CLOSURE: Assessing Systematic Generalization of CLEVR Models

Dzmitry Bahdanau
Mila, Université de Montréal
Element AI

Harm de Vries
Element AI

Shikhar Murty
Stanford University

Philippe Beaudoin
Element AI

Yoshua Bengio
Mila, Université de Montréal
CIFAR Senior Fellow

Aaron Courville
Mila, Université de Montréal
CIFAR Fellow

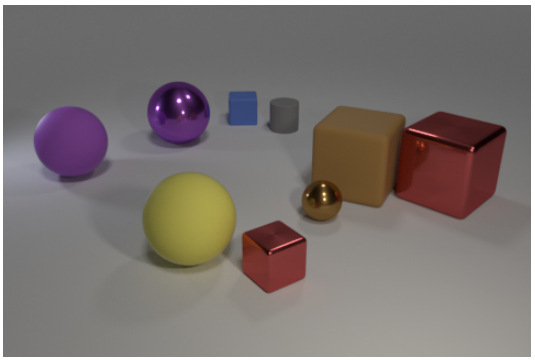
Abstract

The CLEVR dataset of natural-looking questions about 3D-rendered scenes has recently received much attention from the research community. A number of models have been proposed for this task, many of which achieved very high accuracies of around 97-99%. In this work, we study how systematic the generalization of such models is, that is to which extent they are capable of handling novel combinations of known linguistic constructs. To this end, we define 7 additional question families which test models' understanding of similarity-based references (such as e.g. "the object that has the same size as ...") in novel contexts. Our experiments on the thereby constructed CLOSURE benchmark show that state-of-the-art models often do not exhibit systematicity after being trained on CLEVR. Surprisingly, we find that the explicitly compositional Neural Module Network model also generalizes badly on CLOSURE, even when it has access to the ground-truth programs at test time. We improve the NMN's systematic generalization by developing a novel Vector-NMN module architecture with vector-shaped inputs and outputs. Lastly, we investigate the extent to which few-shot transfer learning can help models that are pretrained on CLEVR to adapt to CLOSURE. Our few-shot learning experiment contrast the adaptation behavior of the models with intermediate discrete programs with that of the end-to-end continuous models.

1 Introduction

Being able to communicate in natural language is a crucial skill that we expect from artificial agents of the future. Many of such intelligent agents will be robots that will perceive our rich unstructured 3D reality from raw sensory data. There has been a long line of work on developing agents capable of understanding language grounded in perception. A popular task to benchmark progress towards this goal is *Visual Question Answering* (VQA), in which one must give a (typically short) answer to a question about the content of an image. The release of the relatively large VQA 1.0 dataset by Agrawal et al. (2015) ignited the interest for the VQA setup, but researchers soon found that the biases of natural data (such as the heavily skewed answer distribution for certain question types) make it hard to interpret the VQA 1.0 results (Agrawal, Batra, and Parikh, 2016). To complement biased natural data, Johnson et al. (2016) constructed the CLEVR dataset of complex synthetic questions about 3D-rendered scenes to be free of such biases (see Q1 and Q2 in Figure 1 for examples of CLEVR questions). The CLEVR dataset has spurred VQA modeling research, and many models were designed for and showcased on it (Santoro et al., 2017; Perez et al., 2017a; Johnson et al., 2017; Hudson and Manning, 2018; Mascharka et al., 2018).

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



Q1 (CLEVR): There is another cube that is the same size as brown cube; what is its color?
Q2 (CLEVR): There is a thing that is in front of the yellow thing; does it have the same color as cylinder?
Q3 (CLOSURE): There is another rubber object that is the same size as the gray cylinder; does it have the same color as the tiny shiny block?

Figure 1: CLEVR questions (Q1 and Q2) require complex multi-step reasoning about the contents of 3D-rendered images. We construct CLOSURE questions (Q3) by using the similarity-based references (e.g. the red fragment in Q1) in novel contexts, such as e.g. comparison questions with two referring expressions (Q2).

The high complexity and diversity of CLEVR questions and the reported 97-99% accuracies may lead to the impression that these high-performing models are capable of answering any possible question that uses the same linguistic constructs as in CLEVR. Such an intuitive expectation corresponds to the concept of *systematicity* (Fodor and Pylyshyn, 1988), which characterizes the ability of humans to interpret arbitrary combinations of known primitives. One can argue that systematicity is also a highly desirable property for AI systems. For example, suppose you refer to an object by relating its appearance to another one, as in "the object that is the same size as ...". If a CLEVR model understands such reference, it is likely that you will expect this model to understand it in other, more complex contexts. This includes cases in which such reference is nested in a more complex referring expression, e.g. "the cylinder to the left of the object that has the same size as ...", or is an argument to a logical operation, e.g. "things that are either cubes or have the same size ...". For learning-based systems, unless the training distribution uniformly covers all sensible compositions of interest (which is nearly impossible to achieve for natural data), systematicity requires a particular kind of out-of-domain generalization, whereby the test distribution is different from the training one but follows the same rules of semantic and syntactic composition. Such *systematic generalization* of modern neural models has been recently studied in the context of artificial sequence transduction and VQA tasks (Lake and Baroni, 2018; Bahdanau et al., 2019), the latter done in a setup that is much simpler and less diverse than CLEVR.

