Neural Module Networks

Berthy Feng

Outline

Presentation Overview

1. Neural Module Networks
2. Learning to Reason: End-to-End Neural Module Networks

Neural Module Networks

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. CVPR, 2016.

Motivations

Neural Module Networks

1. VQA Overview
2. Compositional Visual Intelligence

VQA Overview

$$(I, Q) \rightarrow A$$

Previous approaches:

1. Classifier on encoded Q + I
2. Semantic parsers → logical expressions

QA: (What is behind the table?, window)

VQA: Problems with Previous Approaches

- Classifier on Encoded Question + Image
  - **Monolithic** — difficult to understand/explain models
  - **Limited visual reasoning** — probably just memorizing statistics
  - **Limited linguistic reasoning** — bag-of-words, RNN can't capture linguistic complexity
- Logical Expressions
  - Purely logical representations of world might ignore visual features and attentions
  - Semantic parsers not optimized for VQA

Insight: Compositional Nature of VQA

Each question requires different # of reasoning steps and different *kinds* of reasoning

“Is this a truck?”

1. Classify
- Convolutional

“How many objects are to the left of the toaster?”

1. Find toaster
2. Look left
3. Find objects
4. Count
- Convolutional
- Recurrent

Insight: Compositional Nature of VQA

Questions are composed of “reasoning modules” and might share substructures

Insight: Combine Both Approaches

Representational Approaches

Neural network structures are not universal, but at least modular in applications

Compositional Approaches

Logical expressions and functional programs provide computational structure

Proposed Approach

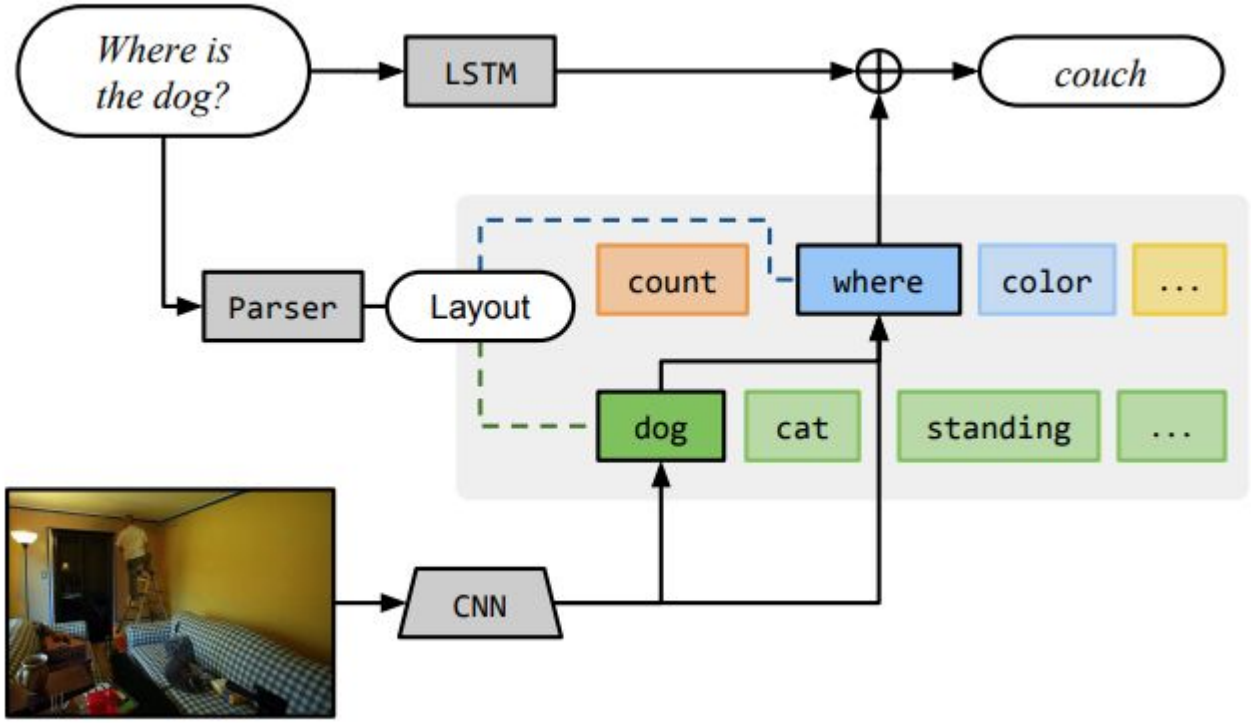
1. Predict computational structure from question
2. Construct modular neural network from this structure

Neural Modules

Neural Module Networks

1. Neural Module Networks
2. Module Types

Neural Module Networks



Model = collection of modules + network layout predictor

Output = distribution over answers

$$p(y \mid w,\ x;\ \theta)$$

Modules

Data Types:

1. Images
2. Attentions
3. Labels

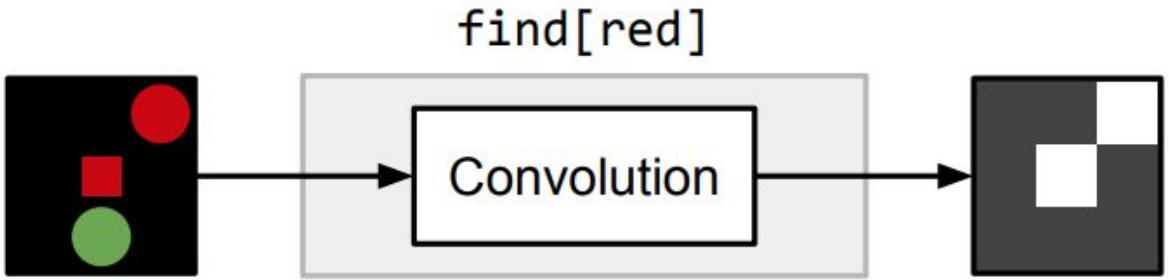
Module Types:

- `FIND`
- `TRANSFORM`
- `COMBINE`
- `DESCRIBE`
- `MEASURE`

$$\texttt{TYPE[INSTANCE]}(\texttt{ARG}_1, \ldots)$$

Find Module

`Image → Attention`

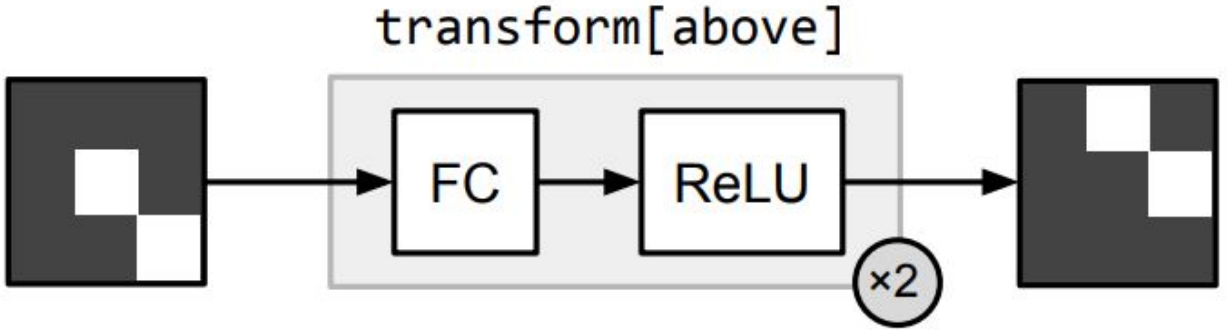


Convolve every position in input image w/ weight vector (distinct for each instance) to produce attention heatmap

`find[dog]` = matrix with high values in regions containing dogs

Transform Module

`Attention` → `Attention`

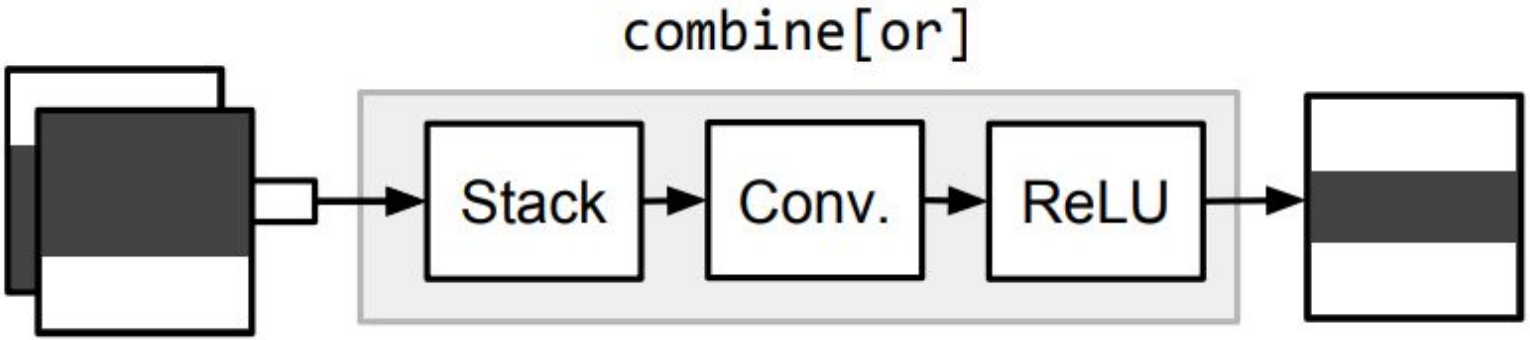


Fully connected mapping from one attention to another (MLP w/ ReLUs)

