deep imaging

Lecture 14: Object detection and Segmentation

Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

deep imaging

Training error

$L_{in}(\mathbf{y}, f(\mathbf{W},\mathbf{x})) =$

cross_entropy($\mathbf{y}$, $f(\mathbf{W},\mathbf{x})$)

Output
$\mathbf{y}^*$

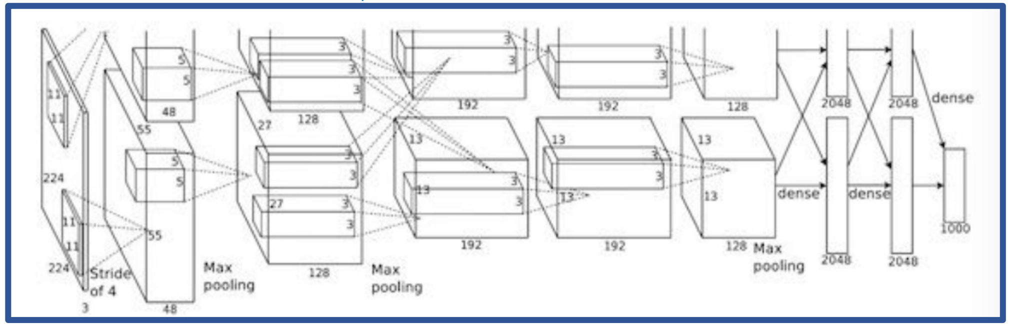
Model

$\mathbf{y}^* = f(\mathbf{W},\mathbf{x})$

Ex. $[\mathbf{x}_1,\mathbf{y}_1]$ ... Ex. $[\mathbf{x}_K,\mathbf{y}_K]$

Training Data

dL/d**W**



Dimensional analysis for classification:

Input **x**: $\sim R^{1000}$
Output **y***: $\sim R^2 - R^{10}$

Training error

$L_{in}(\mathbf{y}, f(\mathbf{W},\mathbf{x}))$ =

cross_entropy($\mathbf{y}$, f($\mathbf{W}$,$\mathbf{x}$))

Make y bigger!

Output

$\mathbf{y}^*$

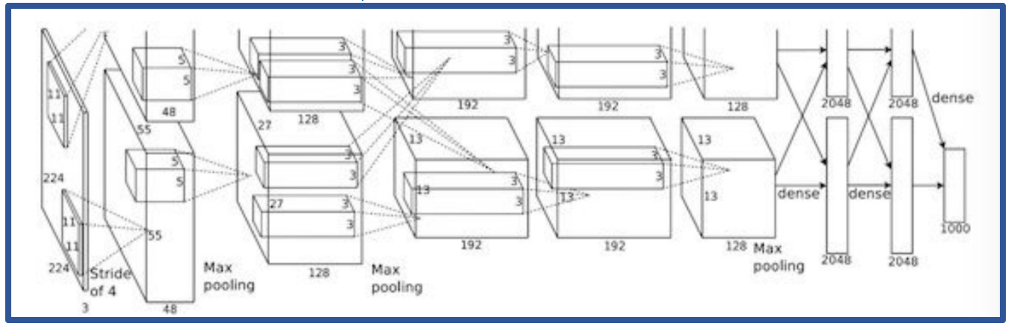
Model

$\mathbf{y}^* = f(\mathbf{W},\mathbf{x})$

Ex. [$\mathbf{x}_1$,$\mathbf{y}_1$] … Ex. [$\mathbf{x}_K$,$\mathbf{y}_K$]

Training Data

dL/d**W**



Dimensional analysis for classification:

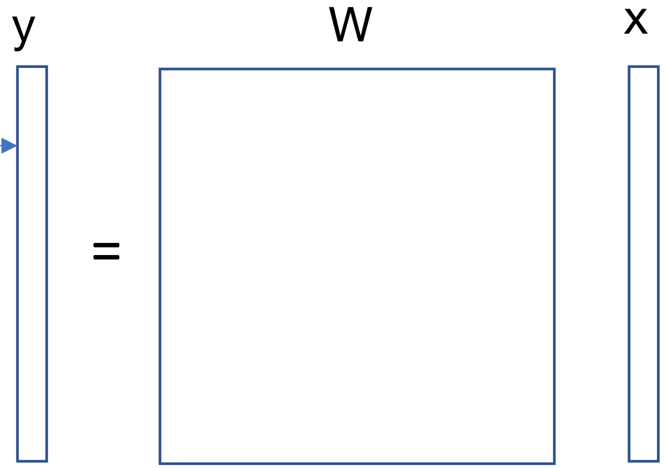
Input **x**: ~R^{1000}
Output **y***: ~$R^2 – R^{10}$

This class – let's make **y*** bigger!

- Object detection
- Segmentation
- Creating 3D volumes
- Better resolution

Over-determined, under-determined and balanced inverse equations

y is what you measure (the image)



