Maximum Entropy, Symmetry, and the Relational Bottleneck: Unraveling the Impact of Inductive Biases on Systematic Reasoning

Simon Segert

a dissertation
presented to the faculty
of princeton university
in candidacy for the degree
of Doctor of Philosophy

recommended for acceptance
by the Department of
Neuroscience

Adviser: Jonathan Cohen

May 2024

Abstract

A major outstanding challenge in cognitive science is to understand how people perform extrapolations generalizations that are far outside the context of their previous experiences. There has been longstanding controversy regarding the ability of connectionist (i.e., neural-network based) models to attain such behavior in the absence of explicit symbolic primitives. In this thesis, we argue for an alternative perspective that neural networks can learn to behave *as if* they were implementing a symbolic procedure, provided that they are endowed with certain general *inductive biases* that shape and constrain the learning process (crucially, these biases do not involve the explicit imposition of any symbolic structure). Specifically, we consider three inductive biases that are deeply motivated by previous work in psychology: The Maximum Entropy Principle, the Relational Bottleneck, and Symmetry.

Said in brief, the Maximum Entropy principle prescribes that an agent ought to disperse its internal representations to the fullest possible extent, while Symmetry refers to propensity to attend to repetitive or self-similar structure within the environment, and the Relational Bottleneck refers to a certain architectural constraint which encourages a model to attend exclusively to higher-order relational information at the expense of lower-order perceptual details. In a series of studies, we characterize the effect of multiple combinations of these three learning biases across a wide range of problem domains (analogical reasoning, function learning, and arithmetic). Taken together, our results demonstrate that these biases in combination can effectively enhance the out-of-domain generalization abilities of neural network models, and consequently, due to the psychological plausibility of the inductive biases, point towards a deeper understanding of how such capabilities may be implemented in the human mind.

Contents

Listing of figures

Acknowledgments

First and foremost I would like to thank my advisor Jon– for constantly pushing my limits as a researcher, allowing me to develop my own ideas, and helping me to maintain enthusiasm during slow periods. I'd also like to thank the other members of the lab for creating an atmosphere of open intellectual exploration and discussion. Thanks to the other friends and colleagues who have made time outside of research enjoyable as well. And finally thank you to my parents for their unwavering support and guidance.

All generalizations are false including this one.

Mark Twain

0 Introduction

One of the more notable abilities of people is our ability to make inferences that are far outside anything we have seen before. For example, as a baby, our motor system may learn how to pick up a block from a table and stack it on top of another block, and later on can seamlessly apply that same procedure to assemble larger block towers out of completely new blocks in a new room. This illustrates generalization *out-of-domain*, in that the objects and

setting may be very different from those seen when we learned the procedure, and also *algorithmic* generalization, in that the procedure for the more complicated task of assembling a tower of blocks is built out of a systematic combination of the simpler stacking procedure which had been learned previously.

Although these examples seem simple, they are in fact somewhat at odds with the foundations of the currently prevailing framework of *connectionism*, which aims to model the mind through a process of *statistical learning* implemented through adjustments to connection weights in neural networks [Rumelhart and McClelland, 1986a]. While this framework has the very notable virtues of being able to fit a wide range of empirical data and being based on neurally-plausible components and learning rules, reconciling it with the forms of generalization described above remains a major outstanding challenge [Lake et al., 2016, Fodor and Pylyshyn, 1988]. Indeed, on the one hand, formal learning theory has shown that there is a fundamental difference between statistical learning [Valiant, 1984] and algorithmic learning [Solomonoff, 1964, AN Kolmogorov, 1965]. Furthermore, almost by definition, an agent trained on the statistical structure of a single environment will not be able to adapt when the statistics change (as in the example of moving to a new room with new blocks). Due to the clear facility that people have with such tasks (in some conditions at least), this provides a serious challenge to connectionist models of the mind, which I plan to address in this thesis.

One may ask, though, why remain tied to neural networks? After all, an entire parallel line of cognitive science research has focused on modeling the mind as a process of symbolic procedures, which are flexible and generalizable by design [Fodor, 1975, Piantadosi et al., 2016]. These have included cognitive architectures, and have later evolved to modern

statistical approaches involving explicit statistical inference over structured combinatorial spaces such as graphs [Kemp and Tenenbaum, 2008] and grammars [Goodman et al., 2008]. Such models can make strong generalizations as described before essentially by construction. Given this state of affairs, why are the problems described above not considered solved? From a practical perspective, models of this form are difficult to scale to rich, high-dimensional datasets and often require extensive hand-engineering of both features and primitive symbolic operators. In contrast, scaling up of connectionist principles often leads to models that "just work" [Vaswani et al., 2017, Krizhevsky et al., 2012]. This makes the current limitations of the connectionist approach described above that much more pressing. Indeed, if one accepts that the brain does not literally have symbolic components but is literally, to a first-order approximation, made up of neurons and adjustable connections between them, then one is forced to confront the question of how it is that such a system can *simulate* or *implement* symbolic procedures.

My main goal, then, will be to imbue artificial systems, such as neural networks, with enhanced abilities for these kinds of generalization, as come naturally to humans. This will allow us to give a direct answer to the challenge raised above: that connectionist models can exhibit systematic generalization, provided that they are endowed with an appropriate collection of *structural inductive biases* that shape and constrain the learning process. By a *structural* bias, I mean one that can be implemented within a pure connectionist setup, and does not explicitly imbue the system with symbolic primitives of any kind; rather, the network must *learn* to behave *as if* it were manipulating a set of symbols using a programmatic procedure. Crucially, I do not treat this as a pure engineering problem, but rather constrain the inductive biases that I consider by the phenomenology of the human mind

and brain.

I will in particular focus attention on three inductive biases that I hypothesize to be useful for this goal: the Maximum Entropy principle [Jaynes, 1957], the Relational Bottleneck [Webb et al., 2024], and Symmetry [Bronstein et al., 2017a]. These form a collection of three complementary, but hitherto separately-studied principles that can each be understood as a prescription for making systematically-structured inferences from limited amounts of data. Said briefly, the Maximum Entropy principle prescribes that an agent ought to "spread out" its internal representations as much as possible, while Symmetry refers to the presence of repetitive or otherwise self-similar structure in the agent's environment, and the Relational Bottleneck denotes a certain architectural bias in the context of neural networks that encourages the model to attend specifically to higher-order relational information, to facilitate generalization across problem instances.

Crucially, as I will demonstrate, each can also be readily implemented within a connectionist framework, giving it a psychological plausibility. To thoroughly understand the effect of each of these and their combinations across a variety of contexts, I will analyze, in each chapter, a different combination of these biases, and across a range of different problem settings: analogical reasoning, function learning, and arithmetic. Each of these settings provides a challenging problem that requires either out-of-domain or algorithmic generalization or both to fully master.

In Chapter 1, I provide a reasonably self-contained introduction to these principles; this chapter is primarily expository in nature. In Chapter 2, I consider a model of analogical reasoning that is based on a simplified version of the Relational Bottleneck, namely Graph Theory. I posit a model that is capable of representing an entity or situation as a discrete

graph (that is, a collection of nodes and edges between them), which then can perform analogies by a process of *continuous graph matching*. This has two important properties: first of all, the graph structure allows the model to focus only on the relational structure between entities rather than the raw features themselves when performing the matching. This implements a form of the Relational Bottleneck principle, and is what allows the model to draw analogical inferences across varied domains. Secondly, the matching process itself is implemented in a continuous fashion using a settling process that can be readily implemented within a neural network architecture. This distinguishes our model from classical graph-based analogical models, which rely on purely-symbolic algorithms to perform the matching. I find that our model is able to flexibly plan on classic logic puzzles such as the Towers of Hanoi, and also that it has a similar pattern of success and failure as people in a logic puzzle.

However, I conclude that this model is limited by the reliance on hand-specified graph structure. Nevertheless, it provides a key insight that the focus on relational information (such as adjacency graphs) is very useful *once such representations have been extracted*, which then motivates us to address the question of how to extract such representations in a more systematic and plausible fashion, which I explore from various angles in the following chapters.

In Chapter 3, I continue to explore the Relational Bottleneck principle, and also introduce the notion of Maximum Entropy. Here, I shift gears from the domain of analogical reasoning and problem solving to that of function learning. This domain has the advantages that it is simpler to describe and experimentally control, while also providing ample opportunity to study the sorts of generalization that interest us (indeed, extrapolating a

scalar function is arguably the clearest and most pared-down example of out-of-domain generalization [Bott and Heit, 2004]). I devise a model that combines the two principles in a natural way and show, using a mathematical result called Burg's Theorem, that this necessarily constrains the model to take the particularly simple form of a linear autoregression. Interestingly, this model is capable of zero-shot extrapolation of simple functional forms such as lines and sinusoids, *without any explicit building-in of such forms*. Furthermore, I show that this model makes several qualitative predictions about the extrapolation patterns of people, and moreover, that these predictions are at odds with the corresponding predictions made with the currently prevailing model of function learning, which is based on Gaussian processes with a hand-coded prior distribution [Schulz et al., 2017]. In a series of experiments, I show that predictions of our model are borne out in human behavior. This model is also notable in that the form of the linear autoregression entails that the predictions are obtained from an algorithmic iterative process. This property presages some forms of Symmetry and algorithmic and generalization that I will study later in Chapter 6.

I therefore conclude that the combination of Relational Bottleneck and Maximum Entropy is sufficient to describe much of peoples' extrapolation abilities for scalar functions. However this model is still fundamentally a mathematical statistical model, as opposed to a neural network. In the following two chapters, I will explore how to distill the insights of this model into a more plausible neural form.

In Chapter 4, I continue the study of function learning, with a goal of implementing the biases of Maximum Entropy and Relational Bottleneck within a neural network architecture. I choose to focus specifically on the transformer architecture, due to its demonstrated flexibility. I propose two modifications of the basic transformer architecture, each

of which implements a key feature of the previous autoregressive model. The first modification involves imposing a finite bound on the maximum "lookback-distance" within the model's self-attention weights. This corresponds to the finite moving-window feature of the linear autoregressive model. Crucially, in the linear autoregressive case, the length of the window was treated as a hyperparameter and not constrained theoretically. In the transformer model, by contrast, it is possible to treat this window length as a differentiable parameter, and learn it in an end-to-end fashion along with the other model parameters. From a technical perspective, this is accomplished by replacing the hard cutoff with a soft one. The second modification of the transformer model is a way to directly instantiate the Relational Bottleneck principle. Specifically, in a typical transformer model, it is trained to predict the next input conditional on the previous ones. To implement the Relational Bottleneck, I propose to instead have the model operate purely at the level of relations between the inputs. I therefore propose a model which predicts the *similarity of the next input to all preceding ones*, conditional on the similarity matrix of the preceding inputs. This can also be thought of as predicting the next row of the similarity matrix, given the matrix.

I show that while both of these modifications can improve the extrapolative ability of the model on the sorts of functions that people can recognize and extrapolate (e.g. lines, sinusoids, and RBF curves), combining both modifications in a single model yields the best overall extrapolation performance. This provides another validation of our observations that the combination of Relational Bottleneck and Maximum Entropy can lead to extrapolation abilities across a wide range of scalar functions. Finally, I show that the second modification, besides providing point estimates of the function, can actually naturally provide uncertainty estimates as well. This distinguishes it from typical neural network models,

and allows it to inherit one of the main advantages of Bayesian models. I provide a proof of concept in which I show that the resulting uncertainty estimates are reasonable when compared with the underlying ground-truth uncertainty in the data generating process.

In Chapter 5, I solidify understanding of function learning through neural networks, and aim to understand how to learn *general-purpose* representations that are useful for extrapolation, through a process of unsupervised learning. I also introduce here the notion of Symmetry. Our basic argument is that an agent can learn such representations through a learning objective that imposes an invariance property with respect to certain common symmetries in the environment. For example, consider a completely straight line, as well as a line that has some smooth, localized perturbations. While these are of course not literally the same input, it is reasonable to suppose that whatever processes generated these two lines probably share many of the same qualitative features, and therefore are related via a symmetry transformation. I therefore study an objective that enforces an invariance with respect to such symmetries by pressuring the representations of these two inputs to be the same. At the same time, the objective has a second term which also imposes a maximal "spreading out" of the representations, which can be thought of as an implementation of a Maximum Entropy prior. Note that the mathematical form of this objective function is not new: it has been previously proposed and studied in a machine learning context [Chen et al., 2020]. I distinguish the present work from this line of work through the use of ecologically-plausible and theoretically-motivated augmentations, and in direct comparison with human data and abilities.

Through a thorough set of experiments, I demonstrate that this method learns representations that are more useful for function learning tasks compared with representations

learned by other existing deep-learning algorithms. I quantify the usefulness by measuring few-shot learning accuracy on downstream function-learning tasks, such as categorization and extrapolation. I also show how this model can be integrated with the previous linear autoregressive model, in order to combine the learned statistical structure from the new model with the free-extrapolation abilities of the autoregressive model.

Furthermore, I perform a direct comparison with human extrapolation data, in which, as shown in [Schulz et al., 2017], people prefer extrapolations that are more "compositional" (that is, built out of several simpler pieces). By modeling their task on my models, I show that my model shows an effect in the same direction (albeit not quantitatively as strong) as people, while none of the other models show such an effect above the level of statistical chance. I conclude that this effect may in fact be explained as a consequence of the basic inductive biases built into the model.

Finally, in Chapter 6, I shift gears to a more targeted study of algorithmic generalization. I use as a test case a symbolic task which requires repeating an underlying kernel (or *symmetry function*) in an iterative fashion: base arithmetic, that is, computing the sum of two positive integers expressed in the standard base format. First of all, I review how the language of Group theory can be used to provide a formal analysis of the symmetries inherent to this problem [Isaksen, 2002]. I then extend this analysis to show how the task can be generalized using *nonstandard carry tables*, which are essentially alternative addition rules that share the same formal symmetries as the standard addition rule. Due to their complexity and intricate structure, these provide a useful testbed for evaluating the systematic generalization capabilities of neural network models. I then evaluate the effect of multiple neural network design choices, including recurrence (that is, a form of symmetry with re-

spect to time) and usage of external memory, with respect to extrapolation ability on this task, and find a combination of these two to be the most promising.

In summary, I analyze the effects, in isolation and in combination, of three basic inductive biases: Maximum Entropy, Symmetry, and the Relational Bottleneck. Across a range of tasks (analogies, function learning, and arithmetic) I show that these biases can both be used to accurately model human data on complex reasoning tasks, as well as employed to build artificial systems that inherit some of the systematic generalization abilities of people. In this way, I contribute to the larger overall goals of modeling human cognition using powerful machine learning models, and of using theories and phenomena of human cognition as a source of inspiration for building more powerful models that can better approximate our abilities.

The beginner should not be discouraged if he finds he does not have the prerequisites for reading the prerequisites.

Paul Halmos

1

Background Material

The purpose of this section is to give a reasonably self-contained introduction to the main theoretical principles that will form the basis of the remainder of the thesis: Maximum Entropy Principle, the Relational Bottleneck, and Symmetry. My purpose here is strictly pedagogical- I do not make claims to originality of any of the material presented in

this chapter. The reader may thus feel free to skip over any sections that cover material with which he or she is already familiar.

1.1 Maximum Entropy Principle

In a general sense, the Maximum Entropy Principle provides a prescription for making inferences from incomplete data. We will provide a thorough treatment, starting with a simple "hands-on" example illustrating how the principle may be used in practice. We then provide a formal treatment of the principle and its fundamental assumptions. Finally, we draw precise and explicit connections between the principle and a number of other established constructs from statistics; in this way we provide convergent justification for the validity of the principle.

We now start with a example, adapted from [Jaynes, 1957]. Suppose you are the owner of a casino. One of the games that you offer is Roulette. In this version of the game, there are 50 slots labeled from 1 to 50. At each draw, a ball is randomly dropped onto the board and settles in one of the slots. A player can pay one dollar to play: to do so, he or she would select one of the 50 numbers, and if the ball lands in that slot, the player would be paid 50 dollars (you are an honest casino owner who pays out even odds).

Unfortunately, you have noticed that one of the tables is slightly miscalibrated- over a large number of games, the average slot that the ball lands in is 30, while for a completely unbiased table the average should be

$$\frac{1+2+\cdots+50}{50} = 25.5$$

Even more unfortunately, some of your records have been lost, so you do not how exactly often each particular number came up, but you do know that the records span over a period of several months and thousands of games, so the average is quite well estimated. Since it is too expensive for you to buy a new table, you instead decide to modify the odds you offer on each slot, in order to prevent observant customers from taking advantage of the biases. However, without having the actual empirical frequencies of each slot, this is a bit difficult to do.

The Maximum Entropy Principle is designed to deal with situations such as the one in the example above. Namely, we have some unknown probability distribution (in this case, the probability p_i of landing on each slot, for $i = 1, 2, \ldots, 50$), but we only have access to some small set of constraints on the distribution (in this case, knowing that $\sum_{i=1}^{50} i p_i = 30$). In this case, the Maximum Entropy principle gives a principled estimate for the full distribution p_i, while respecting the known constraints. As the name suggests, it makes crucial usage of the notion of *entropy*, which can be thought of as a principled metric to measure the flatness or uniformity of a probability distribution.

In this example, it turns out that, according to the predictions of the principle, the estimated probability of landing in slot 1 is approximately %1.1, while the estimated probability of landing in slot 50 is approximately %3.3 percent. Thus if we want to modify the odds of the table, we should pay 1/.011=$90.9 dollars for correctly guessing slot 1, while paying $30.3 dollars for correctly guessing slot 50.

So on the one hand, this principle seems almost supernaturally powerful: given just one constraint, we have fitted 49 free parameters in a seemingly-reasonable way. But would you as a casino owner actually heed the predictions in this way? This is where things get tricky,

and depend on an understanding of the motivation and rationale for the principle itself.

1.1.1 Formal assumptions and setup of Maximum Entropy Principle

The above example illustrates the key assumptions underlying the Maximum Entropy Principle. Namely, to apply the principle, we must have

- An underlying sample space S,
- An unknown probability distribution μ that we want to estimate. The distribution may be either discrete or continuous.
- Some collection of "observables" $f_k, k = 1, \ldots, K$, which are functions from the sample space to $\mathbb{R}$. It is assumed that we know both $y_k := \mathbf{E}_{X\sim\mu} f_k(X)$ for each k, as well as the explicit functional form of each f_k.

If we are given these ingredients, then the Maximum Entropy Principle provides us with a probability distribution μ_{ME} such that $\mathbf{E}_{X\sim\mu_{ME}} f_k(X) = y_k$ for each k. That is, the estimated distribution μ_{ME} is guaranteed to satisfy the known constraints.

Definition 1. *Using the above notations, the associated Maximum Entropy distribution μ_{ME} is defined as*

$$\mu_{ME} = argmax_{\mu\in P(S):\mathbf{E}_{X\sim\mu} f_k(X)=y_k, k=1,\ldots,K} H(\mu) \tag{1.1}$$

where $P(S)$ denotes the set of all probability distributions on S, and $H(\mu)$ denotes the Shannon Entropy[*] of the distribution: either the discrete or differential entropy depending on the nature of S.

[*]Recall that for a discrete probability distribution this is defined as $H(\mu) := -\sum_k P(\mu = k) log P(\mu = k)$, with an analogous extension for the continuous case

Although this definition is rather simple, there are some fundamental and non-obvious questions that it leaves unanswered. For example, under what conditions is a maximizer guaranteed to exist? Fortunately, it can be shown that the maximizer exists and is unique under very general conditions, a more thorough treatment of which is given in the Appendix.

1.1.2 Justifications for the Maximum Entropy Principle

Having described *what* the Maximum Entropy principle prescribes, we may move to the more interesting and difficult question of why and under what conditions the prescriptions are reasonable. There are several distinct *a priori* justifications that may be given for the principle:

Physical

According to the Second Law of Thermodynamics, closed systems will increase entropy. Therefore, when observing a closed system, it is reasonable to suppose that it is in a high-entropy state by the time we observe it. This justification is especially compelling for original subjects to which the principle was applied, such as gases.

Simplicity

Entropy can conceptually be thought of as a measure of "uninformativeness", with high entropy corresponding to an uninformative or unstructured distribution. In this way, the Maximum Entropy principle can be seen as a quantitative interpretation of Occam's Razor. However, this interpretation is actually a bit more subtle than it seems at first. To illustrate,

consider the related principle of Minimum Description Length [Rissanen, 1978]. In the derivation of that principle, high entropy is taken as a measure of *complexity*, which seems to be the exact opposite interpretation as implied by Maximum Entropy. This seeming inconsistency can be resolved by noting that, in the case of Maximum Entropy, we are trying to describe the *distribution* as simply as possible, while in the Minimum Description Length case, we are trying to describe *samples from the distribution* as simply as possible.

We can intuitively see that these two are somewhat at odds with each other. Consider a 50-50 Bernoulli distribution, for example as in a fair coin toss. On the one hand, this distribution has the maximum possible entropy among all distributions supported on two point masses. This means that a given sample from this distribution would carry more information than a sample from any other such distribution-consequently, the former sample will take *more* bits to describe completely. Thus, from the MDL perspective, it has a high description length. On the other hand, the form of the distribution itself is maximally simple. As a contrast, if we consider a completely rigged 100-0 coin toss, then each individual sample carries no information, but the description of the distribution is more complex because we have to arbitrarily "pick out" one of the two outcomes, which intuitively costs one bit. To summarize the difference, if we care about minimizing the complexity of *individual samples* from our distribution, we should consider MDL; but if we care about minimizing the complexity of *our description of the distribution*, then we should consider MaxEnt.

Maximum Likelihood

The method of Maximum Likelihood estimation [Fisher, 1922] forms the backbone of most statistical model-fitting procedures today. The Maximum Entropy solution can be

shown to be equivalent to Maximum-Likelihood estimation when using a certain hypothesis class of distributions.

Definition 2. *Let $f_1, \ldots, f_K$ be some functions on the sample space S. The Exponential Family of distributions $EF(f_1, \ldots, f_K)$ consists of all probability distributions of the form*

$$\mu(x) = \frac{1}{Z(\lambda_1, \ldots, \lambda_K)} e^{\sum_{k=1}^{K} f_k(x)\lambda_k} \tag{1.2}$$

where $\lambda_i \in \mathbb{R}$ and Z is a normalizing constant.

The key observation here is that fixing the values of λ_i in order to match the observed values of the statistics is the same as selecting the values of λ_i so as to maximize the likelihood within this class of distributions.

Theorem 1. *Let $x_1, \ldots, x_n$ be iid samples from some known distribution on S, and let $f_1, \ldots, f_K$ be some functions on S. Let μ_{MLE} denote the maximum-likelihood estimate from within the exponential family, that is*

$$\mu_{MLE} = argmax_{\mu \in EF(f_1, \ldots, f_K)} \prod_{i=1}^{n} \mu(x_i) \tag{1.3}$$

Let μ_{ME} denote the maximum entropy distribution, with respect to the statistics f_k and the empirical mean values on the data, that is

$$y_k := \frac{1}{n} \sum_{i=1}^{n} f_k(x_i) \tag{1.4}$$

Then $\mu_{ME} = \mu_{MLE}$.

We provide a proof in the Appendix.

Casting the Maximum Entropy principle in this way allows us to analyze it from a new angle, since we can see that it inherits many of the advantages and disadvantages of the Maximum Likelihood principle. On the one hand, Maximum Likelihood provides a general and principled method for fitting of statistical models. On the other hand, it can be sensitive to the particular choice of hypothesis class- the above proposition provides an explicit description of the effective hypothesis class implied by the Maximum Entropy Principle.

Information-Theoretic

One common intuitive way to describe the Maximum Entropy distribution is as the "flattest" or "most uniform" distribution subject to a set of constraints. This interpretation can actually be made rigorous using the language of Information Theory. The intuition is that we can reasonably define the "flatness" of a distribution as its distance from the uniform distribution, under a suitable information-theoretic metric (namely, Kullback-Leibler divergence). This allows us to rigorously specify the flattest distribution subject to a set of constraints by setting up an appropriate optimization problem. The content of the following theorem is that this procedure recovers the Maximum Entropy distribution.

Theorem 2. *Let S be a finite set, and let U be the uniform distribution on S. Let f_k and y_k be constraints as before. Then among all distributions that satisfy the constraints, the Maximum Entropy solution is the closest to U, where distance is measured using Kullback-Leibler divergence. That is,*

$$\mu_{ME} = argmin_{\mu: E_{X\sim\mu} f_k(X) = y_k} KL(\mu|U)$$

Proof. By the definition of Kullback-Leibler divergence, we have

$$KL(\mu|U) = \sum_{x \in S} \mu(x) \log(\mu(x)) - \mu(x) \log U(x) \tag{1.5}$$

Rearranging and recalling that U is uniform:

$$KL(\mu|U) = -H(\mu) - \log |S| \tag{1.6}$$

The $\log |S|$ term is a constant that does not depend on μ, so we see that minimizing $KL(\mu|U)$ is equivalent to maximizing $H(\mu)$. □

This theorem justifies the intuitive pictures of starting with a uniform prior distribution, and modifying it in a minimal way so as to satisfy the constraints.

Game Theoretic

We will mention a final interesting perspective due to [Grünwald and Dawid, 2004], which shows that in some ways the Max Ent corresponds to making inferences under an adversarial or "worst case" model of the world.

In this setting, we imagine we are playing a game against an adversary. The rules of the game are as follows: the adversary first reveals to us a collection of statistics and values corresponding to f_k and y_k as before. We then "play" some probability distribution μ satisfying the constraints (i.e. $\mathbb{E}_{X \sim \mu} f_k(X) = y_k$ for each k). The adversary, after having seen our play, then plays another μ' which also is required to satisfy the constraints. Our score from this outcome is the negative of the cross entropy $-CE(\mu, \mu')$, while the adversary's score is $CE(\mu, \mu')$ (i.e., it is a zero sum game).

Intuitively, we are trying to guess the adversary's "secret" distribution μ' as accurately as possible, while the adversary is trying to make our guess as bad as possible. However, the adversary is not actually required to commit to a particular μ' until after seeing our guess. Thus, we should assume that no matter which μ we play, the adversary will choose the μ' that leads to the worst possible outcome for us- implying that we should choose μ so as to obtain the *best worst-case scenario*.

Theorem 3. *[Grünwald and Dawid, 2004]. In the above game, the optimal move is to play the maximum entropy distribution μ_{ME} with respect to the given constraints.*

From this interpretation, we can see that the MaxEnt principle implies a rather pessimistic world view– it can be seen as a maximally conservative, which is appropriate for situations in which the world is "out to get us". On the one hand, this seems somewhat absurd-if the world we lived in was really like that adversary in the game, then science would not be possible. On the other hand, if we consider that we have already incorporated some prior knowledge into the form of the constraints, then the prescription seems more reasonable. In other words, the principle implies we should be maximally conservative *with respect to factors of which we have no knowledge*, all other factors having already been incorporated into our constraints. Thus, this illustrates a distinctive property of the Maximum Entropy formalism– it supposes that our prior knowledge is either known exactly (i.e., we know that *exact* value of the observable) or we are in a state of complete uncertainty.

1.2 Symmetry and Groups

The Symmetry Principle is motivated by the goal of developing models that can make reasonable predictions on inputs that are "out-of-domain" with respect to inputs that the

model has seen before. However, fundamentally, this is possible only if we make the a priori assumption that there is some amount of structure shared between the original inputs the model has been trained on, and the new out-of-domain inputs. As a simple illustration, suppose we have trained some binary classification model m, and we want to see how well it generalizes to completely arbitrary new data, which take the form (x, l), x being an input and $l \in \{0, 1\}$ being the desired label. It is obvious that m can only attain an accuracy of 50 % when considered with respect to *all possible* new data, since if m correctly predicts the label for the datum (x, l), then it will incorrectly predict the label for the datum $(x, 1 - l)$. This is a similar observation to the "No-Free Lunch" theorem of learning theory[Wolpert, 1996].

Thus, it is clear that we cannot expect our models to generalize to completely arbitrary new domains, and we must make some assumption that there is some shared structure or underlying symmetry relating the domains. This leads us to the mathematical field of *Group theory*,which provides a powerful and useful language for describing and analyzing symmetries at a very general level. This formalism will provide a convenient language for our discussion of symmetry in later chapters, so we provide an overview here.

1.2.1 Formal Treatment

Definition 3. *A Group is a set G together with a binary operation $* : G \times G \to G$ which satisfies the following axioms:*

1. *(Associativity) $(a * b) * c = a * (b * c)$ for all $a, b, c \in G$*
2. *(Identity element) There exists some element $e \in G$ such that $a = a * e = e * a$ for all $a \in G$*

3. *(Existence of inverses) For each $a \in G$, there exists some $b \in G$ such that $a * b = b * a = e$, where e is as in Property 2.* [†]

In this definition, the group elements are interpreted as abstract symmetries, and the group operation encodes the relations between them. For example, if we want to model reflectional symmetry through a single axis, we might take the group $G = \{g, e\}$, where e is the identity element and g is a non-identity element which satisfies $g * g = e$. The preceding equation encodes the observation that if we reflect some object across an axis, then reflect the result again across the same axis, the final result is to leave the input unchanged. Note that this is a property of reflections themselves, rather than of any particular input that we may apply the reflection to. This illustrates a crucial property of the group definition: that it completely abstracts away all information about the objects being acted on by any symmetries, as well as how such symmetries act on those objects. All that is relevant is the structure *of the set of symmetries itself*.

It is also worth noting that all specific groups that we will have occasion to consider will satisfy the additional property of *commutativity*, meaning that $gh = hg$ for any $h, g \in G$. Such groups are called *Abelian*. In what follows, we will assume that any group under discussion is Abelian, unless explicitly stated otherwise.

While formally rigorous, this definition is more abstract than we would like for specific applications, in which we would like to consider the "symmetry" of particular transformations or operations on a set (such as translation or reflection). That is, we would like to have a way to instantiate a group as a specific set of transformations of some space. The following definition specifies what it takes to do so.

[†] It is simple to see as a consequence of the axioms that any group can have only one identity element, and each element can have only one inverse.

Definition 4. *Let G be a group and let X be a set. A Group Action of G on X consists of a set of transformations of X, $\{T_g\}_{g \in G}$ [‡], with one transformation for each element of G. The transformations are required to satisfy the following axioms:*

- *(Preservation of identity) $T_e = Id_X$, where $e \in G$ is the identity element and Id_X is the identity function on X,*
- *(Preservation of Group structure) $T_{g*h} = T_g \circ T_h$ for each $g, h \in G$, where $\circ$ denotes functional composition.*

A group action can thus be thought of instantiating the group within a specific set of transformations of a set.

Example 1. *Consider the space of 2-dimensional images. Since images can be subject to affine translations, we intuitively expect that there should be some group action that implements this structure. Since a translation can be described by two parameters, we may take $G = \mathbb{Z}^2$ to correspond to the group of translations, and let $(x, y) \in G$ correspond to the transformation*

$$T_{(x,y)} = \textit{translation to the right by x pixels and upward by y pixels} \tag{1.7}$$

It is easy to see that this satisfies the Preservation of Identity property. However, the Preservation of Group structure property raises an issue with respect to boundary conditions. To illustrate, let's suppose our images have width W and height H. If we translate by $H/2$ units to the right, then the entire rightward half of the image would go over the boundary, while the left half would be padded with zeros. If we then translated the resulting image back by $W/2$

[‡] That is, each T_g is a function from X to itself

units to the left, we would have no way to recover the overflowed pixels. This is schematically illustrated below

$$\begin{pmatrix} *_1 & *_2 \\ *_3 & *_4 \end{pmatrix} \rightarrow^{T_{(W/2,0)}} \begin{pmatrix} 0 & *_1 \\ 0 & *_3 \end{pmatrix} \rightarrow^{T_{(-W/2,0)}} \begin{pmatrix} *_1 & 0 \\ *_3 & 0 \end{pmatrix} \tag{1.8}$$

where $*_i$ denote sub-images of appropriate sizes. So the property is violated. The reason is that our transformations can lose information when some portion of the image overflows past the boundary; in a valid group action each transformation must be invertible. In the case of images there are several ways to fix this. One would be to impose periodic boundary conditions, so that any portion of the image that overflows the boundary would "wrap around" to the other side. Another would be to regard each image as being padded with infinitely many zeros in all 4 directions.

Having studied how to describe the structure of a set of symmetries, let us think through more specifically how we may use this in a learning setting. For example, consider the problem of extrapolating a sinusoidal function. The key property of such a function is that it is completely determined by its values on a bounded interval which spans one period length. Thus the underlying symmetry has reduced the problem of extrapolating out to possibly arbitrarily large distances to one of learning the function on a *bounded* interval. But additionally, we also must assume that the sinusoidal function is "compatible" with the underlying symmetry in some way. For example, if the period length is T, then knowing the values of the function on the interval $[0, T/2]$ would not allow us to extrapolate.

We thus want to formalize what it means for a function on some space to be compatible with the symmetry structure of that space. The next definition provides the simplest such

form of compatibility.

Definition 5. *Let X be a set and f a function defined on X. Let G be a group that acts on X. Then f is invariant with respect to G if for all $x \in X$ and $g \in G$ we have*

$$f(T_g x) = f(x) \tag{1.9}$$

It is also possible to have more complicated forms of compatibility. While the invariance condition completely ignores the action of any group element on its domain, we can instead require that the group action on the domain transforms the value of the function in a consistent way- say $f(T_g x) = T'_g f(x)$ for some other transformation T'_g. Since we are transforming the values of f now in a non-trivial way, we can see that we would actually need to impose some structure on the codomain of f for this to make sense. This motivates the following definition

Definition 6. *Let X and Y be sets, and let $f : X \to Y$ be a function. Let G be a group that acts on both X and Y. Denote the corresponding transformations of X by $T^X_g, g \in G$ and analogously for Y. We say that f is equivariant with respect to G if for all $x \in X$ and $g \in G$ we have*

$$f(T^X_g x) = T^Y_g(f(x)) \tag{1.10}$$

To show how both of these properties – invariance and equivariance – can be useful in different ways, we contrast them in the domain of 2D images used in the example above.

Example 2. *Consider the setting of affine translations of 2D images as in Example 1. As discussed there, we will regard each image as being padded with infinitely many zeros so that the*

translations form a valid group action. For the sake of this example, we'll assume further that the images are grayscale, so they can be represented as scalar arrays indexed by the horizontal and vertical positions. Two basic properties of images are the total luminance and the center of mass. Regarding an image I as an array, we can formally define these as

$$Lum(I) = \sum_{i,j} I[i,j] \tag{1.11}$$

$$CM(I) = \left(\frac{\sum_{i,j} I[i,j]i)}{Lum(I)}, \frac{\sum_{i,j} I[i,j]j)}{Lum(I)} \right) \tag{1.12}$$

(Note that even though the array I has infinitely many entries, all but finitely many of them are equal to zero, so the sums are well-defined and finite).

It is clear that

$$Lum(T_{(x,y)}I) = Lum(I) \tag{1.13}$$

$$CM(T_{(x,y)}I) = CM(I) + (x,y) \tag{1.14}$$

The first equation says that Lum is invariant to the action of G, while the second says that CM is equivariant, if we regard G as acting on the codomain $\mathbb{R}^2$ by vector addition.

1.2.2 Group Extensions and Cocyles

A situation that often arises is one in which two groups need to be combined into a larger group. A paradigmatic example, that we return to in Chapter 6, is integer addition. Following [Isaksen, 2002], consider the structure of the group of integers modulo 100 $G = \mathbb{Z}_{100}$. Intuitively, we can think of this group as being combined from two simpler groups: the

integers that are divisible by 10 $G_T = \{0, 10, \ldots, 90\}$, and the integers mod 10 $G_O = \{0, 1, \ldots, 9\}$[§] Clearly the group G can in some sense be described as multiple copies of G_O, with the copies being "parametrized" by G_T. The simplest such structure would be the Cartesian product group $G_0 \times G_T$, but this is not the same group structure as G. Indeed, if we start with $[0] \in G$ and [1], we would arrive at $[10] \in G$, a number which has the same ones digit but differs in the tens digit. More formally, if we want to add two numbers $[a_1 * 10 + b_1] + [a_2 * 10 + b_2]$ with $a_i \in \{0, \ldots, 9\}$, we can express this as

$$\begin{aligned}
[a_1 * 10 + b_1] + [a_2 * 10 + b_2] &= [a_1 * 10] + [b_1] + [a_2 * 10] + [b_2] && (1.15)\\
&= ([a_1 * 10] + [a_2 * 10]) + ([b_1] + [b_2]) && (1.16)\\
&= [a_1 * 10] + [a_2 * 10] && (1.17)\\
&+ [carry(b_1, b_2) * 10] + [(b_1 + b_2)\%10] && (1.18)
\end{aligned}$$

where $carry(b_1, b_2)$ is the number that would be carried to the tens place when adding b_1 and b_2 (for example $carry(8, 6) = 1$). This illustrates how the addition rule in $\mathbb{Z}_{100}$ can be expressed in terms of the two constituent addition rules in G_T and G_O (since $[a * 10] \in G_T$ for any a, and addition in G_O is done by reducing modulo 10). But in addition to these two rules, we also need to specify the "carry" function. The carry function may thus be interpreted as providing what is needed to glue together the two groups G_T and G_O in a consistent way.

[§]Note that both of these groups happen to have exactly the same abstract group structure, but for a general extension this need not be the case; we will see examples of such in Chapter 6.

Definition 7. *Let G and H be groups. A cocycle is a function $z : G \times G \to H$ that satisfies:*

$$z(g_1 +_G g_2, g_3) +_H z(g_1, g_2) = z(g_2, g_3) +_H z(g_1, g_2 +_G g_3) \tag{1.19}$$

for all $g_1, g_2, g_3 \in G$. This equation is called the Cocycle Condition.

The definition may seem arbitrary, but it is actually a restatement of the associative property. We make this precise below:

Definition 8. *Let $(G, *_G)$ and $(H, *_H)$ be groups, and let $z : G \times G \to H$ be any function (not necessarily a cocycle). The Extension $H *_z G$ is defined as the set of all pairs (h, g) with $h \in H$ and $g \in G$ together with the binary operation $*$ given by:*

$$(h_1, g_1) * (h_2, g_2) = (h_1 *_H h_2 *_H z(g_1, g_2), g_1 *_G g_2) \tag{1.20}$$

Now we can precisely formulate the significance of the cocyle condition.

Proposition 1. *In the above setting, $(H *_z G, *)$ is a group if and only if z is a cocyle.*

The proof essentially consists of unraveling the relevant definitions, see [Isaksen, 2002] for details.

Thus we have seen how cocyles can be used to define Extensions; i.e., a group that "combines" the two constituent groups. But two different cocyles may create the same extension. It is useful to know when this is the case.

Proposition 2. *Two cocyles z and z' generate isomorphic extensions $H *_z G$ and $H *_{z'} G$ if*

and only if there exists some function[¶] $h : G \to H$ *such that* $h(0) = 0$ *and*

$$z(g_1, g_2) = z'(g_1, g_2) + h(g_1 - g_2) \tag{1.21}$$

This is called the Coboundary condition

A noteworthy property of Group extensions is that they combine both invariant and equivariant structures. To unpack this, let us observe that given an extension $H *_z G$, both of the groups H and G act on the extension in the evident ways:

$$h + (a, b) := (h, 0) + (a, b) = (h + a + z(0, b), b) \tag{1.22}$$

$$g + (a, b) = (a, b + g) \tag{1.23}$$

Moreover, there are functions π_G and π_H defined on $H *_z G$ which project onto the given groups.

We see that

$$\pi_G(g + (a, b)) = \pi_G((a, g + b)) = b + g = g + \pi_G(b) \tag{1.24}$$

showing that π_G is equivariant with respect to the action of G. At the same time, for some $h \in H$,

$$\pi_G(h + (a, b)) = \pi_G((a + h + z(0, b), b) = b = \pi_G(b) \tag{1.25}$$

showing that π_G is invariant with respect to the action of H. This illustrates as well that the same function can be both invariant and equivariant, depending on which group action is

[¶] not necessarily a homomorphism

being considered.

We can relate both of these properties back to basic facts about addition. The equivariance property can be summarized as "the ones digit of a sum depends only on the ones digits of the two summands", while the invariance property can be summarized as "the ones digit of a number does not change if we add a multiple of 10". Said another way, the invariance property here pertains to a finite set of tokens (namely the digits 0-9) and their cyclic ordering, while the equivariance property here pertains to the way in which these same tokens are used at different scales.

1.3 Relational Bottleneck

At a high level, the Relational Bottleneck falls under the heading of an inductive learning bias that encourages neural networks to learn representations that behave in a discrete or symbolic fashion, thus facilitating systematic generalizations to new problems [Webb et al., 2024]. In this way, it can be seen as an answer to the symbolic modeling approach discussed in the Introduction, with the crucial difference that the symbolic-like behavior is *learned* in a network with the help of appropriate inductive biases, rather than built in explicitly. Further, it can be seen as complementary to the group theory discussion. There, we considered group structure as a particular kind of inductive bias that would be helpful for making predictions about structured settings which have some underlying symmetry or repetitions. In a similar fashion, the Relational Bottleneck can be regarded as a natural bias for a setting in which the underlying symmetry of interest is fundamentally relational in character. A paradigmatic example of such a setting is the "Same-Different" task [Webb et al., 2021, Marcus et al., 1999], in which the learner is presented with two separate objects and simply

has to indicate whether they are identical or not. The key aspect of this task is that the relevant information concerns only the relationship between the two objects, and can be applied regardless of the specific character of the objects. Thus, although the task may sound trivial to a person, it can be challenging for neural networks which may struggle to generalize the relation learned on one class of objects to a superficially different class (e.g. learning the relation on images of animals, and generalizing it to images of cars) [Webb et al., 2021]. By contrast, children can learn to do this task and similar ones at a few years of age [Marcus et al., 1999].

