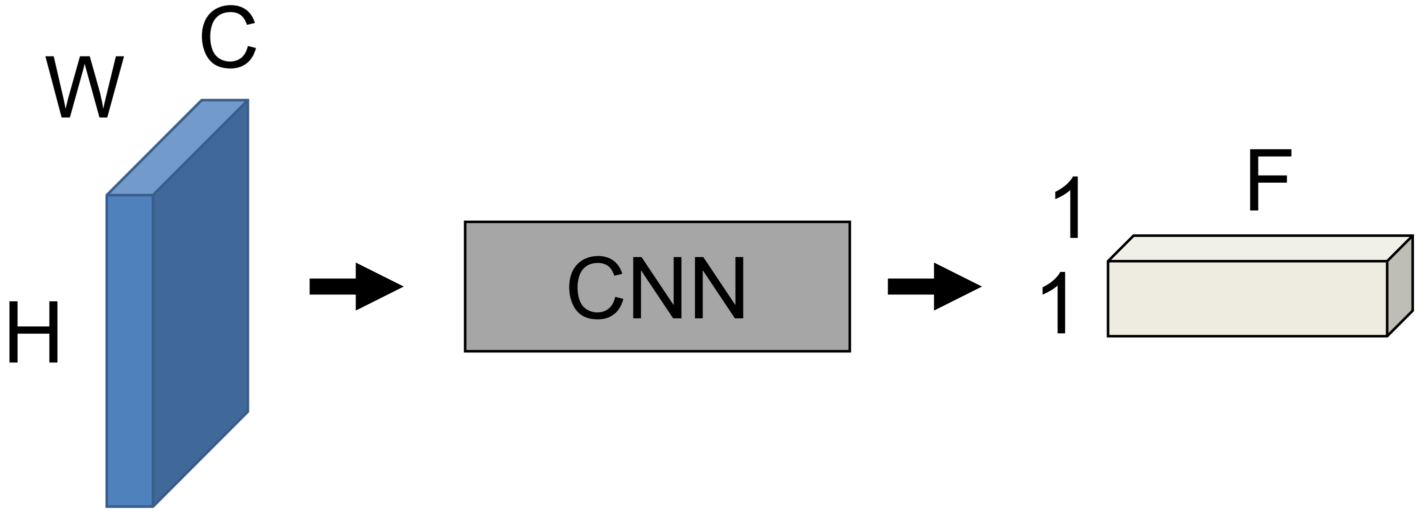
Pixel Labeling

EECS 442 – David Fouhey

Fall 2019, University of Michigan

http://web.eecs.umich.edu/~fouhey/teaching/EECS442_F19/

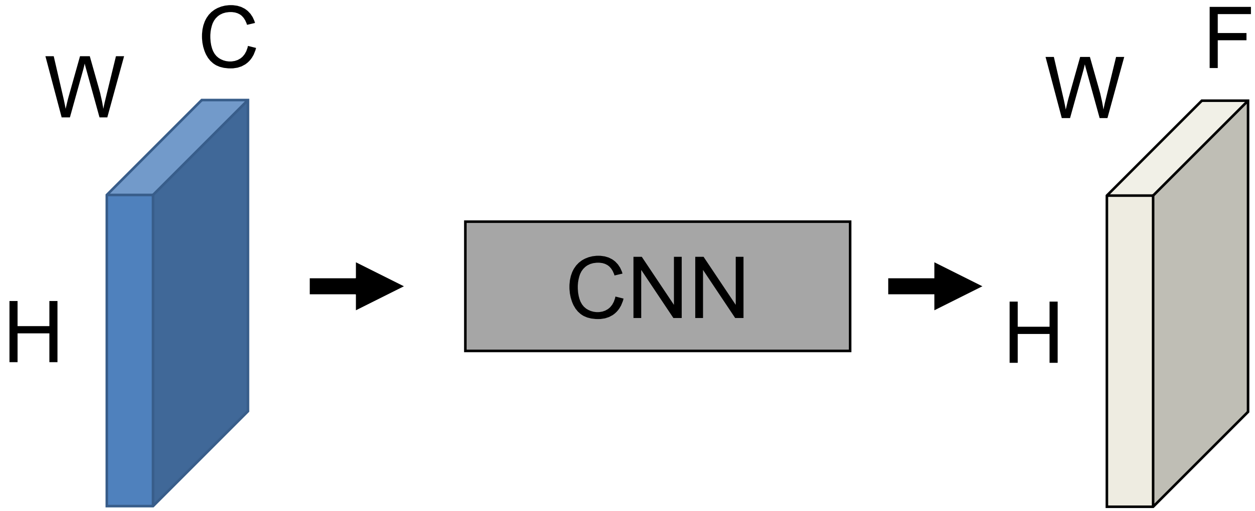
So Far



Convert HxW image into a F-dimensional vector

Is this image a cat?
At what distance was this photo taken?
Is this image fake?

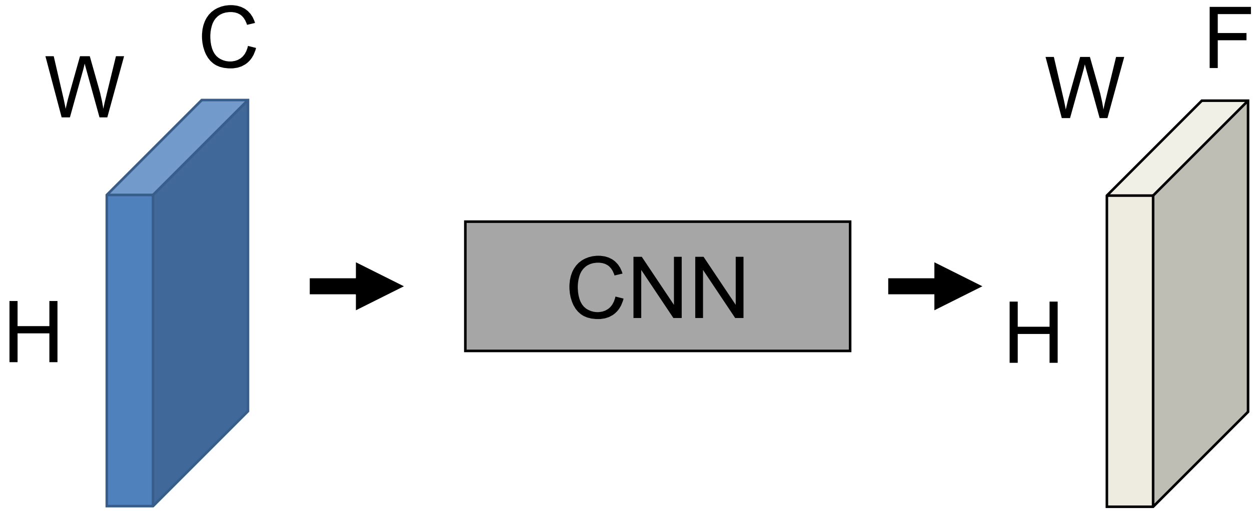
Now



Convert HxW image into a F-dimensional vector

Which pixels in this image are a cat?
How far is each pixel away from the camera?
Which pixels of this image are fake?

Semantic Segmentation



Today's Running Example

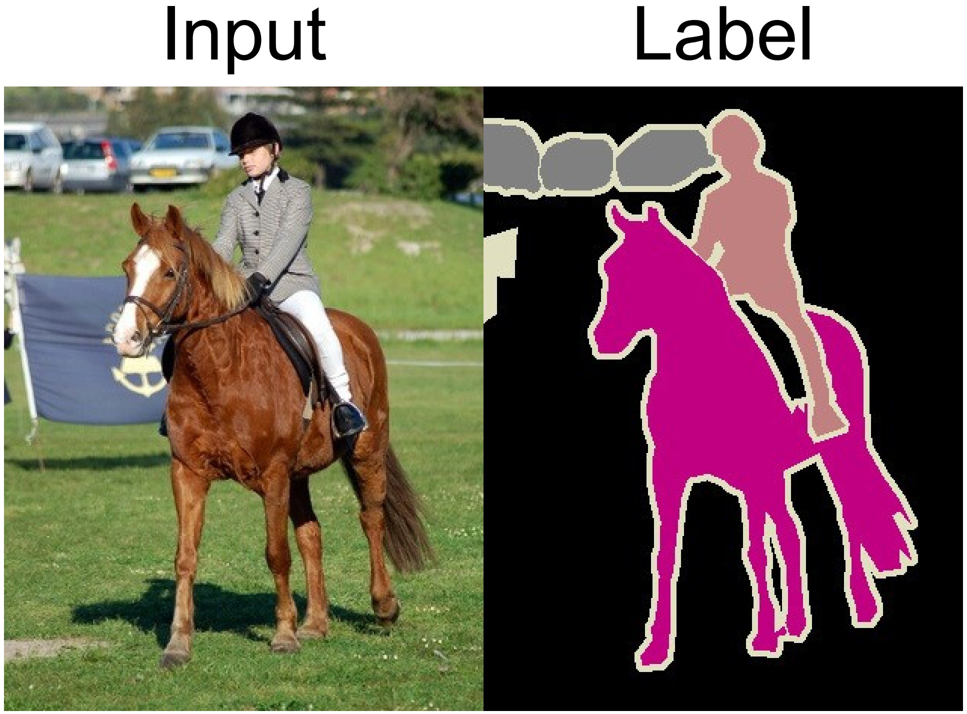
- Predict F-dimensional vector representing probability of each of F classes at every pixel
- Loss computed/backprop'd at *every* pixel.

Semantic Segmentation

Each pixel has label, inc. **background**, and unknown

Usually visualized by colors.

Note: don’t distinguish between object *instances*



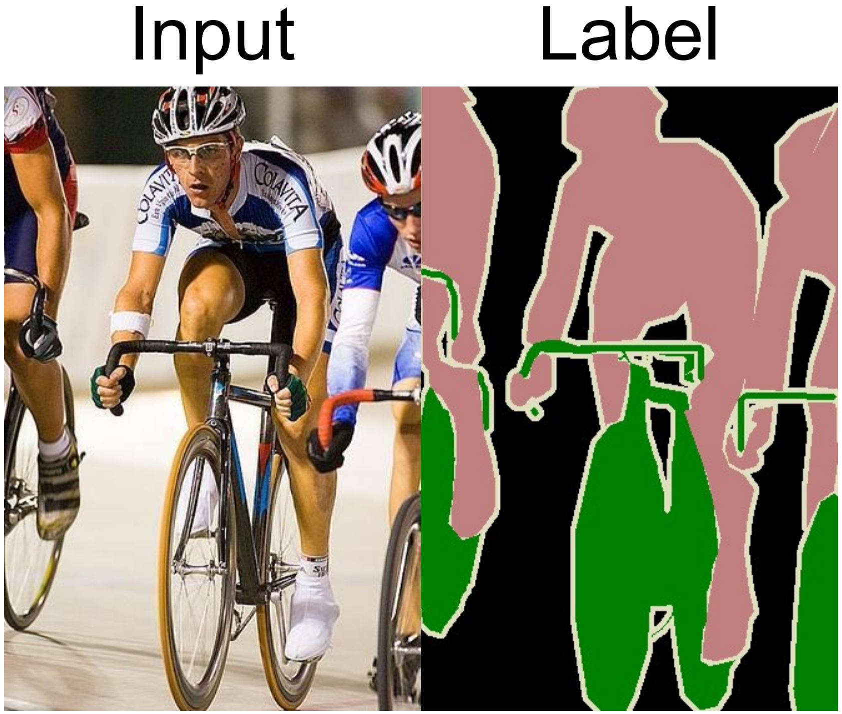
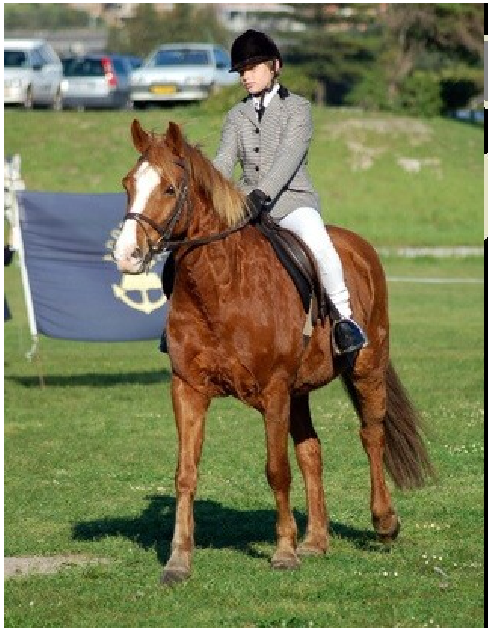


Image Credit: Everingham et al. Pascal VOC 2012.

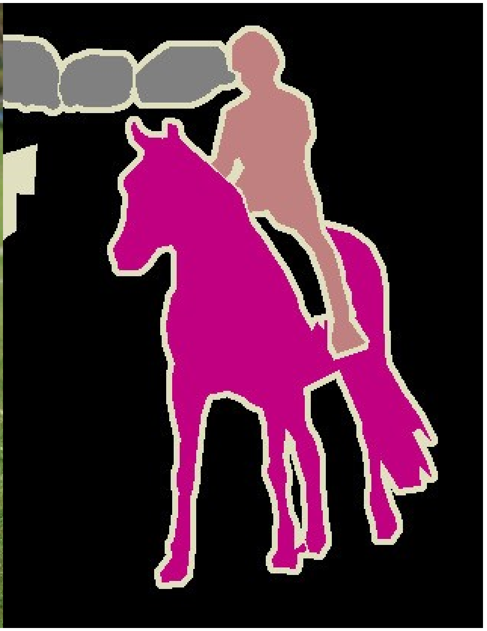
Semantic Segmentation

“Semantic”: a usually meaningless word.
Meant to indicate here that we’re **naming** things.

