Lecture 12:

Imposing Task-Specific Structure on DNNs

Visual Computing Systems
CMU 15-769, Fall 2016

Today

- Four 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?

- Image compression networks [Toderici ICLR 16]
- Cross-stitch networks: sharing lower DNN layers [Misra CVPR 2016]
- Spatial transformers [Jaderberg NIPS 2015]
- Convolutional pose machines for pose estimation [Wei CVPR16]

Image compression using DNNs

Review: JPG image compression

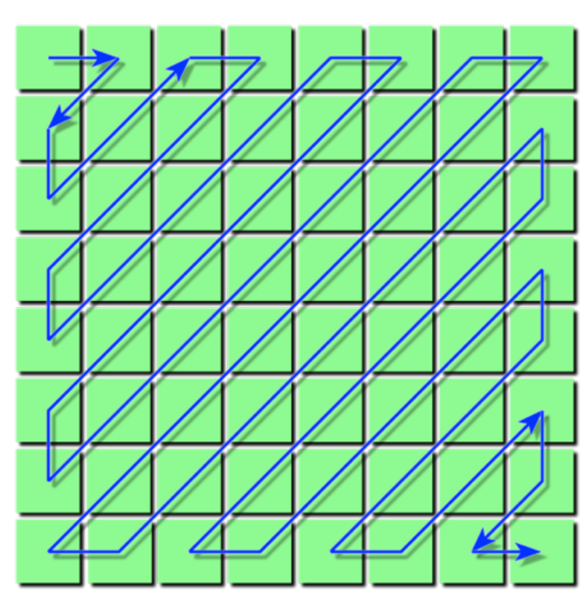
- **Lossy compression designed to retain information that is most important to human perception**
- **Human-designed compact representation**

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

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**

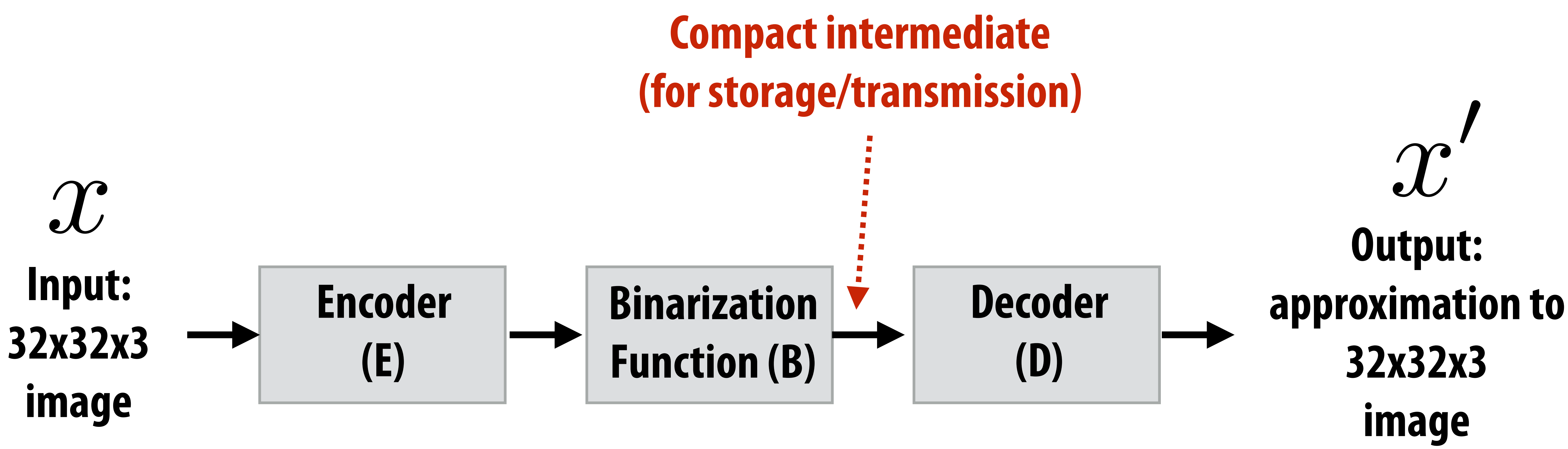
Deep learning learns useful representations

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

- **Class discussion: Why?**

Example use case

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

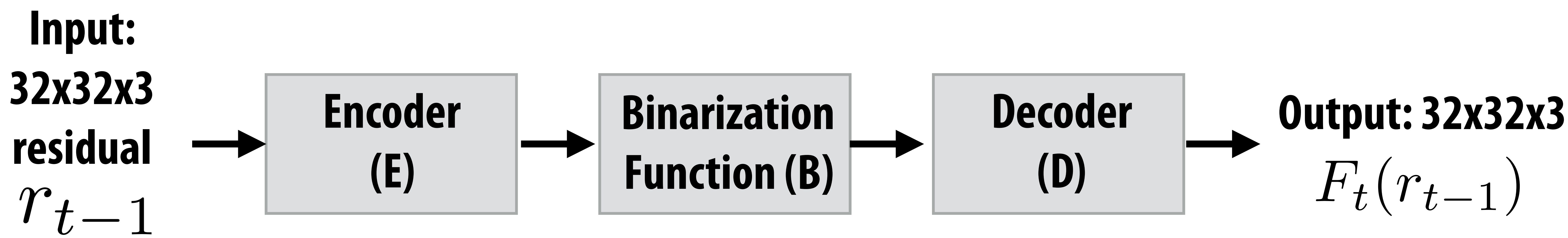


$$x' = D(B(E(x)))$$

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 bits)



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

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**

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

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

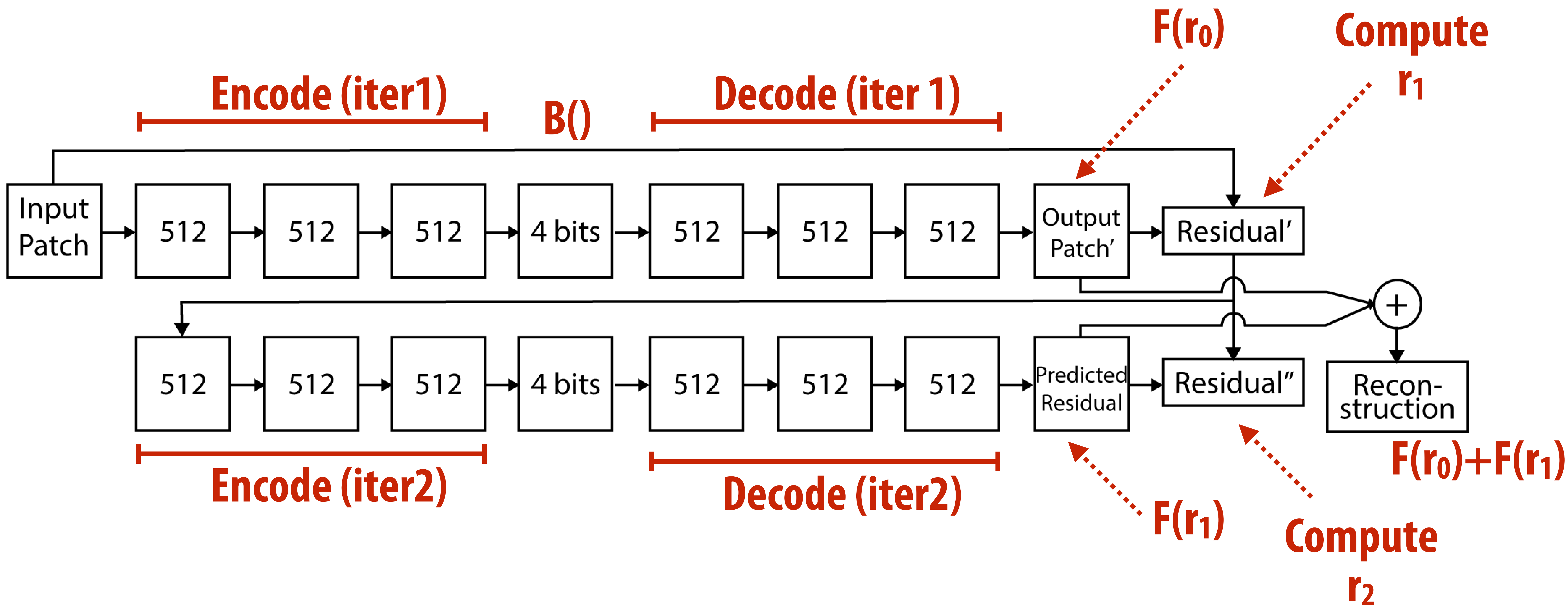
Add random perturbation during training (regularization):