This motivates the typical implementational form of the Relational Bottleneck, which consists of modify the architecture of a neural network in such a way that the final output of the network does not directly depend on surface-level properties of the input, but rather depends only on the pattern of similarities between the input constituients. We will discuss possible implementations in further detail below.

1.3.1 Relational Bottleneck Models

While the preceding discussion in this section been at a high level, we now ground it by describing some specific instantiations of the principle. The simplest example is the *CoRelNet* model of [Kerg et al., 2022]. This model learns a mapping from an ordered sequence of inputs to a discrete label. For example, we can express the same-different task in this format, if we take each individual input to be a 2D image, and the sequence to have length 2. Other tasks are possible as well, such as the Distribution of Three. Here, we may again take each individual input as a 2D image, and construct the sequence such that the first three images form a valid set of three (i.e., three different images), the next two form a partial valid set

(i.e. two different images selected from the first set of three), and the remaining images are candidate choices to complete to a valid set of three. The network would then be trained to output the index of the correct choice among the candidates.

The architecture of the model consists of two components: an encoder, *ENC* which processes each individual input to a vectorial representation, and a readout network φ, which produces the final answer. Typically, *ENC* would be a convolutional net and φ would be an MLP. The inputs are processed in three steps: first each input is run through the encoder to obtain a vector $h_i = ENC(X_i)$. Then, the resulting vectors are combined into a similarity matrix S, defined as

$$S_{ij} := Sim(h_i, h_j) \tag{1.26}$$

where *Sim* can be any similarity measure (often in practice a normalized dot product). Finally, the output of the network is given as $\varphi(S)$, with φ being some learnable function (often an MLP yields the best performance in practice). Crucially, the output depends on the inputs only through S and φ cannot "see" the object-level features X_i or h_i directly. This provides a simple and clear illustration of a Relational Bottleneck. On the one hand, it throws away any information not directly contained in the matrix S, which clearly limits the expressivity of the network. On the other, this removal of information facilities generalization under potentially extreme OOD circumstances (namely, to any inputs that can be encoded and have similar similarity patterns compared to inputs seen during training). In the extreme, if we know the output of the network on some given sequence $X_1, \ldots, X_n$, then it is *guaranteed* to give the same output on some other sequence $X'_1, \ldots, X'_n$ provided that $Sim(X_i, X_j) = Sim(X'_i, X'_j)$ for all i and j. This provides the basic mechanism that allows the network to generalize on purely-relational tasks such as the same-different task.

Other notable models based on the Relational Bottleneck principle are the ESBN [Webb et al., 2021] and Abstractor [Altabaa et al., 2024]. The former enforces an explicit separation between processing of "domain-specific" information derived from input stimuli and "abstract" information which is what the network uses to determine its final answer. The abstract representations depend on the domain-specific representations only indirectly through a similarity-based retrieval operation. The abstractor similarly enforces an explicit representation between two processing streams, however it is done within the Transformer architecture through constraints on the cross-attention mechanism.

These architectures are schematically illustrated in Figure 1.1.

1.4 Psychological Justification

So far, we have described and motivated each of the three principles at a formal level. However, as laid out in the Introduction, our goal is ultimately to use these as tools for understanding human cognition. Therefore, it is crucial that we have some theoretical justification for each of these as reasonable inductive learning biases used by people.

The Maximum Entropy principle. This has a long history in psychology, with some of the earliest work using it being the work of Myung [Myung, 1994, Myung and Shepard, 1996], showing that classical categorization models can be cast in terms of Maximum Entropy. More recently, the principle is related to the Free Energy framework [Friston, 2010]. While this framework is significantly more complex and makes more theoretical commitments, one of the key features is that an organism acts as if to maximize a "free energy functional," the functional form of which includes a term that measures the entropy of the agent's estimated posterior over the environment. In other words, maximizing the free

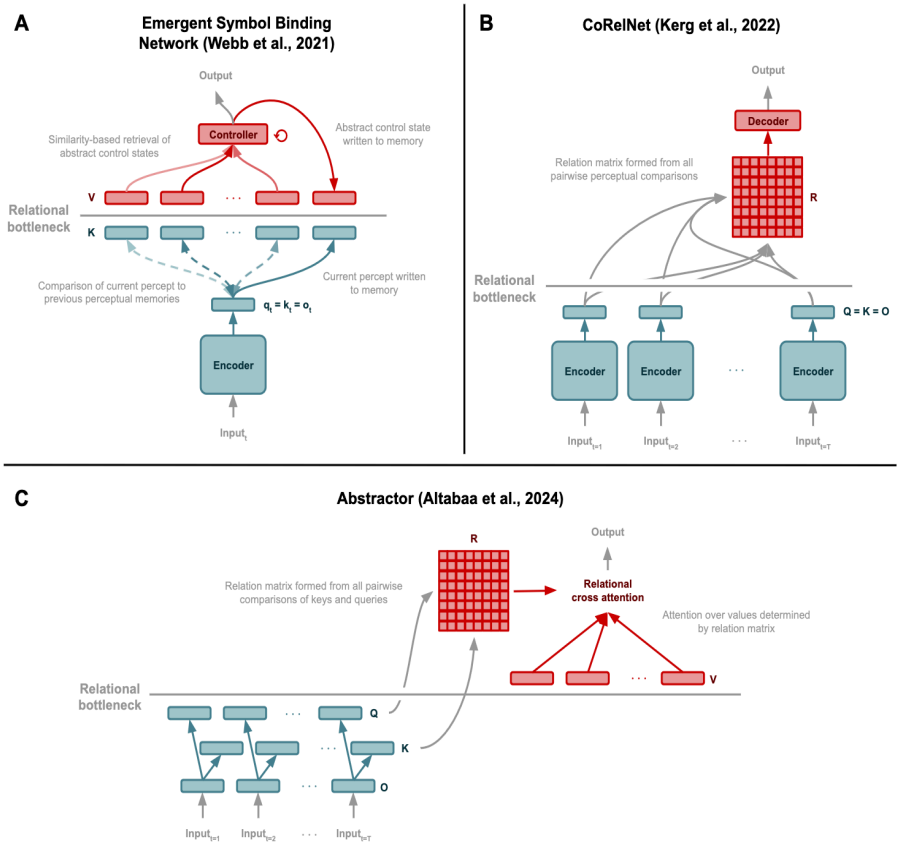

Figure 1.1: Illustration of different Relational Bottleneck architectures: A, ESBN [Webb et al., 2021], B CoRelNet [Kerg et al., 2022], C Abstractor [Altabaa et al., 2024]. Figure reproduced from [Webb et al., 2024].

energy entails maximizing the entropy of the agent's representation of the world, at least to some degree.

More recent work by [Frankland et al., 2021a] has elevated maximization of entropy as a central commitment. By combining this principle with a biologically plausible learning framework based on the interplay between slow and fast systems, it has shown how a variety of seemingly disparate psychophysical phenomena can be accurately and parsimoniously modeled.

The principle is also conceptually related to work on the Mutual Exclusivity Bias [Markman and Wachtel, 1988]. This principle refers to the observed tendency of people to assign new words to unknown concepts. Loosely speaking, these can both be understood as a maximal "spreading out" of representations, and some work [Frankland and Cohen, 2020] has modeled this formally using the mathematical formalism of a Determinantal Point Process (DPP). This can be understood in terms of entropy maximization by noting that the entropy of a multivariate Gaussian is equal (up to an affine scaling) of the log determinant of the covariance matrix. The DPP model, similarly, selects a maximally-informative subset of items based on maximizing the determinant of their similarity which, subject to Gaussian assumptions, is monotonically related to the entropy.

Symmetry. This has been most heavily studied in the domain of vision. For example the Gestalt psychologists posited that symmetry is a fundamental aspect of visual perception, identifying it as one of five basic perceptual "principles of organization" [Koffka, 1935, Wagemans et al., 2012]. Albeit, the forms of symmetry they considered were primarily restricted to axial reflections and rotations, and therefore constitute a less general sense of symmetry than we have considered so far in this section. Later experimental work has

shown that people can very efficiently detect axial symmetry [HB Barlow, 1979] and that the assumption of axial symmetry can bias peoples' similarity judgments [Freyd and Tversky, 1984]

Similar sensitivities to axial or reflectional symmetries have also been reported in the audio domain [Mongoven and Carbon, 2016], geometric shape perception [Sablé-Meyer et al., 2021] and working memory [Rossi-Arnaud et al., 2006, Peterson and Berryhill, 2013]. People also use symmetry as a guide for underconstrained problems, such as predicting the hidden side of a three-dimensional object [Yiu et al., 2022]. Developmental work has shown that sensitivity to symmetry arises at only a few months to a few years of age [Pornstein and Krinsky, 2985, Bornstein and Stiles-Davis, 1984]. This argues that the preference for certain basic forms of symmetry such as reflections may be innate or develop very early, and thus serve as a guiding principle for later learning. However, our work differs from these in an important way: whereas the above works are concerned with symmetries of specific forms or in specific modalities, we are more interested in the ability to learn or detect *general* symmetries, in potentially abstract spaces. It is thus unlikely that the forms of symmetry we study are "hard-wired" in the brain in the same way that those studied by the Gestalt psychologists may be.

Furthermore, many recent works in Machine Learning have integrated ideas from Group theory , in particular the ideas of invariance and equivariance, into neural network architectures [Bronstein et al., 2017a, Cohen and Welling, Gerken et al.]. Although such works do not directly speak to the psychological plausibility of this bias, they at least provide support for the idea of Symmetry as a normatively beneficial inductive bias for neural network learning.

The Relational Bottleneck. As the most recently articulated of the three principles, this has been the subject of less direct empirical psychological work. Nonetheless, it is firmly grounded in principles of human learning. For example, consider the original Relational Bottleneck style model, the ESBN [Webb et al., 2021]. This model uses a mechanism for key-value binding in an external memory module to implement the Relational Bottleneck. This mechanism bears a strong similarity to known processes of memory formation in the hippocampus, which is known for its ability for flexibly form "zero-shot" associations between arbitrary pairs of internal representations [Whittington et al., 2020]. The RB is also related at a high level to classical models of analogy formation such as the Structure mapping Theory [Gentner, 1983]. This theory models the human capacity for analogical reasoning as a process of (1) extracting symbolic, graph-structured representations of inputs, and (2) performing analogies by a process of mapping one graph to another via an approximate isomorphism. There is a strong affinity between these concepts, in that symbolic graph structures by their nature draw a hard distinction between "surface-level" information (nodes) and "purely relational" information. This connection is particularly relevant to the work described in Chapter 2: While the Structure Mapping Theory (as well as the graph-based model we will present in Chapter 2) can fit human data well, it is hampered by its reliance on hand-coded graph structures (i.e., step (1) is not systematically specified). The RB can be thought of as a way of imposing a pressure to extract such representations *through end-to-end learning*. The explicit factorization into surface-level and abstract relational information also mirrors a separation observed in the dorsal and ventral streams in primates, including humans [Webb et al., 2024].

Moreover, the Relational Bottleneck has been used to model the emergence of "number

sense" in development [Dulberg et al., 2021]. That is, when children learn to count they often show distinct stages in which they can reliably count to N but not larger. Past a small value (typically around $N = 5$), however, they are rapidly able to generalize to much larger values. As per this study, the ESBN displayed a qualitatively similar developmental time-course while other models such as Transformers and LSTMs did not, suggesting that the relational bottleneck may facilitate the learning and application of abstract rules.

In summary, each of these three principles provides a psychologically plausible inductive bias for facilitating strong generalizations: Maximum entropy encourages an agent to maintain maximal representational flexibility, the Relational Bottleneck enforces a strong attentional filtering in favor of information that is relational in character, and Symmetry encourages a representation of the domain which consists of multiple copies of a smaller subdomain glued together in a particular way. In the following chapters, we will demonstrate specifically how these biases in combination with each other can encourage out-of-domain and algorithmic generalizations.

What if the way we perceive a problem is already part of the problem?

Slavov Zizek

2

Analogical Reasoning with Relaxed Graph Matching

The material in this chapter is adapted from the previously-published work Segert et al. [2020].

2.0.1 Abstract

Analogical reasoning has classically been modeled using a graph matching framework. However, this problem is known to be computationally intractable, raising the question of finding approximate solution methods that perform well on the sorts of varied domains in which humans excel. In this work, we propose such an algorithm that is based on a continuous relaxation of the graph matching objective. We show the algorithm is capable of knowledge transfer in the domain of abstract multi-step reasoning tasks.

2.0.2 Introduction

Analogical reasoning, defined as the ability to transfer knowledge from one domain to another, is a crucial component of the ability to generalize out-of-domain [Gentner, 2003], an ability in which machines still lag behind humans. Modern approaches to analogy in artificial systems, such as variational autoencoders (VAEs), [Chen et al., 2018, Higgins et al., 2017] or word embeddings [Mikolov et al., 2013b] often rely on strong assumptions about the structure of the underlying domain, such as the existence of linearly disentanglable representations in the former, or of parallelogram structure in the latter. By contrast, classical models of human analogy, such as Gentner's Structure Mapping Theory (SMT) [Gentner, 1983], allow for domains of essentially arbitrary complexity, albeit the basic objective becomes computationally intractable in general. In this work, we propose an approach intermediate between these two extremes: capable of handling non linearly-separable data such as the SMT approach, but also differentiable and implementable by a neural network architecture such as those used in modern approaches to machine learning problems.

To do this, we assume that we are operating on a graph-structured domain. Many natu-

rally occurring domains can be cast in this way. For example, navigation and learning problems can be described in terms of Markov Decision Processes [Sutton and Barto, 2018] in which the graph is composed of possible states connected by the allowed actions. Similarly, conceptual knowledge can be described in terms of knowledge graphs [Gentner, 1983], in which the edges encode various kinds of relations between concepts. Complementing these approaches, recent work in neuroscience has suggested that certain graph-structured domains are represented in the brain via a successor representation [Dayan, 1993], which encodes the graph in an approximate vectorized form [Stachenfeld et al., 2017]. Crucially, our approach allows for such a representation and does not require exact specification of the underlying graph structure.

Following Gentner [1983], we formulate analogical inference as a graph matching problem. In this classical computer science problem, we seek to establish a map between vertices in a source graph to vertices in a target graph. Our key contribution lies in moving down the Marrian hierarchy [Marr and Poggio, 1976] from the Computational level to the Algorithmic level, by proposing a computationally tractable neural network architecture. Our approach yields approximate solutions using an appropriate continuous relaxation [Fiori et al., 2013, Vogelstein et al., 2014], allowing it to gracefully handle the kinds of inherently-nonlinear structural forms that commonly occur in the environment (circles, trees, etc.).

Once we have matched the graphs, we gain the ability to solve simple 4-term analogies "for free": to complete an analogy of the form "A is to B as C is to what?", we simply have to see where B gets sent in the obtained matching. However, learning a matching also allows us to transfer sequences of points from one domain to another, rather than just a single point as in a classical analogy. We show how this can be leveraged to perform multi-step

"planning by analogy" in the domain of abstract problem solving.

2.1 A neural network for graph matching

Assume we are given graphs G_1 and G_2, with vertex sets denoted $V_1 = \{v_1^1, \ldots, v_{|V_1|}^1\}$, $V_2 = \{v_1^2, \ldots, v_{|V_2|}^2\}$ and edge sets denoted E_1, E_2. The respective adjacency matrices are denoted $\mathbf{A}_1$ and $\mathbf{A}_2$. Note that in general $|V_1| \neq |V_2|$. Furthermore, we suppose that G_1 and G_2 contain certain distinguished points, the images of which are constrained a priori under any mapping. Formally, let $X = \{(v_{i_k}^1, v_{j_k}^2)\}_{k=1}^m$ be a collection of m ordered pairs of vertices in G_1 and G_2. We refer to these as Marked Points. Here $1 \leq i_k \leq |V_1|$ and $1 \leq j_k \leq |V_2|$ for all k. * As above, the marked points could correspond to the "A" and "C" terms of an analogy; we will also see examples in section 2.2 that naturally involve multiple pairs of marked points.

Following the basic Structure-Mapping theory, we pose the problem of finding a function $F : V_1 \to V_2$ that respects the connectivities of the two graphs as nearly as possible. That is, we want to minimize the number of pairs $(v, v') \in V_1 \times V_1$ for which the proposition $(v, v') \in E_1 \Leftrightarrow (F(v), F(v')) \in E_2$ is violated. We also require that $F(v_{i_k}^1) = v_{j_k}^2$ for every k. A straightforward argument show that for any such function F, the number $N(F)$ of such violating pairs is given by

$$N(F) = \|\mathbf{Q}_F \mathbf{A}_2 \mathbf{Q}_F{}^\top - \mathbf{A}_1\|_1. \tag{2.1}$$

The matrix is defined as $\mathbf{Q}_F[i, j] = 1$ if $F(v_i^1) = v_i^2$ and 0 otherwise, and the $L1$ norm of a

*In order to rule out inconsistent constraints, we require that $i_k \neq i_l$ for $k \neq l$.

matrix is defined as the sum of absolute values of its entries. A detailed derivation is given in section 2.7. Minimizing N is known to be NP-hard [Loiola et al., 2007].

Note that $\mathbf{Q}_F$ is a quasi-permutation matrix, meaning that every row has exactly one entry equal to 1, with the rest being 0. In particular, all entries are non-negative and each row sums to 1. A matrix with these properties is called Stochastic. This suggests that the problem can be "relaxed" by dropping the constraint that each entry of $\mathbf{Q}_F$ be either 0 or 1, and requiring only that the matrix be stochastic [Lyzinski et al., 2015, Vogelstein et al., 2014]. We denote the set of stochastic matrices $\mathbf{P}$ of shape $|V_1|$ rows by $|V_2|$ columns such that $\mathbf{P}[i_k, j_k] = 1$ for all k by SM_X.[†] This leads us to consider the loss function:

$$L_X(\mathbf{P} \mid \mathbf{A}_1, \mathbf{A}_2) = \|\mathbf{P}\mathbf{A}_2\mathbf{P}^\top - \mathbf{A}_1\|_1, \mathbf{P} \in SM_X. \tag{2.2}$$

The vertical bar notation is meant to indicate that $\mathbf{A}_1$ and $\mathbf{A}_2$ are regarded as inputs and can be replaced with other matrices of the appropriate sizes. We will discuss this further in the next section.

In contrast to the set of mappings $V_1 \to V_2$, SM_X is a continuous space, so we can use gradient-based techniques to perform the optimization. This is done by parametrizing the space of stochastic matrices with a simple neural network, with details given in section 2.8 [‡].

Next, there is the matter of obtaining a valid mapping once we have solved the relaxed version . This can be done in a variety of ways; here we will do so by simply taking a row-wise argmax of $\mathbf{P}$. More explicitly, given a solution $\mathbf{P}$ to the relaxed problem, define the

[†] In general, we use the letter $\mathbf{Q}$ for quasi-permutation matrices, and letter $\mathbf{P}$ for stochastic matrices.
[‡] Note that we solve each matching problem online, rather than employing a train-test paradigm.

associated mapping $F_{\mathbf{P}} : V_1 \to V_2$ by:

$$F_{\mathbf{P}}(v_i^1) = v^2_{argmax(\mathbf{P}[i])}, \tag{2.3}$$

where $\mathbf{P}[i]$ denotes the ith row of the matrix $\mathbf{P}$.

VARIANTS. While the above section dealt with the adjacency matrices of the graphs, the interpretation is not fundamentally changed if these are replaced with other types of affinity matrices on the vertices. Of particular interest to us is the successor representation $\mathbf{SR}_G = (1 - \gamma\overline{\mathbf{A}_G})^{-1}$, where $0 < \gamma < 1$ is a parameter and $\overline{\mathbf{M}}$ denotes the operation of normalizing each row of matrix $\mathbf{M}$ to sum to 1. As alluded to in section 2.0.2, this matrix has a well-known interpretation in the context of reinforcement learning [Dayan, 1993]. This leads us to consider the modified objective function $L_X(\mathbf{P} \mid \mathbf{S}_1, \mathbf{S}_2)$, where $\mathbf{S}_i = \mathbf{SR}_{G_i}$ and $\mathbf{P} \in SM_X$ as before.

Secondly, we consider a variant that often performs better in practice. Taking inspiration from the definition of a normalized graph Laplacian,we consider the modified objective:

$$\overline{L}_X(\mathbf{P} \mid \mathbf{A}_1, \mathbf{A}_2) := \|\overline{\mathbf{P}\mathbf{A}_2\mathbf{P}^\top} - \overline{\mathbf{A}_1}\|_1, \mathbf{P} \in SM_X, \tag{2.4}$$

where, as above, $\overline{\mathbf{M}}$ denotes the matrix obtained by normalizing each row of the matrix $\mathbf{M}$ to sum to 1. These two modifications can be applied in tandem (i.e., replace the adjacency matrix with the SR and normalize both terms), with the corresponding loss denoted $\overline{L}_X(\mathbf{P} \mid \mathbf{S}_1, \mathbf{S}_2)$. In what follows, we will always assume the loss function to be of this form and, for the successor representation, $\gamma = .8$ unless otherwise specified.

Finally, the effect of different relaxations than in eq. (2.2) is discussed in sections 2.5 and 2.6.

2.2 Applications to knowledge transfer

We now formalize how the model may be used for knowledge transfer of multistage planning. Suppose we have two state graphs G_1 and G_2 corresponding to two different planning or reasoning tasks. These tasks come with distinguished nodes $s_1, t_1 \in V_1$ and $s_2, t_2 \in V_2$ corresponding respectively to the starting and target states of each task. Finally, we suppose that we have a solution to the first task, namely a path of vertices $s_1, v_1, \ldots, v_k, t_1$ of vertices on G_1 from the initial state to the target state. To transfer our knowledge, we then solve the matching problem from $G_1 \to G_2$, subject to the constraints that t_1 maps to t_2 and s_1 maps to s_2, in the way described in section 2.1. We then obtain a candidate solution to G_2 by simply mapping each vertex v_i on the path according to the solution matrix $\mathbf{P}$. That is, we obtain a candidate path $\tilde{v_1}, \ldots, \tilde{v_k}$ from s_2 to t_2 given by $\tilde{v_i} = F_{\mathbf{P}}(v_i)$, with $F_{\mathbf{P}}$ as in section 2. Note that, a priori, this need not be a valid path on G_2 (i.e. successive vertices might not be connected); however we will see that when the graphs have common structure the paths obtained in this way do indeed tend to be valid.

As a first illustration, we consider the well-known Tower of Hanoi task, in which the player is presented with a set of disks, each of a different size and distributed over some number of pegs, and given the goal of moving all of the pegs to a target disk (the target state) under the constraint that only one disk can be moved at a time (to any peg) and a larger disk can never be placed over a smaller one on a given peg. For simplicity, we consider two cases, both of which have three disks and three pegs. In the first case, all of the disks

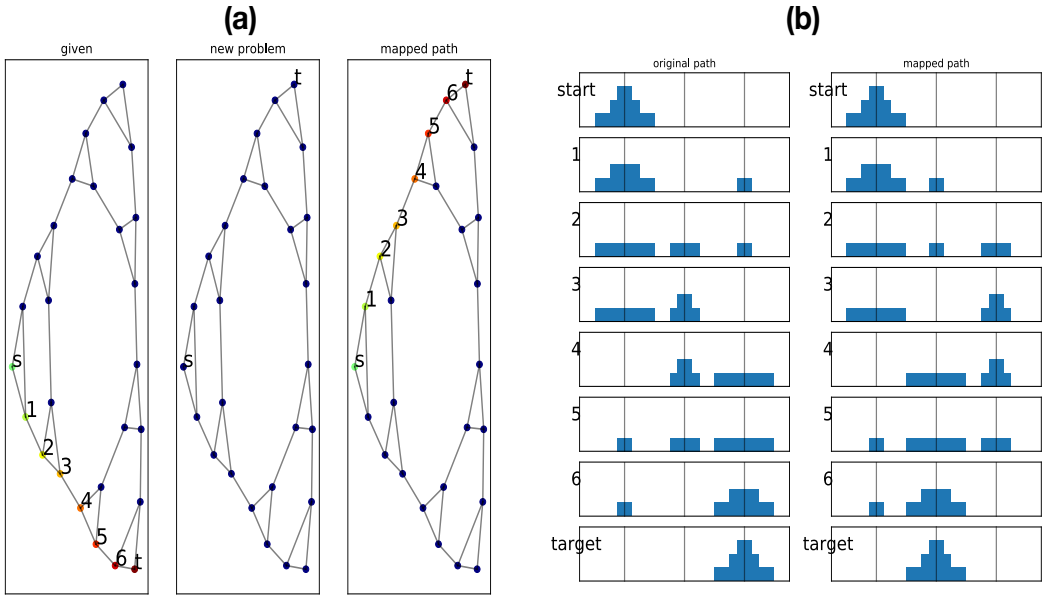

Figure 2.1: Mapping a solution to the original Tower of Hanoi task to a solution to the modified task. (a) Left: Original task with solution path indicated. Note "s" and "t" stand for "start" and "target" respectively. Middle: Modified task. Right: The image of each point on the path in the left image under the obtained mapping. (b) Rendering of the states in the original and mapped path from part (a). (Figure best viewed in zoom).

start on the leftmost peg, and the goal is to move them all to the rightmost peg. Second, we consider a simple variant in which the objective is instead to move all disks from the first peg to the middle one. In both cases, a graph can be described with a node for each valid configuration of the disks, and edges corresponding to valid moves. In the notation used above, the two graphs G_1 and G_2 are identical, and the initial states s_1 and s_2 coincide, but the target states t_1 and t_2 are different. As shown in fig. 2.1, the algorithm discovers a mapping that carries a solution to the first task into a valid solution of the second.

As a slightly more complex example, we let G_1 be the state graph corresponding to the "Farmer/Goat" problem and G_2 that of the "Missionaries and Cannibals" problem. We give precise descriptions of these puzzles in section 2.9. These are both "river crossing" puzzles in which the objective is to transport various parties across the river using a small boat,

subject to constraints about which parties can be left alone with each other. Similar to the previous example, the nodes of the graphs correspond to valid configurations of the various parties on the two shores. It was shown in [Holyoak and Thagard, 1989] that children who had first learned how to solve the simpler Farmer/Goat problem were were more successful in finding a solution to the more complex Missionaries/Cannibals problem. However, no such transfer was obtained in the opposite direction.

In fig. 2.2 (shown in section 2.4) we show how a matching from the Farmer/Goat graph to the Missionaries/Cannibals graph indeed carries a solution of the former to a solution of the latter (in fact there are two solutions to the Farmer/Goat problem, both of which have this property). However, as with people, the matching obtained in the opposite direction is of low quality. Indeed, of the 768 possible shortest solution paths to the Missionaries/Cannibals problem, only 4.7% are mapped to valid solutions of the Farmer/Goat problem. An example solution path obtained by this mapping is shown in fig. 2.3 (in section 2.4).

Finally, we provide a different type of example that shows how our algorithm can exploit state space abstraction for efficient problem solving, by relating the simpler, abstract representation of a state space to a more complex but veridical representation of that space. Often problems are solved hierarchically: one first solves a suitably abstracted version of the problem and then "fills in the details" when solving the actual problem (this can be repeated at several levels of abstraction) [Botvinick, 2012]. Spatial navigation tasks provide an example: if one's goal is to travel from New York to Los Angeles, then the initial planning stage would probably consist of steps like "Drive to JFK, board plane, ..." etc. rather than "Move car into left lane on 2nd Avenue to merge onto I-495 East, ..." etc. In order to implement such a strategy, it is necessary to have a correspondence between the abstracted

state space and the original.

As an example of finding such a correspondence, we return to the Tower of Hanoi. The state graph for this problem has a recursive structure: the graph for n disks can be obtained from the graph for $n - 1$ disks by replacing each node in the smaller graph with a clique of size 3. We see in fig. 2.4 (shown in section 2.4) that the algorithm is able to reveal this structure in the course of mapping the larger graph onto the smaller one. We can interpret this mapping as a coarsening of the larger graph such that the groups correspond exactly to these cliques.

2.3 Discussion

We have described a differentiable algorithm that is capable of structure mapping in wide range of problems, and illustrated its use in multi-step analogical reasoning and problem solving. The algorithm draws inspiration both from cognitive psychology (representing problem spaces) and mathematical optimization (approximating solutions to combinatorial problems). Many problems that people must solve, such as language processing [Barton et al., 1987] and spatial navigation [Macgregor, 2011], are intractably combinatorial; our approach gives a guide for how approximate solutions to them may be found by mapping known structure to novel situations. This exploits the fact that, although graph matching is NP hard in the worst cases, not all *cases* are necessarily so. We argue that the kinds of problems that people encounter and are able to solve are not the "worst case" ones, but rather closer to the structured examples considered in this paper. Our method is thus parsimonious in that it (1) works well in these structured cases and (2) these structures are not explicitly "built in" to the algorithm itself. Furthermore, our approach complements recent

work on graph similarity learning,in particular Li et al. [2019], in which the authors use a graph network to learn a scalar matching-based similarity measure, rather than an explicit node-wise correspondence as we do.

Finally, there are several possible extensions that we plan to address in future work, for example extension to the case in which the solution paths have different lengths, and a more systematic treatment of graphs with recursive or self-similar structure.

2.4 Additional figures

fig. 2.2 shows the mapping of the Farmer/Goat problem onto the missionaries and cannibals problem. Here each solution of the simper problem is mapped to a valid solution of the more complex problem. fig. 2.3 shows the image of an example solution path under the matching in the reversed direction. Unlike in the previous case, the image is not a valid solution path.

fig. 2.4 shows the mapping of the Hanoi graph onto the graph of the 2-disk variant. We see that the mapping reveals the recursive structure by mapping all points in the indicate cliques to the corresponding nodes on the smaller graph.

2.5 Comparison with indefinite Frobenius relaxation

In [Lyzinski et al., 2015] a similar matching problem is considered. The authors prove optimiality results for the objective function $||\mathbf{P}\mathbf{A}_2\mathbf{P}^T - \mathbf{A}_1||_F^2$, $P \in DSM$, with DSM denoting the space of doubly-stochastic matrices and the subscript F denoting the Frobenius norm of a matrix. They term this the "Indefinite loss function". In view of their theoretical results, it is interesting to compare our L1 loss with theirs.

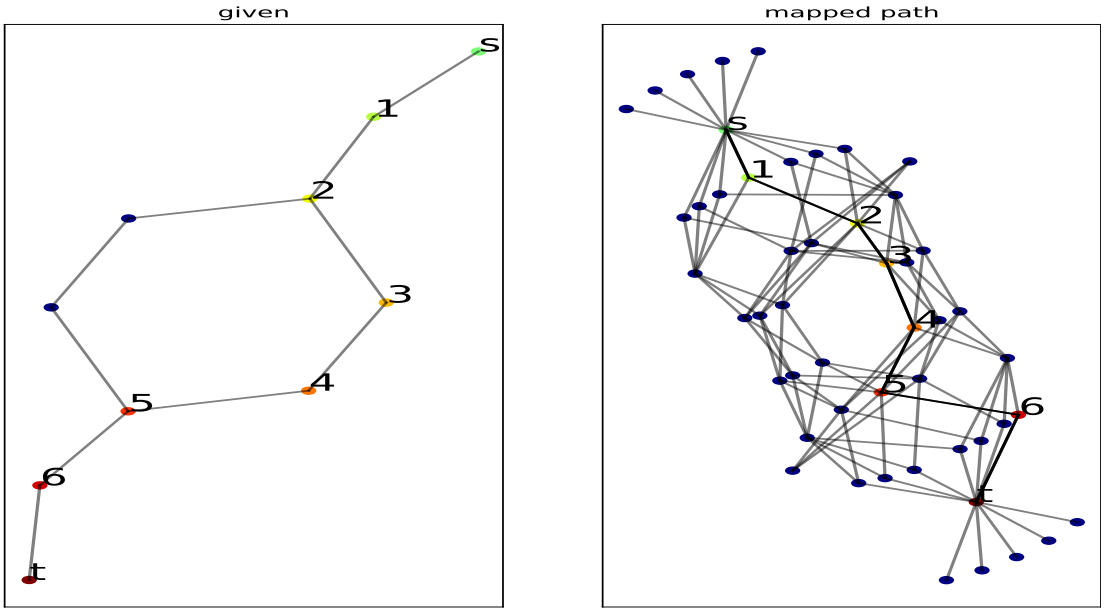

Figure 2.2: Left: The state graph for the Farmer/Goat problem with a solution path indicated. Right: The state graph for the Missionaries/Cannibals problem, with the images of the solution path indicated. Note that the mapped path is a valid path on the target graph (and the same is true of the alternative solution path). Note that the mapping here uses the adjacency matrices rather than the Successor Representations. As in Figure fig. 2.1, "s" and "t" denote the starting and target states in each problem.

Note that their assumptions differ from ours in the following ways: (1) they assume a particular random graph model for G_1 and G_2, while we consider highly structured graphs and (2) they assume G_1 and G_2 to have the same size.

In particular, in our setup, it is impossible to use the space of doubly stochastic matrices. This is because we do not assume the graphs to have the same size, and a doubly stochastic matrix is necessarily square. However, if we drop this assumption and assume P to be only row-stochastic, then we can use their loss function as a drop-in replacement for ours.

In the next sections, we directly compare the obtained results with our own loss function.

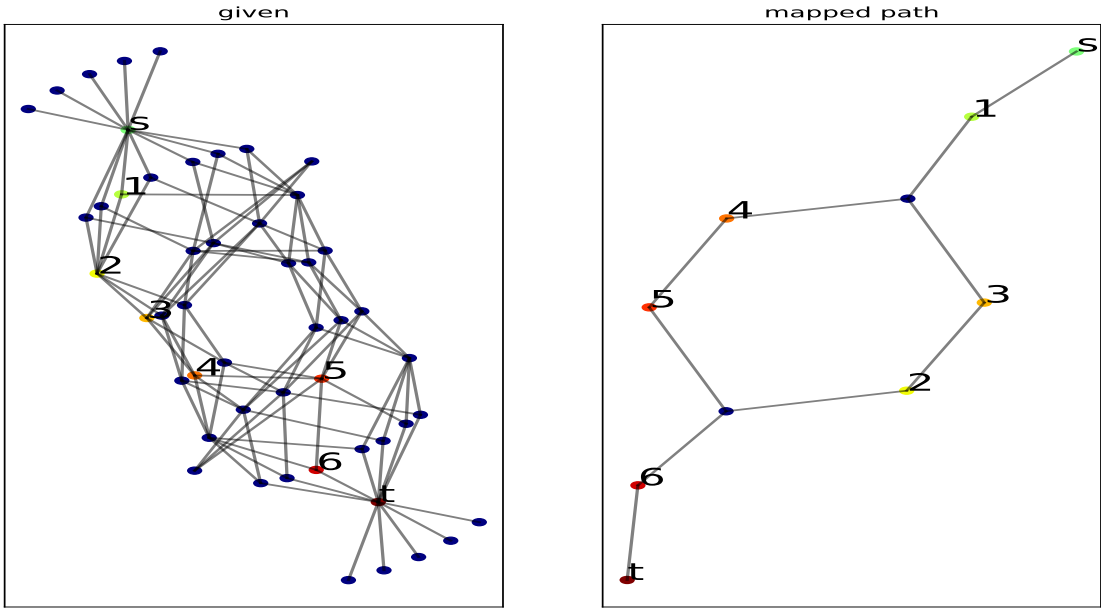

Figure 2.3: Similar to fig. 2.2, but for the mapping in the reverse order. An example solution path is shown for the Missionaries/Cannibals problem on the left, and the corresponding mapped path to the Farmer/Goat problem on the right. Note that the mapped path is not a valid solution to the Farmer/Goat problem.

2.5.1 Figures with Frobenius norm

Here, we use the same figures as in the main text and section 2.4 , but using the "indefinite loss" $\|\mathbf{P}\mathbf{A}_2\mathbf{P}^T - \mathbf{A}_1\|_F^2$ as in [Lyzinski et al., 2015]. In fig. 2.5, fig. 2.6 and fig. 2.7, we reproduce the settings of fig. 2.1, fig. 2.2 and fig. 2.4, except using the indefinite loss in place of our L_1 loss. We see that the matchings obtained in fig. 2.5 and fig. 2.7 of worse quality compared to their L1 counterparts, with the results for the Farmer/Goat in fig. 2.6 being similar to the L1 case.

2.6 Results on Lines, Circles, Trees

We consider the problem of solving simple 4-term analogies on model linear and non-linear spaces. In what follows, let G be either a directed linear graph, a directed circle graph, or

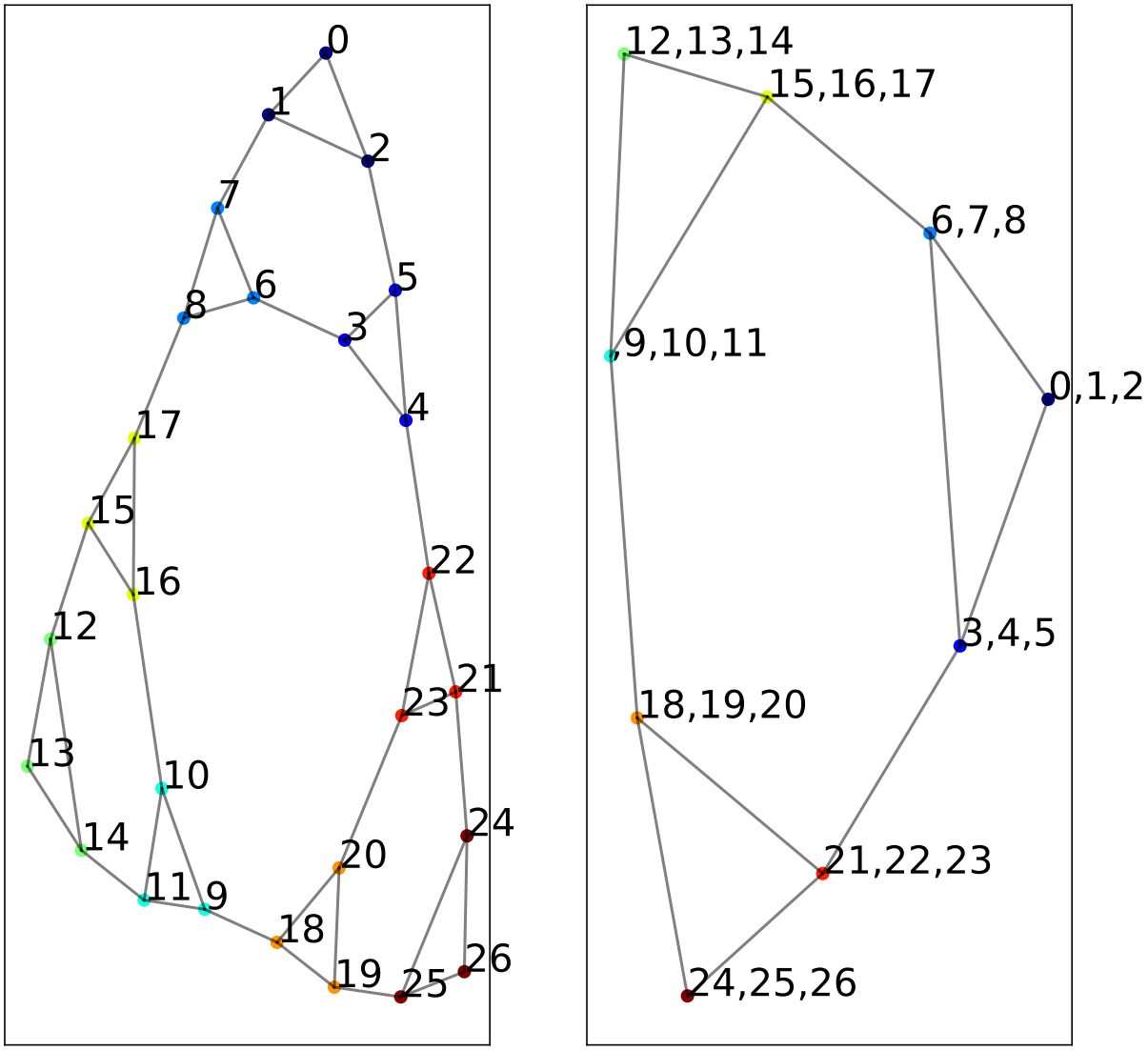

Figure 2.4: Mapping the state graph for Towers of Hanoi (left) to that of two-disk variant (right). Labels on left graph are arbitrary, and the image of each point on the right graph is shown. Colors are used for additional visual aid. Colors on right graph are arbitrary, and points on left graph are colored according to their image under the mapping. Marked points are the 3 extremal corner points on each graph (which correspond to states in which all disks are on top of a single peg).

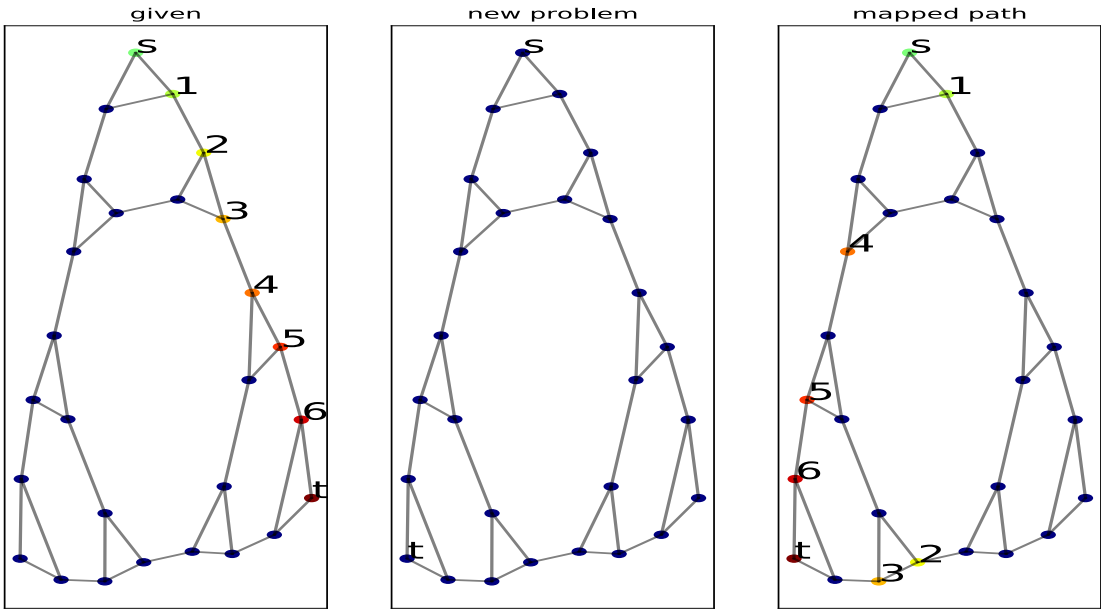

Figure 2.5: Same as fig. 2.1, but with indefinite loss function. Comparison with fig. 2.1 shows the L1 loss yields a better matching.

