Systematic Generalization in Language Models Scales with Information Entropy

Sondre Wold, Lucas Georges Gabriel Charpentier, Étienne Simon
Language Technology Group
University of Oslo

Abstract

Systematic generalization remains challenging for current language models, which are known to be both sensitive to semantically similar permutations of the input and to struggle with known concepts presented in novel contexts. Although benchmarks exist for assessing compositional behavior, it is unclear how to measure the difficulty of a systematic generalization problem. In this work, we show how one aspect of systematic generalization can be described by the entropy of the distribution of component parts in the training data. We formalize a framework for measuring entropy in a sequence-to-sequence task and find that the performance of popular model architectures scales with the entropy. Our work connects systematic generalization to information efficiency, and our results indicate that success at high entropy can be achieved even without built-in priors, and that success at low entropy can serve as a target for assessing progress towards robust systematic generalization.

1 Introduction

Human language is characterized by its combinatorial properties in syntax and semantics (Hadley, 1994). For example, when someone knows what it means for Person A to see Person B, they can readily generalize to what it means for Person B to see Person A, without having been exposed to examples where Person B is used in the subject position (Fodor and Pylyshyn, 1988). This type of bidirectional systematic generalization has long been considered a key feature of human language understanding (Chomsky, 1957), and is closely related to the compositionality of natural language semantics, a contentious but widely studied topic in both formal semantics (van Bethem et al., 1991) and NLP (McCurdy et al., 2024).

Early critics of connectionism, such as Fodor and Pylyshyn (1988), argued that systematic generalization was incompatible with continuous represen-
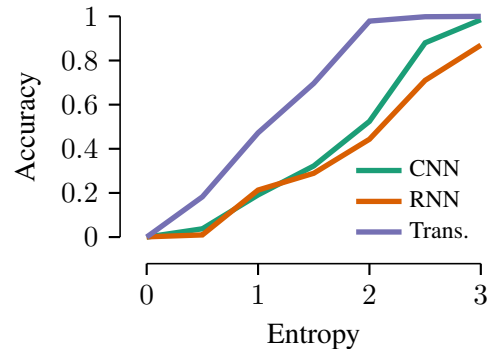

Figure 1: Systematic generalization increases with the entropy of the training data for popular sequence-to-sequence models, even without built-in priors.

tations. More than thirty years since this critique was initially presented, researchers still agree that achieving robust compositional behavior in neural networks remains an open problem, as shown by a recent survey by McCurdy et al. (2024). Research continues to demonstrate that current models struggle with systematic generalization (Lake and Baroni, 2018; Hupkes et al., 2020; Dziri et al., 2024; Wold et al., 2024; Mirzadeh et al., 2024).

Some of the limitations of current models can be alleviated by implementing compositional priors, either in the architecture (Gordon et al., 2019) or through the training procedure (Lake and Baroni, 2023), but it remains unclear whether architectural constraints cause the limitations or whether they arise due to distributional properties of the data used for training.

In this paper, we ask the following question: Can neural networks display systematic out-of-distribution generalization without *any* built-in priors? To answer this, we take a data-centric approach. Specifically, we show how the degree of systematic generalization of common architectures relates to the information entropy of the distribution of component parts found in the training data.

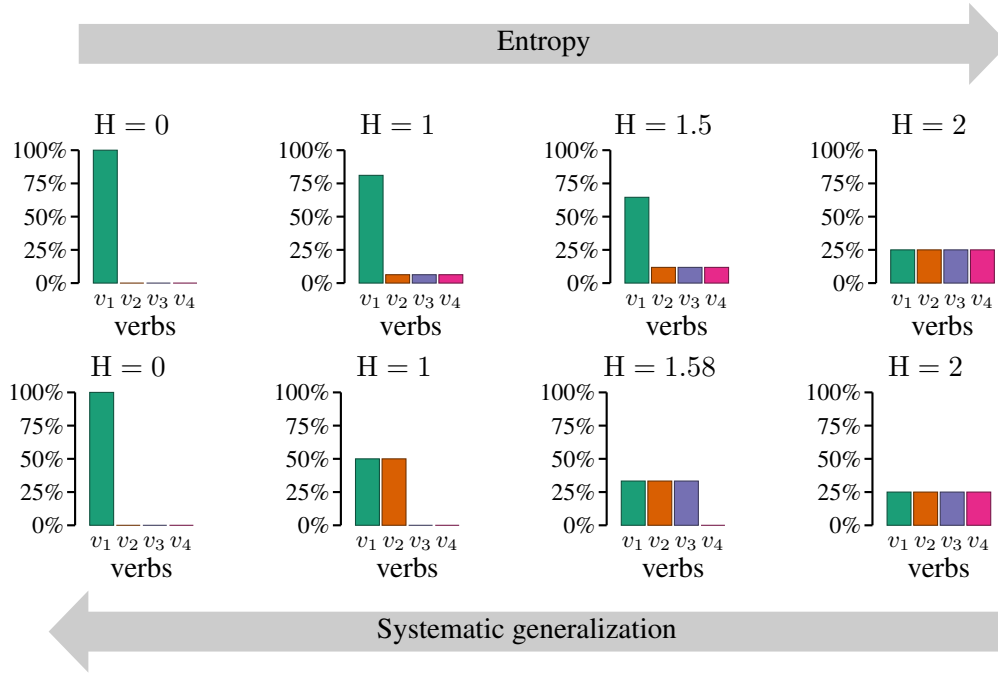Contrary to previous work, we find that popu-

Figure 2: A schematic overview of the two approaches to increasing the entropy level. Top: Vertical scaling by distribution mixing. Bottom: Horizontal scaling by incrementing the support of the distribution.

lar sequence-to-sequence models used in NLP are capable of systematic generalization, even without built-in architectural priors. However, this is only the case when the training dataset contains high entropy. On a series of sequence-to-sequence datasets based on a modified version of the SCAN (Lake and Baroni, 2018) grammar, we show that model performance scales with the entropy of the training dataset, as illustrated in Figure 1, thereby creating a link between systematic generalization and information efficiency.