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

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

[Toderici ICLR 16]

Version 1 autoencoder



Fully-connected version:

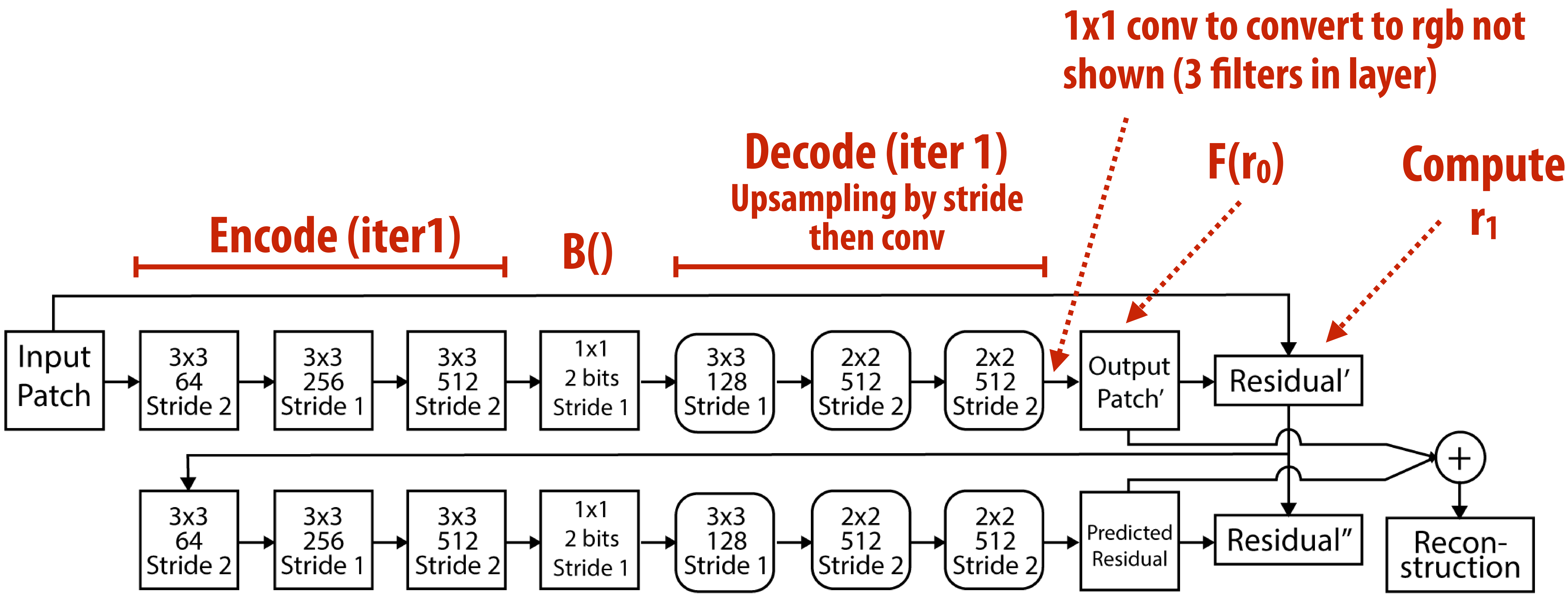
Input is 8x8 block

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:

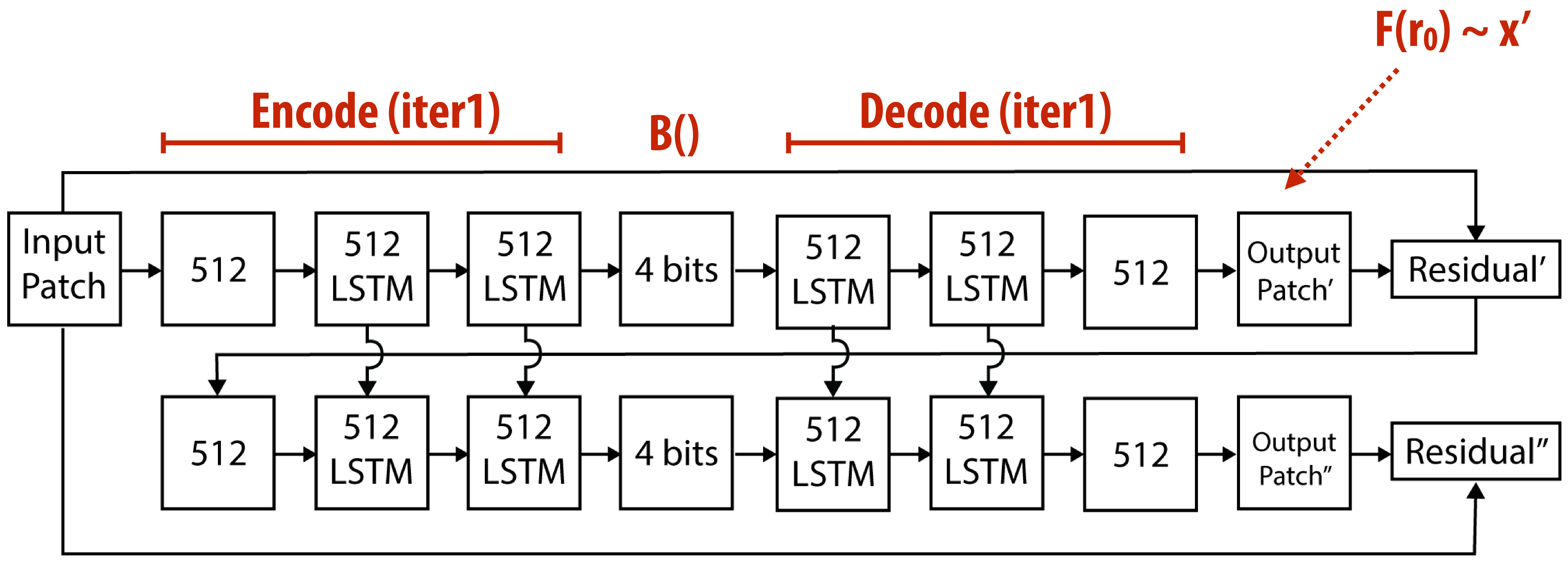
2 bits per spatial location of output per iteration

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

[Toderici ICLR 16]

Version 2 autoencoder (LSTM-based)

(Convolutional form also exists)



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

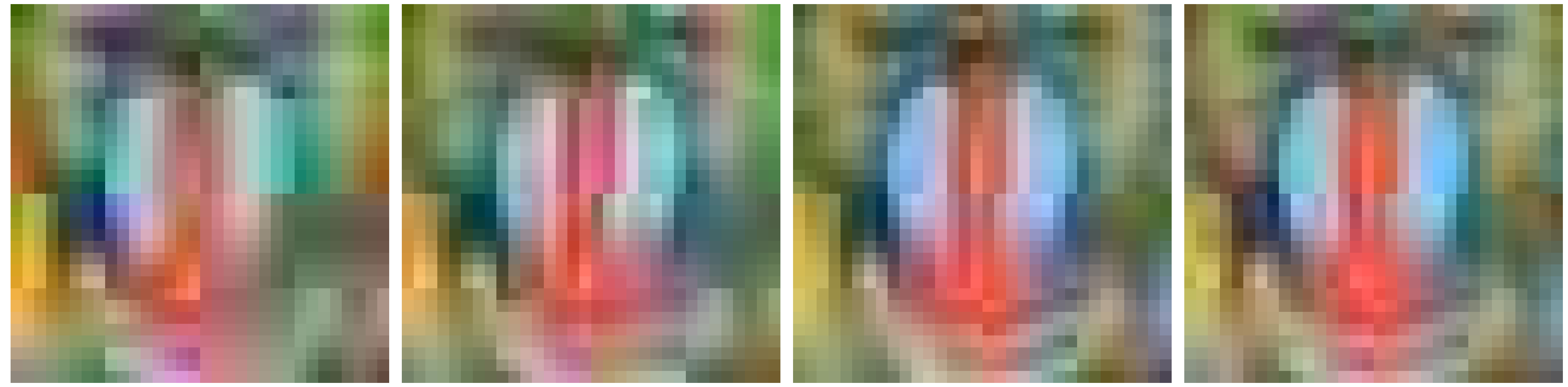
LSTM units:

