Lecture 11:

Imposing Task-Specific Structure on DNNs

Visual Computing Systems
Stanford CS348V, Winter 2018

Today

- Examples of DNN authors imposing structure on networks to better perform a desired task

- For each example, consider:
  - What knowledge does the human inject?
  - What does the computer learn?

- Tasks:
  - Image/video compression networks [Toderici 16, Tsai AAAI 18]
  - Visual Question Answering via Neural Module Networks [Johnson 17, 17]

Image compression using DNNs

Review: JPG image compression

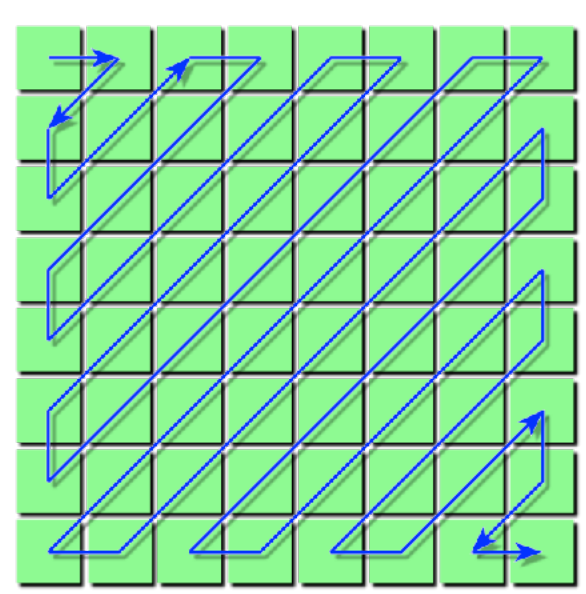
- Lossy compression designed to retain information that is most important to human perception
- Human-designed compact representation

$$
\begin{bmatrix}
-415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\
4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\
-47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\
-49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\
12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\
-8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\
-1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\
0 & 0 & -1 & -4 & -1 & 0 & 1 & 2
\end{bmatrix}
\;/\;
\begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99
\end{bmatrix}
$$

DCT **Quantization Matrix**

$$
= \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\
0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Quantized DCT



Coefficient reordering

→ RLE compression of zeros
Entropy compression of non-zeros → Compressed bits

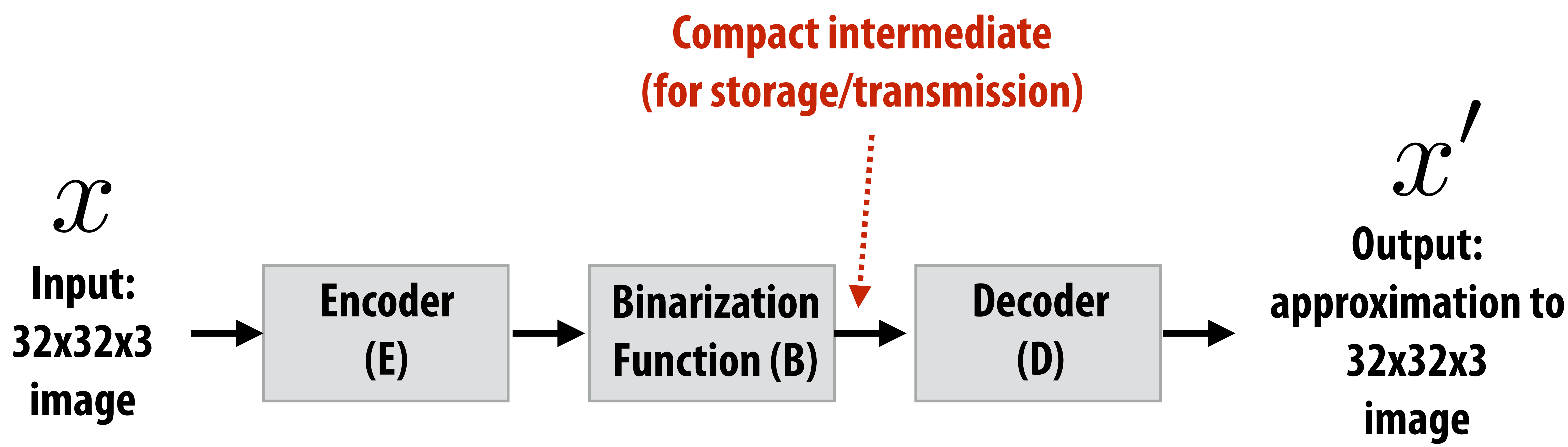
Deep learning learns useful representations

- **Can we apply deep learning techniques to obtain compact image representations for efficient storage and transmission?**

- **Discussion: why should we even consider this? (isn't JPG a pretty darn good representation?)**

Example: autoencoder

Compressing 32x32 8-bit RGB thumbnails (24 bpp)



$$x' = D(B(E(x)))$$

Auto-encoder: learn to compress (encode) and reconstruct (decode) the input signal

- **Jointly train D, B, and E using supervision from Loss(x, x')**

Progressive encoding: chain copies of autoencoder
(each iteration contributes additional bits to stored output)

Input: 32x32x3 residual r_{t-1} → **Encoder (E)** → **Binarization Function (B)** → **Decoder (D)** → **Output: 32x32x3** $F_t(r_{t-1})$

$$F_t(r_{t-1}) = D_t(B(E_t(r_{t-1})))$$

r_0 **= input image to compress**

Version 1:
each iteration predicts the residual

$$r_t = F_t(r_{t-1}) - r_{t-1}$$

$$x' = \sum_{t=1}^{N} F_t(r_{t-1})$$

Version 2: (stateful E() and D() units)
each iteration predicts input image

$$r_t = F_t(r_{t-1}) - r_0$$

$$x' = F_N(r_{N-1})$$

In both cases, loss given by $\|r_t\|_2^2$ **for all *t***

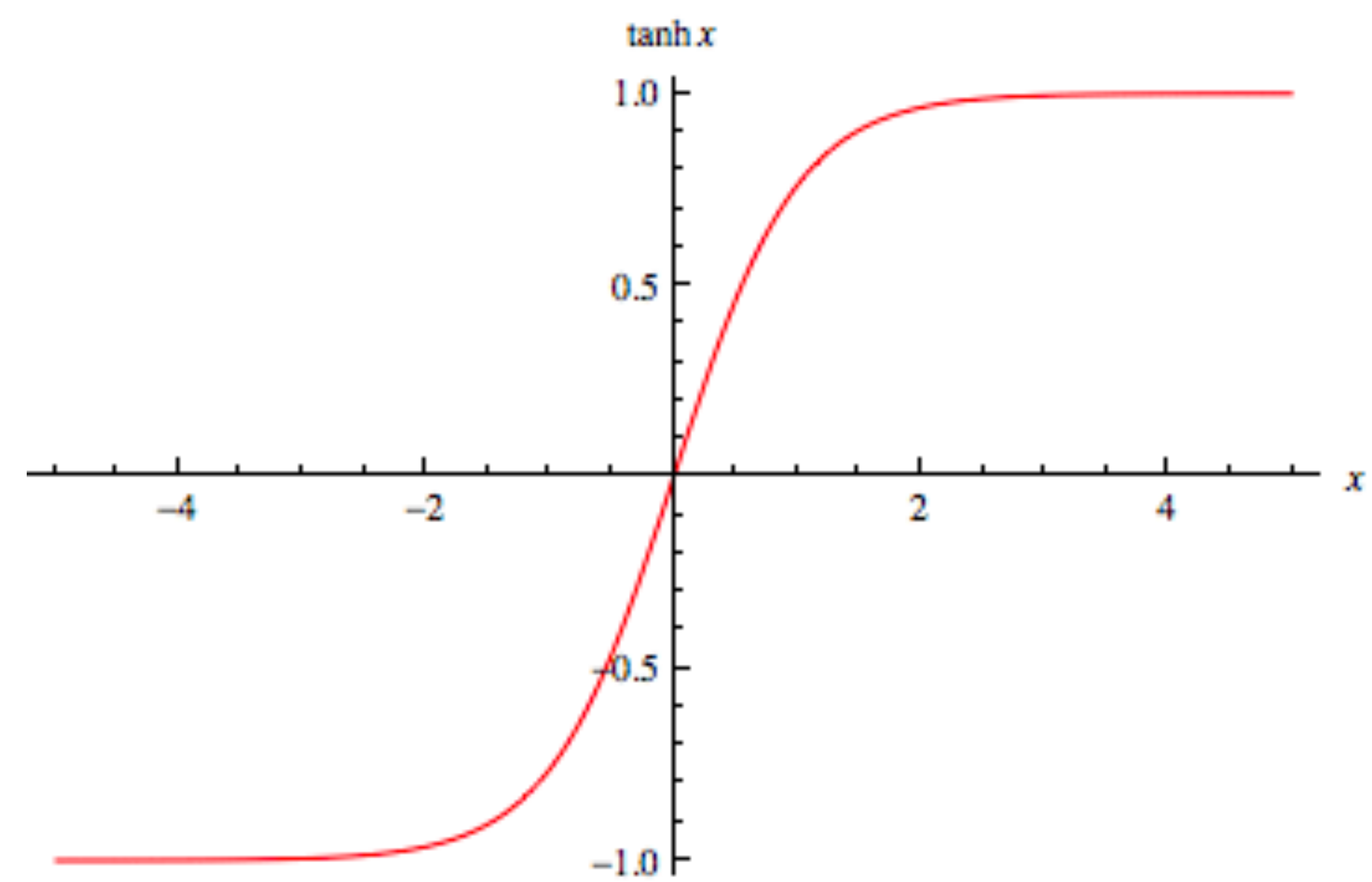
[Toderici ICLR 16]

Binarization

- **Step 1: output of encoder passes through fully-connected layer with *m* outputs (to "squeeze" to desired number of outputs)**
- **Step 2: quantize each output to a bit**

$$B(x) = f(tanh(Wx + b))$$

$$f(x) = \begin{Bmatrix} -1 & x < 0 \\ +1 & x \geq 0 \end{Bmatrix}$$



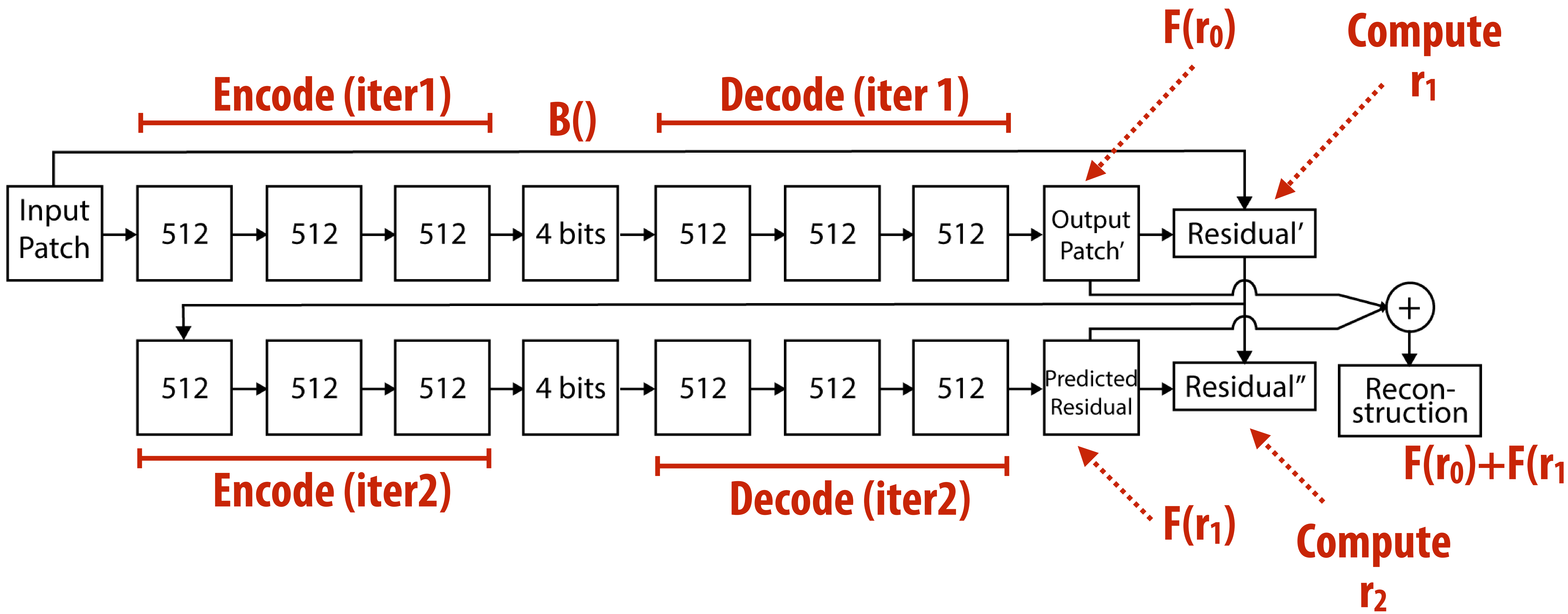
Add random perturbation during training (regularization):

$$f(x) = x + \epsilon$$

$$\epsilon \sim \begin{cases} 1 - x & \text{with probability } \frac{1+x}{2}, \\ -x - 1 & \text{with probability } \frac{1-x}{2}, \end{cases}$$

