Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures

Daniel Furrer* **Marc van Zee*** **Nathan Scales** **Nathanael Schärli**
Google Research, Brain Team

{danielfurrer,marcvanzee,nkscales,schaerli}@google.com

Abstract

While mainstream machine learning methods are known to have limited ability to compositionally generalize, new architectures and techniques continue to be proposed to address this limitation. We investigate state-of-the-art techniques and architectures in order to assess their effectiveness in improving compositional generalization in semantic parsing tasks based on the SCAN and CFQ datasets. We show that masked language model (MLM) pre-training rivals SCAN-inspired architectures on primitive holdout splits. On a more complex compositional task, we show that pre-training leads to significant improvements in performance vs. comparable non-pre-trained models, whereas architectures proposed to encourage compositional generalization on SCAN or in the area of algorithm learning fail to lead to significant improvements. We establish a new state of the art on the CFQ compositional generalization benchmark using MLM pre-training together with an intermediate representation.

1 Introduction

Human intelligence exhibits *systematic compositionality* [Fodor and Pylyshyn, 1988], the capacity to understand and produce a potentially infinite number of novel combinations of known components, i.e., to make "infinite use of finite means" [Chomsky, 1965]. Humans demonstrate this ability across diverse domains, including natural language understanding (NLU) [Lake and Baroni, 2018] and visual scene understanding [Johnson et al., 2017b, Higgins et al., 2018]. For example, we can learn the meaning of a new word and then apply it to other language contexts. As Lake and Baroni [2018] put it: "Once a person learns the meaning of a new verb 'dax', he or she can immediately understand the meaning of 'dax twice' and 'sing and dax'."

In contrast, state-of-the-art machine learning (ML) methods often fail to capture the compositional structure that is underlying the problem domain and thus fail to generalize compositionally [Keysers et al., 2020, Lake and Baroni, 2018, Bastings et al., 2018, Loula et al., 2018, Johnson et al., 2017b].

The past several years have seen an increased focus on compositional generalization in both natural language and other domains, leading to the creation of datasets and specialized train-test splits to test compositional generalization [Lake and Baroni, 2018, Finegan-Dollak et al., 2018, Keysers et al., 2020, Johnson et al., 2017b, Hudson and Manning, 2019a, Sinha et al., 2019] and the development of specialized architectures and training techniques intended to enforce a bias toward compositional generalization [Li et al., 2019, Gordon et al., 2020, Nye et al., 2020, Hudson and Manning, 2019a, Kaiser and Sutskever, 2015, Freivalds et al., 2019, Andreas, 2019, Hill et al., 2019, Lake, 2019].

In the area of natural language – and semantic parsing in particular – SCAN [Lake and Baroni, 2018] has become a de facto standard benchmark and test bed for research in compositional generalization, inspiring a series of new techniques [Li et al., 2019, Russin et al., 2019, Andreas, 2019, Lake, 2019, Gordon et al., 2020, Nye et al., 2020]. These techniques indeed lead to significant gains over vanilla baselines, including up to perfect performance on the primitive holdout splits. However, they frequently involve tailoring to the SCAN task, which raises the question to what degree these improvements would translate to other tasks.

Recently, Keysers et al. [2020] introduced the Compositional Freebase Questions (CFQ), a more realistic and larger scale semantic parsing dataset with rich compositional structure, together with the Distribution Based Compositionality Assessment (DBCA) method, which seeks to measure compositional generalization more comprehensively by making the distributions of compounds as different as possible between train and test, while keeping

* Equal contribution. A description of each author's contribution is available in Appendix A.

the distribution of atoms the same. While baseline evaluations in Keysers et al. [2020] show that three sequence-to-sequence architectures all struggle with compositional generalization on DBCA-based splits of both SCAN and CFQ, techniques designed specifically for compositional generalization on SCAN or other tasks have yet to be evaluated with DBCA.

At the same time, significant gains have been achieved in recent years in natural language tasks including semantic parsing through the use of MLM pre-training with systems such as BERT [Devlin et al., 2018] and T5 [Raffel et al., 2019] and in semantic parsing in particular through the introduction of intermediate representations that simplify the output format or map more closely to the input syntax [Guo et al., 2019]. Neither of these techniques has yet been evaluated, however, in semantic parsing scenarios specifically designed to test compositional generalization.

In this paper, we address some of those open questions and make the following contributions:

- We provide an overview of a range of architectures and techniques that have previously been applied to SCAN or CFQ and provide the most comprehensive summary of results so far (Table 1), including results on detailed SCAN splits of two architectures (Transformer, Universal Transformer) that were previously evaluated only on CFQ and SCAN MCD splits.
- Using the DBCA method on CFQ and SCAN, we evaluate two representative architectures that had been proposed to improve compositional generalization (CGPS, Neural Shuffle Exchange Network) and find that neither show significant improvement compared to general-purpose sequence-to-sequence architectures.
- We evaluate the use of MLM pre-training on SCAN and CFQ and demonstrate that off-the-shelf pre-training is sufficient to nearly match the gains on SCAN's "Add jump" primitive holdout split achieved to date by SCAN-inspired specialized techniques, while obtaining significant improvements over comparable non-pre-trained models on CFQ.
- By combining pre-training with an intermediate representation, we obtain a new state-of-the-art score for CFQ of **42.1%** on the MCD-mean split, beating the previous best results of 18.9%.

This paper is organized as follows: Section 2 describes SCAN and CFQ, our two compositional generalization benchmarks of interest; Section 3 provides and overview of all architectures and techniques, which are then evaluated and analyzed in Section 4, and Section 5 discusses related work.

2 Background: Measuring Compositional Generalization

2.1 SCAN

SCAN [Lake and Baroni, 2018] is a semantic parsing dataset of natural language navigation commands (e.g. `turn left twice and jump`) that are mapped to corresponding action sequences (e.g. `LTURN LTURN JUMP`). The SCAN dataset is compositional in the sense that it is constructed from a simple grammar consisting of rules correponding to primitive commands such as `jump` and `turn left` as well as rules that compose actions into more complex sequences.

Based on this explicit construction, Lake and Baroni [2018] and Loula et al. [2018] introduce various experiments using train/test splits that are specifically designed to measure compositional generalization. For example, some experiments perform the train/test split based on the length of the commands, while other experiments make sure that a certain primitive command (such as `jump`) only occurs in very limited combinations during training. Yet other experiments hold out whole subcommands (such as `jump around right`) or templates (such as `$Primitive around left`) in training.

2.2 DBCA and CFQ

While all SCAN experiments are insightful, it is not clear which experiments are most relevant for measuring compositional generalization [Bastings et al., 2018], and the performance of individual architectures varies significantly across different experiments (see Table 1). Keysers et al. [2020] address this concern by introducing distribution-based compositionality assessment (DBCA), which is a novel method to construct compositionality benchmarks more systematically.

Given a dataset (like SCAN) that is generated from a set of rules, they track for each example the individual rules (atoms) and rule combinations (compounds) that were used to construct the example. Using this information, they then generate "maximum compound divergence" (MCD) splits, which maximize the compound divergence while guaranteeing a small atom divergence between train and test sets. MCD splits are well suited for assessing compositionality because they are both fair (because the distribution of individual rules is similar) and compositionally challenging (because the distribution of compounds is as different as possible).

