Relational Reasoning and Generalization Using Nonsymbolic Neural Networks

Atticus Geiger[1], Alexandra Carstensen[2], Michael C. Frank[2], and Christopher Potts[1]

[1] Department of Linguistics, Stanford University

[2] Department of Psychology, Stanford University

The notion of equality (identity) is simple and ubiquitous, making it a key case study for broader questions about the representations supporting abstract relational reasoning. Previous work suggested that neural networks were not suitable models of human relational reasoning because they could not represent mathematically identity, the most basic form of equality. We revisit this question. In our experiments, we assess out-of-sample generalization of equality using both arbitrary representations and representations that have been pretrained on separate tasks to imbue them with structure. We find neural networks are able to learn (a) basic equality (mathematical identity), (b) sequential equality problems (learning ABA-patterned sequences) with only positive training instances, and (c) a complex, hierarchical equality problem with only basic equality training instances ("zero-shot" generalization). In the two latter cases, our models perform tasks proposed in previous work to demarcate human-unique symbolic abilities. These results suggest that essential aspects of symbolic reasoning can emerge from data-driven, nonsymbolic learning processes.

Keywords: relational reasoning, generalization, neural networks

One of the key components of human intelligence is our ability to reason about abstract relations between stimuli. Many of the most unremarkable human activities—scheduling a meeting, following traffic signs, assembling furniture—require a fluency with abstraction and relational reasoning that is unmatched in nonhuman animals. An influential perspective on human uniqueness holds that relational concepts are critical to higher-order cognition (e.g., Gentner & Goldin-Meadow, 2003). By far the most common case study of abstract relations has been equality.[1] Equality is a valuable case study because it is simple and ubiquitous, but also completely abstract in the sense that it can be evaluated regardless of the identity of the stimuli being judged.

Human equality reasoning has been studied extensively across a host of systems and tasks, with wildly variant conclusions. In some studies, equality is very challenging to learn: only great apes with either extensive language experience or specialized training succeed in matching tasks in which a *same* pair, AA, must be matched to a novel same pair, BB (Thompson et al., 2001; Premack, 1983). Preschool children also struggle to learn these regularities in a seemingly similar task (Walker et al., 2016). In contrast, other studies suggest that equality is simple: bees are able to learn abstract identity relationships from only a small set of training trials

(Avarguès-Weber et al., 2011; Giurfa et al., 2001), and human infants can generalize identity patterns (Anderson et al., 2018) and succeed in relational matching tasks (Ferry et al., 2015). We take the central challenge of this literature to be characterizing the conditions that lead to success or failure in learning an abstract relation in a way that can be productively generalized to new stimuli (Carstensen & Frank, 2021).

The learning task in all of these cases can be described using the predicate *same* (or equivalently, =), which operates over two inputs and returns true if they are identical in some respect, else false. One perspective in the literature is that success in these learning tasks implies the presence of an equivalent symbolic description in the mind of the solver (Marcus et al., 1999; Premack, 1983). This view does not provide a lever to distinguish which of these tasks are trivial and which are difficult, however. Further, it can fall prey to circularity: because newborns show sensitivity to identity relations (Gervain et al., 2012), then it would follow from this argument that they must have symbolic representations. If this logic applies also to bees, then we presuppose symbolic representations universally and

---

[1] We use the term "equality" here, though different literatures have also used "identity."

Atticus Geiger https://orcid.org/0000-0002-9170-506X

Alexandra Carstensen https://orcid.org/0000-0003-0582-9152

Michael C. Frank https://orcid.org/0000-0002-7551-4378

Christopher Potts https://orcid.org/0000-0002-7978-6055

Correspondence concerning this article should be addressed to Atticus Geiger, Department of Linguistics, Stanford University, Stanford, CA 94305, United States. Email: atticusg@stanford.edu

have no account of the gradient difficulty of different tasks for different species.

An explanation of when same–different tasks are trivial and when they are difficult requires a theoretical framework beyond the symbolic/nonsymbolic distinction. To make quantitative predictions about task performance, such a framework should ideally be instantiated in a computational model that takes in training data and learns a solution that generalizes when assessed with stimuli analogous to those used in experimental assessments. Symbolic computational models (e.g., Frank & Tenenbaum, 2011) can be used to make contact with data about the breadth of generalizations that humans make. But such models require the existence of a symbolic equality predicate and hence again presuppose symbolic abilities in every case of success. Ideally, we would want a model that describes under what conditions *same* is easy and under what conditions it is hard or unlearnable—and how learning proceeds in hard cases. Here, we aim to lay the foundation for the development of such an account.

We are inspired by an emergent perspective in the animal learning literature that the representations underlying nonhuman animals' and human infants' successes in equality reasoning tasks are graded (Wasserman et al., 2017). This view acknowledges the increasing evidence that other species like pigeons (Cook & Wasserman, 2007), crows (Smirnova et al., 2015), and baboons (Fagot & Thompson, 2011) can make true, out-of-sample generalizations of *same* and *different* relations, but it also recognizes that the observed patterns of behavior do not show the hallmarks of all-or-none symbolic representations. Instead, performance is graded. Out-of-sample generalization is possible but the level of performance depends critically on the diversity of the training stimuli (e.g., Castro et al., 2010). Success requires hundreds, thousands, or even tens of thousands of training trials. And the outcome of learning is noisy and imperfect. These learning signatures appear to be a close match to the kind of learning exhibited by neural networks. Such networks are a flexible framework for arbitrary function learning which have enjoyed a huge resurgence of interest in recent years in the fields of artificial intelligence, neuroscience, and cognitive science (e.g., LeCun et al., 2015; Saxe et al., 2019).

In an influential rebuttal of the use of neural network models for capturing relational reasoning, Marcus et al. (1999) argued that a broad class of recurrent neural networks were unable to learn a general solution to sequential equality problems and Marcus (2001) argued that feed-forward neural networks are even unable to learn a general solution even to mathematical identity, which is arguably the simplest version of *same/different* reasoning conceptually. These claims were subsequently challenged by the presentation of evidence that some forms of neural networks are able to learn (at least aspects of) Marcus et al.'s equality tasks (Dienes et al., 1999; Elman, 1999; Negishi, 1999; Seidenberg & Elman, 1999a, 1999b) yet these examples were controversial. The resulting debate (reviewed in Alhama & Zuidema, 2019) revealed a striking lack of consensus on some of the ground rules regarding what sort of generalization would be required to show that the learned function was suitably abstract.

In the time since these debates, successful neural network models have been developed for tasks such as natural language inference (Bowman, Angeli, et al., 2015; Williams et al., 2018), question answering (Rajpurkar et al., 2016, 2018), and visual reasoning (Johnson et al., 2017). In many respects, all of these tasks are far more complex than equality-based tasks. In light of these findings, it

may be surprising that the debate over equality-based reasoning is unresolved (Alhama & Zuidema, 2019). Yet even recent work on equality-based reasoning tasks takes as its starting point the conclusion that neural networks are unable to succeed using standard architectures and general purpose learning algorithms (Alhama & Zuidema, 2018; Kopparti & Weyde, 2020; Weyde & Kopparti, 2018, 2019). Further, though tasks and contexts vary, work in both computer vision (Fleuret et al., 2011; Gülçehre & Bengio, 2016; Kim et al., 2018) and machine reasoning (Palm et al., 2018; Raposo et al., 2017; Santoro et al., 2017, 2018) has presupposed that relational reasoning generally—and sometimes equality-based reasoning specifically—is difficult or impossible in standard network architectures.

Modern deep learning models have been so successful that it seems odd that they would be completely unable to learn equality-based reasoning tasks. We suspect these claims remain in the literature in part because only a narrow range of network architectures and representations were explored in the earlier debate, and in part because the debate predated many important innovations in neural network design. Thus we revisit the debate here, using a broader range of architectures and representations and adopting stringent criteria for generalization. In particular we explore random and pretrained representations, which have facilitated many of the successes of modern artificial intelligence (Collobert et al., 2011; Devlin et al., 2019; Mikolov et al., 2013; Pennington et al., 2014; Peters et al., 2018). The use of pretrained representations to solve downstream tasks in particular is argued to be a hallmark of natural learning systems (Zador, 2019), has been an important feature of historical models from cognitive science (e.g., Landauer & Dumais, 1997; McRae & Hetherington, 1993), and is essential in the latest wave of state-of-the-art natural language processing models (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019; T. Brown et al., 2020).

In our current work, we model three cases of equality-based reasoning that have featured prominently in discussions of the role of symbols in relational reasoning (Figure 1): (a) learning to discriminate pairs of objects that exemplify the relation *same* or *different*, (b) learning sequences with repeated *same* elements (Marcus et al., 1999), and (c) learning to distinguish hierarchical *same* and *different*

Figure 1
Relational Reasoning Tasks
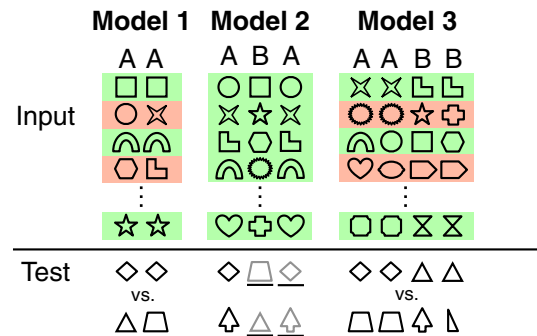
Note. Green and red mark positive and negative training examples, respectively. The sequential task (Model 2) uses only positive instances, and a model succeeds if, prompted with α, it produces a sequence βα, where β ≠ α. For the hierarchical task (Model 3), we show that a model trained on the basic task (Model 1) is effective with no additional training. See the online article for the color version of this figure.

relations in a context with pairs of pairs exemplifying these relations (Premack, 1983). Across these three models, we find strong support for their ability to learn equality relations. These results should serve to revise the conclusions of the earlier debate.

Marcus (2001) and Marcus et al. (1999) showed experimentally that certain neural networks could not generalize the equality relation to stimuli unseen in training. We agree with this claim (and support it with a direct mathematical argument). But we show that the claim is specific to networks that use particular kinds of representations, rather than neural networks in general. The representations that are problematic for this task are *featural* representations (a term that we define in more detail below): representations where either specific properties (e.g., shape, color) or entire items are represented as discrete dimensions in the network's inputs.

Marcus et al. (2019) concluded from their results that neural networks need to have primitive symbolic operators to solve equality-based relational reasoning tasks, which is a solution that has been pursued in recent machine learning research (Kopparti & Weyde, 2020; Weyde & Kopparti, 2018, 2019). On this point, we disagree. Our experiments show that networks without such primitives can solve a range of these tasks using the sort of random or pretrained representations that are now the norm throughout artificial intelligence research. Overall, these findings suggest that some essential aspects of symbolic reasoning can emerge from entirely data-driven, nonsymbolic learning processes.

Our work here makes three contributions. First, we resolve this longstanding debate by demonstrating neural networks are able to learn equality relations when provided with pretrained or random representations. Second, we modify the standard architecture of a recurrent neural network to allow it to learn the sequential equality task with no negative feedback. Negative evidence was dismissed as an unreasonably strong learning regime in the original debate over these issues (e.g., Marcus, 1999), and and we show that this learning regime is not necessary. Third, we show that a model pretrained on the simple equality task can achieve zero-shot generalization (generalization with no new training instances) to the hierarchical equality task, suggesting that pretraining might provide an account of how some organisms succeed on hard relational learning tasks. We believe these three contributions represent significant progress in our understanding of neural networks' ability to perform equality-based reasoning. Taken together, these contributions lay the groundwork for further nonsymbolic neural network models of relational reasoning and abstract thought more broadly.

In all our experiments, we define *same* and *different* in terms of mathematical identity. These are the terms of the original debate, and learning these unambiguous relations is likely a prerequisite for learning real-world equality tasks grounded in some perceptual domain, such as a vision network trained for facial recognition (Wang & Deng, 2021). Thus, following the protocols of Marcus (2001) and Marcus et al. (1999) as closely as possible, we assign every object a unique representation that is reliably retrieved by the model. We demonstrate that, starting from such preindividuation, general solutions to equality relations can be learned if the right representations are used.

General Methods: Designing Theoretical Models of Equality Learning

We begin by discussing two critical design considerations for our models: (a) the standards for generalization by which models should be evaluated and (b) the type of representations they should use. To summarize this discussion: we select generalization tasks with fully disjoint training and test vocabularies to provide the most stringent test of generalization. Next, we show analytically that featural representations, which represent either object identities or object properties as discrete representational dimensions, limit successful out-of-sample generalization. Thus, we adopt randomly initialized representations or random representations that have been pretrained for our subsequent models. The studies reported in this article were not preregistered. The data sets used in this article are publicly available online.[2]

Generalization

The standard approach to training and evaluating neural networks is to choose a data set, divide it randomly into training and test sets, train the system on the training set, and then use its performance on the test set as a proxy for its capacity to generalize to new data. There is a sizable literature focused on ensuring that models learn from the training set in ways that allow them to generalize; for a rich overview, see Goodfellow et al., 2016, Part II.

The standard approach is fine for many purposes, but it raises concerns in a context in which we are trying to determine whether a network has truly acquired a global solution to a target function. In particular, where there is any kind of overlap between the training and test vocabularies, we cannot rule out that the network might be primarily taking advantage of idiosyncrasies in the underlying data set to "cheat." This cheating would happen because the network would simply memorize aspects of the training set and learn a local approximation of the target function that happens to provide traction during assessment. In recent years, these concerns have motivated two complementary innovations in model assessment within AI: *systematic generalization tasks* and *adversarial testing*.

In systematic generalization tasks, one centers the model evaluation around a specific phenomenon that is not represented directly in the test set but that is determined by the training set together with an assumption about how knowledge should generalize. For example, building on ideas from Lake and Baroni (2018) and Wu et al. (2021) define test sets around specific adjective–noun modification structures AN, ensuring that the training set includes only examples involving AN' and $A'N$ for $A \neq A'$ and $N \neq N'$. Given a general assumption of systematicity (Fodor & Pylyshyn, 1988), we expect this training set to support learning to make accurate predictions for AN, and the design of the train–test split gives us confidence that such predictions are not based on mere memorization of AN itself. For other examples involving systematic generalization tests, see Geiger et al., 2020; Goodwin et al., 2020; Hupkes et al., 2020; Yanaka et al., 2020.

In adversarial testing (sometimes called *challenge testing*), one uses behavioral assessments to identify examples and phenomena that fool top-performing models but are natural for humans. This can follow a similar pattern to systematic generalization tasks, but here the models are generally ones that have been trained on large benchmark data sets intended to circumscribe a specific capability. The intent of the adversarial test is to show that models trained on these resources have persistent gaps and other weaknesses.

---

[2] All models, data sets, and code in this article are available at https://github.com/atticusg/NeuralEqualityExperiments.

The intuitive idea behind adversarial testing traces to Winograd (1972; see also Levesque, 2013), and it has been used effectively in a number of domains, including object recognition in digital images (Goodfellow et al., 2015), basic factual question answering (Jia & Liang, 2017), and natural language inference (Glockner et al., 2018; Nie et al., 2019). More recently, adversarial testing has been extended to include training on adversarial examples in order to develop more robust models (Bartolo et al., 2020; Kiela et al., 2021; Nie et al., 2020; Potts et al., 2021).

In our work, we adopt elements of both these modes for assessing generalization. Following Marcus et al. (1999), we propose that networks must be evaluated on assessment sets that are completely disjoint in every respect from the training set, all the way down to the entities involved. For example, below, we train on pairs (a, a) and (a, b), where a and b are representations from a train vocabulary V_T. At test time, we create a new assessment vocabulary V_A, derive equality and inequality pairs (α, α) and (α, β) from that vocabulary, and assess the trained network on these new examples. This is an adversarial setting in service of a specific, systematic learning target (equality). In adopting these methods, we get a clear picture of the system's capacity to generalize, and we can safely say that its performance during assessment is a window into whether a global solution to identity has been learned. This is a very challenging setting for any machine learning model. For the sequential same–different task, we will see that it even requires us to depart from the usual formulations for predictive language models, in that such models standardly cannot even be evaluated on examples that are completely out of vocabulary.

