Inductive Bias Is in the Eye of the Beholder

Michael Wilson[1] **Robert Frank**[2]
[1] Department of Linguistics & Cognitive Science, University of Delaware
[2] Department of Linguistics, Yale University
mawilson@udel.edu bob.frank@yale.edu

Abstract

Due to the finite nature of any evidence used in learning, systematic generalization is crucially reliant on the presence of inductive bias (Mitchell, 1980). We examine inductive biases in different types of sequence-to-sequence neural network models, including CNNs, LSTMs (with and without attention), and transformers, inspired by Kharitonov and Chaabouni (2021). Crucially, however, we consider a wider range of possible inductive biases than their study did. Investigating preferences for hierarchical generalization compared to other types of generalization, we find that, contrary to their results, transformers display no preference for hierarchical generalization, but instead prefer a counting strategy. We also investigate biases toward different types of compositionality. By controlling for a confound in Kharitonov and Chaabouni (2021)'s test set, we find much less consistent generalization overall, and find that a large number of responses were among types other than the two types of generalization they considered. Nevertheless, we observe consistent compositional generalization to held out combinations of primitives and functions on a SCAN task (Lake and Baroni, 2017) by models of all types, but only when primitives occur with other functions in the training set. The pattern of success indicates generalization in models of these types is highly sensitive to distributional properties of their training data.

1 Introduction

Learners, both human and machine, systematically generalize from finite sets of data, and it is such generalization that makes them such effective agents. Hupkes et al. (2023a,b) review the wealth of work focused on understanding the efficacy of different models for different types of generalization in NLP tasks. As Mitchell (1980) notes, the kind of systematic generalization we hope our models will show is only possible in the presence of *inductive bias*, a preference for some generalization over others. Inductive bias can derive from inherent properties of a model or from previous training. In this paper, we focus on the former. While such inherent inductive bias can be read off to a reasonable degree from the structure of a symbolic model, it is much less easy to understand the biases of a neural network architecture trained with some variant of backpropagation.

Work that documents variation among models in their ability to solve a certain NLP task can be understood as illuminating inductive biases in these models: ones that fare better are more biased toward the correct solution (modulo training differences). Yet because of the complexity of most such tasks, it is typically difficult to identify the specific preferences that lead to a model's generalization behavior. The recent work of Kharitonov and Chaabouni (2021, henceforth KC) aims to avoid this issue by focusing on carefully controlled induction problems, which we can think of as "model induction organisms," where the range of solutions is limited. We aim to build on KC's important groundwork, focusing on two of the tasks that they proposed: Hierarchical-or-Linear and Composition-or-Memorization. We show that the evaluation of model behavior and assessment of inductive bias with even apparently trivial tasks requires great care. In particular, assessing inductive bias requires the consideration of the widest possible range of possible hypotheses, and failing to do so can lead to over- or under-estimating inductive bias for a certain type of hypothesis. In addition, we demonstrate that the process of constructing such model organism tasks must avoid the presence of quirks that lead even appropriately biased models astray.

2 Experiment 1: Hierarchy vs. Counting

In experiment 1, we adapt KC's **hierarchical-or-linear** task to further explore the question of whether any of a range of model architectures displays a bias toward hierarchical generalizations.

Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP, pages 152–162

2.1 Materials & methods

In this task, we train sequence-to-sequence models on four example mappings of the form $x^d y x^d \rightarrow y$, where $x, y \in \{a, b\}$ and $d = 4$. This describes the following four pairs of inputs and outputs.

1. Input: *aaaaaaaaa*; Output: *a*
2. Input: *aaaabaaaa*; Output: *b*
3. Input: *bbbbbbbbb*; Output: *b*
4. Input: *bbbbabbbb*; Output: *a*

As KC observe, this training set is consistent with multiple rules characterizing the mapping between inputs to outputs. A **hierarchical** rule could assign to the input a center-embedded structure that associates matching symbols in the prefix and suffix, so that the target output is the most deeply embedded element, i.e., the middle symbol. A **linear** rule instead identifies the output through its absolute sequential position in the source, in this case the fifth symbol. A third rule that KC do not consider involves a **counting** strategy, where the output is the symbol that occurs least frequently, but at least once, in the source.

KC's test set went beyond this training set to include inputs of the form $x^m y x^m$ for $m \in [2, 6]$. This set of inputs allows the hierarchical and linear rules to be distinguished: the former would yield output *y*, while the latter would yield whatever element occurs in fifth position. This set does not, however, allow the counting rule to be distinguished from the hierarchical rule, which would both predict output *y*. As a result, our testing regime evaluated models on outputs of the form $x^m y x^n$, where $m \geq 0$, $n \geq 0$, and $m + n = d$, for $d \in [2, 6]$, which includes strings like *abbbb*, *babbb*, *bbabb*, etc. To see how this expanded test set distinguishes hierarchical, linear, and counting rules, consider the input *abaaa*. Both hierarchical and linear learners would produce *a* (which occurs both as the middle and the fifth symbol in this string). However, a learner with a bias toward counting would produce *b*, the least frequently occurring symbol in the input. On the other hand, an example like *aabaa* would lead hierarchical- and counting-biased learners to produce *b* (the middle and least frequent symbol), while linear learners would produce *a* (the fifth symbol).

