Data Factors for Better Compositional Generalization

Xiang Zhou **Yichen Jiang** **Mohit Bansal**
UNC Chapel Hill
{xzh, yichenj, mbansal}@cs.unc.edu

Abstract

Recent diagnostic datasets on compositional generalization, such as SCAN (Lake and Baroni, 2018) and COGS (Kim and Linzen, 2020), expose severe problems in models trained from scratch on these datasets. However, in contrast to this poor performance, state-of-the-art models trained on larger and more general datasets show better generalization ability. In this work, to reconcile this inconsistency, we conduct an empirical analysis by training Transformer models on a variety of training sets with different data factors, including dataset scale, pattern complexity, example difficulty, etc. First, we show that increased dataset complexity can lead to better generalization behavior on multiple different generalization challenges. To further understand this improvement, we show two axes of the benefit from more complex datasets: they provide more diverse examples so compositional understanding becomes more effective, and they also prevent ungeneralizable memorization of the examples due to reduced example repetition frequency. Finally, we explore how training examples of different difficulty levels influence generalization differently. On synthetic datasets, simple examples invoke stronger compositionality than hard examples do. On larger-scale real language datasets, while hard examples become more important potentially to ensure decent data coverage, a balanced mixture of simple and hard examples manages to induce the strongest generalizability.[1]

1 Introduction

Many recent diagnostic datasets, e.g., SCAN (Lake and Baroni, 2018), COGS (Kim and Linzen, 2020), etc., have exposed the compositional generalization problem of neural sequence-to-sequence (seq2seq) models. They show that seq2seq models that are trained from scratch fail miserably when tested on examples containing novel combinations of seen elements. However, in contrast to these results, models trained (or pretrained) on larger datasets, (including T5 (Raffel et al., 2020), PaLM (Chowdhery et al., 2022), etc.) show substantially better performance (Furrer et al., 2020; Drozdov et al., 2023) for compositional generalization. While some factors behind many of these improvements are the emergent abilities from scaling up (Wei et al., 2022), we explore an alternative and complementary angle: *how and why does training on* ***more complex*** *datasets help a general Transformer model, with limited size and capacity, generalize compositionally to unseen natural language queries?*

To answer this question, we empirically study how changes in data factors (e.g., scale, diversity, difficulty, etc.) influence the generalization ability of models trained from scratch. Our first analysis is inspired by Patel et al. (2022) and Jiang et al. (2022), who make the model generalize significantly better on SCAN simply by increasing the number of unique primitives. We extend this observation in several directions and further explore where this gain comes from. Our experiments show that this is not a special observation on SCAN for the *Jump* and *Around Right* split. Instead, the same effect can be observed on multiple different types of generalization challenges (e.g., primitive-level generalization, length-level extrapolation, etc.), and on both synthetic and real-language datasets. To summarize our first direction of analysis, there is a strong connection between *increased dataset complexity* and *better compositional generalization*.

Second, we try to understand *why* more complex datasets lead to better generalization. We build up our analysis by comparing two potentially competing behaviors of the models: *surface memorization* (i.e., memorizing the direct mapping from inputs to outputs without understanding the underlying composition), and *compositional understanding*

[1]The code and data for this work are available at https://github.com/owenzx/data4comp.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 14549–14566
December 6-10, 2023 ©2023 Association for Computational Linguistics

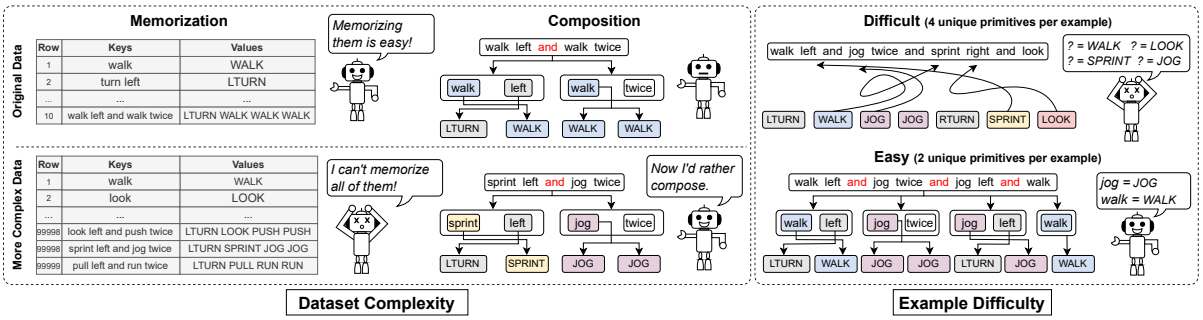

Figure 1: Model's compositional generalization ability is affected by the complexity of the dataset (left) and the difficulty of training examples (right).

(i.e., first discovering a set of compositional rules from examples and then applying the rules for predictions). Please also refer to the left part in Figure 1 for an illustration. While both behaviors can lead to near-perfect training-set performance, only the latter guarantees good generalization performance. We argue that more complex datasets improve compositional generalization as they provide substantial obstacles to surface memorization. Specifically, we make two hypotheses: (1) **diversity**: more diverse patterns (e.g., more unique primitives) exemplified in complex datasets increase the difficulty of surface memorization; (2) **frequency**: the larger dataset size causes a decrease in the frequency of seeing similar examples, thus preventing their memorization. To empirically confirm these hypotheses, we provide detailed ablations showing how both factors contribute to empirical gains in data augmentation. To further isolate the effect of frequently recurring examples, we show that deliberately encouraging example repetition even in a large dataset brings substantial detriment to the generalization performance. Furthermore, as a corollary to our hypotheses, we provide a simple yet effective data augmentation method AugZero that satisfies both contributing factors without utilizing any dataset-specific knowledge.

Finally, we provide a more fine-grained analysis to study how examples of different difficulties inside a dataset affect generalization. Our analysis covers both smaller, synthetic datasets and larger-scale, real-language datasets. On synthetic datasets (e.g., SCAN), we create multiple versions of the datasets strictly controlling the difficulty of the examples in each version, and show that simpler examples facilitate compositional generalization more than difficult examples. On sub-sampling experiments in real-language datasets (e.g., ATIS, SMCalFlow), we observe similar but more complicated trends. Potentially due to the need to cover all the diverse natural language phenomena, simple examples alone are not enough for achieving good performance, however mixing difficult examples with simple examples is still beneficial on the compositional dataset SMCalFlow-CS.

In conclusion, we present an empirical study on how data factors influence compositional generalization behavior. We notice: (1) increased dataset complexity can improve the model's generalization ability; (2) increased complexity creates obstacles for surface memorization by having a higher cost of memorization and less recurring frequency of examples; (3) example difficulty can influence generalization substantially with simpler examples benefiting compositional generalization more.

2 Tasks and Setup

2.1 Datasets

In this work, we focus on semantic parsing datasets. We use both datasets with synthetic examples and datasets with natural examples. We experiment with synthetic datasets to have fine-grained control and experiment with flexible natural language datasets to demonstrate the generalizability of our findings. For **synthetic datasets**, we use both the original version of SCAN (Lake and Baroni, 2018) as well as an expanded version with increased complexity, denoted as SCAN* in this work. We introduce SCAN* to control the overall complexity of the SCAN-like dataset.[2] Compared to SCAN, we make two major changes: (1) We remove the constraint that at most one conjunction (*and* or *after*) is allowed for every example. Instead, in SCAN*, every sentence can have an unlimited number of conjunctions. To avoid adding ambiguity brought by multiple conjunctions, we assign higher operation priority to *after* than *and*. (2) We use a larger

[2]The original grammar in SCAN does not allow examples with more complexity.