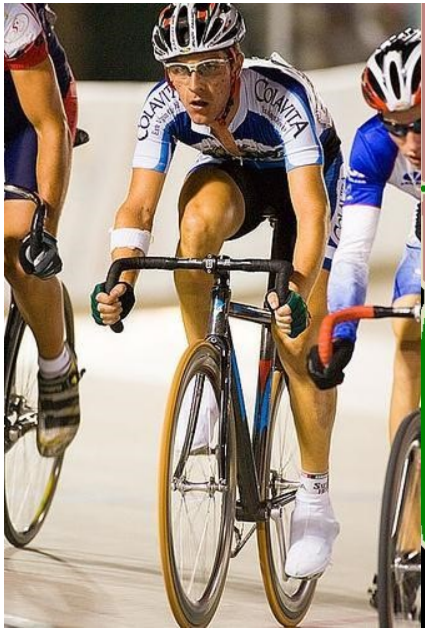
Input



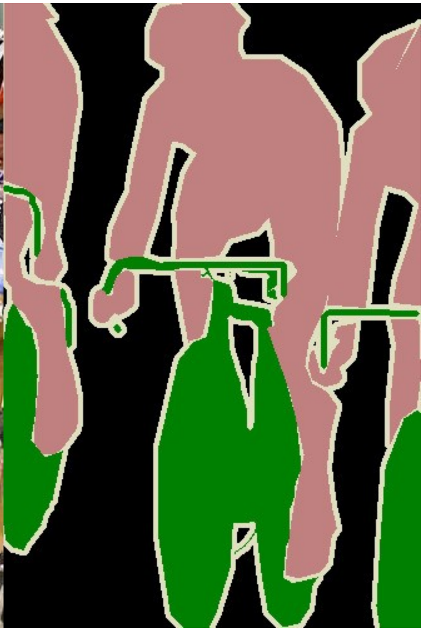
Label



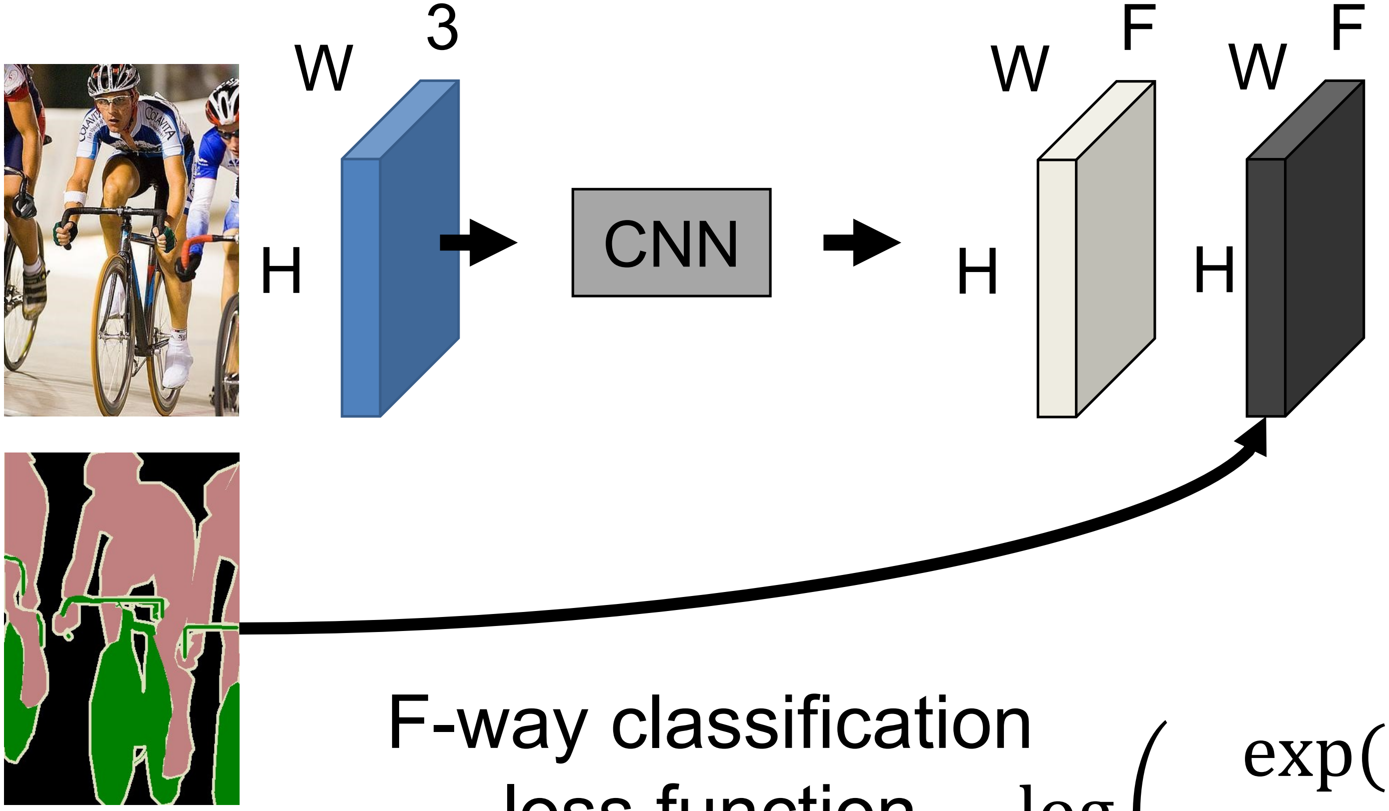
Input



Label



Semantic Segmentation



F-way classification
loss function
at every pixel:

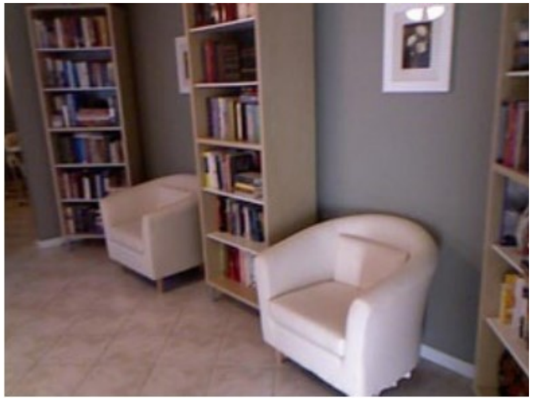
$$-\log\left(\frac{\exp((Wx)_{y_i}}{\sum_k \exp((Wx)_k))}\right)$$

Image Credit: Everingham et al. Pascal VOC 2012.

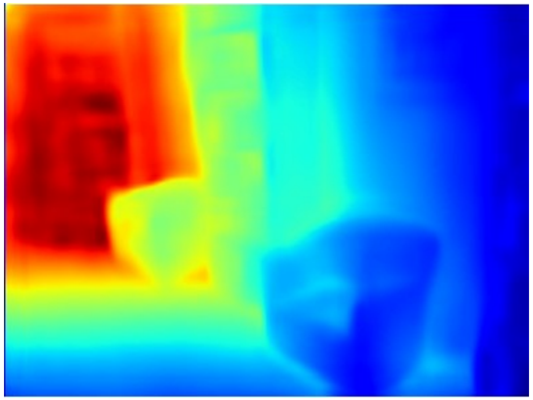
Other Tasks – Depth Prediction

Instead: give label of depthmap, train network to do regression (e.g., $||z_i - \widehat{z_i}||$ where z_i is the ground-truth and $\widehat{z_i}$ the prediction of the network at pixel i).

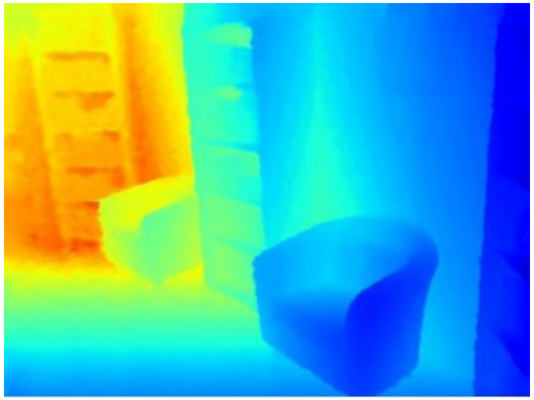
Input HxWx3
RGB Image



Output HxWx1
Depth Image

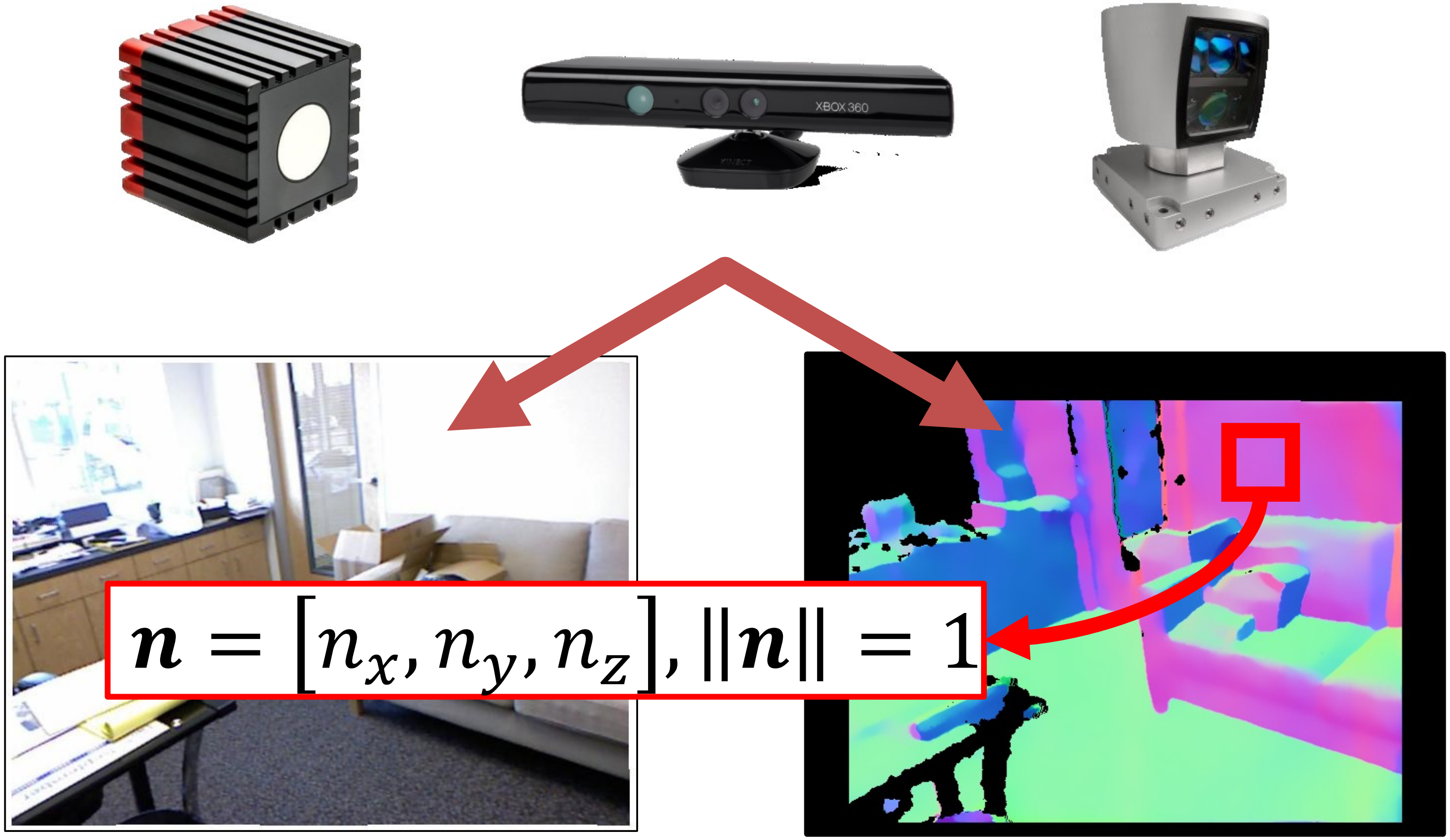


True HxWx1
Depth Image



Result credit: Eigen and Fergus, ICCV 2015

Other Tasks – Surface Normals



$$\boldsymbol{n} = [n_x, n_y, n_z], \|\boldsymbol{n}\| = 1$$

Color Image

Normals

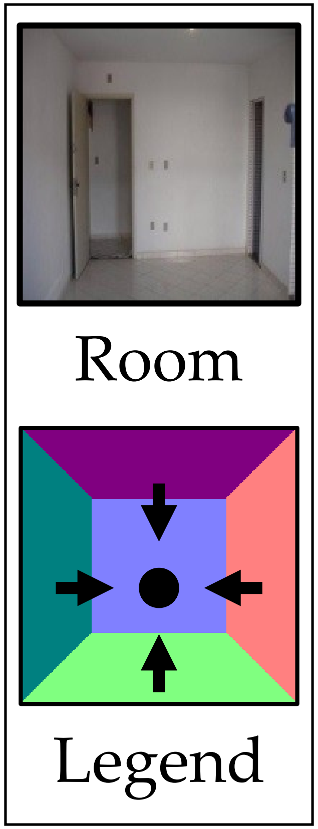


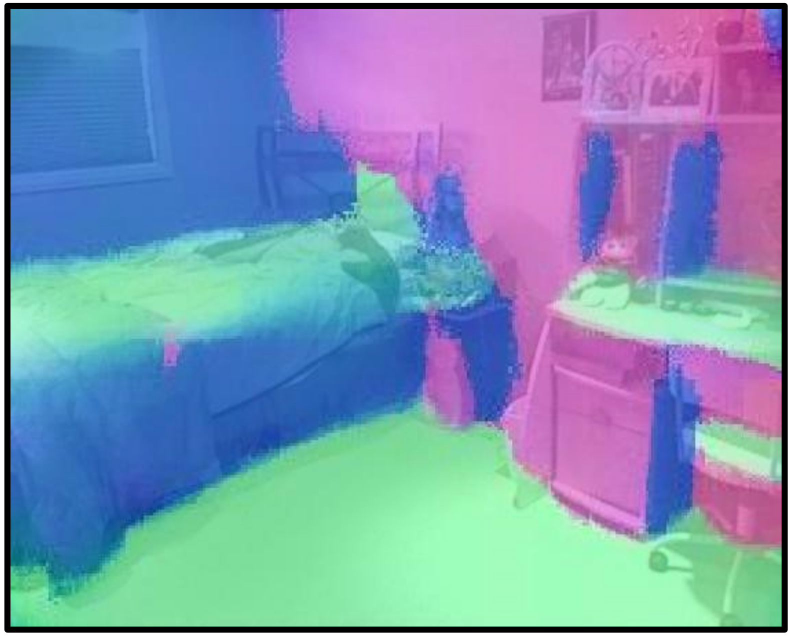
Image credit: NYU Dataset, Silberman et al. ECCV 2012

Surface Normals

Instead: train normal network to minimize $||\boldsymbol{n}_i - \widehat{\boldsymbol{n}_i}||$ where $\boldsymbol{n}_i$ is ground-truth and $\widehat{\boldsymbol{n}_i}$ prediction at pixel i.

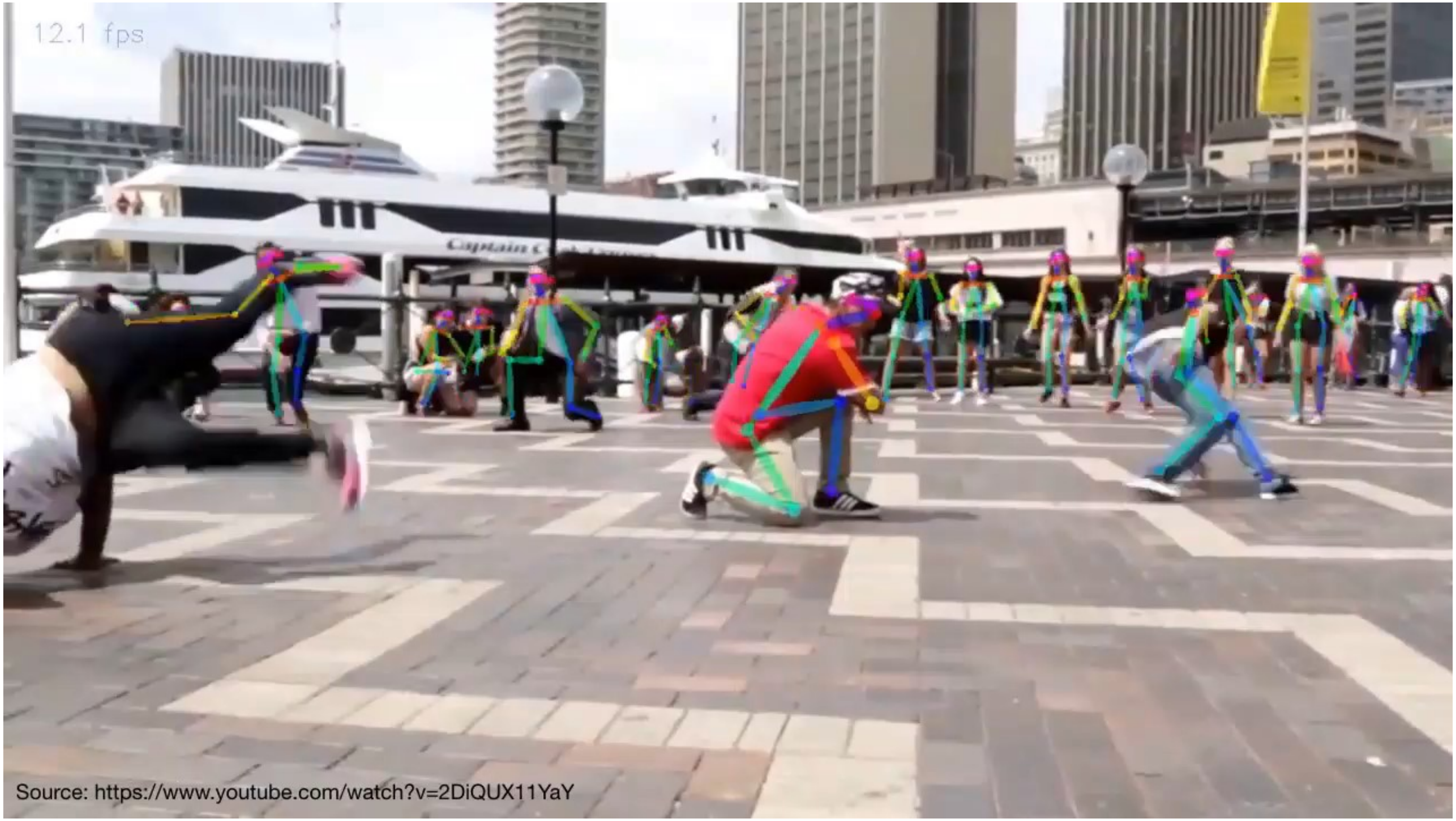
Input: HxWx3
RGB Image

Output: HxWx3
Normals



Result credit: X. Wang, D. Fouhey, A. Gupta, *Designing Deep Networks for Surface Normal Estimation.* CVPR 2014

