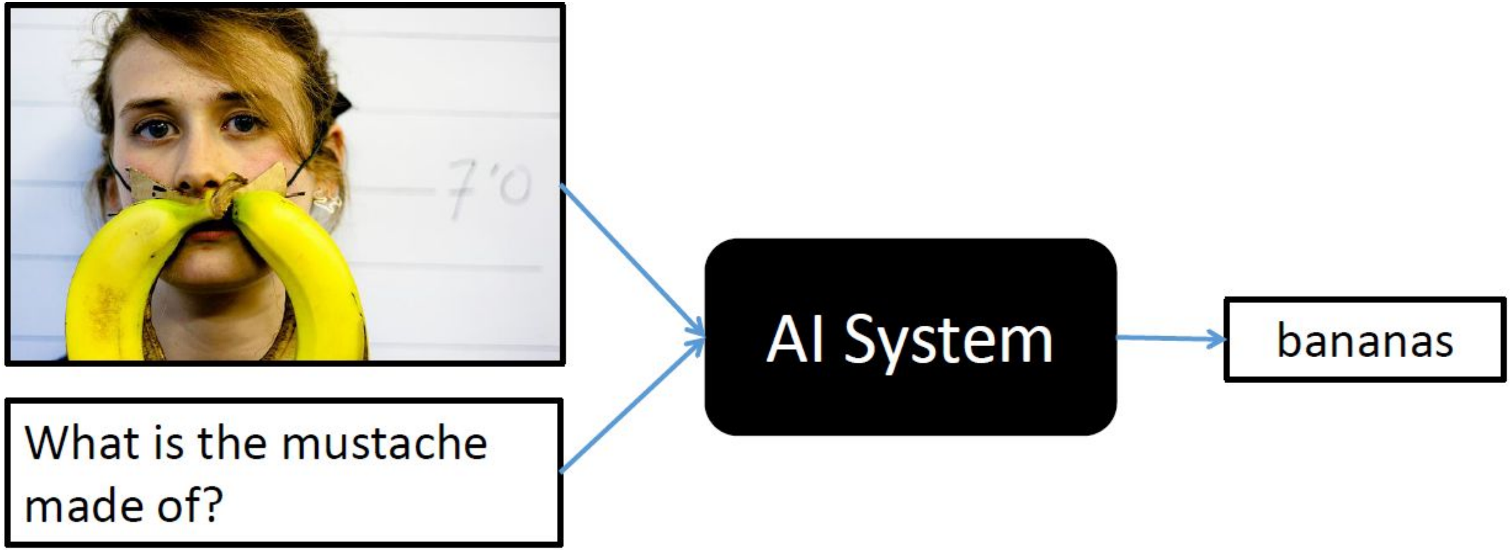
Learning to Reason: End-to-End Module Networks for Visual Question Answering

(Hu & Andreas, 2017)

Andrii Skliar, Gabriele Bani and Andreas Hadjipieri

The Problem: VQA



Belief: multi-modal tasks like image captioning are a closer step to solving AI.

Drawbacks of previous approaches

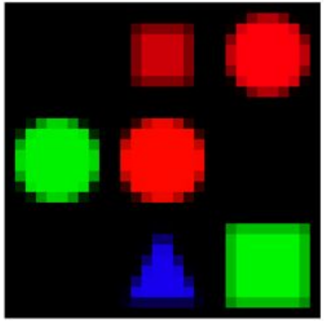
General VQA algorithms:

- N-grams combined with simple scene understanding give sufficient results
- Neural networks do not explicitly perform reasoning
- Often, the models fit to dataset bias
- Black boxes
- With standard datasets, no complex reasoning is required to get high scores on the datasets

Neural Module Networks

Pros:

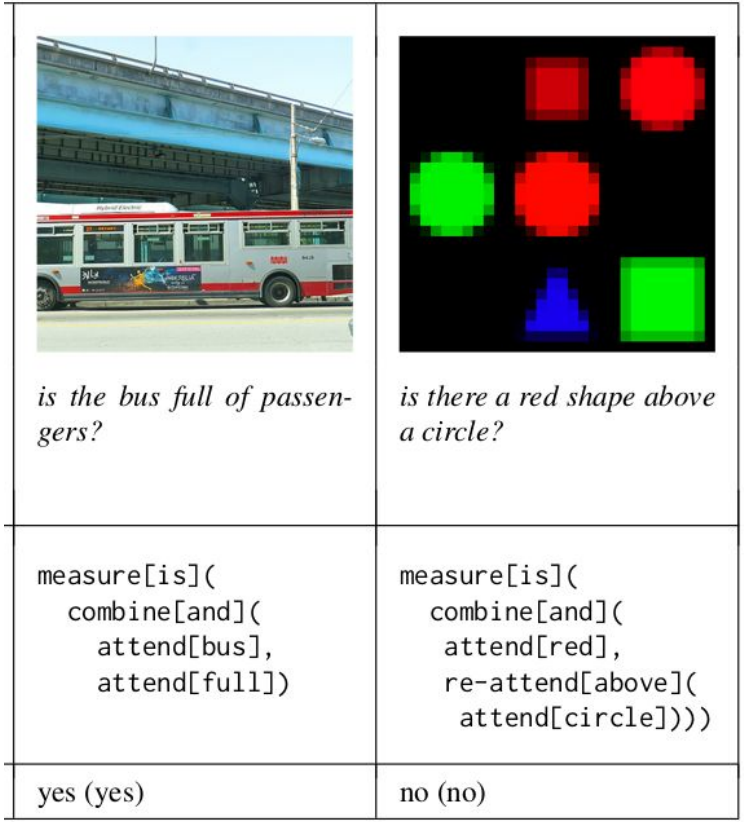
- Good generalization on the length of modules
- Compositional reasoning
- Good with SQL-like languages

	
is the bus full of passengers?	*is there a red shape above a circle?*
`measure[is]( combine[and]( attend[bus], attend[full])`	`measure[is]( combine[and]( attend[red], re-attend[above]( attend[circle])))`
yes (yes)	no (no)