Representations

Essentially all modern machine learning models represent objects using vectors of real numbers. However, there are important differences in how these vectors are used to encode the properties of objects. The method of representation chosen for a particular model impacts both whether there is a natural notion of similarity between entities and whether the model can generalize to examples unseen in training. These two attributes are deeply related; if there is a natural notion of similarity between vector representations, then models can generalize to inputs with representations that are similar to those seen in training.

We characterize two broad approaches to such property encoding— which we call *featural representations* and *nonfeatural representations*—and argue that the differences between them have not been given sufficient attention in the debate about the ability of neural networks to perform relational reasoning. We acknowledge that a dimension of any vector representation is a "feature," but we adopt a usage that is common in cognitive science, namely, that a feature is an interpretable semantic primitive.[3]

We ground our discussion in a hypothetical universe of blocks which vary by shape and color. Figure 2a is a partial view of them, and Figures 2b–2e present four different ways of encoding the properties of these objects in vectors.

Featural Representations

The defining characteristic of *featural* vector representations is that each dimension encodes the value of a single, semantically

interpretable property. The properties can be binary, integer-valued, or real-valued.

We use the term *localist* for the special case of featural representations in which only objects are represented and there is a feature corresponding to each object. In Figure 2b, each column represents the property of being an object, and every object is represented as a vector that has a single unit with value 1 (which is why these representations are often called "one-hot" vectors). There is no shared structure across objects; all are equally (un)related to each other as far as the model is concerned.

We will refer to featural representations that are not localist as *property representations*. Here, column dimensions encode specific, meaningful properties of objects. In our example, we can represent the properties of being red and being blue with two different binary features, and the property of having a certain number of sides as a single integer feature, as in Figure 2c. Unlike with localist representations, objects in this space can have complex relationships to each other, as encoded in the shared structure given by the columns.

Nonfeatural Representations

A *nonfeatural representation* is a vector that encodes property values implicitly across many dimensions. Perhaps the simplest nonfeatural representations are *completely random* vectors, as in Figure 2d. Random representations can be seen as the nonfeatural counterpart to localist representations. In both of these representation schemes, all the objects are equally (un)related to each other, since column-wise patterns are unlikely in random representations and, to the extent that they are present, they exist completely by chance. However, in random representations, all the column dimensions can contribute meaningfully to identifying objects, whereas a localist representation has only one vector unit that determines the identity of any given object.
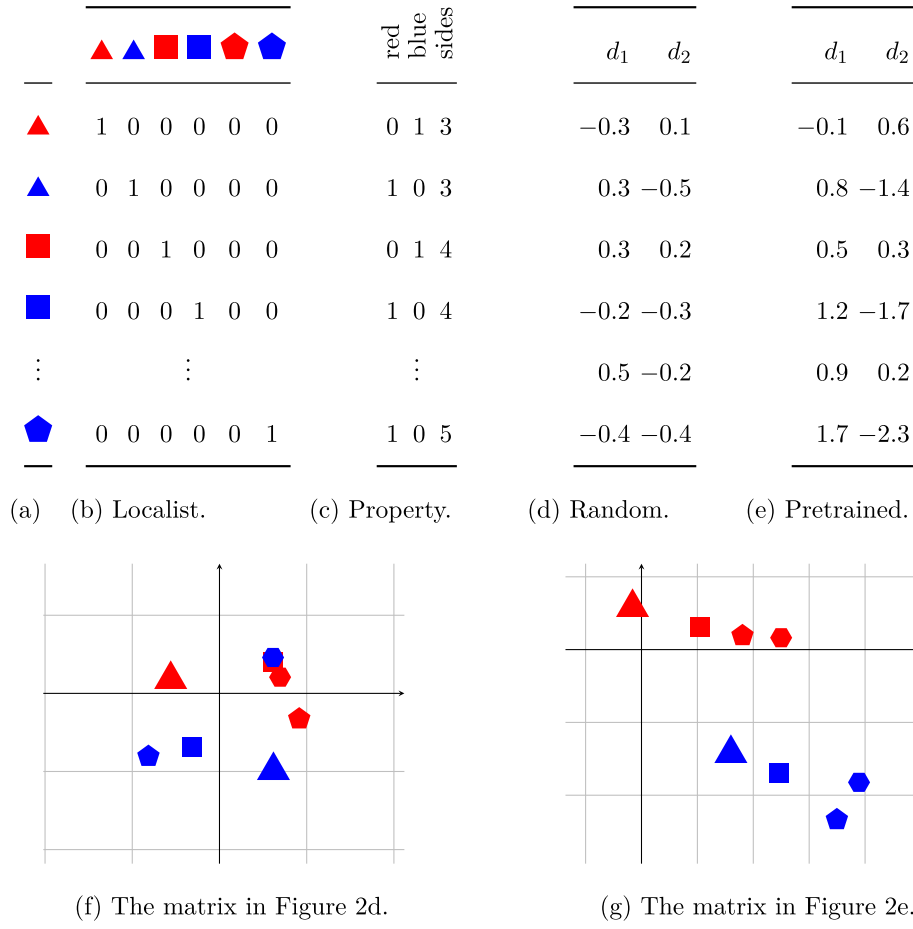
Pretraining

Random representations are a starting point that encodes object identity, but we can *pretrain* these representations via a learning process, imbuing them with rich structure that implicitly encodes property values across many dimensions. Figure 2e provides a simple example. This matrix is the result of pretraining the representations in Figure 2d on three tasks: whether the object is blue, whether the object is red, and the number of sides the object has. Superficially, the two matrices look equally random, but the random representations in Figure 2f have no such structure, while the pretrained representations in Figure 2g do. For example, there is a line that separates blue and red objects.

In our main experiments, we explore pretraining as a path to more efficient learning. There are many potential approaches to pretraining that could be effective in this context. We adopt a simple one based in a multitask classifier. Intuitively, this works

---

[3] The term *distributed representations* is used to refer jointly to what we call property representations, random representations, and pretrained representations. We opted not to use this term because it does not separate property representations from random and pretrained representations, which is the relevant division here. Distributed representations are often contrasted in the neural network literature with *local* or *localist* representations; as discussed below, here we define these terms specifically to refer to representations whose features correspond to specific entities.

Figure 2

Each Matrix Is a Method for Representing the Shapes in Figure 1, Where Each Row Is a Vector Representation of One Shape

		red blue sides		d_1 d_2		d_1 d_2
▲	1 0 0 0 0 0	0 1 3		−0.3 0.1		−0.1 0.6
▲	0 1 0 0 0 0	1 0 3		0.3 −0.5		0.8 −1.4
■	0 0 1 0 0 0	0 1 4		0.3 0.2		0.5 0.3
■	0 0 0 1 0 0	1 0 4		−0.2 −0.3		1.2 −1.7
⋮	⋮	⋮		0.5 −0.2		0.9 0.2
⬠	0 0 0 0 0 1	1 0 5		−0.4 −0.4		1.7 −2.3

(a)   (b) Localist.   (c) Property.   (d) Random.   (e) Pretrained.

(f) The matrix in Figure 2d.   (g) The matrix in Figure 2e.

Note. Localist and property are featural representations where each vector dimension encodes the value of a single, semantically interpretable property. Random and pretrained are nonfeatural representations where the values of properties are encoded implicitly in two dimensions. Random representations and localist representations encode only identity, whereas property representations and pretrained representations encode color and number of sides. See the online article for the color version of this figure.

as follows: random representations (vectors) are paired with labels corresponding to properties like those in our illustration: whether the object is blue, whether the object is red, and so forth. The multitask classifier is trained as usual to predict these labels. However, rather than using the trained classifier to predict such labels, we focus instead on the way the input random representations are affected (pretrained) by this learning process. Once the process is complete, the richly structured inputs that result are then used in other tasks.

In detail, our pretraining model is a feed-forward network with a multitask objective. For an example i and task j, the model is defined as follows:

$$h_i = \text{ReLU}\left(E[i]W_{xh} + b_h\right), \qquad (1)$$

$$y_{i,j} = \textbf{softmax}\left(h_i W_{hy}^j + b_y^j\right). \qquad (2)$$

Here, $E[i]$ is the vector representation for example i in an embedding matrix E of dimension $|V| \times m$, where V is the "vocabulary" of vectors to be pretrained. These representations are multiplied by a vector of weights W_{xh} of dimension $m \times n$ and a bias vector b_h of dimensions n is added to the result. These two steps create a linear projection of the input representation, and the bias term is the value of this linear projection when the input representation is the zero vector. Then, the nonlinear activation function $\text{ReLU}(x) = \max(0, x)$ is applied element-wise to this linear projection. This nonlinearity is what gives the neural model more expressive power than a logistic regression (Cybenko, 1989; Hornik et al., 1989). The result is the hidden representation h_i.

The hidden representation is the input to a separate classification layer for each class j. Each of these classification layers is defined by weights W_{hy}^j and bias b_y^j. Each of these again defines a linear projection, now of h_i. The result is fed through the softmax

activation function: $\mathbf{softmax}\,(x)_i = \frac{\exp x_i}{\sum_j \exp x_j}$. This creates a probability distribution over the classes (in our tasks, "positive" and "negative"). For a given input and task j, the model computes this probability distribution and the input is categorized as the class for j with the higher probability.

Figure 3 depicts this model in detail for three tasks ($J = 3$). The diagram helps reveal that the embedding E and the parameters W_{xh} and b_h are jointly trained against all the tasks, whereas the parameters $h_i W_{hy}^j$ and b_y^j are specific to each task j.

The overall objective of the model is to maximize the sum of the task objective functions. For $|V|$ examples, J tasks, and K_j the number of classes for task j:

$$\max(\theta) \quad \frac{1}{|V|} \sum_{i=1}^{|V|} \sum_{j=1}^{J} \sum_{k=1}^{K_j} y^{i,j,k} \log\left(h_\theta(i)^{j,k}\right), \quad (3)$$

where $y^{i,j,k}$ is the correct label for example i in task j and $h_\theta(i)^{j,k}$ is the predicted value for example i in task j. We use θ to abbreviate all the model parameters: E, W_{xh}, b_h, and W_{hy}^j and b_y^j for each task j.

Our motivation for pretraining is to update the embedding E so that its representations encode properties that can be used by subsequent models. To achieve this, we backpropagate errors through the network and into E. This backpropagation is the step that pretrains E itself.

For the experiments reported below, we initialize E randomly and then, for pretraining on J tasks, we create a random binary vector of length J for each row in E. Each dimension (task) in J is independent of the others. We always pretrain for 10 epochs, where each epoch consists of an example for each item in the vocabulary. This choice is motivated primarily by computational costs; additional pretraining epochs greatly increase experiment run-times, though they do have the potential to imbue the representations with even more useful structure. Pretraining with the optimal hyperparameter settings always led to perfect accuracy on the pretraining tasks.

Pretraining need not be restricted to input representations; all the parameters of a model can be pretrained, offering the possibility that networks might be used as modular components to solve more complex tasks. We realize this possibility with our Experiment 4, where a model pretrained on a simple equality is used as a modular component to compute hierarchical equality.

Localist and Binary Property Representations Prevent Generalization

Featural representations—both localist and property—have the appealing property that they are easy for researchers to interpret because of the tight correspondence between column dimensions and properties. However, this transparency actually inhibits neural networks from discovering general solutions. The core insight is that networks cannot learn anything about column dimensions that are not represented in their training data; whatever weights are associated with those dimensions are unchanged by the learning process, so predictions about those dimensions remain random at test time.

In order to see this, we need to attend the details of how neural models are trained. All the neural models in this article are trained using the back-propagation learning algorithm. An easily observed fact about this algorithm is that, if a unit of the input vector is always zero during training, then any weights connected to that unit and only that unit will not change from their initialized values during training. This means that, when a standard neural model is evaluated on an input vector that has a nonzero value for a unit that was zero throughout training, untrained weights are used to make predictions, and so the network's behavior is unpredictable.

Of all the representation schemes we consider, localist representations are the one that most severely limit generalization. No two representations share a nonzero unit, and so when models are presented with inputs unseen in training, untrained weights are used, and the resulting behavior is unpredictable. In addition, all such representations are orthogonal and equidistant from one

Figure 3

Pretraining Architecture Shown for Three Pretraining Tasks
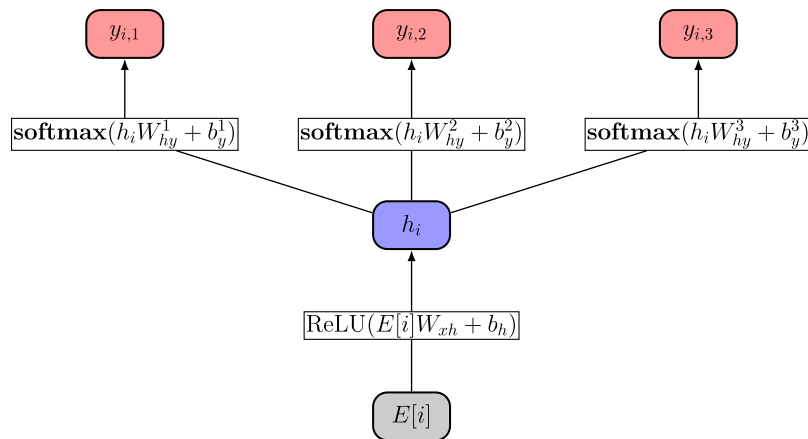
Note. The network is trained as a classifier, but the target of learning is the matrix of input representations $E[i]$, which are imbued with structure from the tasks via backpropagation. See the online article for the color version of this figure.

another, so there is no notion of similarity, and consequently there is no usable transfer of information from training to assessment.

Property representations with binary features also limit generalization, though less severely than localist representations. Localist representations prevent generalization to *entities* unseen during training, while binary feature representations prevent generalization to *properties* unseen during training. For example, if color and shape are represented as binary features, and a red square and blue circle are seen in training, then a model could generalize to the unseen entities of a blue circle or a red square. However, if no entity that is a circle is seen during training, then the binary feature representing the property of being a circle is zero throughout training and untrained weights are used when the model is presented with a entity that is a circle during testing, which once again results in unpredictable behavior.

Property representations with analog features do not inhibit generalization in the same way. If height is represented as an analog feature, then a single unit represents all height values and is always nonzero. Of course, if the height feature is held out entirely during training (i.e., seen only at test time), then generalization will be limited just as it is for binary property representations.

Importantly, nonfeatural representations do not inhibit generalization in these ways, because all units for all representations are nonzero and the network can learn parameters that create complex associations between these entities and its task labels. These representations then can be generalized to new entities, even those unseen during training. In machine learning, learned nonfeatural representations are now the norm, not only because they address this limitation but also because they have proven superior even in settings where all column dimensions are well represented in the training data.

Recent work in machine learning (Kopparti & Weyde, 2020; Weyde & Kopparti, 2018, 2019) attempts to overcome the analytic limitations of binary featural representations by modifying standard neural architectures to have symbolic primitives or changing network weight priors. In our work, we instead opt for nonfeatural representations, which do not have this analytic limitation and are the norm in state-of-the-art artificial intelligence models. There is no need to introduce symbolic primitives or modify network weight priors when nonfeatural representations are used.

Experiment 1: A Feed-Forward Neural Model of Same–Different Relations

We first investigate whether a basic supervised feed-forward neural network can learn the equality relation in the strict setting we describe above, where the training and test vocabularies are disjoint.

