Jump to better conclusions: SCAN both left and right

Joost Bastings[1] **Marco Baroni**[2] **Jason Weston**[2] **Kyunghyun Cho**[2,3,4] **Douwe Kiela**[2]
[1]ILLC, University of Amsterdam
[2]Facebook AI Research
[3]New York University
[4]CIFAR Global Scholar
bastings@uva.nl {mbaroni,jase,kyunghyuncho,dkiela}@fb.com

Abstract

Lake and Baroni (2018) recently introduced the SCAN data set, which consists of simple commands paired with action sequences and is intended to test the strong generalization abilities of recurrent sequence-to-sequence models. Their initial experiments suggested that such models may fail because they lack the ability to extract systematic rules. Here, we take a closer look at SCAN and show that it does not always capture the kind of generalization that it was designed for. To mitigate this we propose a complementary dataset, which requires mapping actions back to the original commands, called NACS. We show that models that do well on SCAN do not necessarily do well on NACS, and that NACS exhibits properties more closely aligned with realistic use-cases for sequence-to-sequence models.

1 Introduction

In a recent paper, Lake and Baroni (2018) (L&B) investigate if recurrent sequence-to-sequence models can exhibit the same strong generalization that humans are capable of, by virtue of our capacity to infer the meaning of a phrase from its constituent parts (i.e., compositionality), providing empirical tests for this long-standing goal (Fodor and Pylyshyn, 1988). Compositional generalization might be a fundamental component in making models drastically less sample-thirsty than they currently are. L&B introduce the SCAN data set (§2), meant to study such generalization to novel examples. It consists of simple command-action pairs, in which more complex commands are composed of simpler ones (see Figure 1 for examples).

SCAN comprises several tests of generalization, namely with respect to (1) a random subset of the data ('simple'), (2) commands with action sequences longer than those seen during training ('length'), and (3) commands that compose a primitive in novel ways that was only seen in isolation during training ('primitive'). In the latter case, the training set would for example only include the command 'jump', after which the test set includes all other commands containing 'jump', e.g. 'jump opposite left after walk twice'.

jump
`JUMP`

turn around left
`LTURN LTURN LTURN LTURN`

jump thrice and turn left twice
`JUMP JUMP JUMP LTURN LTURN`

jump opposite left after walk twice
`WALK WALK LTURN LTURN JUMP`

Figure 1: SCAN maps commands to actions

In this paper we take a closer look at SCAN. We start with the observation (§3) that there are few target-side dependencies in the data, a consequence of SCAN being generated from a phrase-structure grammar. We show (§6) that this allows simple sequence-to-sequence models (§5) to obtain good accuracies e.g. on tasks involving a new primitive, even without access to previous outputs. However, these simple models do not use composition in any interesting way, and their performance is therefore not a realistic indicator of their generalization capability. We hence propose NACS (§4) as a more realistic alternative: SCAN with commands and actions flipped, i.e., mapping actions back to their original commands. This is harder, because different commands may map to the same action sequence, and it introduces target-side dependencies, so that previous outputs need to be remembered. We show in particular that well-tuned attention-based models do achieve a certain degree of generalization on SCAN, and, as predicted, simpler models do better there. However, the models still struggle in the more demanding NACS setup, which we offer as a challenge for future work.

Our contributions can be summarized as follows:

1. we provide an analysis of SCAN and make the important observation that it does not test for target-side dependencies, allowing too simple models to do well;
2. we propose NACS to introduce target-side dependencies and remedy the problem;
3. we repeat all experiments in Lake and Baroni (2018) using early-stopping on validation sets created from the training data.

