Revisiting the Compositional Generalization Abilities of Neural Sequence Models

Arkil Patel[a] **Satwik Bhattamishra**[b] **Phil Blunsom**[b] **Navin Goyal**[a]

[a] Microsoft Research India

[b] University of Oxford

arkil.patel@gmail.com, navingo@microsoft.com

{satwik.bmishra,phil.blunsom}@cs.ox.ac.uk

Abstract

Compositional generalization is a fundamental trait in humans, allowing us to effortlessly combine known phrases to form novel sentences. Recent works have claimed that standard seq-to-seq models severely lack the ability to compositionally generalize. In this paper, we focus on one-shot primitive generalization as introduced by the popular SCAN benchmark. We demonstrate that modifying the training distribution in simple and intuitive ways enables standard seq-to-seq models to achieve near-perfect generalization performance, thereby showing that their compositional generalization abilities were previously underestimated. We perform detailed empirical analysis of this phenomenon. Our results indicate that the generalization performance of models is highly sensitive to the characteristics of the training data which should be carefully considered while designing such benchmarks in future.

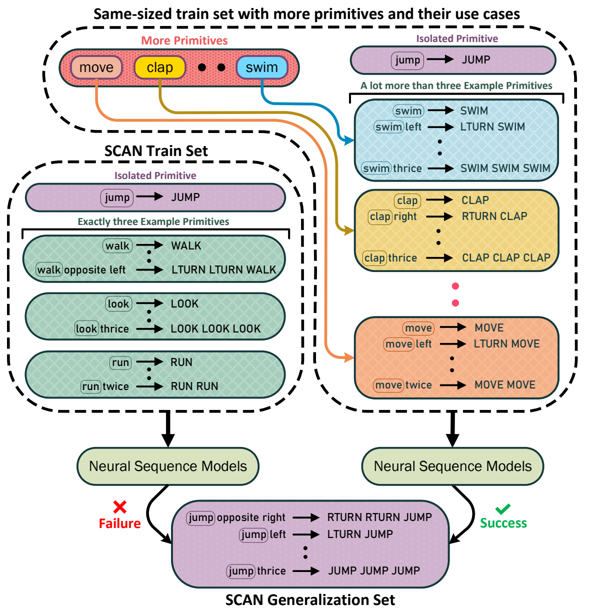


Figure 1: Overview of the SCAN generalization task (left) and our approach (right) that enables standard neural sequence models to generalize compositionally.

1 Introduction

According to the *principle of compositionality*, the meaning of a complex expression (e.g., a sentence) is determined by the meaning of its individual constituents and how they are combined. Humans can effectively recombine known parts to form new sentences that they have never encountered before. Despite the unprecedented achievements of standard seq-to-seq networks such as LSTMs and Transformers in NLP tasks, previous work has suggested that they are severely limited in their ability to generalize compositionally (Lake and Baroni, 2018; Furrer et al., 2020).

Problem Statement. Our work relates to a central challenge posed by compositional generalization datasets such as SCAN (Lake and Baroni, 2018) and Colors (Lake et al., 2019), which we refer to as *one-shot primitive generalization*: The dataset consists of *input-output sentence* pairs (e.g. 'walk twice → WALK WALK'); input sentences are formed from primitive words ('walk') and function words ('twice') and are generated by a context-free grammar (CFG); output sentences are obtained by applying an interpretation function. Crucially, there is a systematic difference between the train and test splits[1]: While the former has a *single* example of an *isolated primitive* (e.g., the primitive definition 'jump → JUMP' in SCAN), the latter consists of compositional sentences with this isolated primitive (e.g. 'jump twice → JUMP JUMP'). See Fig. 1 (left) for an overview of the task.

A model with the right inductive bias should generalize on the test data after having seen compositional expressions with other primitives during training. The need for such inductive bias is justified via psychological experiments (Lake et al., 2019) indicating that humans do have the ability to

[1] We use the term *systematicity* in the rest of the paper to refer to this difference.

generalize on such tasks. Previous works have suggested that seq-to-seq models lack the appropriate inductive bias necessary to generalize on this task since they achieve near-zero accuracies on both SCAN and Colors benchmarks. This has led to the development of many specialized architectures (Li et al., 2019; Gordon et al., 2020; Chen et al., 2020; Akyurek and Andreas, 2021), learning procedures (Lake, 2019; Conklin et al., 2021) and data augmentation methods (Andreas, 2020; Guo et al., 2020) to solve the task.

Contributions. The primary claim of our paper is that, contrary to prior belief, neural sequence models such as Transformers and RNNs do have an inductive bias[2] to generalize compositionally which can be enabled using the right supervision. **(i)** We show that by making simple and intuitive changes to the training data distribution, standard seq-to-seq models can achieve high generalization performance even with a training set of size less than 20% of the original training set. In particular, if we incorporated examples with more novel primitives in the training set without necessarily increasing the size of the training set (see right part of Fig. 1), then the generalization performance of standard seq-to-seq models improves and reaches near-perfect score after a certain point. Our results also exemplify the importance of the training distribution apart from architectural changes and demonstrate that providing the right supervision can significantly improve the generalization abilities of the models. **(ii)** We investigate the potential cause behind the improvement in generalization performance and observe that the embedding of the isolated primitive becomes more similar to other primitives when the training set has higher number of primitives and their use cases. **(iii)** To understand the phenomenon better, we characterize the effect of different training distributions and model capacities. Our results show that the parameters of the experimental setting play a crucial role while evaluating the generalization abilities of models.