In summary, our contributions are: *(i)* We develop and formalize a framework for studying systematic generalization based on information entropy; *(ii)* We experiment with two different approaches to increasing entropy, and we compare the performance of four different model architectures for these two settings; and *(iii)* we show that neural networks achieve systematic out-of-distribution generalization when the entropy of the training distribution is high, independently of the number of unique samples seen during training.

2 Background

Systematic generalization is typically described as a robust mode of composition, where a system is able to combine parts of its training data in a way that enables generalization to novel combinations. As such, systematic generalization relates to com-

positionality, a topic which has both a long tradition and multiple formal definitions, most prominently in formal semantics (van Bethem et al., 1991; Pagin and Westerståhl, 2010). These definitions, however, often prove too restrictive when working with natural languages. As a result, researchers frequently introduce problem-specific relaxations, leading to ambiguity over time regarding the exact meaning of terms such as systematic generalization and compositionality.

Despite there being some ambiguity regarding the formal definitions, recent work shows that there is high agreement among researchers on the following, more informal definition of what constitutes compositional behavior:

Definition 1 (CB) "When a model receives an input I that humans conceive as composed of component parts, if the model produces correct outputs for those parts (in isolation or in other combinations), then it will also produce a correct output for I (McCurdy et al., 2024, p. 9324)."

This definition, however, does not consider the conditions from which a system learns to produce these outputs. Consequently, compositional behavior differs from compositional *generalization*. While the former can be assessed through normal benchmarks, the latter requires information about the distribution the model generalizes from.

Although generalization is a loosely defined term

in machine learning research, it generally refers to how well a model performs on a held-out dataset after being trained on data that has some distributional relationship with the held-out data (Hupkes et al., 2023). When training and test distributions are similar, we refer to this as *in-distribution*, and when they differ, we typically refer to this as *out-of-distribution*. Naturally, the classification of a generalization problem as in-distribution or out-of-distribution depends on which aspects of the distribution are considered. In this work, we focus on *systematic* generalization as an out-of-distribution problem, where the differences in the distribution of component parts and their combinations between the training and test data are the relevant aspects.

2.1 Systematic Generalization

Following earlier works, we will refer to the systematic generalization abilities of a model as the *systematicity* of that model. Our view of systematicity is based on Hadley (1994), who in contrast to earlier work by Fodor and Pylyshyn (1988) distinguished between *degrees* of systematicity. In the following paragraphs, we present and define these degrees under the assumption that we are considering a system tasked to generalize from a training corpus to a held-out test corpus:[1]

Definition 2 (Weak systematicity) A system displays *weak systematicity* if the system requires that all component parts that occur in the training data also occur at every permissible syntactic position. Even if the model succeeds at a test set that contains novel samples, none of the components in these samples occur at a novel syntactic position, essentially making the problem in-distribution with respect to the systematic properties of the data.

Definition 3 (Strong systematicity) Now, a system displays *strong systematicity* if it *(i)* displays weak systematicity, meaning that it can generalize in-distribution, and *(ii)* it can process novel embedded sentences containing previously learned component parts that occur in syntactic positions where they do not appear in the training data, neither in embedded sentences nor in non-embedded sentences, and *(iii)* a significant fraction of the possible combinations of component parts and their permissible syntactic position should be left out, based on

the observation that humans learn grammars based on largely incomplete data Pinker (1989). In this context, an embedded sentence can be thought of as a syntactic constituent in the overall sentence. For example, if a system can process the following novel sentence *Mathilde knows Sigurd and Sigurd knows Olav*, which has the sentences *Mathilde knows Sigurd* and *Sigurd knows Olav* as embedded sentences, without having seen the name *Mathilde* previously used in the subject position in a sentence with constituents nor in a sentence with no constituents, then it is strongly systematic w.r.t to how it can generalize to the novel use of the name *Mathilde*. On the other hand, if the system can only process this novel sentence if it has seen *Mathilde* being used in the same syntactic position, for example in the sentence *Mathilde knows Olav*, then it is not strongly systematic.

Systematic generalization in embedded sentences. In this work, we focus on a relaxed version of strong systematicity as defined in Definition 3. We relax requirement *(ii)* from the original definition so that we only work with samples that have embedded sentences. As we show in the following section, this makes it easier to quantify the level of systematicity required for a given generalization problem, which is the main focus of this work. Furthermore, we also relax *(ii)* so that it allows component parts of the test set to occur at the same syntactic position in *some* embedded sentence in the train set, but not in an embedded sentence that occurs at the same position in the overall sequence.

In order to compare models with and without compositional priors from the literature, we base our experiments on the SCAN grammar (Lake and Baroni, 2018). This grammar has few permissible syntactic positions for each part of speech, making the original requirement from Hadley (1994) too restrictive with respect to the overall sample size needed by neural networks. In the following section, we formalize our framework and elaborate on these choices.

3 Measuring Systematicity

In this work, we demonstrate that sequence-to-sequence models can generalize to embedded sentences where component parts occur at novel syntactic positions in the overall sequence. For our experiments, we construct datasets where each dataset sample contains an input sequence $x \in X$

---

[1]Hadley (1994) also discusses an intermediate degree between weak and strong systematicity, so-called *quasi-systematicity*, but we leave this out for brevity, as it does not relate directly to our use case.

and an output sequence $y \in Y$, based on a modified version of the SCAN grammar. We discuss the specifics of this dataset generation procedure, including details about the context-free grammar that generates x, in Section 3.5. In this section, we define a framework for measuring the *degree* of systematicity displayed by a systematic generalization problem.

