Systematic Generalization: What Is Required and Can It Be Learned?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Numerous models for grounded language understanding have been recently proposed, including (i) generic models that can be easily adapted to any given task and (ii) intuitively appealing Neural Module Networks (Andreas et al., 2016a) that require background knowledge to be instantiated. We compare generic and modular models in how much they lend themselves to a particular form of systematic generalization. Our findings show that the generalization of modular models is much more systematic and that it is highly sensitive to the module layout, i.e. to how exactly the modules are connected. We furthermore investigate if modular models that generalize well could be made more end-to-end by learning their layout and parametrization. We show how end-to-end methods from prior work often learn spurious layouts and parametrizations that do not facilitate systematic generalization. Our results suggest that, in addition to modularity, systematic generalization in language understanding may require explicit regularizers or priors.

1 Introduction

In recent years, neural network based models have become the workhorse of natural language understanding and generation showing state-of-the-art performance on numerous benchmarks, including Recognizing Textual Entailment (RTE) (Gong et al., 2017), Visual Question Answering (VQA) (Jiang et al., 2018), and Reading Comprehension (Wang et al., 2018). Despite these successes, a growing body of literature suggests that these approaches latch onto statistical regularities which are omnipresent in existing datasets (Agrawal et al., 2016; Gururangan et al., 2018; Jia & Liang, 2017) and do not generalize outside of the specific distributions they are trained on. These findings have recently been corroborated by Lake & Baroni (2018), who showed that seq2seq models (Sutskever et al., 2014; Bahdanau et al., 2015) show little systematicity (Fodor & Pylyshyn, 1988) in how they generalize, i.e. they fail to learn general rules on how to compose words.

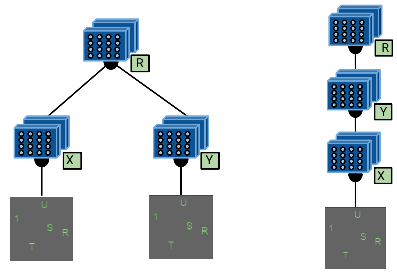
Introduced by Andreas et al. (2016b), Neural Module Networks (NMNs) approach aims to improve the generalization capabilities of neural models by adding *modularity* and *structure* to their design to make them resemble the kind of rules they are supposed to learn. The NMN approach, while intuitively appealing, has seen limited adoption because of the large amount of domain knowledge it requires to decide (Andreas et al., 2016a) or predict (Johnson et al., 2017; Hu et al., 2017) how the modules should be created (parametrization) and how they should be connected (layout) based on a query. Besides, their performance has often been matched by more generic neural models, such as FiLM (Perez et al., 2017), Relations Networks (Santoro et al., 2017), and CAN (Hudson & Manning, 2018), and their generalization, to the best of our knowledge, has not been a subject of a focused study. In this work we investigate the impact of explicit modularity and structure on systematic generalization by studying the generalization of NMNs and contrasting it to those of generic models. We choose to focus on the following basic generalization requirement: *a good model should be able*

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

a: Is there an S above T? **True**

b: Is there a W left of an A? **False**

Figure 1: SQOOP tests model's ability to reason about all object pairs after being trained on a small subset of them. We show a positive (left) and negative (right) example from SQOOP.



a: NMN-tree

b: NMN-chain

Figure 2: The different NMN layouts for the question "is there a S above T".

to reason about all possible object combinations despite being trained on a small subset of them. We instantiate this requirement in form of a simple yes-no Visual Question Answering (VQA) dataset.

Our first finding is that NMNs do generalize better than other neural models when an appropriate choice of layout and parametrization is made. We furthermore experiment with existing methods for making NMNs more end-to-end by inducing the module layout (Johnson et al., 2017) or learning module parametrization through soft-attention over the question (Hu et al., 2017) . We show how such end-to-end approaches often fail to find the right structural settings and instead prefer a wrong chain layout or spurious parametrization and do not generalize better than the generic models. We believe that our findings challenge the intuition of researchers in the field and provide a foundation for improving systematic generalization of neural approaches to language understanding.

2 Setup

Dataset: SQOOP (Spatial Queries Over Object Pairs, Figure 1) is a minimalistic VQA task designed to test a particular type of generalization: the ability to disentangle the meaning of relation words and object words and then compose these meanings in novel contexts to perform basic relational reasoning in a consistent way. Concretely, SQOOP requires answering a yes-no question $q = \mathrm{X\,R\,Y}$ about whether objects X and Y are in a spatial relation R, given a 64×64 RGB image x. x contains 5 randomly chosen and randomly positioned objects. There are 36 objects: letters A-Z and digits 0-9, and 4 relations: LEFT_OF, RIGHT_OF, ABOVE, and BELOW. Our goal is to discover which models can correctly answer questions about all $36 \cdot 36$ possible object pairs in the SQOOP dataset after having been trained on only a subset. Therefore, we train on $36 \cdot 4 \cdot k$ unique questions, where for every left-hand-side (LHS) object X, we randomly sample k different right-hand-side objects (RHS), and test on the remaining $36 \cdot 4 \cdot (36 - k)$ questions. We refer to k as the *#rhs/lhs parameter* of the dataset. To exclude a possible compounding factor of overfitting on training images, all our training sets contain 1 million examples obtained by sampling multiple images per question.

Models: We experiment with models from 2 broad categories. *Generic* models such as FiLM (Perez et al., 2017), Relation Networks (RelNet, Santoro et al. (2017)) and CAN Hudson & Manning (2018), and *modular* and *structured* Neural Module Networks (NMN). NMNs (Andreas et al., 2016b) construct question-specific networks by composing together trainable neural modules. To answer a question with an NMN, a computation graph is constructed by making 2 decisions: *layout* - the number of modules, their types and how they are connected, and *parametrization* - how these modules are parametrized based on the question. For our study we adapt the N2NMN (Hu et al., 2017) paradigm from this family, which predicts the layout with a seq2seq model (Sutskever et al., 2014) and computes the parametrization of the modules using a soft attention mechanism. Since all the questions in SQOOP have the same structure, we can get away with a single trainable layout variable and separate trainable attention variables per each module. We also experiment with hard-coded layout and parametrization setting, in the spirit of original NMN (Andreas & Klein, 2015).

Formally, our NMN is constructed by repeatedly applying a *generic neural module* $f(\theta, \gamma, h_l, h_r)$, which takes as inputs the shared parameters θ, the question-specific parametrization γ and the left-

Table 1: Comparing the performance of generic models to the structured NMN-Tree model on the hardest version of our dataset (lower #rhs/lhs is more difficult).

model	train. acc (%)	test acc. (%)
Conv+LSTM	97.9	64.4 ± 1.8
RelNet	95.6	63.1 ± 1.0
FiLM	100	66.6 ± 2.5
MAC	99.5	72.6 ± 3.4
NMN-Tree (Residual)	100	100.0 ± 0.0
NMN-Tree (Find)	100.0	99.7 ± 0.3
NMN-Chain (Find)	99.2	51.4 ± 2.8
NMN-Chain-XYR (Residual)	100	51.6 ± 1.6
NMN-Chain-XRY (Residual)	99.7	54.1 ± 1.7
NMN-Chain-RXY (Residual)	98.7	50.5 ± 0.9

hand side and right-hand side inputs h_l and h_r. M such modules are connected and conditioned on a question $q = (q_1, q_2, q_3)$ as follows:

$$\gamma_k = \sum_{i=1}^{s} \alpha^{k,i} e(q_i) \tag{1}$$

$$h_k = f(\theta, \gamma_k, \sum_{j=-1}^{k-1} \tau_0^{k,j} h_j, \sum_{j=-1}^{k-1} \tau_1^{k,j} h_j) \tag{2}$$

In the equations above, $h_{-1} = 0$ is the zero tensor, $h_0 = h_x$ are the image features outputted by a CNN network referred to as the *stem*, and e is the embedding table for the questions words. We refer to $A = (\alpha^{k,i})$ and $T = (\tau_0^{k,i}, \tau_1^{k,i})$ as the *parametrization attention matrix* and the *layout tensor* respectively. The output of the final module, h_M is fed into a fully connected classifier network to make predictions. We perform our experiments with the Find module from Hu et al. (2017) and the $\mathrm{Residual}$ module from Johnson et al. (2017) with minor modifications (Appendix-A.3 for details).

Based on the generic NMN model described above, we experiment with several specific architectures. Each of the models uses $M = 3$ modules, which are connected and parametrized differently. In **NMN-Chain** the modules form a sequential chain as shown in Figure 1b. Modules 1, 2 and 3 are parametrized based on the first object word, second object word and the relation word respectively, which is achieved by setting the attention α_1, α_2, α_3 to the corresponding one-hot vectors. **NMN-Tree** has similar hard-coded attention vectors, with a tree-like connectivity between the modules. 1a. In the **Stochastic N2NMN**, similar to the N2NMN (Hu et al., 2017), the layout T is treated as a stochastic latent variable that takes two values: T_{tree} as in NMN-Tree, and T_{chain} as in NMN-Chain. The output probabilities are computed by marginalizing out T, i.e. probability of label "yes" is computed as $p(\text{yes}|x, q) = \sum_{T \in \{T_{tree}, T_{chain}\}} p(\text{yes}|T, x, q)p(T)$. **Attention N2NMN** also from (Hu et al., 2017), is structured just like NMN-Tree but has α^k computed as $\mathrm{softmax}(\tilde{\alpha}^k)$, where $\tilde{\alpha}^k$ is a trainable vector. We use Attention N2NMN only with the Find module, which was designed by (Hu et al., 2017) specifically parametrized with the help of soft attention.

3 Experiments

Which Models Generalize Better: We report the performance for all models on the hardest version of our dataset (#rhs/lhs = 1) in Table 1. These results show that generic models do not generalize well, while the NMN-Tree model does. To understand better the cause of NMN-Tree's advantage we compare the performance of the NMN-Tree and NMN-Chain models. The results show that for both Find and $\mathrm{Residual}$ architectures, using a tree layout is crucial for generalization as NMN-Chain performs barely above random chance.

Can the Right Kind of NMN Be Induced: The generalization of NMN-Tree model, while impressive, is somewhat unsurprising because both the *layout* and *parametrization* of this model encode a significant amount of prior knowledge about the task. We therefore investigate whether the amount of such prior knowledge can be reduced by fixing one of the structural aspects and inducing the other. For inducing a layout, we use the Stochastic N2NMN model. We experiment with both Find and

Table 2: Layout induction results for Stochastic N2NMNs using Residual modules and Find modules. For each setting of $p_0(tree)$, we report results on 1 rhs/lhs and 18 rhs/lhs datasets.

(a) Residual modules

#rhs/lhs	$p_0(tree)$	test acc. (%)	$p_{50K}(tree)$
1	0.1	52.7 ± 2.2	0.003
	0.5	57.0 ± 4.4	0.026
	0.9	99.9 ± 0.1	0.997
18	0.1	100.0 ± 0.0	0.999
	0.5	97.7 ± 5.1	0.999
	0.9	99.1 ± 2.3	0.999

(b) Find modules

#rhs/lhs	$p_0(tree)$	test acc. (%)	$p_{50K}(tree)$
1	0.1	51.2 ± 2.9	0.00
	0.5	93.2 ± 7.1	0.999
	0.9	95.9 ± 1.6	0.999
18	0.1	78.6 ± 20.7	0.2
	0.5	91.6 ± 6.5	0.999
	0.9	97.3 ± 3.4	0.999

Residual modules and report results with diverse initial conditions, $p_0(tree) = 0.1, 0.5, 0.9$, where $p_0(tree)$ is the initial value of $p(T_{tree})$. The results obtained on the #rhs/lhs=1 dataset (Table 2) show that the correct layout was not induced for $p_0(tree) = 0.1$ and $p_0(tree) = 0.5$ with the Residual module and for $p_0(tree) = 0.1$ with the Find module. We also run similar experiments on an easy-to-generalize #rhs/lhs=18 version. Here, the NMN with Residual module preferred the tree layout for all initializations. It is notable, however, that in the setting with #rhs/lhs=1 where the correct choice of the layout is the *only* way to generalize, only a very lucky initialization $p_0(tree) = 0.9$ resulted in successful layout induction for the Residual module.

For parameterization induction, we experiment with the Attention N2NMN model on #rhs/lhs=1 and #rhs/lhs=18. The model often did not find the attention settings that lead to generalization on the challenging #rhs/lhs=1 split (83.8% test accuracy). That should be contrasted with the close-to-perfect 99.2% accuracy of the model that was trained on #rhs/lhs=18 version of the task, suggesting that the parametrization induction did not work due to the difficulty of our #rhs/lhs=1 split. To analyze the learnt attention weights, we compute a sharpness ratio $\rho = \max(\alpha^{k,X}, \alpha^{k,Y}) / \min(\alpha^{k,X}, \alpha^{k,Y})$ for modules $k = 1$ and $k = 2$ for each of the trained modules. We find that the learnt attention weights on #rhs/lhs=1 are generally blurry with $\rho < 2$ for 40% of the modules (details in Appendix-A.4).

4 Related Work

Using synthetic VQA datasets to study grounded language understanding is a recent trend started by CLEVR (Johnson et al., 2016), and recently the ShapeWorld dataset (Kuhnle & Copestake, 2017), that involves a number of VQA generalization tests. Closely related to our work is the recent study on generalization to long-tail questions about rare objects by Bingham et al. (2018). They do not, however, consider as many models as we do and do not study whether the best-performing models can be made end-to-end. Andreas et al. (2016a) introduced NMNs as a modular, structured VQA model where a fixed number of hand-crafted neural modules are chosen and composed together in a layout determined by the dependency parse of the question. Hu et al. (2017) and Johnson et al. (2017) followed up with end-to-end NMNs, removing the non-differentiable parser. Recent concurrent work by (Hu et al., 2018) attempts to remove the need for hard stochastic layout decisions.

5 Conclusion and Discussion

We have conducted a rigorous investigation of an important form of systematic generalization required for grounded language understanding: the ability to reason about all possible pairs of objects despite being trained on a small subset. The intuitive appeal of modularity and structure in designing neural architectures for language is now supported by our results. Our other key finding is that coming up with an end-to-end and/or soft version of modular models may be not sufficient for strong generalization, because in the very setting where strong generalization is required, end-to-end methods may find a different, less compositional solution (e.g. a chain layout or blurred attention). This conclusion is relevant in the view of recent work done in the direction of making Neural Module Networks more end-to-end (Suarez et al., 2018; Hu et al., 2018; Hudson & Manning, 2018). We hope that our findings will inform researchers working on language understanding and provide them with a useful intuition about what facilitates strong generalization and what is likely to inhibit it.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the Behavior of Visual Question Answering Models. *arXiv:1606.07356 [cs]*, June 2016. URL http://arxiv.org/abs/1606.07356. arXiv: 1606.07356.

Jacob Andreas and Dan Klein. Alignment-Based Compositional Semantics for Instruction Following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015. URL https://arxiv.org/abs/1508.06491.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39–48, 2016a.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural Module Networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b. URL http://arxiv.org/abs/1511.02799.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the ICLR 2015*, 2015.

Eli Bingham, Piero Molino, Paul Szerlip, Obermeyer Fritz, and Goodman Noah. Chacterizing how Visual Question Answering scales with the world. In *NIPS Workshop on Visually-Grounded Interaction and Language*, 2018.

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

Yichen Gong, Heng Luo, and Jian Zhang. Natural Language Inference over Interaction Space. *arXiv:1709.04348 [cs]*, September 2017. URL http://arxiv.org/abs/1709.04348. arXiv: 1709.04348.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation Artifacts in Natural Language Inference Data. *arXiv:1803.02324 [cs]*, March 2018. URL http://arxiv.org/abs/1803.02324. arXiv: 1803.02324.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to Reason: End-to-End Module Networks for Visual Question Answering. *arXiv:1704.05526 [cs]*, April 2017. URL http://arxiv.org/abs/1704.05526. arXiv: 1704.05526.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 53–69, 2018.

Drew A. Hudson and Christopher D. Manning. Compositional Attention Networks for Machine Reasoning. In *ICLR 2018*, February 2018.

Robin Jia and Percy Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, 2017. doi: 10.18653/v1/D17-1215. URL https://aclanthology.coli.uni-saarland.de/papers/D17-1215/d17-1215.

Yu Jiang, Vivek Natarajan, Xinlei Chen, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia v0.1: The winning entry to the vqa challenge 2018. https://github.com/facebookresearch/pythia, 2018.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *arXiv:1612.06890 [cs]*, December 2016. URL http://arxiv.org/abs/1612.06890. arXiv: 1612.06890.

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 *ICCV*, 2017. URL http://arxiv.org/abs/1705.03633.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

Alexander Kuhnle and Ann Copestake. ShapeWorld - A new test methodology for multimodal language understanding. *arXiv:1704.04517 [cs]*, April 2017. URL http://arxiv.org/abs/1704.04517. arXiv: 1704.04517.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2018.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer. In *In Proceedings of the AAAI Conference on Artificial Intelligence*, 2017. URL http://arxiv.org/abs/1709.07871.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv:1706.01427 [cs]*, June 2017. URL http://arxiv.org/abs/1706.01427. arXiv: 1706.01427.

Joseph Suarez, Justin Johnson, and Fei-Fei Li. DDRprog: A CLEVR Differentiable Dynamic Reasoning Programmer. *arXiv:1803.11361 [cs]*, March 2018. URL http://arxiv.org/abs/1803.11361. arXiv: 1803.11361.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pp. 3104–3112, 2014.

Wei Wang, Ming Yan, and Chen Wu. Multi-Granularity Hierarchical Attention Fusion Networks for Reading Comprehension and Question Answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1705–1714, Melbourne, Australia, 2018. Association for Computational Linguistics. URL http://aclweb.org/anthology/P18-1158.

A Appendix

A.1 Additional Details

Dataset: To make negative examples in SQOOP challenging, we ensure that both X and Y of a question are always present in the associated image and that there are always distractor objects $\mathrm{Y}' \neq \mathrm{Y}$ and $\mathrm{X}' \neq \mathrm{X}$ such that $\mathrm{X}\,\mathrm{R}\,\mathrm{Y}'$ and $\mathrm{X}'\,\mathrm{R}\,\mathrm{Y}$ are both true for the image. These extra precautions guarantee that answering a question requires the model to locate all possible X and Y then check if any pair of them are in the relation R.

Hyperparameters: All models share the same stem architecture which is a CNN based architecture of 6 layers. Each layer is a $\mathrm{Conv} \rightarrow \mathrm{BatchNorm} \rightarrow \mathrm{ReLU}$ with a $\mathrm{MaxPool}$ after layers 1 and 3. The input to the stem is a $64 \times 64 \times 3$ image, and the feature dimension used throughout the stem is 64. All models are optimized using Adam Kingma & Ba (2014) with a learning rate of 3e-4, and with minibatches of size 128.

A.2 Generic Models

We consider four generic models in this paper: CNN+LSTM, FiLM, Relation Networks (RelNet), and Compositional Attention Networks (CAN). For CNN+LSTM, FiLM, and RelNet models, the question q is first encoded into a fixed-size representation h_q using a unidirectional LSTM network.

CNN+LSTM flattens the 3D tensor h_{x} to a vector and concatenates it with h_q to produce $h_{q\,\mathrm{x}}$.

$$h_{q\,\mathrm{x}} = [vec(h_{\mathrm{x}}); h_q] \tag{3}$$

RelNet uses a network g which is applied to all pairs of feature columns of h_{x} concatenated with the question representation h_q, all of which is then pooled to obtain $h_{q\,\mathrm{x}}$:

$$h_{q\,\mathrm{x}} = \sum_{i,j} g(h_{\mathrm{x}}(i), h_{\mathrm{x}}(j), h_q) \tag{4}$$

where $h_x(i)$ is the i-th feature column of h_x.

FiLM networks use N convolutional FiLM blocks applied to h_{x}. A FiLM block is a residual block (He et al., 2016) in which a feature-wise affine transformation (FiLM layer) is inserted after the 2nd convolutional layer. The FiLM layer is conditioned on the question at hand via prediction of the scaling and shifting parameters γ_n and β_n:

$$[\gamma_n; \beta_n] = W_q^n h_q + b_q^n \tag{5}$$

$$\tilde{h}_{q\,\mathrm{x}}^n = BN(W_2^n * ReLU(W_1^n * h_{q\,\mathrm{x}}^{n-1} + b_n)) \tag{6}$$

$$h_{q\,\mathrm{x}}^n = h_{q\,\mathrm{x}}^{n-1} + ReLU(\gamma_n \odot \tilde{h}_{q\,\mathrm{x}}^n \oplus \beta_n) \tag{7}$$

where BN stands for batch normalization, $*$ stands for convolution and $\odot$ stands for element-wise multiplications. $h_{q\,\mathrm{x}}^n$ is the output of the n-th FiLM block and $h_{q\,\mathrm{x}}^0 = h_{\mathrm{x}}$. The output of the last FiLM block $h_{q\,\mathrm{x}}^N$ undergoes an extra 1×1 convolution and max-pooling to produce $h_{q\,\mathrm{x}}$.

CAN networks of Hudson & Manning (2018) produces $h_{q\,\mathrm{x}}$ by repeatedly applying a Memory-Attention-Control (MAC) cell that is conditioned on the question through an attention mechanism. The CAN model is quite complex and we refer the reader to the original paper for details.

A.3 NMN Modules

As mentioned in the text, our experiments are performed with the Find module from Hu et al. (2017) and the $\mathrm{Residual}$ module from Johnson et al. (2017) with very minor modifications - we use 64 dimensional CNNs in our $\mathrm{Residual}$ blocks since our dataset consists of 64×64 images. The equations for the $\mathrm{Residual}$ module are as follows:

$$\theta = \emptyset, \tag{8}$$

$$\gamma = [W_1; b_1; W_2; b_2; W_3; b_3], \tag{9}$$

$$\tilde{h} = ReLU(W_3 * [h_l; h_r] + b_3), \tag{10}$$

$$f_{Residual}(\gamma, h_l, h_r) = ReLU(\tilde{h} + W_1 * ReLU(W_2 * \tilde{h} + b_2)) + b_1), \tag{11}$$

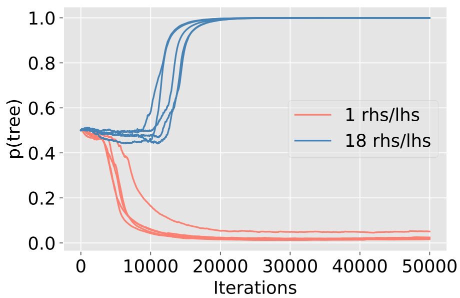


Figure 3: Learning dynamics of layout induction on 1 rhs/lhs and 18 rhs/lhs datasets using the $\mathrm{Residual}$ module with $p_0(tree) = 0.5$. All 5 runs of the model do not learn to use the tree layout for 1 rhs/lhs, the very setting where the tree layout is necessary for generalization.

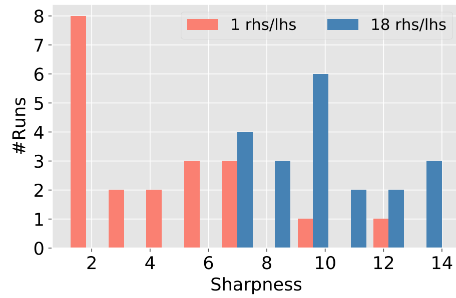


Figure 4: Histogram of sharpness (ρ) values for attention weights induced on the 1 rhs/lhs and 18 rhs/lhs datasets. We can observe that the attention is much sharper for 18 rhs/lhs.

and for Find module as follows:

$$\theta = [W_1; b_1; W_2; b_2], \tag{12}$$

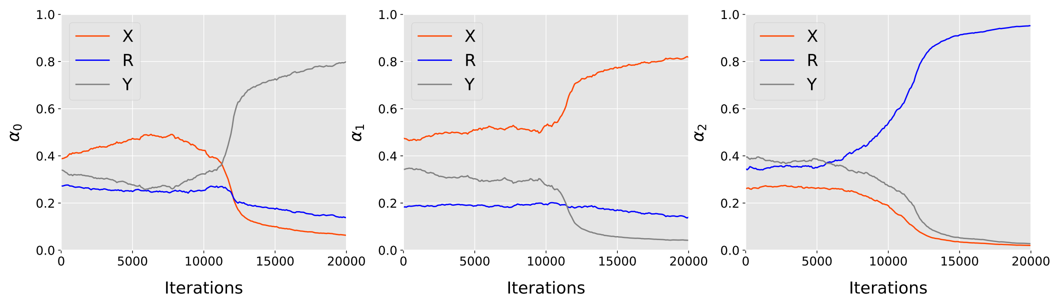
$$f_{Find}(\gamma, h_l, h_r) = ReLU(W_1 * \gamma \odot ReLU(W_2 * [h_l; h_r] + b_2) + b_1). \tag{13}$$

In formulas above W_1, W_2, W_3 are convolution weights, and b_1, b_2, b_3 are biases. The main difference between $\mathrm{Residual}$ and Find is that in $\mathrm{Residual}$ all parameters depend on the questions words, where as in Find convolutional weights are the same for all questions, and only the element-wise multipliers γ vary based on the question.

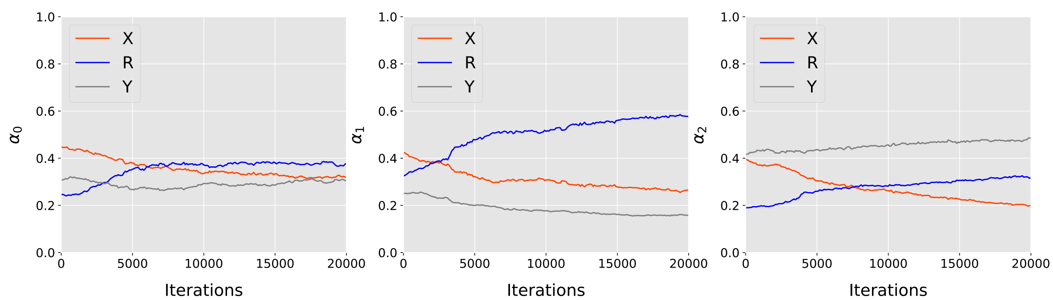
A.4 Additional Results

Structure Induction: We visualize the progress of structure induction for the $\mathrm{Residual}$ module with $p_0(tree) = 0.5$ in Figure 3. The figure shows $p(tree)$ saturates to 0.0 or 1.0 eventually in #rhs/lhs=1 and #rhs/lhs=18 settings respectively.

Parametrization Induction: Figure 5 shows how attention weights evolve for an Attention N2NMN model in the same context. It is notable that unlike in the gold-standard NMN-Tree model, the relation word is mixed with the object words for modules 1 and 2. We also noticed that the model did not learn to focus modules 1 and 2 on different words in the #rhs/lhs=1 setting (Figure 5b) as sharply as it did in #rhs/lhs=18 (Figure-5a). To substantiate this observation with quantitative results, we compute a sharpness ratio $\rho = \max(\alpha^{k,X}, \alpha^{k,Y})/\min(\alpha^{k,X}, \alpha^{k,Y})$ for modules $k = 1$ and $k = 2$ for each of the 20 modules that we have trained. One can observe from the histogram in Figure 4 that attention weights learnt on #rhs/lhs=1 are generally blurry, with ρ being less than 2 for 8 modules out of 20.



(a) 18 rhs/lhs



(b) 1 rhs/lhs

Figure 5: All three modules' attention weights for parametrization of the three question words for (a) 18 rhs/lhs and (b) 1 rhs/lhs version of SQOOP. The model learns to disentangle between X and Y much better with more rhs/lhs.