2 Enabling Generalization by Providing the Right Supervision

Setup. We focus on the SCAN and Colors datasets.[3] Both these datasets have exactly one *isolated primitive*. We refer to all other primitives

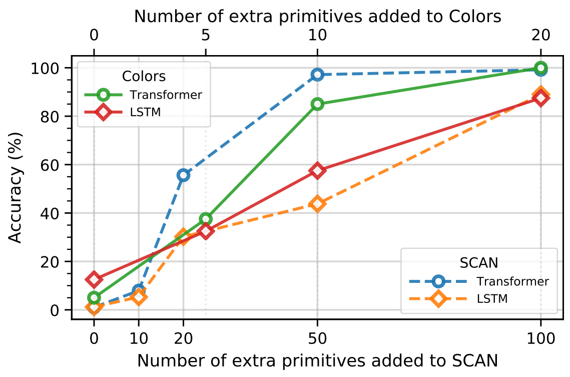


Figure 2: Generalization performance (↑) on SCAN and Colors improves with higher number of example primitives in the training set.

(i.e., those that are also composed with other words to form sentences in the training set) as *example primitives*. Both the SCAN and Colors training sets have exactly three example primitives. The training set of SCAN has 13.2k examples while the test set has 7.7k examples. Colors has just 14 training examples and 8 test examples. More details on implementation and datasets can be found in Appendix A & B. Our source code is available at https://github.com/arkilpatel/Compositional-Generalization-Seq2Seq.

Adding More Primitives. We modify the training set such that the number of distinct example primitives present in the dataset is higher. To do so, we add new primitives to the language which are simply random words (e.g., 'swim', 'clap', etc.) that have the same semantics and follow the same grammar rules as other existing primitives (see Fig. 1 (right) for illustration). These new primitives act as example primitives in our training set. For SCAN, we control the size of the training set such that it is at most the size of the original dataset.[4] To generate the training set, we randomly sample the examples from the new grammar and discard all compositional sentences with the isolated primitive. For each example primitive and the isolated primitive, a primitive definition (such as 'walk → WALK') is also added to the training set. The test set is untouched and remains the same.

Main Observation. Fig. 2 shows the generalization performance of Transformer and LSTM based seq-to-seq models. We observe that there is a clear trend of improvement in compositional gen-

[2]However, note that this inductive bias is not as strong as that of specialized architectures designed for these tasks.

[3]Results on COGS (Kim and Linzen, 2020) can be found in Appendix C.

[4]The training set size $|T|$ is kept fixed by discarding original examples and adding $(|T|/\#primitives)$ examples per primitive. Because of extremely small data size, we cannot do this for Colors while also trying to illustrate our idea.

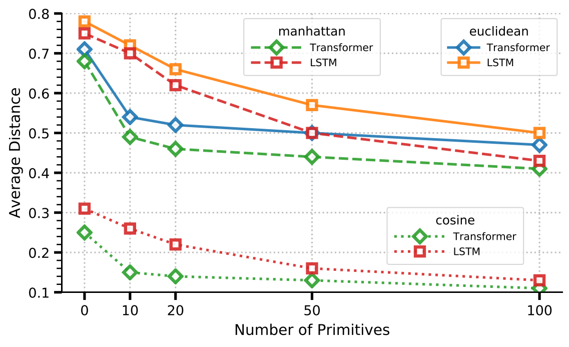


Figure 3: Measuring the distance of embedding of *isolated primitive* with embeddings of example primitives for learned Transformer and LSTM models as we increase the number of example primitives in SCAN.

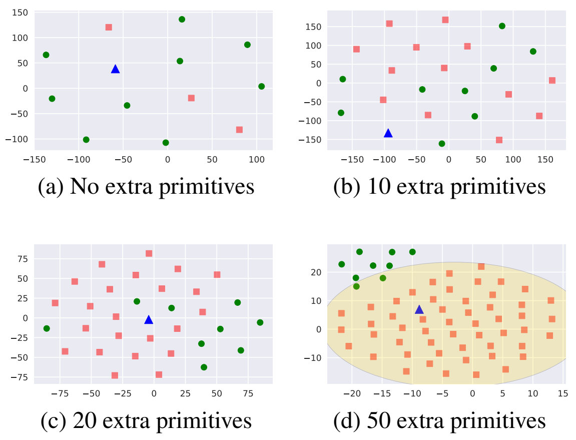


Figure 4: Visualizing the t-SNE reduced embeddings of isolated primitive (▲), example primitives (■) and non-primitives (●) from a learned Transformer model as we increase number of example primitives in SCAN.

eralization as we increase the number of example primitives and their use cases. It is surprising to see that on SCAN, Transformers perform on par with some recently proposed specialized architectures (Li et al., 2019; Gordon et al., 2020) and even better than certain architectures (Russin et al., 2019).