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

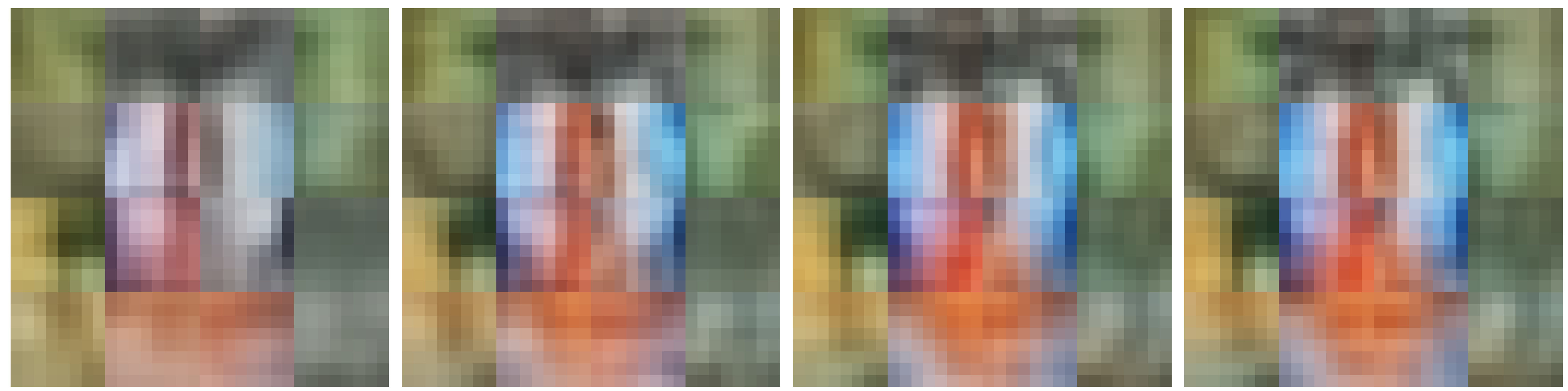
Compression results

[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

[Toderici ICLR 16]

Compression results

SSIM: structural similarity index

	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)

Camera-specific compression?

Security cameras (stationary)

Head mounted cameras

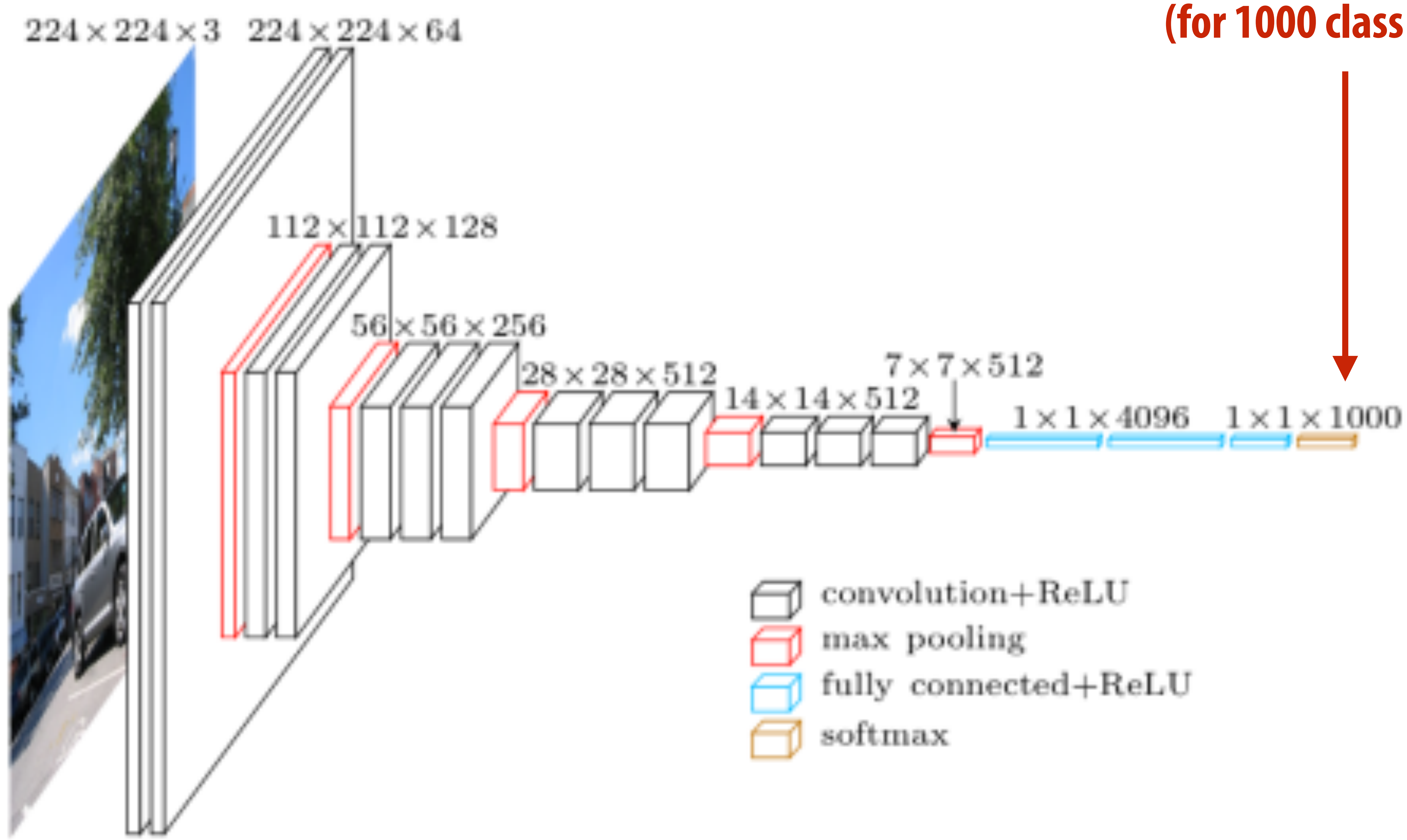
Cross-Stitch Networks

Recall object classification networks

Lower levels of network "shared" across all categories
(Lower level convolutions produce useful features)