Together with the DBCA method, Keysers et al. [2020] also provide CFQ, which is a simple but realistic and large natural language dataset that is specifically designed to measure compositional generalization using DBCA. The task of interest is semantic parsing from a natural language question

(such as 'Which art director of [Stepping Sisters 1932] was a parent of [Imre Sándorházi]?') to a SPARQL query, which can then be executed against the Freebase knowledge base. Named entities are anonymized, which is standard practice and ensures that the models do not have to learn all the entities.

The authors release a number of MCD splits for both SCAN and CFQ, and they show that there is a strong negative correlation between the accuracy of three standard sequence-to-sequence architectures and the compound divergence. They investigate this for *LSTM+attention* (an LSTM [Hochreiter and Schmidhuber, 1997] with attention mechanism [Bahdanau et al., 2015]), for *Transformer* [Vaswani et al., 2017], and for *Universal Transformer* [Dehghani et al., 2018].

3 Architectures and Techniques

In this section we give an overview of general sequence-to-sequence architectures that have been applied in the past to the SCAN or CFQ benchmarks, together with specialized architectures and techniques (SCAN-inspired and algorithm learning-based approaches), and MLM pre-trained models. We present in Section 4 the results that each of these approaches obtains.

3.1 General Sequence-to-sequence Architectures

LSTM Lake and Baroni [2018] evaluate simple recurrent networks (SRNs; Elman [1990]), long short-term memory networks [Hochreiter and Schmidhuber, 1997], and gated recurrent units (GRUs; Chung et al. [2014]), each with and without an attentional mechanism [Bahdanau et al., 2015] on a number of SCAN splits. They find the overall-best architecture (determined by hyperparameter search) to be an LSTM without attention.

CNN Dessì and Baroni [2019] evaluate convolutional networks on some SCAN splits and find that "CNNs are dramatically better than RNNs at compositional generalization".

GRU, GRU-dep Bastings et al. [2018] analyze SCAN and conclude that it has very few target-side dependencies, and therefore simpler architectures tend to perform better on it. They focus on RNNs with GRU cells and also evaluate a GRU-based architecture that is non-autogressive [GRU-dep] which indeed performs better.

LSTM+A, Transformer, Universal Transformer Keysers et al. [2020] evaluate these three architectures using DCBA splits of CFQ and SCAN. Hyperparameters are obtained using a search on a random CFQ split.

Evolved Transformer So et al. [2019] found this architecture via an evolutionary neural architecture search that was seeded with a standard Transformer architecture. The most notable difference from a standard Transformer is the use of wide depth-wise separable convolutions in the early layers of both the encoder and decoder. The evolution was done on an English-German translation task, and the final architecture consistently beats the standard Transformer in several translation tasks as well as language modeling. Yun et al. [2019] independently show that depth-wise separable convolutions have interesting theoretical properties, being more efficient than attention while keeping the transformer architecture functionally universal.

3.2 SCAN-inspired Approaches

The SCAN dataset has inspired many models that improve compositional generalization in some ways. Most of these approaches are guided by the observation that the mapping of input to output surface forms (e.g. the command `jump` to the action `JUMP`) can be handled separately from the mapping of the structure (e.g. `X twice after Y` to `Y X X`). A model that decouples these concerns - i.e. is equivariant to substitution of primitives - generalizes better.

Syn-att Russin et al. [2019] call this decoupling "separating syntax from semantics" and draw on neurosience research to further motivate their approach. They implement two separate encodings for each input token: one which maps each token independently (without context) and one which uses recurrence. This latter representation is used as an attention map over the former.

CGPS Li et al. [2019] propose an almost identical architecture but add entropy regularization to the encodings and show that this is important for reducing variance. The model is thoroughly evaluated on many SCAN splits and performs well on most. Like Syn-att, it is also conceptually simple and does not require complicated learning setups or additional supervision (unlike many other models mentioned below).

We select this model as a representative of SCAN-inspired approaches to evaluate the degree to which its gains carry over to the MCD splits of SCAN and CFQ. For CFQ we adapt the setup slightly to deal with the fact that the architecture assumes that every output token is directly mappable from an input token: We add a prefix whose token length is equal to the number of tokens not otherwise directly mappable (in particular, SPARQL syntactic tokens such as `SELECT`, etc.) to all inputs. We also run a hyperparameter search to optimize for the CFQ task (see Appendix B).

Equivariant Gordon et al. [2020] seek to achieve

a similar equivariance across primitive substitutions, building on the conceptually appealing notion of group convolutions [Cohen and Welling, 2016, Kondor and Trivedi, 2018]. The applicability of this approach in its current form is limited by the need for task-specific pairs of related input/output forms to be explicitly provided as a side input.

GECA Andreas [2019] introduces a data augmentation technique called Good Enough Compositional Data Augmentation (GECA) that seeks to provide a compositional inductive bias by automatically detecting "templates" that occur multiple times in training and then mechanically constructing additional training examples from these templates by filling them with different text "fragments" that were observed in similar contexts to the original fragments. Despite the tendency to introduce grammatically or semantically "incorrect" examples, the authors report significant improvements on several SCAN splits. (See also Ruis et al. [2020], however, which reports largely negative results for GECA when applied to a different compositional task).

LANE Concurrently with this work, Liu et al. [2020] propose a modular architecture with memory, which consists of two cooperating modules – a Composer and a Solver – trained with hierarchical reinforcement learning using a curriculum. The Composer obtains "analytical expressions" (which are sequences with words and variables) from the inputs, while the Solver converts a variable-based source expression into a corresponding variable-based destination expression and then assigns the variables through interaction with memory. This approach has the benefit of not depending on the sorts of task-specific extra resources required by the meta-learning approaches described below, while demonstrating perfect generalization on SCAN splits including length and MCD. We expect that the method for identifying and composing expressions would need modification in order to apply it to CFQ, which the authors propose to do in future work.

3.2.1 Meta-learning for SCAN

Another line of research seeks to improve performance on SCAN through meta-learning, in which a system "learns to learn" by being trained on batches of examples generated by grammars that are distinct from, but structurally related to, the true SCAN grammar that is used in evaluation. These meta-learning approaches suffer the disadvantage of requiring task-specific supplementary data at training time, in this case in the form of the meta-learning training data, whose generation requires access to (or manual construction of) a family of grammars that are known to be structurally close to the grammar to be learned.

Meta seq2seq Lake [2019] introduces a meta-learning model which takes as an additional input a set of support input/output pairs. When trained with episodes of carefully crafted problems that are similar to the problem of interest the model learns to use the support set to aid generalization. For example, a model trained with episodes of SCAN samples where the command to action meaning was shuffled (e.g. `jump` → `LOOK`) can solve the SCAN "Add jump" task if provided with the true command to action mapping as support set.

Synth Nye et al. [2020] propose a neural program synthesis approach, in which the system, when exposed to a carefully crafted meta-learning training regime, is able to learn to reverse engineer a symbolic grammar equivalent to the original SCAN grammar. They evaluate their approach on a variety of SCAN splits and achieve perfect scores including on the challenging length split.