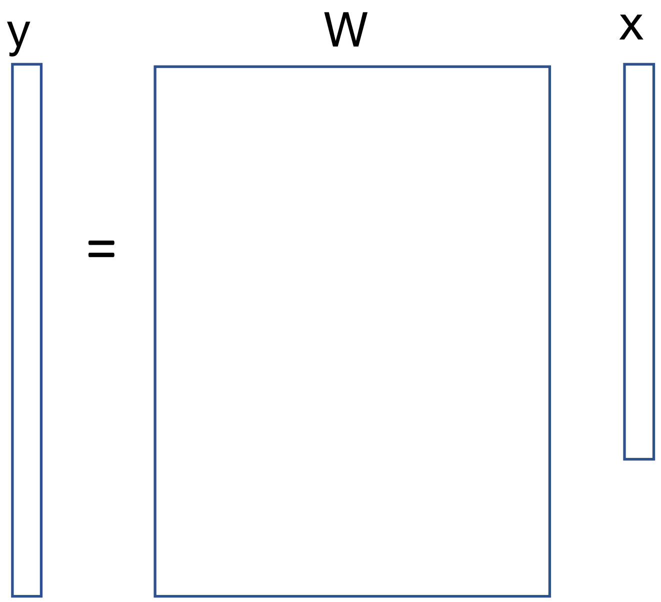
Balanced equation

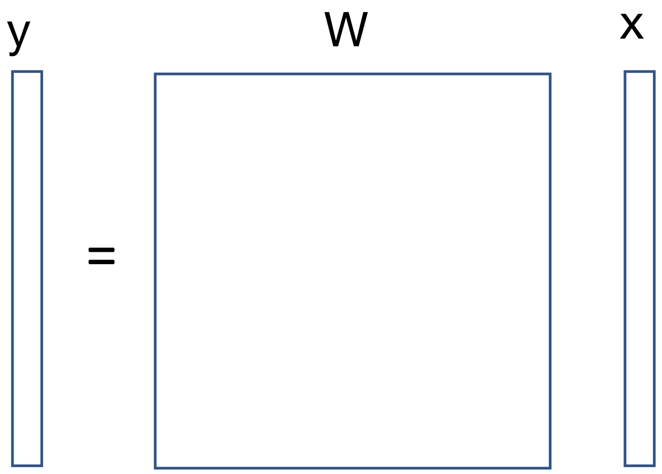
- Invertible if W is “nice”
- Hard to invert otherwise

$$\boxed{x = W^{-1} y}$$

x is what you want to figure out

Over-determined, under-determined and balanced inverse equations





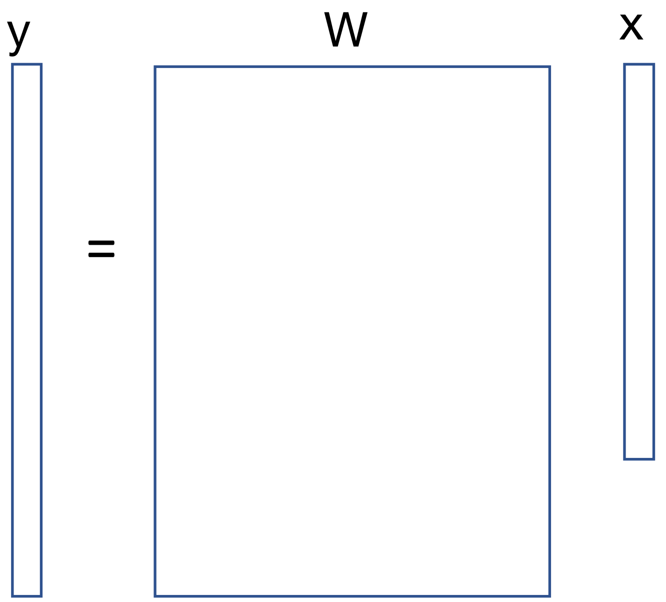
Balanced equation
- Invertible if W is "nice"
- Hard to invert otherwise

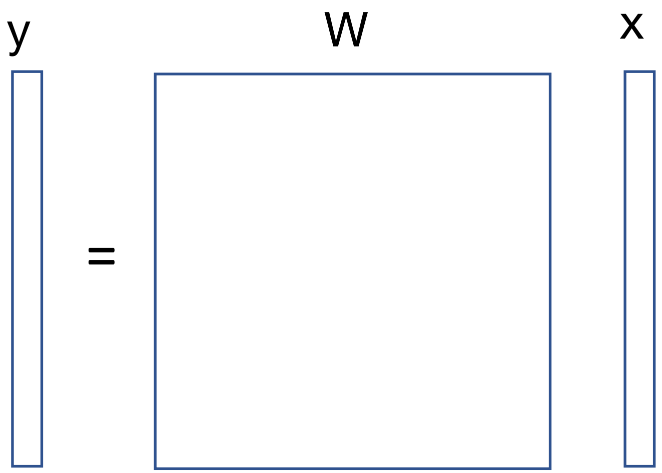
$$\boxed{x = W^{-1} y}$$

Over-determined equation
- Unique solution can exist
- If not, it's easy to get close
- Good place – more measurements than unknowns

$$\boxed{x = W^{+} y}$$

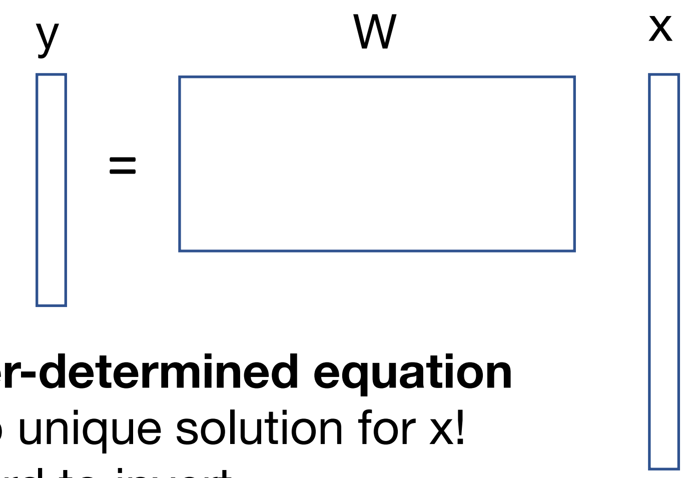
Over-determined, under-determined and balanced inverse equations





Balanced equation
- Invertible if W is “nice”
- Hard to invert otherwise

$$\boxed{x = W^{-1} y}$$

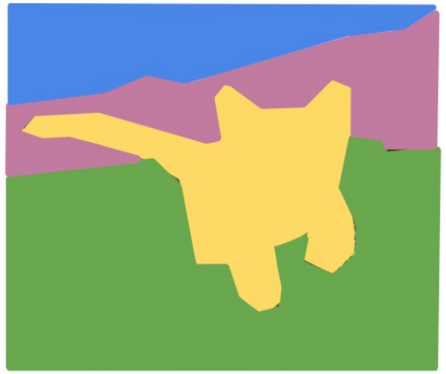


Over-determined equation
- Unique solution can exist
- If not, it’s easy to get close
- Good place – more measurements than unknowns

Under-determined equation
- No unique solution for x!
- Hard to invert
- Not a good place to be