Implication. Since the training set still contains only one non-compositional example with the isolated primitive[5] and the test set is untouched, one-shot primitive generalization setting is preserved. Hence our results clearly show that standard neural sequence models have 'some' inductive bias required to generalize on such out-of-distribution tasks even if it is not as strong as that of specialized architectures designed primarily to solve these tasks. Our results are in contradiction to previously suggested limitations of standard seq-to-seq models in terms of primitive generalization (Lake and Baroni, 2018; Furrer et al., 2020; Baroni, 2020). While it is important to develop architectures with better compositional generalization abilities, we wish to highlight that synthetic benchmarks such as SCAN require a model with very strong inductive biases and tend to underestimate the generalization abilities of baseline models.

While we have shown that these models can generalize from one-shot exposure to primitive definitions, our results also hold for the more general case where the one-shot exposure of the primitive is in a sentence (e.g. 'jump twice $\rightarrow$ JUMP JUMP'). More details regarding these experiments can be found in Appendix D.

Prior Work. Note that our work is unrelated to previous works that propose data augmentation approaches for compositional generalization tasks (Andreas, 2020; Guo et al., 2020; Akyürek et al., 2021). (1) The datasets created by some of these augmentation methods do not preserve the systematic differences between train and test sets, while our datasets do.[6] (2) The objective of these works was to devise a method to improve compositional generalization performance whereas the focus of our work is not to develop a general method; rather we want show that **baseline seq-to-seq models are capable of generalizing compositionally even without breaking systematicity**. (3) These methods add additional data resulting in datasets of larger sizes whereas we control for data size.

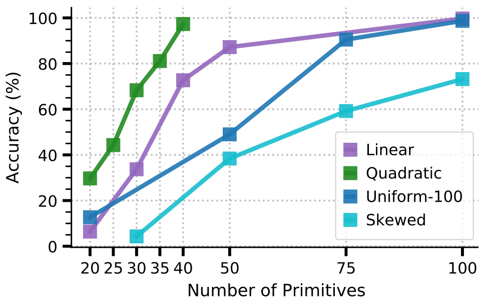
2.1 Analyzing the Embedding of the Isolated Primitive

Our results raise the question: Why do Transformers and LSTMs generalize better when the training data has more example primitives? Compositional generalization in our setting requires a model to learn to apply the same rules to the isolated primitive as it does to the other example primitives. Thus, we analyze the change in the learned embedding of the isolated primitive (such as 'jump') with respect to other primitives in different settings.

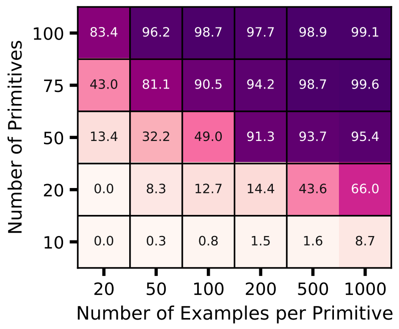
In particular, we compare the average distance with other primitives before and after adding certain number of primitives to training data (this is the same setting that was explained earlier in this section). We find that as we increase the number of example primitives in the training set, the em-

[5]Note that our results also hold when there are multiple isolated primitives in the dataset at the same time. This is discussed in Appendix E.5.

[6]We discuss this in more detail in Appendix F.



(a) Other Distributions



(b) Uniform Distribution

Figure 5: Measuring the generalization performance of Transformer on different types of training set distributions of the SCAN dataset.

bedding of the isolated primitive gets closer to the example primitives (Fig. 3) in terms of Euclidean, Manhattan and Cosine distances. If the embedding of the isolated primitive is closer to the embeddings of the other primitives, then the model is more likely to operate over it in a similar fashion and apply the same rules as it does over the other primitives.

This phenomenon is also illustrated in t-SNE plots (Fig. 4) of the learned embeddings where the embedding of the isolated primitive seems closer to the embeddings of the example primitives when there are more example primitives in the dataset. Hence, a possible reason behind improved generalization performance could be the difference in the learned embeddings.[7] Additional results with the LSTM model and Colors dataset can be found in Appendix E.1.

3 Exploring the Impact of the Parameters of the Experimental Setup

3.1 Impact of Training Distributions

In this section, we analyze the influence of different training distributions on the generalization performance of the model. In the previous experiments, the data generating distribution was uniform over all possible samples. Here, we alter the training data distribution by varying the number of examples for each example primitive. The test set remains unchanged and there will still be only one non-compositional example of the isolated primitive (i.e., the primitive definition) in the training set. We experiment with linearly, quadratically and exponentially increasing probability distribution functions. For instance, in the quadratically increasing case, a training set with 10 example primitives will have one example primitive with 1 compositional example, the next one with 4 compositional examples, another one with 9 compositional examples and so on.[8] Similarly, in the exponentially increasing case (which we also call ‘skewed’), 10% example primitives have 500 compositional examples each, 30% have 10 compositional examples each and the remaining have just one compositional example each in the training set. The general idea is that all the example primitives do not have equal representation in the training data. Upon training the models on different distributions, we observed that the models generalize well even with fewer number of example primitives when their distribution is linearly or quadratically increasing (Fig. 5a). On the other hand models struggle to generalize when the distribution is skewed. In that case, most primitives appear in only one or very few compositional sentences in the training data. The failure to generalize on such data implies that extra primitives must be added as part of multiple compositional sentences; just adding the primitive definition or a single example for each example primitive does not help the model to leverage it.