3.3 Algorithm Learning

NSEN The Neural GPU [Kaiser and Sutskever, 2015, Freivalds and Liepins, 2017] achieves perfect generalization on a number of algorithmic tasks such as binary addition and multiplication. More recently, Freivalds et al. [2019] show how their Neural Shuffle-Exchange Network (NSEN) outperforms the Neural GPU in many ways and does well on algorithmic problems as well as a language task. We select NSEN as a representative of algorithm learning approaches to evaluate the degree to which it can systematically learn the algorithms of SCAN and CFQ. Both of these tasks can be expected to be learnable algorithmically due to their rule-based nature.

3.4 Masked Language Model Pre-training

Pre-training large, deep neural language models and successive fine-tuning on a variety of downstream natural language processing (NLP) tasks has yielded impressive results [Devlin et al., 2018, Yang et al., 2019, Raffel et al., 2019]. The general idea behind pre-training is that it provides the model with general knowledge of syntax and "world knowledge". While MLM pre-training is a common approach in some semantic parsing domains [Hwang et al., 2019, Guo et al., 2019, Choi et al., 2020, Wang et al., 2019], this has not yet been tried on SCAN or CFQ.

The Text-to-Text Transfer Transformer (T5) [Raffel et al., 2019] treats every NLP task as a text-to-text problem, and is therefore suitable for the semantic parsing tasks we consider in this paper. Raffel et al. [2019] show state-of-the-art results on many NLP tasks such as SQuAD, MultiRC, and BoolQ. T5 is released with multiple pre-trained models, ranging from "small"

(60 million parameters) to "11B" (11 billion parameters).

T5-small, T5-small-NP, T5-base, T5-large, T5-3B, T5-11B In our experiments, we fine-tune all T5 variations on both SCAN and CFQ. For comparison we also train a T5-small model without loading the pre-trained weights (**T5-small-NP**). We use the T5 public pre-trained models[1] and fine-tune them with an inverse square root learning rate schedule on CFQ with 10^4 warm-up steps[2], and a constant learning rate of 0.003 on SCAN (as opposed to the constant learning rate of 0.001 that is used for fine-tuning in the original T5 paper). We use a batch size of 2^{17} tokens per batch. In the original T5 paper, all models are fine-tuned for 256k steps, but we found this number to be too large for SCAN due to the small size of the dataset, which led to overfitting. In order to determine the number of steps to fine-tune, we fine-tune on the "simple" split and choose the number of steps when the model converges, which is around 20k. We replicate most runs 5 times and report the average accuracy and confidence intervals, except for the non-MCD splits where we run only 1 replica for models larger than T5-base. As the variance in the runs with 5 replicas was relatively small, we do not expect the use of single replicas on some splits to affect the overall results we present.

T5's vocabulary does not contain curly braces (see Raffel et al. [2019]; Section 2.2 for their rationale). To work around this, we relax the evaluation to accept an out-of-vocabulary token in the output wherever a curly brace is expected.

3.5 Intermediate SPARQL Representation

Guo et al. [2019] show that significant gains can be obtained on text-to-SQL semantic parsing tasks by using an intermediate representation to help address the mismatch between intents expressed in natural language and SQL. We observe that the structure of natural language questions in CFQ and their semantic parses as SPARQL query can be quite different and that ML models struggle with such examples. Consider for example the question

Did M0 and M1 direct M2 and M3?

which is interpreted with SPARQL clauses like

```
M0 directed M2 .  M1 directed M2 .
M0 directed M3 .  M1 directed M3
```

It is straightforward to translate this to a version that is structurally more aligned with the question

```
{M0, M1} directed {M2, M3}
```

by preprocessing the example outputs for training and then post-processing to obtain the original valid SPARQL form at inference time. We show that such an intermediate representation results in better performance on CFQ and provide more details in Appendix E.

T5-11B-mod We evaluate T5-11B on the CFQ MCD split using the intermediate SPARQL representation where we group both subjects and objects.

4 Results and Analysis

Table 1 summarizes results on all evaluated models, together with results from prior work on SCAN and CFQ. Accuracy results with a white background are existing results, while those with a grey background are results we obtain in this paper. We provide additional experiments on the MCD splits for both SCAN and CFQ in Appendix D. We make four main observations, which we describe in the next subsections.

4.1 Masked Language Model Pre-training

Pre-training helps for compositional generalization, but doesn't solve it.

Comparing T5-small-NP to T5-small, we see that pre-training helps significantly on CFQ, and an increase in T5 model size is correlated with an increase of accuracy on MCD. Furthermore, pre-training is beneficial for all SCAN splits involving holdouts of primitives, subcommands, or templates, with the biggest gains on the "Add jump" primitive holdout split. However, it leads to an average decrease in accuracy of 8.5% on the length split. In Appendix F, we show that on the CFQ MCD split, gains were limited to lengths that were seen in training.

We hypothesize that the primary benefit provided by pre-training is to improve the model's ability to substitute similar words or word phrases by ensuring they are close to each other in the representation space. The scenario which we would expect to benefit most from such a mechanism is that of substitution of a single-word primitive, typified in the "Add jump" task, where pre-training indeed succeeds in achieving near-perfect performance, with lesser gains on the splits requiring substitution of multi-word phrases, subcommands, or groups of phrases described by templates. Pre-training does not, however, fully solve the compositional generalization tasks. We would need further investigation to determine whether pre-training's negative effect on length generalization means that pre-training actively harms the model's ability to learn to build larger constructs through composition of known building blocks.

[1] https://console.cloud.google.com/storage/browser/t5-data/

[2] This is the same schedule that has been used to pre-train the T5 models.