We consider two measures of performance. The first we adopt from KC: "fraction of perfect agreement" (FPA). We train and evaluate 100 models of each architecture we consider with different random initial states. FPA is the proportion of these models for which *all* outputs adhere to a particular rule, whether hierarchical, linear, or counting.

Dataset	Architecture	FPA Counting	FPA Linear	FPA Hierar.	PrAg with counting (100 seeds/row)
KC H-or-L	LSTM (w/o attn.)	0.00	0.00	0.00	
	LSTM (w/ attn.)	0.00	0.00	0.00	
	CNN	0.00	0.84	0.00	
	Transformer	0.79	0.00	0.00	
Mirror CFG	LSTM (w/o attn.)	0.00	–	0.00	
	LSTM (w/ attn.)	0.00	–	0.00	
	CNN	0.00	–	0.00	
	Transformer	1.00	–	0.00	
Mirror CFG (brackets)	LSTM (w/o attn.)	0.00	–	0.00	
	LSTM (w/ attn.)	0.00	–	0.00	
	CNN	0.00	–	0.74	
	Transformer	0.70	–	0.00	
Mirror PCFG	LSTM (w/o attn.)	0.00	–	0.00	
	LSTM (w/ attn.)	0.00	–	0.00	
	CNN	0.00	–	0.00	
	Transformer	0.95	–	0.00	
Mirror PCFG (brackets)	LSTM (w/o attn.)	0.00	–	0.00	
	LSTM (w/ attn.)	0.00	–	0.00	
	CNN	0.00	–	0.55	
	Transformer	0.18	–	0.00	

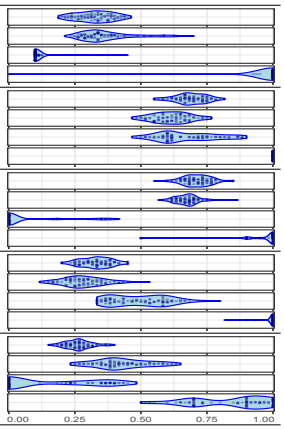

Table 1: Results of experiment 1 and follow-up experiments. Overlapping points are jittered on the *y*-axis.

Our second measure is the proportion of agreement (PrAg) with a particular generalization for individual examples in our test set in a single model. Note that for this task, no example can be entirely unambiguous due to the simplicity of the training language: if one response unambiguously signals the counting generalization (e.g., $baaaa \rightarrow b$), then the other possible response, *a*, is compatible with both the linear and hierarchical generalizations. For this reason, for this initial task, we provide plots showing only the PrAg with the counting generalization as compared to the hierarchical generalization out of examples where the two would predict different responses.

We train the same types of networks as KC: CNNs, LSTMs (with and without attention), and transformers, using the same hyperparameters; see KC for network architecture and training details.[1]

2.2 Results

Results for experiment 1 are shown in the first set of rows in table 1 (under KC H-or-L). Notably, no model consistently generalizes in accordance with the hierarchical rule on our test set. While we observe, like KC, that CNNs display an inductive bias toward linear generalizations, we find quite different results for transformers: rather than a hierarchical bias, it appears that they actually display a bias toward counting or determination of majority

[1]Our data and code are available at github.com/ma-wilson1234/FIND.

(Merrill and Sabharwal, 2023), revealed by examining performance on a test set that distinguishes these competing possibilities.

The PrAg results reflect this bias for CNNs and Transformers. LSTMs show a moderate but not overwhelming proportion of counting responses—less than might have been expected from the results of Weiss et al. (2018). However, the plots in the table of necessity only present PrAg with the counting generalization for examples whose possible responses distinguish counting, on the one hand, from hierarchical or linear strategies, on the other. This means these plots alone do not reveal whether LSTMs tend to produce more linear or hierarchical responses in cases that could distinguish those two possibilities. When we consider such examples, we in fact find that LSTMs of both types produce more responses consistent with the hierarchical strategy compared to the linear strategy (50%ile for hierar. responses for LSTMs w/o attn.: 0.875; LSTMs w/ attn.: 0.75), which echoes what KC found using their description length-based measure.

2.3 Attempts to induce hierarchical generalization

The results of experiment 1 indicate that no model architecture exhibits a strong bias toward hierarchical generalizations when we evaluate behavior against a wider space of hypotheses. However, the success of various model architectures, especially transformers, on linguistic data raises the question of whether a richer kind of training data could be sufficient to induce such a bias. In particular, language is replete with rules and constraints that make crucial reference to hierarchical structure. The fact that large pre-trained transformer models have shown general success on tasks probing their sensitivity to such structure clearly shows us that they are able to learn hierarchical generalizations (Mueller et al., 2022). Such large pre-trained models receive input much richer than the four examples given to the models in experiment 1.

Rather than attempting to replicate this full richness, we ask instead whether three changes that begin to approach the ways in which language is richer than the original four-example training set could suffice to induce hierarchical generalizations as opposed to counting generalizations.