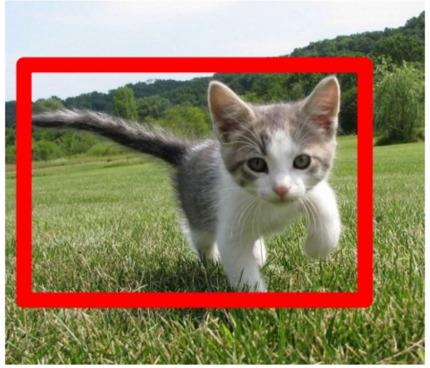
Other Computer Vision Tasks

Semantic Segmentation	**Classification + Localization**	**Object Detection**	**Instance Segmentation**	**Super-resolution**



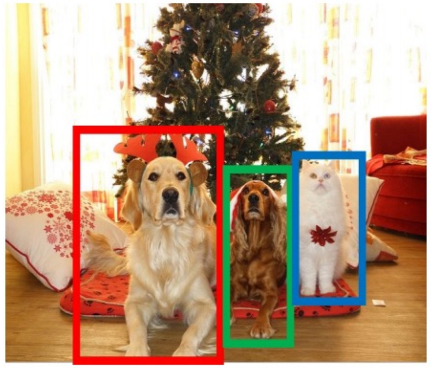
GRASS, CAT, TREE, SKY

No objects, just pixels

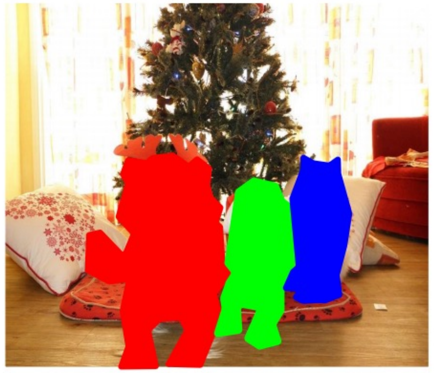


CAT

Single Object



DOG, DOG, CAT



DOG, DOG, CAT

Multiple Object

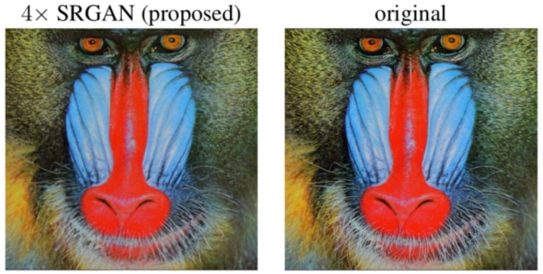


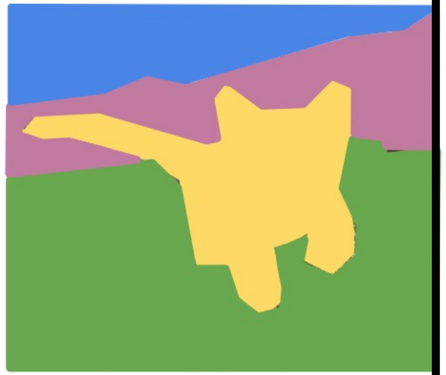
Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Balanced equation	**Over-determined**	**Over-determined**	**Over-determined**	**Under-determined**