Other Tasks – Human Pose Estimation



Source: https://www.youtube.com/watch?v=2DiQUX11YaY

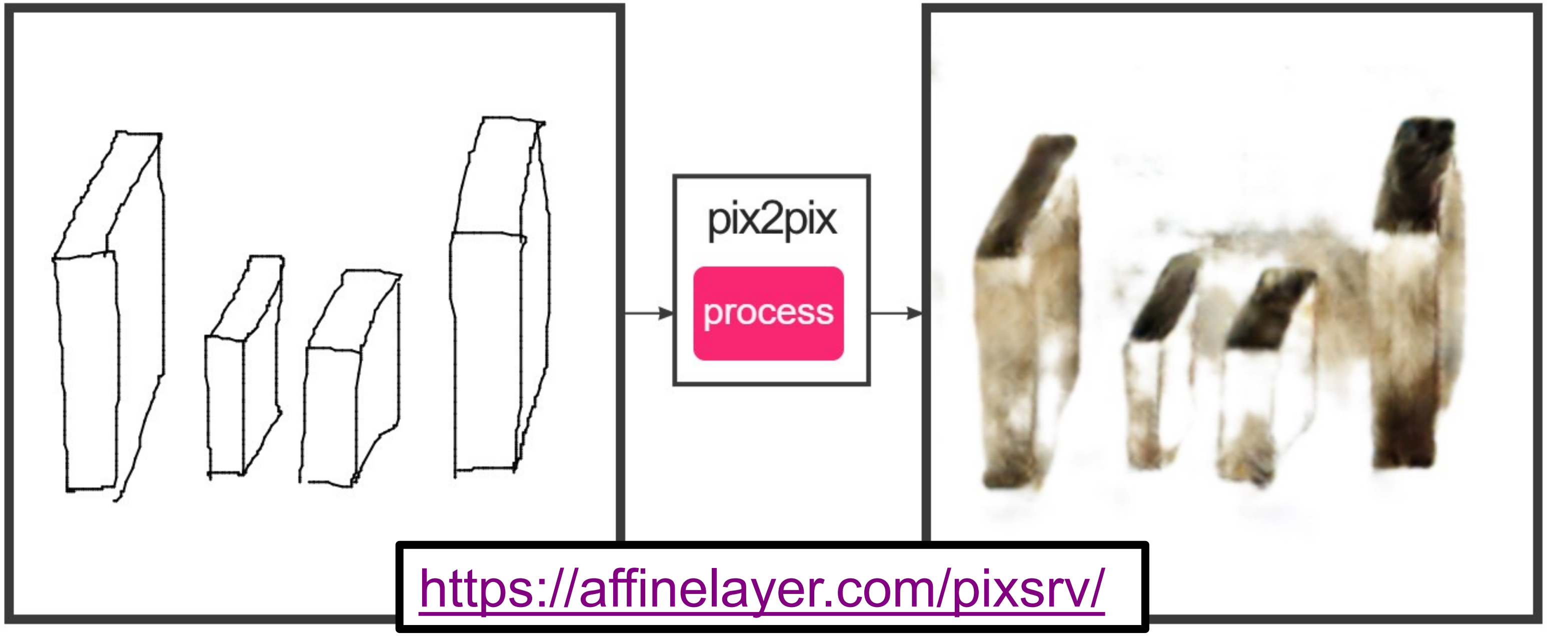
Result credit: Z. Cao et al. *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. CVPR 2017.

Other Task – Edges to Cats

Train network to minimize $\|I_j - \widehat{I_j}\|$ where I_j is GT and $\widehat{I_j}$ prediction at pixel j (*plus other magic*).

Input: HxWx1
Sketch Image

Output: HxWx3
Image



https://affinelayer.com/pixsrv/

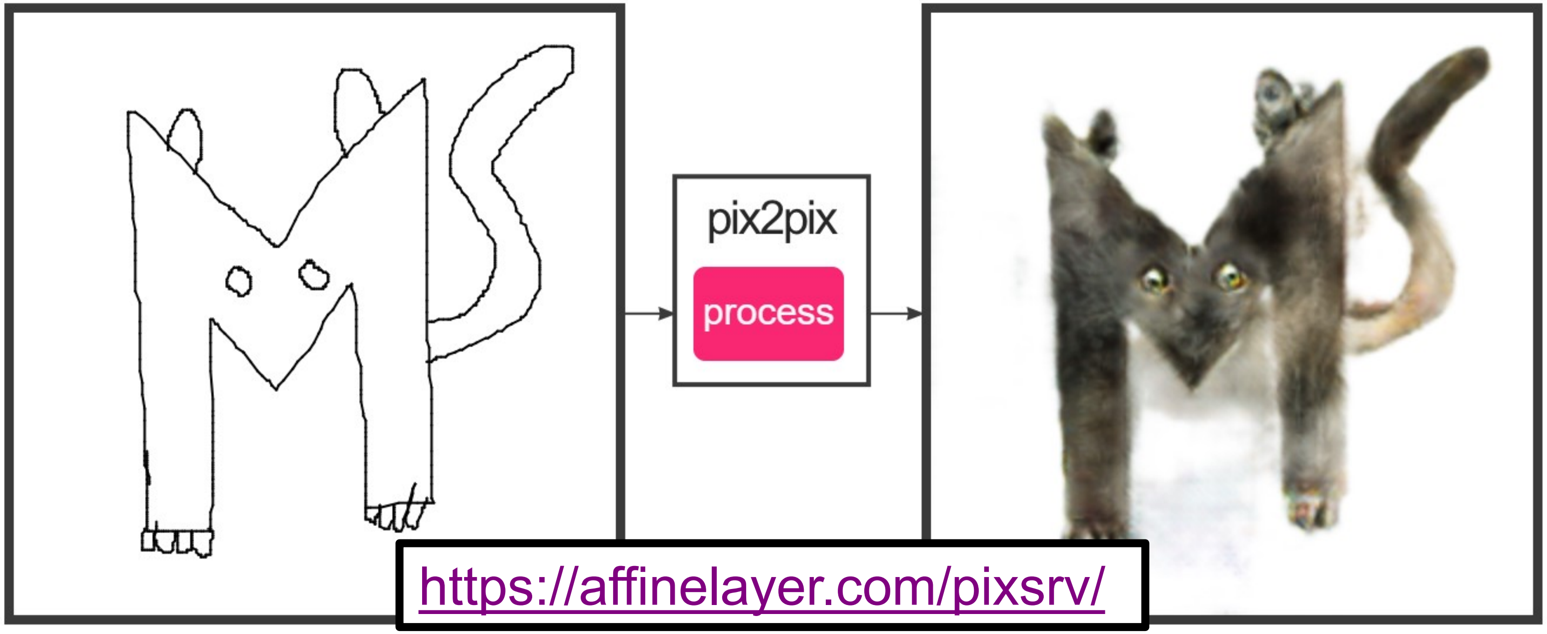
Result credit: Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. CVPR 2017.

Other Task – Edges to Cats

Train network to minimize $\|I_j - \widehat{I_j}\|$ where I_j is GT and $\widehat{I_j}$ prediction at pixel j (*plus other magic*).

Input: HxWx1
Sketch Image

Output: HxWx3
Image



https://affinelayer.com/pixsrv/

Result credit: Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. CVPR 2017.

Why Is This Task Hard?

Image credit: A. Torralba

Why Is This Task Hard?

What’s this? (No Cheating!)

(a) Keyboard?

(b) Hammer?

(c) Old cell phone?

(d) Xbox controller?

Image credit: COCO dataset

Why Is This Task Hard?

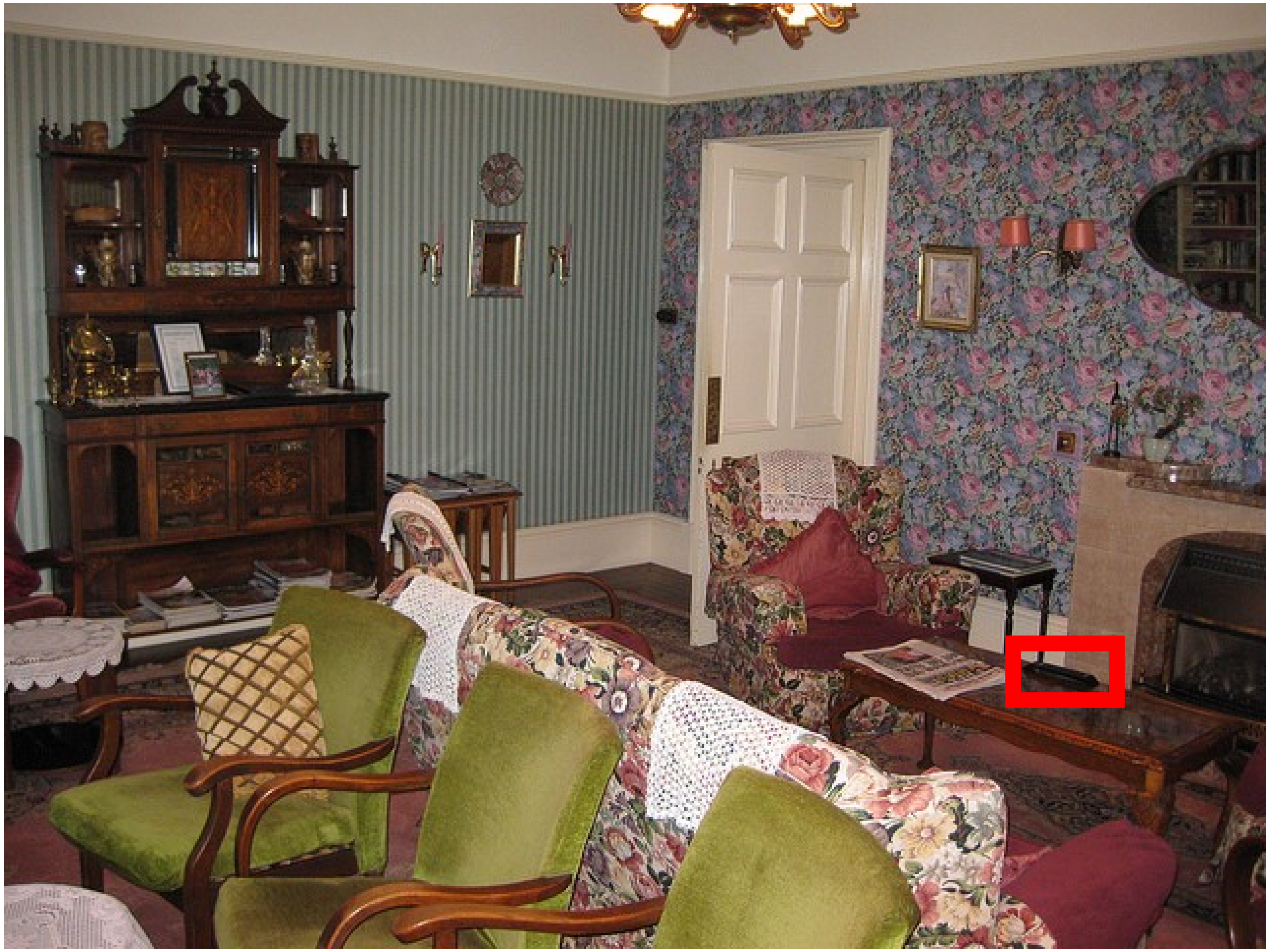
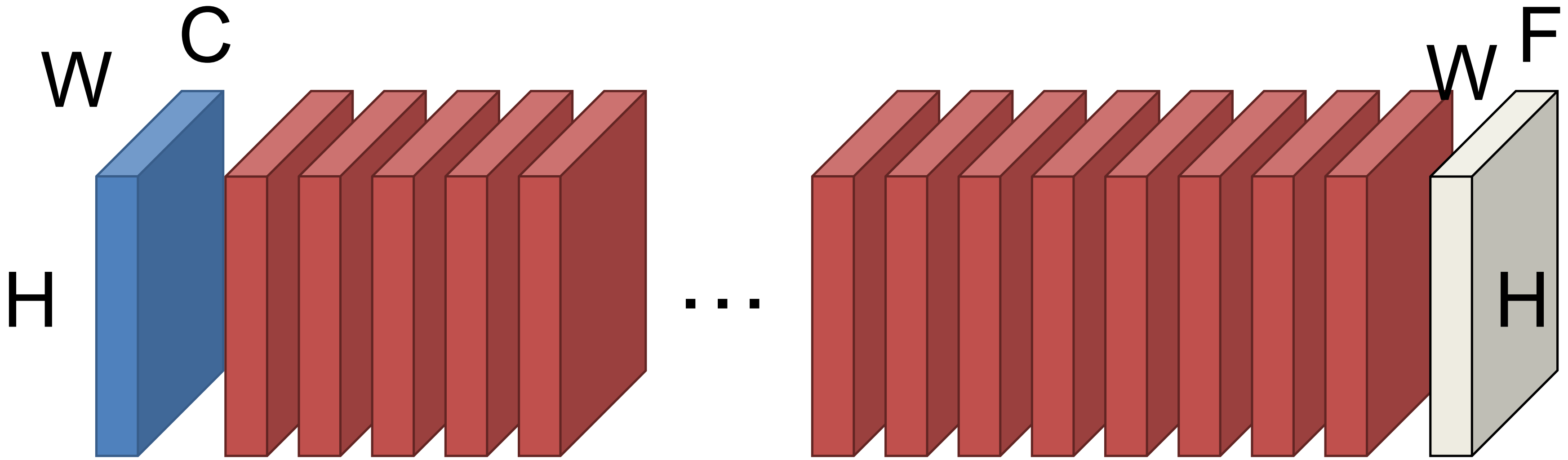


Image credit: COCO dataset

First – Two “Wrong” Ways

- It’s helpful to see two “wrong” ways to do this.

Why Not Stack Convolutions?

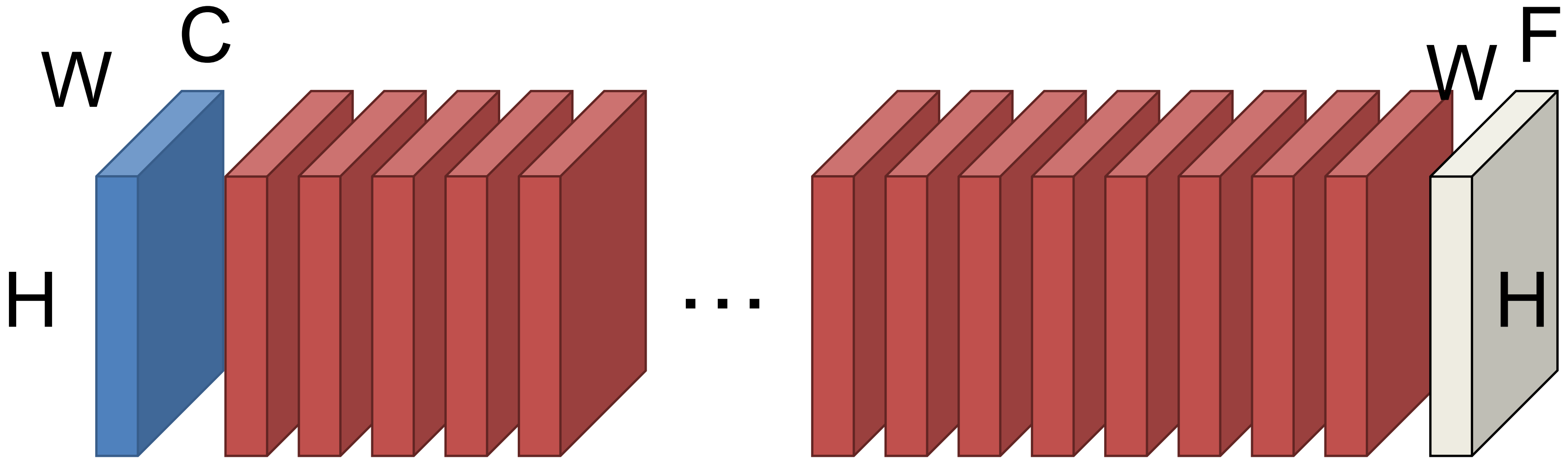


n 3x3 convs have a receptive field of 2n+1 pixels

How many convolutions until >=200 pixels?

