How Modular should Neural Module Networks Be for Systematic Generalization?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural Module Networks (NMNs) aim at VQA via composition of modules that tackle a sub-task. NMNs are a promising strategy to achieve systematic generalization, *ie.* overcoming biasing factors in the training distribution. However, the aspects of NMNs that facilitate systematic generalization are not fully understood. In this paper, we demonstrate that the stage and the degree at which modularity is defined has large influence on systematic generalization. In a series of experiments on three VQA datasets (MNIST with multiple attributes, SQOOP, and CLEVR-CoGenT), our results reveal that tuning the degree of modularity in the network, especially at the image encoder stage, reaches substantially higher systematic generalization. These findings lead to new NMN architectures that outperform previous ones in terms of systematic generalization.

1 Introduction

The combinatorial nature of vision [1] is an open challenge for learning machines tackling Visual Question Answering (VQA) [2, 3, 4, 5, 6, 7]. The amount of possible combinations of object categories, attributes, relations, context and visual tasks is exponentially large, and any training set contains a limited amount of such combinations. The ability to generalize to combinations not included in the training set, *ie.* the so-called *systematic generalization* [8, 9, 10], is a hallmark of intelligence. Systematic generalization can be evaluated with the out-of-distribution (OOD) accuracy using examples that contain combinations not included in the training set, as in the example in Fig. 1a [8, 4, 2, 11, 5, 12, 13].

Recent research has shown that Neural Modular Networks (NMNs) are a promising approach in the context of systematic generalization in VQA [11, 5]. NMNs infer a program layout containing a sequence of sub-tasks, where each sub-task is associated to a module consisting of a neural network [14, 15].

The aspects that facilitate systematic generalization in NMNs are not fully understood. Bahdanau *et al.* [11] has shown the importance of using an appropriate program layout to solve a VQA task. Also, Bahdanau *et al.* [5] in another paper reports that the architecture of the NMN's module can impact systematic generalization. These works highlight the relevance of evaluating systematic generalization alongside with in-distribution generalization, as many VQA approaches achieve high generalization only when tested in-distribution.

In this paper, we investigate what sub-tasks a module should perform in order to facilitate systematic generalization, *ie.* we ask how modular should NMNs be. We analyse different degrees of modularity as in the example of Fig. 2b, where NMNs could be composed by a unique module for extracting color and object category, or by a module for extracting color and another one for extracting category, or by many modules each one for detecting a specific color or category. Also, we investigate the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

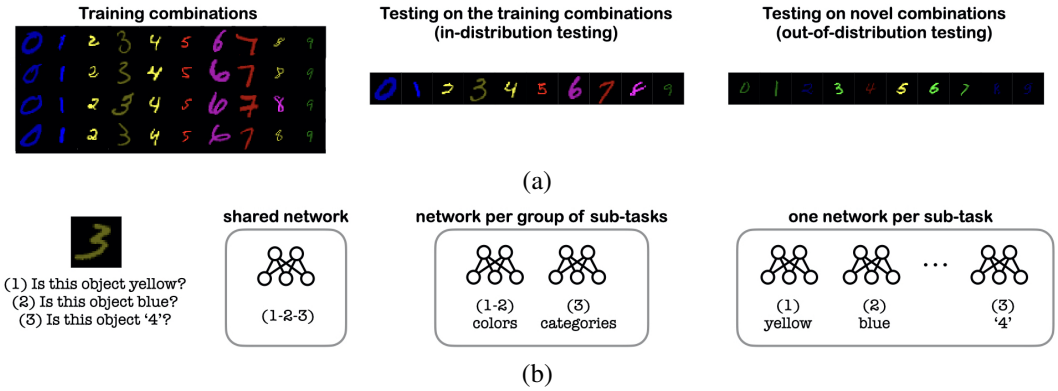

Figure 1: (a) Example of datasets to measure in-distribution and out-of-distribution (systematic) generalization: Given an attribute extraction task (category, color, size, light condition) the digits appears in a limited amount of combinations for those attributes during training, on the left. A model can be tested on an in-distribution test set, with data points identically distributed to training set, at the center, or on an out-of-distribution test set, on the right. (b) Different architectural choices given a VQA task: a single network for all the sub-tasks on the left, a set of networks each tackling on group of sub-tasks at the center, a network per sub-task on the right.

degree of modularity at all the stages of the network, as NMNs have a structure divided in three stages as in Fig. 2a: an *image encoder* for the extraction of visual features, a set of *intermediate modules*, selected based on program layout, and a *classifier*.