Variable length We enrich our training set, so that it is now described by the following recursive phrase structure rules:

- $S \rightarrow a\ S'\ a \mid b\ S'\ b$
- $S' \rightarrow a\ S'\ a \mid b\ S'\ b \mid a \mid b$

We refer to this as the "Mirror CFG" set. We cap the maximum length of an example to 11, and randomly generate 2 examples for each length in {3, 5, 7, 9, 11} for each center symbol *a* or *b*.[2] We also include two instances of each of these lengths where all symbols are identical. For examples where symbols were not all identical, we ensured that the center symbol was also the least frequently occurring symbol for each example. This creates a set of 40 sentences. We train models using the same hyperparameters as above, with all 40 examples run as a single batch. Note that including differing example lengths means that a linear hypothesis, in which an element at a fixed position is output, is no longer compatible with the training set.

Our test set consists of all examples of lengths {3, 5, 7, 9, 11} for which the two possible responses distinguish between the counting and hierarchical strategies (i.e., for which the middle symbol and the least frequent symbol are different). For the test set, we ensured that the center symbol was always the *most* frequently occurring symbol, which will allow us to distinguish the hierarchical and linear generalization strategies. Because of this property of our test items, a model with low PrAg score for counting will have a correspondingly high score for the hierarchical hypothesis. Results are shown in the second row of table 1. Both FPA and PrAg measures indicate that all architectures show a bias toward counting, transformers most strongly.

Statistical signature of recursion Next, we consider a dataset generated by a probabilistic version of the grammar above, where the probability of recursion is 0.5, which we call the "Mirror PCFG" set. This provides information not only about examples of different lengths, but also about the relative frequencies of different lengths. The resulting geometric distribution over example lengths is consistent with generation by a recursive process: at each point in generation the structure can either recurse with probability with p or stop with probability $1-p$, resulting in structures with k levels of recursion being generated with probability $p^k(1-p)$. If each step of recursion introduces a fixed amount of material, this will yield a geometric distribution on string lengths. We cap examples to at most length

[2]The 2 examples per length were randomly generated only once, and reused for every architecture/seed.

11, and otherwise randomly generate 10 examples for each center symbol in {a, b}, and 10 examples where all symbols are identical for each symbol in {a, b} (40 total). We train the same way as before, and use the same Mirror CFG test set.

Results are again shown in table 1. Interestingly, the CNNs and LSTMs now show a bias toward hierarchical generalization revealed by the PrAg measure, though none generalized consistently. Nevertheless, transformers remain strongly biased toward the counting generalization.

Explicit Encoding of Structure Finally, we modified the previous two datasets adding left and right brackets to mark constituency (e.g., *babab* becomes $[b[a[b]a]b]$). We also added brackets to the test set. As pointed out by an anonymous reviewer, this means there is now a *linear* way of producing a "hierarchical" response—choose the symbol with a preceding "[" and a following "]". Our goal was to see how far we needed to go to produce any kind of generalization consistent with hierarchical structure—even if such a generalization could also be a linear generalization. In other words, if we make the hierarchical generalization "easier," will the models be more likely to take the bait?

Table 1 shows that adding brackets changes the behavior of the CNNs, which show a bias toward the "hierarchical" generalization. We suspect this sharp change is due to the fact that in the bracketed dataset, the "hierarchical" generalization can now be expressed in linear terms (as detailed above). What we find more interesting is that in spite of the availability of this simple strategy, no substantial change in overall bias is seen for LSTMs or transformers—the transformers remain biased toward a counting strategy, and the LSTMs' preferences do not change.[3] For the constant-frequency training set, the LSTMs retain their counting bias; for the PCFG training set, the LSTMs retain their hierarchical bias without much change. The transformers in both cases retain a bias toward the counting generalization, though it is somewhat less pronounced with the addition of brackets.

2.4 Discussion

Experiment 1 showed that despite KC's claims, transformers do not have a bias toward hierarchical generalizations. When we considered a richer set of possible generalizations, we found that transformers favor counting generalizations over hierarchical generalizations when both are compatible with the input. This shows the importance of considering ambiguities in the hypothesis space when discussing inductive biases.

This result led us to see whether we could enrich the training set to induce a hierarchical bias. We considered two simple changes that could make the simple training set more like human language: first, we considered inputs of different lengths described by a CFG. Second, we considered inputs of different lengths generated by a recursive PCFG so that shorter strings were more frequent than longer strings. Finally, we considered both of these manipulations with the addition of brackets in the input that marked the underlying hierarchical structure.

Two manipulations made a difference. First, going from a uniform distribution over example lengths to a geometric distribution of lengths produced by a PCFG reversed the bias of the CNNs and LSTMs from counting to hierarchical, with the change being more noticeable for LSTMs. Second, adding brackets made CNNs strongly biased toward apparently hierarchical responses, but had relatively little effect on other model types.[4] Most interestingly, despite the high success of transformer-based models on linguistic tasks that require reference to hierarchical structure, we found that none of our manipulations sufficed to induce a hierarchical bias in these models—they remained stubbornly in favor of the counting generalization.

3 Experiment 2: Compositionality

In experiment 2, we examine the question of whether different model architectures display a bias toward compositional generalizations of various types. By "compositional," we mean a process whose output is a function of its individual input symbols and their mode of combination (Szabó, 2022). This question has been explored using a variety of tasks and datasets, e.g., SCAN (Lake and Baroni, 2017) and COGS (Kim and Linzen, 2020a), which both explore the problem of assigning structured semantic interpretations to English sentences. While offering useful measures of compositional generalization, the complexity of these datasets makes it difficult to assess the propensity for com-

[3] See also McCoy et al. (2020) for a similar lack of change in LSTM performance in the face of explicit evidence about hierarchical structure.