100

Why Not Stack Convolutions?



Suppose 200 3x3 filters/layer, H=W=400

Storage/layer/image: 200 * 400 * 400 * 4 bytes = 122MB

Uh oh!*

*100 layers, batch size of 20 = 238GB of memory!

If Memory’s the Issue…

Crop out every sub-window and predict the label in the middle.

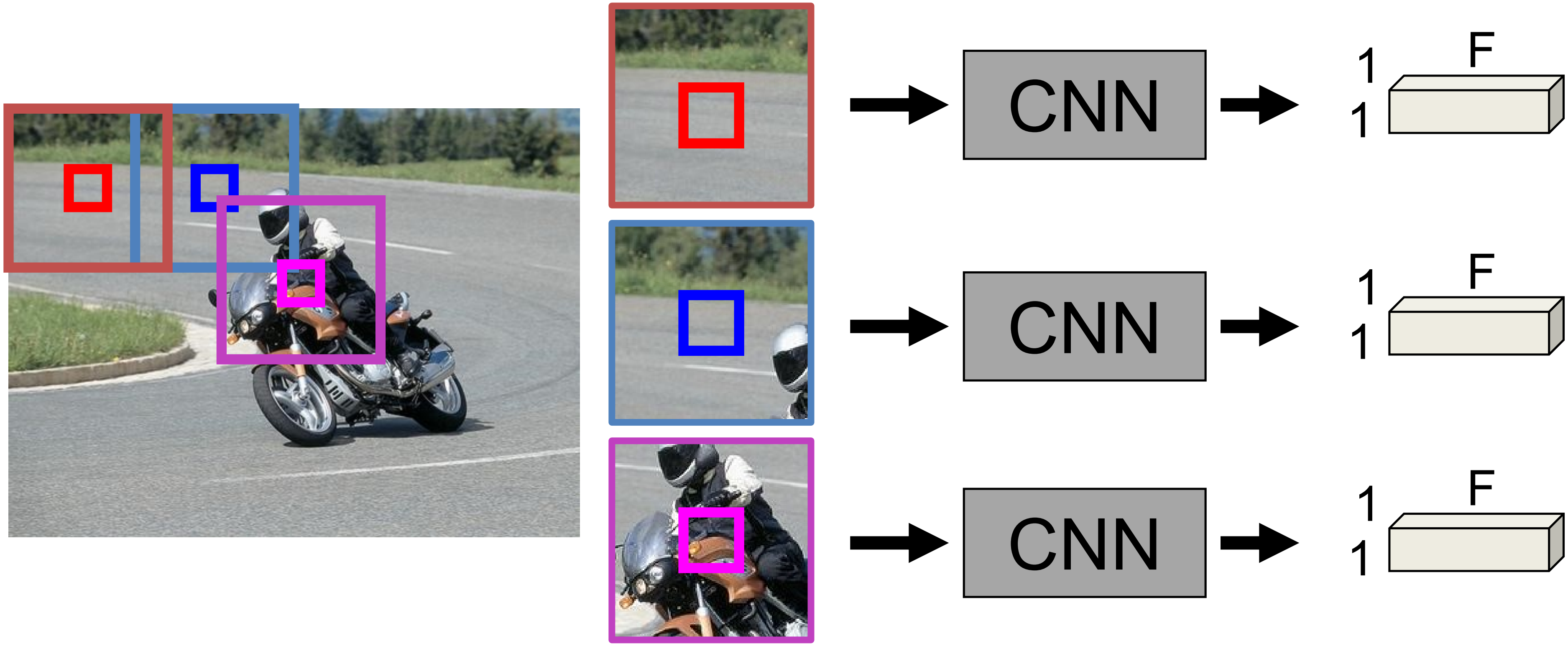
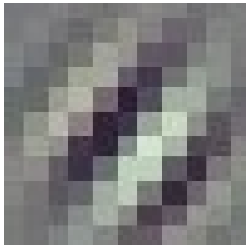


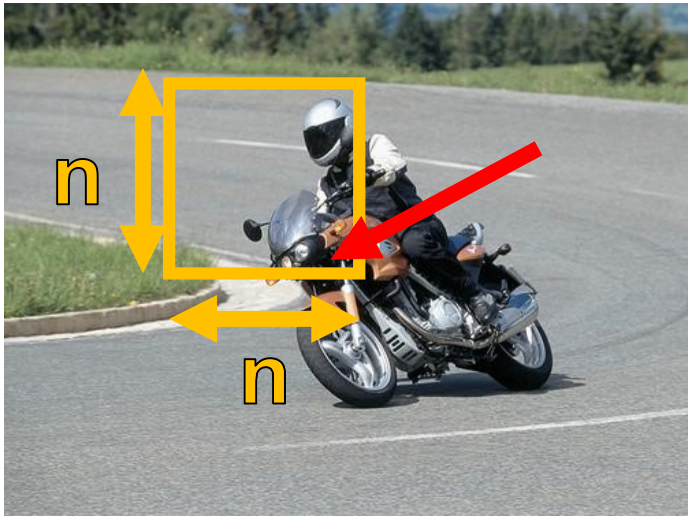
Image credit: PASCAL VOC, Everingham et al.

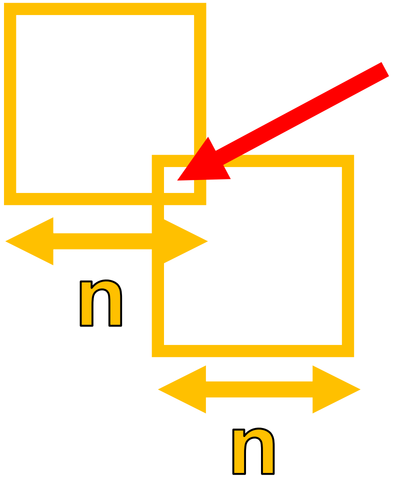
If Memory's the Issue…

Meet "Gabor". We extract NxN patches and do independent CNNs. **How many times does Gabor filter the red pixel?**



Gabor





Answer:
(2n-1)*(2n-1)
Gabor's looking for a better job with a smarter boss.

Image credit: PASCAL VOC, Everingham et al.

The Big Issue

We need to:

1. Have large receptive fields to figure out what we’re looking at
2. Not waste a ton of time or memory while doing so

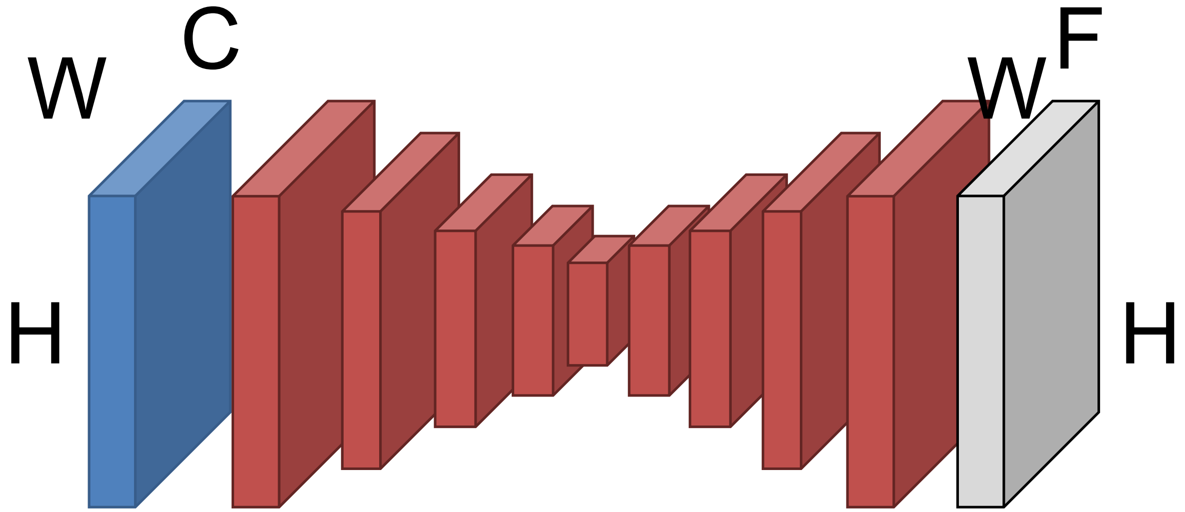
These two objectives are in total conflict

Encoder-Decoder

Key idea: First **downsample** towards middle of network. Then **upsample** from middle.

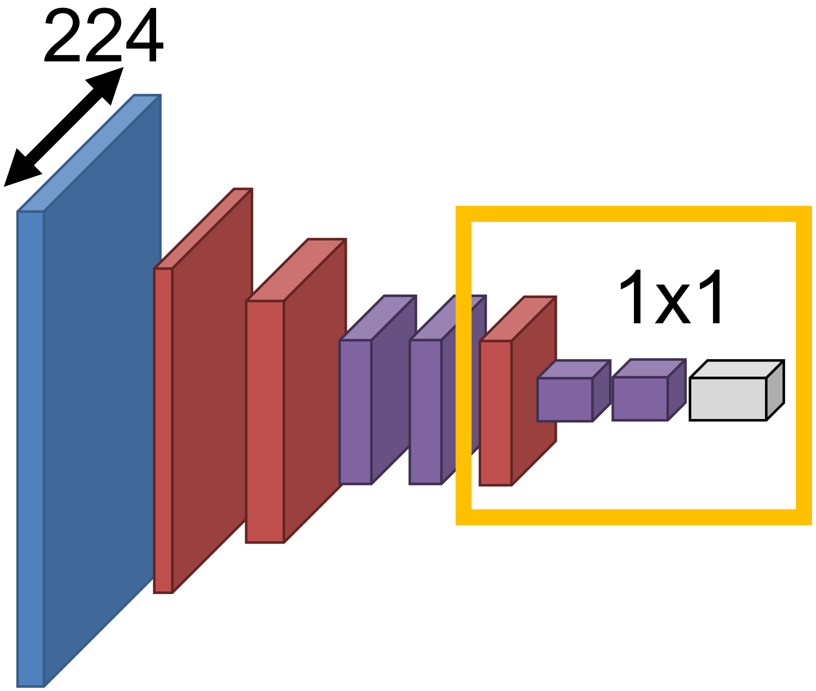
How do we downsample?

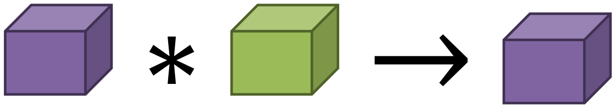
Convolutions, pooling



Where Do We Get Parameters?

Convnet that maps images to vectors



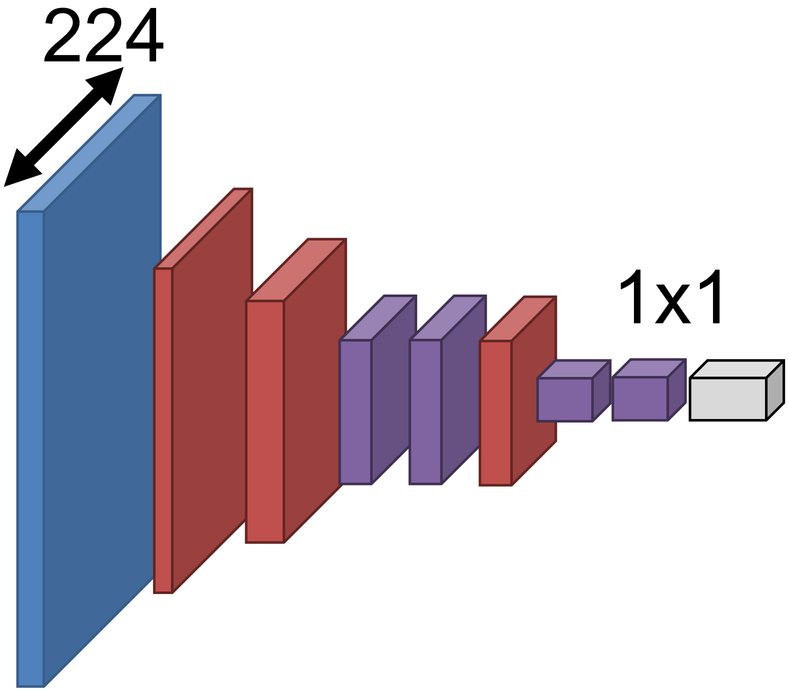


Recall that we can rewrite any vector-vector operations via 1x1 convolutions

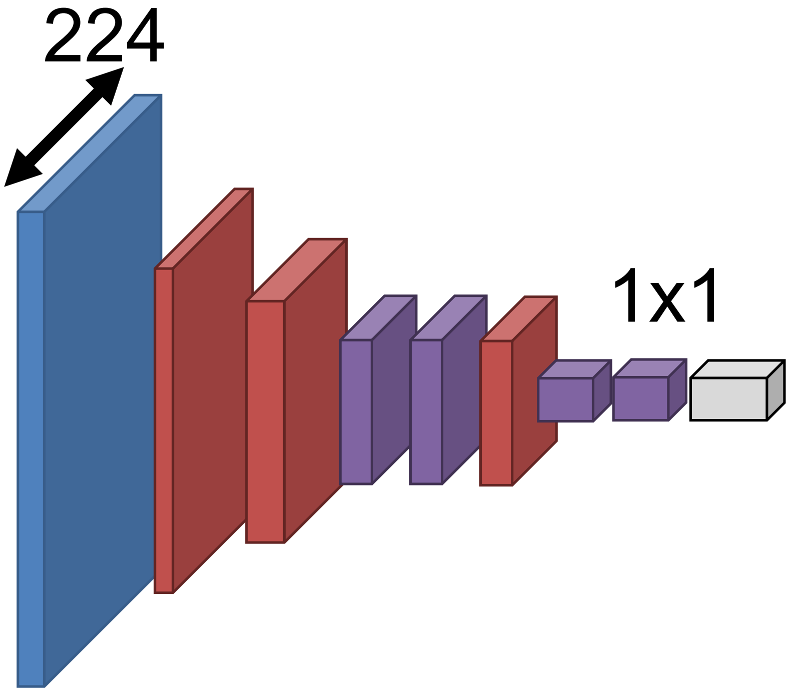
Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014

Where Do We Get Parameters?

Convnet that maps images to vectors



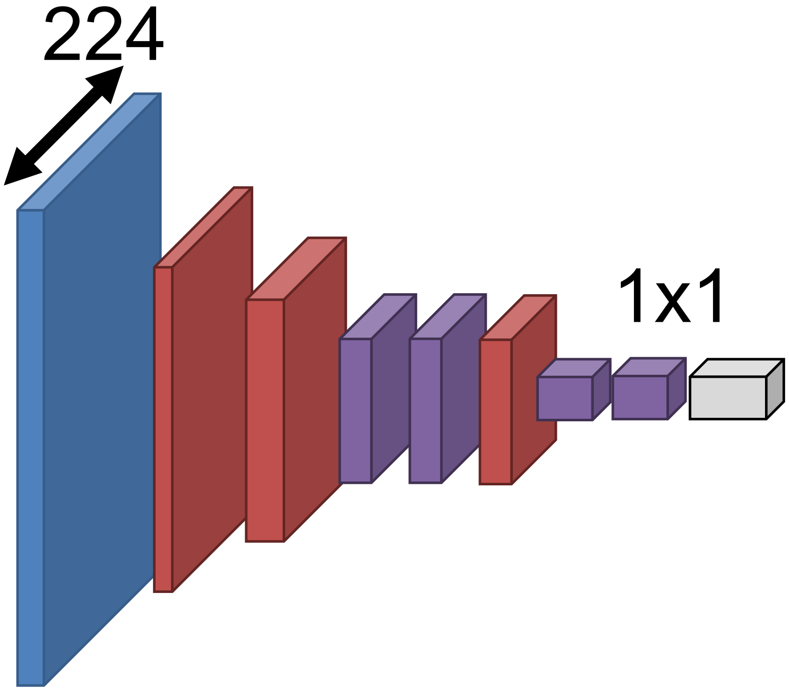
Convnet that maps images to images



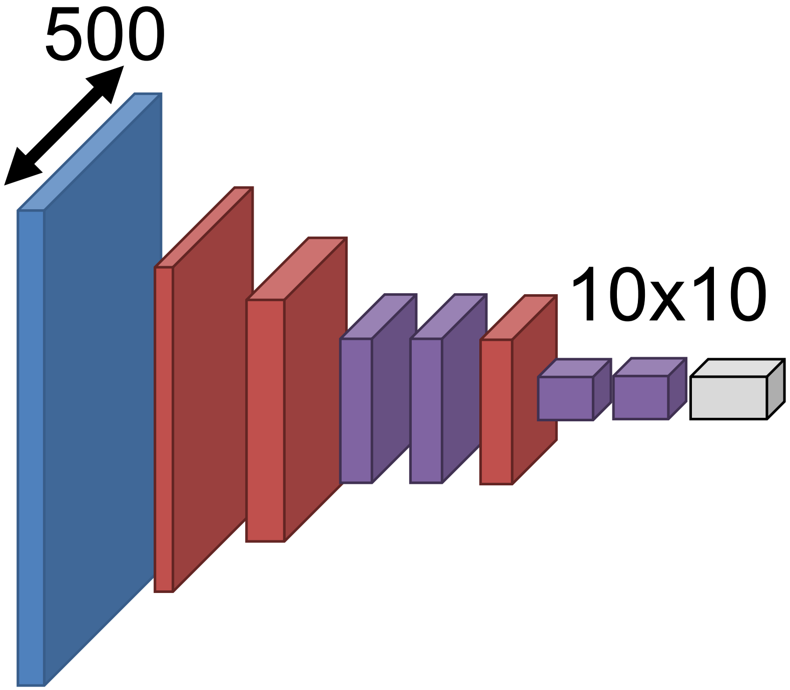
What if we make the input bigger?