Model	Add jump	Add turn left	Jump around right	Around right	Opposite right	Right	Length	SCAN MCD	CFQ MCD
LSTM	0.1	90.3	98.4 ±0.5	2.5 ±2.7	47.6 ±17.7	23.5 ±8.1	13.8	-	-
LSTM+A	0.0 ±0.0	82.6 ±8.2	**100.0 ±0.0**	0.0 ±0.0	16.5 ±6.4	30.0 ±7.8	14.1	6.1 ±1.7	14.9 ±1.1
CNN	69.2 ±9.2	-	-	56.7 ±10.2	-	-	0.0	-	-
GRU	12.5 ±6.6	59.1 ±16.8	-	-	-	-	18.1	-	-
GRU-dep	0.7 ±0.4	90.8 ±3.6	-	-	-	-	17.8	-	-
Transformer	1.0 ±0.6	**99.6 ±0.8**	**100.0 ±0.0**	53.3 ±10.9	3.0 ±6.8	92.0 ±15.1	0.0	0.9 ±0.3	17.8 ±0.9
Univ. Trans.	0.3 ±0.3	**99.4 ±1.4**	**100.0 ±0.0**	47.0 ±10.0	15.2 ±13.0	83.2 ±18.2	0.0	1.1 ±0.6	18.9 ±1.4
Evol. Trans.	0.6 ±0.6	**100.0 ±0.0**	**100.0 ±0.0**	30.2 ±28.4	11.6 ±14.6	**99.9 ±0.3**	19.8 ±0.0	1.6 ±0.6	20.8 ±0.7
Syn-att	91.0 ±27.4	**99.9 ±0.2**	98.9 ±2.3	28.9 ±34.8	10.5 ±8.8	99.1 ±1.8	15.2 ±0.7	-	-
CGPS	98.8 ±1.4	**99.7 ±0.4**	**100.0 ±0.0**	83.2 ±13.2	89.3 ±5.5	**99.7 ±0.5**	20.3 ±1.1	2.0 ±0.7	7.1 ±1.8
Equivariant*	99.1 ±0.0	-	-	92.0 ±0.2	-	-	15.9 ±3.2	-	-
GECA*	87.0 ±1.0	-	-	82.0 ±4.0	-	-	-	-	-
LANE	**100.0**	-	-	**100.0**	-	-	**100.0**	**100.0**	-
Meta seq2seq*	**99.9**	-	-	**99.9**	-	-	16.6	-	-
Synth*	**100.0**	-	-	**100.0**	-	-	**100.0**	-	-
NSEN	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	1.7 ±0.9	2.8 ±0.3
T5-small-NP	1.4 ±0.8	45.7 ±15.4	**100.0 ±0.0**	5.3 ±4.6	30.5 ±8.7	44.6 ±11.2	19.4 ±0.8	0.9 ±0.5	21.4 ±1.5
T5-small	84.1 ±1.0	73.0 ±5.8	**100.0 ±0.0**	31.8 ±1.0	58.2 ±10.4	88.7 ±8.9	10.9	6.9 ±1.1	28.0 ±0.6
T5-base	**99.5 ±0.0**	62.0 ±0.9	99.3 ±0.3	33.2 ±0.5	99.2 ±0.2	73.5 ±1.8	14.4	15.4 ±1.1	31.2 ±1.3
T5-large	98.3	69.2	**99.9**	46.8	**100.0**	91.0	5.2	10.1 ±1.6	34.8 ±1.5
T5-3B	99.0	65.1	**100.0**	27.4	90.0	76.6	3.3	11.6	40.2 ±4.2
T5-11B	98.3	87.9	**100.0**	49.2	99.1	91.1	2.0	9.1	40.9 ±4.3
T5-11B-mod	-	-	-	-	-	-	-	-	**42.1 ±9.1**

Table 1: Accuracy of various models (first column) on the traditional SCAN splits, the SCAN MCD-mean split (second-to-right column), and the CFQ MCD-mean split (rightmost column). Models in the first group are general-purpose architectures; the second group are SCAN-inspired approaches; the third group are SCAN-inspired approaches requiring a special meta-learning task setup; the fourth group are architectures designed for algorithmic learning; the last group are T5-based models. For the MCD splits, the reported variance is the 95% confidence interval, while for all other splits it is the stdev. Models marked with * take additional knowledge as side inputs. Cells with a white background are results obtained in previous papers; cells with a grey background are results obtained in this paper. Boldfaced results are 0.5% points within the best result.

Note that while the pre-trained T5 models consistently outperform T5-small-NP on splits other than the length split, relative performance is not fully consistent between Transformer and T5-small-NP, which are both (non-pre-trained) Transformers. These differences are unrelated to pre-training per se and, similarly to inconsistent relative performance described in Section 4.3, would require additional investigation to explain.

4.2 Specialized Architectures

For the specialized architectures we evaluated, improvements obtained on one compositional generalization benchmark do not transfer to others.

CGPS – while showing strong performance on traditional SCAN splits – performs poorly on MCD splits, and is outperformed by all general-purposes architectures on CFQ and by the basic LSTM+A architecture on SCAN. As shown in Appendix D, the underperformance of CGPS vs. other general-purpose architectures is even more noticeable at moderate levels of compound divergence. While it would be unsurprising for CGPS to show only limited gains in high compound divergence splits due to the relatively small number of examples in those that can be directly solved by primitive substitution alone, this is insufficient to explain why CGPS performs worse than general-purpose architectures. It appears rather that the CGPS mechanism, unlike pre-training, is not robust to shifts in compound distribution and even introduces negative effects in such circumstances.

NSEN performs very poorly on all of the splits we evaluated. It may be noted that while we did manage to train NSEN to 98.3% on a SCAN random split, on CFQ the maximum accuracy we were able to obtain is 56.6% even on a random split. One noteworthy difference between NSEN and the other architectures is that NSEN is not autoregressive. However, in Appendix C, we show through an ablation study that autoregression is only a small factor in the stronger performance of LSTM+A, suggesting it is unlikely to fully explain the poor performance of NSEN. We hypothesize that while

NSEN has been shown to perform well on algorithms and simple language tasks that align cleanly to its shuffle exchange architecture, it is insufficiently general to efficiently learn the more complex task of semantic parsing. This hypothesis is supported by NSEN's relatively poor performance on even the random split of the more structurally complex CFQ task.

A remaining question for follow-up research is whether the most recent specialized approaches LANE and Synth (which have been developed concurrently with this work) could break this trend, although we expect that both of them would have to be adapted substantially for CFQ.

4.3 General Sequence-to-sequence Architectures

General ML architecture improvements yield incremental, if limited, improvements in compositional generalization settings.

On CFQ MCD, the performance order of models (Evolved Transformer > Transformer > LSTM+A) corresponds with what we would expect based on results obtained in other areas such as machine translation [Vaswani et al., 2017, So et al., 2019]. On SCAN MCD, LSTM+A outperforms all other general architectures, but this seems to be a peculiarity not present in lower compound divergence DBCA splits (see Figure 2 in Appendix D). On 5 out of 7 of the other SCAN splits, Transformer outperforms LSTM+A, while Evolved transformer outperforms Transformer.

Compared to the Transformer and the Universal Transformer, the Evolved Transformer's performance on the length split is much better, and mirrors a similar performance improvement in T5-small-NP, another Transformer variant. Further comparison between these architectures is needed to explain this surprising result.

4.4 Interpretation of Prior Work

It is challenging to evaluate compositional generalization using the traditional SCAN splits.

While the SCAN splits have served an important purpose in motivating innovative research on compositional generalization, the number of splits is somewhat unwieldy, and perhaps as a result, many models have not been evaluated on all splits. Furthermore, as seen, for example, in comparison of performance between Transformer and Evolved Transformer on "Around right" vs. "Opposite right", or between Transformer and T5-small on "Add jump" vs. "Add turn left", the relative performance of models is not strongly correlated across splits. This makes it difficult to perform an unambiguous evaluation of the compositional generalization abilities of different architectures. MCD splits alleviate this problem by providing a single yet comprehensive measure for compositional generalization.

5 Related Work

We discuss here additional lines of research related to improving compositional generalization of ML systems and performance of semantic parsing systems in settings other than SCAN or CFQ. Many involve ideas potentially applicable to CFQ or SCAN and suggest potential areas of future work.