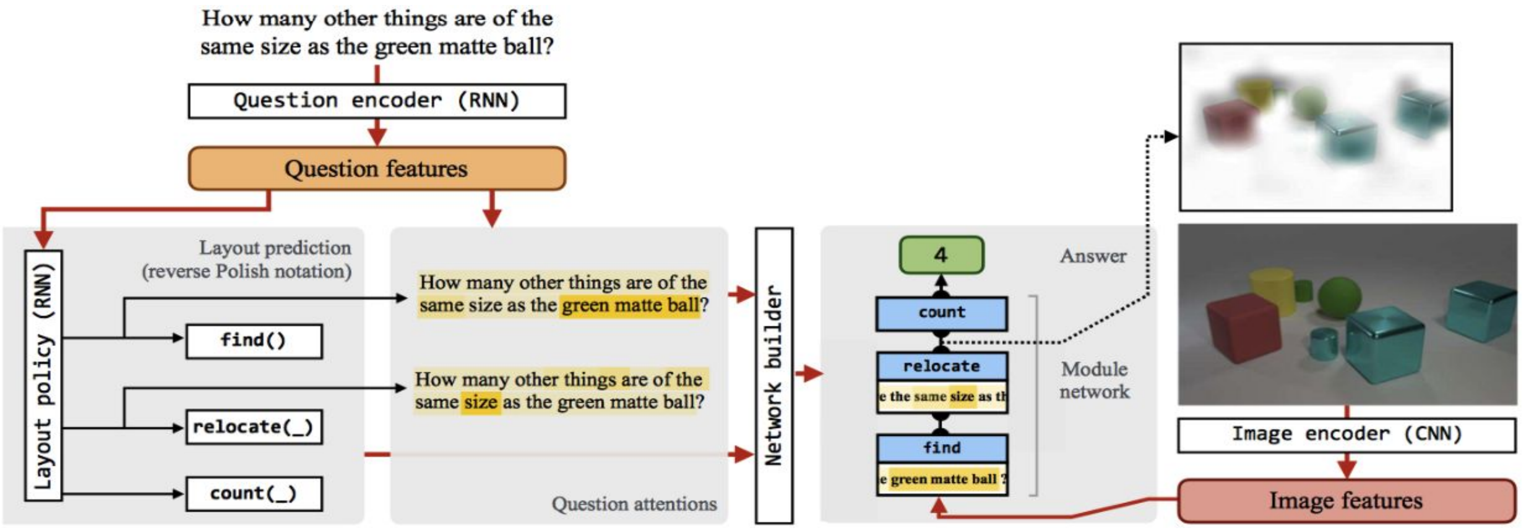
Neural Module Networks

Drawbacks:

- Use of a fixed parser to find the layout structure
- Does not generalize to new concepts
- Hard coded textual components (Every module has different parameters for every word)



End-to-End Module Networks



Two main components:

- Set of co-attentive neural modules that provide parameterized functions for solving sub-tasks
- Layout policy to predict a question specific layout from which a neural network is dynamically assembled.

Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

- Informally, a neural attention mechanism equips a neural network with the ability to focus on a subset of its inputs (or features)

- Attention can be applied to any kind of inputs, regardless of their shape. In the case of matrix-valued inputs, such as images, we can talk about visual attention.

$$\mathbf{a} = f_\phi(\mathbf{x}),$$

$$\mathbf{g} = \mathbf{a} \odot \mathbf{z},$$

Attentional Neural Modules

Module name	Att-inputs	Features	Output	Implementation details
`find`	(none)	x_{vis}, x_{txt}	att	$a_{out} = \text{conv}_2\left(\text{conv}_1(x_{vis}) \odot W x_{txt}\right)$
`relocate`	a	x_{vis}, x_{txt}	att	$a_{out} = \text{conv}_2\left(\text{conv}_1(x_{vis}) \odot W_1 \text{sum}(a \odot x_{vis}) \odot W_2 x_{txt}\right)$
`and`	a_1, a_2	(none)	att	$a_{out} = \text{minimum}(a_1, a_2)$
`or`	a_1, a_2	(none)	att	$a_{out} = \text{maximum}(a_1, a_2)$
`filter`	a	x_{vis}, x_{txt}	att	$a_{out} = \text{and}(a, \texttt{find}[x_{vis}, x_{txt}]())$, *i.e.* reusing `find` and `and`
`[exist, count]`	a	(none)	ans	$y = W^T \text{vec}(a)$
`describe`	a	x_{vis}, x_{txt}	ans	$y = W_1^T \left(W_2 \text{sum}(a \odot x_{vis}) \odot W_3 x_{txt}\right)$
`[eq_count, more, less]`	a_1, a_2	(none)	ans	$y = W_1^T \text{vec}(a_1) + W_2^T \text{vec}(a_2)$
`compare`	a_1, a_2	x_{vis}, x_{txt}	ans	$y = W_1^T \left(W_2 \text{sum}(a_1 \odot x_{vis}) \odot W_3 \text{sum}(a_2 \odot x_{vis}) \odot W_4 x_{txt}\right)$

- Parametrized function:
  - Inputs:
    - zero, one or multiple tensors (attention maps over the convolutional image feature grid)
    - internal parameters from the image
    - question to perform some computation on the input
  - Outputs:
    - image attention map
    - probability distribution over possible answers
- For each module, we predict an attention map over the question words and obtain the textual feature for each module:

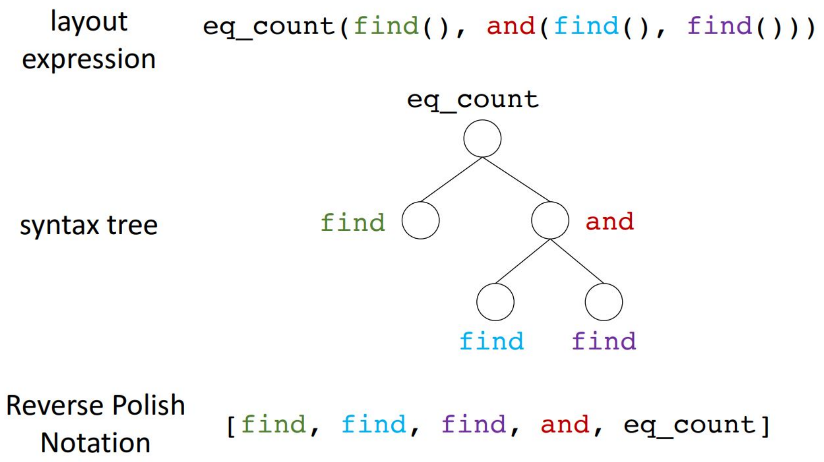
$$x_{txt}^{(m)} = \sum_{i=1}^{T} \alpha_i^{(m)} w_i$$

Layout prediction - seq2seq model

- Model the probability $p(l|q)$ for every possible layout

- Transform the layout into reverse polish notation

- Output the reverse Polish notation of the best sequence of modules.



Layout prediction - seq2seq model

Transform the layout into reverse polish notation with the help of Attentional Recurrent Neural Network

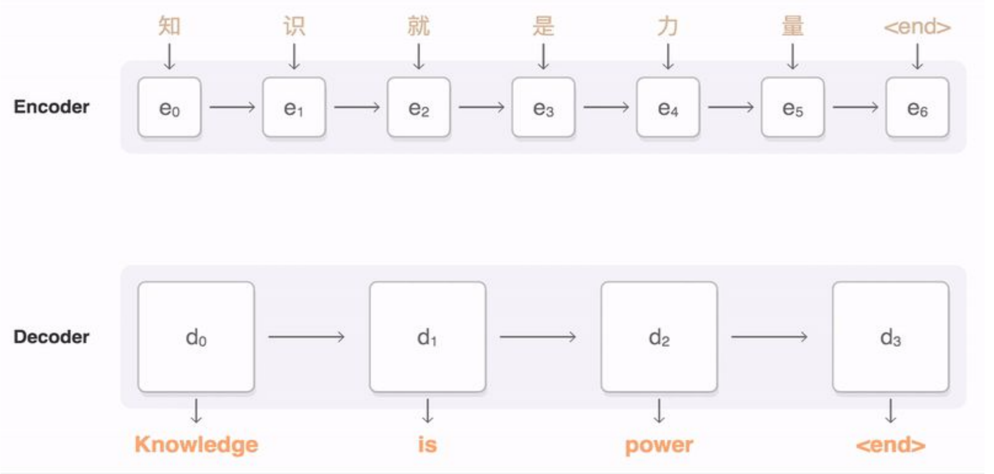
- Encode the words of the question and the layout tokens, into embeddings.

- Encoder:
    - Inputs: T word embeddings
    - Outputs: T sized sequence

- Decoder:
    - Inputs: T sized sequence
    - Outputs: T sized sequence of modules

