Solving SCAN Tasks with Data Augmentation and Input Embeddings

Michal Auersperger
Charles University
Faculty of Mathematics and Physics
auersperger@ufal.mff.cuni.cz

Pavel Pecina
Charles University
Faculty of Mathematics and Physics
pecina@ufal.mff.cuni.cz

Abstract

We address the compositionality challenge presented by the SCAN benchmark. Using data augmentation and a modification of the standard seq2seq architecture with attention, we achieve SOTA results on all the relevant tasks from the benchmark, showing the models can generalize to words used in unseen contexts. We propose an extension of the benchmark by a harder task, which cannot be solved by the proposed method.

1 Introduction

Compositionality describes the property of language that the syntactic and semantic aspects of complex language units are composed of the syntactic and semantic aspects of primitive units (Fodor and Lepore, 2002). The SCAN benchmark (Lake and Baroni, 2018) has been designed to assess the ability of current neural networks to utilize compositionality of language to deal with systematic difference between the training and test data distributions (see Table 1 for examples).

We use a novel combination of existing ideas (data augmentation, adding noise and predicting outputs based only on the weighted average of input embeddings) to achieve high accuracy in all the relevant tasks of the SCAN benchmark. An approach working well for all the tasks has not been reported yet.[1] Seeing the good results, we analyze some underlying assumptions of the tasks and propose a new split of the SCAN data (i.e. a new task) that proves to be more difficult for our approach.

In Section 2, we describe the SCAN benchmark for testing the systematic use of compositionality in detail. Then we present some approaches tested on the benchmark so far (Section 3), introduce our own method of solving the tasks (Section 4) and its results and analysis of some of the decisions (Section 5). In Section 6, we discuss the existing tasks and propose a new one. We conclude the paper in Section 7.

[1]The model by Lake (2019) achieves very good results, but it relies on seeing test input sequences (tohough with modified output) during training.

2 The tasks (SCAN dataset)

Lake et al. (2017) discussed the differences between human and machine learning and stressed systematic compositionality as an important ingredient. It makes human learning fast and data efficient, compared to slow and data-hungry training of current deep learning models. Neural networks struggle when provided with familiar concepts in new combinations, while people can use known concepts productively.

The SCAN dataset (Lake and Baroni, 2018) was designed to test this specific aspect of learning. It represents the problem of sequence transduction (sequence to sequence transformation). An agent in a grid world environment is supposed to perform a sequence of primitive actions (output) based on a sequence of commands (input). However, neither the agent nor the environment play any role in the generation of the next examples (there is no 'state' as known from reinforcement learning tasks) or in the interpretation of the commands. SCAN is designed to test traditional supervised learning models, namely those that translate from the "command language" into the "action language".

Some examples taken from the dataset are given in Table 1. There are 13 input tokens (*jump, look, run, walk, turn, left, right, and, after, opposite, around, twice, thrice*). These are governed by an underlying non-recursive phrase-structure grammar that produces $20,910$ possible input sequences in total. Each command sequence is translated into an output action sequence. The output vocabulary contains 6 tokens: (JUMP, LOOK, RUN, WALK,

Proceedings of Recent Advances in Natural Language Processing, pages 86–91
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_011

Commands	Actions
(look twice) and (turn right twice)	(LOOK LOOK) (RIGHT RIGHT)
(run thrice) after (look left)	(LEFT LOOK) (RUN RUN RUN)
(walk around right)	(RIGHT WALK RIGHT WALK RIGHT WALK RIGHT WALK)
jump	JUMP
(walk opposite left) after (run)	(RUN) (LEFT LEFT WALK)
...	...
(walk opposite left) after (jump)	(JUMP) (LEFT LEFT WALK)
jump twice	JUMP JUMP
(jump thrice) and (walk)	(JUMP JUMP JUMP) (WALK)
...	...