[Toderici ICLR 16]

Version 1 autoencoder



Fully-connected version:

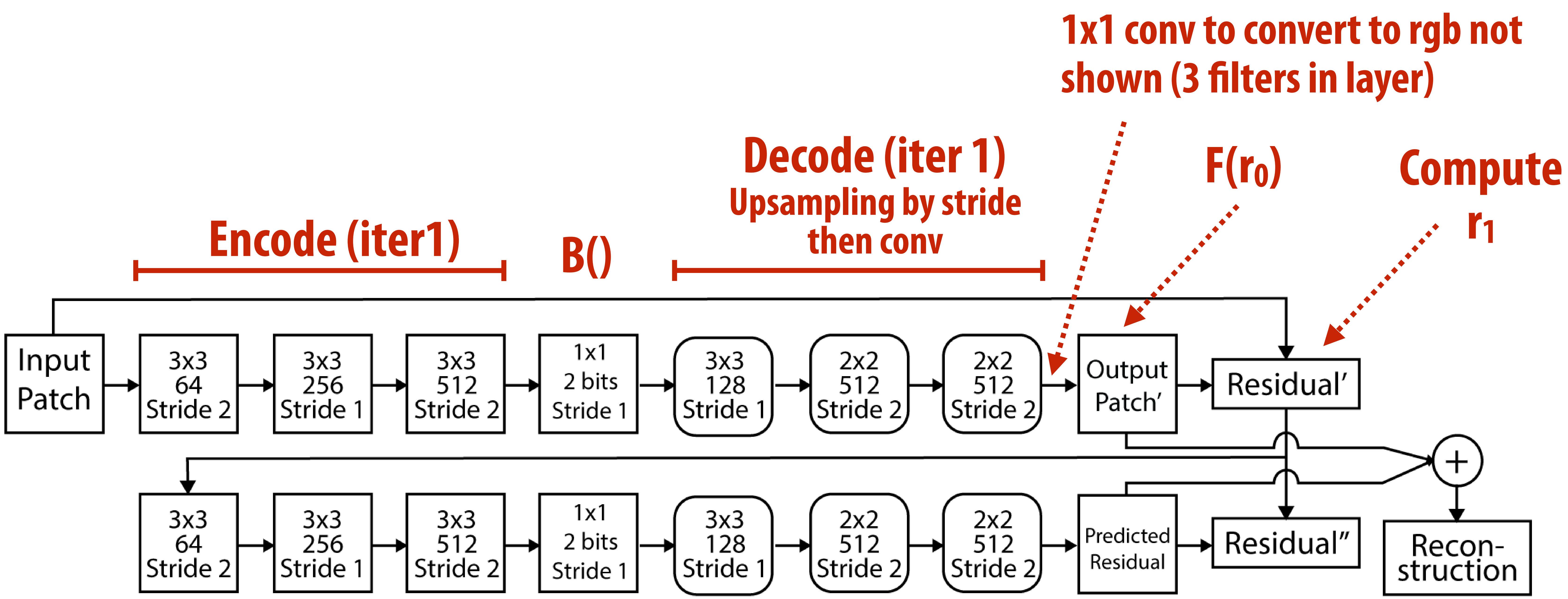
Input is 8x8 block

Each fully connected layer has 512 outputs and tanh non-linearity

Each iteration through auto encoder yields 4 bits (two iterations shown)

[Toderici ICLR 16]

Version 1 autoencoder (convolutional)



Convolutional version:

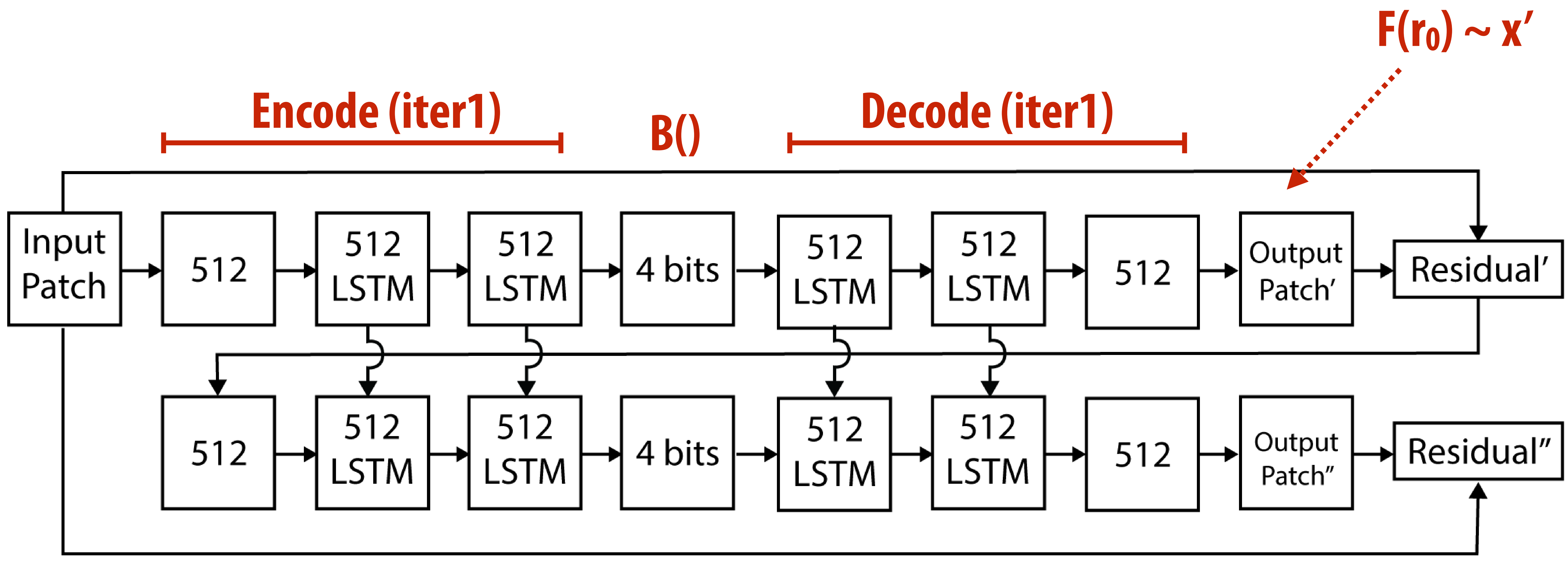
2 bits per spatial location of output per iteration

32x32 input → 8x8 spatial outputs (128 bits per iteration)

[Toderici ICLR 16]

Version 2 autoencoder (LSTM-based)

(Convolutional form of the LSTM auto encoder also exists)



LSTM version: predicts source image each iteration (not a residual)

LSTM units:

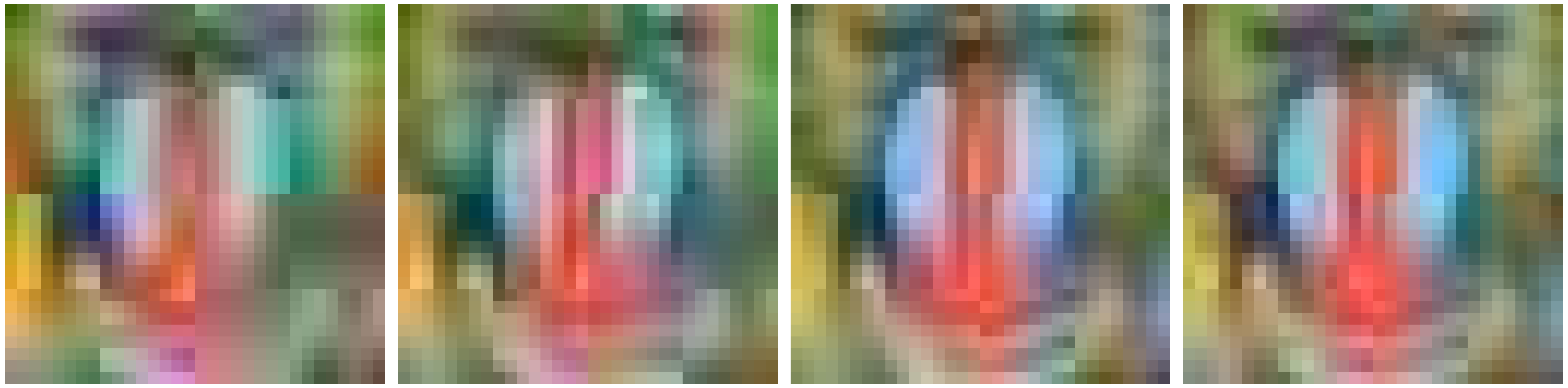
- **Recurrent: output from iteration t-1 fed into unit in iteration t**
- **Stateful: each unit maintains its own hidden state**

[Toderici ICLR 16]

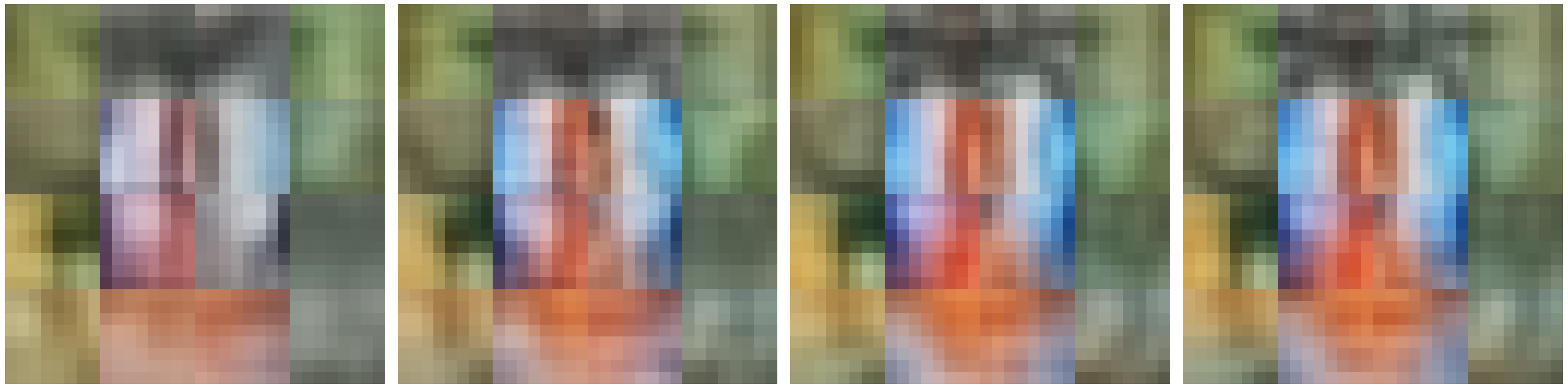
Compression results

[Toderici ICLR 2016]

