

# Structure Learning for Neural Module Networks

Vardaan Pahuja<sup>◇1</sup>      Jie Fu<sup>†‡</sup>  
Sarath Chandar<sup>†§</sup>      Christopher J. Pal<sup>†‡</sup>

†Mila    §Université de Montréal  
‡Polytechnique Montréal    ◇The Ohio State University

October 23, 2019

---

<sup>1</sup>Work done when the author was a student at Mila, Université de Montréal.

# Overview

- 1 Motivation
- 2 Module Layout Controller
- 3 Training Algorithm
- 4 Experiments
- 5 Results
- 6 Measuring the sensitivity of modules
- 7 Visualization of module network parameters

- Visual Reasoning using Neural Module Networks (NMN).
- Previous Approaches (e.g. NMN [Andreas et al., 2016], End-to-end Neural Module Networks [Hu et al., 2017], Stack-NMN [Hu et al., 2018]) use hand-designed modules
- This may not be feasible when one has limited knowledge of the kinds of questions or associated visual reasoning required to solve the task.
- Our model: learn the module structure and the module parameters together

# Structure of the Generic Module I

- The cell denotes a generic module, which we suppose can span all the required modules for a visual reasoning task.
- Each cell contains a certain number of nodes.
- We consider two broad kinds of modules: (i) *Attention modules* which output an attention map (ii) *Answer modules* which output memory features to be stored in the memory.
- The function of a node (denoted by  $O$ ) is to perform a weighted sum of outputs of different arithmetic operations applied on the input feature maps  $\mathbf{x}_1$  and  $\mathbf{x}_2$ .

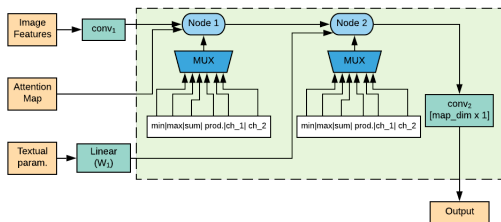
## Structure of the Generic Module II

- The function of a node (denoted by  $O$ ) is to perform a weighted sum of outputs of different arithmetic operations applied on the input feature maps  $\mathbf{x}_1$  and  $\mathbf{x}_2$ .

$$O(\mathbf{x}_1, \mathbf{x}_2) = \alpha'_1 * \min(\mathbf{x}_1, \mathbf{x}_2) + \alpha'_2 * \max(\mathbf{x}_1, \mathbf{x}_2) + \alpha'_3 * (\mathbf{x}_1 + \mathbf{x}_2) + \alpha'_4 * (\mathbf{x}_1 \odot \mathbf{x}_2) + \alpha'_5 * \text{choose}_1(\mathbf{x}_1, \mathbf{x}_2) + \alpha'_6 * \text{choose}_2(\mathbf{x}_1, \mathbf{x}_2)$$

where  $\alpha' = \sigma(\alpha)$ ,  $\text{choose}_1(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1$  and  $\text{choose}_2(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_2$ .

# Generic Module with 3 inputs

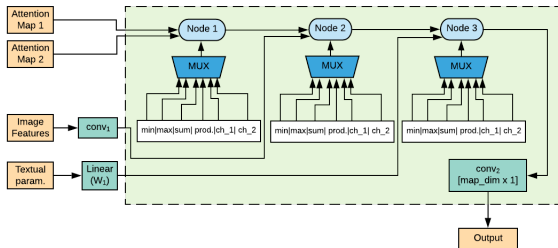


This module type receives 3 inputs:

- image features
- textual parameter
- An attention map

Output: An attention map

# Generic Module with 4 inputs



This module type receives 4 inputs:

- Two attention maps
- image features
- textual parameter

Output: An attention map

# Hand-designed Modules

| module name | input attention | output type | implementation details<br>( $x$ : image feature map, $c$ : textual parameter)     |
|-------------|-----------------|-------------|---|
| Find        | (none)          | attention   | $a_{out} = \text{conv}_2(\text{conv}_1(x) \odot Wc)$                              |
| Transform   | $a$             | attention   | $a_{out} = \text{conv}_2(\text{conv}_1(x) \odot W_1 \sum(a \odot x) \odot W_2 c)$ |
| And         | $a_1, a_2$      | attention   | $a_{out} = \text{minimum}(a_1, a_2)$  |
| Or          | $a_1, a_2$      | attention   | $a_{out} = \text{maximum}(a_1, a_2)$  |
| Filter      | $a$             | attention   | $a_{out} = \text{And}(a, \text{Find}())$ , i.e. reusing Find and And              |
| Scene       | (none)          | attention   | $a_{out} = \text{conv}_1(x)$  |
| Answer      | $a$             | answer      | $y = W_1^T (W_2 \sum(a \odot x) \odot W_3 c)$                                     |
| Compare     | $a_1, a_2$      | answer      | $y = W_1^T (W_2 \sum(a_1 \odot x) \odot W_3 \sum(a_2 \odot x) \odot W_4 c)$       |
| NoOp        | (none)          | (none)      | (does nothing)  |

**Table:** Neural modules used in [Hu et al., 2018]. The modules take image attention maps as inputs, and output either a new image attention  $a_{out}$  or a score vector  $y$  over all possible answers ( $\odot$  is elementwise multiplication;  $\sum$  is sum over spatial dimensions).

Credits: Stack -NMN [Hu et al., 2018]



# Module Layout Controller

**Data:** Question (string), Image features ( $\mathcal{I}$ )

Encode the input question into  $d$ -dimensional sequence

$[\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_S]$  using Bidirectional LSTM.

$A^{(0)} \leftarrow$  Initialize the memory stack ( $A; p$ ) with uniform image attention and set the stack pointer  $p$  to point at the bottom of the stack (one-hot vector with 1 in the 1<sup>st</sup> dim.).

**for** each time-step  $t = 0, 1, \dots, (T-1)$  **do**

$$\mathbf{u} = \mathbf{W}_2[\mathbf{W}_1^{(t)} \mathbf{q} + \mathbf{b}_1; \mathbf{c}_{t-1}] + \mathbf{b}_2;$$

$$\mathbf{w}^{(t)} = \text{softmax}(\text{MLP}(\mathbf{u}; \theta_{\text{MLP}}));$$

$$cv_{t,s} = \text{softmax}(W_3(\mathbf{u} \odot \mathbf{h}_s));$$

$$\mathbf{c}_t = \sum_{s=1}^S cv_{t,s} \cdot \mathbf{h}_s$$

**for** every module  $m \in M$  **do**

    Produce updated stack and stack pointer:

$$(\mathbf{A}_m^{(t)}, \mathbf{p}_m^{(t)}) = \text{run-module}(m, \mathbf{A}^{(t)}, \mathbf{p}^{(t)}, \mathbf{c}_t, \mathcal{I});$$

**end**

$$\mathbf{A}^{(t+1)} = \sum_{m \in M} \mathbf{A}_m^{(t)} \cdot \mathbf{w}_m^{(t)};$$

$$\mathbf{p}^{(t+1)} = \text{softmax}(\sum_{m \in M} \mathbf{p}_m^{(t)} \cdot \mathbf{w}_m^{(t)})$$

**end**

**Algorithm 1:** Operation of Module Layout Controller and Memory Stack.

# Final Classifier

- At each time-step of module execution, the weighted average of output of the *Answer* modules is called memory features (denoted by  $f_{mem}^{(t)} = \sum_{m \in \text{ans. module}} o_m^{(t)} w_m^{(t)}$ ).
- Here,  $o_m^{(t)}$  denotes the output of module  $m$  at time  $t$ . The memory features are given as one of the inputs to the *Answer* modules at the next time-step.
- The memory features at the final time-step are concatenated with the question representation, and then fed to an MLP to obtain the logits.

# Training Algorithm

**while** *not converged* **do**

1. Update weights  $\mathcal{W}$  by descending

$$\nabla_{\mathbf{w}} \left[ \mathcal{L}_{train}(\mathcal{W}, \alpha) - \frac{\lambda_w}{T} \sum_{t=1}^T \mathcal{H}(\mathbf{w}^{(t)}) \right];$$

2. Update architecture  $\alpha$  by descending

$$\nabla_{\alpha} \left[ \mathcal{L}_{val}(\mathcal{W}, \alpha) - \lambda_{op} \sum_{m=1}^M \sum_{k=1}^p \frac{\|\sigma(\alpha^{m,k})\|_2}{\|\sigma(\alpha^{m,k})\|_1} \right];$$

**end**

**Algorithm 2:** Training Algorithm for LNMN Modules. Here,  $\alpha$  denotes the collection of module network parameters i.e.  $\{\alpha_i^{m,k}\}_{i=1}^6$  for  $k^{th}$  node of module  $m$ ,  $\mathcal{W}$  denotes the collection of weight parameters of modules and all other non-module parameters.

- CLEVR [Johnson et al., 2017a]: synthetic dataset for visual reasoning which contains questions that test visual reasoning abilities such as counting, comparing, logical reasoning based on 3D shapes like cubes, spheres, and cylinders of varied shades.
- Natural Image datasets: VQA v1 [Antol et al., 2015] and VQA v2 [Goyal et al., 2017].

# Results on CLEVR dataset

| Model   | CLEVR Overall | Count | Exist | Compare Numbers | Query Attribute | Compare Attribute | CLEVR Humans |
|---|---------------|-------|-------|-----------------|-----------------|-------------------|--------------|
| Human [Johnson et al., 2017b]                     | 92.6          | 86.7  | 96.6  | 86.5            | 95.0            | 96.0              | -            |
| Q-type baseline [Johnson et al., 2017b]           | 41.8          | 34.6  | 50.2  | 51.0            | 36.0            | 51.3              | -            |
| LSTM [Johnson et al., 2017b]                      | 46.8          | 41.7  | 61.1  | 69.8            | 36.8            | 51.8              | 36.5         |
| CNN+LSTM [Johnson et al., 2017b]                  | 52.3          | 43.7  | 65.2  | 67.1            | 49.3            | 53.0              | 43.2         |
| CNN+LSTM+SA+MLP [Johnson et al., 2017a]           | 73.2          | 59.7  | 77.9  | 75.1            | 80.9            | 70.8              | 57.6         |
| N2NMN* [Hu et al., 2017]                          | 83.7          | 68.5  | 85.7  | 84.9            | 90.0            | 88.7              | -            |
| PG+EE (700K prog.)* [Johnson et al., 2017b]       | 96.9          | 92.7  | 97.1  | 98.7            | 98.1            | 98.9              | -            |
| CNN+LSTM+RN <sup>‡</sup> [Santoro et al., 2017]   | 95.5          | 90.1  | 97.8  | 93.6            | 97.9            | 97.1              | -            |
| CNN+GRU+FiLM [Perez et al., 2017]                 | 97.7          | 94.3  | 99.1  | 96.8            | 99.1            | 99.1              | 75.9         |
| MAC [Hudson and Manning, 2018]                    | 98.9          | 97.1  | 99.5  | 99.1            | 99.5            | 99.5              | 81.5         |
| TbD [Mascharka et al., 2018]                      | 99.1          | 97.6  | 99.2  | 99.4            | 99.5            | 99.6              | -            |
| Stack-NMN (9 mod.) <sup>†</sup> [Hu et al., 2018] | 91.41         | 81.78 | 95.78 | 85.23           | 95.45           | 95.68             | 68.06        |
| LNMN (9 modules)                                  | 89.88         | 84.28 | 93.74 | 89.63           | 89.64           | 94.84             | 66.35        |
| LNMN (11 modules)                                 | 90.52         | 84.91 | 95.21 | 91.06           | 90.03           | 94.97             | 65.68        |
| LNMN (14 modules)                                 | 90.42         | 84.79 | 95.52 | 90.52           | 89.73           | 95.26             | 65.86        |

**Table:** CLEVR and CLEVR-Humans Accuracy by baseline methods and our models. (\*) denotes use of extra supervision through program labels. (‡) denotes training from raw pixels. † Accuracy figures for our implementation of Stack-NMN.

# Ablation Studies

| Model   | Overall | Count | Exist | Compare number | Query attribute | Compare Attribute |
|---|---------|-------|-------|----------------|-----------------|-------------------|
| Original setting<br>( $T = 5, L = 5, \text{map\_dim} = 384$ )   | 89.78   | 84.54 | 93.46 | 88.70          | 89.59           | 94.87             |
| Use hard-max for operation weights<br>(for inference only)<br>( $T = 5, L = 5, \text{map\_dim} = 384$ ) | 87.99   | 81.53 | 94.11 | 87.70          | 88.27           | 91.55             |
| $T = 9, L = 9, \text{map\_dim} = 256$   | 89.96   | 84.03 | 93.45 | 89.98          | 90.75           | 93.10             |
| Concatenate all inputs<br>followed by conv. layer   | 47.03   | 42.5  | 61.15 | 68.64          | 38.06           | 49.43             |

**Table:** Model Ablations for LNMN (CLEVR Validation set performance). The term 'map\_dim' refers to the dimension of feature representation obtained at the input or output of each node of cell.

# Results on Natural Image VQA datasets

| Model            | VQA v2 | VQA v1 |
|------------------|--------|--------|
| Stack-NMN        | 58.23  | 59.84  |
| LNMN (9 modules) | 54.85  | 57.67  |

Table: Test Accuracy on Natural Image VQA datasets

# Measuring the sensitivity of modules I

- We use an attribution technique called Integrated Gradients [Sundararajan et al., 2017] to study the impact of module structure parameters (denoted by  $\{\alpha_i^{m,k}\}_{i=1}^6$  for  $k^{th}$  node of module  $m$ ) on the probability distribution in the last layer of LNMN model.

$$\text{IG}(\alpha_i^m) = \sum_{j=1}^N \sum_{k=1}^p \left[ (\alpha_i^{m,k} - (\alpha_i^{m,k})') \times \int_{\xi=0}^1 \frac{\partial F(\mathcal{I}_j, \mathbf{q}_j, (1-\xi) \times (\alpha_i^{m,k})' + \xi \times \alpha_i^{m,k})}{\partial \alpha_i^{m,k}} \right]$$

Here,  $\mathcal{I}_j$  and  $\mathbf{q}_j$  denote the (image, question) pairs for the  $j^{th}$  example respectively.  $F(\mathcal{I}_j, \mathbf{q}_j, \alpha)$  denotes the function that assigns the probability corresponding to the correct answer index in the softmax distribution.



# Measuring the sensitivity of modules II

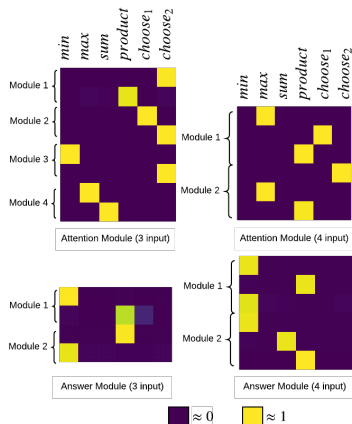
| Module ID | Module type     | min    | max    | sum    | product | choose_1 | choose_2 |
|-----------|-----------------|--------|--------|--------|---------|----------|----------|
| 0         | Attn. (3 input) | 6.3 e4 | 2.7 e4 | 3.6 e4 | 1.1 e5  | 5.1 e4   | 1.6 e4   |
| 1         | Attn. (3 input) | 4.4 e4 | 1.8 e4 | 6.2 e4 | 1.4 e4  | 2.8 e4   | 1.7 e5   |
| 2         | Attn. (3 input) | 7.0 e4 | 3.3 e4 | 3.8 e4 | 1.1 e5  | 5.2 e4   | 1.5 e4   |
| 3         | Attn. (3 input) | 8.6 e3 | 6.2 e4 | 1.7 e4 | 1.8 e4  | 4.7 e4   | 3.0 e4   |
| 4         | Attn. (4 input) | 4.5 e4 | 3.2 e4 | 7.6 e4 | 1.7 e4  | 3.6 e4   | 2.1 e5   |
| 5         | Attn. (4 input) | 1.1 e5 | 5.6 e5 | 2.3 e5 | 8.5 e3  | 2.8 e4   | 1.8 e5   |
| 6         | Ans. (3 input)  | 2.1 e6 | 4.3 e6 | 4.4 e6 | 8.3 e6  | 2.3 e6   | 4.9 e5   |
| 7         | Ans. (4 input)  | 1.2 e5 | 5.8 e4 | 1.7 e5 | 5.2 e3  | 1.0 e5   | 4.5 e5   |

**Table:** Analysis of gradient attributions of  $\alpha$  parameters corresponding to each module (LNMN (9 modules)), summed across all examples of CLEVR validation set.

- The module structure parameters ( $\alpha$  parameters) of the *Answer* modules have their attributions to the final probability around 1-2 orders of magnitudes higher than rest of the modules.

- The higher influence of Answer modules can be explained by the fact that they receive the memory features from the previous time-step and the classifier receives the memory features of the final time-step.

# Visualization of module network parameters I



**Figure:** Visualization of module structure parameters (LNMN (11 modules)). For each module, each row denotes the  $\alpha' = \sigma(\alpha)$  parameters of the corresponding node.




# Visualization of module network parameters II

| Module type             | Module implementation   |
|-------------------------|---|
| Attention<br>(3 inputs) | $O(img, a, c_{txt}) = conv_2(choose_2(conv_1(\mathcal{I}), a) \odot W_1 c_{txt}) = conv_2(a \odot W_1 c_{txt})$ $O(img, a, c_{txt}) = conv_2(choose_2(choose_1(conv_1(\mathcal{I}), a), W_1 c_{txt})) = conv_2(W_1 c_{txt})$ $O(img, a, c_{txt}) = conv_2(choose_2(\min(conv_1(\mathcal{I}), a), W_1 c_{txt})) = conv_2(W_1 c_{txt})$ $O(img, a, c_{txt}) = conv_2(\max(conv_1(\mathcal{I}), a) + W_1 c_{txt})$ |
| Attention<br>(4 inputs) | $O(img, a_1, a_2, c_{txt}) = conv_2(choose_1(\max(a_1, a_2), conv_1(\mathcal{I})) \odot W_1 c_{txt})$ $= conv_2(\max(a_1, a_2) \odot W_1 c_{txt})$ $O(img, a_1, a_2, c_{txt}) = conv_2(\max(choose_2(a_1, a_2), conv_1(\mathcal{I})) \odot W_1 c_{txt})$ $= conv_2(\max(a_2, conv_1(\mathcal{I})) \odot W_1 c_{txt})$   |
| Answer<br>(3 inputs)    | $O(img, a, c_{txt}) = W_2[\sum \min(conv_1(\mathcal{I}), a) \odot W_1 c_{txt}, W_1 c_{txt}, f_{mem}]$ $O(img, a, c_{txt}) = W_2[\sum \min((conv_1(\mathcal{I}) \odot a), W_1 c_{txt}), W_1 c_{txt}, f_{mem}]$   |
| Answer<br>(4 inputs)    | $O(img, a_1, a_2, c_{txt}) = W_2[\sum \min((\min(a_1, a_2) \odot conv_1(\mathcal{I})), W_1 c_{txt}), W_1 c_{txt}, f_{mem}]$ $O(img, a_1, a_2, c_{txt}) = W_2[\sum ((\min(a_1, a_2) + conv_1(\mathcal{I})) \odot W_1 c_{txt}), W_1 c_{txt}, f_{mem}]$  |




**Table:** Analytical expression of modules learned by LNMN (11 modules). In the above equations,  $\sum$  denotes sum over spatial dimensions of the feature tensor.

- A differentiable approach to learn the modules needed in a visual reasoning task automatically.
- Extensive analysis of the degree to which each module influences the prediction function of the model, the effect of each arithmetic operation on overall accuracy and the analytical expressions of the learned modules.




# References I

-  Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016).  
Neural module networks.  
*In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.
-  Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015).  
VQA: Visual Question Answering.  
*In International Conference on Computer Vision (ICCV)*.
-  Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017).  
Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering.  
*In Conference on Computer Vision and Pattern Recognition (CVPR)*.

## References II




-  Hu, R., Andreas, J., Darrell, T., and Saenko, K. (2018). Explainable neural computation via stack neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–69.
-  Hu, R., Andreas, J., Rohrbach, M., Darrell, T., and Saenko, K. (2017). Learning to reason: End-to-end module networks for visual question answering. *CoRR*, *abs/1704.05526*, 3.
-  Hudson, D. A. and Manning, C. D. (2018). Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*.

## References III

-  Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. (2017a).  
Clevr: A diagnostic dataset for compositional language and elementary visual reasoning.  
*In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE.
-  Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Zitnick, C. L., and Girshick, R. (2017b).  
Inferring and executing programs for visual reasoning.  
*In ICCV*.
-  Mascharka, D., Tran, P., Soklaski, R., and Majumdar, A. (2018).  
Transparency by design: Closing the gap between performance and interpretability in visual reasoning.  
*2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4942–4950.



## References IV

-  Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. (2017).  
Film: Visual reasoning with a general conditioning layer.  
*CoRR*, abs/1709.07871.
-  Santoro, A., Raposo, D., Barrett, D. G., 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*, pages 4967–4976.
-  Sundararajan, M., Taly, A., and Yan, Q. (2017).  
Axiomatic attribution for deep networks.  
*arXiv preprint arXiv:1703.01365*.

The End