$$u_{ti} = v^T \tanh(W_1 h_i + W_2 h_t)$$

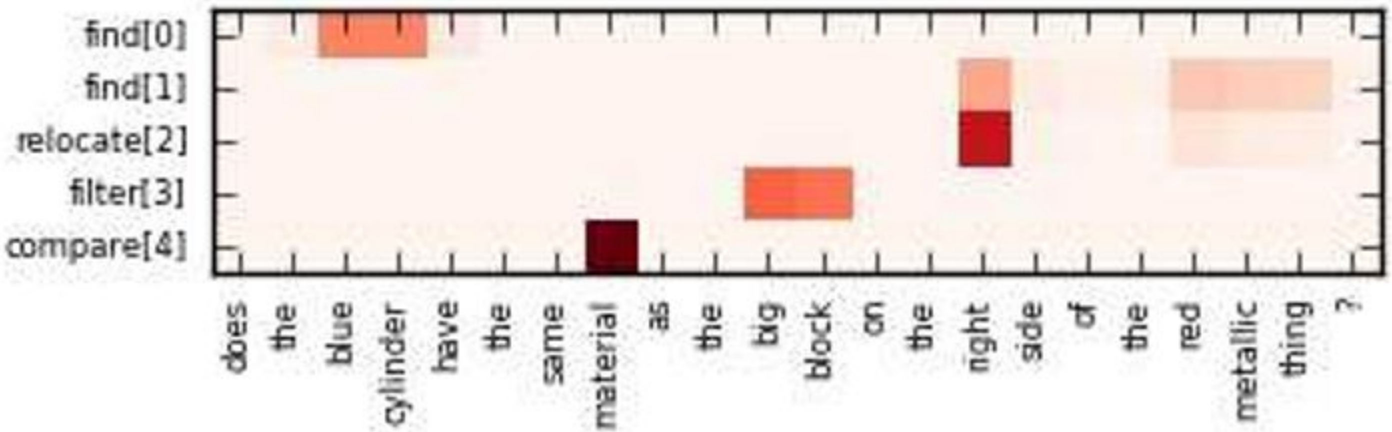
$$\alpha_{ti} = \frac{\exp(u_{ti})}{\sum_{j=1}^{T} \exp(u_{tj})}$$



Layout prediction - seq2seq model

2. Output the reverse Polish notation of the best sequence of modules.

- A context vector is obtained as $\sum_{i=1}^{T} \alpha_{ti} h_i$
- The next module probability is $p(m^{(t)} | m^{(1)}, \cdots, m^{(t-1)}, q) = \mathrm{softmax}(W_3 h_t + W_4 c_t)$
- We sample from this distribution to get the next token and construct the textual input $x_{txt}^{(t)}$



Layout prediction - seq2seq model

- Train Time :
    - Seq2seq outputs distribution: $p(l|q) = \prod_{m^{(t)} \in l} p(m^{(t)}|m^{(1)}, \cdots, m^{(t-1)}, q)$
    - Sample from the distribution to get the high probability layout

- Test Time:
    - Deterministically predict a maximum-probability layout from $p(l|q)$ sing beam search

End-to-End training

- Layout policy $p(l|q)$
- Parameters of the modules θ

Given a question q and an image I, the answer loss is given by the cross entropy loss over answer scores $\tilde{L}(\theta, l; q, I)$

Since we sample from the layout policy, the total loss is

$$L(\theta) = E_{l \sim p(l|q;\theta)}[\tilde{L}(\theta, l; q, I)]$$

End-to-End training - Monte-Carlo Sampling

- $L(\theta)$ is not fully differentiable, because we sample a discrete layout.
- We can train only the modules by backprop, not the seq2seq model
- Use policy gradient to calculate the loss
- Use Monte-Carlo sampling to estimate the policy gradient

$$\nabla_\theta L \approx \frac{1}{M} \sum_{m=1}^{M} \left(\tilde{L}(\theta, l_m) \nabla_\theta \log p(l_m | q; \theta) + \nabla_\theta \tilde{L}(\theta, l_m) \right)$$

Behavioral cloning from expert policies

- Learning the seq2seq RNN and neural modules parameters together from scratch is very challenging

Idea:

- Use an already existing “expert” policy, which produces reasonable candidate layouts
- Train to imitate the expert policy
- Minimize the KL divergence between the expert policy and our policy, while also minimizing the question answering loss

Behavioral cloning from expert policies

- Provides a good set of initial parameters
- After this, train End-to-End as described
- Only used at training time
- Produces much better results!