We examined three families of VQA datasets: (i) Multi-attribute MNIST, (ii) the SQOOP dataset [11], and (iii) the Compositional Generalization Test (CoGenT) of CLEVR [2]. In Multi-attribute MNIST, we generate four different VQA tasks, two for attribute extraction and two for comparison where we limit the combinations of objects and attributes during training. In the SQOOP dataset, we introduce a bias in the co-occurrence of objects in questions and scenes. In the CLEVR-CoGenT, limited combinations of shapes and colors appear during training in complex 3D scenes, containing multiple objects. This collection of datasets provides a comprehensive evaluation of systematic generalization that highlights the generality of our results.

Our results reveal two main findings that are consistent across the tested datasets: (i) an intermediate level of modularity by grouping sub-tasks leads to much higher test accuracy on novel combinations than for the standard NMNs' modules definition, and (ii) modularity is mostly effective when imposed at the image encoder stage, which is rarely done in the literature. Furthermore, we show that these findings are directly applicable to improve the systematic generalization of state-of-the-art NMNs.

2 Inducing Modularity in Neural Module Networks

NMNs consist of two trainable components: a program generator and an execution part. The former translates the VQA question into a program layout, consisting of a sequence of connected sub-tasks. The execution part implements the network by composing the image encoder, the sequence of sub-tasks of the intermediate modules stage, and the classifier, as in Fig. 2a. The use of a correct program layout plays a fundamental role in solving VQA with NMNs [2, 11, 5]. To study the aspects of NMNs that facilitate systematic generalization, we follow the approach in previous works that fix the program generator to the ground-truth program layout, such that it does not interfere in the analysis [11, 5]. This reductionist approach facilitates gaining an understanding of each individual aspect of NMNs, and as we gain such understanding, we will be in a position to study the interactions between these in future works.

Fig. 2a shows four NMNs implementations where we define modularity at different stages and in different degrees. Each row identifies a stage in the network, while the use of single or multiple modules identifies the degree of modularity for each stage. In particular, a single module shared across all sub-tasks is expressed by a single block named *all*. The case of grouped sub-tasks is pictured as a group of blocks named *group*, while a module for each single sub-task are shown through the largest