In this work, we perform a case-study of how systematic CLEVR models are. In doing so, we seek to provide important context to the near-perfect CLEVR accuracies that are measured by the usual methodology, as well as to contribute to the literature on systematic generalization. The specific aspect of systematicity that we analyze is the one exemplified in the previous paragraph: the ability to interpret known ways of referring to objects in arbitrary contexts. We focus on the *similarity-based references* (see e.g. "another cube that it is the same size as the brown cube" in Figure 1), which in the context of CLEVR amounts to both objects having the same size, color, material or shape. We construct 7 new question families that highly overlap with CLEVR questions and yet are different (see the Q3 in Figure 1 for an example from one such family), aiming to cover all contexts in which object similarity is not used to refer to objects in the original CLEVR. We call the resulting benchmark CLOSURE referring to the underlying idea of taking a closure of CLEVR questions under the operation of reference substitution and keeping those questions that are similar enough to the original ones.

We evaluate a number of different CLEVR models on CLOSURE, including end-to-end differentiable ones, like FiLM (Perez et al., 2017a) and MAC (Hudson and Manning, 2018), and models using intermediate symbolic programs, like NS-VQA (Yi et al., 2018) and the variety of Neural Module

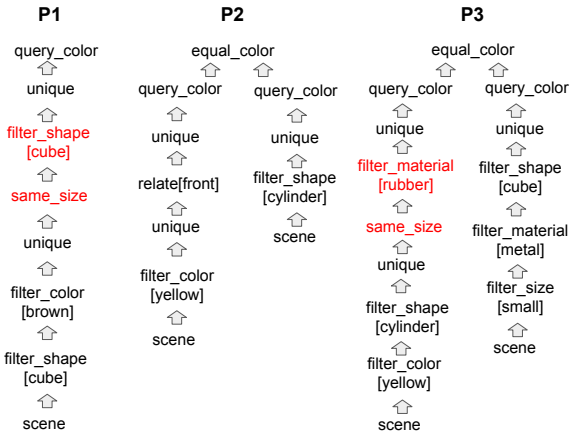


Figure 2: Programs P1, P2, P3 that define the ground-truth meaning for the questions Q1, Q2 and Q3 in Figure 1. The fragments in red correspond to the similarity-based references in the respective questions.

Networks (NMN, Andreas et al. (2016)) proposed by Johnson et al. (2017). We show that all aforementioned models often struggle on CLOSURE questions. For models using symbolic programs, we observe a generalization gap at both the program generation and the program execution stage when such execution is performed by the NMN. This result is remarkable, given that the original motivation for NMNs is to decompose the model into components that can be recombined arbitrarily. To improve the NMN's generalization, we develop a new Vector-NMN module with a vector-valued (as opposed to tensor-valued) inputs and outputs. We show that the Vector-NMN modules perform much better than prior work when assembled in configurations that are different from the training ones. Lastly, we complement our 0-shot systematic generalization analysis with a few-shot transfer learning study and contrast the few-shot adaptation behavior of models with and without symbolic programs.

2 CLOSURE: A Systematic Generalization Benchmark for CLEVR

Our CLOSURE benchmark tests whether models that are trained on the popular CLEVR dataset can generalize to questions that require recombining familiar components of CLEVR questions in novel contexts. In particular, CLOSURE tests if a model can apply its ability to understand similarity-based references to novel questions with different and more complex structures.

The CLEVR dataset contains 700K synthetic questions about 70K 3D-rendered scenes. The internal representation of the question's meaning is a program in a functional domain-specific language (DSL). See Figure 2 for example programs P1 and P2. The DSL functions operate on sets of objects, where each object is represented by the values of its four attributes (shape, color, size and material) and its spatial coordinates. We will focus on two kinds of the DSL operations: *filtering operations*, which filter the input set of objects by the value of an attribute (e.g. "filter_color[brown]", "filter_shape[cube]"), and *relational operations*, which given a single object, return a set of all other objects that are related to the given object. Two kinds of relations are supported. These are location relations: being to the left ("relate[left]"), to the right ("relate[right]"), in front ("relate[front]") or behind ("relate[behind]") of the given object; and similarity relations: having the same shape ("same_shape"), color ("same_color"), material ("same_material") or size ("same_size") as the given object. We will use the shortcuts "loc" and "sim" to refer to the two types of relational operations.

The programs and questions of the original CLEVR dataset come from 90 diverse question families, each generated from a question template. The question template defines the text of the question and the structure of the corresponding program, leaving a number of slots to be filled with specific filtering and location-relational operations (the similarity relations are fixed for each template). Given an image, these slots are synchronously filled in the question and in the program in a way that the resulting question is valid for the given image. We classify the question templates either as chain or tree-structured, following the topology of their programs. See programs P1 and P2 in Figure 2 for an

example of a chain and tree-structured program, respectively. The tree-structured templates contain 2 branches whose outputs are combined with a logical "or", a logical "and", or a comparison operator. We will use the shortcut names "chain", "and", "or" and "compare" to refer to these different template classes, as well as to the programs instantiated from them.

A notable asymmetry in the CLEVR dataset design[1] is that similarity-based relational operations only occur in chain-structured templates. Moreover, a similarity relation must be the only one in a chain-structured template. On the contrary, up to three location-based relational operations can be chained together. As a consequence of this asymmetry in the templates, only CLEVR questions with a single non-nested referring expression contain similarity-based references. This excludes questions with two referring expressions (i.e. with a tree-structured program) or a single nested referring expression (i.e. with chained relational operations in the programs). We construct CLOSURE by generating questions with these structures that also contain similarity-based references. To produce such questions, we compose seven new templates, aiming to exhaustively cover the combinations of the structural template classes ("chain", "or", "and", "compare") and the relation types ("loc", "'sim"). An example CLOSURE question generated from the "compare_sim" template is shown in Figure 1. We make sure that for each new template there exists a CLEVR template with the same structure and complexity that differs only in that a similarity relation is substituted for a location one. Below we list the exact definitions of all CLOSURE templates. Note that $\langle Z \rangle$, $\langle C \rangle$, $\langle M \rangle$, $\langle S \rangle$, and $\langle R \rangle$ are slots to be filled by specific size, color, material, shape, and location words respectively.

- **chain_sim_loc**: Is there a $\langle Z \rangle$ $\langle C \rangle$ $\langle M \rangle$ $\langle S \rangle$ $\langle R \rangle$ the $\langle C3 \rangle$ $\langle M3 \rangle$ $\langle S3 \rangle$ that is the same size as $\langle C2 \rangle$ $\langle M2 \rangle$ $\langle S2 \rangle$?
- **chain_loc_sim**: Is there anything else that is the same size as the $\langle C \rangle$ $\langle M \rangle$ $\langle S \rangle$ $\langle R \rangle$ the $\langle Z2 \rangle$ $\langle C2 \rangle$ $\langle M2 \rangle$ $\langle S2 \rangle$?
- **or_loc_sim**: How many things are [either] $\langle Z2 \rangle$ $\langle C2 \rangle$ $\langle M2 \rangle$ $\langle S2 \rangle$ [that are] $\langle R \rangle$ the $\langle Z \rangle$ $\langle C \rangle$ $\langle M \rangle$ $\langle S \rangle$ or $\langle Z4 \rangle$ $\langle C4 \rangle$ $\langle M4 \rangle$ objects that are the same shape as the $\langle Z3 \rangle$ $\langle C3 \rangle$ $\langle M3 \rangle$ object?
- **or_sim**: How many things are [either] $\langle Z \rangle$ $\langle C \rangle$ $\langle M \rangle$ $\langle S \rangle$s or $\langle Z3 \rangle$ $\langle C3 \rangle$ $\langle M3 \rangle$ objects that are the same shape as the $\langle Z2 \rangle$ $\langle C2 \rangle$ $\langle M2 \rangle$ object?
- **and_sim**: What is the shape of the $\langle C3 \rangle$ $\langle M3 \rangle$ $\langle S3 \rangle$ that is $\langle R2 \rangle$ the $\langle Z2 \rangle$ $\langle C2 \rangle$ $\langle M2 \rangle$ $\langle S2 \rangle$ and is the same size as the $\langle C \rangle$ $\langle M \rangle$ $\langle S \rangle$?
- **compare_sim_loc**: There is another $\langle M2 \rangle$ $\langle S2 \rangle$ that is the same size as the $\langle C \rangle$ $\langle M \rangle$ $\langle S \rangle$; does it have the same color as the $\langle Z4 \rangle$ $\langle M4 \rangle$ $\langle S4 \rangle$ [that is] $\langle R2 \rangle$ the $\langle Z3 \rangle$ $\langle C3 \rangle$ $\langle M3 \rangle$ $\langle S3 \rangle$?
- **compare_sim**: There is another $\langle M2 \rangle$ $\langle S2 \rangle$ that is the same size as the $\langle C \rangle$ $\langle M \rangle$ $\langle S \rangle$; does it have the same color as the $\langle Z3 \rangle$ $\langle M3 \rangle$ $\langle S3 \rangle$?

For all templates, the first part of the name identifies the structural template class and the remaining components define the types of relational operations used in the question. For questions with a nested referring expression, the location- and similarity-based relational operations can be chained in two different orders, resulting in two question families: "chain_loc_sim" and "chain_sim_loc".

3 Experiments on CLOSURE

See the Appendix for evaluation of state-of-the-art CLEVR models on CLOSURE.

[1] We are not aware of the reasoning behind this design decision, but we found it useful as it allowed us to define a systematic generalization test without changing the training settings.

References

Agrawal, A.; Batra, D.; and Parikh, D. 2016. Analyzing the Behavior of Visual Question Answering Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Agrawal, A.; Lu, J.; Antol, S.; Mitchell, M.; Zitnick, C. L.; Batra, D.; and Parikh, D. 2015. VQA: Visual Question Answering. *arXiv:1505.00468 [cs]*. arXiv: 1505.00468.

Agrawal, A.; Kembhavi, A.; Batra, D.; and Parikh, D. 2017. C-VQA: A Compositional Split of the Visual Question Answering (VQA) v1.0 Dataset. *arXiv:1704.08243 [cs]*. arXiv: 1704.08243.

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016. Neural Module Networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bahdanau, D.; Murty, S.; Noukhovitch, M.; Nguyen, T. H.; Vries, H. d.; and Courville, A. 2019. Systematic Generalization: What Is Required and Can It Be Learned? In *International Conference on Learning Representations*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 2015 International Conference on Learning Representations*.

Bastings, J.; Baroni, M.; Weston, J.; Cho, K.; and Kiela, D. 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*, 47–55. Brussels, Belgium: Association for Computational Linguistics.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bougares, F.; Schwenk, H.; and Bengio, Y. 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)*.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hu, R.; Andreas, J.; Rohrbach, M.; Darrell, T.; and Saenko, K. 2017. Learning to Reason: End-to-End Module Networks for Visual Question Answering. In *Proceedings of 2017 IEEE International Conference on Computer Vision*. arXiv: 1704.05526.

Hu, R.; Andreas, J.; Darrell, T.; and Saenko, K. 2018. Explainable Neural Computation via Stack Neural Module Networks. *arXiv:1807.08556 [cs]*. arXiv: 1807.08556.

Hudson, D. A., and Manning, C. D. 2018. Compositional Attention Networks for Machine Reasoning. In *Proceedings of the 2018 International Conference on Learning Representations*.

Ioffe, S., and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 448–456.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. 2016. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv: 1612.06890.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Hoffman, J.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. 2017. Inferring and Executing Programs for Visual Reasoning. In *Proceedings of 2017 IEEE International Conference on Computer Vision*.

Kuhnle, A., and Copestake, A. 2017. ShapeWorld - A new test methodology for multimodal language understanding. *arXiv:1704.04517 [cs]*. arXiv: 1704.04517.

Lake, B. M., and Baroni, M. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 36th International Conference on Machine Learning*. arXiv: 1711.00350.

Loula, J.; Baroni, M.; and Lake, B. M. 2018. Rearranging the Familiar: Testing Compositional Generalization in Recurrent Networks. In *Proceedings of the 2018 BlackboxNLP EMNLP Workshop*.

Mascharka, D.; Tran, P.; Soklaski, R.; and Majumdar, A. 2018. Transparency by Design: Closing the Gap Between Performance and Interpretability in Visual Reasoning. *arXiv:1803.05268 [cs]*. arXiv: 1803.05268.

Perez, E.; de Vries, H.; Strub, F.; Dumoulin, V.; and Courville, A. 2017a. Learning Visual Reasoning Without Strong Priors. *arXiv:1707.03017 [cs, stat]*. arXiv: 1707.03017.

Perez, E.; Strub, F.; de Vries, H.; Dumoulin, V.; and Courville, A. 2017b. FiLM: Visual Reasoning with a General Conditioning Layer. In *In Proceedings of the 2017 AAAI Conference on Artificial Intelligence*.

Russin, J.; Jo, J.; and O'Reilly, R. C. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv:1904.09708 [cs, stat]*. arXiv: 1904.09708.

Santoro, A.; Raposo, D.; Barrett, D. G. T.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems 31*. arXiv: 1706.01427.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, 3104–3112.

Yi, K.; Wu, J.; Gan, C.; Torralba, A.; Kohli, P.; and Tenenbaum, J. B. 2018. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. *arXiv:1810.02338 [cs]*. arXiv: 1810.02338.

A Models

A large number of models for the CLEVR task have been recently proposed, and it would be impossible for us to evaluate all of them. We therefore choose several models that vary in how CLEVR-specific their design is, aiming to cover the whole spectrum of "CLEVR-awareness" that such models possess. In addition to existing models, we experiment with a novel Vector-NMN neural module that we employ in the context of the Neural Module Network paradigm.

Throughout this section we use capital letters for matrix- or tensor-shaped parameters of all models and small letters for the vector-shaped ones. We use $*$ to denote convolution as well as to inform the reader that the symbols on the left and right sides of the operator are a 4D and a 3D tensor respectively. $\odot$ and $\oplus$ are used to denote element-wise multiplication and addition for the case where the left argument is a tensor and the right is a vector. The respective operation is applied independently to all subtensors of the left argument obtained by fixing all its indices but for the last one (the approach known as "broadcasting"). We will use square brackets $[x; y]$ to denote tensor concatenation performed along the last dimension.

A.1 Generic Models

The most generic method that we consider is Feature-wise Linear Modulation (**FiLM**) by Perez et al. (2017b). In this approach, an LSTM recurrent network transforms the question q into biases β and element-wise multipliers α that are then applied in the blocks of a deep residual convolutional network (He et al., 2016). A FiLM-ed residual block takes a tensor-valued input h_{in} and performs the following computation upon it:

$$[\gamma; \beta] = W \cdot LSTM(q) + b, \tag{1}$$

$$\tilde{h} = BN(W_2 * R(W_1 * h_{in} \oplus b_1)), \tag{2}$$

$$h_{out} = h_{in} + R(\gamma \odot \tilde{h} \oplus \beta), \tag{3}$$

where R stands for the Rectified Linear Unit, BN denotes batch normalization (Ioffe and Szegedy, 2015). Several such blocks are stacked together and applied to a 3D feature tensor h_x that is produced by several layers of convolutions, some of them pretrained. The FiLM-ed network thereby processes the input image x in a manner that is modulated by the question q. Despite its simplicity, FiLM achieves a remarkably high reported accuracy of 97.7% on the CLEVR task.

A more advanced model that we include in our evaluation is Memory-Attention-Composition (**MAC**) by Hudson and Manning (2018). In the MAC approach, the input and control components of the model first produce a sequence of control vectors c_i from the question q. A visual attention component (called the read unit in the original paper) is then recurrently applied to a preprocessed version h_x of the image x. The i-th application of the read unit is conditioned on the respective control vector c_i and on a memory m_i of the unit's outputs at the previous steps:

$$r_i = read\_unit(h_x, c_i, m_{i-1}), \tag{4}$$

$$m_i = memory\_unit(r_i, m_{i-1}). \tag{5}$$

Such read operations and memory updates are performed for T steps, after which the last memory vector m_T and a question representation q are concatenated and passed to the classifier. Different versions of the MAC model reach near-perfect 98.9-99.4% performance on CLEVR.

A.2 Modular and Symbolic Approaches

In addition to the end-to-end differentiable models, we experiment with methods that rely on intermediate structured symbolic meaning representations. We adhere to the common practice of using programs expressed in the CLEVR DSL as such representations, although in principle logical formulae or other formalisms from the field of formal semantics could be used for this purpose. Under the assumption that a symbolic *execution engine* for the programs is available, the task of VQA can be reduced to parsing the question and the image into a program and a symbolic scene representation respectively. Such an approach has been proposed by Yi et al. (2018) under the name Neural-Symbolic VQA (**NS-VQA**) with a reported CLEVR accuracy of 99.8%. This excellent performance, however, is achieved by relying heavily on the prior knowledge about the task, meaning

that applying NS-VQA in conditions other than CLEVR could require significantly more adaptation and data collection than needed for the more generic methods, such as FiLM and MAC.

Intermediate symbolic programs can also be used without apriori knowledge of the semantics of the symbols, in which case the execution engine for the programs is either fully or partially learned. In the Neural Module Network (**NMN**) paradigm, proposed by Andreas et al. (2016), the meanings of symbols are represented in the form of trainable neural modules. Given a program, the modules that correspond to the program's symbols are retrieved and composed following the program's structure. Formally, a program in CLEVR DSL can be represented as a (P, L, R) triple, where $P = (p_1, p_2, \ldots, p_T)$ is the sequence of function tokens[2], $L = (l_1, \ldots, l_T)$ and $R = (r_1, \ldots, r_T)$ are the indices of the left and right arguments for each function call respectively (some DSL functions only take one or zero arguments, in which case the respective r_i and l_i are undefined). Using this formalism, a step of the NMN computation can be expressed as follows:

$$h_i = \begin{cases} M_{p_i}(h_x), & arity(p_i) = 0, \\ M_{p_i}(h_x, h_{l_i}), & arity(p_i) = 1, \\ M_{p_i}(h_x, h_{l_i}, h_{r_i}), & arity(p_i) = 2. \end{cases} \tag{6}$$

Here, M_{p_i} is the neural module corresponding to the function token p_i and h_i is its output, while h_x is a tensor representing the image. Similarly to MAC, the output h_T of the last module is fed to the classifier, after which the modules are jointly trained by backpropagating the classifier's loss.

A number of NMN-based approaches have been proposed for the CLEVR task, including those where different modules perform different operations (e.g. the module corresponding to logical "and" might compute an element-wise maximum of two vectors (Hu et al., 2017; Mascharka et al., 2018)), and those where all modules perform similar computations but use different parameters (Johnson et al., 2017). We focus on the latter variety of NMNs, since such models rely less on the domain knowledge and thus complement well the NS-VQA approach in our evaluation. In both cases, a *program generator* can be pretrained with a small seed set of (question, program)-pairs and then fine-tuned, e.g. with REINFORCE (Hu et al., 2017; Johnson et al., 2017), on the rest of the dataset, using only (image, question, answer)-triplets as supervision. The programs produced by such a program generator can then be used at test time, meaning that after training the complete model takes the same inputs as end-to-end continuous models, such as FiLM and MAC.

The first NMN model that we consider is the one proposed by Johnson et al. (2017), in which residual blocks (He et al., 2016) are used as neural modules M_{p_i}. For example, modules corresponding to functions of arity 2, (such as e.g. "and", "equal_color", etc.), perform the following computation in their approach:

$$h_{proj} = R(W_1 * [h_{l_i}; h_{r_i}]), \tag{7}$$

$$\tilde{h} = R(W_2 * h_{proj} \oplus b_2), \tag{8}$$

$$h_i = R(W_3 * \tilde{h} \oplus b_3) + h_{proj}. \tag{9}$$

Note that the module described above does not use the image representation h_x as an input; only the M_{scene} module—the root node in all CLEVR programs—does so.

Our preliminary experiments showed that such modules often perform much worse when assembled in novel combinations. We hypothesized that this could be due to the fact that high-capacity 3D tensors h_i are used in this model as the interface between modules. In order to test this hypothesis, we have designed a new module with a lower-dimensional vector output. We will henceforth refer to the module by Johnson et al. (2017) and our new module as **Tensor-NMN** and **Vector-NMN** respectively. The computation of our Vector-NMN is inspired by the FiLM approach to conditioning residual blocks on external inputs:

$$\tilde{h}_1 = R(U_1 * (\gamma_1 \odot h_x \oplus \beta_1)), \tag{10}$$

$$\tilde{h}_2 = R(U_2 * (\gamma_2 \odot \tilde{h}_1 \oplus \beta_2) + h_x), \tag{11}$$

$$h_i = \text{maxpool}(\tilde{h}_2), \tag{12}$$

[2]In this work we treat composite functions like e.g. "filter_color[brown]" as standalone ones, not as "filter_color" parameterized by "brown"

where "maxpool" denotes max pooling of the 3D-tensor across all locations. The above equations describe a 1-block version of Vector-NMN, but in general several FiLM-ed residual blocks described by Equations 10 and 11 can be stacked prior to the max-pooling. The FiLM coefficients β_1, β_2, γ_1 and γ_2 are computed with 1-hidden-layer MLPs from the concatenation $h_{cond} = [e(p_i); h_{l_i}; h_{r_i}]$ of the embedding $e(p_i)$ of the function token p_i and the module inputs h_{l_i} and h_{r_i}

$$[\beta_k, \tilde{\gamma_k}] = W_2^k(R(W_1^k h_{cond} + b_1^k) + b_2^k), \tag{13}$$

$$\gamma_k = 2\tanh(\tilde{\gamma}_k) + 1. \tag{14}$$

The extra $\tanh$ nonlinearity in Equation 14 was required to achieve stable training. Note that unlike in Tensor-NMN, the convolutional filters U_1 and U_2 are reused among all modules. To make this possible, we feed zero vectors instead of h_{r_i} or h_{l_i} when the function p_i takes less than two inputs.

B Experiments

We use the original implementation for FiLM and Tensor-NMN and train these models with the hyperparameter settings suggested by the authors. For the MAC model, we use a PyTorch reimplementation by Bahdanau et al. (2019) that is close to the original one. We report results for a 2-block version of Vector-NMN, as our preliminary experiments on the original CLEVR dataset showed that it performs better than the 1-block version.

For all models that rely on symbolic programs, i.e. NS-VQA and the NMNs, we use a standard seq2seq model with an attention mechanism (Bahdanau, Cho, and Bengio, 2015) as the program generator. Our preliminary investigations showed that this model generalizes better than the seq2seq models without attention (Sutskever, Vinyals, and Le, 2014; Cho et al., 2014), as used in (Johnson et al., 2017), and better than the seq2seq model used in the reference implementation of NS-VQA, in which the decoder does not take the attention outputs as inputs. We report results for program generators trained on the (question, program)-pairs from all 700k CLEVR examples[3]. In addition to evaluating the NMNs with the predicted programs, we also measure their performance when the ground-truth programs are given at test time. For the latter setting, we prepend **GT** to the model's name (as in GT-Vector-NMN), as opposed to prepending **PG** (as in PG-Vector-NMN).

All numbers that we report are averages over 5 or 10 runs. Where relevant, we also report the standard deviation σ in the form of $\pm\sigma$ or as a vertical black bar in figures.

B.1 Zero-shot Generalization

In our first set of experiments, we assess 0-shot systematic generalization of models trained on CLEVR by measuring their performance on CLOSURE question families. We find that on average, all models perform much worse on CLOSURE than they do on CLEVR (see Figure 3). Surprisingly, a performance drop is observed even for the NS-VQA model, whose only learnable component is the program generator. To put these 0-shot generalization results in context, we retrain the same models from scratch on an extended version of CLEVR to which we add 8K examples from each CLOSURE family. This simulates the original composition of CLEVR, whereby each of the 90 original question families is represented with 8K examples in the training set. After retraining on the extended dataset, the performance of all models goes up and gets close to their performance on CLEVR (see Figure 3), showing that the models are, in fact, capable of the reasoning that CLOSURE requires, but do not perform it well after training on CLEVR only.

To get a better understanding of the generalization challenge that CLOSURE represents, we report results for all models on all 7 question families in Figure 5. We find that for all models, the chain-structured question families "chain_loc_sim" and "chain_sim_loc" (see Section 2 for the corresonding templates) are easier to generalize to, as opposed to questions with logical operations ("and_sim", "or_sim" and "or_loc_sim"). Notably, comparison question families "compare_sim" and "compare_loc_sim" champion the generalization abilities of models with symbolic programs as compared to the end-to-end continuous models.

[3]Similarly to prior work we found that pretraining on as few as 300-1000 ground-truth programs, followed by REINFORCE finetuning, is sufficient to achieve near-perfect program generation performance. We chose, however, to use all available data to keep the study focused on systematic generalization.

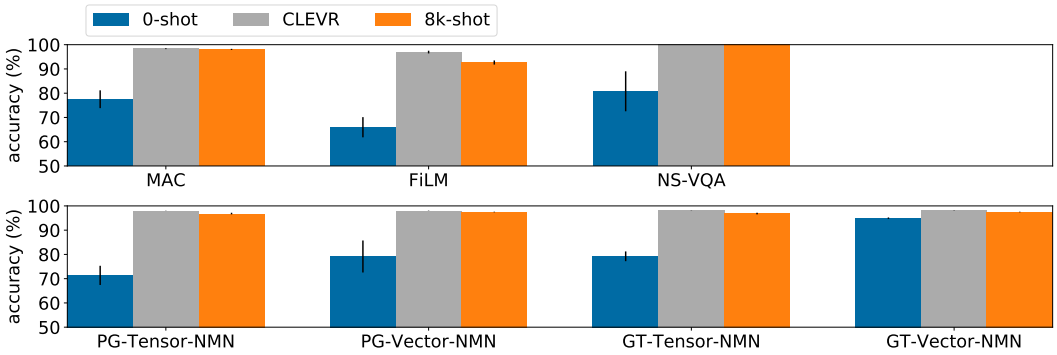


Figure 3: Average accuracy of all models on the 7 CLOSURE question families. For comparison, we also plot the models' performance on CLEVR as well as the their accuracies on CLOSURE when 8K questions from each CLOSURE family were used for training. The hatching is used for "GT-..." models to emphasize that the ground-truth programs were used at test time.

B.2 Tensor-NMN vs Vector-NMN

One surprising finding of our experiments is that the Tensor-NMN model often generalizes badly even when the ground-truth programs are given at test time (see the hatched columns in Figure 5). This phenomenon is especially pronounced for question families with logical operations, on which the accuracy ranges from $\sim 40\%$ to $\sim 70\%$, but can also be observed for the chain-structured "chain_loc_sim" questions, on which the model reaches only 83.2% accuracy on average. In comparison, our new Vector-NMN architecture fares much better on all the aforementioned tests. The better generalization ability of Vector-NMN is however not perfect. In Figure 4 we compare 0-shot and 8K-shot results for Vector-NMN and find that for the questions with logical operations, 0-shot accuracy of Vector-NMN is between $\sim 3\%$ and $\sim 9\%$ lower than what it could be.

B.3 Few-shot transfer learning

The results above show that generalization abilities of existing models are often not sufficient for 0-shot systematic generalization. A natural question to ask in these circumstances is whether just a few examples would be sufficient to correct models' extrapolation behavior. To answer this question, we finetune MAC, PG-Vector-NMN and NS-VQA models that are pretrained on CLEVR using 30 examples from each CLOSURE family, for a total of 210 new examples. For PG-Vector-NMN and NS-VQA we consider two fine-tuning scenarios: one where the programs are provided for new examples and one where they are not given and must be inferred, for which purpose we are using a basic REINFORCE-based program induction approach (Johnson et al., 2017; Hu et al., 2017). We will refer to the two said scenarios as strong and weak supervision respectively. To get the best finetuning performance we oversample 300 times the resulting 210 CLOSURE examples, add them to the CLEVR training set and train on the resulting mixed dataset. We report the *generalization ratio* ρ which we define as $\rho = a_{30}/a_{8K}$, where a_{30} is the best validation accuracy achieved by the fune-tuning described above, and a_{8K} is the best validation accuracy of the model trained on CLEVR that is extended with 8K examples from each CLOSURE family.