We then try to characterize the relationship between the number of example primitives and the amount of data required for the model to generalize well on the test data, when the example primitives are uniformly distributed. We create different training sets by varying the total number of example primitives, $\#primitives$; for each example primitive, we draw $\#examples$ number of samples uniformly from the CFG. Fig. 5b shows the generalization performance of Transformers for each of these training sets. The size of each training set is the product of the row and column values ($\#primitives \times \#examples$). As expected, the

[7] More fundamental reasons for difference in learned embeddings, such as learning dynamics, are beyond our scope.

[8] In all experimental setups considered in this paper, each example primitive will always have a primitive definition in the training set.

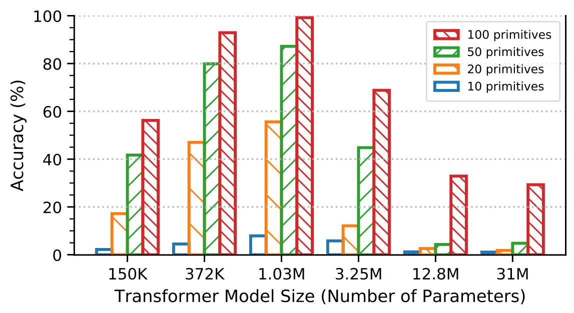


Figure 6: Measuring the generalization performance of a Transformer of varying capacity across increasing number of primitives in the SCAN training set.

upper-right triangle has higher scores indicating that the sample requirement decreases as we add more primitives to the dataset. Surprisingly, the top-left cell indicates that Transformers can achieve high performance even with 2k training examples which is less than **20%** of the original SCAN training set. Additional results with the LSTM model can be found in Appendix E.2.

3.1.1 Understanding Transferability

We wish to check whether the inductive bias that is enabled when a model is trained on more number of example primitives can be transferred to a scenario where the number of example primitives is limited. We create a *pretraining* set with 50 example primitives uniformly distributed, each of them having 200 examples. The *finetuning* set is the original SCAN training set and the test set is the original SCAN test set. The model is first trained from scratch on the pretraining set and then finetuned on the finetuning set.

We find that if we allow all the parameters of the Transformer model to be updated during the finetuning phase on the original SCAN training set, then the model generalizes very poorly. On the other hand, when we freeze the weights of the encoder and decoder after the pretraining phase, and only allow the embedding and output layers to be updated, then the model generalizes near-perfectly on the test set. Our hypothesis is that in the latter setting, the task becomes simpler for the model since it only has to align the embeddings of the newly seen primitives in the finetuning phase with the embeddings of the primitives seen during the pretraining phase. This experiment also indicates that the previously learned rules during pretraining can help a model to compositionally generalize on novel primitives.

3.2 Impact of Model Capacity

We analyze the relationship between the model capacity and the number of example primitives in the training set. We vary the number of primitives as per the description in Section 2. We evaluate the generalization performance of the models while gradually increasing the number of parameters by increasing the size of its embeddings and intermediate representations. For each experiment, we exhaustively finetune the rest of the hyperparameters (e.g., dropout, learning rate, batch size, etc.) to select the best model. Looking at Fig. 6, we observe a general trend in which the model starts to overfit and has poor generalization performance as we increase the model size. Note that all these model configurations are able to achieve near-perfect accuracies on the SCAN random split that does not test for compositional generalization. This shows that carefully controlling the model size is important for achieving compositional generalization. On such small datasets, larger models might simply memorize the input-output mappings in the training set. Indeed, such memorization has been cited as a potential reason to explain why models fail at compositional generalization (Conklin et al., 2021). We also find that as we increase the number of example primitives, the models are less susceptible to overfitting and achieve relatively better generalization performance. Additional results with the LSTM model and Colors dataset can be found in Appendix E.3.

4 Conclusion

While it is essential to make progress in building architectures with better compositional generalization abilities, we showed that the generalization performance of standard seq-to-seq models (often used as baselines) is underestimated. A broader implication of our experiments is that although systematicity must be preserved when designing such benchmarks, it is imperative to carefully explore different parameters associated with the experimental setup to draw robust conclusions about a model's generalization abilities.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. We would also like to thank Kabir Ahuja, Zihuiwen Ye and our colleagues at Microsoft Research India for their valuable feedback and helpful discussions.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations*.

Ekin Akyurek and Jacob Andreas. 2021. Lexicon learning for few shot sequence modeling. 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 4934–4946, Online. Association for Computational Linguistics.

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

Marco Baroni. 2020. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B*, 375(1791):20190307.