Method

Our basic model for equality is a feed-forward neural network with a single hidden representation layer. The model is depicted graphically in Figure 4a and defined as follows:

$$h = \text{ReLU} \left([a; b] W_{xh} + b_h \right), \tag{4}$$

$$y = \textbf{softmax} \left(h W_{hy} + b_y \right). \tag{5}$$

The input is a pair of vectors (a, b), which correspond to the two stimulus objects. These vectors are nonfeatural representations that do not have features encoding properties of the objects or their identity (though they may be pretrained). These are concatenated to form a single vector $[a; b]$ of dimension $2m$, which is the simplest way of merging the two representations to form a single input.

This representation is multiplied by a matrix of weights W_{xh} of dimension $2m \times n$ and a bias vector b_h of dimension n is added to this result, where n is the hidden layer dimensionality, with the ReLU activation applied element-wise to the result. This yields the hidden representation h, which is multiplied by a second matrix of weights W_{hy}, dimension $n \times 2$, and a bias term b_y (Dimension 2) is added to this. A final softmax activation function creates a probability distribution over the classes. For a given input, the model computes this probability distribution and the input is categorized as the class with the higher probability.

During training, this model is presented with positive and negative labeled examples and the parameters W_{xh}, W_{hy}, b_y, and b_h are learned using backpropagation with a cross entropy loss function. This function is defined as follows, for a corpus of N examples and K classes:

$$\max(\theta) \ \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y^{i,k} \log \left(h_\theta(i)^k \right), \tag{6}$$

where θ abbreviates the model parameters (W_{xh}, W_{hy}, b_y, b_h), $y^{i,k}$ is the actual label for example i and class k, and $h_\theta(i)^k$ is the corresponding prediction.
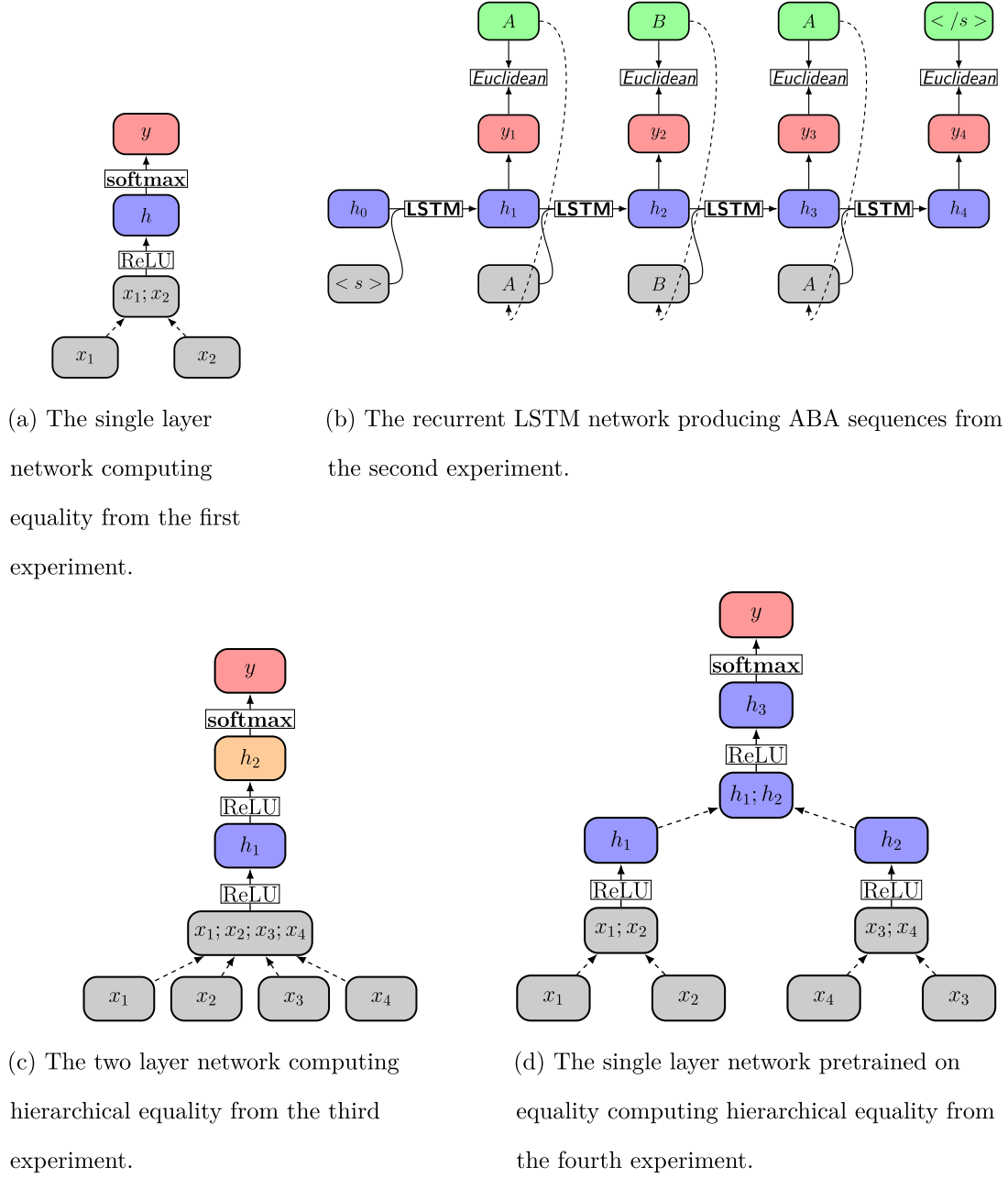
During testing, this model is tasked with categorizing inputs unseen during training. It is straightforward to show that a network like this is capable of learning equality as we have defined it. Indeed, in Appendix B, we provide one possible analytic solution to the equality relation using this neural model. Here we illustrate with a small example network that maps all identity pairs to [0.5, 0.5] and all nonidentity pairs to $[y, 1 - y]$ where $y > 0.5$, which supports a trivial classification rule:

$$\textbf{softmax} \left(\text{ReLU} \left([a; b] \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} + 0 \right) \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} + 0 \right). \tag{7}$$

This result shows that equality in our sense is learnable in principle, but it doesn't resolve the question of whether networks can find this kind of solution given finite training data. To address this issue, we train networks on a stream of pairs of random vectors. Half of these are identity pairs (a, a), labeled with "positive," and half are nonidentity pairs (a, b), labeled with "negative." Trained networks are assessed on the same kind of balanced data set, with vectors that were never seen in training so that, as discussed earlier, we get a clear picture of whether they have found a generalizable solution.

We implemented this network using the multilayer perceptron from scikit-learn (Pedregosa et al., 2011), and we conducted a wide

Figure 4
The Neural Models Used in Each of Our Four Experiments

(a) The single layer network computing equality from the first experiment.

(b) The recurrent LSTM network producing ABA sequences from the second experiment.

(c) The two layer network computing hierarchical equality from the third experiment.

(d) The single layer network pretrained on equality computing hierarchical equality from the fourth experiment.

Note. See the online article for the color version of this figure.

hyperparameter search. See Appendix A for a full specification of our experimental protocol. In the main text, we graph results for the single best hyperparameter setting, reflecting our goal of assessing the possibility of high performance in this task from a network of this type. All the models are trained on balanced data sets, and they never see the same input more than once. The test set is a fixed set of 500 vectors, disjoint from the training sets, and also balanced across the two classes.

Results

Figure 5 shows our results, for a model with an embedding dimension of 10 and a hidden-layer dimension of 100. (In Appendix D, Figure D1 provides results for other network dimensionalities, and Figure D7a provides comparable results for evaluations on the training set.) The representations used in these experiments are random representations that were pretrained using a linear classifier

Figure 5

Experiment 1 Results: Feed-Forward Neural Networks With One Hidden Layer Trained on Our Simple Equality Task (10-Dimensional Inputs, 100-Dimensional Hidden Layers)
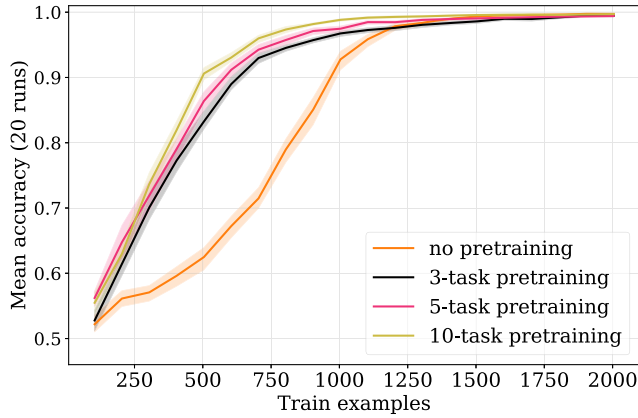
Note. The "no pretraining" model is provided random representations and the "*k*-task pretraining" models are provided random representations that are grounded in *k* binary property domains. See the online article for the color version of this figure.

for 0, 3, 5, or 10 different independent binary feature discrimination tasks (with 0 corresponding to purely random representations). For example, following Figure 2, a three-task model might be trained to encode binary properties that we might gloss as "being blue," "having four sides," and "being red." For all representations, this neural model reached above-chance performance almost immediately, but required upward of 1,000 examples to achieve near perfect accuracy. Interestingly, we observed a clear speed-up with pretraining, with more pretraining tasks resulting in the largest gains. It seems that, by grounding our representations in "property domains" (as represented by the different task dimensions), we imbue them with implicit structure that makes learning easier.

Discussion

Our assessment pairs have nothing in common with the training pairs except insofar as both involve vectors of real numbers of the same dimensionality. During training, the network is told (via labels) which pairs are equality pairs and which are not, but the pairs themselves contain no information about equality per se. It thus seems fair to us to say that these networks have learned equality—or at least how to simulate that relation with near perfect accuracy. Further, the use of representations that are structured by pretraining results in faster learning.

Experiment 2: A Recurrent Neural Model of Sequential Same–Different

Our first model is simple and successfully learns equality. However, this model is supervised with both positive and negative evidence. In the initial debate around these issues, supervision with negative evidence was dismissed as an unreasonably strong

learning regime (e.g., Marcus, 1999). While this argument likely holds true for language learning (in which supervision is generally agreed not to be binary or direct; R. Brown & Hanlon, 1970; Chouinard & Clark, 2003), it is not necessarily true for learning more generally. Nevertheless, learning of sequential rules without negative feedback is possible for infants (Marcus et al., 1999; Rabagliati et al., 2019). In experiments of this type, infants are presented with a set of positive examples of the form ABA. Then their responses are measured for examples using entirely new stimuli that either conform to the ABA regularity or not. Our next model explores whether neural networks can learn this sequential equality task in the same challenging regime with no negative supervision.

Method

To explore learning with only positive instances, we use a neural Long Short-Term Memory (LSTM) language model, a recurrent network with the ability to selectively forget and remember information (Hochreiter & Schmidhuber, 1997). Language models are sequential: at each timestep, they predict an output given their predictions about the preceding timesteps. As typically formulated, the prediction function is just a classifier: at each timestep, it predicts a probability distribution over the entire vocabulary of options, and the item with the highest probability is chosen as a symbolic output. This output becomes the input at the next timestep, and the process continues.

As we noted above, this standard formulation will not work in situations in which we want to make predictions about test items with an entirely disjoint vocabulary from the training sample. The classifier function will get no feedback about these out-of-vocabulary items during training, and so it will never predict them during testing.

To address this issue, we reformulate the prediction function. Our proposal is to have the model predict output vector representations—instead of discrete vocabulary items—at each timestep. During training, the model seeks to minimize the distance between these output predictions and the representations of the actual output entities. During assessment, we take the prediction to be the item in the entire vocabulary (training and assessment) whose representation is closest to the predicted vector (in terms of Euclidean distance). This fuzzy approach to prediction creates enough space for the model to predict sequences from an entirely new vocabulary.

The specific model we use for this is as follows:

$$h_t = \textbf{LSTM}\,(x_t, h_{t-1}), \tag{8}$$

$$y_t = h_t\,W + b. \tag{9}$$

This holds for $t > 0$, and we set $h_0 = 0$. **LSTM** is a long short-term memory cell (Hochreiter & Schmidhuber, 1997). We visualize this model in Figure 4b.

The input is a sequence of vectors x_1, x_2, $x_{3,}$ . . ., each of dimension m, which correspond to a sequence of stimulus objects. These vectors are, again, nonfeatural representations that do not have features encoding properties of the objects or their identity, though they may be pretrained to encode such properties more abstractly.

At each timestep t, the input vector x_t is fed into the **LSTM** cell along with the previous hidden representation h_{t-1}. The defining

feature of an **LSTM** is the ability to decide whether to store information from the current input, x_t, and whether to remember or forget the information from the previous timestep h_{t-t}. The output of the **LSTM** cell is the hidden representation for the current time step h_t. The dimension of the hidden representations is n. The hidden representation is multiplied by a matrix W with dimensionality $n \times m$ to produce y_t. This result, y_t, is a linear projection of the hidden representation into the input vector space, which is necessary because y_t is a prediction of what the next input, x_{t+1}, will be.

The objective function is as follows:

$$\max(\theta) - \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \|h_\theta(x^{i,0:t-1}) - x^{i,t}\|^2, \quad (10)$$

for N examples. Here, T_i is the length of example i. As before, θ abbreviates the parameters of the model as specified in Equations 8 and 9. We use $h_\theta(x^{i,0:t-1})$ for the vector predicted by the model for example i at timestep t, which is compared to the actual vector at timestep t via squared Euclidean distance (i.e., the mean squared error).

We implemented this recurrent LSTM network for the sequential ABA task using PyTorch (Paszke et al., 2019). As in our previous experiment, we conducted a wide hyperparameter search (see Appendix A for details). In the main text, we graph results for the single best hyperparameter setting. We trained and tested on vocabularies of size 20, which are reconstructed between each run.

Appendix C provide an analytic solution to the ABA task using this model. To see how well the model performs in practice, we trained networks on sequences $a\ b\ a$, where $b \neq a$. We show the network every such sequence during training, from an underlying vocabulary of 20 items (creating a total of 380 examples). To assess how well the model learns this pattern, we seed it with x where x is an item from a disjoint vocabulary from that seen in training, and we say that a prediction is accurate if the model continues with $y\ x$, where y is any character (from the training or assessment vocabulary) except x. In our experiments, we use a test vocabulary of 26 items, which creates 52 distinct $y\ x$ continuations and hence 52 distinct test examples.

Results

Figure 6 shows our results, for networks with two-dimensional embeddings and 100-dimensional hidden representations (Figure D2 provides results for a wide range of network dimensionalities.). Unlike for the previous equality experiment, we found that we had to allow the model to experience multiple epochs of training on the same set in order to succeed, and tens of thousands of training examples were necessary. We considered a range of representations (as in Experiment 1), and the model was again successful with all representations. However, unlike in Experiment 1, we do not find evidence in favor of pretraining for this task.

Figure D7b provides training-set results that are comparable to those in Figure 6. These models very quickly learn to perform with near perfect accuracy on the training set, creating a large gap between training-set performance and test-set performance. This contrasts with our other models, for which the differences between these two modes of evaluation are small. This difference likely traces to the nature of language modeling tasks (compared with

Experiment 2 Results: LSTM Recurrent Neural Networks Trained on Our Sequential Equality Task (2-Dimensional Inputs, 100-Dimensional Hidden Layers)
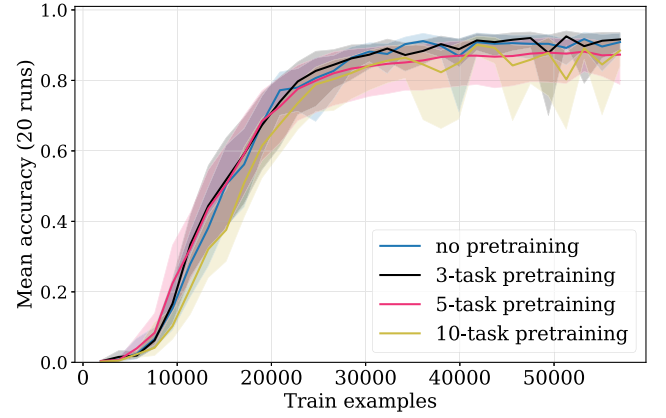
Note. The "no pretraining" model is provided random representations and the "k-task pretraining" models are provided random representations that are grounded in k binary property domains via pretraining. All the training examples are presented at once over multiple epochs. See the online article for the color version of this figure.

classifiers) as well as our decision to allow the model to experience multiple epochs of the same training data.