Input:
fixed size image

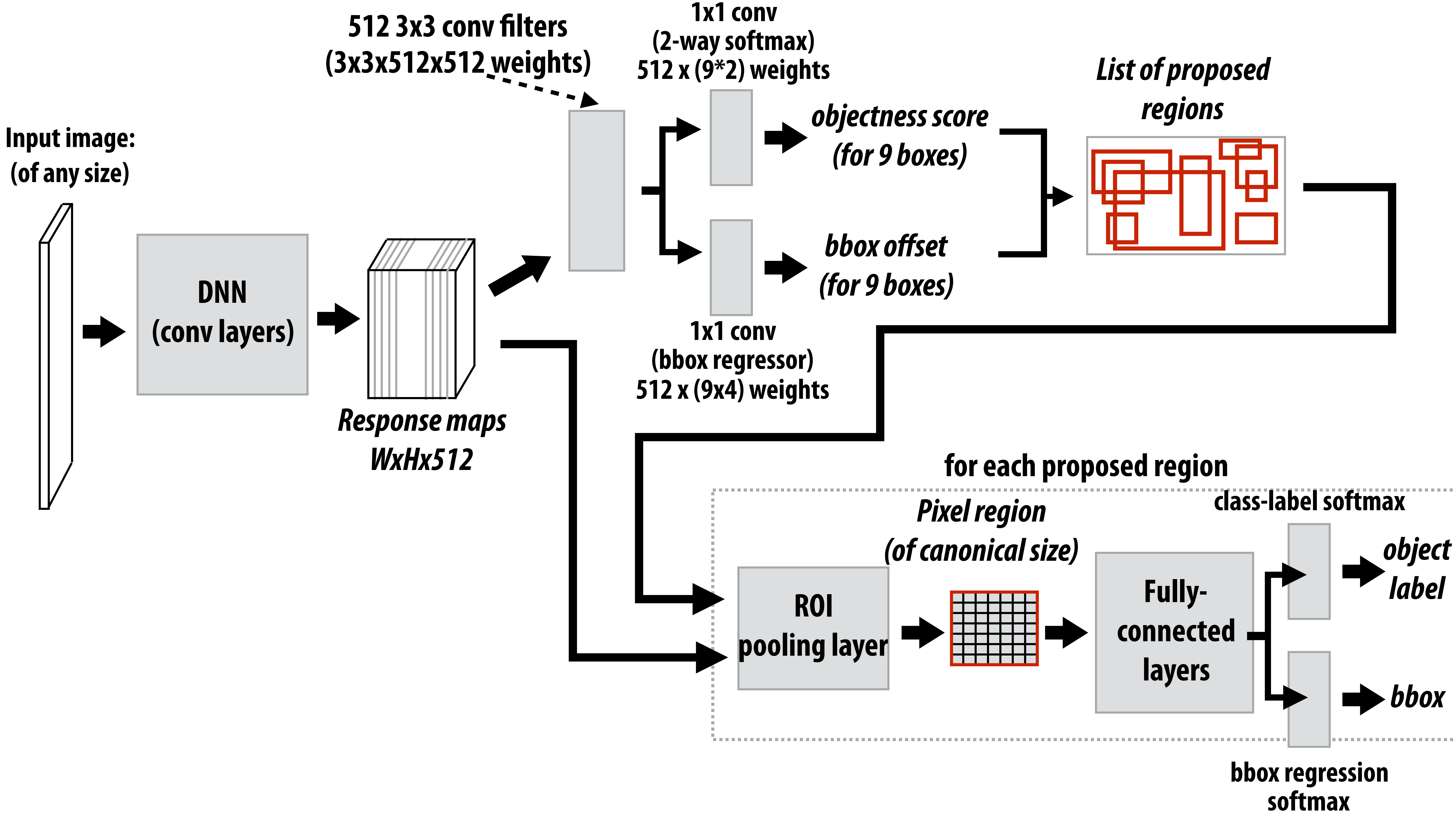
Output:
probability of label
(for 1000 class labels)



VGG-16

Recall Faster R-CNN

Lower conv layers shared between two tasks:
(1) object bounding box prediction and (2) object detection

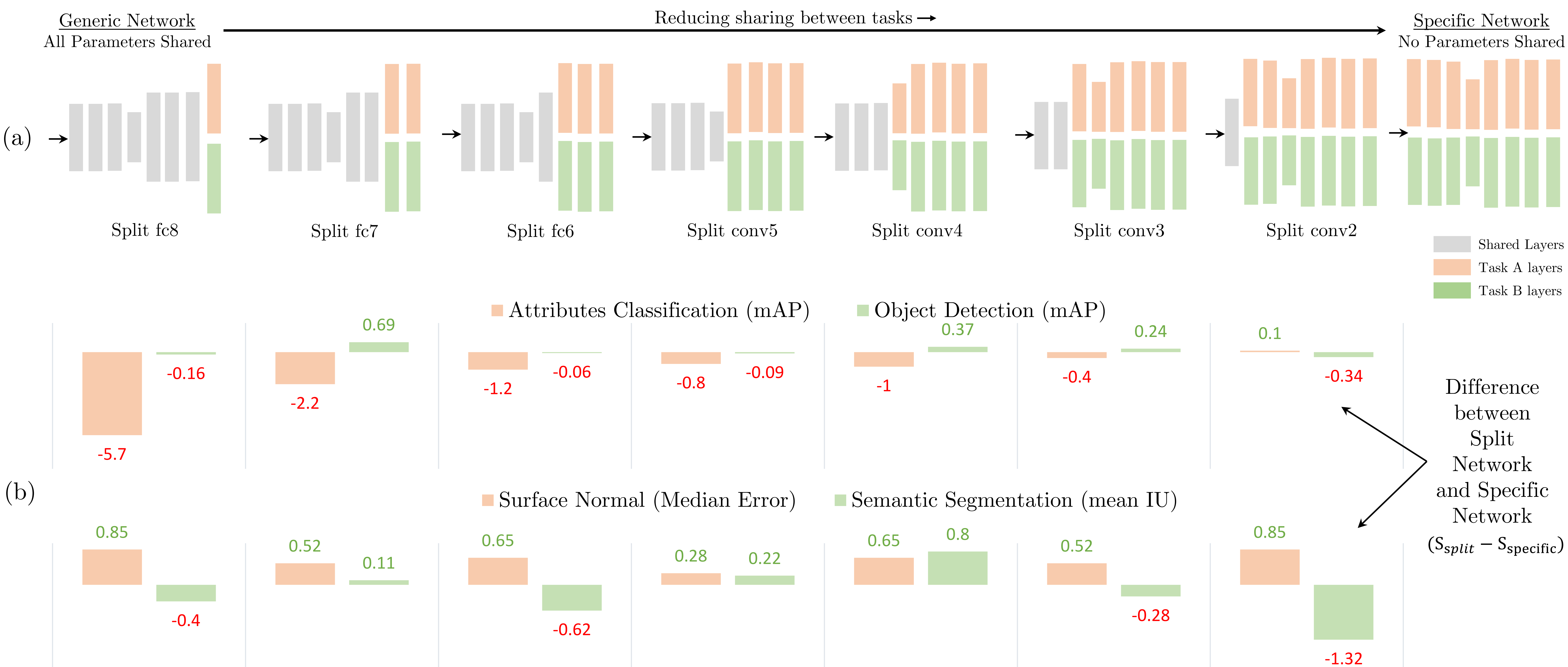


Multi-task learning

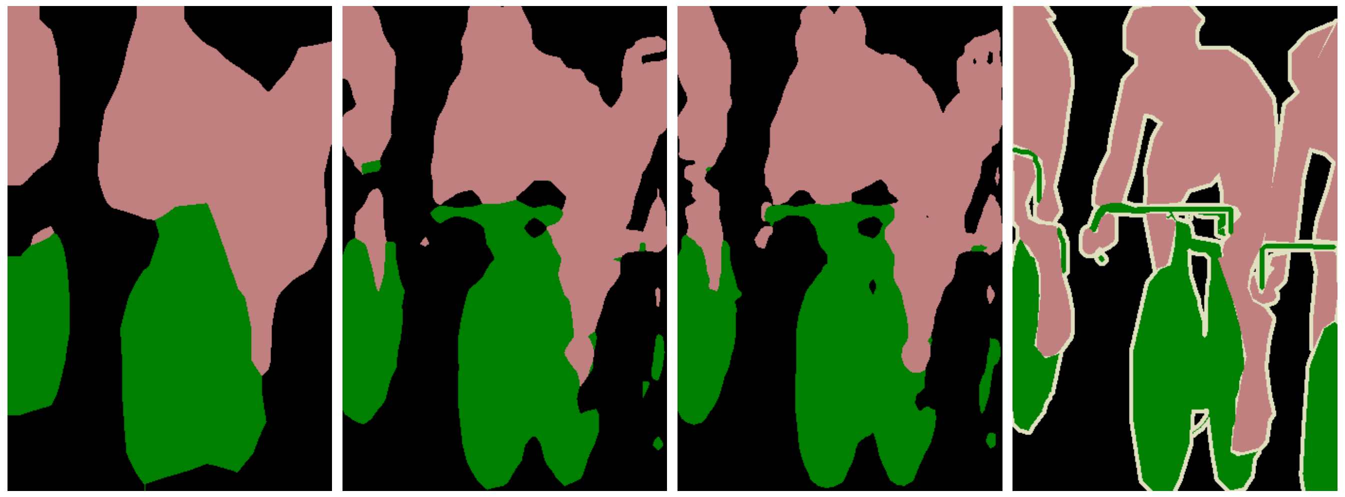
- **For better accuracy**
  - Representations learned to successfully perform multiple tasks A, B, C may be more general, less prone to overfitting, etc.
  - One task serves as a form of supervision for another (as there are effectively more examples available to train the shared lower layers of the network)