Paul Bloom. 2000. *How Children Learn the Meanings of Words*. MIT Press.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. In *Advances in Neural Information Processing Systems*, volume 33, pages 1690–1701. Curran Associates, Inc.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. 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 3322–3335, Online. 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.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Demi Guo, Yoon Kim, and Alexander Rush. 2020. Sequence-level mixed sample data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5547–5552, Online. Association for Computational Linguistics.

Najoung Kim and Tal Linzen. 2020. 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.

Brenden 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*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Brenden M. Lake, Tal Linzen, and Marco Baroni. 2019. Human few-shot learning of compositional instructions.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4293–4302, Hong Kong, China. Association for Computational Linguistics.

Santiago Ontañón, Joshua Ainslie, Vaclav Cvicek, and Zachary Fisher. 2021. Making transformers solve compositional tasks.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jake Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics.

Ning Shi, Boxin Wang, Wei Wang, Xiangyu Liu, Rong Zhang, Hui Xue, Xinbing Wang, and Zhouhan Lin. 2021. From scan to real data: Systematic generalization via meaningful learning.

A Implementation Details

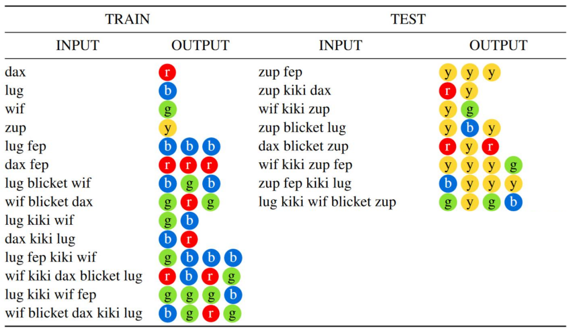
We use 8 NVIDIA Tesla P100 GPUs each with 16 GB memory to run our experiments. All models are implemented in PyTorch (Paszke et al., 2019). We do not use any pretrained models and all embeddings are learnt from scratch. Parameters are updated using Adam optimization. All results are an average of 5 different runs with random seeds. The dataset-specific hyperparameters used for each model are shown in Table 1.

B Primitive Generalization Datasets

In this paper, we show results on three datasets that evaluate primitive generalization.

SCAN (Lake and Baroni, 2018) is a supervised sequence-to-sequence semantic parsing task wherein the natural language input command has to be transformed to the corresponding set of actions. The complete dataset consists of all the commands (a total of 20,910) generated by a phrase-structure grammar and the corresponding sequence of actions, produced according to a semantic interpretation function. The benchmark consists of 4 splits: random, add jump, turn left and length. We work on the ‘add jump’ split which was designed to test primitive generalization. In this split, the test set (size: 7706) is made up of all the compositional sentences with the primitive ‘jump’ (which we refer to as the *isolated primitive*). The train set (size: 13,204[9]) has just one example of the isolated primitive (i.e. the primitive definition ‘jump $\rightarrow$ JUMP’) and other examples demonstrating the definitions and compositions of the three other primitives (which we refer to as the *example primitives*). Table 2 illustrates the task.

Colors (Lake et al., 2019) is a sequence-to-sequence task that was designed to measure human inductive biases. Apart from the challenge of primitive generalization, this dataset poses an additional challenge of low-resource learning for neural sequence models. The train set has just 14 examples that are either primitive definitions of the four primitives or examples with compositions of the three example primitives and three operations (concatenation, repetition and wrapping). The test set has 8 examples[10] with compositions of the isolated primitive (‘zup’). Fig. 7 illustrates the task.



TRAIN		TEST	
INPUT	OUTPUT	INPUT	OUTPUT
dax	r	zup fep	y y y
lug	b	zup kiki dax	r y
wif	g	wif kiki zup	y g
zup	y	zup blicket lug	y b y
lug fep	b b b	dax blicket zup	r y r
dax fep	r r r	wif kiki zup fep	y y y g
lug blicket wif	b g b	zup fep kiki lug	b y y y
wif blicket dax	g r g	lug kiki wif blicket zup	g y g b
lug kiki wif	g b		
dax kiki lug	b r		
lug fep kiki wif	g b b b		
wif kiki dax blicket lug	r b r g		
lug kiki wif fep	g g g b		
wif blicket dax kiki lug	b g r g		

Figure 7: The primitive generalization task in Colors[11]. Note that the test set does not contain the two length generalization examples.

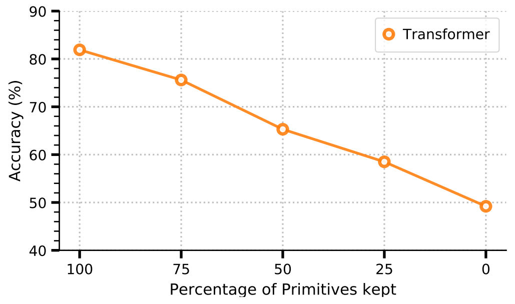


Figure 8: Decrease in generalization performance on our COGS primitive generalization test set with a decrease in the percentage of example primitives and their use cases present in the train set.

