CS 4803 / 7643: Deep Learning

Topics:

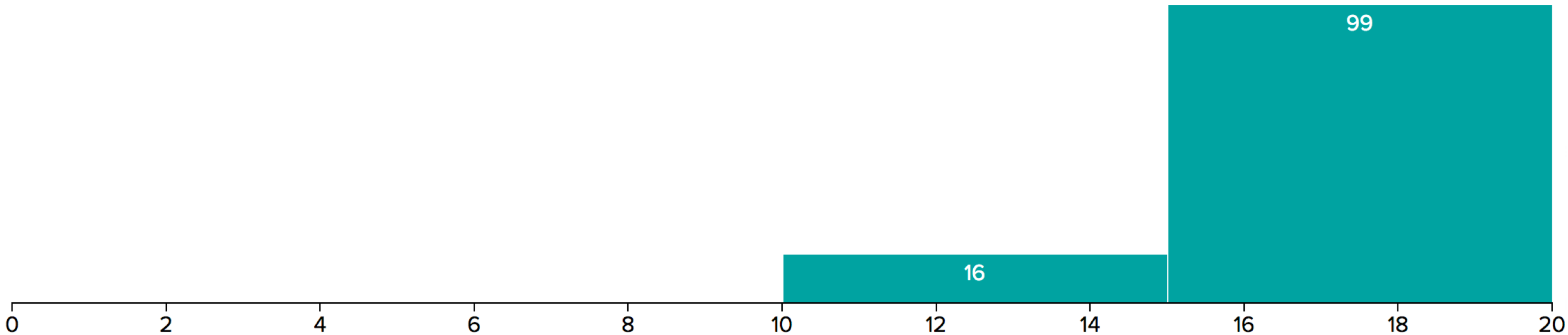
- Convolutional Neural Networks
  - (Finish) Backprop in conv layers
  - Toeplitz matrices and convolutions = matrix-mult
  - Dilated/a-trous convolutions
  - Transposed convolutions

Dhruv Batra

Georgia Tech

Administrativia

- HW0 Grades Released
  - 1 week to talk to TAs about any concerns.



MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
11.6	18.15	20.0	17.55	2.11

Administrativia

- HW1 Analysis
  - https://evalai.cloudcv.org/web/challenges/challenge-page/132/leaderboard/377
  - https://docs.google.com/spreadsheets/d/1k-ePFv77CUMhnzLFvVmR2k9JP0MYLX3eaLQS84vsVtI/edit#gid=1412360458
  - Overfitting plots

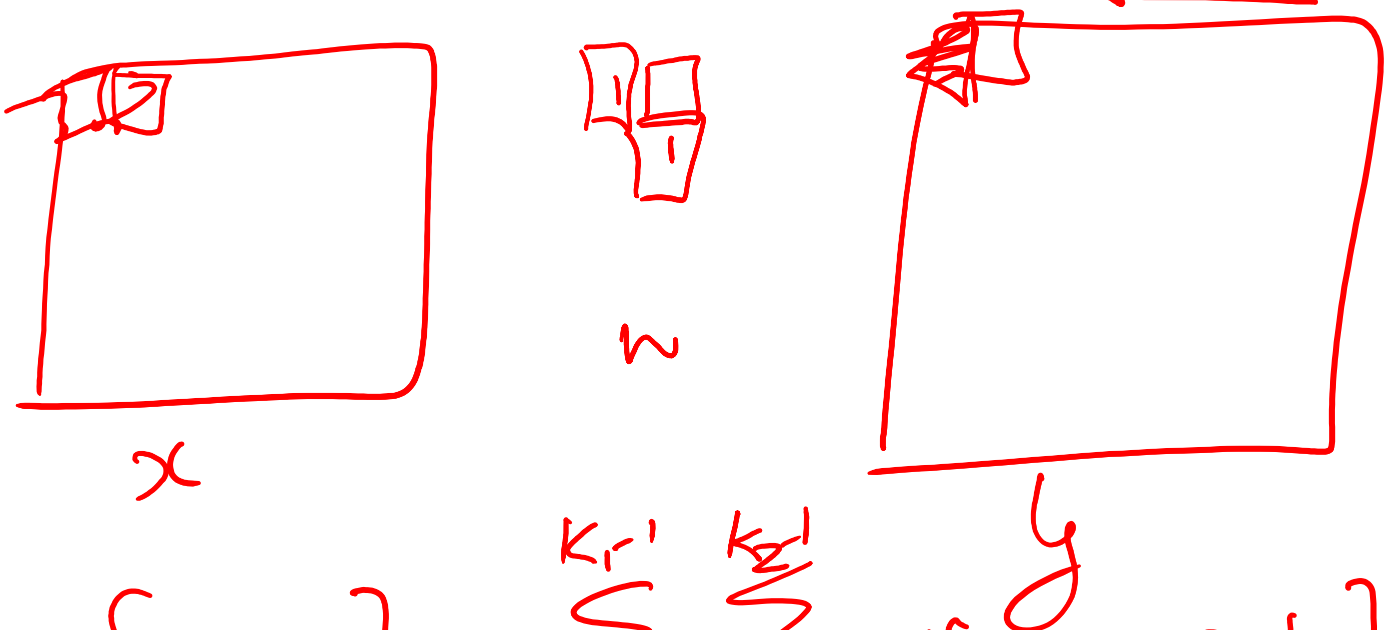
Administrativia

- HW2 Released
  - Due: 10/18, 11:55pm
  - https://www.cc.gatech.edu/classes/AY2019/cs7643_fall/assets/hw2.pdf

- Project Sign-up
  - Due: 10/05
  - https://reproducibility-challenge.github.io/iclr_2019/
  - https://docs.google.com/spreadsheets/d/1BipWLvvWb7Fu6OSDd-uOCF1Lr_4drKOCRVdhxm_eSHc/edit#gid=0

Recap from last time

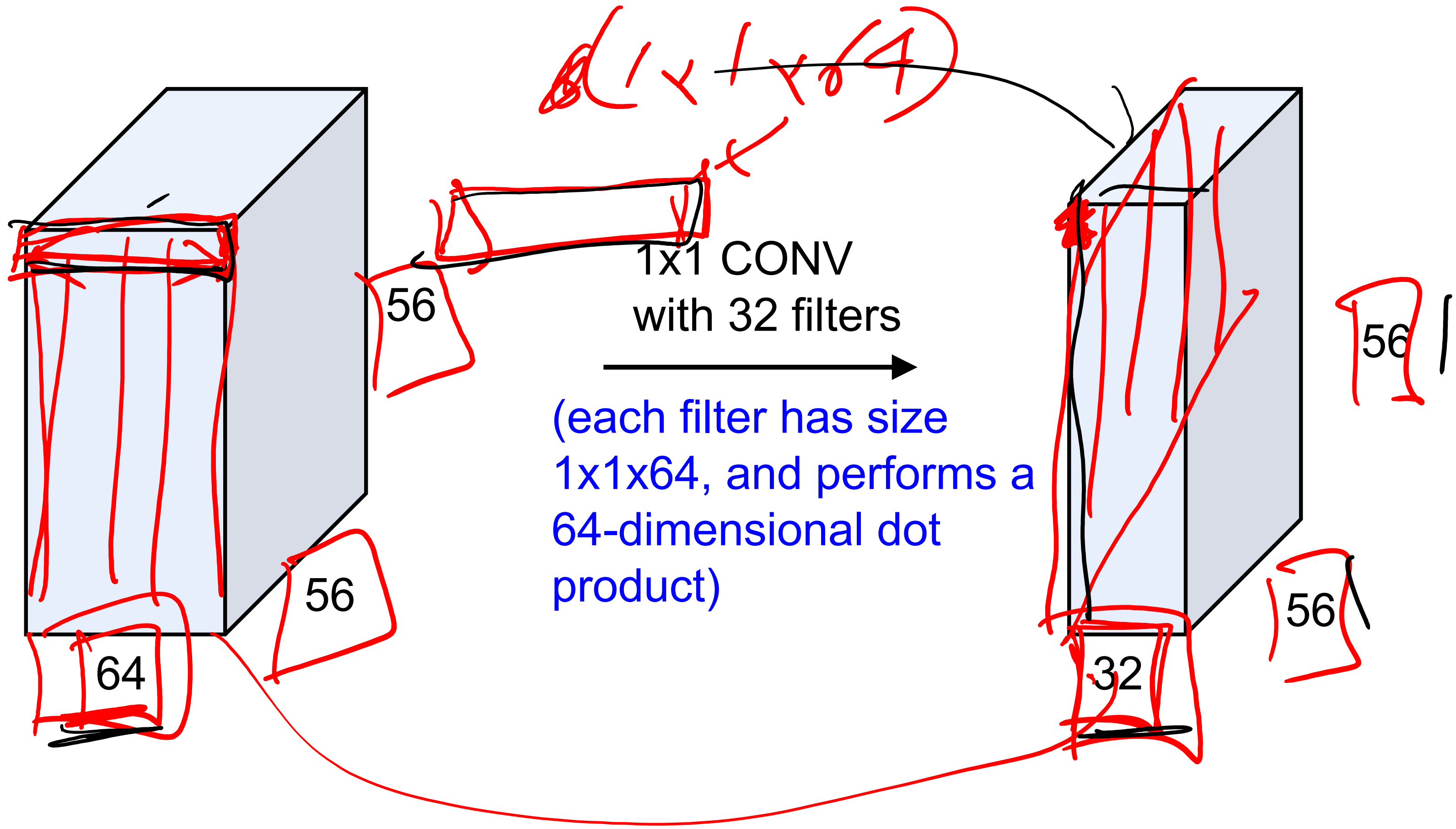
Can we have $\boxed{1\text{x}1}$ filters?



$$y[r,c] = \sum_{a=0}^{K_1-1} \sum_{b=0}^{K_2-1} x[r+a, c+b]\, w[a,b]$$

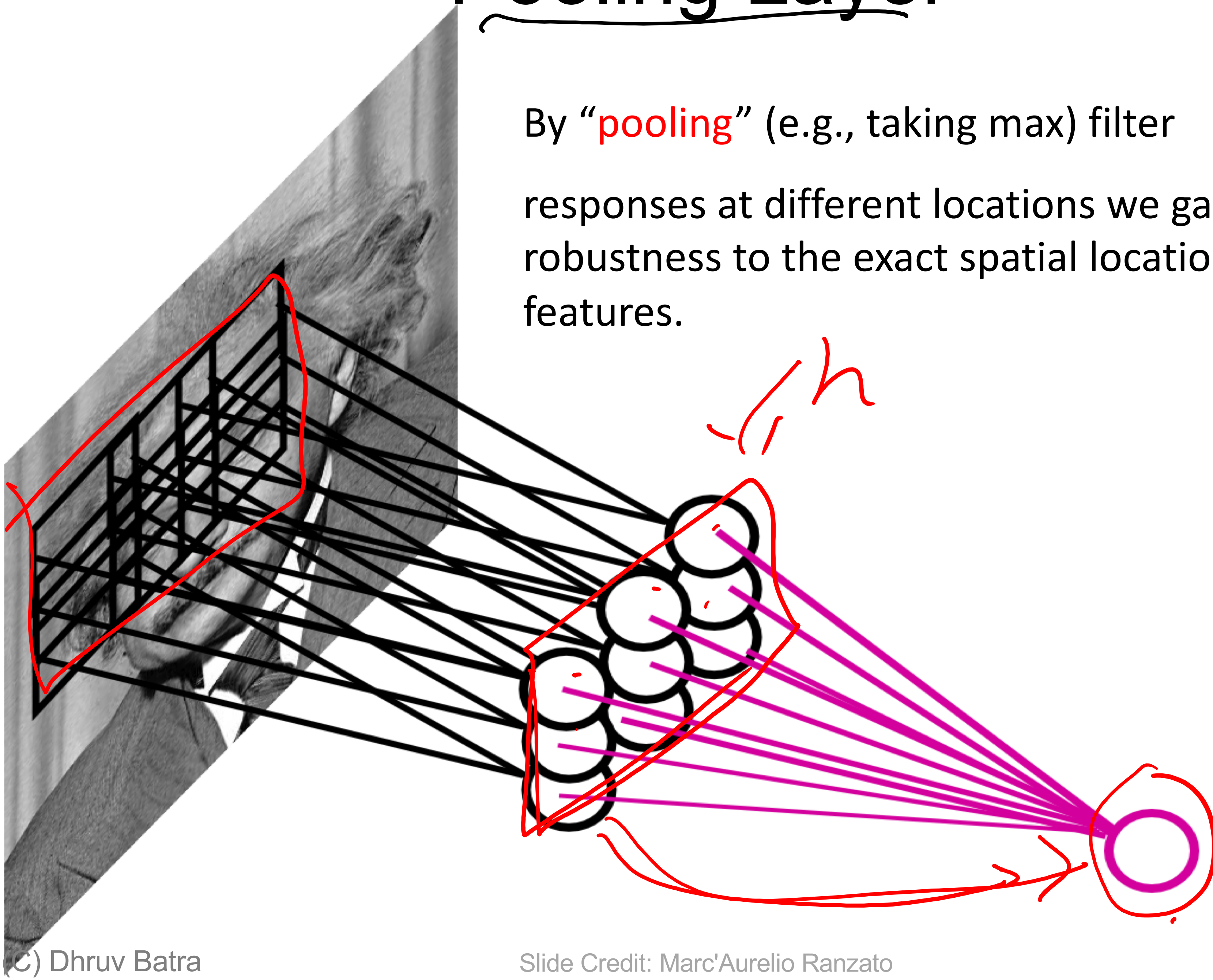
$$\boxed{y[r,c] = x[r,c]} \cdot w[0,0]$$

(btw, 1x1 convolution layers make perfect sense)