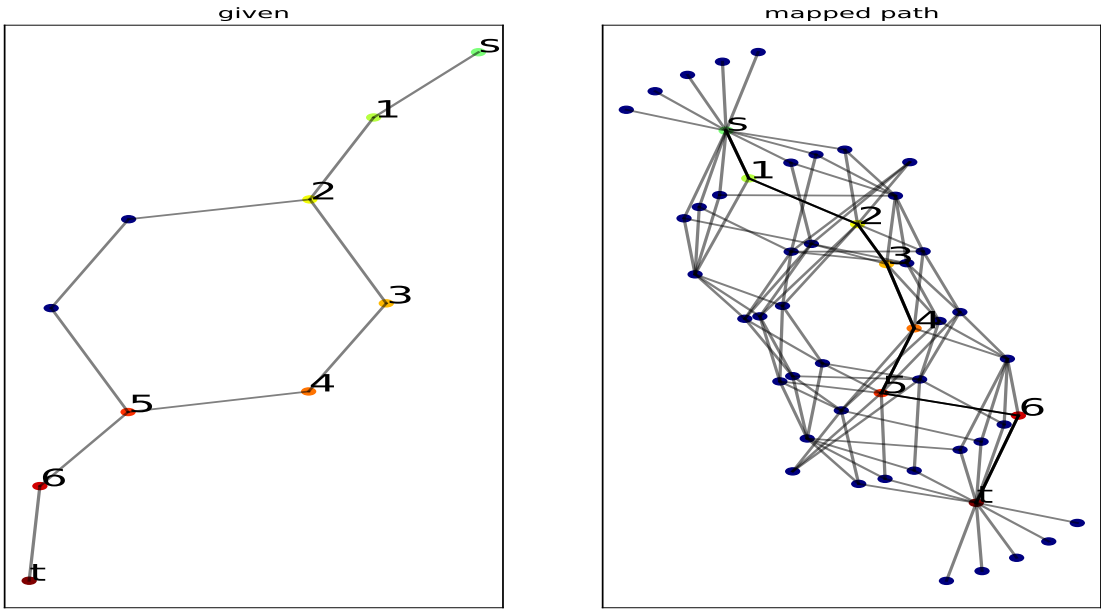

Figure 2.6: Same as fig. 2.2, but with indefinite loss function. Results are comparable to L1 loss.

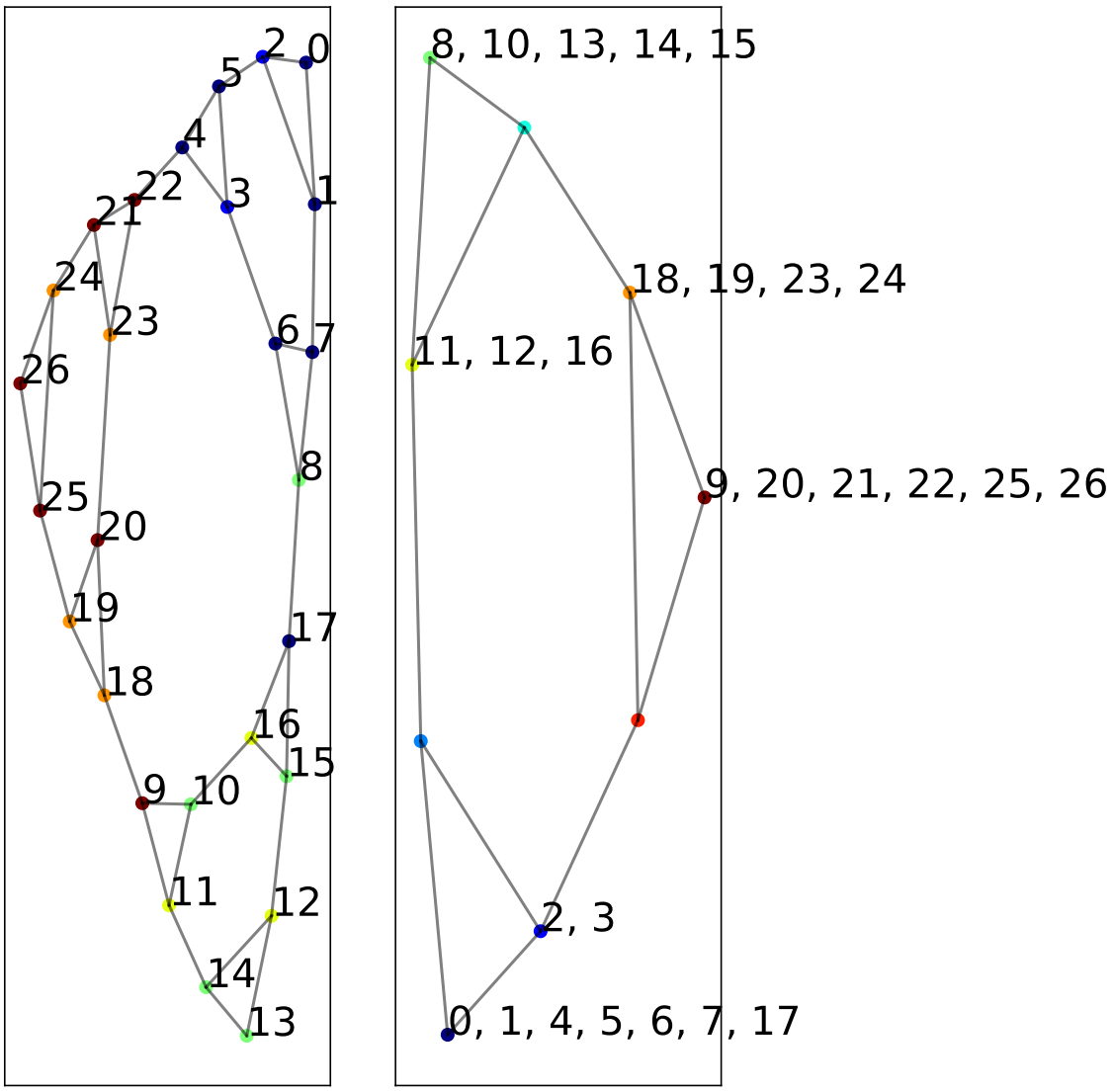

Figure 2.7: Same as fig. 2.4, but with indefinite loss function.Comparison with fig. 2.4 shows the L1 loss yields a better matching.

a directed binary tree. Denote by d_G the undirected geodesic distance function on G (i.e., the length of the shortest undirected path between two points). Denote by S_γ the successor representation of G, defined using the directed adjacency matrix of G. As in the main text, we set $\gamma = .8$ in what follows.

On these experiments, we used a binary tree with 63 vertices, a linear graph with 31 vertices, and a circle with 31 vertices and 100 analogies for each graph. Each matching problem is solved using 10 random intializations.

2.6.1 Generating analogies

To generate analogies, we choose A, B, C randomly from the vertices of G subject to the constraints that $A \neq B$ and $A \neq C$, $d_G(A, B) \leq M$ and the analogy has at least one valid answer. Here M is set to 4 for the tree and 8 for the other two graphs.

In the case of lines and circles, each analogy has an unambiguous correct answer, denoted by D_{true}. In the case of a tree, an analogy may have several acceptable answers. For example, if A is a parent of B, then either child of C would be an acceptable answer to the analogy. In general, we consider D to be correct if (1) $d_G(A, B) = d_G(C, D)$ and (2) the shortest path from C to D has the same pattern of forward/backtracking with respect to the edge directions on G as the shortest path from A to B. We thus propose the following unified measure of analogy quality on the three graphs: Consider the shortest undirected path from A to B. Associated to this path is a binary sequence which encodes whether or not each step in the path goes "against the grain" with respect to the edge directions. The path from C to D_{pred} has a similar sequence. We then define the error of the analogy to be the normalized levenshtein distance between these two sequences. This has the property

that the values lie in $[0, 1]$ with 0 indicating a perfect fit. [§]

2.6.2 Defining the matching problem

Define G_1 to be the induced subgraph of G containing all nodes no more than distance $d_G(A, B)$ from A and G_2 the induced subgraph contains all nodes no more than that distance from C. Let $\mathbf{S}_1$ and $\mathbf{S}_2$ denote the matrices obtained by keeping only the rows and columns of $\mathbf{S}$ corresponding to points in G_1/G_2.

We then solve the marked matching problem $\overline{L}_{\{(A,C)\}}(\mathbf{P}|\mathbf{S}_1, \mathbf{S}_2)$. We also compare this to the indefinite loss function $\|\mathbf{P}\mathbf{A}_2\mathbf{P}^T - \mathbf{A}_1\|_F^2$ from [Lyzinski et al., 2015] and the variant of replacing the L1 norm in ours with the Frobenius: $\|\overline{\mathbf{P}\mathbf{S}_1\mathbf{P}^T} - \overline{\mathbf{S}_2}\|_F^2$.

As described previously, we complete the analogy by taking an argmax of $\mathbf{P}[B]$. Let us denote this point by D_{pred}.

2.6.3 Results

Comparison of our loss with the indefinite loss is shown in fig. 2.8. Our loss $\overline{L}_{(A,C)}(\cdot|\mathbf{S}_1, \mathbf{S}_2)$ achieves highest quality on the tree and circle, and is only slightly worse than the Frobenius variant on the line. The worst performing variant is the $L1$ norm combined with the adjacency matrix.

2.7 Derivation of matrix form for matching problem

Continuing the notation from section 2.1, we see that

[§]Note that on these graphs each pair of points has a unique shortest path so there is no ambiguity in this measure. The unique exception is the pair of antipodal points on the circle, however this does not occur in our examples due to the restriction on the distance between the two terms in the analogy.

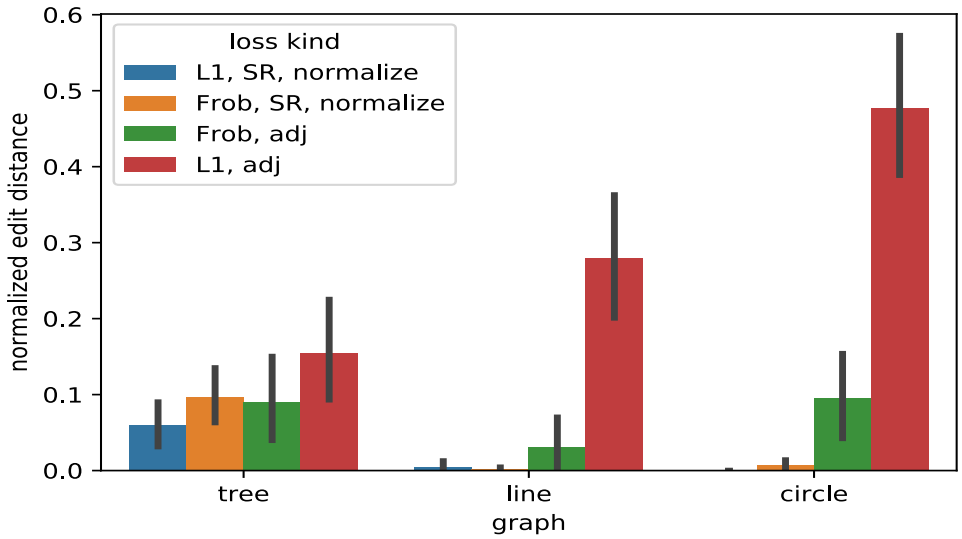

Figure 2.8: Comparison of quality of generated analogies using different loss functions on different model graphs (lower is better). Blue is our default loss $\overline{L}(\cdot|\mathbf{S}_1, \mathbf{S}_2)$ as described in Section 2. Green is "indefinite loss" of [Lyzinski et al., 2015]. Orange is variant of our default loss, replacing L1 with Frobenius norm. Red is the loss from eq. (2.2). Our loss (blue) generally outperforms the indefinite loss on these examples.

$$N(F) = \Big|\{(v_1, v_2) : (v_1, v_2) \in E_1, (F(v_1), F(v_2)) \notin E_2 \text{ or } (v_1, v_2) \notin E_1, (F(v_1), F(v_2)) \in E_2\}\Big| \tag{2.5}$$

$$= \Big|\{(v_1, v_2) \in V_1 \times V_1 : \mathbf{1}_{(v_1,v_2)\in E_1} \neq \mathbf{1}_{(F(v_1),F(v_2))\in E_2}\}\Big| \tag{2.6}$$

$$= \sum_{(v_1,v_2)\in V_1\times V_1} \Big|\mathbf{1}_{(v_1,v_2)\in E_1} - \mathbf{1}_{(F(v_1),F(v_2))\in E_2}\Big|, \tag{2.7}$$

where $\mathbf{1}_A$ is equal to 1/0 depending on whether the proposition A is true/false.

Now recall the matrix matrix $\mathbf{Q}_F$ defined by $\mathbf{Q}_F[i,j] = 1$ if $F(i) = j$ and 0 otherwise. We

see that

$$(\mathbf{Q}_F \mathbf{A}_2 \mathbf{Q}_F^T)[i,j] = \sum_{k,l} \mathbf{Q}_F[i,k]\mathbf{A}_2[k,l]\mathbf{Q}_F[j,l] \tag{2.8}$$

$$= \sum_{k,l} \mathbf{1}_{F(i)=k} \mathbf{1}_{F(j)=l} \mathbf{1}_{(k,l)\in E_2} \tag{2.9}$$

$$= \mathbf{1}_{(F(i),F(j))\in E_2}. \tag{2.10}$$

Since $\mathbf{A}_1[i,j] = \mathbf{1}_{(v_i,v_j)\in E_1}$, we see that $\|\mathbf{Q}_F \mathbf{A}_2 \mathbf{Q}_F^T - \mathbf{A}_1\|_1 = N(F)$ as was to be shown.

2.8 Solving the continuous problem.

To minimize the continuous objective eq. (2.2), let $\mathcal{M}_{n,m}$ denote the space of all matrices with n rows and m columns. We parametrize SM_X by the map $Mat_{|V_1|-|X|,|V_2|} \to SM_X$ which takes a softmax across each row and then inserts the clamped rows specified by X in the appropriate order. This allows us to consider L_X as a function on $Mat_{|V_1|-|X|,|V_2|}$; because this is a linear space, we can minimize the objective using standard gradient descent. The variants in the "Variants" subsection are treated in the same way. We use 20 random initializations for each problem.

2.9 Description of Farmer/Goat and Missionaries/Cannibals

As in [Holyoak and Thagard, 1989], the Farmer/Goat problem consists of a farmer, a wolf, a goat, and a cabbage. All parties start on the same shore of the river, and the objective is to transport them all to the opposite shore. The boat can only hold one item in addition to the farmer. The wolf cannot be left alone with the goat, nor can the goat with the cabbage.

The Missionaries/Cannibals problem consists of 3 missionaries and 2 cannibals which must all cross the river. The boat can hold at most two parties, and the missionaries can never be outnumbered the cannibals on either shore. Additionally, it is disallowed to make a trip across the river with an empty boat.

It is utterly implausible that a mathematical formula should make the future known to us, and those who think it can would once have believed in witchcraft.

Daniel Bernoulli

3

Maximum Entropy Function Learning

The material in this chapter is mostly adapted from the previously published work [Segert and Cohen, 2022a]. The exception is Section 3.8, which contains results that have not been previously published.

3.0.1 ABSTRACT

Understanding how people generalize and extrapolate from limited amounts of data remains an outstanding challenge. We study this question in the domain of scalar function learning, and propose a simple model based on the Principle of Maximum Entropy [Jaynes, 1957]. Through computational modeling, we demonstrate that the theory makes two specific predictions about peoples' extrapolation judgments, that we validate through experiments. Moreover, we show that existing Gaussian Process models of function learning cannot account for these effects. We also demonstrate that the MaxEnt model can yield close fits to empirical data in a generative extrapolation paradigm.

3.1 Introduction

One of the most impressive aspects of human intelligence is the ability to detect and extrapolate a variety of abstract patterns that commonly occur in the world. Here, for tractability, we focus on patterns that can be expressed using one-dimensional functions, the class of which is still sufficiently rich to encompass many real-world tasks, such as deciding whether to invest in a certain stock, or predicting how hard to hit a golf ball so that it will travel a certain distance. Work on *function learning* [McDaniel and Busemeyer, 2005a, DeLosh et al., 1997, Bott and Heit, 2004] has catalogued the form of several inductive biases that people employ in this setting, including a preference for positive linear forms [Kwantes and Neal, 2006a] or compositional construction from a small number of simple forms ("atoms;" [Schulz et al., 2017]). It is an outstanding open question whether this collection of biases is sufficient to fully characterize peoples' extrapolation judgements of scalar func-

tions. that commonly occur in the world. Here, for tractability, we focus on patterns that can be expressed using one-dimensional functions, the class of which is still sufficiently rich to encompass many real-world tasks, such as deciding whether to invest in a certain stock, or predicting how hard to hit a golf ball so that it will travel a certain distance. Work on *function learning* [McDaniel and Busemeyer, 2005a, DeLosh et al., 1997, Bott and Heit, 2004] has catalogued the form of several inductive biases that people employ in this setting, including a preference for positive linear forms [Kwantes and Neal, 2006a] or compositional construction from a small number of simple forms ("atoms;" [Schulz et al., 2017]). It is an outstanding open question whether this collection of biases is sufficient to fully characterize peoples' extrapolation judgements of scalar functions.

Most current models of function extrapolation cast it as a Bayesian inference problem [Lucas et al., 2015a, Schulz et al., 2017, Wilson et al., 2015b], the solution to which can be expressed using the mathematical formalism of Gaussian Processes (GPs) [Rasmussen and Williams, 2006a]. In contrast, we propose a novel, basic inductive bias in the context of function extrapolation, the form of which is based on the Principle of Maximum Entropy ("MaxEnt;" [Jaynes, 1957]). Put simply, we posit that peoples' extrapolations arise as samples from the "maximally indeterminate" distribution that is consistent with certain observed structural features of the function. This proposal is supported by the key role that the MaxEnt principle has played in other contexts within cognitive science. For example, an early such application was the work of Myung [Myung, 1994, Myung and Shepard, 1996], showing that certain classical categorization models could be seen as special cases of the MaxEnt principle. The principle has also been used to derive the functional form of psychoeconomic weighting functions [Bhui and Gershman, 2018], and to provide a normative

account of many perceptual capacity limitations [Frankland et al., 2021a].

Here, we leverage a classic theorem of Burg 1967 to show that MaxEnt functional extrapolation takes a relatively simple and psychologically plausible form. To test this hypothesis, we derive two specific behavioral predictions from it, that we term *roughness calibration* and *typicality preference*. In a pair of experiments, we test whether people show these effects when making extrapolation judgements, and contrast these with predictions of GP-based models in our experimental setting. We find that the data are consistent with predictions of the MaxEnt model and not existing GP-based models. We also consider the predictions of the MaxEnt model in a generative, rather than discriminative, extrapolation paradigm and show that it can provide close fits to an empirical dataset collected by [Ciccione and Dehaene, 2021], with no hyperparameter tuning. In light of these results, we discuss how the MaxEnt model and GP model may potentially complement each other in a Resource Rational analysis of function learning [Lieder and Griffiths, 2020].

3.1.1 Maximum Entropy extrapolation

In typical function extrapolation experiments [McDaniel and Busemeyer, 2005a, DeLosh et al., 1997, Schulz et al., 2017], the experimenter chooses some function $f : \mathbb{R} \to \mathbb{R}$, and the participant is shown the values of f in a bounded region of the domain. The participant is then tasked with inferring the values of the function at points that lie outside of the region. For simplicity, we make the additional assumption that the points are equally spaced along the x axis; this permits rescaling the domain, if necessary, so that the points lie on the positive integers. Thus, the participant is shown the values $f(1), f(2), \ldots, f(N)$, and then is tasked with inferring the values $f(N+1), \ldots, f(N+M)$.

This may be naturally cast as a problem of probabalistic inference, in which the target of the inference is the distribution of future values of f, conditional on the observed ones. In general, MaxEnt posits that the preferred solution to such an inference problem takes the form that requires the least amount of additional information to satisfy the constraints imposed by the data, among all those that are consistent with those constraints [Jaynes, 1957]; this can be thought of as making the smallest representational commitment needed to accommodate the data, so that new information can be accommodated with a minimum of disruption to existing representations. More formally, the inference may be cast as a constrained maximization problem, with the entropy of the distribution being the objective, and the observed values of certain statistics (e.g. means or covariances) being the constraints. For the case in which the data correspond to values of a scalar function, Burg's theorem implies that the solution to the optimization takes a remarkably simple form, as shown below:

Theorem 4. *[Burg, 1967] Let* $(\hat{Y}_1, \ldots, \hat{Y}_k)$ *be an observation of a time series**, *and let let* $\alpha_j = \sum_i \hat{Y}_i \hat{Y}_{i+j}, 1 \leq j \leq L$ *be the empirical lagged covariances, for some fixed* L. *The maximal entropy distribution of* $(Y_1, \ldots, Y_k)$, *subject to the constraints* $\sum_i Y_i Y_{i+j} = \alpha_j$ *is given by*

$$Y_i = \sum_{j=1}^{L} w_j Y_{i-j} + \varepsilon_i$$

for some choice of coefficients w_i. *Here* ε_i *are iid Gaussian variables with mean* 0 *and variance* σ^2.

(For a proof, see [Cover and Thomas, 1991], Section 12.6.) Burg's Theorem thus sug-

*Note that a time series is merely a notational variant of the ordered vector of values $(f(1), f(2), \ldots, f(N))$ of a function as formulated above.

gests a natural way to estimate the values of f on points outside the domain of definition, in both generative and evaluative paradigms. In the generative case, the value at $N+1$ is estimated by applying the autoregressive weights to the last L observations, $\hat{f}(N+1) := \sum_{j=1}^{L} w_j f(N-j+1)$ (optionally with normal additive random noise). We may then append this value to the vector of observed values and repeat the process iteratively, obtaining estimates of $\hat{f}(N+k)$ for arbitrarily large k. In the evaluative case (e.g., multiple choice), the participant must judge the quality of a proposed completion $(\hat{f}(N+1), \ldots \hat{f}(N+M))$ of f. This may be done in a in a manner that uses the same machinery as the generative case; we describe this process in greater detail further below.

3.2 Gaussian Process models of function learning

The MaxEnt model differs from approaches to modeling function learning that use Gaussian Processes [Schulz et al., 2017, Lucas et al., 2015a, Wilson et al., 2015b]. In this section, we briefly outline the key assumptions of the GP framework.

Formally, a Gaussian process (GP) defines a probability distribution on the space of all possible one-dimensional functions $f : \mathbb{R} \to \mathbb{R}$ [Rasmussen and Williams, 2006a]. This assumes that, for any finite set of observations of values of the function $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$, that elements of the vector $(f(x_1), \ldots, f(x_n))$ are distributed according to a zero-mean multivariate Gaussian, with the covariance structure satisfying $K(x_1, x_2) = Cov(f(x_1), f(x_2))$, where K is a referred to as a *kernel function*.

Following the previous convention, we assume that the x_i values are always given on the set of positive integers, $x_i = i$. In this case, a GP with kernel function K is equivalent to a mean-zero multivariate normal distribution with covariance $C_{ij}^{K} = K(i, j)$. Thus, given the

first N values of the function $f(1), \ldots, f(N)$, and a kernel function K, the posterior over later values $(f(N+1), \ldots, f(N+M))$ is given by conditionalizing the distribution $N(0, C^K)$ on the values of the first N components. The posterior is also Gaussian, the mean and variance of which can be explicitly expressed in terms of C^K and the vector $(f(1), \ldots, f(N))$ [Lucas et al., 2015a, Rasmussen and Williams, 2006a].

In the context of function learning, kernel functions are used to encode an agent's prior beliefs about likely kinds of functions. Given such priors, the problem of inferring the values of a function at a new set of points can be recast as a form of rational probabilistic inference [Lucas et al., 2015a]. In order to complete the specification of such a model, it is necessary to posit a specific kernel or collection thereof. A common choice is the Radial Basis Function (RBF) kernel, defined by the formula $K_\sigma^{rbf}(x, y) = e^{-(x-y)^2/\sigma}$ [Lucas et al., 2015a, Schulz et al., 2017]. Samples from this distribution do not generally satisfy any global parametric form, but rather obey a form of local statistical regularity: the curves are smooth, but tend to "meander" randomly due to the lack of long-range correlations. This kernel has thus been used as a model for a basic smoothness prior. Other kinds of kernels, that encode more specific forms of structure, such as linearity or periodicity, are also possible [Schulz et al., 2017]. A final type of kernel, which is common in machine learning applications but less common in studies of human function learning, is the Matern kernel [Rasmussen and Williams, 2006a]. This kernel is a generalization of the RBF, which can generate curves that are locally "rough". The kernel contains an additional hyperparameter ν that controls this degree of roughness (see Figure 3.1). We use this kernel to generate stimuli for Experiment 1. This kernel has been previously used in human function learning experiments by [Schulz et al., 2015]. In our experiments, all stimuli were generated by sampling from Gaussian

kernels, either RBF or Matern.

3.3 Modeling of multiple choice extrapolations

Following [Schulz et al., 2017], the experiments we report below used a multiple-choice (evaluative) extrapolation paradigm. In this paradigm, participants are shown a static graphical representation of a prompt curve $y_{prompt} = (f(1), \ldots, f(N))$ as well as of several candidate completions, and tasked with selecting the most plausible completion. We denote the ith candidate completion by y_i, which is represented as a vector of potential values of f at the unseen points $N+1, \ldots, N+M$. In all experiments, we take $N = M = 100$.

3.3.1 GP model

Following previous accounts of GP inference [Schulz et al., 2017, Duvenaud et al., 2013a], we assume that the GP agent makes a multiple choice decision using the following two-step process:

1. Given y_{prompt}, estimate the kernel most likely to have generated it: $\hat{K} = argmax_K P_K(y_{prompt})$
2. Evaluate the posterior likelihood of each completion $P_{\hat{K}}(y_i|y_{prompt})$, using the kernel inferred in step 1.

Here, P_K denotes the probability distribution defined by the kernel K. We then assume that the choice probabilities are determined by the conditional likelihood of each candidate. Thus, given prompt y_{prompt} and candidate completions y_i, we assume that the GP agent selects choice i with probability

$$p_i^{GP} \propto P_{\hat{K}}(y_i|y_{prompt})^{\gamma}$$

where $\gamma > 0$ is an inverse temperature parameter. We treat γ as a participant-specific hyperparameter, and fit it to each participant using maximum likelihood. Finally, for simplicity and conceptual clarity, we will assume that the inference in step 1 is perfectly accurate. That is, if y_{prompt} was generated from some kernel K, then the GP agent can exactly recover K after seeing y_{prompt}. Thus the GP agent acts as an Ideal Observer.

3.3.2 MaxEnt model

For this model, rather than relying on specific prior distributions, the choice probabilities are based on the simple iterative extrapolation process implied by Burg's theorem. When presented with a prompt and set of candidate completions, we assume that the MaxEnt agent first fits the parameters w, σ of an autoregressive linear model on y_{prompt}. The agent then uses these parameters to evaluate the plausibility of a given completion. The form of the MaxEnt solution gives a natural way to make such judgments. Let $\{r_{ij}\}_j$ denote the set of residuals (i.e., prediction errors) along the candidate completion y_i with respect to the fitted regression model. That is, $r_{ij} = y_i[j] - \sum_{k=1}^{L} w_k y_i[j-k]$ (where $y_i[j]$ denotes the jth entry of the vector y_i). Burg's theorem implies that *if the completion was generated by the same process as the prompt, then the residuals are independent samples from* $N(0, \sigma)$. Conversely, systematic departure of the distribution of residuals from $N(0, \sigma)$ indicates that the completion is unlikely to have been generated by the same process as generated the prompt. Accordingly, a simple way to assess the fit is to compute the empirical mean $\hat{\mu}_i = \mathbb{E}_j r_{ij}$ and standard deviation $\hat{\sigma}_i = \sqrt{\mathbb{E}_j (r_{ij} - \hat{\mu}_i)^2}$ of the residuals, and compare them to the expected values 0 and σ. The further these values deviate from 0 and σ, respectively, the less likely it is that the completion y_i was generated from the same process as y_{prompt}.

Based on this, we assume that the MaxEnt agent makes its decision using the following three-step process:

1. Compute the regression parameters w, σ on the prompt curve

2. For each candidate completion y_i, compute the regression residuals $\{r_{ij}\}_j$ along y_i, as well as the mean $\hat{\mu}_i$ and standard deviation $\hat{\sigma}_i$ of these residuals.

3. Select the choice i with probability

$$p_i^{MaxEnt} \propto KL(N(\hat{\mu}_i, \hat{\sigma}_i)||N(0, \sigma))^{-\gamma}$$

where KL denotes the Kullback-Liebler divergence [†] between the two Gaussian distributions, and $\gamma > 0$ is an inverse temperature parameter . The exponent is negative because high values of KL correspond to poor fits, and thus to low choice probabilities. This model thus contains two participant-specific hyperparameters: γ and the window length L. We fit these two parameters independently for each participant using maximum likelihood. Accordingly, we report model comparison results via BIC in which the MaxEnt model is considered to have two parameters, and the GP to have one. [‡]

3.4 Experiment 1: Roughness calibration

The previous function learning literature has focused primarily on smooth curves [McDaniel and Busemeyer, 2005a, DeLosh et al., 1997], or ones with piecewise discontinuities

[†]Using KL divergence is not essential, and other metrics could also be used without changing the model's qualitative behavior.

[‡]Strictly speaking, this is an accommodation that disadvantages the MaxEnt. Since one of the parameters of the MaxEnt model is discrete, the parameter space of this model is geometrically a finite set of disjoint one-dimensional intervals. Thus, arguably, the MaxEnt model should be considered as having only one free parameter, since finitely many small intervals can be concatenated into one large interval.

[Wilson et al., 2015b]. Here we test the hypothesis that people also have consistent preference patterns for rough curves, and that the pattern of responses to both smooth and rough curves can be explained by the MaxEnt model. More specifically, it predicts that people will be sensitive to the degree of *local* roughness along a curve, and will tend to prefer completions that are calibrated (i.e., matched) to the prompt in this regard.

We used the Matern kernel to generate multiple choice extrapolation problems, in which the roughness of the prompt curve, as well as the roughness of the individual completions, were independently varied. More specifically, to generate a prompt curve, we first sampled $\nu_0 \in \{.25, 1, 4\}$, and then sampled from the corresponding Matern kernel $y_{prompt} \sim K_{\nu_0}^{matern}$. We then generated three candidate choice curves y_i, defined as samples from the three posterior distributions $y_i \sim P_{K_{\nu_i}^{matern}}(\cdot|y_{prompt})$, where $\nu_1 = .25$, $\nu_2 = 1$ and $\nu_3 = 4$. An example is shown in Figure 3.1. We generated a total of 15 stimuli for each prompt roughness level, for a total of 45 stimuli. Participants were instructed to choose the completion they judged to be the best completion of the prompt. The order of prompt curves was randomized for each participant, and the order in which the three candidate completions were displayed was randomized within each trial.

52 participants were recruited from Prolific (www.prolific.co). After exclusion of participants who failed attention check trials, $N = 43$ participants remained (15 female, mean age=27.0 $\pm$ 10.3). Participants were paid \$1.59 for their participation. On average, the experiment took 4.5 minutes to complete.

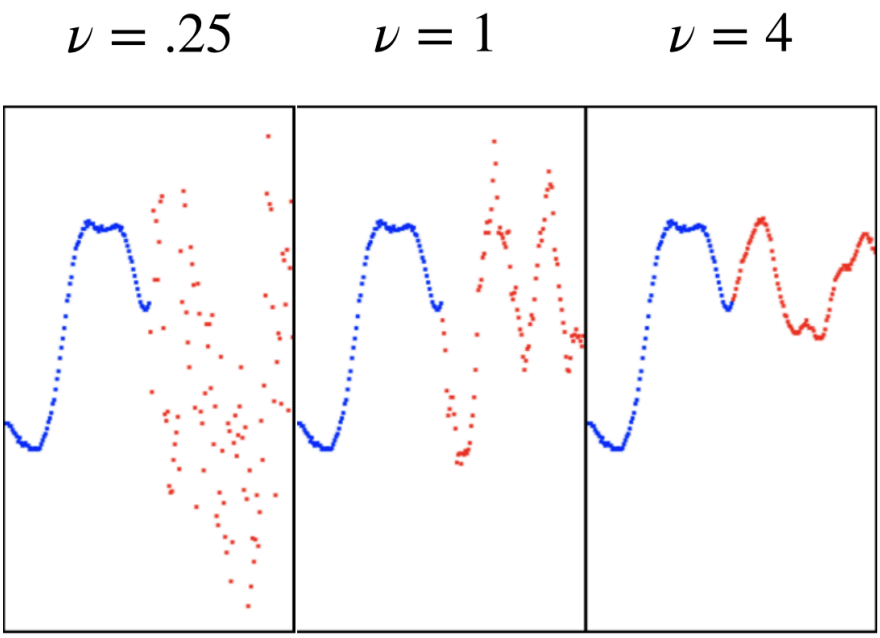

Figure 3.1: Example stimulus in Experiment 1. The prompt curve (blue) was in this case sampled from a Matern kernel with $\nu = 1$. The plot labels indicate the ν corresponding to each respective completion. In this case, the MaxEnt model would prefer the middle completion (assuming a small L), while the GP would prefer the right one.

3.4.1 Behavioral results

On average, participants selected the completion that was matched to the prompt in terms of roughness in 79 percent of trials, suggesting a strong preference for roughness-calibrated completions. This preference was strongest when the prompt curves were maximally smooth (i.e. $\nu_0 = 4$).

We next examined individual differences in response patterns by performing K-means clustering on the confusion matrix of each participant. The best fit (according to a silhouette score [Rousseeuw, 1987]) was obtained for $K = 2$, with the two groups corresponding to qualitatively different response patterns. The first group, that we refer to as *calibrated* participants, chose the completion with the noise level matched to the prompt on nearly all

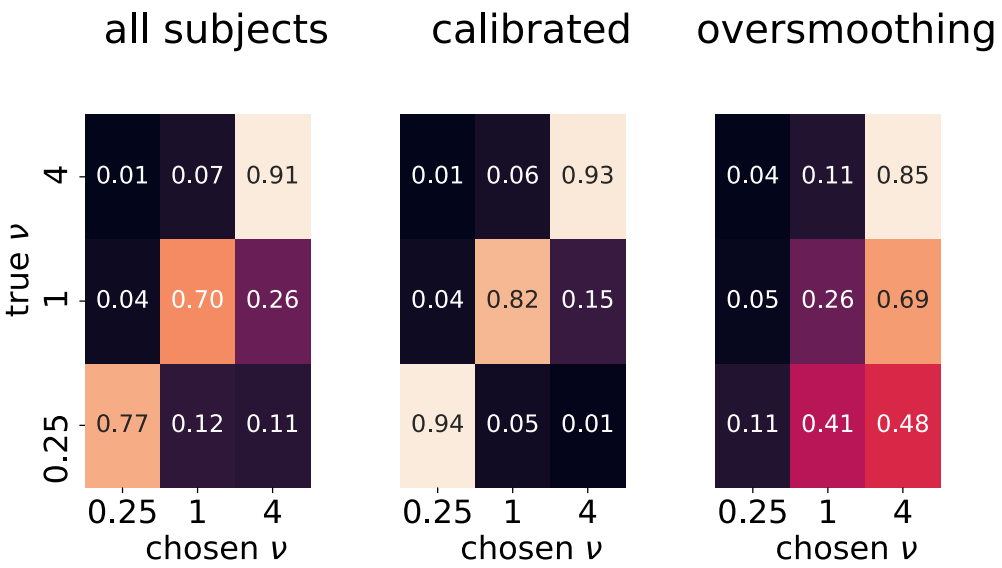

Figure 3.2: Empirical confusion matrices. The left matrix is the average response over all participants, while the other two are averages only over participants classified as calibrated or oversmoothing, respectively (see text for explanation)

trials. This group comprised 34/43 subjects. In contrast, the second group, that we refer to as *oversmoothing* participants, consistently chose completions that were smoother than the prompt (or matched it, in the case of maximally smooth prompts). This group comprised the remaining 9/43 participants. In Figure 3.2 we show the empirical confusion matrices averaged across all participants and for each of the two groups individually.

3.4.2 Model results

Before presenting formal model fits, we first consider the qualitative behavior of the models. In Figure 3.3 we plot the confusion matrices for the MaxEnt model for both large and small L values; the figure also shows the confusion matrix for the GP model, which has no free parameters after fixing the value of γ. The results show that the MaxEnt model can exhibit both calibrated response patterns (for low values of L) as well as oversmoothing response patterns (for higher values of L), thus offering a potential account of individual

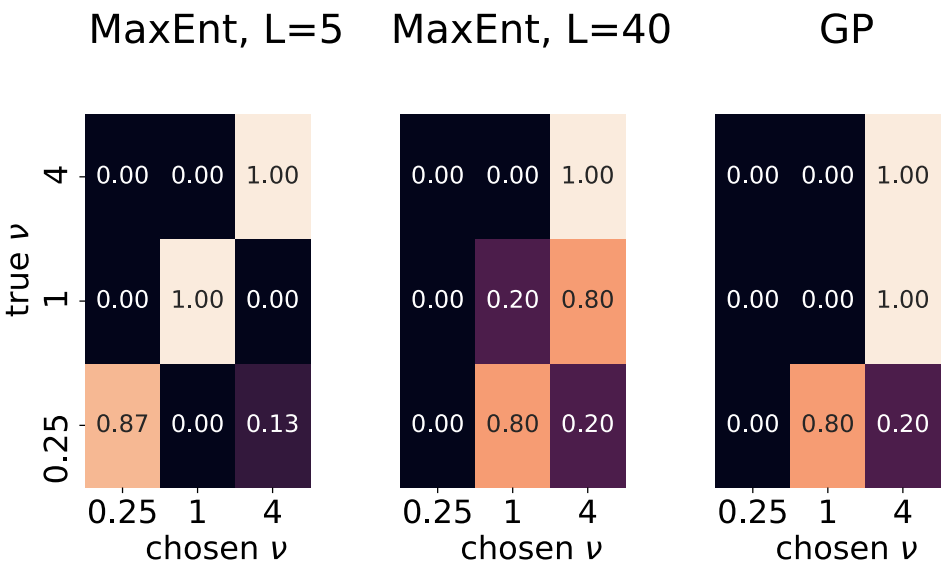

Figure 3.3: Model response patterns. For simplicity, in all models $\gamma = \infty$ (corresponding to a deterministic argmax decision rule). In MaxEnt models, L is window length (see text).

variability. In contrast, the GP model is constrained to consistently oversmooth relative to the prompt (i.e. chooses curves with larger values of ν), and thus can account only for the less common of the two observed behavioral patterns.

Figure 3.4 shows the fits of the MaxEnt model in greater detail. As expected, calibrated participants exhibit smaller L values, whereas oversmoothing participants exhibit larger values. Thus the L parameter can account for this difference in response patterns.

These observations are supported by formal model fits. When fitted to each individual participant,the MaxEnt model attains a log-likelihood (llh) value of -0.502 ± 0.259 (mean $\pm$ std), whereas the GP model attains a fit of -0.729 ± 0.147. A random guessing model attains $\log 1/3 = -1.099$. Additionally, the MaxEnt model provides a better MLE fit to 41 of 43 participants than the GP model, and attains a lower BIC value on 35 of 43 participants.

These results may seem surprising: Why couldn't the GP model "pick out the right"

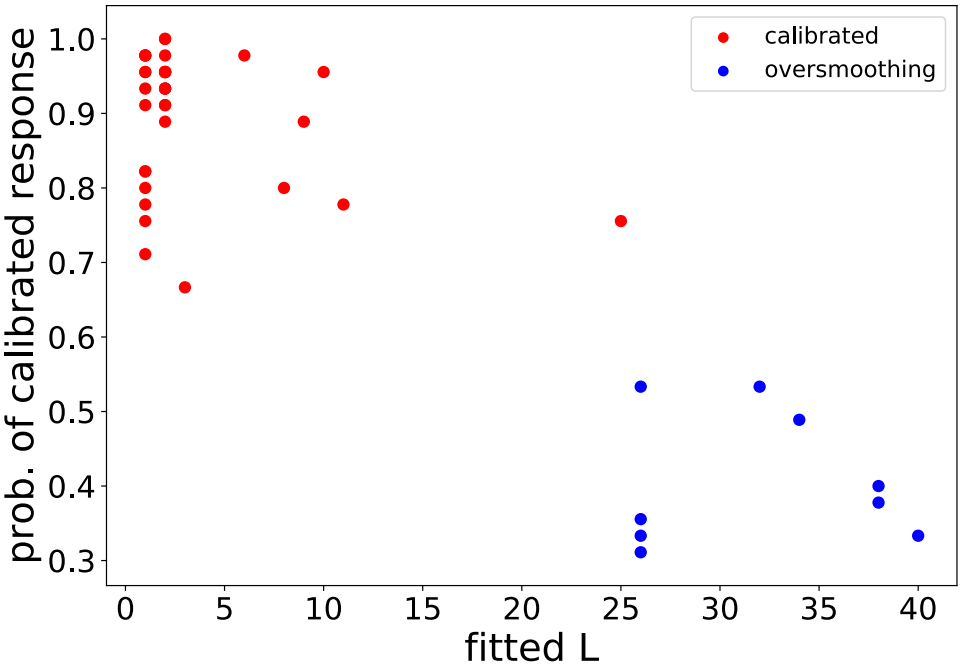

Figure 3.4: Fitted window length parameters, and calibration probabilities (i.e. average diagonal entry of the confusion matrix) for each participant. Large L values correspond to low calibration probabilities, with the majority of participants having high calibration. The colors indicate the cluster assignment of each participant.

completion, given that it has access to the ground truth generative distribution of each prompt curve? This is because, regardless of the value of ν, the posterior *mean* closely approximates a horizontal line, with the likelihood of a given completion being inversely related to the deviations from this mean. Since, in general, smooth candidate curves will tend to deviate less from a line than will rough candidate curves, it is likely that a *given* smooth curve may have a higher posterior probability than a *given* rough curve, despite the fact that the *set of all* rough curves may have far higher posterior probability than the *set of all* smooth curves.