2 SCAN

SCAN stands for Simplified version of the CommAI Navigation tasks (Mikolov et al., 2016). Each example in SCAN is constructed by first sampling a command $X = (x_1, \ldots, x_T)$ from a finite phrase-structure grammar with start symbol C:

$$
\begin{aligned}
&C \rightarrow S \text{ and } S \mid S \text{ after } S \mid S\\
&S \rightarrow V \text{ twice } \mid V \text{ thrice } \mid V\\
&V \rightarrow D_{[1]} \text{ opposite } D_{[2]} \mid D_{[1]} \text{ around } D_{[2]} \mid D \mid U\\
&D \rightarrow U \text{ left } \mid U \text{ right } \mid \text{ turn left } \mid \text{ turn right}\\
&U \rightarrow \text{ walk } \mid \text{ look } \mid \text{ run } \mid \text{ jump}
\end{aligned}
$$

For each command, the corresponding target action sequence $Y = (y_1, \ldots, y_{T'})$ then follows by applying a set of *interpretation functions*, such as:

$$
\begin{aligned}
[\![\text{jump}]\!] &= \text{JUMP}\\
[\![u \text{ around left }]\!] &= \text{LTURN } [\![u]\!] \text{ LTURN } [\![u]\!]\\
&\quad\ \text{LTURN } [\![u]\!] \text{ LTURN } [\![u]\!]\\
[\![x_1 \text{ after } x_2]\!] &= [\![x_2]\!]\ [\![x_1]\!]
\end{aligned}
$$

of which only the last function requires global reordering, which occurs at most once per command. See the supplementary materials for the full set. Figure 1 shows examples of commands and their action sequences as obtained by the interpretation functions. The commands can be decoded compositionally by a learner by discovering the interpretation functions, enabling generalization to unseen commands. The total data set is finite but large (20910 unambiguous commands).

3 SCAN prefers simple models

We observe an important property of the data set generation process for SCAN: temporal dependencies of the action sequence are limited to the phrasal boundaries of each sub-phrase, which span at most 24 actions (e.g. jump around left thrice). Crucially, even rules that require repetition (such as 'thrice') as well as global reordering, can be resolved by simple counting and without remembering previously generated outputs, due to the limited depth of the phrase-structure grammar (see e.g. Rodriguez and Wiles (1998)).

This observation has two important implications. First, because SCAN is largely a phrase-to-phrase conversion problem, any machine learning method that aims at solving SCAN needs to have an *alignment mechanism* between the source and target sequences. Such an alignment mechanism could work fairly accurately by simply advancing a pointer. Somewhat contrary to the observation by Lake and Baroni (L&B), we therefore hypothesize that an attention mechanism (Bahdanau et al., 2015) always helps when a neural conditional sequence model (Sutskever et al., 2014; Cho et al., 2014) is used to tackle any variant of SCAN. Second, we speculate that any algorithm with strong long-term dependency modeling capabilities can be *detrimental* in terms of generalization, because such an approach might inappropriately capture spurious target-side regularities in the training data. We thus hypothesize that *less powerful decoders generalize better* on to unseen action combinations on SCAN when equipped with an attention mechanism.

To summarize: good performance on SCAN does not necessarily indicate the capability of a model to strongly generalize. SCAN favors simpler models that need not capture target-side dependencies, which might not work well on more realistic sequence-to-sequence problems, such as machine translation, where strong auto-regressive models are needed for good results (Bahdanau et al., 2015; Kaiser and Bengio, 2016).

4 NACS: actions to commands

By simply flipping the source X and target Y of each example, we obtain a data-set that suddenly features strong target-side dependencies. Even when the mapping $p(Y|X)$ from the source to target is simple, the opposite $p(X|Y) \propto p(Y|X)p(X)$ is non-trivial due to the complexity

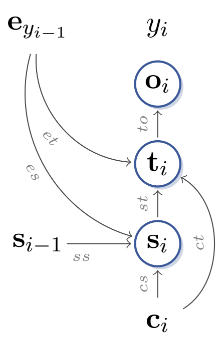

Figure 2: The decoder of Bahdanau et al. (2015)

of the prior $p(X)$. The inclusion of $p(X)$ naturally induces strong dependencies among the output tokens, while maintaining the original properties of SCAN that were intended to test various aspects of systematic generalization.

NACS naturally makes the mapping that needs to be learned stochastic and multi-modal (sensitive to both commands and actions). For instance, an action sequence of the form $[\![x_1]\!][\![x_2]\!]$ could be mapped to either $[\![x_1$ and $x_2]\!]$ or $[\![x_2$ after $x_1]\!]$, both of which are correct. In order for a model to decide whether to output "and" or "after", it is necessary for it to remember what has already been generated (i.e., $[\![x_1]\!]$ or $[\![x_2]\!]$).