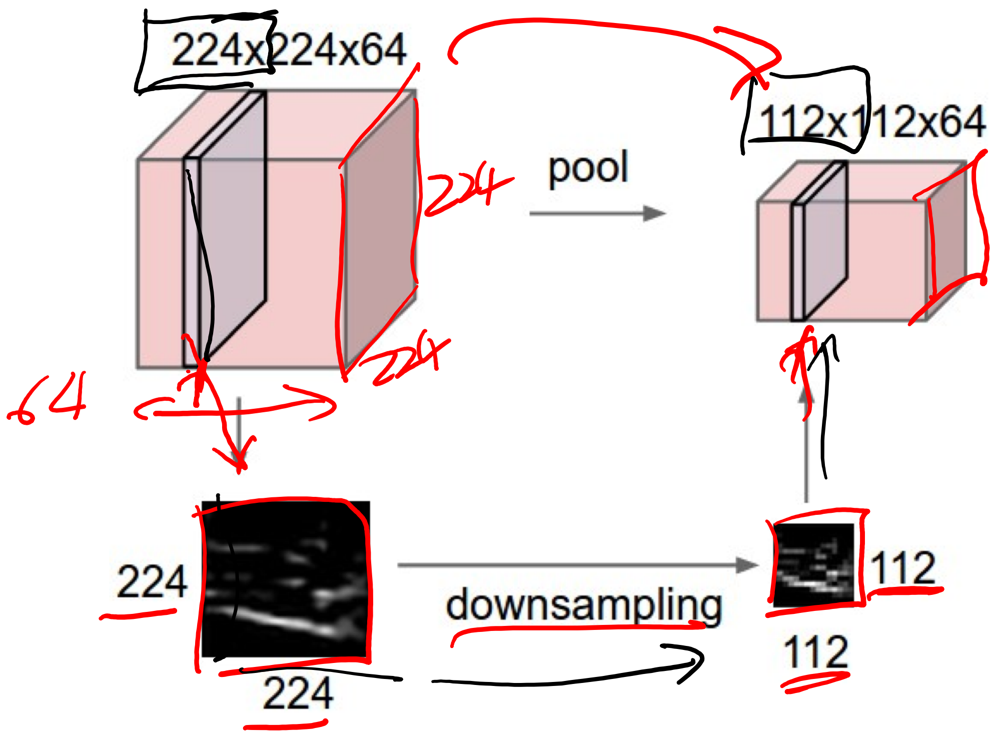
Pooling Layer

By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.



Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

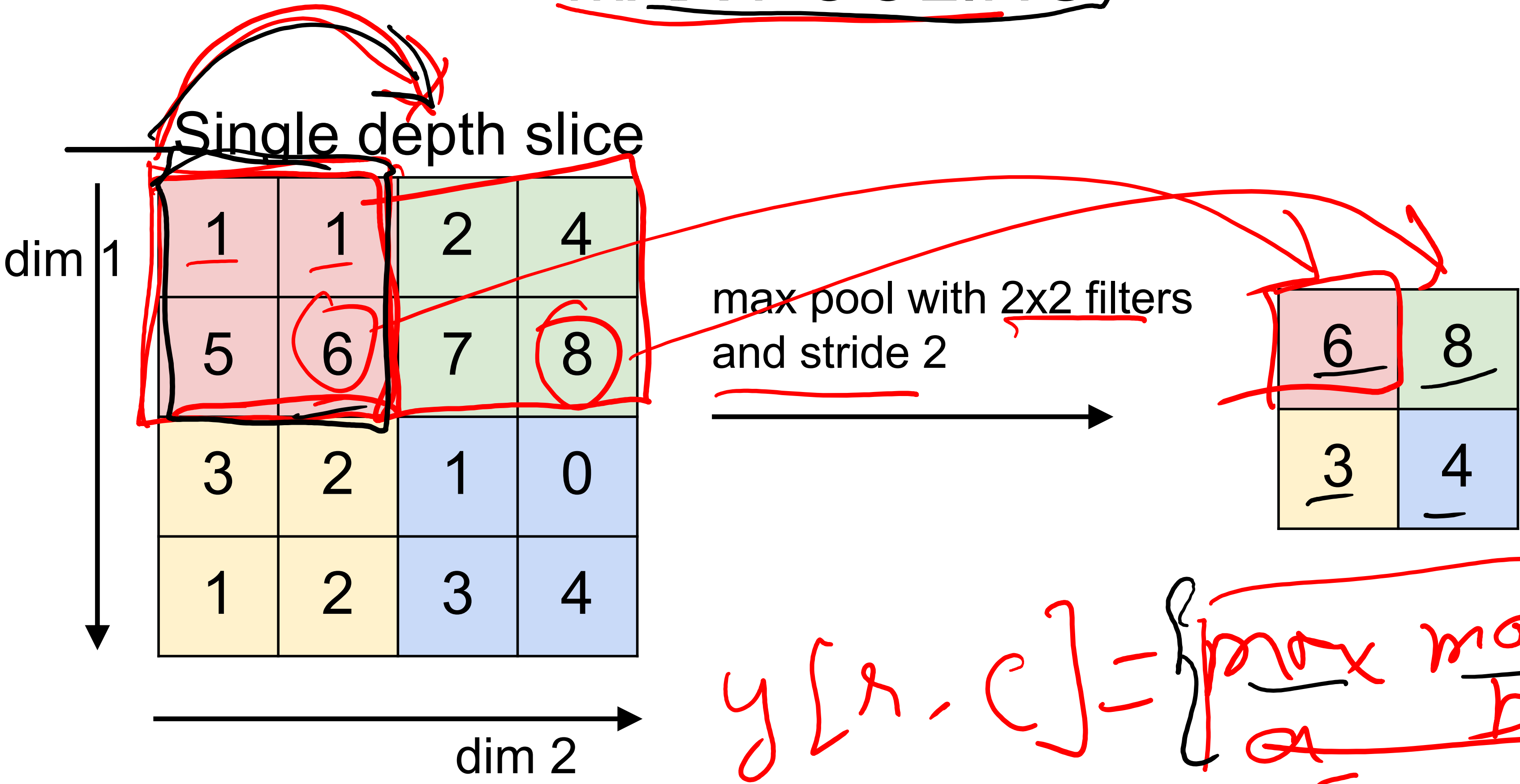


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

stride=2

MAX POOLING

2x2



$$y[r,c] = \max_{a} \max_{b} x[r+a, c+b]$$

1	3	2	9
7	4	1	5
8	5	2	3
4	2	1	4

Pooling Layer: Examples

Max-pooling:

$$h_i^n(r,c) = \boxed{\max_{\bar{r}\in N(r),\ \bar{c}\in N(c)}} \boxed{h_i^{n-1}(\bar{r},\bar{c})}$$

Average-pooling:

$$h_i^n(r,c) = \boxed{\operatorname*{mean}_{\bar{r}\in N(r),\ \bar{c}\in N(c)}} \boxed{h_i^{n-1}(\bar{r},\bar{c})}$$

L2-pooling:

$$h_i^n(r,c) = \sqrt{\sum_{\bar{r}\in N(r),\ \bar{c}\in N(c)} h_i^{n-1}(\bar{r},\bar{c})^2}$$

Classical View

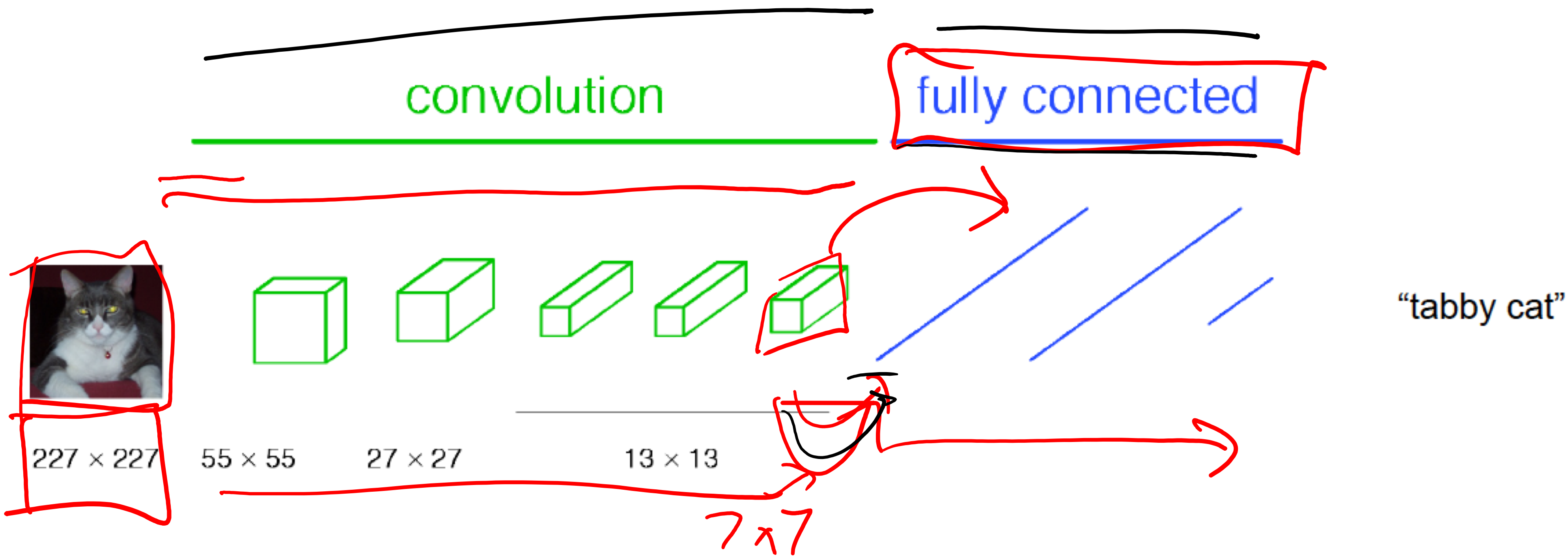
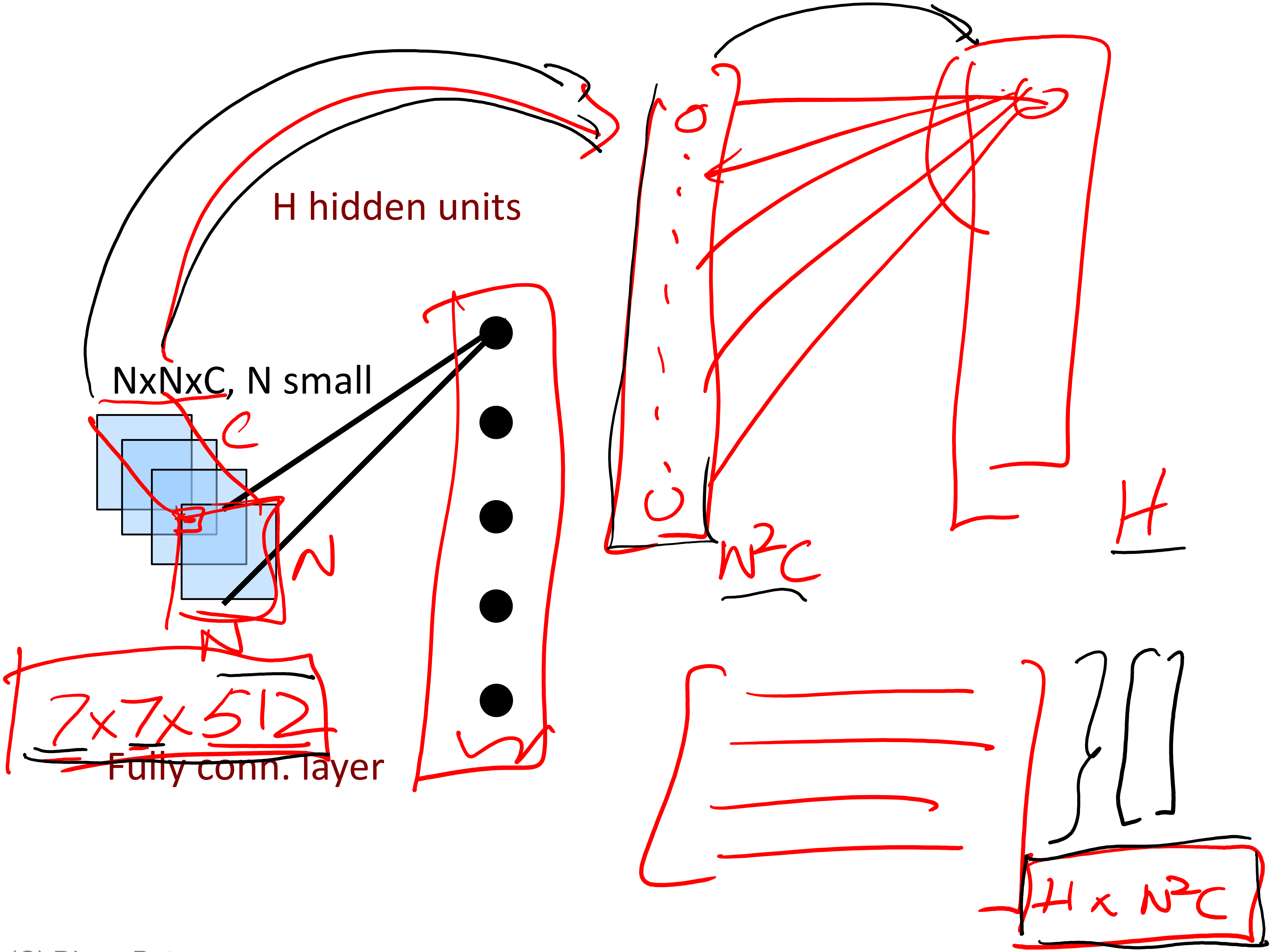
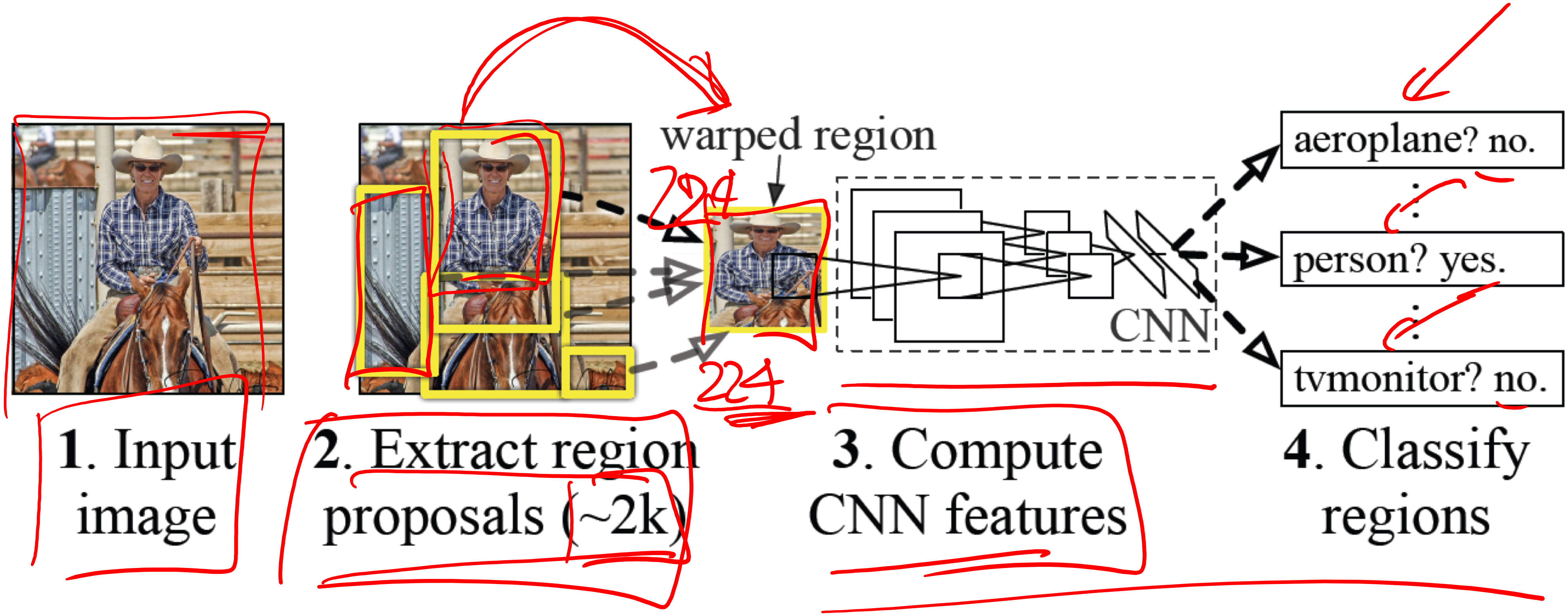