3.1 Task Description

Throughout this paper, x comprises two sentences conjoined by a conjunctive phrase.[2] Consequently, x can be modeled as a triple, with a conjunction $c \in C$, and the two embedded sentences $e_1, e_2 \in \Sigma^*$ as arguments, $x = (e_1, c, e_2)$. The component parts of both embedded sentences are drawn from the same vocabulary $\Sigma = V \cup A$, where V is a set of verbs and A is a set of adverbial phrases, e.g.: JUMP, RUN $\in V$, TWICE $\in A$, and AND $\in C$. There exists a deterministic procedure φ for transforming x into y:

$$\varphi(\text{RUN TWICE AND JUMP}) = \textit{RUN RUN JUMP}.$$

The goal of the described task is to learn a function f that approximates φ, which requires modeling the conditional distribution of output sequences given input sequences in the training data, i.e. $p^{\text{train}}(y \mid x)$. See Table 1 for more examples generated by our modified SCAN grammar.

3.2 Component Distributions

Since we are interested in systematic generalization, we also need to define the distribution of component parts in e_1 and e_2, and we are specifically interested in the case where the distribution of e_1 and e_2 is different in the training and test data, but they share a similar distribution when marginalized over position, which in this case refers to which side of the conjunctive phrase they appear in.

In SCAN (Appendix A), e_1 and e_2 always contain **one** verb in V and zero or more adverbial phrases from A that control how many times that verb should be repeated, and in what direction it is executed. To create a systematic generalization problem that adheres to the definition given in Section 2.1, it is necessary to set the support of e_1 and e_2 to be different in the training and test distributions. For simplicity, we limit our study to focus on the setting where the support of the verbs V is different between the two distributions. This means
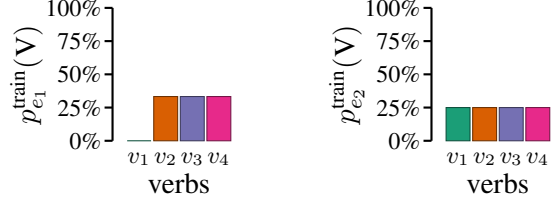

Figure 3: Example of distributions of verbs for e_1 (left) and e_2 (right).

that we can create a generalization problem by restricting a verb v_1 to never occur in e_1 but to occur in e_2 for p^{train}, as illustrated by Figure 3.

For p^{test} we invert the distribution of e_1, making it a degenerate distribution with single-point support for the verb that is not in e_1 for p^{train}, while e_2 has the same distribution as in p^{train}. When generalizing from p^{train} to p^{test} in this scenario, a model will be exposed to all verbs in all permissible syntactic positions in *some* embedded sentence, but v_1 will never be seen in e_1 during training, while e_1 in p^{test} always contains v_1.
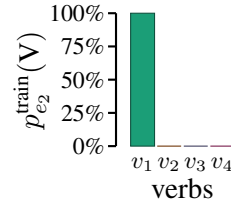

Figure 4: Probabilities in the degenerate case for e_2

3.3 Systematicity as Entropy

The main contribution of this work is to demonstrate that the difficulty of generalizing from p^{train} to p^{test} in the scenario described above depends on the *entropy* H of the distribution of verbs in e_2 for p^{train}:

$$\text{H}_{e_2}^{\text{train}}(\text{V}) = - \sum_{v \in V} p_{e_2}^{\text{train}}(v) \log_2 p_{e_2}^{\text{train}}(v),$$

where V is the verb random variable and V is the set of all verbs. For brevity, we will refer to $\text{H}_{e_2}^{\text{train}}(\text{V})$ as H. When the verbs in e_2 have a degenerate distribution, as shown in Figure 4, the entropy is zero, and when e_2 is distributed uniformly, as shown in Figure 3, the entropy is at its maximum. Decreasing H within this interval increases the difficulty of the systematic generalization problem. Note that even at maximum entropy, the generalization problem is still out-of-distribution, as the restricted verb is never seen in e_1. In Section 4, we

show empirically that the performance of neural network architectures scales with H on this type of generalization problem.

The degenerate case. We note that H $= 0$ constitutes a special case in the described framework. If e_2 is described by a degenerate distribution, then instances of e_2 will always contain the same $v_1 \in V$. Now, if $|C| = 1$, there is no way of decoupling the semantics of the conjunctive phrase and v_1, as these will always appear together in X. In this case, we argue, no system will be able to generalize from p^{train} to p^{test}. Consequently, it is necessary to set $|C| > 1$ for evaluating performance at H $= 0$.

3.4 Increasing Entropy

In this work, we experiment with two different approaches to increasing H, as illustrated in Figure 2. In the following sections, we formalize these approaches, and in Section 4 we demonstrate how the performance of neural architectures scales for each type of increment.

3.4.1 Distribution Mixing

Let $\mathcal{U}$ be a uniform distribution with support over $V \setminus \{v_1\}$, and let $\mathcal{D}$ be a degenerate distribution over V with single-point support for v_1. Our first approach consists in defining a family of distributions parametrized by λ mixing $\mathcal{U}$ and $\mathcal{D}$:

$$p_{e_2}^{\text{train}}(\text{V}) = \lambda \mathcal{U} + (1 - \lambda)\mathcal{D},$$

Then, the entropy H is at the minimum when $\lambda = 0$ and at the maximum when $\lambda = 1 - 1/|V|$, which is equal to a uniform distribution over V. Consequently, increasing H can be achieved by tending λ towards $1 - 1/|V|$. This approach to increasing H is also illustrated in the top part of Figure 2.