Another example is `LTURN LTURN LTURN LTURN`, which can be translated into either "turn around left" or two repetitions of "turn opposite left". Deciding whether to output "and" after the first phrase requires the model to remember whether "around" was generated previously.

In §6 we experimentally evaluate the proposed NACS task using the same scenarios as SCAN (simple, length and primitive). We observe that NACS prefers more advanced models that could capture long-term dependencies in the output (now a command sequence) better. However, we notice that even these powerful models, equipped with GRUs and attention, cannot systematically generalize to this task, as was also observed by Lake and Baroni (2018). Based on this observation, we believe that NACS (or perhaps a combination of SCAN and NACS) is better suited for evaluating any future progress in this direction.

5 Sequence-to-sequence models

In this section, we describe the sequence-to-sequence models we use for evaluating on SCAN and its proposed sibling NACS.

We directly model the probability of a target sequence given a source sequence $p(Y|X)$. Our encoder-decoder is modeled after Cho et al. (2014) and our attention-based encoder-decoder after Bahdanau et al. (2015). The attention-based decoder is a function that takes as input the previous target word embedding $\mathbf{e}_{y_{i-1}}$, the context vector $\mathbf{c}_i$, and the previous hidden state $\mathbf{s}_{i-1}$ (see also Figure 2): $\mathbf{s}_i = f(\mathbf{e}_{y_{i-1}}, \mathbf{c}_i, \mathbf{s}_{i-1})$.

The prediction for the current time step is then made from a pre-output layer $\mathbf{t}_i$: $\mathbf{t}_i = W_e\mathbf{e}_{y_{i-1}} + W_c\mathbf{c}_i + W_s\mathbf{s}_i$. We do not apply a max-out layer and directly obtain the output by $\mathbf{o}_i = W_o\mathbf{t}_i$. For the encoder-decoder without attention, the prediction is made directly from decoder state $\mathbf{s}_i$. We vary the recurrent cell, experimenting with simple RNN (Elman, 1990), GRU (Cho et al., 2014), and LSTM cells (Hochreiter and Schmidhuber, 1997). For conciseness we only report results with RNN and GRU cells in the main text, and LSTM results in the appendix.

In this paper, we investigate the properties of both SCAN and NACS using RNN-based sequence-to-sequence models for evaluation. We leave further investigation of alternative architectures (see, e.g., Vaswani et al., 2017; Gehring et al., 2017; Chen et al., 2018) for the future.

6 Experiments

6.1 Settings

Our models are implemented in PyTorch and trained using mini-batch SGD with an initial learning rate of 0.2, decayed by 0.96 each epoch. We use a batch size of 32, 256 hidden units (64 for embeddings), and a dropout rate of 0.2. We test on all SCAN/NACS tasks[1], as well as on the Fr-En Machine Translation (MT) task that L&B used. The reported results are averaged over three runs for each experiment. Models *with* attention are marked as such with *+Attn*, e.g. 'GRU +Attn'.

Validation Set. L&B split each SCAN subtask into a training set (80%) and a test set (20%). They train for a fixed number of updates (100k) and evaluate on the test set. Because any training run without early stopping may have missed the optimal solution (Caruana et al., 2001), we believe their results may not reflect the reality as closely as they could. We thus augment each of the SCAN variants with a validation set that follows the training distribution but contains examples that are not contained in the corresponding training set. This