Analysis on the SHAPES dataset

Dataset Structure:

- 15616 image-question pairs with 244 unique questions
- Each image consists of shapes of different colors and sizes aligned on a 3 by 3 grid.
- Ground-truth parsing result for each question

Training and Results:

- Simple randomly initialized two-layer convolutional neural network to extract visual features from the image, trained together with other parts of our model.
- Model ("ours - behavioral cloning from expert") already achieves 100% accuracy.
- Model achieves a good performance on this dataset by performing policy search from scratch

Method	Accuracy
NMN [3]	90.80%
ours - behavioral cloning from expert	100.00%
ours - policy search from scratch	96.19%

Table 2: Performance of our model on the SHAPES dataset.

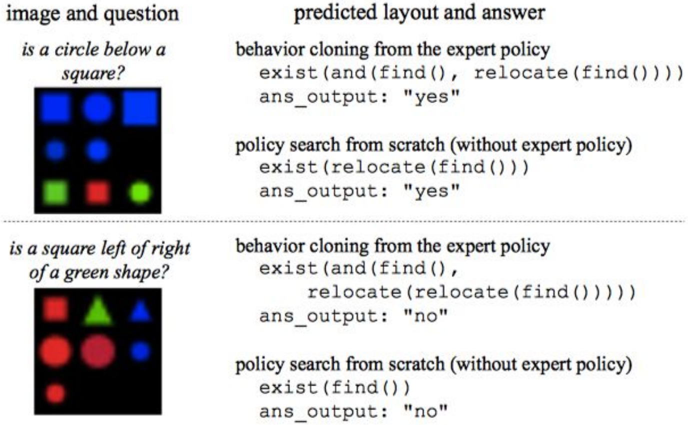
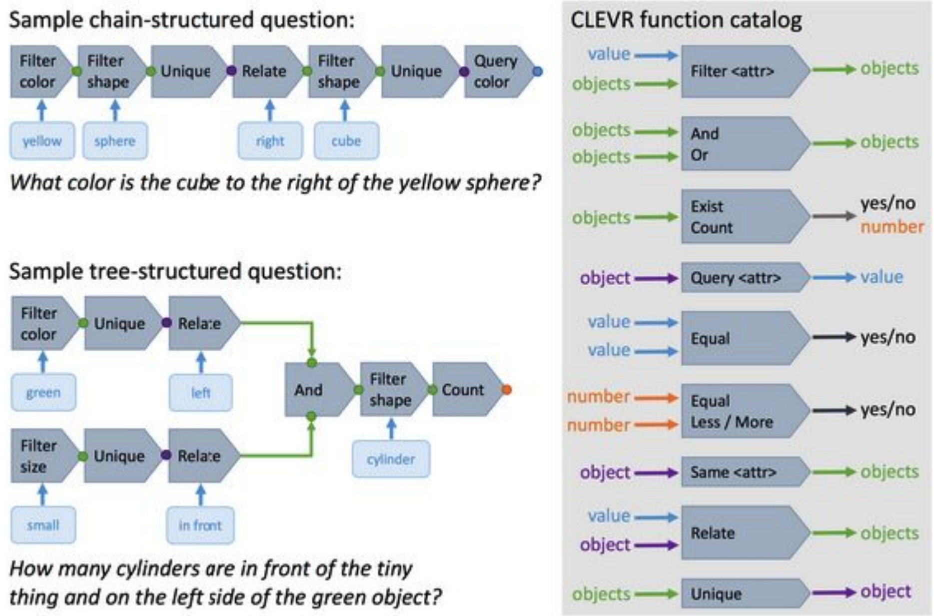


Figure 4: Examples of layouts predicted by our model on the SHAPES dataset, under two training settings (Sec. 4.1).

