Inferring and Executing Programs for Visual Reasoning

Justin Johnson, Bharath Hariharan, Laurens van der Maaten,
Judy Hoffman, Li Fei-Fei, C.Lawrence Zitnick, Ross Girshick

Presenter: Siliang Lu
9/26/2017

What is visual reasoning?

Is the person with the blue hat touching the bike in the back?

- In order to deal with complex visual question answering, it might be necessary to explicitly incorporate compositional reasoning in the model.
- I.e. Without having seen ”a person touching a bike”, the model should be able to understand the phrase by putting together its understanding of “**person**”, “**bike**” and “**touching**”.
- Different from visual recognition where models learn direct input-output mappings to learn dataset biases

What is visual reasoning?

Is the person with the blue hat touching the bike in the back?

- **Inputs:**
  An image *x* and a visual question *q* about the image
- **Intermediate outputs:**
  A predicted program *z* = $\pi(q)$ representing the reasoning steps required to answer the question and an execution engine $\phi(x, z)$ executing the program on the image to predict an answer
- **Output:**
  An answer *a* ∈ A to the question from a fixed set *A* of possible answers

Program generator *z* and execution engine ϕ

Innovations compared with state-of-arts

- Module network: a syntactic parse of a question to determine the architecture of the network

  Existing research: hand-designed off-the-shelf syntactic parser
  Current research: a learnt program generator that can adapt to the task at hand

- Semantic parser

  Existing research: the semantics of the program and the execution engine are fixed and known a priori
  Current research: learn both the program generator and the execution engine

- Program-induction methods

  Existing research: the interpretation of neural program considers only simple algorithms and program-induction assumes knowledge of the low-level operations
  Current research: the program generator consider inputs comprising an image and an associated question while assume minimal prior knowledge

What is program generator and execution engine?

Programs: focused on learning semantics for a fixed syntax

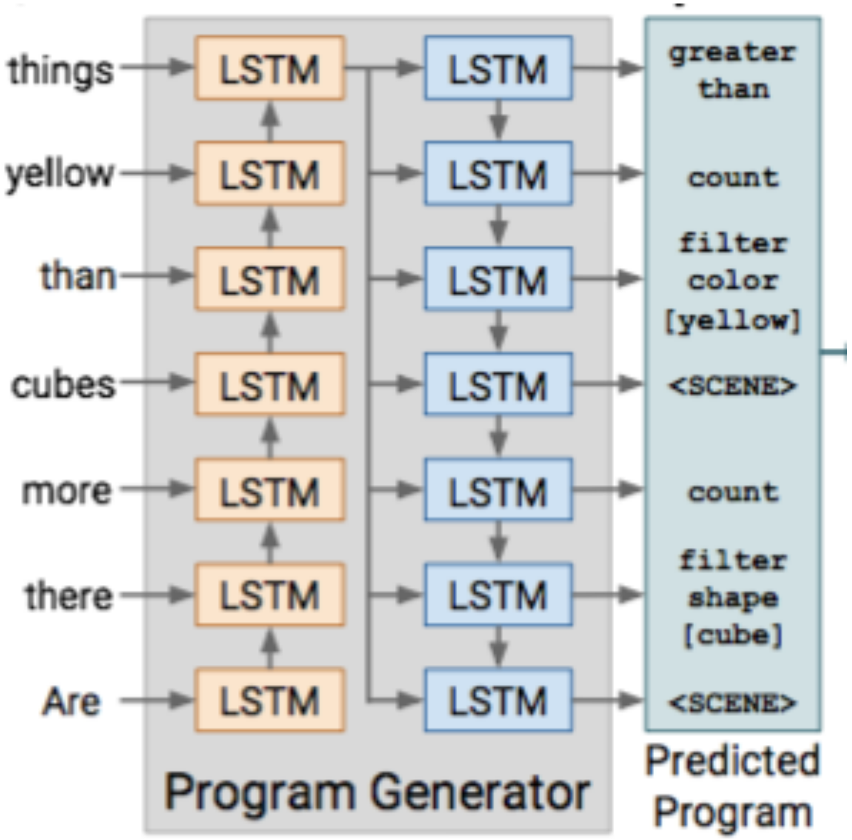
- Pre-specifying a set *F* of functions *f*, each of which has a fixed arity $n_f = \{1,2\}$
- Including in the vocabulary a special constant *Scene* representing the visual features of the image
- A valid program *z* is represented as syntax tress where each node contains a function *f*