Where Do We Get Parameters?

Convnet that maps images to vectors

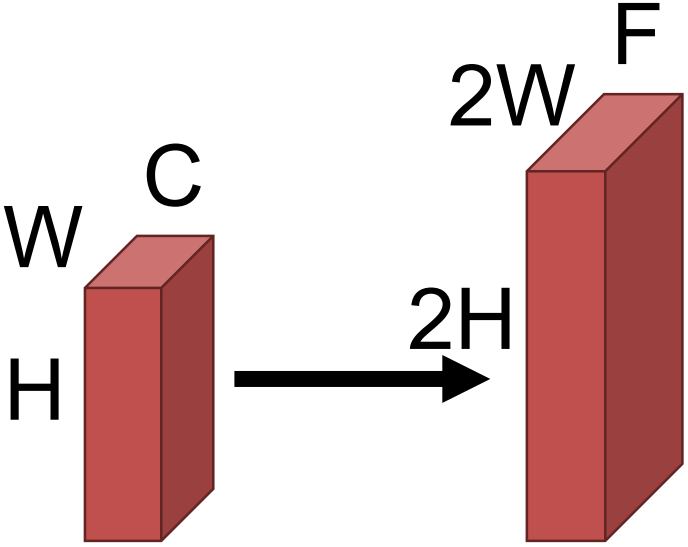


Convnet that maps images to images



Since it’s convolution, can reuse an image network

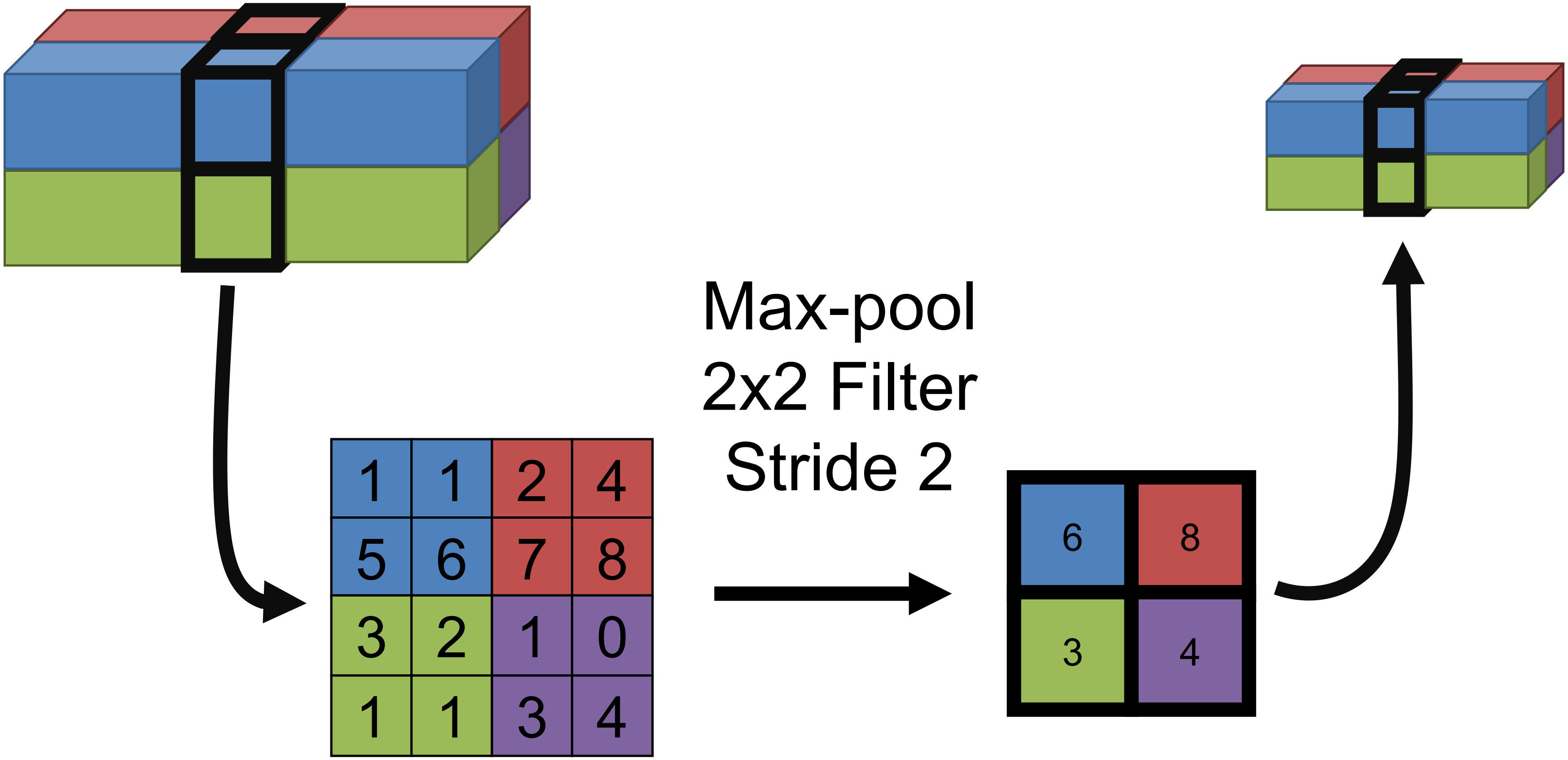
How Do We Upsample?



Do the opposite of how we downsample:

1. Pooling → “Unpooling”
2. Convolution → “Transpose Convolution”

Recall: Pooling

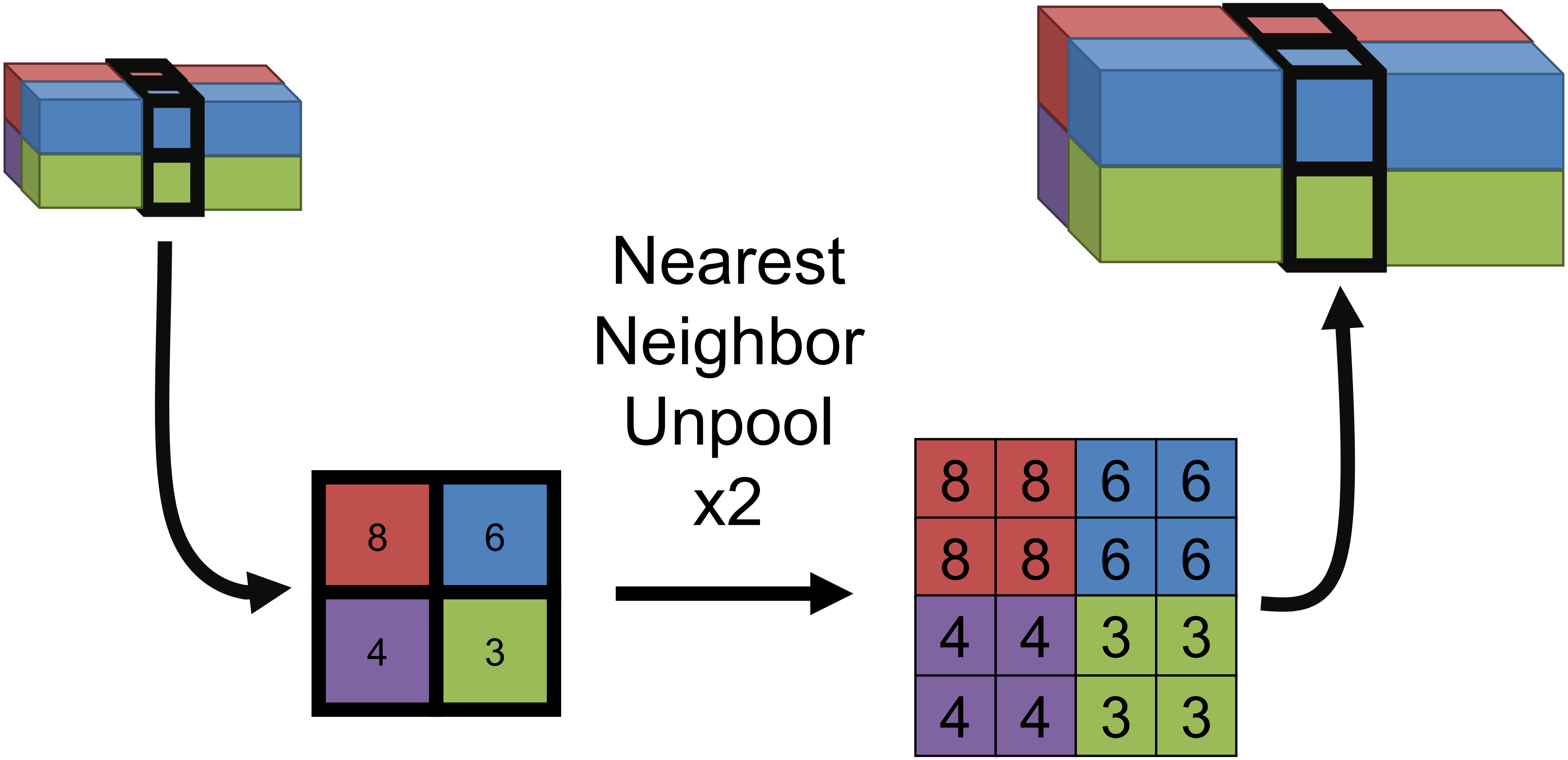


Max-pool
2x2 Filter
Stride 2

1	1	2	4
5	6	7	8
3	2	1	0
1	1	3	4

6	8
3	4

Now: Unpooling



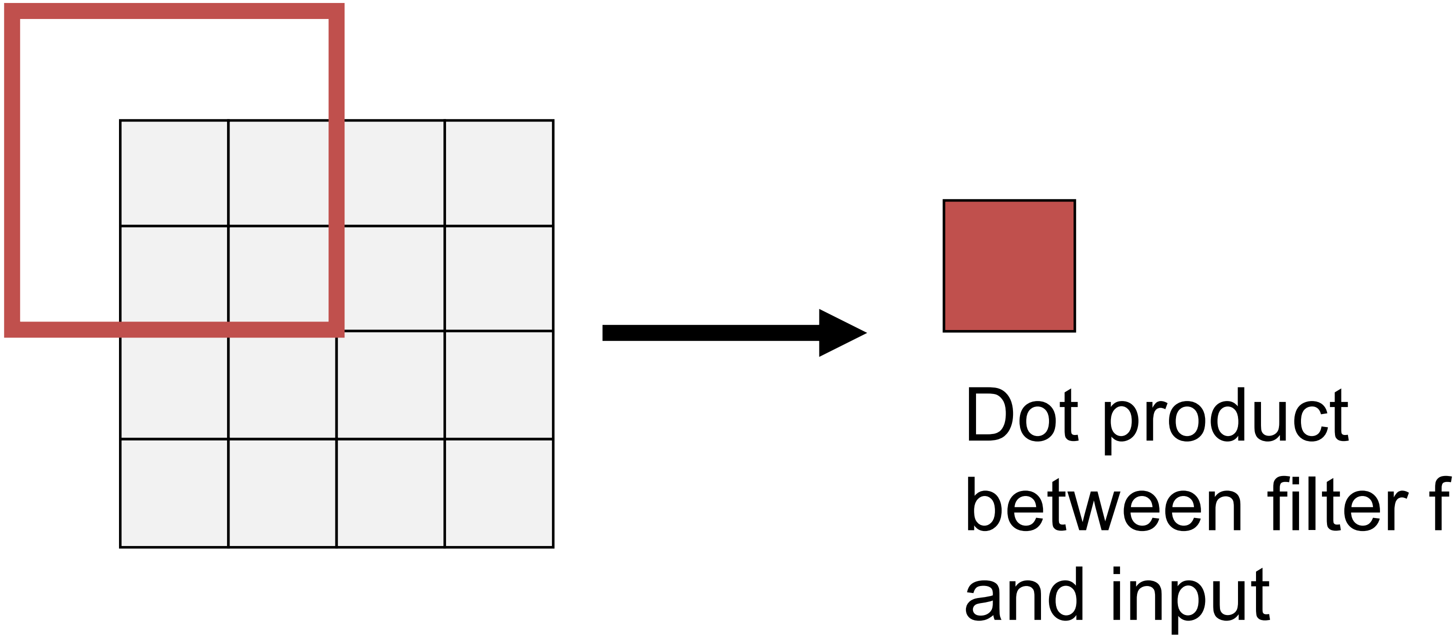
Nearest
Neighbor
Unpool
x2

8	6
4	3

8	8	6	6
8	8	6	6
4	4	3	3
4	4	3	3

Recall: Convolution

3x3 Convolution, Stride 2, Pad 1

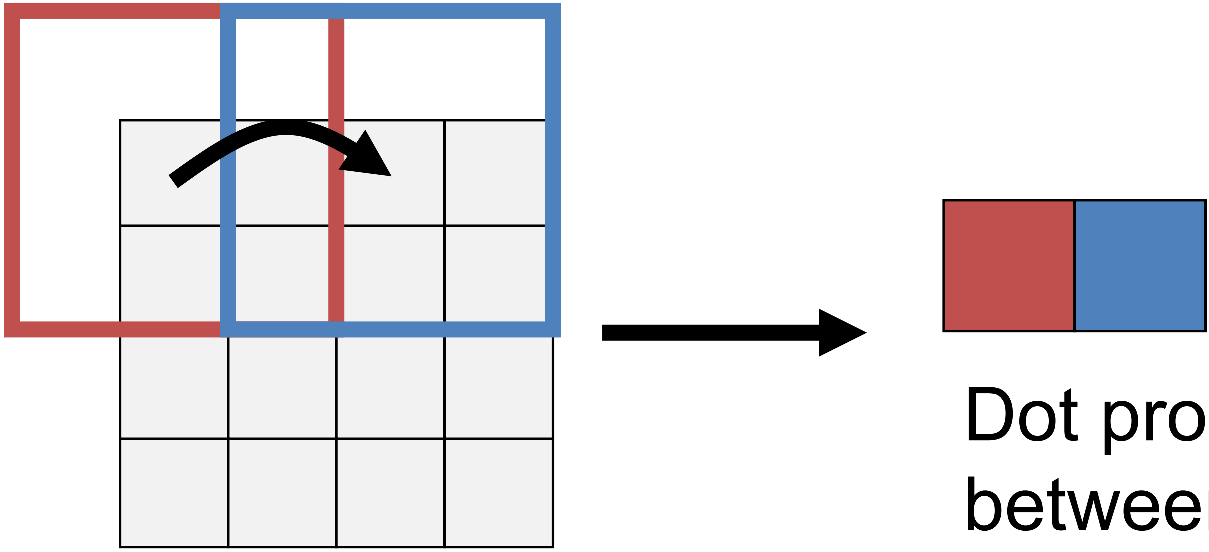


Dot product between filter f and input

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung

Recall: Convolution

3x3 Convolution, Stride 2, Pad 1

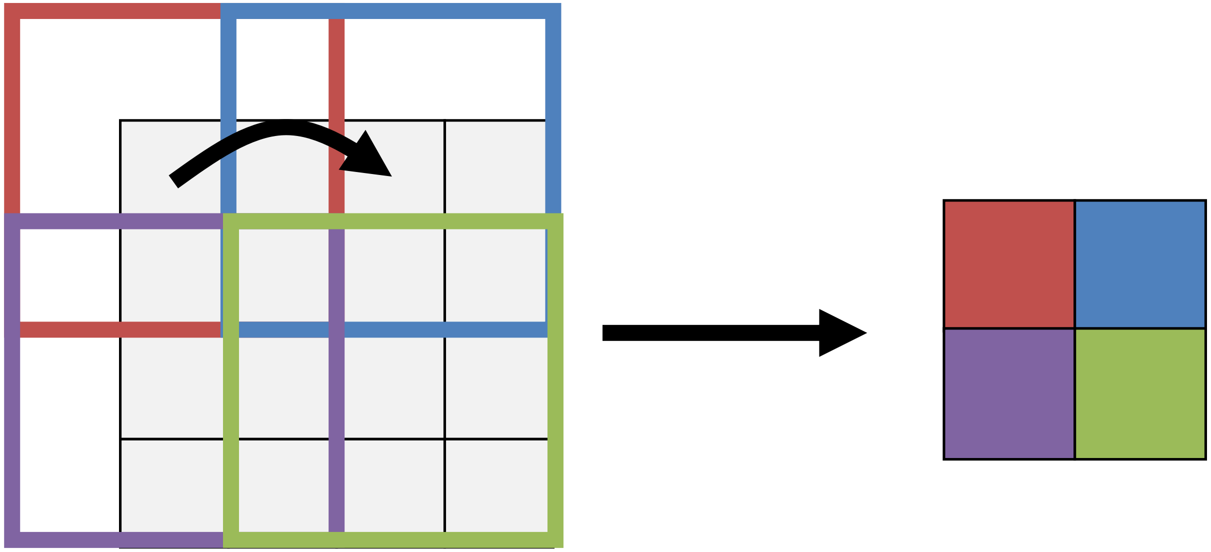


Dot product between filter f and input

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung

Recall: Convolution

3x3 Convolution, Stride 2, Pad 1

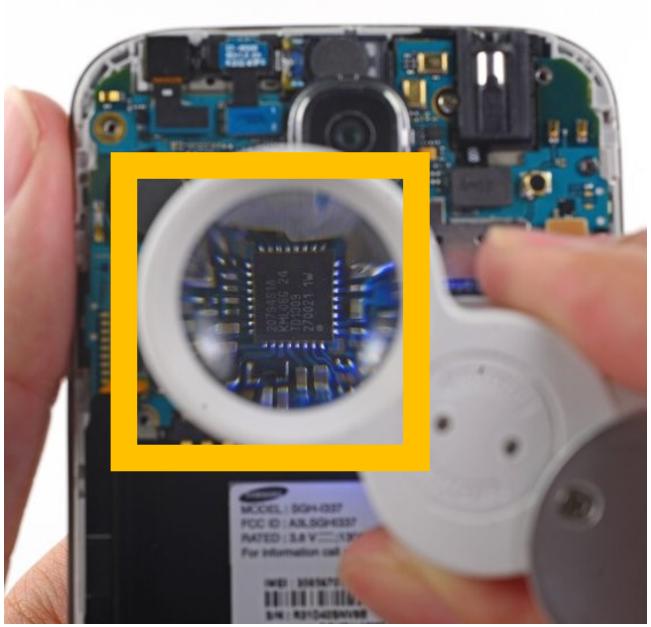


Example Credit: L. Fei-Fei, J. Johnson, S. Yeung

Transpose Convolution

Convolution

Filter: little lens that looks at a pixel.



Transpose Conv.

Filter: tiles used to make image

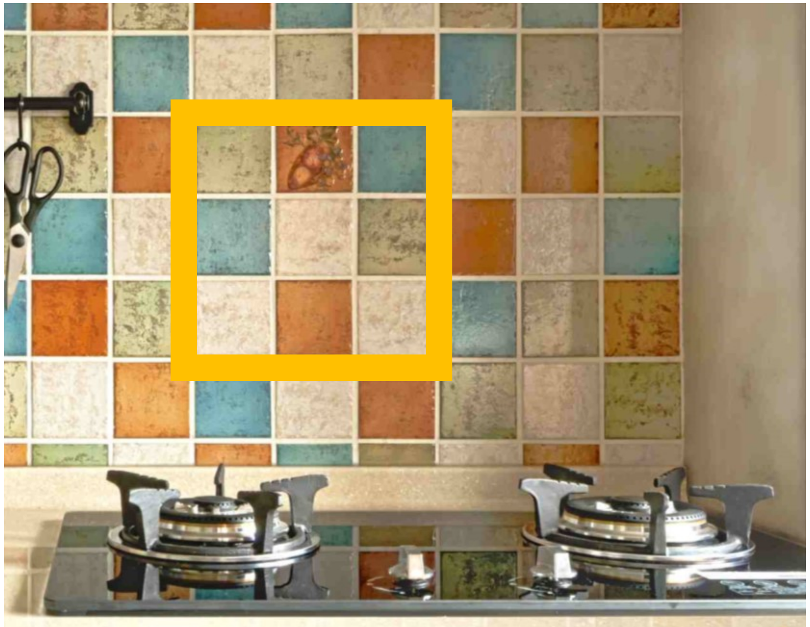
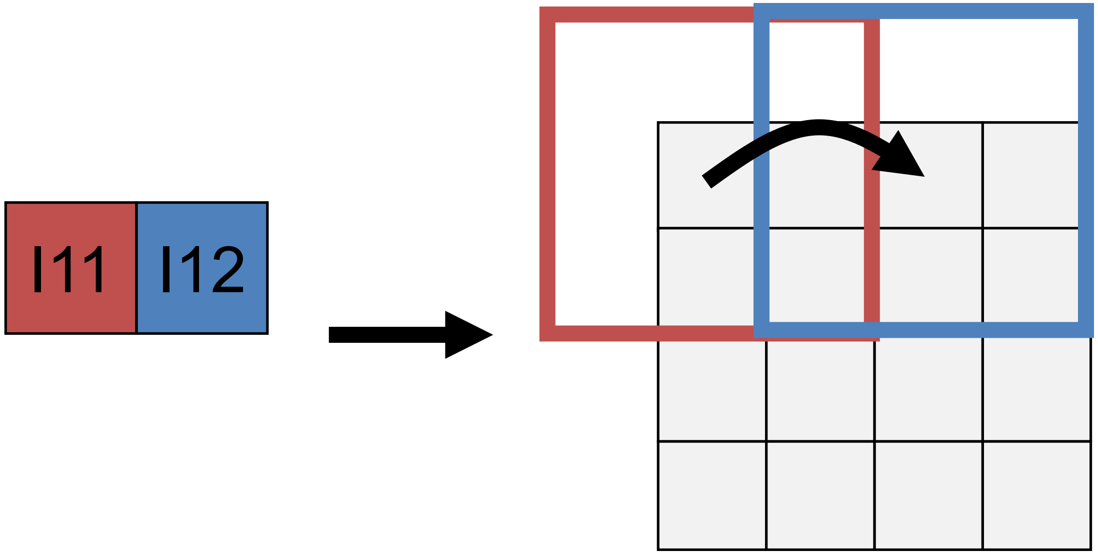


Image credit: ifixit.com, thespruce.com

Transpose Convolution

3x3 Transpose Convolution, Stride 2, Pad 1



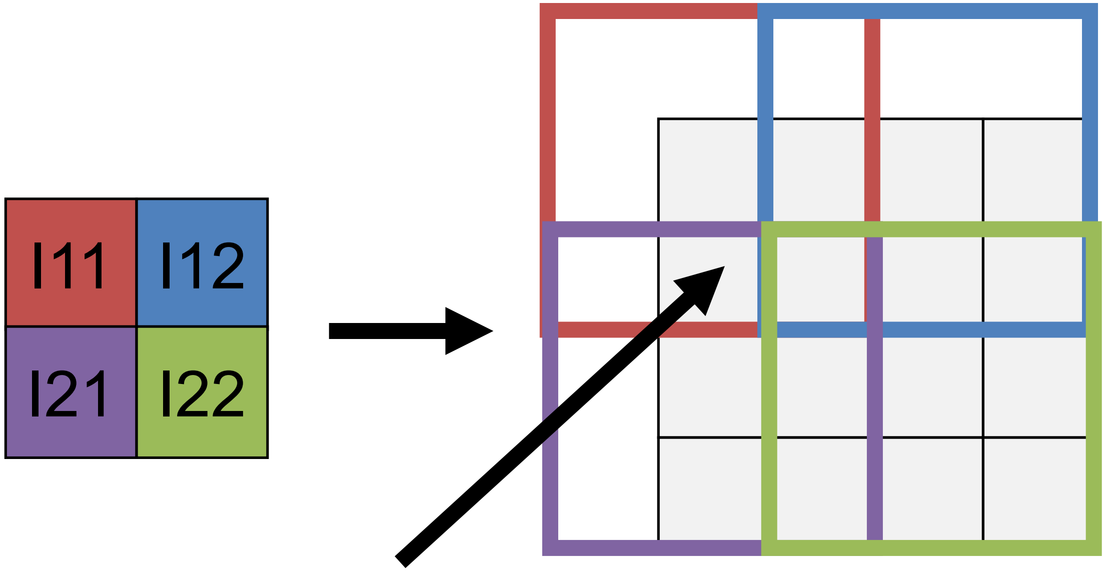
Output is filter F weighted by input I

$I_{11}\boldsymbol{F}$ $I_{12}\boldsymbol{F}$

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung

Transpose Convolution

3x3 Transpose Convolution, Stride 2, Pad 1



Output is filter F weighted by input I

$I_{11}F$ $I_{12}F$

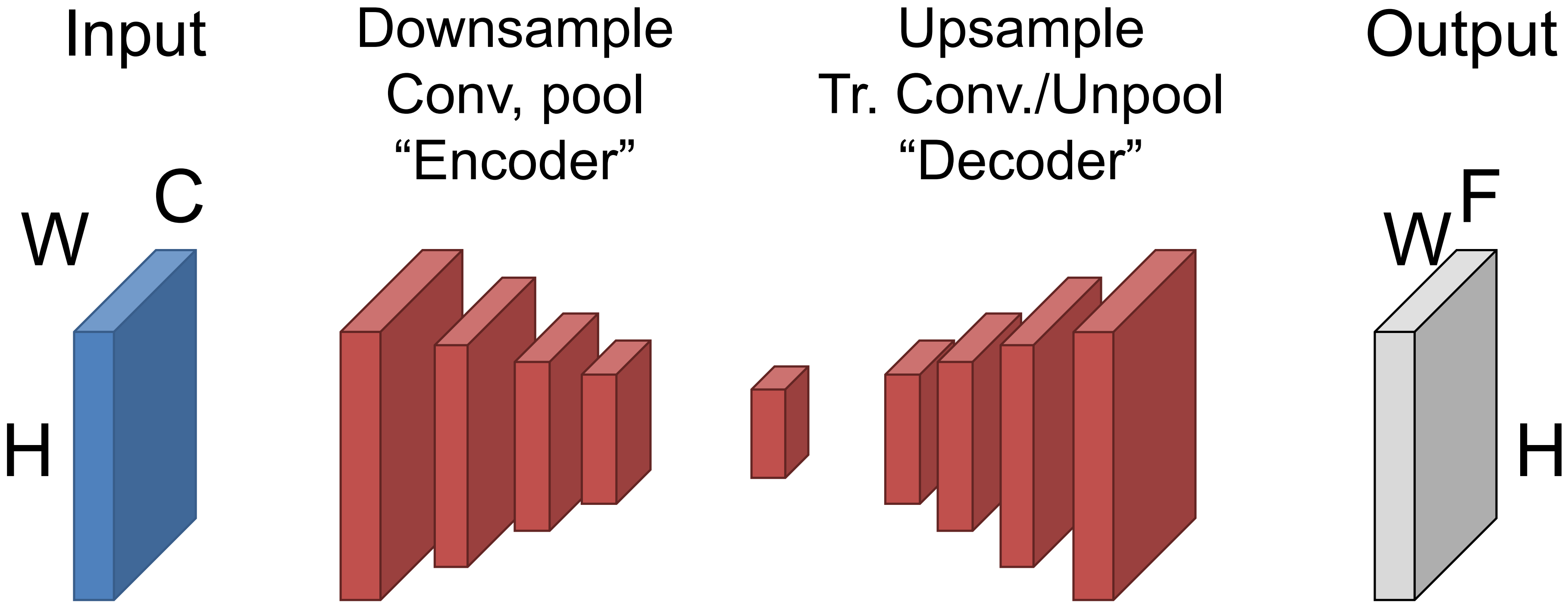
$I_{21}F$ $I_{22}F$

Sum outputs at overlap
(e.g., from $I_{11}F$ and $I_{21}F$)

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung

Putting it Together

Convolutions + pooling downsample/compress/encode
Transpose convs./unpoolings upsample/uncompress/decode



Input

Downsample
Conv, pool
“Encoder”

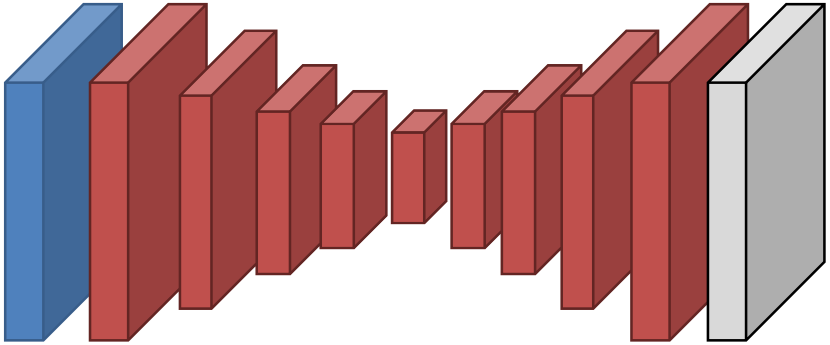
Upsample
Tr. Conv./Unpool
“Decoder”

Output

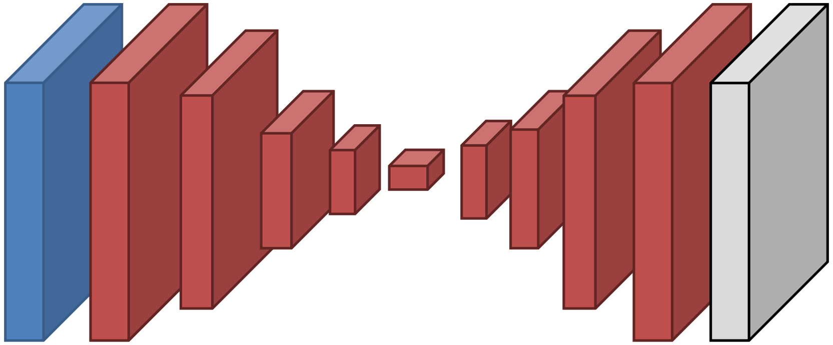
Putting It Together – Block Sizes

- Networks come in lots of forms
- **Don’t take any block sizes literally.**
- Often (not always) keep some spatial resolution

Encode to spatially smaller tensor, then decode.