Discussion

These sequential models are given no negative examples and they must predict into a totally new vocabulary. Despite these challenges, they succeed at learning the underlying patterns in our data. On the other hand, the learning process is slow and data intensive. We hypothesized that grounding representations in property domains via pretraining might lead to noticeable speed-ups, as it did in for our simple same–different task (Experiment 1), but we did not see this effect in practice. We speculate that there may be model variants that reduce these demands, given that learning is in principle possible in this architecture, but we leave them to future work.

Experiment 3: A Feed-Forward Neural Model of Hierarchical Same–Different Relations

Given the strong results found for simple equality relations, we can ask whether more challenging equality problems are also learnable in our setting. The hierarchical equality task used by Premack (1983) is an interesting test case: given a pair of pairs ((a, b), (c, d)), the label is "positive" if ($a = b$) and ($c = d$) or ($a \neq b$) and ($c \neq d$). The mixed cases (i.e., ($a = b$) and ($c \neq d$); ($a \neq b$) and ($c = d$)) are labeled "negative."

Premack suggested that the ability exemplified by this task— reasoning about hierarchical *same* and *different* relations—could represent a form of symbolic abstraction uniquely enabled by language. Given the nonsymbolic nature of our models, our simulations provide a test of this hypothesis, though we should look critically at their ability to find good solutions with reasonable amounts of training data.

Method

We can approach this task using the same model and methods as we used for equality, with the relatively minor change of providing the network four vector representations instead of two. We found that feed-forward neural networks with only one hidden layer required nearly 100,000 training examples to solve this task (see Figure D3). We hypothesized that a single hidden layer network might be suboptimal here. This task is intuitively hierarchical: if one works out the equality labels for each of the two pairs, then the further classification decision can be done entirely on that basis. Our current neural network might be too shallow to find this kind of decomposition. To address this issue, we use a feed-forward network with two hidden layers. We visualize this model in Figure 4c and we define

$$h_1 = \text{ReLU}\left([a;b;c;d]\, W_{xh} + b_{h_1}\right), \tag{11}$$

$$h_2 = \text{ReLU}\left(h_1 W_{hh} + b_{h_2}\right), \tag{12}$$

$$y = \textbf{softmax}\left(h_2 W_{hy} + b_y\right). \tag{13}$$

We implemented this network using PyTorch and conducted a wide hyperparameter search (see Appendix A for details). In the main text, we graph results for the single best hyperparameter setting. The training and testing protocol is the same as in Experiment 1, and we further ensure that the train and test sets are balanced across the four distinct input types for the hierarchical task (same/same, different/different, same/different, and different/same).

Results

Figure 7a shows our results for networks with 10-dimensional embeddings and 100-dimensional hidden layers (Figure D4 shows results for other network dimensionalities). We again considered a range of representations, and again the network succeeded across this range, with pretraining increasing performance dramatically. The network required more than 20,000 training instances to reach top performance, and upward of 10,000 examples with pretrained representations.

Discussion

Our model with two hidden layers requires vastly more data than human participants get in similar experiments. For example, sequential rule learning experiments typically involve short exposures in the range of dozens to hundreds of examples (e.g., Endress et al., 2005; Marcus et al., 1999). Thus, it is worth asking whether there are other solutions that would be more data efficient and more in line with human capabilities. We next seek to further capitalize on the hierarchical nature of this task by defining a modular pretraining regime in which previously learned capabilities are recruited for new tasks.

Experiment 4: A Tree-Structured Model of Hierarchical Same–Different Relations Pretrained on Simple Equality

Our successful results training neural networks on simple equality suggested another strategy for solving the hierarchical equality task. Rather than requiring our networks to find solutions from scratch,

we pretrained them on basic equality tasks and then used those parameters as a starting point for learning hierarchical equality. This set of simulations was conceptually similar to our previous experiments with pretrained input representations, but now we pretrained an entire subpart of the model, rather than just input representations. This approach parallels the experimental paradigm used by Thompson et al. (1997), in which chimpanzees that received pretraining on a basic equality (*same/different* judgment) task—but not naive chimpanzees—succeed in a hierarchical equality task.

Method

The hierarchical equality task requires computing the equality relation three times: compute whether the first two inputs are equal, compute whether the second two inputs are equal, then compute whether the truth-valued outputs of these first two computations are equal. We propose to use the same network pretrained on basic equality to perform all three equality computations. The model is depicted in Figure 4d and defined as follows:

$$h_1 = \text{ReLU}\left([a;b]\, W_{xh} + b_h\right), \tag{14}$$

$$h_2 = \text{ReLU}\left([c;d]\, W_{xh} + b_h\right), \tag{15}$$

$$h_3 = \text{ReLU}\left([h_1;h_2]\, W_{xh} + b_h\right), \tag{16}$$

$$y = \textbf{softmax}\left(h_3 W_{hy} + b_y\right). \tag{17}$$

Here, W_{xh}, W_{hy}, b_h, and b_y are the parameters from the model in Equations 4 and 5 already trained on basic equality. Crucially, the same parameters, W_{xh} and b_h, are used three times: twice to compute representations encoding whether a pair of input entities are equal (h_1, h_2), and once to compute a representation (h_3) encoding whether the truth values encoded by h_1 and h_2 are equal. This final representation is then used to compute a probability distribution over two classes, and the class with the higher probability is predicted by the model.[4]

We implemented this network using PyTorch using the same protocols as in Experiments 1–3 (see Appendix A for details). In the main text, we graph results for the single best hyperparameter setting. The training and testing protocol matches the one used in Experiment 3.
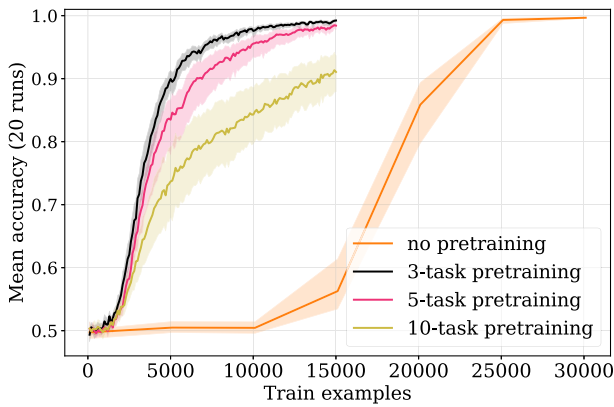
Results

Figure 7b shows our results for networks with 25-dimensional inputs, which sets the dimensionality of the hidden representations to 50. (See Figure D4 for results for other network dimensionalities.) All the models have above chance performance after being trained only on the simple equality task—that is, they achieve zero-shot generalization to the hierarchical task and within two thousand examples, the models achieve near perfect accuracy. Additionally pretraining the input representations leads to modest but consistent gains in terms of learning speed (In Figure D7d, we provide

---

[4] A clear limitation of this approach is that it presupposes the hierarchical nature of the task rather than learning it from the data. Recent developments with modular networks and routing networks might open up a path to removing this limitation (Andreas et al., 2016; Cases et al., 2019; Chang et al., 2019; Kirsch et al., 2018). We briefly return to this point in our General Discussion section below.

Figure 7

Results for (a) Hierarchical Sequential Equality Task Without Pretraining on Simple Equality and (b) With Pretraining on Simple Equality

(a) Experiment 3 results: feed-forward neural networks with two hidden layers trained on our hierarchical equality task (10-dimensional inputs; 100-dimensional hidden layers).
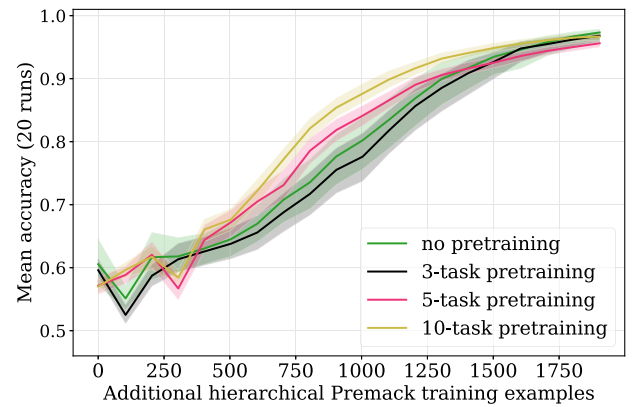
(b) Experiment 4 results: simple equality networks (with 25-dimensional inputs and, in turn, 50-dimensional hidden representations) applied to the hierarchical equality task as pretrained components. Even with no additional training instances for this task ("zero shot"), all models achieve greater than chance accuracy, and even modest amounts of additional training on the task lead to excellent performance. Additionally pretraining the input representations leads to modest improvements.

Note. The "no pretraining" models are provided random representations and the "*k*-task pretraining" models are provided random representations that are grounded in *k* binary property domains via pretraining learning tasks. The Experiment 3 model requires vastly more task-specific training data than the Experiment 4 model to achieve good results, which helps to show the value of modular reuse of pretrained components. See the online article for the color version of this figure.

comparable training set assessments; in that context, pretraining has the opposite effect, slightly slowing down learning).

Discussion

It is remarkable that a model trained only on equality between entities is able to get traction on a problem that requires determining whether equality holds between the truth values encoded in two learned representations. What is more, even small amounts of additional training data lead to excellent performance on the task. This shows the value of pretraining regimes involving entire networks, and it begins to show how trained networks can serve as modular components that are useful for solving more complex tasks.

Intuitively, this problem presents four different input class types: the two "positive" classes (same/same and different/different), and

the two "negative" classes (same/different and different/same). We see noteworthy variation in model performance across these classes: where there is little or no training data, accuracy is high for the "negative" classes and low for the "positive" ones. This pattern stabilizes quickly, and then we find that the different/different class is persistently the most difficult, which aligns with general qualitative findings from behavioral experiments. Figure D6 depicts these findings for our best model.

General Discussion

Equality is a key case study for understanding the origins of human relational reasoning. This case study has been puzzling for symbolic accounts of reasoning because such accounts do not provide a compelling explanation for why some equality tasks

are so easy to learn and others are so hard. In addition, evidence of graded learning and generalization in nonhuman species suggests that a gradual learning account might provide more traction in explaining the empirical data (Wasserman et al., 2017). Inspired by this work, we revisited a longstanding debate about whether neural network models can learn equality relations from data (Alhama & Zuidema, 2019).

Our work here makes three contributions to this debate. First, we show that nonfeatural representations—both random and pretrained—allow standard neural networks to learn simple, sequential, and hierarchical equality tasks. Both the research that originated this debate (Dienes et al., 1999; Elman, 1999; Marcus, 2001; Marcus et al., 1999; Negishi, 1999; Seidenberg & Elman, 1999a, 1999b) and more recent work (Alhama & Zuidema, 2018; Kopparti & Weyde, 2020; Weyde & Kopparti, 2018, 2019) only involve experiments where featural representations are used. We suggest that this choice led directly to the negative conclusions from this body of work. Second, we show that neural networks can achieve high test accuracy on the sequential equality task with no negative feedback, suggesting that a negative feedback learning regime is not critical for learning equality. Finally, we show that a neural network trained only on simple equality can generalize to hierarchical equality, even in a "zero-shot" evaluation. Although pretrained representations sometimes led to faster learning, they were not a necessary component for models to succeed, and success was possible even using random representations.

The Implications of Pretraining

In some settings, our current models require many more training instances than humans seem to need. However, our pretraining approach suggests a path forward: by using pretrained models as modular components, we can get traction on challenging tasks without any training specifically for those tasks. In some cases, even a small amount of additional training can make a substantial difference.

One implication of our pretraining findings is that it should be possible to scaffold nonhuman animals' performance in complex, hierarchical equality tasks via training on simpler ones. Indeed, Smirnova et al. (2015) show just this result in crows, consistent with our findings. Although we do not discount the potential role of linguistic labels in informing adult humans' expertise in such tasks (Gentner & Goldin-Meadow, 2003), pretraining also provides a potential account of how infants and young children might succeed in a range of equality reasoning tasks without access to specific linguistic symbols like "same" (Ferry et al., 2015; Hochmann et al., 2016; Walker et al., 2016).

While pretraining in the sparse, artificial contexts of our models is far from accurately capturing the experience of infants and other animals, we believe it is reasonable to think of cognitive systems as "pretrained" by their perceptual experiences and by their evolved perceptual architecture. This "pretraining" might lead them to have continuous representations of perceptual information that are nonfeatural in the sense of our models, but which still encode psychologically meaningful properties. In this sense, human generalizations could be similar to those made by our models in that both encode meaningful properties of stimuli in a way that is not perfectly individuated or discrete. Further, both can make rapid (and sometimes one-shot) generalizations, especially in cases where the

agent has had prior experience with the stimuli (Rabagliati et al., 2019). This kind of viewpoint on our pretraining simulations is consistent with a range of views of how complex human abilities are created from simpler one (e.g., Frank et al., 2008; Heyes, 2018).

We have not offered a theory of how an agent might come to be able to combine pretrained components to solve more complex tasks, however. Rather, we simply pieced these components together by hand to achieve our desired outcomes. A natural next step would be to learn how to effectively combine these modular pieces. The artificial intelligence literature currently offers a range of techniques that could be used for this purpose, including neural module networks (Andreas et al., 2016), compositional recursive learners (Chang et al., 2019), modular networks (Kirsch et al., 2018), and recursive routing networks (Cases et al., 2019). These techniques might provide a fruitful avenue for simulating the emergence of more complex cognitive abilities.

Beyond Exact Equality

We have so far confined our discussion to idealized notions of *same* and *different* that are defined in terms of complete mathematical identity. These are the terms of the original debate involving Marcus et al. (1999), Dienes et al. (1999), Seidenberg and Elman (1999a, 1999b), Elman (1999), Negishi (1999), and others. We hope to have established that neural networks can solve this version of the problem. However, the cognitive notions of *same* and *different* are richer and more multifaceted than this.

Our models and findings hold for the case where judgments are made about one and the same image, but it is an open question how to extend them to situations in which one has two distinct images of the same entity and the intended label is "same." Even this basic expansion of the problem space poses many challenging new questions concerning context dependence (e.g., when are two different apples of the same type judged to be "the same") and human fallibility (one could fail to perceive relevant differences or fail to ignore irrelevant ones), and thus the very notion of success is also likely to become more graded and context dependent. Webb et al. (2021) begin to explore this problem, and those initial results give us confidence that the kinds of networks we propose here could be conditioned contextually to provide flexible, context-sensitive interpretations for sameness and graded similarity (Medin et al., 1993), holding the promise of unifying models of identity, similarity, and related notions.

More broadly still, our work suggests a possible way forward in understanding the acquisition of logical semantics. Graded logical functions like those our models learned here could form the foundation for a semantics of words like "same" (Potts, 2019). Such an option is appealing because it escapes from the circularity of defining the semantics of linguistic symbols as originating in a mental primitive same. A semantics for "same" requires defining its inputs and outputs as well as how it composes with other symbols. Further, understanding the pragmatics of "same" then requires understanding how exact the similarity must be for the statement to be true in particular contexts.

Conclusions

Even the best present-day deep learning models often fail to achieve systematic and general solutions (Geiger et al., 2019;

Hupkes et al., 2020; Lake & Baroni, 2018; Linzen, 2020; Marcus, 2018; Wu et al., 2021). However, as our experiments demonstrate, abstract equality reasoning is too simple of a domain to expose this failure. Indeed, at this point, even very complicated problems in logical semantics might not suffice to make this point, given the high level of performance that can sometimes be achieved (Bowman, Potts, et al., 2015; Mul & Zuidema, 2019).