Stanford CS231n - http://cs231n.stanford.edu

Other Computer Vision Tasks

Semantic Segmentation

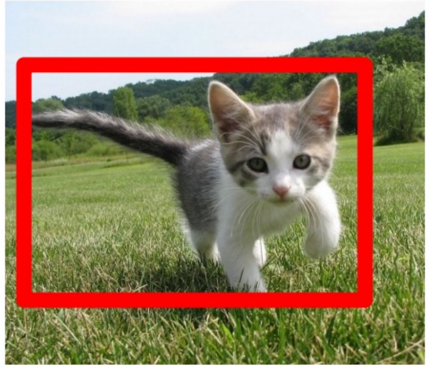


GRASS, CAT, TREE, SKY

No objects, just pixels

Balanced equation

Classification + Localization

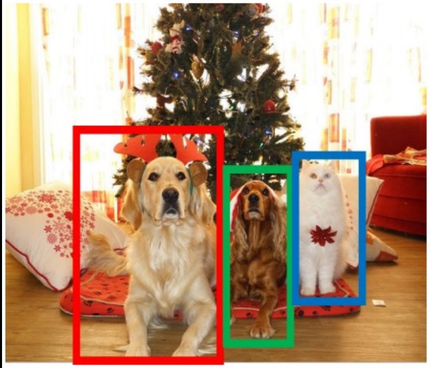


CAT

Single Object

Over-determined

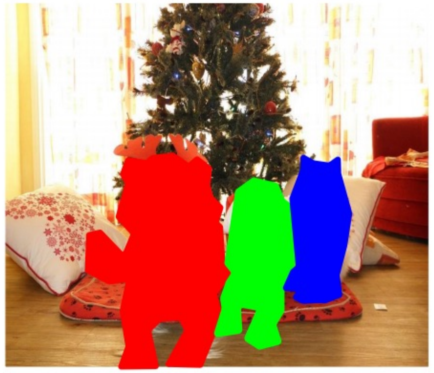
Object Detection



DOG, DOG, CAT

Over-determined

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Over-determined

Super-resolution

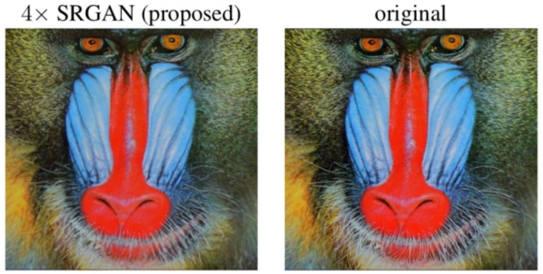
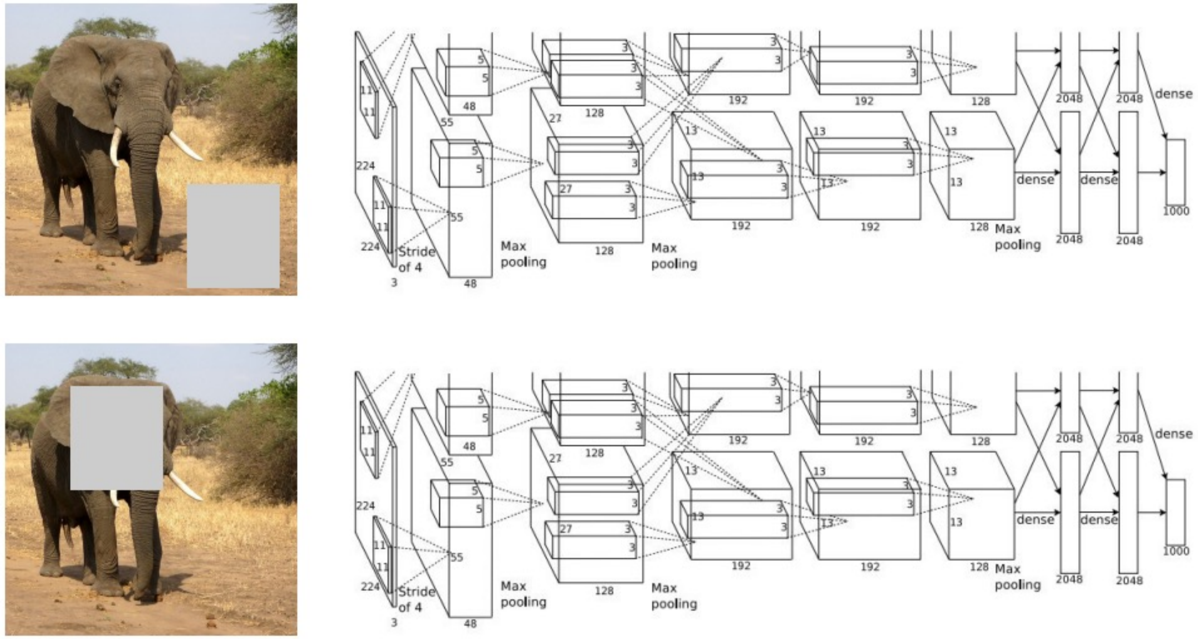


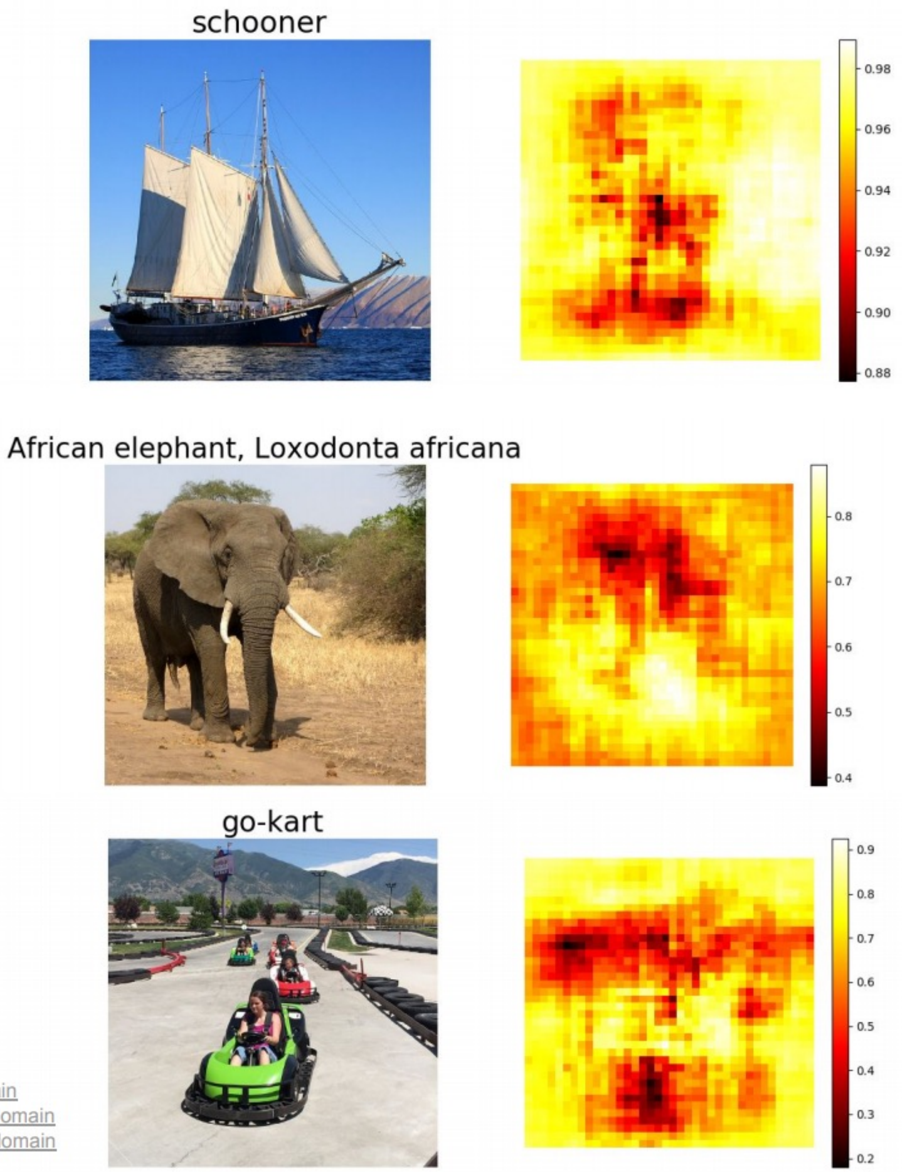
Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Under-determined

Stanford CS231n - http://cs231n.stanford.edu

Approach #1: Sliding window + occlusion map (last lecture)