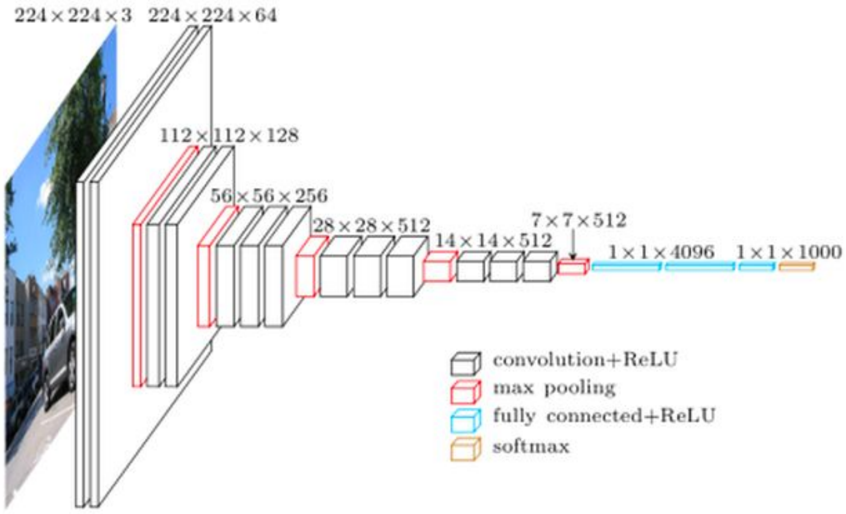
CLEVR dataset structure

- 100,000 images
- 853,554 questions
- Images are photorealistic rendered images with objects of different shapes, colors, materials and sizes and possible occlusions
- Questions are synthesized with functional programs
- Questions have much longer question length, and require handling long and complex inference chains to get an answer

Each question in CLEVR is represented both in **natural language** and as a **functional program**. The functional program representation allows for precise determination of the reasoning skills required to answer each question.

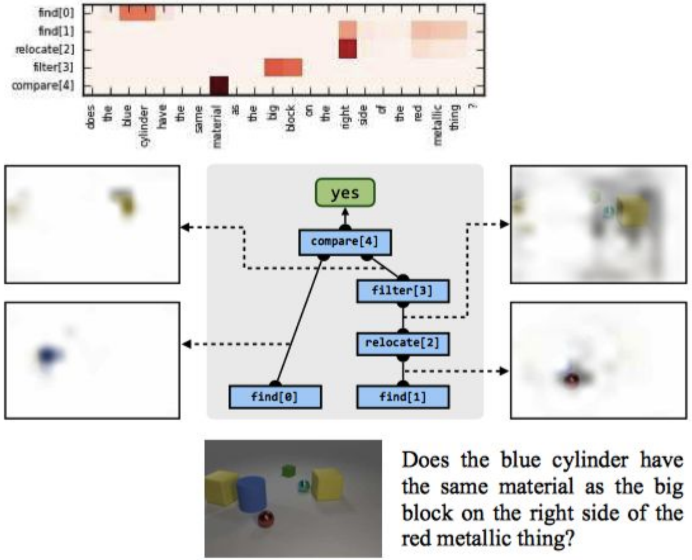


CLEVR dataset training



- Resize each image to 480 × 320
- Extract a 15 × 10 convolutional feature map from each image using VGG-16 network trained on ImageNET classification, and take the 512-channel pool5 output
- Add two extra dimensions $x = \frac{i}{15}$ and $y = \frac{j}{10}$ to each location (i, j), so the final visual feature on each image is a 15 × 10 × 514 tensor
- Construct an expert layout policy, which converts the annotated functional programs in this dataset into a module layout with manually defined rules
- During behavioral cloning, train model with two losses added together:
  - KL-divergence
  - Question answering loss
- After the first training stage, we discard the expert policy and continue to train our model with end-to-end reinforcement learning.