	Jump	*Around Right*	GeoQuery (query)	GeoQuery (question)
baseline	3.49±1.65	19.86±10.41	42.37±3.26	64.52±1.44
2x Augmentation	77.37±17.74	73.02±20.12	45.92±3.17	69.10±0.85
20x Augmentation	99.68±0.32	99.38±0.68	47.85±3.89	68.17±2.13
200x Augmentation	99.93±0.04	99.01±0.95	45.70±1.66	65.45±2.43

Table 1: Datasets with increased complexity via data augmentation lead to better compositional generalization. We use logic form outputs for GeoQuery in this table. For SQL results with similar trends, see Table 7 in the Appendix.

set of verb-type primitives (i.e., *run*, *walk*, etc.). The first change allows us to increase the number of possible example structures and lengths, and the second allows the increased lexicon-level complexity. For **natural language datasets**, we use three English datasets with human-written queries: GeoQuery (Zelle and Mooney, 1996), ATIS (Price, 1990; Dahl et al., 1994) and SMCalFlow (Andreas et al., 2020; Yin et al., 2021). For GeoQuery, we use both the *query* split and the *question* split following Andreas (2020). And for SMCalFlow, we include the 32-shot version in Yin et al. (2021) as a compositional split and denote it as SMCalFlow-CS. For fair comparison across splits, we preprocess SMCalFlow and SMCalFlow-CS in the same way. More details are in Appendix A.

2.2 Implementation Details

We focus on seq2seq Transformers as they are the most prevalent choices for most semantic parsing datasets. For our main experiments, we train the Transformer from scratch so that our analysis is not influenced by the existing knowledge from pre-training. Our model configuration mainly follows Csordás et al. (2021) to use a 3-layer Transformer which works well in most compositional tasks. We provide additional results on models with other configurations in Appendix H. We decode using a beam size of 5 and select the best model based on their dev-set exact-match accuracy. For any experiments in the same table or figure, we train every model using the same amount of total steps even though the training dataset size may vary. Unless otherwise mentioned, all the results are the mean of 5 runs. More implementation details are in Appendix B.

3 Increased Training Set Complexity Leads to Better Generalization.

On the SCAN (Lake and Baroni, 2018) dataset, models trained from scratch show very poor performance, especially on the *Jump* split. However, recent works (Patel et al., 2022; Jiang et al., 2022) use data augmentation methods to significantly improve the performances. Notably, the main effect of these methods is to increase the number of primitives. They neither provide any structure-level change nor create any overlapping between the training and the testing set. Therefore, the improvement seems to be solely from a *more complex* training set. In this section, we replicate and extend their findings, showing this phenomenon is in fact consistent on multiple generalization challenges.

3.1 Experiment Design

We first follow the same setup as Jiang et al. (2022) to conduct data augmentation experiments on the SCAN *Jump* and *Around Right* split, creating augmented training sets with more primitives. We also extend this process to GeoQuery dataset (Zelle and Mooney, 1996) to investigate the effect on datasets with more natural and realistic language. Please refer to Appendix C for details of our replication.

Additionally, we include the SCAN* *Length* split to investigate the effect of data complexity on a different type of generalization: length extrapolation. To allow more flexible control of length, we use the expanded SCAN* dataset described in Sec. 2.1. Specifically, we train the model on training sets containing examples with length $0 < l \leq L$, and challenge it on test examples with length $L < l \leq 2L$. To control the complexity, we create four splits with different values of $L \in \{31, 62, 125, 250\}$. For all four splits, we make sure they have similar dataset statistics whenever possible. See Appendix A.2 for more details.

3.2 Results

In Table 1, we show the generalization performances when trained with different complexities (i.e., different numbers of total primitives). On both SCAN and GeoQuery, increasing the training-set complexity substantially improves the generalization performance. On SCAN, the models obtain huge improvements with 2x augmentation and reach a near-perfect accuracy with 20x or more primitives. Similarly, on GeoQuery, the augmented performance substantially outperforms the base-

	jump	*walk*	*look*	*run*
1x	-0.84, 0.57	0.02, 0.59	0.75, 0.15	0.20, -0.13
20x	0.41, 0.11	0.49, -0.01	0.45, 0.50	0.75, -0.10

Table 2: The projections of "jump, walk, run, look" onto the two most principal components of the embedding space, when trained on *Jump* 1x or *Jump* 20x.

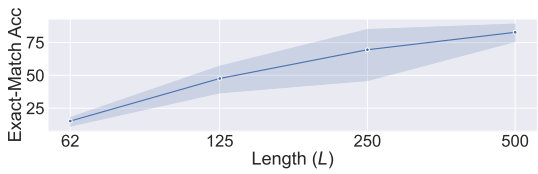

Figure 2: Different generalization performances on SCAN* *Length*. The challenge is always to train on examples with length $0 < l \leq L$ and test on examples with length $L < l \leq 2L$.

line, reaching around 5% accuracy improvement with the best augmentation.[3] Interestingly, there is a difference in the optimal augmentation between SCAN and GeoQuery. On SCAN, larger datasets always lead to better results until the model reaches near-perfect accuracy. However, on GeoQuery, the best augmentation is *not* the most complex 200x data, though a larger amount of augmentation still leads to improvements. We suspect this phenomenon is related to the frequency of primitives, and please see Sec. 4.3 for more discussion.

Besides primitive-level generalization, increased complexity is also beneficial for simple structure-level challenges such as length generalization. In Figure 2, we see steadily increasing performance when both the training set contains longer examples. For the same generalization challenge (i.e., generalizing to twice the length seen during training), the same model can reach over 80% accuracy when the training set contains sentences with 250-token length, but only reaching around 15% if the training sentences are all within a length of 62. These results demonstrate that across multiple different generalization challenges, more complex training sets seem to bring a consistent gain.

Analysis on learned primitive embeddings. To better understand how training on a more complex dataset (e.g., *Jump* 20x) changes the model's behavior, we conduct an analysis on the word embeddings of four primitives: "jump, walk, run, look". First, we perform a linear Principal Component Analysis on the entire embedding matrix and then project these four embeddings to the first two principal components. In Table 2, we show the projection of the embeddings trained on *Jump* 1x versus the embeddings trained on the larger *Jump* 20x. We observe that the four primitives trained on *Jump* 20x are closer together in the first principal component, ranging from 0.41 to 0.75 (0.53 ± 0.13), compared to -0.84 to 0.75 for the baseline (0.13 ± 0.57). This suggests that compositionality arises because the model can better represent the syntactic similarity between the rare primitive "jump" and other common primitives, which appear in very different contexts during most of the training.

4 Understanding the Advantages Brought by Increased Complexity

We further explore *how* increased complexity improve generalization. We will first provide a discussion on dataset complexity and model behaviors. Then, we will state our hypotheses on how data complexity impacts generalization by influencing the model behaviors. Finally, we provide supporting experiments.

4.1 Preliminaries

Two types of data complexity. For simplicity, we provide an informal discussion of "data complexity" for semantic parsing tasks. Data complexity can be measured from many different angles. In this study, we only consider the *training set* and mainly focus on two types of data complexity: (1) **Pattern complexity**: a training set with larger pattern complexity will contain more unique patterns (e.g., more unique primitives, more diverse example length, etc.) in the examples. (2) **Scale complexity**: a training set with larger scale complexity will contain more different examples. Note that these two properties are usually positively correlated in most datasets as a larger training set usually contains more diverse patterns. In this section, we aim to explain how increments in these complexities influence the model's generalization.

Two competitive model behaviors. One way to understand the different generalization behaviors is to focus on how models achieve good training performance. We argue the difference in how models achieve good training performances heavily influences the generalization performance. Intuitively, for a seq2seq task, there are two extremes of behaviors that models could adopt: (1) **surface mem-**

[3]Note that unlike SCAN, primitive-level generalization is not the only challenge in GeoQuery, so the scale of the improvement is noticeably smaller. Nonetheless, the improvements shown in Table 1 are consistent across different splits, output formats, and augmentation sizes.

orization: the model only establishes one-to-one maps from the inputs and the outputs and does not correctly infer the composition;[4] (2) **compositional understanding**: the model makes predictions based on a correct understanding of how the semantics are composed by smaller sub-structures.[5] An illustration of these two behaviors is shown in Figure 6 in the Appendix. In practice, the model's behavior is not a binary choice between these two extremes, but oftentimes a complicated combination depending on the inputs. Nonetheless, here we use these two concepts to denote two trends of behaviors, and we will show that increased data complexity biases the model toward the second behavior, and hence leads to better compositional generalization.

4.2 Hypotheses

We elaborate on our hypotheses of the connection between data complexity and model behaviors. Note that our two hypotheses are interrelated and not contradictory to each other. They are just two different perspectives on how this connection.

(1) Pattern complexity (i.e., more diverse examples) increases the difficulty of surface memorization. For surface memorization, the models need to memorize all the individual mappings from each example input to each example output. The difficulty of achieving a specific training loss through surface memorization is proportional to the total sum of the complexity of *every* unique example in the dataset. With a larger training set containing more diverse examples, the difficulty of surface memorization increases substantially. However, with correct compositional understanding, the difficulty will remain roughly the same as the previously-learned composition remains correct. Hence, with more complex datasets containing more different examples, the difficulty of surface memorization increases much faster than the difficulty of compositional understanding, leading to a preference for the latter.

Dataset	SCAN (*Around Right*)	GeoQuery (query)
Origin Dataset	$19.86_{\pm 10.41}$	$42.37_{\pm 3.26}$
2x Augmentation	$73.02_{\pm 20.12}$	$45.92_{\pm 3.17}$
+ More primitives	$96.09_{\pm 4.33}$	$43.55_{\pm 1.82}$
+ More prim. & larger size	$99.38_{\pm 0.68}$	$47.85_{\pm 3.89}$
+ AugZero	$95.50_{\pm 7.48}$	$43.55_{\pm 1.01}$

Table 3: Both dataset size and the number of primitives contribute to the performance improvement.

(2) Scale complexity (i.e., larger datasets) avoids memorization by reducing frequently recurring examples. Another effect usually brought by increased dataset complexity is more training examples. This naturally leads to a decreased frequency with which similar examples occur during the training. As is shown by Hernandez et al. (2022), even a small portion of recurring data can bring substantial damage to the copying performance and the scaling law of a language model. We argue that similar to this finding, reduced example recurring frequency will make surface memorization harder and thus encourage compositional understanding.

4.3 Supporting Experiments

In this section, we provide empirical evidence supporting our previous two hypotheses.

Two sources of empirical improvements. First, we try to separate the benefits of data augmentation shown in Sec 3 into two parts: benefits from increasing pattern complexity and from increasing scale complexity. In Table 3, we present models trained with data of three different level of complexities. The "+ More prim. & larger size" row is the model trained with the full x20 data augmentation with more primitives *and* a larger dataset size. For the "+ More primitives" row, we down-sample the augmented part of the x20 data[6] so that the total size is the same as the x2 augmented dataset.[7] On both SCAN and GeoQuery, by comparing 'Origin Dataset' and '+ More primitives' rows, we find that increasing pattern complexity can encourage stronger compositional generalization. Additionally, by comparing the '2x Augmentation' and the '+ More primitives' row, we notice that the trade-off of having more primitives but fewer examples

[4] This may remind readers of classic *overfitting* behaviors. Indeed, traditional loss regularization methods (Li et al., 2019; Yin et al., 2023) can improve compositional generalization to an extent. However, finding the best regularization method is not trivial, and we also need to understand methods (e.g., LLMs) free of explicit regularization. Our work aims to provide related insights from the dataset angle.

[5] Depending on the actual dataset properties, compositional understanding may not always be the correct behavior (e.g., in translation as shown in Dankers et al. (2022)). However, for the scope of this study, we focus on how more complex datasets encourage compositional generalization, so we assume compositional understanding as the ideal behavior.

[6] We do not down-sample the whole dataset as it will create an unwanted side-effect of removing a large portion of the original data compared to the x2 data.

[7] Note that there is no ablation *only* with larger sizes, since if we keep the original examples unchanged, increasing dataset size will inevitably require new primitives.

Data Size	Repetition	GeoQuery (query)	GeoQuery (question)
Original	/	42.37$\pm$3.26	64.52$\pm$1.44
Original	Example	24.84$\pm$3.64	55.48$\pm$4.51
Original	Primitive	1.65$\pm$1.69	43.08$\pm$0.93
20x	/	47.85$\pm$3.89	68.17$\pm$2.13
20x	Example	44.95$\pm$3.20	68.46$\pm$1.13
20x	Primitive	31.40$\pm$4.82	49.11$\pm$0.84

Table 4: The performance of models trained with different types of example repetition curriculum.

for each primitive is not always beneficial, leading to improvement on SCAN, but not on GeoQuery. Finally, by comparing '+ More primitives' and '+ More prim. & larger size', we can further conclude that increasing scale complexity given the same amount of distinct primitives can provide further gain in generalization results.

Performance detriment from example repetition. To further support our second hypothesis, we present experiments to show that highly-recurring examples can cause a significant performance decrease. We compare three training curricula with the same total steps in Table 4. The first curriculum with no repetition is a normal curriculum where every epoch contains the entire training set. The other two curricula are specifically designed to aid surface memorization by having more recurring examples. For the Example Repetition curriculum, at the first 20% of the training steps, the model is only trained on a small subset containing 20% of the examples, then the remaining data are gradually put back into the training set until the training set becomes the full dataset at 80% of the total steps. This curriculum ensures that the model is repeatedly trained on a small set of examples for a long period. The Primitive Repetition curriculum is similar to the Example Repetition curriculum except that it first clusters the training examples by the primitives and then starts to train the model with data containing only 20% of the primitives. In Table 4, we confirm that example repetition can bring substantial damage to the generalization performance. On smaller original datasets, repetition at both the example level and the primitive level substantially hurt the performance. On GeoQuery, Example Repetition leads to a 10 to 20 accuracy drop, and the damage brought by Primitive Repetition is even larger. With a larger dataset resulting from data augmentation, the trend is slightly different. Repetition at the example level causes a much smaller drop, possibly because that 20% of the augmented dataset is still relatively large so there is not much repetition. However, primitive-level repetition still causes substantial damage, showing over 15 points drop. To summarize, earlier in this paper, we show that increasing the dataset size generally helps generalization. However, Table 4 show that naturally increasing the dataset size while still explicitly adding highly recurring examples actually leads to worse results. Combining these two observations, we provide evidence supporting our hypothesis on the importance of reducing frequently recurring examples.

Simple Data Augmentation with *Zero* Prior Knowledge Finally, we provide a data augmentation method as a direct corollary of our hypotheses. We call our method *Augmentation with Zero Prior Knowledge* (AugZero) as it brings zero additional knowledge but can still be effective as it satisfies our two previous two hypotheses. The main idea is to simply copy the *entire* vocabulary and use the newly copied vocabulary to re-tokenize the entire dataset. Figure 3 shows an illustration for AugZero with k times augmentation. A more detailed description is in Appendix C.3. The results with $k = 200$ are in Table 3. Despite how simple the algorithm is, AugZero can still substantially improve the performance, reaching close to perfect performance on the SCAN *Around Right* split and achieving decent improvement on GeoQuery. While the gain is smaller than the primitive-aware augmentation result (+ More prim. & larger size row), its success further supports our hypotheses.

5 Which Examples Benefit Generalization the Most: A Difficulty Perspective

Previously, we demonstrate and explain that increasing the complexity of the whole dataset can improve compositional generalization. In practice, examples *inside* a dataset have different properties (e.g., difficulty, topic, etc.) and thus may influence generalization differently. In this section, we present a study from the difficulty perspective. We will start with a discussion on the definition of difficulty, and then show how example-level difficulty can substantially influence compositional generalization on both synthetic and real datasets. For this section, we report results with three runs as the dataset sizes are substantially larger.

5.1 Example-Level Difficulty Metrics

Complexity-based difficulty. For synthetic datasets like SCAN, we can intuitively define the difficulty of an example by the complexity of the

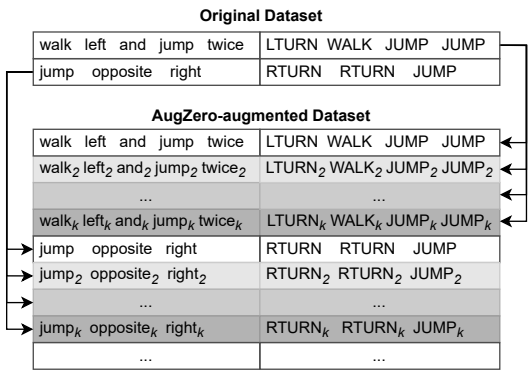

Figure 3: The AugZero data augmentation process.

example itself, for example (1) the length of the input instruction, or (2) the number of unique primitives, both reflecting the pattern complexity of the example. These metrics are the most reliable as they directly reflect the complexity of correctly generating the target output examples. However, these metrics can be hard to measure on real natural language datasets.

Prototype-based difficulty. Another limitation of the complex-based difficulty is that it treats each example independently, while in practice the learning is conducted on an entire dataset instead of individual examples. One metric addressing this issue is to see how prototypical (i.e., if there are other similar examples in the dataset) one example is. To measure this, we follow the process in Sorscher et al. (2022). We use SimCSE (Gao et al., 2021) to encode all the input of the examples, then cluster the encoded vectors using K-means, and use the L2 distance to the centroid as the difficulty measure. An outlier that is far from any cluster is deemed difficult. See Appendix G.1 for clustering examples for this method. In our experiments, we will use this difficulty as the metric when the ground truth complexity metric is not available.[8]

5.2 Experiments

We show how models behave differently with different distributions of example difficulties.

Simpler examples make generalization easier on SCAN*. First, as we have full control over SCAN*, we directly use example complexity (including both length-based and primitive-based complexity) as the difficulty metric. For evaluation, we use the *Jump* split as its difficulty is substantially influenced by both types of complexity. For length-based complexity, we generate training sets with different maximum lengths, the same as described in Sec. 3. For primitive-based complexity, we keep the number of total possible primitives in the vocabulary fixed but generate multiple different training sets only varying the maximum number of unique primitives *per example*. Intuitively, it will be easier to infer the correct composition from shorter examples or examples containing fewer primitives. The results are shown in Fig. 5. All the models are tested on the same testing set similar to the original SCAN *Jump* testing set. For both settings, we can see a steadily decreasing trend when the examples become harder. When the maximum length is reduced from 500 to 62, the performance is increased from 16.65% to 47.31%.[9] With only 2 unique primitives per example, the performance is also increased to 49.14%. These results demonstrate that easier examples can make the correct composition easier to learn.

Datasets	Simple	Hard	Mix
ATIS	$40.86_{\pm 0.43}$	$52.26_{\pm 1.05}$	$48.11_{\pm 0.94}$
SMCalFlow	$37.35_{\pm 0.26}$	$46.38_{\pm 0.55}$	$43.69_{\pm 0.35}$
SMCalFlow-CS	$36.72_{\pm 0.70}$	$38.43_{\pm 1.03}$	$39.41_{\pm 1.11}$

Table 5: Different performance when trained subsets with different difficulty. Simple and Hard denotes the simplest and hardest subset as in Figure 4. Mix is a mixture of both subsets.

Mix of simple and hard examples needed on real language datasets. We next examine the impact of example difficulty on more complex larger-scale natural language datasets. Due to the flexible and diverse nature of natural language in real datasets, models now not only need to understand the correct composition but also need to capture other language uses through potentially non-compositional ways. Therefore, the trends in natural language datasets can be different from the previous observation. For this study, we conduct experiments on ATIS (Price, 1990; Dahl et al., 1994), SMCalFlow (Andreas et al., 2020) and the compositional version SMCalFlow-CS (Yin et al., 2021). For all three datasets, we train models on multiple subsets with different difficulties but all con-

[8] Another way to measure difficulty is to directly use the model's performance (e.g., accuracy). Using these metrics, models will perform very poorly when trained on the hardest subset as they fail to learn from most examples, making the analysis results less interesting. See Appendix G.2 for more discussion on these metrics and detailed results.

[9] Note that the results here and the results in Fig. 2 are not contradictory. Here we show more examples with longer lengths make primitive-level generalization worse, while in Fig. 2 we show such data make length generalization better.

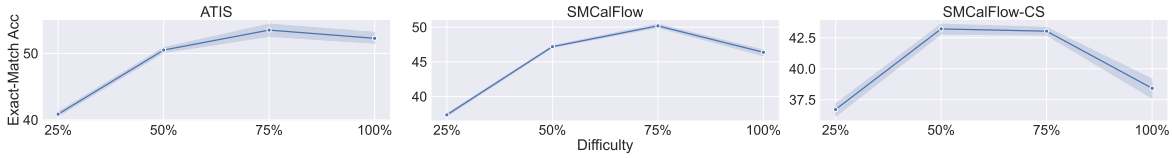

Figure 4: Results on ATIS and SMCalFlow with training sets of different difficulties. The X-axis represents the quantiles of example difficulty, the smaller the easier.

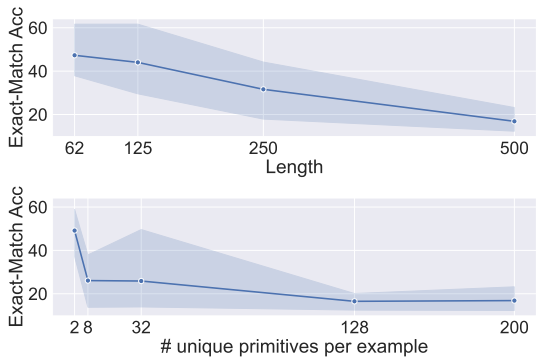

Figure 5: On SCAN* *Jump* split, simpler examples facilitate better compositional generalization.

tain 25% of the training examples. Since on these datasets, we no longer have access to the ground truth example complexity, so we present results with prototype-based difficulty in Figure 4.

We observe similar but more complicated trends on larger-scale natural language datasets. First, we observe that *difficult examples become important on these datasets*. On all three datasets, using only the easiest examples leads to the worst results. We suspect that only using the simplest examples does not provide enough coverage for all the diverse linguistic phenomena in the dataset. However, the best performance is also not achieved with the most difficult examples, but at the medium level. Additionally, we notice that *even in larger-scale natural language datasets, simpler examples are still important for compositional generalization.* Here we specifically focus on the trend difference between the results on SMCalFlow-CS (a compositional split), and the other two non-compositional-split datasets. In Figure 4, we see SMCalFlow-CS achieves the best performance with the second easiest split, different from the other two datasets. Additionally, in Table 5, we show the performance by mixing the hardest examples with the easiest examples with a 50%/50% ratio, and we also observe a unique trend on SMCalFlow-CS. Only on SMCalFlow-CS, the mixed performance outperforms the performance by only using the difficult data, showing the advantage of having simpler examples for compositional challenges still persists.

6 Related Work

Compositional generalization. Earlier related works in compositional generalization study the systematic behavior of neural networks in language learning (Wong and Wang, 2007; Brakel and Frank, 2009), compositional counting (Wiles, 1998; Weiss et al., 2018), syntax learning (Linzen et al., 2016), etc. Recently, many recent datasets (Lake and Baroni, 2018; Kim and Linzen, 2020; Loula et al., 2018; Bastings et al., 2018; Keysers et al., 2020; Tsarkov et al., 2021; Hupkes et al., 2020) substantially facilitates the development and evaluation of related method improvements. Since then, many different methods have been proposed, including architecture improvements (Dessì and Baroni, 2019; Gordon et al., 2020; Oren et al., 2020; Zheng and Lapata, 2021), grammar-based approaches (Shaw et al., 2021; Kim, 2021), task decomposition (Herzig et al., 2021), data augmentation (Andreas, 2020; Akyürek et al., 2021; Akyurek and Andreas, 2023), and novel learning methods (Jiang and Bansal, 2021; Lake, 2019; Conklin et al., 2021; Jiang et al., 2022). Of all these methods, the biggest breakthrough (Drozdov et al., 2023; Qiu et al., 2022) still mainly comes from using large language models (Chowdhery et al., 2022; Raffel et al., 2020). Our work aims to provide insight into how (pre-) training on a large corpus is beneficial for compositional generalization. Besides data factors, the generalization behavior of models can also be studied from many different angles, including architecture (Li et al., 2019), regularization (Yin et al., 2023), representation geometry (Montero et al., 2021; Ito et al., 2022; Murty et al., 2023), etc. All these directions are complementary towards the same goal of understanding the mechanism of generalization.

Dataset influence on generalization. Prior to the development of LLMs, most dataset quality-related research focuses on dataset artifacts (Gururangan et al., 2018), or label quality (Maynez et al., 2020;

Pavlick and Kwiatkowski, 2019). After the recent success of LLMs, a number of works pointed out that large-scale datasets can be a major contributing factor to the success. Chan et al. (2022) shows that the scale of datasets is as important as the scale of the model. Hoffmann et al. (2022) show how example distribution can significantly impact the emergence of in-context few-shot learning. Hernandez et al. (2022) notice that even a small portion of repeated data can have a significant impact on the generalization performance.

7 Discussion

Advantage of large-scale pretraining. In this work, we have demonstrated how different data factors (e.g., scale, diversity, example difficulty, etc.) can substantially improve the model's ability to generalize compositionally. While our experiments are done on a smaller scale, these results also hint at why models pretrained on larger datasets show better generalization ability. We argue that doing large-scale pretraining implicitly satisfies many beneficial data factors studied in this work. During pretraining, models are exposed to a large set of different structures and primitives. Additionally, pretraining datasets have a large size that prevents frequent repetition and contains a diverse set of simple and difficult examples. All these conditions incentives the emergence of compositional generalization ability.

How about fine-tuning models? In our main experiments, we choose to not use pretrained models as they already possess a substantial amount of knowledge, which makes it very difficult to evaluate how models initially acquire compositionality. Nonetheless, as fine-tuning models are of great practical importance, here we discuss related issues about fine-tuning models and summarize our preliminary findings with T5 models (more details are in Appendix H.2). The behaviors of fine-tuning models are slightly different. We notice that the size and the diversity of the dataset can still be important as repetitive training on similar examples (e.g., with the same primitive) will cause substantial overfitting and hurt the pretrained model. However, as pretrained models already have a good understanding of basic language compositionality, they may need less assistance from the data, so maintaining a large number of different primitives or simple examples may be less important. Overall, our results suggest that when fine-tuning on small datasets, data augmentation methods increasing the size and diversity may help the model's compositional generalization performance.

8 Conclusion

We empirically study how dataset factors influence compositional generalization. We show that increased dataset complexity facilitates compositional generalization, with dataset size and pattern complexity as two important factors behind the gain. We also provide an analysis of example difficulty and discuss the implication of our work on pertaining and fine-tuning models.

Limitations

The analyses in this study focused on relatively small-scale Transformer seq2seq models. While such architectural choice has been shown to be effective on compositional generalization datasets when trained from scratch (Csordás et al., 2021), very large models may demonstrate different behaviors due to their *emergence abilities* (Wei et al., 2022). The data factor analysis is not meant to be a complete investigation about *all* possible data factors. In this paper, we focus on scale, difficulty, and data complexity, which are all shown to be important factors for compositional generalization. However, there are other factors also being important for compositional generalization. The specific evaluation setup of compositional generalization also varies in different works. In this work, we focus on synthetic diagnostic datasets ensuring sufficient supervision and larger-scale natural language datasets with minimal manual control of the data distribution. Additionally, compositional generalization challenges include both lexicon-level generalization and structure-level generalization. This work mainly focuses on lexicon-level generalization and relatively simple structure-level generalization as in our length-related and SMCalFlow-CS experiments. We assume these results will also generalize to more complicated structure-level generalization problems. For example, some more challenging setups may involve splits maximizing the compound divergence (MCD) (Keysers et al., 2020). MCD complexity does not naturally change when collecting larger datasets. To test a related hypothesis, one may check if increasing the number of unique local structures improves performance. However, such dataset modification is non-trivial and we leave the exploration as future work.

Acknowledgements

We thank Dipanjan Das, Colin Raffel, and the reviewers for their helpful comments. This work was supported by ONR Grant N00014-18-1-2871, NSF-CAREER Award 1846185, and DARPA MCS Grant N66001-19-2-4031, and an Apple PhD Fellowship. The views are those of the authors and not of the funding agency.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Ekin Akyurek and Jacob Andreas. 2023. LexSym: Compositionality as lexical symmetry. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 639–657, Toronto, Canada. 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.

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: SCAN both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55, Brussels, Belgium. Association for Computational Linguistics.

Philémon Brakel and Stefan Frank. 2009. Strong systematicity in sentence processing by simple recurrent networks. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 31.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, Jay McClelland, and Felix Hill. 2022. Data distributional properties drive emergent in-context learning in transformers.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *ArXiv preprint*, abs/2204.02311.

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.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022. The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland. Association for Computational Linguistics.

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.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings*

of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 351–360, Melbourne, Australia. 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. *ArXiv preprint*, abs/2007.08970.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. Permutation equivariant models for compositional generalization in language. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, et al. 2022. Scaling laws and interpretability of learning from repeated data. *ArXiv preprint*, abs/2205.10487.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *ArXiv preprint*, abs/2104.07478.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

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.