The MaxEnt model also exhibits a tendency to oversmooth, but only for larger values

of L, and for a different reason. When L is very large, the autoregressive model will overfit to the prompt curve, and the estimated σ value will be too small, relative to the intrinsic volatilty along the curve. Since the decisions are made by comparing the residual variance along each candidate to the fitted residual variance σ along the prompt, this model will thus have a bias towards smoother completions when L is large. Psychologically, a large L may correspond to an increased reliance on the "global character" of the function at the expense of local features such as roughness.

3.5 Experiment 2: Typicality preference

Previous work on function learning has focused on curves generated from an RBF kernel (e.g., [Lucas et al., 2015a, Schulz et al., 2017]). The MaxEnt model predicts that, in this more restricted case, completions that look "representativve" of the posterior distribution will be seen as more plausible completions than will the mode of that distribution. This contrasts with the GP model which predicts the opposite, since the mode has (by definition) a higher posterior likelihood value than any other possible completion.

To generate stimuli, we sampled prompt curves from an RBF, and generated two choices for each curve: the first was the posterior mode (with respect to the underlying RBF kernel), and the second was an unbiased sample from the posterior distribution. We refer to these as "modal" and "typical" completions, respectively. 64 participants were recruited from Prolific using the same criteria and payments as Experiment 1. After excluions, N=52 participants remained (24 female, mean age=32.3 $\pm$ 10.6). On average the experiment took 5.2 minutes to complete .

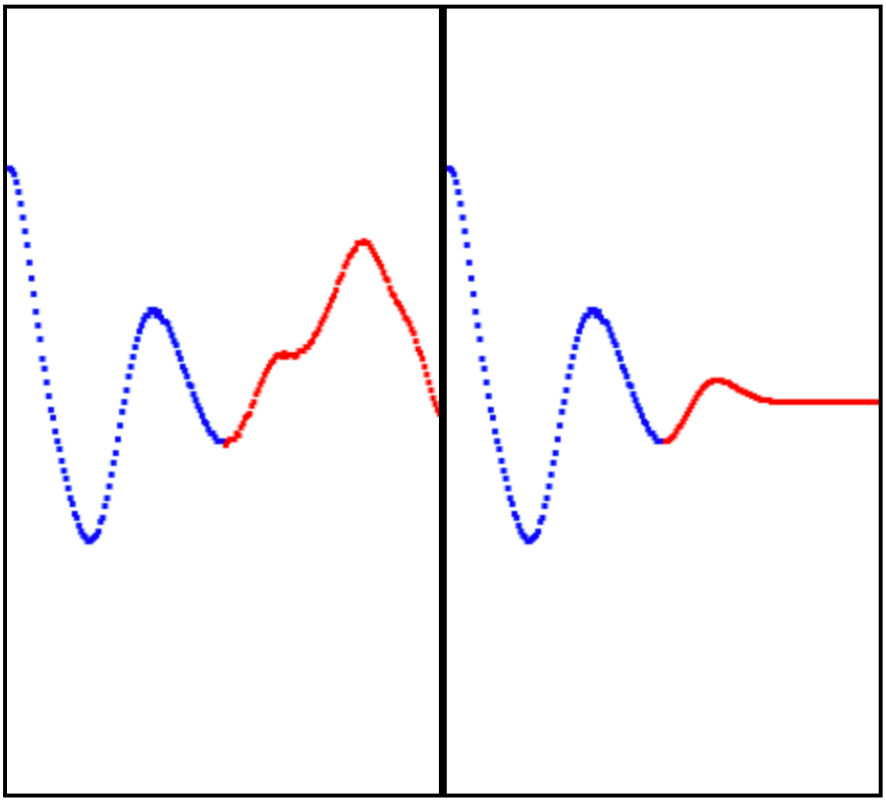

Figure 3.5: Example stimulus from Experiment 2. A typical completion is on the left, and the modal completion is on the right. Note the regression to the mean in the latter.

3.5.1 Behavioral results

Overall, the typical completion was chosen on 74.9 percent of all trials. 43/53 participants chose typical completions on the majority of trials they performed; and for 47/50 stimuli, the typical completion was chosen by the majority of participants. Furthermore, as in Experiment 1, there was considerable individual variability in response patterns, with the proportion of trials for which the typical completion was selected ranging from .38 to 1.0 across participants.

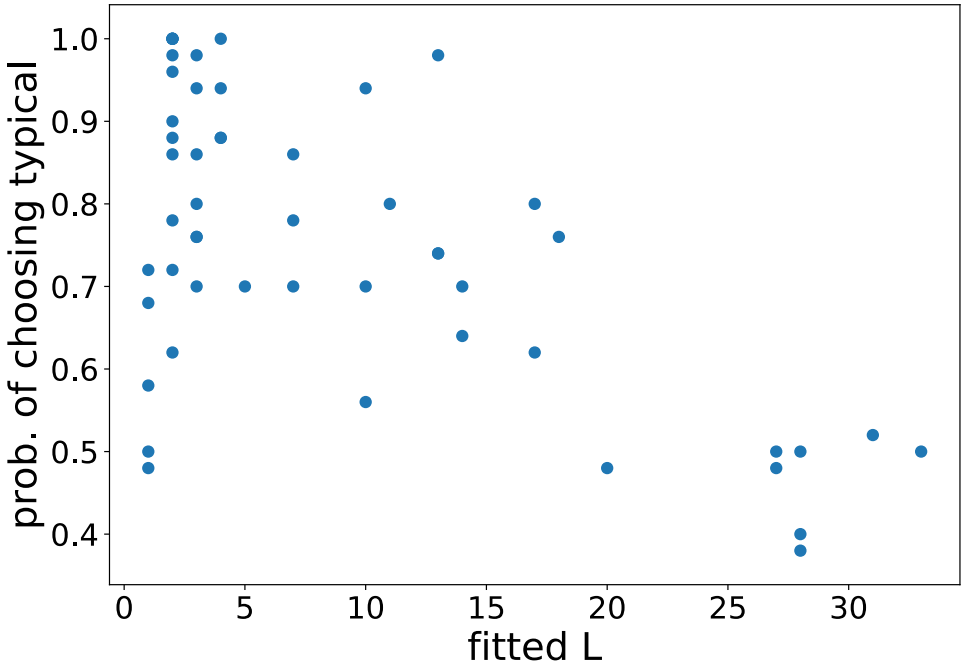

Figure 3.6: Individual response patterns in Experiment 2. Fitted L value of each participant tracks the proportion of trials on which participants chose the typical completion.

The MaxEnt model exhibits a strong bias for selecting the typical completions, similar to what is observed in the empirical data. In contrast, the GP model provides a poor qualitative fit. This is because it cannot assign $>$ 50 percent probability to the typical completion for any stimulus, regardless of the value of the inverse temperature parameter γ. Thus, the GP agent assigns a higher choice probability to the *modal* completion on *all* trials. This contrasts with the empirical observation that the typical completion was preferred for the vast majority (94 percent) of stimuli.

In formal model fits, the MaxEnt model attains a llh of $-0.450 \pm .222$ on an average

participant, compared to $-.692 \pm .006$ for the GP model which is not appreciably different than a random guessing model ($\log 1/2 = -.693$). The MaxEnt model also provides a higher MLE for every participant individually, and exhibits a lower BIC for 41 out of 52 participants. Finally, Figure 3.6 also shows that the fitted L value for each participant closely tracks the participant's preference for typical completions, with smaller L values corresponding to greater preference for typical completions.

3.7 Related work

Classic studies of function learning [McDaniel and Busemeyer, 2005a, DeLosh et al., 1997, Bott and Heit, 2004] have focused on peoples' ability to learn curves generated by simple parametric families, such as linear, quadratic or sinusoidal. [Little and Shiffrin, 2016] considered a more complex set of functions by using polynomials of varying degrees, and found that peoples' preferred completions were biased towards lower dimensional polynomials. In [Lucas et al., 2015a], it was shown that many phenomena could be explained using the Gaussian Process framework. This work relied on a prespecified set of kernel functions, raising the general question of what class of kernels should be considered. [Wilson et al., 2015b] explored the question of discovering an appropriate kernel function using unsupersived methods. [Schulz et al., 2017] argued that the class of kernels is constrained by a principle of compositionality, in which more complex kernels are constructed through combinations of simpler atomic ones.

Although, to our knowledge, no previous studies have specifically tested the MaxEnt model (or a version thereof), several recent studies have generated data relevant to its assumptions, and the design of the experiments used here. For example, [Gelpi et al., 2021]

used an active function learning paradigm, in which only parts of the entire function were shown, and participants had to choose locations at which to query the value of the function. They found that people tended to prefer an even spacing of such points along the x-axis, consistent with our assumption of evenly-spaced observations. Furthermore, [León-Villagrá et al., 2018] incorporated explicit memory constraints into the GP modeling framework. Their strategy for this was very similar to the autoregressive form of the MaxEnt model implemented here, in that they considered only the rightmost previously encountered k points at any given location along the x axis. The autoregressive model also bears a notable similarity to the nearest-neighbor heuristics used in the context of graph structured spaces by [Wu et al., 2021]. Additionally, in a machine learning context, [Segert and Cohen, 2022b] showed how to use the MaxEnt principle for unsupervised learning of "intuitive functions" (that is, smooth functions with simple underlying statistical structure, similar to those considered in our experiments). But, they considered only synthetic data and did not model peoples' extrapolation judgements directly.

3.8 Fit to Free extrapolation data

Up to this point we have discussed the fit of the Maximum Entropy model in the context of forced choice paradigms. While this can allow us to disambiguate the model predictions from those of the traditional Guassian Process model of Schulz et. al., this provides only one way to evaluate the function learning capabilities of the model. However, it has been argued that the key test of whether one has truly learned a function is whether one can generatively extrapolate beyond the observation points [Bott and Heit, 2004]. Moreover, the free extrapolation paradigm is inherently richer and makes it easier to qualitatively assess

the predictions of the model.

3.8.1 Experimental Task

With this motivation, we turn to an experimental dataset collected by Ciccione and Dehaene [Ciccione and Dehaene, 2021] which collects peoples' free extrapolation predictions on a variety of linear and non-linear functions. More specifically, the functions were generated based on one of four families: linear, quadratic, sinusoidal, and piecewise linear. In the experiment, each function is shown in a graphical scatterplot display, with a total of 66 equally-spaced observation points $(x_i, y_i), i = 1, \ldots, 66$, and the participant is asked to predict, using an adjustable slider, the value of the function at an indicated x location. There are two possible locations for this x-location: a near condition, in which the x location is x_{66+19}, and a far condition in which it is x_{66+39}. An example display from a single trial is shown in Figure 3.7. There are a total of 10 ground-truth functions, and the stimulus for a given trial is constructed by adding normal noise to one of these functions. That is,

$$y_{i,j} = f_j(x_i) + \varepsilon_{i,j} \tag{3.1}$$

where $y_{i,j}$ is the displayed y value for point i in trial j, f_j is the ground-truth function chosen for trial j, and $\varepsilon_{i,j}$ is a sample from a centered normal distribution. The x_i points are fixed across trials.

There were a total of 10 participants, each of whom gave a total of 10 extrapolations of each curve in both the near and far conditions, yielding a total of 2000 free extrapolation judgments.

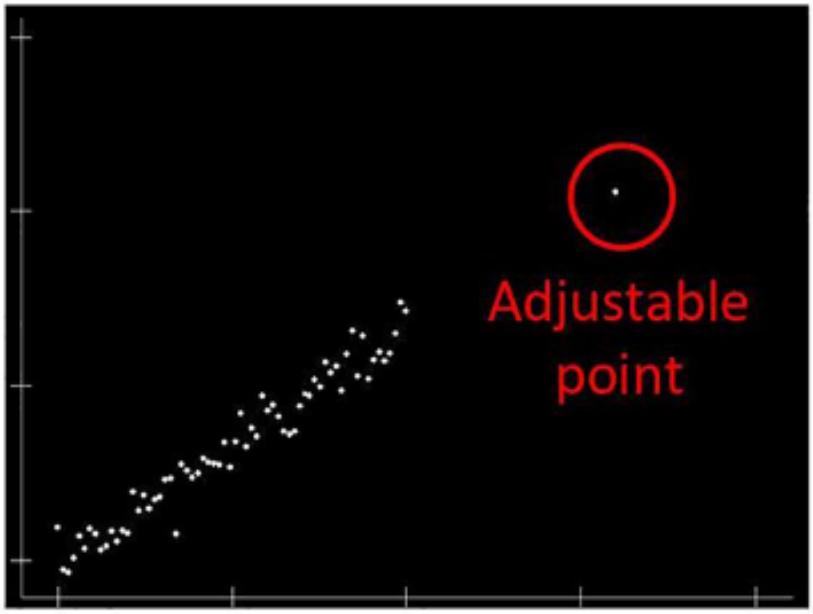

Figure 3.7: An example trial from the free extrapolation experiment performed by Ciccone and Dehaene. The participant can adjust the vertical position of the circled point using a slider. Figure adapted from [Ciccione and Dehaene, 2021].

3.8.2 Model Fitting Procedure

To fit the MaxEnt model to this data, there is only one free parameter that we need to constrain, namely the autoregressive window-length L. We used the particular structure of this experiment to constrain this parameter. Namely, in a given trial, the participants knows that they will need to extrapolate the function *to a known number of steps in the future*. For definiteness, let us say that $(x_i, y_i), 1 \leq i \leq N$ are the displayed values of the function, where x_i are increasing and evenly-spaced, and the task is to predict y_{N+K}. We selected L so as to maximize the accuracy of K-step autoregressive extrapolation within the observed data.

To describe this in more detail, let us introduce some notation. Assume that we have fit a linear autoressive model with window length L, thus obtaining a vector w_L of autoregression parameters, as well as a residual variance σ_L^2. Now, suppose that we have some fixed vector $v \in \mathbb{R}^L$, and we want to "roll out" the predictions starting from v for K steps into the future, obtaining some value v^K. Due to the presence of the noise terms in the autoregressive equation, v^K is actually a random variable. In fact, we have

$$v^K = \sigma_L \varepsilon^K + \sum_{i=1}^{L} w_i v^{N+K-i} \tag{3.2}$$

where ε_{N+K} is a unit-normal random variable, and by convention $v^k = v_{L+k}$ for $k < 0$. We can thus see that v^K is Gaussian-distributed (since by induction v_{K-i} are Gaussian, and a linear combination of jointly-Gaussian random variables is Gaussian). We denote the mean and standard deviation of v^{N+K} by $\mu(v)_L^k$ and σ_L^k, where we keep the L to emphasize this dependence. Note that we do not include the dependence of v on σ_L^k in our notation. This is because it turns out that this variance does not actually depend on v, so we drop it for brevity.

We can now describe how we fit L, given the distance K we need to extrapolate. The idea in words is to first fit the autoregressive parameters w_L and σ_L on the full observed data $(x_i, y_i), 1 \leq i \leq N$, and next use the fitted parameters to compute the K-step rollouts of each window $(x_i, y_i), j \leq i \leq j + L$ and compare the predictions to the actual value y_{L+K}. In symbols, we define a quantity $Q_K(L)$ which measures the extrapolation quality of the

window length L to K steps as

$$Q_K(L) = \prod_{i=L}^{N-K} \varphi(y_{i+K}, y^K_{i-L:i}, \sigma^K_L) \tag{3.3}$$

where we have used the slice notation $y_{i-L:i}$ to denote the vector of all indices within the indicated range, and where $\varphi(\cdot, \mu, \sigma)$ denotes the standard normal probability density function with mean μ and variance σ^2. Finally, for a given K, we choose the L that maximizes Q_K, and take the predictions of the corresponding autoregressive model.

Finally, due to the small number $N = 66$ of observation points in the stimuli, we found that fitting the autoregressive model only on the raw data can sometimes lead to instability when the value of L is comparable to N (since this entails fitting a regression with only a few observations per feature). To remedy this, we employ a mild form of data augmentation, and fit the autoregressive model on data both from the raw stimulus $(x_i, y_i), 1 \leq i \leq N$, as well as the horizontally-reversed version $(x_i, y_{N-i}), 1 \leq i \leq N$. That is, the training data of the autoregressive model consists of both pairs of the form $(y_i, \ldots, y_{i+L-1}), y_{i+L}$ as well as of the form $(y_i, y_{i-1}, \ldots, y_{i-L+1}), y_{i-L}$.

Note that our fitting procedure is adaptive; that is, L is separately estimated for each individual trial (and may differ across different trials with the same underlying function due to the presence of the noise in the stimuli). Furthermore, the resulting model has *no* free parameters.

We will be interested in whether the model can capture the distribution of human extrapolation judgments across the variety of curves used in the experiment. For each of the 10 ground truth curves, and each of the 2 extrapolation distances (near and far), we will

compare the mean and variance in human predictions to those made by the model. As described above, this represents an aggregation of 100 judgments for each curve/distance combination.

Finally, we discuss the aggregation of model predictions across different presentations of the same ground truth function. Recall that for a given input and extrapolation distance, the model gives a *distribution* over the possible values of the function at the query point. We assume that people make their extrapolation judgment by sampling from this distribution. Thus, it is important to realize that there are actually two different sources of variance in the model predictions. The first source arises from the variation in the estimated model parameters w, σ across different presentations of the same ground-truth stimulus (each such presentation having a different pattern of added observation noise). The second arises from the Gaussian noise distribution inherent to the autoregressive model. That is, for a given ground truth function f and extrapolation distance K, let us use $i = 1, \ldots, 100$ to index the different trials in which this combination occurs throughout the experiment. As described above, the model predicts a mean and variance for the query point on each trial, say[§] μ_i and σ_i^2, and we model the actual extrapolation judgment as $\mu_i + \varepsilon_i \sigma_i$ where ε_i is a sample from a unit normal distribution. To aggregate these model judgments over all presentations of the ground truth function, it is necessary to marginalize over both i and ε_i. Thus, the mean prediction over presentations of f, K is given by

$$\mu_{f,K} = \mathbb{E}_{i,\varepsilon_i} \mu_i + \varepsilon_i \sigma_i = \mathbb{E}_i \mu_i \tag{3.4}$$

[§]each of these quantities also depends on f and K, but we drop these from the notation to avoid clutter

while the variance is given by

$$\begin{aligned}
\sigma^2_{f,K} &= Var_{i,\varepsilon}(\mu_i + \varepsilon_i\sigma_i) \\
&= \mathbb{E}_{i,\varepsilon}(\mu_i + \varepsilon_i\sigma_i)^2 - \mu^2_{f,K} \\
&= \mathbb{E}_{i,\varepsilon}\mu_i^2 + 2\mathbb{E}_{i,\varepsilon}\varepsilon_i\sigma_i\mu_i + \mathbb{E}_{i,\varepsilon}\varepsilon_i^2\sigma_i^2 - \mu^2_{f,K} \\
&= \mathbb{E}_i^2\mu_i^2 + \mathbb{E}_i\sigma_i^2 - \mu^2_{f,K} \\
&= Var_i\mu_i + \mathbb{E}_i\sigma_i^2
\end{aligned}$$

We can see that the two terms in the final expression map onto the two variance contributions described above.

3.8.3 Results

Figures 3.8 and 3.9 show the model predicted mean and variance for each function in each distance condition. We can see that in general the model can accurately extrapolate the qualitative shape of each function, and generally attains a close fit to the human data. However, there are some discrepancies, most notably the model typically overestimates the variance in the linear condition. We speculate that this may be a reflection of people having stronger priors for linear relationships compared to other functional forms (which is consistent with prior modeling approaches, e.g. [Lucas et al., 2015a]). Since the autoregessive model has no such prior, it may need more observations in order to reach the same level of confidence in its predictions for a linear function compared to people.

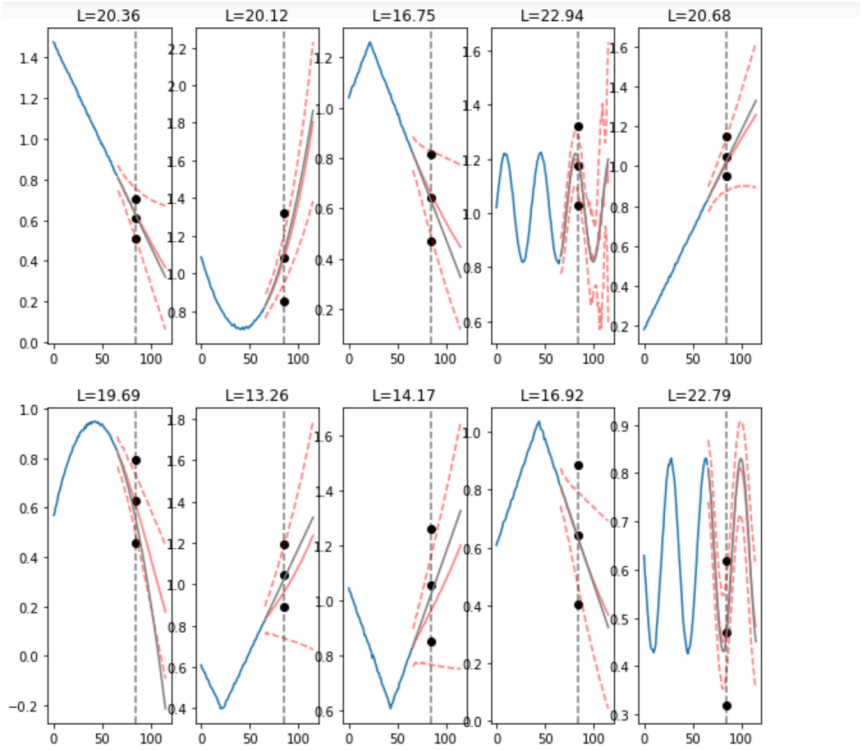

Figure 3.8: Free extrapolation predictions of the MaxEnt model, using stimuli in Experiment 4 in [Ciccione and Dehaene, 2021] in the Near condition. Black dots denote the mean human response, plus-or-minus one standard deviation. Red lines are the model predictions, with the solid being the mean and the dashed plus-or-minus one standard deviation. The gray line is the ground truth extension of the function. We also show the average estimated L value across different presentations of each function.

3.9 Discussion

We have presented a model of function extrapolation that is based on the Principle of Maximum Entropy. The model is based on a general inductive bias that is not specific to function learning, and provides a simple algorithmic process (viz., linear autoregression) by which extrapolation judgments are made. We derived two behavioral predictions from the model: roughness calibration and typicality preference, and showed that both of these effects are robustly present in human judgments. Furthermore, we showed that the L hyperparameter in the MaxEnt model (that determines the autoregressive window length) can

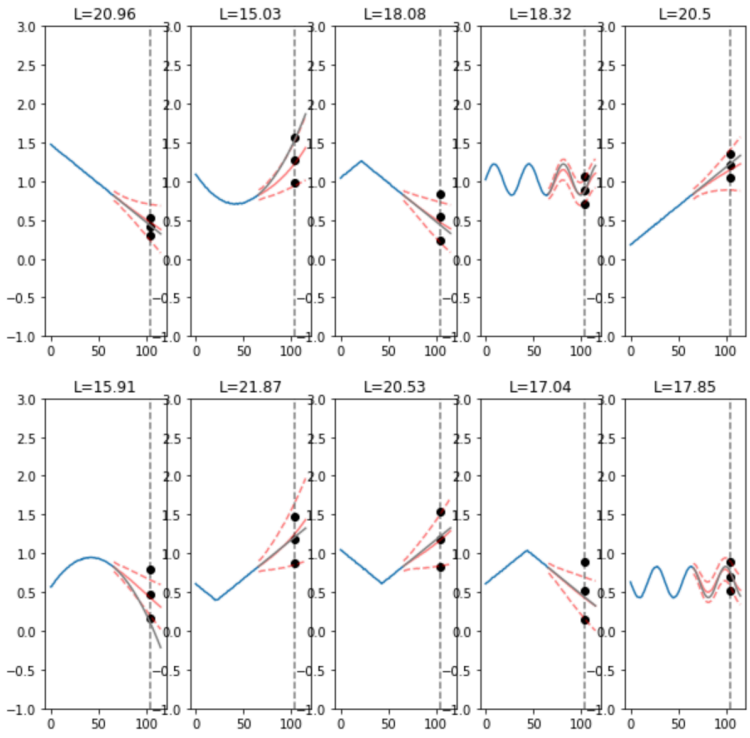

Figure 3.9: Same as Figure 3.8, except in the Far condition.

explain participant-level variation in response patterns, with larger values of L corresponding to a preference for smoother completions. We also saw that the MaxEnt model can be used to successfully model free extrapolation judgments, and can provide a close qualitative fit to empirical data without any parameter tuning. Moreover, in the setting of free extrapolation with a known horizon, the model gives a precise normative prediction for L. In future work, it will be very interesting to directly compare these predicted L values with corresponding L values that are empirically estimated through the forced-choice paradigm as we did in the two experiments.

The MaxEnt model contrasts with existing models of function learning, based on Bayesian inference in the space of curves with a Gaussian Process prior. The latter has proven to be

a useful and conceptually clarifying framework, providing one possible normative benchmark against which to evaluate performance. However, its use has focused on computational-level accounts of the problem. Furthermore, the resulting posterior computation, despite having a closed-form mathematical solution, is computationally intensive, generally requiring the inversion of a large matrix [Rasmussen and Williams, 2006a]. It is unclear how people might perform this computation, or what approximation they might use; though it is possible that identifying such an approximation might also account for the experimental results we report here.

The MaxEnt approach is also grounded in a potentially normative account of function learning, in which the MaxEnt principle can be thought of as a form of regularization that protects against overfitting the data (and thereby maximizing the potential for generalization and/or future flexibility in learning). Thus, in line with recent work on Resource Rationality [Lieder and Griffiths, 2020, Dasgupta et al., 2020], the MaxEnt model may complement the GP perspective by providing a simple and psychologicaly plausible algorithm for performing inference over the space of functions.

It is possible that a GP model could account for the experimental data, by using some kernel besides the veridical one (a "human kernel"). But since the true kernel already oversmooths, such a model could only account for calibrated response patterns if the putative kernel placed most of its probability mass on *rougher* curves-and would thus be very different from previously proposed "human kernel" candidates [Schulz et al., 2017, Wilson et al., 2015b].

At the same time, the MaxEnt model has several limitations, that suggest directions for future work. First, it is primarily a model of extrapolation, and it is not immediately clear

how it can be applied to interpolation tasks. Second, our implementation assumed observation points to be equally spaced, which is not always borne out in real functions. Finally, one desirable property of GP models, which we have not utilized within the context of forced-choice judgments, is their ability to quantify uncertainty in a natural way-it remains to be further explored how to best accomplish this within the MaxEnt framework.

In summary, the MaxEnt perspective on function learning, in addition to its success in fitting empirical data, provides several theoretical advantages: leveraging Burg's theorem, it provides a solution to inference problems using a tractable algorithm that is simple and parsimonious, relying only on a domain-general inductive bias. The algorithm provides the basis for a mechanistic, process-level understanding of human function learning, while the generic nature of the inductive bias suggests that the insights obtained in this domain may apply more broadly to other domains requiring generalization and/or extrapolation.

To be aware of limitations is already to be beyond them.

Georg Friedrich Hegel

4

Beyond Transformers for Function Learning

The material in this chapter is adapted from the previously published work [Segert and Cohen, 2023].

4.0.1 ABSTRACT

The ability to learn and predict simple functions is a key aspect of human intelligence. Recent works have started to explore this ability using transformer architectures, however it remains unclear whether this is sufficient to recapitulate the extrapolation abilities of people in this domain. Here, we propose to address this gap by augmenting the transformer architecture with two simple inductive learning biases, that are directly adapted from recent models of abstract reasoning in cognitive science. The results we report demonstrate that these biases are helpful in the context of large neural network models, as well as shed light on the types of inductive learning biases that may contribute to human abilities in extrapolation.

4.1 Introduction

People often reliably identify patterns or rules in small amounts of data, and extrapolate them to domains out of the range of the training data. Despite its seeming simplicity, this process has proved deceptively hard for most neural networks, which often require orders of magnitude more training data. The extrapolation of scalar functions has also been used as an informative test case for abstract reasoning for both humans [DeLosh et al., 1997] and neural networks [**?**] alike.

Recently, the transformer architecture has shown promising abilities to interpolate simple scalar functions [von Oswald et al., 2022, Garg et al., 2022]. However, it remains unclear whether such models can match human abilities on the more challenging and informative task of *extrapolating* simple, scalar functions. Moreover, to the extent that they fall

short in doing so, an important question is: what kinds of modifications or inductive biases would improve their performance? In particular, can ideas from cognitive science be scaled up to aid performance of models using this architecture?

In this work, we propose two such learning biases that are heavily adapted from previous cognitive models, and that we hypothesize would lead to such an improvement. We term them "Relational Bottleneck" and "Adaptive Attentional Window" . The first has been used in models of abstract rule learning, as a way to enforce a separation between content and relations [Webb et al., 2021, Kerg et al., 2022]. The second has been proposed in the context of a cognitive model of function learning [?], and may be thought of as the assumption that the value of a function at a given timepoint depends only the values at a small number of nearby timepoints, with the exact length scale needing to be learned.

Our contribution is first to show how each of these biases can be implemented in the transformer architecture, and secondly to show that their presence improves the performance of this model on extrapolation of the sorts of scalar functions that are easy for people.

Finally, as a collateral benefit, we show that the method we use to implement the relational bottleneck yields *uncertainty estimates* without any additional computation, which standard function learning transformers (such as in [Garg et al., 2022]) do not provide.

The remainder of the paper is structured as follows: We first provide background and further explanation of the two learning biases. We also describe how we implemented these in the context of a transformer architecture, and how the resulting architecture can also be used to naturally model uncertainty. We then empirically evaluate the resulting model and compare it with a standard transformer. Finally we conclude with an overview of related

work and general discussion.

4.2 Background

4.2.1 Linear Autoregressive Function Learning-Motivating Adaptive Attention Window

Function learning, that is, the study of peoples' abilities to interpolate and extrapolate scalar functions from a small number of observations, is a classic topic in cognitive psychology [DeLosh et al., 1997, McDaniel and Busemeyer, 2005a]. The most popular models of peoples' function learning abilities are based on Gaussian Processes [Lucas et al., 2015a, Schulz et al., 2017, Wilson et al., 2015b]. While this framework is able to explain many specific phenomena, it has the disadvantage of requiring pre-specification of a class of functional or distributional forms, and also makes some predictions that are qualitatively at odds with human behavior, especially in cases where there is not a simple parametric underlying pattern to the data [**?**]. It has subsequently been proposed that these issues can be addressed using linear autoregressive models (ibid.) In this case, the value of the function at a location $i \in \mathbf{N}$ is assumed to be a linear combination of a small number of preceding values: $f(i) = \sum_{j=1}^{L} w_j f(i-j)$, with the parameters w_j fit to a given function, and the "window length" L being a hyperparameter. A crucial feature of this model is the assumption of dependence of $f(i)$ on only a small number of nearby values of f. This property is psychologically appealing, as it naturally maps onto constraints on human memory or processing. It is also mathematically convenient, since it limits the total number of parameters w_j that must be fit. Due to the success of this simple and general model at fitting human extrapolation performance, we hypothesize that an adaptation of the finite window length will be

beneficial for transformers in function learning tasks as well.

4.2.2 Systematic Generalization with Neural Networks-Motivating Relational Bottleneck

While the first bias was motivated by a cognitive model of function learning, use of a relational bottleneck was motivated by models of abstraction and relational reasoning [Webb et al., 2021, Kerg et al., 2022], and a correspondence between such tasks and function learning. To see this, we first consider that a simple function such as a line can be thought of as a kind of simple abstract *rule* (e.g., increment a fixed amount for each segment) or, alternatively, a relation between each point and the next. The aforementioned works have considered how to design neural networks to systematically extract and apply abstract rules in visual reasoning settings, such as detecting whether two images are the same or different. These models have the ability to learn a rule on a given collection of inputs (e.g., stars) and then apply the same rule to a different set of unseen images (e.g., circles). At a high level, such models work by constructing a similarity matrix between encoded representations of objects, and forcing the remainder of processing to be done on this pattern of similarities. In this way, the part of the model responsible for learning and applying the rule do not (by design) have access to "sensory" details of specific objects, and thus are forced to learn and use second-order relational patterns. We refer to this as the Relational Bottleneck assumption. Due to the success of these models, we hypothesize that the inductive bias of focusing on the pattern of relations between the values of a function, rather than the raw values themselves, will also aid transformers in function learning tasks.

To give a bit more detail on the models, the ESBN model [Webb et al., 2021] uses a

mechanism of binding and similarity-based retrieval in an external memory, in order to force a reasoning module to operate only on patterns of similarities. The ESBN mechanism has also been adapted for another simple form of extrapolation, namely counting with integers [Dulberg et al., 2021]. This further suggests that it may be possible to also adapt it to the case of extrapolating smooth function as well (after all, an increasing linear function is very similar to a sequence of consecutive integers).

The CoRelNet model [Kerg et al., 2022] can be regarded an abstraction of ESBN model, that implements the bottleneck in a simpler and more explicit form. This model takes a sequence of objects as input, and runs each through a shared encoder module, to obtain representations $z_i \in \mathbf{R}^d$ for each object $i = 1, \ldots, n$. These are used to construct a representational similarity matrix $S_{ij} =< z_i, z_j >$. Finally a reasoning module (architecturally an MLP) is used to read out the answer from the similarity matrix.

4.3 Transformers and Function Learning

Finally, we give a brief overview of the Transformer architecture [Vaswani et al., 2017] as we employ it. As in previous work on modeling sequences, we will employ a decoder-only transformer architecture [Radford et al., 2019]. At the specification level, the transformer architecture TF takes as input a list of vectors $\{v_i\}_{i=1}^N$ and returns as output a list consisting of a "transformed" version of each input vector: $TF(\{v_i\}_{i=1}^N) = \{\tilde{v}_i\}_{i=1}^n$. Some recent works have used this architecture as a model for function learning [von Oswald et al., 2022, Garg et al., 2022]. In these works, the input is a sequence of ordered-pairs $\{(x_i, y_i)\}_i$ of observations of the values of a function f, together with a query point of the form $(x_q, 0)$, where x_q is a point at which the value of the function is to be estimated. The set of such

points are passed through a transformer, and the transformed vector corresponding to the query point is finally passed through a learned linear decoder in order to obtain an estimate for $f(x_q)$.

Since the details of the architecture are described in many other papers (e.g. [Vaswani et al., 2017]) we will instead highlight two main features that are particularly relevant to our purposes. The first is *permutation equivariance*, meaning that if the input vectors are re-ordered according to some permutation, then the output vectors will also be reordered in the same way. Symbolically, $TF(\{v_{\sigma(i)}\}_{i=1}^{N}) = \{\tilde{v}_{\sigma(i)}\}_{i=1}^{n}$, where σ is any permutation. This property means that the input to the transformer can be regarded as an *unordered set*, rather than as an ordered list. This is important because it means that the input does not have to be a one-dimensional sequence, which property we will exploit subsequently. The second property is the presence of masking. When computing self-attention weights between elements in the input sequence, it is common to impose a value of 0 on the weight between certain pairs. This is often done, for example, when the inputs have a canonical temporal ordering, and one does not want to allow one input to attend to those that occur in the future [Radford et al., 2019]. This process can be regarded as a special case of multiplicative weighting of the attention weights, where each gating factor is either 0 or 1. We exploit this in the next section when discussing the Adaptive Window.

4.4 Methods

As indicated above, we aim to incorporate two inductive biases from the cognitive science literature into a transformer, to determine whether these improve its ability to extrapolate: (1)Relational Bottleneck, and (2) Adaptive Attention Window. We begin by describing the

form of the similarity metric we use for the relational bottleneck (tailored to the context of function learning), followed by our implementation of the two inductive biases in the context of the transformer architecture.

In what follows, we will represent a scalar function as a list of x-y observations: $\{(x_i, y_i)\}_{i=1}^{N}$ where $x_i, y_i \in \mathbf{R}$ Furthermore, we will assume for the sake of simplicity, and following [**?**], that the x points are equally spaced along the horizontal axis: $x_i = i, i = 1, \ldots, N$.

4.4.1 Form of Similarity Measure

In the ESBN and CoRelNet, the form of the similarity is given by either a cosine similarity or dot product similarity between the vectors. However, our case differs in that we are dealing with scalar functions (that is, each value of the function lives in a one-dimensional vector space), and we want to consider the similarity between the values of the function at different time points. The cosine and dot product similarity metrics have undesirable properties for such one-dimensional spaces, which make them unsuitable for our application. More specifically, the cosine similarity is degenerate in this context, which follows directly from the definition $\frac{\langle v,w\rangle}{\|v\|\|w\|}$ that the cosine similarity of any two non-zero one-dimensional vectors will be equal to 1. The dot product similarity $\langle v, w\rangle$ is also not suitable for one-dimensional vectors, due to its dependence on the scale of the values. For example, the dot product similarity between two small numbers is smaller than between a large number and a much larger number, even if the two small numbers are extremely close to each other while the two large numbers are very far from each other. For these reasons, we use negative Euclidean distance as our measure of similarity, so that the (dis)similarity between two

observations is defined as

$$S(x, y) = x - y$$

where values with absolute value near zero correspond to very similar observations, and those with large absolute values correspond to dissimilar ones. We choose a signed rather than unsigned distance because this will make it easier to translate from a similarity value to a raw function value, as we discuss further in the next section.

4.4.2 Incorporating Similarity Matrix Bottleneck

As discussed in the previous section, the critical inductive bias in the ESBN and CoRelNet architectures is the imposition of an architectural constraint to allow the "reasoner" in the model to see only similarity scores between pairs of inputs, rather than raw input values. Accordingly, in the case of predicting a scalar function, we restrict the reasoning component of our network, rather than seeing the raw values $\{y_i\}_{i \leq N}$ of the function, to see only the similarity matrix $\{S(y_i, y_j)\}_{i \leq N, j \leq N}$, with S as defined in the previous section. Having specified the desired input, we now consider the desired target. We argue that, given that we restrict the network to consider only relational information, it would defeat the purpose if the prediction target was the raw value y_{N+1}. What, then, should the network predict? We posit that the natural prediction target, given the desideratum of imposing a Relational Bottleneck, is for the network to predict the *N+1 st row of the similarity matrix*. Fortunately, this can easily be accomplished using a standard transformer architecture. We exploit the fact mentioned in the transformer description above that the input to a transformer can be an arbitrary set. We simply construct a set which consists of all values of the similarity matrix, together with their positions within the matrix, as well as a set of query

points that encode the locations at which we want the model to predict the similarity values. That is, given an input $\{(x_i, y_i)\}_{i=1}^{N}$, we construct the following set:

$$X = \{(x_i, x_j, S(y_i, y_j))\}_{i<j\leq N} \cup \{(i, N+1, 0)\}_{i\leq N+1}$$

Due to (anti)symmetry of the similarity function, it is enough to consider only the lower triangular part of the matrix $i < j$. This also helps to alleviate memory demands of the implementation. This set is first passed through a learned linear projection into $\mathbf{R}^{d_{model}}$, and then fed into a transformer. The outputs of the transformer corresponding to the elements $(i, N+1, 0)$ are finally passed through a learned linear decoder in order to obtain predictions for the next row of the matrix. This model is trained by minimizing the mean-square-error of the output relative to the vector $\{Sim(y_i, y_{N+1})\}_{i=1,\ldots,N+1}$. We refer to this model as the *Relational Transformer*.

4.4.3 Incorporating Adaptive Attention Window

Following the autoregressive linear function learning model from [**?**], as well as the general transformer architecture, we map the window length L to the transformer implementations simply by masking out any attention weights between observations that are separated by a distance $> L$ on the x-axis. However, in order to facilitate gradient-based learning, we do not actually impose such a hard-cutoff, but rather parameterize a multiplicative mask on input-to-input attention weights, using a learnable monotonic function of the distance between the inputs, thus allowing the network to learn the most effective interaction length scale. We refer to this as a *Learned Attention Window*.

We first explain the implementation of the Learned Window for the case of a standard

transformer model, such as in [Garg et al., 2022], that takes as input $\{(x_i, y_i)\}_{i=1}^{N} \cup \{(x_{N+1}, 0)\}$ and is trained to predict y_{N+1}. In this case, we would impose a multiplicative gating on the corresponding attention self-attention weight between (x_i, y_i) and (x_j, y_j), by a factor of $F_\theta(|x_i - x_j|)\mathbf{1}_{x_i > x_j}$, where F is a positive decreasing function such that $F(0) = 1$, with θ being learnable parameters. Here the indicator function $\mathbf{1}_{x_i > x_j}$ enforces the standard causal constraint on mask values [Radford et al., 2019]. Note that a gating of only $\mathbf{1}_{x_i > x_j}$ would correspond to the standard upper-triangular mask. We parameterize F using a decreasing sigmoidal form, $F_{a,b}(x) = \frac{1-\sigma(x/b-a)}{1-\sigma(-a)}, a, b > 0, \sigma(x) = \frac{1}{1+e^{-x}}$ although other choices are possible.