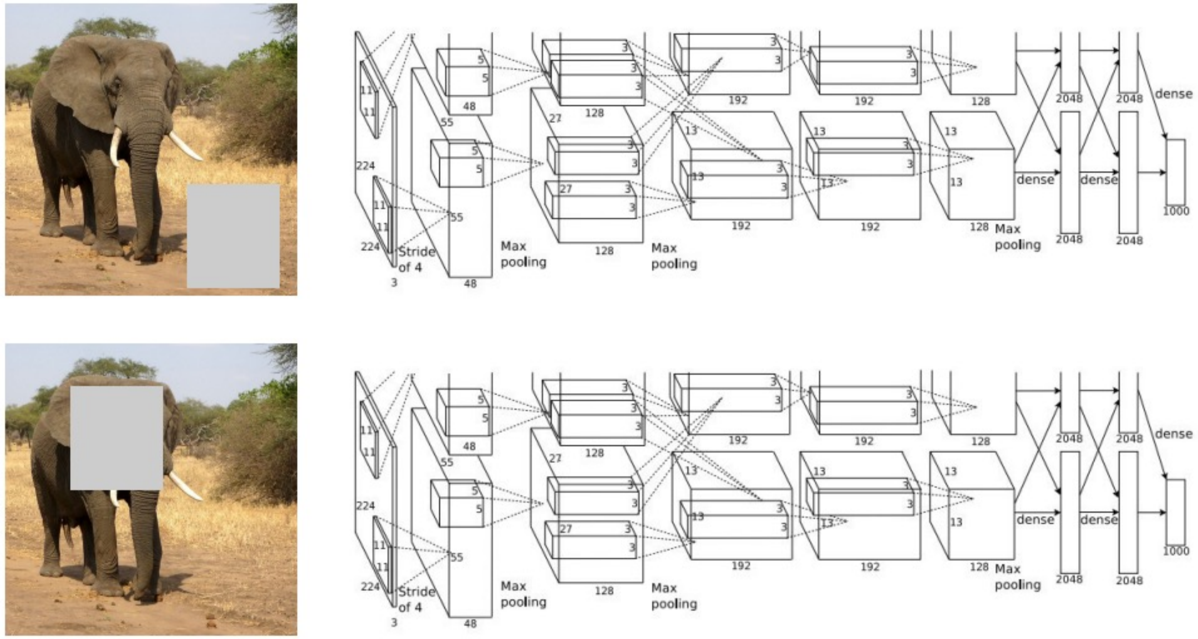
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

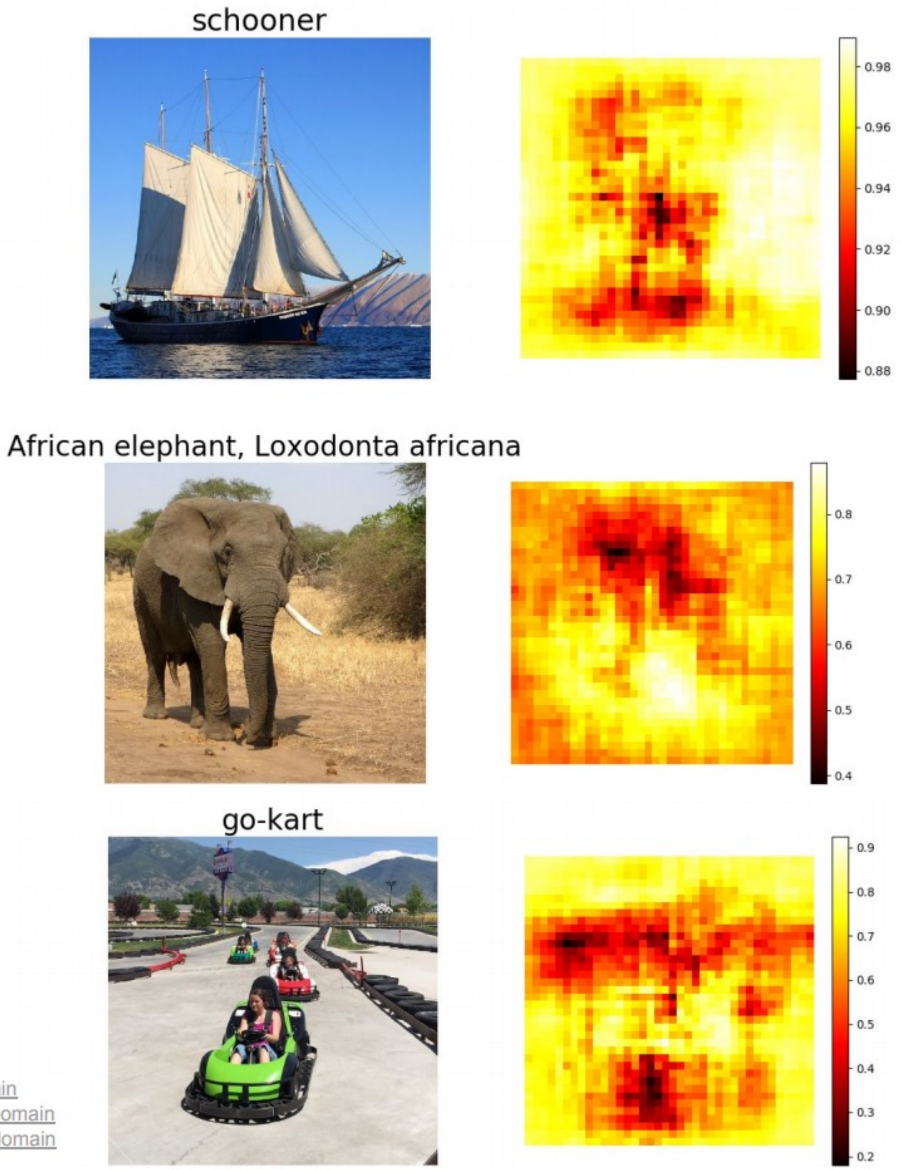
Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain

Stanford CS231n - http://cs231n.stanford.edu

Approach #1: Sliding window + occlusion map (last lecture)

Problem: Inefficient – not sharing information between different sliding window positions (even w/ lots of overlap)





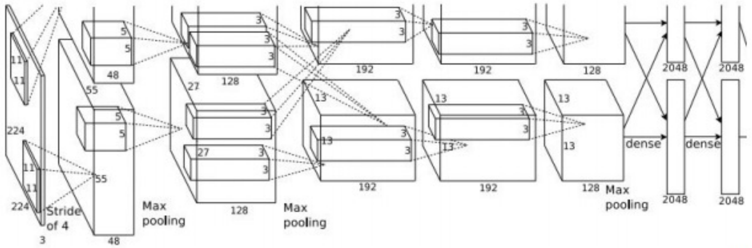
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain

Stanford CS231n - http://cs231n.stanford.edu

Classification + Localization

This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

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

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

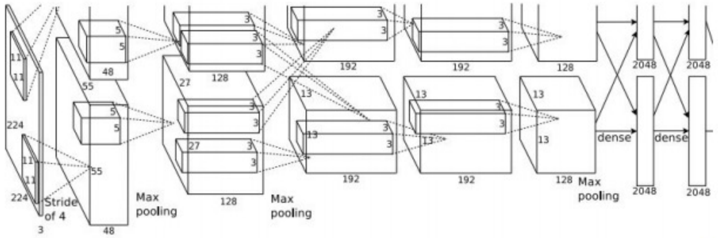
Treat localization as a regression problem!