Pre-training on artificial languages While we show that MLM pre-training already yields significant performance improvements on both CFQ and SCAN, it has likely not achieved its full potential on these tasks, as we pre-train currently only on natural language – the input language – whereas the output (SPARQL or SCAN actions) is in an artificial language with a distinct syntax, and whose tokens have at most a loose connection to English words. Recent research has shown success in applying language model pre-training to artificial languages as well, which could potentially be applicable to CFQ, given a sufficiently large SPARQL corpus. Specifically, Lachaux et al. [2020] apply unsupervised machine translation techniques to the problem of source-to-source compilation, achieving competitive performance by training a language model on documents from each of three programming languages, while relying on tokens with common meaning across these languages to achieve a joint embedding space. Feng et al. [2020] treat natural language (NL) and programming language (PL) as distinct modes and use a combination of paired examples and monolingual documents to train a bi-modal pre-trained model, which they apply to natural language code search and code documentation generation (i.e., PL-to-NL translation). The semantic parsing task of CFQ would involve similar requirements as PL-to-NL translation, but with the direction of translation reversed, while ideally using techniques of Lachaux et al. [2020] to depend only on monolingual sources.

Intermediate representations Intermediate representations to simplify the task of mapping input to output have been applied to a text-to-SQL task in the form of SemQL [Guo et al., 2019], which provides a simplified output space at the cost of some reduced expressivity, and to earlier text-to-SPARQL tasks using representations such as lambda-DCS [Liang, 2013, Berant et al., 2013]. Based on the improvements yielded by the basic intermediate representation adopted in this paper, we see potential for further benefit from development of improved intermediate representations that maintain precision and expressive power while supporting programmatic manipulation and closer mapping

to natural language structures.

Data augmentation Additional evidence in favor of GECA-like data augmentation is provided by Hill et al. [2019], who show that implicit data augmentation effects in a situated agent task can improve compositional generalization. They also observe that generalization to previously unseen action combinations improves as the absolute number of primitive actions and objects observed in training increases, potentially motivating data augmentation techniques that combine known grammatical patterns with additional (possibly artificially generated) primitives. Kagitha [2020] reports preliminary results suggesting that such a technique could improve performance on some SCAN splits. While the tendency to introduce incorrect examples is likely to remain a challenge, data augmentation has the advantage of architecture-independence and as future work could be combined with other approaches evaluated in this paper.

Syntax-constrained decoders In semantic parsing research motivated by the text-to-SQL datasets Spider [Yu et al., 2018b] and WikiSQL [Zhong et al., 2017], a common technique is to use a syntax-constrained decoder to reduce the effective size of the output space and avoid errors in which the system produces malformed output. State-of-the-art systems on these two datasets constrain output via either a grammar-based decoder [Wang et al., 2019] or a sketch-based decoder [Choi et al., 2020]. Grammar-based decoders [Yin and Neubig, 2017] typically involve outputting a leftmost derivation sequence of the output parse tree, with the main gains coming from the relative ease of integration of syntax-based pruning [Xiao et al., 2016]. Sketch-based decoders [Xu et al., 2017] constrain output by forcing the output to follow a given high-level format called a "sketch", containing "holes" that the learned model must fill in. Both approaches are potentially applicable to the text-to-SPARQL task of CFQ, given the strict grammatical structure of SPARQL and, based on the results shown in this paper from simplification of output format, are likely to provide at least marginal improvements in performance.

Order independence in decoder Tasks like CFQ that output SPARQL or SQL have the potential to suffer from the "order matters" problem [Vinyals et al., 2015], due to the mismatch between the sequential form of the expected output and the inherent order invariance of SPARQL constraints and SQL WHERE clauses. Early work on the WikiSQL dataset sought to mitigate this issue through integration of a sequence-to-set architecture [Xu et al., 2017] or through use of a non-deterministic oracle [Shi et al., 2018] to allow for multiple possible correct outputs, leading to modest gains in performance. Current state-of-the-art approaches for WikiSQL or Spider, however, do not adopt such mechanisms.

DB schema encoders Another standard technique on the Spider dataset is the use of graph encoders [Li et al., 2015, Gilmer et al., 2017] to encode the schema of the target database into a representation that can be attended to to influence both the encoding of the question and the decoding of the final output [Shaw et al., 2019, Bogin et al., 2019b,a, Wang et al., 2019]. This technique is more directly relevant to Spider due to its specialized task setup, which focuses on generalization to databases unseen in training and thus depends on the database schema being passed as an additional input. We do believe, however, that encoding DB schema as explicit knowledge could potentially benefit compositional generalization by helping to disentangle the generic language understanding algorithm from the knowledge that should parameterize it and could thus be a relevant future area to investigate for tasks such as CFQ as well.

Other approaches Appendix G describes approaches that are of interest from the broader perspective of compositional generalization, but are not directly applicable to the SCAN or CFQ tasks.

6 Conclusion and Outlook

We investigate state-of-the-art techniques and architectures to assess their effectiveness in improving compositional generalization in semantic parsing tasks based on the SCAN and CFQ datasets.

Our four main findings are as follows: First, pre-training helps for compositional generalization, but does not solve it. Secondly, for the specialized architectures we evaluated, improvements obtained on one compositional generalization benchmark do not transfer to others. Thirdly, improvements to general-purpose sequence-to-sequence architectures generally lead to corresponding incremental improvements in compositional settings. Fourthly and lastly, it is challenging to unambiguously and comprehensively evaluate compositional generalization using the traditional SCAN splits, which may be a reason for researchers to focus more on the MCD splits as a comprehensive measure of compositional generalization.

Our findings suggest several promising routes to further improve compositional generalization. As our pre-trained models were pre-trained on English (the input language) alone, further improvements amy be possible be pre-training on SPARQL (the output language) as well. Performance gains from simplification of output format suggest potential further gains could be achieved through adoption of improved intermediate representations and/or syntax-constrained decoding. We are interested in

evaluating promising approaches such as LANE and Synth that would require more substantial adaptation to CFQ. We also welcome future work evaluating architectures and techniques from related domains on CFQ, as described in the related work.

7 Acknowledgements

We thank: Lukasz Kaiser for useful discussions and pointing us to NSEN; Yuanpeng Li, for interesting discussions and pointers to get CGPS working; Emīls Ozoliņš, for helping us set up NSEN; Adam Roberts for helping us debug the T5 pipeline; Pete Shaw, for pointing us to the intermediate representations that were used in text-to-SQL datasets; Olivier Bousquet for providing guidance and insights, and finally, we thank Xiao Wang for being involved in many of the discussions around our progress and providing useful insights.

References

Jacob Andreas. Good-enough compositional data augmentation. *arXiv preprint arXiv:1904.09545*, 2019.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. URL http://arxiv.org/abs/1409.0473.

Dzmitry Bahdanau, Harm de Vries, Timothy J O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*, 2019a.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? In *ICLR*, 2019b. URL http://arxiv.org/abs/1811.12889.

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. Jump to better conclusions: SCAN both left and right. In *BlackboxNLP@EMNLP*, 2018. URL https://www.aclweb.org/anthology/W18-5407.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, 2013. URL https://www.aclweb.org/anthology/D13-1160.

Ben Bogin, Matt Gardner, and Jonathan Berant. Global reasoning over database structures for text-to-sql parsing. *arXiv preprint arXiv:1908.11214*, 2019a.

Ben Bogin, Matt Gardner, and Jonathan Berant. Representing schema structure with graph neural networks for text-to-sql parsing. *arXiv preprint arXiv:1905.06241*, 2019b.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *arXiv preprint arXiv:2004.03125*, 2020.

Noam Chomsky. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge, 1965.

