Visual Reasoning

Mohan Bhambhani

Topics in Deep Learning, 2017

Outline

1. Introduction
2. Datasets
3. Models

- Too much bias in VQA datasets.

- Too much bias in VQA datasets.
- Maybe What covers the ground? is answerable not because it understands the scene but because biased datasets often ask questions about the ground when it is snow-covered.

- Too much bias in VQA datasets.
- Maybe What covers the ground? is answerable not because it understands the scene but because biased datasets often ask questions about the ground when it is snow-covered.
- **Solution:** Focus on the ultimate goal of visual reasoning.

Introduction

Visual reasoning vs VQA

- Too much bias in VQA datasets.
- Maybe What covers the ground? is answerable not because it understands the scene but because biased datasets often ask questions about the ground when it is snow-covered.
- **Solution:** Focus on the ultimate goal of visual reasoning.
- **Goal of Visual systems:** To completely understand information contained in the images.

Introduction

Visual reasoning vs VQA

- Too much bias in VQA datasets.
- Maybe What covers the ground? is answerable not because it understands the scene but because biased datasets often ask questions about the ground when it is snow-covered.
- **Solution:** Focus on the ultimate goal of visual reasoning.
- **Goal of Visual systems:** To completely understand information contained in the images.

Image understanding → Visual reasoning → VQA

Outline

CLEVR

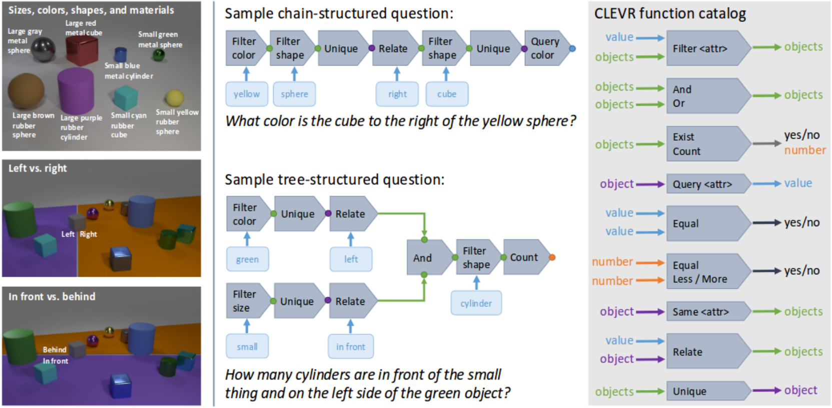
- To avoid bias added by humans, questions must be generated automatically. But, at the same time they must be good.

CLEVR

- To avoid bias added by humans, questions must be generated automatically. But, at the same time they must be good.
- Getting this done is easier in synthetic images.

CLEVR

- To avoid bias added by humans, questions must be generated automatically. But, at the same time they must be good.
- Getting this done is easier in synthetic images.
- Interface:



CLEVR

Images:

- Synthetically generated

CLEVR

Images:

- Synthetically generated
- 3 object shapes (cube, sphere, and cylinder)

CLEVR

Images:

- Synthetically generated
- 3 object shapes (cube, sphere, and cylinder)
- 2 sizes (small and large)

CLEVR

Images:

- Synthetically generated
- 3 object shapes (cube, sphere, and cylinder)
- 2 sizes (small and large)
- 2 materials (shiny ‘metal’ and matte ‘rubber’)

CLEVR

Images:

- Synthetically generated
- 3 object shapes (cube, sphere, and cylinder)
- 2 sizes (small and large)
- 2 materials (shiny 'metal' and matte 'rubber')
- 8 colors

CLEVR

Images:

- Synthetically generated
- 3 object shapes (cube, sphere, and cylinder)
- 2 sizes (small and large)
- 2 materials (shiny ‘metal’ and matte ‘rubber’)
- 8 colors
- 4 relationships between objects: (‘left’,‘right’, ‘behind’, and ‘in front’)

CLEVR

Images:

- Synthetically generated
- 3 object shapes (cube, sphere, and cylinder)
- 2 sizes (small and large)
- 2 materials (shiny 'metal' and matte 'rubber')
- 8 colors
- 4 relationships between objects: ('left','right', 'behind', and 'in front')
- Scene graph annotated

CLEVR

Images:

- Synthetically generated
- 3 object shapes (cube, sphere, and cylinder)
- 2 sizes (small and large)
- 2 materials (shiny 'metal' and matte 'rubber')
- 8 colors
- 4 relationships between objects: ('left','right', 'behind', and 'in front')
- Scene graph annotated
- **To generate image:** Randomly sample a scene graph. Then generate image using Blender (a tool to generate 3d images from scene graph).

CLEVR

Questions:

CLEVR

Questions:

- Define question families.

CLEVR

Questions:

- Define question families.
- Example: How many red things are there? can be formed from How many $< C > < M >$ things are there?, binding the parameters $< C >$ and $< M >$ (with types color and material) with values red and nil.

CLEVR

Questions:

- Define question families.
- Example: How many red things are there? can be formed from How many $< C > < M >$ things are there?, binding the parameters $< C >$ and $< M >$ (with types color and material) with values red and nil.
- Each question in CLEVR is associated with a functional program that can be executed on an image's scene graph.

CLEVR

Questions:

- Define question families.
- Example: How many red things are there? can be formed from How many $< C > < M >$ things are there?, binding the parameters $< C >$ and $< M >$ (with types color and material) with values red and nil.
- Each question in CLEVR is associated with a functional program that can be executed on an image's scene graph.
- The functional program $count(filterColor(red, scene()))$ for this question can be formed by instantiating the associated program template $count(filterColor(< C >, filterMaterial(< M >, scene())))$.

CLEVR

Questions:

- Define question families.
- Example: How many red things are there? can be formed from How many $< C > < M >$ things are there?, binding the parameters $< C >$ and $< M >$ (with types color and material) with values red and nil.
- Each question in CLEVR is associated with a functional program that can be executed on an image's scene graph.
- The functional program $count(filterColor(red, scene()))$ for this question can be formed by instantiating the associated program template $count(filterColor(< C >, filterMaterial(< M >, scene())))$.
- CLEVR contains a total of 90 question families, each with a single program template and an average of four text templates.

CLEVR

Questions:

- Define question families.
- Example: How many red things are there? can be formed from How many $< C > < M >$ things are there?, binding the parameters $< C >$ and $< M >$ (with types color and material) with values red and nil.
- Each question in CLEVR is associated with a functional program that can be executed on an image's scene graph.
- The functional program $count(filterColor(red, scene()))$ for this question can be formed by instantiating the associated program template $count(filterColor(< C >, filterMaterial(< M >, scene())))$.
- CLEVR contains a total of 90 question families, each with a single program template and an average of four text templates.
- So to generate question: choose a question family, select values for each of its template parameters, execute the resulting program on the image's scene graph to find the answer.

CLEVR

Quality of questions :

- Still many ill-posed questions can be formed.

CLEVR

Quality of questions :

- Still many ill-posed questions can be formed.
- The question What color is the cube to the right of the sphere? would be ill-posed if there were many cubes right of the sphere, or degenerate if there were only one cube in the scene.

