Neural program synthesis

Rakitin Denis

Higher School of Economics

Overview

- Introduction
- Programming by example
- FlashFill and RobustFill
- Neural-Symbolic VQA
- Neuro-Symbolic Concept Learner

Motivation

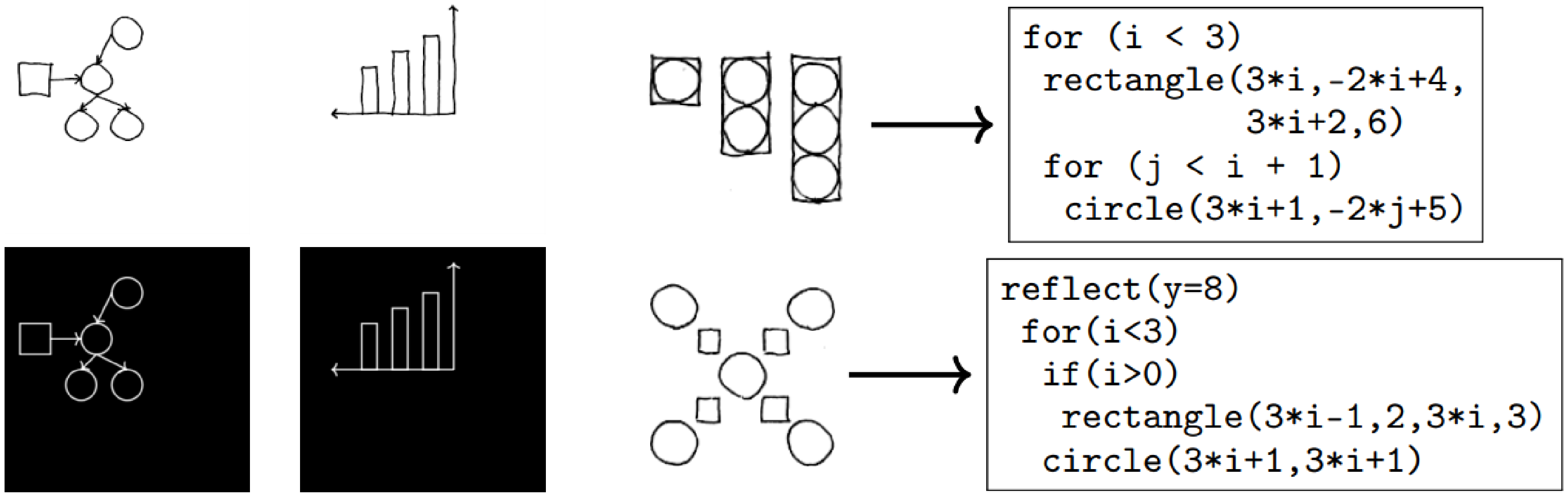
- Learning concepts from a small amount of data
- Could be seen as a potential AI benchmark

- Combining with neural models could result in benefits in both directions

Examples: strings editing

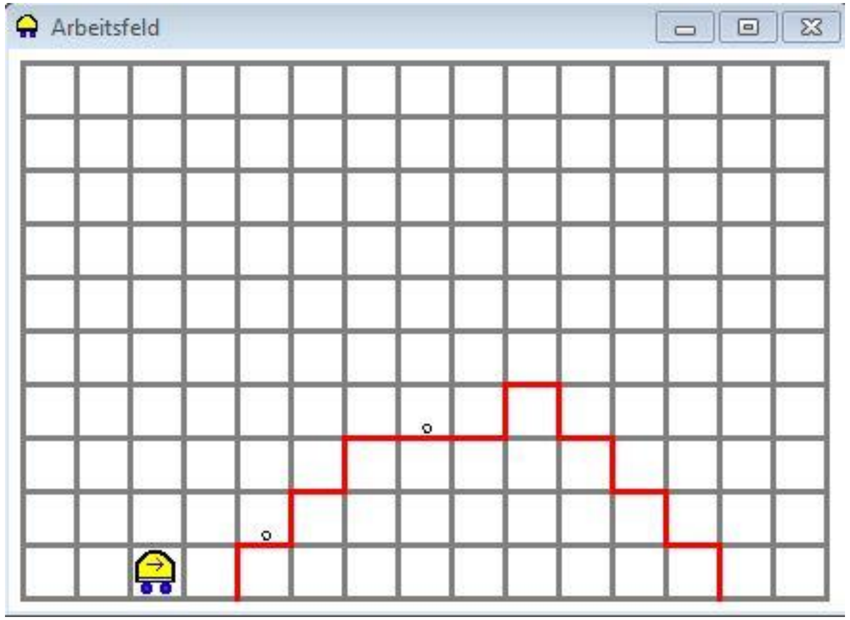
<table>
<tr><th>Input String</th><th>Output String</th></tr>
<tr><td>john Smith
DOUG Q. Macklin
Frank Lee (123)
Laura Jane Jones</td><td>Smith, Jhn
Macklin, Doug
LEe, Frank
Jones, Laura</td></tr>
<tr><td>Steve P. Green (9)</td><td>?</td></tr>
<tr><th colspan="2">Program</th></tr>
<tr><td colspan="2"><code>GetToken(Alpha, -1) | ',' | ' ' | ToCase(Proper, GetToken(Alpha, 1))</code></td></tr>
</table>

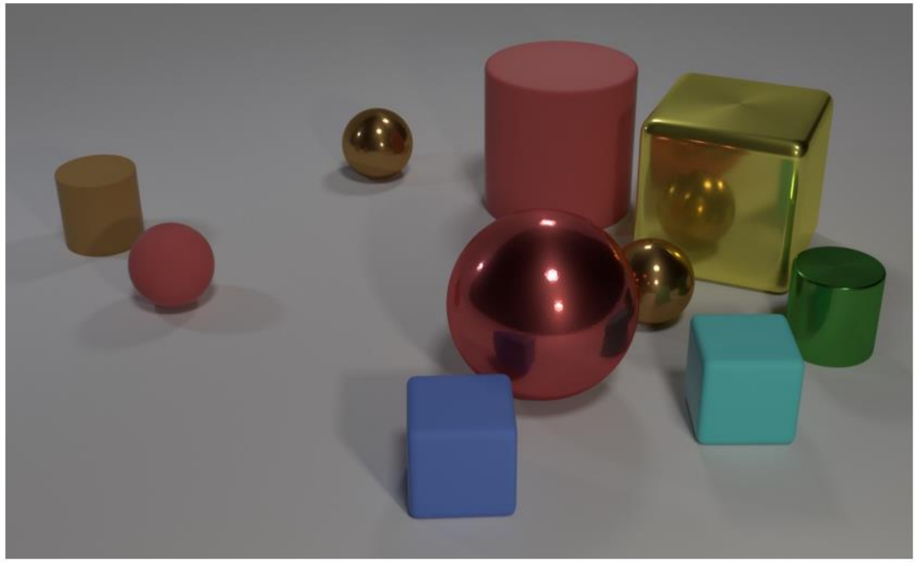
Examples: graphics generation



https://arxiv.org/abs/1707.09627

Examples: many more





Q: Are there an equal number of large things and metal spheres?

Program learning as a main task

Program learning basics

Problems:

- Giving a full specification is intractable in practice
- A programming language is needed

Solutions:

- An approximate description defined by I/O examples
- Design a domain-specific language (DSL)

Programming by example

Inputs:

$\{x_k, y_k\}_{k=1}^{n}$ - observed I/O examples

$\{x_k^{test}\}_{k=1}^{m}$ - test (assessment) input examples

Outputs:

$\{\hat{y}_k^{test}\}_{k=1}^{m}$ - outputs for corresponding test inputs

If program π is derived explicitly, we set

$$\hat{y}_k^{test} = \pi(x_k^{test})$$

and require

$$\pi(x_k) = y_k \ \forall k \in \ \{1, \ldots n\}$$

Neural program learning

Usually modeled as a sequence-to-sequence task

Two main approaches:

- Neural program induction: generate an output directly using latent program representation
- Neural program synthesis: generate a program and evaluate it

Tokens are provided by a DSL

FlashFill

A legendary tool for string editing in Microsoft Excel

- Uses a very small number of I/O examples
- Works invisibly fast

A competitive rule-based PL algorithm

- Expressive DSL with a large number of editing procedures
- Smart search techniques and heuristics

Drawbacks:

- Fragile to noise (e.g. typos)
- A lot of engineering

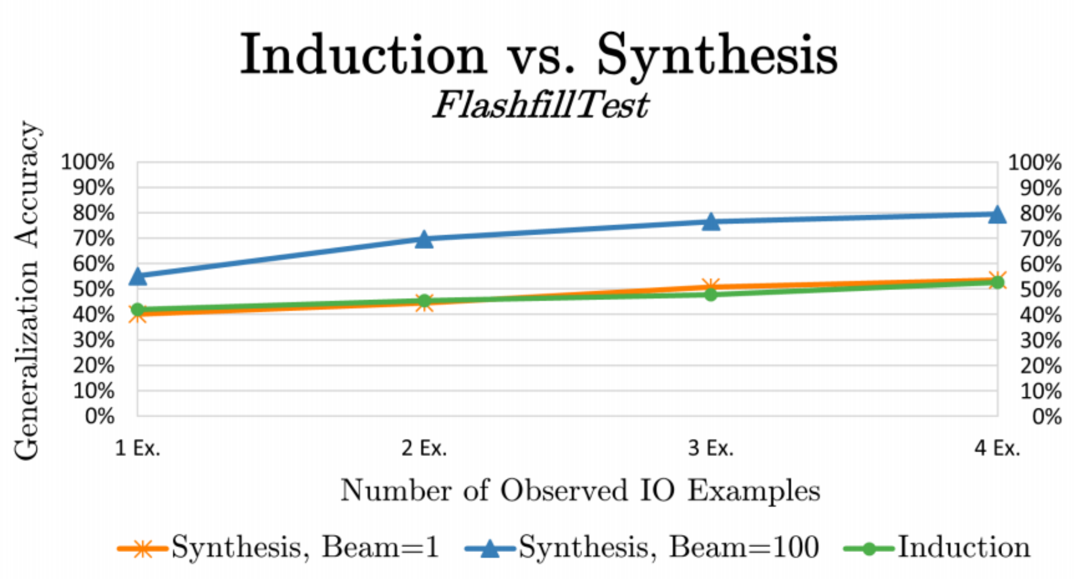
Original paper by Gulwani et al., 2011

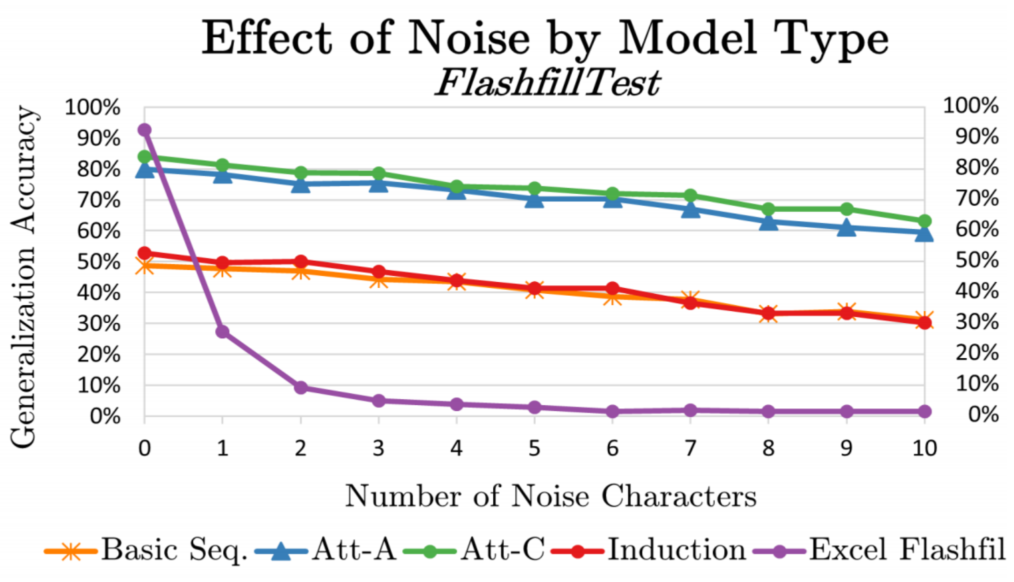
RobustFill: properties

- Uses a neural PL approach: synthesis and induction are compared
- Bases on a hand-crafted DSL
- Multi-attentional architecture
- Synthetically generated programs for learning
- Achieves comparable (with FlashFill) and even better results

https://arxiv.org/abs/1703.07469

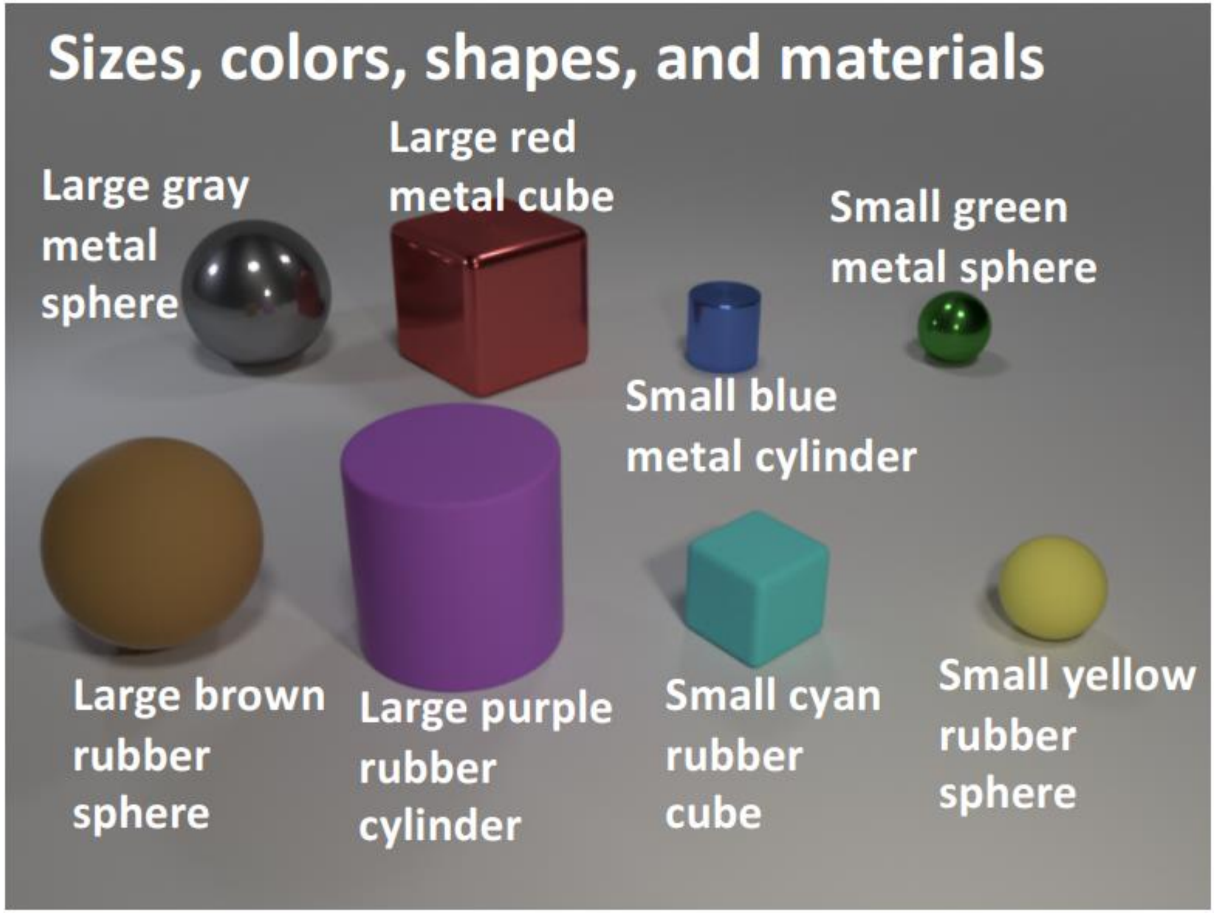
RobustFill: results





PL module as a building block

CLEVR


Sizes, colors, shapes, and materials
Large gray metal sphere
Large red metal cube
Small green metal sphere
Small blue metal cylinder
Large brown rubber sphere
Large purple rubber cylinder
Small cyan rubber cube
Small yellow rubber sphere

CLEVR

- Example question:

How many red things are there?

How many <C> and <M> things are there?

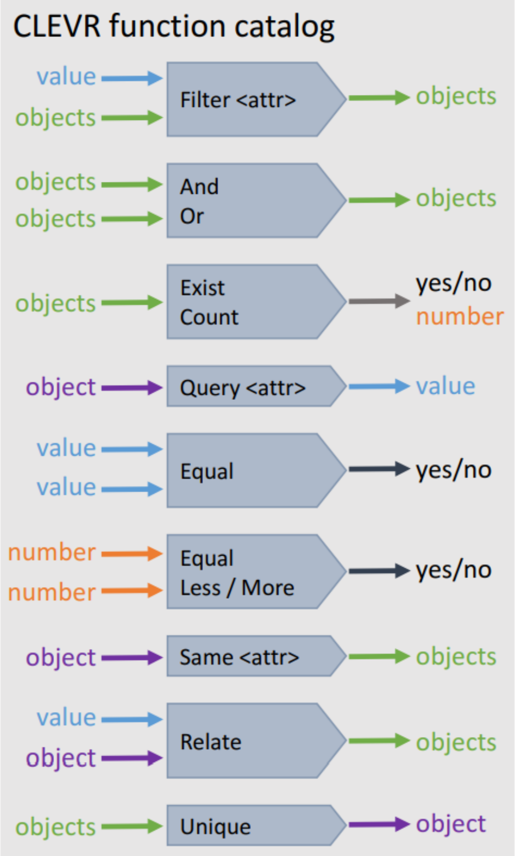
<C> -> red, <M> -> nil

- Example program:

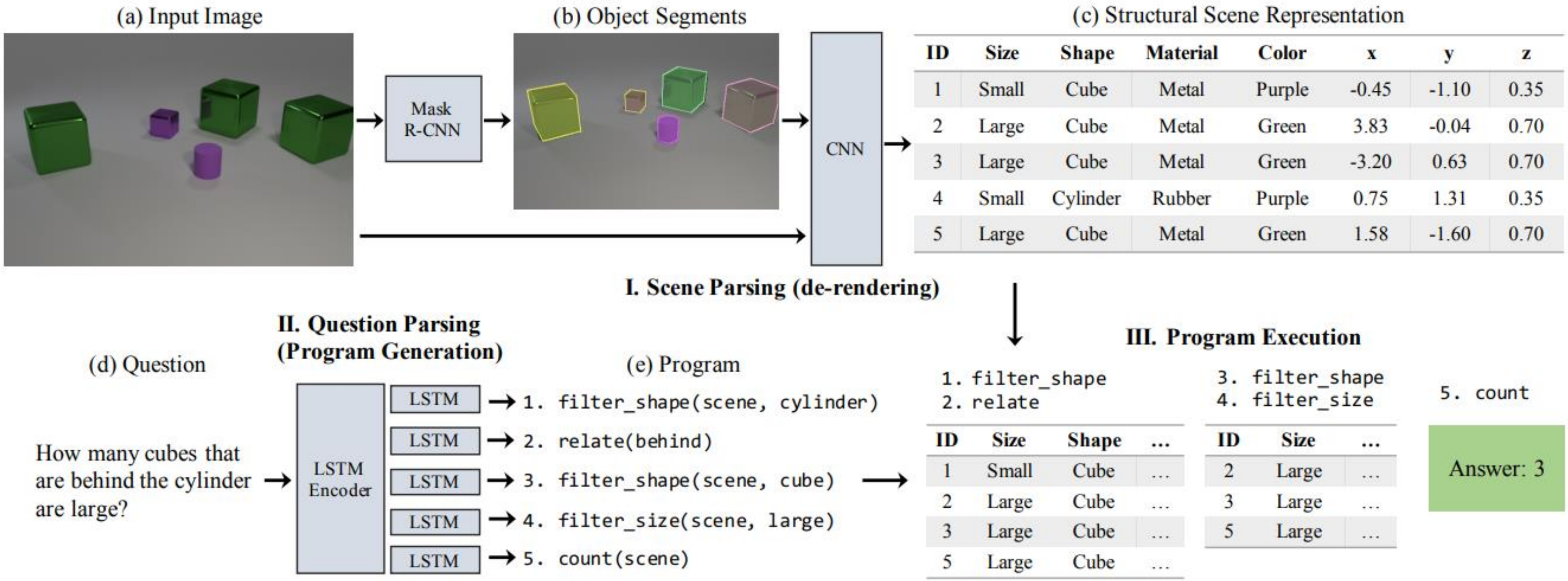
count(filter color(

<C>, filter material(<M>, scene())

))

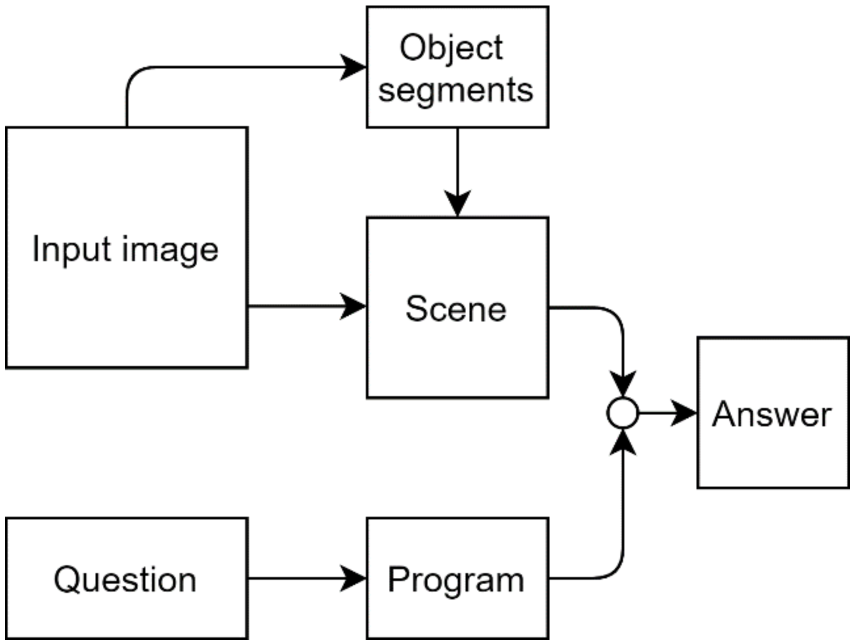


Neural-Symbolic(NS) VQA



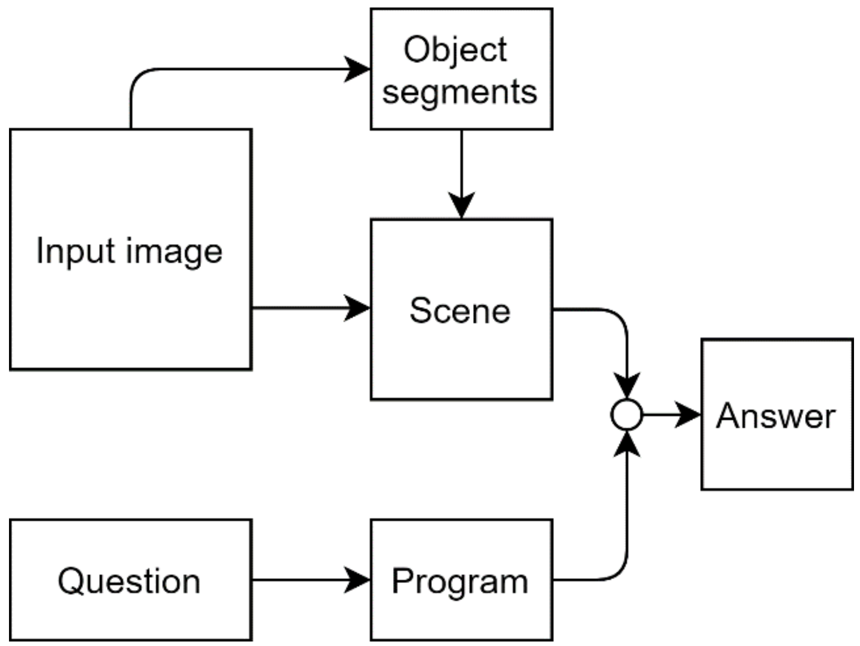
NS-VQA: scene parser

- Mask R-CNN -> segment proposals
- A set of categorical values -> class
- Bounding box score < 0.9 -> drop
- Object -> (image, segment) -> ResNet-34



NS-VQA: question parser

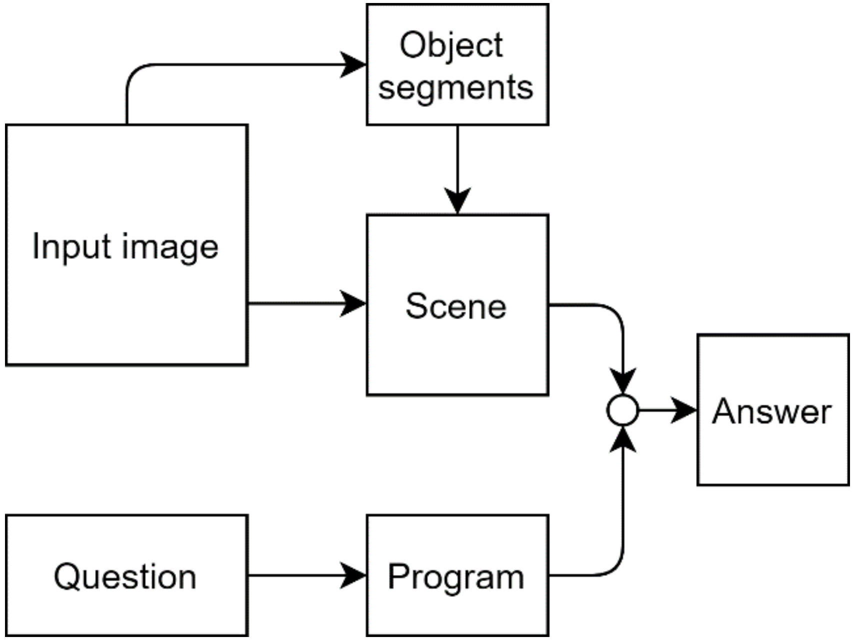
- Attention-based seq2seq model
- Encoder is a bidirectional LSTM
- Decoder is an LSTM



- 2 hidden layers with hidden dim 256
- Encoder and decoder activations have dim 300

NS-VQA: program executor

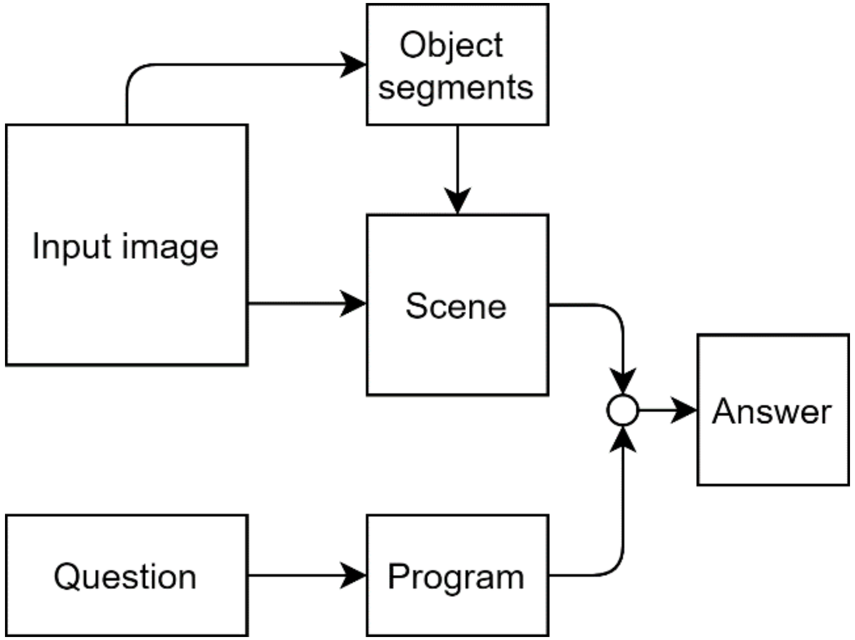
- DSL consists of functional Python modules
- Python modules <-> CLEVR DSL operators
- Program has sequential structure



NS-VQA: training scene parser

Scene parser:

- 4000 CLEVR images with annotations
- Mask R-CNN: 30000 iterations.
- Feature extractor CNN: MSE Loss, 30000 iterations.



NS-VQA: training question parser

First stage:

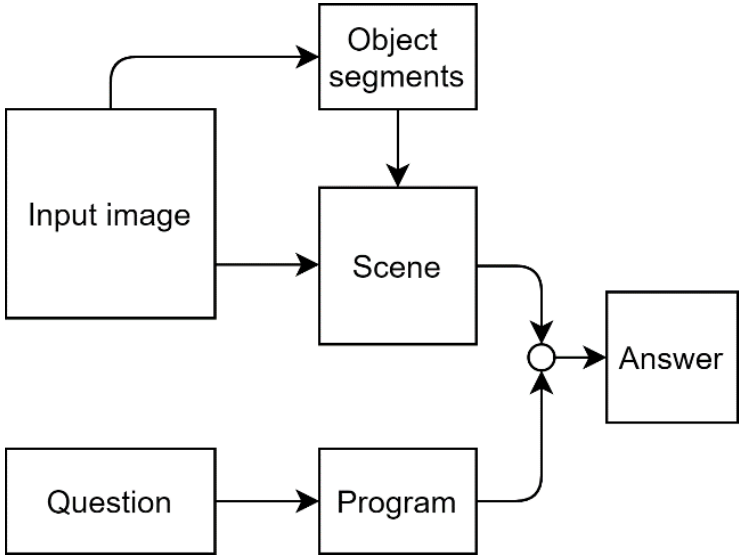
- Supervised pretraining on (q, p) pairs

Second stage:

- REINFORCE :

θ $-$ scene parser parameters, φ $-$ question parser parameters

$$\mathbb{E}_{I,Q,A\sim p_{data}(I,Q,A)}\mathbb{E}_{q_{\varphi}(\pi|Q)}I\{\,\pi(S_{\theta}(I)) = A\} \to \max_{\varphi}$$



NS-VQA: results

Methods	Count	Exist	Compare Number	Compare Attribute	Query Attribute	Overall
Humans [Johnson et al., 2017b]	86.7	96.6	86.4	96.0	95.0	92.6
CNN+LSTM+SAN [Johnson et al., 2017b]	59.7	77.9	75.1	70.8	80.9	73.2
N2NMN* [Hu et al., 2017]	68.5	85.7	84.9	88.7	90.0	83.7
Dependency Tree [Cao et al., 2018]	81.4	94.2	81.6	97.1	90.5	89.3
CNN+LSTM+RN [Santoro et al., 2017]	90.1	97.8	93.6	97.1	97.9	95.5
IEP* [Johnson et al., 2017b]	92.7	97.1	98.7	98.9	98.1	96.9
CNN+GRU+FiLM [Perez et al., 2018]	94.5	99.2	93.8	99.0	99.2	97.6
DDRprog* [Suarez et al., 2018]	96.5	98.8	98.4	99.0	99.1	98.3
MAC [Hudson and Manning, 2018]	97.1	99.5	99.1	99.5	99.5	98.9
TbD+reg+hres* [Mascharka et al., 2018]	97.6	99.2	99.4	99.6	99.5	99.1
NS-VQA (ours, 90 programs)	64.5	87.4	53.7	77.4	79.7	74.4
NS-VQA (ours, 180 programs)	85.0	92.9	83.4	90.6	92.2	89.5
NS-VQA (ours, 270 programs)	**99.7**	**99.9**	**99.9**	**99.8**	**99.8**	**99.8**

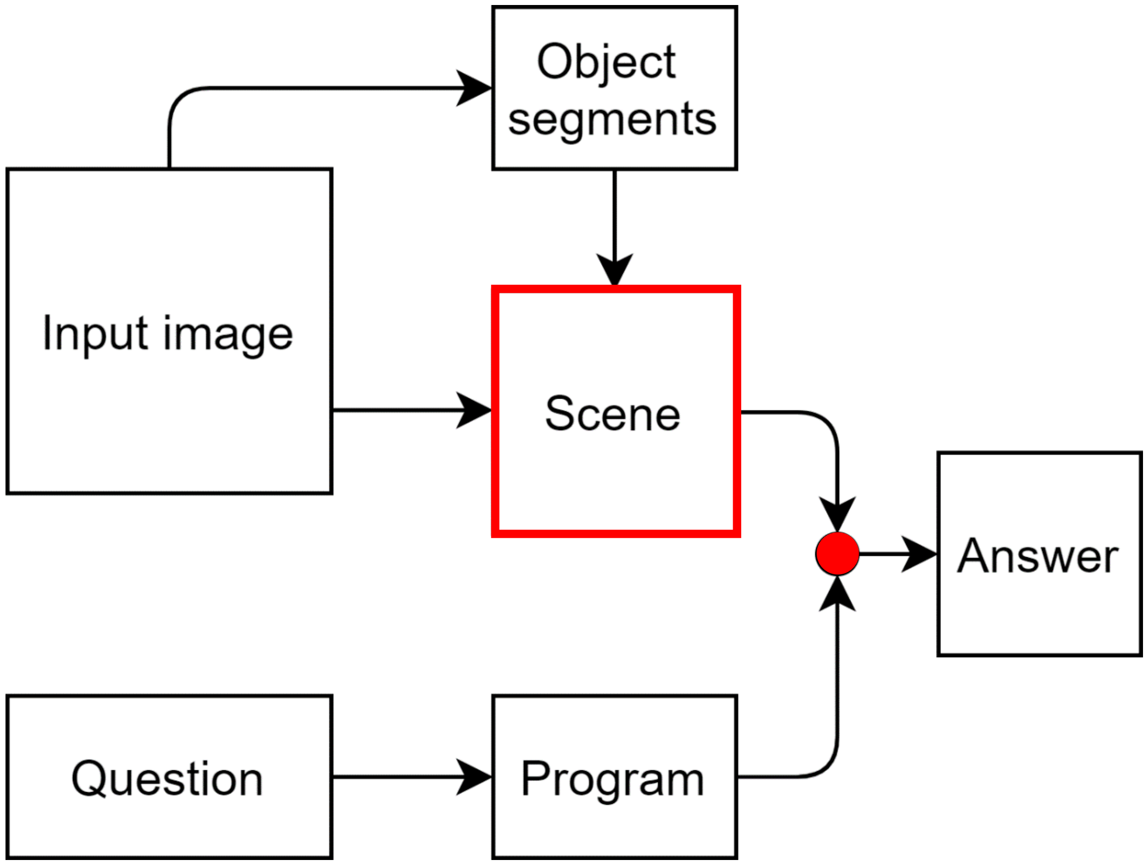
# Programs	NS-VQA	IEP
100	**60.2**	38.7
200	**65.2**	40.1
500	**67.8**	49.2
1K	**67.8**	63.4
18K	**67.0**	66.6

(b) Question answering accuracy on CLEVR-Humans.

NS-VQA: conclusion

- Near-perfect performance on CLEVR dataset
- A few number of ground-truth programs
- Heavily relies on the ground-truth scene decomposition

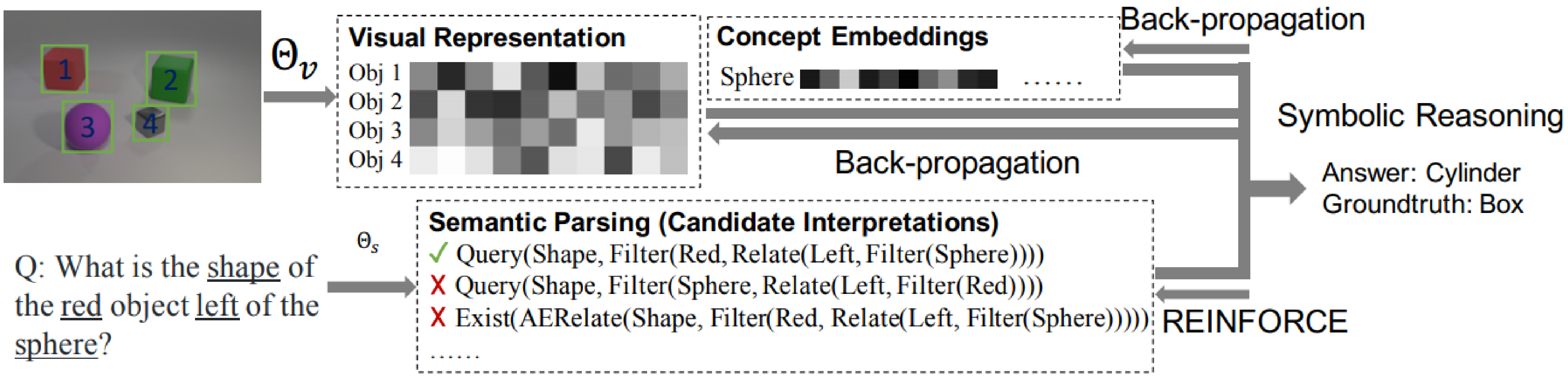
Neuro-symbolic concept learner (CL)



NS-CL: motivation

- No ground-truth scene representation
- No ground-truth programs
- How to propagate gradients?

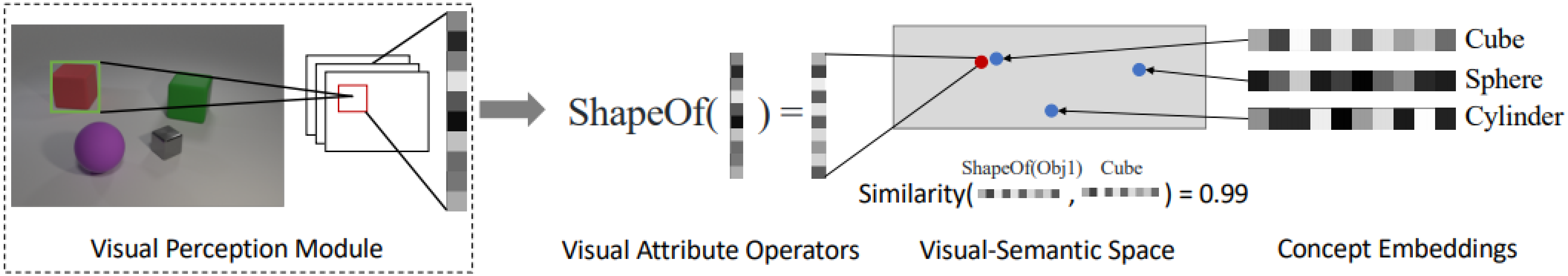
NS-CL: scheme



1
2
3
4
Θ_v
Visual Representation
Obj 1
Obj 2
Obj 3
Obj 4
Concept Embeddings
Sphere ......
Back-propagation
Symbolic Reasoning
Back-propagation
Answer: Cylinder
Groundtruth: Box
Q: What is the shape of the red object left of the sphere?
Θ_s
Semantic Parsing (Candidate Interpretations)
✓ Query(Shape, Filter(Red, Relate(Left, Filter(Sphere))))
X Query(Shape, Filter(Sphere, Relate(Left, Filter(Red))))
X Exist(AERelate(Shape, Filter(Red, Relate(Left, Filter(Sphere)))))
......
REINFORCE

NS-CL: quasy-symbolic properties

- Object concept-> learnable vector representation
- Object attribute -> differentiable operation



NS-CL: classifying

- Distance between embeddings is cosine distance $\langle \, , \rangle$
- Example:

$$P(o_i \; is \; cube) = \sigma_{\gamma,\tau}(\langle ShapeOf(o_i), v^{Cube} \rangle),$$

$$\sigma_{\gamma,\tau}(x) = \sigma((x \; - \gamma)/\tau)$$

- Each *concept* corresponds to an *attribute*
- Red -> Color, Cube -> Shape

NS-CL: properties

Feature extraction:

- Pretrained Mask R-CNN -> object proposals
- Image -> region-based(by ROI-align) , image-based features -> concat
- By adding image we add contextual information

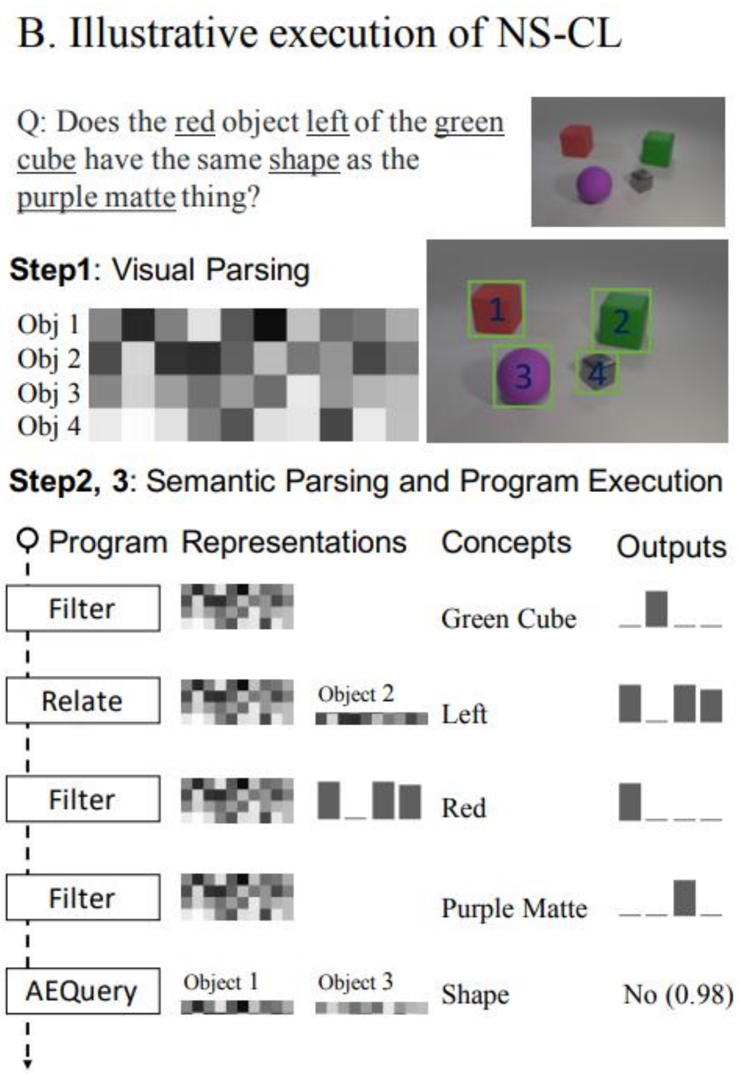
DSL and question parser:

- Question parser: bidirectional GRU encoder and GRU decoder
- DSL similar to NS-VQA one

NS-CL: quasy-symbolic executor

Probabilistic operations for differentiability:

- A set of output objects -> a set of probabilities to be in output, denoted as $Mask_i \in [0, 1]$
- Next input: (scene, mask)



NS-CL: training

$$\mathbb{E}_{q(\pi|\theta_s,\,Q)}P\big(A = Executor(Perception(S,\theta_v),\pi)\big) \to \max_{\theta_s,\theta_v}$$

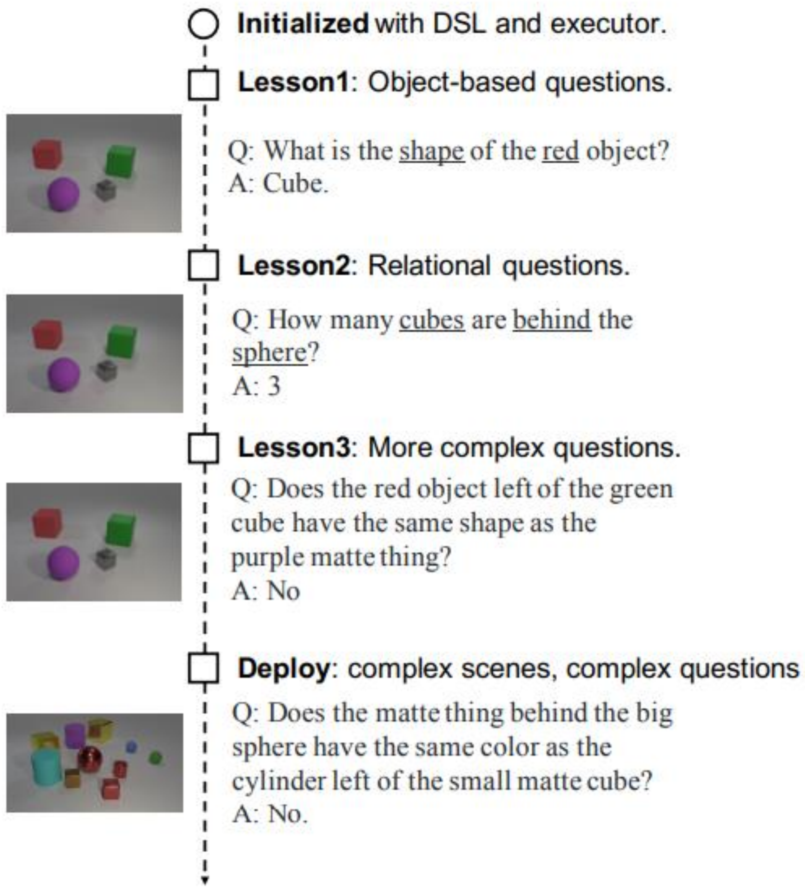
Visual scene parser (θ_v):

- Cross-entropy minimization between A and execution result

Semantic parser (θ_s):

- REINFORCE

NS-CL: curriculum learning



NS-CL: training

- 5000 CLEVR images
- 20 questions for each image
- Mask R-CNN pretrained on 4000 images as in NS-VQA

NS-CL: results

Validation dataset

Model	Prog. Anno.	Overall	Count	Cmp. Num.	Exist	Query Attr.	Cmp. Attr.
Human	N/A	92.6	86.7	86.4	96.6	95.0	96.0
NMN	700K	72.1	52.5	72.7	79.3	79.0	78.0
N2NMN	700K	88.8	68.5	84.9	85.7	90.0	88.8
IEP	700K	96.9	92.7	98.7	97.1	98.1	98.9
DDRprog	700K	98.3	96.5	98.4	98.8	99.1	99.0
TbD	700K	**99.1**	**97.6**	**99.4**	**99.2**	**99.5**	**99.6**
RN	0	95.5	90.1	93.6	97.8	97.1	97.9
FiLM	0	97.6	94.5	93.8	99.2	99.2	99.0
MAC	0	98.9	97.2	**99.4**	**99.5**	99.3	**99.5**
NS-CL	0	**98.9**	**98.2**	99.0	98.8	**99.3**	99.1

NS-CL: results

Data efficiency

Model	Visual	Accuracy (100% Data)	Accuracy (10% Data)
TbD	Attn.	99.1	54.2
TbD-Object	Obj.	84.1	52.6
TbD-Mask	Attn.	99.0	55.0
MAC	Attn.	98.9	67.3
MAC-Object	Obj.	79.5	51.2
MAC-Mask	Attn.	98.7	68.4
NS-CL	Obj.	**99.2**	**98.9**

Combinatorial generalization

Model	Test			
	Split A	Split B	Split C	Split D
MAC	97.3	N/A	92.9	N/A
IEP	96.1	92.1	91.5	90.9
TbD	98.8	94.5	94.3	91.9
NS-CL	**98.9**	**98.9**	**98.7**	**98.8**

NS-CL: conclusion

- Structural PL part with differentiable executor
- Near-perfect accuracy $\approx$ 99%
- No ground-truth programs
- No ground-truth scene representations

Thanks for attention!

NS-VQA: training details

Scene parser:

- Mask R-CNN: 30000 iterations, batch size = 8.
- Feature extractor CNN: MSE Loss, 30000 iterations, lr = 0.002, batch size = 50. ResNet-34 architecture.

Question parser:

- Pretraining: 20000 iterations, lr = $7 \cdot 10^{-4}$
- REINFORCE: at most 2M iterations, early stopping, lr = 10^{-5}, weight decay = 0.9, constant baseline

NS-CL: differentiable operations

- Differentiable in DL sense: some pooling-like operations are present
- Smooth versions of deterministic operations on sets

`Filter`(in: ObjectSet, oc: ObjConcept) $\rightarrow$ out: ObjectSet	$out_i := \min(in_i, \mathrm{ObjClassify}(oc)_i)$
`Relate`(in: Object, rc: RelConcept) $\rightarrow$ out: ObjectSet	$out_i := \sum_j (in_j \cdot \mathrm{RelClassify}(rc)_{j,i}))$
`Intersection`($in^{(1)}$: ObjectSet, $in^{(2)}$: ObjectSet) $\rightarrow$ out: ObjectSet	$out_i := \min(in_i^{(1)}, in_i^{(2)})$
`Union`($in^{(1)}$: ObjectSet, $in^{(2)}$: ObjectSet) $\rightarrow$ out: ObjectSet	$out_i := \max(in_i^{(1)}, in_i^{(2)})$
`Exist`(in: ObjectSet) $\rightarrow$ b: Bool	$b := \max_i in_i$
`Count`(in: ObjectSet) $\rightarrow$ i: Integer	$i := \sum_i in_i$