Takuya Ito, Tim Klinger, Doug Schultz, John Murray, Michael Cole, and Mattia Rigotti. 2022. Compositional generalization through abstract representations in human and artificial neural networks. *Advances in Neural Information Processing Systems*, 35:32225–32239.

Yichen Jiang and Mohit Bansal. 2021. Inducing transformer's compositional generalization ability via auxiliary sequence prediction tasks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6253–6265, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yichen Jiang, Xiang Zhou, and Mohit Bansal. 2022. Mutual exclusivity training and primitive augmentation to induce compositionality. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11778–11793, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Yoon Kim. 2021. Sequence-to-sequence learning with latent neural grammars. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 26302–26317.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Brenden M. Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9788–9798.

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

Stockholm, Sweden, July 10-15, 2018, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

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.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Milton Llera Montero, Casimir J. H. Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. 2021. The role of disentanglement in generalisation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. 2023. Characterizing intrinsic compositionality in transformers with tree projections. In *The Eleventh International Conference on Learning Representations*.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, Online. Association for Computational Linguistics.

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

Ellie Pavlick and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7:677–694.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

P. J. Price. 1990. Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Tianze Shi, Jonathan Herzig, Emily Pitler, Fei Sha, and Kristina Toutanova. 2022. Evaluating the impact of model scale for compositional generalization in semantic parsing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9157–9179, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? 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 922–938, Online. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Dmitry Tsarkov, Tibor Tihon, Nathan Scales, Nikola Momchev, Danila Sinopalnikov, and Nathanael Schärli. 2021. *-cfq: Analyzing the scalability of machine learning on a compositional task. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9949–9957. AAAI Press.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research.*

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Paul Rodriguez Janet Wiles. 1998. Recurrent neural networks can learn to implement symbolsensitive counting. *Advances in Neural Information Processing Systems*, 10:87.

Francis CK Wong and William SY Wang. 2007. Generalisation towards combinatorial productivity in language acquisition by simple recurrent networks. In *2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 139–144. IEEE.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

Yongjing Yin, Jiali Zeng, Yafu Li, Fandong Meng, Jie Zhou, and Yue Zhang. 2023. Consistency regularization training for compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1294–1308, Toronto, Canada. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Hao Zheng and Mirella Lapata. 2021. Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1022–1032, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Appendix

A Dataset Details

A.1 Natural Dataset Details

We use three English datasets with human-written queries: GeoQuery (Zelle and Mooney, 1996), ATIS (Price, 1990; Dahl et al., 1994) and SMCalFlow (Andreas et al., 2020; Yin et al., 2021).

GeoQuery (Zelle and Mooney, 1996) dataset contains 880 questions about US geography with the corresponding SQL queries or logical expressions of the question. Following Andreas (2020), we report performance on both the *query* split and the *question* split. The *question* split is the original split proposed in the dataset, while the *query* is a more compositionally challenging split proposed in Finegan-Dollak et al. (2018) ensuring no overlapping logical forms between the splits. We also use the standard dev-test split for GeoQuery.

ATIS (Price, 1990; Dahl et al., 1994) is a larger-scale semantic parsing dataset with 3809 examples. Each example contains a flight-related query as well as the corresponding SQL query. Due to the nesting format difference between the train/dev set and the test set of ATIS, we simplify our experiment setting to report dev set results only for the ATIS dataset.

SMCalFlow (Andreas et al., 2020) is a large-scale dialog dataset. All the examples in the dataset are from natural conversations and they are annotated with an executable dataflow program. Yin et al. (2021) proposed a modified version of this dataset focusing on compositional generalization. We use the 32-shot version in Yin et al. (2021) and denote it as SMCalFlow-CS in later experiments. As our experiments also aim to provide a fair comparison between the original version and the compositional version of SMCalFlow, we preprocess both versions using the same way as the preprocessing steps for the compositional version

described in Yin et al. (2021). Due to this preprocessing detail, our result on the non-compositional split of SMCalFlow may not be comparable to other works.

A.2 Dataset Construction Details for SCAN* Length Generalization Experiment

SCAN* is an extended version of SCAN created in this work so that we can analyze how dataset complexity (especially length-related complexity) influences the model's generalization ability. This is an essential design choice as the complexity of the original SCAN is bounded. For all SCAN* related experiments, we apply two changes: first, every dataset will contain 200 different primitives; second, every sentence can have an unlimited number of *and* and *after* as long as the total length is within the limit for the length generalization experiments. To avoid ambiguity, we assign different orders of priority to *and* and *after*, with *after* having the higher priority and will be executed first. Additionally, we make two minor simplifications to the SCAN grammar that do not influence the trends substantially. We removed the verb *turn* and we unify the grammar for *around* and *opposite*[10]. To generate the data for the length generalization experiment in Sec. 3, and the length ablation experiment in Sec. 5, we try to ensure similar dataset statistics across datasets with different lengths as much as possible. To achieve this, we always first generate the dataset with the maximum length. Then, to create all the other shorter experiments, we repeatedly create a half-length truncated version of the longer original dataset. Specifically, for every example with length l in the longer dataset, we first split each example by its conjunctions (*and* or *after*), then we keep the most amount of the split parts so that the total length is under the half-length $l/2$ of the original example. For our experiments, we use the length of the input as the length measure.

B Implementation Details

B.1 Models and Training Details

In this study, we focus on seq2seq Transformers as they are the most prevalent choices for most compositional generalization (and more broadly, semantic parsing) datasets. For our main experiments, we focus on models trained from scratch so that our analysis is not influenced by the existing knowledge from pretraining. For the experiments in the main paper, we follow Csordás et al. (2021) to use 3 layers in both encoder and decoder, and use relative positional embeddings (Shaw et al., 2018). In Appendix H, we provide additional results on models with other configurations. We set both the hidden size and the embedding size to 256, and the dimension of the feed-forward layer to 512. We use a dropout rate of 0.1 and 4 self-attention heads. For the optimizer, we use Adam (Kingma and Ba, 2015) and a batch size of 128 examples. We use the Noam Learning rate scheduling with a peak learning rate of 2.0, 50000 total steps, and 5000 warmup steps. We use a beam size of 5 to get all the results in our experiments.

B.2 Implementation of Difficulty Metrics

For the prototype-based difficulty metric, we follow the process in Sorscher et al. (2022). For every example, we feed the input side to the SimCSE (Gao et al., 2021) model to get an example embedding. We choose to use the input side instead of the output side to compute the embedding since the inputs are natural language so they are more suitable for off-the-shelf models like SimCSE. Then, we run k-means to get the distance between each example embedding and its corresponding cluster centroid. Our k-means implementation is from (Pedregosa et al., 2011). We set k to 100 in our experiments, and use the best clustering result from 10 different initializations.

For the learning learning-based difficulty metric, we follow our standard setup to evaluate our model on the training set for every 500 steps. We then log the performance of each example. Then for every example, we find the earliest step when the model has predicted the example correctly for 10 consecutive steps. The earlier this step is, the simpler this example is. If the model can never predict the example correctly, this step number is set to the final step. In our experiment, we compute this metric on multiple random seeds. Therefore, for every example, the final step number is the average over all the different seeds.