[4] As noted above, the presence of the brackets gives rise a non-hierarchical alternative, which the CNN may be exploiting.

positional generalization in its simplest guise. Here, we focus on a distilled task probing compositional generalization proposed by KC, which they call **Composition-or-Memorization**. This task targets compositional generalization in a way that is not specific to its role in natural language.

3.1 Materials & methods

In KC's **Composition-or-Memorization** task, models are trained on two types of examples. In one type, input symbols (which we represent as natural numbers) are presented in isolation in the input and are mapped to corresponding output symbols (which we represent as lower case letters) in a one-to-one way. Thus an input symbol a would be mapped to the corresponding output symbol A (which we encode as the corresponding upper case letters). The examples are "non-compositional examples". For a second type of input, there is a modifier, F, which precedes one of these input symbols. Under the intended interpretation of our dataset, F is interpreted like "thrice," so that input $F\ a$ is mapped to a three copies of the corresponding output symbol, namely $A\ A\ A$ ("compositional examples"). For experiment 2, we consider the same model architectures as in experiment 1, and train 100 random seeds/architecture.

KC vary the number of input symbols that are presented in compositional form in the training data, and consider how models perform on held-out compositional examples. They define M as the number of distinct compositional examples in the training set, and N as the number of distinct input (and output) symbols that occur in non-compositional examples in the training set (in all cases, such symbols are a superset of the ones that occur in compositional examples). They consider $N = 40$ with $M \in \{6, 24, 36\}$; that is, they train learners on all 40 non-compositional examples and M compositional examples.[5] Their test set consists of all unobserved compositional examples within the 40 symbol range. For example, when $M = 36$, their test set consists of the four inputs $F\ x_{37}$, $F\ x_{38}$, $F\ x_{39}$, $F\ x_{40}$; when $M = 24$, their test set consists of sixteen inputs, and so on.

A problem with this approach is that it makes it easier for a model to show perfect agreement with a generalization as the number of training examples increases, because this corresponds to the size of the test set decreasing. In other words, a model trained with $M = 36$ need only consistently generalize on 4 examples to be counted as perfectly agreeing with a particular generalization. A model trained with $M = 24$ would instead need to consistently generalize on 16 examples, and so on, making perfect agreement more difficult to achieve.

To address this, we increase N to 100. We still train on $M \in \{6, 24, 36\}$ and all non-compositional examples. However, we test only on compositional examples from $F\ x_i$ for $i \in [50, 100]$, i.e., compositional examples that did not occur for any value of M. This means that the size of the test set remains identical for all M, and eliminates the confound between M and the size of the test set.

3.2 Results

KC evaluate model performance by considering two possible hypotheses. "Composition" for them means that a model generalizes such that $F\ a \rightarrow A\ A\ A$. In contrast, they interpret "memorization" to indicate that a model produces the single symbol output associated with the non-modifier input symbol, i.e., $F\ a \rightarrow A$. However, this is certainly not the full space of possible hypotheses, and narrowly restricts what constitutes compositional behavior. For instance, suppose that the model learns that F means "produce any two identical symbols, followed by the symbol a maps to in non-compositional examples" (i.e, $F\ a = B\ B\ A$, with B an output symbol other than the one associated with input a). This is also a kind of compositional generalization: the output related to the complex input is a function of its individual input symbols and their mode of combination (Szabó, 2022). In this case, $[\![F]\!] = \lambda x.B\ B\ x$ and $[\![a]\!] = A$ with concatenation associated with function application. Following this intuition, we consider as compositional any mapping in which the input a is associated with at least one occurrence of A in the output, and which includes at least two other symbols in output, possibly, but not necessarily identical to A. Many other generalizations are possible under this interpretation of compositionality, even if they don't correspond to the "thrice" interpretation, i.e., $[\![F]\!] = \lambda x.x\ x\ x$. Determining that models fail to generalize according to one compositional interpretation does not preclude the possibility that they

[5] Of course, we have presented this task as mapping lowercase letters to uppercase letters, and English would only provide a maximum of 26 possible input-output pairs for this task—fewer than the 40 described. In practice, we used the natural numbers as input symbols, and the natural numbers prefixed with O as output symbols, to avoid this complication. We use the alphabetic notation for presentational convenience.

have generalized compositionally according to another.

For this reason, we classify responses according to a considerably wider range of possible templates. In particular, we consider all ways in which symbols in the output might be identical or differ for numbers of output symbols from 1–3.[6] We define A as the output symbol corresponding to input symbol a, and B, C, D as symbols other than A that do not correspond to a. For instance, the generalization $F\ a = B\ B\ A$ means the model produced two (identical) non-A symbols, followed by the symbol corresponding to a; while $F\ a = B\ C\ A$ means that the model produced two (distinct) non-A symbols, followed by the symbol corresponding to a. Non-A symbols B, C, and D identify identical output symbols within a response, but may differ across responses from the same model.[7] Both of these responses would, we argue, qualify as compositional in a broader sense, even if they don't instantiate the "thrice" interpretation: the output is a function of both input symbols, with a being mapped to A, and F requiring an output sequence of length 3.