Original (32×32)



JPEG compressed images



Compressed images with LSTM architecture

Compressed images with conv/deconv LSTM architecture

Average bpp:

	From left to right			
JPEG	0.641	0.875	1.117	1.375
LSTM	0.625	0.875	1.125	1.375
(De)Convolutional LSTM	0.625	0.875	1.125	1.375

Compression results

SSIM: structural similarity index (attempt to quantify perceived similarity of images)

	Patch Size	SSIM / 64B Target (Header-less Size)	SSIM / 128B Target (Header-less Size)
Header-less JPEG	8×8	0.70 (72.5 bytes avg.)	0.80 (133 bytes avg.)
Header-less JPEG 2000		0.66 (73 bytes avg.)	0.77 (156 bytes avg.)
Header-less WebP		0.62 (80.7 bytes avg.)	0.73 (128.2 bytes avg.)
Fully Connected Residual Encoder (Shared Weights)	8×8	0.46	0.48
Fully Connected Residual Encoder (Distinct Weights)	8×8	0.65	0.75
LSTM Compressor	**8×8**	**0.69**	**0.81**
Conv/Deconv Residual Encoder (Shared Weights)	32×32	0.45	0.46
Conv/Deconv Residual Encoder (Distinct Weights)	32×32	0.65	0.75
Convolutional/Deconvolutional Autoencoder	32×32	0.76	0.86
Conv/Deconv LSTM Compressor	**32×32**	**0.77**	**0.87**

[Toderici ICLR 16]

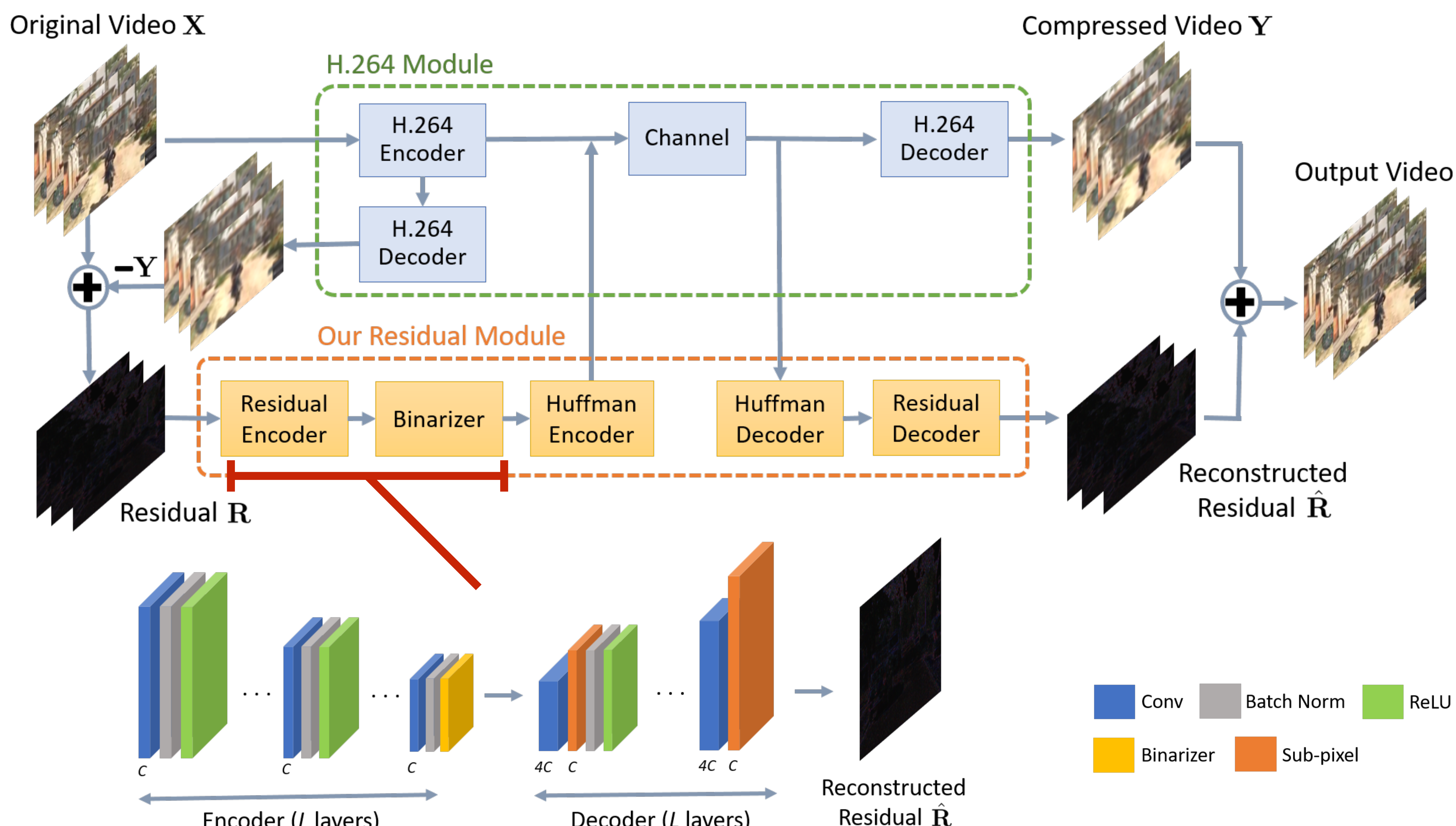
Summary / thoughts

- Idea: learn how to compress thumbnail-sized images by trying to compress large database of tiny images
  - Loss is not perceptually motivated (if there was a differentiable perceptual loss metric, they would have used it instead of L2 on pixel residual)
- Improvement on JPG for small images, future work extends to large images by exploiting global redundancy [Toderici 2016]
- Why use learning for this problem?
  - Potential for higher quality encode (learn better representations than humans can manually craft)
  - General mechanism to specialize representations for task
    - [Toderici 2016]: specific to thumbnail images
    - What about camera-viewpoint specific compression?
    - Task-based definition of loss rather than pixels (compress subject to still being able to recognize objects)

Video compression

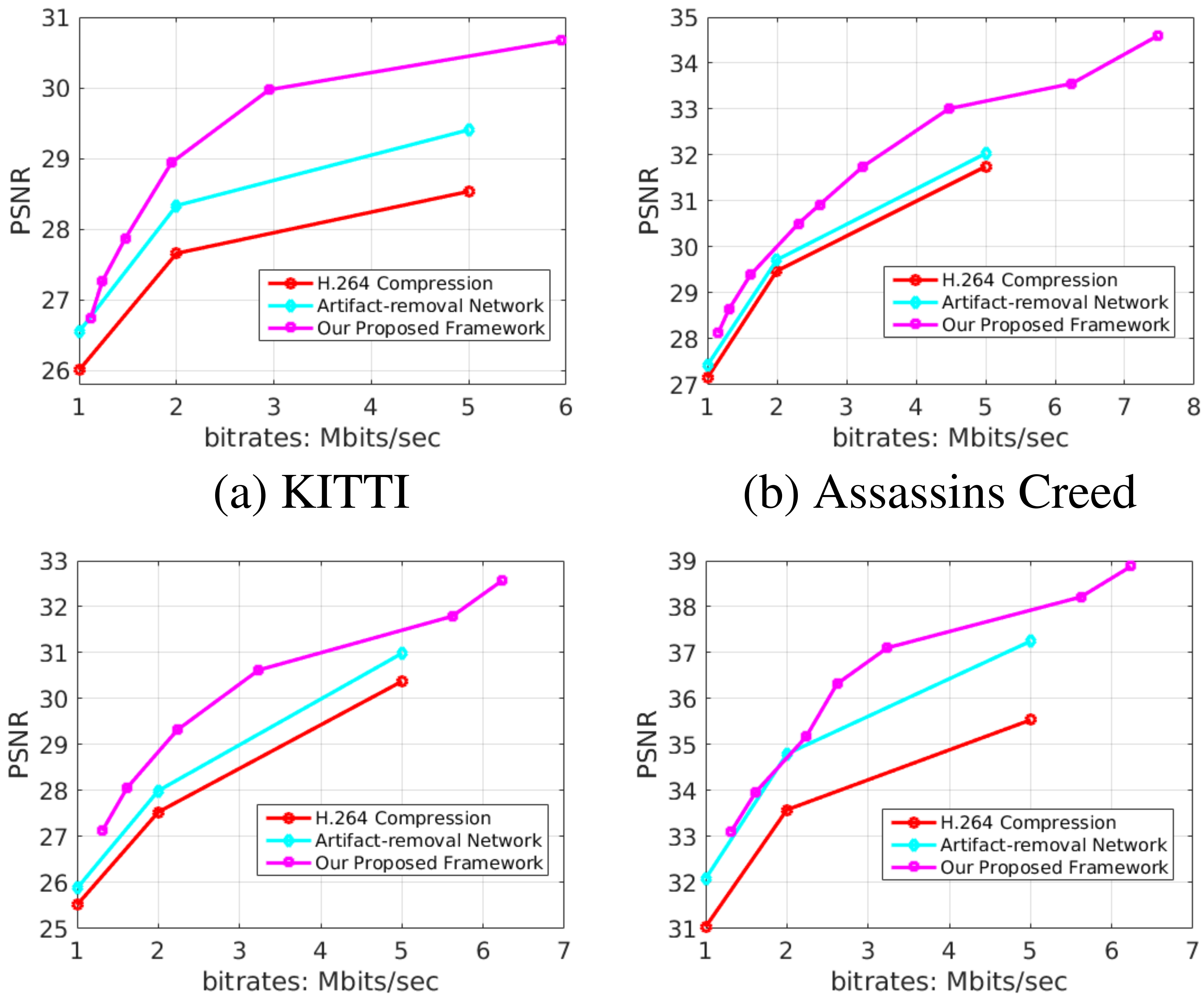
[Liu 2018]

- Idea: use deep network to compress only the residual of H.264 compressed video stream (recall: residual = loss due to H.264 compression)



Takeaway:

- **Can learn to compress residuals effectively when compressor tailored to specific type of video stream**



(a) KITTI (b) Assassins Creed

(c) Skyrim (d) Borderlands

[Liu 2018]

Camera-specific compression?

Security cameras (stationary)

Head mounted cameras
(person only sees so many things)

On-vehicle cameras
(specialization to neighborhood or city?)