Table 1: Examples from the SCAN dataset (brackets are added for clarity). Each example consists of a command sequence (input) and a corresponding action sequence (output). Two horizontal lines separate the training (top) from the test (bottom) data. This particular split illustrates the *jump* task, where *jump* appears in the training data in isolation only.

LEFT, RIGHT). For more detail, see the Supplementary in Lake and Baroni (2018).

As far as the training and test data are produced randomly, standard seq2seq models achieve high accuracy of 99.8% (Lake and Baroni, 2018). (The accuracy is computed over whole sequences, i.e. the models make errors in about 0.2% of the testing sequences). Once the training and test data reflect a systematic distribution change, the models struggle. In this work, we address two main sets of tasks presented by SCAN: generalization to a new primitive (JUMP) and generalization to a new combination of learned concepts (AROUND RIGHT). We discuss them in the following subsections.

2.1 Generalization to a new primitive

In this scenario, a primitive command (*jump*) and its corresponding action are excluded from the training data except for the simplest case where the command and action stand in isolation (*jump* $\rightarrow$ JUMP). The test data consist of examples with the primitive in all possible contexts (e.g. *jump twice; walk opposite twice after jump around thrice*). The models need to learn to generalize the information about the contextualized behavior of other primitives (*run twice, walk right*) and apply this information to a new primitive (*jump twice, jump right*).

A simplified version of the task above arises when *turn left* is held out from the training data instead of *jump*. The difference is that the corresponding output action (LEFT) still remains in the training data in different contexts: e.g. (*jump left* $\rightarrow$ LEFT JUMP)

2.2 Generalization to a new combination of learned concepts

Loula et al. (2018) pointed out that while the tasks above are aimed at testing compositionality, there is not much information about the newly added primitives that the models are asked to utilize. We return to this argument later in Section 6. For this reason, a new set of tasks has been added to the benchmark. In these, it is a single combination of well established words that is unique to the test set. There are four such tasks, given here in increasing complexity:

- *jump around right* – unlike the jump task in Section 2.1, here *jump* appears during training in different contexts (e.g. *jump around left; jump right;* ...), the only instances left out from the training contain the *jump around right* sequence
- *Primitive right* – during training, *right* can only follow the two manner adverbs (*around*, *opposite*) but it never follows a verb (*jump, look, run, walk, turn*) directly
- *Primitive opposite right* – the models are asked to infer the meaning of this sequence (i.e. RIGHT RIGHT ACTION) from seeing examples such as (*jump opposite left; look around right, walk right*)
- *Primitive around right* – is analogous to the previous task, the only difference being the complexity (length) of the targeted output sequences (*jump around right* $\rightarrow$ RIGHT JUMP RIGHT JUMP RIGHT JUMP RIGHT JUMP)

3 Existing methods

We describe notable approaches to the SCAN benchmark, selected because of their good performance and/or relevance to our proposed method:

Seq2seq model with the LSTM encoder and decoder and the attention (Bahdanau et al., 2015) mechanism is the baseline method evaluated by Lake and Baroni (2018).

Dessì and Baroni (2019) employ convolutional neural networks (**CNN**) and observe substantial increase in accuracy in the SCAN tasks. This seems to support the conclusion of Bastings et al. (2018), who argue that SCAN tasks inherently prefer simpler models (mainly because of very limited temporal dependencies within the output sequences).

Andreas (2020) employs the baseline recurrent models (Lake and Baroni, 2018) together with a general data augmentation technique (**GECA**) to expand the training data. After the augmentation, the training set contains 5% (JUMP) and 1% (AROUND RIGHT) of the test data, which leads to an increase in performance.

Russin et al. (2020) modify the baseline model by using different attention values: in each decoding step, they keep the computation of the attention weights over the input sequence, but for attention values, the encoder hidden states are replaced with a second set of input word embeddings. (These word embeddings are different from the embeddings used as the encoder input). It is an attempt to separate syntactic (computation of attention weights) and semantic (attention values) information (**SyntAttn**).