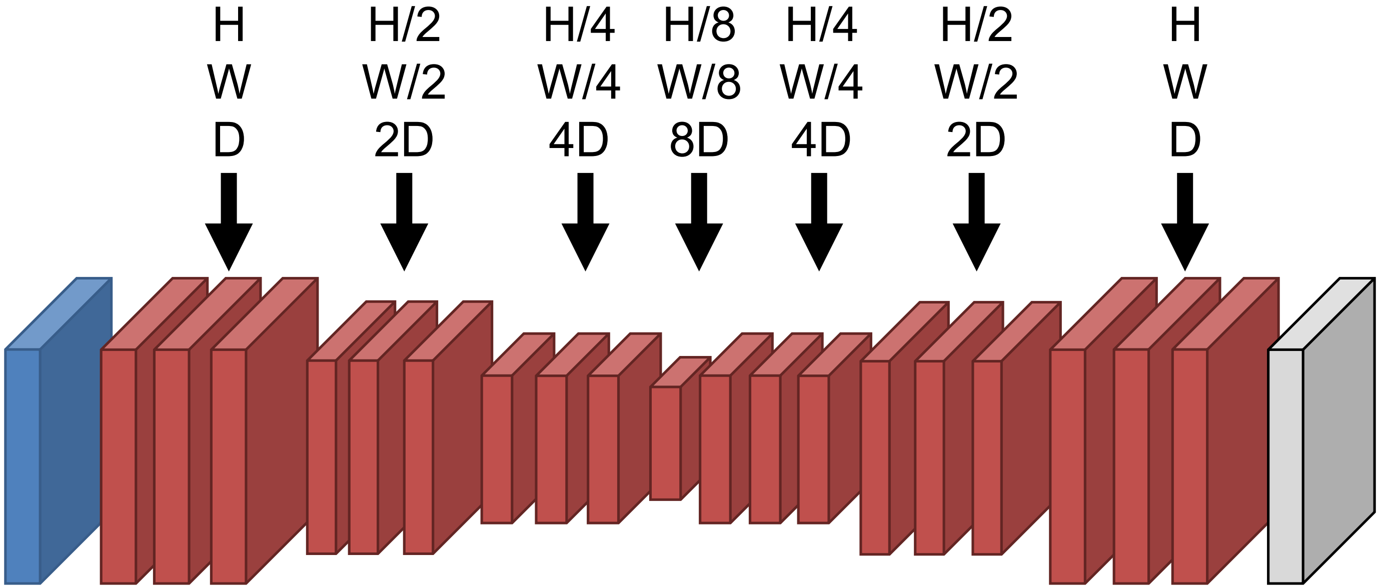


Encode to 1D vector then decode



Putting It Together – Block Sizes

- Often multiple layers at each spatial resolution.
  - Often halve spatial resolution and double feature depth every few layers



H W D

H/2 W/2 2D

H/4 W/4 4D

H/8 W/8 8D

H/4 W/4 4D

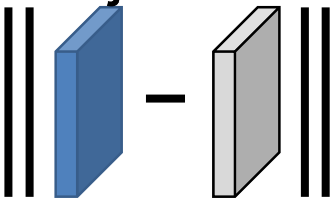
H/2 W/2 2D

H W D

An Aside: Autoencoders

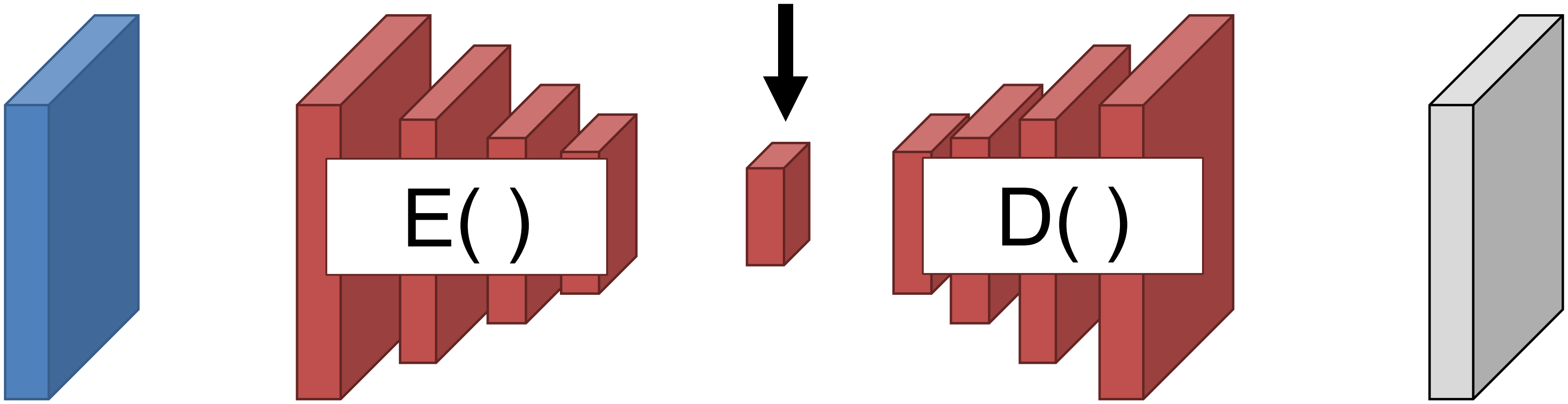
Network compresses input to “bottleneck”, decodes it back to input.

Objective:

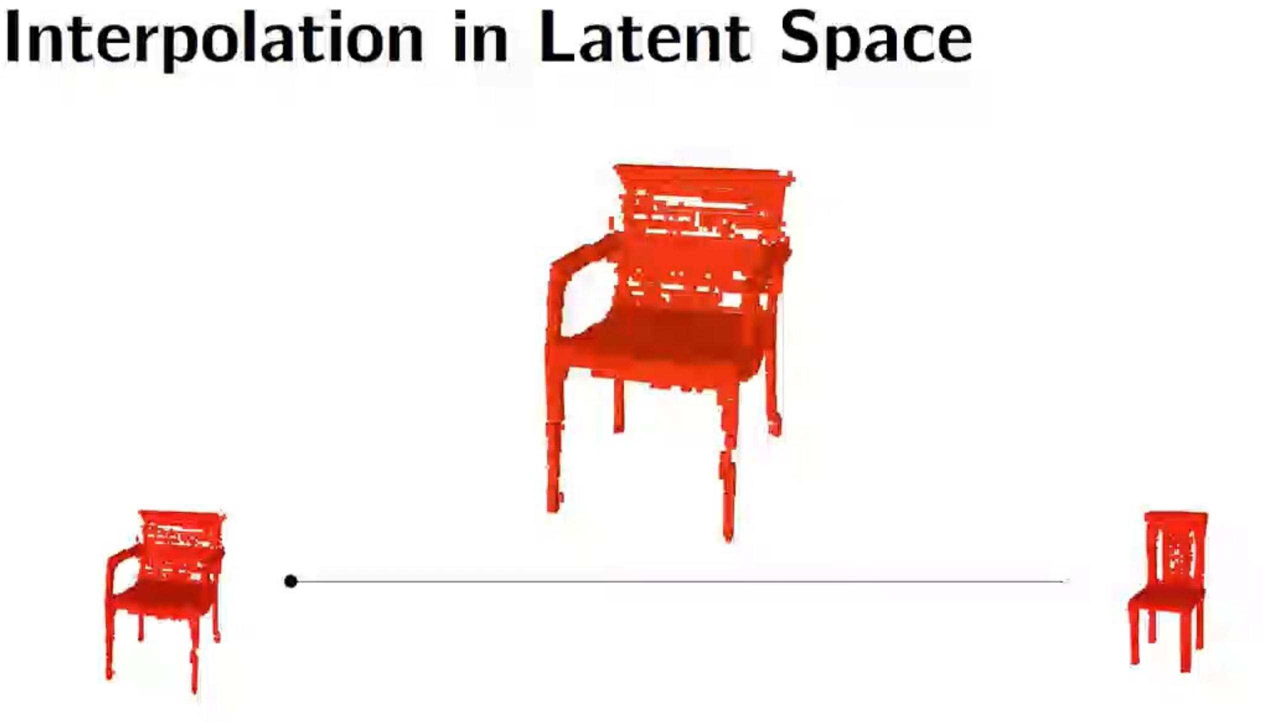


$$\|D(E(\boldsymbol{X})) - \boldsymbol{X}\|$$

Bottleneck/
Latent Space/
Latent Code



Walking the Latent Space*



*In the interest of honesty in advertising: not an autoencoder, but a similar method with the same goal of learning a latent space

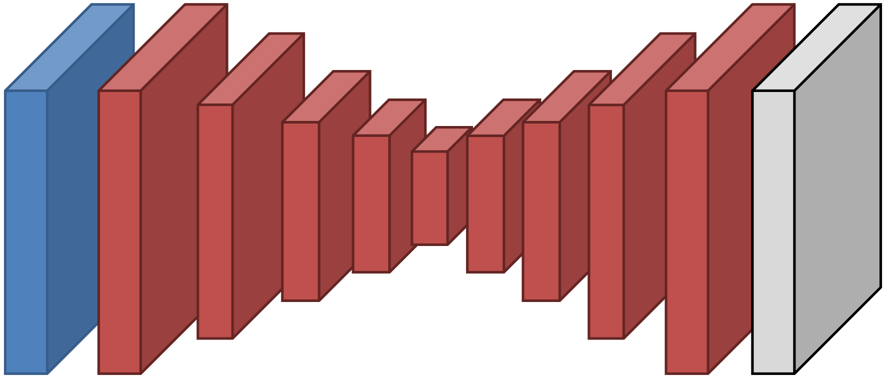
Result from Wu et al. *Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*. NIPS 2016

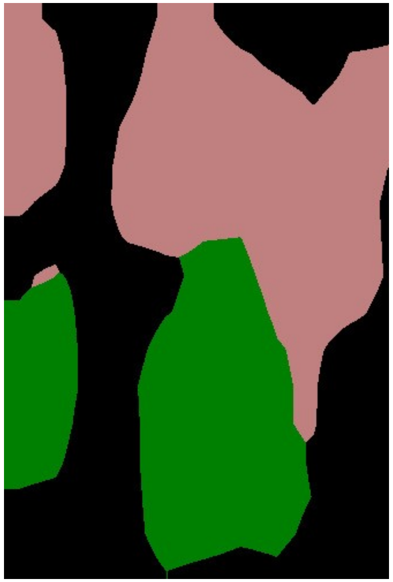
Missing Details

While the output *is* HxW, just upsampling often produces results without details/not aligned with the image.

Why?

Information about details lost when downsampling!

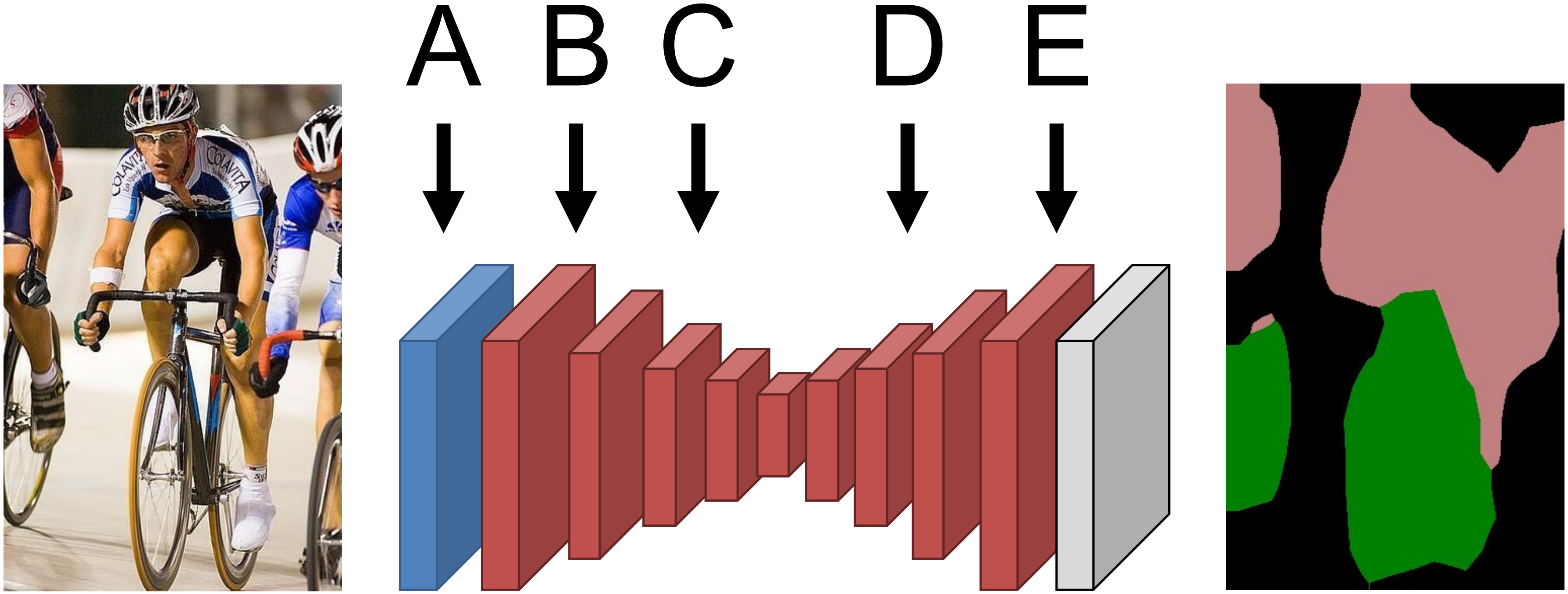




Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014

Missing Details

Where is the useful information about the high-frequency details of the image?



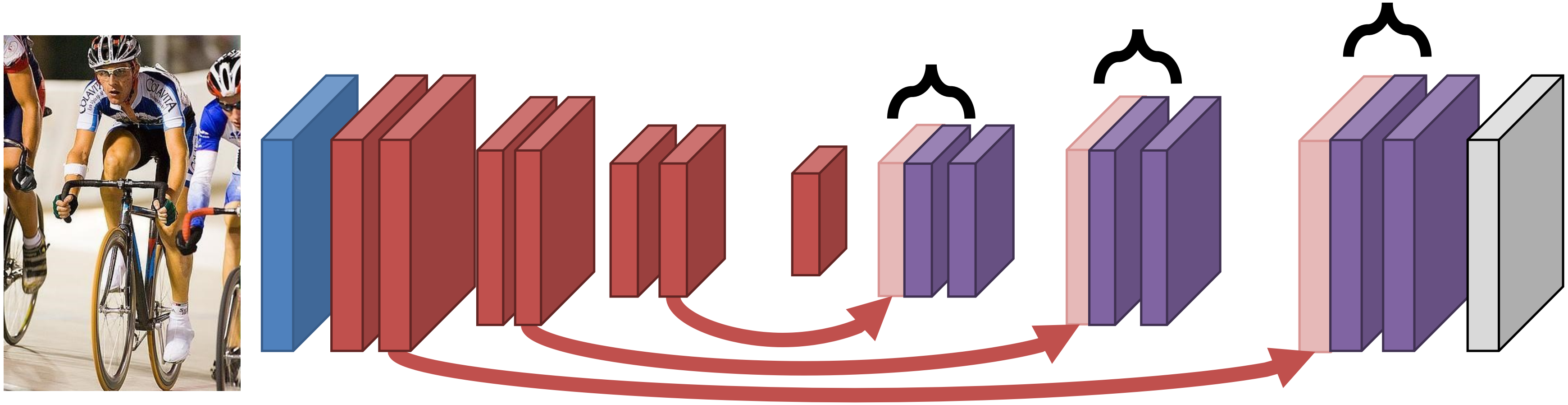
Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014

Missing Details

How do you send details forward in the network?

You copy the activations forward.