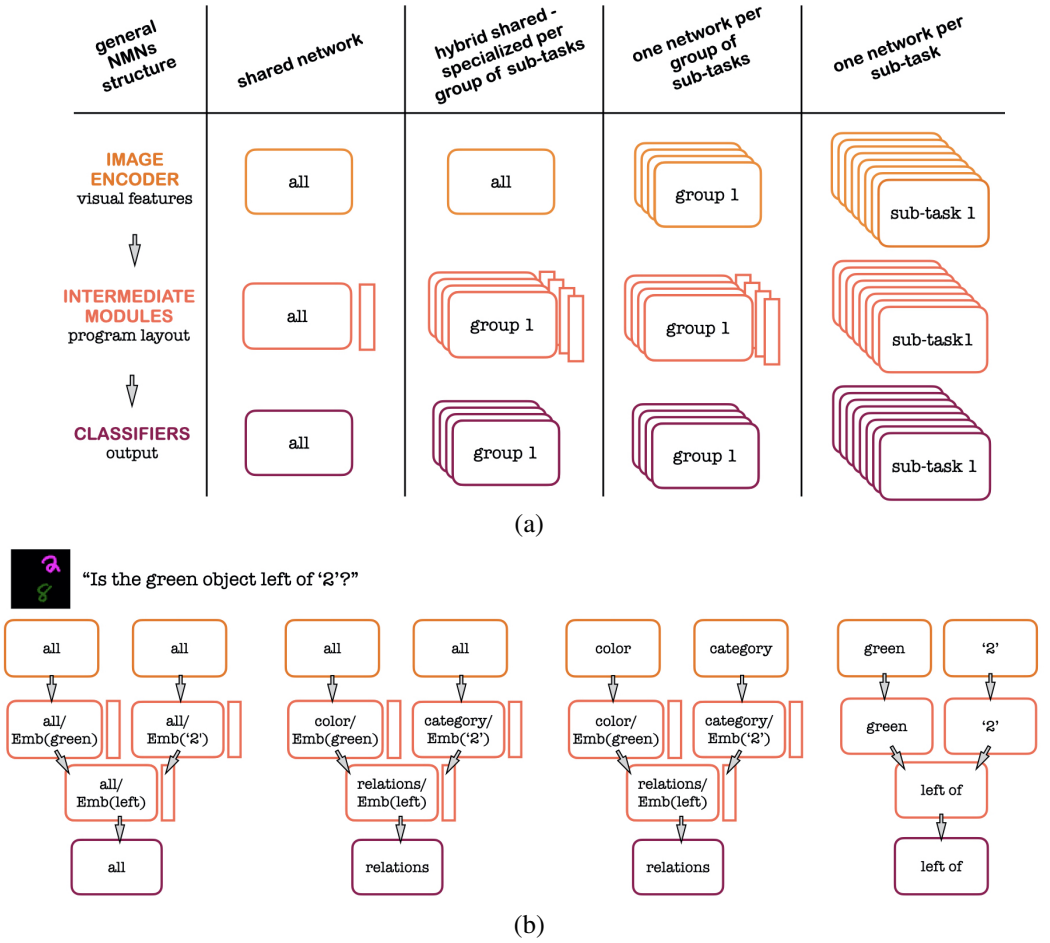

Figure 2: (a) NMNs consist of three stages: the image encoder, the intermediate modules, which reflect the program layout, and the classifier. Modularity is applied at different stages and in different degrees. From the left, the NMN has single module common across tasks. The bar in the intermediate modules represents the embedding vector encoding a sub-task. Going to the right, we impose modularity by group of sub-task or by single sub-task. (b) NMN implementation given a scene with a pair of objects (an object for which we know the color, and a second object, a digit) and the question *"Is the green object left of '2'?"*

set of blocks identified by the label *sub-task*. In the following, we first define modularity for the image encoder and the classifier stages, then we resort to existing implementations of intermediate modules and propose a new one, and finally we provide an example of the implementation of this network.

2.1 Modularity in the Image Encoder and the Classifier

The image encoder is usually a single convolutional neural network common to all sub-tasks (first two columns of Fig. 2a). We further define modularity per group of sub-tasks, as for the orange blocks in the third column in Fig. 2a. For this case, we assign to each group an image encoder, which will be shared among sub-tasks of the same group and will be independent of the other image encoders. The maximum degree of modularity corresponds to having a number of image encoders equivalent to the number of sub-tasks of the VQA problem (see orange blocks, last column of Fig. 2a). In presence of a binary classification task, modularity can be equally imposed on the classifier, as shown in Fig. 2a in violet.

2.2 Modularity in the Intermediate Modules

The definition of intermediate modules varies across the literature. We recall two state-of-the-art definitions which previous works [11, 5] have focused on: the *Find* [16] and the *Residual* [15] modules. Let $f(k, s_x, s_y)$ be the function that represents an intermediate module, where k indexes the module with the sub-task, and (s_x, s_y) are two inputs to the module, which can be the representation of the scene, or the outcome from the precedent intermediate modules. If a module requires a single input for the given sub-task, s_y corresponds to the output of the image encoder given a null input scene. In the following equations, $[s_x; s_y]$ represents the concatenation of the two inputs.

Next, we show that the *Find module* can be interpreted as a *single intermediate module* shared across sub-tasks, where the sub-task to be performed is indicated through the modulation of neural representation with a vector representing the sub-task. Then, we show that the *Residual module* consists of *one intermediate module per sub-task*, where each intermediate module with independent weights corresponds to a specific sub-task. Finally, we introduce a module with a degree of modularity between the *Find module* and the *Residual module*.

A single intermediate module (aka. Find module). The definition is the following:

$$[W_1; b_1; W_2; b_2] = \theta, \tag{1}$$
$$\gamma_k = \text{Embedding}(k), \tag{2}$$
$$f(k, s_x, s_y) = \text{ReLU}(W_1 * (\gamma_k \odot \text{ReLU}(W_2 * [s_x; s_y] + b_2)) + b_1). \tag{3}$$

The parameter θ is the same across sub-tasks, *ie.* it is a single module. The module is encouraged to perform a sub-task through the element-wise product involving γ_k, which modulates the representation in Eq. 3. This choice corresponds to the first column, second row, in Fig. 2a.

One intermediate module per sub-task (aka. Residual module). This module has a much finer degree of modularity than the previous one. For each sub-task k, we define γ_k as an independent set of convolutional weights for each sub-task, as in the following:

$$\gamma_k = [W_1^k; b_1^k; W_2^k; b_2^k, W_3^k; b_3^k], \text{ for each } k, \tag{4}$$
$$\tilde{s}_k = \text{ReLU}(W_3^k * [s_x; s_y] + b_3^k), \tag{5}$$
$$f(k, s_x, s_y) = \text{ReLU}(\tilde{s}_k + W_1^k * \text{ReLU}(W_2^k * \tilde{s}_k + b_2^k) + b_1^k). \tag{6}$$

The representation of this intermediate module is in Fig. 2a, last column.