Earlier debates about the nature of equality computations centered around the question of whether models included explicitly symbolic elements. We believe ours do not; but it is of course possible to quibble with this judgment. For example, since the supervisory signals used in Experiments 1 and 3 are generated based on a symbolic rule, perhaps that makes these models symbolic under some definition (Of course, the same argument could be applied to the supervision signal that is provided to crows, baboons, and human children in some tasks).

This kind of argument increasingly seems terminological, rather than substantive, however. Neural models may learn symbolic computations in ways that do not contain explicit operators but that nevertheless have elements that play the same causal role. Geiger et al. (2020) and Geiger et al., (2021) perform causal abstraction analyses of trained neural models and find that they define and use symbolic variables at an abstract level of computation. In related work, Geiger, Wu, et al. (2021) train neural models to implement abstract symbolic computations in order to solve systematic generalization tasks. These techniques leverage recent innovations in models of causality and abstraction (Beckers & Halpern, 2019; Chalupka et al., 2016; Rubenstein et al., 2017), and they provide more evidence for the classic view that neural networks are capable of implementing symbolic computations (Fodor & Pylyshyn, 1988; Smolensky, 1986) which has received recent focus again Piantadosi (2021); Santoro et al. (2021). So an advocate for a symbolic view could even say that such networks (and likely ours as well) have learned symbols, though this move might lead to more confusion than clarity with respect to the original debate.

In the end, the primary goal is not to resolve the question of whether explicit symbolic representations are necessary (although by most reasonable definitions we believe we have presented a negative answer to that question). Rather, the hope is to create an explicit learning theory for relational reasoning, which might in turn explain the rich and puzzling empirical landscape (Carstensen & Frank, 2021). Our hope is that—by providing a range of architectures that learn equality functions—the work described here takes a first step in this direction.

References

Alhama, R. G., & Zuidema, W. (2018). Pre-wiring and pre-training: What does a neural network need to learn truly general identity rules? *Journal of Artificial Intelligence Research*, *61*, 927–946. https://doi.org/10.1613/jair.1.11197

Alhama, R. G., & Zuidema, W. (2019). A review of computational models of basic rule learning: The neural-symbolic debate and beyond. *Psychonomic Bulletin & Review*, *26*(4), 1174–1194. https://doi.org/10.3758/s13423-019-01602-z

Anderson, E. M., Chang, Y.-J., Hespos, S., & Gentner, D. (2018). Comparison within pairs promotes analogical abstraction in three-month-olds. *Cognition*, *176*, 74–86. https://doi.org/10.1016/j.cognition.2018.03.008

Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016, June). Learning to compose neural networks for question answering. In K. Knight, A. Nenkova, & O. Rambow (Eds.), *Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies* (pp. 1545–1554). Association for Computational Linguistics. https://aclanthology.org/N16-1181

Avarguès-Weber, A., Deisig, N., & Giurfa, M. (2011). Visual cognition in social insects. *Annual Review of Entomology*, *56*(1), 423–443. https://doi.org/10.1146/ento.2011.56.issue-1

Bartolo, M., Roberts, A., Welbl, J., Riedel, S., & Stenetorp, P. (2020). Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, *8*(2), 662–678. https://doi.org/10.1162/tacl_a_00338

Beckers, S., & Halpern, J. Y. (2019). Abstracting causal models. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(1), 2678–2685. https://doi.org/10.1609/aaai.v33i01.33012678

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015, September). A large annotated corpus for learning natural language inference. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, & Y. Marton (Eds.), *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 632–642). Association for Computational Linguistics. https://aclanthology.org/D15-1075

Bowman, S. R., Potts, C., & Manning, C. D. (2015, July). Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality* (pp. 12–21). Association for Computational Linguistics. https://aclanthology.org/W15-4002

Brown, R., & Hanlon, C. (1970). Derivational complexity and order of development in speech. In J. R. Hayes (Ed.), *Cognition and the development of language*. Wiley.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 1877–1901). Curran Associates. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Carstensen, A., & Frank, M. C. (2021). Do graded representations support abstract thought? *Current Opinion in Behavioral Sciences*, *37*, 90–97. https://doi.org/10.1016/j.cobeha.2020.10.009

Cases, I., Rosenbaum, C., Riemer, M., Geiger, A., Klinger, T., Tamkin, A., Li, O., Agarwal, S., Greene, J. D., Jurafsky, D., Potts, C., & Karttunen, L. (2019, June). Recursive routing networks: Learning to compose modules for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies* (pp. 3631–3648). Association for Computational Linguistics. https://www.aclweb.org/anthology/N19-1365

Castro, L., Kennedy, P. L., & Wasserman, E. A. (2010). Conditional same-different discrimination by pigeons: Acquisition and generalization to novel and few-item displays. *Journal of Experimental Psychology: Animal Behavior Processes*, *36*(1), 23–38. https://doi.org/10.1037/a0016326

Chalupka, K., Eberhardt, F., & Perona, P. (2016, May 09–11). Multi-level cause-effect systems. In A. Gretton & C. C. Robert (Eds.), *Proceedings of the 19th international conference on artificial intelligence and statistics* (Vol. 51, pp. 361–369). PMLR. http://proceedings.mlr.press/v51/chalupka16.html

Chang, M., Gupta, A., Levine, S., & Griffiths, T. L. (2019). Automatically composing representation transformations as a means for generalization. In *International conference on learning representations*. OpenReview.net. https://openreview.net/forum?id=B1ffQnRcKX

Chouinard, M. M., & Clark, E. V. (2003). Adult reformulations of child errors as negative evidence. *Journal of Child Language*, *30*(3), 637–669. https://doi.org/10.1017/S0305000903005701

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*, 2493–2537. https://dl.acm.org/doi/10.5555/1953048.2078186

Cook, R. G., & Wasserman, E. A. (2007). Learning and transfer of relational matching-to-sample by pigeons. *Psychonomic Bulletin & Review*, *14*(6), 1107–1114. https://doi.org/10.3758/BF03193099

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314. https://doi.org/10.1007/BF02551274

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. https://www.aclweb.org/anthology/N19-1423

Dienes, Z., Altmann, G. T., & Gao, S.-J. (1999). Mapping across domains without feedback: A neural network model of transfer of implicit knowledge. *Cognitive Science*, *23*(1), 53–82. https://doi.org/10.1207/s15516709cog2301_3

Elman, J. L. (1999). *Generalization, rules, and neural networks: A simulation of Marcus et al.* (HTML document)

Endress, A. D., Scholl, B. J., & Mehler, J. (2005). The role of salience in the extraction of algebraic rules. *Journal of Experimental Psychology: General*, *134*(3), Article 406. https://doi.org/10.1037/0096-3445.134.3.406

Fagot, J., & Thompson, R. K. (2011). Generalized relational matching by guinea baboons (papio papio) in two-by-two-item analogy problems. *Psychological Science*, *22*(10), 1304–1309. https://doi.org/10.1177/0956797611422916

Ferry, A. L., Hespos, S. J., & Gentner, D. (2015). Prelinguistic relational concepts: Investigating analogical processing in infants. *Child Development*, *86*(5), 1386–1405. https://doi.org/10.1111/cdev.2015.86.issue-5

Fleuret, F., Li, T., Dubout, C., Wampler, E., Yantis, S., & Geman, D. (2011). Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences of the United States of America*, *108*, 17621–17625. https://doi.org/10.1073/pnas.1109168108

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, *28*(1–2), 3–71. https://doi.org/10.1016/0010-0277(88)90031-5

Frank, M. C., Everett, D. L., Fedorenko, E., & Gibson, E. (2008). Number as a cognitive technology: Evidence from Pirahã language and cognition. *Cognition*, *108*(3), 819–824. https://doi.org/10.1016/j.cognition.2008.04.007

Frank, M. C., & Tenenbaum, J. B. (2011). Three ideal observer models for rule learning in simple languages. *Cognition*, *120*(3), 360–371. https://doi.org/10.1016/j.cognition.2010.10.005

Geiger, A., Cases, I., Karttunen, L., & Potts, C. (2019, November). Posing fair generalization tasks for natural language inference. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing* (pp. 4475–4485). Association for Computational Linguistics. https://www.aclweb.org/anthology/D19-1456

Geiger, A., Lu, H., Icard, T., & Potts, C. (2021). Causal abstractions of neural networks. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems*. Curran Associates. https://arxiv.org/abs/2106.02997

Geiger, A., Richardson, K., & Potts, C. (2020, November). Neural natural language inference models partially embed theories of lexical entailment and negation. In A. Alishahi, Y. Belinkov, G. Chrupala, D. Hupkes, Y. Pinter, & H. Sajjad (Eds.), *Proceedings of the third blackboxnlp workshop on analyzing and interpreting neural networks for NLP* (pp. 163–173). Association for Computational Linguistics. https://www.aclweb.org/anthology/2020.blackboxnlp-1.16

Geiger, A., Wu, Z., Lu, H., Rozner, J., Kreiss, E., Icard, T., Goodman, N. D., & Potts, C. (2021). *Inducing causal structure for interpretable neural networks*. To appear in Proceeding of the International Conference on Machine Learning. https://arxiv.org/abs/2112.00826 (arXiv:2112.00826)

Gentner, D., & Goldin-Meadow, S. (2003). *Language in mind: Advances in the study of language and thought*. MIT Press.

Gervain, J., Berent, I., & Werker, J. F. (2012). Binding at birth: The newborn brain detects identity relations and sequential position in speech. *Journal of Cognitive Neuroscience*, *24*(3), 564–574. https://doi.org/10.1162/jocn_a_00157

Giurfa, M., Zhang, S., Jenett, A., Menzel, R., & Srinivasan, M. V. (2001). The concepts of 'sameness' and 'difference' in an insect. *Nature*, *410*(6831), 930–933. https://doi.org/10.1038/35073582

Glockner, M., Shwartz, V., & Goldberg, Y. (2018, July). Breaking NLI systems with sentences that require simple lexical inferences. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th annual meeting of the Association for Computational Linguistics (volume 2: Short papers)* (pp. 650–655). Association for Computational Linguistics.

Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). *Explaining and harnessing adversarial examples* [Conference session]. Proceedings of the International Conference on Learned Representations, San Diego, California, United States.

Goodwin, E., Sinha, K., & O'Donnell, T. J. (2020, July). Probing linguistic systematicity. In D. Jurafsky, J. Chai, N. Schluter, & J. R. Tetreault (Eds.), *Proceedings of the 58th annual meeting of the Association for Computational Linguistics* (pp. 1958–1969). Association for Computational Linguistics.

Gülçehre, Ç., & Bengio, Y. (2016). Knowledge matters: Importance of prior information for optimization. *Journal of Machine Learning Research*, *17*(1), 226–257. http://jmlr.org/papers/v17/gulchere16a.html

Heyes, C. (2018). *Cognitive gadgets: The cultural evolution of thinking*. Harvard University Press.

Hochmann, J.-R., Mody, S., & Carey, S. (2016). Infants' representations of same and different in match-and non-match-to-sample. *Cognitive Psychology*, *86*(6), 87–111. https://doi.org/10.1016/j.cogpsych.2016.01.005

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Hupkes, D., Dankers, V., Mul, M., & Bruni, E. (2020). Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, *67*, 757–795. https://doi.org/10.1613/jair.1.11674

Jia, R., & Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 2021–2031). Association for Computational Linguistics.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., & Girshick, R. (2017). CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, & R. B. Girshick (Eds.), *2017 IEEE conference on computer vision and pattern recognition (cvpr)* (pp. 1988–1997). IEEE. https://doi.org/10.1109/CVPR.2017.215

Kiela, D., Bartolo, M., Nie, Y., Kaushik, D., Geiger, A., Wu, Z., Vidgen, B., Prasad, G., Singh, A., Ringshia, P., Ma, Z., Thrush, T., Riedel, S., Waseem, Z., Stenetorp, P., Jia, R., Bansal, M., Potts, C., & Williams, A. (2021, June). Dynabench: Rethinking benchmarking in NLP. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, & Y. Zhou (Eds.), *Proceedings of*

the 2021 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies (pp. 4110–4124). Association for Computational Linguistics. https://www.aclweb.org/anthology/2021.naacl-main.324

Kim, J., Ricci, M., & Serre, T. (2018). Not-so-CLEVR: learning same–different relations strains feedforward neural networks. Interface Focus, 8(4). https://doi.org/10.1098/rsfs.2018.0011

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization [Conference session]. Proceedings of the 3rd International Conference on Learning Representations, San Diego, California, United States.

Kirsch, L., Kunze, J., & Barber, D. (2018). Modular networks: Learning to decompose neural computation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), Advances in neural information processing systems (Vol. 31). Curran Associates. https://proceedings.neurips.cc/paper/2018/file/310ce61c90f3a46e340ee8257bc70e93-Paper.pdf

Kopparti, R., & Weyde, T. (2020). Weight priors for learning identity relations.

Lake, B., & Baroni, M. (2018, July 10–15). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In J. Dy & A. Krause (Eds.), Proceedings of the 35th international conference on machine learning (Vol. 80, pp. 2873–2882). PMLR. https://proceedings.mlr.press/v80/lake18a.html

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological Review, 104(2), Article 211. https://doi.org/10.1037/0033-295X.104.2.211

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Levesque, H. J. (2013, August). On our best behaviour [Conference session]. Proceedings of the Twenty-Third International Conference on Artificial Intelligence, Beijing, China.

Linzen, T. (2020, July). How can we accelerate progress towards human-like linguistic generalization? In Proceedings of the 58th annual meeting of the Association for Computational Linguistics (pp. 5210–5217). Association for Computational Linguistics. https://aclanthology.org/2020.acl-main.465

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. CoRR, abs/1907.11692. http://arxiv.org/abs/1907.11692

Marcus, G. F. (1999). Rule learning by seven-month-old infants and neural networks. Response to Altmann and Dienes. Science, 284, Article 875. https://doi.org/10.1126/science.283.5398.77

Marcus, G. F. (2001). The algebraic mind: Integrating connectionism and cognitive science. MIT Press.

Marcus, G. F. (2018). Deep learning: A critical appraisal. CoRR, abs/1801.00631. https://arxiv.org/pdf/1801.00631.pdf

Marcus, G. F., Vijayan, S., Rao, S. B., & Vishton, P. M. (1999). Rule learning by seven-month-old infants. Science, 283(5398), 77–80. https://doi.org/10.1126/science.283.5398.77

McRae, K., & Hetherington, P. A. (1993). Catastrophic interference is eliminated in pretrained networks [Conference session]. Proceedings of the 15th annual conference of the Cognitive Science Society, Hillsdale, New Jersey, United States.

Medin, D. L., Goldstone, R. L., & Gentner, D. (1993). Respects for similarity. Psychological review, 100(2), 254–278. https://doi.org/10.1037/0033-295X.100.2.254

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Q. Weinberger (Eds.), Advances in neural information processing systems (Vol. 26, pp. 3111–3119). Curran Associates. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

Mul, M., & Zuidema, W. (2019). Siamese recurrent networks learn first-order logic reasoning and exhibit zero-shot compositional generalization. arXiv preprint arXiv:1906.00180.

Negishi, M. (1999). Do infants learn grammar with algebra or statistics? Science, 284(5413), Article 435. https://www.science.org/doi/10.1126/science.284.5413.433f

Nie, Y., Wang, Y., & Bansal, M. (2019). Analyzing compositionality-sensitivity of NLI models. Proceedings of the AAAI conference on artificial intelligence (Vol. 33, pp. 6867–6874). AAAI Press.

Nie, Y., Williams, A., Dinan, E., Bansal, M., Weston, J., & Kiela, D. (2020, July). Adversarial NLI: A new benchmark for natural language understanding. In D. Jurafsky, J. Chai, N. Schluter, & J. R. Tetreault (Eds.), Proceedings of the 58th annual meeting of the Association for Computational Linguistics (pp. 4885–4901). Association for Computational Linguistics.