`transform[not]`: move attention away from active regions

Combine Module

`Attention x Attention → Attention`

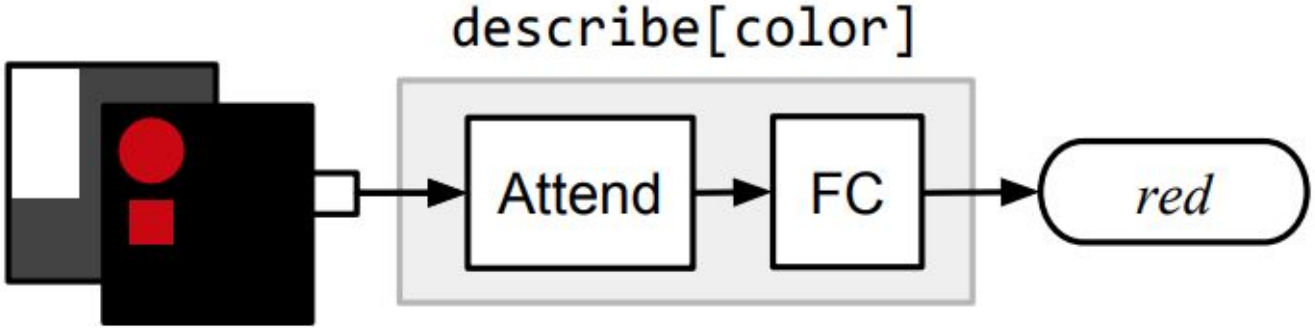


Merge two attentions into one

`combine[and]`: activate regions active in both inputs

Describe Module

`Image x Attention → Label`

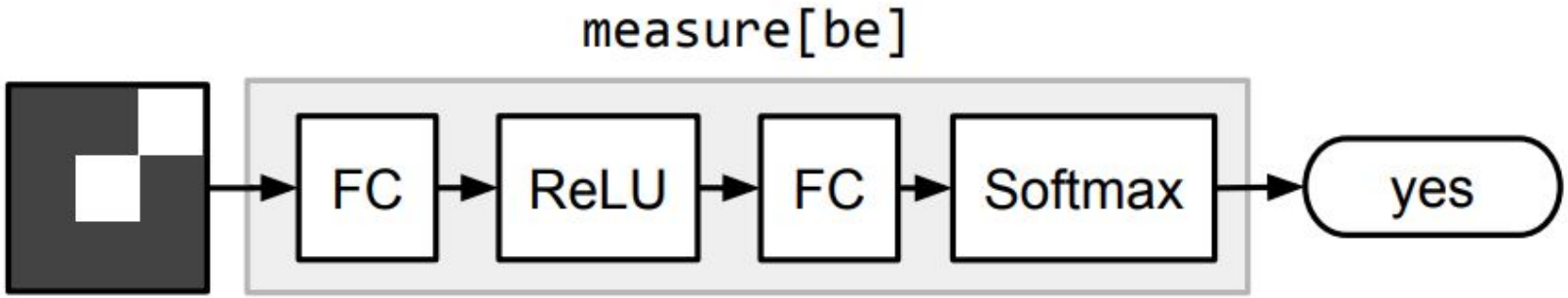


Average image features weighted by attention, then pass averaged feature vector through FC layer

`describe[color]` = representation of colors in the region attended to

Measure Module

Attention → Label



Map attention to a distribution over labels

Can evaluate existence of detected object or count sets of objects

Question Parsing

Neural Module Networks

1. Question Parsing
2. Question Tree
3. Question Encoding
4. Predicting an Answer
5. Note on Training

Question Parsing

Question → dependency representation

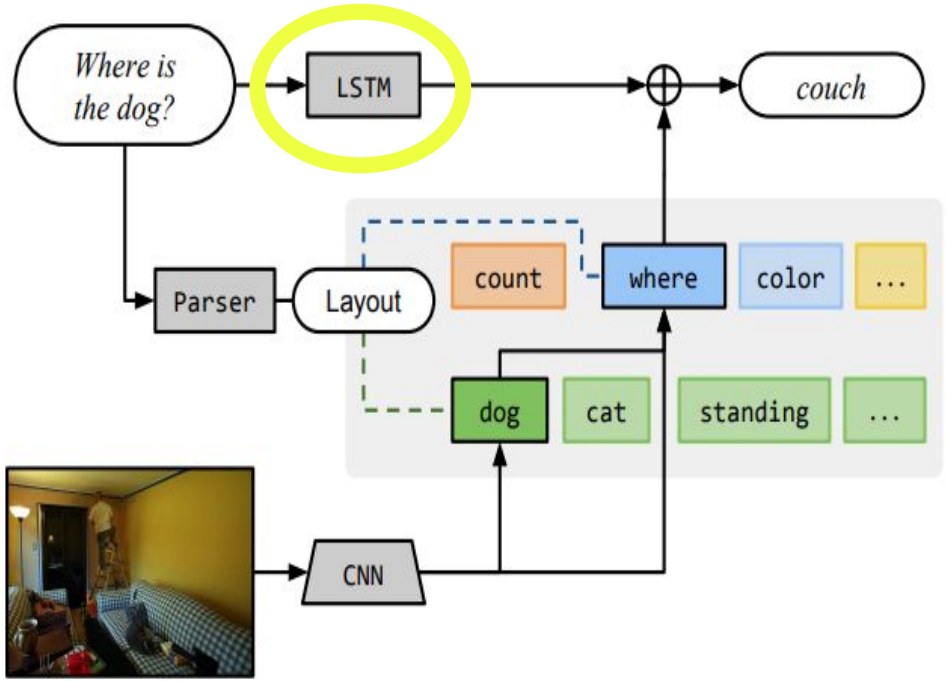
Filter dependencies to those connected by *wh*-word or connecting verb

“What is standing in the field?”	what(stand)
“What color is the truck?”	color(truck)
“Is there a circle next to a square?”	is(circle, next-to(square))

Question Tree

leaves:	`find`
internal nodes:	`transform, combine`
root nodes:	`describe, measure`

Question Encoding

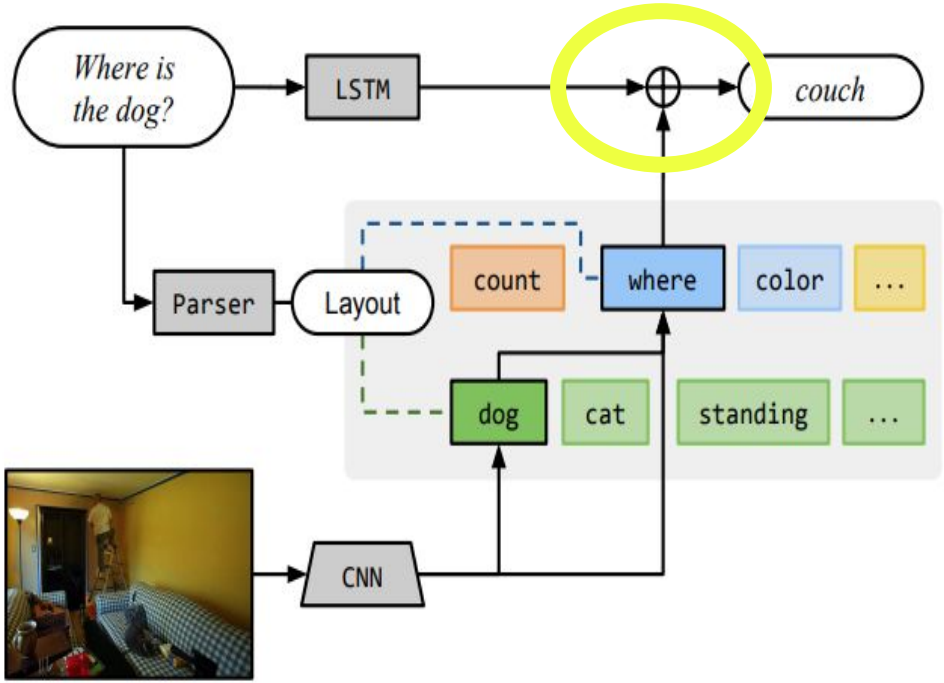


Full model = NMN +

LSTM question encoder

- LSTM models attributes lost in parsing: “underlying syntactic, semantic regularities in the data”
- Single-layer LSTM w/ 1000 hidden units