[1]`github.com/facebookresearch/NACS`

	Simple		Length		Turn left		Jump	
	SCAN	NACS	SCAN	NACS	SCAN	NACS	SCAN	NACS
GRU	100.0 ±0.0	99.0 ±0.1	14.4 ±0.8	12.9 ±1.2	53.4 ±11.7	47.5 ±4.7	0.0 ±0.0	0.0 ±0.0
RNN +Attn	100.0 ±0.0	99.8 ±0.1	9.6 ±0.9	19.4 ±0.7	81.1 ±14.7	44.1 ±0.9	1.9 ±1.2	0.3 ±0.3
RNN +Attn -Dep	100.0 ±0.0	61.1 ±0.3	11.7 ±3.2	0.5 ±0.2	92.0 ±5.8	18.6 ±1.0	2.7 ±1.7	0.0 ±0.0
GRU +Attn	100.0 ±0.0	99.8 ±0.1	18.1 ±1.1	17.2 ±1.9	59.1 ±16.8	55.9 ±3.5	12.5 ±6.6	0.0 ±0.0
GRU +Attn -Dep	100.0 ±0.0	51.2 ±1.2	17.8 ±1.7	2.0 ±1.4	90.8 ±3.6	16.9 ±1.2	0.7 ±0.4	0.0 ±0.0
L&B best	99.8	-	20.8	-	90.3	-	1.2	-
L&B best overall	99.7	-	13.8	-	90.0	-	0.1	-

Table 1: Test scores on the simple, length, and primitive (turn left, jump) tasks. *+Attn* marks attention, *-Dep* has the connections from the previous target word embedding removed (es and et in Figure 2). L&B$_{\text{best}}$ is the best reported score for each task by L&B, and L&B$_{\text{best overall}}$ is the score for their best-scoring model all tasks considered.

allows us to incorporate early stopping in our experiments so that they are better benchmarks for evaluating future progress. For each experiment we remove 10% of the training examples to be used as a validation set.

Accuracy. Following L&B we measure performance according to sequence-level accuracy, i.e., whether the generated sequence entirely matches the reference. This metric is also used for early stopping. For NACS, an output (command) is considered correct if its interpretation ('backmapping') produces the input action sequence.

Ablations. To validate our analysis, we remove the connections from the previous target word embedding $\mathbf{e}_{y_{i-1}}$ to the decoder state and the pre-output layer (es and et in Figure 2), so that the current prediction is not informed by previous outputs. If our analysis in §3 is correct, then these simpler models should still be able to make the correct predictions on SCAN, but not on NACS.

6.2 Results and Analysis

Results on the three SCAN and NACS tasks are listed in Table 1. The full results including models with LSTM cells and MT experiments may be found in the supplementary materials. We will now discuss our observations.

SCAN is not enough. Table 1 shows that all model variants perform (near) perfectly on the SCAN simple task. While this is impressive, results for the severed models (+Attn -Dep) on the simple task for NACS show that it is possible to have a perfect accuracy on SCAN, while at the same time failing to do well on NACS.[2] Crucially, a (near) perfect score on SCAN does not imply strong generalization. A model can exploit the determinism and lack of target-side dependencies of SCAN by developing a simple translation strategy such as advancing a pointer and translating word by word, and the use of such a simple strategy is not revealed by SCAN.

NACS is harder. NACS is a harder problem to solve compared to SCAN, as evidenced by consistently lower accuracies in Table 1 for all tasks. The discrepancy between SCAN and NACS performance is the most extreme when we look at the primitive tasks (turn left and jump). For turn left, the severed models (+Attn -Dep) obtain the highest scores on SCAN, but are the worst on NACS.

The 'turn left' task benefits from `TURNL` occurring on the target-side in other contexts during training, which is not the case for 'jump'.[3] Since there is no evidence in the training data that 'jump' is a verb, Table 2 shows results where additional (composed) 'jump' commands were added for training. We see that performance quickly goes up when adding more commands.[4] Again here the simpler models (+Attn -Dep) perform better.

Machine Translation. We repeat L&Bs English-French MT experiment for both directions. Table 3 shows that models that perform well on NACS also perform well here, with the GRU outperforming the other cells (see appendix

[2]We made similar observations using LSTM cells, as we show in the appendix.

[3]See Lake and Baroni (2018) for a discussion.

[4]L&B performed this experiment without attention, which we show has a large positive impact.

		1	2	4	8	16	32
RNN +Attn	SCAN	35.0 ±2.8	48.6 ±8.1	77.6 ±2.6	89.2 ±3.8	98.7 ±1.3	99.8 ±0.1
RNN +Attn -Dep	SCAN	29.5 ±10.5	53.3 ±10.2	82.4 ±4.7	98.8 ±0.8	99.8 ±0.1	100.0 ±0.0
GRU +Attn	SCAN	58.2 ±12.0	67.8 ±3.4	80.3 ±7.0	88.0 ±6.0	98.3 ±1.8	99.6 ±0.2
GRU +Attn -Dep	SCAN	70.9 ±11.5	61.3 ±13.5	83.5 ±6.1	99.0 ±0.4	99.7 ±0.2	100.0 ±0.0
RNN +Attn	NACS	2.8 ±0.8	9.3 ±7.3	24.7 ±4.2	43.7 ±4.4	57.1 ±5.2	69.1 ±2.1
RNN +Attn -Dep	NACS	0.4 ±0.1	0.9 ±0.2	2.4 ±0.3	3.9 ±0.3	9.3 ±0.3	15.9 ±1.4
GRU +Attn	NACS	5.5 ±1.8	9.2 ±2.8	11.0 ±1.5	21.9 ±2.4	23.5 ±0.6	42.0 ±1.5
GRU +Attn -Dep	NACS	0.1 ±0.1	0.6 ±0.2	2.0 ±0.2	3.2 ±0.2	5.8 ±1.1	10.9 ±0.8
L&B	SCAN	0.1	0.1	4.1	15.3	70.2	89.9

Table 2: Test scores on the 'jump' task with additional commands. +Attn marks attention, -Dep has the es and et connections removed (Figure 2). The test set contains all jump commands except the 32 used for training. Columns indicate how many commands with 'jump' were added to the training set, such as 'jump around left thrice.'

	En-Fr	Fr-En
GRU +Attn	32.1 ±0.3	37.5 ±0.6
GRU +Attn -Dep	30.2 ±0.3	35.9 ±0.3

Table 3: Results (BLEU) on the Machine Translation experiment for both directions using a GRU. See appendix for results using SRN and LSTM cells.

for other cell types). In a setting similar to the jump task, the sentence pair 'I am daxy' ('je suis daxiste') was added to the training set. The goal is now to test if eight novel sentences that contain 'daxy' are correctly translated.

In our setting with mini-batching and early-stopping, the GRU gets 70.8% (En-Fr) and 54.2% (Fr-En) of the daxy-sentences right, which is surprisingly good compared to L&B (12.5%).

Other observations. As expected, Table 1 shows that attention always helps. Generalizing to longer sequences is generally hard, and this remains an open problem.

7 Related Work

Ever since Fodor and Pylyshyn (1988) conjectured that neural networks are unable to show strong generalization, many attempts were made to show that the opposite is true, leading to inconclusive evidence. For example, Phillips (1998) found that feed-forward nets and RNNs do not always generalize to novel 2-tuples on an auto-association task, while Wong and Wang (2007) and Brakel and Frank (2009) found that RNNs can show systematic behavior in a language modeling task.

In the context of analyzing RNNs, Rodriguez and Wiles (1998) found that simple RNNs can develop a symbol-sensitive counting strategy for accepting a simple (palindrome) context-free language. Weiss et al. (2018) show that LSTMs and simple RNNs with ReLU-activation can learn to count unboundedly, in contrast to GRUs.

Linzen et al. (2016) probed the sensitivity of LSTMs to hierarchical structure (not necessarily in novel constructions). Instead of a binary choice, with SCAN a sequence-to-sequence model productively generates an output string.

Liska et al. (2018) found that a small number of identical RNNs trained with different initializations show compositional behavior on a function composition task, suggesting that more specific architectures may not be necessary.

Finally, Lake and Baroni (2018) introduced the SCAN data set to study systematic compositionality in recurrent sequence-to-sequence models, including gating mechanisms and attention. This work is a direct response to that and aims to facilitate future progress by showing that SCAN does not necessarily test for strong generalization.

8 Conclusion

In the quest for strong generalization, benchmarks measuring progress are an important component. The existing SCAN benchmark allows too simple models to shine, without the need for compositional generalization. We proposed NACS to remedy this. NACS still requires systematicity, while introducing stochasticity and strong dependencies on the target side. We argue that a good benchmark needs at least those properties, in order not to fall prey to trivial solutions, which do not work on more realistic use-cases for sequence-to-sequence models such as machine translation.

Acknowledgments

We would like to thank Brenden Lake and Marc'Aurelio Ranzato for useful discussions and feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA.

Philémon Brakel and Stefan Frank. 2009. Strong systematicity in sentence processing by simple recurrent networks. In *31th Annual Conference of the Cognitive Science Society (COGSCI-2009)*, pages 1599–1604. Cognitive Science Society.

Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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

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

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Łukasz Kaiser and Samy Bengio. 2016. Can active memory replace attention? In *Advances in Neural Information Processing Systems*, pages 3781–3789.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *International Conference on Machine Learning (ICML)*.

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.

Adam Liska, Germán Kruszewski, and Marco Baroni. 2018. Memorize or generalize? searching for a compositional RNN in a haystack. *CoRR*, abs/1802.06467.

Tomas Mikolov, Armand Joulin, and Marco Baroni. 2016. A roadmap towards machine intelligence. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 29–61. Springer.

Steven Phillips. 1998. Are feedforward and recurrent networks systematic? analysis and implications for a connectionist cognitive architecture. *Connection Science*, 10(2):137–160.

Paul Rodriguez and Janet Wiles. 1998. Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in Neural Information Processing Systems*, pages 87–93.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

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)*, Melbourne, Australia. Association for Computational Linguistics.

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

Supplementary Materials

⟦walk ⟧ = WALK
⟦look⟧ = LOOK
⟦run⟧ = RUN
⟦jump⟧ = JUMP

⟦turn left⟧ = LTURN
⟦turn right⟧ = RTURN

⟦u left⟧ = LTURN ⟦u⟧
⟦u right⟧ = RTURN ⟦u⟧

⟦x twice⟧ = ⟦x⟧ ⟦x⟧
⟦x thrice⟧ = ⟦x⟧ ⟦x⟧ ⟦x⟧

⟦turn around left⟧ = LTURN LTURN LTURN LTURN
⟦turn around right⟧ = RTURN RTURN RTURN RTURN
⟦u around left⟧ = LTURN ⟦u⟧ LTURN ⟦u⟧ LTURN ⟦u⟧ LTURN ⟦u⟧
⟦u around right⟧ = RTURN ⟦u⟧ RTURN ⟦u⟧ RTURN ⟦u⟧ RTURN ⟦u⟧

⟦turn opposite left⟧ = LTURN LTURN
⟦turn opposite right⟧ = RTURN RTURN
⟦u opposite left⟧ = ⟦turn opposite left⟧ ⟦u⟧
⟦u opposite right⟧ = ⟦turn opposite right⟧ ⟦u⟧

⟦x_1 and x_2⟧ = ⟦x_1⟧ ⟦x_2⟧
⟦x_1 after x_2⟧ = ⟦x_2⟧ ⟦x_1⟧

Figure 3: The interpretation functions for translating SCAN commands to actions.

	Simple	Length	Turn left	Jump
RNN	75.6 $\pm$5.4	0.2 $\pm$0.0	26.7 $\pm$12.8	0.0 $\pm$0.0
GRU	100.0 $\pm$0.0	14.4 $\pm$0.8	53.4 $\pm$11.7	0.0 $\pm$0.0
LSTM	99.8 $\pm$0.1	10.1 $\pm$2.0	56.5 $\pm$0.8	0.1 $\pm$0.0
RNN $_{\text{+Attn}}$	100.0 $\pm$0.0	9.6 $\pm$0.9	81.1 $\pm$14.7	1.9 $\pm$1.2
RNN $_{\text{+Attn-Dep}}$	100.0 $\pm$0.0	11.7 $\pm$3.2	92.0 $\pm$5.8	2.7 $\pm$1.7
GRU $_{\text{+Attn}}$	100.0 $\pm$0.0	18.1 $\pm$1.1	59.1 $\pm$16.8	12.5 $\pm$6.6
GRU $_{\text{+Attn-Dep}}$	100.0 $\pm$0.0	17.8 $\pm$1.7	90.8 $\pm$3.6	0.7 $\pm$0.4
LSTM $_{\text{+Attn}}$	100.0 $\pm$0.0	15.6 $\pm$1.6	83.8 $\pm$16.8	9.7 $\pm$2.9
LSTM $_{\text{+Attn-Dep}}$	100.0 $\pm$0.0	12.5 $\pm$1.3	57.6 $\pm$3.8	0.8 $\pm$0.5
L&B $_{\text{best}}$	99.8	20.8	90.3	1.2
L&B $_{\text{best overall}}$	99.7	13.8	90.0	0.1