The implementation of the Adaptive Window for the Relational Transformer is similar. In this case, the input to the model is instead a set of tuples of the form $(x_i, x_j, S(y_i, y_j))$. Given two such tuples $(x_i, x_j, S(y_i, y_j))$ and $(x_{i'}, x_{j'}, S(y_{i'}, y_{j'}))$, we impose a multiplicative gating factor on the self-attention weight between them as follows:

$$F_\theta(|x_i - x_{i'}|)F_\theta(|x_j - x_{j'}|)\mathbf{1}_{i>i'}\mathbf{1}_{j>j'}.$$

In this way, the gating factor decays with both horizontal and vertical distance within the similarity matrix. Furthermore, the indicator functions enforce a constraint similar to the standard causality constraint in the one-dimensional case. In that case, each element in a sequence is only allowed to attend to elements to the left of itself, whereas in the relational case, each entry in the similarity matrix is only allowed to attend to elements to the left and above itself.

4.4.4 Function Extrapolation with Relational Transformer

Given a set of values $\{(x_i, y_i)\}_{i=1}^{N}$, the Relational Transformer model predicts the "similarity profile" of y_{N+1}, that is to say, the vector containing the similarity of y_{N+1} with all preceding y_i. However, at test time what we want is y_{N+1} itself. In order to recover this value from the predicted similarity profile, we exploit the fact that the similarity function has a known and simple mathematical form, namely an arithmetic difference.

Let $\hat{z}_{N+1}$ denote the predicted similarity profile. By definition, the i-th component $(\hat{z}_{N+1})_i$ is the model's prediction of $S(y_{N+1}, y_i)$. That is, $(\hat{z}_{N+1})_i$ is the model's estimate of $y_{N+1} - y_i$. Thus by simply adding y_i, we can convert this estimate of $y_{N+1} - y_i$ into an estimate of y_{N+1} itself. Doing this for each $i \leq N$, we obtain an ensemble of estimates $(\hat{y}_{N+1})_i$ for y_{N+1}, defined by

$$(\hat{y}_{N+1})_i = (\hat{z}_{N+1})_i + y_i$$

Note that we have used the *invertibility* of the similarity function in a key way (more precisely, of the function $S(x, \cdot)$ for any x). This is why we used signed distance rather than euclidean distance when defining the similarity function. We return to this point in the discussion section.

Thus the output of the Relational Transformer model can be modified to yield an ensemble of estimates for $\hat{y}_{N+1}$. In turn, given an ensemble, we can naturally give both a point estimate and an uncertainty estimate for y_{N+1}. We define the point estimate as

$$\hat{y}_{N+1} = \mathit{Median}_{i \leq N}(\hat{y}_{N+1})_i$$

.

The uncertainty estimates are treated similarly, with the uncertainty in the estimate being defined as the sample standard deviation of the ensemble $(\hat{y}_{N+1})_i, i \leq N$.

4.4.5 Comparison model

Since our objective is to understand what inductive biases are useful to transformers, we will consider as a control a model following [Garg et al., 2022]. In this model, we first construct a set $\{(x_i, y_i)\}_{i=1}^N \cup \{(x_{N+1}, 0)\}$, and then pass each vector in the set through a shared learned linear embedding to $\mathbf{R}^2 \to \mathbf{R}^{d_{model}}$. The resulting set of vectors is then fed through a transformer, and the output vector corresponding to the token $(x_{N+1}, 0)$ is passed through a learned linear decoder $\mathbf{R}^{d_{model}} \to \mathbf{R}^1$ to obtain the model's prediction of y_{N+1}. This model is trained using mean-square-error of the prediction relative to the true value y_{N+1}. We will sometimes refer to this as the 1d Transformer model, to distinguish it from the Relational Transformer model above.

4.5 Experiment details

Models were trained on a next-timestep prediction objective using squared error loss as described in the previous section.

In our experiments we consider functions of length $N = 20$, which is similar to values used in similar psychological experiments, e.g. [Ciccione and Dehaene, 2021]. The training data for each model consisted of a combination of lines, sinusoids, and Radial Basis Function (RBF) curves, following previous works such as [Schulz et al., 2017, **?**].

The lines were sampled with slopes in $[-.1, .1]$. The sinusoids were sampled with peri-

ods in $[5, 12]$, amplitudes in $[.8, 1.2]$ and phases in $[0, 2\pi]$. The RBF curves were sampled from a Gaussian distribution with mean 0 and covariance $C_{ij} = e^{-.5*(i-j)^2/\sigma^2}$, where $\sigma = 3$. When sampling training curves, each of the above three classes was sampled with probability $1/3$, and then the parameters within each class were sampled as described above in order to generate the actual curve. Additionally, all curves had random uniform noise added to the y values with mean 0 and $\sigma = .1$.

Both the Relational Transformer and one-dimensional transformer models have an embedding dimension of $d_{model} = 256$, with 8 attention heads and 12 layers. We do not use dropout.

All models were trained on a total of 320000 curves using a batch size of 32. We used the Adam optimizer with default parameters and learning rate of 10^{-4}. Each model was trained 3 times from different random initializations. All simulations were done using PyTorch.

4.6 Results

In table 4.1, we consider extrapolation results on the three classes of curves. We sampled a total of 2500 new curves evenly split among the three classes linear,sine, and rbf. We used each model to extrapolate the function to the points $x_{N+1}, \ldots, x_{N+10}$ in an autoregressive fashion, similar to [Radford et al., 2019]. That is, after we have obtained the model's prediction $\hat{y}_{N+1}$ for the value at x_{N+1}, we construct a new input set consisting of the original observations $\{(x_i, y_i)\}_{i=1}^{N}$ together with the observation $(x_{N+1}, \hat{y}_{N+1})$ and a query for the next point $(x_{N+2}, 0)$. We repeat this process until we have obtained the requisite number of extrapolated values. We then computed the mean square error of the model extrapolation with the true value of the function at the corresponding 10 points.

Table 4.1: Extrapolation accuracy, mean square error. Values are mean and standard error, over 3 copies of each network. We show an average over all test curves, as well as broken down by curve type.

	all	lin	rbf	sine
1d transformer	0.545 ± 0.060	0.109 ± 0.005	1.501 ± 0.164	0.213 ± 0.047
1d transformer, learned window	0.415 ± 0.022	0.368 ± 0.045	0.810 ± 0.020	0.056 ± 0.005
relational transformer	0.401 ± 0.017	0.073 ± 0.020	1.109 ± 0.044	0.163 ± 0.042
relational transformer, learned window	0.365 ± 0.016	0.069 ± 0.026	1.063 ± 0.030	0.085 ± 0.014

Table 4.2: Estimated uncertainty results for the Relational Transformer model. Note that the baseline transformer model does not have any way to natively estimate uncertainty, so corresponding values are not shown. Values are mean and standard error, over 3 copies of each network. Optimal values are defined as in main text.

	all	lin	rbf	sine
relational transformer	0.211 ± 0.019	0.068 ± 0.015	0.399 ± 0.028	0.241 ± 0.023
relational transformer, learned window	0.281 ± 0.050	0.092 ± 0.024	0.603 ± 0.135	0.238 ± 0.022
optimal		.1	.802	.1

We can see that the omnibus effect of introducing either the finite window length or the relational transformer is a significant improvement in performance (compare second and third rows of table with first). However, the breakdown according to curve types is quite different. In particular, the 1d transformer with Learned Window attains poor performance on linear curves, but compensates with large improvements for sines and RBF curves, compared to the baseline. This is somewhat at odds with psychological data suggesting that people can extrapolate lines more accurately than oscillations [Ciccione and Dehaene, 2021, Kalish, 2013]. By contrast, the similarity transformer improves on both lines and sines compared to the baseline, while preserving the relative difficulty between them. Finally, the variant with both a learned window and relational bottleneck attains the best performance of all, suggesting that both biases together are helpful for accurately predicting simple functions, moreso than either on its own.

4.7 Uncertainty Estimation Results

As mentioned previously, the relational transformer model has the property of natively estimating uncertainty of its own estimates. To evaluate this capability, we generate extrapolations out to $t = 10$ steps as before, and average the predicted standard deviation at each step. We show in table 4.2 the results for the Relational Transformer, both with and without the learned window.

For the sake of having a comparison, we now consider the question of an optimal value for these estimates. For the cases of lines and sinusoids, we recall that they are generated using an underlying deterministic function corrupted by iid exogenous noise. A perfectly predictive and calibrated model, therefore, would be able to infer the underlying function, and would thus have an uncertain value that is equal to the exogenous noise, which in our case was $\sigma = .1$. The RBF functions are slightly different, because even in the absence of exogenous noise, the functions are sampled from a distribution rather than generated according to a deterministic formula, and thus they have an irreducible amount of unpredictability. However we can still define an optimal uncertainty estimate using the underlying kernel of the RBF process. More precisely, we define the optimal uncertainty estimate σ_t to be the standard deviation of the RBF posterior distribution of $y_{t+N}|y_1, \ldots, y_N$. For direct comparison with the models, we average these standard deviation values over $t = 1, \ldots, 10$ to obtain the value in the table. It is a mathematical fact that the posterior variance does not in fact depend on observed values $y_1, \ldots, y_n$ of the function [Lucas et al., 2015a], and consequently all RBF curves have the same optimal uncertainty value.

In this case, the uncertainty estimates for the similarity transformer with and without

the masking are essentially indistinguishable, for the cases of lines and sines. Both tend to underestimate the uncertainty for lines, and overestimate for sinusoids. We thus see the same order-of-difficulty effect as in the MSE values. Interestingly, the similarity transformer significantly underestimates the variance of RBF curves compared to the version with the masking. Thus we see another benefit to having both biases in the model, rater than just the Similarity Bottleneck.

4.8 Related work

4.8.1 Scalar Function Learning

Scalar function learning has been a classic topic in cognitive psychology [DeLosh et al., 1997, McDaniel and Busemeyer, 2005a, Bott and Heit, 2004], and has recently gained popularity as a test case for large neural network models as well.

Modern modeling approaches of function learning are typically based on Gaussian Processes [Schulz et al., 2017, Wilson et al., 2015b, Lucas et al., 2015a] and autoregressive linear models [**?**]. Suggestively, the Gaussian Process approach relies very heavily and explicitly on the notion of a similarity matrix, namely the covariance kernel of the process. There, however, the usage of the matrix conceptually differs somewhat from its usage in the present work, in that the matrix is there assumed to have a known parametric form, rather than constructed from the input data as we do.

The general simplicity/tractability of the space and grounding in psychological data have also made it appealing as a test bed for neural networks. For example, in [**?**], the authors used scalar function learning tasks analogous to those from the psychological literature to analyze a variety of self-supervised learning models, as well as to build models that more

closely match patterns of human behavior. Works such as [Garg et al., 2022, von Oswald et al., 2022] have begun to systematically evaluate the function learning capabilities of transformers, however they have focused more on iid interpolation, and less on the extrapolation/prediction setting that concerns us.

4.8.2 Relational Reasoning with Neural Networks

As outlined previously, our approach on the Relational Bottleneck is directly inspired by [Webb et al., 2021, Kerg et al., 2022], in which a form of abstract reasoning is attained by forcing a network to attend to strictly relational information, encoded as a pattern of similarities between a sequence of inputs to the network.

The same tasks in the ESBN, as well as some generalization to more complex images have been addressed in the recent GAMR model [Vaishnav and Serre, 2023];however despite the similar objective, the mechanism of the model is quite different from ESBN and CoRelNet, relying instead on a learned visual attention policy.

The property of working with similarity and relational information has also been a key ingredient in many previous models, albeit not with the absolute separation imposed by CoRelNet and ESBN. For example, the transformer architecture itself [Vaswani et al., 2017] is built on a simple form of all-to-all attention determined by patterns of similarity among inputs. Models such as the Differentiable Neural Dictionary [Pritzel et al., 2017] and Neural Turing Machine [Graves et al., 2014] also make use of similarity computations in a key way, namely as attention patterns to lookups in external memory. Another strategy is to impose strong priors on the pattern of attention weights by imposing a graph structure on the data and allowing each node to attend directly only to nodes that are connected by

short paths along the graph [Battaglia et al., 2018, Veličković et al., 2017, Bronstein et al., 2017b] This technique often requires strong assumptions about graph structure, and may not generalize to continuous cases such as scalar functions.

The general motivation behind the similarity matrix itself, namely of separating "sensory" from "abstract" processing, have also been employed in the context of neuroscience. For example [Whittington et al., 2020] achieves a form of flexible generalization in navigation tasks by enforcing a separate "abstract" network whose processing is divorced from the explicit perceptual information coming into the network. This model has been argued to be formally equivalent to a basic transformer with a specific form of positional encoding and key lookup [Whittington et al., 2022].

4.9 Limitations and Future Work

While we have tried to motivate our specific choice of similarity function from mathematical considerations, it is not clear whether this is the best possible choice. In future work, it will be interesting to explore whether a model would be able to learn an appropriate similarity function directly from data, rather than having it pre-specified.

Further, we recall that the current form of the model also requires an invertible similarity function in order to read out raw values. Thus signed distance is workable but unsigned is not. As not all similarity functions of interest have this property, it is an interesting direction for future work to relax this assumption while keeping the same ensembling property of the model.

4.10 Discussion

Inspired by abilities of recent Cognitive Science models (namely ESBN/CoRelNet, and MaxEnt Function Learning), we have built models to predict scalar functions that impose a bottleneck through computation of a relational matrix, and imposition an adaptive attention window. We proposed how to implement both of these biases in the context of a standard transformer model, and found that both individually improved the extrapolation performance of a transformer model on scalar functions, with the greatest gain coming in a model that incorporated both. Furthermore,we showed that our Relational Transformer method can naturally be extended to give uncertainty estimates, differentiating it from transformer models of this task such as [Garg et al., 2022].

Thus, we have shown that ideas from cognitive science can be profitably adapted to large deep learning models to improve performance on the kinds of tasks that people are good at. On the other hand, our work also generalizes and extends aspects of the cognitive models which it adapts. Firstly, while the ESBN and CoRelNet models were trained in a supervised multiple-choice setting, we have here extended the Relational Bottleneck property of these models to the case of a generative, self-supervised prediction objective. Secondly, in [**?**], the window length parameter L is treated as a descriptive hyperparameter, and is not given a principled normative account. In our Learnable Window implementation, by contrast, we effectively promote L to a learnable parameter which can be optimized alongside the rest of the model in an end-to-end fashion. In future work, it would be extremely interesting to consider the L of the learned model with those estimated from data on people.

The greater the contrast, the greater the potential. Great energy only comes from a correspondingly great tension of opposites.

Carl Jung

5

Unsupervised Function Learning

The material in this chapter is adapted from the previously published work Segert and Cohen [2022b].

5.0.1 ABSTRACT

Understanding how agents learn to generalize — and, in particular, to extrapolate — in high-dimensional, naturalistic environments remains a challenge for both machine learning and the study of biological agents. One approach to this has been the use of function learning paradigms, which allow agents' empirical patterns of generalization for smooth scalar functions to be described precisely. However, to date, such work has not succeeded in identifying mechanisms that acquire the kinds of general purpose representations over which function learning can operate to exhibit the patterns of generalization observed in human empirical studies. Here, we present a framework for how a learner may acquire such representations, that then support generalization — and extrapolation in particular — in a few-shot fashion in the domain of scalar function learning. Taking inspiration from a classic theory of visual processing, we construct a self-supervised encoder that implements the basic inductive bias of invariance under topological distortions. We show the resulting representations outperform those from other models for unsupervised time series learning in several downstream function learning tasks, including extrapolation.

5.1 Introduction

A key feature of an intelligent agent is the ability to recognize and extrapolate a variety of abstract patterns that commonly occur in the world. Here, we focus on a tractable but still highly general special case of such patterns, that take the form of one-dimensional smooth functions. From a formal perspective, the space of all such functions is vast [Reed and Simon, 1980], necessitating the use of inductive biases for making useful inferences. Thus,

while this setting does not encompass all possible kinds of structures that can be generalized or extrapolated, it is a sufficiently rich space that insights gained here are likely to shed light on the ability of biological and artificial agents to generalize more broadly. At the same time, the abstract structure of this space is relatively simple and well-understood, thus making it amenable to precise analysis and interpretable experimental manipulations.

A further virtue of the space of functions is the existence of detailed experimental data from humans in this domain, thus facilitating a direct comparison of models with natural agents. Indeed, over the past few decades, the empirical studies of function learning in humans has catalogued the forms of several such commonly applied biases, including associative similarity, rule based categorization [McDaniel and Busemeyer, 2005b], bias towards positive linear forms [Kwantes and Neal, 2006b] and compositional construction from small number of basis elements [Schulz et al., 2017]. Taken together, such results describe a class of "intuitive functions," which are functions that people appear readily able to recognize and use.

While it is generally accepted that efficient generalization implies the existence of some previous expectations about the structure of the space of functions, what is not obvious is how such expectations are acquired; that is, what mechanisms are capable of learning and encoding abstract structure through unsupervised or self-supervised experience, in such a way that features relevant to any particular task may be easily "read out" as required. Here, we propose to address these challenges by adapting and extending the general framework of the field of self-supervised learning [Chen et al., 2020, He et al., 2020]. The framework consists of two components: a "slow" encoder that learns general purpose representations of one-dimensional functions using a standard self-supervised learning algorithm,

and a collection of "fast" heads, which can rapidly adapt to different function learning paradigms, based on a small amount of task-specific annotated data, using a simple form of supervised learning (linear or logistic regression). The heads are trained on top of the representations learned previously by the encoder, allowing the model to make use of its general knowledge to rapidly adapt to the particular task demands.

While efforts have taken this general approach [Chen et al., 2020, He et al., 2020], none to our knowledge have specifically considered the domain of function learning with intuitive functions – that is, ones that people have been empirically observed to use [DeLosh et al., 1997, McDaniel and Busemeyer, 2005b, Schulz et al., 2017]. Our approach is further distinguished in the design of the encoder used for self-supervised learning. For this, we treat a scalar function as a (typically very short) time series. The crucial feature of our encoder is a novel family of augmentations of time series, derived from the theory and phenomenology of topological visual processing [Zeeman, 1965, Chen, 2005]. This theory holds that the visual system is invariant to certain kinds of local topological distortions of stimuli, distortions that we design our augmentations to mimic. We hypothesize that such distortions reflect commonly occurring structure in the world, that may in turn have been discovered by the brain, either through evolution or early development, and used as a basis for generalization. Following this idea, we train on a self-supervised objective that tries to enforce invariance across these augmentations, adapting the framework of Chen et al. [2020].

We demonstrate that our choice of encoder and training procedure learns representations that perform better on a collection of downstream function learning and generalizaton tasks than do comparison models for learning and/or representing time series. This

should be of particular interest to the field of semi-supervised learning, since works in that field have not yet systematically analyzed time series that correspond to intuitive functions. Moreover, we directly compare the generalization patterns of the model with those of humans asked to perform a multiple-choice extrapolation paradigm modeled after an empirical study by Schulz et al. [2017]. We find that the model exhibits a qualitatively similar bias as people in this setting, namely, a greater accuracy on functions that are compositionally structured. This should also be of interest to psychologists, since it suggests that behavioral biases in function learning may arise as consequences of a more general representation-learning procedure.

5.2 Background

5.2.1 Contrastive Learning

Here we provide a brief summary of the elements of contrastive learning that are necessary to define our encoder. This is adapted from Chen et al. [2020] and van den Oord et al. [2018]. The basic assumption is that we are provided with a set of positive pairs $(v_i, v_i'), i = 1, \ldots, N$, which are taken as inputs that we wish to consider similar to each other. All other pairs of inputs are considered as negative pairs, which the objective will attempt to push apart in the latent space. For convenience, we will treat the input as a single flattened dataset of size $2N$, in which the positive pairs are those of the form $(v_i, v_{i+N}), i \leq N$. Let $f_\theta : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ and $g_\varphi : \mathbb{R}^{n_2} \to S^{n_3-1}$ be two parametric families of functions (e.g. neural networks). Here n_1 is the dimensionality of the inputs v_i, while n_2 and n_3 are arbitrary, and S^{n_3-1} denotes the hypersphere consisting of all vectors in $\mathbb{R}^{n_3}$ of unit norm. The objective is

$$\max_{\theta,\varphi} \sum_{i=1}^{2N} < (g_\varphi \circ f_\theta)(v_i), (g_\varphi \circ f_\theta)(v_{i+N}) > -LSE^{\tau}_{j\neq i}(< (g_\varphi \circ f_\theta)(v_i), (g_\varphi \circ f_\theta)(v_j) >) \quad (5.1)$$

Here, $\tau > 0$ is a hyperparameter, and LSE denotes the logsumexp function $LSE^{\tau}_i(z_i) := \tau * \log \sum_i e^{z_i/\tau}$. The brackets $< \cdot, \cdot >$ are the Euclidean dot product. After optimizing this objective, we discard the function g and take the encoder to be the function f.

Informally, the first term of the objective function acts to push positive pairs together in the latent space, since it is maximized when both elements in the pair have equal representations. Conversely, the second term acts to push apart all other pairs of inputs. This is because the logsumexp is a monotonically increasing function of each of its inputs; therefore it will be minimized when all of the pairwise similarities are as small as possible. A more precise analysis of properties of this objective may be found in Wang and Isola [2020].

5.2.2 A Generative model of Intuitive Functions

To define a generative model for reference curves that plausibly resemble the distribution encountered by people, we adapt the generative process proposed in Schulz et al. [2017]. This generative model uses the formalism of Gaussian processes Rasmussen and Williams [2006b]; we provide further general background in the Appendix.

Schulz et al. [2017] define the Compositional Grammar by starting from three basic Gaussian Process kernels:

$$\begin{aligned} K_{linear}(x,y) &= (x-\theta_1)(y-\theta_1) \\ K_{rbf}(x,y) &= \theta_3 e^{-(x-y)^2/\theta_2^2} \\ K_{periodic}(x,y) &= \theta_4 e^{-\sin^2(2\pi|x-y|\theta_5)/\theta_6^2} \end{aligned}$$

where θ_i are hyperparameters. In addition, the authors include in the CG ten kernels which are defined using pointwise sums and products of these above three. We refer to the Appendix for a more detailed description.

A natural point of comparison is the Spectral Mixture (SM) kernel [Wilson and Adams, 2013], which is a flexible non-parametric kernel family defined by the formula

$$K_{mix}(x,y) = \sum_{i=1}^{m} w_i e^{-2\pi^2(x-y)^2\sigma_i} \cos(2\pi(x-y)\mu_i)$$

Schulz et al. [2017] demonstrated that people learn curves generated from the CG more easily than ones generated from the SM . Therefore the family of kernels in the CG are good candidates for generating curves that are both naturalistic and are easily recognized by people.

Lastly, it is important to note that, due to the nature of continuous space, in practice it is necessary to represent functions by their values on a finite set $x_1 < \cdots < x_N$ of ordered sample points. In our case, we take the points to be evenly-spaced, and use the same set of points for every function. Thus any function $\{(x_i, y_i)\}$ may be treated as a time se-

ries and vice versa*. In what follows we will use the terms "curve," "function" and "time series" interchangeably, with the understanding that the points x_i remain fixed across all functions. Also, since the positions of the x_i's are the same for all functions, we omit them from explicit notation, and use y to denote the vector with components $\{y_i\}_i$ that defines a function.

5.3 A Contrastive Encoder for Intuitive Functions

To define the encoder, following Section 5.2.1, we need to specify the architecture and the family of positive pairs. For the encoder architecture, we simply take f to be a feedforward network of several 1D convolutions, and g to be an MLP with a single hidden layer. We set $n_2 = n_3 = 128$ and $\tau = .5$. For the class of augmentations, we take inspiration from the field of topological visual perception [Chen, 2005, Zeeman, 1965], which posits that the visual system maintains an invariance to local topological distortions (or "tolerances") of stimuli in order to facilitate global processing.

In our case, we consider 1-dimensional functions rather than 2-dimensional images, but similar principles apply. We propose a family of transformations that implements localized topological distortions to the function, together with several basic global distortions: (1) random vertical reflection, (2) random jittered upsampling and (3) random rescaling. We denote these respectively by stochastic transformations T_1, T_2, T_3, which we describe in more detail below.

The first transformation, that accommodates vertical reflection, is defined by $T_1(y) = -y$ with 50 percent probability and $T_1(y) = y$ otherwise. The second, that accommodates

*In function learning, the x-axis does not necessarily correspond to time. The same is true of a "time series," despite the name: it is merely an ordered list of numbers.

horizontal bending, is the most elaborate. To evaluate $T_2(y)$, we first select a random interval $[a, b] \supset [x_1, x_T]$. We then randomly select points $x'_1 < \cdots < x'_T$ in $[a, b]$. These points are not required to be uniformly spaced. We generate them by sampling uniformly and independently at random from $[a, b]$ and then sorting the samples, and then define $T_2(y)_i = \sum_j C_i e^{-\frac{(x'_j - x_i)^2}{2\sigma^2}} y_j$ where $1/C_i = \sum_j e^{-\frac{(x'_j - x_i)^2}{2\sigma^2}}$. In other words, the values of $T_2(y)$ are given by a Gaussian Kernel Density Estimator (KDE) at the points x_i. The effect is three-fold: since the points x_i lie in a proper sub-interval of $[a, b]$, this crops a portion of the function and up-samples to the original resolution. Secondly, because the points x'_i are not evenly spaced, some inhomogeneous horizontal stretching or contraction is introduced. Thirdly, the nature of the Gaussian KDE means that the augmented functions are smoothed with respect to the originals. Finally, we apply T_3 that accommodates vertical rescaling. For this, we choose a random interval $[a, b] \subset [0, 1]$ and then apply an affine transformation such that the maximum value of the new function is b and the minimum is a. More explicitly, $T_3(y)_i = (b - a)\frac{y_i - \min_j y_j}{\max_j y_j - \min_j y_j} + a$. Since this is applied last, the resulting functions always take values in the interval $[0, 1]$. Therefore the positive pairs take the form $(T_3 T_2 T_1(y), T_3 T_2 T_1(y))$ where y is a function. We reiterate that T_i are stochastic transformations, so despite the notational appearance, the two functions comprising a given positive pair will not be equal, since they are generated using two separate evaluations of a stochastic transformation. In our experiments the function y itself is generated from the Compositional Grammar over intuitive functions described in Section 5.2.2. An illustration of the augmentations is provided in Figure 1.Furthermore, we provide ablation studies on the effect of each augmentation individually in the Appendix.

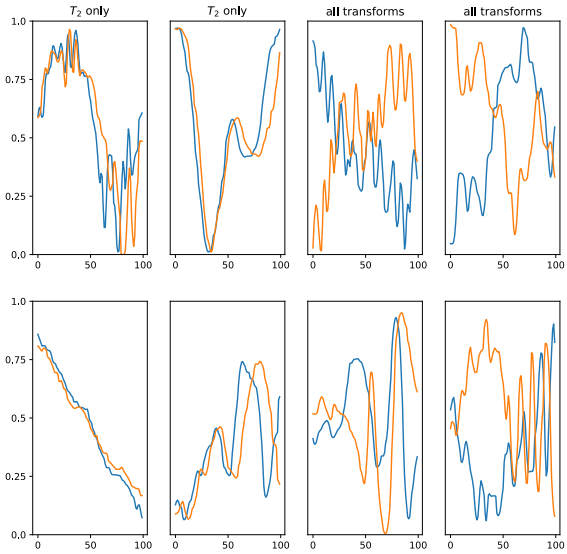

Figure 5.1: Illustrations of the augmentations. Each plot consists of two functions comprising a positive pair. In the four plots on the left, only the horizontal stretch transformation T_2 is applied. In the four plots on the right, all three transformations are applied.

5.4 Data description and Encoder training

To evaluate the ability of the encoder to learn a representation of intuitive functions, we generated and trained it on two types of functions: one generated from the family of 13 kernels defined by the CG (see Section 5.2.2); and the other (used as a control) from a non-compositional SM Kernel, for a total of 14 kernels. As noted above, Schulz et al. [2017] showed that human completions are closer to those generated by the CG than by the SM. We included the SM in our generative distribution to allow a similar comparison. Each function was generated by first sampling one of these 14 kernels, then sampling any hyperparameters of that kernel, and finally sampling from the resulting covariance matrix evaluated on $T = 100$ evenly horizontally spaced points. We sampled from the SM ker-

nel 50 percent of the time, and each of the remaining 13 CG kernels $3.85 (= 50/13)$ percent of the time. Therefore, any differences in the representations between the SM and the CG cannot be ascribed to data availability. We normalized all functions to lie in the interval $[0, 1]$. All encoders were trained using a batch size of 512, with an Adam optimizer with learning rate of .001 and weight decay of 10^{-6}. All encoders were exposed to 500,000 curves during training. We trained three copies of each encoder using random initializations and averaged results over these copies.

As described above, our model consists of a combination of a contrastive loss function, and a convolutional encoder architecture. In addition, we considered eight comparison models, six of which are encoding models trained using different objectives than the contrastive loss, and two of which are architectural ablations trained using the same contrastive loss, but with non-convolutional encoder architectures. Of the first six, four were unsupervised time series models: Triplet Loss (tloss) [Jean-Yves et al., 2019], Temporal Neighborhood coding (tnc) [Tonekaboni et al., 2021], Contrastive Predictive Coding (cpc) [van den Oord et al., 2018], and Conditional Neural Processes (cnp) [Garnelo et al., 2018a]. We also tested a Variational Autoencoder (vae) [Kingma and Welling, 2014] as an example of an unsupervised algorithm that has been successful in other domains, but does not exploit any structure particular to time series. Finally, we included a baseline encoder ("raw") that simply copies the raw input. To control for the latent space capacity, all encoders except for the baseline had representations of equal dimensionality (128).

The first of the two architectural ablations consisted of replacing the convolutional encoder with a multi-layer perceptron. We denote this by "contrastive-mlp." In the second, we replaced the convolutional encoder with the same permutation-invariant encoder as was

used in the CNP model, which we denote by "contrastive-perm-inv". In this encoder, the representation of the function $\{(x_i, y_i)\}_{i=1}^n$ takes the form $\frac{1}{n}\sum_{i=1}^n MLP(x_i, y_i)$ [Garnelo et al., 2018a,b]. This encoder is thus permutation invariant in the sense that the representation of the function does not depend on the ordering of its constituent x-y observations. For both of these ablations, the architectures were constrained to have approximately the same number of parameters as the convolutional encoder. Further implementational details of all comparison models are provided in the Appendix.

5.5 Results on Downstream Classification and Extrapolation tasks

To evaluate the quality of the learned representations, we adapted three function learning paradigms that are either directly translated from or inspired by paradigms from studies of human performance: (1) kernel classification, (2) multiple choice extrapolation, and (3) freeform extrapolation. The first one corresponds directly to the standard paradigm for unsupervised learning evaluation in computer vision [Chen et al., 2020], in which an unsupervised algorithm is trained on a dataset for which ground truth annotations are available, and then a supervised classifier such as a logistic regression is fit on top of the frozen representations. Although to our knowledge this has not been used directly in the analysis of empirical results concerning human function learning, it may be regarded as an abstraction of the experiments from Leon-Villagra and Lucas [2019], which showed that people's completions depend on their judgements about the category to which a function belongs, suggesting that people make use of categories when judging functions. The two extrapolation tasks are drawn directly from Schulz et al. [2017]. A version of the third task also appears inWilson et al. [2015a], however using a different generative process for the probe

curves.

For each task, we define a head that transforms the encoder representations to a task-specific output, and train the head on a small amount of labeled data. In all cases when training the heads, the weights of the encoder are frozen.

In addition, we note that several of the tasks required that we compute the posterior mean with respect to a Gaussian kernel in order to construct the training data, which required that we initially fit the kernel hyperparameters. For example, in the multiple choice task, to construct two candidate completions we took the posterior mean of the prompt curve with respect to both the SM and the CG kernels, which required we first fit the hyperparameters for those kernels. The fitting of GP kernel hyperparameters is known to suffer from under-fitting and instability problems (see Wilson et al. [2015a], including the supplementary material). To address this, we fixed the hyperparameter values of each kernel class, and evaluated the downstream performance of all encoders on curves generated with this fixed set of hyperparameters. We repeated this 10 times using different random choices of hyperparameters each time, and averaged the results. This ensured that the hyperparameter values were correctly specified within each task, and that our results were not influenced by the imperfections of any particular hyperparameter optimization procedure.

We trained three copies of each of the six encoders (contrastive encoder and five comparisons) using different initializations, and all reported results were averaged over the 3 copies and 10 hyperparameter choices, for a total of 30 measurements. The error bars are 95 percent confidence intervals of the standard error of the mean over those measurements.

Table 5.1: Accuracy on the categorization task, as a function of the number of training examples per category. Chance performance is 7.14 percent

	3	10	30	100	300
contrastive	**40.95± 1.83**	**55.27± 1.41**	**64.81± 1.80**	**72.00± 1.52**	**76.23± 1.31**
cnp	15.07± 1.28	19.25± 1.35	22.69± 0.96	26.19± 0.95	28.10± 0.73
cpc	25.06± 1.29	35.45± 1.57	46.38± 1.14	54.94± 1.18	58.48± 1.04
raw	11.86± 1.47	15.54± 2.02	14.27± 1.96	15.73± 1.51	14.25± 1.64
t-loss	30.41± 1.73	41.31± 1.89	52.16± 1.20	59.78± 1.24	63.35± 1.19
tnc	23.15± 1.14	31.22± 1.09	38.55± 1.23	44.85± 1.08	49.16± 1.20
vae	9.27± 1.13	12.77± 1.65	21.51± 1.89	29.17± 1.56	33.74± 1.50
contrastive-perm-inv	16.24± 1.48	22.89± 1.43	27.09± 0.88	29.37± 0.70	31.75± 0.87
contrastive-mlp	33.58± 2.05	46.08± 1.37	53.96± 0.90	57.79± 0.67	60.30± 0.65

5.5.1 Kernel classification

Here, the task was to predict which of the 14 kernels was used to generate a given function. The head was simply a linear layer + softmax, the outputs of which were interpreted as the probabilities of each class. Thus it is equivalent to a 14-way logistic regression on the encoder representations. In all cases, we fit the head using the SGDClassifier class from scikit-learn. Additionally, we separately chose an L2 penalty for each head using cross validation. We report the accuracy of each such classifier on a collection of 2800 held-out curves (200 per class). As shown in Table 5.1, the contrastive encoder is able to attain approximately 55 percent accuracy using only 10 labeled examples per class, which improves to approximately 75 percent when using 300 examples per class, improving upon the second-best model (t-loss) by around 10 percentage points.

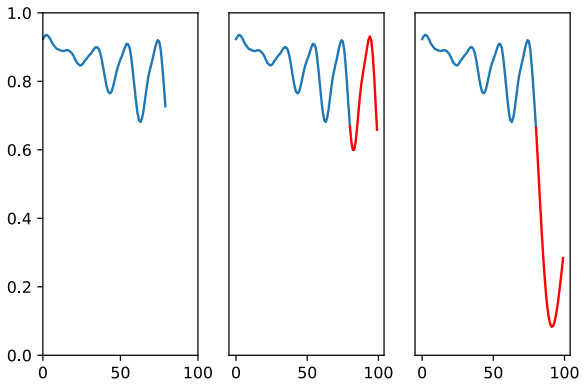

Figure 5.2: An example multiple choice completion problem. The prompt curve is on the left. The compositional completion is in the middle and the mixture completion is on the right. In this case, the correct answer is the compositional completion. The coloring of the candidate curves is for visual aid only.

5.5.2 Multiple Choice Extrapolation

In the multiple choice completion paradigm, the models were presented with a prompt curve $y \in \mathbb{R}^{80}$, as well as several candidate completions curves $y^i \in \mathbb{R}^{100}$ with the properties that $y^i_j = y_j$ for $j <= 80$ and were required to select the correct completion. Following Schulz et. al., we constructed the candidate completion curves by computing the posterior mean with respect either to the SM kernel, or to the best-fitting CG kernel, with the correct answer corresponding to which of these two kernels was used to generate the prompt curve. [†]The training data for the head consisted of 50 percent prompt curves sampled from the SM and 50 percent curves sampled from the CG.

Since this task required comparing the prompt curve to each of the candidate curves, we used a quadratic decision rule for the head. Let h_0 denote the encoder representation of the prompt (upsampled to 100 points prior to being fed into the encoder), and $h_i, i > 0$

[†]More explicitly, to generate the CG completion, we computed the likelihood of the prompt with respect to each of the CG kernels, and then took the posterior mean of the kernel that attained the highest likelihood, mimicing the procedure of Schulz et. al.

Table 5.2: Performance on the Multiple Choice completion task, as a function of the number of training examples per category. Chance performance is 50 percent.

	3	10	30	100	300
contrastive	**67.74± 2.48**	**73.68± 1.69**	**78.09± 1.47**	**79.07± 1.75**	**80.90± 1.93**
cnp	59.30± 2.35	59.80± 2.35	59.72± 2.10	60.04± 2.14	61.56± 2.30
cpc	62.80± 2.60	67.55± 2.26	70.55± 1.93	71.58± 1.97	74.48± 1.81
raw	51.64± 2.83	54.80± 2.13	54.28± 2.20	54.68± 1.85	56.96± 2.22
t-loss	60.18± 1.44	62.74± 1.62	65.12± 1.57	65.31± 1.55	67.76± 1.73
tnc	58.38± 2.46	63.65± 1.89	68.91± 1.64	70.05± 1.68	72.25± 1.99
vae	52.48± 0.74	53.03± 0.83	53.32± 0.85	53.57± 0.84	55.00± 1.29
contrastive-perm-inv	60.38± 2.70	60.63± 2.67	60.68± 2.68	62.30± 2.58	63.32± 2.54
contrastive-mlp	**67.05± 3.00**	67.49± 2.56	70.00± 2.85	71.71± 2.50	72.79± 2.24

denote the representations of the choices. The head linearly projected these vectors into a lower-dimensional space, and chose between the alternatives using a dot product in this space. That is, we fit a model of the form $p_i \propto e^{(wh_i, wh_0)}$ where w is a linear projection from the encoder space to $\mathbb{R}^{32}$ and $p_i, i = 1, 2$ are the choice probabilities. All heads were trained on a cross-entropy loss using the Adam optimizer with a learning rate of .01. We report the accuracy on a collection of 400 held-out curves (200 per class). In this case, we see from Table 5.2 an improvement in accuracy of approximately 5 to 10 percent compared with the second best model. Interestingly, the rank ordering of the models also differs compared to the categorization task: here cpc and tnc both outperform t-loss, and the vae generally matches performance of the raw encoder baseline.

5.5.3 Freeform extrapolation

In this task, for a given function $y \in \mathbb{R}^{100}$, the model was presented with an initial portion $y_{1:80}$ and required to make a prediction $\hat{y} \in \mathbb{R}^{20}$ that extended it for a fixed sized window

(length 20). Performance was measured by $Sim(\hat{y}, y_{80:100})$, for some choice Sim of similarity function. It has been argued that, due to its high-dimensional and underconstrained nature, this task provides a more rigorous test of extrapolation than do discrete categorization tasks and that, in an empirical setting, it may provide finer-grained insights into peoples' inductive biases [DeLosh et al., 1997]. However, it may be unreasonable to expect that an algorithm trained without *any* predictive experience can exhibit a reasonable ability to perform free-form extrapolation. Here, we tested the hypothesis that this capacity can arise from modest amounts of supervised (predictive) training based on categorization judgements among function representations acquired in a self-supervised manner from the contrastive learning mechanism described above. To test this, we implemented a simple form of curriculum learning.

In the first phase of the curriculum, we trained a logistic regressor on the encoder features learned during unsupervised training, to predict the generative kernel of an input function, exactly as in the categorization task from Section 5.5.1. Here we presented the regressors with 300 functions and corresponding category annotations from each kernel. In the second phase of the curriculum, we present a small number of functions y^k with no label annotations. We then fit a simple class-dependent forecasting model of the form:

$$y_i^k \sim w_0^{\hat{c_k}} + w_0 + \sum_{j=1}^{L} (w_j^{\hat{c_k}} + w_j) y_{i-j}^k \tag{5.2}$$

where $\hat{c_k} \in \{1, 2, \ldots, 14\}$ denotes the kernel class of the function y^k predicted by the logistic regressor. Here L is a hyperparameter that controls the autoregressive time lag. We set $L = 20$ in all cases. The parameters $\{w_j^m\}_{0 \leq j \leq L, 1 \leq m \leq 14}$ are weights that are fit using

least-squares. When given a function y to extrapolate at test, we first estimated the class $\hat{c} \in \{1, 2, \ldots, 14\}$ of y using the logistic regressor and encoder features. We then forecast it using the autoregression weights $\{w_j + w_j^{\hat{c}}\}_{0 \leq j \leq L}$.

We compared the results of this procedure with two controls. The first was an Ideal Observer model that was given access to the true underlying Gaussian kernel used to generate each function, information to which the other models were not privy. This model forecast a given function by computing the posterior mean with respect to the kernel on which it was trained. Since this model used Bayesian inference on the exact underlying distribution over functions, it represented the best performance that any model could attain. We refer to this as the "GPIO" (Gaussian process Ideal Observer) model. The second control was a simple autoregression model, that removed the categorization step in order to evaluate its contribution to the forecast quality. It used an unconditional forecasting model of the form $y_i^k \sim w_0 + \sum_{j=1}^{L} w_j y_{i-j}^k$ that ignored any category structure. The autoregression model was trained on exactly the same number of functions as the other forecasting models (with the number of curves used to train the logistic regressor included in this count).