Execution engine: creating a neural network mapping to each function *f*

- The program z is used to assemble a question-specific neural network composed from a set of modules
- Generic architecture for all unary module, binary module and Scene module

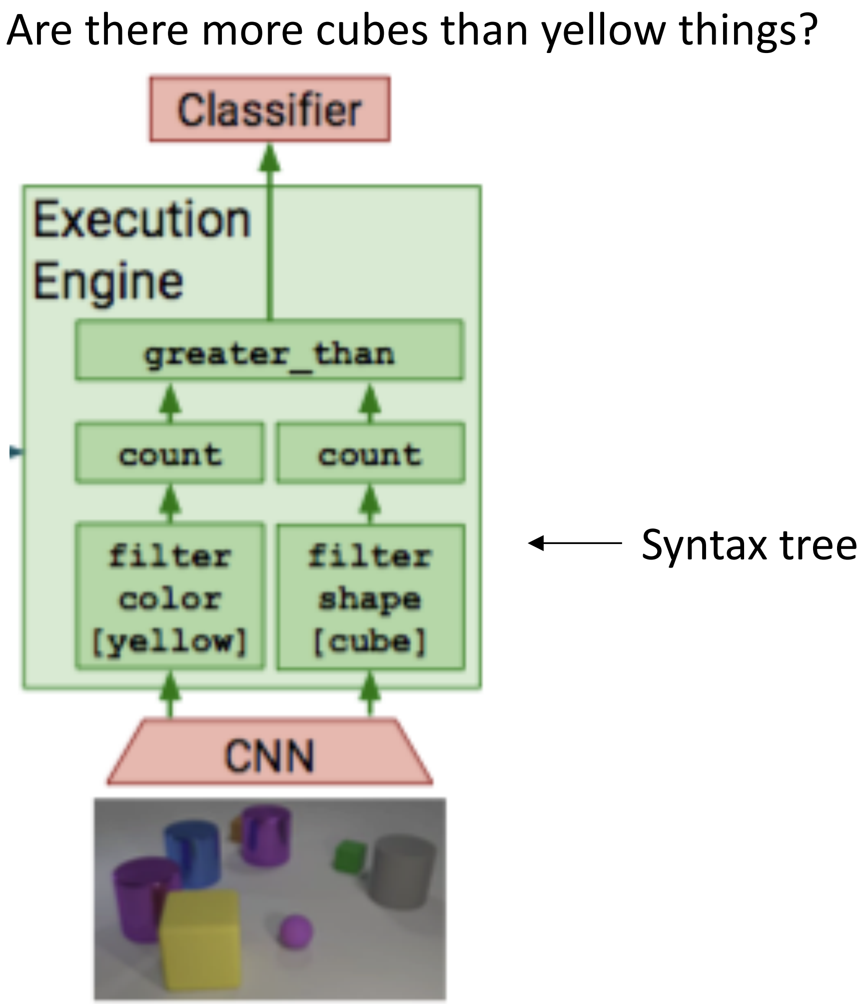
Program generator

Are there more cubes than yellow things?



- LSTM sequence-to-sequence model
- The resulting sequence of functions is converted to a syntax tree with prefix traversal
- If the sequence is too short, we pad the sequence with Scene constants
- If the sequence is too long, unused functions are discarded

Execution engine



- Scene module takes visual features as input with CNN

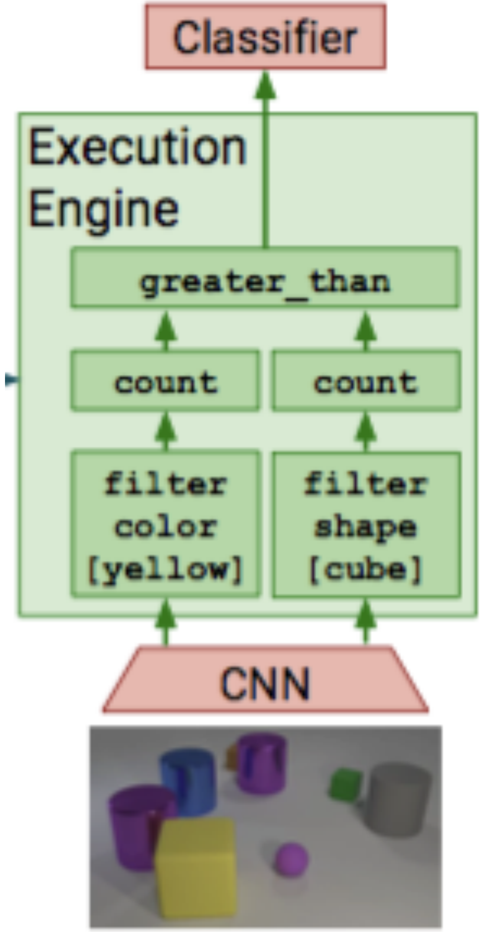
Layer	Output size
Input image	$3 \times 224 \times 224$
ResNet-101 [14] conv4_6	$1024 \times 14 \times 14$
Conv(3×3, $1024 \to 128$)	$128 \times 14 \times 14$
ReLU	$128 \times 14 \times 14$
Conv(3×3, $128 \to 128$)	$128 \times 14 \times 14$
ReLU	$128 \times 14 \times 14$

- The final feature map is flattened and passed into a multilayer perception classifier

Layer	Output size
Final module output	$128 \times 14 \times 14$
Conv(1×1, $128 \to 512$)	$512 \times 14 \times 14$
ReLU	$512 \times 14 \times 14$
MaxPool(2×2, stride 2)	$512 \times 7 \times 7$
FullyConnected($512 \cdot 7 \cdot 7 \to 1024$)	1024
ReLU	1024
FullyConnected($1024 \to \lvert\mathcal{A}\rvert$)	$\lvert\mathcal{A}\rvert$

Execution engine

Are there more cubes than yellow things?



Syntax tree

- Unary module

Index	Layer	Output size
(1)	Previous module output	$128 \times 14 \times 14$
(2)	Conv(3×3, $128 \to 128$)	$128 \times 14 \times 14$
(3)	ReLU	$128 \times 14 \times 14$
(4)	Conv(3×3, $128 \to 128$)	$128 \times 14 \times 14$
(5)	Residual: Add (1) and (4)	$128 \times 14 \times 14$
(6)	ReLU	$128 \times 14 \times 14$

- Binary module

Index	Layer	Output size
(1)	Previous module output	$128 \times 14 \times 14$
(2)	Previous module output	$128 \times 14 \times 14$
(3)	Concatenate (1) and (2)	$256 \times 14 \times 14$
(4)	Conv(1×1, $256 \to 128$)	$128 \times 14 \times 14$
(5)	ReLU	$128 \times 14 \times 14$
(6)	Conv(3×3, $128 \to 128$)	$128 \times 14 \times 14$
(7)	ReLU	$128 \times 14 \times 14$
(8)	Conv(3×3, $128 \to 128$)	$128 \times 14 \times 14$
(9)	Residual: Add (5) and (8)	$128 \times 14 \times 14$
(10)	ReLU	$128 \times 14 \times 14$

Execution engine

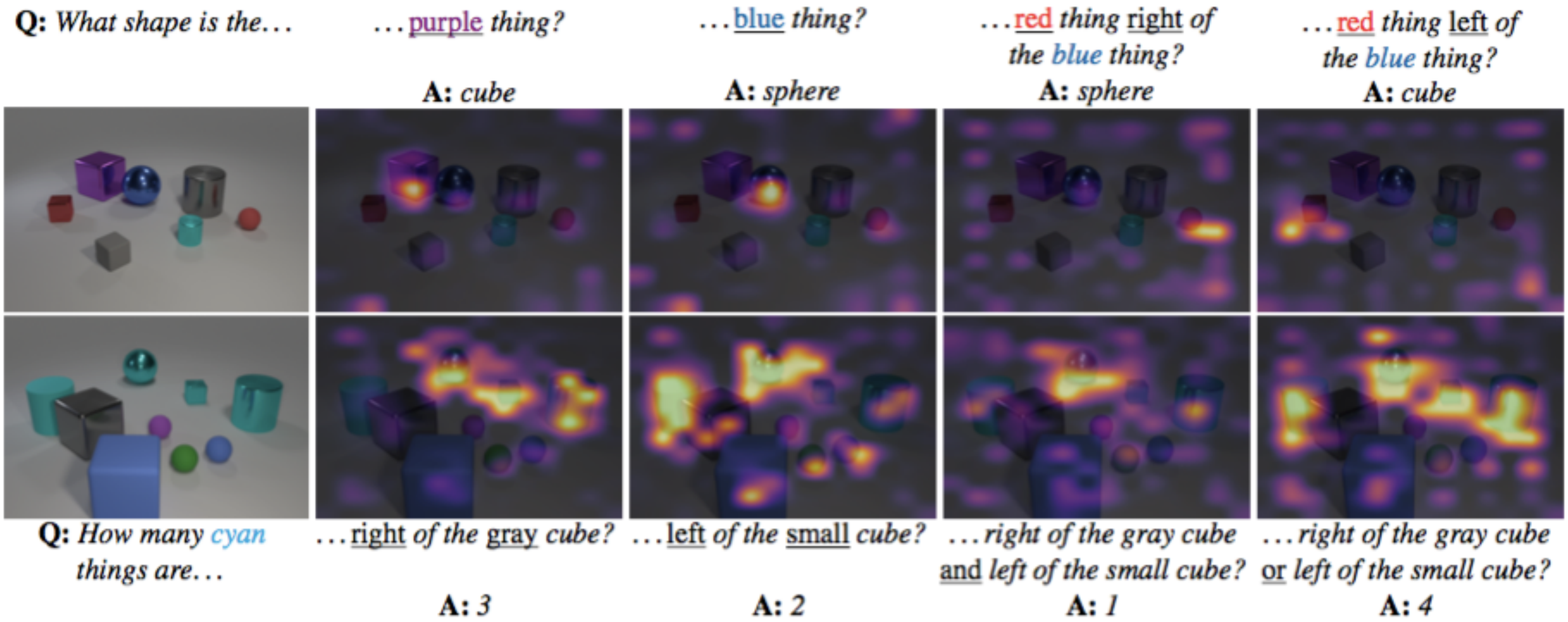


Figure 3. Visualizations of the norm of the gradient of the sum of the predicted answer scores with respect to the final feature map. From left to right, each question adds a module to the program; the new module is *underlined* in the question. The visualizations illustrate which objects the model attends to when performing the reasoning steps for question answering. Images are from the validation set.

Training

Separate training with ground-truth programs

- Given VQA dataset containing (x,q,z,a) tuples with ground truth z
- Use pairs *(q,z)* of questions and corresponding programs to train the program generator
- Use triplets *(x,z,a)* of the image, program, and answer to train the execution engine with backpropagation to compute the gradients

Joint training without ground-truth programs

- Use REINFORCE to estimate gradients on the outputs of the program generator.
- **The reward for each of its outputs is the negative zero-one loss of the execution engine, with a moving-average baseline.**

Training

Semi-supervised learning

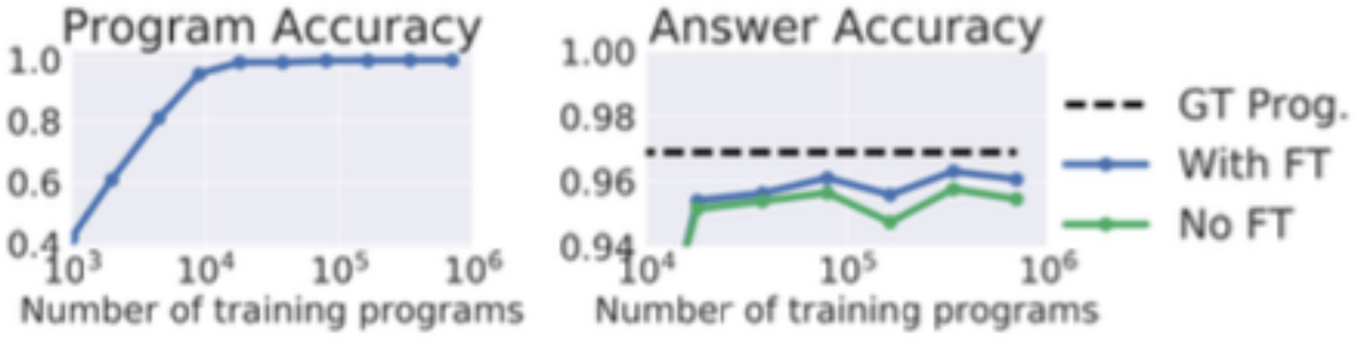
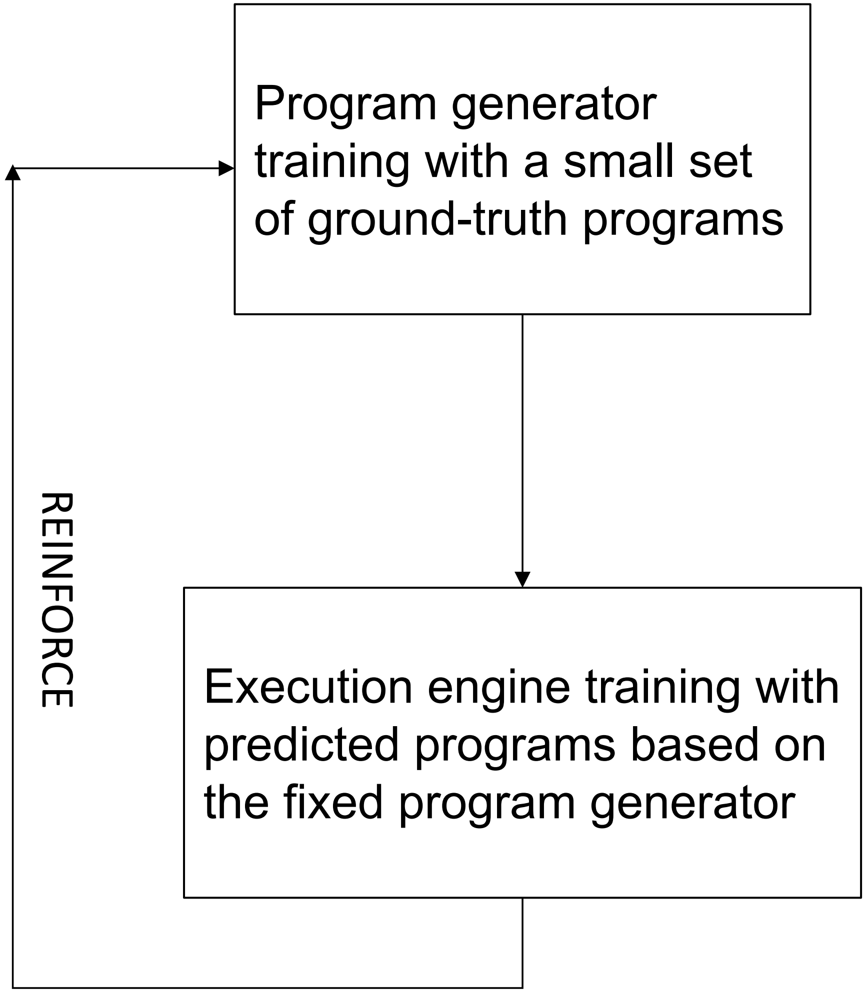


Figure 4. Accuracy of predicted programs (left) and answers (right) as we vary the number of ground-truth programs. Blue and green give accuracy before and after joint finetuning; the dashed line shows accuracy of our strongly-supervised model.



Training

Method	Exist	Count	Compare Integer			Query				Compare				Overall
			Equal	Less	More	Size	Color	Mat.	Shape	Size	Color	Mat.	Shape	
Q-type mode	50.2	34.6	51.4	51.6	50.5	50.1	13.4	50.8	33.5	50.3	52.5	50.2	51.8	42.1
LSTM	61.8	42.5	63.0	73.2	71.7	49.9	12.2	50.8	33.2	50.5	52.5	49.7	51.8	47.0
CNN+LSTM	68.2	47.8	60.8	74.3	72.5	62.5	22.4	59.9	50.9	56.5	53.0	53.8	55.5	54.3
CNN+LSTM+SA [46]	68.4	57.5	56.8	74.9	68.2	90.1	83.3	89.8	87.6	52.1	55.5	49.7	50.9	69.8
CNN+LSTM+SA+MLP	77.9	59.7	60.3	83.7	76.7	85.4	73.1	84.5	80.7	72.3	71.2	70.1	69.7	73.2
Human† [19]	96.6	86.7	79.0	87.0	91.0	97.0	95.0	94.0	94.0	94.0	98.0	96.0	96.0	92.6
Ours-strong (700K prog.)	**97.1**	**92.7**	**98.0**	**99.0**	**98.9**	**98.8**	**98.4**	**98.1**	**97.3**	**99.8**	**98.5**	**98.9**	**98.4**	**96.9**
Ours-semi (18K prog.)	95.3	90.1	93.9	97.1	97.6	98.1	97.1	97.7	96.6	99.0	97.6	98.0	97.3	95.4
Ours-semi (9K prog.)	89.7	79.7	85.2	76.1	77.9	94.8	93.3	93.1	89.2	97.8	94.5	96.6	95.1	88.6

Table 1. Question answering accuracy (higher is better) on the CLEVR dataset for baseline models, humans, and three variants of our model. The strongly supervised variant of our model uses all 700K ground-truth programs for training, whereas the semi-supervised variants use 9K and 18K ground-truth programs, respectively. †Human performance is measured on a 5.5K subset of CLEVR questions.

Experiments

Generalizing to new attribute combinations

Compositional Generalization Test (CoGenT)

This data was used in Section 4.7 of the paper to study the ability of models to recognize novel combinations of attributes at test-time. The data is generated in two different conditions:

Condition A

- Cubes are gray, **blue**, **brown**, or **yellow**
- Cylinders are **red**, **green**, **purple**, or **cyan**
- Spheres can have any color

Condition B

- Cubes are **red**, **green**, **purple**, or **cyan**
- Cylinders are gray, **blue**, **brown**, or **yellow**
- Spheres can have any color

Experiments

Generalizing to new attribute combinations

- Top 1st column :

Train on A and test on A

- Top 2nd column:

Train on A and test on B

- Top 3rd column:

Train A and finetune on B and test on A

- Top 4th column:

Train A and finetune on B and test on B

- Bottom Figure 1:

Finetune on B and test on B with overall questions

- Bottom Figure 2:

Finetune on B and test on B with color-query

- Bottom Figure3:

Finetune on B and test on B with shape-query

	Train A		Finetune B	
Method	A	B	A	B
LSTM	55.2	50.9	51.5	54.9
CNN+LSTM	63.7	57.0	58.3	61.1
CNN+LSTM+SA+MLP	80.3	68.7	75.7	75.8
Ours (18K prog.)	**96.6**	**73.7**	**76.1**	**92.7**

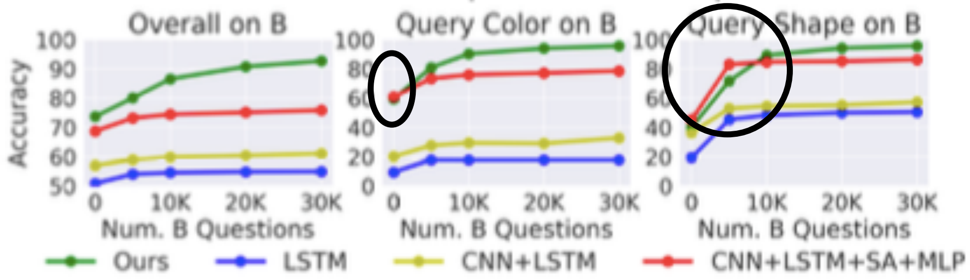


Figure 5. Question answering accuracy on the CLEVR-CoGenT dataset (higher is better). **Top:** We train models on Condition A, then test them on both Condition A and Condition B. We then finetune these models on Condition B using 3K images and 30K questions, and again test on both Conditions. Our model uses 18K programs during training on Condition A, and does not use any programs during finetuning on Condition B. **Bottom:** We investigate the effects of using different amounts of data when finetuning on Condition B. We show overall accuracy as well as accuracy on color-query and shape-query questions.

Experiments

Generalizing to new type of questions

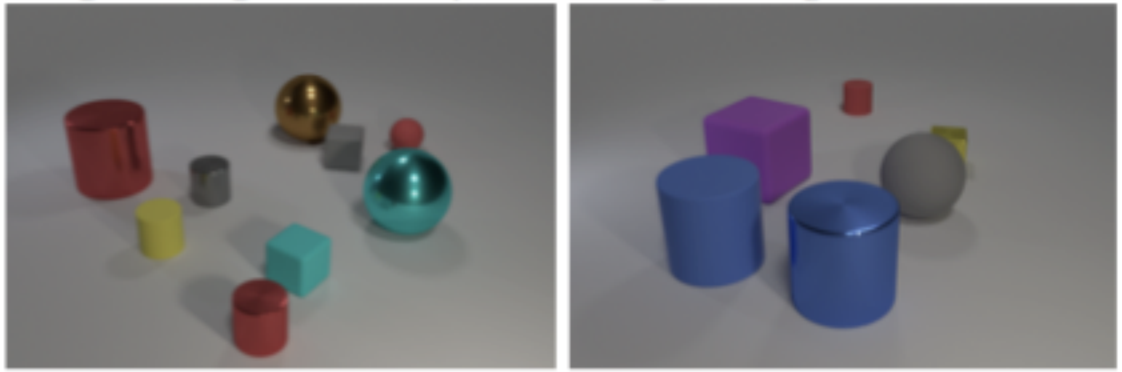
- Able to generalize to questions with **program structures** without observing associated ground-truth programs.

Method	Train Short: Short	Train Short: Long	Finetune Both: Short	Finetune Both: Long
LSTM	46.4	48.6	46.5	49.9
CNN+LSTM	54.0	52.8	54.3	54.2
CNN+LSTM+SA+MLP	74.2	**64.3**	74.2	67.8
Ours (25K prog.)	**95.9**	55.3	**95.6**	**77.8**

Table 2. Question answering accuracy on short and long CLEVR questions. **Left columns**: Models trained only on short questions; our model uses 25K ground-truth short programs. **Right columns**: Models trained on both short and long questions. Our model is trained on short questions then finetuned on the entire dataset; no ground-truth programs are used during finetuning.

Ground-truth question: *Is the number of matte blocks in front of the small yellow cylinder greater than the number of red rubber spheres to the left of the large red shiny cylinder?*
Program length: 20 **A:** *yes* ✓

Ground-truth question: *How many objects are big rubber objects that are in front of the big gray thing or large rubber things that are in front of the large rubber sphere?*
Program length: 16 **A:** *1* ✓

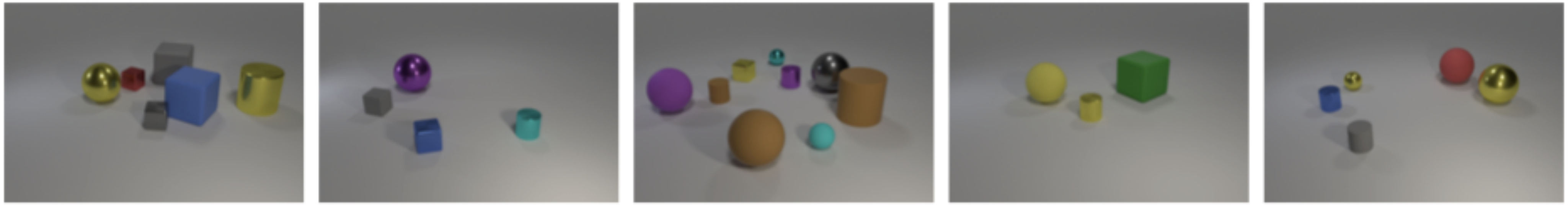


Predicted program (translated): *Is the number of matte blocks in front of the small yellow cylinder greater than the number of large red shiny cylinders?*
Program length: 15 **A:** *no* ✗

Predicted program (translated): *How many objects are big rubber objects in front of the big gray thing or large rubber spheres?*
Program length: 12 **A:** *2* ✗

Experiments

Human-composed questions



Q: *Is there a blue box in the items?* **A:** *yes*

Predicted Program:
exist
filter_shape[cube]
filter_color[blue]
scene

Predicted Answer:
✓ *yes*

Q: *What shape object is farthest right?* **A:** *cylinder*

Predicted Program:
query_shape
unique
relate[right]
unique
filter_shape[cylinder]
filter_color[blue]
scene

Predicted Answer:
✓ *cylinder*

Q: *Are all the balls small?* **A:** *no*

Predicted Program:
equal_size
query_size
unique
filter_shape[sphere]
scene
query_size
unique
filter_shape[sphere]
filter_size[small]
scene

Predicted Answer:
✓ *no*

Q: *Is the green block to the right of the yellow sphere?* **A:** *yes*

Predicted Program:
exist
filter_shape[cube]
filter_color[green]
relate[right]
unique
filter_shape[sphere]
filter_color[yellow]
scene

Predicted Answer:
✓*yes*

Q: *Two items share a color, a material, and a shape; what is the size of the rightmost of those items?* **A:** *large*

Predicted Program:
count
filter_shape[cube]
same_material
unique
filter_shape[cylinder]
scene

Predicted Answer:
✗ *0*

Figure 7. Examples of questions from the CLEVR-Humans dataset, along with predicted programs and answers from our model. Question words that do not appear in CLEVR questions are underlined. Some predicted programs exactly match the semantics of the question (green); some programs closely match the question semantics (yellow), and some programs appear unrelated to the question (red).

Future work

- How to add new modules by automatically identifying and learning without supervision program?

  i.e. “What color is the object with a unique shape?”
  solution: a Turing-complete set of modules

- Control-flow operators could be incorporated into the framework

- Learning programs with limited supervision

Thanks!