Palm, R., Paquet, U., & Winther, O. (2018). Recurrent relational networks. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), Advances in neural information processing systems (pp. 3368–3378). Curran Associates.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche-Buc, E. B. Fox, & R. Garnett (Eds.), Advances in neural information processing systems (Vol. 32, pp. 8024–8035). Curran Associates. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://dl.acm.org/doi/10.5555/1953048.2078195

Pennington, J., Socher, R., & Manning, C. D. (2014, October). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), Proceedings of the 2014 conference on empirical methods in natural language processing (pp. 1532–1543). Association for Computational Linguistics. http://www.aclweb.org/anthology/D14-1162

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In M. A. Walker, H. Ji, & A. Stent (Eds.), Proceedings of the 2018 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies, volume 1 (long papers) (pp. 2227–2237). Association for Computational Linguistics. http://aclweb.org/anthology/N18-1202

Piantadosi, S. T. (2021). The computational origin of representation. Minds and Machines, 31(1), 1–58. https://doi.org/10.1007/s11023-020-09540-9

Potts, C. (2019). A case for deep learning in semantics: Response to pater. Language, 95(1), e115–e124. https://doi.org/10.1353/lan.2019.0003

Potts, C., Wu, Z., Geiger, A., & Kiela, D. (2021, August). DynaSent: A dynamic benchmark for sentiment analysis. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), Proceedings of the 59th annual meeting of the Association for Computational Linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers) (pp. 2388–2404). Association for Computational Linguistics. https://aclanthology.org/2021.acl-long.186

Premack, D. (1983). The codes of man and beasts. Behavioral and Brain Sciences, 6(1), 125–136. https://doi.org/10.1017/S0140525X00015077

Rabagliati, H., Ferguson, B., & Lew-Williams, C. (2019). The profile of abstract rule learning in infancy: Meta-analytic and experimental evidence. Developmental Science, 22(1), Article e12704. https://doi.org/10.1111/desc.2019.22.issue-1

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*.

Rajpurkar, P., Jia, R., & Liang, P. (2018, July). Know what you don't know: Unanswerable questions for SQuAD. In I. Gurevych & Y. Miyao (Eds.), *Proceedings of the 56th annual meeting of the Association for Computational Linguistics (volume 2: Short papers)* (pp. 784–789). Association for Computational Linguistics. https://www.aclweb.org/anthology/P18-2124

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016, November). SQuAD: 100,000+ questions for machine comprehension of text. In J. Su, X. Carreras, & K. Duh (eds.), *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2383–2392). Association for Computational Linguistics. https://www.aclweb.org/anthology/D16-1264

Raposo, D., Santoro, A., Barrett, D., Pascanu, R., Lillicrap, T., & Battaglia, P. (2017). *Discovering objects and their relations from entangled scene representations*. arXiv preprint arXiv:1702.05068.

Rubenstein, P. K., Weichwald, S., Bongers, S., Mooij, J. M., Janzing, D., Grosse-Wentrup, M., & Schölkopf, B. (2017, August). Causal consistency of structural equation models. In G. Elidan, K. Kersting, & A. T. Ihler (Eds.), *Proceedings of the 33rd conference on uncertainty in artificial intelligence (uai)*. Association for Uncertainty in Artificial Intelligence (AUAI). http://auai.org/uai2017/proceedings/papers/11.pdf

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., & Lillicrap, T. (2018). Relational recurrent neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 7299–7310). Curran Associates.

Santoro, A., Lampinen, A. K., Mathewson, K. W., Lillicrap, T. P., & Raposo, D. (2021). *Symbolic behaviour in artificial intelligence*. CoRR, abs/2102.03406. https://arxiv.org/abs/2102.03406

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., & Lillicrap, T. (2017). A simple neural network module for relational reasoning. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 4967–4976). Curran Associates.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2019). A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, *116*(23), 11537–11546. https://doi.org/10.1073/pnas.1820226116

Seidenberg, M. S., & Elman, J. L. (1999a). Do infants learn grammar with algebra or statistics? *Science*, *284*(5413), 433–433. https://www.science.org/doi/10.1126/science.284.5413.433f

Seidenberg, M. S., & Elman, J. L. (1999b). Networks are not 'hidden rules'. *Trends in Cognitive Sciences*, *3*(8), 288–289. https://doi.org/10.1016/S1364-6613(99)01355-8

Smirnova, A., Zorina, Z., Obozova, T., & Wasserman, E. (2015). Crows spontaneously exhibit analogical reasoning. *Current Biology*, *25*(2), 256–260. https://doi.org/10.1016/j.cub.2014.11.063

Smolensky, P. (1986). Neural and conceptual interpretation of PDP models. In J. L. McClelland, D. E. Rumelhart & the PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Psychological and biological models* (Vol. 2, pp. 390–431). MIT Press.

Thompson, R. K. R., Oden, D. L., & Boysen, S. T. (1997). Language-naive chimpanzees (pan troglodytes) judge relations between relations in a conceptual matching-to-sample task. *Journal of Experimental Psychology: Animal Behavior Processes*, *23*(1), 31–43. https://doi.org/10.1037/0097-7403.23.1.31

Thompson, R. K. R., Rattermann, M. J., & Oden, D. L. (2001). Perception and judgement of abstract same-different relations by monkeys, apes and children: Do symbols make explicit only that which is implicit? *Hrvatska revija za rehabilitacijska istraživanja*, *37*(1), 9–22. https://hrcak.srce.hr/100732

Walker, C. M., Bridgers, S., & Gopnik, A. (2016). The early emergence and puzzling decline of relational reasoning: Effects of knowledge and search on inferring abstract concepts. *Cognition*, *156*(4), 30–40. https://doi.org/10.1016/j.cognition.2016.07.008

Wang, M., & Deng, W. (2021). Deep face recognition: A survey. *Neurocomputing*, *429*, 215–244. https://doi.org/10.1016/j.neucom.2020.10.081

Wasserman, E., Castro, L., & Fagot, J. (2017). Relational thinking in animals and humans: From percepts to concepts. In J. Call, G. M. Burghardt, I. M. Pepperberg, C. T. Snowdon & T. Zentall (Eds.), *APA handbook of comparative psychology: Perception, learning, and cognition* (Vol. 2). American Psychological Association.

Webb, T. W., Sinha, I., & Cohen, J. (2021). *Emergent symbols through binding in external memory* [Conference session]. International Conference on Learning Representations, Austria. https://openreview.net/forum?id=LSFCEb3GYU7

Weyde, T., & Kopparti, R. (2018). Feed-forward neural networks need inductive bias to learn equality relations. In A. Grover, P. Varma, F. Sala, S. Holtzen, J. Neville, S. Ermon, & C. Re (Eds.), *Proceedings of the NeurIPS 2018 relation representation learning workshop*. Curran Associates.

Weyde, T., & Kopparti, R. (2019). Modelling identity rules with neural networks. *Journal of Applied Logic*, *6*(4), 745–769. https://doi.org/10.48550/arXiv.1812.02616

Williams, A., Nangia, N., & Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In M. A. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies, volume 1 (long papers)* (pp. 1112–1122). Association for Computational Linguistics. http://aclweb.org/anthology/N18-1101

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, *3*(1), 1–191. https://doi.org/10.1016/0010-0285(72)90002-3

Wu, Z., Kreiss, E., Ong, D. C., & Potts, C. (2021). ReaSCAN: Compositional reasoning in language grounding. In J. Vanschoren & S. Yeung (Eds.), *Proceedings of the neural information processing systems track on datasets and benchmarks*. https://arxiv.org/abs/2109.08994

Yanaka, H., Mineshima, K., Bekki, D., & Inui, K. (2020, July). Do neural models learn systematicity of monotonicity inference in natural language? In D. Jurafsky, J. Chai, N. Schluter, & J. R. Tetreault (Eds.), *Proceedings of the 58th annual meeting of the Association for Computational Linguistics* (pp. 6105–6117). Association for Computational Linguistics.

Zador, A. M. (2019). A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, *10*(1), 1–7. https://doi.org/10.1038/s41467-019-11786-6

(*Appendices follow*)

Appendix A

Model Optimization Details

The feed-forward networks for basic and hierarchical equality were implemented using the multilayer perception from scikit-learn and a cross entropy function was used to compute the prediction error. The recurrent LSTM network for the sequential ABA task was implemented using PyTorch and a mean squared error function was used to compute the prediction error. The networks for pretraining representations and for the hierarchical equality task were also implemented using PyTorch, with cross-entropy loss functions used to compute the prediction errors.

For all models, Adam optimizers (Kingma & Ba, 2014) were used. For all models, we used a batch size of 1 and ran a hyperparameter search over learning rate values of {0.00001, 0.0001, 0.001} and l2 normalization values of {0.0001, 0.001, 0.01} for each hidden dimension and input dimension. We considered hidden dimensions of {2, 10, 25, 50, 100} and input dimensions of {2, 10, 25, 50, 100}. In the main text, we graph results for the single best hyperparameter setting, averaged over 20 runs with different random model and input initializations. See Additional Results Plots section below for details on how model performance is affected by changes in hidden dimensionality and input dimensionality.

Appendix B

An Analytic Solution to Identity With a Feed-Forward Network

We now show that our feed-forward networks with one hidden layer can solve the same–different problem we pose, in the following sense: for any set of inputs, we can find parameters θ that perfectly classify those inputs. At the same time, we also show that there are always additional inputs for which θ makes incorrect predictions.

Here are the parameters of a feed-forward neural network that performs a binary classification task

$$
\text{ReLu}\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} W^{11} & W^{12} \\ W^{21} & W^{22} \end{pmatrix}\right)\begin{pmatrix} v^{11} & v^{12} \\ v^{21} & v^{22} \end{pmatrix} + (b_1 \quad b_2)
$$
$$
= (o_1 \quad o_2),
$$

where, if n is the dimension of entity embeddings used, then

$$
x, y, v^{11}, v^{12}, v^{21}, v^{22} \in \mathbb{R}^{n \times 1},
$$
$$
W^{11}, W^{12}, W^{21}, W^{22} \in \mathbb{R}^{n \times n},
$$
$$
b_1, b_2, o_1, o_2 \in \mathbb{R}.
$$

Given an input (x_1, x_2), if the output o_1 is larger than o_2, then one class is predicted; if the output o_2 is larger than o_1, then the other class is predicted. When the two outputs are equal, the network has predicted that both classes are equally likely and we can arbitrarily decide which class is predicted. In this case, the output o_1 predicts the two inputs, x_1 and x_2, are in the identity relation and the output o_2 predicts the two inputs are not.

Now, we specify parameters to provide an analytic solution to the identity relation using this network:

$$
\text{ReLu}\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} I & -I \\ -I & I \end{pmatrix}\right)\begin{pmatrix} \vec{1} & \vec{0} \\ \vec{1} & \vec{0} \end{pmatrix} + (b_1 \quad b_2) = (o_1 \quad o_2),
$$

where I is the identity matrix, $-I$ is the negative identity matrix, and $\vec{1}$ and $\vec{0}$ are the two vectors in $\mathbb{R}^n$ that have all zeros and all ones, respectively. The output values, given an input, are

$$
o_1 = \sum_{i=1}^{n} |(x_1)_i - (x_2)_i| + b_1 \qquad o_2 = b_2,
$$

where two parameters are left unspecified, b_1, b_2. We present a visualization in Figure B1 of how the analytic solution to identity of this network changes depending on the values of two bias terms. In this example, the network receives two one-dimensional inputs, x_1 and x_2. If the ordered pair of inputs is in the shaded area on the graph, then they are predicted to be in the identity relation. If in the unshaded area, they are predicted not to be. The dotted line is where the network predicts the two classes to be equally likely.

Figure B1

A Visual Representation of How the Analytic Solution to Identity of a Single Layer Feed-Forward Network Changes Depending on the Values of Two Bias Terms, b_1, b_2
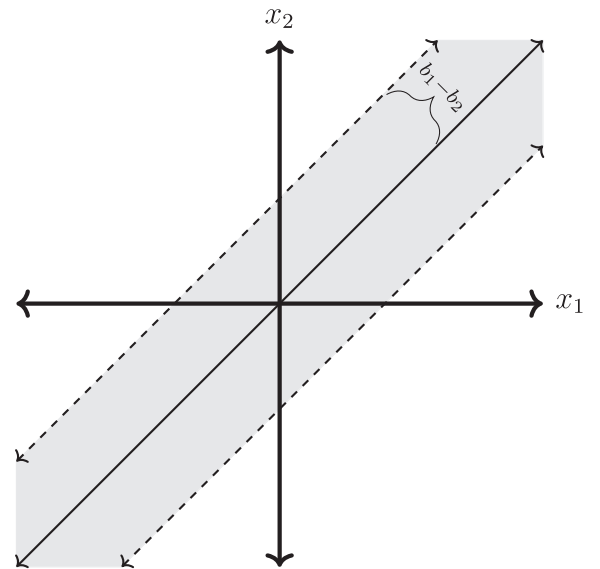
The network predicts x_1 and x_2 to be in the identity relation if $\sum_{i=1}^{n} |x_i - y_i| < b_1 - b_2$ which is visualized as the points between two parallel lines above and below the solution line $x_1 = x_2$. As the difference $b_1 - b_2$ gets smaller and smaller, the two lines that bound the network's predictions get closer and closer to the solution line. However, as long as $b_1 - b_2$ is positive, there will always be inputs of the form $(r, r + (b_1 - b_2)/2)$ that are false positives. For any set of inputs, we can find bias values that result in the network correctly classifying those inputs, but for any bias values, we can find an input that is incorrectly classified by those values. In other words, we have an arbitrarily good solution that is never perfect. We provide a proof below that there is no perfect solution and so this is the best outcome possible. However, if we were to decide that, if the network predicts that an input is equally likely in either class, then this input is predicted to be in the identity relation, we could have a perfect solution with $b_1 = b_2$.

Here is proof that a perfect solution is not possible. A basic fact from topology is that the set $\{x: f(x) < g(x)\}$ is an open set if f and g are continuous functions. Let N_{o1} and N_{o2} be the functions that map an input (x_1, x_2) to the output values of the neural network, o_1 and o_2, respectively. These functions are continuous. Consequently, the set $C = \{(x_1, x_2): N_{o2}(x_1, x_2) < N_{o1}(x_1, x_2)\}$, which is the set of inputs that are predicted to be in the equality relation, is open.

With this fact, we can show that, if the neural network correctly classifies any point on the solution line $x_1 = x_2$, then it must incorrectly classify some point not on the solution line. Suppose that C contains some point (x, x). Then, by the definition of an open set, C contains some ε ball around (x, x), and therefore C contains $x, x + \epsilon)$, which is not on the solution line $x_1 = x_2$. Thus, C can never be equal to the set $\{(x_1, x_2): x_1 = x_2\}$. So, because C is the set of inputs classified as being in the equality relation by the neural network, a perfect solution cannot be achieved. Thus, we can conclude our arbitrarily good solution is the best we can do.

Appendix C

An Analytic Solution to ABA Sequences

Here are the parameters of a long short-term memory recurrent neural network (LSTM):

$$f_t = \sigma(x_t W_f + h_{t-1} U_f + b_f),$$

$$i_t = \sigma(x_t W_i + h_{t-1} U_i + b_i),$$

$$o_t = \sigma(x_t W_o + h_{t-1} U_o + b_o),$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \text{ReLU}(x_t W_c + h_{t-1} U_c + b_c),$$

$$h_t = o_t \circ \text{ReLU}(c_t),$$

$$y_t = h_t V,$$

where, if n is the representation size and d is the network hidden dimension, then

$$x_t \in \mathbb{R}^n, f_t, i_t, o_t, h_t, c_t \in \mathbb{R}^d,$$

$$W \in \mathbb{R}^{n \times d}, U \in \mathbb{R}^{d \times d},$$

$$V \in \mathbb{R}^{d \times n}, b \in \mathbb{R}^d,$$

and σ is the sigmoid function. The initial hidden state h_0 and initial cell state c_0 are both set to be the zero vector. We say that an LSTM model with specified parameters has learned to produce ABA sequences if the following holds: when the network is seeded with some entity vector representation as its first input, x_1, then the output y_1 is not equal to x_1 and at the next time step the output y_2 is equal to x_1.