3.4.2 Incremental Support

In our second approach, we define another family of distributions, all uniforms over increasingly larger supports $S_i \subseteq V$:

$$\{v_1\} = S_1 \subset S_2 \subset \cdots \subset S_{|V|} = V$$

The uniform distribution over the smallest support S_1 is equivalent to the degenerate distribution $\mathcal{D}$. As i increases, H increases logarithmically until reaching maximum entropy at $i = |V|$ where we have the uniform distribution over V. This approach to increasing H is also illustrated in the bottom part of Figure 2.

3.5 Data Generation

As posited in the previous sections, the quantification of systematic generalization requires information about the distribution of parts and combinations observed during training. This level of granularity is generally intractable when dealing with real-world corpora. Without this information, however, it is difficult to control for statistical patterns that permit a non-compositional solution. Studies on systematicity typically overcome this problem by studying synthetic languages where the distribution can be known a priori. This is also the approach taken in this work.

To generate datasets samples $(x, y) \in X \times Y$ we define a context-free grammar (CFG) based on the SCAN grammar (Lake and Baroni, 2018). An overview of the complete vocabulary can be found in Appendix A, while Table 1 shows examples from our generated data. Importantly, we set $|V|$ to 8, as opposed to the original four verbs, thereby increasing the number of possible sequences generated by the grammar. We also remove the turn operators, as they are more syntactically constrained than other verbs, as was already done in Gordon et al. (2019). As previously stated, all instances of x contain embedded sentences e_1 and e_2.

In our experiments, we use the standard conjunctive phrases in SCAN $|C| = 2$. Furthermore, we let the two conjunctive phrases take the embedded sentences in the opposite order: if v_1 always occurs in e_2 for c_1, then v_1 will always occur in e_1 for c_2.

4 Experiments

We validate our framework by evaluating the three most common sequence-to-sequence architectures: the Transformer, the RNN, and the CNN. We also run a model with built-in compositional priors to showcase how a model that encodes verb equivariance can solve the task even when H is low. We have two main experimental settings, corresponding to the two methods for increasing H described in Section 3.4.1 and Section 3.4.2. We also conduct a supporting experiment to control for a potential confounding effect resulting from an increased sample size, and an experiment that estimates the influence of different position encoding types. For all experimental settings, H $= 0$ constitutes the most difficult generalization problem, and H $= \log_2 |V| = 3$ constitutes the easiest problem. We average and report results from five differently seeded runs. The following sections provide details

Input sequence x	Output sequence y
squat opposite right and squat	RTURN RTURN SQUAT SQUAT
squat twice and crawl opposite left	SQUAT SQUAT LTURN LTURN CRAWL
sprint right twice after sprint left	LTURN SPRINT RTURN SPRINT RTURN SPRINT
lunge opposite left and look thrice	LTURN LTURN LUNGE LOOK LOOK LOOK

Table 1: Examples from our generated datasets, with green indicating e_1 and orange indicating e_2.

about our experimental setup.[3]

4.1 Models

RNN. For our RNN we use a similar configuration as previous work on SCAN (Lake and Baroni, 2018; Gordon et al., 2019): a bidirectional encoder-decoder with attention. Details on the implementation and choice of hyperparameters can be found in Appendix B.1.

CNN. For our CNN we use the encoder-decoder architecture from Gehring et al. (2017), which has been proven effective on SCAN in previous work (Dessì and Baroni, 2019). As in the RNN, this CNN uses an attention mechanism between the encoder and decoder. Hyperparameters can be found in Appendix B.2.

Transformer. We use the original encoder-decoder Transformer from Vaswani (2017). We use this for two reasons. Firstly, this makes comparison with previous work on systematicity easier, as this has been used in previous work such as Hupkes et al. (2020) and Lake and Baroni (2023). Secondly, recent work on machine translation has shown that encoder-decoders outperform the more widely used decoder-only formulation of the Transformer on similar parameter sizes (Pitorro et al., 2024). As SCAN is framed and modeled as a sequence-to-sequence task, the encoder-decoder formulation is the most reasonable choice. We use absolute position encodings for the two main experiments, as well as Gated Linear Units as the activation function (Shazeer, 2020). Details on the implementation and choice of hyperparameters can be found in Appendix B.3.

Permutation-equivariant model. Lastly, we experiment with a model that has built-in architectural priors. We use the encoder-decoder from Gordon et al. (2019), which enforces the verb equivariance

found in SCAN using a cyclic group. This group is used in the encoding of the input sequences in a way that pools the representation of any verb with the representation of the other verbs, even when they are not present in the input. This is a strong structural prior that enables the model to solve the original SCAN splits efficiently. However, this approach is dataset-dependent, as the elements of the cyclic group must be defined manually. We use this model primarily as a sanity check for our experiments, but also as a comparison for the other architectures, comparing their efficiency to an idealized case.

4.2 Experiment 1: Vertical scaling

For Experiment 1, p^{test} has a degenerate distribution for e_1, where $v_1 \in V$ always occur, while e_2 has a uniform distribution over V. For p^{train}, e_1 has a uniform distribution over $V \setminus \{v_1\}$, and e_2 is a mixture of a degenerate distribution at v_1 and a uniform distribution over $V \setminus \{v_1\}$, where we increase the total samples drawn from the uniform while decreasing the number drawn from the degenerate distribution. By doing this, we increase H as described in section Section 3.4.1. We refer to this increase in H as *vertical scaling*, as the distribution over the verbs increases towards the uniform. The models are trained on $6\,000$ unique samples and evaluated on $7\,056$ samples for all values of H.