Intermediate modules separated per group. We introduce a halfway degree of modularity between the two presented. Let G be a set of non-intersecting sets $(\mathcal{G}_1, \dots \mathcal{G}_G)$ of sub-tasks. We define G single intermediate modules with a modulation mechanism, such as the one of the single intermediate module. Sub-tasks belonging to the same group leverage on the same convolutional weight θ^g, which are independent across groups:

$$\text{For each } g \text{ and } k \in \mathcal{G}_g: \tag{7}$$
$$[W_1^g; b_1^g; W_2^g; b_2^g] = \theta^g, \tag{8}$$
$$\gamma_k = \text{Embedding}(k), \tag{9}$$
$$f(k, s_x, s_y) = \text{ReLU}(W_1^g * (\gamma_k \odot \text{ReLU}(W_2^g * [s_x, s_y] + b_2^g)) + b_1^g). \tag{10}$$

We now provide a comparative example among the different degrees of modularity we just introduced. Given the question *e.g., "Is the green object left of '2'?"*, Fig. 2b shows the implementations of the networks in Fig. 2a. The ground truth program layout is a tree structure, with two leaves and a root that identify respectively the pairs of objects and the spatial relation. Our sub-task separation divides sub-tasks related to colors, categories and spatial relations in three different groups. Starting from the first column in Fig. 2b, the network *all - all - all* share the same image encoder across leaves. *All* stands for a single block shared across all sub-tasks. The intermediate module is shared and the modulation mechanism makes each module sub-task selective (*Emb*). The network *all - group - group* has a single image encoder and intermediate modules separated per group. The weights for color and category are independent. The network *group - group - group* has G image encoders. The leaves in the tree receive two different input representations, obtained from the image encoder of colors

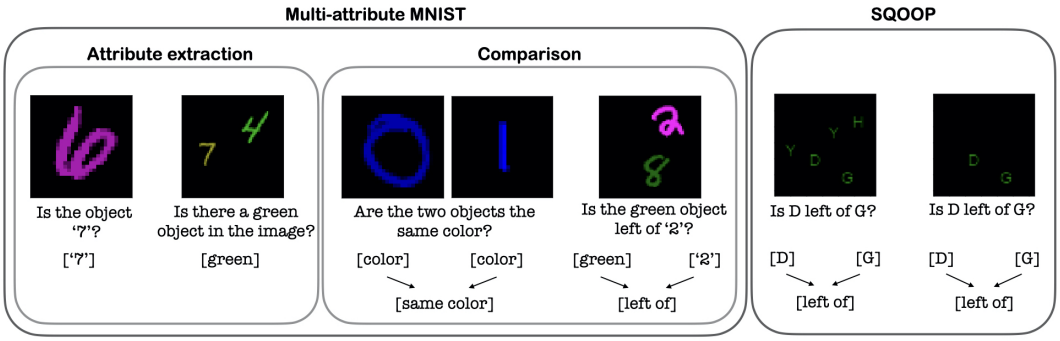

Figure 3: VQA datasets to test systematic generalization: visual input at the top, followed by the question and the correspondent program layout. The Multi-attribute MNIST, left block, contains attribute extraction tasks (from single object and from multiple objects), and comparison (between attribute of separated object pairs and spatial locations). The SQOOP datasets, right block, as in its original version of the dataset on the left, and as our modified version, on the right.

(first object in the question) and categories (second object in the question). The network *sub-task - sub-task - sub-task* has a single and independent representation per sub-task. The image encoder, as well as the intermediate modules, varies across different objects.

3 Datasets

To study the effect of modularity on systematic generalization to novel combinations, it is necessary to limit the data variability of the training set and having test combinations different from those at training (OOD test set). Having control over all these variables is possible for synthetically generated datasets. With this goal, we build several VQA datasets, divided in two main families. The former, Multi-attribute MNIST is a modified version of MNIST, where the digits appear in different colors, sizes, and light conditions. In Multi-attribute MNIST, we limit the combinations of visual attributes for the objects in the scenes. The latter is a modification of the Spatial Queries On Objects Pairs (SQOOP) dataset [11], where originally the amount of possible combinations of objects in the question is limited. In our modification, we limit the amount of combinations of objects in the scene as well.

3.1 Multi-attribute MNIST: limited amount of visual combinations

We designed Multi-attribute MNIST, a family of four VQA datasets. For two datasets, the VQA task is attribute extraction, while the others deal with comparisons of attributes and spatial positions. All tasks admit binary response. Across VQA tasks, each object is a MNIST digit which additionally assumes a color, a size and a light condition among respectively five, three, and three different options (see App. A.1), for a total amount of combinations of attribute instances equal to 450.

Across all datasets, we focus on the bias at the visual level, obtained by reducing the amount of attribute combinations in the image. To generate a limited set of combinations of attributes, we fix a parameter r and we randomly extract combinations among all the possible 450, with the requirement that each of the twenty-one attribute instances appears at least r times, as detailed in App. A.

To study the effect of data variability, we define *data-run* as six different training datasets originated from training combinations obtained with r fixed at $[1, 2, 5, 8, 10, 20]$. Within the same data-run, the set of training combinations extracted for smaller r falls in the training combinations for larger r values. The VQA training datasets generated within a single data-run are tested on novel combinations (OOD test points), obtained by fixing $r = 5$. The amount of training points for each dataset in the data-run is fixed to 210000, the OOD test points are 42000. In Fig. 3, in the Multi-attribute MNIST block, we report the four VQA tasks.