We evaluated the extrapolation performance of each model using the Pearson correlation coefficient and L2 distance (see the Appendix for results of L2 distance) between the actual and predicted curves. We report the average values for 4200 held-out curves (300 per class). The results, shown in Table 5.3, are similar to those for the categorization task. All models substantially outperformed the autoregression baseline, indicating that even imperfect category information is helpful for extrapolation. The contrastive model performed better than any other model except the GPIO model. Several example extrapolations from the contrastive model are shown in Figure 5.3.

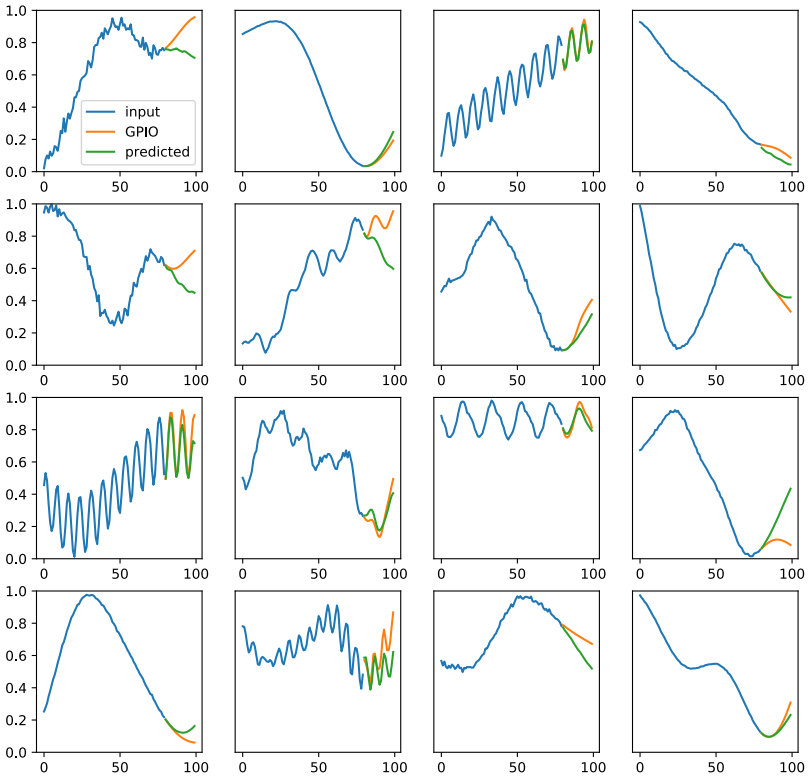

Figure 5.3: Freeform completions generated by the contrastive model, using the maximal amount of training data. The GPIO completions are also shown for comparison.

5.6 Comparison with human data

The multiple choice completion task from Section 5.2 was modeled after Experiment 1 in Schulz et al. [2017]. An intriguing result of that experiment was that people were more likely to select the CG completion than the SM completion. We asked whether any of the models shares this property. To do this, we measured the difference in accuracy when the

Table 5.3: Results on the freeform task, as a function of the number of samples per category used to train the regression (not counting the functions used to train the kernel classifier). Values are the Pearson correlation of the predicted to the true completion. The value for the GPIO model is 83.70± 1.78.

	1	3	10	30	100
autoregression	18.48± 4.46	18.48± 4.48	18.47± 4.52	18.50± 4.47	18.66± 4.47
contrastive	**30.15± 2.36**	**49.05± 2.00**	**60.78± 1.59**	**63.34± 1.39**	**63.91± 1.43**
cnp	24.29± 2.07	33.12± 1.98	40.91± 1.44	43.60± 1.33	45.37± 1.29
cpc	25.13± 2.83	42.32± 2.61	51.04± 2.39	53.06± 1.92	54.79± 1.92
raw	19.89± 4.20	23.13± 4.81	27.02± 5.82	28.18± 5.81	29.14± 5.89
t-loss	**29.45± 3.28**	44.48± 2.74	54.62± 2.00	56.87± 1.93	58.10± 1.95
tnc	**26.72± 3.22**	39.11± 2.84	48.30± 1.81	51.52± 1.60	52.78± 1.74
vae	**27.36± 3.09**	33.51± 2.53	37.18± 2.67	39.27± 2.51	40.89± 2.39
contrastive-perm-inv	22.51± 2.65	32.33± 2.10	39.53± 1.52	42.57± 1.49	44.04± 1.33
contrastive-mlp	**26.72± 2.21**	45.45± 2.41	56.19± 1.55	58.47± 1.65	59.64± 1.35

prompt curve was sampled from the CG compared to when it was sampled from the SM. More precisely, let $\{y_0^i\}_i$ denote a collection of prompt curves, $\{y_{CG}^i\}_i$ the corresponding completions generated by the CG, and $\{y_{SM}^i\}_i$ the completions generated by the SM. Furthermore, define z_i to be a binary variable that indicates whether y_0^i was sampled from the CG or from the SM. In our design, half of the prompt curves were sampled from the CG, meaning that z_i assumes each of the two values with 50 percent probability. For a given model, the choice probabilities $\{(p_{CG}^i, p_{SM}^i)\}_i$ are given as in Section 5.2. Then the accuracy difference is defined by

$$\Delta_{acc} := \mathbb{E}_i(p_{CG}^i | z_i = CG) - \mathbb{E}_i(p_{SM}^i | z_i = SM) \tag{5.3}$$

A short calculation shows that Δ_{acc} is directly related to the model's propensity to favor

the CG completion over the SM completion:

$$\begin{aligned}
\mathbb{E}_i(p^i_{CG}) &= \frac{1}{2}\mathbb{E}_i(p^i_{CG}|z_i = CG) + \frac{1}{2}\mathbb{E}_i(p^i_{CG}|z_i = SM) && (5.4)\\
&= \frac{1}{2}\mathbb{E}(p^i_{CG}|z_i = CG) + \frac{1}{2}(1 - \mathbb{E}_i(p^i_{SM}|z_i = SM)) && (5.5)\\
&= \frac{1}{2} + \frac{1}{2}\Delta_{acc} && (5.6)
\end{aligned}$$

In other words, Δ_{acc} is positive exactly when the CG completion is chosen more often than the SM completion. Note that an unbiased model would have $\mathbb{E}_i(p^i_{CG}) = \mathbb{P}_i(z_i = CG) = \frac{1}{2}$ and thus $\Delta_{acc} = 0$. Note also that, as described in Section 4, all models were trained on an equal proportion of curves from the SM and CG, so any resulting bias cannot be due to differential data availability between the two classes of curves.

We see from Table 5.4 that all models had at least a weak form of the CG bias, in that they attained higher accuracy on the multiple choice task when the prompt was sampled from the CG. The contrastive model had the highest value of this bias, albeit the error bars overlap with the values for t-loss and contrastive-mlp.

Crucially, however, even the baseline "raw" model showed a significant positive bias, thus indicating that the observed biases may be due to statistical properties of the curves themselves, independent of the properties of the learned representations of the models. The contrastive and contrastive-mlp were the only models that attained a Δ_{acc} value significantly higher than that of raw, thus indicating that the observed bias for these models is partially due to the properties of their representations.

However, all of these biases are smaller quantitatively than reported in Schulz et al. [2017]. There it was found that people attain an accuracy of 32 percent when the prompt

Table 5.4: Value of Δ_{acc} on the multiple choice task, as a function of number of labeled training samples per category. The corresponding value for people is approximately 39. We do not bold the highest number because, unlike in the other tables, the values here do not correspond to a normative performance metric.

	3	10	30	100	300
contrastive	19.82± 5.88	21.49± 5.26	20.53± 4.87	18.57± 4.84	19.39± 4.81
cnp	0.73± 5.39	5.25± 3.01	6.61± 4.35	8.17± 3.00	9.61± 3.32
cpc	-3.92± 7.59	1.50± 5.50	4.73± 5.60	6.38± 4.43	10.33± 4.02
raw	5.19± 5.36	2.25± 3.79	3.87± 2.45	5.09± 1.81	7.92± 2.61
t-loss	0.57± 5.04	5.46± 4.37	8.24± 3.95	10.30± 3.44	12.72± 3.75
tnc	-3.23± 6.13	4.70± 4.07	6.42± 2.81	8.18± 3.60	9.55± 3.97
vae	2.01± 1.34	2.62± 1.25	3.12± 1.42	3.30± 1.41	4.19± 1.79
contrastive-perm-inv	1.95± 5.01	4.60± 2.88	5.98± 4.05	8.17± 3.20	9.74± 3.55
contrastive-mlp	16.06± 3.74	16.56± 3.35	16.14± 4.01	16.48± 4.30	18.32± 3.84

curve is from the SM (that is, they choose the CG completion 68 percent of the time), while they attain an accuracy of at least 71 percent when the prompt curve is from the CG. We say "at least", because in that experiment, there were actually three choices presented to the participants in the CG case: the CG completion, the SM completion, and an additional distractor completion. Thus we can estimate that, for people, the accuracy difference is given by

$$\Delta_{acc}^{people} \geq 39$$

5.7 Related Work

5.7.1 Contrastive Learning and Time Series Representation Learning

The idea of learning representations by maximizing an information theoretic criterion can be traced at least back to Linsker's InfoMAX [Linsker, 1988] principle, in which it was shown that certain properties of neurons in visual cortex could be replicated by training the

encoder to maximize mutual information between the input and the encoder representation. This principle was subsequently extended to the problem of unsupervised deconvolution of time series [Bell and Sejnowski, 1995] by extraction of independent components. A network that learns by instead trying to maximize representational similarity between two different parts of the same input, presaging the modern approach to contrastive learning, was introduced by Becker and Hinton [1992]. In a similar spirit, the BCM learning rule [Bienenstock et al., 1982], introduced as a model of synaptic plasticity in the visual cortex, can be shown to be equivalent to projecting the data onto subspaces that are "maximally discriminative" [Intrator and Cooper, 1992], and thus, most likely to be useful for downstream classification tasks. Rather than trying to optimize the mutual information directly, however, most modern implementations of this idea use a form of the InfoNCE loss, introduced by van den Oord et al. [2018]. There it was shown that this objective is a tractable lower bound to the mutual information criterion, which can be difficult to estimate directly. This loss is also strongly reminiscent of the older technique of Contrastive Hebbian learning [Hinton, 1989], insofar as both involve computing average network activations over a set of "positive pairs" of inputs as well as over a set of "negative pairs", and try to maximize the difference between the two averages. The authors incorporated this objective in their Contrastive Predictive Coding model (CPC), in which a recurrent encoder is trained to predict its own future outputs. This basic loss function has been adapted and modified in several ways. In Chen et al. [2020] it is used in tandem with a siamese network architecture, as we described in Section 5.2.1, while Aberdam et al. [2020] extends this setup to a seq-to-seq objective and Li et al. [2020] integrates the contrastive objective with a reconstructive one. A similar contrastive objective is used in He et al. [2020], except with

a memory bank used to sample negative examples, with this approach extended to videos in Pan et al. [2021]. The approach of Hyvarinen and Morioka [2016] is also very similar in spirit, in which the idea is to learn temporal features of time series that differ across different time windows.

Although some of the works above deal with sequential data, they tend to be high-dimensional (videos [Pan et al., 2021], image patches [Aberdam et al., 2020]) or data that is otherwise not directly interpretable by humans (audio [van den Oord et al., 2018], radio frequency signals [Li et al., 2020]), and do not consistently yield lower dimensional, readily interpretable, and easily composable functions of the form studied here. Fewer works have considered whether and how representation learning of such simple functions, such as the 1-dimensional time series of the sort used in function learning experiments and studied here-this omission is notable since, despite their simplicity, such functions occur in a wide range of naturalistic settings [Duvenaud et al., 2013b] . Two particularly notable models of unsupervised time series learning are Triplet loss [Jean-Yves et al., 2019] and Temporal Neighborhood Coding [Tonekaboni et al., 2021]. The first is inspired by word2vec [Mikolov et al., 2013a], and relies on predicting the representation of a "word" (here a short window of the time series) from the representation of its "context" (here a longer window containing the "word"). In TNC, the timeseries is divided into disjoint segments. The encoder is jointly learned alongside a discriminator, in such a way that the discriminator is able to tell the difference between distant and proximal observations. In both TNC and Triplet loss, a recurrent encoder is used. An alternative approach to unsupervised learning of 1D time series learning is through autoencoders. A popular choice here is a seq2seq architecture with a reconstruction loss [Amiriparian et al., 2017, Lyu et al., 2018, Malhotra

et al., 2017]. Ma et al. [2019] augmented this setup with a k-means objective to encourage clustering in the latent space. Compared to our approach, these involve considerably more complexity, through the use of an additional decoding step, as well as more intricate seq2seq architectures.

5.7.2 Function Learning and Gaussian Processes

The dominant framework for modeling of human function learning uses Gaussian processes, a statistical model that specifies a probability distribution over the infinite-dimensional space of functions and allows for tractable inference procedures [Rasmussen and Williams, 2006b]. Lucas et al. [2015b] used Gaussian processes to capture a wide range of empirical function learning phenomena, while Wilson et al. [2015a] and Schulz et al. [2017] proposed specific families of kernels to model human extrapolation judgements. A limitation of the basic GP framework is its dependence on a choice of specific kernels or kernel families. Our approach sought to address this dependence through the use of unsupervised learning. Other approaches have taken a similar tack. The Spectral Mixture Kernel [Wilson and Adams, 2013] and Variational GP [Tran et al., 2016] do so by introducing nonparametric families of kernels that can approximate arbitrary kernels to an arbitrary level of precision. Duvenaud et al. [2013b] implement a similar idea, except by building up a family of kernels using operations of a small number of atomic kernels, and performing a search over the resulting combinatorial space. Sun et al. [2018] perform a similar search except using a continuous relaxation and neural network. In Hinton and Salakhutdinov [2007], an appropriate kernel is found by fitting a Boltzmann machine. Neural Processes [Garnelo et al., 2018b] go further and replace the Gaussian kernel with a more flexible parametric family of

distributions that can be learned using a neural network. This approach naturally extends to modeling of conditional distributions[Garnelo et al., 2018a, Kim et al., 2019, Gondal et al., 2021]. This approach also has the advantage that the encoder can accommodate both variable number of observations, as well as variable x-locations. Such approaches share our broad goal of trying to learn the structure of a space of curves without assuming any particular functional forms ahead of time. However, both differ from ours in that they build in more statistical machinery, by positing an explicit generative probabalistic model of the input distribution of curves.

5.8 Limitations

There are several limitations to our encoder. First, it differs from other models in that it was not designed to scale to very long time series. In particular, we use a feedforward convolutional encoder that processes the entire time series at once, while other techniques use some combination of recurrence and/or local windowing of the time series. Our time series have only 100 points, which is extremely short from the viewpoint of typical time series in learning models. However, in the context of function learning, such short time series have face validity, because during function learning experiments people can only make use of limited information at a time [Villagra et al., 2018]. Thus, while it is not clear how well our encoder architecture would scale to very long time series, it is also not clear how well humans would do so either; and it remains to be determined how useful doing so would be for generalization in natural environments. These remain subjects for future research. Second, the feedforward nature of our encoder also restricts it to processing time series of a fixed length and sampling frequency, as opposed to the recurrent encoders of the other

models which can handle time series of variable lengths, or the CNP-style permutation-invariant encoder, which can handle both variable lengths and variable sampling frequencies. In principle this could be overcome by upsampling or downsampling as necessary (and this was the approach we took in the multiple choice completion task). While this kind of resampling may be benign or helpful in certain circumstances (e.g., as a form of context normalization [Webb et al., 2020]), there are also many applications in which it would instead be preferable to preserve the original resolution and accommodate variable lengths. It is also possible that an appropriate modification of the CNP-style encoder could be used to overcome this difficulty; but due to the relatively poor results of the contrastive-perm-inv model, further work is necessary to fully flesh out this idea.

5.9 Discussion

The contrastive encoder we presented exhibited superior performance to comparison models in tests of generalization involving categorization as well as free form extrapolation. This was the case, despite its greater simplicity than those models. Moreover, we found that the performance was significantly degraded if an MLP was used in place of the convolutional layers in the encoder, and was even further degraded when using a Neural Process-style permutation-invariant architecture. On the other hand, the usage of a convolutional encoder is not sufficient on its own to achieve this level of performance, as shown by the poor results of the VAE, which employed such an encoder. This suggests that the combination of 1d convolutions,together with the contrastive loss and specific family of augmentations, may be key to learning good representations of intuitive functions.

In the multiple choice task, all models found it easier to correctly extrapolate prompt

curves generated from the CG than curves generated from the SM, with this effect being most pronounced in the contrastive model. This is qualitatively similar to the corresponding empirical result from Schulz et al. [2017], regarding peoples' judgements in an analogous task. Thus our analysis suggests that such a bias may simply "fall out" as a consequence of a more general representation-learning procedure. More generally, we regard this as a proof of concept that the properties of representation learning algorithms can serve as an explanatory tool in the study of high-level human cognition such as function learning.

Indeed, several influential accounts of human intelligence posit the existence of elements of "core knowledge," such as an abstract number sense and fundamental notions of Euclidean geometry [Spelke and Kinzler, 2007, Chollet, 2019]. Sometimes referred to as "atoms" of knowledge, these primitives are assumed to be low dimensional forms of representation and/or simple constructs and functions (e.g., continuity of processing, simple forms of causality), on which more complex cognitive abilities responsible for human intelligence are built. It has been proposed that the availability and use of such primitives is a critical factor in distinguishing human generalization capabilities from that of existing artificial systems [Lake et al., 2016], that rely on statistical estimation, and recent empirical evidence has been proposed in support of this claim [Kumar et al., 2020] Major efforts in cognitive science have assumed that such primitives are either genetically pre-specified, or arise sufficiently early and predictably in development that they can be treated as predetermined. Based on this assumption, such efforts have focused research on the kinds of inference and learning mechanisms that, operating on such primitives, can compose them into more complex forms of processing [Lake et al., 2015, Ellis et al., 2020]. Similarly, it has been proposed that such primitives should be considered as inductive biases when de-

signing and comparing candidate computational architectures that seek to emulate human generalization capabilities. In contrast to this approach, some have argued that it is neither necessary nor accurate to assume that such primitives are pre-specific, but rather they arise from and are shaped by general purpose learning mechanisms interacting with and encoding statistical of present in the environment [Rumelhart and McClelland, 1986b]. While examples have been provided of how human-like concept formation and generalization can arise in this way [McClelland and Rogers, 2003], these have generally relied on externally supervised forms of learning that are explicitly trained on tasks that elicit such structure. To date, it has been difficult to design artificial systems that can discover low dimensional, simple forms of structure that can be exploited for generalization, using unsupervised or self-supervised forms of learning. Here, we have proposed one such mechanism, through a combination of contrastive learning and topological augmentations, and have demonstrated its ability learn to basic classes of functions, and simple compositions thereof.

For testing free form extrapolation, we used a curriculum learning strategy that involved first learning categories and then learning category-specific forecasting rules. While this procedure was more complex than for the other heads, there is reason to believe that it in fact resembles the process by which people may learn to make inferences in sparse and underdetermined settings. The most direct evidence of this in the realm of function learning comes from Leon-Villagra and Lucas [2019], which showed that peoples' extrapolations of curves were dependent on whether they judged the curves to lie in a previously encountered category, suggesting that people use category-dependent forecasting rules. More generally, our approach may be viewed as implementing a form of a Hierarchical Bayesian model, which have been show to capture the structure of peoples' intuitive theories about

abstract structures in the world [Gershman and Niv, 2010, Tenenbaum et al., 2011, Kemp and Tenenbaum, 2008].

Our approach also fits with the idea that learning in natural agents involves adjudicating a tension between maintaining as much flexibility as possible (by optimizing a Maximum Entropy objective) while at the same time maximizing efficiency of computation (e.g., by optimizing a Minimum Energy objective). From this perspective, the contrastive encoder can be viewed as maximizing entropy, as implemented by the InfoNCE objective that we used, as it may be shown [Wang and Isola, 2020] that the second term in that objective is an estimator of the entropy of the distribution of codes in the latent space. Complementing this, our curriculum learning can be viewed as minimizing the energy of representations generated by a given category of function when presented with an instance of that function. This may strike a balance between flexibility (of generalization) and efficiency (of inference) that begins to approximate the balance observed in natural agents, and humans in particular [Frankland et al., 2021b].

5.10 Code Availability

We provide code and pretrained models at: https://github.com/SimonSegert/functionlearning-contrastive-tmlr.

5.11 Encoder architecture

Let Linear(m) denote a linear layer with an output size of m. Further, let 1dConv(c,k,s) denote a 1-dimensional convolution layer with a filter size of k, a stride of s and c output channels, and let 1dMaxPool(k) denote a 1-dimensional max pooling with kernel size of k.

Then our encoder is given by the following sequence of layers:

```
1dConv(64,5,2)
1dMaxPool(2)
LeakyReLU()
BatchNorm()
1dConv(64,5,1)
1dMaxPool(2)
LeakyReLU()
BatchNorm()
1dConv(64,3,1)
LeakyReLU()
Linear(128)
```

5.12 Comparison models

For the "contrastive-mlp" model, we replaced the convolutional encoder as described in Appendix B with the following architecture:

```
Linear(256)
LeakyReLU()
BatchNorm()
Linear(150)
LeakyReLU()
BatchNorm()
```

Linear(128)

LeakyReLU()

BatchNorm()

Linear(128)

For the "contrastive-perm-inv" model, the encoder takes the form $Enc(\{(x_i, y_i)\}_{i=1}^n) = \frac{1}{n}\sum_i \varphi(x_i, y_i)$, where φ is an MLP given by the following architecture:

Linear(512)

LeakyReLU()

BatchNorm()

Linear(150)

LeakyReLU()

BatchNorm()

Linear(128)

The total number of parameters in each of these encoders match the total number of parameters in the convolutional encoder described in Appendix B, to within 2 percent. Each of these models was trained in an identical fashion to the "contrastive" model.

For t-loss [Jean-Yves et al., 2019] and tnc [Tonekaboni et al., 2021] we used the implementations made available on the authors' Github pages. Since the authors of cpc [van den Oord et al., 2018] do not provide an official public implementation, we used the cpc implementation provided in the tnc repo. The t-loss repository is licensed under Apache License 2.0 and the tnc repository is not licensed. For the vae, we wrote our own implementation using the same encoder architecture described in the previous section. The decoder con-

sisted of the same pieces in the opposite order, except with the convolutional layers replaced with ConvolutionTranspose layers, and the MaxPooling layers replaced with MaxUnpooling layers. Also, we do not use batch normalization in the decoder.

Both tnc and cpc require choice of a window size, which as the authors of tnc note [Tonekaboni et al., 2021], must in general be chosen using domain knowledge of the time series in question. In our case, the time series are very smooth, meaning that small windows will generally fail to contain distinguishing features. For example, if the time series contains a slow linear trend (which many curves from the CG do), this may be hard to detect in a small window. Thus for both of these models we chose the window size to be the largest divisor of 100 (the number of observations in each time series) that could be accommodated by the implementation.

For t-loss and cpc, extracting a global representation of a time series is straightforward, and we did so as described in the respective papers. For tnc, a key aspect of the model is that it learns representations of windows of the time series, rather than a representation of the full time series itself, which allows it to deal with non-stationarity. To convert this to a global representation, we divided the time series into disjoint windows and concatenated the window representations. This was the representation used in all of our downstream analyses. We also tried a variant in which the window representations were averaged rather than concatenated, but found that the results were slightly worse.

For CNP [Garnelo et al., 2018a], we wrote our own implementation, closely following the description at `https://github.com/deepmind/neural-processes/blob/master/conditional_neural_process.ipynb`. The model was trained to maximize the likelihood of 10 randomly-selected observations from each input function, conditioned on the re-

maining 90 observations from that function. The encoder architecture was identical to the that of the "contrastive-perm-inv" model.

As described in the main text, we modified all comparison models so that the resulting representations had equal dimensionality of 128 (for tnc this refers to the global representation formed by concatenating window representations). Other than this modification, and the choice of window size described above, we used the hyperparameter settings provided in the respective implementations.

5.13 Further Description of Compositional Grammar and Spectral Mixture Kernels

Here we review the set of kernels used by Schulz et al. [2017] as well as the spectral mixture kernel [Wilson and Adams, 2013]. We also specify our choice of kernel hyperparameter distributions, which were used to sample curves as described in the main text.

We begin with a brief refresher on Gaussian process formalism, and refer to Rasmussen and Williams [2006b] for further details. Recall that a *kernel function* is a function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, which may be intuitively thought of as an infinite-dimensional covariance matrix. More precisely, a Gaussian Process with kernel K is a distribution over the space of all functions $f : \mathbb{R} \to \mathbb{R}$ with the property that for any finite set of points $\{x_i\}_{i=1}^n \subset \mathbb{R}$, the vector $(f(x_1), \ldots, f(x_n))$ has a multivariate normal distribution with mean 0 and covariance matrix $C_{ij} = K(x_i, x_j)$. The kernel function is required to be such that this matrix is always symmetric and non-negative definite.

In our case, the points x_i are always fixed to be 100 evenly spaced points between 0 and 10. Thus the Gaussian process formalism is equivalent to a multivariate normal distribu-

tion, but we will retian the language of Gaussian processes for consistency with Schulz et. al.

A Gaussian process kernel can be used to predict future values of a curve. Suppose that we are given some partial set of observations $y_{obs} := \{y_i\}_{i=1}^{m}$ where $m < n$, and we want to estimate the values $y_{ext} := \{y_i\}_{i=m+1}^{n}$ on the remaining points. It may be shown [Rasmussen and Williams, 2006b] that the posterior mean estimate is given by

$$\mathbb{E}(y_{ext}|y_{obs}, K) = K(x_{m+1:n}, x_{1:m})K(x_{1:m}, x_{1:m})^{-1}y_{obs} \tag{5.7}$$

We used this to formula to generate completions in the Multiple Choice Completion task and the Freeform Completion task.

As described in the main text, Schulz et al. [2017] define the Compositional Grammar by starting from three basic kernels:

$$\begin{aligned} K_{linear}(x_i, x_j) &= (x_i - \theta_1)(x_j - \theta_1) \\ K_{rbf}(x_i, x_j) &= \theta_3 e^{-(x_i - x_j)^2/\theta_2^2} \\ K_{periodic}(x_i, x_j) &= \theta_4 e^{-\sin^2(2\pi|x_i - x_j|\theta_5)/\theta_6^2} \end{aligned}$$

where θ_i are hyperparameters. In addition to these three, the authors take advantage of the algebraic fact that the pointwise sum and product of two kernels are themselves valid kernel functions. The remaining kernels in the CG are given by

$$
\begin{aligned}
K_{linear} &+ K_{periodic} \\
K_{linear} &+ K_{rbf} \\
K_{rbf} &+ K_{periodic} \\
K_{linear} &* K_{periodic} \\
K_{linear} &* K_{rbf} \\
K_{rbf} &* K_{periodic} \\
K_{linear} &+ K_{rbf} + K_{periodic} \\
K_{linear} &+ K_{periodic} * K_{rbf} \\
K_{periodic} &+ K_{linear} * K_{rbf} \\
K_{linear} &* K_{rbf} * K_{periodic}
\end{aligned}
$$

Several examples of curves generated from the CG are shown in Figure 5.4.

In our experiments, we sampled the hyperparameters of each of these kernel families at random. To specify these hyperparameter distributions, we use the convention that $N(a, b)$ denotes a normal distribution with mean a and standard deviation b. Also $U[a, b]$ denotes a uniform distribution between a and b, and $U(\{z_1, \ldots, z_k\})$ denotes a categorical uniform distribution on the finite set $\{z_1, \ldots, z_k\}$. The distributions over kernel hyperparameters

are:

$$
\begin{aligned}
\theta_1 &\sim N(0, 2) \\
\theta_2 &\sim U[1, 5] \\
\theta_3 &\sim U[1, 3] \\
\theta_4 &\sim U[1, 3] \\
\theta_5 &\sim U[0, .5] \\
\theta_6 &\sim U[1, 5] \\
\mu_i &\sim U[0, .01] \\
\sigma_i &\sim U[0, .02] \\
w_i &\sim U[0, 1] \\
m &\sim U(\{2, 3, 4, 5, 6\})
\end{aligned}
$$

where μ_i, σ_i and m are the hyperparmaters in the Spectral Mixture kernel.

5.14 Further details on augmentations

When constructing the augmentation T_2, we choose the random interval $[a, b] \supset [x_1, x_T]$ such that $a > x_1 - .4 * (x_T - x_1)$ and $b < x_T + .4(x_T - x_1)$. The bandwidth parameter in the Gaussian KDE is set to $\sigma = .1$. When constructing the augmentation T_3, we define the random interval $[a, b] \subset [0, 1]$ such that $b - a > .8$.

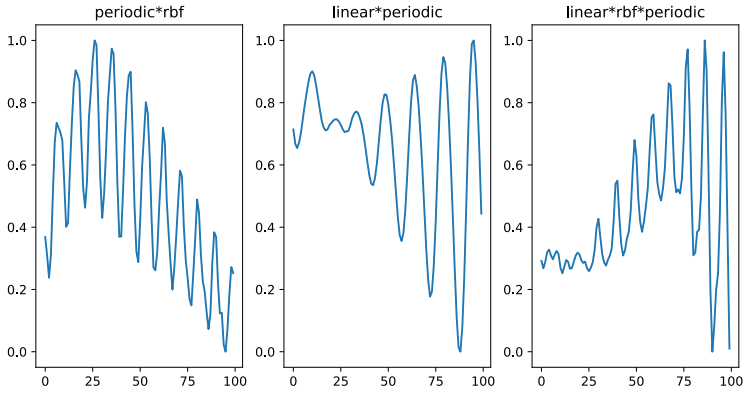

Figure 5.4: Several examples of curves sampled from the CG.

5.15 Further results on Freeform Task

In Table 5.6, we report values on the freeform task using $L2$ distance inplace of Pearson correlation. The results are qualitatively similar, but there is overall less between-model variance in the values.

5.16 Effect of augmentations

We consider the marginal effect of each of the three augmentations on the quality of the learned representations. For this purpose, we compared our contrastive model to 6 ablated models. Each such model was trained identically to the contrastive model, except that one or more of the augmentations was omitted during training (we did not include the degenerate case in which all three augmentations were omitted). The number 6 thus arises as the number of subsets of $\{1, 2, 3\}$ which contain at least 1 and at most 2 elements. As in the main text, we trained three copies of each such ablated model, hen froze the weights and evaluated the downstream performance of each copy on 10 random GP hy-

Table 5.5: Similar to Table 3 in the main text, except using L2 distance instead of Pearson correlation. The value for the GPIO is .0561 ± .0082.

	1	3	10	30	100
ar	**0.1449± 0.0165**	0.1450± 0.0165	0.1451± 0.0165	0.1451± 0.0164	0.1453± 0.0164
contrastive	0.2058± 0.0101	**0.1326± 0.0085**	**0.0989± 0.0062**	**0.0945± 0.0058**	**0.0926± 0.0057**
cnp	0.2120± 0.0145	0.1584± 0.0056	0.1400± 0.0046	0.1346± 0.0043	0.1319± 0.0042
cpc	0.2223± 0.0164	0.1444± 0.0073	0.1143± 0.0064	0.1108± 0.0065	0.1086± 0.0066
raw	0.1758± 0.0201	0.1562± 0.0173	0.1423± 0.0160	0.1384± 0.0151	0.1367± 0.0154
t-loss	0.2044± 0.0120	**0.1374± 0.0078**	0.1085± 0.0056	0.1037± 0.0058	0.1017± 0.0058
tnc	0.2019± 0.0131	0.1453± 0.0064	0.1199± 0.0066	0.1147± 0.0066	0.1131± 0.0066
vae	0.1921± 0.0128	0.1504± 0.0081	0.1330± 0.0072	0.1287± 0.0074	0.1273± 0.0076
contrastive-perm-inv	0.2115± 0.0120	0.1681± 0.0112	0.1392± 0.0047	0.1338± 0.0043	0.1313± 0.0042
contrastive-mlp	0.2257± 0.0133	**0.1362± 0.0058**	0.1083± 0.0040	0.1033± 0.0041	0.1013± 0.0038

perparameters, at 5 different labeled training set sizes. In this way, we obtained a total of $(6 + 1) * 3 * 10 * 5 = 1050$ observations of downstream accuracy as a function of the presence of each augmentation and training set size (the +1 corresponds to the non-ablated contrastive model containing all augmentations).

For the categorization and the multiple choice completion tasks, we fit the following ANOVA:

$$acc \sim \beta_0 + \sum_{i=1}^{3} \beta_1 \mathbf{1}_i + C(n_{tr}) \tag{5.8}$$

where *acc* is the percentage of correct answers given by the model instance, n_{tr} is the number of labeled training samples (which, as indicated by the C, is treated as a categorical variable), and $\mathbf{1}_i$ is an indicator variable which is 1/0 depending on whether the augmentation T_i was included in the training of the encoder (where T_i are as in Section 3). Note that we exclude the freeform completion task from this analysis, because the generated completions depend both on properties of the learned encoder representations, as well as of

Table 5.6: Effect of augmentations on model performance. Values are linear regression coefficients and 95 percent confidence intervals. Thus for example, the presence of T_1 (random reflection) increases the downstream categorization accuracy by approximately 3 percentage points, all else equal.

task	β_1	β_2	β_3	R^2
categorization	.031, (.025,.036)	.052, (.047,.058)	.002, (-.003,.008)	.911
multiple choice completion	.028, (.020,.037)	.099, (.090,.108)	.012, (.003,.021)	.507

an autoregressive linear model. Performance on the other two tasks, by contrast, depends only on the properties of the learned representations. So focusing on these two allows us to isolate the effects of the augmentations on the learned representations.

We fit the ANOVA using ordinary (non-regularized) least squares, with the python statsmodels package.

The resulting coefficients are shown in Table 5.6. In general, we see that all three augmentations had positive effects on downstream performance. Moreover, the effect was significant for each augmentation, with the exception of T_3 (rescaling) in the categorization task. In general, we see that T_2 (horizontal bending) had the largest marginal effect while T_3 had the smallest. Intriguingly, T_2 is the augmentation that is closest to the topological transformations originally proposed by Chen [2005] ("tolerance spaces"). This bolsters our hypothesis from the Introduction that the augmentations will prove useful to the extent that they reflect commonly occurring structure in the world-since the most useful augmentation ended up being the one that had previously been singled out for its ecological relevance.

Finally, we see from the residual diagnostic plots in Figure 5.5 that the ANOVA assumptions are met reasonably well (residuals are approximately normal, and do not depend systematically on predicted value), thus giving credence to the estimated β weights.

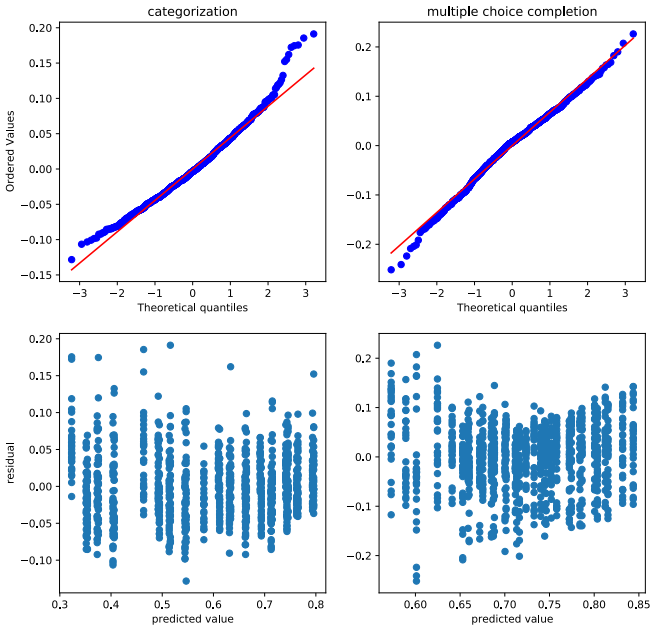

Figure 5.5: Residual diagnostic plots for the ANOVA in Equation 5.8. Top row shows residual QQ plots, with the red line corresponding to a perfect normal fit. Bottom row shows residuals plotted against predicted values.

5.17 Effect of optimizing GP hyperparameters

In the multiple choice completion task, the hyperparameters of the GP kernels were fixed independently of the prompt curves. Thus for example, to compute the SM completion of a prompt curve y_{obs}, we computed

$$\hat{y}_{SM} = \mathbb{E}_{SM}(y_{ext}|y_{obs}, \theta_0)$$

, where θ_0 are some fixed values of the hyperparameters of the SM kernel (with the CG completion constructed similarly). In this section, we consider the effect of a modified construction, which is arguably closer to standard practice with GPs, and instead define the completion as

$$\hat{y}_{SM} = \mathbb{E}_{SM}(y_{ext}|y_{obs}, \theta^*)$$

where the hyperparameters θ^* are optimized with respect to y_{obs}:

$$\theta^* = argmax_\theta \mathbb{P}_{SM}(y_{obs}|\theta),$$

and similarly for the CG case. To be more precise about the CG case, we separately optimized the hyperparameters for each of the constituient kernel classes in the CG, and then selected the one that attained the highest marginal likelihood, as in Schulz et al. [2017]. We performed the optimizations using the pyGPs library Neumann et al. [2015], with the default "Minimize" optimizer.

In Table 5.7 we show the results of the multiple choice analysis under this change. Due to the increased computational demands of performing the optimization for every prompt

Table 5.7: Downstream accuracy on multiple choice extrapolation task, with GP hyperparameters optimized to prompt curve prior to computing posterior means (cf. Table 2 in main text)

train size model name	3	10	30	100	300
contrastive	59.77± 1.83	63.91± 1.83	62.33± 2.56	61.87± 0.76	62.74± 1.82
cnp	50.81± 0.54	52.38± 1.42	50.84± 0.97	50.72± 0.22	50.68± 0.45
cpc	52.18± 2.31	52.89± 2.25	54.80± 2.87	56.21± 1.87	58.55± 1.65
raw	50.86± 0.55	51.39± 2.94	51.36± 2.25	50.87± 0.70	50.80± 0.80
t-loss	50.71± 0.83	51.35± 0.81	50.80± 0.83	51.74± 0.79	54.33± 1.39
tnc	57.03± 1.99	56.52± 2.33	55.04± 3.07	56.04± 1.26	56.87± 1.11
vae	50.23± 0.19	50.24± 0.28	50.10± 0.22	50.15± 0.18	50.48± 0.28
contrastive-perm-inv	50.00± 1.38	52.04± 1.41	50.92± 1.58	51.61± 0.51	51.69± 0.97
contrastive-mlp	56.44± 1.52	57.52± 1.40	57.22± 1.27	58.58± 0.78	59.07± 1.00

Table 5.8: Value of Δ_{acc} on multiple choice task, with optimized GP hyperparameters (cf. Table 4 in main text)

train size model name	3	10	30	100	300
contrastive	20.61± 7.84	16.49± 4.42	13.66± 3.64	17.81± 1.92	20.08± 4.50
cnp	1.03± 6.53	2.45± 7.90	-0.71± 2.85	1.27± 1.87	0.48± 0.42
cpc	-3.64± 7.93	-8.97± 7.34	-7.98± 4.58	2.01± 3.75	9.31± 1.21
raw	3.56± 2.48	2.11± 8.63	3.58± 2.34	0.84± 1.14	1.23± 0.61
t-loss	-1.29± 3.19	-1.17± 3.06	-2.73± 2.13	-0.77± 2.02	6.22± 2.55
tnc	25.63± 5.76	10.71± 8.90	7.63± 7.80	9.35± 2.47	10.26± 1.56
vae	-0.54± 0.75	-0.25± 0.76	-0.33± 0.74	-0.16± 0.47	1.04± 0.41
contrastive-perm-inv	-5.73± 7.19	1.92± 5.67	-2.18± 3.00	0.44± 1.00	0.28± 0.46
contrastive-mlp	5.92± 3.32	1.91± 4.65	2.02± 3.50	5.66± 3.41	7.04± 1.68

curve, we averaged over 3 different ground truth hyperparmeter values, rather than 10 as in the main text. As with Table 2 in the main text, we find that the contrastive model performs the best. However, the overall accuracies are markedly lower than in Table 2. This is to be expected, since allowing for the GP hyperparameters to be optimized transforms the task to the more difficult one of distinguishing between kernel classes rather than individual kernels, as in the main text.

As regards the difference in accuracies for SM/CG curves, we find that the bias is reduced for most models with the exception of the contrastive, as in Table 5.8.

However, we caution that since the GP marginal likelihood optimization is non-convex, it is possible that the results are influenced by some imperfection in the optimization algorithm itself.

The aspects of things that are most important for us are hidden because of their simplicity and familiarity.

Ludwig Wittgenstein

6

Base Addition with Neural Networks

The material in this chapter is adapted from an in-preparation manuscript done in collaboration with Kamesh Krishnamurthy and Jonathan Cohen.

6.0.1 ABSTRACT

The conditions under which neural networks develop abstractions remain a subject of intense debate. While the prevalent approach to evaluating performance centers on statistical generalization, there has been relatively little exploration of algorithmic generalization. To address this gap, we employ arithmetic as a foundational framework. Our aim is to uncover the inductive biases crucial for fostering algorithmic, rather than solely statistical, generalization. We investigate the impact of key design considerations such as recurrence versus self-attention mechanisms, the significance of the relational bottleneck, and the utilization of external memory. Our overarching objective is to shed light on how neural networks acquire and utilize abstract procedures, providing insights to enhance their capabilities beyond traditional statistical measures.