C Data Augmentation Details

C.1 Data Augmentation Details of SCAN *Jump* and *Around Right*

We first follow the exact same setup in Jiang et al. (2022) to conduct data augmentation experiments on the SCAN *Jump* and *Around Right* split to create

[10] In Appendix E, we show this simplification does not influence the trends shown in this paper

training sets with different numbers of primitives. The augmentation follows a two-stage procedure, which we explain here.

Building a Dictionary. In the first stage, we use a dataset-agnostic, rule-based algorithm to build a dictionary that maps certain input tokens to their output forms (e.g., "*look* $\mapsto$ `LOOK`" and "*jump* $\mapsto$ `JUMP`"). Given the source vocabulary as V and the target vocabulary as W, the algorithm iterates through the training set to identify pairs $(v, w), v \in V, w \in W$ such that the presence of v in the input is both *necessary* and *sufficient* for the presence of w in the output.

$$\begin{aligned} \text{suff}(v, w) &= \forall(x, y), (v \in x) \rightarrow (w \in y) \\ \text{ness}(v, w) &= \forall(x, y), (w \in y) \rightarrow (v \in x) \end{aligned} \tag{1}$$

This algorithm ignores those functional words (e.g., "*around*" and "*twice*") that only decide the syntactic structure of the outputs but cannot be translated to a specific target token.

Mutating Primitives. In stage two, we iterate through every example in the training set and randomly select some primitive pairs that exist in the previously built lexicon for mutation. In mutating them, we simply add a suffix to their source and target forms. Given an original example "*walk left twice*", we select a primitive "*walk*" and mutate it to a new example "***walk1*** *left twice* $\mapsto$ `TL` **`WALK1`** `TL` **`WALK1`**". For our experiments in Sec. 3, we create three different augmented training sets (2x, 20x, 200x) with increasing complexities. Specifically, for a training example in the SCAN *Jump* or *Around Right*, we first identify all primitive pairs in the example. Then, in Kx augmentation (K=2,20,200), for each primitive pair (e.g., "(*walk*, *WALK*)"), we randomly replace the source token in the input with one of its $K+1$ mutated form (*walk*, *walk1*, *walk2*, ..., *walk*K) and the target token in the output with the corresponding form. We repeat this process $2K$ times or until we collect K distinct augmented examples.

C.2 Data Augmentation Details for GeoQuery

Here we describe the detailed data augmentation process for the GeoQuery dataset as used in Sec. 3 and Sec. 4. As one crucial part of the questions in GeoQuery are geography entities (e.g., *Oregon*, *Springfield*, etc.), our augmentation focuses on increasing the diversity of these entities similar to the primitive-based augmentation method in Patel et al. (2022) and Jiang et al. (2022). Specifically, we first get all the geography entities in the dataset by parsing the output predictions. Then, for each entity, we create multiple copies (e.g., 19 new copies for the x20 experiments) for the entity. Then, for every example, we identify the entities contained in every example, and then augment new examples containing the new copies of these entities. We augment the same amount of new examples for every example. The final size of the x20 augmented dataset will be 20 times of the original dataset.

C.3 AugZero Augmentation Details

Below we describe *Augmentation with Zero Prior Knowledge* (AugZero). The general idea is just to satisfy our hypotheses by providing multiple copies of the training set using different copies of the vocabulary. Interestingly, this process requires *zero* knowledge about the actual data. Specifically, assume the original vocabulary is $V^1 = \{w_j^1|_{j=1}^m\}$, where $w_1^1 \ldots w_m^1$ are all the m possible tokens. AugZero increases the vocabulary by k times. The augmented vocabulary will be $V^{ADV} = V^1 \cup V^2 \cup \ldots \cup V^k = \{w_j^i : |_{j=1}^m{}_{i=1}^k\}$, where each w_j^i is a corresponding new token for w_j^1. Given a tokenized example with length l in the original dataset: $x = w_{u_1}^1, w_{u_2}^1, \ldots, w_{u_l}^1$, where $u_1, \ldots, u_l$ are the token indexes in V^1. In AugZero, we create $k-1$ additional copies of this example, from $w_{u_1}^2, \ldots, w_{u_l}^2$ to $w_{u_1}^k, \ldots, w_{u_l}^k$. See Figure 3 for an illustration. One crucial difference between AugZero and the data augmentation described in Sec. 3 is that AugZero no longer distinguishes the primitive and non-primitive tokens, hence requiring zero knowledge about the actual task.

Other variants of AugZero. In AugZero, we select a different set of vocabulary for each example. For example, the original "*walk left and jump*" may become "*walk2 left2 and2 jump2*". If we combine this idea and syntax induction methods, we can get additional augmentation examples such as "walk left2 and2 jump2". This can further improve the performance (as shown by the prim2primX results using very similar ideas). However, doing such augmentation requires inducing the syntax mapping beforehand, which is not trivial on natural language datasets and loses the advantage of simplicity. Additionally, we explored another variant that maps each original token in the dataset randomly into all the corresponding words in the augmented vocabulary. This makes the one-to-one map "*walk* $\mapsto$ `WALK`" becomes a many-to-many map "{*walk*,

Dataset Size	Repetition	GeoQuery (query)	GeoQuery (question)
baseline	/	29.67±5.31	60.79±1.69
baseline	Example	8.68±4.17	56.63±2.50
baseline	Primitive	13.87±2.70	36.77±1.90
20x	/	32.97±2.60	62.44±1.80
20x	Example	30.00±1.85	62.65±1.38
20x	Primitive	6.48±2.21	40.05±1.07

Table 6: The effect of examples repetition on different datasets with SQL outputs. We use SQL outputs for the GeoQuery dataset in this table.

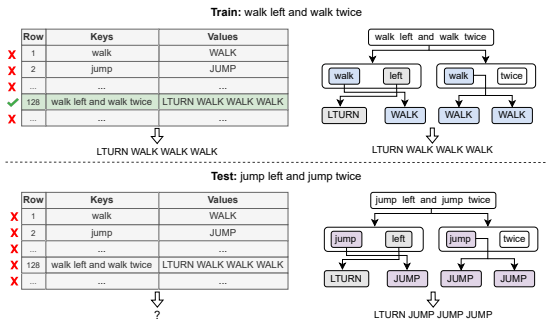

Figure 6: Two different ways to achieve low training losses: surface memorization (left) and compositional understanding (right).

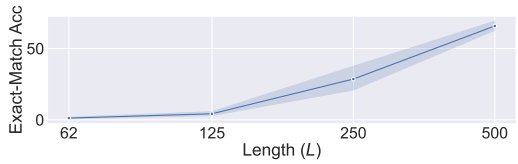

Figure 7: Different generalization performances on without the grammar simplification of *around* and *opposite*. Same to Figure 2, the challenge is always to train on examples with length $0 < l \leq L$ and test on examples with length $L < l \leq 2L$.

walk1, *walk2*, ...} ↦ {WALK, WALK1, WALK2, ...}", but do not show further improvements.

D Illustration for the Two Model Behaviors

In Figure 6, we provide an illustration for the two model behaviors mentioned in Sec. 4.1. Both behaviors can lead to good performance on the training set (as shown in the upper half of the figure). However, given a new example with a novel combination of seen structures and primitives, only models with correct compositional understanding can provide the correct prediction (as shown in the lower half of the figure).

	GeoQuery (query)	GeoQuery (question)
baseline	29.67±5.31	60.79±1.69
2x Augmentation	26.05±4.34	61.72±2.35
20x Augmentation	32.97±2.60	62.44±1.80
200x Augmentation	32.64±1.00	61.08±2.11

Table 7: Datasets with increased complexity via data augmentation are easier for compositional generalization. We use SQL outputs for the GeoQuery dataset in this table.