4.3 Experiment 2: Horizontal Scaling

For Experiment 2, p^{test} is the same as for Experiment 1. For p^{train}, e_1 is also the same as for Experiment 1, but e_2 is a uniform distribution over an increasing number of verbs, starting with $\{v_1\}$ and ending at V, as described in Section 3.4.2. We refer to this increase in H as *horizontal scaling*, as the increase of H is achieved by adding another verb to the support of e_2. The models are trained on all grammatical combinations in Σ^* permitted by the support at each level of H. When $H = 2$, for example, the support of the distribution of e_2 contains four verbs, and the training set contains all possible

---

[3]Data generation scripts and code to reproduce our experiments can be found at `https://github.com/ltgoslo/systematicity-entropy`.

(a) Experiment 1: Vertical scaling of entropy (b) Experiment 2: Horizontal scaling of entropy
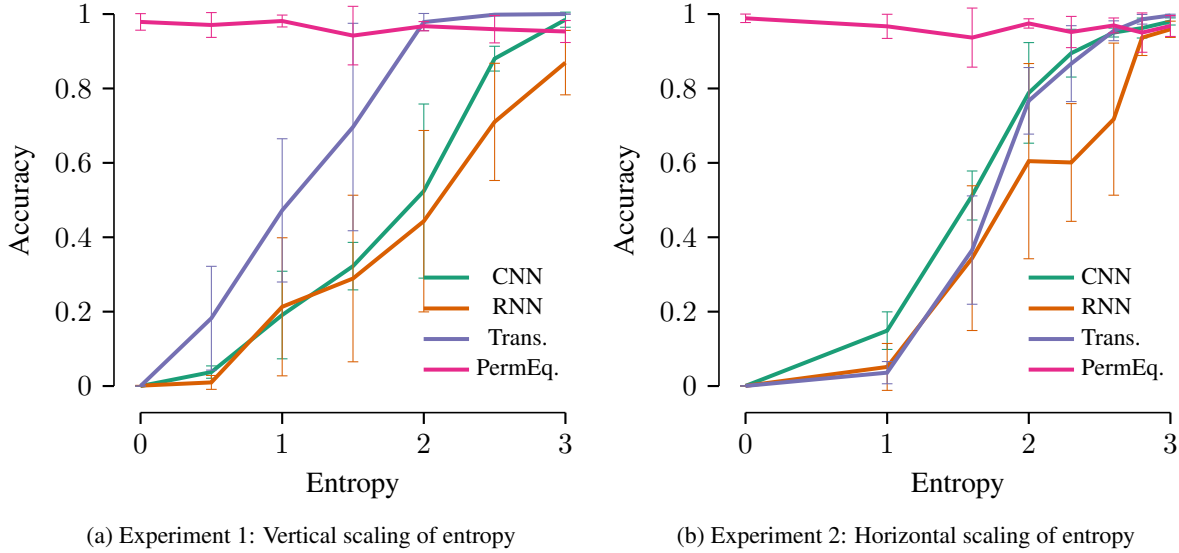
Figure 5: Results from the Experiments 1 & 2. The accuracy and standard deviations are from five seeds.

pairs $(x, y) \in X \times Y$ under these constraints.

To ensure a fair comparison across entropy levels, we train all models for the same number of optimization steps, so for $H = 1$, which has fewer training samples than $H = 3$, we do more iterations over the full data.

4.4 Effect of Number of Unique Samples

As H describes the relative distribution of verbs in e_2, it is possible to construct datasets of varying sizes that have the same H. This can be achieved by increasing the total volume of unique samples while maintaining the same relative distribution. To verify that performance scales with H independently of the number of unique samples seen during training, we create datasets that have the same entropy as in the previous experiments, but where we vary the number of total unique samples. We experiment with three different sample sizes: 3 000, 4 000, and 6 000, using the vertical scaling setting. All models run for the same number of optimization steps for all sample sizes.

4.5 Effect of Positional Embedding Type

In our experiments, there is an intuitive connection between the generalization from p^{train} to p^{test} and the ability to encode verb equivariance. This is exemplified by Gordon et al. (2019), which achieves high performance on the original SCAN tasks precisely by enforcing verb equivariance in the model architecture explicitly.

As the verbs always occur on the first token position of e_1 and e_2, positional information is central

to our experiments. To estimate the effect of positional information, we show how the choice of positional encoding scheme affects performance in the vertical scaling setting. In addition to the original absolute position embeddings from Vaswani (2017), we experiment with embeddings that encode relative positional information through disentangled attention (He et al., 2021) and rotary position embeddings (RoPE; Su et al., 2024). In this experiment, we focus on the Transformer. Details on hyperparameter selection can be found in Appendix B.4

4.6 Results and Discussion

The results of Experiment 1 and Experiment 2 can be found in Figure 5. We observe a clear positive relationship between performance and H for both experiments, demonstrating that systematic generalization scales with entropy. Unsurprisingly, the permutation-equivariant model, which has a strict compositional prior, solves both tasks at all entropy levels. This shows that the task is solvable even at $H = 0$ when enforcing such a prior, but that vanilla architectures generally require much higher entropies for this type of generalization.

In Experiment 1, the Transformer outperforms both the RNN and CNN across all levels of H. In Experiment 2, the performances of all models are statistically similar, falling within one standard deviation of each other, but with the CNN performing the best on average in the reported runs.