6.1 Introduction

A key ingredient of (arguably the only necessary ingredient for) generalization is symmetry. Indeed, in a fundamental sense, generalizing from one domain to another can only make sense under the assumption that there is some shared structure common to the two domains. In physics, this has been systematized and exploited via famous results such as Noether's Theorem, to the point that Phil Anderson has remarked that "It is only slightly overstating the case to say that physics is the study of symmetries" [Anderson, 1972]. Crucial to this viewpoint is the notion of a *symmetry function* [Hamermesh, 1989], which is a function that is invariant under symmetry of the relevant system. An example of such a function is Newton's law for the force of gravitational attraction between two objects;

the strength of this force will be the same for two objects regardless of whether they are located on Earth or Mars, provided that the relative distance between the objects is kept unchanged.

Here, we aim to carry over this viewpoint to the study of algorithmic generalization. The basic thought is that, just as many physical laws can be succinctly described using the language of symmetry (for example, the force of gravity acts in the same manner for any location on the earth), many algorithmic procedures can also be described in terms of a repeating "kernel" of computation, which can be considered as the analogue of a symmetry function in this setting. To take a simple example, consider the computation of F_n, the nth Fibonacci number. Recall that these are defined by setting $F_0 = 0, F_1 = 1$, and extended inductively for larger n through the relation $F_n = F_{n-1} + F_{n-2}$. A simple * algorithm is to start with the vector $(0, 1)$, and then iteratively apply the function $T : (x, y) \mapsto (y, x + y)$ $n - 1$ times in succession; the result will be (F_{n-1}, F_n). Crucially, this computation involves only iterative application of the *same* function T; thus, this function may be regarded as a symmetry function for this algorithm. A similar important example, studied extensively in Chapter 3, is a linear autoregressive model.

Here, we aim to uncover the inductive biases that encourage neural networks to discover and apply such symmetry functions in algorithmic tasks. We study a seemingly-simple setting which, like the Fibonacci example, requires the iterative application of a symmetry function: integer addition in base arithmetic. We first analyze this problem at an abstract Group-theoretic level: Using the tools described in Chapter 1, and in particular the analysis of [Isaksen, 2002], we precisely characterize the symmetry function involved in the

*but relatively inefficient

addition operation. We then show how to use Group-theoretic results to generalize the operation to a family of *nonstandard addition rules*, which can be used as a rich testbed for the algorithmic generalization abilities of neural networks. We conjecture that our standard carry table is distinguished among these through its simple recursive form, which allows for the possibility to systematically and parametrically construct more difficult generalization tasks using appropriate alternative tables. Next, we present learnability results using multiple neural network architectures and training paradigms. In particular, we hypothesize that the Relational Bottleneck may provide a useful inductive bias in this setting, which leads us to consider the Abstractor architecutre [Altabaa et al., 2024]. We consider as well recurrent networks, because the abstract symmetry structure of the problem (namely, repeatedly applying a computational kernel) maps very cleanly onto the architectural form of such models, in which the weight-sharing through time allows the network to implement the same function at different positions in its input. Taken together, we aim to isolate the neural network primitives which encourage the learning and systematic application of algorithmic symmetry functions.

6.2 Carry tables and Cocyles

We consider the problem of addition of two positive integers in a fixed base representation. While this problem is very familiar, the underlying structure is surprisingly rich and subtle, which provides many opportunities to understand the benefits and limitations of various neural network design choices. Furthermore, to our knowledge this is not yet any neural architecture that has been shown to be able to learn and use (for unbounded extrapolation) the underlying symmetry structure of base arithmetic, and use this in demonstrably algo-

rithmic form.

In what follows, we will consider some fixed integer $d > 1$ which serves as the modulus of the base representation (for example, $d = 10$ corresponds to the standard decimal base representation, while $d = 2$ corresponds to binary). Let us first introduce a useful notation: if a and b are non-negative integers in the range $[0, d)$, then we use $+_d$ to denote their sum modulo d (which also lies in the range $[0, d)$), and reserve $+$ for the standard integer sum. For example, $3 +_6 4 = 1$ and $3 + 4 = 7$. The set of integers $[0, d)$ together with the operation $+_d$ is denoted by $\mathbf{Z}/d$, and forms a Group, as in the sense of Chapter 1.

We will denote the base representation of an integer $n \geq 1$ in the following form:

$$n = [a_k, a_{k-1}, \ldots, a_1]_d \tag{6.1}$$

where a_i are integers in the range $[0, d)$. The a_i are determined by the relation $n = \sum_{i=1}^{k} a_i d^{i-1}$, and it is a basic fact of arithmetic that any n has such a unique base representation (up to zero-padding on the left side) and conversely any such set of integers a_i corresponds to an integer n. Also, we will drop the d subscript when it is clear from context (as it usually is).

The basic task we study is as follows: given two integers n and m represented in their corresponding base formats, return the base representation of the sum $n + m$.

For the sake of clarity, let us review the familiar algorithm which is typically taught in school. Let $[a_k, \ldots, a_1]$ and $[b_k, \ldots, b_1]$ be the base representations of n and m, respectively. We can assume that they have the same number of digits, since if not we can always zero-pad the shorter one. The algorithm proceeds as follows. We first introduce an auxiliary

variable $c_1 = 0$. We can then successively read off the digits s_i of the sum as follows:

$$s_i = c_i +_d a_i +_d b_i \tag{6.2}$$

$$c_{i+1} = 1_{c_i + a_i + b_i \geq d} \tag{6.3}$$

The algorithm starts at $i = 1$ and terminates once we get to $i = k+1$. Here, c_i is interpreted as the amount that is "carried" from the previous position. We can now make several simple but important observations about this procedure. First, it is inherently iterative in that it requires performing an operation at each "position" i in the input. Secondly, it contains a repeating "kernel" (or "symmetry function") which is applied in exactly the same way at all places, namely the above equations are identical for s_i and c_{i+1}. Crucially, the kernel requires learning only finitely-many input-output associations (since the inputs a_i, b_i, c_i are constrained to take only finitely many values). This contrasts strongly with the full addition operation, which can in principle be applied to arbitrarily large inputs, and so cannot be reduced to a lookup table. Thus abstractly, this algorithm requires both learning the underlying symmetry function (invariance) as well as sliding and applying it at multiple positions (equivariance).

We can also see that this is a special case of a more abstract algebraic structure which also shares the above two properties of invariance and equivariance. Namely, starting from the two input lists $[a_k, \ldots a_1]$ and $[b_k, \ldots, b_1]$, we can *define* their sum as the list $[s_{k+1}, \ldots, s_1]$

where the entries are given inductively by initializing $c_1 = 0$ and

$$s_i = c_i +_d a_i +_d b_i \tag{6.4}$$

$$c_{i+1} = f(a_i, b_i, c_i) \tag{6.5}$$

where f is some as-yet-unspecified function. Now, while we certainly *can* do this, it is worth pausing at this point to explain why we would *want to*. The main motivation is that this construction has the same high-level properties and structure as the standard carry table (namely, the invariant and equivariant aspects), yet has a different underlying kernel/symmetry function (namely, the function z in place o the standard carry rule). As such, understanding the differences in learnability between these two case can allow us to dissociate the learning of an individual symmetry function from the learning of how to systematically use such a symmetry function. Additionally, it can possibly allow us to understand any distinguishing features of our standard carry rule that make it somehow more efficient or natural than alternatives.

Let us now turn to the unspecified function f and understand what constraints to place on it. An important property of the standard addition operation, which we want to impose *a priori* on any other alternative addition operation is *associativity*. Informally, this means that if we want to add more than two elements together, then the final answer does not depend on the order in which we group the terms. In symbols:

$$x + (y + z) = (x + y) + z \tag{6.6}$$

for all x, y, z. [†] To understand when the generalized operation is associative, let us first consider a simplified version, in which $k = 2$ and we ignore the last digit of the sum. In this case, the addition rule may be parametrized as

$$[a_2, a_1] +_z [b_2, b_1] = [a_2 +_d b_1 +_d z(a_1, a_2), a_1 +_d b_1] \tag{6.7}$$

where $z : \mathbb{Z}/d \times \mathbb{Z}/d \to \mathbb{Z}/d$ is some function. This is precisely the Group Extension setting that we discussed in Chapter 1, and as described there, it is possible to completely characterize which zs yield an associative operation.

Proposition 3. *The addition rule $+_z$ defined in Equation 6.7 is associative if and only if z satisfies*

$$z(a, b +_d c) +_d z(b, c) = z(a +_d b, c) +_d z(a, b) \tag{6.8}$$

for all $a, b, c \in \mathbb{Z}/d$.

For details of the proof see [Isaksen, 2002]. The above equation is often called the *Cocycle condition* and functions z which satisfy it are called *cocycles*. In the present setting, we will also refer to such a function z as a *Carry table*, because it specifies what digit gets carried to the next place. Now let us consider the next simplest case of lists of length 3:

$$[a_3, a_2, a_1] +_z [b_3, b_2, b_1] = [a_3 +_d b_3 +_d z_2(a_2, a_1, b_2, b_1), a_2 +_d b_2 +_d z(a_1, a_2), a_1 +_d b_1] \tag{6.9}$$

where z_2 is the digit that is carried from the addition of the last two digits. How should z_2 be defined? To understand this, let us understand how it would look in the standard carry

[†] It is easy to show that this condition implies that if there are more than 3 terms in the sum, then grouping the terms according to any valid set of parentheses will give the same answer.

table. In this case, suppose we have two integers $[a_2, a_1]$ and $[b_2, b_1]$. The sum will take the form $[c_3, *, *]$, and we want to characterize c_3 in terms of the two summands. On the one hand, if $a_2 + b_2 \geq d$, then we will carry a 1 to the next place. On the other hand, we could still carry a 1 if $a_2 + b_2 = d - 1$, and $a_1 + b_1 \geq d$. In any other situation, there is no carried digit i.e. $c_3 = 0$. This can be summed up as

$$\begin{aligned} c_3 &= z_{standard}(a_2, b_2) +_d z_{standard}(a_2 +_d b_2, z_{standard}(a_1, b_1)) && (6.10) \\ &= z_{standard}(a_2, b_2) +_d z_{standard}(a_2 +_d b_2, c_2) && (6.11) \end{aligned}$$

where $z_{standard}$ denotes the standard table $z_{standard}(a, b) = 1_{a+b\geq d}$. In general, by the same argument we can see that

$$c_{k+1} = z_{standard}(a_k, b_k) +_d z_{standard}(a_k +_d b_k, c_k) \tag{6.12}$$

Thus, we see that the natural definition of f in terms of the carry table z is

$$f(a, b, c) := z(a, b) +_d z(a +_d b, c) \tag{6.13}$$

When z is the standard carry table, it is easy to check that $f(a, b, c) = 1_{a+b+c\geq d}$, as per Equation 6.3. However, now we have an expression that can be generalized to any carry table z.

Now let us return to the question of associativity. While we have seen that associativity in the 2-digit case reduces to Cocyle condition, the general case is more complicated. In particular, even if z is a cocyle, it may happen that the generalized addition rule fails to be

associative for inputs of length ≥ 3. This leads us to introduce

Definition 9. *A function $z : \mathbb{Z}/d \times \mathbb{Z}/d \to \mathbb{Z}/d$ is a k-Recursive Cocyle if the corresponding addition rule*

$$
\begin{aligned}
[a_k, \ldots, a_1] +_z [b_k, \ldots, b_1] &= [s_k, \ldots, s_1] && (6.14)\\
s_i &= c_i +_d a_i +_d b_i && (6.15)\\
c_{i+1} &= z(a_i +_d b_i, c_i) +_d z(a_i, b_i) && (6.16)\\
c_1 &= 0 && (6.17)
\end{aligned}
$$

is associative. We say that z is an Infinitely Recursive Cocyle if it is a k-Recursive Cocyle for each $k \geq 1$.

Clearly, the standard carry table is an Infinitely Recursive Cocyle. In general, however, determining whether a given z is an Infinitely Recursive Cocyle is very challenging, and we have not found a systematic way to do so. Even verifying that such a z is an Infinitely Recursive Cocyle of some finite order can quickly become computationally intractable due to the exponential growth in the number of elements that must be checked for the associativity constraints. This leads us to make the following conjectures, in the hope of spurring research into such questions:

Conjecture 1. *For any modulus $d > 2$, there exist Infinitely Recursive Cocyles besides $z_{standard}$.*

Conjecture 2. *Any 3-Recursive Cocyle is an Infinitely Recursive Cocyle.*

Finally, we note that it is often useful for the purpose of intuition and visualization to think about the recursive addition operation in terms of *higher order carry tables*. Simply put, if we start with some carry table z, then this also induces tables $z_{(k)}$ for any $k \geq 1$, which encode the final carried digit for each pair of numbers with k digits. In symbols:

$$[a_k, \ldots, a_1] +_z [b_k, \ldots, b_1] = [z_{(k)}([a_k, \ldots, a_1], [b_k, \ldots, b_1]), *_1, \ldots, *_k] \tag{6.18}$$

where the $*_i$ are some digits whose precise values are immaterial for this definition. Each $z_{(k)}$ is therefore a matrix of size $d^k \times d^k$, where d is the modulus.

Using the iterative definition of $+_z$, we can readily derive a recursive formula for $z_{(k)}$:

$$z_{(1)}([a_1], [b_1]) = z(a_1, b_1) \tag{6.19}$$

$$z_{(k)}([a_k, \ldots, a_1], [b_k, \ldots, b_1]) = z(a_k +_d b_k, z_{(k-1)}([a_{k-1}, \ldots, a_1], [b_{k-1}, \ldots, b_1])) \tag{6.20}$$

$$+_d \; z(a_k, b_k) \tag{6.21}$$

6.3 Exploration of Non-standard Carry Tables

We first provide an exploration of different kinds of non-standard carry tables that can arise. The goal here is to simply visualize the kinds of structures that are possible and get an intuition for what these look like. From a learnability viewpoint, this is interesting because while the alternative tables are equivalent to the standard table in an abstract group-theoretic sense, they are also in some sense less efficient and more complex; this suggests that they may be discovered by architectures that are less constrained or do not have a bias towards abstraction, while providing potential insight into the distinguishing features of

our standard table that have caused us to use it instead of an alternative.

We first show an example of a non-standard carry table for addition modulo 10 in Figure 6.1. As an example usage, let us consider the addition rule $34 +_z 17$ where $+_z$ denotes the addition operation induced by this table. The result is 91. We can verify this by simply walking through the definition of the addition rule. The ones digit is given by $(4 + 7)\%10 = 1$. To determine the tens digit, we first look up the $(4, 7)$ entry in the table which happens to be 5. The tens digit is then given by $(3 + 1 + 5)\%10 = 9$.

Surprisingly, this addition operation turns out to be isomorphic to the usual cyclic addition operation on $\mathbb{Z}/100$, meaning that the two groups differ only by a relabeling of the elements. This is illustrated by the second panel in the figure, in which we see that successive addition by 1 can generate all integers up to 100 using the nonstandard table.

At the same time, this table differs fundamentally from the standard table in that recursively applying the table with more digits (that is, passing to the higher-order carry tables $z_{(k)}$) does not yield an associative operation.

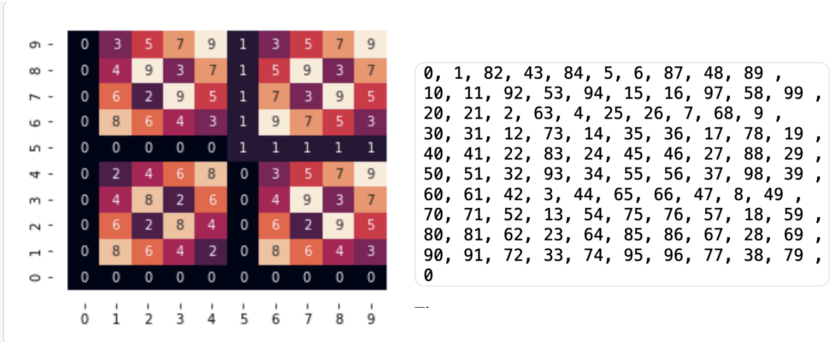

Figure 6.1: An illustration of a non-standard carry table for addition modulo 10. The left panel shows the table itself, and the right shows the corresponding counting sequence, i.e. the result of successive addition by 1.

We now explore the iteration rule for carry tables and the resulting higher-order tables. In Figure 6.2, we start with a non-standard carry table for $\mathbb{Z}/4$, as shown in the first panel.

We then successively apply the rule given by Equation 6.19 to recursively generate higher-order carry tables from the same rule. In contrast to the carry table shown in Figure 6.1, all of these recursively-generated tables define associative addition rules. Furthermore, we can see from the figure that the higher-order tables display fractal self-similar structure.

Figure 6.2 shows more examples of possible carry tables for different modulus values, as well as the results of iterating these tables to higher order in 6.4. Again we see that the non-standard tables display intricate fractal structure under the iteration operation.

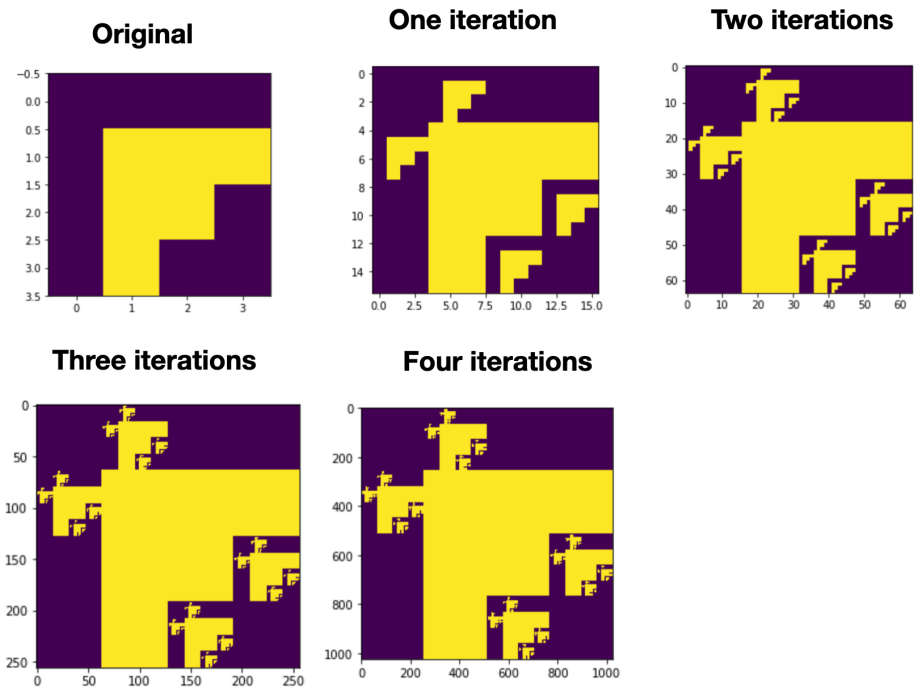

Figure 6.2: Illustration of the recursive expansion of a non-standard carry table for $\mathbb{Z}/4$. Note that the scale of the axes grows by a factor of 4 with each additional iteration, corresponding to the addition of an additional digit.

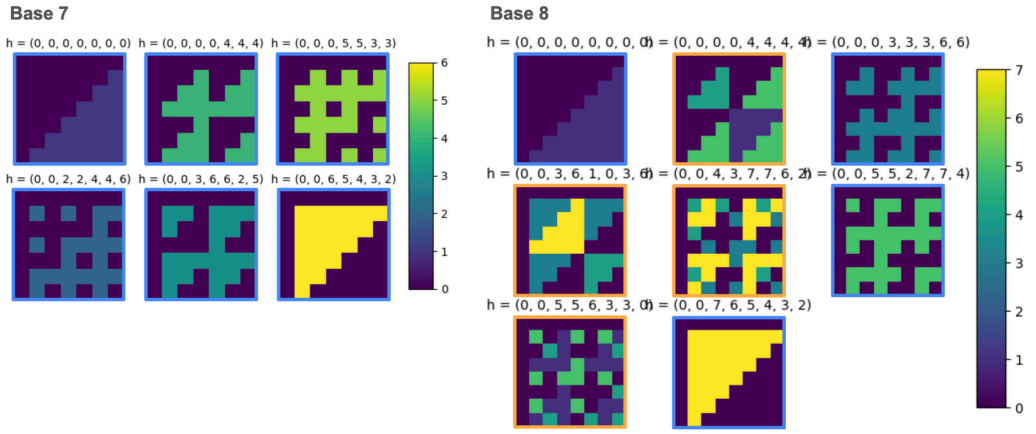

Figure 6.3: Illustration of possible carry tables for $\mathbb{Z}/7$ and $\mathbb{Z}/8$ that preserve associativity under iteration. Each table shows the coboundary function h which was used to generate it, as per Proposition 2 in Chapter 1. Figure courtesy of Cutter Dawes.

6.4 Results on Neural Network learnability

We now study the relative difficulty of learning different carry tables for neural networks, as well as their generalization abilities to longer sequences than seen during training. As discussed before,our intention with these experiments is to both rigorously understand the design decisions that can facilitate systematic generalization across a wide range of problems (i.e., not just the single problem of addition with the standard table), as well as to determine whether our own carry table is easier to learn than the alternatives, thus pointing towards a possible normative justification for using the standard table.

6.4.1 Tasks and Inputs

We represent each addition problem as a sequence of one-hot vectors, with each vector representing an individual digit. There is also a special token, which we denote "?" which

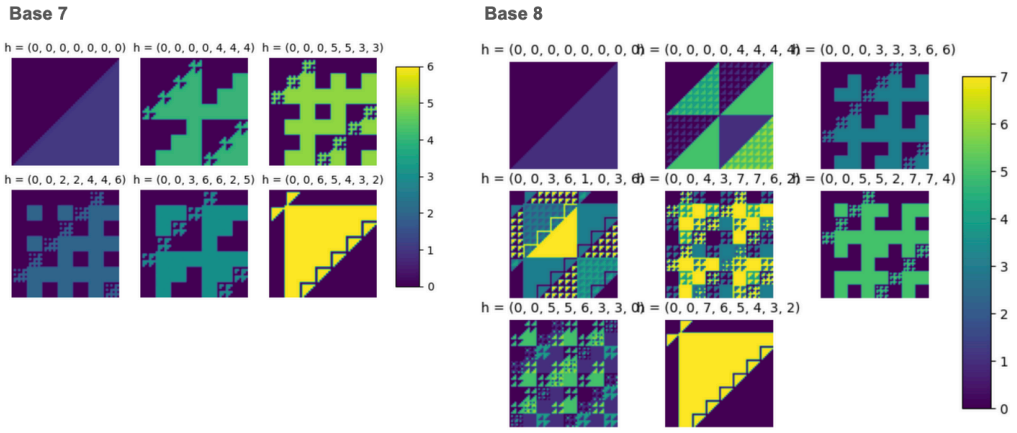

Figure 6.4: Illustration of one step of recursion for the tables shown in Figure 6.3. Figure courtesy of Cutter Dawes.

indicates that the model should output the appropriate digit of the answer at that position in the sequence.

We consider three different formats in different levels of difficulty, as illustrated in Figure 6.5.

Intuitively, the interleaved condition is easiest because it maps most cleanly onto the abstract structure of the problem. Namely, after seeing the ones digits of the two constituent numbers, one can immediately deduce the ones digit of the sum, and report it at the first "?" token, without any need to "remember" that digit, and report it in the proper place later. Similarly, we can deduce the tens digit of the sum after seeing the tens digits of the two constituent numbers (provided we have also encoded the appropriate carry digit from the ones place), report that result (again, obviating the need to remember it), and then move on to the next place. The partial interleaved condition provides more of a challenge, because while the model can still in principle compute the digits incrementally as in the interleaved case, it must at the same time store the intermediate results of the computations so that it can report them at the end of the sequence. Finally, the blocked condition further

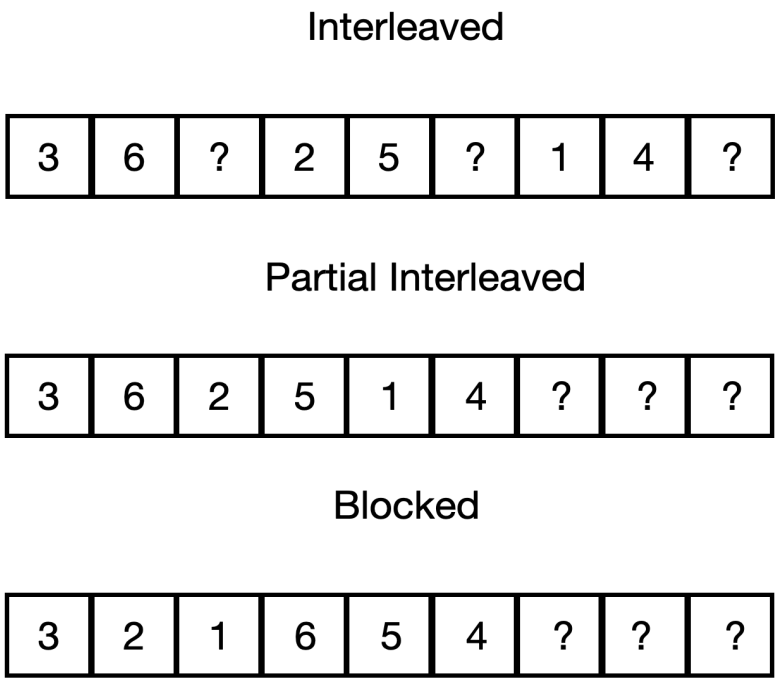

Figure 6.5: Illustration of different input formats for example addition problem $123 + 456$. Here "?" denotes a special token which indicates that the model should output the corresponding digit of the answer at that position. The blocked condition also includes a delimiter between the two digit sequences, not shown here.

exacerbates this problem, because the inputs to the partial computations must themselves also be stored in memory, since they do not appear in adjacent positions (i.e. after seeing 3 in the example, the model must wait until encountering 1 to perform the first partial computation, namely the ones digit of the sum, and maintain this value 3 the entire time so it may be used for this computation alter).

In all cases, the models take as input a sequence of shape $N \times d_{in}$, and return an output of shape $N \times d$. Here N may potentially be variable. This flexibility with respect to input size is accommodated by both recurrent networks and transformer-encoder based models. Moreover d is here the modulus of the addition rule, and the outputs of the network are

interpreted as logits for each of the d possible digits at the respective position. Finally d_{in} is the dimensionality of the input vectors. For example, in the interleaved case this would be $d + 1$, corresponding to one token for each of the possible digits, plus an extra for the "?" token.

The models are trained to output the digits of the sum in the respective "?" positions. Importantly, the model outputs are ignored for the other tokens. That is, if we let $T_i \in \{0, \ldots, d-1\}$ denote the digits of the sum, then the loss function is given by

$$L(\theta) = \sum_i CE(m_\theta(X)_{j_i}, T_i) \tag{6.22}$$

where X is the input sequence, $m_\theta(X)_j$ is the model output in the jth position of the sequence, and j_i denotes the position of the ith "?" token in the input sequence [‡]. Finally CE denotes the standard cross-entropy loss function: $CE(Z, i) = -Z_i + \log \sum_j e^{Z_j}$.

6.4.2 Model Architectures and Implementation

We tested the following architectures: LSTMs and transformers, as baseline comparisons; and an Abstractor [Altabaa et al., 2024], to evaluate whether a Relational Bottleneck-based architectural constraint that is otherwise comparable to a standard transformer yields an advantage in this setting.

For the self-attention based models (Transformer and Abstractor) we also employ causal masking and learnable position encodings. The position ids are taken to be random contiguous subsequences from the interval $[0, N_{max}]$ where N_{max} is taken to be much larger than the sequences seen during training. That is to say, given a sequence of length N, the

[‡]for example, in the interleaved condition as in Figure 6.5, we would have $j_1 = 3, j_2 = 6$ and $j_3 = 9$

positions would be defined to be $i, i+1, \ldots, i+N-1$ for some randomly-sampled i, and the corresponding position codes for these elements would be added to the sequence prior to feeding it to the model. The reasoning for this approach is that the model sees all possible position codes during training time. During test time it sees longer sequences but never encounters a completely new and uninitialized position token. At the same time, this allows the position codes to be learned and allows us to avoid using hard-coded formulas for the position embeddings as in [Vaswani et al., 2017].

Finally, in order to quantify the differences between the standard carry table and the alternatives, we consider two conditions for each model/task combination: one in which the model is trained on the standard addition rule, and another in which the model is trained on a randomly-generated alternative carry table, as described in Section 6.2.

All models are trained using the Adam optimizer using a learning rate of .001 and batch size of 64. In all experiments we use a modulus of $d = 6$ and train on sequences with at most 6 digits per number. We arbitrarily define one epoch of training as 10000 example sequences, and use this unit in the training graphs shown subsequently. Models were trained using 500 epochs (i.e. 5 million example sequences) except where otherwise indicated.

6.4.3 Results

We first consider the performance of the self-attention based models (i.e. Transformer and Abstractor) in the interleaved condition. We see in Figure 6.6 that while the models can perfectly fit the training data, they do not learn a completely systematic procedure for performing the addition. Further, we see that the incorporation of the Relational Bottleneck through the Relational Cross Attention of the Abstractor substantially harms the general-

ization performance in this case, relative to a standard Transformer.

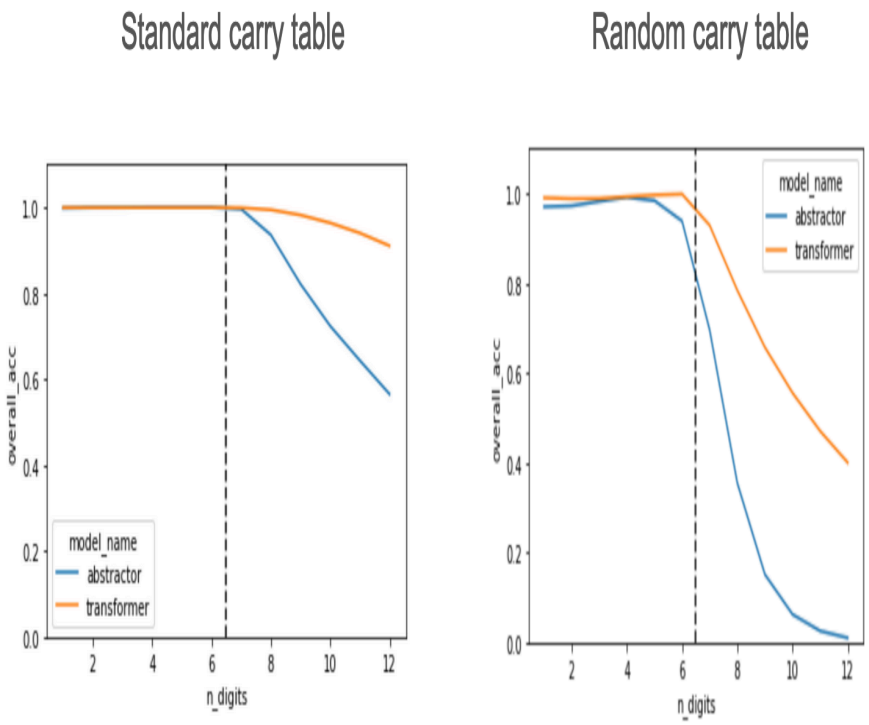

Figure 6.6: Results of transformer and Abstractor in the Interleaved condition. The models can learn to fit the training distribution but fail to learn a completely systematic solution.

In contrast, as shown in Figure 6.7, we can see that the LSTM model can not only learn the task but can achieve perfect generalization on sequences up to (at least) double the size of the training sequence. We interpret this as the LSTM having learned the "symmetry kernel" of this problem. Due to the structure of the Interleaved inputs, the LSTM only needs to learn to store the previously-carried digit in its hidden state, as well as learn the rule for modular arithmetic for three digits (i.e. the two digits of the addends as well as the carried digit). Once it has learned these two successfully, it will be able to successfully apply it arbitrarily many steps due to the recurrent nature of the network. In this case, we see that the basic inductive bias of a recurrent network of *weight sharing through time* is nearly

perfectly suited to the recursive structure of this task.

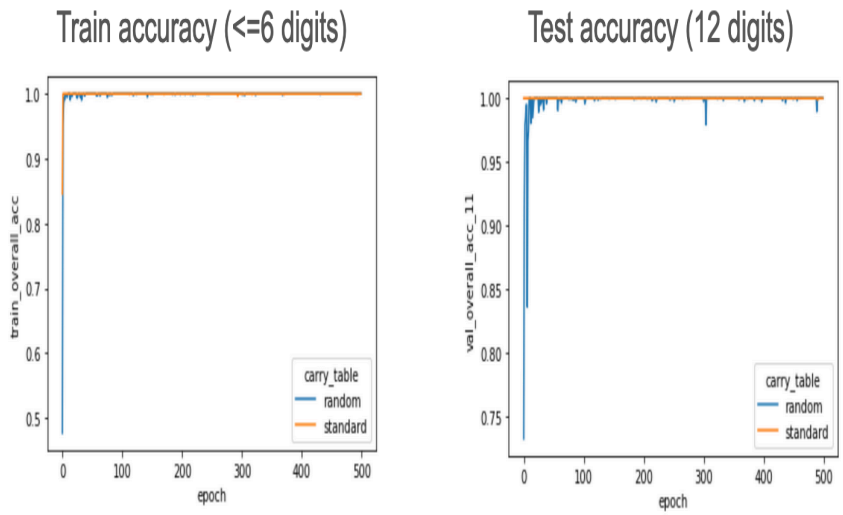

Figure 6.7: Results for the LSTM in the interleaved condition. The left shows training accuracy on sequences with up to 6 digits, while the right shows validation accuracy on 12 digits. The model quickly attains ceiling performance and learns a generalizable algorithm.

This hypothesis about the functioning of the LSTM in the interleaved case is confirmed by results in the Partial Interleaved case. Here, as described above, the network cannot immediately externalize its partial computations and thus must store them internally, which has the effect of "distributing" the symmetry over multiple processing steps. Accordingly, see as in Figure 6.8 that the LSTM struggles heavily. We also evaluated an mGRU [Krishnamurthy et al., 2022] model to see whether this was a particular weakness of the LSTM or was a general feature of recurrent models and found it displayed the same behavior as the LSTM.

As per the discussion in Section 6.4.1, we hypothesized that allowing a recurrent network to store its intermediate states in an external memory would allow it to overcome this challenge. That is, by storing the intermediate inputs and computations and reading from

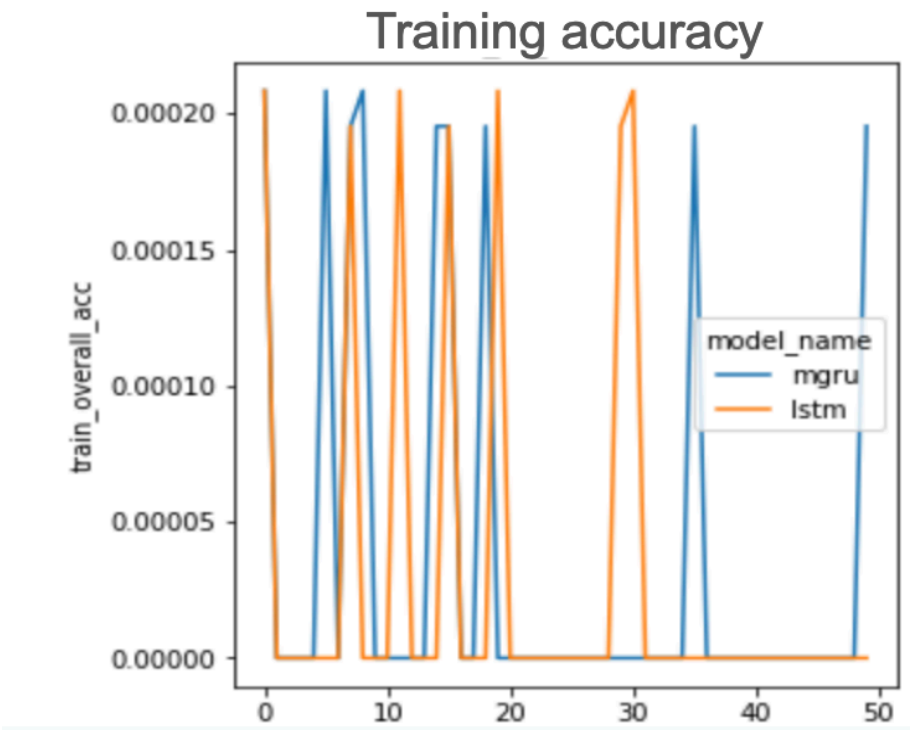

Figure 6.8: Learning curves for LSTM and mGRU in the partial interleaved condition (note the vertical scale!). Both models fail to get off the ground.

them as needed, the network may be able to effectively rearrange the input into an ordering in which the computation itself is easy (that is, the interleaved ordering). To test this, we combined the LSTM with a Differentiable Neural Dictionary [Pritzel et al., 2017], which is a simple form of external memory that accommodates write operations and similarity-based retrieval. More specifically, at each position in the input, we write a key/value pair to the memory consisting of the current hidden state/output vector of the LSTM. The output of the model at the position is then obtained by querying the dictionary using the current LSTM hidden state. For this model, we trained on only 100 epochs due to the added complexity of the architecture, but found this training time was sufficient for convergence. As shown in Figure 6.9 this modification does indeed allow the model to fit the partial inter-

leaved condition; however it is not able to generalize to longer sequences than seen during training. Indeed, the model must implicitly perform a kind of index arithmetic when iterating through the "?" tokens at the end of the sequence: when it is at the ith query token, it must know to retrieve the $(2i-1)$th and $(2i)$th digits in the input sequence. For sequences of a fixed length it is possible to simply memorize these index arithmetic results, however in order to legnth generalize, it is necessary for the model to generate an internal representation of positional indices that are larger than those encountered during training, in such a way that it can perform this index arithmetic operation correctly on these new representations. As we do not constrain the hidden states of the model in any way, it is likely difficult for it to accomplish this. We speculate that imposing stronger inductive biases on the position encodings used by the model, possibly using a code similar to that from the Temporal Context Model [Howard and Kahana, 2002] may help with this issue.

6.5 Related work

In [Lee et al., 2024] the authors systematically consider the effects of various data formats and network design choices on the ability of transformers models to learn standard integer base addition. One finding is that chain of thought [Wei et al., 2022], that is, guiding the model to take a certain reasoning path to solve a problem, can help dramatically; however, this introduces a substantial amount of supervision, whereas our interest in mechanisms that require less supervision. More importantly, they find that the models struggle to generalize to different lengths than seen in training (i.e., are constrained to within distribution generalization), suggesting that the models have not fully discovered the underlying symmetries required to perform the task algorithmically.

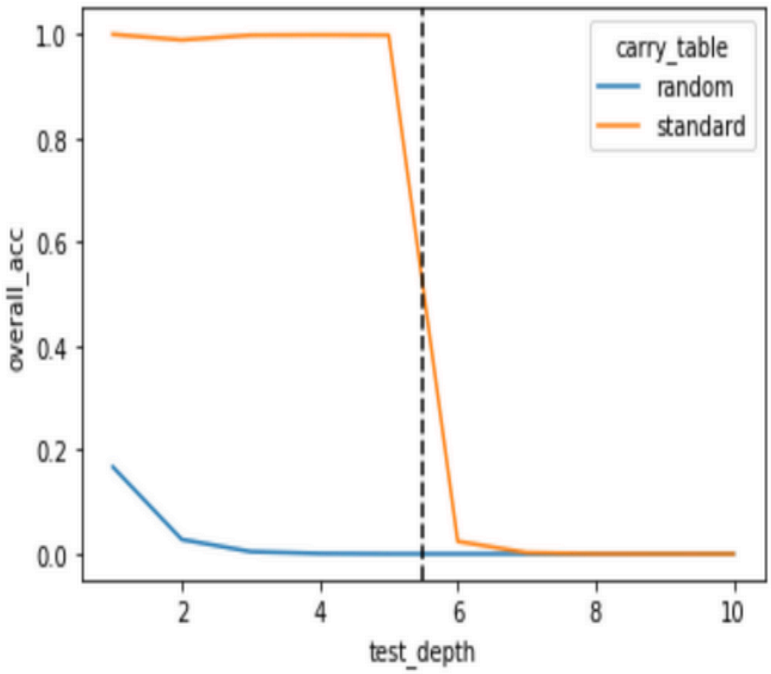

Figure 6.9: Results of an LSTM augmented with a Differentiable Neural Dictionary external memory module, in the setting of Figure 6.8. The model can now fit the training data, but does not learn a generalizable solution.

[Zhou et al., 2023] did study the issue of length generalization in transformers, and showed that in several tasks, including base addition, it is possible to obtain near-perfect generalization by the inclusion of so-called index hints, which are special tokens inserted in the input that explicitly encode the relative position of each digit within the sequence. Although this is a very interesting observation, it involves some degree of hand-engineering of the input, making it closer to neuro-symbolic forms of models, that prespecify symbolic primitives, rather than allowing the model to discover the underlying symmetries (i.e., "symbolic" structure) of the problem on its own. This is also consistent with the study of [Shen et al., 2023], which found that the performance of transformers on the addition task can be highly sensitive to the details of the positional encoding used in the input.

Another interesting recent study [Jelassi et al., 2024]compared the performance of transformers with a class of recurrent models called *State Space models* [Gu et al., 2021, Albert Gu, 2023]. They found that the recurrent models were able to perform better at a task that required retrieving arbitrary parts of the input. While not specifically involving an arithmetic task, this provides an interesting demonstration of the hypothesis that recurrence may facilitate the learning of certain algorithmic procedures (that involve caching intermediate products) more effectively than simply self-attention.

In [Dulberg et al., 2021] the authors used the ESBN architecture to model a simple counting task (in effect, addition by one). They found that the ESBN-based model showed a developmental timecourse which was closer to that of children learning the task than comparison models (including transformers). Furthermore, after successfully learning how to count to 5, the model more effectively generalized out of range (i.e., to larger numbers). This suggests that inclusion of a relational bottleneck may also facilitate the learning of systematic algorithms.