Discussion:
Neural Module Networks

Visual question answering

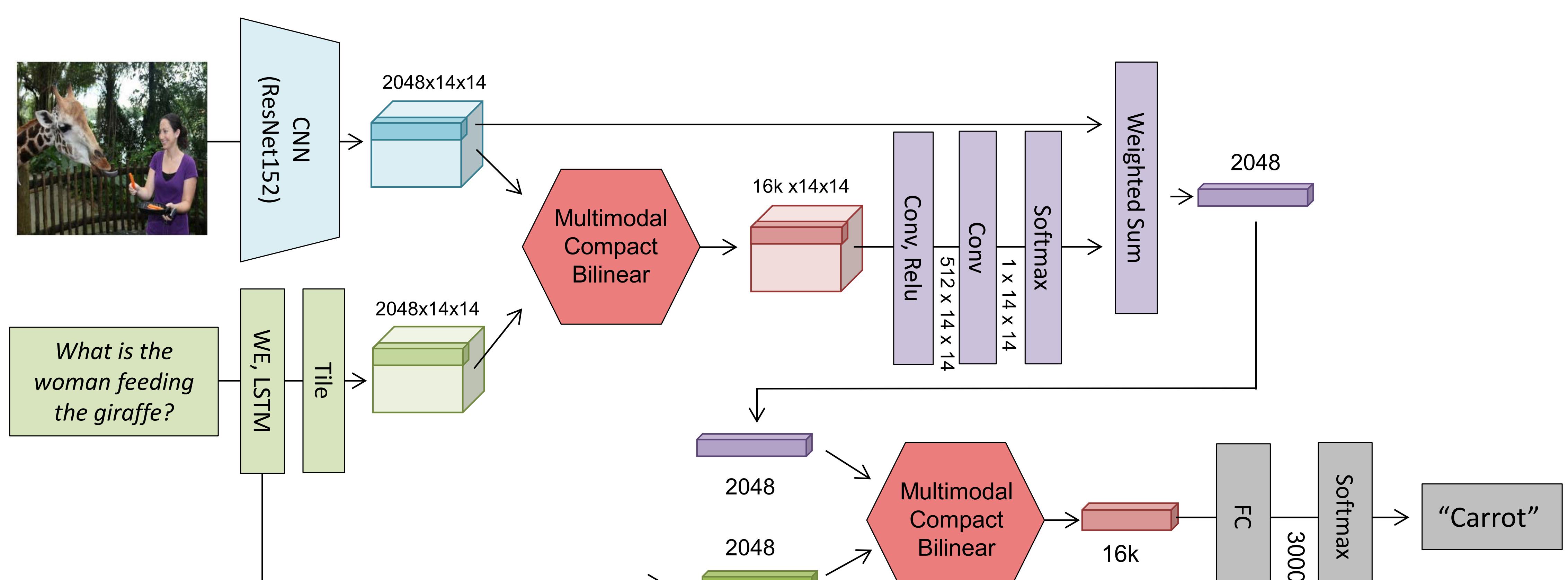
How many children are in the bed?

2 1

Does it appear to be rainy?
Does this person have 20/20 vision?

How many slices of pizza are there?
Is this a vegetarian pizza?

- **Common approach at the time:**
  - **DNN (LSTM) to compute embedding of the question**
  - **Use the question to "attend" to the relevant part of the image**
  - **Mix visual features in attended region and question embedding to answer question**



[Fukui 2016]

Example challenge: debugging failures

Clear

what is the color of the shirt on the man

red (0.24)
blue (0.19)
orange (0.18)
gray (0.14)
white (0.09)

Took 0.049 sec

Clear

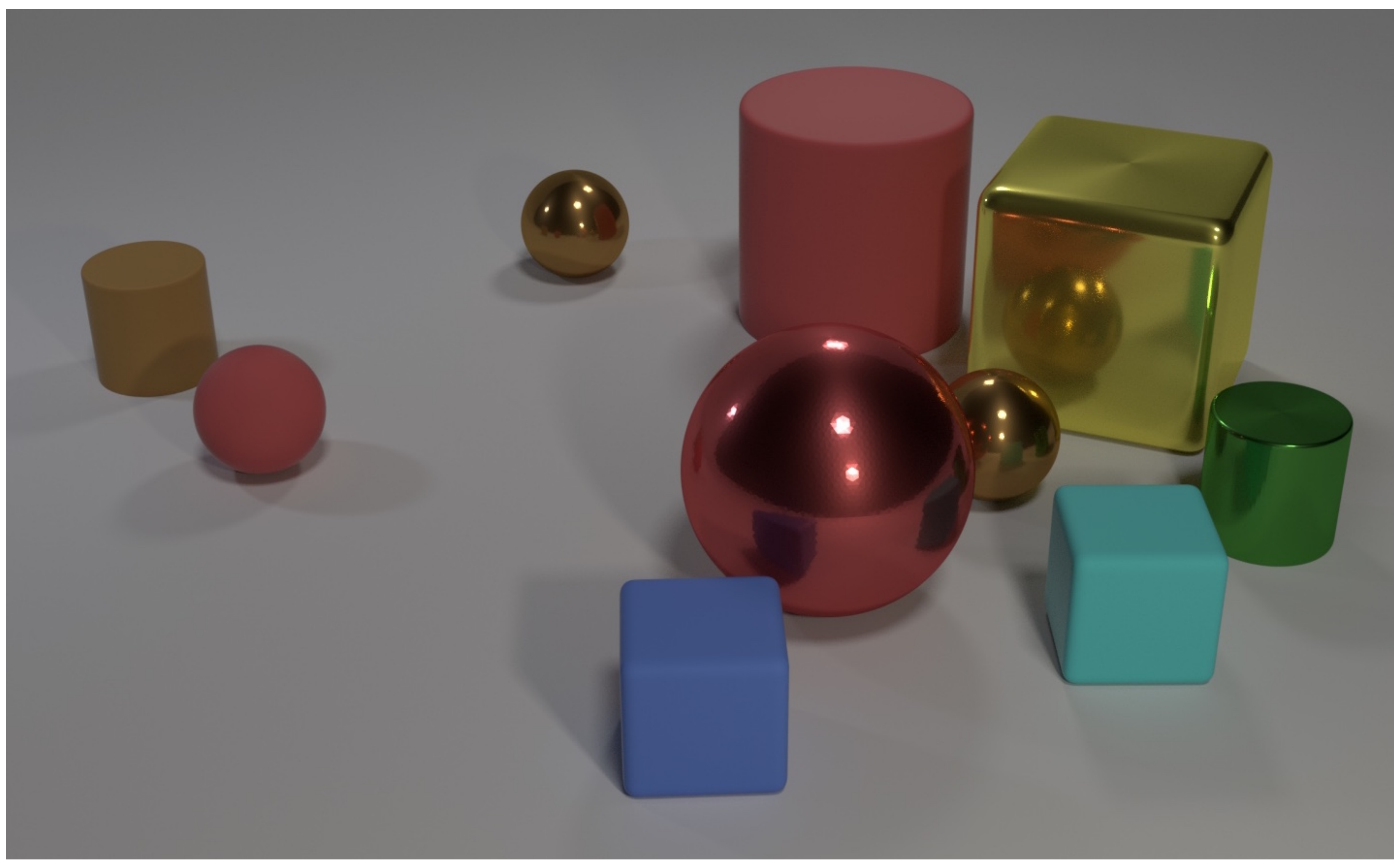
what is the color of the shirt on the person on the left

red (0.39)
orange (0.30)
pink (0.15)
blue (0.07)
white (0.03)

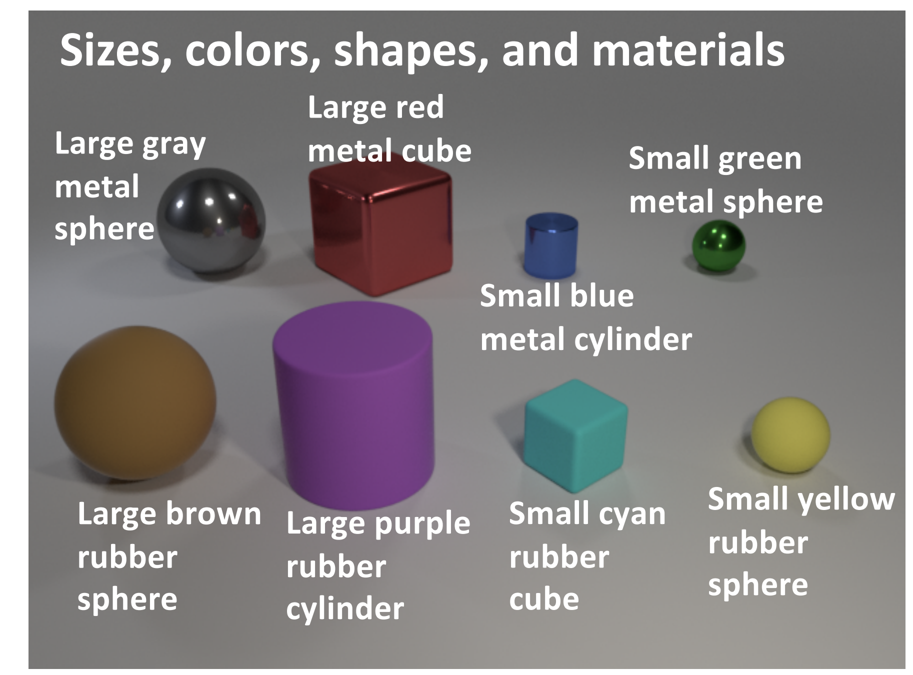
Took 0.051 sec

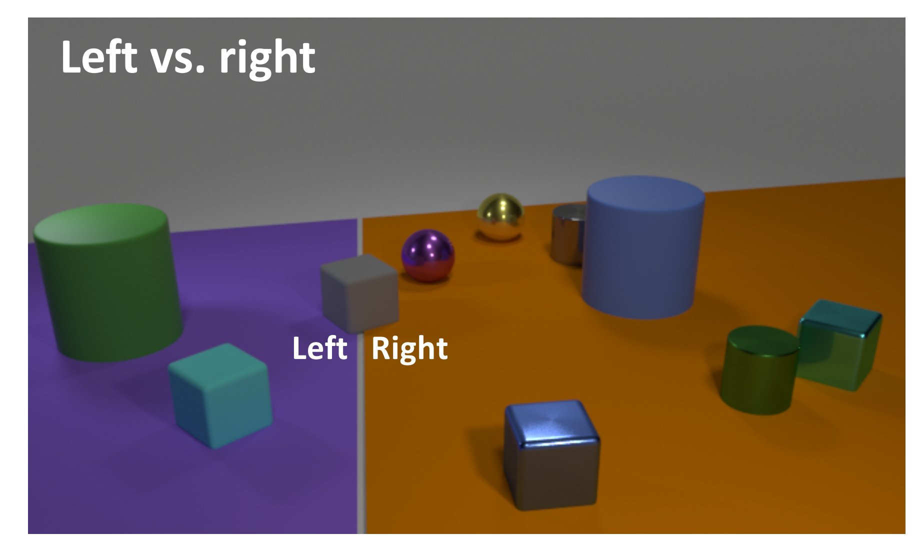
CLEVR dataset

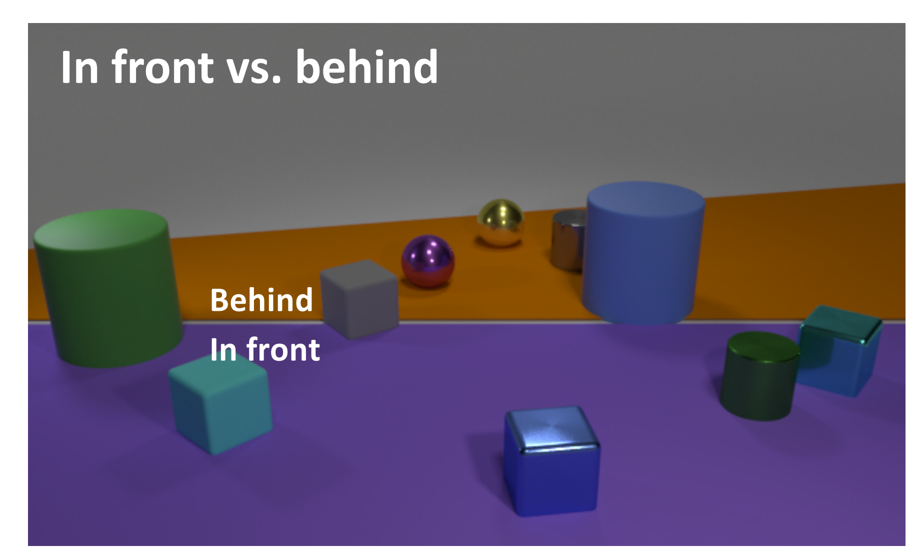
- **Main idea: images and questions can be answered from contents of images alone (no external knowledge)**



Q: Are there an equal number of large things and metal spheres? **Q:** What size is the cylinder that is left of the brown metal thing that is left of the big sphere? **Q:** There is a sphere with the same size as the metal cube; is it made of the same material as the small red sphere? **Q:** How many objects are either small cylinders or metal things?