We also observe that for the Transformer, a reduction in the support of the distribution is more

(a) Effect of unique sample size on the Transformer

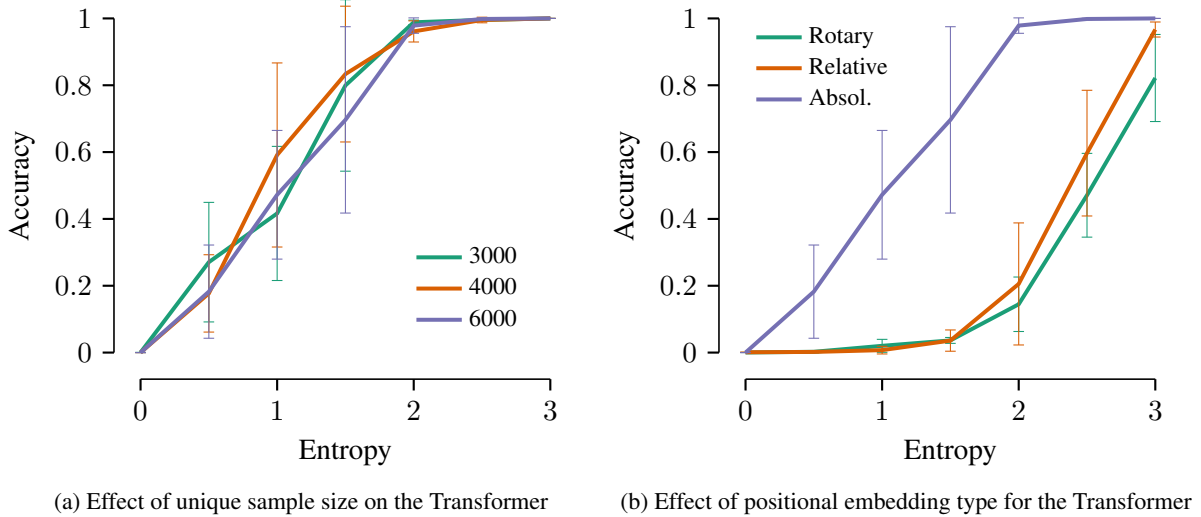(b) Effect of positional embedding type for the Transformer

Figure 6: The result of the supplemental experiments. The accuracy and standard deviations are from five seeds.

detrimental to the performance than a step towards the degenerate distribution: At H = 2, the Transformer has close to 100% accuracy for Experiment 1, but for the same level of H in Experiment 2, it achieves around 80% accuracy. This is not the case for the RNN and CNN, where we observe the opposite: The increase in support that increases H from 1 to 2 in Experiment 2 gives a higher performance increase than what the same increase in H does for Experiment 1.

We argue that these results indicate that the Transformer is more information-efficient than the RNN and CNN under full support, but that the models are similar under lacking support. In practice, this means that the Transformer is quicker to generalize to novel uses of component parts, as long as the overall syntactic pattern—constituted by the conjunctive phrase in our experiments—is frequent in the training data. This interpretation aligns with the findings of Loula et al. (2018), showcasing how our framework can provide a theoretical explanation for previous results.

We note that even though all models can generalize under high entropy for both experiments, their performance is subpar for low entropy, e.g. H = 1. This indicates a rather high lower bound on the entropy required for systematic generalization, both for vertical and horizontal scaling, and it shows that all models are unable to meet requirement *iii* from Definition 3. Furthermore, our results show that systematic generalization remains an open problem when there is no prior, and it remains unclear whether or not it is possible to build a model that performs well on low entropy without architectural

priors as in Gordon et al. (2019) or priors in the training procedure as in Lake and Baroni (2023).

4.6.1 Supplemental results

In Figure 6a, we show the results of varying the number of unique samples at each level of H. We observe that the positive relationship between performance and H is independent of the number of unique samples. The Transformer performs similarly when trained on 3 000 to 6 000 unique samples. During hyperparameter selection (Appendix B), we also observe that performance does not scale with model size. This indicates that achieving higher accuracy on lower levels of H is not achievable by scaling the number of unique data samples or model size for the same architecture.

In Figure 6b, we report the performance of different positional encoding types. Although all encoding types achieve near-ceiling performance at high entropy, absolute encoding performs better for all levels of H. From our task formulation in Section 3, it is clear that positional information is crucial for solving the tasks. We attribute the efficiency of absolute encodings to the fact that the positional information of the verbs is independent of the remainder of the sequence. This invariance has to be learned by the other embedding types, making the absolute variant more information-efficient.

5 Previous Work

In this work, we base our experiments on a version of the SCAN grammar from Lake and Baroni (2018). The original grammar has been studied in

other works, such as Loula et al. (2018); Dessì and Baroni (2019); Gordon et al. (2019), and there has also been work on the related PCFG SET dataset from Hupkes et al. (2020).

The work most similar to ours is Zhou et al. (2023), who also studies how data-centric aspects affect compositional generalization. In our work, we see compositionality as characterized by a set of primitives composed by a composition function, and we focus on how models capture this composition function depending on properties of the primitives. Their work, however, does not make this distinction, as their complexification process modify both the composition function and the distribution of primitives. The main distinction is hence that our work relates compositionality to dataset-level regularities, while they relate it to sample-level features.

Our work is also similar to Keysers et al. (2020), who quantifies the level of systematic generalization using distributional information. Specifically, they quantify the compositionality of an experiment as the divergence between the distribution of component parts and combinations in the training and test data. Their work differs from ours since they focus on the case where the distribution of component parts is as similar as possible between the training and test data, while our work focuses on the case where the distribution of component parts is inverted from the training to the test set for some parts of the input.

In another domain, Wiedemer et al. (2023) approaches systematicity in an image generation problem by factorizing the latent variables of the input image. The authors show that the support of these variables must be the same in the training and test distributions for any model to generalize systematically. We take a more fine-grained approach and our results demonstrate that support alone is insufficient as a predictor of systematic generalization; the performance also depends on the entropy, and higher levels of entropy in the training distribution can compensate for lower support.