CLEVR

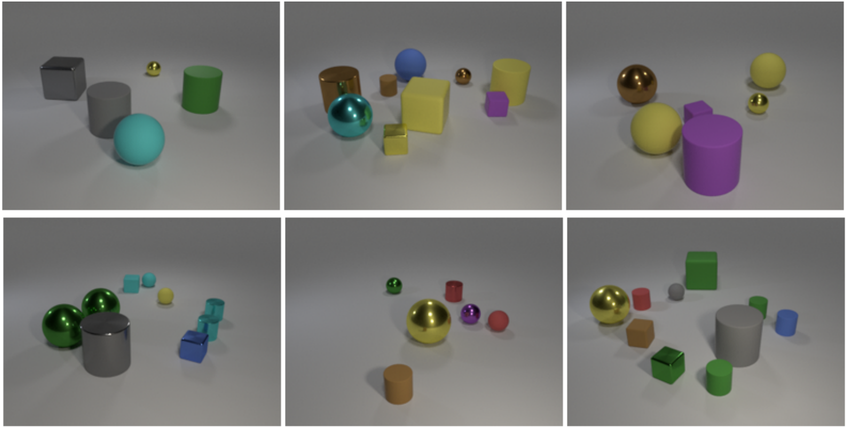
Quality of questions :

- Still many ill-posed questions can be formed.
- The question What color is the cube to the right of the sphere? would be ill-posed if there were many cubes right of the sphere, or degenerate if there were only one cube in the scene.
- Employ a depth-first search. At each step of assigning values to the vaiables, use ground-truth scene information to prune nodes which are guaranteed to produce undesirable questions.

Quality of questions :

- Still many ill-posed questions can be formed.
- The question What color is the cube to the right of the sphere? would be ill-posed if there were many cubes right of the sphere, or degenerate if there were only one cube in the scene.
- Employ a depth-first search. At each step of assigning values to the vaiables, use ground-truth scene information to prune nodes which are guaranteed to produce undesirable questions.
- Maintain uniform question distribution across families.

CLEVR

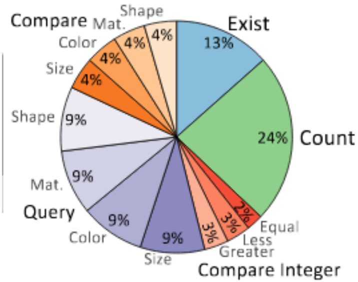


CLEVR

```
"params": [
  {"type": "Size", "name": "<Z>"},
  {"type": "Color", "name": "<C>"},
  {"type": "Material", "name": "<M>"},
  {"type": "Shape", "name": "<S>"},
  {"type": "Relation", "name": "<R>"},
  {"type": "Size", "name": "<Z2>"},
  {"type": "Color", "name": "<C2>"},
  {"type": "Material", "name": "<M2>"},
  {"type": "Shape", "name": "<S2>"}
],
"text": [
  "What size is the <Z2> <C2> <M2> <S2> [that is] <R> the <Z> <C> <M> <S>?",
  "What is the size of the <Z2> <C2> <M2> <S2> [that is] <R> the <Z> <C> <M> <S>?",
  "How big is the <Z2> <C2> <M2> <S2> [that is] <R> the <Z> <C> <M> <S>?",
  "There is a <Z2> <C2> <M2> <S2> [that is] <R> the <Z> <C> <M> <S>; what size is it?",
  "There is a <Z2> <C2> <M2> <S2> [that is] <R> the <Z> <C> <M> <S>; how big is it?",
  "There is a <Z2> <C2> <M2> <S2> [that is] <R> the <Z> <C> <M> <S>; what is its size?"
],
```

CLEVR

Split	Images	Questions	Unique questions	Overlap with train
Total	100,000	999,968	853,554	-
Train	70,000	699,989	608,607	-
Val	15,000	149,991	140,448	17,338
Test	15,000	149,988	140,352	17,335



Outline

NMNs

- **Inspiration:** Different architectures are needed for different tasks - CNNs for object detection, RNNs for counting.

NMNs

- **Inspiration:** Different architectures are needed for different tasks - CNNs for object detection, RNNs for counting.
- Decompose a question into its linguistic substructures and train a neural network module for each substructure.

NMNs

- **Inspiration:** Different architectures are needed for different tasks - CNNs for object detection, RNNs for counting.
- Decompose a question into its linguistic substructures and train a neural network module for each substructure.
- Jointly train the modules and dynamically compose them into deep networks which can learn to answer the question.

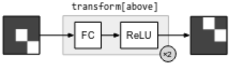
NMNs

- **Inspiration:** Different architectures are needed for different tasks - CNNs for object detection, RNNs for counting.
- Decompose a question into its linguistic substructures and train a neural network module for each substructure.
- Jointly train the modules and dynamically compose them into deep networks which can learn to answer the question.
- Start by analyzing the question and decide what logical units are needed to answer the question and what should be the relationship between them.

NMNs

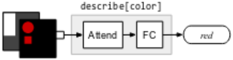
- Find: Finds objects of interest.

- Transform: Shift regions of attention.

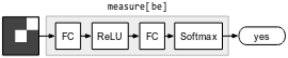


- Combine: Merge two attention maps into a single one.

- Describe: Map a pair of attention and input image to a distribution over the labels.



- Measure: Map attention to a distribution over the labels.



NMNs

- Model is specified by collection of modules $\{m\}$ and a network layout predictor P which maps from strings to networks.

NMNs

- Model is specified by collection of modules $\{m\}$ and a network layout predictor P which maps from strings to networks.
- Map question to the layout which specifies the set of modules and connections between them.

NMNs

- Model is specified by collection of modules $\{m\}$ and a network layout predictor P which maps from strings to networks.
- Map question to the layout which specifies the set of modules and connections between them.
- Assemble the final network using the layout.

NMNs

- Model is specified by collection of modules $\{m\}$ and a network layout predictor P which maps from strings to networks.
- Map question to the layout which specifies the set of modules and connections between them.
- Assemble the final network using the layout.
- Parse the input question to obtain set of dependencies and obtain a representation similar to combinatory logic.

NMNs

- Model is specified by collection of modules $\{m\}$ and a network layout predictor P which maps from strings to networks.
- Map question to the layout which specifies the set of modules and connections between them.
- Assemble the final network using the layout.
- Parse the input question to obtain set of dependencies and obtain a representation similar to combinatory logic.
- For example, what is standing in the field becomes what(stand), what color is the truck becomes color(truck), and is there a circle next to a square becomes is(circle, next-to(square)).

NMNs

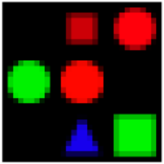
- Model is specified by collection of modules $\{m\}$ and a network layout predictor P which maps from strings to networks.
- Map question to the layout which specifies the set of modules and connections between them.
- Assemble the final network using the layout.
- Parse the input question to obtain set of dependencies and obtain a representation similar to combinatory logic.
- For example, what is standing in the field becomes what(stand), what color is the truck becomes color(truck), and is there a circle next to a square becomes is(circle, next-to(square)).
- The symbolic representation is mapped to a layout:
  - All leaves become find module.
  - All internal nodes become transform/combine module.
  - All root nodes become describe/measure module.

NMNs

- Final model combines the output from the NMN with predictions from a simple LSTM question encoder. This helps in modelling the syntactic and semantic regularities of the question.

NMNs

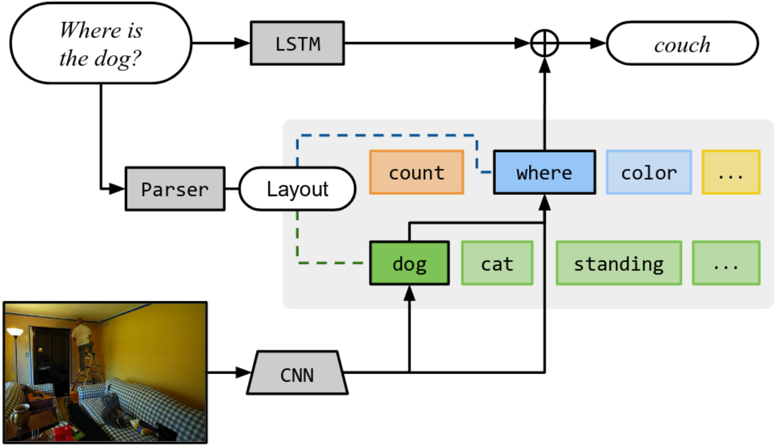
- Final model combines the output from the NMN with predictions from a simple LSTM question encoder. This helps in modelling the syntactic and semantic regularities of the question.

 *how many different lights in various different shapes and sizes?*	 *what is the color of the horse?*	 *what color is the vase?*	 *is the bus full of passengers?*	 *is there a red shape above a circle?*

`describe[count](`
`find[light])`
	`describe[color](`
	`find[horse])`
		`describe[color](`
		`find[vase])`
			`describe[is](`
			`combine[and](`
			`find[bus],`
			`find[full])`
				`measure[is](`
				`combine[and](`
				`find[red],`
				`transform[above](`
				`find[circle])))`

four (four)	brown (brown)	green (green)	yes (yes)	yes (yes)

NMNs



Where is the dog?
LSTM
couch
Parser
Layout
count
where
color
...
dog
cat
standing
...
CNN

Implementation details:

- Since some modules are updated more frequently than others, adaptive per weight learning rates are better.
- Training: Adadelta with standard parameter settings.
- Stanford parser has F1 score of 87%.

NMNs

Model	Overall	Count	Exist	Compare numbers	Query	Compare attribute
Human	92.6	86.7	96.6	86.5	95.0	96.0
LSTM	46.8	41.7	61.1	69.8	36.8	51.8
CNN+RNN	52.3	43.7	65.2	67.1	49.3	53.0
SA	68.5	52.2	71.1	73.5	85.3	52.3
NMN	72.1	79.3	52.5	71.4	78.9	78

N2NMNs

- NMNs are restricted to parsers for model configuration. A better model would be that learns it from the data.
- This paper learns to both parse the language into linguistic structures and compose them into appropriate layouts.
- Model has 2 parts. First, a set of co-attentive neural modules that provide parameterized functions for solving sub-tasks, and a layout policy to predict a question-specific layout from which a neural network is dynamically assembled.

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$
- **And:** To take and of 2 feature maps $a_{out} = minimum(a_1, a_2)$

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$
- **And:** To take and of 2 feature maps $a_{out} = minimum(a_1, a_2)$
- **Or:** To take or of 2 feature maps $a_{out} = maximum(a_1, a_2)$

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$
- **And:** To take and of 2 feature maps $a_{out} = minimum(a_1, a_2)$
- **Or:** To take or of 2 feature maps $a_{out} = maximum(a_1, a_2)$
- **Filter:** Added this to simplify layout. $a_{out} = and(a, find[x_{vis}, x_{txt}]())$

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$
- **And:** To take and of 2 feature maps $a_{out} = minimum(a_1, a_2)$
- **Or:** To take or of 2 feature maps $a_{out} = maximum(a_1, a_2)$
- **Filter:** Added this to simplify layout. $a_{out} = and(a, find[x_{vis}, x_{txt}]())$
- **Exist, Count:** For simple inference from attention map.
  $y = W^T vec(a)$

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$
- **And:** To take and of 2 feature maps $a_{out} = minimum(a_1, a_2)$
- **Or:** To take or of 2 feature maps $a_{out} = maximum(a_1, a_2)$
- **Filter:** Added this to simplify layout. $a_{out} = and(a, find[x_{vis}, x_{txt}]())$
- **Exist, Count:** For simple inference from attention map.
  $y = W^T vec(a)$
- **Describe:** For complex inference $y = W_1^T(W_2 sum(a \odot x_{vis}) \odot W_3 x_{txt})$

N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$
- **And:** To take and of 2 feature maps $a_{out} = minimum(a_1, a_2)$
- **Or:** To take or of 2 feature maps $a_{out} = maximum(a_1, a_2)$
- **Filter:** Added this to simplify layout. $a_{out} = and(a, find[x_{vis}, x_{txt}]())$
- **Exist, Count:** For simple inference from attention map.
  $y = W^T vec(a)$
- **Describe:** For complex inference $y = W_1^T (W_2 sum(a \odot x_{vis}) \odot W_3 x_{txt})$
- **Eq-count, More, Less:** For pairwise comparison over action maps.
  $y = W_1^T vec(a_1) + W_2^T vec(a_2)$

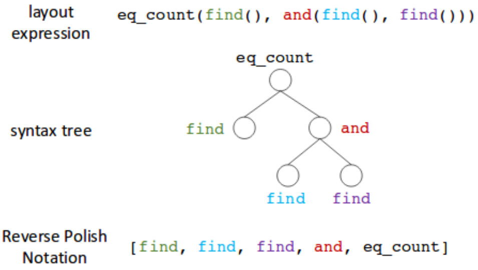
N2NMNs

- x_{vis} - image feature map, x_{txt} - text features
- **Find:** to localize some objects and output an attention map over the image. $a_{out} = conv2(conv1(x_{vis}) \odot Wx_{txt})$
- **Relocate:** to transform the input image attention map and output a new attention map
  $a_{out} = conv2(conv1(x_{vis}) \odot W_1 sum(a \odot x_{vis}) \odot W_2 x_{txt})$
- **And:** To take and of 2 feature maps $a_{out} = minimum(a_1, a_2)$
- **Or:** To take or of 2 feature maps $a_{out} = maximum(a_1, a_2)$
- **Filter:** Added this to simplify layout. $a_{out} = and(a, find[x_{vis}, x_{txt}]())$
- **Exist, Count:** For simple inference from attention map.
  $y = W^T vec(a)$
- **Describe:** For complex inference $y = W_1^T(W_2 sum(a \odot x_{vis}) \odot W_3 x_{txt})$
- **Eq-count, More, Less:** For pairwise comparison over action maps.
  $y = W_1^T vec(a_1) + W_2^T vec(a_2)$
- **Compare:** For complex pairwise comparison
  $y = W_1^T(W_2 sum(a_1 \odot x_{vis}) \odot W_3 sum(a_2 \odot x_{vis}) \odot W_4 x_{txt})$

N2NMNs

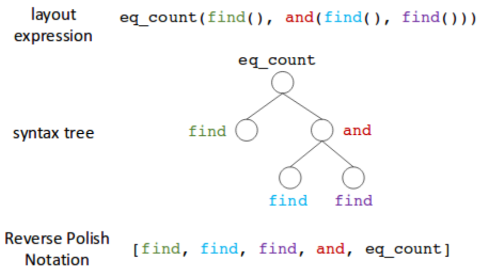
Layout:

- To predict the layout represent the layout in term of Reverse Polish notation and convert it to a sequence.



Layout:

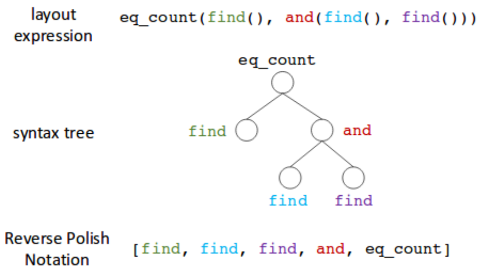
- To predict the layout represent the layout in term of Reverse Polish notation and convert it to a sequence.
- This can now be addressed using Encoder-decoder framework with both encoder and decoder being LSTMs.



N2NMNs

Layout:

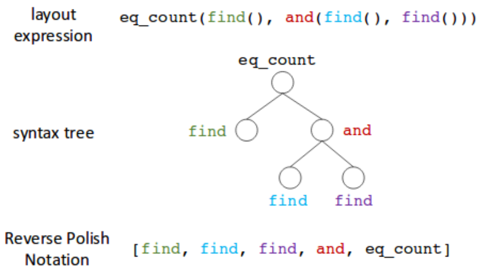
- To predict the layout represent the layout in term of Reverse Polish notation and convert it to a sequence.
- This can now be addressed using Encoder-decoder framework with both encoder and decoder being LSTMs.



- At each time step in the decoder LSTM, a soft attention map over the input sequence is predicted.

Layout:

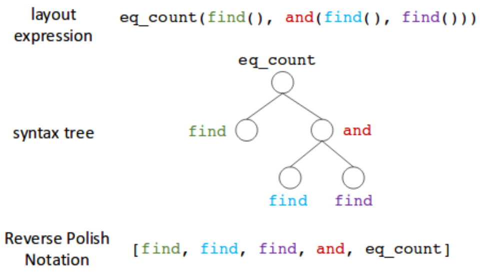
- To predict the layout represent the layout in term of Reverse Polish notation and convert it to a sequence.
- This can now be addressed using Encoder-decoder framework with both encoder and decoder being LSTMs.



- At each time step in the decoder LSTM, a soft attention map over the input sequence is predicted.
- In the end of each cell of decoder we have softmax layer to predict the module.

Layout:

- To predict the layout represent the layout in term of Reverse Polish notation and convert it to a sequence.
- This can now be addressed using Encoder-decoder framework with both encoder and decoder being LSTMs.



- At each time step in the decoder LSTM, a soft attention map over the input sequence is predicted.
- In the end of each cell of decoder we have softmax layer to predict the module.
- At each step the text features passed to the module are the attended features obtained from the LSTM

N2NMNs

Training:

- Training is done end to end.

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.
- Layout network: $p(l|q;\theta)$

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.
- Layout network: $p(l|q;\theta)$
- Loss: $L(\theta) = E_{l \sim p(l|q;\theta)}[\widetilde{L}(\theta, l; q, Im)]$

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.
- Layout network: $p(l|q;\theta)$
- Loss: $L(\theta) = E_{l \sim p(l|q;\theta)}[\widetilde{L}(\theta, l; q, Im)]$
- Gradient from PG:
  $\nabla_\theta L = E_{l \sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)]$

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.
- Layout network: $p(l|q;\theta)$
- Loss: $L(\theta) = E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l; q, Im)]$
- Gradient from PG:
  $\nabla_\theta L = E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)]$
- This is estimated using Monte-Carlo sampling. (M=1)
  $\nabla_\theta L \sim \frac{1}{M}\sum_{m=1}^{M}(E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)])$

N2NMNs

Training:

- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.
- Layout network: $p(l|q;\theta)$
- Loss: $L(\theta) = E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l; q, Im)]$
- Gradient from PG:
  $\nabla_\theta L = E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)]$
- This is estimated using Monte-Carlo sampling. (M=1)
  $\nabla_\theta L \sim \frac{1}{M}\sum_{m=1}^{M}(E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)])$
- To reduce variance of estimated gradient baseline is added.

N2NMNs

Training:

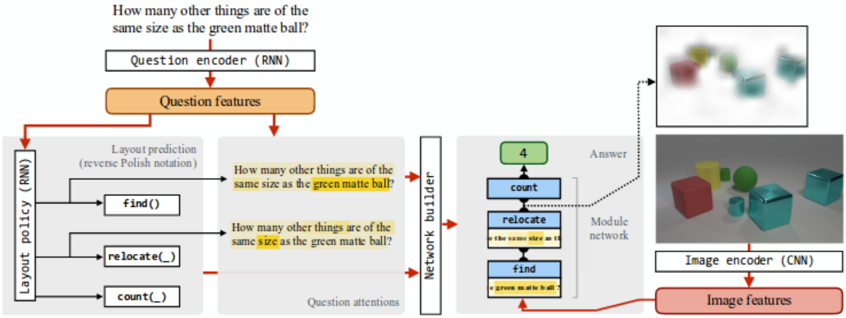
- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.
- Layout network: $p(l|q;\theta)$
- Loss: $L(\theta) = E_{l \sim p(l|q;\theta)}[\widetilde{L}(\theta, l; q, Im)]$
- Gradient from PG:
  $\nabla_\theta L = E_{l \sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)]$
- This is estimated using Monte-Carlo sampling. (M=1)
  $\nabla_\theta L \sim \frac{1}{M}\sum_{m=1}^{M}(E_{l \sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)])$
- To reduce variance of estimated gradient baseline is added.
- Layout is pretrained with KL-divergence between expert policy and layout policy loss.

N2NMNs

Training:

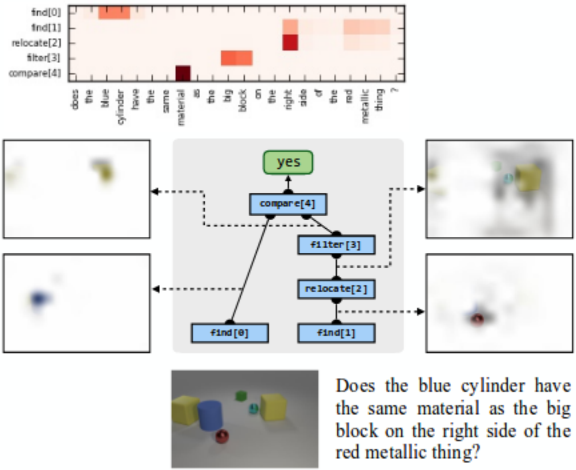
- Training is done end to end.
- Cross entropy loss function is used for all the modules.
- Loss function is not fully differentiable since the layout is discrete.
- Policy gradient is used for layout selection network.
- q- question, Im- image, l -layout.
- Layout network: $p(l|q;\theta)$
- Loss: $L(\theta) = E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l; q, Im)]$
- Gradient from PG:
  $\nabla_\theta L = E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)]$
- This is estimated using Monte-Carlo sampling. (M=1)
  $\nabla_\theta L \sim \frac{1}{M}\sum_{m=1}^{M}(E_{l\sim p(l|q;\theta)}[\widetilde{L}(\theta, l)\nabla_\theta \log p(l|q;\theta) + \nabla_\theta \widetilde{L}(\theta, l)])$
- To reduce variance of estimated gradient baseline is added.
- Layout is pretrained with KL-divergence between expert policy and layout policy loss.
- Adam optimiser was used for training.

Architecture overview:



N2NMNs

Example:



N2NMNs

Model	Overall	Count	Exist	Compare numbers	Query	Compare attribute
Human	92.6	86.7	96.6	86.5	95.0	96.0
LSTM	46.8	41.7	61.1	69.8	36.8	51.8
CNN+RNN	52.3	43.7	65.2	67.1	49.3	53.0
SA	68.5	52.2	71.1	73.5	85.3	52.3
NMN	72.1	79.3	52.5	71.4	78.9	78
N2NMN	83.7	85.7	68.5	83.7	90	88.7

Program generator

- Very similar architecture and same training procedure.

Program generator

- Very similar architecture and same training procedure.
- More modules

Program generator

- Very similar architecture and same training procedure.
- More modules
- Unary: Scene(to get action map), filter-color, filter-shape, filter-material, filter-size, unique, query-color, query-shape, query-material, query-color, query-size, exist, relate, count

Program generator

- Very similar architecture and same training procedure.
- More modules
- Unary: Scene(to get action map), filter-color, filter-shape, filter-material, filter-size, unique, query-color, query-shape, query-material, query-color, query-size, exist, relate, count
- Binary: intersect, union, equal-size, equal-color, equal-material, equal-shape, equal-integer, less-than, greaterthan.

Program generator

- Very similar architecture and same training procedure.
- More modules
- Unary: Scene(to get action map), filter-color, filter-shape, filter-material, filter-size, unique, query-color, query-shape, query-material, query-color, query-size, exist, relate, count
- Binary: intersect, union, equal-size, equal-color, equal-material, equal-shape, equal-integer, less-than, greaterthan.
- All the modules have residual layers.

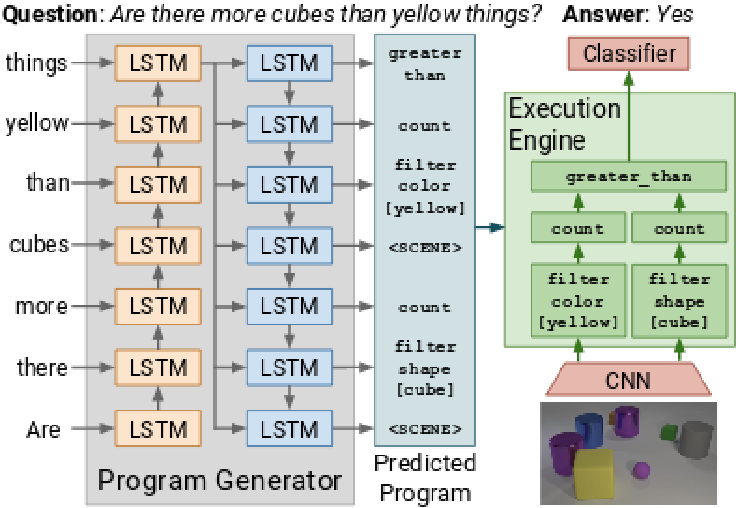
Program generator

- Very similar architecture and same training procedure.
- More modules
- Unary: Scene(to get action map), filter-color, filter-shape, filter-material, filter-size, unique, query-color, query-shape, query-material, query-color, query-size, exist, relate, count
- Binary: intersect, union, equal-size, equal-color, equal-material, equal-shape, equal-integer, less-than, greaterthan.
- All the modules have residual layers.
- At the end there is output module with fully connected layers.

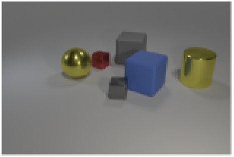
Program generator

- Very similar architecture and same training procedure.
- More modules
- Unary: Scene(to get action map), filter-color, filter-shape, filter-material, filter-size, unique, query-color, query-shape, query-material, query-color, query-size, exist, relate, count
- Binary: intersect, union, equal-size, equal-color, equal-material, equal-shape, equal-integer, less-than, greaterthan.
- All the modules have residual layers.
- At the end there is output module with fully connected layers.
- Program generation is top-down.

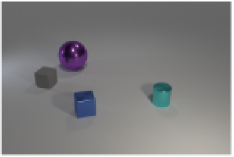
Program generator



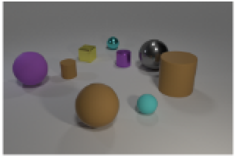
Program generator



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



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



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

Predicted Program:

```
exist
filter_shape[cube]
filter_color[blue]
scene
```

Predicted Answer:
✓ *yes*

Predicted Program:

```
query_shape
unique
relate[right]
unique
filter_shape[cylinder]
filter_color[blue]
scene
```

Predicted Answer:
✓ *cylinder*

Predicted Program:

```
equal_size
query_size
unique
filter_shape[sphere]
scene
query_size
unique
filter_shape[sphere]
filter_size[small]
scene
```

Predicted Answer:
✓ *no*

Program generator

Model	Overall	Count	Exist	Compare numbers	Query	Compare attribute
Human	92.6	86.7	96.6	86.5	95.0	96.0
LSTM	46.8	41.7	61.1	69.8	36.8	51.8
CNN+RNN	52.3	43.7	65.2	67.1	49.3	53.0
SA	68.5	52.2	71.1	73.5	85.3	52.3
NMN	72.1	79.3	52.5	71.4	78.9	78
N2NMN	83.7	85.7	68.5	83.7	90	88.7
PG	96.9	97.1	92.7	98.6	98.1	98.9

Relational networks

- Symbolic approaches to artificial intelligence are inherently relational.

Relational networks

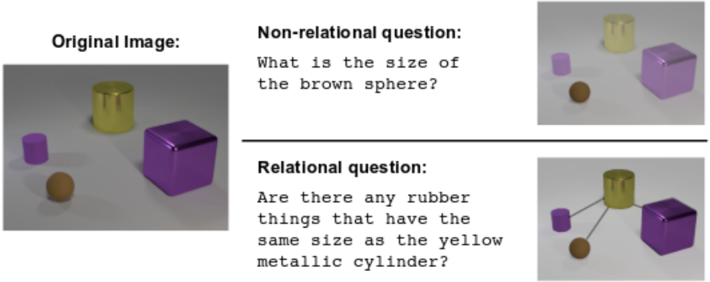
- Symbolic approaches to artificial intelligence are inherently relational.
- In deep learning, many approaches often struggle in data-poor problems where the underlying structure is characterized by sparse but complex relations.

Relational networks

- Symbolic approaches to artificial intelligence are inherently relational.
- In deep learning, many approaches often struggle in data-poor problems where the underlying structure is characterized by sparse but complex relations.
- This paper proposes relational network for relational reasoning with neural networks.

Relational networks

- Symbolic approaches to artificial intelligence are inherently relational.
- In deep learning, many approaches often struggle in data-poor problems where the underlying structure is characterized by sparse but complex relations.
- This paper proposes relational network for relational reasoning with neural networks.



Relational networks

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

Relational networks

- An RN is a neural network module with a structure primed for relational reasoning.
- The design philosophy behind RNs is to constrain the functional form of a neural network so that it captures the core common properties of relational reasoning.

Relational networks

- An RN is a neural network module with a structure primed for relational reasoning.
- The design philosophy behind RNs is to constrain the functional form of a neural network so that it captures the core common properties of relational reasoning.
- $RN(O) = f_\phi(\sum_{i,j} g_\theta(o_i, o_j))$

Relational networks

- An RN is a neural network module with a structure primed for relational reasoning.
- The design philosophy behind RNs is to constrain the functional form of a neural network so that it captures the core common properties of relational reasoning.
- $RN(O) = f_{\phi}(\sum_{i,j} g_{\theta}(o_i, o_j))$
- The input here is set of objects.

Relational networks

- An RN is a neural network module with a structure primed for relational reasoning.
- The design philosophy behind RNs is to constrain the functional form of a neural network so that it captures the core common properties of relational reasoning.
- $RN(O) = f_{\phi}(\sum_{i,j} g_{\theta}(o_i, o_j))$
- The input here is set of objects.
- f_{ϕ} and g_{θ} are MLPs, and the parameters are learnable synaptic weights, making RNs end-to-end differentiable.

Relational networks

- An RN is a neural network module with a structure primed for relational reasoning.
- The design philosophy behind RNs is to constrain the functional form of a neural network so that it captures the core common properties of relational reasoning.
- $RN(O) = f_\phi(\sum_{i,j} g_\theta(o_i, o_j))$
- The input here is set of objects.
- f_ϕ and g_θ are MLPs, and the parameters are learnable synaptic weights, making RNs end-to-end differentiable.
- **RNs learn to infer relations** The functional form in Equation shows that an RN should consider the potential relations between all object pairs.

Relational networks

- An RN is a neural network module with a structure primed for relational reasoning.
- The design philosophy behind RNs is to constrain the functional form of a neural network so that it captures the core common properties of relational reasoning.
- $RN(O) = f_\phi(\sum_{i,j} g_\theta(o_i, o_j))$
- The input here is set of objects.
- f_ϕ and g_θ are MLPs, and the parameters are learnable synaptic weights, making RNs end-to-end differentiable.
- **RNs learn to infer relations** The functional form in Equation shows that an RN should consider the potential relations between all object pairs.
- Not knowing the actual relations, RNs must learn to infer the existance of relations.

Relational networks

- **RNs are data efficient** RNs use a single function g_θ to compute each relation.

Relational networks

- **RNs are data efficient** RNs use a single function g_θ to compute each relation.
- This mode of operation encourages greater generalization for computing relations, since g_θ is encouraged not to over-fit to the features of any particular object pair.

Relational networks

- **RNs are data efficient** RNs use a single function g_θ to compute each relation.
- This mode of operation encourages greater generalization for computing relations, since g_θ is encouraged not to over-fit to the features of any particular object pair.
- There will be $O(n^2)$ passes over g_θ. At the same times each example has so many forward passes to learn.

Relational networks

- **RNs are data efficient** RNs use a single function g_θ to compute each relation.
- This mode of operation encourages greater generalization for computing relations, since g_θ is encouraged not to over-fit to the features of any particular object pair.
- There will be $O(n^2)$ passes over g_θ. At the same times each example has so many forward passes to learn.
- **RNs operate on a set of objects** RNs are order invariant of object sequence. This invariance ensures that the RN's output contains information that is generally representative of the relations that exist in the object set.

Relational networks

- In CLEVR, many questions are explicitly relational in nature. Remarkably, powerful QA architectures are unable to solve CLEVR, presumably because they cannot handle core relational aspects of the task.
  Model:

Relational networks

- In CLEVR, many questions are explicitly relational in nature. Remarkably, powerful QA architectures are unable to solve CLEVR, presumably because they cannot handle core relational aspects of the task.
  Model:
- Get $d \times d$ feature map from CNN.

Relational networks

- In CLEVR, many questions are explicitly relational in nature. Remarkably, powerful QA architectures are unable to solve CLEVR, presumably because they cannot handle core relational aspects of the task.
  Model:
- Get $d \times d$ feature map from CNN.
- Each of the d^2 k-dimensional cells in the $d \times d$ feature maps was tagged with an arbitrary coordinate indicating its relative spatial position, and was treated as an object for the RN.

Relational networks

- In CLEVR, many questions are explicitly relational in nature. Remarkably, powerful QA architectures are unable to solve CLEVR, presumably because they cannot handle core relational aspects of the task.
  Model:
- Get $d \times d$ feature map from CNN.
- Each of the d^2 k-dimensional cells in the $d \times d$ feature maps was tagged with an arbitrary coordinate indicating its relative spatial position, and was treated as an object for the RN.
- The existence and meaning of an object-object relation should be question dependent.

Relational networks

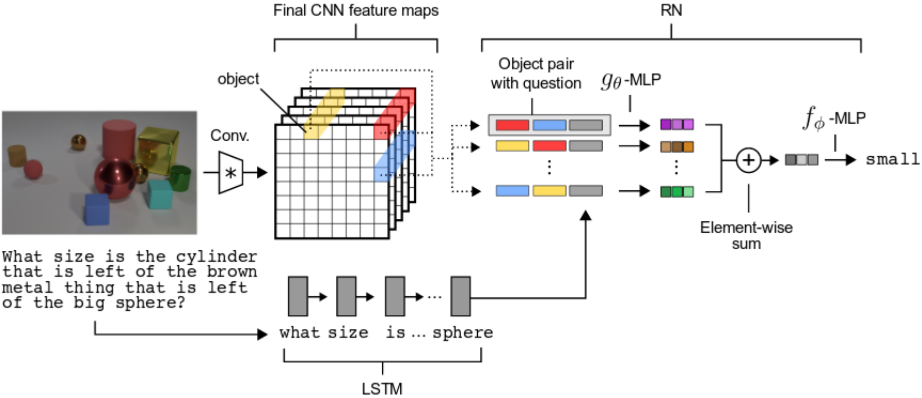
- In CLEVR, many questions are explicitly relational in nature. Remarkably, powerful QA architectures are unable to solve CLEVR, presumably because they cannot handle core relational aspects of the task.

  Model:
- Get $d \times d$ feature map from CNN.
- Each of the d^2 k-dimensional cells in the $d \times d$ feature maps was tagged with an arbitrary coordinate indicating its relative spatial position, and was treated as an object for the RN.
- The existence and meaning of an object-object relation should be question dependent.
- So the RN architecture is modified such that g_θ could condition its processing on the question: $a = f_\phi(\sum_{i,j} g_\theta(o_i, o_j, q))$.

Relational networks

- In CLEVR, many questions are explicitly relational in nature. Remarkably, powerful QA architectures are unable to solve CLEVR, presumably because they cannot handle core relational aspects of the task.
  Model:
- Get $d \times d$ feature map from CNN.
- Each of the d^2 k-dimensional cells in the $d \times d$ feature maps was tagged with an arbitrary coordinate indicating its relative spatial position, and was treated as an object for the RN.
- The existence and meaning of an object-object relation should be question dependent.
- So the RN architecture is modified such that g_θ could condition its processing on the question: $a = f_\phi(\sum_{i,j} g_\theta(o_i, o_j, q))$.
- LSTM is used to get question embedding.

Relational networks



Relational networks

- CNN- 4 convolutional layers each with 24 kernel

Relational networks

- CNN- 4 convolutional layers each with 24 kernel
- 128 unit LSTM for question processing, 32 unit word-lookup embedding

Relational networks

- CNN- 4 convolutional layers each with 24 kernel
- 128 unit LSTM for question processing, 32 unit word-lookup embedding
- 4-layer MLP consisting of 256 units per layer with ReLU non-linearities for g_θ.

Relational networks

- CNN- 4 convolutional layers each with 24 kernel
- 128 unit LSTM for question processing, 32 unit word-lookup embedding
- 4-layer MLP consisting of 256 units per layer with ReLU non-linearities for g_θ.
- 3-layer MLP consisting of 256, 256 (with 50% dropout), and 29 units with ReLU non-linearities for f_ϕ.

Relational networks

- CNN- 4 convolutional layers each with 24 kernel
- 128 unit LSTM for question processing, 32 unit word-lookup embedding
- 4-layer MLP consisting of 256 units per layer with ReLU non-linearities for g_θ.
- 3-layer MLP consisting of 256, 256 (with 50% dropout), and 29 units with ReLU non-linearities for f_ϕ.
- **No attention No ResNet**

Relational networks

- CNN- 4 convolutional layers each with 24 kernel
- 128 unit LSTM for question processing, 32 unit word-lookup embedding
- 4-layer MLP consisting of 256 units per layer with ReLU non-linearities for g_θ.
- 3-layer MLP consisting of 256, 256 (with 50% dropout), and 29 units with ReLU non-linearities for f_ϕ.
- **No attention No ResNet**
- Training was end-to-end with Adam optimizer with a learning rate of $2.5e^{-4}$

Relational networks

Model	Overall	Count	Exist	Compare numbers	Query	Compare attribute
Human	92.6	86.7	96.6	86.5	95.0	96.0
LSTM	46.8	41.7	61.1	69.8	36.8	51.8
CNN+RNN	52.3	43.7	65.2	67.1	49.3	53.0
SA	68.5	52.2	71.1	73.5	85.3	52.3
NMN	72.1	79.3	52.5	71.4	78.9	78
N2NMN	83.7	85.7	68.5	83.7	90	88.7
PG	96.9	97.1	92.7	98.6	98.1	98.9
RN	95.5	90.1	97.8	93.6	97.9	97.1

Performance on bAbI:

- Famous dataset for text-based QA.

Relational networks

Performance on bAbI:

- Famous dataset for text-based QA.
- There are 20 tasks, each corresponding to a particular type of reasoning.

Performance on bAbI:

- Famous dataset for text-based QA.
- There are 20 tasks, each corresponding to a particular type of reasoning.
- Each question is associated with a set of supporting facts.

Relational networks

Performance on bAbI:

- Famous dataset for text-based QA.
- There are 20 tasks, each corresponding to a particular type of reasoning.
- Each question is associated with a set of supporting facts.
- For example, the facts Sandra picked up the football and Sandra went to the office support the question Where is the football? (answer: **office**).

Relational networks

Performance on bAbI:

- Famous dataset for text-based QA.
- There are 20 tasks, each corresponding to a particular type of reasoning.
- Each question is associated with a set of supporting facts.
- For example, the facts Sandra picked up the football and Sandra went to the office support the question Where is the football? (answer: **office**).
- A model succeeds on a task if its performance surpasses 95%.

Relational networks

Performance on bAbI:

- Famous dataset for text-based QA.
- There are 20 tasks, each corresponding to a particular type of reasoning.
- Each question is associated with a set of supporting facts.
- For example, the facts Sandra picked up the football and Sandra went to the office support the question Where is the football? (answer: **office**).
- A model succeeds on a task if its performance surpasses 95%.
- Memory networks pass 14 out of 20.

Relational networks

Model:

- Identify up to 20 sentences in the support set that were immediately prior to the probe question.

Relational networks

Model:

- Identify up to 20 sentences in the support set that were immediately prior to the probe question.
- Tag these sentences with labels indicating their relative position in the support set, and processed each sentence word-by-word with an LSTM.

Relational networks

Model:

- Identify up to 20 sentences in the support set that were immediately prior to the probe question.
- Tag these sentences with labels indicating their relative position in the support set, and processed each sentence word-by-word with an LSTM.
- Previous bAbI models processed all word tokens from all support sentences sequentially.

Relational networks

Model:

- Identify up to 20 sentences in the support set that were immediately prior to the probe question.
- Tag these sentences with labels indicating their relative position in the support set, and processed each sentence word-by-word with an LSTM.
- Previous bAbI models processed all word tokens from all support sentences sequentially.
- The final state of the sentence-processing-LSTM is considered to be an object.

Relational networks

Model:

- Identify up to 20 sentences in the support set that were immediately prior to the probe question.
- Tag these sentences with labels indicating their relative position in the support set, and processed each sentence word-by-word with an LSTM.
- Previous bAbI models processed all word tokens from all support sentences sequentially.
- The final state of the sentence-processing-LSTM is considered to be an object.
- The model succeeded on 18 tasks.

Model:

- Identify up to 20 sentences in the support set that were immediately prior to the probe question.
- Tag these sentences with labels indicating their relative position in the support set, and processed each sentence word-by-word with an LSTM.
- Previous bAbI models processed all word tokens from all support sentences sequentially.
- The final state of the sentence-processing-LSTM is considered to be an object.
- The model succeeded on 18 tasks.
- Where other models failed miserably on these tasks, the model missed the 95% cut by 3.1% and 11.5%.

Performance on dynamic physical systems

- The paper introduces a dataset for 2 tasks on MuJoCo physics engine.

Performance on dynamic physical systems

- The paper introduces a dataset for 2 tasks on MuJoCo physics engine.
- Each scene contained 10 colored balls moving on a table-top surface.

Performance on dynamic physical systems

- The paper introduces a dataset for 2 tasks on MuJoCo physics engine.
- Each scene contained 10 colored balls moving on a table-top surface.
- Randomly selected ball pairs were connected by invisible springs or a rigid constraint. These connections prevented the balls from moving independently.

Performance on dynamic physical systems

- The paper introduces a dataset for 2 tasks on MuJoCo physics engine.
- Each scene contained 10 colored balls moving on a table-top surface.
- Randomly selected ball pairs were connected by invisible springs or a rigid constraint. These connections prevented the balls from moving independently.
- The paper introduces 2 tasks: We defined two separate tasks: 1) infer the existence or absence of connections between balls when only observing their color and coordinate positions across multiple sequential frames, and 2) count the number of systems on the table-top.

Performance on dynamic physical systems

- Both of these tasks involve reasoning about the relative positions and velocities of the balls to infer.

Performance on dynamic physical systems

- Both of these tasks involve reasoning about the relative positions and velocities of the balls to infer.
- In the connection inference task, our model correctly classified all the connections in 93% of the sample scenes in the test set.

Relational networks

Performance on dynamic physical systems

- Both of these tasks involve reasoning about the relative positions and velocities of the balls to infer.
- In the connection inference task, our model correctly classified all the connections in 93% of the sample scenes in the test set.
- In the counting task, the RN achieved similar performance, reporting the correct number of connected systems for 95% of the test scene samples.

Relational networks

Performance on dynamic physical systems

- Both of these tasks involve reasoning about the relative positions and velocities of the balls to infer.
- In the connection inference task, our model correctly classified all the connections in 93% of the sample scenes in the test set.
- In the counting task, the RN achieved similar performance, reporting the correct number of connected systems for 95% of the test scene samples.
- An MLP with comparable number of parameters was unable to perform better than chance for both tasks.

Relational networks

- This paper shows the difference between processing and reasoning. Saying, ResNets are highly capable visual processors, but they may not be the most appropriate choice for reasoning about arbitrary relations.

Relational networks

- This paper shows the difference between processing and reasoning. Saying, ResNets are highly capable visual processors, but they may not be the most appropriate choice for reasoning about arbitrary relations.
- Demonstrate the RN's rich capacity for structured reasoning even with unstructured inputs and outputs.

FiLM

- A model made from general-purpose components that could learn to visually reason, would likely be more widely applicable across domains.

FiLM

- A model made from general-purpose components that could learn to visually reason, would likely be more widely applicable across domains.
- A FiLM layer carries out a simple, feature-wise affine transformation on a neural network's intermediate features, conditioned on an arbitrary input.

FiLM

- A model made from general-purpose components that could learn to visually reason, would likely be more widely applicable across domains.
- A FiLM layer carries out a simple, feature-wise affine transformation on a neural network's intermediate features, conditioned on an arbitrary input.
- FiLM can be thought of as a generalization of Conditional Normalization, which has proven highly successful for image stylization.

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated
- $\gamma_{i,c} = f_c(x_i)$

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated
- $\gamma_{i,c} = f_c(x_i)$
- $\beta_{i,c} = h_c(x_i)$

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated
- $\gamma_{i,c} = f_c(x_i)$
- $\beta_{i,c} = h_c(x_i)$
- $\gamma_{i,c}$ and $\beta_{i,c}$ modulate a neural network's activations $F_{i,c}$.

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated
- $\gamma_{i,c} = f_c(x_i)$
- $\beta_{i,c} = h_c(x_i)$
- $\gamma_{i,c}$ and $\beta_{i,c}$ modulate a neural network's activations $F_{i,c}$.
- $FiLM(F_{i,c}|\gamma_{i,c}, \beta_{i,c}) = \gamma_{i,c} F_{i,c} + \beta_{i,c}$

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated
- $\gamma_{i,c} = f_c(x_i)$
- $\beta_{i,c} = h_c(x_i)$
- $\gamma_{i,c}$ and $\beta_{i,c}$ modulate a neural network's activations $F_{i,c}$.
- $FiLM(F_{i,c}|\gamma_{i,c}, \beta_{i,c}) = \gamma_{i,c}F_{i,c} + \beta_{i,c}$
- FiLM layers empower the FiLM generator to manipulate feature maps of a target, FiLM-ed network by scaling them up or down, negating them, shutting them off, selectively thresholding them (when followed by a ReLU), and more.

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated
- $\gamma_{i,c} = f_c(x_i)$
- $\beta_{i,c} = h_c(x_i)$
- $\gamma_{i,c}$ and $\beta_{i,c}$ modulate a neural network's activations $F_{i,c}$.
- $FiLM(F_{i,c}|\gamma_{i,c}, \beta_{i,c}) = \gamma_{i,c}F_{i,c} + \beta_{i,c}$
- FiLM layers empower the FiLM generator to manipulate feature maps of a target, FiLM-ed network by scaling them up or down, negating them, shutting them off, selectively thresholding them (when followed by a ReLU), and more.
- Each feature map is conditioned independently, giving the FiLM generator moderately fine-grained control over activations at each FiLM layer.

FiLM

- FiLM learns to adaptively influence the output of a neural network by applying an linear transformation.
- FiLM - Feature-wise Linearly Modulated
- $\gamma_{i,c} = f_c(x_i)$
- $\beta_{i,c} = h_c(x_i)$
- $\gamma_{i,c}$ and $\beta_{i,c}$ modulate a neural network's activations $F_{i,c}$.
- $FiLM(F_{i,c}|\gamma_{i,c}, \beta_{i,c}) = \gamma_{i,c}F_{i,c} + \beta_{i,c}$
- FiLM layers empower the FiLM generator to manipulate feature maps of a target, FiLM-ed network by scaling them up or down, negating them, shutting them off, selectively thresholding them (when followed by a ReLU), and more.
- Each feature map is conditioned independently, giving the FiLM generator moderately fine-grained control over activations at each FiLM layer.
- It is a scalable and computationally efficient conditioning method.

FiLM

Model:

- Model consists of a FiLM-generating linguistic pipeline and a FiLM-ed visual pipeline.

FiLM

Model:

- Model consists of a FiLM-generating linguistic pipeline and a FiLM-ed visual pipeline.
- The FiLM generator processes a question x_i using a Gated Recurrent Unit (GRU) network with 4096 hidden units that takes in learned, 200-dimensional word embeddings.

FiLM

Model:

- Model consists of a FiLM-generating linguistic pipeline and a FiLM-ed visual pipeline.
- The FiLM generator processes a question x_i using a Gated Recurrent Unit (GRU) network with 4096 hidden units that takes in learned, 200-dimensional word embeddings.
- The final GRU hidden state is a question embedding, from which the model predicts (γ_i^n, β_i^n) for each n-th residual block.

FiLM

Model:

- Model consists of a FiLM-generating linguistic pipeline and a FiLM-ed visual pipeline.
- The FiLM generator processes a question x_i using a Gated Recurrent Unit (GRU) network with 4096 hidden units that takes in learned, 200-dimensional word embeddings.
- The final GRU hidden state is a question embedding, from which the model predicts (γ_i^n, β_i^n) for each n-th residual block.
- The visual pipeline extracts 128 14×14 image feature maps from a resized, 224×224 image input using CNN.

FiLM

Model:

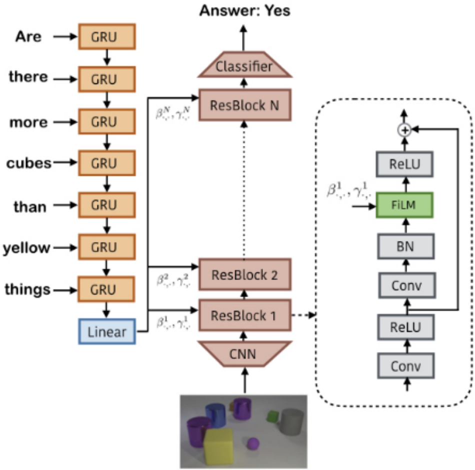
- Model consists of a FiLM-generating linguistic pipeline and a FiLM-ed visual pipeline.
- The FiLM generator processes a question x_i using a Gated Recurrent Unit (GRU) network with 4096 hidden units that takes in learned, 200-dimensional word embeddings.
- The final GRU hidden state is a question embedding, from which the model predicts (γ_i^n, β_i^n) for each n-th residual block.
- The visual pipeline extracts 128 14×14 image feature maps from a resized, 224×224 image input using CNN.
- The CNN architecture is similar to ResNet-101 with a added FiLM module. It was pretrained on Image-Net.

FiLM

Model:

- Model consists of a FiLM-generating linguistic pipeline and a FiLM-ed visual pipeline.
- The FiLM generator processes a question x_i using a Gated Recurrent Unit (GRU) network with 4096 hidden units that takes in learned, 200-dimensional word embeddings.
- The final GRU hidden state is a question embedding, from which the model predicts (γ_i^n, β_i^n) for each n-th residual block.
- The visual pipeline extracts 128 14×14 image feature maps from a resized, 224×224 image input using CNN.
- The CNN architecture is similar to ResNet-101 with a added FiLM module. It was pretrained on Image-Net.
- Model is trained end-to-end from scratch with Adam (learning rate 3e-4), weight decay (1e-5), batch size 64,and batch normalization and ReLU throughout FiLM-ed network.

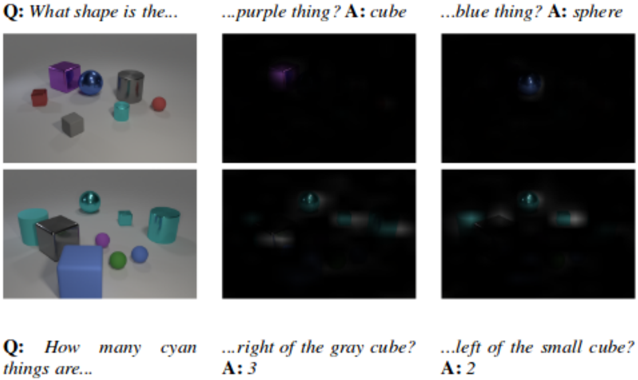
FiLM



Answer: Yes
Are
there
more
cubes
than
yellow
things
GRU
Linear
Classifier
ResBlock N
ResBlock 2
ResBlock 1
CNN
ReLU
FiLM
BN
Conv
ReLU
Conv

FiLM

- Images reveal that the FiLM model predicts using features areas near answer-related or question-related objects.



FiLM

Model	Overall	Count	Exist	Compare numbers	Query	Compare attribute
Human	92.6	86.7	96.6	86.5	95.0	96.0
LSTM	46.8	41.7	61.1	69.8	36.8	51.8
CNN+RNN	52.3	43.7	65.2	67.1	49.3	53.0
SA	68.5	52.2	71.1	73.5	85.3	52.3
NMN	72.1	79.3	52.5	71.4	78.9	78
N2NMN	83.7	85.7	68.5	83.7	90	88.7
PG	96.9	97.1	92.7	98.6	98.1	98.9
RN	95.5	90.1	97.8	93.6	97.9	97.1
FiLM	97.7	94.3	99.1	96.8	99.1	99.1

We saw 3 different innovative ways to achieve super human performance in this task.

Thank You.