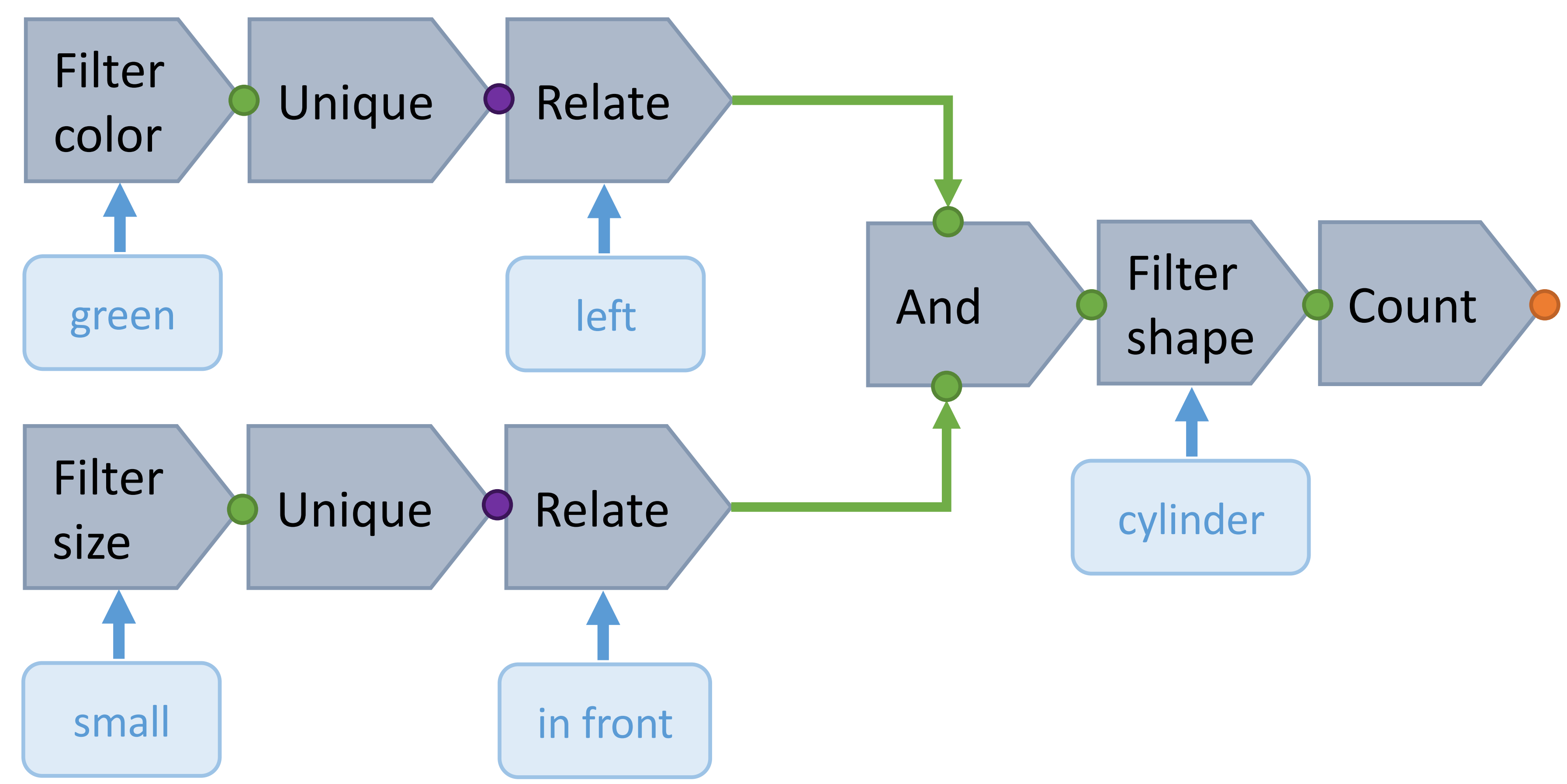
[Johnson 2017]

Second major idea: neural module networks

- **If a question is just a logical expression, why isn't the program used to answer it dependent on the structure of a question?**

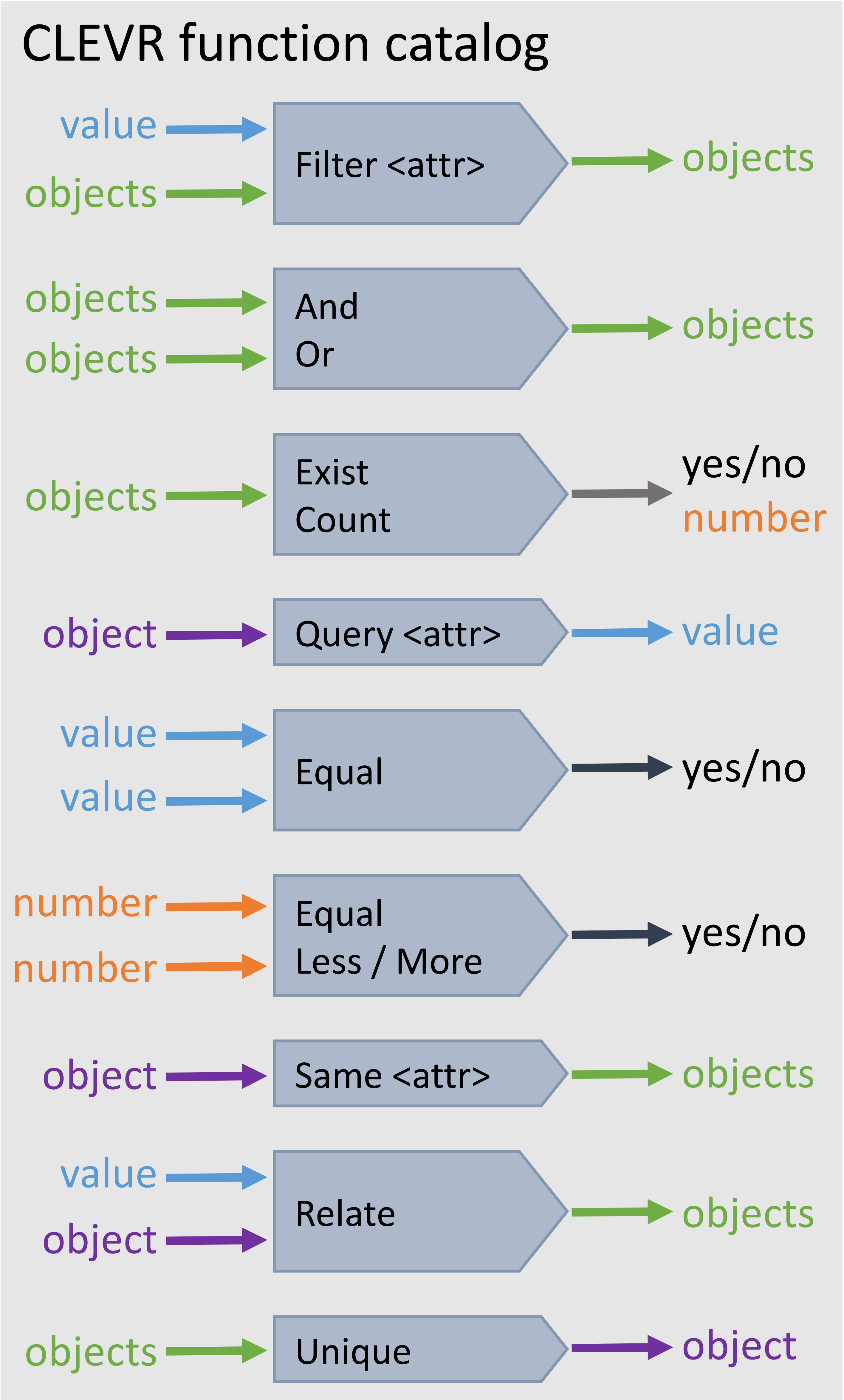
Question text:
How many cylinders are in front of the small thing and on the left side of the green object?

NLP semantic parser
question to "program"
(program = neural module network)



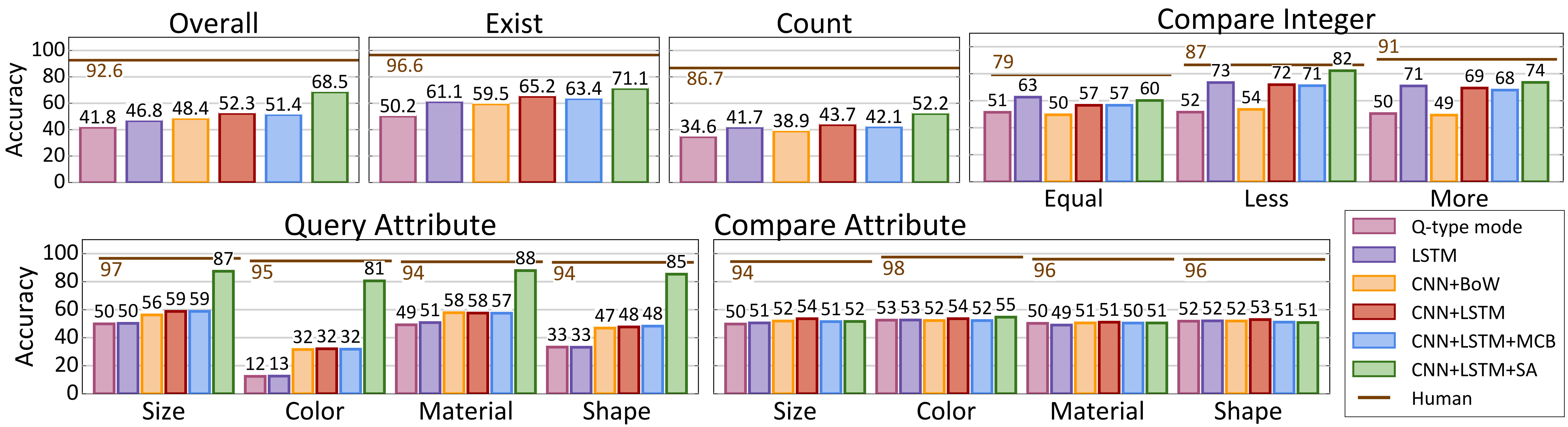
DNN Topology specific to question:
(Unique DNN per question)

Grammar of CLEVR questions



[Johnson 2017]

State-of-the-art at time of CLEVR



Accuracy of state-of-art VQA techniques on CLEVR dataset

Hand-engineering CLEVR

- **Simplification:**
  - Assume question text to program parse is perfect
  - Just answer question based on program+image