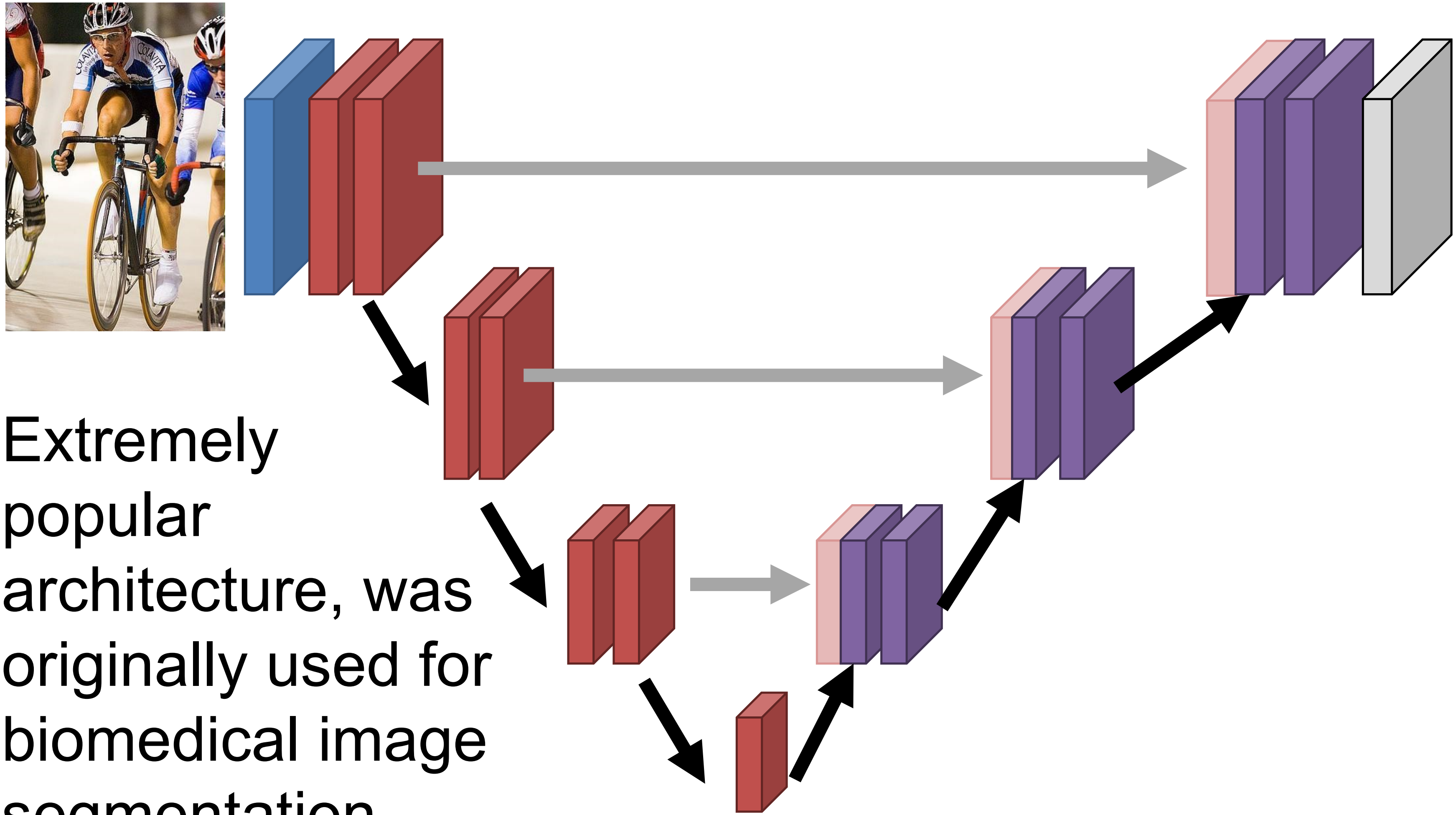
Subsequent layers at the same resolution figure out how to fuse things.



Copy

Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014

U-Net



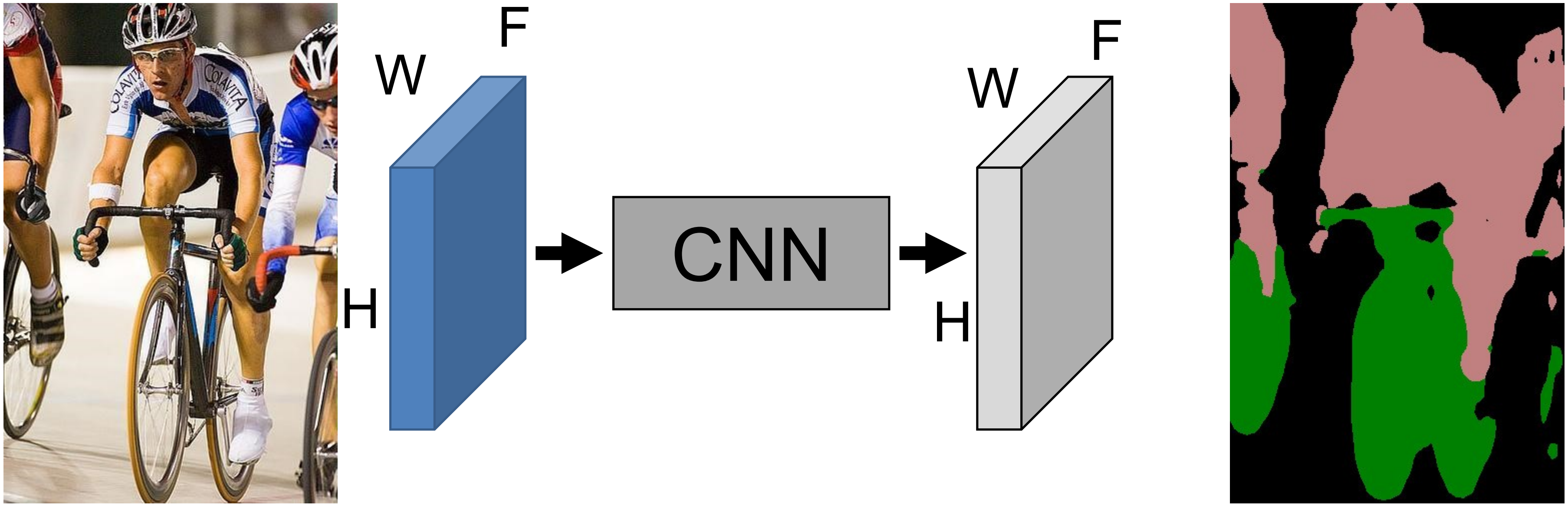
Extremely popular architecture, was originally used for biomedical image segmentation.

Ronneberger et al. “U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI 2015

Evaluating Pixel Labels

Input Image

Predicted Classes



How do we convert final HxWxF into labels?

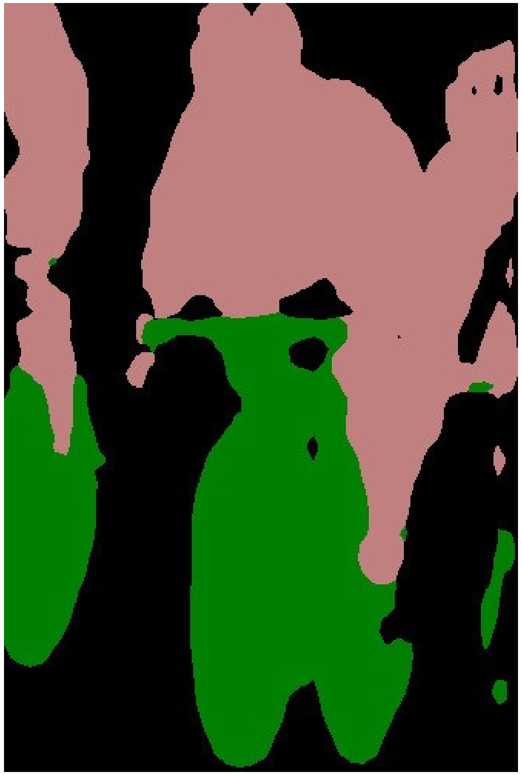
argmax over labels

Evaluating Semantic Segmentation

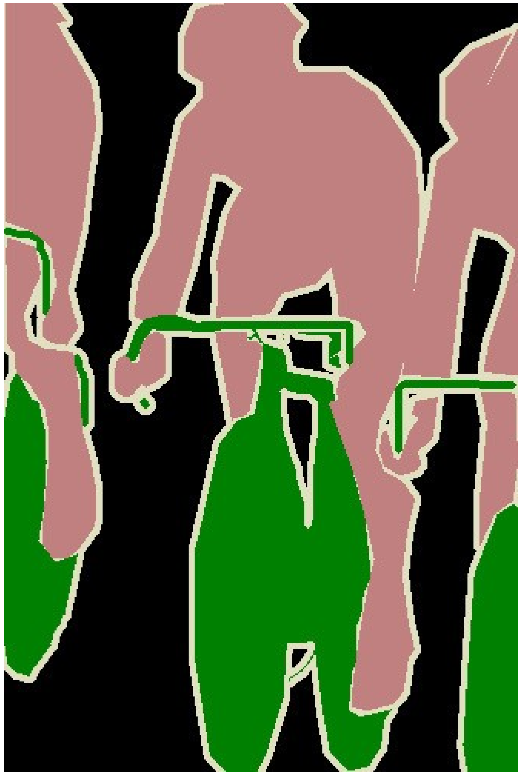
Given predictions, how well did we do?

Input

Prediction ($\hat{y}$)



Ground-Truth (y)

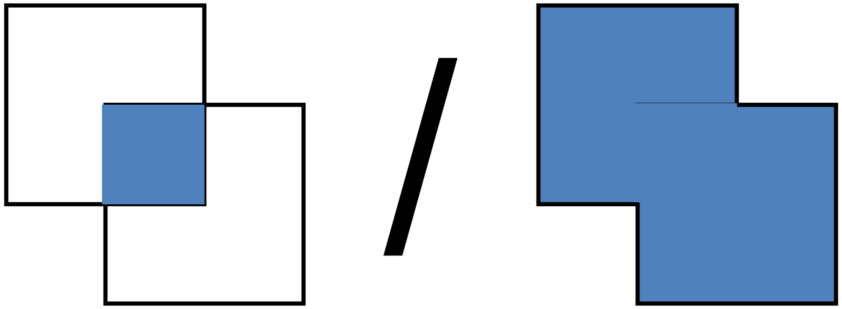


Evaluating Semantic Segmentation

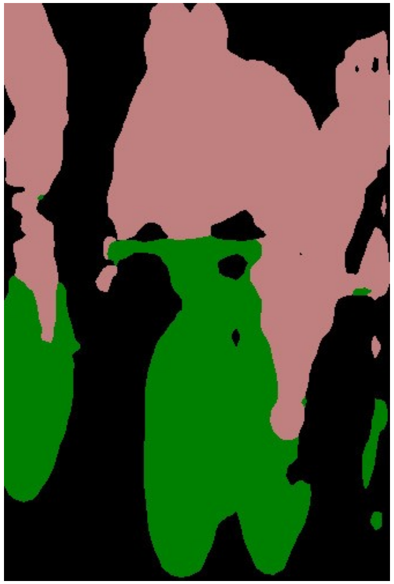
Prediction and ground-truth are images where each pixel is one of F classes.

Accuracy: mean($\hat{\boldsymbol{y}} = y$)

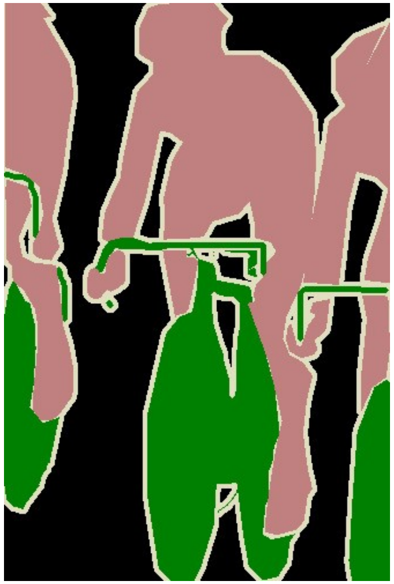
Intersection over union, averaged over classes



Prediction ($\hat{\boldsymbol{y}}$)



Ground-Truth ($\boldsymbol{y}$)



Next Time

- Detecting Objects (drawing boxes around them)
- I’ll have some extra time. Should I cover:
    1. How sketches to cats works?
    2. How you can learn features from the data itself?
    3. How you can learn to identify people?

More Info

Why “Transpose Convolution”?

Can write convolution as matrix-multiply
Input: 4, Filter: 3, Stride: 1, Pad: 1

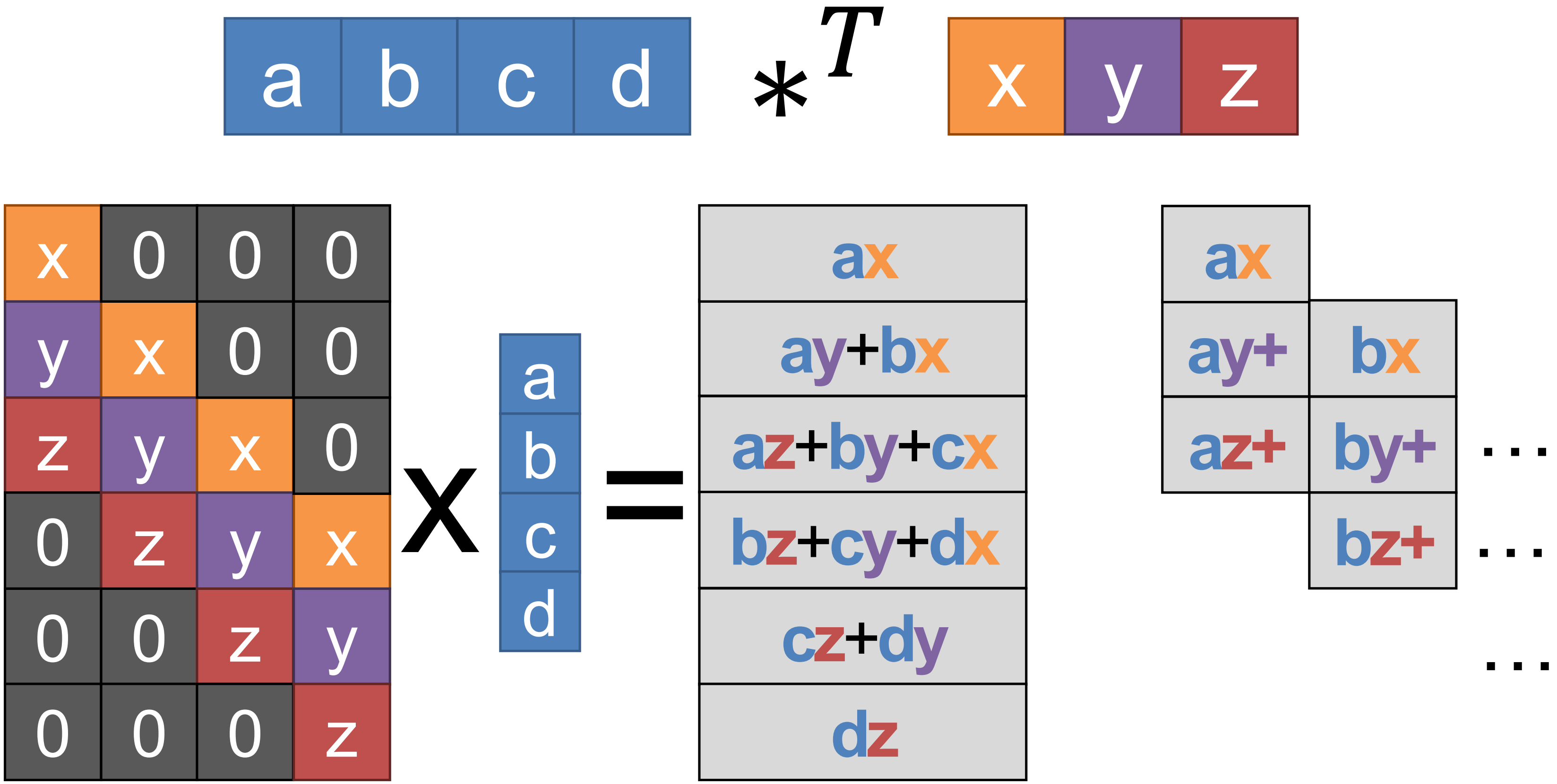
$$\begin{bmatrix} a & b & c & d \end{bmatrix} * \begin{bmatrix} x & y & z \end{bmatrix}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \times \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay+bz \\ ax+by+cz \\ bx+cy+dz \\ cx+dy \end{bmatrix}$$

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung

Why “Transpose Convolution”?

Transpose convolution is convolution transposed



$$\begin{bmatrix} a & b & c & d \end{bmatrix} *^{T} \begin{bmatrix} x & y & z \end{bmatrix}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay+bx \\ az+by+cx \\ bz+cy+dx \\ cz+dy \\ dz \end{bmatrix}$$

$$\begin{array}{ccc} ax & & \\ ay+ & bx & \\ az+ & by+ & \cdots \\ & bz+ & \cdots \\ & & \cdots \end{array}$$

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung