Detecting Compositionally Out-of-Distribution Examples in Semantic Parsing

Denis Lukovnikov and **Sina Däubener** and **Asja Fischer**
Ruhr University Bochum
Bochum, Germany
{denis.lukovnikov,sina.daeubener,asja.fischer}@rub.de

Abstract

While neural networks are ubiquitous in state-of-the-art semantic parsers, it has been shown that most standard models suffer from dramatic performance losses when faced with compositionally out-of-distribution (OOD) data. Recently several methods have been proposed to improve compositional generalization in semantic parsing. In this work we instead focus on the problem of *detecting* compositionally OOD examples with neural semantic parsers, which, to the best of our knowledge, has not been investigated before. We investigate several strong yet simple methods for OOD detection based on predictive uncertainty. The experimental results demonstrate that these techniques perform well on the standard SCAN and CFQ datasets. Moreover, we show that OOD detection can be further improved by using a heterogeneous ensemble.

1 Introduction

Neural network (NN) based models are ubiquitous in natural language processing (NLP). In particular, sequence-to-sequence models have found adoption in neural machine translation (NMT (Bahdanau et al., 2014; Luong et al., 2015)), neural semantic parsing (NSP (Dong and Lapata, 2016)), and beyond. While basic sequence-to-sequence models have shown impressive results on these tasks, recent work (Lake and Baroni, 2018; Keysers et al., 2019; Kim and Linzen, 2020) have presented the disconcerting finding that these models fail to generalize to novel combinations of elements observed in the training set (see Section 2). Therefore, several models and methods with improved compositional generalization have recently been proposed (Liu et al., 2020; Li et al., 2019; Russin et al., 2020; Guo et al., 2020a; Gordon et al., 2019; Herzig and Berant, 2020; Furrer et al., 2020; Andreas, 2020; Guo et al., 2020b; Herzig et al., 2021)

In this work, we consider the task of **detecting** compositionally out-of-distribution (OOD) examples, which, to the best of our knowledge has not been investigated before. The ability to detect OOD inputs is important, as it helps us to decide whether the model's prediction on the input can be trusted, which is crucial for safe deployment of the model and could be useful to build more efficient systems.

To this end, we analyse the OOD detection performance of recurrent neural network (RNN) and transformer-based models using methods relying on predictive uncertainty. In addition, we propose to use a heterogeneous ensemble of transformer and RNN-based models that combines the strengths of both to improve the detection of compositionally OOD examples.

2 Background

Several recent works have investigated the generalization properties of commonly used sequence-to-sequence models, in particular their ability to learn to process and produce novel combinations of elements observed during training (Lake and Baroni, 2018; Keysers et al., 2019; Kim and Linzen, 2020).

Lake and Baroni (2018) propose the SCAN dataset, which consists of natural language utterances (input) and action sequences (output), and perform an analysis of the generalization performance of sequence-to-sequence models on different splits of the dataset. The different splits are aimed at testing the ability of networks to (1) generalize to novel combinations of tokens observed only in isolation during training (the JUMP and TURN_LEFT settings) and (2) generalize to longer sequence lengths (the LENGTH setting). For the JUMP setting, the training set consists of the basic example "jump" $\rightarrow$ [JUMP] as well as all other simple and composed examples (e.g. "run twice") while the test set contains all composed examples with "jump" (e.g. "jump twice"). They observe that standard sequence-to-sequence models fail on the JUMP and LENGTH splits (accuracy below 10%) while they perform well (near 100%) on a random

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 591–598
November 7–11, 2021.

test split.

Keysers et al. (2019) performed their analysis on the CFQ dataset that provides tens of thousands of automatically generated question/SPARQL-query pairs and provides maximum compound divergence (MCD) splits. The MCD splits are generated such that the distributions over compounds (phrases combining atomic elements) are maximally different between the train and test sets while the distributions over the atomic elements (entities, relations, question patterns) are kept similar. Keysers et al. (2019) also provide MCD splits for the SCAN dataset. Experiments using standard neural sequence-to-sequence models (transformers and RNN+attention) reveal that while the random splits result in near-perfect accuracy, the MCD splits suffer dramatic losses in performance ($< 20\%$ accuracy for CFQ's MCD splits and $< 10\%$ for SCAN's MCD splits).

3 Detecting OOD examples

In this work, we focus on OOD detection methods that build on the predictive distributions of discriminative task-specific models (extending the work of Hendrycks and Gimpel (2017)). These methods have the advantage that they are easy to use in existing models and do not require additional models or additional training (unlike for example generative modeling (Nalisnick et al., 2018; Ren et al., 2019)). Previous work has shown that neural network models can produce incorrect predictions with high confidence on OOD inputs (Nguyen et al., 2015), which can be detrimental for detecting such inputs. We investigate whether this is the case for compositionally OOD examples in semantic parsing models as well.

We compare the following measures quantifying the uncertainty of the prediction based on the output distributions of a trained model: (1) the average negative log-likelihood (NLL) for the generated sequence, (2) the sum of the NLLs, and (3) the average entropy of the output distributions. More specifically, our approach for measuring uncertainty proceeds as outlined: First, the input x is encoded and an output sequence $\hat{y}$ is generated by the decoder. The model's output probability distributions $p(\hat{y}_i|\hat{y}_{<i}, x)$ for every decoding step i are then used to compute the sum of NLL as

$$-\log p(\hat{y}_i|\hat{y}_{<i}, x) \ . \quad (1)$$

The average entropy is given by

$$-\frac{1}{T}\sum_{i=1}^{T}\sum_{y_i \in \mathcal{V}} p(y_i|\hat{y}_{<i}, x)\log p(y_i|\hat{y}_{<i}, x) \ , \quad (2)$$

where $\mathcal{V}$ is the set of all output tokens.

3.1 MC Dropout

To take model uncertainty into account, Bayesian approaches can be used (Louizos and Welling, 2017; Maddox et al., 2019; Malinin and Gales, 2018). A simple method for approximating the predictive uncertainty under a Bayesian posterior distribution over model parameters, is MC Dropout (Gal and Ghahramani, 2016).

In our work, we use MC dropout as follows: First, we encode the input x and run the decoder to generate an output sequence $\hat{y}$. Then, we obtain K output probability distributions $p_k(y_i|\hat{y}_{<i}, x)$, $k = 1, \dots K$, for each decoding step by feeding x and the generated $\hat{y}$ through the model K times while randomly dropping neurons with the same probability as during training. Finally, the posterior predictive distribution is approximated by $\frac{1}{K}\sum_{k=1}^{K} p_k(y_i|\hat{y}_{<i}, x)$ and is used with the metrics described previously.

3.2 Homogeneous ensemble

Another method often used for uncertainty quantification are deep ensembles (Lakshminarayanan et al., 2017), where K models with the same architecture and hyperparameters are trained in parallel starting with different initalizations. The final prediction is given as the average over the single predictions. For our sequence models, we average the predictive distributions of the ensembled models at every decoding step.

3.3 Heterogeneous ensemble

In our experiments, we found that different underlying architectures are better at detecting different types of OOD examples. To further improve detection performance, we propose to use a heterogeneous ensemble of different models for compositional OOD detection in semantic parsing. Concretely, given M different architectures (in our case $M = 2$), we first train an ensemble (in our case of size 3) of each architecture, and during prediction, analogously to the regular ensemble, we average the predictive distributions of *all* the models at every time step.

Method	MC Drop	Ensemble	JUMP			TURN_LEFT			LENGTH			MCD		
			AUROC	AUPRC	FPR90	AUROC	AUPRC	FPR90	AUROC	AUPRC	FPR90	AUROC	AUPRC	FPR90
TM+avgENT	No	No	99.5	99.9	1.4	86.7	82.3	40.8	79.8	91.2	61.5	88.1	91.4	43.5
TM+sumNLL	No	No	99.5	99.9	1.4	82.5	78.7	58.8	88.4	94.8	32.7	91.5	93.6	29.0
GRU+avgENT	No	No	89.9	98.2	38.9	84.5	84.6	51.5	93.6	96.9	20.0	91.0	92.8	25.9
GRU+sumNLL	No	No	85.9	97.0	40.3	81.6	81.3	59.8	97.7	98.9	6.6	91.2	92.4	23.0
TM+avgENT	5	No	99.5	99.9	1.2	92.4	88.8	22.3	78.5	89.2	62.1	92.8	94.0	19.0
TM+sumNLL	5	No	99.5	99.9	0.6	91.1	87.2	26.5	85.3	93.6	43.1	94.4	95.6	16.8
GRU+avgENT	5	No	90.1	98.2	34.3	87.2	86.5	47.7	88.8	93.8	29.0	93.5	94.5	17.7
GRU+sumNLL	5	No	85.6	96.6	39.5	84.8	83.1	55.5	93.0	96.4	18.3	92.8	93.9	17.7
TM+avgENT	No	5	99.9	**100**	0.2	**98.5**	97.2	**4.2**	93.0	96.8	22.8	96.8	97.7	7.2
TM+sumNLL	No	5	**100**	**100**	**0.0**	**98.5**	**97.7**	4.6	96.0	98.4	14.9	98.3	98.6	5.3
GRU+avgENT	No	5	91.4	98.4	29.7	97.4	96.6	7.0	94.6	96.8	14.0	97.7	97.9	7.6
GRU+sumNLL	No	5	86.6	97.1	34.9	96.5	95.7	12.8	97.0	98.5	7.9	95.2	93.4	12.4
HE+avgENT	5	1+1	99.2	99.9	2.0	95.2	91.5	11.8	89.3	94.6	29.5	98.6	98.9	2.9
HE+sumNLL	5	1+1	98.7	99.8	2.8	94.1	89.4	13.5	92.7	96.5	21.8	98.7	98.9	2.6
HE+avgENT	No	3+3	**100**	**100**	0.1	97.1	96.2	6.2	97.3	98.6	5.7	**99.5**	**99.6**	**0.8**
HE+sumNLL	No	3+3	99.9	**100**	0.1	96.0	94.7	10.4	**98.3**	**99.2**	**4.4**	**99.5**	**99.6**	**0.4**

Table 1: OOD Detection performance on SCAN's splits for the transformer (TM), the GRU-based sequence-to-sequence model with attention (GRU), and heteogenous ensembles (HE). The results for MCD correspond to the average over the three MCD splits. If "MC Drop" is "No", MC dropout is not used during prediction, otherwise the value of "MC Drop" specifies the number of samples (K in Section 3.1). If "Ensemble" is "No", homogeneous ensemble is not used, otherwise, its value specifies the number of models in the ensemble. 3+3 specifies that we use ensembles of 3 transformer models and 3 GRU models in the heterogenous ensemble. Best result is shown in bold, close to best are underlined.

We also combine heterogeneous ensembles and MC dropout, the approach for which is described in Appendix C.

4 Experiments[1]

Datasets: We experiment with the SCAN (Lake and Baroni, 2018) and CFQ (Keysers et al., 2019) datasets mentioned in Section 2. Table 4 in Appendix A provides some statistics on the number of examples in each split.

Models: We consider both a transformer based (Vaswani et al., 2017) and a GRU+attention (Cho et al., 2014) based sequence-to-sequence model in our experiments. Both are randomly initialized. For transformers, we use six layers with six heads, learned position embeddigns, dimension 384 and dropout rate 0.25. For the GRU-based model, we use two-layers, dimension 384 hidden layers and dropout probability 0.1[2]. The models are trained using Adam (Kingma and Ba, 2015), with an initial learning rate of $5 * 10^{-4}$. For more details, see Appendix B. We ran all experiments with three[3] different seeds and report the average.

Evaluation: To evaluate the ability of the techniques presented in Section 3 to detect OOD examples, the following metrics are computed: (1) AUROC↑[4], (2) AUPRC↑[5], and (3) FPR90↓[6]. These metrics are commonly used to measure the performance in OOD detection as well as for binary classifiers in general.

Data splits: The experiments are conducted on a slightly different data split compared to previous work and thus the obtained accuracies might not be directly comparable. To evaluate OOD detection performance, the test set must contain both in-distribution (ID) and out-of-distribution (OOD) examples. The ID test examples must be similar to the training data but must **not** have been seen during training. Due to the lack of a predefined ID test set (that does not overlap with the validation set), we randomly split off 10% of the training examples

[1]Code is available at https://github.com/lukovnikov/parseq/tree/emnlpcr/

[2]0.25 for homogeneous ensemble.

[3]We used only one or two seeds for each of the three MCD splits of CFQ experiments because of the long training times.

[4]Area under the receiver operating characteristic.

[5]Area under the precision-recall curve.

[6]FPR90 is the false positive rate at 90% true positive rate.

Method	SCAN				CFQ
	JUMP	T._LEFT	LEN.	MCD	
TM	0.4	90.0	0.0	1.7	17.0
GRU	0.2	62.8	12.1	20.9	9.4

Table 2: Logical form accuracy of the considered models on the OOD test sets.

as the ID test set and train only on the remaining 90%. The reported AUROC, AUPRC, and FPR90 metrics are then computed by taking the original test set (which is assumed to be OOD) as examples of the positive class and the ID test examples as the negative class. Note that OOD data are not used in any way during training.

Prediction accuracy: The accuracy of the models is evaluated using a tree-based logical form accuracy. See Appendix B for details. The results reported in Table 2 verify that the query accuracy is similar to previously reported numbers. They show that the standard sequence-to-sequence models fail on all compositional generalization scenarios except on the TURN_LEFT split from SCAN. In contrast, the ID test accuracy was near 100% for both datasets.

5 Results

The OOD detection performance for the different splits of the SCAN dataset are reported in Table 1, and for the CFQ dataset in Table 3. SCAN's random split obtains 50% AUROC, which is expected since it does not contain OOD data.[7]

The effect of MC-dropout: The method described in Section 3.1 leads to improvements across different settings and architectures, with the exception of SCAN's length-based split.

The effect of architecture: Different architectures appear to produce markedly different results for different types of splits on SCAN. The transformer performs better than the GRU-based model on the primitive generalization splits (SCAN's JUMP and TURN_LEFT splits), slightly underperforms on the MCD splits of both SCAN and CFQ and is worse on the length-based SCAN split.

How difficult are the different splits? Some of the splits are more challenging to detect than others. The JUMP split appears the easiest to detect (see Table 1). The TURN_LEFT split is more challenging. The high query accuracy on this test set in Table 2 might indicate that it is closer to the training distribution than the others. Nevertheless, several methods are able to achieve high detection performance for TURN_LEFT. The transformer fails to produce any correct output on the length-based split of SCAN and is also bad at *detecting* when it encounters such examples.

The effect of homogeneous ensemble: The regular (homogeneous) deep ensemble (Lakshminarayanan et al., 2017) leads to significant improvements of the OOD detection ability across all tested architectures and datasets. However, using an ensemble is not always sufficient to close the performance gap to the best performing architecture on a certain split (e.g. GRU on the length-based split). Note that a disadvantage of using ensembles is the increased computational requirements, which can be especially prohibitive for large transformer-based models.

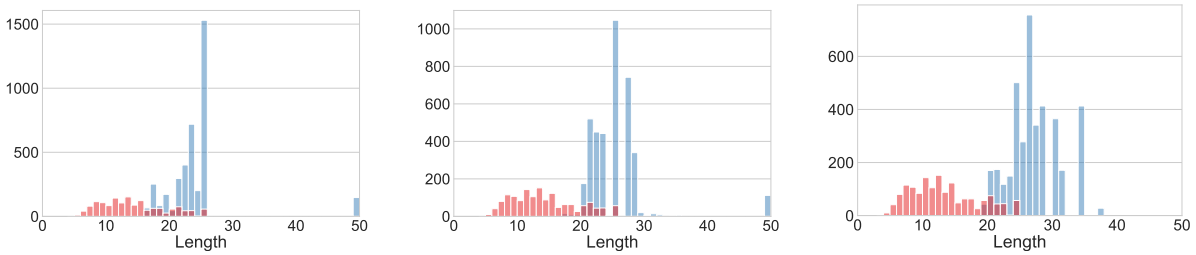
The effect of heterogeneous ensemble: Using the heterogeneous ensemble of a transformer and a GRU-based sequence-to-sequence model to detect OOD examples yields the best overall results. The heterogeneous ensemble leads to an overall improvement both in combination with MC dropout and with regular ensemble. Most notable are the gains on SCAN's MCD splits, reaching an FPR90 of less than 5% with MC Dropout and below 1% with regular ensemble. It also appears to improve results on the TURN_LEFT split and beats the detection performance of the ensembled GRU-based model on the length-based SCAN split.

6 Analysis and Discussion

In the results obtained in Table 1, two things stand out: (1) the gap in OOD detection performance between the transformer and the GRU-based model on the length-based split and (2) the extremely high OOD detection performance of the transformer on the JUMP split. In this section, we perform a further analysis to try to better understand these findings.

Length-based split: To analyse what may have caused the poor performance of the transformer on the length-based split, we investigate the lengths of the generated outputs (see Figure 1). We found that the transformer with absolute position encodings (PE) that produced the results in Table 1 and 3 is more biased towards generating shorter sequences than a transformer with relative PE or the GRU. However, the transformer with relative PE, which

[7] Note that 50% AUROC corresponds to a random classifier. We leave these results out of the tables because of space constraints.



(a) Transformer, absolute PE. (b) Transformer, relative PE. (c) GRU-based model.

Figure 1: Histograms of lengths for outputs generated by two different transformer models for SCAN's length-based split. Blue is on ID inputs, red is on OOD inputs. Note that here we also count the tokens added at the beginning and end of the sequences.

Method	MC Drop	Ensemble	MCD AUROC	MCD AUPRC	MCD FPR90
TM+avgENT	No	No	92.9	94.8	18.7
TM+sumNLL	No	No	91.5	93.5	21.3
GRU+avgENT	No	No	93.3	95.1	16.4
GRU+sumNLL	No	No	91.9	93.7	19.6
TM+avgENT	5	No	95.0	96.5	11.9
TM+sumNLL	5	No	93.7	95.2	15.6
GRU+avgENT	5	No	94.5	96.0	12.7
GRU+sumNLL	5	No	93.1	94.7	16.3
TM+avgENT	No	5	95.9	97.2	8.0
TM+sumNLL	No	5	95.0	96.3	10.5
GRU+avgENT	No	5	96.3	97.5	6.1
GRU+sumNLL	No	5	95.6	96.9	8.1
HE+avgENT	5	1+1	95.0	96.4	11.4
HE+sumNLL	5	1+1	93.9	95.3	13.7
HE+avgENT	No	3+3	**96.7**	**97.8**	**4.8**
HE+sumNLL	No	3+3	95.9	97.2	6.9

Table 3: OOD Detection performance on CFQ's MCD splits, averaged over the three provided MCD splits. The table is structured similarly to Table 1.

reaches 86.0 AUROC and 41.1 FPR90, still performs poorly compared to the GRU-based model (AUROC: 93.6, FPR90: 20.0).

SumNLL sums over the entire sequence and simply producing longer sequences, even with similar per-timestep entropies to ID data, would lead to better distinguishable examples. However, for AvgENT, which is averaged over time steps and therefore not influenced by the length, the GRU-based model still performs better than the transformer.

Thus, we believe that while the length of the generated sequences can be an important signal for detection, and may give a slight benefit to the GRU-based model, it is not the only reason of the high performance of the GRU-based model.

Transformer on JUMP split: To ensure that the high performance of the transformer on JUMP is not just due to the exploitation of trivial input features we experimented with additional JUMP examples that put the word "jump" in all other positions to avoid correlation with the position vectors. This indeed resulted in slightly worse OOD detection ability. However, with an FPR90 of 4.6, the transformer was still better than the GRU-based model.

7 Conclusion

In this work, we investigate how easy it is for neural semantic parsers to detect out-of-distribution examples in the context of compositional generalization. While some recent works (Fomicheva et al., 2020; Malinin and Gales, 2021) investigate similar methods for structured prediction (for NMT and automated speech recognition), to the best of our knowledge, our work is the first to investigate compositional OOD detection for NSP. Our analysis shows that relatively simple uncertainty based methods perform well for RNN as well as transformer-based models in most settings. Ensemble provide the best results, while MC dropout leads to improvements at no extra training cost. OOD detection can further be improved by using an ensemble of RNN and transformer-based models.

Acknowledgements

This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

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.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *ACL (2016)*. The Association for Computer Linguistics.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2019. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020a. Hierarchical poset decoding for compositional generalization in language. *Advances in Neural Information Processing Systems*.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2020b. Revisiting iterative back-translation from the perspective of compositional generalization. *arXiv preprint arXiv:2012.04276*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*.

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040*.

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

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*.

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 4284–4293.

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

Christos Louizos and Max Welling. 2017. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*.

Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*.

Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. 2018. Do deep generative models know what they don't know? In *International Conference on Learning Representations*.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. 2019. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural In-formation Processing Systems*.

A Dataset statistics

See Table 4 for statistics of the used datasets and our splits. Note that the datasets come with three predefined MCD-based splits, which are referred to by MCD-{X} for $X \in \{1, 2, 3\}$ in the table.

Table 4: Number of examples per dataset and split.

Dataset	Split	#Train	# Dev	#Test	
				ID	OOD
SCAN	JUMP	12.0k	1.5k	1.3k	7.7k
	TURN_LEFT	18.0k	2.2k	2.0k	1.2k
	LENGTH	14.0k	1.7k	1.5k	3.9k
	MCD-{X}	7.5k	1.0k	0.8k	1.1k
CFQ	MCD-{X}	86.0k	12.0k	9.6k	12.0k

B Experimental details

Early stopping: We initially applied early stopping based on the *original* validation set for all splits. While for SCAN's original splits (JUMP, LENGTH and TURN_LEFT) these validation sets contain ID examples, SCAN's and CFQ's MCD splits have a validation set consisting out of OOD examples. In early experiments we found that early stopping based on OOD examples results in poorer OOD performance with high variance because the resulting model is often retained from very early training steps. For this reason, we do not use early stopping for the two MCD splits and instead train for a fixed number of epochs (25 for transformer on SCAN, 40 for GRU on SCAN, 20 for transformer on CFQ and 40 for GRU on CFQ.

Preprocessing: The CFQ dataset is preprocessed using a simple reversible transformation of SPARQL queries into LISP-style s-expressions. This includes converting the set of triple patterns of the form "?x :rel ?y. ?a :r ?b" to s-expressions of the form "(AND (COND ?x :rel ?y) (COND ?x : rel ?y))", where the order of the arguments of "AND" does not matter during evaluation.

Accuracy: The logical form accuracy considers an example correct if the logical form is equivalent to the target logical form, and which is invariant to the effects of linearization order. In the case of CFQ, which uses SPARQL, this means that the order of conditions does not affect the accuracy of the obtained results and is therefore ignored. In the case of SCAN, whose outputs are action sequences, this simply becomes the sequence-level accuracy.

C MC dropout with heterogeneous ensemble

When we apply MC dropout to the heterogeneous ensemble, we train only **one** model for each of the two different architectures. These models are used to independently predict a sequence $\hat{y}_m$ with $m \in \{1, 2\}$ given x with dropout disabled. Next, we feed x and $\hat{y}_m$ through both models K times with dropout enabled, leading to two averaged outputs $\bar{y}_m$ over $K * 2$ distributions. Finally, we perform max-pooling over the NLL-based metrics computed for each $\bar{y}_m$ such that the most pessimistic score is retained.