Li et al. (2019), in the same vein, separate the flow of information into two streams, which they call primitive (i.e. semantic) and functional (i.e. syntactic). Again, syntactic embeddings are used to determine the attention weights over the input sequence and semantic embeddings are used as the actual values the attention mechanism produces. Moreover, the approach uses regularization (adding noise to the embeddings, L_2 norm) leading to more stable training (and better results) compared to the previous method (**Li19**).

Finally, Lake (2019) uses a seq2seq model with the combination of external memory, data augmentation and meta-learning and achieves good performance on multiple SCAN tasks (**MetaSeq2seq**).

4 Model

Our architecture is a straightforward extension of the baseline LSTM seq2seq model with attention (Lake and Baroni, 2018). We modify the attention mechanism by using input word embeddings as the attention values (as opposed to using contextualized representations produced by the encoder). This can also be seen as a simplified version of the architectures introduced by Li et al. (2019); Russin et al. (2020). Unlike them, we train only one set of embeddings that are used both as the encoder input and as attention values. Also, we use the traditional autoregressive decoder (the previous predicted output serves as the input for the decoder at each step), which is not the case in the two cited approaches.

We added standard Gaussian noise to the input embeddings during training, as suggested by Li et al. (2019), but did not use the L2 regularization.

We also experimented with weight tying, known for example, from some language modelling literature (Inan et al., 2017). We tried to avoid using a separate output layer and instead, we computed similarity (dot product) of the attention output with the output token embeddings (the embeddings used by the decoder) and used these vectors as logits.

Similarly to Lake (2019), we experiment with adding artificial primitives to the training data. We hope this would make the encoder more robust as it should recognize the syntactic patterns (X *opposite right*) rather than memorize each instance (*walk opposite right*) using the available embedding (*walk*).

For the *jump* task, we introduce artificial action commands *action1, action2, action3, ...* with the corresponding actions ACTION1, ACTION2, ACTION3, ... For all the other tasks (*lturn, jump around right, Primitive right, Primitive opposite right, Primitive around right*), we introduce artificial directions: *dir1* $\rightarrow$ DIR1, *dir2* $\rightarrow$ DIR2, *dir3* $\rightarrow$ DIR3 .... Unlike Lake (2019), we are not permuting the command – action assignment during training and we do not introduce the held-out phrases to either the input or the output of the training examples.

4.1 Training details

We trained all the models using the Adam optimizer with the learning rate 0.001 exponentially decaying to 0.0001 over 60 epochs (regardless of the task). The batch size was set to 32. We experimented with different sizes of the input and output embeddings and LSTM dimensions (64, 128, 256, 512), and the

Method	jump	around right
seq2seq	1.2	2.46 ± 2.68
CNN	69.2 ± 8.2	56.7 ± 10.2
GECA	87 ± 2	82 ± 4
SyntAttn	91.0 ± 27.4	28.9 ± 34.8
Li19	98.8 ± 1.4	83.2 ± 13.2
MetaSeq2seq	99.95 ± 0.08	99.96 ± 0.08
ours	98.90 ± 1.40	99.82 ± 0.47

Table 2: Test accuracy (mean ± std %) on the two most challenging SCAN tasks. Results are given as reported by their authors. Seq2seq is from Lake and Baroni (2018) for the jump and Loula et al. (2018) for the around right task, CNN from Dessì and Baroni (2019), GECA from Andreas (2020), SyntAttn from Russin et al. (2020), Li19 from Li et al. (2019) and MetaSeq2seq from Lake (2019). Results are reported over 5 random seeds with the exception of GECA (10 seeds), SyntAttn (median value, 25 seeds) and ours (25 seeds).

encoder layers (1, 2). The decoder was initialized by the final hidden state of the encoder. We did not employ dropout and we clipped gradients whose norm was larger than 1. Teacher forcing was used during training. In preliminary experiments, we settled on the model with one layer bidirectional LSTM encoder, embedding size of 256 and LSTM dimension of 256. We experimented with adding 10, 50, 75 and 100 artificial command – action pairs.