COGS (Kim and Linzen, 2020) is a semantic parsing task of mapping English natural language sentences to their corresponding logical forms. Apart from primitive generalization, COGS also evaluates other types of systematic generalization such generalizing to higher depths or generalizing to novel syntactic structures. The size of the train set is 24,155 and that of the test set is 21,000.

C Removing Primitives Hurts Generalization on COGS

Unlike SCAN and Colors, both of which have a single isolated primitive and only 3 example primitives, COGS has 3 isolated primitives - a verb, a common noun and a proper noun which are supported by 80 verbs, 40 common nouns and 20 proper nouns as example primitives. We hypothesize that this high number of example primitives might be one of the reasons behind the high performance of Transformers on COGS (Csordás et al.,

[9]The dataset released by (Lake and Baroni, 2018) is of size 14,670 which has many repetitions of the ‘jump $\rightarrow$ JUMP’ primitive definition. In this work, we remove all these repetitions since they do not significantly help in generalization.

[10]The original dataset has two additional examples which evaluate length generalization. Since we focus only on primitive generalization, we do not evaluate on these.

[11]Image taken from Akyurek and Andreas (2021).

	SCAN		**COLORS**		**COGS**
Hyperparameters	Transformer	LSTM	Transformer	LSTM	Transformer
Embedding Size	[64, **128**, 256]	[64, **128**, 256]	[16, **32**, 64]	[16, **32**, 64]	[**384**, 512]
Hidden/FFN Size	[**256**, 512]	[**64**, 128]	[16, **32**, 64]	[16, 32, **64**]	[**512**, 1024]
Heads	[**2**, 4]	N/A	[**4**, 8]	N/A	[2, **4**]
Number of Layers	[2, **3**]	[1, **2**]	[**2**, 3]	[1, **2**]	[**2**, 3]
Learning Rate	[3e-4, **5e-4**, 8e-4]	[5e-3, **8e-3**, 1e-2]	[**8e-4**, 1e-3]	[5e-3, **8e-3**, 1e-2]	[3e-4, **5e-4**, 8e-4]
Batch Size	[**128**, 256]	[128, **256**]	[**1**, 2]	[**1**, 2]	[**128**, 256]
Dropout	[**0.1**, 0.2]	[**0.1**, 0.2]	[**0.1**, 0.2]	[**0.1**, 0.2]	[**0.1**, 0.2]
Epochs	150	150	150	150	150
Avg Time/Epoch	30	40	2	3	60

Table 1: Different hyperparameters and the values considered for each of them in the models. The best hyperparameters for each model for all the datasets (with maximum number of primitives of all the settings studied in this paper) are highlighted in bold. Average Time/Epoch is measured in seconds.

TRAIN:	
jump	JUMP
run after run left	LTURN RUN RUN
run	RUN
look left twice	LTURN LOOK LTURN LOOK
TEST:	
jump twice after look	LOOK JUMP JUMP
turn left and **jump**	LTURN JUMP
jump right twice	RTURN JUMP RTURN JUMP

Table 2: An illustration of the primitive generalization task in SCAN.

2021; Ontañón et al., 2021), as far as primitive generalization is concerned.

To validate our hypothesis, we systematically reduce the number of example primitives in COGS and evaluate the model. The test set of COGS focusing on primitive generalization consists of 5000 examples. If we directly start removing the primitives from the train set, we risk having out-of-vocabulary tokens in the test set. Hence we select a portion of the test set of size 1218 which exludes 129 example primitives. We will hold this test set fixed and vary the percentage of the 129 example primitives to be inserted in the train set. For each example primitive, samples are drawn uniformly from the original COGS train set. Note that even though the number of example primitives and their use cases will vary in the train set, we control the total train set size to be always 2500 for fair evaluation.

The results of our experiment can be seen in Fig. 8. We see a clear trend of decrease in generalization performance as we decrease the number of example primitives and their use cases. This is in tandem with the results shown in Section 2 and further validates the idea that providing more example primitives and their use cases helps neural sequence models generalize on the primitive generalization task. Our results help explain that the gap in performance of neural sequence models on primitive generalization tasks in COGS and primitive generalization tasks in SCAN or Colors is at least partially caused by the difference in the number of example primitives and their use cases in these datasets.

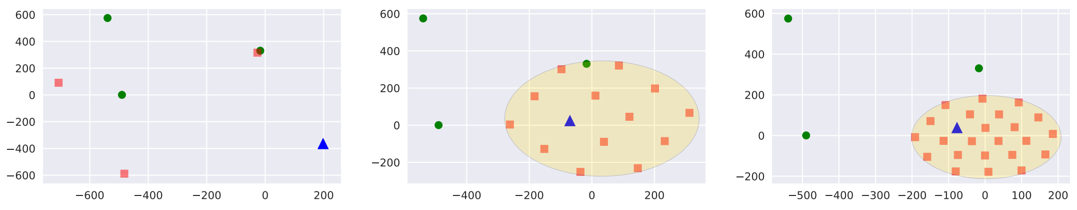
COMPLEXITY	SENTENCE
1	**jump** twice
2	**jump** thrice and look
3	run twice after **jump** opposite left
4	**jump** around left and walk opposite left twice

Table 3: Sentences of varying complexities featuring the isolated primitive 'jump'.