We report the same measures as for experiment 1: the fraction of perfect agreement (Table 2), and the proportion of productions matching a particular generalization (Figure 1). Due to the high number (22) of generalizations we consider, for FPA we only report non-zero results. Only CNNs ever responded completely consistently, and even then, only a very small number of seeds did so (3 of 300 models). Instead, the vast majority of models produced responses consistent with a variety of answer types. Making the size of the test set consistent for all M appears to have made it harder for models to consistently generalize, as expected.

Architecture	M	Generalization	FPA
CNN	6	$F\ a = B\ B\ B$	0.02
	24	$F\ a = A\ A\ A$	0.01

Table 2: FPA results from experiment 2. Architectures, M, and generalizations not shown have FPA = 0.00.

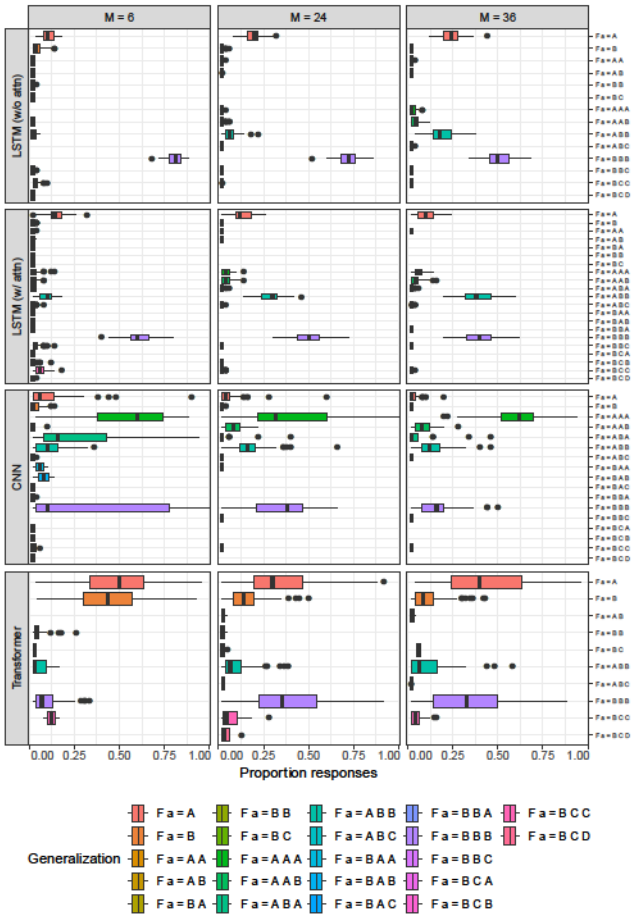

Figure 1: Distributions of PrAg with generalizations of up to 3 symbols in experiment 2. Within each row of plots, generalizations are omitted from the y-axis if no model of that architecture produced any response fitting that template for any M.

The distribution of response proportions (figure 1) reveals a more nuanced picture of model preferences. Both types of LSTMs have a preference for non-compositional $F\ a = B\ B\ B$ responses for $M = 6$, with KC's memorization $F\ a = A$ responses also among the more common. As M increases, however, the proportion of the (compositional) $F\ a = A\ B\ B$ responses increases, with LSTMs with attention appearing to have equal preference for $F\ a = B\ B\ B$ and the compositional $F\ a = A\ B\ B$ when $M = 36$. CNNs, in contrast, display a wider

[6] Models sometimes produced longer responses (up to the maximum length of 200 symbols). We present results for responses up to at most 3 symbols for perspicuity. In practice, longer responses were entirely absent from CNNs (0% of responses), and almost entirely absent from LSTMs w/o attention ($M = 6$: 0.04%, $M = 24$: 0.16%, $M = 36$: 0%) and LSTMs w/attention ($M = 6$: 0.04%, $M = 24$: 0.08%, $M = 36$: 0.04%). Consequently, our analysis for these networks is essentially exhaustive. For transformers, longer responses were more common, but still a minority, with differences depending on M: longer responses accounted for 29.2% of responses for $M = 6$, 9.2% of responses for $M = 24$, and 2.5% of responses for $M = 36$. Of these, for $M = 6$ the most common lengths were 200 (the maximum length, 20.9% of responses) and 4 (2.86%). For $M = 24$, 1.3% of responses were length-200 and 7.06% were length-4; and for $M = 36$, 0.46% of responses were length-200 and 1.86% were length-4.

[7] In the computation of the FPA measure, we did not require that a model make use of a consistent choice of symbols. That is, in order to count as consistently $F\ a = B\ B\ A$, the model's choice of B in the output could vary by response, but was required to be distinct from A. Since the PrAg measure was calculated at the level of the individual trial, no consistent choice of non-A symbols was required.

range of common response types for smaller M, but as M increases, the compositional "thrice" response $F\ a = A\ A\ A$ becomes the modal response. Finally, transformers with $M = 6$ show equal preference for non-compositional $F\ a = A$ and $F\ a = B$, but with the preference for $F\ a = B$ being replaced with a preference for the similarly non-compositional $F\ a = B\ B\ B$ for $M \in \{24, 36\}$.

Non-A symbols for responses of length > 1 overwhelmingly were drawn from the symbols seen with compositional training examples, with the mean proportion of these symbols in outputs with non-A symbols uniformly being between $[0.976, 0.999]$. However, we did not observe any tendency for particular models to use the same non-A symbols across responses.