We let $d = 2n + 1$ and assign the following parameters, which provide an analytic solution to producing ABA sequences:

$$f_t = \sigma(x_t 0_{n \times d} + h_{t-1} 0_{d \times d} + N_d),$$

$$i_t = \sigma\left(x_t 0_{n \times d} + h_{t-1} \begin{bmatrix} -4 \dots -4 \\ 0_{2n \times n} \end{bmatrix} + N_d\right),$$

$$o_t = \sigma\left(x_t 0_{n \times d} + h_{t-1} \begin{bmatrix} 1 \dots 1 \\ 0_{2n \times n} \end{bmatrix} + 0_d\right),$$

$$c_t = f_t \circ c_{t-1} +$$

$$i_t \circ \text{ReLU}\left(x_t \begin{bmatrix} 0 \\ \vdots & -I_{n \times n} & I_{n \times n} \\ 0 \end{bmatrix} + h_{t-1} 0_{d \times d} + \begin{bmatrix} N & 0 \dots 0 \end{bmatrix}\right),$$

$$h_t = o_t \circ \text{ReLU}(c_t),$$

$$y = h_t \begin{bmatrix} 0 \dots 0 \\ -I_{n \times n} \\ I_{n \times n} \end{bmatrix},$$

where $0_{j \times k}$ is the $j \times k$ zero matrix, $\mathbf{m}_k$ is a k dimensional vector with each element having the value m, $I_{n \times n}$ is the $n \times n$ identity matrix, and N is some very large number. Now, we show that these parameters achieve an increasingly good solution as N increases. When a value involves the number N, we will simplify the computation by saying what that value is equal to as N approaches infinity. We begin with an arbitrary input x_1 and the input and hidden state initialized to zero vectors:

$$h_0 = 0_d \qquad c_0 = 0_d.$$

The gates at the first time step are easy to compute, as the cell state and hidden state are zero vectors so the gates are equal to the sigmoid function applied to their respective bias vectors. The forget gate is completely open, the output gate is partially open, and the input gate is fully open:

$$f_1 = \sigma(N_d) \approx 1_d,$$

$$o_1 = \sigma(0_d) = 0.5_d,$$

$$i_1 = \sigma(N_d) \approx 1_d.$$

(*Appendices continue*)

Then, we compute the cell and hidden states at the first timestep. The cell state encodes the information of the input vector, so it can be used to recover the vector at a later time step and receives no information from the previous cell state despite the forget gate being open, because the previous cell state is a zero vector. The hidden state is the cell state scaled by one half.

$$c_1 = 1_d \circ 0_d +$$

$$1_d \circ \mathrm{ReLU}\left(x_1 \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad -I_{n\times n} \quad I_{n\times n} \end{bmatrix} + \begin{bmatrix} N & 0\ldots0 \end{bmatrix}\right),$$

$$= \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right),$$

$$h_1 = 0.5_d \,\mathrm{ReLU}\left(\mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right)\right),$$

$$= 0.5_d \circ \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right).$$

At the next time step, the forget gate remains fully open, the output gate changes from partially open to fully open, and the input gate changes from fully open to fully closed:

$$f_2 = 1_d,$$

$$o_2 = \sigma\left(x_2\, 0_{n\times d} + h_1 \begin{bmatrix} 1\ldots1 \\ 0_{2n\times n} \end{bmatrix} + 0_d\right),$$

$$= \sigma\left(0.5_d \circ \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right)\begin{bmatrix} 1\ldots1 \\ 0_{2n\times n} \end{bmatrix} + 0_d\right),$$

$$= \sigma\left(0.5_d \circ N_d\right) \approx 1_d,$$

$$i_2 = \sigma\left(x_2\, 0_{n\times d} + h_1 \begin{bmatrix} -4\ldots-4 \\ 0_{2n\times n} \end{bmatrix} + N_d\right),$$

$$= \sigma\left(0.5_d \circ \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right)\begin{bmatrix} -4\ldots-4 \\ 0_{2n\times n} \end{bmatrix} + N_d\right),$$

$$= \sigma\left(0.5_d \circ -4N_d + N_d\right) \approx 0_d.$$

Then, we compute the cell and hidden states for the second timestep. Because the forget gate is completely open and the input gate is completely closed, the cell state remains the same. Because the output gate is completely open, the hidden state is the same as the cell state.

$$c_2 = 1_d \circ \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right) +$$

$$0_d \circ \mathrm{ReLU}\left(x_2 \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad -I_{n\times n} \quad I_{n\times n} \end{bmatrix} + \begin{bmatrix} N & 0\ldots0 \end{bmatrix}\right),$$

$$= \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right),$$

$$h_2 = 1_d \circ \mathrm{ReLU}\left(\mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right)\right),$$

$$= \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right).$$

With the hidden states for the first and second time steps, we can compute the output values and find that the output at the first time step is the initial input vector scaled by one half and the output at the second time step is the initial input vector.

$$y_1 = h_1 \begin{bmatrix} 0\ldots0 \\ -I_{n\times n} \\ I_{n\times n} \end{bmatrix} = 0.5_d \circ \mathrm{ReLU}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right)\begin{bmatrix} 0\ldots0 \\ -I_{n\times n} \\ I_{n\times n} \end{bmatrix},$$

$$= 0.5_d \circ x_1,$$

$$y_2 = h_2 \begin{bmatrix} 0\ldots0 \\ -I_{n\times n} \\ I_{n\times n} \end{bmatrix} = \mathrm{ReLu}\left(\begin{bmatrix} N & -x_1 & x_1 \end{bmatrix}\right)\begin{bmatrix} 0\ldots0 \\ -I_{n\times n} \\ I_{n\times n} \end{bmatrix} = x_1.$$

Then, because $y_1 = 0.5_d \circ x_1 \neq x_1$ and $y_2 = x_1$, this network produces ABA sequences.

Appendix D

Additional Results Plots

Experiment 1 Results for Different Hidden Dimensionalities

Figure D1 explores a wider range of hidden dimensionalities for our Experiment 1 model (feed-forward network with a single hidden layer; Equations 4 and 5) applied to the basic same–different task. The lines correspond to different embedding dimensionalities.

Experiment 2 Results for Different Hidden Dimensionalities

Figure D2 explores a wider range of hidden dimensionalities for our Experiment 2 model (LSTM; Equations 8 and 9) applied to the sequential ABA task. The lines correspond to different embedding dimensionalities. The full training set is presented to the model in multiple epochs.
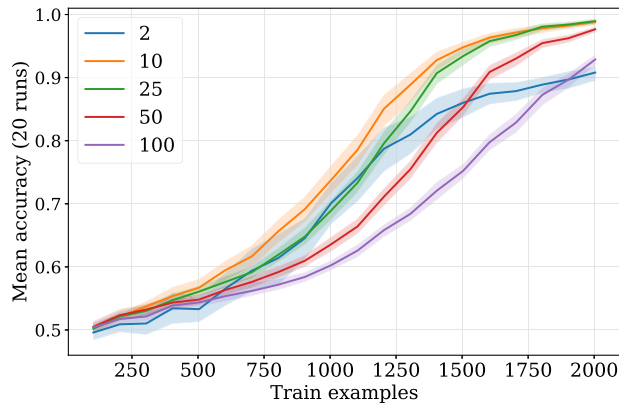
Figure D1

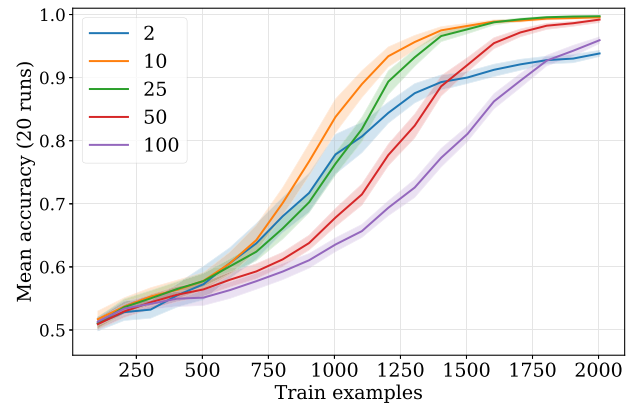*Experiment 1 Model for Basic Same–Different*
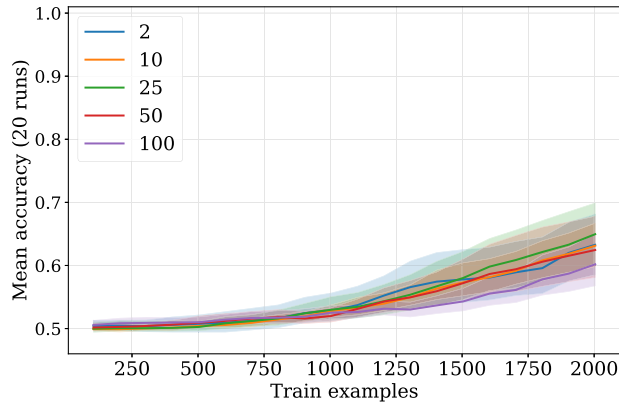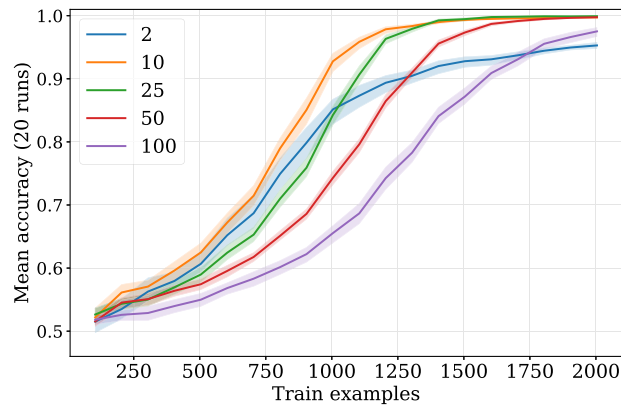

(a) Hidden dimensionality 2.
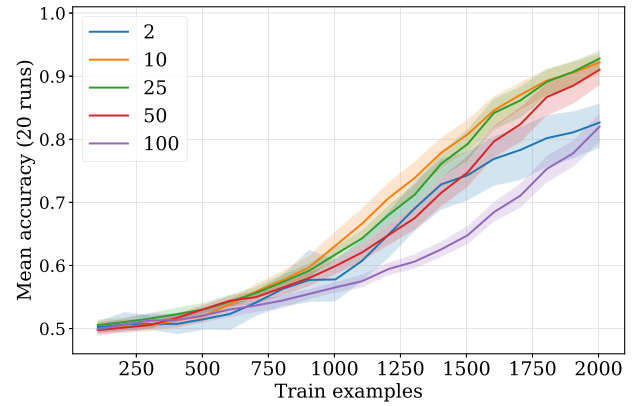
(b) Hidden dimensionality 10.
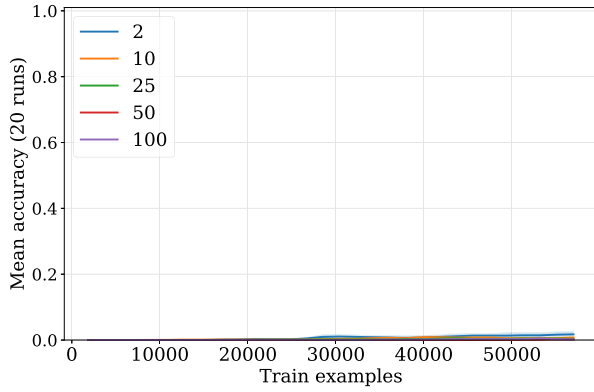
(c) Hidden dimensionality 25.
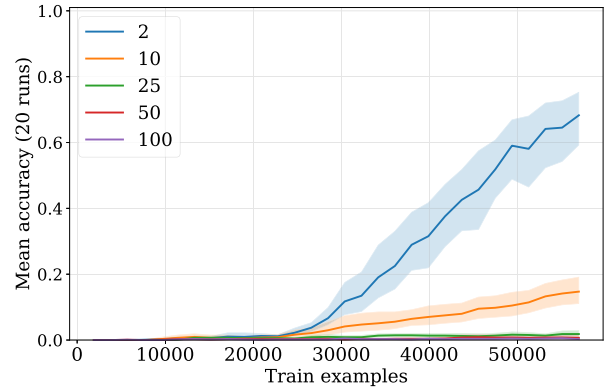
(d) Hidden dimensionality 50.

(e) Hidden dimensionality 100.

Note. Lines correspond to different input dimensions. These models were provided random input representations. See the online article for the color version of this figure.
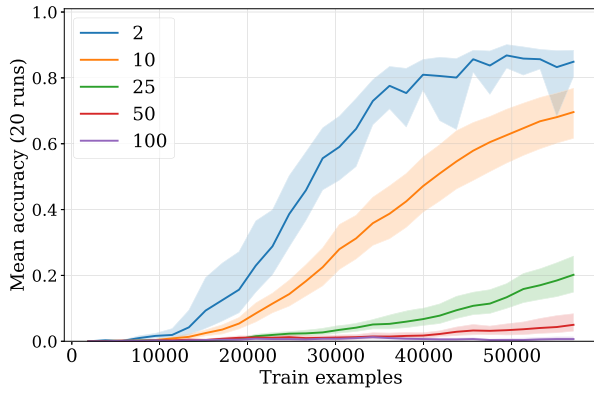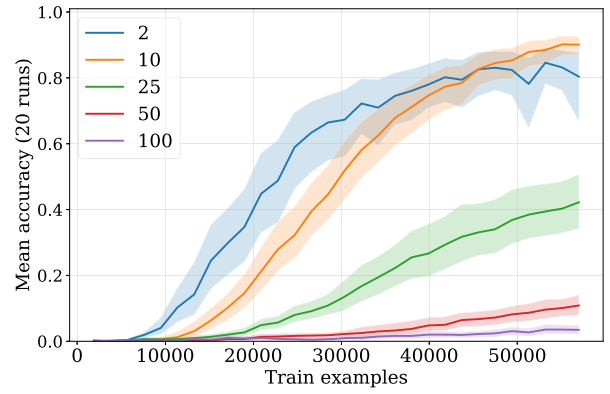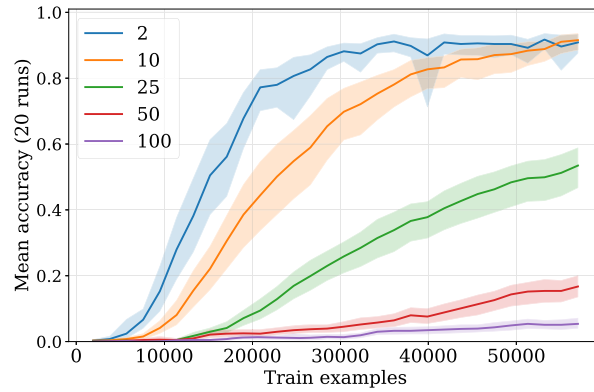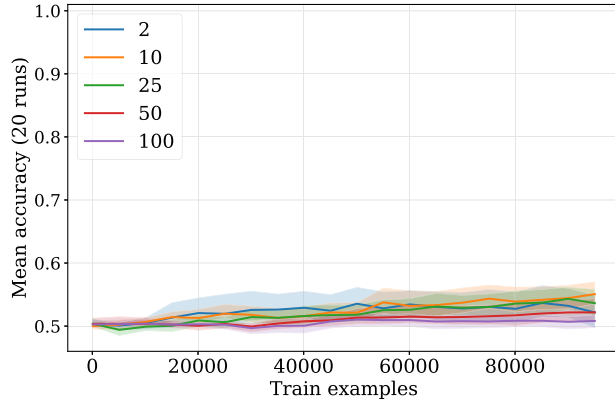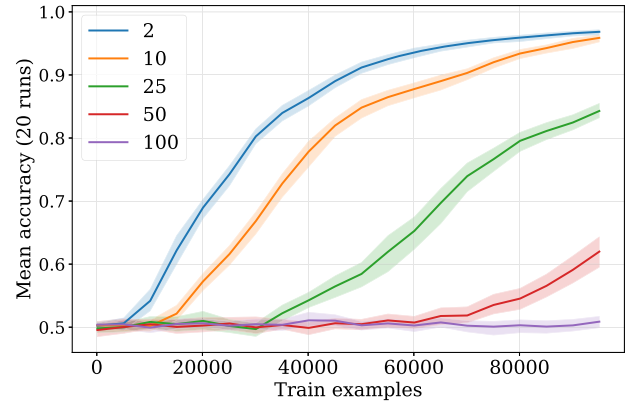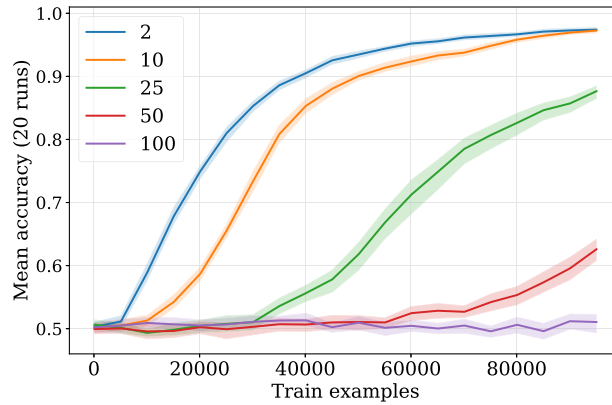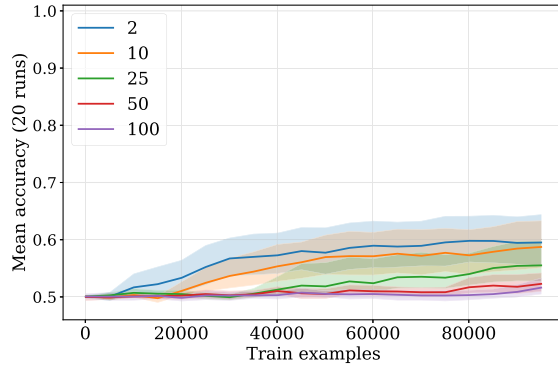
Figure D2