Evaluation of CLEVR dataset



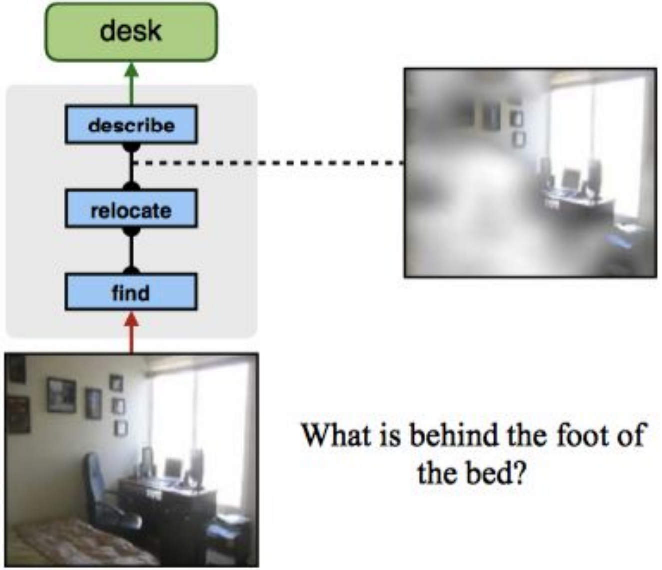
				Compare Integer			Query Attribute				Compare Attribute			
Method	**Overall**	Exist	Count	equal	less	more	size	color	material	shape	size	color	material	shape
CNN+BoW [26]	48.4	59.5	38.9	50	54	49	56	32	58	47	52	52	51	52
CNN+LSTM [4]	52.3	65.2	43.7	57	72	69	59	32	58	48	54	54	51	53
CNN+LSTM+MCB [9]	51.4	63.4	42.1	57	71	68	59	32	57	48	51	52	50	51
CNN+LSTM+SA [25]	68.5	71.1	52.2	60	82	74	87	81	88	85	52	55	51	51
NMN (expert layout) [3]	72.1	79.3	52.5	61.2	77.9	75.2	84.2	68.9	82.6	80.2	80.7	74.4	77.6	79.3
ours - policy search from scratch	69.0	72.7	55.1	71.6	85.1	79.0	88.1	74.0	86.6	84.1	50.1	53.9	48.6	51.1
ours - cloning expert	78.9	83.3	63.3	68.2	87.2	85.4	90.5	80.2	88.9	88.3	89.4	52.5	85.4	86.7
ours - policy search after cloning	**83.7**	**85.7**	**68.5**	**73.8**	**89.7**	**87.7**	**93.1**	**84.8**	**91.5**	**90.6**	**92.6**	**82.8**	**89.6**	**90.0**

Evaluation on the VQA dataset

- Construct an expert layout policy using a syntactic parse of questions

- Extract visual features from the ResNet-152 and train the model similarly to CLEVR dataset

- Second training stage of policy search after cloning does not lead to noticeable improvement in the accuracy



Method	Accuracy
MCB [9]	64.7
NMN [3]	57.3
D-NMN [2]	57.9
ours	64.2

Future Research

- DDRprog; IEP: Generating programs consisting of modules with textual parameters hard-coded in module instantiation.

- FiLM + CBN: learn directly from language and image inputs and without an architectural prior on modularity.

- RN: neural network module with a structure primed for relational reasoning.

Table 1. Accuracy on all CLEVR question types for baselines and competitive models. The Human baseline is from the original CLEVR work. * denotes additional program supervision. SA refers to stacked spatial attention (Yang et al., 2015)

Model	Parameters	Epochs	Exist	Count	Compare Integer	Query	Compare	Overall
End-to-End NMN*	-	-	85.7	68.5	84.9	89.9	88.7	83.7
IEP*	41M	12	97.1	92.7	**98.7**	98.1	98.8	96.9
DDRprog*	9M	52	98.8	**96.5**	98.4	99.1	**99.0**	**98.3**
RN	500k	1000	97.8	90.1	93.6	97.9	97.1	95.5
FiLM/CBN	>50M	80	**99.2**	94.5	93.8	**99.2**	**99.0**	97.6

Summary

- Fully End-to-End training of module networks
- No fixed parser required
- Capable of very complex compositional reasoning
- Hard to train from scratch, because of RL
- Good initial policies can greatly boost results

Thank you for the Attention!