5 Results and analysis

We provide the test accuracy of our model on the two most widely discussed SCAN tasks (*jump* and *around right*) in Table 2. For completeness, the results for the remaining tasks are given in Table 3. The reported model was trained using data augmentation with 75 additional command – action pairs, Gaussian noise added to the input embeddings during training and no parameter tying. Below, we investigate the effect of these choices. If not stated otherwise, we always report a mean and standard deviation computed over 25 runs with different random seeds.

5.1 Data augmentation

Arguably, the biggest benefit comes from the data augmentation. We illustrate this in Table 4, where we compare the baseline model trained with and without data augmentation on all the discussed SCAN tasks. With the exception of the *jump* and *around right* tasks, data augmentation seems to be enough to achieve good results.

5.2 Parameter tying

The effect of replacing a separate output layer with a dot product with decoder embeddings was most prominent with less intensive data augmentation. When adding only 10 additional command – action pairs to the training data of the *around right* task, our top model achieved the mean accuracy of 59.47% (±18.27%) with the output layer and 90.4% (±7.5%) without it. The effect disappeared with heavier data augmentation and/or with other tasks. Therefore, parameter tying was not included in the overall results.

5.3 Input embeddings as attention values

In contrast, the other architectural change with respect to the seq2seq baseline, i.e. using the input embeddings as attention values, had an effect even with stronger data augmentation. Models with modified attention trained with 100 additional command – action pairs achieved the mean accuracy of 99.44% (±0.61%) on the *jump* task, compared to 68.51% (±10.67%) achieved by corresponding models, where the attention values were formed by the encoder output.

5.4 Embedding noise

Adding standard Gaussian noise to the input embeddings plays also an important role by making the training more stable. Our overall best model without the noise added in training achieves only 89.12% (±12.91%) accuracy on the *jump* task (compared to 98.9% (±1.4%) by the same model with the noise).

5.5 Single jump proportion

During training on the *jump* task, the models were sensitive to the proportion of isolated *jump* examples in the training data. In the original dataset, such examples are upsampled so that they form about 10% of the training data (without this, models would encounter *jump* only once each episode). We found the best results, when these examples were treated as all the other ones, i.e. they were also eligible for command replacement with uniform probability.

6 Discussion

We show that task-specific data augmentation with simple architecture modification leads to good re-

Method	turn left	jump ar. right	right	opposite right
seq2seq	90.3	98.43 ± 0.54	23.49 ± 8.09	47.62 ± 17.72
SyntAttn (Russin et al., 2020)	99.9 ± 0.16	98.9 ± 2.3	99.1 ± 1.8	10.5 ± 8.8
Li et al. (2019)	99.7 ± 0.4	100.0 ± 0.0	99.7 ± 0.5	89.3 ± 5.5
ours	99.99 ± 0.02	99.98 ± 0.05	99.99 ± 0.04	99.98 ± 0.04

Table 3: Test accuracy (mean ± std %) on the remaining SCAN tasks. The baseline (seq2seq) results were reported by Lake and Baroni (2018) for the turn left task and by Loula et al. (2018) for the other tasks. Results are reported over 5 random seeds with the exception of ours (25 seeds).

Task	extra0	extra100
jump	0.11 ± 0.11	57.92 ± 15.91
around r.	0.0 ± 0.0	98.69 ± 1.33
turn l.	49.55 ± 8.89	99.92 ± 0.24
jump ar. r.	82.97 ± 11.14	99.62 ± 0.96
right	3.18 ± 2.3	99.90 ± 0.2
opposite r	0.04 ± 0.07	99.97 ± 0.04

Table 4: The effect of data augmentation: the same baseline seq2seq model trained on the original data (extra0) and with 100 additional command – action pairs (extra100). Test accuracy (mean ± std %, 25 seeds)

sults on tasks from the SCAN benchmark.