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

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Roberto Dessì and Marco Baroni. 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, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

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

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. Improving text-to-SQL evaluation methodology. In *ACL*, 2018. URL http://aclweb.org/anthology/P18-1033.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988. URL https://pdfs.semanticscholar.org/d806/76034bfabfea59f35698af0f715a555fcf50.pdf.

Karlis Freivalds and Renars Liepins. Improving the neural gpu architecture for algorithm learning. *arXiv preprint arXiv:1702.08727*, 2017.

Karlis Freivalds, Emīls Ozoliņš, and Agris Šostaks. Neural shuffle-exchange networks-sequence processing in o (n log n) time. In *Advances in Neural Information Processing Systems*, pages 6630–6641, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylVNerFvr.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*, 2019.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. *arXiv preprint arXiv:1912.04971*, 2019.

Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matko Bosnjak, Murray Shanahan, Matthew Botvinick, Demis Hassabis, and Alexander Lerchner. Scan: Learning hierarchical compositional visual concepts. In *ICLR*, 2018. URL https://openreview.net/pdf?id=rkN2Il-RZ.

Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L McClelland, and Adam Santoro. Emergent systematic generalization in a situated agent. *arXiv preprint arXiv:1910.00571*, 2019.

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

Drew Hudson and Christopher D Manning. Learning by abstraction: The neural state machine. In *Advances in Neural Information Processing Systems*, pages 5903–5916, 2019a.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019b. URL https://arxiv.org/pdf/1902.09506.pdf.

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*, 2019.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998, 2017a.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Fei-Fei Li, Lawrence C. Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017b. URL https://arxiv.org/pdf/1612.06890.pdf.

Prabhu Prakash Kagitha. Systematic generalization emerges in seq2seq models with variability in data. *Bridging AI and Cognitive Science. ICLR*, 2020. URL https://baicsworkshop.github.io/pdf/BAICS_11.pdf.

Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3781–3789. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6295-can-active-memory-replace-attention.pdf.

Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

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. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR*, 2020. URL https://arxiv.org/abs/1912.09713.pdf.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690*, 2018.

Marie-Anne Lachaux, Baptiste Roziere, Lowik Chanussot, and Guillaume Lample. Unsupervised translation of programming languages. *arXiv preprint arXiv:2006.03511*, 2020.

Brenden M Lake. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, pages 9788–9798, 2019.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2018. URL https://arxiv.org/pdf/1711.00350.pdf.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. Compositional generalization for primitive substitutions. *arXiv preprint arXiv:1910.02612*, 2019.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Percy Liang. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*, 2013.

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. Compositional generalization by learning analytical expressions, 2020.

João Loula, Marco Baroni, and Brenden Lake. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *BlackboxNLP@EMNLP*, 2018. URL https://www.aclweb.org/anthology/W18-5413.

Maxwell I Nye, Armando Solar-Lezama, Joshua B Tenenbaum, and Brenden M Lake. Learning compositional rules via neural program synthesis. *arXiv preprint arXiv:2003.05562*, 2020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. A benchmark for systematic generalization in grounded language understanding. *arXiv preprint arXiv:2003.05161*, 2020.

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*, 2019.

Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Yasemin Altun. Generating logical forms from graph representations of text and entities. *arXiv preprint arXiv:1905.08407*, 2019.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*, 2018.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text. *arXiv preprint arXiv:1908.06177*, 2019.

David R So, Chen Liang, and Quoc V Le. The evolved transformer. *arXiv preprint arXiv:1901.11117*, 2019.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*, 2019.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *NAACL*, 2018. URL https://www.aclweb.org/anthology/N18-1059.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*, 2019.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, 2016.

Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*, 2017.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769*, 2018a.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex

and cross-domain semantic parsing and text-to-sql task. In *EMNLP*, 2018b. URL https://arxiv.org/pdf/1809.08887.pdf.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

Appendix

A Contributions

Daniel ran the experiments for the Evolved transformer, NSEN and CGPS. Marc set up the T5 pre-training infrastructure, performed the experiments, and developed the intermediate SPARQL representation. Nathan researched related work. Nathanael provided high-level project direction. All authors helped set the scope and research direction and contributed to the writing of the paper.

B Hyperparameters

We provide the hyperparameters used for our experiments in Table 2. All hyperparameters searches were done on a random split of the CFQ dataset.

C Ablation Study: Autoregression

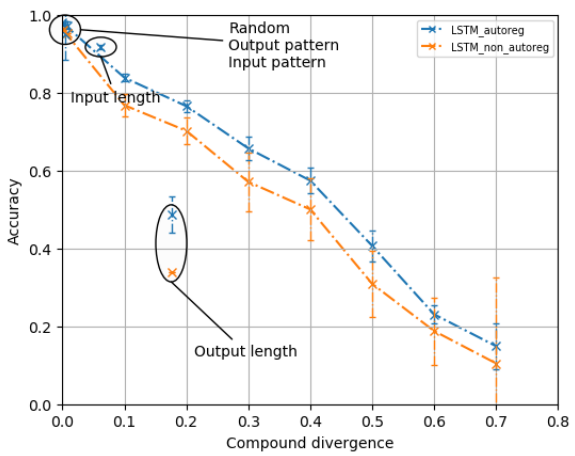

Figure 1: Autoregression ablation, LSTM+A on CFQ.

One significant difference between NSEN and other architectures is that the NSEN is not autoregressive. Autoregression has been found to be important for generative natural language tasks such as machine translation and the NACS (reverse SCAN) task [Kaiser and Bengio, 2016, Bastings et al., 2018]. We show that this is unlikely to explain why NSEN is performing so poorly, however, by running an ablation experiment where we remove autoregression from the LSTM baseline and observe only minimal degradation in performance (see Figure 1). It is also worth noting that Bastings et al. [2018] observes that removing autoregression improves performance on SCAN, which is then given as an argument that simpler architectures do better on SCAN. Our results show that this is not the case for CFQ.

D Additional Experiments on the MCD Splits

We apply the DBCA method to evaluate CGPS and NSEN on both CFQ and SCAN, and compare them with the baselines of Keysers et al. [2020]: LSTM+A, Transformer, and Universal Transformer (Figure 2). Our observations are as follows:

The relative performance of architectures on the CFQ MCD splits are observed consistently across lower compound divergences, starting from even modest divergences of 0.1 to 0.2.

Benefits of Evolved Transformer on SCAN are more noticeable at medium divergences. In particular, at all points other than the MCD, Evolved Transformer matches or outperforms LSTM+A and CGPS (as well as the other architectures), consistent with the improvement observed in the MCD split of CFQ.

NSEN performance drops particularly steeply (on CFQ even more than on SCAN), suggesting that this architecture is particularly brittle with respect to distribution shifts.

We also provide full accuracy results of all MCD splits on both SCAN (Table 3) and CFQ (Table 4) for the architecture that we evaluated in Figure 2, as well as the full pre-training results.

E Intermediate SPARQL Representation

In this section we provide details on the intermediate representation used in the model T5-11B-mod.

Recall from Section 3.5 that the question "Did M0 and M1 direct M2 and M3?" has the following corresponding SPARQL clauses.

```
M0 directed M2 .  M1 directed M2 .
M0 directed M3 .  M1 directed M3
```

Representing the clauses of a SPARQL query by $\{c_1, \ldots, c_n\}$, where $c_i = (subj_i, rel_i, obj_j)$ with $1 \leq i \leq n$, we build the intermediate representation by applying the following three transformations in sequence to the original clauses.