In spite of our use of the PrAg measure as opposed to KC's minimal description length-based measure, our results are generally compatible with theirs, especially when we limit our focus to the two hypotheses they considered, namely $F\ a = A$ (their "memorization") and $F\ a = A\ A\ A$ (their "composition"). For LSTMs, KC found a general preference for memorization over composition, except for LSTMs with attention with $M = 36$, where the preference is (barely) reversed. In figure 1, we similarly see that the proportion of $F\ a = A$ responses for LSTMs with and without attention is consistently higher than that of the $F\ a = A\ A\ A$ responses. Our more fine grained analysis reveals however that LSTMs show even stronger preferences for $F\ a = B\ B\ B$ or $F\ a = A\ B\ B$ than either of the hypotheses KC considered. As noted above, this latter rule is plausibly interpreted as compositional, as its output does not include a reflection of the input symbol a. This suggests that LSTMs do not show such a uniform dispreference for compositional generalization. For CNNs, KC report a strong preference for composition over memorization for larger values of M, and our Figure 1 reveals the same preference. We note, however, that many of the responses we observed fell into the $F\ a = B\ B\ B$, $F\ a = A\ B\ B$ and $F\ a = A\ B\ A$ categories, of which the former is plausibly non-compositional. Finally, for transformers, KC report a uniform preference for memorization over composition, which we also see in figure 1. Again, Figure 1 reveals that other preferences are often stronger than either of these. Transformers, for $M = 6$, show a nearly equal preference for $F\ a = B$ as compared to KC's memorization hypothesis, and for larger values of M, $F\ a = B\ B\ B$ is comparably frequent to memorization. Nonetheless, these are both plausibly considered non-compositional responses, once again because the output lacks a symbol corresponding to the input a.

To take stock, we see that expanding the space of hypotheses we consider reveals that learners are less likely to generalize consistently. Further, while CNNs retained a bias for $F\ a = A\ A\ A$ responses with larger Ms, LSTMs and transformers behave considerably differently from what KC reported.

3.3 Inducing compositional generalization

Why are the networks we studied in our previous experiment resistant to systematic compositional generalization? This seems surprising in the face of apparent compositionality in state of the art language models, which are constructed from some of the same architectures we have explored.

One difference between our experiments and such models lies in the training dataset. Our dataset is extremely simple, consisting of single symbol input-output mappings and one compositional operator. It is possible that a model trained on a rich variety of structures, each showing the kind of combinatorics associated with compositionality, could evince more compositional generalization.[8]

To explore this possibility, we conducted an evaluation of the SCAN dataset (Lake and Baroni, 2018). SCAN consists of a finite set of English-like inputs that represent intended movements of a robot, and outputs are step-by-step instructions for achieving these movements. SCAN includes a fair number of different predicates that can combine in different ways. This permits a training set to exhibit broad evidence for compositional combination, and allows us to test for systematic compositional generalization. We experiment with two splits of the SCAN dataset. The first is the `addprim_jump` split of Lake and Baroni (2018). In this split, the training set includes the full range of possible sentences from SCAN, except for those involving compositional uses of the primitive `jump`. The test set consists of all compositional sentences containing this primitive. This split poses the question of whether a model learns to generalize the use of `jump` to all positions in which other simple predicates occur.

Our second split, `addtwicethrice_jump`, comprises a training set that includes all sentences without `jump` as well as sentences in which `jump` is

[8] See Patel et al. (2022) for a related proposal.

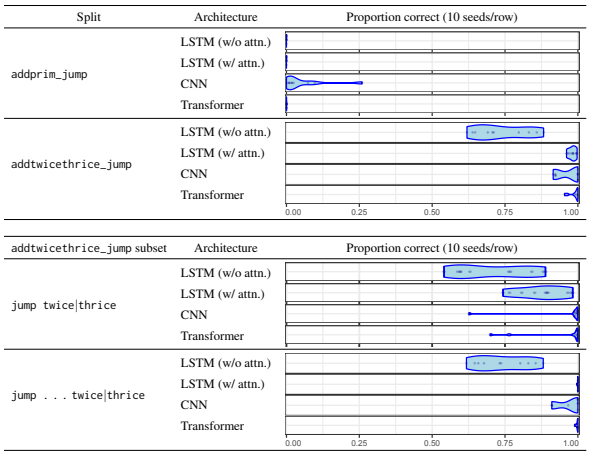

Table 3: Results of SCAN experiments. Overlapping points are jittered on the *y*-axis.

not under the scope of `twice` or `thrice`, operators whose outputs require repeating the sequence of instructions corresponding to the trajectory to their left 2 or 3 times. The test set contains all inputs where `jump` occurs under the scope of `twice` or `thrice`. This means that, unlike the `addprim_jump` split, `jump` does occurs in the training set in the presence of other compositional operators, just not with `twice` or `thrice`. This avoids the potential problem that a model might learn `jump` has the distinctive property of only appearing by itself.