Further, the models used in our study are quite similar to the Neural Turing Machine [Graves et al., 2014],both with respect to the architecture, which combines recurrence and an external memory, and with respect to some of the tasks tested, which included the "copy-repeat" task, in which the model must repeat an essentially arbitrary pattern a specified number of times, which requires some form of counting.

Finally, our high level goal of understanding the process by which neural networks can learn symbolic procedures is somewhat similar to the topic of *program induction* [Solomonoff, 1964], in which the goal is to infer a computer program (i.e., algorithm) that can generate the observed data, and then generalize to new observations by executing the program [Ellis

et al., 2020, Lake et al., 2015]. However, our approach differs from this in a fundamental way: We do not want to implement any symbolic primitives needed to compose such programs, but rather identify architectural features of the model that predispose it to learning these on its own.

6.6 Discussion

We have identified base arithmetic as a useful test case for evaluating the learning of symmetry functions in neural networks. We have additionally identified the effect of key design choices, such as recurrence vs. self attention and/or the presence of an external memory, on the performance of the models.

Intriguingly, we found that recurrent networks (i.e. LSTMs) can perfectly generalize given a particular input format, which roughly matches the format of the standard algorithm that is taught in schools. We attribute this to the fact that in this format the symmetries of the problem (namely, the same carry operation applied at each place) are perfectly aligned with the symmetry built into the network (weight sharing across time), without being complicated by the need to cache intermediate values. Bolstering this, we found that the same recurrent network breaks when it must instead wait until the end of the sequence to output its answer. However, this can be ameliorated by allowing the network to make use of an external memory module. We are thus led to posit that the combination of recurrence (that is, symmetry through time) and external memory as a promising path forward for building models capable of systematic generalization using psychologically plausible components. We saw in the easiest interleaved condition that a simple recurrent architecture is sufficient to obtain length generalization, and in the more difficult partial interleaved con-

dition, such an architecture in tandem with a simple form of external memory is sufficient to obtain in-distribution generalization (but not length generalization). We take this as an encouraging sign that these two ingredients in some form can lead to strong generalization on the addition task, and in future work we plan to explore different forms of external memory mechanisms with a view towards generalizing in both the partial interleaved and most-difficult blocked conditions, and achieving the capability for full out of distribution generalization.

Interestingly, we also did not find an advantage for the Relational Bottleneck in this setting (as implemented in the Abstractor architecture). One possible reason for this is that the inputs to the network are already represented in a somewhat abstract fashion through the structured base representation. Another may be that this task requires generalization along a different axis than other tasks in which the Relational Bottleneck has been shown to be useful, such as the Distribution of Three [Webb et al., 2021] [§] In that task, the model must generalize a relation to different tokens, but always within a fixed-size sequence; whereas here the tokens are always the same (i.e. the numbers between 0 and d) but what changes instead is the number of times the basic operation must be applied. It is an interesting question to consider how the Relational Bottleneck may be modified to more effectively encourage this form of generalization as well.

We have also generalized the base addition task via the introduction of non-standard carry tables, and performed some initial explorations of the effect of standard vs. non-standard carry tables on the performance of the neural networks. We found, not surpris-

[§]Note that [Altabaa et al., 2024] did find an advantage of the Abstractor in a mathematical problem solving dataset. However, the problems in that dataset were stated in natural language, meaning it is possible for the model to learn a more abstracted representation of the inputs; whereas in our case, the inputs are already represented in a maximally abstracted form as a list of digits.

ingly, that networks robustly tend to perform better on the standard carry operation compared with addition operations defined by nonstandard carry tables. Due to the richness and intricate structure of these alternative carry rules, we believe this to be a very promising space for evaluating the ability of neural networks to learn systematic or recursive rules (i.e., symmetry functions), and plan to more systematically evaluate models in this space in future work.

The introduction of these alternative carry tables, and the relative performance advantage on the standard table also introduces an intriguing question of whether there is a canonical distinguishing feature of our standard carry table? Identifying such a criterion could shed light on the abstract characteristics of problems that make them easier or harder for people to learn. This is something we are currently investigating in collaboration with Cutter Dawes and Mark McConnell.

Finally, a fascinating future direction would be to directly compare the efficacy of the learning strategies in our models with developmental data on how children learn to count and perform arithmetic. For example it has been suggested [Butterworth, 2005] that children first learn to count to small numbers (up to about 10), and then use this as the basis of arithmetic. Whereas, in our models, we instead trained directly on arithmetic and did not train the model to count as a preliminary step. It would thus be interesting to investigate where there are benefits from using a developmentally-inspired curriculum (e.g., pretraining on counting). Conversely, the modeling work points to subtler questions about details of human learning, that could be addressed both in empirical and further modeling work. For example, whereas children do learn to count up to some range before they learn arithmetic, it is not clear that at that point they appreciate the recursive nature of count-

ing (and can fully extrapolate the procedure) before they learn arithmetic, or whether that latter is necessary for that to occur.

7

Conclusion and future directions

In this work, we have examined how inductive learning biases grounded in psychology can be implemented in artificial systems such as neural networks in order to improve their capacity for systematic generalization and learning of symmetry functions. We have explored this from a combinatorially rich perspective, using combinations of multiple inductive biases (Maximum Entropy, Symmetry, Relational Bottleneck) across multiple domains

(analogical reasoning, function learning, and arithmetic).

In Chapters 2 and 3, we considered versions of the Relational Bottleneck and Maximum Entropy principles, respectively, instantiated in the context of mathematical and statistical (i.e., not neural-network based) models. This allowed us to understand the effect of these inductive biases in precise and controlled settings, however the approach is somewhat limited by the reliance on hand-coded features (e.g., graph-based problem representations) and lack of end-to-end learning.

In Chapters 4 and 5 we considered how the Maximum Entropy and Relational Bottleneck principles, along with the principle of Symmetry, can be successfully implemented in modern neural network architectures, trained end-to-end, as well as the extent to which their performance fits empirical human data. In particular, we saw how the technique of *contrastive learning*, which we showed can be interpreted as a combination of Maximum Entropy and Symmetry, naturally yields a preference for compositional function extrapolations as an emergent property. Notably, this points to a middle ground between the longstanding debate between connectionist and symbolist theories of the mind, and in particular the famous critique raised by Fodor and Pylytyn Fodor and Pylyshyn [1988] regarding the alleged inability of neural networks to exhibit systematic compositionality. Our result demonstrates that such a preference can in fact emerge in a neural network through unsupervised statistical learning, provided that the network is endowed with appropriate inductive learning biases. This points towards a view that such abilities may arise as emergent property of learning, together with an appropriate collection of general-purpose inductive biases, rather than something that must be explicitly built-in as suggested by, e.g. [Lake et al., 2016, Marcus, 2003]

Finally, in Chapter 6, we continued to explore the effect of these inductive biases on the ability of neural networks to exhibit symbolic reasoning, in the context of an integer base addition task. Here we found recurrence (that is, a form of symmetry with respect to time) to be an especially useful design choice for this task. We also showed, through a Group-theoretic analysis, how this task can be naturally generalized through *non-standard carry tables*, which can provide a rich test bed for the systematic generalization abilities of neural networks.

There are multiple interesting and promising avenues for future work. While we have hypothesized some specific ingredients (recurrence, external memory) that are likely to be useful for learning a systematic procedure for base counting, we were not able as yet to construct a model that fully and systematically generalizes on the addition task in all conditions. Doing so remains the most obvious and pressing avenue for future work. Succeeding at this would also open up many exciting follow ups. For example, if such a model can learn a symmetry function in the setting of addition, can it re-purpose this same operation in the context of a different task with similar structure, for example, solving arithmetic word problems? Further, the assumptions of such a model could be tested empirically. A natural way to do this would be to replace the memory component of the model with one that is closer to the human memory process, such as the Temporal Context [Howard and Kahana, 2002] rather than assume that the memory operations are learned through optimization on the specific addition task, as we have done for simplicity.

Another potential avenue would be to give a more formal account of the benefits of the inductive biases we have considered. Recent advances in theoretical machine learning may provide a roadmap for this goal. On the one hand, it has been argued in works such as Saxe

et al. [2013], Jacot et al. [2020] that many important properties of neural network learning dynamics can be recapitulated in well-chosen "model systems" such as linear networks or kernel machines (i.e., linear models in a featurized space). Thus, the reduction to an appropriate high-dimensional linear model may already be sufficient to give insight. On the other hand, a recent work Segert [2024] has systematically characterized the effect of a wide family of inductive biases in the context of regularized linear regression; it is possible that the results there could be extended to also rigorously and theoretically analyze the effects of the inductive biases we have considered here.

A

Proofs Related to the Maximum Entropy Principle

Proposition 4. *The optimization problem defined by 1.1 is strictly concave, meaning that the objective function is strictly concave, and the domain of the optimization is a convex set.*

Proof. The discrete and differential entropy functions are both known to be strictly con-

cave (e.g.[Boyd and Vandenberghe, 2004]). The constraints $E_{X\sim\mu} f_k(X) = y_k$ are linear with respect to the distribution μ. Since the set of probability distributions itself is convex (geometrically a simplex), the domain of the optimization is thus convex, since it is the intersection of a convex with with a linear subspace. Thus the problem amounts to maximizing a strictly concave function on a convex set, i.e. it is strictly concave. □

Corollary 1. *There is at most one solution to the optimization problem in 1.1. In other words, the Maximum Entropy distribution is unique if it exists.*

Proof. It is a very standard result in convex analysis that a strictly concave optimization problem has at most one solution, see e.g. Boyd and Vandenberghe [2004]. For convenience, we restate the argument here. Suppose we had two distinct maximizers μ^1_{ME} and μ^2_{ME}. Consider the average

$$\overline{\mu} := \frac{1}{2}(\mu^1_{ME} + \mu^2_{ME}) \tag{A.1}$$

.Since $\mu^1_{ME} \neq \mu^2_{ME}$, it follows that $\overline{mu} \neq \mu^1_{ME}, \mu^2_{ME}$. By strict concavity,

$$H(\overline{\mu}) > \frac{1}{2}H(\mu^1_{ME}) + \frac{1}{2}H(\mu^2_{ME}) = H(\mu^1_{ME}) \tag{A.2}$$

Moreover, since the domain of the optimization convex, the average μ lies in the domain. However, this contradicts the maximality of μ^1_{ME} over this domain. Thus, there cannot be more than one maximizer. □

Thus we have answered one of the fundamental questions, namely that there can be at most one solution to the Maximum Entropy problem. But does a solution always exist in the first place? Unfortunately, the below counterexample shows that the answer can be no:

Example 3. *Consider probability distributions μ on the set of all integers $S = \mathbb{Z}$. We consider only one constraint defined by $f(x) = x$ and $y = 0$, i.e. $E_{X\sim\mu}x = 0$. In other words, our constraint amounts to considering only mean-zero distributions on $\mathbb{Z}$. Now we ask: subject to this constraint, is there a distribution that attains the maximal possible entropy?*

To see that the answer is "no", consider the family of distributions μ_n, where μ_n is uniformly distributed over the interval $[-n, n]$ and zero outside of the interval. An easy calculation implies that

$$H(\mu_n) = \sum_{i=-n}^{n} \frac{1}{2n+1} log(2n+1) = log(2n+1) \tag{A.3}$$

On the other hand, clearly each μ_n has zero mean. Since $\lim_{n\to\infty} H(\mu_n) = \infty$, this means that there are mean-zero distributions with arbitrarily large entropy, and thus there cannot be a maximizer.

Fortunately, though, we can guarantee that μ_{ME} exists is we are willing to restrict to finite domains. In practice, this is typically not a severe assumption, since on a computer we can only implement finite systems anyway.

Theorem 5. *Let S be a finite set defining the domain of μ. Then the maximum entropy distribution, as defined in 1.1 always exists.*

Proof. In this case, the set of all probability distributions on S $P(S)$ can be identified with the standard simplex in the finite-dimensional Euclidean space $\mathbb{R}^{|S|}$. Furthermore, the constraints $E_{\mu\sim X}f_k(X) = y_k$ can be written as $\sum_{s\in S}\mu(s)f_k(s) = y_k$, which shows that each one is a linear constraint on μ. Thus the set of allowable μ geometrically consists of the (compact) unit simplex, intersected with a finite set of hyperplanes- the resulting intersection is compact because it is the intersection of a compact set with finitely many closed sets. By

basic calculus (e.g. the Extreme Value Theorem), any function on a compact set attains a maximum. □

At this point, we have completely answered the original questions at least in the finite case, and for reasonable restrictions. Namely, we have shown that if the sample space S is finite, then

1. μ_{ME} always exists, and
2. μ_{ME} is always unique.

So far, this discussion has been purely theoretical: we have not considered how to actually compute μ_{ME}. We will discuss this in the next section.

A.0.1 Computation with Lagrange multipliers

We first prove a standard theorem which is often quite useful for computing μ_{ME} in practice:

Theorem 6. *If μ_{ME} exists, then it is given in the form*

$$\mu_{ME}(x) \propto e^{\lambda_i \sum_{k=1}^{K} f_k(x)}, \textit{discretecase}$$

$$d\mu_{ME}/dx \propto e^{\lambda_i \sum_{k=1}^{K} f_k(x)}, \textit{continuouscase}$$

where λ_i are real numbers.

Proof. We first consider the discrete case. By standard convex analysis, the optimization 1.1

may be expressed in the Lagrange form by solving the following unconstrained problem:

$$argmax_{\mu \in P(S)} H(\mu) + \sum_{k=1}^{K} \lambda_k E_{X \sim \mu} f_k(X) + \nu \sum_{s \in S} \mu(s) \tag{A.4}$$

for real parameters $\lambda_1, \ldots, \lambda_k, \nu$. We have written the condition for the total probability to sum to 1 as another constraint, and introduced its own Lagrange multiplier ν.

Writing this out in terms of the components, is

$$argmax_{\mu \in P(S)} - \sum_{s \in S} \mu(s) \log \mu(s) + \sum_{s \in S} \sum_{k=1}^{K} \lambda_k f_k(s) \mu(s) + \nu \sum_{s \in S} \mu(s) \tag{A.5}$$

By differentiating with respect to $\mu(s)$ and setting equal to zero, we see that the first-order condition for an extremal point is

$$-1 - \log \mu_{ME}(s) + \sum_{k=1}^{K} \lambda_k f_k(s) + \nu \tag{A.6}$$

Rearranging,

$$\mu_{ME}(s) = e^{-1+\nu} e^{\sum_{k=1}^{K} \lambda_k f_k(s)}$$

The parameters ν and λ_k, in turn, must be chosen to satisfy the original constraints $\sum_{s \in S} \mu_{ME} s = 1$ and $\sum_{s \in S} f_k(s) \mu_{ME}(s) = y_k$.

For the continuous case, the argument is analogous. The difference is that rather than taking the standard derivative with respect to the components $\mu(s)$, we instead take the "functional derivative" with respect to the values $d\mu/ds$ of the density, which can be rigorously justified using the calculus of variations. □

These examples show the power of Theorem 6.However, it can also be turned on its head to illustrate a potential weakness of the Max Ent framework:

Corollary 2. *(of Theorem 6) Let μ be any distribution with finite entropy. Then μ is the Maximum Entropy distribution with constraint $f(x) = -log\mu(x), y = H(\mu)$.*

This corollary can be interpreted in two ways. On the one hand, there is the "glass half empty" interpretation, according to which the corollary implies that MaxEnt is too flexible to be useful-after all, if any distribution whatsoever can expressed as the solution to a Maximum Entropy problem, then how can there be any significance to the fact that a particular distribution is the solution to a particular Maximum Entropy problem?

Needless to say, we do not accept this interpretation. For one thing, the argument that MaxEnt is too flexible is itself likely too flexible. Consider that any distribution can be obtained as a Bayesian posterior for an appropriate choice of prior and likelihood- does this imply that Bayesian inference is too flexible to be useful? On the other hand, we do take Corollary 2 as an important instantiation of the old adage "garbage in,garbage out". That is, we should make sure that the inputs to the maxEnt problem (namely, the forms of the constraints) are meaningful constructs, just like when performing Bayesian inference we should make sure that we are using meaningful and appropriate prior distributions. Indeed, since means and covariances are quantities that are generally interesting in themselves, the MaxEnt distribution using these quantities as constrains is meaningful (in fact, we will later treat this example in more detail). By contrast, the MaxEnt distribution corresponding to an arbitrary constraint function like $f(x) = x^{3\sin(x)+log(x+1)^2}$ would likely not be interesting except as a mathematical curiosity.

We now show that this Theorem allows us to compute μ_{ME} in some interesting cases.

Example 4. *Let us return to the roulette example from the opening of Chapter 1. By direct application of Theorem 6, we see that the estimated probability of each bin is*

$$p_i = \frac{e^{\lambda i}}{\sum_{j=1}^{50} e^{\lambda j}} \tag{A.7}$$

for some $\lambda \in \mathbb{R}$. The value of λ in turn is determined by requiring that

$$\sum_{i=1}^{50} i \frac{e^{\lambda i}}{\sum_{j=1}^{50} e^{\lambda j}} = 30 \tag{A.8}$$

While this is not quite a "closed form" solution, we have at least reduced it from the original optimization as in Equation 1.1 over the 50 unknown variables to p_i to a root-finding problem in a single variable λ, as in Equation A.8. Many standard techniques exist for solving equations such as A.8, such as Newton's Method.

One of the more well-known examples of the Maximum Entropy principle in practice is to justify the usage of the Gaussian distribution. We consider this in our next example.

Example 5. *Let $S = \mathbb{R}$ and let $f(x) = x^2$. By Theorem 6, the Maximum Entropy distribution (if it exists) is given by*

$$d\mu_{ME}/dx \propto e^{\lambda x^2} \tag{A.9}$$

Evidently, λ must be negative, for otherwise $\int d\mu_{ME}/dx$ would diverge. We recognize this as the density for a mean-zero normal distribution with variance σ^2, where $\sigma := \sqrt{-2\lambda}$. Thus in order to satisfy the original constraint $\mathbb{E}_{X\sim\mu} X^2 = y$, we should choose the Lagrange multiplier to be $\lambda = -y/2$.

What if we also add a mean constraint, say $f_1(x) = x$ and $f_2(x) = x^2$? We could just

apply Theorem 6 again and crank through the resulting algebra to see that we get a normal distribution again, except shifted to have mean μ_1. But we can also see this without doing any further calculations. On the one hand, for the case of just one constraint $f(x) = x^2$, we saw that the resulting Maximum Entropy distribution was a centered (i.e. mean-zero) Gaussian. Therefore, this distribution must remain the Maximum Entropy distribution if we impose the additional constraint $E_{X\sim\mu}X = 0$. On the other hand, since the differential entropy function is translationally invariant (meaning $H(X) = H(X + c)$ for any random variable X and constant c), it follows that the Maximum Entropy distribution with values $y_1 = \mu$ and $y_2 = \sigma^2$ can be obtained by taking the Maximum Entropy distribution with $y_1 = 0$ and $y_2 = \sigma^2$ and translating it by μ.

As a final extension to this example, which will also be useful later, we will consider the multivariate case. To wit, we now set $S = \mathbb{R}^2$. For simplicity we consider only two dimensions, although the d-dimensional case is only harder in a notational sense. We consider now three constraints of the form $f_1(x, y) = x^2, f_2(x, y) = y^2$ and $f_3(x, y) = xy$. These correspond to observing the mean of each component, as well as their covariance. As in the 1d case, we will not impose a mean constraint, but we will see that the Maximum Entropy solution has mean zero anyway. Applying Theorem 6 again, we obtain the form

$$\mu_{ME}(x, y) \propto e^{\lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 xy} \tag{A.10}$$

$$= e^{-\frac{1}{2}(x,y)C^{-1}(x,y)^t} \tag{A.11}$$

where

$$C = \begin{pmatrix} -2\lambda_1 & -\lambda_3 \\ -\lambda_3 & -2\lambda_2 \end{pmatrix}^{-1} \tag{A.12}$$

$$\tag{A.13}$$

We recognize the density as a 2d Gaussian with mean zero and covariance matrix C. In order to satisfy the constraints, we thus need

$$\begin{pmatrix} -2\lambda_1 & -\lambda_3 \\ -\lambda_3 & -2\lambda_2 \end{pmatrix} = \begin{pmatrix} y_1 & y_3 \\ y_3 & y_2 \end{pmatrix}^{-1} \tag{A.14}$$

$$= \frac{1}{y_1 y_2 - y_3^2} \begin{pmatrix} y_2 & -y_3 \\ -y_3 & y_1 \end{pmatrix} \tag{A.15}$$

Thus, we see that the 2d Gaussian is the Maximum Entropy distribution given constraints on the variances and covariances. The d-dimensional case is analogous.

A.0.2 Proof of Theorem 1

Proof. For any values $\lambda_i \in \mathbb{R}$, the log-likelihood of the corresponding exponential family distribution is given by

$$LLH(\lambda_1, \ldots, \lambda_K) = -nlogZ(\lambda_1, \ldots, \lambda_k) + \sum_{i=1}^{n} \sum_{k=1}^{K} \lambda_k f_k(x_i) \tag{A.16}$$

Setting the gradient equal to zero, we see that at the maximum we should have

$$\frac{\partial}{\partial \lambda_k} log Z(\lambda_1, \ldots, \lambda_K) = \frac{1}{n} \sum_{i=1}^{n} f_k(x_i) \tag{A.17}$$

for each k. On the other hand,

$$\begin{aligned} \frac{\partial}{\partial \lambda_k} Z(\lambda_1, \ldots, \lambda_K) &= \frac{\partial}{\partial \lambda_k} \sum_{x \in S} e^{\sum_{k=1}^{K} f_k(x)\lambda_k} && \text{(A.18)} \\ &= \sum_{x \in S} f_k(x) e^{\sum_{k=1}^{K} f_k(x)\lambda_k} && \text{(A.19)} \\ & && \text{(A.20)} \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{\partial}{\partial \lambda_k} log Z(\lambda_1, \ldots, \lambda_K) &= Z^{-1} \frac{\partial}{\partial \lambda_k} Z(\lambda_1, \ldots, \lambda_K) && \text{(A.21)} \\ &= \frac{\sum_{x \in S} f_k(x) e^{\sum_{k=1}^{K} f_k(x)\lambda_k}}{\sum_{x \in S} e^{\sum_{k=1}^{K} f_k(x)\lambda_k}} && \text{(A.22)} \\ &= \mathbb{E}_{X \sim \mu_{\lambda_1, \ldots, \lambda_K}} f_k(X) && \text{(A.23)} \end{aligned}$$

where $\mu_{\lambda_1, \ldots, \lambda_K}$ denotes the distribution from the exponential framily $ME(f_1, \ldots, f_K)$ with the corresponding λ_i values. Putting together equations A.17 and A.21, we see that the condition for maximizing the likelihood is

$$\mathbb{E}_{X \sim \mu_{\lambda_1, \ldots, \lambda_K}} f_k(X) = \frac{1}{n} \sum_{i=1}^{n} f_k(x_i) \tag{A.24}$$

for each k. By Theorem 6, this exactly coincides with μ_{ME}, provided again that we take $y_k := \frac{1}{n}\sum_{i=1}^{n} f_k(x_i)$. □

References

A. Aberdam, R. Litman, S. Tsiper, O. Anschel, R. Slossberg, S. Mazor, R. Manmatha, and P. Perona. Sequence-to-sequence contrastive learning for text recognition. *arXiv:2012.10873v1*, 2020.

T. D. Albert Gu. Mamba: Linear-time sequence modeling with selective state spaces. *arxiv.org:2312.00752*, 2023.

A. Altabaa, T. Webb, J. Cohen, and J. Lafferty. Altabaa, a. et al. (2024). abstractors and relational cross-attention: An inductive bias for explicit relational reasoning in transformers. *International Conference on Learning Representations*, 2024.

S. Amiriparian, M. Freitag, N. Cummins, and B. Schuller. Sequence to sequence autoencoders for unsupervised representation learning from audio. In *DCASE Workshop*, 2017.

A. AN Kolmogorov. Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1965.

P. Anderson. More is different. *Science*, 1972.

G. E. Barton, R. C. Berwick, and E. S. Ristad. *Computational Complexity and Natural Language*. MIT Press, 1987.

P. Battaglia, J. Hamrick, . others, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint*, 1806.01261, 2018.

S. Becker and G. Hinton. Self-organizing neural network that discovers surfaces in random dot stereograms. *Nature*, 1992.

A. J. Bell and T. I. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 1995.

R. Bhui and S. Gershman. Decision by sampling implements efficient coding of psychoeconomic functions. *Psychological Review*, pages 985–1001, 2018.

E. Bienenstock, L. Cooper, and P. Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. 1982.

M. Bornstein and J. Stiles-Davis. Discrimination and memory for symmetry in young children. *Developmental Psychology*, 1984.

L. Bott and E. Heit. Nonmonotonic extrapolation in function learning. *Journal of Experimental Psychology:Learning, Memory, and Cognition*, 2004.

M. M. Botvinick. Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology*, pages 956–962, 2012.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017a.

M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017b.

J. Burg. Maximum entropy spectral analysis. In *Proceedings of the 37th meeting of the society of exploration geophysicists.*, 1967.

B. Butterworth. The development of arithmetical abilities. *Journal of Child Psychology and Psychiatry*, 2005.

L. Chen. The topological approach to perceptual organization. *Visual Cognition*, 2005.

R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in Neural Information Processing Systems*, 2018.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

F. Chollet. On the measure of intelligence. *arXiv:1911.01547*, 2019.

L. Ciccione and S. Dehaene. Can humans perform mental regression on a graph? accuracy and bias in the perception of scatterplots. *Cognitive Psychology*, 2021.

T. Cohen and M. Welling. Group equivariant convolutional networks. *International conference on machine Learning*.

T. Cover and J. Thomas. *Elements of information theory*. John Wiley and Sons, 1991.

I. Dasgupta, E. Schulz, J. Tenenbaum, and S. Gershman. A theory of learning to infer. *Psychological Review*, 127:412–441, 2020.

P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 1993.

E. L. DeLosh, J. R. Busemeyer, and M. A. McDaniel. Extrapolation: The sine qua non for abstraction in function learning. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 1997.

Z. Dulberg, T. Webb, and J. Cohen. Modelling the development of counting with memory-augmented neural networks. In *Proceedings of the Cognitive Science Society*, 2021.

D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174, 2013a.

D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, 2013b.

K. Ellis, C. Wong, M. Nye, M. Sable-Meyer, L. Cary, L. Morales, L. Hewitt, A. Solar-Lezama, and J. B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv:2006.08381*, 2020.

M. Fiori, P. Sprechmann, J. Vogelstein, P. Muse, and G. Sapiro. Robust multimodal graph matching:sparse coding meets graph matching. *Advances in Neural Information Processing Systems*, 2013.

R. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society*, 1922.

J. Fodor. *The language of thought*. Harvard University Press, 1975.

J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 1988.

S. Frankland and J. Cohen. Determinantal point processes for memory and structured inference. *Proceedings of the Cognitive Science Society*, 2020.

S. Frankland, T. Webb, and J. Cohen. No coincidence, george: Capacity-limits as the curse of compositionality. *psyarxiv*, 2021a.

S. Frankland, T. Webb, and J. Cohen. No coincidence, george: Capacity-limits as the curse of compositionality. *PsyArxiv:https://doi.org/10.31234/osf.io/cjuxb*, 2021b.

J. Freyd and B. Tversky. Force of symmetry in form perception. *American Journal of Psychology*, 1984.

K. Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 2010.

S. Garg, D. Tsipras, P. Liang, and G. Valiant. What can transformers learn in-context? a case study of simple function classes. *arXiv preprint*, 2208.01066, 2022.

M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami. Conditional neural processes. In *International conference on machine learning*, 2018a.

M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. A. Eslami, and Y. W. Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

R. Gelpi, N. Saxena, G. Lifchits, D. Buchsbaum, and C. Lucas. Sampling heuristics for active function learning. *PsyArXiv*, 2021.

D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7:155–170, 1983.

D. Gentner. Why we're so smart. *Language in mind: Advances in the study of language and thought*, 2003.

J. E. Gerken, J. Aronsson, O. Carlsson, H. Linander, F. Ohlsson, C. Petersson, and D. Persson. Geometric deep learning and equivariant neural networks. *Artificial Intelligence Review*.

S. J. Gershman and Y. Niv. Learning latent structure: carving nature at its joints. *Current Opinion in Neurobiology*, 2010.

M. W. Gondal, S. Joshi, N. Rahaman, S. Bauer, M. Wuthrich, and B. Scholkopf. Function contrastive learning of transferable meta-representations. In *International Conference on Machine Learning*, 2021.

N. Goodman, J. Tenenbaum, J. Feldman, and T. L. Griffiths. A rational analysis of rule-based concept learning. *Cognitive Science*, 2008.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint*, 1410.5401, 2014.

P. Grünwald and A. Dawid. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *Annals of Statistics*, 2004.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv:2111.00396*, 2021.

M. Hamermesh. *Group Theory and Its Application to Physical Problems*. Dover, 1989.

B. R. HB Barlow. The versatility and absolute efficiency of detecting mirror symmetry in random dot displays. *Vision research*, 1979.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Computer Visision and Pattern Recognition*, 2020.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on learning Representations*, 2:891–921, 2017.

G. E. Hinton. Deterministic boltzmann learning performs steepest descent in weight-space. *Neural Computation*, 1989.

G. E. Hinton and R. R. Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In *Conference on Neural Information Processing Systems*, 2007.

K. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 1989.

M. W. Howard and M. J. Kahana. A distributed representation of temporal context. *Journal of Mathematical Psychology*, 2002.

A. Hyvarinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Advances in Neural Information Processing Systems*, 2016.

N. Intrator and L. Cooper. Objective function formulation of the bcm theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 1992.

D. Isaksen. A cohomological viewpoint on elementary school arithmetic. *American Mathematical Monthly*, 2002.

A. Jacot, B. Simsek, F. Spadaro, C. Hongler, and F. Gabriel. Kernel alignment risk estimator: risk prediction from training data. *Advances in Neural Information Processing Systems*, 2020.

E. Jaynes. Information theory and statistical mechanics. *Physical Review*, pages 620–630, 1957.

Jean-Yves, Franceschi, A. Dieuleveut, and M. Jaggi. Unsupervised scalable representation learning for multivariate time series. In *Conference on Neural Information Processing Systems*, 2019.

S. Jelassi, D. Brandfonbrener, S. M. Kakade, and E. Malach. Repeat after me: Transformers are better than state space models at copying transformers are better than state space models at copying. *https://arxiv.org/pdf/2402.01032.pdf*, 2024.

M. Kalish. Learning and extrapolating a periodic function. *Mem Cognit*, 2013.

C. Kemp and J. B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 2008.

G. Kerg, S. Mittal, D. Rolnick, Y. Bengio, B. Richards, and G. Lajoie. On neural architecture inductive biases for relational tasks. *arXiv Preprint*, 2206.05056, 2022.

H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. *arXiv preprint*, 1901.05761, 2019.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2014.

K. Koffka. *Principles of Gestalt psychology*. 1935.

K. Krishnamurthy, T. Can, and D. J. Schwab. Theory of gating in recurrent neural networks. *Physical Review X*, 2022.

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 2012.

S. Kumar, I. Dasgupta, J. Cohen, N. Daw, and T. Griffiths. Meta-learning of structured task distributions in humans and machines. 2020.

P. Kwantes and A. Neal. Why people underestimate y when extrapolating in linear functions. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 2006a.

P. Kwantes and A. Neal. Why people underestimate y when extrapolating in linear functions. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 2006b.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 2016.

N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos. Teaching arithmetic to small transformers. *International Conference on Learning Representations*, 2024.

P. Leon-Villagra and C. G. Lucas. Generalizing functions in sparse domains. In *Proceedings of the Cognitive Science Society*, 2019.

P. León-Villagrá, I. Preda, and C. Lucas. Data availability and function extrapolation. In *Proceedings of the Cognitive Science Society*, 2018.

T. Li, L. Fan, Y. Yuan, H. He, Y. Tian, and D. Katabi. Information-preserving contrastive learning for self-supervised representations. *arXiv:2012.09962v1*, 2020.

Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects. *arXiv:1904.12787.*, 2019.

F. Lieder and T. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43, 2020.

R. Linsker. Self-organization in a perceptual network. *IEEE Computer*, 1988.

D. Little and R. Shiffrin. Simplicity bias in the estimation of causal functions. In *CogSci*, pages 1157–1162, 2016.

E. M. Loiola, N. M. M. de Abreua, P. O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176, 2007.

C. G. Lucas, T. L. Griffiths, J. J. Williams, and M. L. Kalish. A rational model of function learning. *Psychonomic Bulletin and Review*, 2015a.

C. G. Lucas, T. L. Griffiths, J. J. Williams, and M. L. Kalish. A rational model of function learning. *Psychonomic Bulletin and Review*, 2015b.

X. Lyu, M. Hueser, S. L. Hyland, G. Zerveas, and G. Rätsch. Improving clinical predictions through unsupervised time series representation learning. *arXiv:1812.00490*, 2018.

V. Lyzinski, D. E. Fishkind, M. Fiori, J. T. Vogelstein, C. E. Priebe, and G. Sapiro. Graph matching: Relax at your own risk. *IEEE transactions on pattern analysis and machine intelligence*, 38, 2015.

Q. Ma, J. Zheng, S. Li, and G. W. Cottrell. Learning representations for time series clustering. In *Advances in Neural Information Processing Systems*, 2019.

J. N. Macgregor. Human performance on the traveling salesman and related problems: A review. *Journal of Problem Solving*, 2011.

P. Malhotra, L. V. Vishnu TV, P. Agarwal, and G. Shroff. Timenet:pre-trained deep recurrent neural network for time series classification. *arXiv:1706.08838*, 2017.

G. Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. 2003.

G. Marcus, S. Vijayan, S. B. Rao, and P. Vishton. Rule learning in 7-month-old infants. *Science*, 283, 1999.

E. Markman and G. Wachtel. Children's use of mutual exclusivity to constrain the meanings of words. *Cognitive psychology*, 1988.

D. Marr and T. Poggio. From understanding computation to understanding neural circuitry. *Artificial Intelligence Laboratory. A.I. Memo. Massachusetts Institute of Technology*, 1976.

J. McClelland and T. Rogers. The parallel distributed processing approach to semantic cognition. *Nature Reviews Neuroscience*, 2003.

M. McDaniel and J. Busemeyer. The conceptual basis of function learning and extrapolation: Comparison of rule-based and associative-based models. *Psychonomic Bulletin and Review*, 2005a.

M. A. McDaniel and J. R. Busemeyer. The conceptual basis of function learning and extrapolation: Comparison of rule-based and associative-based models. *Psychonomic Bulletin and Review*, 2005b.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013b.

C. Mongoven and C.-C. Carbon. Acoustic gestalt: On the perceptibility of melodic symmetry. *Musicae Scientiae*, 2016.

J. Myung. Maximum entropy interpretation of decision bound and context models of categorization. *Journal of Mathematical Psychology*, 38:335–365, 1994.

J. Myung and R. Shepard. Maximum entropy inference and stimulus generalization. *Journal of Mathematical Psychology*, 40:342–347, 1996.

M. Neumann, S. Huang, D. E. Marthaler, and K. Kersting. pygps – a python library for gaussian process regression and classification. *Journal of Machine Learning Research*, 2015.

T. Pan, Y. Song, T. Yang, W. Jiang, and W. Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples. In *Computer Vision and Pattern Recognition*, 2021.

D. Peterson and M. Berryhill. The gestalt principle of similarity benefits visual working memory. *Psychonomic bulletin review*, 2013.

S. Piantadosi, J. Tenenbaum, and N. Goodman. The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, 2016.

M. H. Pornstein and S. J. Krinsky. Perception of symmetry in infancy: The salience of vertical symmetry and the perception of pattern wholes. *Journal of Experimental Child Psychology*, 2985.

A. Pritzel, B. Uria, S. Srinivasan, A. Puigdomenech, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural episodic control. In *International Conference on Machine Learning*, 2017.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. In *OpenAI Blog*, 2019.

C. Rasmussen and C. Williams. *Gaussian processes for machine learning*. MIT Press, 2006a.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006b.

M. Reed and B. Simon. *Functional analysis*. Academic Press, 1980.

J. Rissanen. Modeling by shortest data description. *Automatica*, 1978.

C. Rossi-Arnaud, L. Pieroni, and A. Baddeley. Symmetry and binding in visuo-spatial working memory. *Neuroscience*, 2006.

P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics*, 20:53–65, 1987.

D. Rumelhart and J. McClelland. Parallel distributed processing: Explorations in the microstructure of cognition. 1986a.

D. E. Rumelhart and J. L. McClelland. *Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations*. MIT Press, 1986b.

M. Sablé-Meyer, J. Fagot, S. Caparos, T. van Kerkoerle, M. Amalric, and S. Dehaene. Sensitivity to geometric shape regularity in humans and baboons: A putative signature of human singularity. *Proceedings of the National Academy of Sciences*, 2021.

A. Saxe, J. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, 2013.

E. Schulz, J. Tenenbaum, D. Reshef, M. Speekenbrink, and S. Gershman. Assessing the perceived predictability of functions. In *CogSci*, 2015.

E. Schulz, J. B. Tenenbaum, D. Duvenaud, M. Speekenbrink, and S. J. Gershman. Compositional inductive biases in function learning. *Cognitive Psychology*, 99:44–79, 2017.

S. Segert. Flat minima in linear estimation and an extended gauss markov theorem. *International Conference on Learning Representations*, 2024.

S. Segert and J. Cohen. Maximum entropy function learning. *Proceedings of the Cognitive Science Society*, 2022a.

S. Segert and J. Cohen. A self-supervised framework for function learning and extrapolation. *Transactions in Machine Learning Research*, 2022b.

S. Segert and J. Cohen. Beyond transformers for function learning. *Proceedings of the Cognitive Science Society*, 2023.

S. Segert, M. Tepper, J. Turek, and J. Cohen. Relaxed graph matching for analogical reasoning. *International Conference on Learning Representations, Workshop on Bridging AI and Cognitive Science*, 2020.

R. Shen, S. Bubeck, R. Eldan, Y. T. Lee, Y. Li, and Y. Zhang. Positional description matters for transformers arithmetic. *https://arxiv.org/abs/2311.14737*, 2023.

R. Solomonoff. A formal theory of inductive inference part i. *Information and Control*, 1964.

E. S. Spelke and K. K. D. Kinzler. Core knowledge. *Developmental Science*, 2007.

K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 20, 2017.

S. Sun, G. Zhang, C. Wang, W. Zeng, J. Li, and R. Grosse. Differentiable compositional kernel learning for gaussian processes. *International Conference on Machine Learning*, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics,structure, and abstraction. *Science*, 2011.

S. Tonekaboni, D. Eytan, and A. Goldengerg. Unsupervised representation learning for time series with temporal neighborhood coding. In *International Conference on Learning Representations*, 2021.

D. Tran, R. Ranganath, and D. M. Blei. The variational gaussian process. In *International Conference on Learning Representations*, 2016.

M. Vaishnav and T. Serre. Gamr: A guided attention model for (visual) reasoning. In *International Conference on Learning Representations*, 2023.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 1984.

A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. *arXiv preprint*, 1710.10903, 2017.

P. L. Villagra, I. Preda, and C. G. Lucas. Data availability and function extrapolation. In *Proceedings of the Cognitive Science Society*, 2018.

J. T. Vogelstein, J. M. Conroy, V. Lyzinski, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe. Fast approximate quadratic programming for large (brain) graph matching. *arxiv:1112.5507v5*, 2014.

J. von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov. Transformers learn in-context by gradient descent. *arXiv preprint*, 2212.07677, 2022.

J. Wagemans, J. Elder, M. Kubovy, S. Palmer, M. Peterson, M. Singh, and R. V. der Heydt. A century of gestalt psychology in visual perception i: Perceptual grouping and figure–ground organization. *Psychological bulletin*, 2012.

T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, 2020.

T. Webb, I. Sinha, and J. Cohen. Emergent symbols through binding in episodic memory. In *international Conference in Learning Representations*, 2021.

T. W. Webb, Z. Dulberg, S. M. Frankland, A. A. Petrov, R. C. O'Reilly, and J. D. Cohen. Learning representations that support extrapolation. In *International Conference on Machine learning*, 2020.

T. W. Webb, S. M. Frankland, A. Altabaa, K. Krishnamurthy, D. Campbell, J. Russin, R. O'Reilly, J. Lafferty, and J. D. Cohen. The relational bottleneck as an inductive bias for efficient abstraction. *arXiv:2309.06629v2*, 2024.

J. Wei, W. Xuezhi, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 2022.

J. Whittington, T. Muller, S. Mark, G. Chen, C. Barry, N. Burgess, and T. Behrens. The tolman-eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 2020.

J. Whittington, J. Warren, and T. Behrens. Relating transformers to models and neural representations of the hippocampal formation. In *International Conference on Learning Representations*, 2022.

A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning*, 2013.

A. G. Wilson, C. Dann, C. G. Lucas, and E. P. Xing. The human kernel. In *Conference on Neural Information Processing Systems*, 2015a.

A. G. Wilson, C. Dann, C. G. Lucas, and E. P. Xing. The human kernel. In *Neural Information Processing Systems*, 2015b.

D. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 1996.

C. M. Wu, E. Schulz, and S. J. Gershman. Inference and search on graph structured spaces. *Computational Brain Behavior*, 4(2):125–147, 2021.

E. Yiu, J. Collins, and A. Gopnik. Three-dimensional object completion in humans and computational models. *Proceedings of the Cognitive Science Society*, 2022.

E. Zeeman. Topology of the brain. In *Mathematics and computer science in biology and medicine: Proceedings of the conference held by the Medical Research Council in Association with the Health Department*, 1965.

H. Zhou, A. Bradley, E. Littwin, N. Razin, O. Saremi, J. Susskind, S. Bengio, and P. Nakkiran. What algorithms can transformers learn? a study in length generalization. *arXiv:2310.16028*, 2023.