We acknowledge that adding tens of additional command–action pairs to the training data might seem problematic. At the same time we argue that, fundamentally, the tasks remain unchanged and the models still need to generalize to a new primitive (*jump*) or combine two concepts in a new way (*around right*). Moreover, we do not present our work to claim superiority over previous research but mainly to shed some light on how compositionality is represented by the SCAN benchmark.

As seen above (Table 4, *around right* task) the baseline seq2seq architecture struggles when asked to cope with examples such as *look around right* if it was only trained on *look right, look left, look around left,* and the like. However, when provided with more diverse training data, it succeeds in the vast majority of instances. Does this mean the network mastered compositionality and is ready to combine the information productively? Or does it mean that the modified task no longer tests for compositionality?

We hypothesize that current methods (including ours) manage to solve this compositionality challenge without using much compositionality. Specifically, what is enough for all the tasks as they have been defined is to learn to attend to correct positions and then use a one to one mapping from the input to the output space. This "align and translate" (Bahdanau et al., 2015) approach exhibits some degree of compositionality. However, having learned somewhat complex information (say in the case of grammatical words such as *around* or *opposite*), this needs to be combined with just the identity of another word. This was actually the motivation of Loula et al. (2018) to introduce new tasks into the benchmark (e.g. *around right*, see Section 2.2). We believe that in this respect, the new tasks suffer from the same weaknesses as the old ones (as evidenced by the possibility of easily using data augmentation).

6.1 Proposed task

Based on the presented results and the discussion above, we propose another task for the SCAN benchmark: *around Direction twice*. [2]

Here, we form the test set by selecting all the SCAN examples where the input string contains the sequence *around right/left twice*. Ideally, a model should be able to combine the information about doing something while turning around and doing something twice based on such examples as *jump around left, jump around right thrice, jump twice, jump opposite right twice*.

At face value, this task is almost identical to the tasks discussed in Section 2.2 (e.g. *around right*): we held out a certain combination of well established words from the training and tested on them. What we believe is the substantial difference is the fact that neither *around* nor *twice* have a direct 'translation' counterpart in the output 'language'. This makes the proposed task less suitable for the above-mentioned "align and translate" approach.

Another practical effect of this is that it is not very obvious how data augmentation could be used. We would need to generate alternatives to either *twice*, which would quickly lead to uncomfortably long output sequences (e.g. *run around right thirty-*

[2]The data are available at https://github.com/michal-au/scan-around-twice

seven times) and/or *around*, which seems even more obscure.

We trained 3 types of models for the new task: the baseline seq2seq architecture without modifications; our seq2seq model with added noise and modified attention values; the same model with 100 additional augmented directions (e.g. *run around dir3 thrice*). All the models failed with results: $10.05\%(\pm 2.30\%)$, $8.78\%(\pm 2.48\%)$ and $5.53\%(\pm 2.12\%)$ respectively (each model was trained with 5 different random seeds).

7 Conclusion

The SCAN benchmark is a very accessible dataset used for investigating compositionality. Cherry-picking from previous research, we introduced the first approach that is successful in all the relevant tasks from the benchmark. Analyzing the important features defining our approach, we introduced a new task, where this approach fails. We hypothesize the new task is more challenging than the existing ones since it tests for deeper aspects of compositionality.

Acknowledgments

The work was supported by the grant 978119 of the Grant Agency of Charles University in Prague (GAUK) and by the grant 19-26934X of the Czech National Science Foundation (GACR).

References

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

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.

Jerry A. Fodor and Ernest Lepore. 2002. *Compositionality Papers*. Oxford University Press UK.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882, Stockholmsmässan, Stockholm Sweden. PMLR.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 9791–9801. Curran Associates, Inc.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253.

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.

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.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. Association for Computational Linguistics.