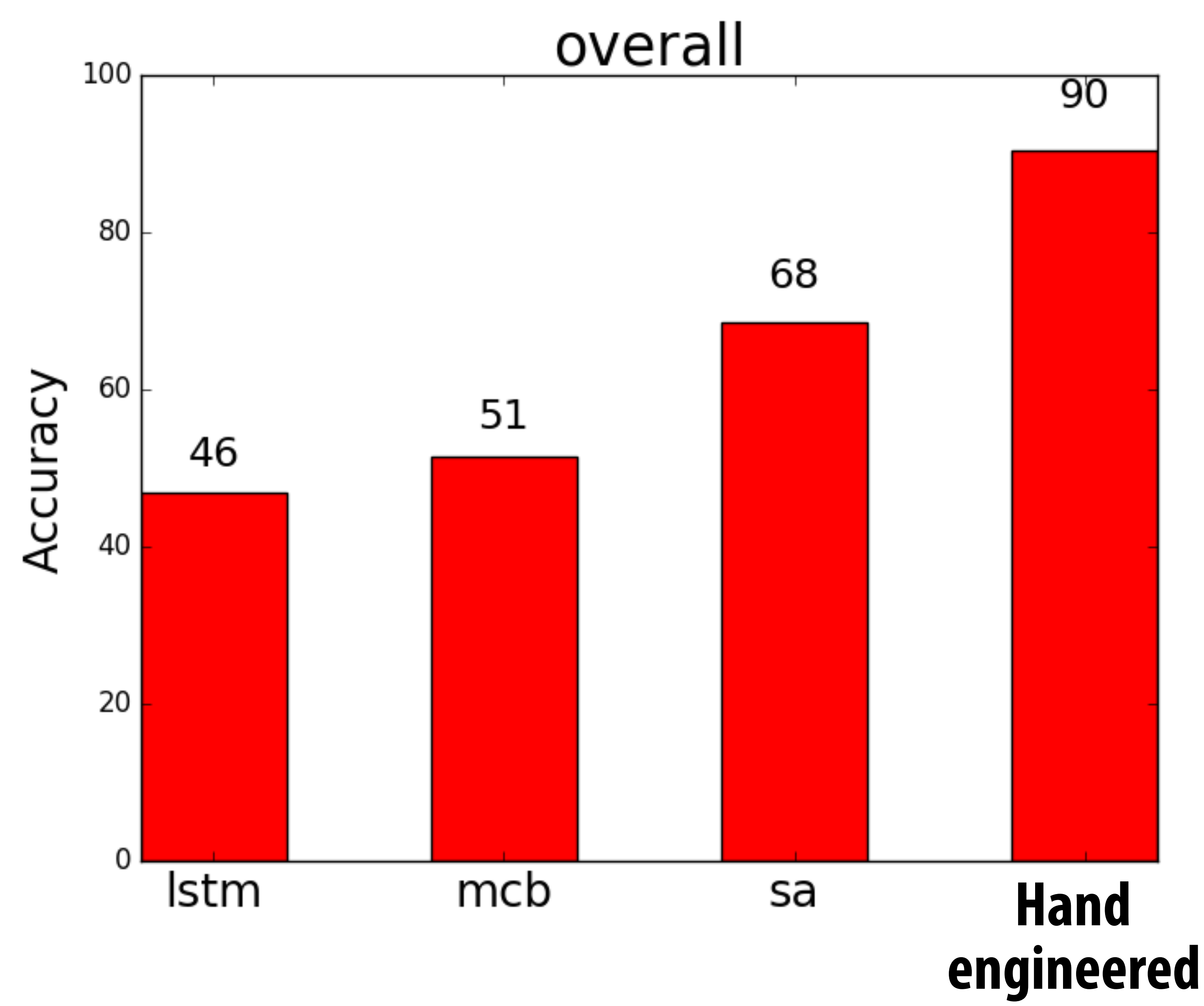
- **Learned modules (manually annotated intermediate results based on data in CLEVR scene graph description)**
  - Detection: YOLO trained on CLEVR objects
  - Color: SVM trained on VGG conv5 features
  - Material: SVM trained on VGG conv5 features
  - Size: SVM trained on bounding box coordinates
  - Depth: LR trained on bounding box coordinates

- **Not learned modules (implemented by hand)**
  - Counting
  - Integer comparison
  - Attribute comparison
  - Spatial relations (left, right, behind, in front)
  - Logical operators (and, or)

[Slide credit: Karima Ma]

Hand-engineered solution

Accuracy	Detect	Shape	Color	Material	Size	Depth	
96.39							
91.8							
94.57							
95.59							
95.72							
96.26							
85.77							*no qtype
90.02							*yes qtype



[Slide credit: Karima Ma]

Testing on real world images*

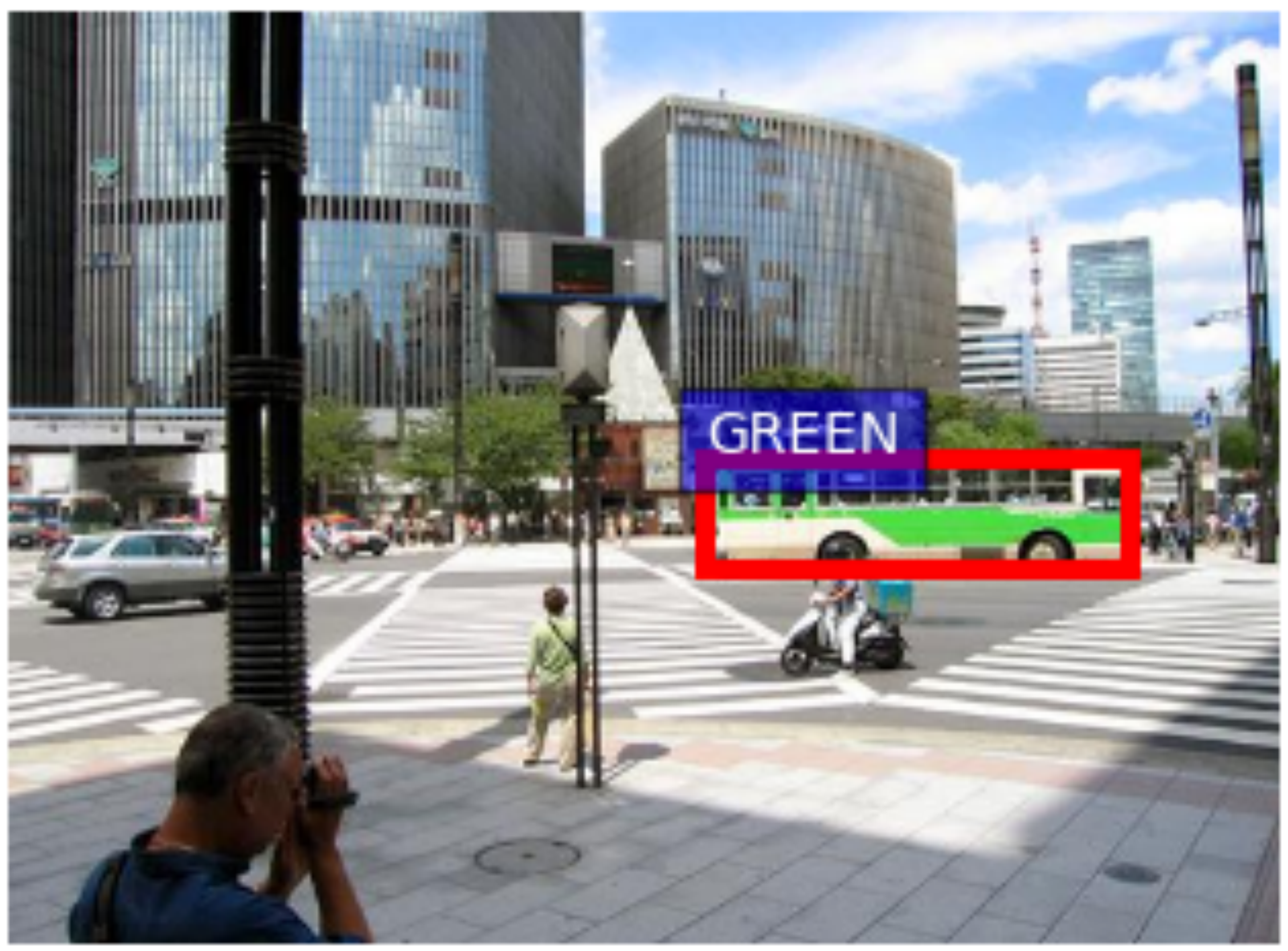
(* Text question rewritten as expression manually. Focus is not NLP.)