Stanford CS231n - http://cs231n.stanford.edu

deep imaging

Classification + Localization

This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

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

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

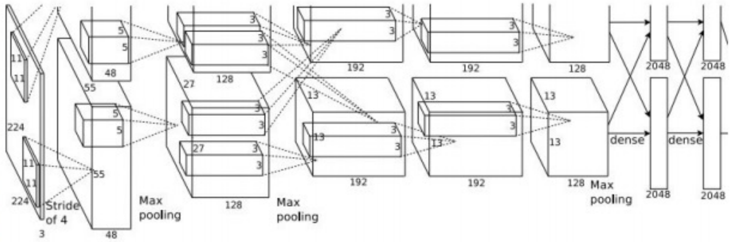
Correct box: (x', y', w', h')

Treat localization as a regression problem!

Stanford CS231n - http://cs231n.stanford.edu

Classification + Localization

This image is CC0 public domain



Correct label:
Cat

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

Fully Connected:
4096 to 1000

Softmax Loss

Vector:
4096

Multitask Loss

+ → Loss

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x', y', w', h')

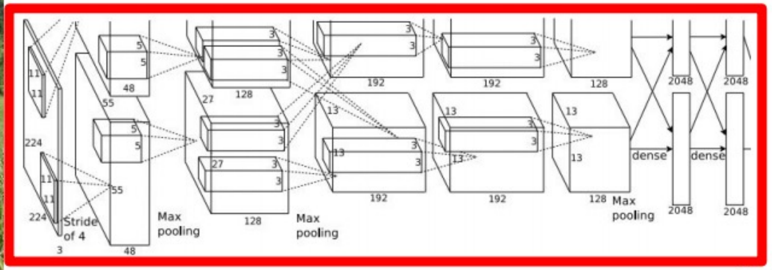
Treat localization as a regression problem!

Stanford CS231n - http://cs231n.stanford.edu

ing

Classification + Localization

This image is CC0 public domain



Often pretrained on ImageNet
(Transfer learning)

Vector: 4096

Fully Connected: 4096 to 1000

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

Correct label: Cat

Softmax Loss

+ → **Loss**

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

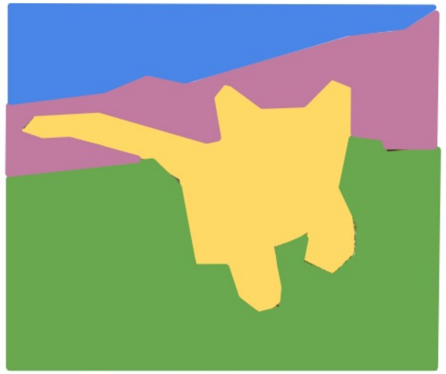
Correct box: (x', y', w', h')

Treat localization as a regression problem!

Stanford CS231n - http://cs231n.stanford.edu

Other Computer Vision Tasks

Semantic Segmentation

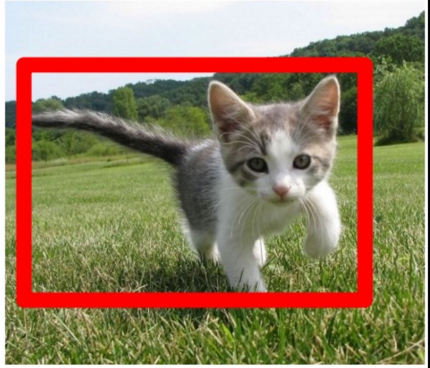


GRASS, CAT, TREE, SKY

No objects, just pixels

Balanced equation

Classification + Localization

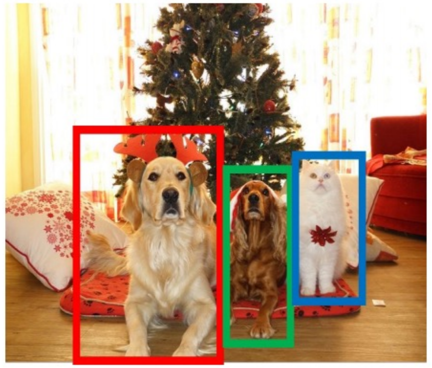


CAT

Single Object

Over-determined

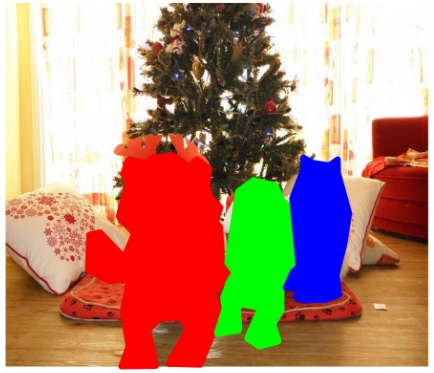
Object Detection



DOG, DOG, CAT

Over-determined

Instance Segmentation



DOG, DOG, CAT

Multiple Object

This image is CC0 public domain

Over-determined

Super-resolution

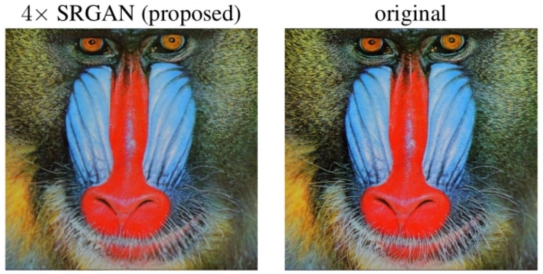


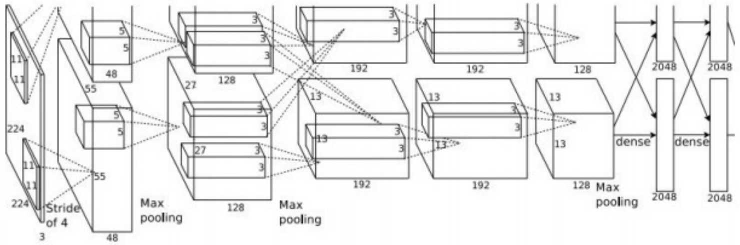
Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Under-determined