Group subjects (f_1) Replace clauses $(subj, rel1, obj1), \ldots (subj, relk, objk)$ by $(subj, (rel1, obj), \ldots, (relk, objk))$. Applying this transformation to our example gives the following intermediate representation.

```
M0 { directed M2 .  directed M3 }
M1 { directed M2 .  directed M3 }
```

Group subjects and objects (f_2) Apply f_1, and replace clauses $(subj, \{(rel, obj_1), \ldots, (rel, obj_p), (rel_1, obj_{p+1}), \ldots, (rel_s, obj_{p+s})\})$ by $(subj, \{(rel, \{obj_1, \ldots, obj_p\}), (rel_1, \{obj_{p+1}\}), \ldots, (rels, \{obj_{p+s}\})\})$. Applying this transformation to our example gives the following intermediate representation.

	CGPS (CFQ)	Evolved Transformer	NSEN
train steps	8,000	100,000	100,000
batch size	2,048	4,096	4,096
hidden size	512	128	384
embedding size	512	–	–
function embedding size	256	–	–
num hidden layers	–	2	2
num heads	–	8	–
learning rate schedule	–	constant*single_cycle_cos_decay	noam
learning rate {,constant}	0.0013319345874256959	0.0011703123695332683	0.1
learning rate warmup steps	–	4,000	100
dropout	–	–	0.1

Table 2: Summary of hyperparameters that deviate from the defaults.

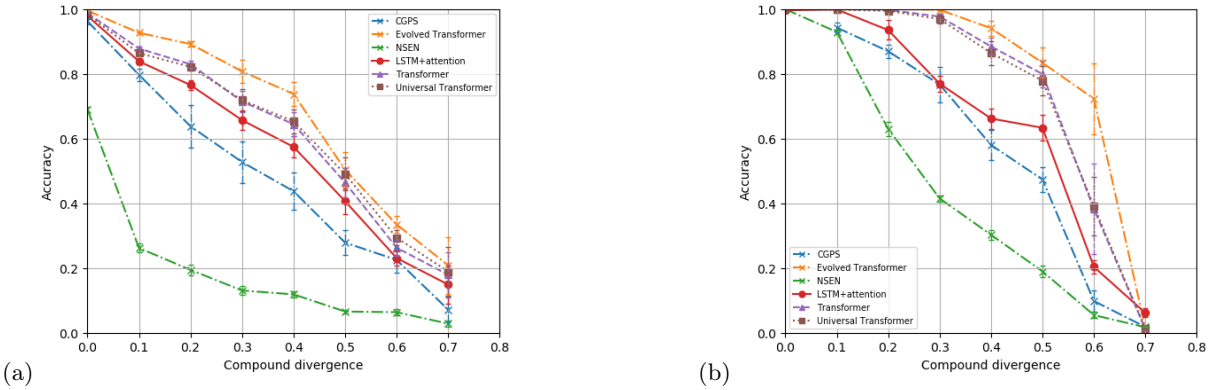

Figure 2: Accuracies of the various models on (a) CFQ and (b) SCAN vs. compound divergence for different split methods and for different target compound divergences.

Model	MCD Mean	MCD1	MCD2	MCD3
LSTM+A	6.1 ±1.7	4.7 ±2.2	7.3 ±2.1	1.8 ±0.7
Transformer	0.9 ±0.3	0.4 ±0.4	1.8 ±0.4	0.5 ±0.1
Univ. Trans.	1.1 ±0.6	0.7 ±1.0	1.5 ±0.2	1.0 ±0.6
Evol. Transf.	1.6 ±0.6	1.4 ±0.2	2.7 ±1.2	0.7 ±0.4
CGPS	2.0 ±0.7	2.6 ±1.2	1.8 ±0.6	1.4 ±0.5
NSEN	1.7 ±0.9	1.2 ±1.0	1.2 ±0.6	2.0 ±1.0
T5-small-NP	0.9 ±0.5	0.2 ±0.3	1.7 ±0.7	0.9 ±0.4
T5-small	6.9 ±1.1	9.5 ±1.2	2.4 ±0.9	8.8 ±1.1
T5-base	**15.4** ±1.1	**26.2** ±1.7	**7.9** ±1.6	12.1 ±0.1
T5-large	10.1 ±1.6	17.7 ±3.4	2.3 ±0.8	10.3 ±0.6
T5-3B	11.6	22.2	3.5	9.2
T5-11B	9.1	7.9	2.4	**16.8**

Table 3: Various models accuracy on SCAN MCD. The variance reported is the 95% confidence interval.

Model	MCD Mean	MCD1	MCD2	MCD3
LSTM+A	14.9 ±1.2	28.9 ±1.8	5.0 ±1.1	10.8 ±0.6
Transformer	17.9 ±0.8	34.9 ±1.1	8.2 ±0.3	10.6 ±1.1
Univ. Trans.	18.9 ±1.4	37.4 ±2.2	8.1 ±1.6	11.3 ±0.3
Evol. Trans.	20.8 ±0.7	42.4 ±1.0	9.3 ±0.8	10.8 ±0.2
CGPS	7.1 ±1.8	13.2 ±3.9	1.6 ±0.8	6.6 ±0.6
NSEN	2.8 ±0.3	5.1 ±0.4	0.9 ±0.1	2.3 ±0.3
T5-small-NP	21.4 ±1.5	42.5 ±2.6	11.2 ±1.5	10.6 ±0.4
T5-small	28.0 ±0.6	54.2 ±0.8	16.0 ±0.3	13.8 ±0.8
T5-base	31.2 ±1.3	57.6 ±1.4	19.5 ±1.0	16.6 ±1.5
T5-large	34.8 ±1.5	63.3 ±0.6	22.2 ±1.5	18.8 ±2.6
T5-3B	40.2 ±4.2	**64.0** ±1.5	29.7 ±2.8	27.0 ±8.3
T5-11B	40.9 ±4.3	61.4 ±4.8	30.1 ±2.2	31.2 ±5.7
T5-11B-mod	**42.1** ±9.1	61.6 ±12.3	**31.3** ±12.8	**33.3** ±2.3

Table 4: Various models accuracy on CFQ MCD. The variance reported is the 95% confidence interval.

```
M0 { directed { M2, M3 } }
M1 { directed { M2, M3 } }
```

Group subjects and objects and sort alphabetically (f_3) The last representation is exactly the same as f_2, but ensuring that the clauses are sorted. (In this particular example, the clauses already happened to be in sorted order, so there would be no effect.)

Figure 3 shows the effect of using different intermediate representations on the compound divergence splits of CFQ for the Transformer. The results indicate that the intermediate representations lead to significant gains for all compound divergence splits. Grouping both subjects and objects leads to the biggest wins, and sorting alphabetically does not seem to affect accuracy significantly.