The few-shot results, reported in Figure 6, show that as few as 30 examples from each family can significantly improve CLOSURE performance for all models. A notable exception from this general observation is the "and_sim" question family, on which weakly supervised program induction for NS-VQA and PG-Vector-NMN most often did not work. We analyzed this case in detail and found that the appropriate programs for "and_sim" would typically have a very low probability, and hence were never sampled in our REINFORCE-based program search.

A closer analysis reveals that the impact of 30 examples varies widely depending on the model and on the test. For chain-structured and comparison questions the models using symbolic programs have consistently achieved the target 8K-example performance, whereas for MAC the generalization ratio was significantly below 1, namely 0.87 ± 0.14, 0.9 ± 0.12 and 0.96 ± 0.03 for "compare_sim", "compare_loc_sim" and "chain_loc_sim" respectively. For logical "or" questions, the generalization

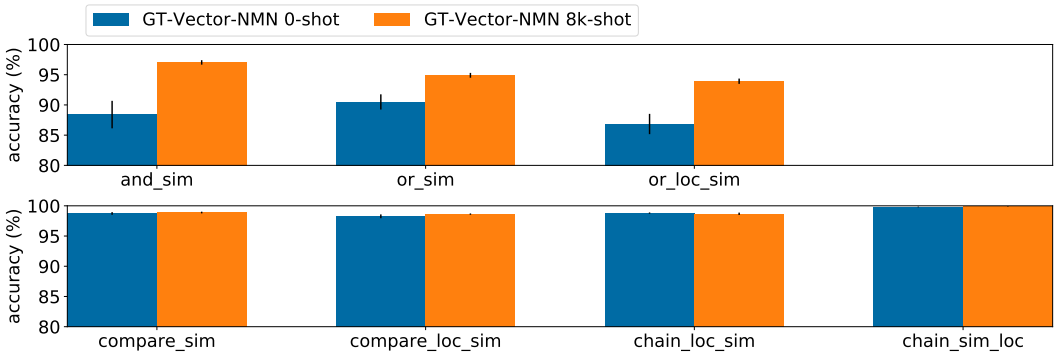


Figure 4: 0-shot accuracy of the new Vector-NMN model when the ground-truth programs are used at test time. For comparison, we also plot the accuracy for the same model retrained on CLEVR that was extended with 8K CLOSURE questions per family.

ratios of PG-Vector-NMN and MAC are similar. The logical "and" questions champion MAC: not only is it the only model that is able to finetune without a stronger form of supervision, but also when such extra supervision is given to PG-Vector-NMN, the generalization ratio of that model is still lower than MAC's.

Another interesting observation that can be made from our 30-shot results is how program induction is sensitive to the performance of the execution engine. With the perfect symbolic execution engine of NS-VQA, whenever the right programs are found during program search (except for the case of "and_sim"), the system ultimately reaches a near perfect performance. To the contrary, in the case of PG-Vector-NMN we observe that weakly-supervised induction finds wrong programs that give the correct answer due to the imperfection of the Vector-NMN execution engine. The resulting overfitting on the training set shows itself in the gap of ~ 5 generalization points between the strong and weak supervision results of this model on the "or_loc_sim" test.

C Related Work

Several related generalization tests that were proposed for CLEVR and other VQA datasets differ from CLOSURE in what they aim to measure and/or how they were constructed. The compositional generalization test (CLEVR-CoGenT) from original paper by Johnson et al. (2016) restricts the colors that cubes and cylinders can have in the training set images and inverts this restriction during the test time. By its design, CoGenT evaluates how robust a model is to a shift in the image distribution. On the contrary, in CLOSURE the image distribution remains the same at test time, but the question distribution changes to contain novel combinations of linguistic constructs from CLEVR. The generalization splits from the ShapeWorld platform (Kuhnle and Copestake, 2017) also focus on the difference in the distribution of images, not questions. The CLEVR-Humans dataset was collected by having crowd workers ask questions about CLEVR images (Johnson et al., 2017). Some questions from this dataset require reasoning that is outside of the scope of CLEVR, such as e.g. quantification ("Are all the balls small?"). In contrast, CLOSURE requires models to recombine only the well-known reasoning primitives. A compositional C-VQA split was proposed for the VQA 1.0 dataset (Agrawal et al., 2017). In C-VQA similar questions must have different answers when they appear in the training and test sets, yet the distributions of questions at training and testing remain similar, unlike CLOSURE.

Perhaps the closest to our work is the SQOOP dataset and the study conducted on it by Bahdanau et al. (2019). SQOOP features questions of the form "Is there an X R of Y", where X and Y are object words and R a location relation. The authors test whether models can answer all possible SQOOP questions after training on a subset that is defined by holding out most of the (X, Y) pairs. The methodology of that study is thus very similar to ours, however the specific nature of the generalization split is different. Similarly to our results, Bahdanau et al. (2019) report significant generalization gaps for a number of VQA models, with a notable exception of the Tensor-NMN, that generalized perfectly in their study when a tree-like layout was used to connect the modules. Our CLOSURE results are an important addition to the SQOOP ones, because the CLEVR questions (and consequently, the

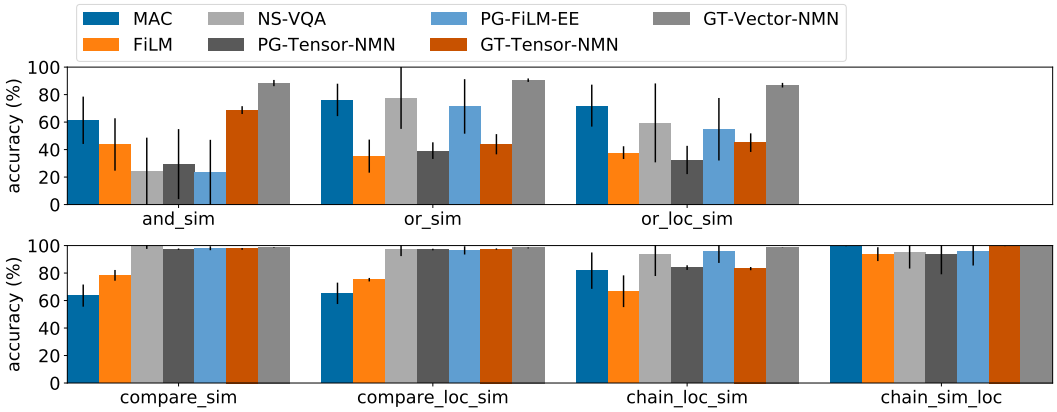


Figure 5: 0-shot accuracy of all models on the 7 CLOSURE question families. The hatching used for "GT-..." models indicates that we used the ground-truth programs at test time.

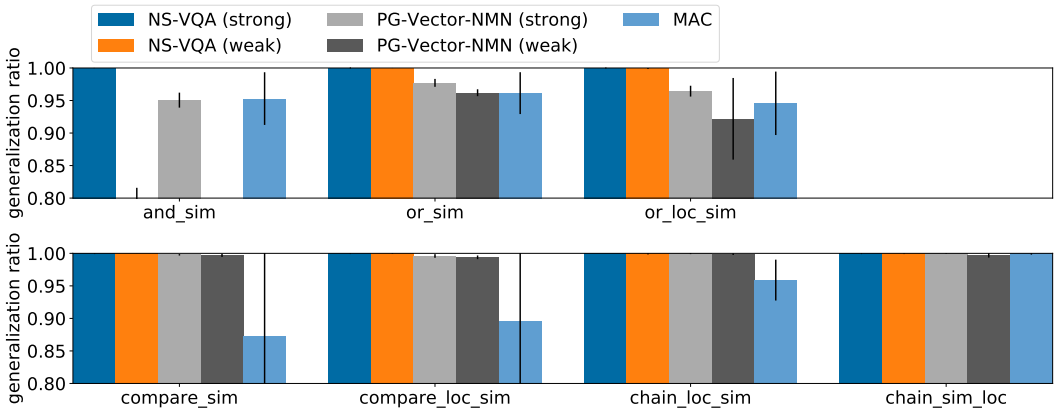


Figure 6: The generalization ratio ρ for NS-VQA, PG-FiLM-EE and MAC after finetuning on 30 examples from each CLOSURE family. See Section B.3 for the definition of ρ and details on how fine-tuning was performed.

CLOSURE questions) are much more complex and diverse than the SQOOP ones. The specific cause of the aforementioned discrepancies between the two studies is, however, an intriguing question for future work.

As can be clearly seen from the performance of the NS-VQA model, much of the performance drop that we reported can be explained by insufficient systematicity of seq2seq models that we use for program generation. The SCAN dataset (Lake and Baroni, 2018) and the follow-up works (Loula, Baroni, and Lake, 2018; Bastings et al., 2018) have recently brought much-needed attention to this important issue. Compared to SCAN, CLOSURE features richer and more natural-looking language, and hence can serve to validate the conclusions drawn in recent SCAN-based studies, e.g. (Russin, Jo, and O'Reilly, 2019).

Prior work on Neural Module Networks features modules that output either attention maps (Andreas et al., 2016; Hu et al., 2017, 2018) or feature tensors (Johnson et al., 2017). The model by (Mascharka et al., 2018) combines modules with both attention- and tensor-valued outputs. Our Vector-NMN generalizes more systematically than its tensor-based predecessor by Johnson et al. (2017), while inheriting that model's simplicity, generality and good performance.

D Discussion

Our study shows that while models trained on CLEVR are very good at answering questions from CLEVR, their high performance quickly deteriorates when the question distribution features unfamil-

iar combinations of well-known primitives. We believe that this is an interesting finding, given that CLEVR puts VQA models in very favorable conditions: the training set is large and well-balanced and complex questions are well represented. One could say that we took advantage of a gap in the CLEVR question distribution to make our point, yet we believe that natural language datasets collected under naturalistic conditions will only have more gaps like this.

While in our 0-shot test of systematic generalization all models fare similarly badly, our few-shot learning study highlights important differences in their behavior. Given few examples, the program-based models either almost perfectly adapt to the target task or completely fail, depending on whether the right programs are found. For end-to-end continuous models back-propagation is always effective in adapting them to the target examples, but the generalization gap is often not fully bridged with a number of examples that is sufficient for the program-based models. An important context for this comparison is that program-based models require seed programs to jump-start the training and are later on constrained to the seed lexicon. It would be highly desirable to combine the strengths of these two types of systems in one model without inheriting any of their limitations, a direction that we would like to explore in our future work.

We hope that the CLOSURE benchmark will facilitate future work in a number of directions. First, our results suggest that parsing (program generation in our case) can be the bottleneck for systematic generalization of grounded language learning. CLOSURE can thus be used for testing systematic generalization of neural parsers, complementing the SCAN benchmark and its variants. Second, while Vector-NMN improves upon prior work, it does not generalize perfectly. Further work on generic trainable interchangeable modules can thus be done using CLOSURE. Lastly, our test set construction methods can be adapted to natural data, yielding more insights and helping researchers make measurable progress towards learning-based models for grounded language understanding that generalize systematically.