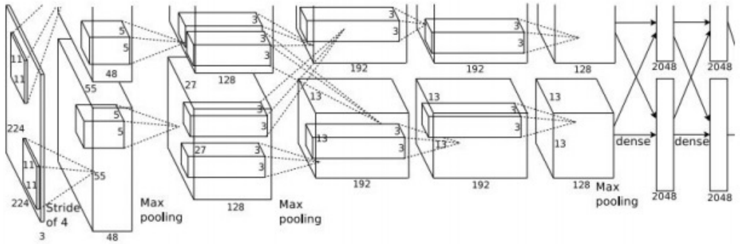
Stanford CS231n - http://cs231n.stanford.edu

Object Detection as Regression?

Each image needs a different number of outputs!

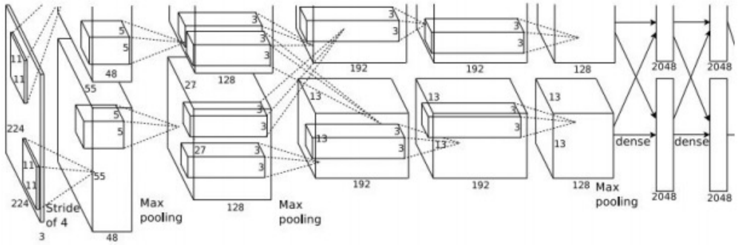


CAT: (x, y, w, h) 4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers



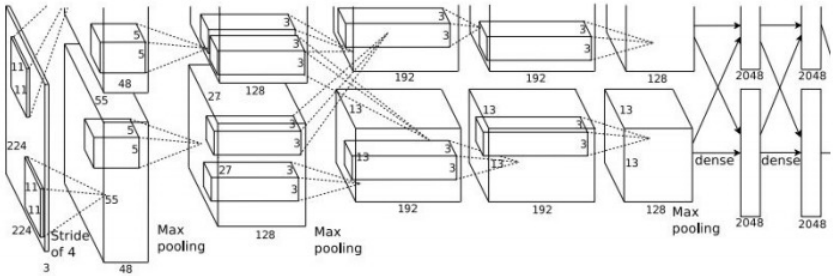
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Stanford CS231n - http://cs231n.stanford.edu

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

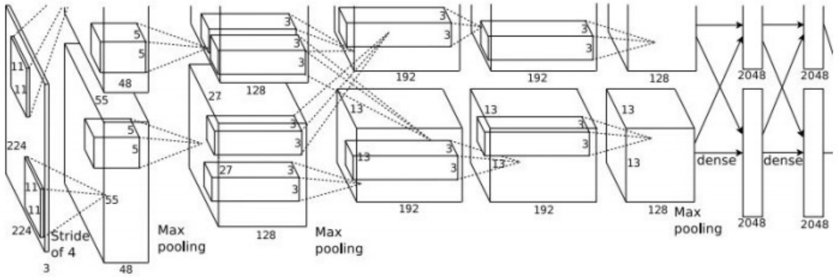


Dog? NO
Cat? NO
Background? YES

Stanford CS231n - http://cs231n.stanford.edu

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

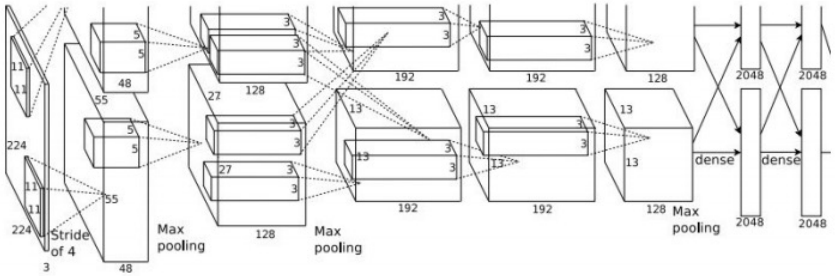


Dog? YES
Cat? NO
Background? NO

Stanford CS231n - http://cs231n.stanford.edu

Object Detection as Classification: Sliding Window

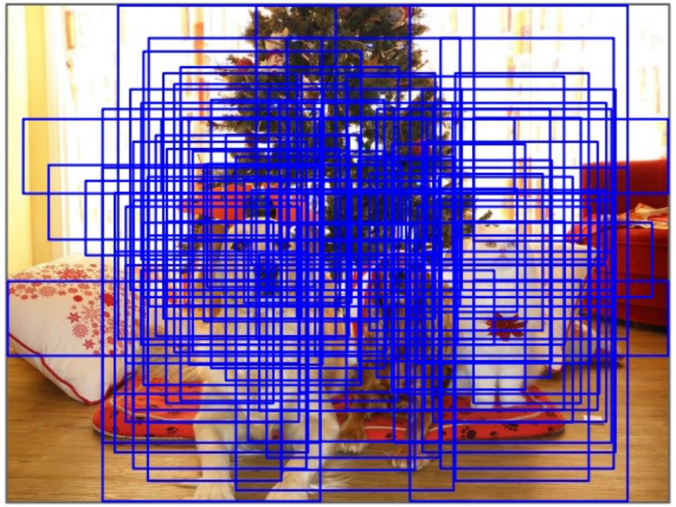
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