D Implicit Word Learning

Drawing analogy from human vocabulary acquisition (Bloom, 2000), our primitive generalization setting corresponds to the case when a child is explicitly explained the meaning of a word. But children can learn word meaning from implicit usage. In our setting this would translate to using a primitive in a more complex construction, say 'jump twice → JUMP JUMP' instead of the original 'jump → JUMP'. It would be interesting to evaluate how well seq-to-seq models learn the meanings of words from a single sentence and whether they learn to use that word compositionally with other words.

We consider the 'add jump' split in SCAN. Instead of providing the 'jump → JUMP' primitive definition in the train set, we provide one compositional sentence featuring 'jump'. We vary the complexity of this sentence as shown in Table 3. Similar to the case of providing only the primitive definition, we observe that models are unable to



(a) No extra primitives (b) 10 extra primitives (c) 20 extra primitives

Figure 9: Visualizing the t-SNE reduced embeddings of isolated primitive (▲), example primitives (■) and non-primitives (●) from a learned LSTM model as we increase the number of example primitives in the Colors train set.

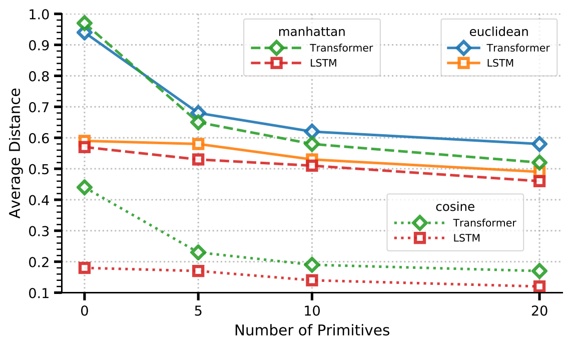
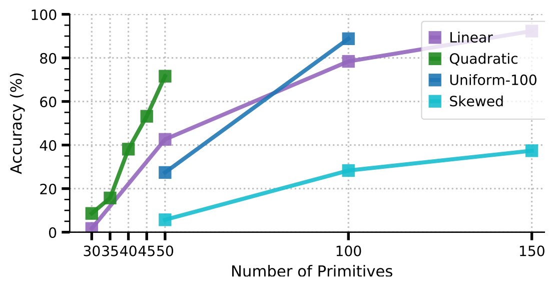
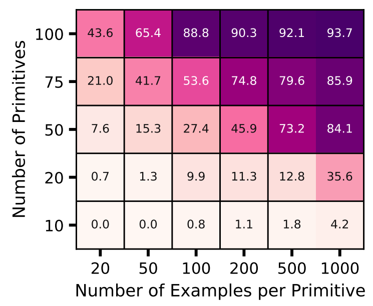


Figure 10: Measuring the similarity of the embedding of *isolated primitive* with the embeddings of example primitives for learned Transformer and LSTM models as we increase the number of example primitives in the Colors train set.



(a) Other Distributions



(b) Uniform Distribution

Figure 11: Measuring the generalization performance of LSTM on different types of train set distributions of the SCAN dataset.

generalize and achieve near-zero accuracies.

We now wish to see whether the presence of more number of primitives and their sentences in the train set helps a model generalize in this scenario (like it did for primitive definitions as shown in Section 2). We consider the setup of having 100 primitives and their sentences in the train set (Section 2) apart from the one compositional sentence with the word 'jump'. We find that models are able to achieve near-perfect generalization accuracies.

This shows that our idea holds more generally: Adding more primitives and their sentences helps a model effectively learn the meaning of a new primitive, whether specified explicitly via a primitive definition or implicitly in a sentence.

E Details of Experimental Setups and Other Results

E.1 Embedding of Isolated Primitive

We scale the embedding vectors to unit $L2$-norm for calculating the euclidean distance and unit $L1$-norm for calculating the manhattan distance. For Colors dataset as well, we compare the average distance with other primitives before and after adding primitives to the training data. We again find that as we increase the number of example primitives in the training set, the embedding of the isolated primitive ('zup') gets closer to the example primitives (refer to Fig. 10) in terms of Euclidean, Manhattan and Cosine Distances.

We additionally show the t-SNE plots of the learned embeddings for the LSTM model on the Colors dataset (Fig. 9).

E.2 Impact of Training Distributions

In Section 3.1, we showed results of the Transformer model on various train set distributions of the SCAN dataset. We also experimented with the LSTM model, the results of which can be found in Fig. 11. We see the same trend as we saw for Transformers.

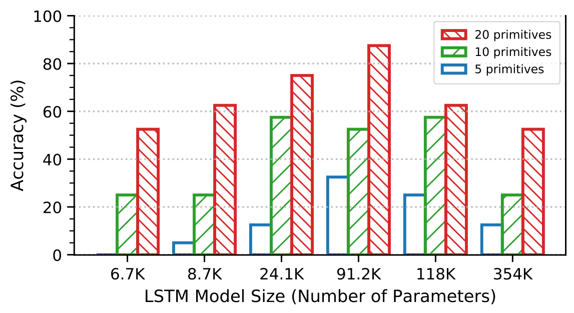


Figure 12: Measuring the generalization performance of an LSTM of varying capacity across increasing number of primitives in the Colors train set.