Predicting an Answer



1. Pass final hidden state of LSTM through FC layer
2. Add output elementwise to representation produced by root module of NMN
3. Apply ReLU, another FC layer, and get distribution over possible answers

$$p(y \mid w,\ x;\ \theta)$$

Training

Modules jointly trained

Some weights updated more frequently than others

- Use adaptive per-weight learning rates (ADADELTA)

Experiments

Neural Module Networks

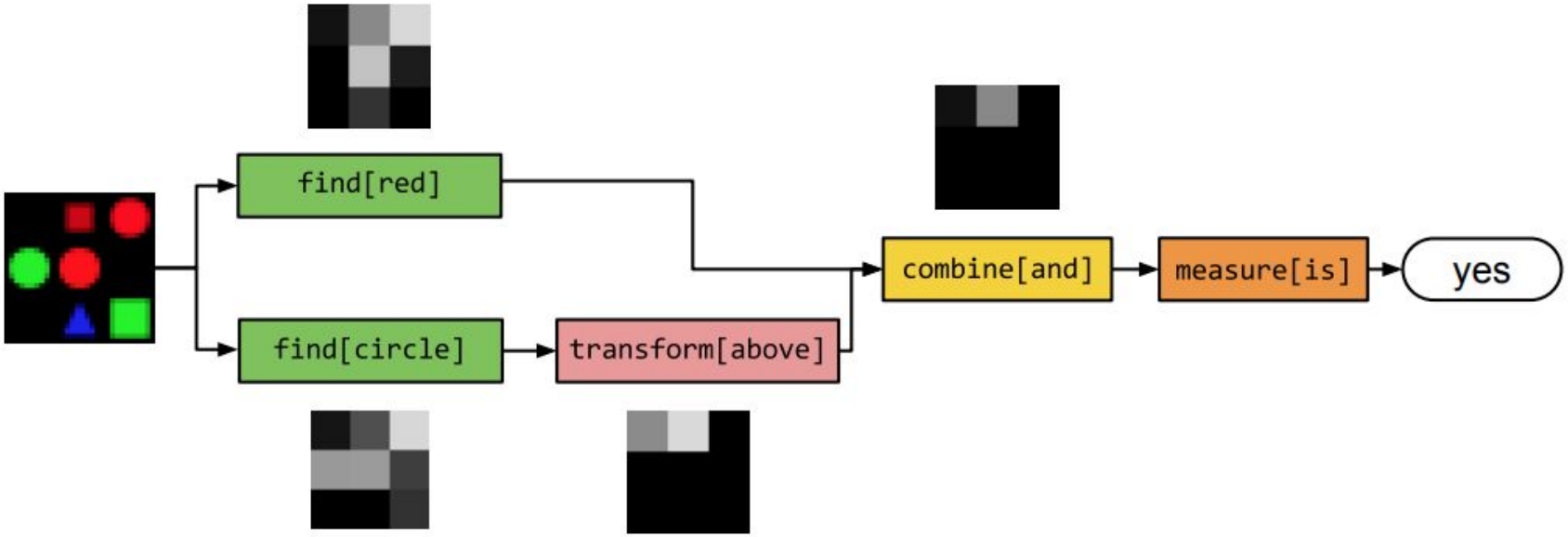
1. Evaluation on SHAPES
2. Evaluation on VQA
3. Future Work

SHAPES Dataset

- Synthetic dataset
- Complex questions about simple arrangements of colored shapes
- To prevent mode-guessing, all answers yes/no

	types	# instances	# layouts	max depth	max size
VQA	`find, combine, describe`	877	51138	3	4
SHAPES	`find, transform, combine, measure`	8	164	5	6

Is there a red shape above a circle?



Performance on SHAPES

modules in neural module network

	size 4	size 5	size 6	All
% of test set	31	56	13	
Majority	64.4	62.5	61.7	63.0
VIS+LSTM	71.9	62.5	61.7	65.3
NMN	89.7	92.4	85.2	**90.6**
NMN (train size ≤ 5)	97.7	91.1	89.7	**90.8**

VQA Dataset

- Contains over 200,000 images from MS COCO
- Each image annotated with 3 questions and 10 answers/question

93213. COCO_val2014_000000212462

Show Image

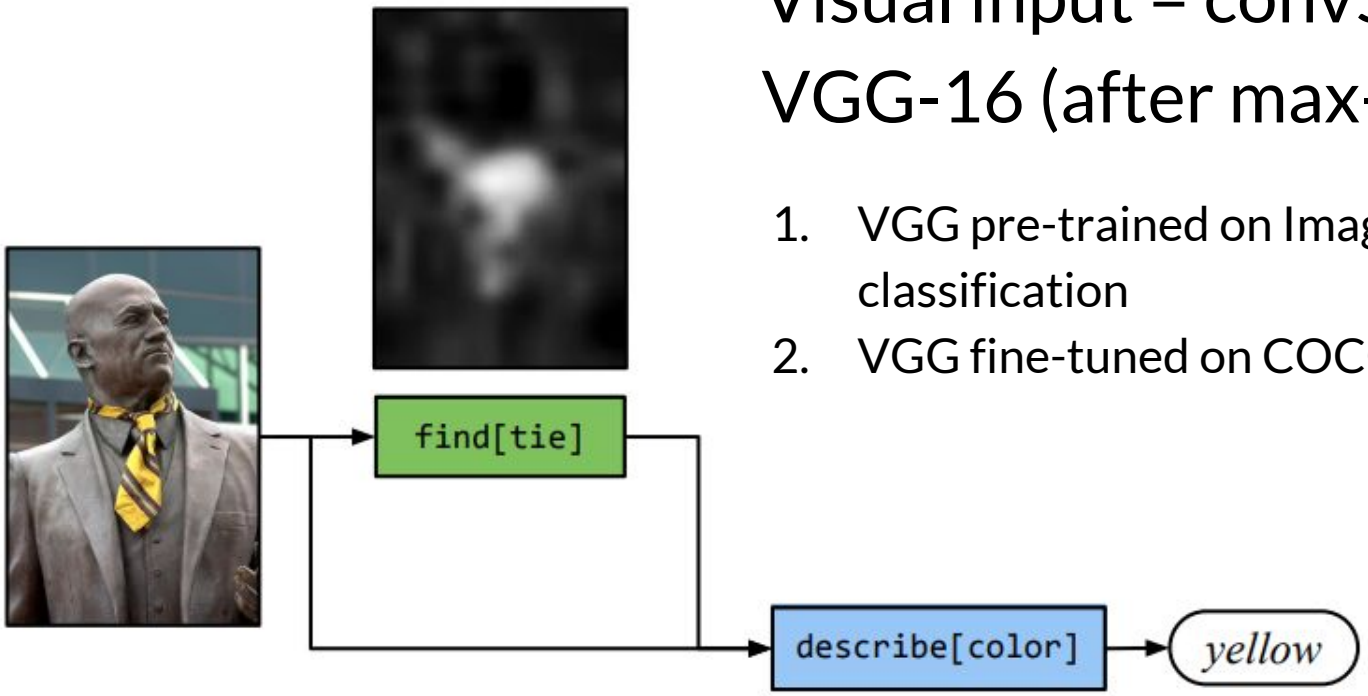
Open-Ended | Multiple-Choice | Ground-Truth | Common-Sense | Captions

Q: Is the fruit cut up?

Q: Is the fruit on a plate?

Q: What kinds of fruit are shown?

What color is his tie?