E Additional Length Generalization Results

In Figure 7, we show length generalization results without the grammar unification of *around* and *opposite*. We can see the trend is very similar to Figure 2. As the maximum length becomes larger in the dataset, generalization performance becomes better. In our preliminary experiments, we have verified that our other findings on SCAN* are also robust against minor grammar changes.

F Additional GeoQuery Results

In the main paper, due to space constraints, we report on the GeoQuery dataset with logical forms as output. In Table 6 and Table 7, we show the corresponding results using SQL as the output. Both tables demonstrate similar trends to the results in the main paper.

G Additional Example Difficulty Results

G.1 Examples of Prototype-Based Difficulty

In Table 8, we show examples of simple and difficult data in the SMCalFlow-CS dataset according to the prototype-based difficulty. In the table, the simple examples are randomly sampled from the simplest 25% of the dataset, while difficult examples come from the hardest 25%. We can see that the simple examples usually refer to common general instructions, while difficult examples tend to be more complex in the language and specify more

Simple Examples	Difficult Examples
Can you create an Meeting for Saturday 1 : 00 pm	I had a meeting with Jesse last week
Schedule a lunch after the meeting on Thursday.	I need my free day on April 7, to be turned to my All out gallery opening.
create a new appointment tomorrow	Add a beer festival in Denver to be for all weekend next week.

Table 8: Simple and difficult data examples in the SMCalFlow-CS dataset according to the prototype-based difficulty metric. In this table, the simple examples are randomly sampled from the simplest 25% of the dataset, while difficult examples come from the hardest 25% of the dataset.

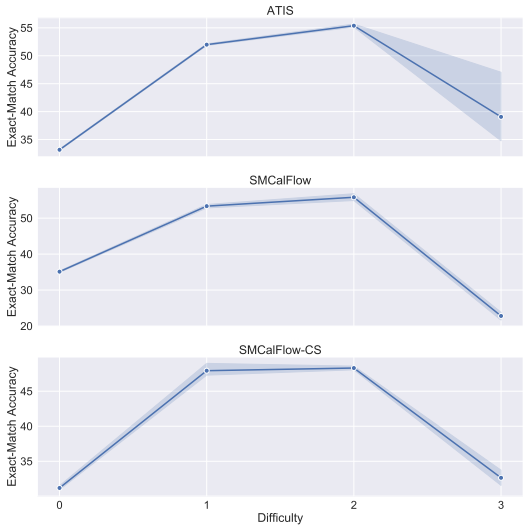

Figure 8: Results on ATIS and SMCalFlow with training sets of different difficulties with learning-based difficulty.

details (e.g., the name Jesse, the place Denver, etc.).

Additionally, we want to point out that while this prototype-based difficulty measure alters the general complexity of the data, it *does not* change the compositional difficulty of the SMCalFlow-CS dataset. This is because the dataset of SMCalFlow-CS is constructed to test the ability to combine multiple instructions in the training set under a few-shot setting (Yin et al., 2021), and the difficulty is controlled by altering the number of few-shot examples. As we always keep the few-shot examples the same in all these training sets and only change the difficulty of the remaining single-skill examples, the clustering step will not introduce other confounding factors to the compositional challenge.

G.2 Learning-Based Difficulty

We can use an empirical metric based on how soon the model predicts an example correctly during training. This is similar to the popular correctness-based metric (Swayamdipta et al., 2020). Since the model could reach perfect training accuracy on many of the datasets in this work, we choose to use the dedicated metric of *how soon* the model predicts the example correctly instead of the plain accuracy, so that we can differentiate between examples. Specifically, during training, we evaluate the model on the entire training set for every checkpoint. Then for every example, we log the earliest step when the model has predicted the example correctly for 10 straight checkpoints. See Appendix B for details.

We report additional results of our difficulty-based sub-sampling results in Sec. 5. In Sec 5 in the main paper, we use prototype-based difficulty metrics to conduct the sub-sampling experiments. We make this design choice as if we use the learning-based difficulty metric, then the models perform extremely badly on the most difficult split, which makes it unsuitable for our experiments. The corresponding results are shown in Figure 8.

H Results with Other Model Configurations

H.1 Data Augmentation Results with Larger Trained-From-Scratch Transformers

For the main results in the main paper, we use 3-layer Transformers as it is suggested as the best choice for many popular compositional generalization datasets (Csordás et al., 2021). In Table 9, we report the data augmentation number for a larger 6-layer model. We see similar trends on all the datasets, confirming that the advantage brought by increased data complexity is not sensitive to the underlying model scale.

H.2 Results with Pre-trained Transformers

As the main motivation of this work is to examine how data factors influence model generalization performance, we mainly experiment with models trained from scratch so that we can have clean con-

	Jump	*Around Right*	GeoQuery (query)	GeoQuery (question)
baseline	4.14±4.71	52.42±21.41	37.63±5.57	58.42±1.56
2x Augmentation	31.79±26.23	79.26±7.78	37.42±1.55	60.57±2.96
20x Augmentation	92.13±4.56	77.50±22.34	41.20±2.68	66.95±1.17
200x Augmentation	99.90±0.09	99.55±0.85	39.68±5.08	62.20±1.89

Table 9: Datasets with increased complexity via data augmentation are easier for compositional generalization. We use 6-layer Transformers and logic form outputs for the GeoQuery dataset in this table.

Dataset Size	Repetition	GeoQuery (query)	GeoQuery (question)
baseline	/	67.74±5.61	80.05±1.97
20x	/	72.04±10.80	79.69±0.20
20x	Example	65.41±2.54	81.72±1.99
20x	Primitive	51.08±7.27	61.65±4.66

Table 10: The effect of examples repetition on different datasets for pre-trained Transformers.

trol over all the influencing factors and avoid the interference of pretraining data and testing data. Nonetheless, when we actually deploy models in practical applications, the more common choice is to use pretrained models (e.g., T5 (Raffel et al., 2020), BART (Lewis et al., 2020), etc.). Below we reproduce some of our main results in this paper with T5-base models and summarize the takeaways. For the hyper-parameters, we mostly follow the same setting described in Sec. 2.2. The only change is that we use a learning rate of 0.001 and a linear learning rate scheduler. For the T5 experiments, we report mean and standard deviations from 3 runs.

Preventing repetition in data is still important. In Table 10, we reproduce the experiments in Table 4 where we test the effect of repeating examples during the training process. From the results, we can see that similar to the results in the main paper. When the model is repetitively trained on a randomly sampled small set (i.e., the 3rd row in the table), we can only see a small drop in the query split. However, the damage is more substantial when the repeated subset contains few primitives, showing a drop from 72.04 to 51.08 on the query split and a drop from 79.69 to 61.65 on the question split. To summarize, example frequency is still very important during finetuning. Highly repetitive examples can cause substantial damage to the generalization performance, which may also be linked with overfitting on the small subset.

Pretrained Transformers need less assistance for simple cases. One crucial difference between the pretrained models and the models trained from scratch is that pretrained models already have a basic understanding of natural language. For example, by pretraining on a large corpus, it already knows that *run* and *jump* are both verbs; *Texas* and *California* are both location entities, and these words in the same category should be treated similarly. Therefore, it gets a 'jump-start' on these compositional generalization datasets, and may not need a large number of unique primitives to infer the equivalence of *run* and *jump*. As a result, we do not observe consistent improvement by using data augmentation in Table 10.