- **For increased computational efficiency**
  - Share early layer computations across many tasks (consider performing N related tasks on an input video stream)

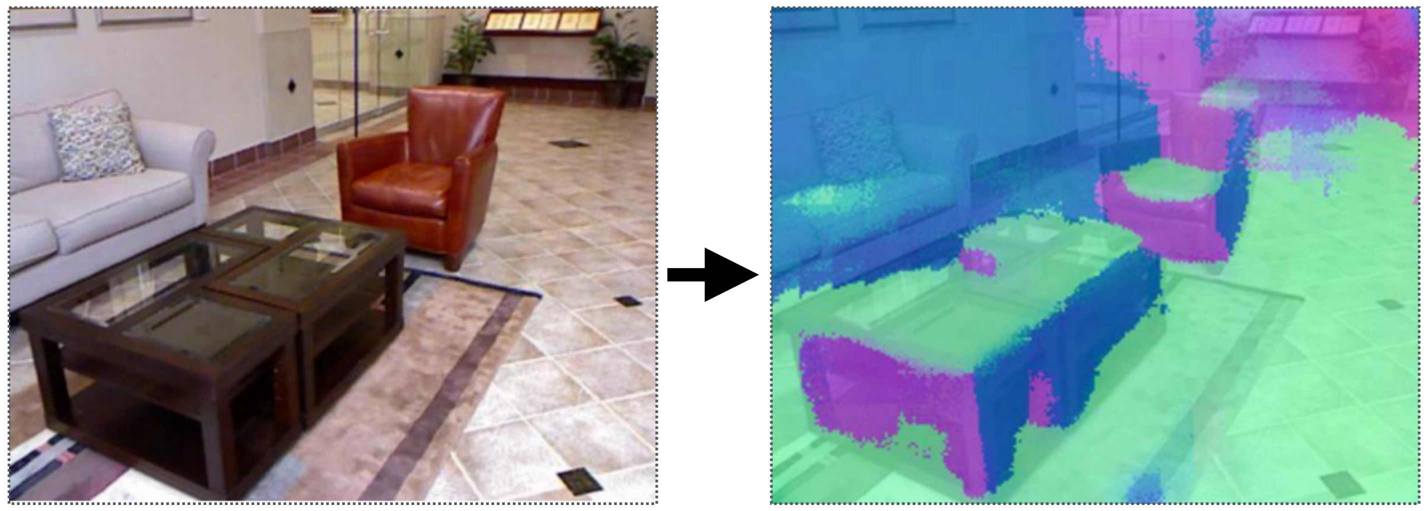
Example: various topologies for multi-class learning



Semantic Segmentation

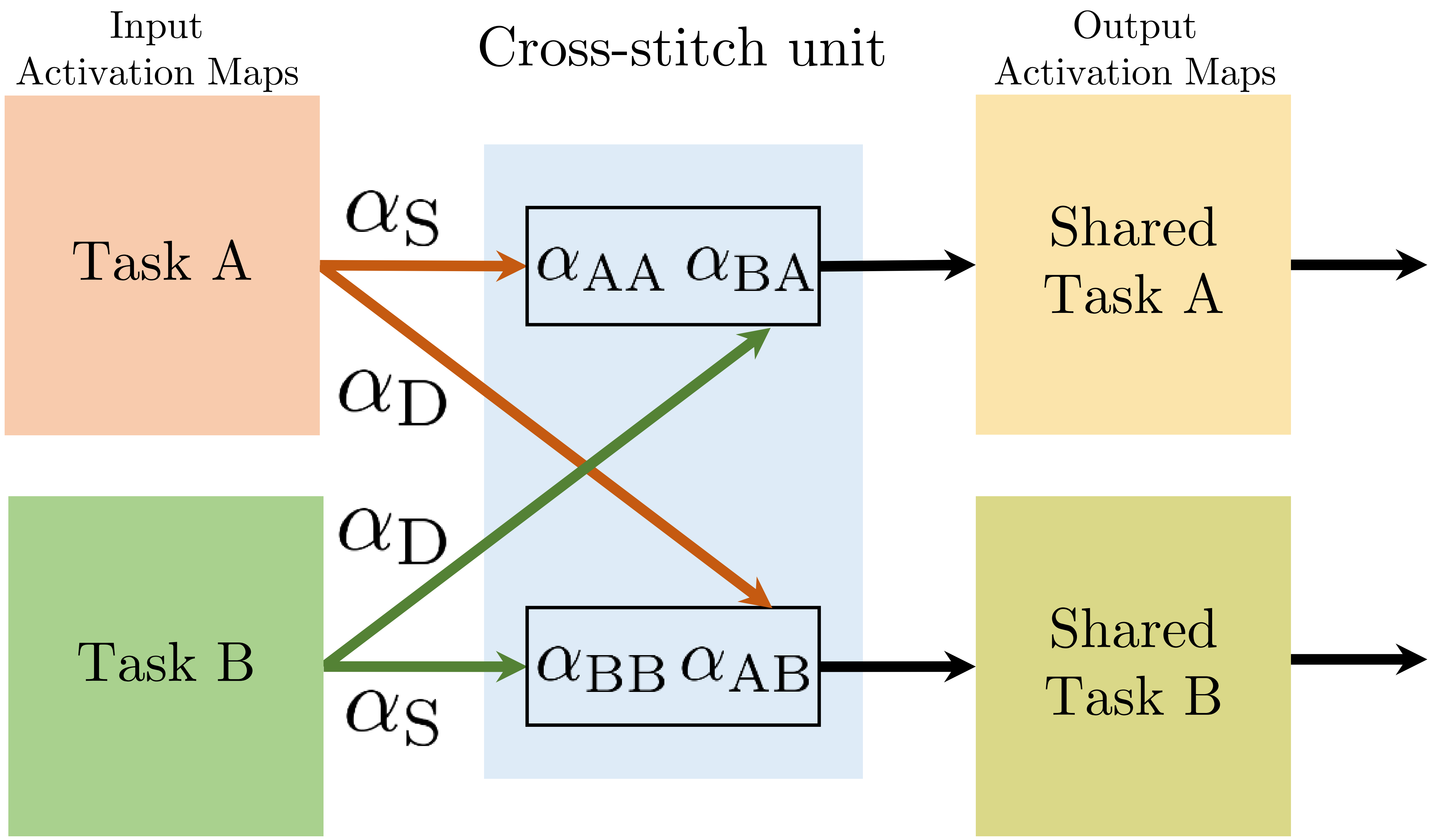


Normal estimation



[Image from: Misra et al. CVPR 2016]