Visual input = conv5 layer of VGG-16 (after max-pooling)

1. VGG pre-trained on ImageNet classification
2. VGG fine-tuned on COCO for captioning

Results on VQA

	test-dev			
	Yes/No	Number	Other	All
LSTM	78.7	36.6	28.1	49.8
VIS+LSTM [3] [2]	78.9	35.2	36.4	53.7
ATT+LSTM	80.6	36.4	42.0	57.2
NMN	70.7	36.8	39.2	54.8
NMN+LSTM	81.2	35.2	43.3	58.0
NMN+LSTM+FT	81.2	38.0	44.0	58.6

Does especially well on questions answered by object, attribute, or number

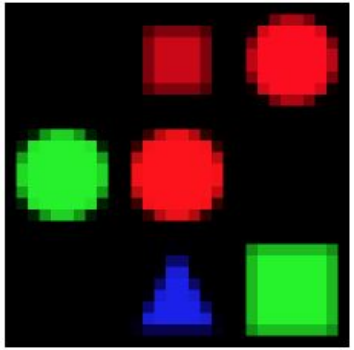
Good Example

what is the color of the horse?

```
describe[color](
  find[horse])
```

brown (brown)

Good Examples (2 of 2)

	
is the bus full of passengers?	*is there a red shape above a circle?*
`describe[is]( combine[and]( find[bus], find[full])`	`measure[is]( combine[and]( find[red], transform[above]( find[circle])))`
yes (yes)	yes (yes)

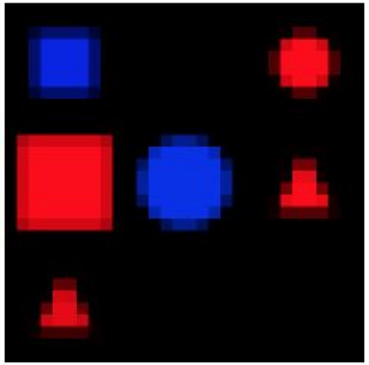
Bad Example

what is stuffed with toothbrushes wrapped in plastic?

```
describe[what](
  find[stuff])
```

container (cup)

Bad Examples (2 of 2)

	
is this a clock?	*is a red shape blue?*

`describe[is](`
`  find[clock])`
	`measure[is](`
	`  combine[and](`
	`  find[red],`
	`  find[blue]))`

yes (no)	yes (no)

Future Work

Better parser

“Are these people most likely experiencing a work day?”

- Parser: *be(people, likely)*
- Correct: *be(people, work)*

Combine predicting network structures and learning network parameters

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

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. ICCV, 2017.

End-to-End Module Networks

Learning to Reason

1. Motivations
2. Proposed Approach
3. Attentional Neural Modules

Motivations

- Standard neural module networks rely on external parser

  “Are these people most likely experiencing a work day?”

  Parser: *be(people, likely)*

- New datasets
  - CLEVR

Proposed Approach

Predict modular network architectures **directly** from textual input

1) Set of co-attentive neural modules

2) Layout policy that predicts question-specific network layout

Attentional Neural Modules

Neural module m := $f_m(a_1, a_2, \ldots; x_{vis}, x_{txt}, \theta_m)$

Each input tensor a_i = image attention map over convolutional feature grid

Output tensor y = image attention map or probability distribution over answers

Attentional Neural Modules

Module name	Att-inputs	Features	Output
`find`	(none)	x_{vis}, x_{txt}	att
`relocate`	a	x_{vis}, x_{txt}	att
`and`	a_1, a_2	(none)	att
`or`	a_1, a_2	(none)	att
`filter`	a	x_{vis}, x_{txt}	att
`[exist, count]`	a	(none)	ans
`describe`	a	x_{vis}, x_{txt}	ans
`[eq_count, more, less]`	a_1, a_2	(none)	ans
`compare`	a_1, a_2	x_{vis}, x_{txt}	ans

Attentional Neural Modules: Implementations

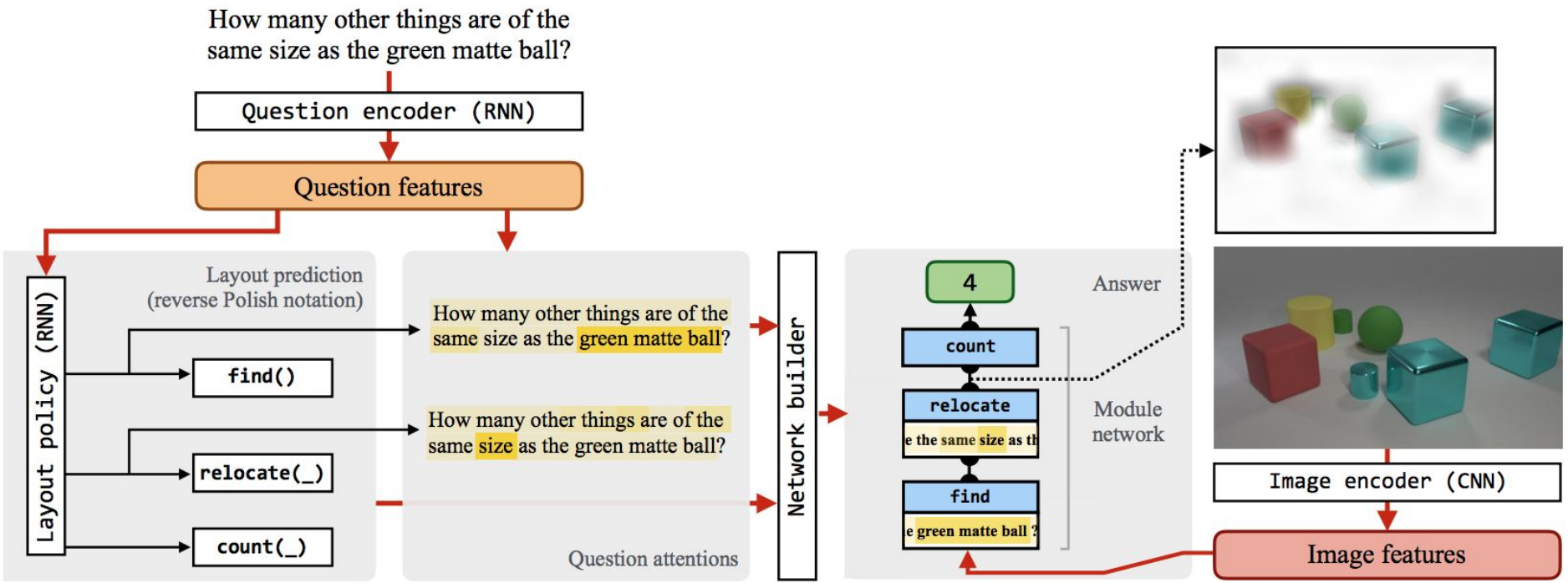
Module name	Implementation details
`find`	$a_{out} = \text{conv}_2\left(\text{conv}_1(x_{vis}) \odot W x_{txt}\right)$
`relocate`	$a_{out} = \text{conv}_2\left(\text{conv}_1(x_{vis}) \odot W_1 \text{sum}(a \odot x_{vis}) \odot W_2 x_{txt}\right)$
`and`	$a_{out} = \text{minimum}(a_1, a_2)$
`or`	$a_{out} = \text{maximum}(a_1, a_2)$
`filter`	$a_{out} = \textbf{and}(a, \textbf{find}[x_{vis}, x_{txt}]())$, *i.e.* reusing `find` and `and`
`[exist, count]`	$y = W^T \text{vec}(a)$
`describe`	$y = W_1^T \left(W_2 \text{sum}(a \odot x_{vis}) \odot W_3 x_{txt}\right)$
`[eq_count, more, less]`	$y = W_1^T \text{vec}(a_1) + W_2^T \text{vec}(a_2)$
`compare`	$y = W_1^T \left(W_2 \text{sum}(a_1 \odot x_{vis}) \odot W_3 \text{sum}(a_2 \odot x_{vis}) \odot W_4 x_{txt}\right)$

Architecture

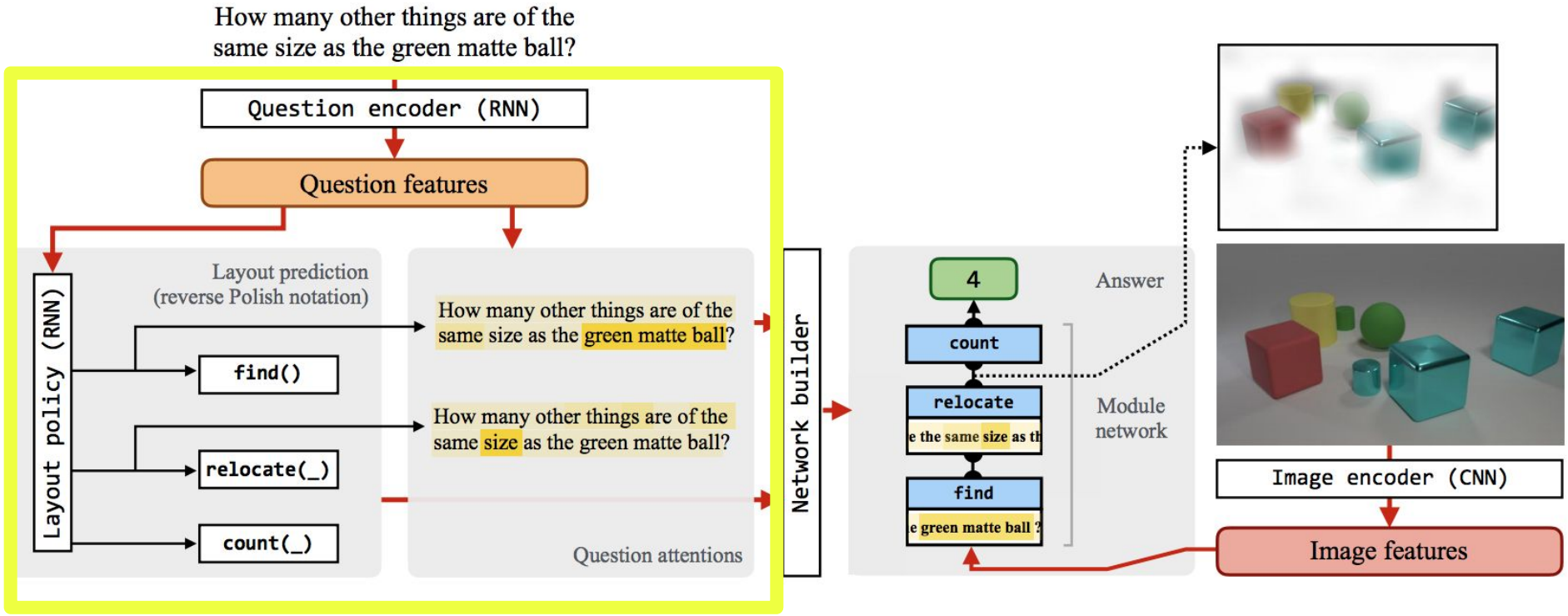
Learning to Reason

1. Model Overview
2. Layout Policy

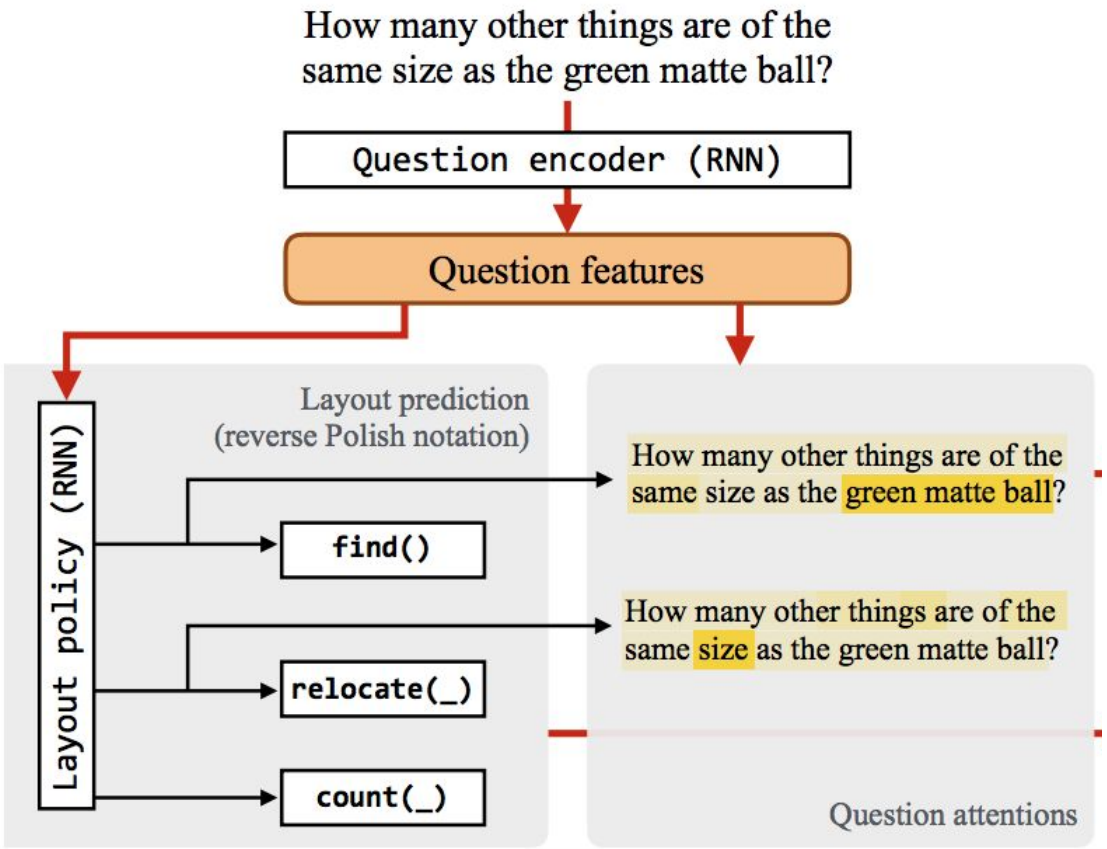
Model Overview



Model Overview



Layout Policy



Input: Question

Output: Probability distribution over layouts, $p(l \mid q)$

Layout Policy: Seq-to-Seq RNN (1 of 3)

1. **Encode question**
   a. Embed every word $i \rightarrow$ vector w_i (using GloVe)
   b. Multi-layer LSTM encodes input question $\rightarrow [h_1, h_2, \ldots, h_T]$, where there are T words in the question
2. Decoder LSTM
3. Predict next module at t

Layout Policy: Seq-to-Seq RNN (2 of 3)

1. Encode question → h_i's: [h_1, h_2, …, h_T]
2. **Decoder LSTM**
   a. Same structure as encoder, but different params
   b. At each time step *t*, predict attention weights a_{ti} of every input word using h_i (encoder output) and h_t (decoder output)

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

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

3. Predict next module at *t*

Layout Policy: Seq-to-Seq RNN (3 of 3)

1. Encode question $\rightarrow [h_1, h_2, \ldots, h_T]$
2. Decoder LSTM $\rightarrow a_{ti}$ = attention at time *t* for word *i*
3. **Predict next module at *t***
   a. Context vector $c_t = \sum_{i=1}^{T} \alpha_{ti} h_i$ (attention heatmap across question words at time *t*)
   b. Probability for next module token (encoded form of a module) $m^{(t)}$ predicted from h_t and c_t

$$p(m^{(t)}|m^{(1)}, \cdots, m^{(t-1)}, q) = \text{softmax}(W_3 h_t + W_4 c_t)$$

Layout Policy: Choosing a Layout

- Each time step *t* corresponds to word in input question
- For each *t*, sample from distribution over all modules to get next module token from $p(m^{(t)}|m^{(1)}, \cdots, m^{(t-1)}, q)$

- Probability of layout *l* is: $\prod_{m^{(t)} \in l} p(m^{(t)}|m^{(1)}, \cdots, m^{(t-1)}, q)$

Layout Policy: Textual Input (1 of 3)

Previous: Textual features hard-coded into module instance

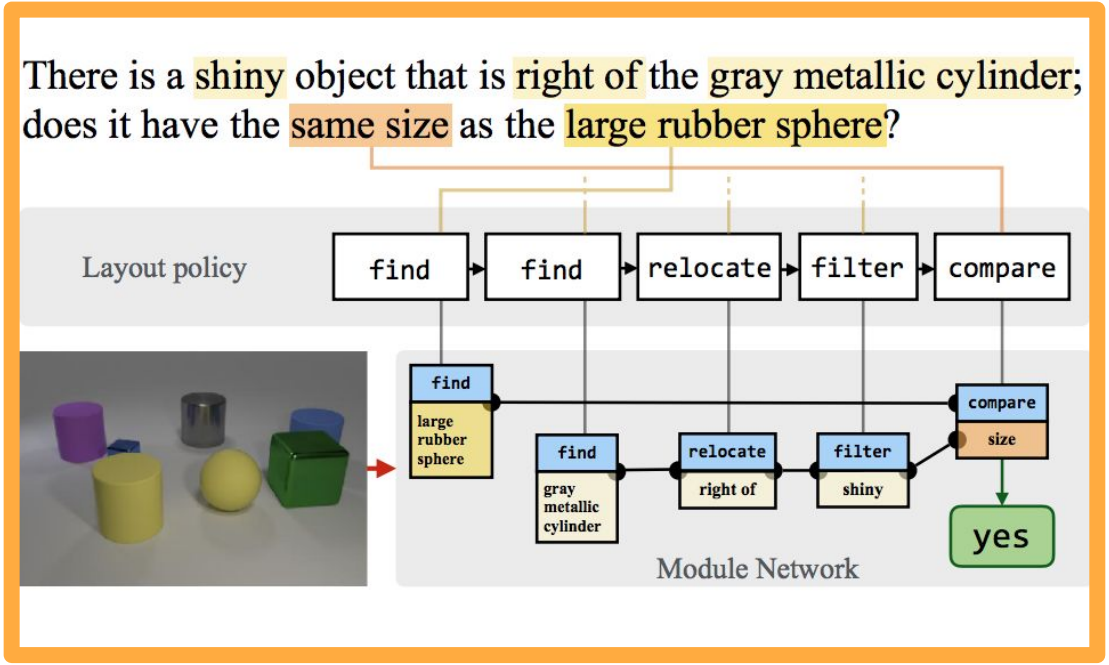
- `describe['shape']` and `describe['where']` different instantiations

Idea: Pass attention-based textual input to each module in network

- “shape” and “where” would be input to general `describe` module

Layout Policy: Textual Input (2 of 3)

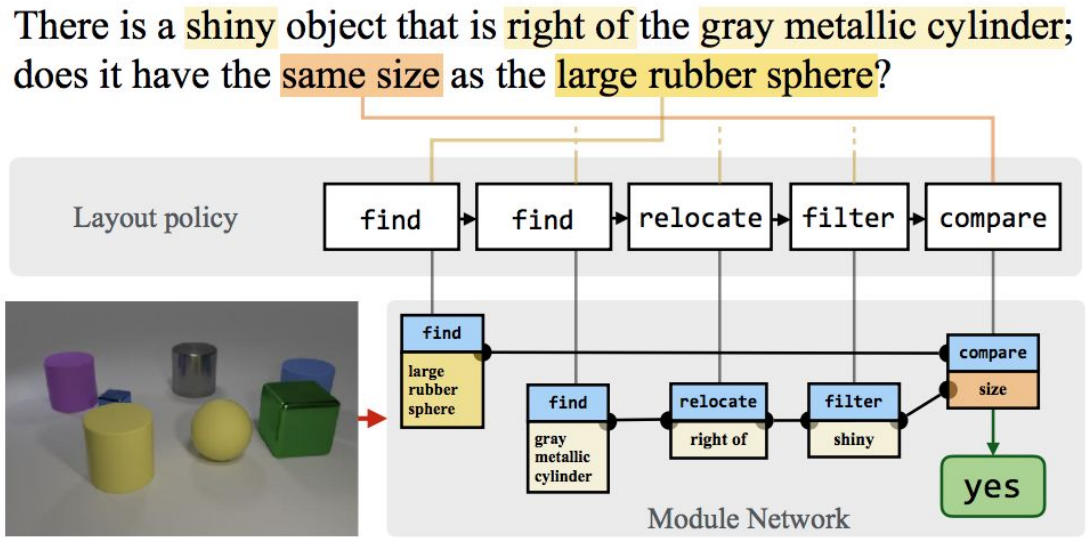
Simplified textual attention heat map:



Construct textual input $x_{txt}^{(t)}$ for each time step t, using a_{ti} from each word i

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

Layout Policy: Textual Input (3 of 3)



Construct textual input $x_{txt}^{(t)}$ for each time step t, using a_{ti} from each word i

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

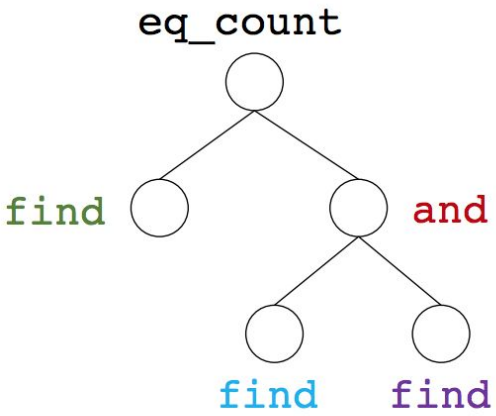
textual attention heatmap

Linearized Layout

layout expression

```
eq_count(find(), and(find(), find()))
```

syntax tree

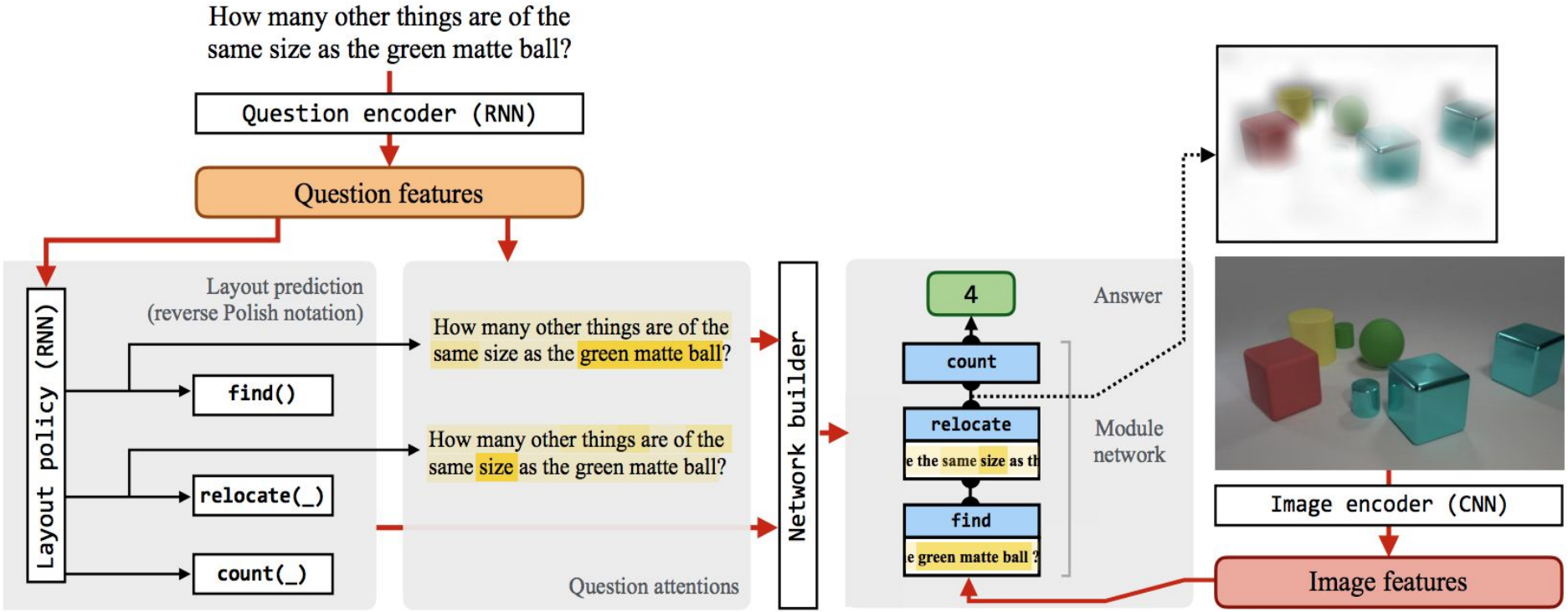


Reverse Polish Notation

```
[find, find, find, and, eq_count]
```

Reverse Polish notation lets us linearize functional programs

Model Overview (Revisited)



Training

Learning to Reason

1. Loss Function
2. Behavioral Cloning

Answer-Based Loss: Function

Training loss function:

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

Softmax loss over output answer scores

Answer-Based Loss: Backpropagation

Loss function not fully differentiable since layout *l* is discrete

- **Differentiable parts:** backpropagation
- **Non-differentiable parts:** policy gradient method in reinforcement learning

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

Behavioral Cloning (1 of 2)

- Difficult to train — 3 sets of params to learn
  - Seq-to-seq RNN params
  - Textual attention weights
  - Neural module params

Behavioral Cloning (2 of 2)

- Pre-train by behavioral cloning from **expert policy** p_e
  - Minimize KL-divergence between p_e and p (our layout policy)
  - Simultaneously minimize question-answering loss using layout l obtained from p_e

- After learning good initialization, further train model end-to-end using previous estimated gradient

Experiments

Learning to Reason

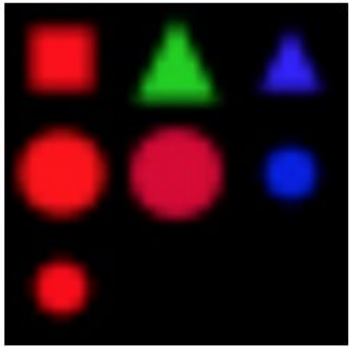
1. Evaluation on SHAPES
2. Evaluation on CLEVR
3. Evaluation on VQA

Evaluation on SHAPES

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

SHAPES Example

is a square left of right of a green shape?



behavior cloning from the expert policy

```
exist(and(find(),
    relocate(relocate(find()))))
ans_output: "no"
```

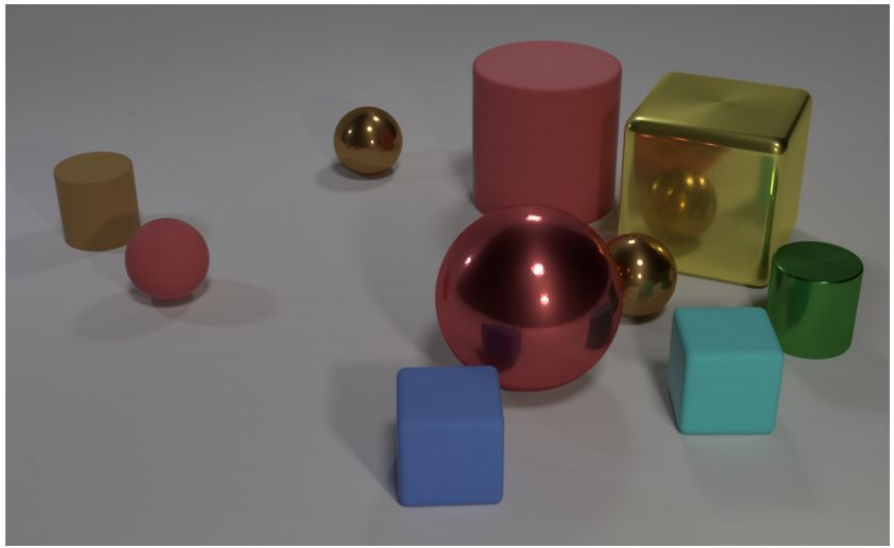
policy search from scratch (without expert policy)

```
exist(find())
ans_output: "no"
```

CLEVR Dataset

Johnson et al., 2016

- 100k images + 850k questions
- Questions synthesized with functional programs



Testing on CLEVR

- Visual inputs
  - For each image, extract 15x10x512 tensor from pool5 output of VGG-16 trained on ImageNet classification
  - Add two extra *x*, *y* dimensions to each location on feature map → 15x10x514 tensor
- Textual inputs
  - Each question word embedded to 300-dimensional vector
- Expert layout policy
  - Convert annotated functional programs in CLEVR into module layout

CLEVR Results

Method	**Overall**	Exist	Count
CNN+BoW [26]	48.4	59.5	38.9
CNN+LSTM [4]	52.3	65.2	43.7
CNN+LSTM+MCB [9]	51.4	63.4	42.1
CNN+LSTM+SA [25]	68.5	71.1	52.2
NMN (expert layout) [3]	72.1	79.3	52.5
ours - policy search from scratch	69.0	72.7	55.1
ours - cloning expert	78.9	83.3	63.3
ours - policy search after cloning	**83.7**	**85.7**	**68.5**

CLEVR Results

Method	Compare Integer		
	equal	less	more
CNN+BoW [26]	50	54	49
CNN+LSTM [4]	57	72	69
CNN+LSTM+MCB [9]	57	71	68
CNN+LSTM+SA [25]	60	82	74
NMN (expert layout) [3]	61.2	77.9	75.2
ours - policy search from scratch	71.6	85.1	79.0
ours - cloning expert	68.2	87.2	85.4
ours - policy search after cloning	**73.8**	**89.7**	**87.7**

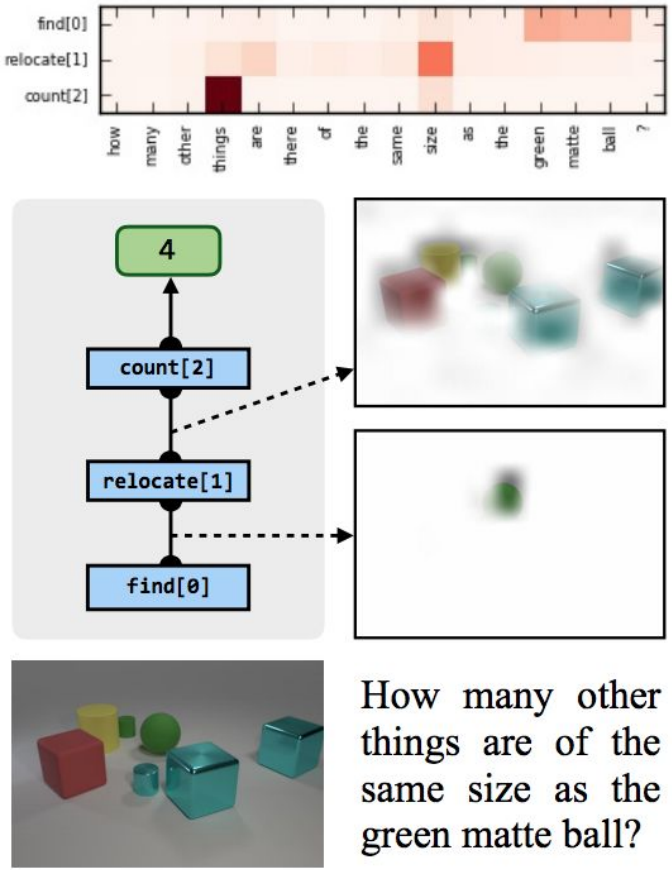
CLEVR Results

Method	Query Attribute			
	size	color	material	shape
CNN+BoW [26]	56	32	58	47
CNN+LSTM [4]	59	32	58	48
CNN+LSTM+MCB [9]	59	32	57	48
CNN+LSTM+SA [25]	87	81	88	85
NMN (expert layout) [3]	84.2	68.9	82.6	80.2
ours - policy search from scratch	88.1	74.0	86.6	84.1
ours - cloning expert	90.5	80.2	88.9	88.3
ours - policy search after cloning	**93.1**	**84.8**	**91.5**	**90.6**

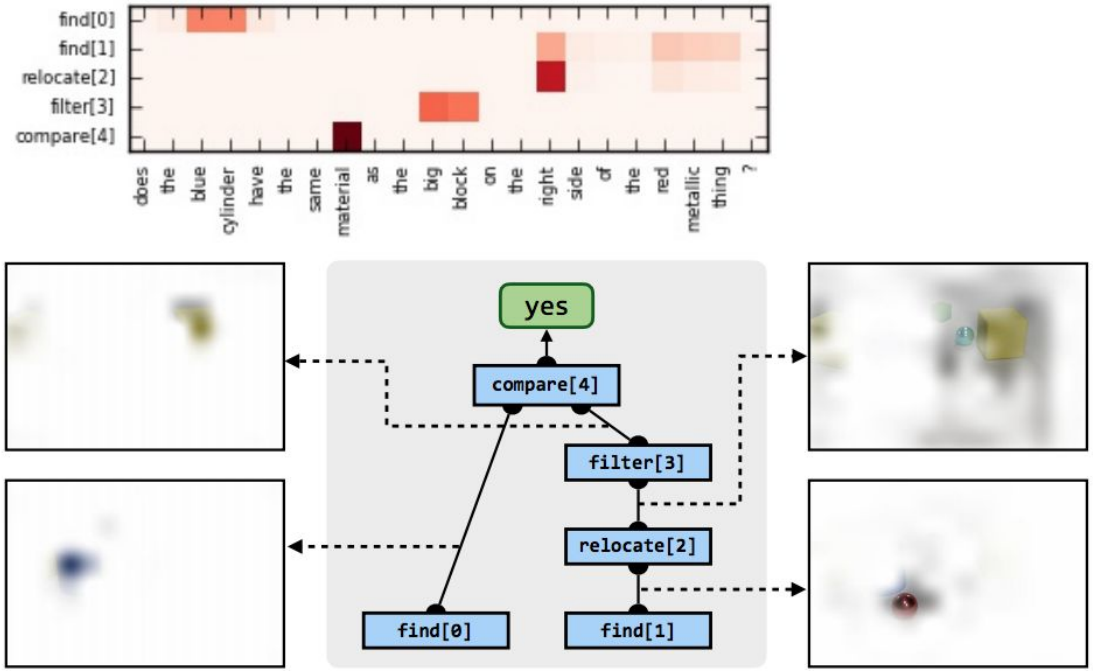
CLEVR Results

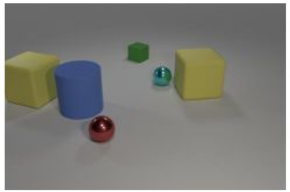
Method	shape	Compare Attribute			
		size	color	material	shape
CNN+BoW [26]	47	52	52	51	52
CNN+LSTM [4]	48	54	54	51	53
CNN+LSTM+MCB [9]	48	51	52	50	51
CNN+LSTM+SA [25]	85	52	55	51	51
NMN (expert layout) [3]	80.2	80.7	74.4	77.6	79.3
ours - policy search from scratch	84.1	50.1	53.9	48.6	51.1
ours - cloning expert	88.3	89.4	52.5	85.4	86.7
ours - policy search after cloning	**90.6**	**92.6**	**82.8**	**89.6**	**90.0**

CLEVR Examples (1 of 2)



CLEVR Examples (2 of 2)





Does the blue cylinder have the same material as the big block on the right side of the red metallic thing?

Testing on VQA

- Same input preparation as for CLEVR
- Expert layout policy same as in NMN paper (using external parser)

VQA Results

Method	Visual feature	Accuracy
NMN [3]	LRCN VGG-16	57.3
D-NMN [2]	LRCN VGG-16	57.9
MCB [9]	ResNet-152	64.7
ours - cloning expert	LRCN VGG-16	61.9
ours - cloning expert	ResNet-152	64.2
ours - policy search after cloning	ResNet-152	**64.9**

MCB

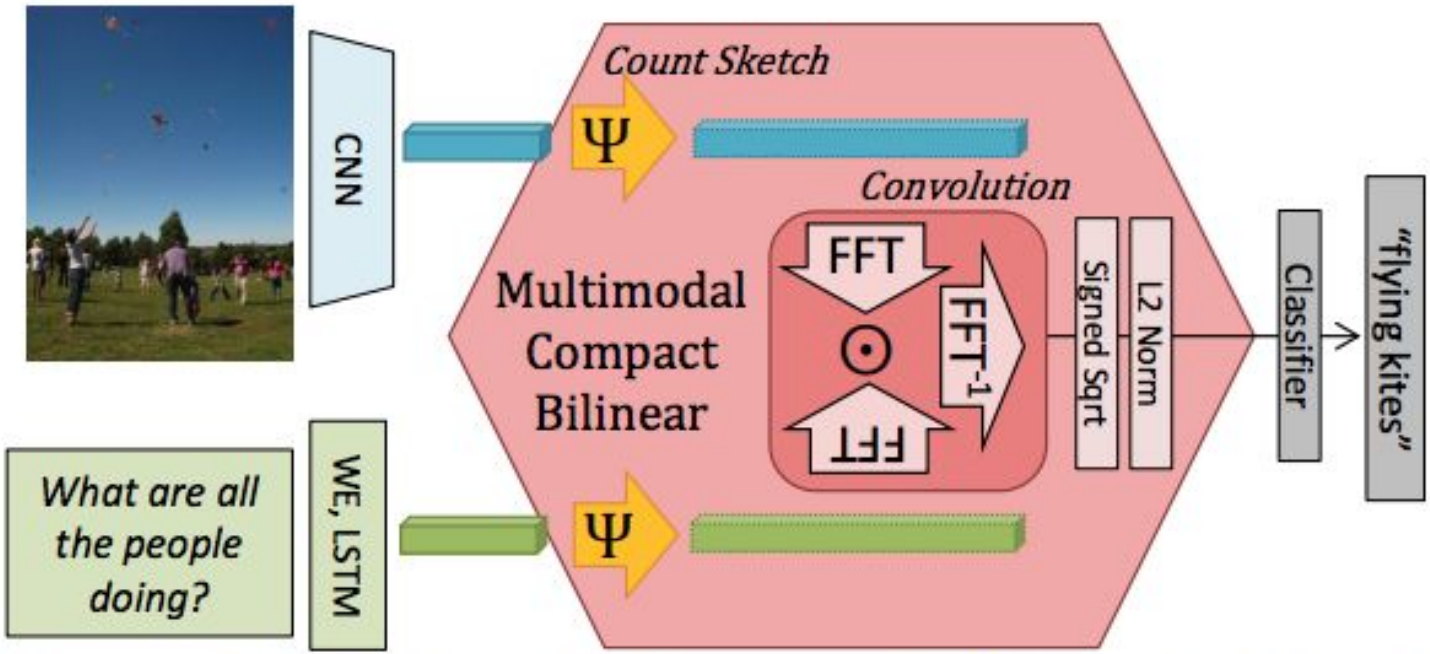
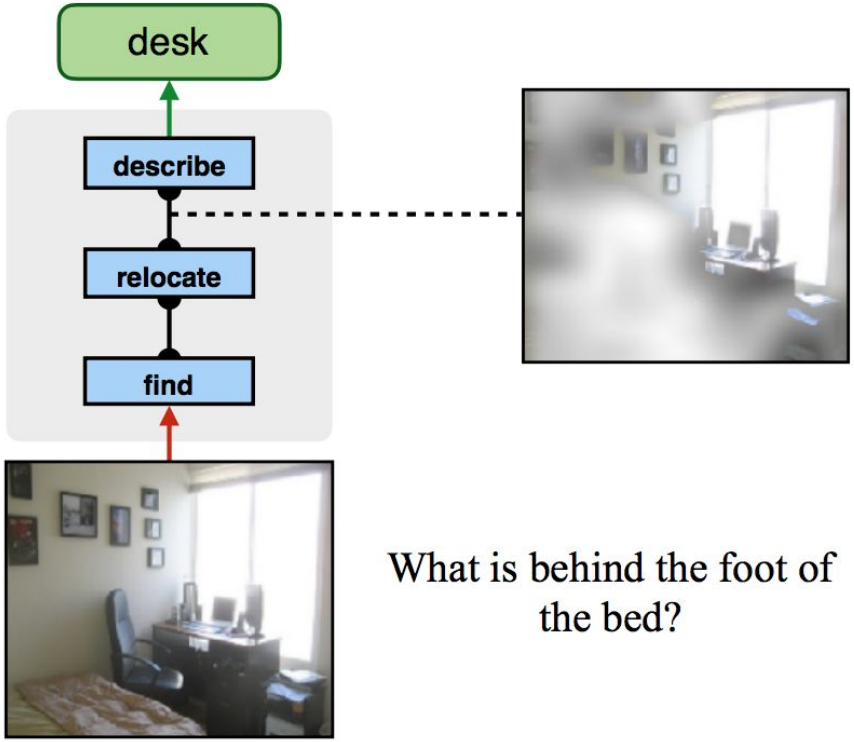


Figure 1: Multimodal Compact Bilinear Pooling for visual question answering.

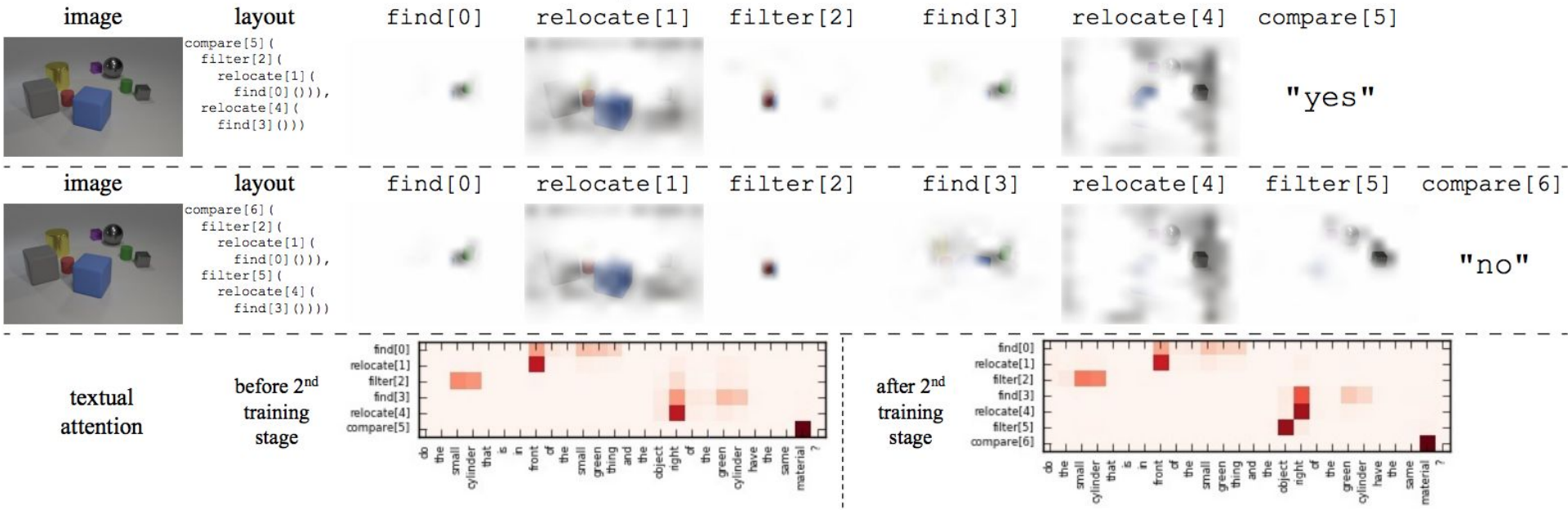
VQA Examples (1 of 2)



VQA Examples (2 of 2)

question: *do the small cylinder that is in front of the small green thing and the object right of the green cylinder have the same material?*

ground-truth answer: *no*



Summary

Neural Module Networks + Learning to Reason

1. **Neural Module Networks**
   a. Compositional reasoning
   b. Module toolbox
   c. Question parsing for layout prediction
2. **Learning to Reason**
   a. End-to-end module networks
   b. Layout policy
   c. Baseline for CLEVR

Extensions

Neural Module Networks + Learning to Reason

1. **Compositional Reasoning for Other Vision Tasks**
2. **More Versions of Compositional VQA**
    a. “Inferring and executing programs for visual reasoning” (Johnson et al., 2017) https://arxiv.org/pdf/1705.03633.pdf