Attribute extraction. Starting from the left of Fig. 3, we have two datasets, *attribute extraction from single object* and *attribute extraction from multiple objects*. In the former case, the image size is 28 and contains a single object, in the latter, the image size is 64 and it always contains a pair of objects.

The second object is a confounding factor, as it is extracted in such a way to never respond positively to the attribute extraction question. In both datasets, the number of questions corresponds to the twenty-one attribute instances. The ground truth program corresponds to a single sub-task, which is identified by the attribute instance. In Fig. 3, for attribute extraction, the sub-tasks correspond to the keys *seven* and *green*.

Attribute comparison. At the center of Fig. 3 we have two datasets for attribute comparison, *attribute comparison between pairs of separated objects* and *comparison between spatial positions*. In the former, we compare a pair of objects contained in separated scenes of size 28. Each pair of scenes has a related question about the comparison of one among the attributes: color, size, category, and light condition, see App.A.1 for details.The ground-truth program layout has a tree structure. For this case, the two leaves can assume the form of one sub-task among category, color, size, and light condition. Sub-tasks and groups of sub-tasks coincides in the leaves. The root tackles the comparison, and the sub-tasks involved in comparison divide in four additional groups for category, color, size, and light condition comparisons, for a total of eight groups. The dataset for comparison between spatial position corresponds to the case at the right of Fig. 3 in the Multi-attribute MNIST section, with image size 64. The program layout has a tree structure, with the leaves assuming the form of one among the twenty-one attribute instances (each instance constitutes a sub-task) to identify the objects in the scene, and the root assuming one relational sub-task among above, below, left, or right. The sub-tasks are divided in five groups, category, color, size, light condition, and spatial relation.

3.2 SQOOP: limited visual and textual co-occurrence

Given a dictionary of thirty-six objects and four spatial relations, the SQOOP dataset tackle the problem of spatial relations between two objects in a 2D environment [11]. The questions consist on comparing the position of two objects, *e.g., Is D left of G?*. To measure systematic generalization, not all pairs of objects are compared during training and OOD evaluates the accuracy to compare novel pairs. Each scene contains five objects, as in the left example of the SQOOP block in Fig. 3. All objects can co-occur in the image, even though specific pairs are not compared during training.

We extend SQOOP by also limiting the co-occurrences of objects in the image. We reduce the objects in the scene to the pair in the question, as shown at the right of the SQOOP block, in Fig. 3. We generate a dataset with the smallest amount of combinations, such that each object only co-occurs with another object. Given a generic training question *e.g., Is D left of G?*, all the training scenes related to this question contain the *D* and *G* objects only, reducing exponentially the data variability of the original SQOOP datasets with five objects.

4 Results

We report the results obtained on Multi-attribute MNIST and the original and modified SQOOP datasets. We measure the systematic generalization of the models introduced in Sec. 2 by quantifying their OOD test accuracy. Despite the illusive simplicity of the tasks and the usage of synthetic datasets, the results highlights the limitations of these methods and the large impact of modularity in the out-of-distribution case.

4.1 Multi-attribute MNIST

The plots in Fig. 4 show the systematic generalization to novel combinations on (a) attribute extraction from single object, (b) attribute extraction from multiple objects, (c) attribute comparison between pairs of separated objects, and (d) comparison between spatial positions. For each task we consider the performance of the NMNs on two data-runs. Across all plots, each value on the x-axis reports the average percentage of training combinations for a fixed r across the two data-runs, where the bars are grouped based on the r value. In the legend we specify the modularity at the three stages of the network, where we use the nomenclature *all*, *group*, and *sub-task* of Sec.2 to denote different degrees of modularity. We report for each NMN mean and standard deviation OOD test accuracy across two data-runs. Details on NMNs' hyper-parameters and model selection are in Sec. B.3. In Fig. 6 we report the striking difference in accuracy between in-distribution and out-of-distribution testing for these experiments.

Generalizing to out-of-distribution test data is challenging. The low OOD accuracy across VQA tasks in Fig. 4 highlights consistently the difficulty to generalize to novel combinations, which does

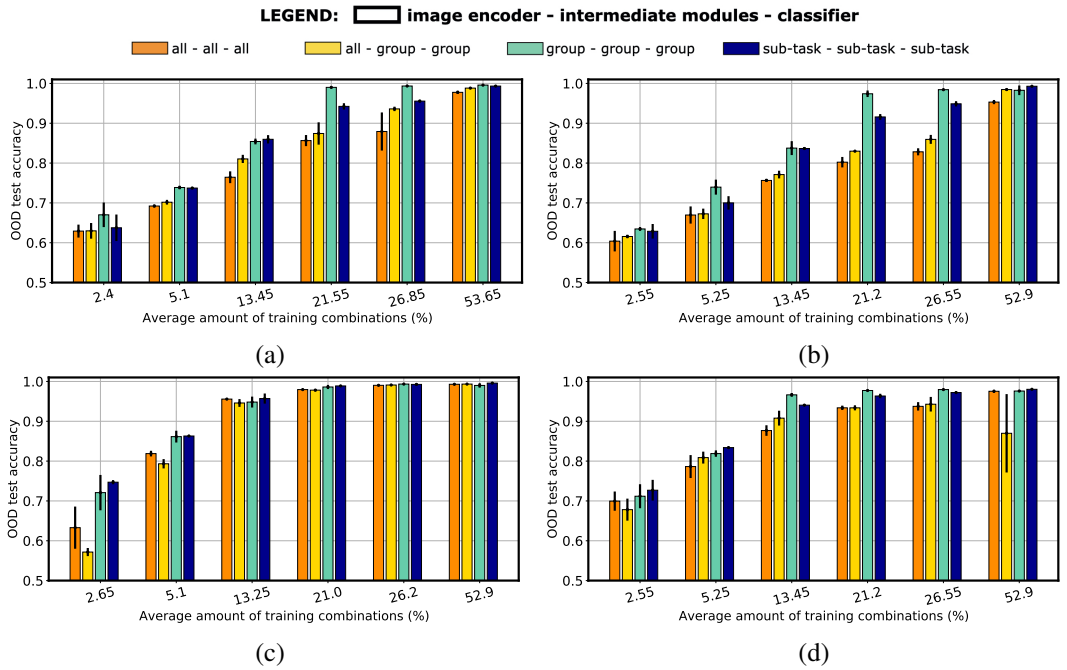

Figure 4: Systematic generalization of NMNs tested on novel combinations of Multi-attribute MNIST. (a) attribute extraction from single object, (b) attribute extraction from multiple objects, (c) attribute comparison between pairs of separated objects, (d) comparison between spatial positions.

not emerge from the in-distribution accuracy (see Fig.6 in Appendix). We underline that this behavior depends on the variability of the training combinations, since the amount of training points is fixed across all experiments and all datasets.

Modularity, especially in the image encoder, improves generalization to novel combinations. NMNs with induced modularity at all stages achieve the highest OOD test accuracy across all VQA tasks (the group - group - group NMN, in green, and sub-task - sub-task - sub-task NMN, in blue). The difference in the definition of intermediate modules and degree of modularity between these networks highlights the generality of this finding. The comparison between the all - group - group NMN, in yellow, and the group - group - group NMN, in green, which differ at the image encoder stage only, further reveals the importance of having a task-based image encoder stage. We report additional NMNs with modularity at the intermediate module stage and with batch normalization [17] trained on the same VQA tasks. Moreover, for the task of attribute comparison, we introduce two further networks with different degrees of modularity in the intermediate modules, which differ for the degree of modularity of their image encoders. All these results provide further evidence to the importance of a modular image encoder (see App. B.1 and App. B.4). This claim is in agreement with recent works where the alignment of image and textual inputs is applied from early stages of the network [18].

The low OOD test accuracy is not dependent on the amount of data points, but data variability. We asked if the performance of all models is impacted by a limited size of the training set. To answer this, we considered the task of attribute extraction from single object and attribute comparison between objects in separated scenes, and increased the training set size of a factor ten (for a total of 2.1M points), while keeping the same percentage of training combinations of the previous experiments in Fig. 4 (a,c), data-run #2. The results in Sec. B.5 are consistent to those in Fig. 4.

4.2 SQOOP

Recall that we focus on the case with the smallest amount of training combinations with one object co-occuring with only another object. We use the two variants of the tree-NMNs proposed in the previous work [11], together with the same optimization parameters. To compute the OOD performance, we consider the model at its last training iteration using the default hyperparameters for SQOOP. The first tree-NMN has a single image encoder, a single intermediate module, and a single classifier (*all* -

Table 1: SQOOP results. Top row: bias in the question, bottom row: bias in the co-occurrence of objects in the scene and question. The NMN with modular intermediate modules reach higher OOD test accuracy in presence of the bias in the co-occurrence.

	all - sub-task - all tree-NMN	all - all - all tree-NMN
Bias in the question (previous work)	$\mathbf{0.9996 \pm 0.0006}$	$\mathbf{0.998 \pm 0.002}$
Bias in the co-occurrence	$\mathbf{0.885 \pm 0.005}$	0.84 ± 0.02

Table 2: CoGenT mean and standard deviation performance across five repetitions. NMNs are trained on condition A and tested on validation condition A and validation condition B.

	GT-Tensor-NMN	GT-Vector-NMN	GT-Vector-NMN with modular image encoder (ours)
Condition A	0.979 ± 0.001	$\mathbf{0.980 \pm 0.002}$	0.944 ± 0.003
Condition B	0.727 ± 0.005	0.732 ± 0.002	$\mathbf{0.773 \pm 0.013}$

all - all), while the second tree-NMN differs in the intermediate module stage, with one intermediate module per sub-task (*all - sub-task - all*). Table 1 shows the OOD test accuracy mean and standard deviation of the two NMNs across five repetitions of the experiment.

Modularity is beneficial for bias in the co-occurrence of objects. On the SQOOP dataset with five objects [11] the tree-NMNs achieve almost perfect systematic generalization over new test questions. In presence of two objects per scene, which leads to bias in the co-occurrences in the image, the modular tree-NMNs achieves higher systematic generalization performance. In both cases, the additional bias in the amount of objects at training and testing causes a decrease in performance. We further report in Tab. 5 the performance of VQA models different from NMNs (FiLM [19] and MAC [20]). The results provide re-assurance that NMNs achieve higher systematic generalization given the program layout.

5 Application on CLEVR Split for Systematic Generalization

CLEVR is a diagnostic dataset in VQA [2] consisting of complex 3D scenes with multiple objects and program layout formed by multiple sub-tasks. This dataset comes with additional splits to test the systematic generalization capacity, including the Compositional Generalization Test (CoGenT). CoGenT divides in two conditions where cubes and cylinders appear in a limited amount of colors, that are inverted between training and OOD testing (see App. D.2). More recently, CLOSURE has been proposed as a OOD test set to measure the ability to refer to known objects in arbitrary contexts, together with Vector-NMN [5], a new NMN architecture outperforming the previous NMNs on CLOSURE (see App. D.1, Eqs. 11-15). The Vector-NMN, as all the previous NMNs, has a single image encoder shared across sub-tasks.

To verify the generality of our findings in this more complex scenario, we induce modularity in the image encoder of the Vector-NMN. The Vector-NMN, besides being the most performant model, has intermediate modules that take in input the original image, and we hypothesize that a modular image encoder could be effective on systematic generalization in presence of visual bias. The Vector-NMN's image encoder takes in input the features extracted from a pre-trained ResNet-101 [21] (see Sec. 4.2 in [2]), and consists of few trainable convolutional layers shared among sub-tasks. To build our Vector-NMN with modular image encoder we maintain the same input features and divide the sub-task into eight groups: (i) counting tasks, (ii) tasks related to colors, (iii) materials, (iv) shapes, (v) sizes, (vi) spatial relations, (vii) logical operations, (viii) input scene (for details see App. D.3).

In our experiments, the optimization hyper-parameters for the NMNs are fixed to the ones in the previous work [5], and the Ground Truth (GT) program layout is given. Tab. 2, 3 show the mean and standard deviation accuracy of Tensor-NMN, Vector-NMN, and our modular Vector-NMN in presence of a limited amount of objects-attribute combinations (CoGenT) across five different initializations. Tab. 2 shows the overall validation accuracy on splits Condition A (in-distribution) and Condition B (out-of-distribution). The analysis of performance for single questions in Condition B is in Tab. 3.

Table 3: Validation performance on Condition B divided by question type across five repetitions.

	GT-Tensor-NMN	GT-Vector-NMN	GT-Vector-NMN with modular image encoder (ours)
`count`	0.697 ± 0.008	0.704 ± 0.004	$\mathbf{0.71 \pm 0.01}$
`equal_color`	0.756 ± 0.008	0.74 ± 0.01	$\mathbf{0.80 \pm 0.01}$
`equal_integer`	0.827 ± 0.003	0.78 ± 0.02	$\mathbf{0.85 \pm 0.02}$
`equal_material`	0.74 ± 0.02	0.742 ± 0.007	$\mathbf{0.84 \pm 0.02}$
`equal_shape`	$\mathbf{0.91 \pm 0.02}$	0.89 ± 0.03	0.79 ± 0.02
`equal_size`	0.75 ± 0.01	0.75 ± 0.01	$\mathbf{0.88 \pm 0.02}$
`exist`	0.842 ± 0.004	$\mathbf{0.844 \pm 0.004}$	0.844 ± 0.005
`greater_than`	0.838 ± 0.006	0.836 ± 0.004	$\mathbf{0.89 \pm 0.01}$
`less_than`	0.807 ± 0.009	0.820 ± 0.005	$\mathbf{0.87 \pm 0.02}$
`query_color`	0.58 ± 0.01	0.60 ± 0.01	$\mathbf{0.67 \pm 0.04}$
`query_material`	0.841 ± 0.009	0.847 ± 0.004	$\mathbf{0.882 \pm 0.008}$
`query_shape`	0.37 ± 0.01	0.40 ± 0.03	$\mathbf{0.52 \pm 0.03}$
`query_size`	0.835 ± 0.006	0.847 ± 0.007	$\mathbf{0.895 \pm 0.005}$

Modularity in Vector-NMN leads to higher systematic generalization on CoGenT. The performance in Tab. 2 highlights the wide gap between classical and systematic generalization across all NMNs in presence of visual bias (Condition B). Our GT-Vector-NMN achieves the best systematic generalization performance. This comes with a reduction of the accuracy on Condition A. Tab. 3 shows large performance variations depending on questions types. On those questions that are most affected by the limited amount of training combinations `query_shape` and `query_color`, modularity brings respectively an improvement of 12% and 7% in accuracy. A broader comparison of our Vector-NMN with FiLM [19], MAC [20], and further NMN variations is reported in Tab. 7 (App. D.4.).