Idea: learn how much to share weights



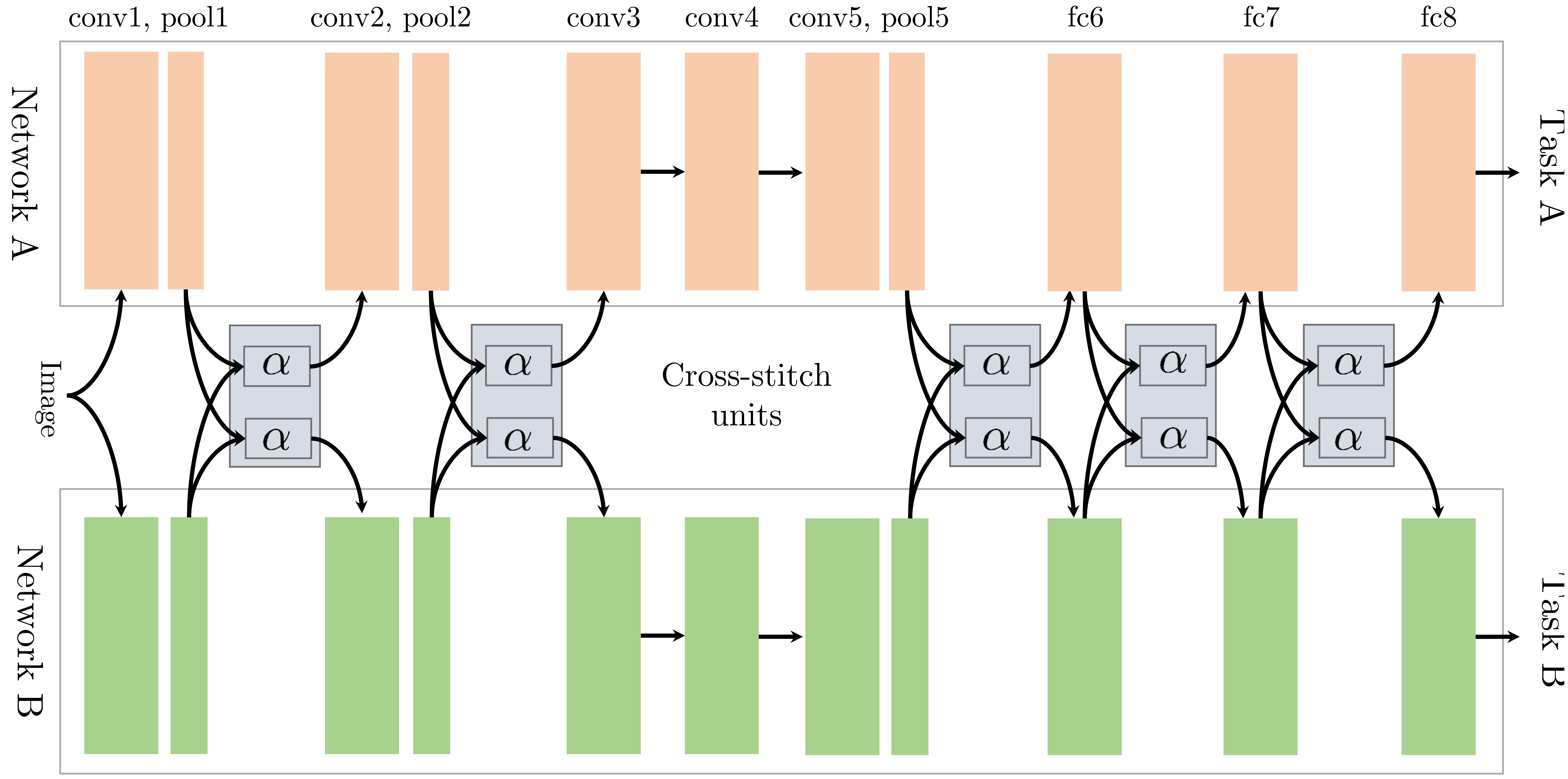
Setting α_{AB} or α_{BA} to 0 implies no input sharing.
Setting row's coefficients to same value implies full sharing .

$$\begin{bmatrix} \tilde{x}^{ij}_{A} \\ \tilde{x}^{ij}_{B} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x^{ij}_{A} \\ x^{ij}_{B} \end{bmatrix}$$

[Image from: Misra et al. CVPR 2016]

Example cross-stitch network

(Two cross-stitched AlexNet's)



[Image from: Misra et al. CVPR 2016]

Example result: segmentation + normal estimation task

Method	Surface Normal: Angle Distance (Lower Better) Mean	Med.	Within t° (Higher Better) 11.25	22.5	30	Segmentation (Higher Better) pixacc	mIU	fwIU
One-task	34.8	19.0	38.3	53.5	59.2	-	-	-
	-	-	-	-	-	46.6	18.4	33.1
Ensemble	34.4	18.5	38.7	54.2	59.7	-	-	-
	-	-	-	-	-	**48.2**	18.9	33.8
Split `conv4`	34.7	19.1	38.2	53.4	59.2	47.8	19.2	33.8
MTL-shared	34.7	18.9	37.7	53.5	58.8	45.9	16.6	30.1
Cross-stitch [**ours**]	**34.1**	**18.2**	**39.0**	**54.4**	**60.2**	47.2	**19.3**	**34.0**

[Misra et al. CVPR 2016]

Spatial Transformer Networks

Common pattern

- Train DNN for perform task on canonical image form
- Example, R-CNN expects contents of boxes to be scaled to fixed-size input
    - This is a crop of original image, followed by resample

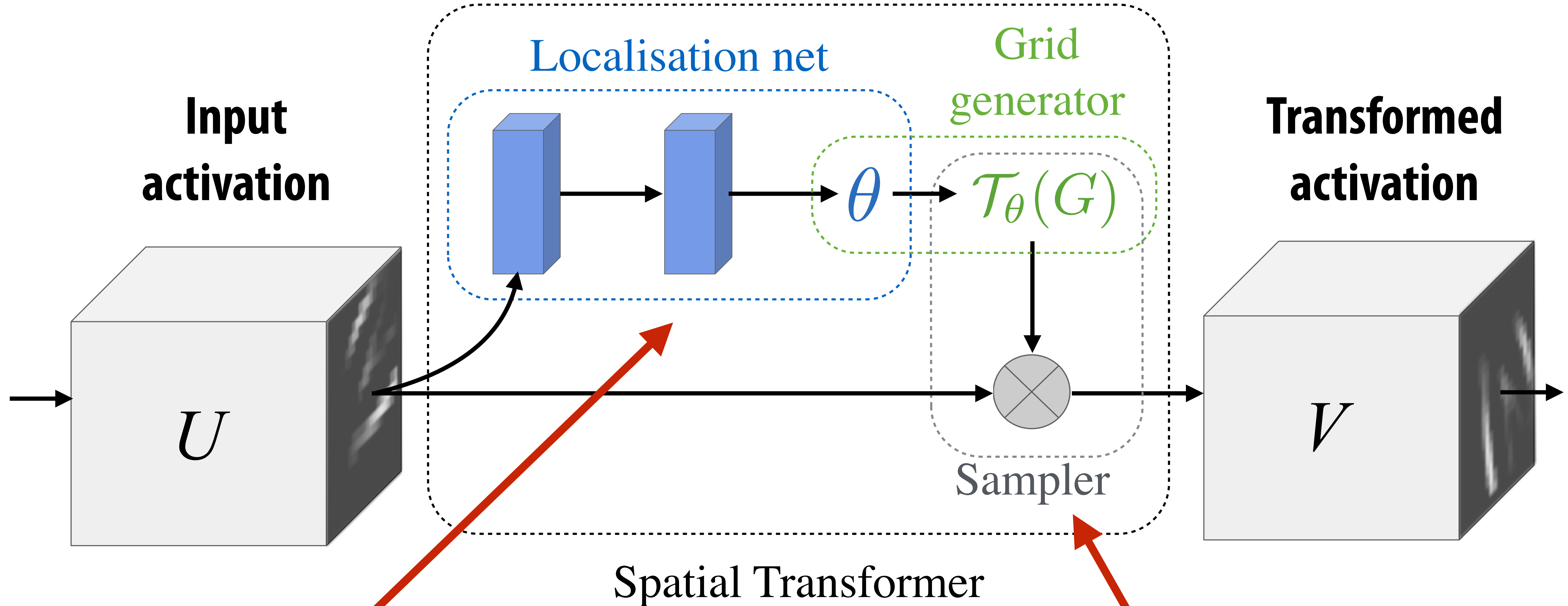
```
for all candidate boxes (x,y,w,h):
    cropped = image_crop(image, bbox(x,y,w,h))
    resized = image_resize(227,227)
    label = detect_object(resized)
    if (label != background)
      // region defined by bbox(x,y,w,h) contains object
      // of class ‘label’
```

- Recall Faster R-CNN: given pixel pattern… predict box, then crop/resample to canonicalize box’s contents

Generalization: learning to canonicalize

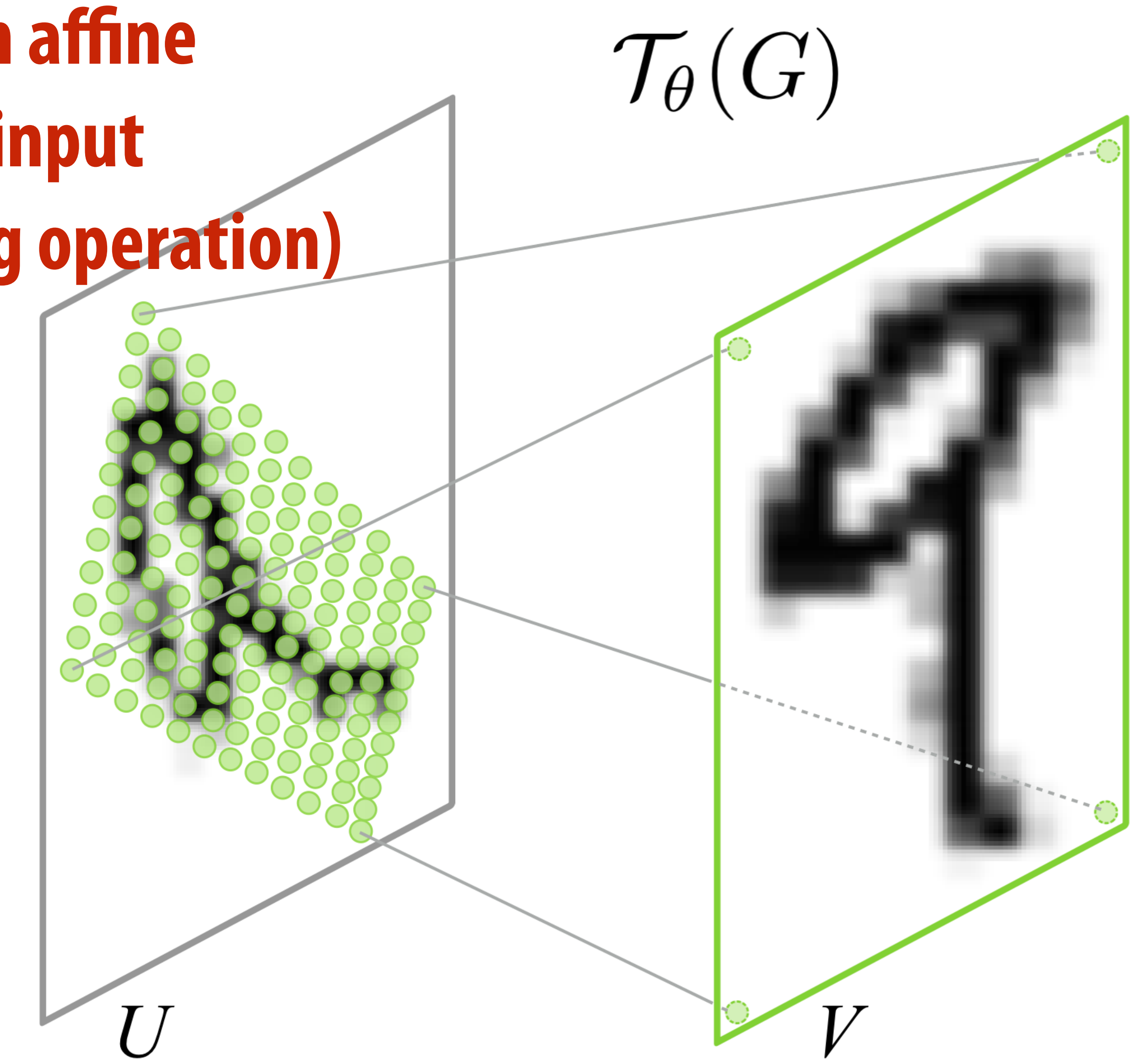
- Modular thinking:
  - Step 1: canonicalize
  - Step 2: perform task (e.g., detect)
- Why not jointly learn network to perform task, and network to canonicalize the input?
  - (In other words, learn to be spatial invariant)

Spatial transformer network



Given input activation predict appropriate transform parameters

Resample to perform affine transformation of input (differentiable resampling operation)



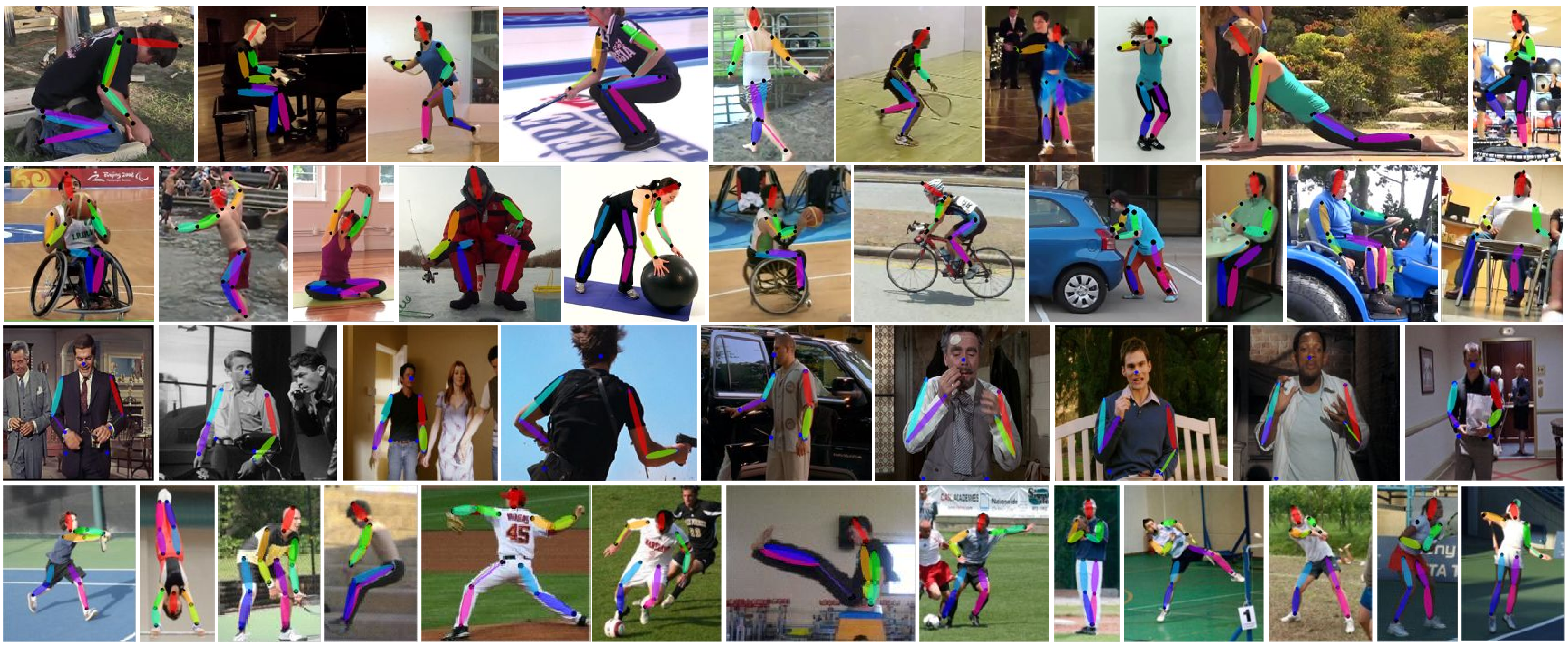
$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

[Jaderberg NIPS 2015]

Convolutional Pose Machines

Problem statement

- **Given image containing a human, output the (x,y) position of each of the human's P parts (parts are joints)**



[Wei CVPR16]

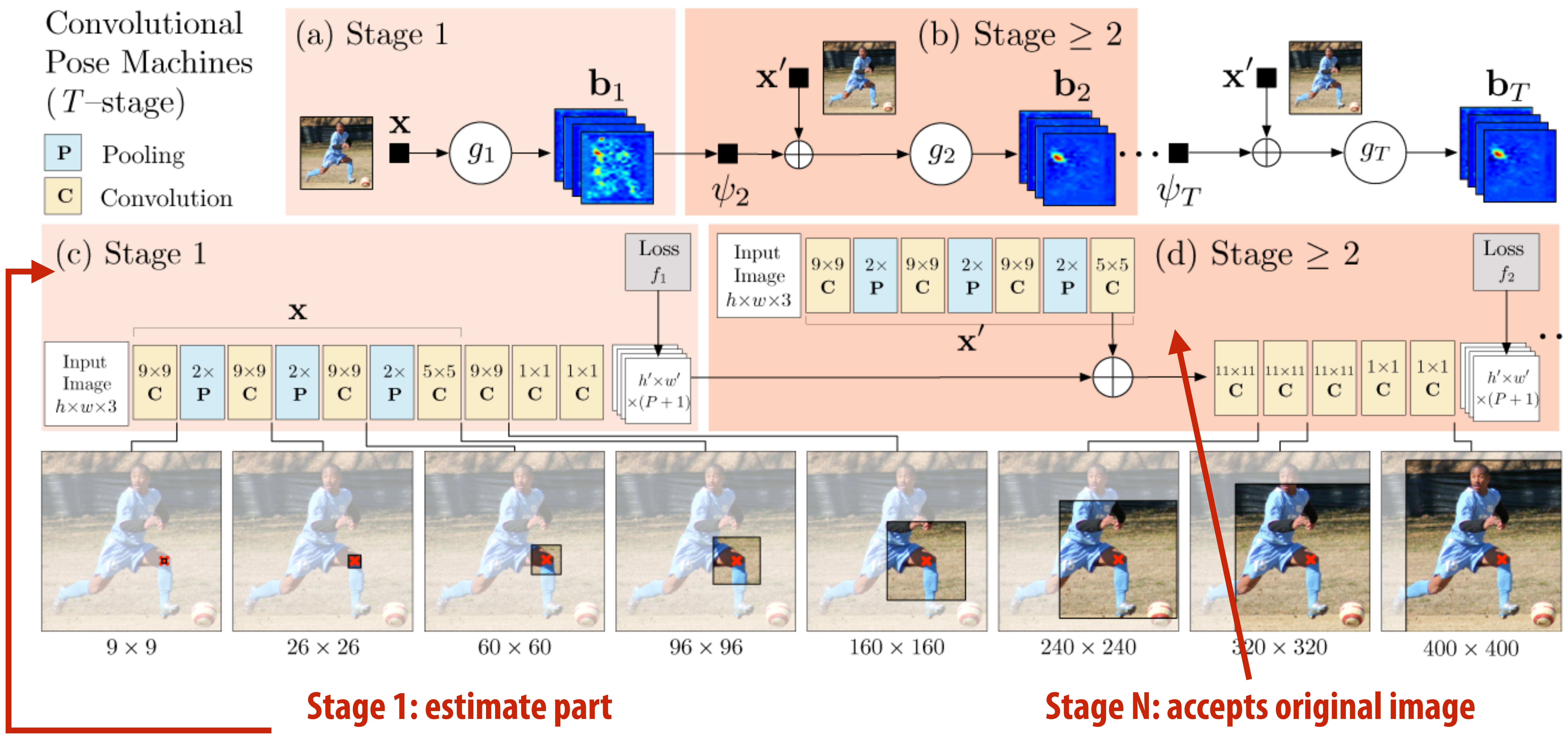
Knowledge

- There are P parts
  - No connectivity or constraints are given

- Context is useful: for any part p_i, knowing the position of other parts may be useful to localizing p_i
  - If I know where the knee is, that should provide information about where the foot is likely to be

- Iteratively update belief about distributions until convergence

Convolutional pose machine (CPM)

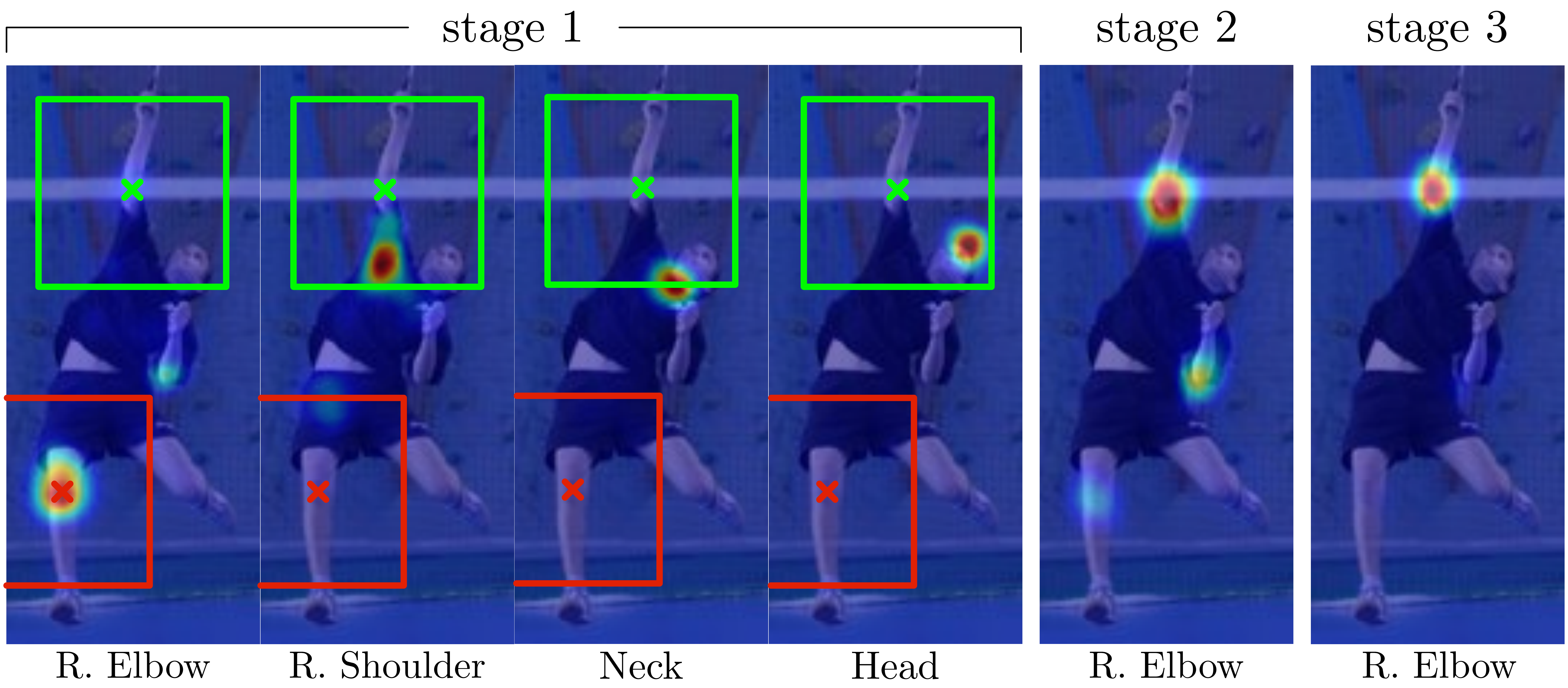


Stage 1: estimate part positions from local information (160x160 receptive field)

Stage N: accepts original image (locally computed parts) and previously predicted locations of all parts from stage N-1

[Wei CVPR16]

Example contextual refinement



Localization of shoulder, head, neck (easy to detect in stage 1) help eliminate incorrect distribution (from stage 1) for position of elbow

Takeaway

- Designers of modern networks impose structure on topology based on human knowledge of how a solution to the task at hand should proceed (topology suggests basic structure of solution)
  - e.g., share these layers
  - e.g., canonicalize and then detect
  - e.g., the number of joints in a human

- Use end-to-end learning to learn the “details” that would be hard (or tedious) for a human to craft
  - e.g., how much to share for each layer
  - e.g., how to canonicalize
  - e.g., how does context help localize human joints in an image