Table 4: SCAN test scores on the simple, length, and primitive (turn left and jump) tasks. For '+Attn-Dep' models we removed the connections from the previous target word embedding to the decoder state and the pre-output layer.

	Simple	Length	Turn left	Jump
RNN	26.9 ± 0.2	0.2 ± 0.1	26.4 ± 12.0	0.0 ± 0.0
GRU	99.0 ± 0.1	12.9 ± 1.2	47.5 ± 4.7	0.0 ± 0.0
LSTM	99.1 ± 0.1	10.9 ± 1.3	42.9 ± 2.9	0.0 ± 0.0
RNN +Attn	99.8 ± 0.1	19.4 ± 0.7	44.1 ± 0.9	0.3 ± 0.3
RNN +Attn-Dep	61.1 ± 0.3	0.5 ± 0.2	18.6 ± 1.0	0.0 ± 0.0
GRU +Attn	99.8 ± 0.1	17.2 ± 1.9	55.9 ± 3.5	0.0 ± 0.0
GRU +Attn-Dep	51.2 ± 1.2	2.0 ± 1.4	16.9 ± 1.2	0.0 ± 0.0
LSTM +Attn	99.1 ± 0.2	17.1 ± 2.0	48.3 ± 1.7	0.0 ± 0.0
LSTM +Attn-Dep	38.9 ± 0.9	1.0 ± 0.5	17.2 ± 1.2	0.0 ± 0.0

Table 5: NACS test scores on the simple, length, and primitive (turn left and jump) tasks. For '+Attn-Dep' models we removed the connections from the previous target word embedding to the decoder state and the pre-output layer.

	0	1	2	4	8	16	32
RNN	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.0	0.1 ± 0.1	0.5 ± 0.3	1.4 ± 0.3
GRU	0.1 ± 0.0	0.2 ± 0.1	0.6 ± 0.2	2.5 ± 1.1	3.3 ± 0.9	13.1 ± 2.4	42.4 ± 2.5
LSTM	0.1 ± 0.0	0.3 ± 0.2	1.3 ± 0.2	3.8 ± 1.8	2.5 ± 1.1	6.5 ± 2.7	21.3 ± 1.4
RNN +Attn	3.5 ± 3.0	35.0 ± 2.8	48.6 ± 8.1	77.6 ± 2.6	89.2 ± 3.8	98.7 ± 1.3	99.8 ± 0.1
RNN +Attn-Dep	2.7 ± 1.7	29.5 ± 10.5	53.3 ± 10.2	82.4 ± 4.7	98.8 ± 0.8	99.8 ± 0.1	100.0 ± 0.0
GRU +Attn	12.5 ± 6.6	58.2 ± 12.0	67.8 ± 3.4	80.3 ± 7.0	88.0 ± 6.0	98.3 ± 1.8	99.6 ± 0.2
GRU +Attn-Dep	0.7 ± 0.4	70.9 ± 11.5	61.3 ± 13.5	83.5 ± 6.1	99.0 ± 0.4	99.7 ± 0.2	100.0 ± 0.0
LSTM +Attn	7.8 ± 0.9	40.2 ± 9.3	37.7 ± 10.7	50.3 ± 13.9	62.2 ± 7.7	94.0 ± 2.7	98.6 ± 1.0
LSTM +Attn-Dep	0.8 ± 0.6	39.0 ± 6.5	43.6 ± 17.6	66.0 ± 1.6	86.1 ± 2.3	98.7 ± 1.6	99.8 ± 0.2
L&B	0.1	0.1	0.1	4.1	15.3	70.2	89.9

Table 6: SCAN test scores for jump with additional composed commands.

	0	1	2	4	8	16	32
RNN	0.0 ± 0.0	0.1 ± 0.0	0.1 ± 0.1	0.2 ± 0.0	0.7 ± 0.2	0.4 ± 0.0	0.8 ± 0.1
GRU	0.0 ± 0.0	0.3 ± 0.2	0.4 ± 0.1	0.3 ± 0.2	1.0 ± 0.4	5.8 ± 0.1	20.8 ± 2.2
LSTM	0.0 ± 0.0	0.6 ± 0.4	0.5 ± 0.3	0.7 ± 0.0	1.0 ± 0.3	3.7 ± 0.4	11.4 ± 1.2
RNN +Attn	0.3 ± 0.3	2.8 ± 0.8	9.3 ± 7.3	24.7 ± 4.2	43.7 ± 4.4	57.1 ± 5.2	69.1 ± 2.1
RNN +Attn-Dep	0.0 ± 0.0	0.4 ± 0.1	0.9 ± 0.2	2.4 ± 0.3	3.9 ± 0.3	9.3 ± 0.3	15.9 ± 1.4
GRU +Attn	0.0 ± 0.0	5.5 ± 1.8	9.2 ± 2.8	11.0 ± 1.5	21.9 ± 2.4	23.5 ± 0.6	42.0 ± 1.5
GRU +Attn-Dep	0.0 ± 0.0	0.1 ± 0.1	0.6 ± 0.2	2.0 ± 0.2	3.2 ± 0.2	5.8 ± 1.1	10.9 ± 0.8
LSTM +Attn	0.0 ± 0.0	2.1 ± 0.2	3.7 ± 0.9	6.6 ± 0.5	12.5 ± 2.5	21.8 ± 2.6	34.2 ± 1.7
LSTM +Attn-Dep	0.0 ± 0.0	0.4 ± 0.2	0.9 ± 0.1	1.5 ± 0.2	1.9 ± 0.3	3.2 ± 0.6	7.4 ± 0.9

Table 7: NACS test scores for jump with additional composed commands.

	En-Fr	Fr-En
RNN +Attn	29.1 ±0.4	34.9 ±0.8
RNN +Attn-Dep	27.5 ±0.7	32.9 ±0.8
GRU +Attn	32.1 ±0.3	37.5 ±0.6
GRU +Attn-Dep	30.2 ±0.3	35.9 ±0.3
LSTM +Attn	31.5 ±0.2	36.9 ±1.1
LSTM +Attn-Dep	28.7 ±0.2	34.0 ±0.1

Table 8: Results (BLEU) on the Machine Translation experiment for both directions.

	En-Fr	Fr-En
RNN +Attn	79.2 ±15.6	41.7 ±5.9
RNN +Attn-Dep	66.7 ±5.9	41.7 ±5.9
GRU +Attn	70.8 ±11.8	54.2 ±5.9
GRU +Attn-Dep	58.3 ±5.9	45.8 ±11.8
LSTM +Attn	75.0 ±10.2	41.7 ±15.6
LSTM +Attn-Dep	50.0 ±10.2	41.7 ±5.9

Table 9: Machine Translation: accuracy on eight novel sentences containing ‘daxy’ (‘daxiste’).

	En-Fr	Fr-En
RNN +Attn	66.7 ±5.9	20.8 ±5.9
RNN +Attn-Dep	66.7 ±5.9	29.2 ±15.6
GRU +Attn	62.5 ±0.0	33.3 ±5.9
GRU +Attn-Dep	66.7 ±5.9	25.0 ±20.4
LSTM +Attn	66.7 ±5.9	25.0 ±10.2
LSTM +Attn-Dep	62.5 ±0.0	25.0 ±17.7

Table 10: Machine Translation: accuracy on eight novel sentences containing ‘tired’ (‘fatigué’).