6 Conclusions and Discussion

Our results highlight the importance of the degree of modularity, especially at the image encoder stage, for systematic generalization, across many datasets, NMN architectures, and tasks of different complexity. These findings are easily applicable on existing NMNs and bring to higher performance than the original methods when tested on a benchmark dataset to measure generalization on novel combinations (CoGenT).

This work also lead to new research questions. While we have shown that modularity has a large impact in systematic generalization, we have tuned the degree of modularity at hand and in practical applications it is desirable to do this automatically. Moreover, since we have fixed the program generator, it is unclear how bias in the program layout could affect systematic generalization. To motivate follow-up work, in App. E we report a pilot experiment that shows that bias in the program layout could degrade systematic generalization of the modular NMNs we introduced. This suggests that there may be a trade-off between modularity and bias in the program layout.

Finally, we are intrigued about the neural mechanisms that facilitate systematic generalization and how modularity positively affects those mechanisms. Some hints towards an answer have been pointed out by [22], which shows that the emergence of invariance to nuisance factors at the individual neural level improves systematic generalization for object recognition.

References

[1] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[2] 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. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2901–2910, 2017.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

[4] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4971–4980, 2018.

[5] Dzmitry Bahdanau, Harm de Vries, Timothy J O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. CLOSURE: Assessing systematic generalization of CLEVR models. *arXiv preprint arXiv:1912.05783*, 2019.

[6] Damien Teney, Kushal Kafle, Robik Shrestha, Ehsan Abbasnejad, Christopher Kanan, and Anton van den Hengel. On the value of out-of-distribution testing: An example of Goodhart's law. In *Advances in Neural Information Processing Systems*, pages 407–417, 2020.

[7] Kushal Kafle and Christopher Kanan. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20, 2017.

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

[9] João Loula, Marco Baroni, and Brenden Lake. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, 2018.

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

[11] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? In *Proceedings of the International Conference on Learning Representations*, 2019.

[12] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *Proceedings of the International Conference on Learning Representations*, 2019.

[13] Nitish Gupta and Mike Lewis. Neural compositional denotational semantics for question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[14] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2016.

[15] 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 *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2989–2998, 2017.

[16] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 804–813, 2017.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[18] Aishwarya Kamath, Mannat Singh, Yann LeCun, Ishan Misra, Gabriel Synnaeve, and Nicolas Carion. MDETR–modulated detection for end-to-end multi-modal understanding. *arXiv preprint arXiv:2104.12763*, 2021.

[19] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[20] Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[21] 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 (CVPR)*, pages 770–778, 2016.

[22] Spandan Madan, Timothy Henry, Jamell Dozier, Helen Ho, Nishchal Bhandari, Tomotake Sasaki, Frédo Durand, Hanspeter Pfister, and Xavier Boix. On the capability of neural networks to generalize to unseen category-pose combinations. *arXiv preprint arXiv:2007.08032*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 5, Sec. 6 and App. E.
   (c) Did you discuss any potential negative societal impacts of your work? [No] This research tackles the problem of dataset bias and propose new solutions in presence of limited amount of training attributes. We think that this type of investigation could help future research to make AI more transparent and interpretable.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] This work should not cause any harm to people or environment, but it is rather an analysis with the goal of studying how the bias affects NMNs and how we can make these models able to systematically generalize. The data used in the dataset are available online, synthetically generated, and do not contain offensive images.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Sec.B.3. For experiments on SQOOP and CLEVR we use respectively the hyper-parameters in [11] and [5]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In Fig. 4 for Multi-attribute MNIST. In Tab. 1 for the SQOOP experiments. In Tabs. 2 3, 7, 8 for the experiments related to the CLEVR dataset
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Supplementary – we give an estimate of resources and time used for the experiments.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] This work extends the result of two previous publications [11], [5] which are recurrently cited in this work
   (b) Did you mention the license of the assets? [No] The code from which we took inspiration has MIT license. Also our code is open-source with MIT license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] The data and the code are open source and reusable `https://github.com/rizar/CLOSURE/blob/master/NOTICE_AND_LICENSE`. The Multi-attribute MNIST dataset we use in the experiments can be easily generated using the code we provide in the Supplementary materials.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]