There is also recent work on formalizing compositionality in terms of data complexity (Elmoznino et al., 2024), kernel-theory (Lippl and Stachenfeld, 2025), using graph formalisms (Ram et al., 2024), and decomposition strategies (Jarvis et al., 2023). Our work is similar to these in the sense that we try to formalize some properties of systematic generalization, which is closely tied to compositionality.

6 Conclusion

In this work, we show that systematic generalization of sequence-to-sequence models scales with information entropy. Through supporting experiments, we also show that this relationship is independent of the number of unique samples seen during training. Contrary to previous work, our findings demonstrate that these models are capable of systematic generalization even without any built-in architectural priors that incentivize compositional solutions. However, this requires that the training data have high entropy w.r.t. the parts that the target composition function operates on. This poses a new question: Is it possible to achieve systematicity also at low entropy *without* compositional priors? Through our formalizations, we hope to facilitate future work that attempts to answer this question, providing a clear method for assessing progress in systematic generalization research.

Limitations

Scope of our findings. In this work, we have considered an aspect of systematicity that is concerned with embedded sentences as defined by Hadley (1994). There is no guarantee that our findings generalize to scenarios that are different from ours, but that still falls in under some definition of systematic generalization.

Use of synthetic data. As is commonplace in compositionality research, our work relies on synthetic data that is generated by a CFG. This is necessary to quantify the entropy and to get a controlled experimental setup where the distribution of component parts in the embedded sentences can be measured precisely. This is generally not tractable for real-world corpora. Consequently, our method is constrained to cases where fine-grained distributional information is obtainable.

Acknowledgements

References

Noam Chomsky. 1957. *Syntactic Structures*. Mouton.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Roberto Dessì and Marco Baroni. 2019. CNNs found to jump around more skillfully than RNNs: Compositional generalization in seq2seq convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3919–3923, Florence, Italy. Association for Computational Linguistics.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Jeffrey L Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7:195–225.

Eric Elmoznino, Thomas Jiralerspong, Yoshua Bengio, and Guillaume Lajoie. 2024. A complexity-based theory of compositionality. *arXiv preprint arXiv:2410.14817*.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2019. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Robert F Hadley. 1994. Systematicity in connectionist language learning. *Mind & Language*, 9(3):247–272.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*.

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

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, 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. 2023. A taxonomy and review of generalization research in NLP. *Nature Machine Intelligence*, 5(10):1161–1174.

Devon Jarvis, Richard Klein, Benjamin Rosman, and Andrew M Saxe. 2023. On the specialization of neural modules. In *The Eleventh International Conference on Learning Representations*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Brenden M Lake and Marco Baroni. 2023. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121.

Samuel Lippl and Kim Stachenfeld. 2025. When does compositional structure yield compositional generalization? a kernel theory. In *The Thirteenth International Conference on Learning Representations*.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, Brussels, Belgium. Association for Computational Linguistics.

Kate McCurdy, Paul Soulos, Paul Smolensky, Roland Fernandez, and Jianfeng Gao. 2024. Toward compositional behavior in neural models: A survey of current views. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9323–9339, Miami, Florida, USA. Association for Computational Linguistics.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *Preprint*, arXiv:2410.05229.

M Ott. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Peter Pagin and Dag Westerståhl. 2010. Compositionality i: Definitions and variants. *Philosophy Compass*, 5(3):250–264.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Steven Pinker. 1989. *Learnability and cognition: The acquisition of argument structure*. MIT Press.

Hugo Pitorro, Pavlo Vasylenko, Marcos Treviso, and André Martins. 2024. How effective are state space models for machine translation? In *Proceedings of the Ninth Conference on Machine Translation*, pages 1107–1124, Miami, Florida, USA. Association for Computational Linguistics.

Parikshit Ram, Tim Klinger, and Alexander G Gray. 2024. What makes models compositional? a theoretical view. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 4824–4832.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

JFAK van Bethem, JAG Groenendijk, DHJ de Jong, MJB Stockhof, and HJ Verkuyl. 1991. *Logic, Language, and Meaning, Volume 2: Intensional Logic and Intensional Grammar*. University of Chicago Press, Chicago, IL.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. 2023. Compositional generalization from first principles. In *Advances in Neural Information Processing Systems*, volume 36, pages 6941–6960. Curran Associates, Inc.

Sondre Wold, Étienne Simon, Lucas Charpentier, Egor Kostylev, Erik Velldal, and Lilja Øvrelid. 2024. Compositional generalization with grounded language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3447–3460, Bangkok, Thailand. Association for Computational Linguistics.

Xiang Zhou, Yichen Jiang, and Mohit Bansal. 2023. Data factors for better compositional generalization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14549–14566, Singapore. Association for Computational Linguistics.

A Modified SCAN grammar

V	A
look	left
jump	right
run	opposite left
walk	opposite right
sprint	around left
crawl	around right
squat	twice
lunge	thrice

Table 2: The verbs and adjective phrases in our modified version of SCAN.

Our modified SCAN grammar consists of the same parts of speech as the original, but we extend the set of verbs V with four additional items. We keep the original eight adjective phrases. The verbs and adjective phrases can be seen in Table 2. Every verb can be combined with every adjective phrase in each embedded sequence (under no distributional constraints), combined with one of the two conjunctive phrases from SCAN (and, after).

B Model details

All models were implemented in Pytorch (Paszke et al., 2019). We conducted hyperparameter searches for each model per experiment. Below is a list of the search space used for each. All configurations used a weight decay of 1×10^{-1} and a 10% dropout rate.

B.1 RNN

We use the basic sequence-to-sequence implementation from Gordon et al. (2019), which is based on the Pytorch MT tutorial. We conduct a grid search over the following learning rates: $\{1 \times 10^{-3}, 1 \times 10^{-4}, 3 \times 10^{-4}\}$; cell type: Elman RNN (Elman, 1991), LSTM (Hochreiter and Schmidhuber, 1997), GRU (Chung et al., 2014); number of hidden units: $\{64, 128, 256\}$; number of layers: $\{1, 2, 3\}$; and teacher-forcing rate: $\{0.0, 0.5\}$. All models ran with an attention mechanism between the encoder and decoder, bidirectional layers, and with no learning rate decay. Following previous work, we used a batch size of 1.

The RNN used for the experiment outlined in Section 4.2 has a learning rate of 1×10^{-4}, normal RNN cell types, a hidden size of 128 and 2 layers (483 438 parameters), while for the experiment

outlined in Section 4.3 it has 1 layer and 64 hidden units (112 238 parameters). Teacher-forcing did not result in increased performance for either experiment.

B.2 CNN

We based our implementation of the CNN from Gehring et al. (2017) on the one found in FAIRSEQ (Ott, 2019). We conduct a grid search over the following learning rates: $\{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 3 \times 10^{-4}\}$; kernel size: $\{3, 5\}$; number of hidden units: $\{64, 128, 256\}$; and number of layers: $\{1, 2, 3\}$; We train with a batch size of 32 and a cosine decay on the learning rate.

The CNN used for the experiment outlined in Section 4.2 has a learning rate of 3×10^{-4}, a kernel size of 5, a hidden size of 64 and 3 layers (298 078 parameters), and the same configuration for Section 4.3.

B.3 Transformer

We implement the Transformer following the original from Vaswani (2017). We conduct a grid search over the following learning rates: $\{1 \times 10^{-3}, 1 \times 10^{-4}, 3 \times 10^{-4}, 1 \times 10^{-5}\}$; number of hidden units: $\{64, 128, 256\}$; and number of layers: $\{1, 2, 3\}$; As for the CNN, we train with a batch size of 32 and a cosine decay on the learning rate.

The Transformer used for the experiment outlined in Section 4.2 has a learning rate of 3×10^{-4}, a hidden size of 128 and 3 layers (1 796 894 parameters), while for the experiment outlined in Section 4.3 it has 2 layers and 256 hidden units (4 763 166 parameters).

B.4 Positional Embedding

In addition to the absolute encodings used in Vaswani (2017), we implement two other variations of positional embeddings: RoPE Su et al. (2024) and Disentangled He et al. (2021). For all variations, we use the same hyperparameters found in the Transformer search (Appendix B.3). Additionally, we conduct a hyperparameter search for the encoding-specific parameters. For RoPE we conduct a search for the base value of the θ. We tried powers of 10, from one to one million. For Disentangled Attention we sweep over the input length for the relative attention. We have a separate length for the input and output encodings. We sweep over the following values: $\{4, 8, 16, 32, 64\}$. As for the Transformer, we train with a batch size of 32 and a cosine decay on the learning rate.

For the models used for the experiment outlined in Section 4.5, we use a RoPE θ value of 1 000, and for the disentangled encodings we use a relative input length of 64 and a relative output length of 4.

B.5 Permutation equivariant sequence model

For the permutation equivariant model, we use the original implementation from Gordon et al. (2019). We extend the cyclic group from four to eight verbs. For the hyperparameter selction, we conducted a grid search over the following learning rates: $\{1 \times 10^{-3}, 1 \times 10^{-4}, 3 \times 10^{-4}, 1 \times 10^{-5}\}$; the number of hidden units: $\{64, 128\}$; and RNN cell type: Elman RNN, GRU. All settings ran with 1 layer.

The model used for the experiment outlined in Section 4.2 has a learning rate of 1×10^{-4}, a hidden size of 64 and the GRU cell types ($\approx 130\,000$ parameters), and the same configuration for Section 4.3.

We note that the model presented in Lake and Baroni (2023) also encodes verb equivariance and achieves high performance on SCAN. However, this approach relies on automatically augmenting the training data by creating examples of the input that have different semantics than the original grammar, making our quantification of H impractical. Consequently, we only evaluate the permutation-equivariant model from Gordon et al. (2019).

C Computational budget

All hyperparameter sweeps and experiments ran on AMD EPYC 7763 CPUs:
- RNN sweep: 2880 hours.
- CNN sweep: 139 hours.
- Transformer sweep: 120 hours.
- Permutation equivariant sweep: 24 hours.
- RNN training run: 600 hours.
- CNN training run: 21 hours.
- Transformer sweep: 120 hours.
- Evaluation: 12 hours.
- **Total:** 3916 core CPU hours.

D Licenses and use

The present work uses the following scientific artifacts:

1. We base our synthetic data on the SCAN grammar (Lake and Baroni, 2018). SCAN is released under a BSD License. We do not redistribute or use any of the source code, nor do we redistribute or use any of the original data files. We implement the grammar based

on the description provided in the paper. Our code implementation of the grammar includes the original license statement from SCAN.

2. Our use of the permutation equivariant model from Gordon et al. (2019) uses the original implementation. The code is released under an MIT license. All source code files redistributed with our work contain the original license statement, as per the license requirement.

3. The Transformer (Vaswani, 2017) and the CNN (Gehring et al., 2017) is implemented by the authors in PyTorch (Paszke et al., 2019) and do not use existing code implementations. However, the CNN was largely based on the implementation found in Fairseq (Ott, 2019), which is released under an MIT license. We do not redistribute any of the original code files.

E The use of AI assistants

The authors used AI assistants for spellchecking and sentence-level editing in the writing of the current manuscript.