Experiment 2 Model for the Sequential ABA Task, With a Vocabulary Size of 20

(a) Hidden dimensionality 2.

(b) Hidden dimensionality 10.

(c) Hidden dimensionality 25.

(d) Hidden dimensionality 50.

(e) Hidden dimensionality 100.

Note. Lines correspond to different input dimensions. These models were provided random input representations. See the online article for the color version of this figure.

Experiment 1 Model Applied to Hierarchical Same–Different

Figure D3 shows the results of applying the Experiment 1 model (Equations 4 and 5) to the hierarchical same–different task. The only change from that model is that the inputs have dimensionality 4 m, since the four distinct representations in task inputs are simply concatenated. The lines correspond to different embedding dimensionalities.
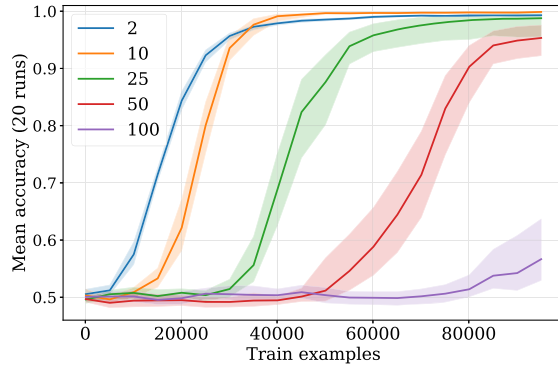
Figure D3

Experiment 1 Model Applied to the Hierarchical Same–Different Task
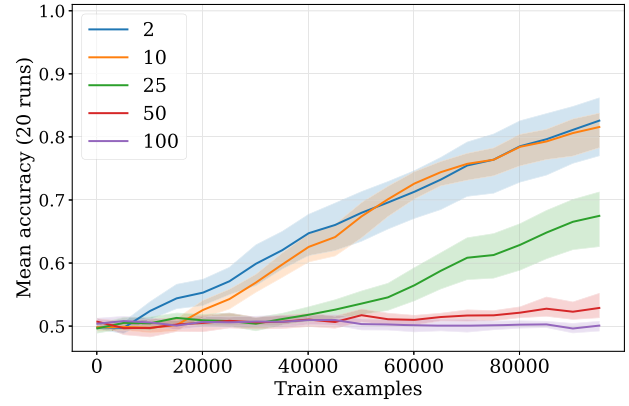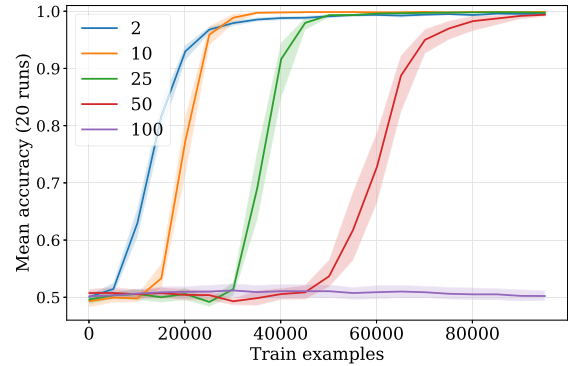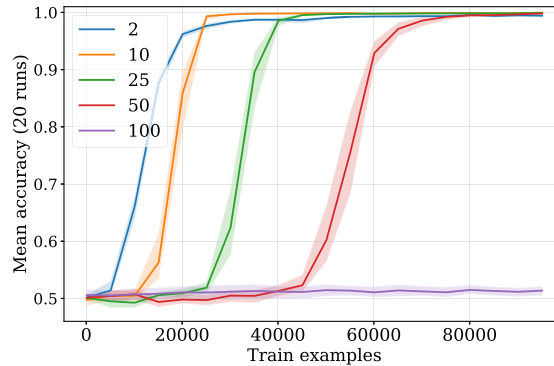

(a) Hidden dimensionality 2.
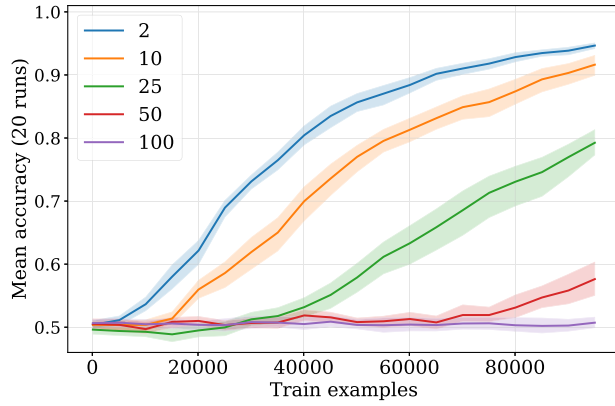
(b) Hidden dimensionality 10.
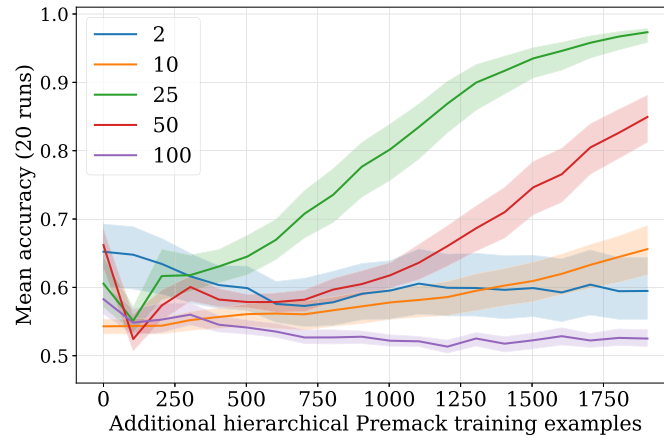
(c) Hidden dimensionality 25.

(d) Hidden dimensionality 50.

(e) Hidden dimensionality 100.

Note. Lines correspond to different input dimensions. These models were provided random input representations. See the online article for the color version of this figure.

Experiment 3 Results for Different Hidden Dimensionalities

Figure D4 shows the results of applying our Experiment 3 model (feed-forward network with two hidden layers; Equations 11–13) to the hierarchical same–different task. The lines correspond to different embedding dimensionalities.

Experiment 4 Results for Different Input Dimensionalities

Figure D5 shows the results of applying our Experiment 4 model (Equations 14–17) to the hierarchical same–different task. In this model, the hidden dimensionality is required to be twice the input dimensionality, due to the way the components are the model are reused in a hierarchical fashion.

Figure D4
Experiment 3 Model Applied to the Hierarchical Same–Different Task
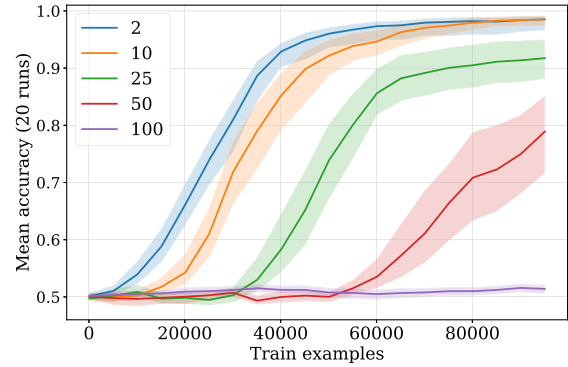
(a) Hidden dimensionality 2.

(b) Hidden dimensionality 10.

(c) Hidden dimensionality 25.

(d) Hidden dimensionality 50.

(e) Hidden dimensionality 100.

Note. Lines correspond to different input dimensions. These models were provided random input representations. See the online article for the color version of this figure.

Figure D5

Experiment 4 Model Applied to the Hierarchical Same–Different Task

Note. Lines correspond to different input dimensionality, which determine the hidden dimensionality. These models were provided random input representations. See the online article for the color version of this figure.

Experiment 4 Results by Input Class

The hierarchical same–different task intuitively has four subclasses corresponding to different inputs:

- same/same: "positive"
- different/different: "positive"
- same/different: "negative"
- different/same: "negative"

Figure D6 depicts model accuracy for each of these input classes, for the "no pretraining" model depicted in Figure 7B, which uses 25-dimensional input representations. It is noteworthy that the different/different class is the most difficult for the model.

Performance on Model Train Sets

Figure D7 presents experimental results comparable to each of our main experiment figures, but with the evaluations now done on the training data.

Figure D6

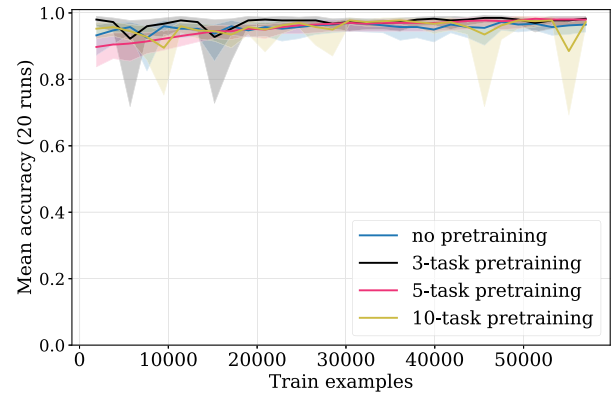*Experiment 4 Model Performance Broken Down by Input Class*
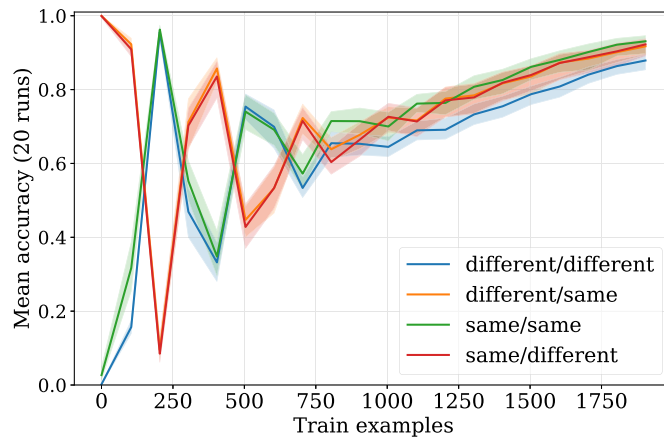
Note. See the online article for the color version of this figure.
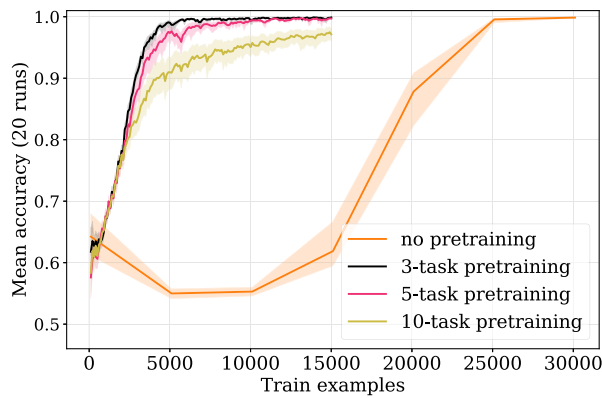
Figure D7

Training Set Results for the Networks Used in Our Primary Assessments in the Main Text
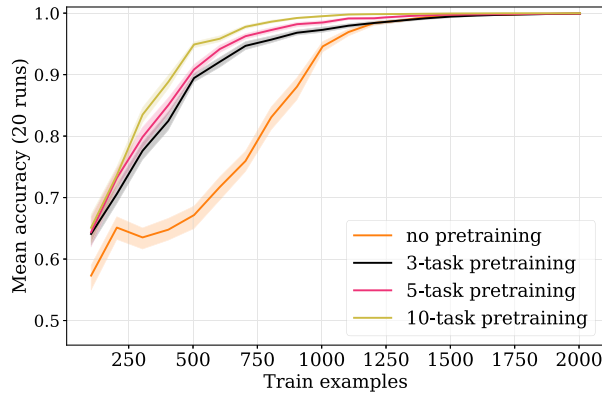
(a) Experiment 1 training set results.

These are qualitatively very much like our

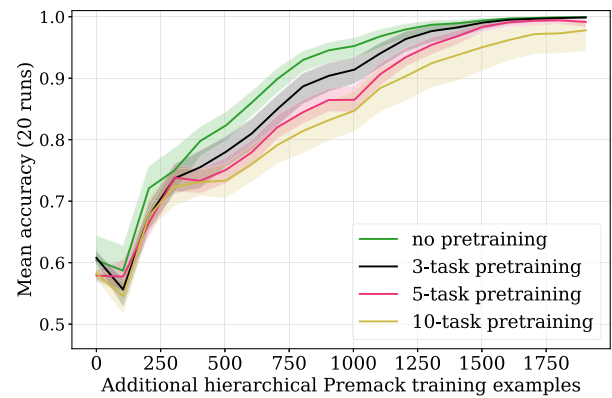held-out assessment results (Figure 5).

(b) Experiment 2 training set results,

indicating that these models very quickly

memorize the training data, creating a

large gap between training performance

and test performance (cf. Figure 6).

(c) Experiment 3 training set results.

These are qualitatively very much like our

held-out assessment results (Figure 7a).

(d) Experiment 4 training set results.

These are very similar to our held-out

assessments (Figure 7b), with one

noteworthy difference: pretraining provides

small improvements in the held-out

assessments, whereas 'no pretraining'

clearly provides the fastest learning in the

training-set evaluations

Note. See the online article for the color version of this figure.