In their appendix D, KC report results from the `addprim_jump` split. Because we were unable to determine all details of their training regimen from their description, we first repeated their experiment of the `addprim_jump` split. For LSTMs and transformers, we use the architectures they reported. For CNNs, we use models with 5 encoder layers, 1 decoder layer, and a kernel size of 8, the best performing of their CNN models. We trained 10 random initializations per architecture on each split, using 1000 batches sampled with replacement for 500 epochs, for a total of 500,000 weight updates. Batch sizes matched KC's (LSTMs, CNNs: 16; transformers: 256). We used the same procedure for our `addtwicethrice_jump` split. Accuracy on the training set was uniformly high: the model with the lowest performance achieved 98.2% accuracy on the training set, with most achieving 100%.

Results from these experiments appear in the upper half of table 3. While we did not replicate KC's results on the `addprim_jump` split exactly, we similarly find that on this split, only CNNs show any sort of generalization, though even their performance is quite low. However, the picture is quite different for the `addtwicethrice_jump` split, where all models generalize correctly to most examples. It seems that showing `jump` in a range of compositional contexts helped the models generalize to this predicate's occurrences in other compositional contexts, compared to when we show the models `jump` in only non-compositional contexts.

To investigate this difference further, we consider performance on two subsets of the `addtwicethrice_jump` test set: (1) a subset comprising all examples where `jump` occurs immediately preceding `twice` or `thrice` in the input, and (2) the complement of set (1).[9] On the face of it, test set (1) seems simpler and might yield better performance, as its examples involve less depth of embedding. However, the results (lower half of table 3) reveal the opposite pattern: performance tends to be worse on examples where `jump` occurs immediately adjacent to `twice` and `thrice`.

We hypothesize this is due to models' dependence on (irrelevant) surface properties of the input in the training set: in the `addtwicethrice_jump` split, the bigrams `jump twice` and `jump thrice` never occur—they have a probability of 0. However, all other bigrams permitted in SCAN occur in the training set of this split. So, even though this training set is sufficient to induce compositional generalization, the presence of non-occurring bigrams yields less accurate performance.[10] The data augmentation reported in Andreas (2020), which also reduces non-occurring bigrams in test items, has a similarly salutary impact on generalization. While speculative, this could explain the sharp distinction between the `addprim_jump` and `addtwicethrice_jump` splits, since in the former, no bigram including `jump` occurs in the training set, since it occurs only in isolation. A similar line of reasoning could be behind the poor compositional performance we saw in Section 3.2.

4 Related Work in Architectural Inductive Bias

Here, we briefly discuss work on inductive bias in neural network models that focuses particularly on those aspects of inductive bias that are traceable to

[9] Note that in set (2), `jump` is still under the scope of `twice` or `thrice` in each example, just not adjacent to either, as in `jump opposite right twice`.

[10] We do not consider bigrams privileged in this regard; this is just the simplest way of characterizing the divide given our training datasets. We might expect a more nuanced picture if we take into account n-grams with $n > 2$, though local relationships will need to play a more important role in order to explain the contrasts reported in the lower part of Table 3.

network architecture; see Hupkes et al. (2023a,b) for a more thorough review.

In some cases, inductive biases have been intentionally built in to particular architectures. The parameter sharing and filter structure of CNNs leads directly to a bias for translation invariance and feature locality, which are useful in a variety of tasks (LeCun and Bengio, 1998; Mitchell, 2017). The inclusion of gates with multiplicative interactions in LSTMs (Hochreiter and Schmidhuber, 1997) explicitly addressed deficiencies of RNNs in modeling the long-distance dependencies found in natural language (Elman, 1990). Weiss et al. (2018) discuss the fact that unbounded growth in LSTMs' hidden state vectors leads to a counting bias.

None of these biases relate explicitly to the questions of structural or compositional generalization that we have explored in this paper. Research on language-related biases has explored structural generalization. White and Cotterell (2021) train LSTMs and Transformers on synthetic corpora that exhibit a range of word order patterns, some attested in natural languages, others not. LSTMs, unlike Transformers, showed a preference for certain word order patterns over others, but neither showed a bias toward attested natural language patterns. McCoy et al. (2020) study the ability of different architectures to generalize structurally-defined mappings between sentences. LSTMs, RNNs, and GRUs with different attention mechanisms failed to generalize structurally, preferring linear generalizations. Petty and Frank (2021) find an even more extreme failure for Transformer models. McCoy et al. only find a bias toward structural generalization in models with explicitly hierarchically-structured recurrence (Chen et al., 2017).

To detect bias toward compositionality, researchers have explored tasks requiring the mapping of a natural language input to an interpretation of some sort, including SCAN (Lake and Baroni, 2017), PCFG SET (Hupkes et al., 2020), COGS (Kim and Linzen, 2020b), and SLOG (Li et al., 2023). Success on these datasets is informative about the kinds of generalizations that networks are capable of. Yet their complexity, compared to the tasks we explored, can make it difficult to identify reasons for success or failure. Indeed, modifications to non-crucial properties of a dataset can yield quite different results (Wu et al., 2023). Of course, considering complex cases is important, as we have indeed seen in Section 3.3 above. However, we see the study of simple tasks as providing complementary understanding.

Recently, it has been found that experiments aiming to identify inductive biases must also carefully control for training regimen. For example, training without early stopping can lead to qualitatively different patterns of generalization through "grokking" (Csordás et al., 2021; Power et al., 2022). It will be important to understand the range of applicability of grokking and the like, as well as the challenge such results pose for assessing inductive bias.