Q. "What color is the bowl"?

A. orange

Q. "What color is the bus?"

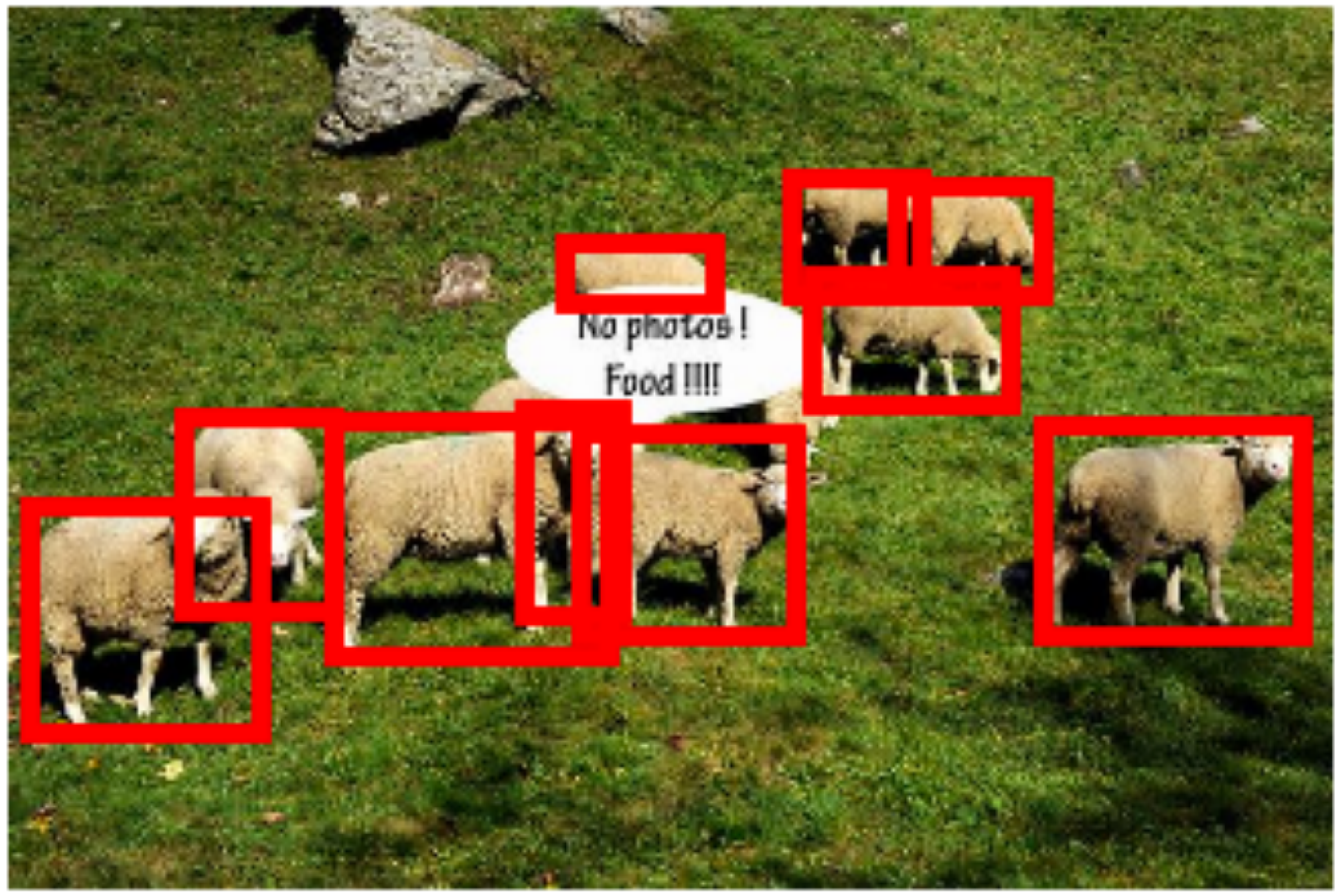
A. green



Testing on real world images

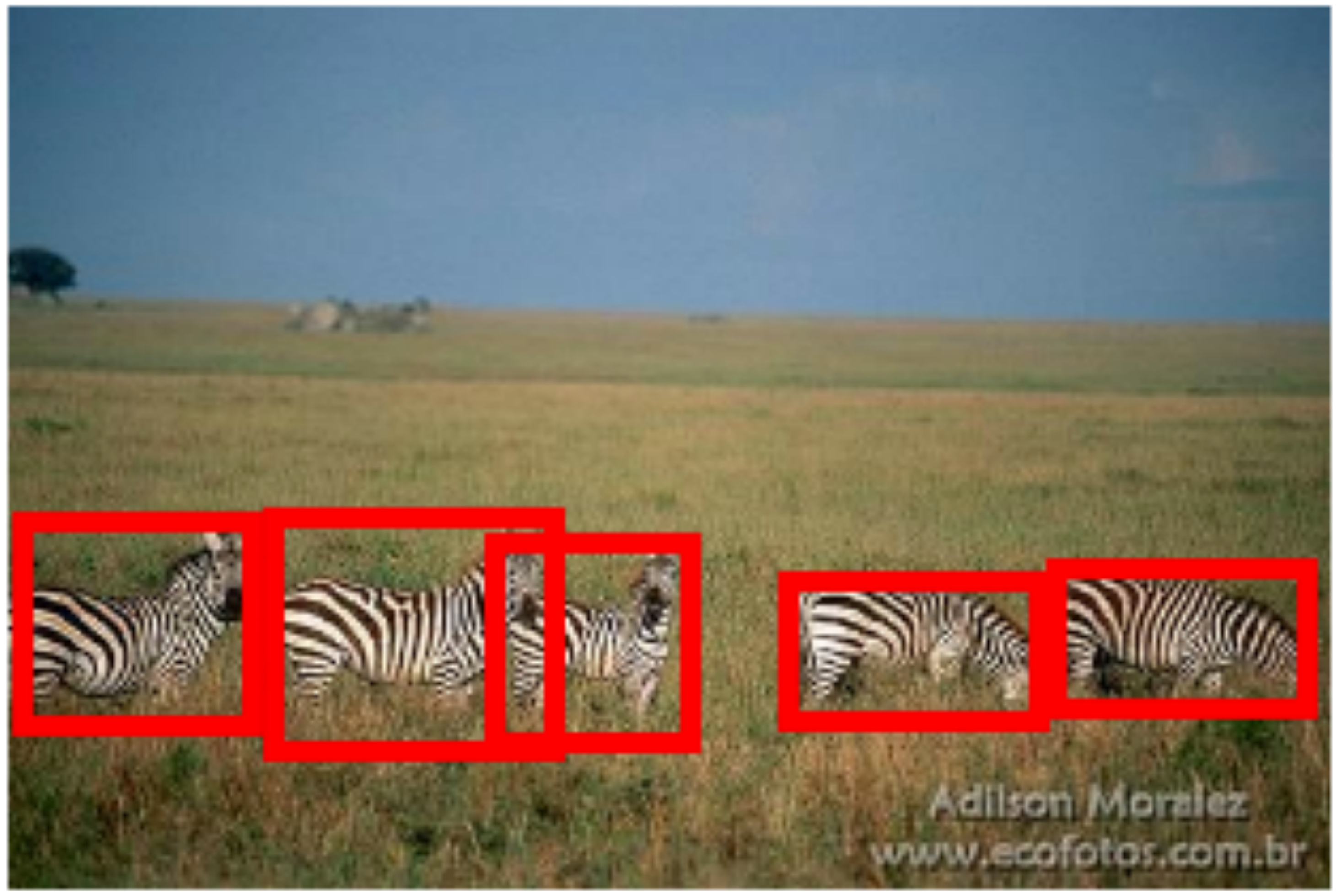
Q. "How many sheep are in the photo?"

A. 10



Q. "How many zebras are there?"

A. 5



[Slide credit: Karima Ma]

Testing on real world images

Q. "What is on the bench?"

A. airplane

detect(bench)

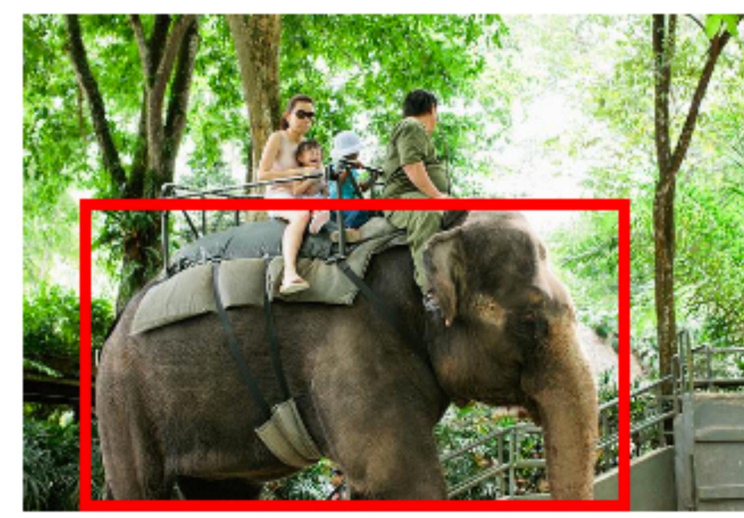
detectany()

on(bench, any)

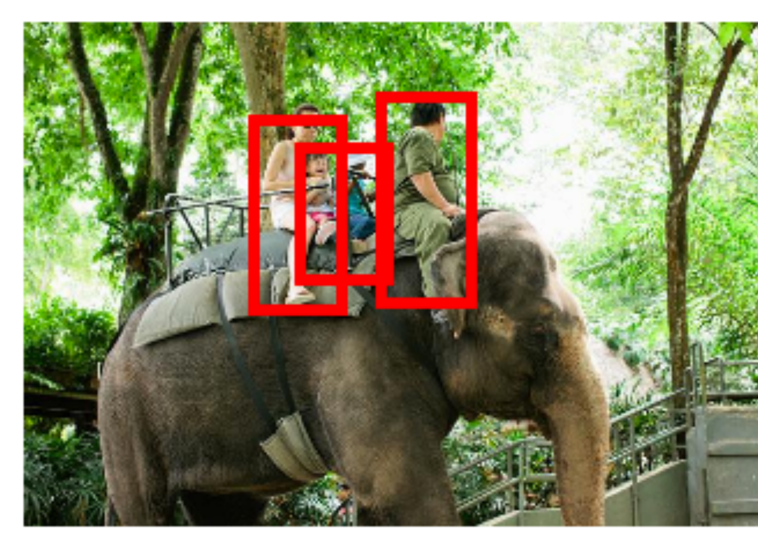
what()

Q. "How many people are on the elephant?"

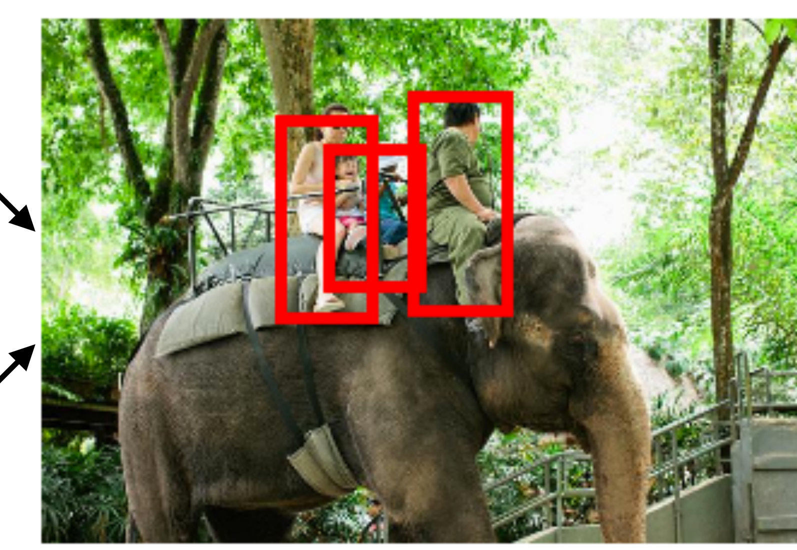
A. 3



detect(elephant)



detect(person)



on(elephant, person)

count()

[Slide credit: Karima Ma]

Back to CLEVR…

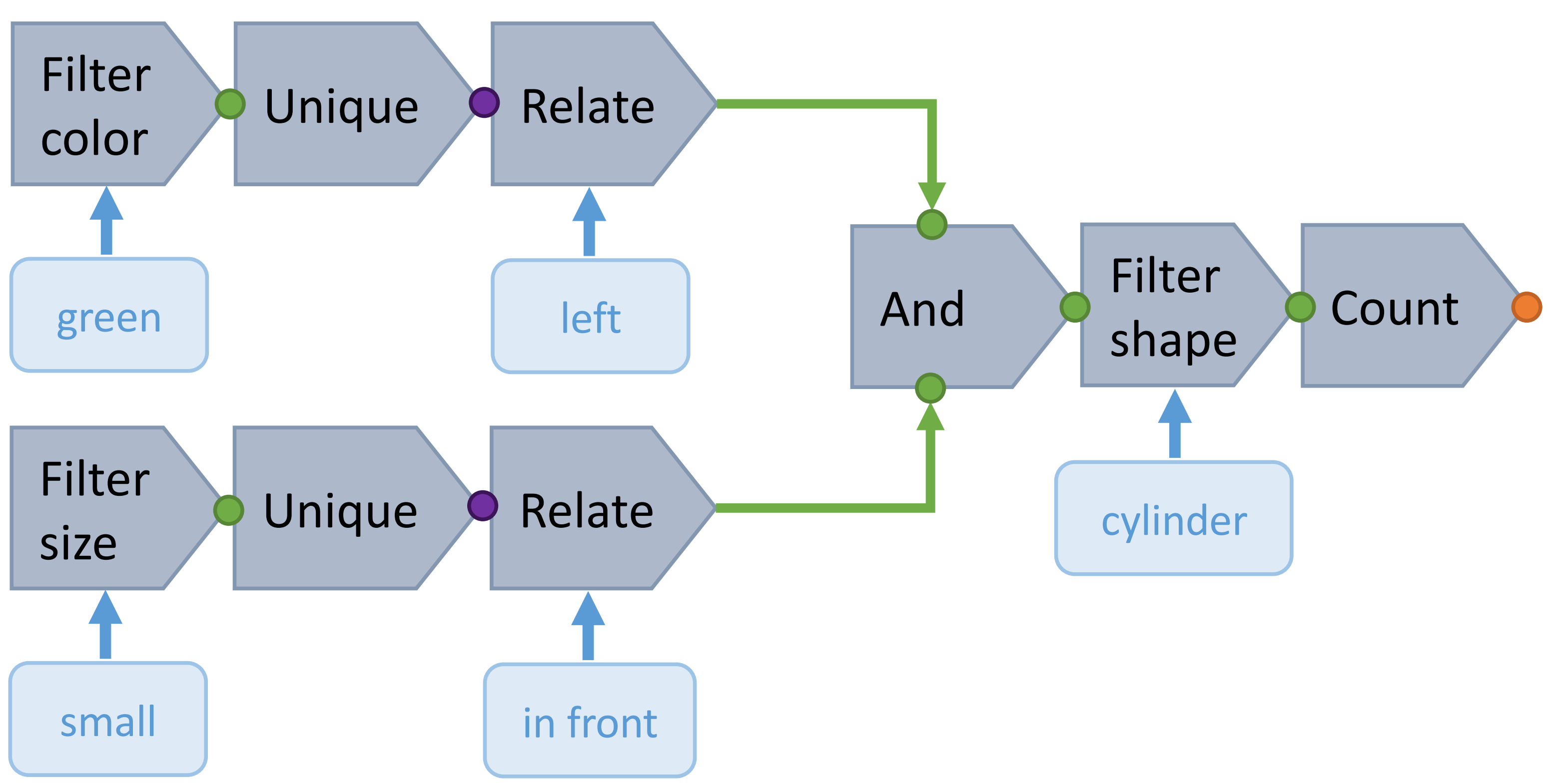
- Challenge 1: convert ambiguous English text to valid program

- Challenge 2: learning behavior of individual modules
  - CLEVR provides module network definitions for a question, as well as the end-to-end answer to the question, but not a specification of how individual modules should behave!