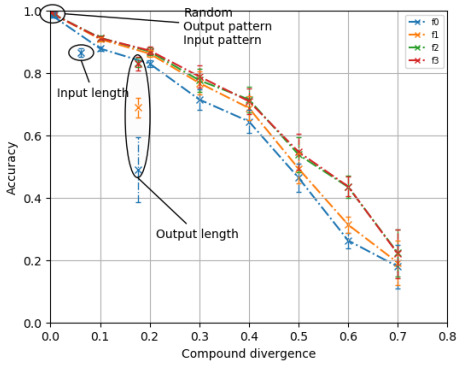

Figure 3: Transformer accuracy on CFQ with intermediate SPARQL representations (see Section E.

F Effect of Pre-training Input and Output Length

We analyze the effect of pre-training further by comparing T5-small-NP (named the "base" model below) with T5-small (the "pre-trained" model). While pre-training is beneficial for many splits, on the length split on SCAN it leads to an average decrease in accuracy of 8.5%. Figure 4 (top) shows a breakdown of the input and output lengths for this split. The test set of this split has input examples of length 8 or 9. As seen in the figure, the relative performance of the pre-trained model compared to the base model largely decreases as the input or output length increases. We provide a similar breakdown for SCAN-MCD3 and for CFQ-MCD1 (Figure 4 middle and bottom). While the pre-trained model outperforms the base model on these two splits, we see that no gains were achieved at sentence lengths beyond the maximum seen in training, at which point accuracy was consistently 0 both with and without pre-training.

G Extended Related Work

This section describes approaches that have been used to encourage compositional generalization or to drive general performance improvements in question answering tasks, but which are not directly applicable to the SCAN or CFQ tasks.

DB content encoders A more extreme approach than that of DB schema encoding is to encode the actual content of the database, so as to attend to some relevant portion of this content when building a representation of the meaning of the questions and when seeking for an answer. This approach is taken by PullNet [Sun et al., 2019], the current state of the art on ComplexWebQuestions [Talmor and Berant, 2018], a benchmark for question answering based on information potentially from web documents in addition to from a knowledge base. Semantic parsing literature typically treats approaches that consider database content as a separate task from approaches that are blind to database content, with the latter approach in some cases considered preferable due to its applicability to situations in which access to DB content is restricted for privacy reasons [Zhong et al., 2017, Xu et al., 2017, Yu et al., 2018a].

Neural module networks A number of benchmarks have been developed with the aim of testing some manner of compositional generalization in the area of visual question answering [Johnson et al., 2017b, Bahdanau et al., 2019a,b, Hudson and Manning, 2019b] and reading comprehension [Dua et al., 2019]. A common approach in both of these spaces is the use of neural module networks [Andreas et al., 2016, Johnson et al., 2017a, Bahdanau et al., 2019b,a, Gupta et al., 2019], which seek to encourage compositional generalization by decomposing the task into re-usable modules which can be combined at evaluation time in ways potentially unseen during training. In its current form, however, this approach depends as one of its components on a semantic parser for mapping the natural language question to a program layout, so is not an alternative for the semantic parsing task itself. In a related line of work, the Neural State Machine [Hudson and Manning, 2019a] also uses a variation of a semantic parser internally to map natural language questions to sequences of reasoning instructions. In both of these cases, the innovations for providing a compositional inductive bias apply primarily to the mechanism by which the answer is retrieved from the relevant data source, rather than to the task of understanding the natural language question itself.

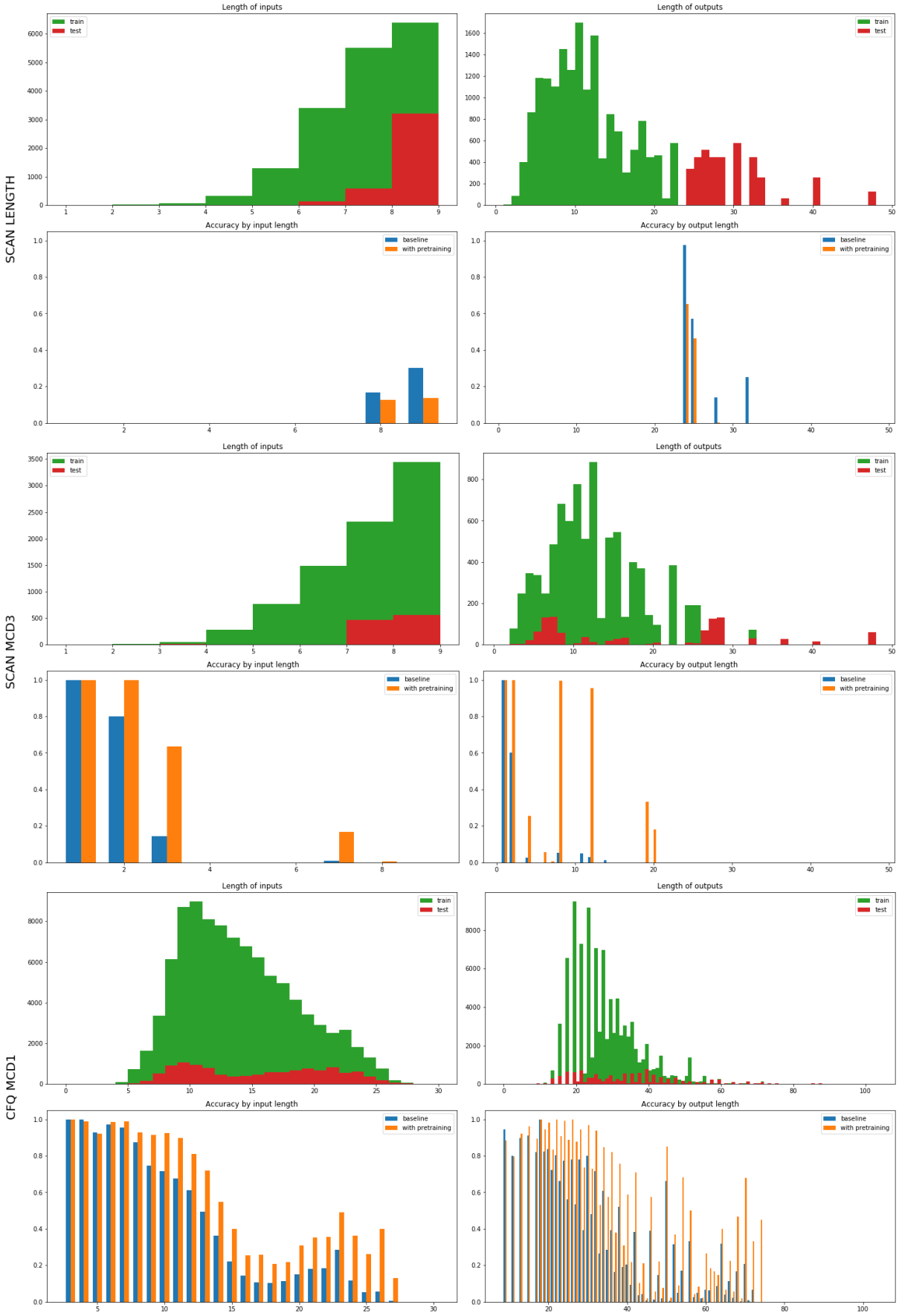

Figure 4: Length analysis of SCAN length, SCAN MCD3 and CFQ MCD1 splits. We show the distribution of train and test samples as well as the performance impact of pre-training broken down by length.