5 Conclusion

A finite set of data may be consistent with an infinite number of possible generalizations (Hume, 1739). It is precisely for this reason that studies of inductive bias must proceed with great care to avoid adducing support for the existence of certain kinds of inductive biases prematurely.

Taking this idea to heart, we found that the KC's claim that transformers *tout court* display hierarchical bias to be premature. Instead, we found their behavior to be most consistent with a counting-based strategy.

Similarly, we found that considering a more consistent test set when assessing compositionality led us to find less support for consistent generalization than KC. Though the pattern of results we found was consistent with theirs when limiting our focus to the hypotheses they considered, we found that expanding the range of possible hypotheses revealed a more nuanced picture.

Nevertheless, we found there is hope for inducing compositional generalization: when we ensure that distributional properties of the training set are more like the those in the test set, compositional generalization appears easily achieved.

This suggests low performance on a particular task may be due to irrelevant factors, like surface-level distributional properties of a training dataset. It may not be that a particular model cannot achieve high levels of success on certain tasks due to the inherent complexity of whatever pattern of behavior the task aims to measure, but because it has fixated on some accidental, irrelevant generalization in its training data that impedes its recognizing the correct generalization. Alleviating models' propensity toward fixation on such irrelevancies, perhaps by altering their training data in systematic ways (as in our SCAN experiments), may prove useful in improving their performance in various domains.

Limitations

The small scale of the datasets used here to train the networks represent both an advantage and limitation. While this scale limits information given to the model during training, the unusually small dataset can lead the model to detect and induce unusual distributional properties as relevant to the task at hand. Indeed, we have seen an instance of this problem in experiment 2. Though there is no magic bullet for avoiding this issue we suspect, it is one that future work that seeks to develop "model induction organisms" will need to take into account.

Experiments reported here all made use of the same procedure for parameter optimization, namely Adam (Kingma and Ba, 2014), and therefore differences observed between the models must be due to their structure. However, it is possible that different optimization methods may lead to different inductive biases for the same network architectures.

Acknowledgments

This work was made possible by support from National Science Foundation grant BCS-1919321.

References

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566. Association for Computational Linguistics.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Vancouver, Canada. Association for Computational Linguistics.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 619–634, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

David Hume. 1739. *A Treatise of Human Nature*. Oxford University Press, Oxford.

Dieuwke Hupkes, Verna Dankers, Mul Mathijs, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2023a. State-of-the-art generalisation research in NLP: A taxonomy and review.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2023b. A taxonomy and review of generalization research in NLP. *Nature Machine Intelligence*, 5:1161–1174.

Eugene Kharitonov and Rahma Chaabouni. 2021. What they do when in doubt: a study of inductive biases in seq2seq learners. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020a. COGS: a compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Najoung Kim and Tal Linzen. 2020b. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Brenden M. Lake and Marco Baroni. 2017. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *CoRR*, abs/1711.00350.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden.

Yann LeCun and Yoshua Bengio. 1998. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA.

Bingzhi Li, Lucia Donatelli, Alexander Koller, Tal Linzen, Yuekun Yao, and Najoung Kim. 2023. SLOG: A structural generalization benchmark for semantic parsing.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2020. Does syntax need to grow on trees? Sources of hierarchical inductive bias in sequence-to-sequence networks. *Transactions of the Association for Computational Linguistics*.

William Merrill and Ashish Sabharwal. 2023. A logic for expressing log-precision transformers.

Benjamin R. Mitchell. 2017. *The Spatial Inductive Bias of Deep Learning*. Ph.D. thesis, Johns Hopkins University.

Tom M. Mitchell. 1980. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, Department of Computer Science.

Aaron Mueller, Robert Frank, Tal Linzen, Luheng Wang, and Sebastian Schuster. 2022. Coloring the blank slate: Pre-training imparts a hierarchical inductive bias to sequence-to-sequence models. In *Findings of the Association for Computational Linguistics 2022*.

Arkil Patel, Satwik Bhattamishra, Phil Blunsom, , and Navin Goyal. 2022. Revisiting the compositional generalization abilities of neural sequence models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 424–434.

Jackson Petty and Robert Frank. 2021. Transformers generalize linearly.

Alethea Power, Yuri Burda, Harrison Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *CoRR*, abs/2201.02177.

Zoltán Gendler Szabó. 2022. Compositionality. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne.

Jennifer C. White and Ryan Cotterell. 2021. Examining the inductive bias of neural language models with artificial languages. In *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)*, pages 454–463, Online. Association for Computational Linguistics.

Zhengxuan Wu, Christopher D. Manning, and Christopher Potts. 2023. ReCOGS: How incidental details of a logical form overshadow an evaluation of semantic interpretation.

GenBench Evaluation Card

Motivation			
Practical	*Cognitive* □ △	*Intrinsic* □ △	*Fairness*
Generalisation type			
Compositional △	*Structural* □	*Cross Task*	*Cross Language*	*Cross Domain*	*Robustness*
Shift type			
Covariate □ △	*Label*	*Full*	*Assumed*
Shift source			
Naturally occurring	*Partitioned natural*	*Generated shift*	*Fully generated* □ △
Shift locus			
Train–test □ △	*Finetune train–test*	*Pretrain–train*	*Pretrain–test*