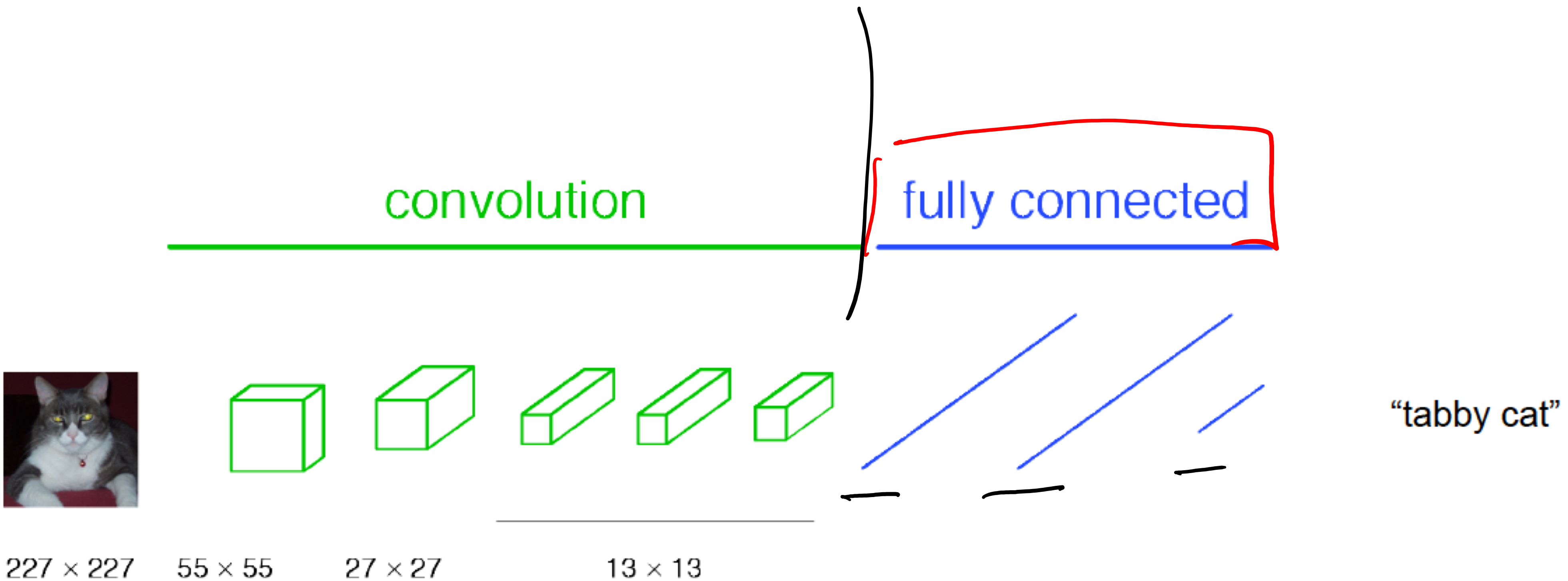
Figure Credit: [Long, Shelhamer, Darrell CVPR15]


H hidden units
NxNxC, N small
C
N
N
7x7x512
Fully conn. layer
N²C
H
H x N²C

Classical View = Inefficient



Classical View



Re-interpretation

- Just squint a little!

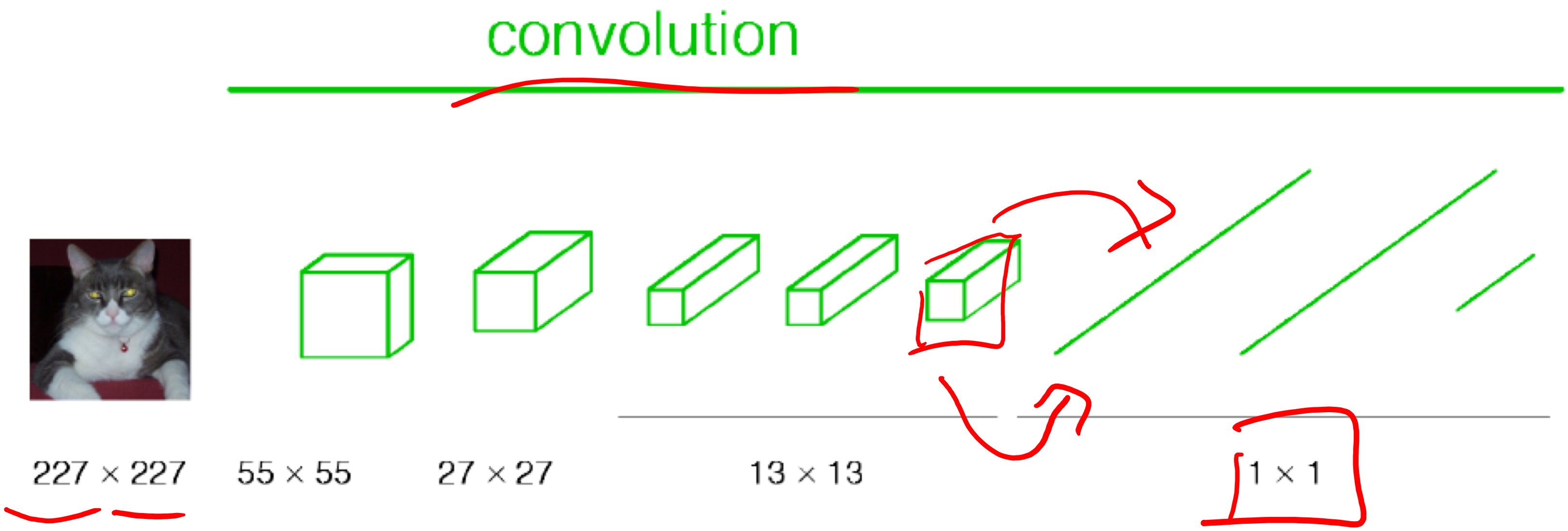


Figure Credit: [Long, Shelhamer, Darrell CVPR15]

“Fully Convolutional” Networks

- Can run on an image of any size!

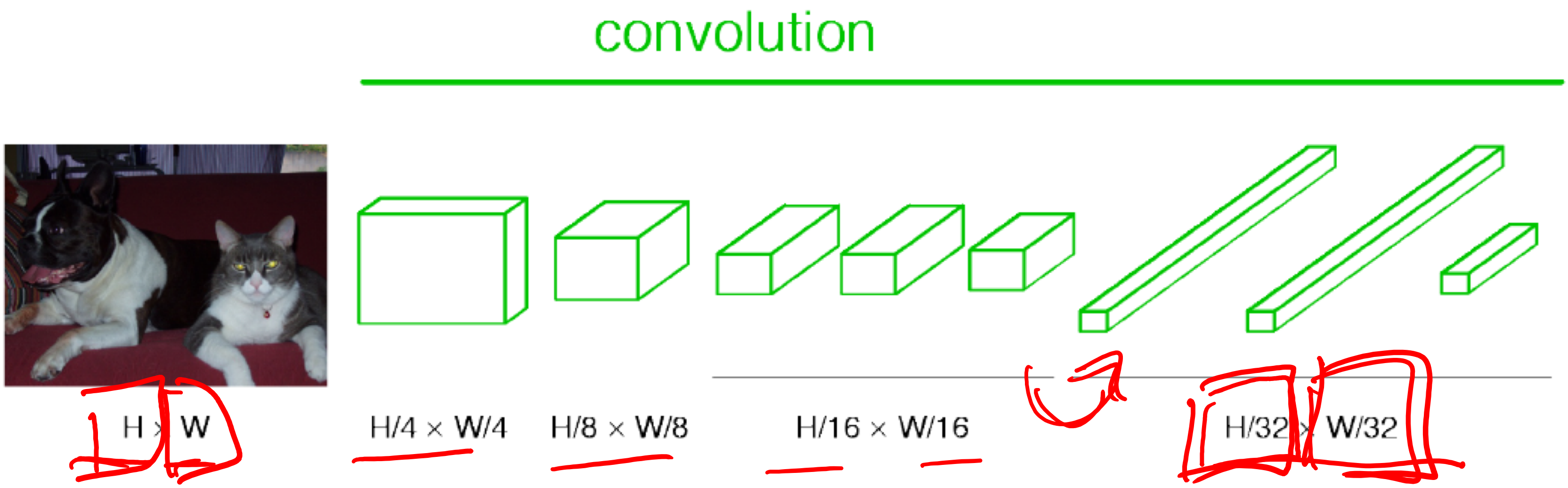
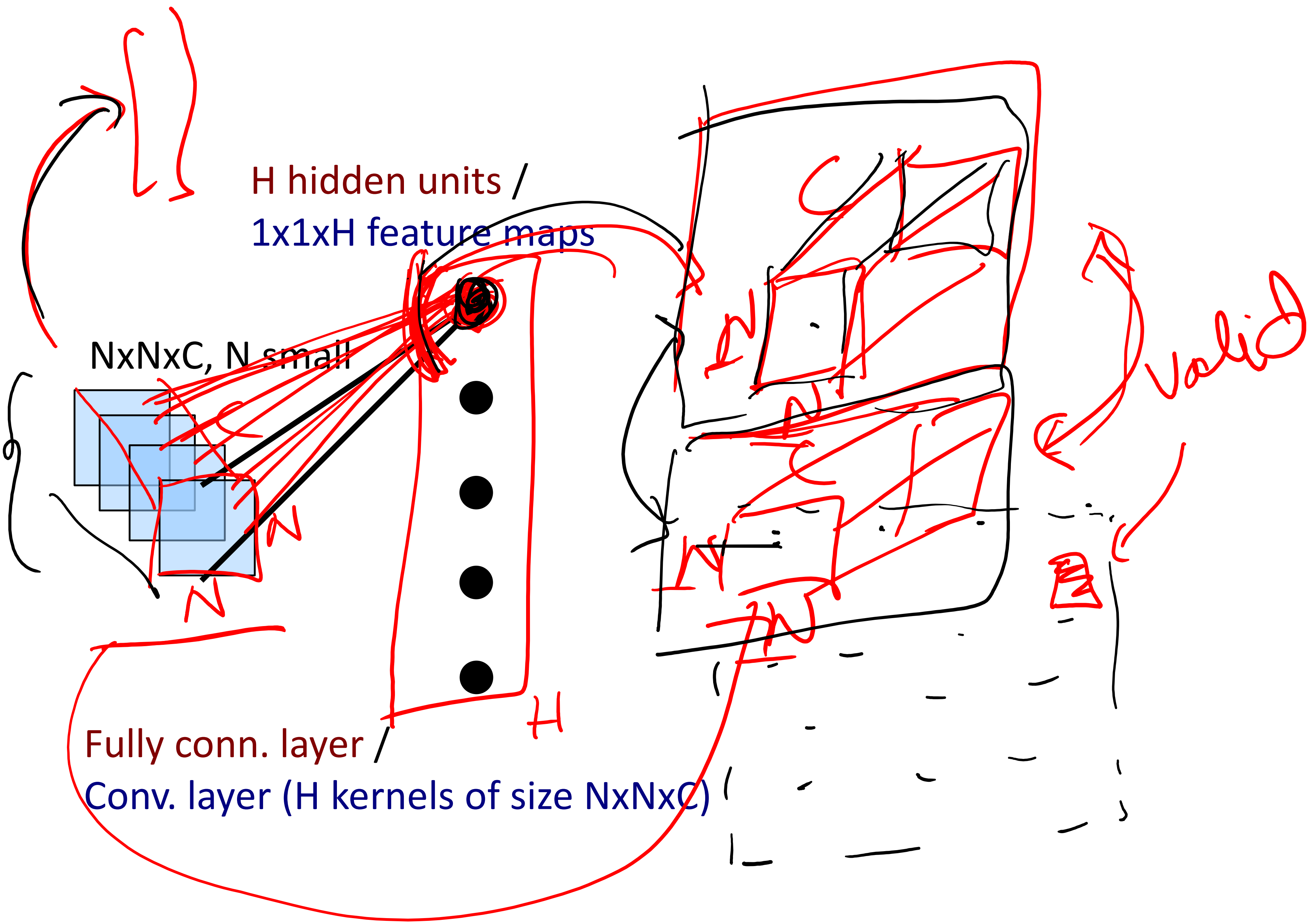


Figure Credit: [Long, Shelhamer, Darrell CVPR15]



H hidden units /
1x1xH feature maps
NxNxC, N small
N
C
H
valid
Fully conn. layer /
Conv. layer (H kernels of size NxNxC)

Benefit of this thinking

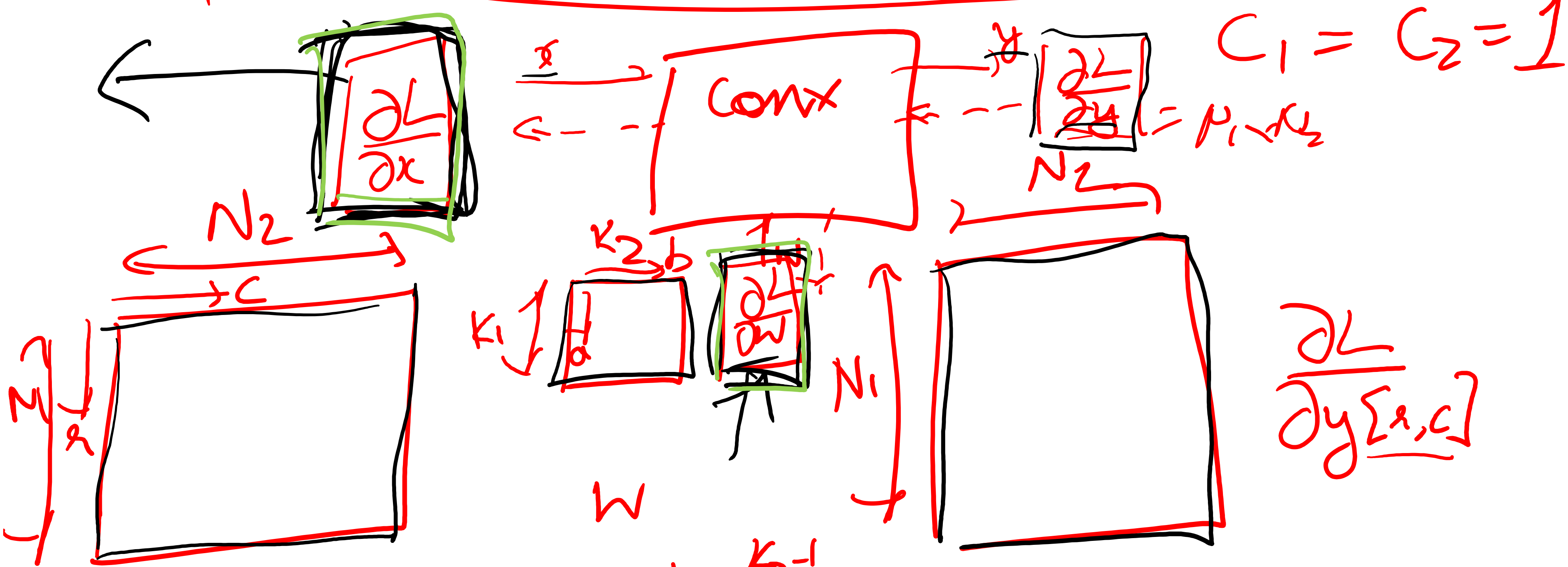
- Mathematically elegant
- Efficiency
  - Can run network on arbitrary image
  - Without multiple crops

Plan for Today

- Convolutional Neural Networks
  - (Finish) Backprop in conv layers
  - Dilated/a-trous convolutions
  - Toeplitz matrices and convolutions = matrix-mult
  - Transposed convolutions

Backprop in Convolutional Layers

- Notes
  - https://www.cc.gatech.edu/classes/AY2018/cs7643_fall/slides/L6_cnns_backprop_notes.pdf

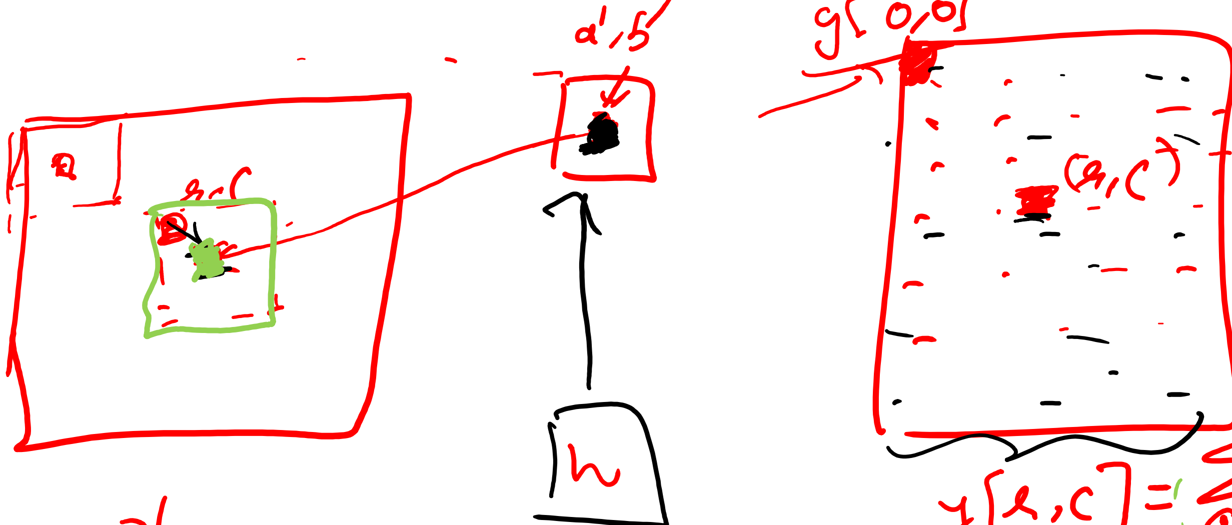


$$y[r,c] = \sum_{a=0}^{K_1-1} \sum_{b=0}^{K_2-1} x[r+a, c+b]\, w[a,b]$$

Backprop in Convolutional Layers

$$\frac{\partial L}{\partial w[a',b']} = \sum_{\text{all pixels } p \text{ affected}} \frac{\partial L}{\partial y[p]} \underbrace{\frac{\partial y[p]}{\partial w[a',b']}}_{\text{Jacobian}}$$

input

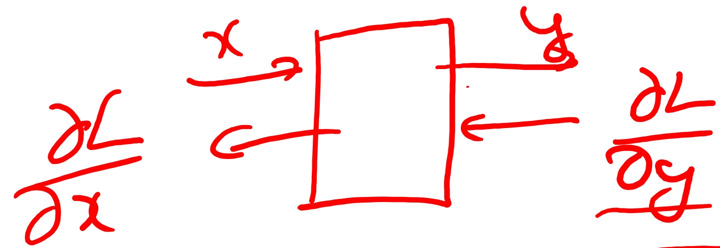


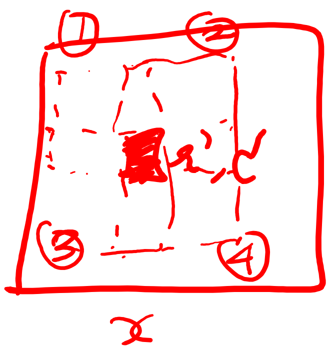
$$y[r,c] = \sum_a \sum_b x[r+a, c+b]\, w[a,b]$$
$$= x[r+0, c+0]\, w[0,0] + x[r+0, c+1]\, w[0,1] + \dots + x[r+a', c+b']\, w[a',b']$$

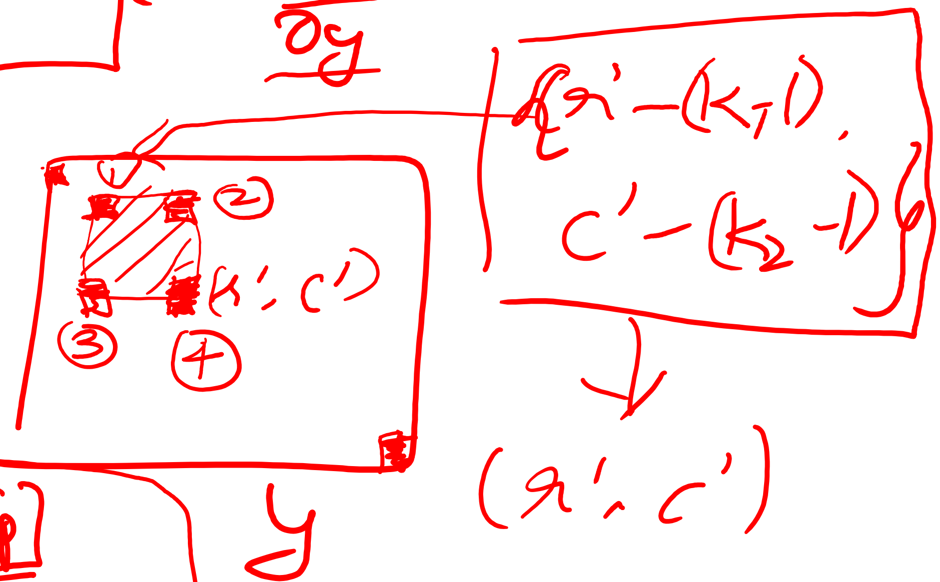
$$\sum \frac{\partial y[r,c]}{\partial w[a',b']} = x[r+a', c+b']$$

$$\frac{\partial L}{\partial w[a',b']} = \sum_{r=0}^{N_1} \sum_{c=0}^{N_2} \frac{\partial L}{\partial y[r,c]}\, x[r+a', c+b'] = x * \frac{\partial L}{\partial y}$$

Backprop in Convolutional Layers







$$\frac{\partial L}{\partial x[r',c']} = \sum_{\text{Pixels affected } p} \frac{\partial L}{\partial y[p]} \frac{\partial y[p]}{\partial x[r',c']}$$

Jacobian / To-be-computed

$$= \sum_{a=0}^{k_1-1} \sum_{b=0}^{k_2-1} \frac{\partial L}{\partial y[r'-a, c'-b]} \underbrace{\frac{\partial y[r'-a, c'-b]}{\partial x[r', c']}}_{w[a,b]}$$

Backprop in Convolutional Layers

$$\boxed{\frac{\partial y[r'-a, c'-b]}{\partial x[r', c']}}$$

$$\boxed{y[r'-a, c'-b]} = \sum_{a'=0}^{k_1-1} \sum_{b'=0}^{k_2-1} x[r'+a'-a, c'+b'-b]\, w[a', b']$$

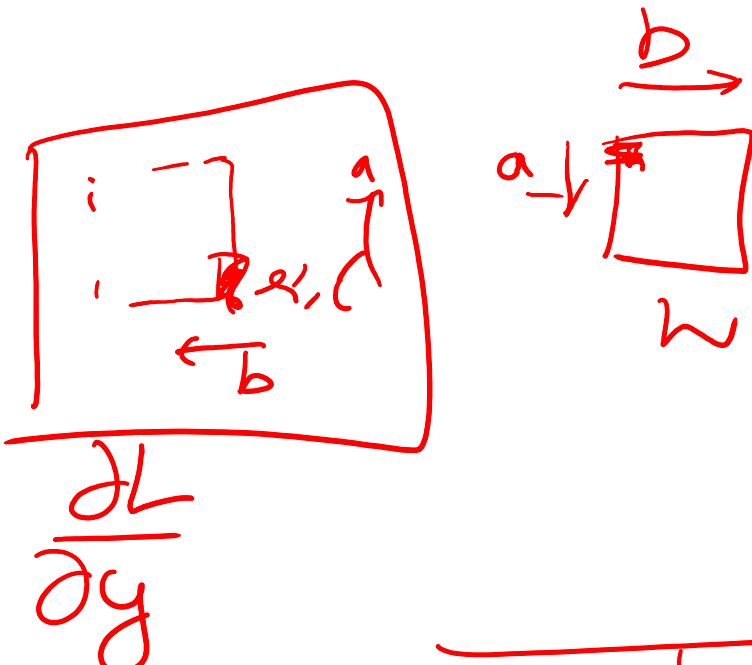
$$\boxed{x[r', c']} \, w[a, b]$$

$$\frac{\partial y[\quad]}{\partial x[r', c']} = w[a, b]$$

Backprop in Convolutional Layers

$$\frac{\partial L}{\partial x[r',c']} = \boxed{\sum_{a=0}^{k_1-1}\sum_{b=0}^{k_2-1}} \underbrace{\frac{\partial L}{\partial y[r'-a,\, c'-b]}} \underbrace{w[a,b]}$$

$$= \underbrace{\boxed{\frac{\partial L}{\partial y}} * \boxed{w^{flip}}}$$



w[0,0]	w[0,1]
w[1,0]	w[1,1]

flip 180°

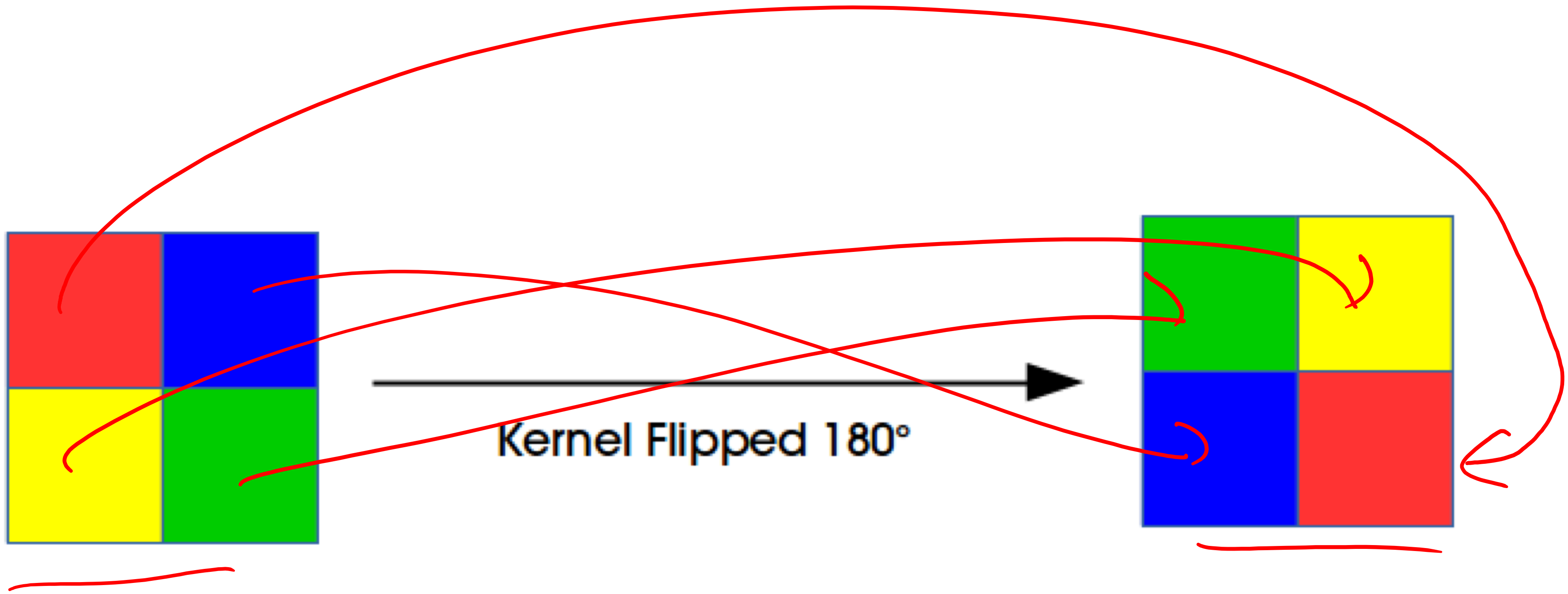
w[1,1]	w[1,0]
w[0,1]	w[0,0]

Backprop in Convolutional Layers

Backprop in Convolutional Layers

Backprop in Convolutional Layers

Backprop in Convolutional Layers

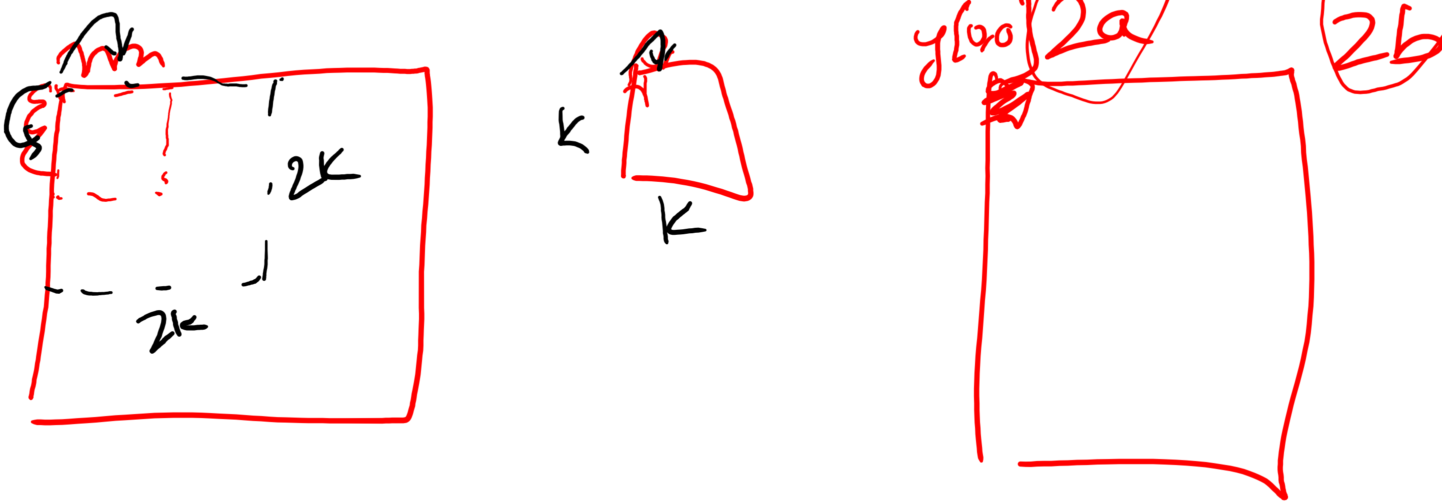


Plan for Today

- Convolutional Neural Networks
  - (Finish) Backprop in conv layers
  - Dilated/a-trous convolutions
  - Toeplitz matrices and convolutions = matrix-mult
  - Transposed convolutions

Dilated Convolutions

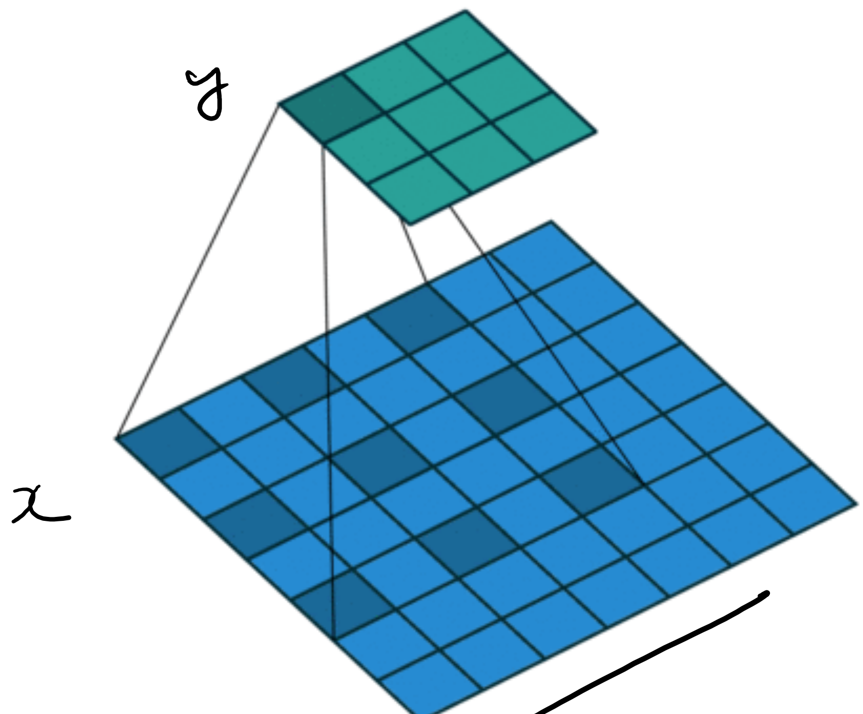
$$y[r,c] = \sum_a \sum_b x[r+a, c+b]\, \underbrace{w[a,b]}$$



Dilated Convolutions

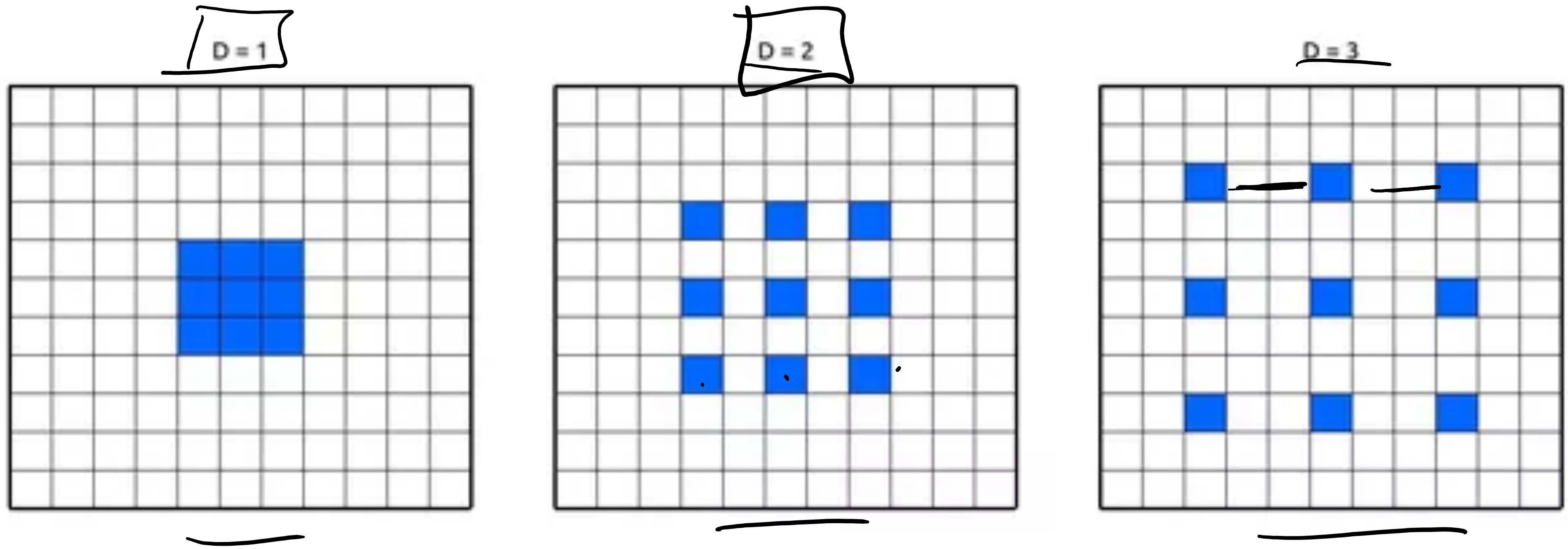
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ & & & & \end{bmatrix}$$

$d-1$



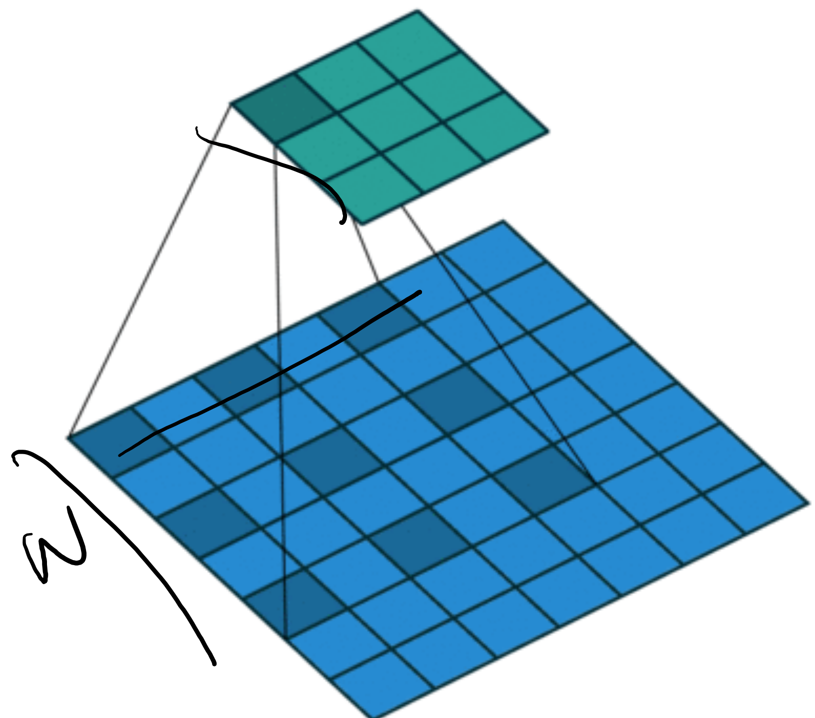
$d=2$

Figure Credit: Dumoulin and Visin, https://arxiv.org/pdf/1603.07285.pdf



$$k \rightarrow \underline{k} + \underbrace{(\underline{k-1})(d-\underline{1})}$$

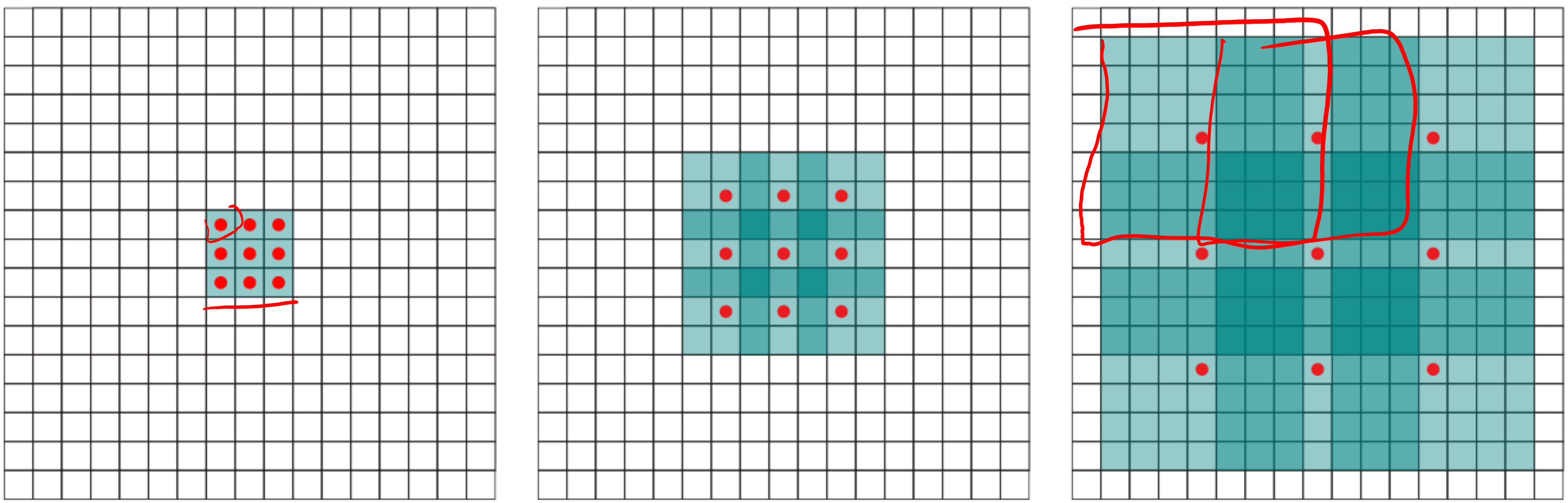
$$3 + (3-1)(2-1) = 5$$



(recall:)

$\boxed{(N - k) / \text{stride} + 1}$

Figure Credit: Dumoulin and Visin, https://arxiv.org/pdf/1603.07285.pdf



(a) (b) (c)

Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

Plan for Today

- Convolutional Neural Networks
  - (Finish) Backprop in conv layers
  - Dilated/a-trous convolutions
  - Toeplitz matrices and convolutions = matrix-mult
  - Transposed convolutions

Toeplitz Matrix

- Diagonals are constants

$$\begin{bmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{bmatrix}.$$

- $\boxed{A_{ij} = a_{\boxed{i-j}}}$

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & \dots & a_{-n+1} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{bmatrix}$$

Why do we care?

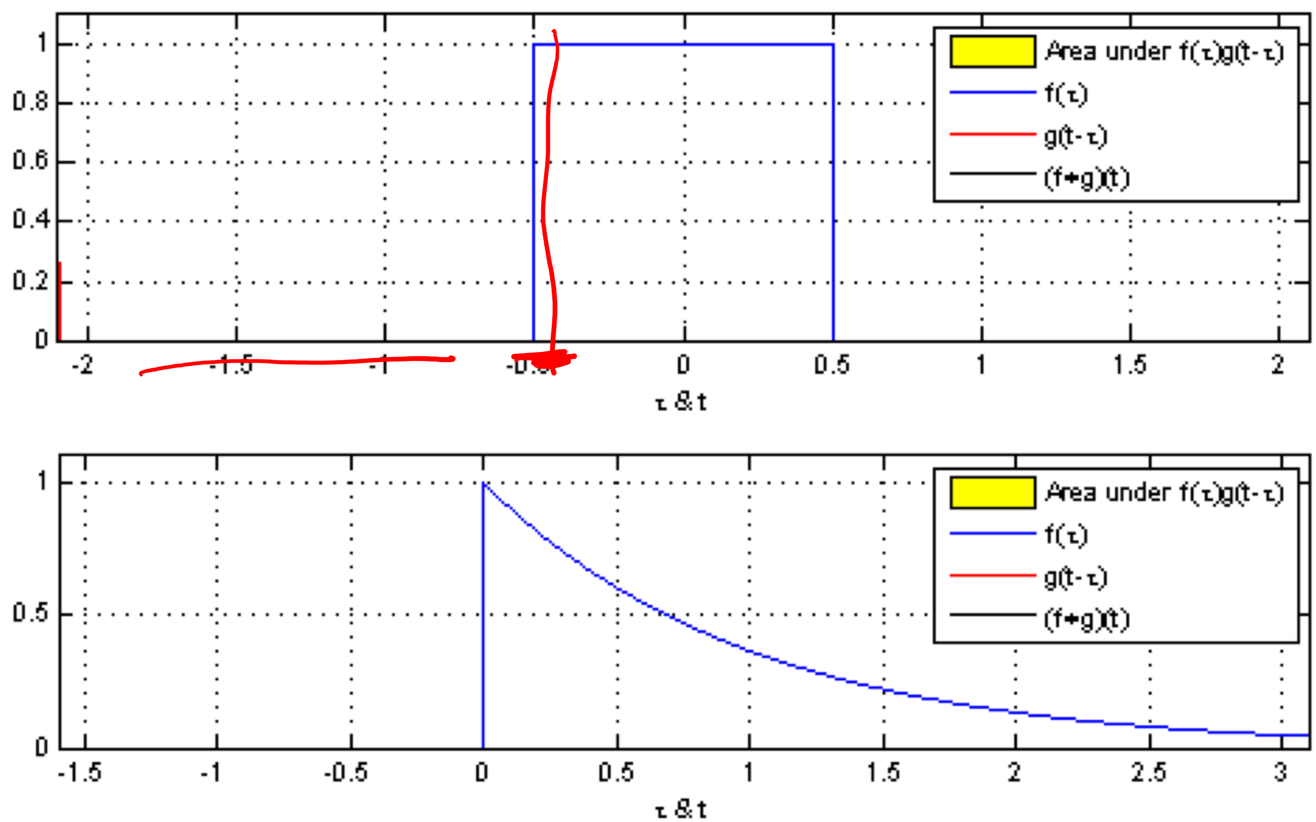
- (Discrete) Convolution = Matrix Multiplication
  - with Toeplitz Matrices

$w = \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix}$

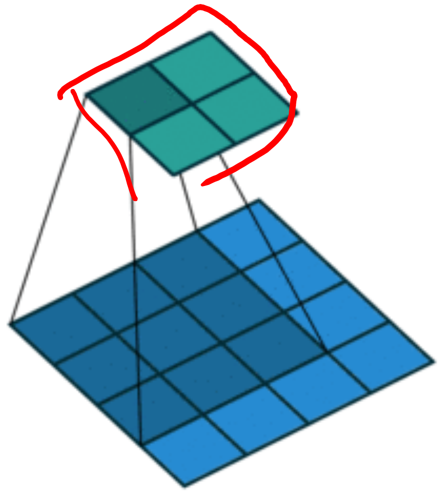
$$y = w * x \qquad \begin{bmatrix} w_k & 0 & \dots & 0 & 0 \\ w_{k-1} & w_k & \dots & 0 & 0 \\ w_{k-2} & w_{k-1} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_1 & w_{k-2} & \dots & w_k & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & w_1 & \dots & w_{k-1} & w_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \vdots & w_1 & w_2 \\ 0 & 0 & \vdots & 0 & w_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

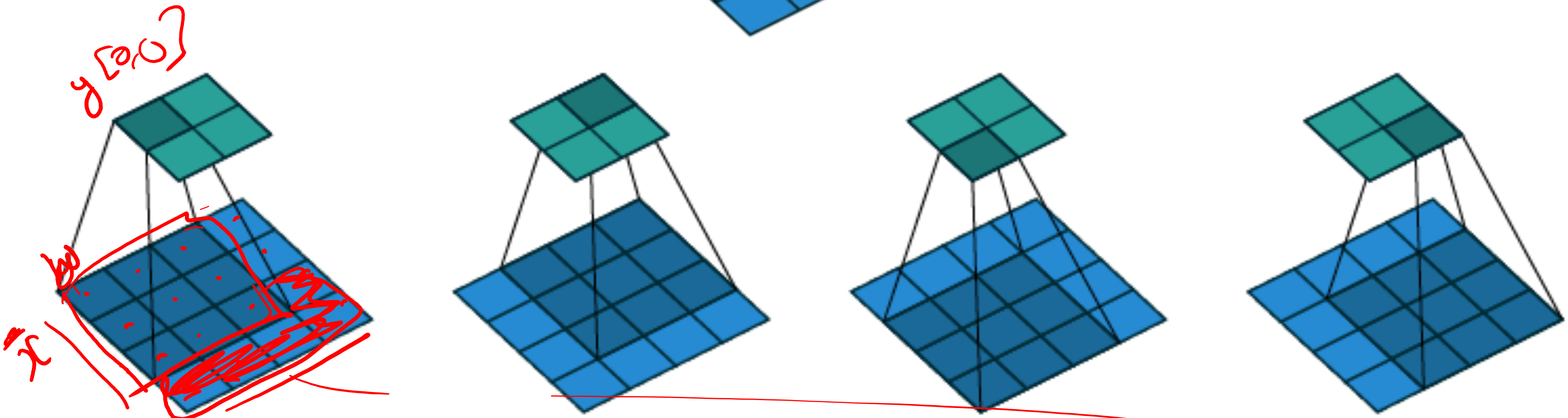
$\vec{y} = W\vec{x}$

$\vec{y} = W\vec{x}$



"Convolution of box signal with itself2" by Convolution_of_box_signal_with_itself.gif: Brian Ambergderivative work: Tinos (talk) - Convolution_of_box_signal_with_itself.gif. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Convolution_of_box_signal_with_itself2.gif#/media/File:Convolution_of_box_signal_with_itself2.gif





$$
\begin{bmatrix} y[0,0] \\ \\ \\ \\ \end{bmatrix}
\leftarrow
\begin{pmatrix}
w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\
0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\
0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2}
\end{pmatrix}
\underbrace{\begin{bmatrix} \vdots \end{bmatrix}}_{16}^{x(:)}
$$

$$y\text{-vec} = W\, x(:)$$

Im2Col

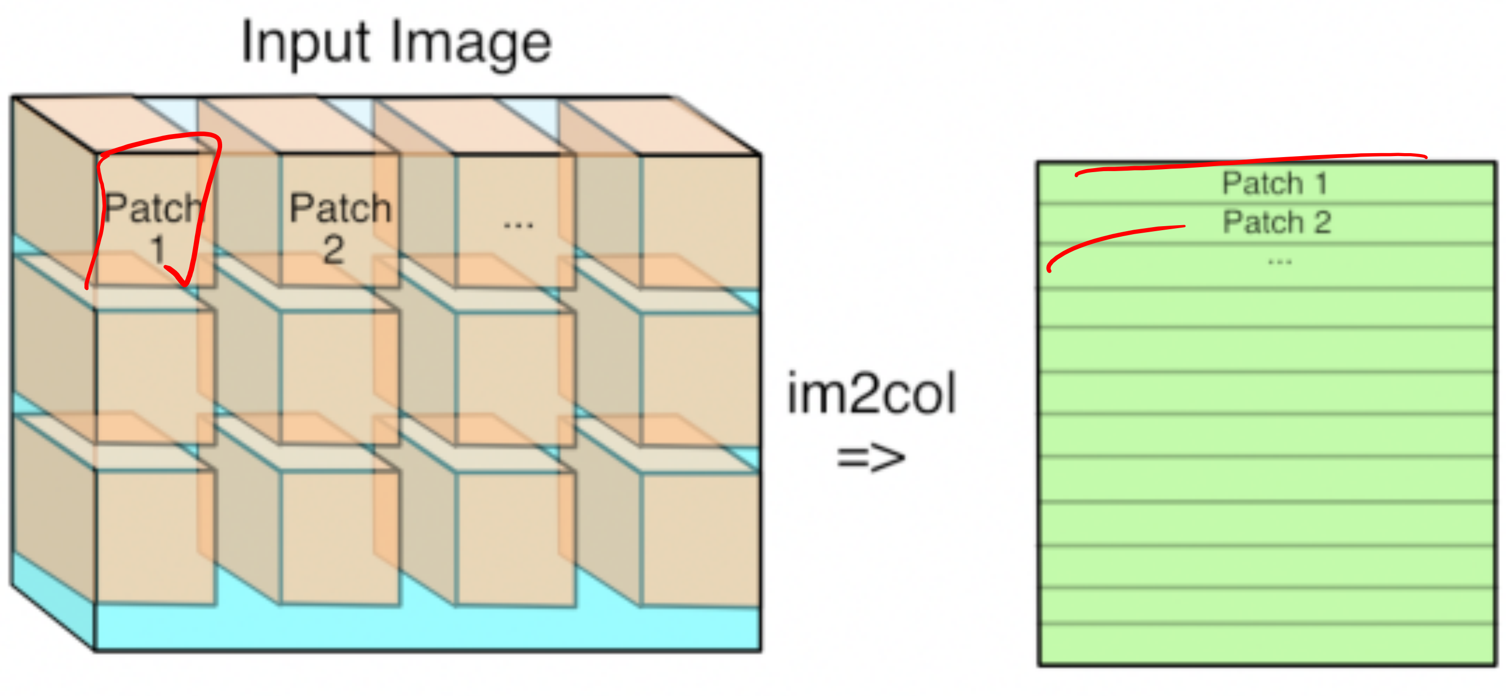
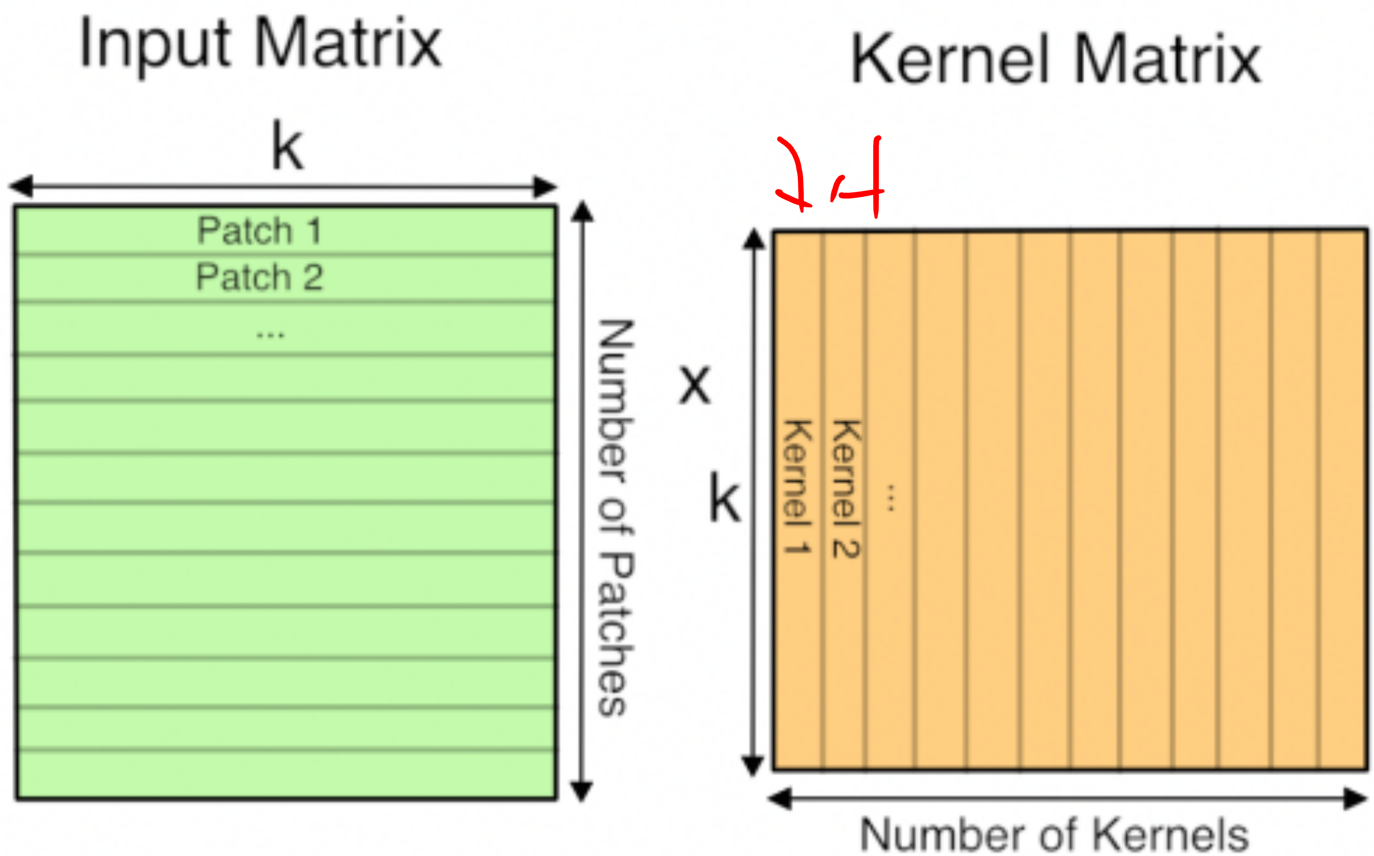


Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

GEMM



Input Matrix
k
Patch 1
Patch 2
...
Number of Patches
x
Kernel Matrix
k
Kernel 1
Kernel 2
...
Number of Kernels

Plan for Today

- Convolutional Neural Networks
  - (Finish) Backprop in conv layers
  - Dilated/a-trous convolutions
  - Toeplitz matrices and convolutions = matrix-mult
  - Transposed convolutions

Transposed Convolutions

- Deconvolution (bad)
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution

So far: Image Classification

This image is CC0 public domain

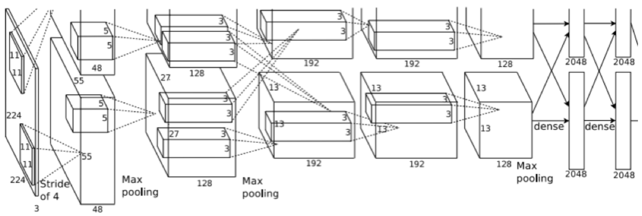


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

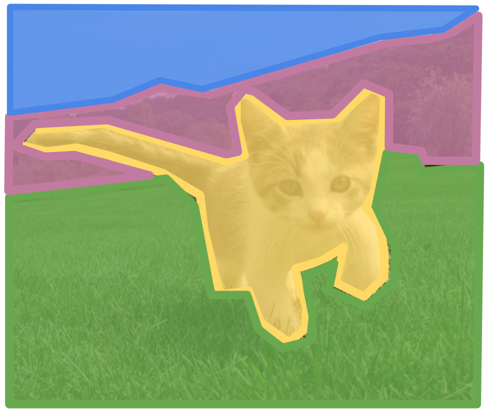
Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Other Computer Vision Tasks

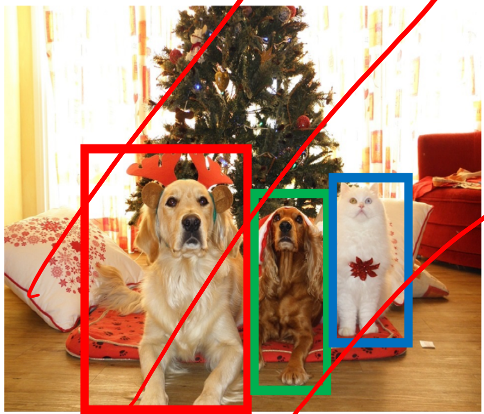
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

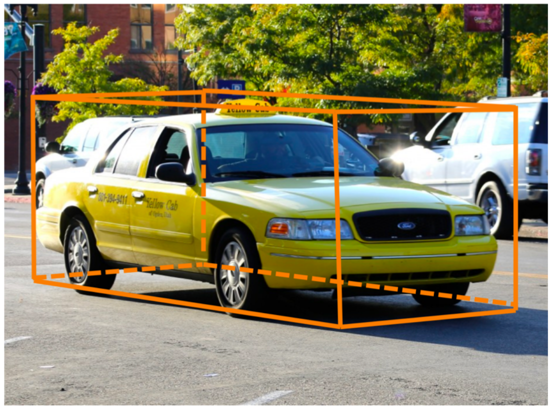
2D Object Detection



DOG, DOG, CAT

Object categories + 2D bounding boxes

3D Object Detection

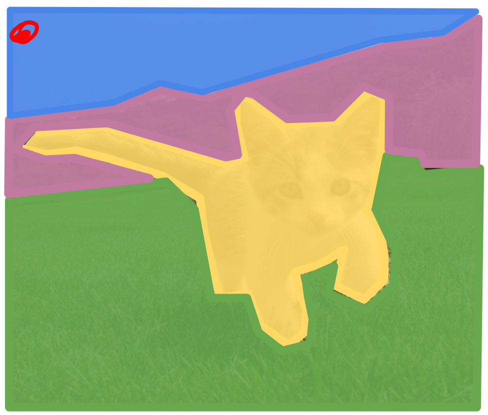


Car

Object categories + 3D bounding boxes

Semantic Segmentation

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

2D Object Detection

DOG, DOG, CAT

Object categories + 2D bounding boxes

3D Object Detection

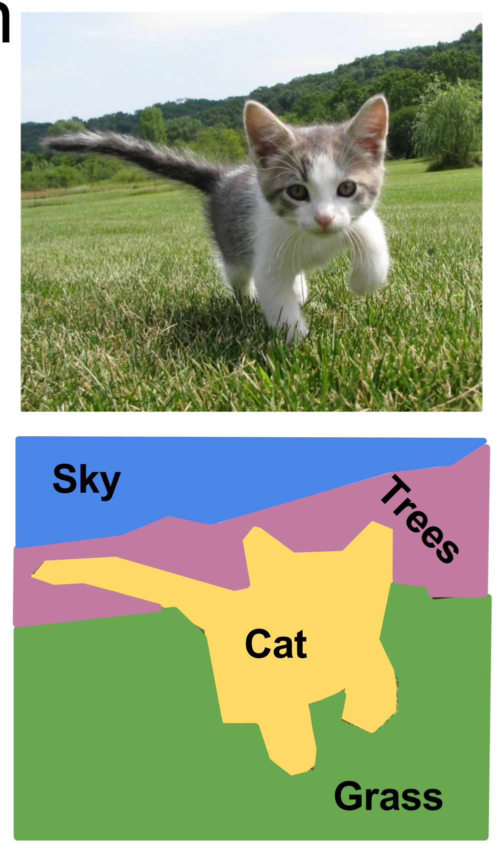
Car

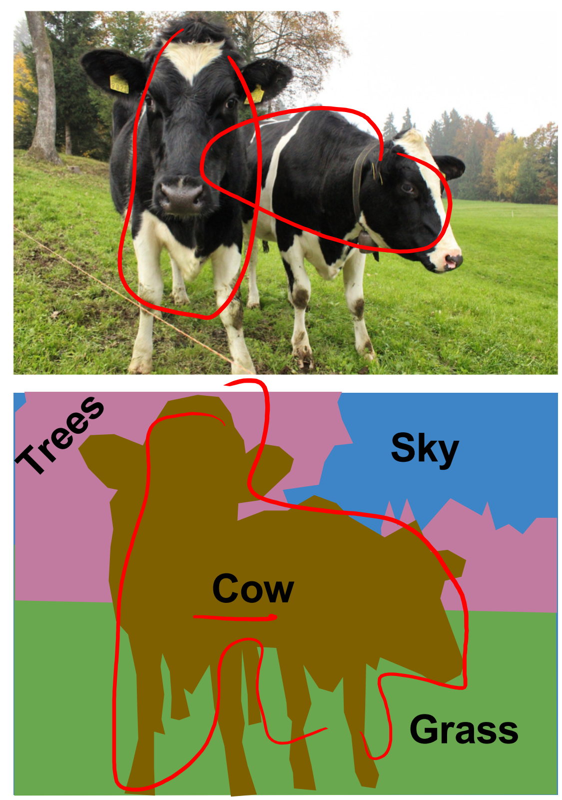
Object categories + 3D bounding boxes

Semantic Segmentation

Label each pixel in the image with a category label

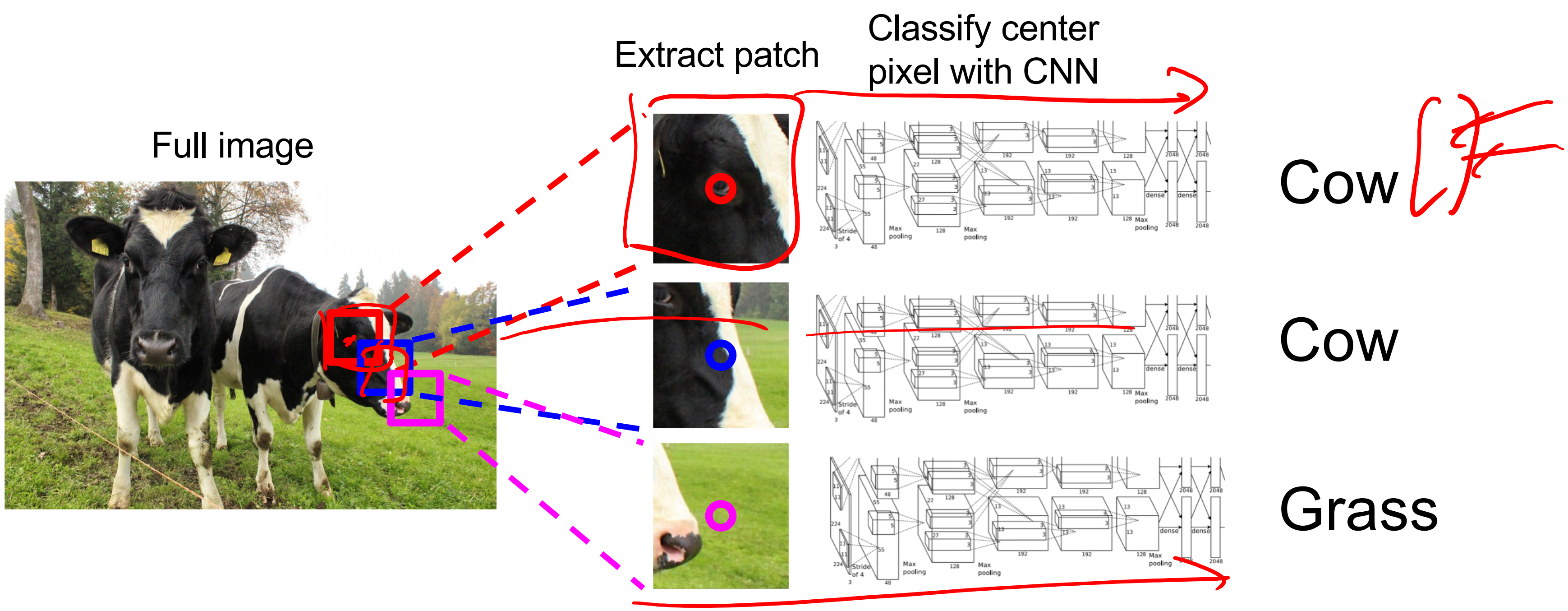
Don't differentiate instances, only care about pixels





Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Semantic Segmentation Idea: Sliding Window



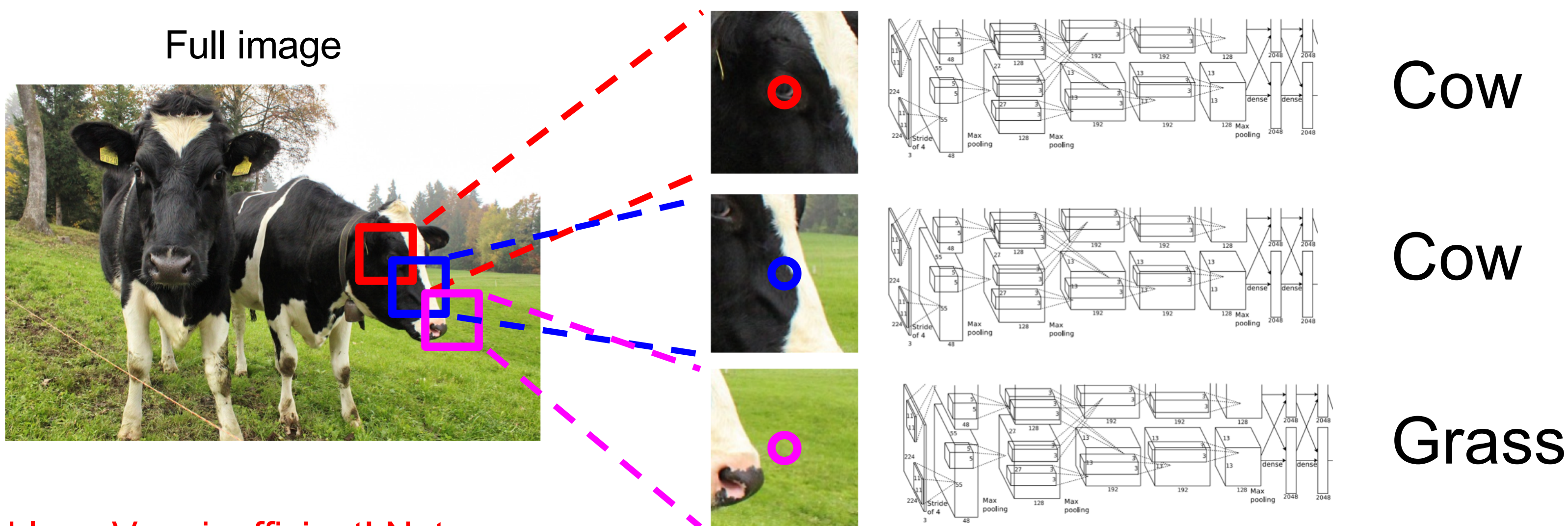
Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Sliding Window

Full image

Extract patch

Classify center pixel with CNN

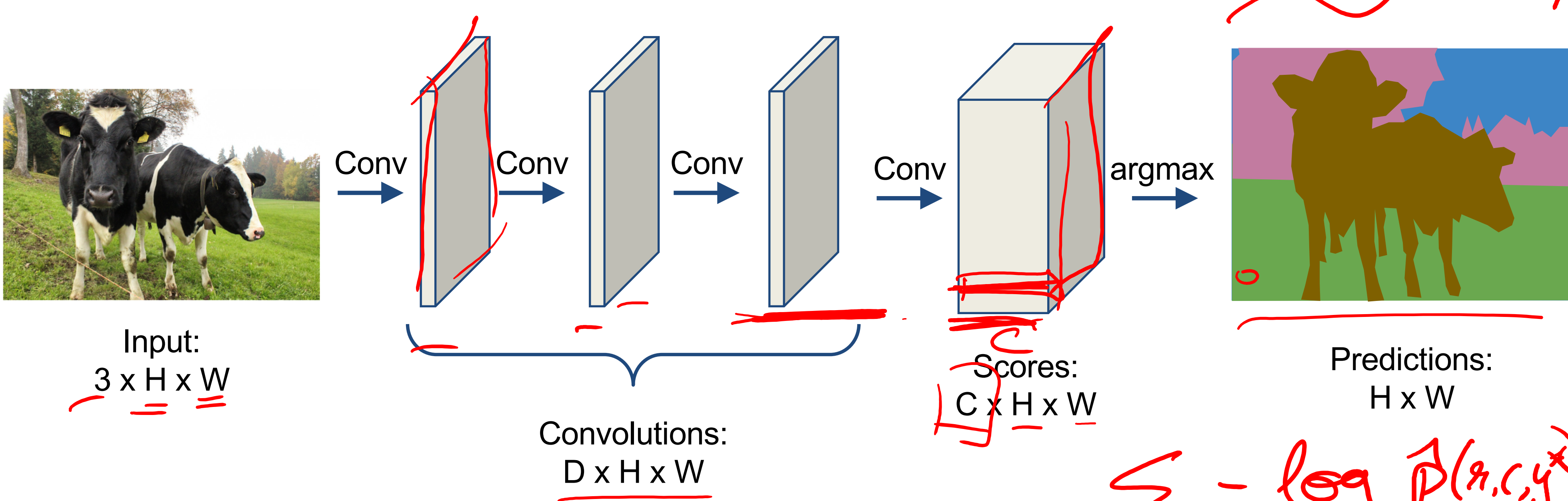


Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Fully Convolutional

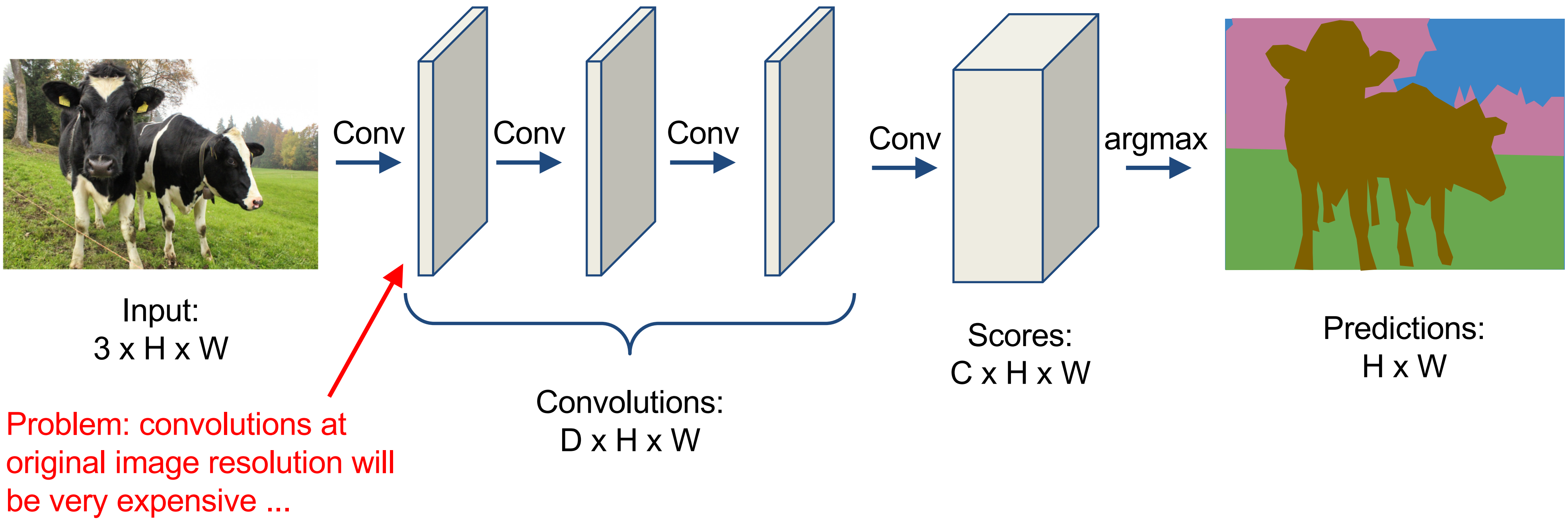
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



$$\sum_{\text{pixel}} - \log p(r, c, y^*)$$

Semantic Segmentation Idea: Fully Convolutional

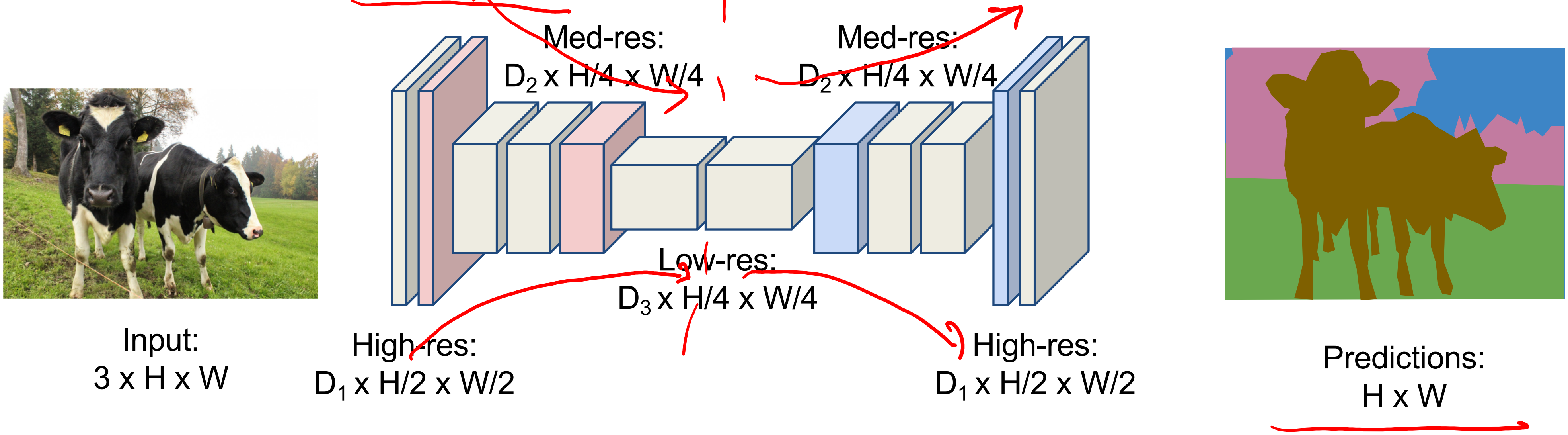
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

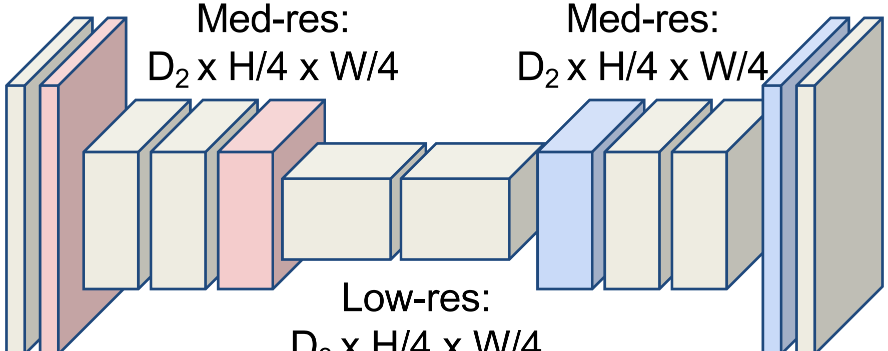
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: ???

Input: $3 \times H \times W$

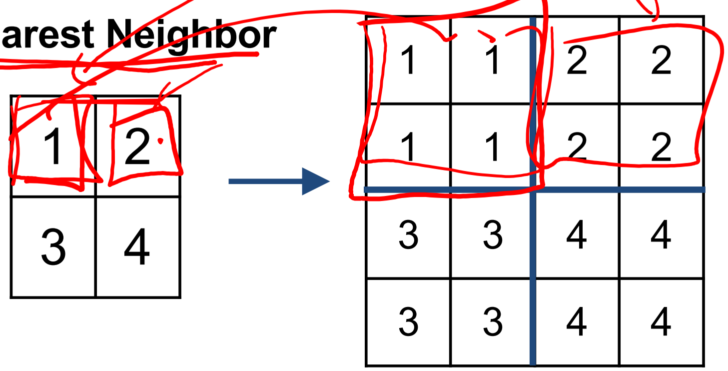


Predictions: $H \times W$

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

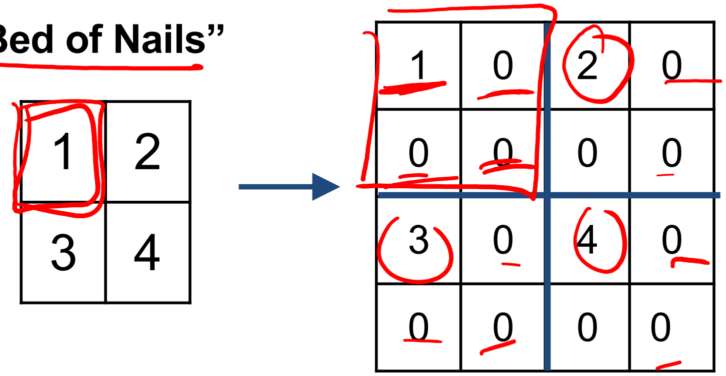
In-Network upsampling: "Unpooling"

Nearest Neighbor



Input: 2 x 2 Output: 4 x 4

"Bed of Nails"



Input: 2 x 2 Output: 4 x 4

In-Network upsampling: "Max Unpooling"

Max Pooling
Remember which element was max!

argmax

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

Input: 4 x 4

max

5	6
7	8

Output: 2 x 2

→ . . . →

Rest of the network

Max Unpooling
Use positions from pooling layer

1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

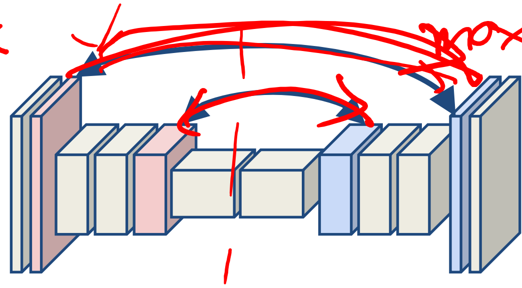
Output: 4 x 4

max

max

unpool

Corresponding pairs of downsampling and upsampling layers



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n