Solving CLEVR

- **Challenge 1: convert ambiguous English text to valid program**
  - [Johnson 2017] trains program generator via REINFORCE <see paper>
  - Or just assume ground truth program known for today's discussion

- **Challenge 2: learning behavior of individual modules**
  - Module definition: WxHxC input —> WxHxC output
  - One ResNet block with 3x3 convs
  - Binary modules concat inputs WxHx2C
  - Compositions of programs as valid DNNs



"How many cylinders are in front of the small thing and on the left side of the green object?"

DNN Topology specific to question:
(Unique DNN per question)

Discussion: end-to-end learning solution

- Construct neural module network
- Evaluate network to compute answer
- Backpropagate loss, updating weights of modules used in neural module network
  - Classification loss since final layer is a classifier over all possible value answers

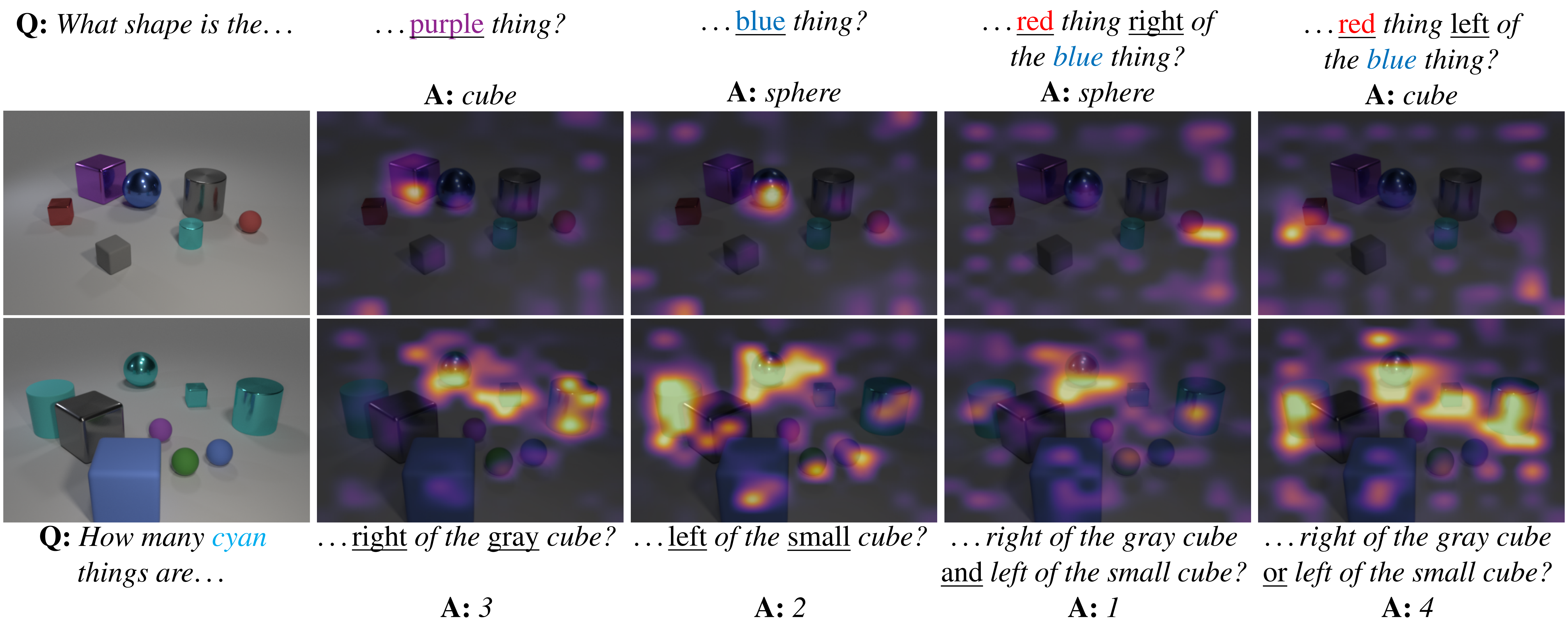
- What has happened:
  - Learning system only knows what module to use when a module of type T is needed
  - Learning system has no definition for each of the module's behavior/semantics
  - By using the module X in many tasks whenever "FilterColor(x)" is needed, module X ultimately should learn to perform a filtering by color operation
  - But there is no guarantee of what each module learns!

First, results…

Method	Exist	Count	Compare Integer Equal	Compare Integer Less	Compare Integer More	Query Size	Query Color	Query Mat.	Query Shape	Compare Size	Compare Color	Compare Mat.	Compare Shape	Overall
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

- **Better than human performance!**
- **Better performance than manually designing modules to perform subtasks!**
  - **Discussion: What does this imply?**

Attempting to introspect network behavior



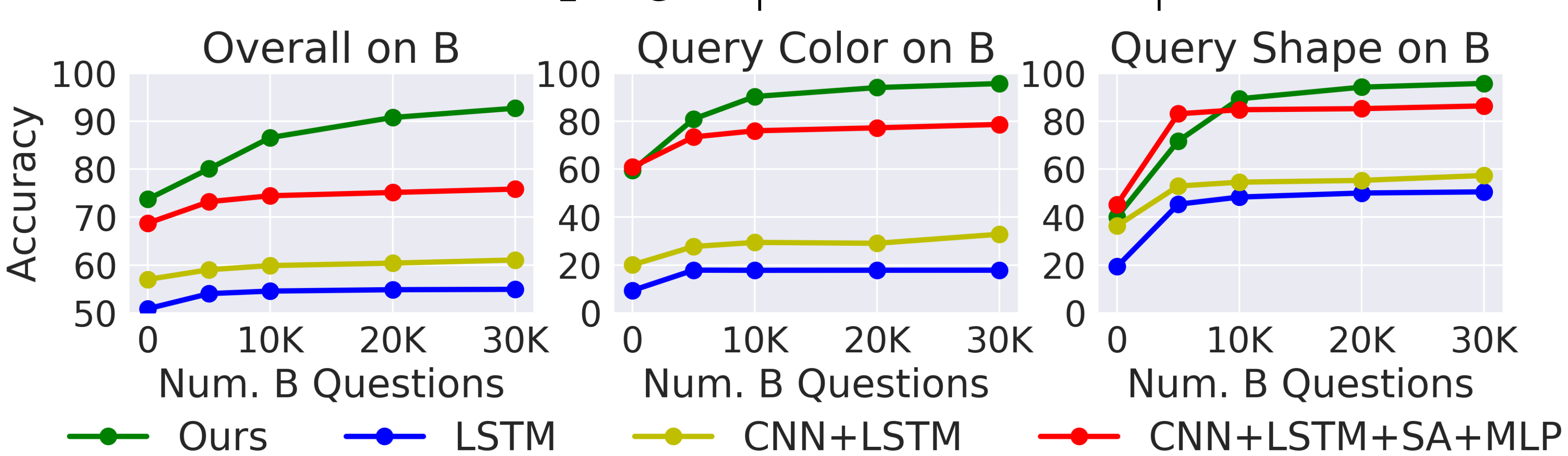
Visualizing loss gradient w.r.t. last WxHxC feature map output by DNN

Location of most salient part of image is intuitive.
(this part of image most impacts loss)

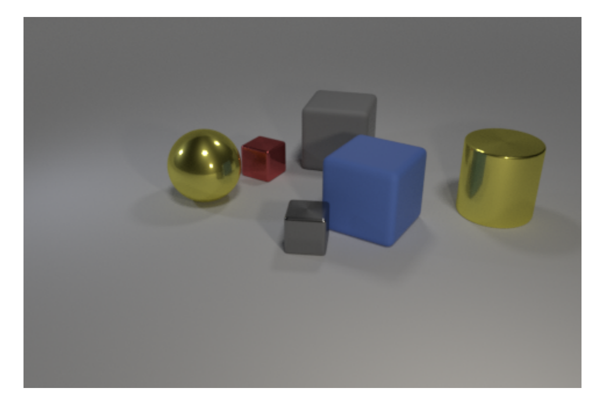
Do modules generalize beyond CLEVR?

- Experiment 1: train only configuration A
- Configuration A
  - all cubes are gray, brown, blue, yellow
  - all cylinders are red
- Configuration B (flip shape/color association)
  - all cubes are red
  - all cylinders are gray, brown, blue, yellow

Method	Train A A	Train A B	Finetune B A	Finetune B 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**



- Experiment 2: have humans write new questions
  - Underlined words are words not in CLEVR dataset
- Fine tune program generator (end-to-end based on final answers)

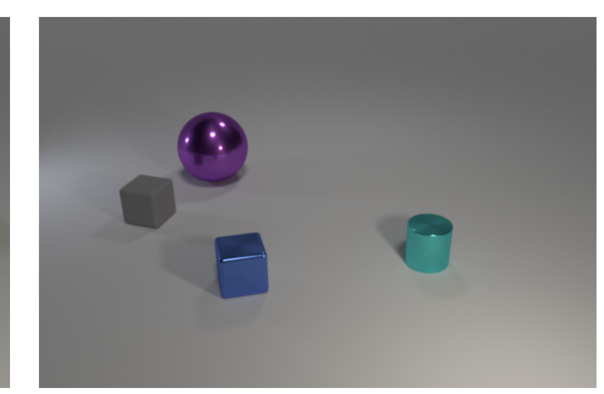


Q: *Is there a blue <u>box</u> in the <u>items</u>?* **A:** *yes*

Predicted Program:

exist
filter_shape[cube]
filter_color[blue]
scene

Predicted Answer:
✓ *yes*

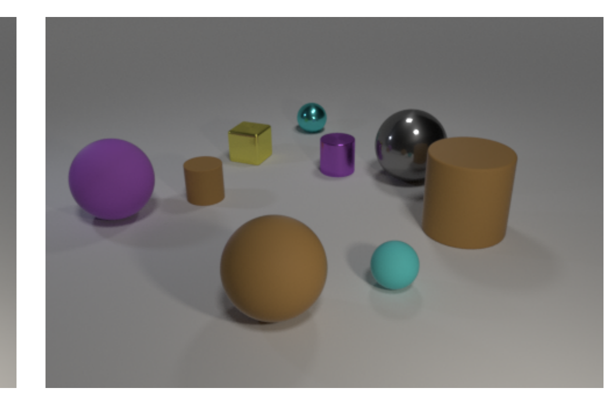


Q: *What shape object is <u>farthest</u> right?* **A:** *cylinder*

Predicted Program:

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

Predicted Answer:
✓ *cylinder*



Q: *Are <u>all</u> 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*

Discussion questions

- Would you agree that training modules for CLEVR is a form of multi-task learning? (each question is a unique task)

- What were the two major benefits of end-to-end training approach?

- What are your thoughts on when learning should be used?
  - Interesting question: would left_of() module work on real-world images?

- Given the results in the paper how would you describe the semantics of a module? (e.g., the module that is supposed to filter_color(red))