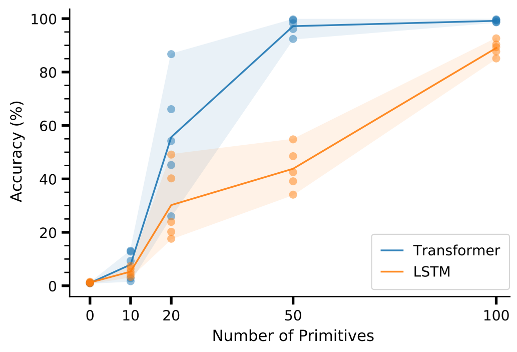


Figure 13: Generalization performance on SCAN across different runs with random seeds.

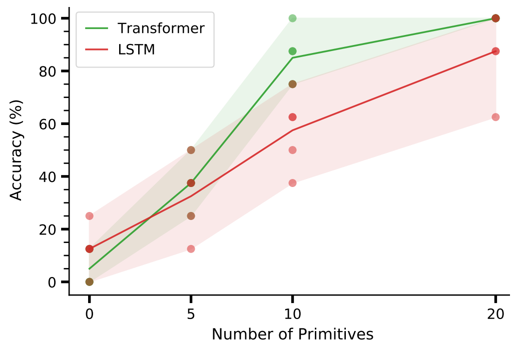


Figure 14: Generalization performance on Colors across different runs with random seeds.

E.3 Impact of Model Capacity

In Section 3.2, we showed results of varying sizes of Transformers trained on datasets with different number of example primitives. We also experimented with the LSTM model, the results of which on the Colors dataset can be found in Fig. 12. We see the same trend as we saw for Transformers.

E.4 Variance Across Different Runs

We plot the generalization accuracies of the Transformer and LSTM models on SCAN and Colors datasets over 5 different runs with random seeds in Fig. 13-14. Both models displayed a high degree of variance in generalization performance on both datasets. It is interesting to see that the variance decreases with increasing number of primitives.

E.5 Evaluation on Multiple Isolated Primitives

Our results are valid not just when there is a single isolated primitive, but even when there are multiple isolated primitives that are used compositionally at test time. While we believe that this holds trivially due to the symmetry of the setup, for completeness, we provide empirical evidence. We consider the setting on SCAN in which the train set has a total of 100 example primitives uniformly distributed. To this train set, in addition to the primitive definition of 'jump' (i.e., 'jump $\rightarrow$ JUMP'), we add 9 other primitive definitions of newly introduced isolated primitives. Thus, while the size of the train set in this setting was 13185, the size of the new train set is 13194. We then extract templates from the original SCAN test set and exhaustively populate these templates with the 10 isolated primitives. Hence, while the size of the original test set was 7706, the size of the new test set is 77060.

We evaluated Transformers on this data. The best model achieved 94.5% accuracy on the complete test set, thereby showing that our methodology and results are valid even when there are multiple isolated primitives in the dataset at the same time.

F A Note on Other Data Augmentation Methods

Applying data augmentation methods such as GECA (Andreas, 2020) on SCAN will lead to addition of training examples in which the input sentences are compositions of the isolated primitive 'jump'. This breaks the systematicity of the setup. While such automatic data augmentation approaches are important resources for enabling compositional generalization, a model that performs well on this modified split cannot be considered to be able to generalize compositionally.

Shi et al. (2021) proposed a data augmentation method based on the theory of meaningful learning. Similar to our work, they also augment the train set by adding more primitives (e.g. 'jump_0', 'jump_1', ..., 'jump_n'). However, compared to our work, their setup is completely different: The new primitives that they add to the train set are all still mapped to the output token of an example primitive ‘jump’, which is ‘JUMP’ (i.e. ‘jump_0 $\rightarrow$ JUMP’, ..., ‘jump_n $\rightarrow$ JUMP’). Their train set has examples showing compositions of ‘jump’ while their test set evaluates for novel compositions of the newly added primitives. We argue that their setup cannot be considered one-shot primitive generalization since now the model can see the output token ‘JUMP’ in composition with other words. We claim that this familiarity with the output token enables a model to generalize well on the test data even if the newly added primitives are only presented one-shot in the train set. Indeed, Lake and Baroni (2018) also suggested that the reason why models are able to do well on the ‘turn left’ split of SCAN is because the train set consists of many examples that have the output token ‘LTURN’ used compositionally.

To validate our claim, we propose a simple experiment. In the original SCAN ‘add jump’ split, we map ‘jump $\rightarrow$ WALK’ instead of ‘jump $\rightarrow$ JUMP’ for all examples (primitive definition as well as compositional sentences) in both the train and test sets. In this setup, even though the input word ‘jump’ is seen only once at train time, it’s mapping ‘WALK’ is used compositionally in many examples. On evaluating a Transformer model on this split, we found that it achieves a near-perfect accuracy. This shows that providing compositional examples with the output token of the isolated primitive not only breaks systematicity, but is the reason behind the high performance of models in that setting.