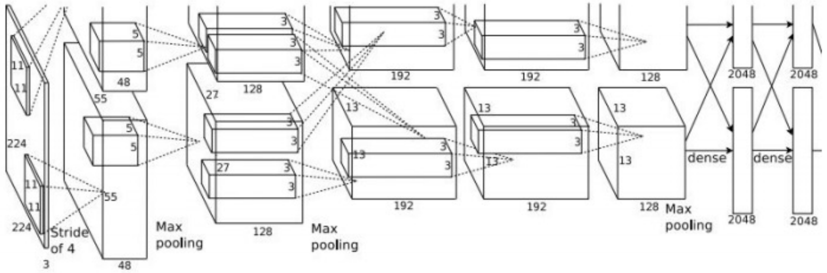
Dog? NO
Cat? YES
Background? NO

Stanford CS231n - http://cs231n.stanford.edu

Object Detection as Classification: Sliding Window



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

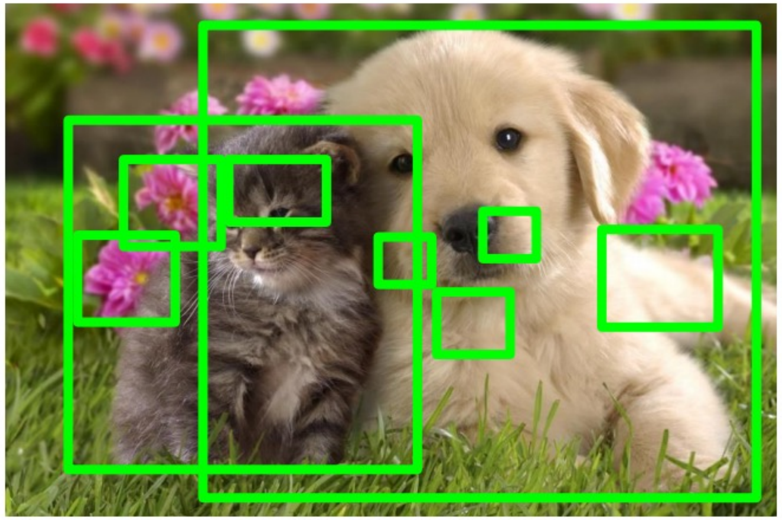
Stanford CS231n - http://cs231n.stanford.edu

Solution: First apply a fixed ROI scheme to pull out "blobs" of interest

(Image source: van de Sande et al. ICCV'11)

Region Proposals / Selective Search

- Find "blobby" image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU

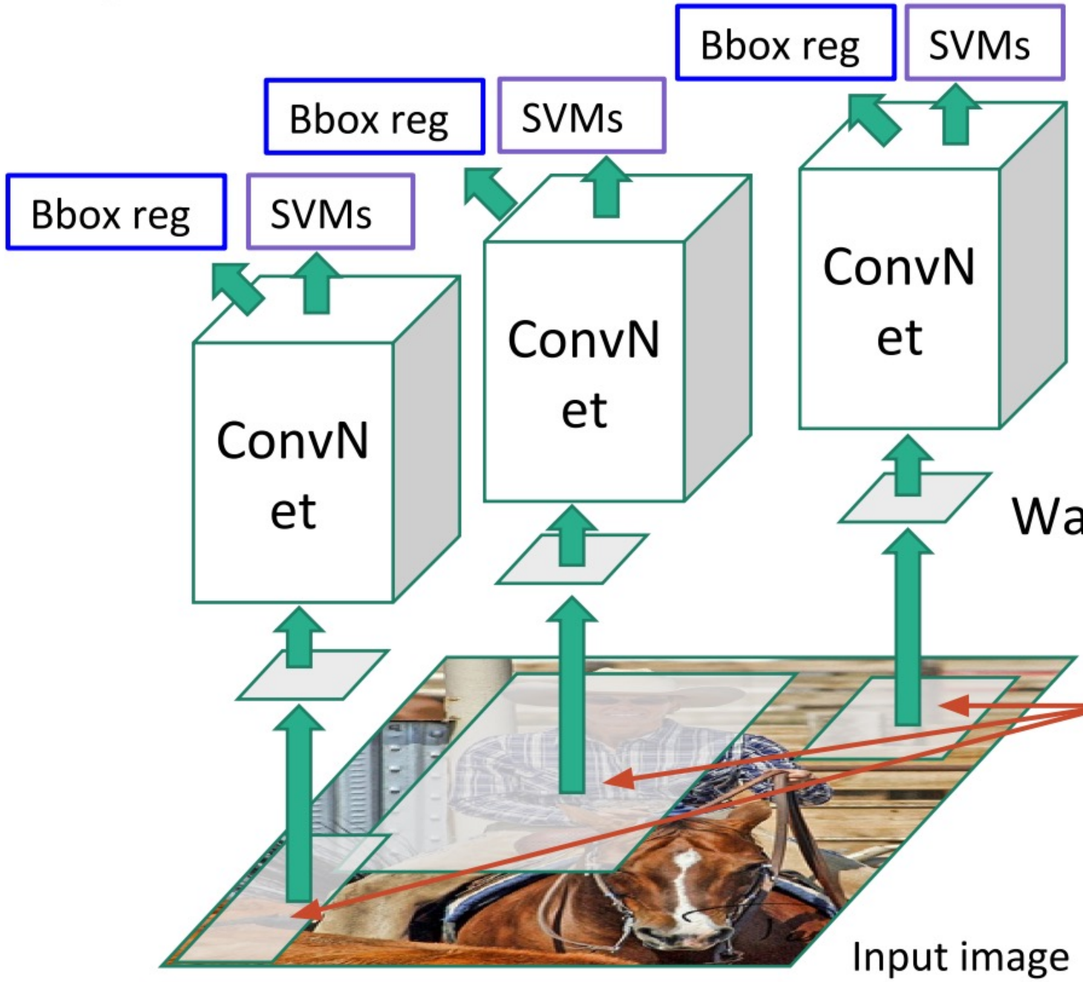


Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

Note: Training dataset has marked boxes, so don't necessarily need to do selective search for training, just evaluation/testing

Stanford CS231n - http://cs231n.stanford.edu

R-CNN



Linear Regression for bounding box offsets

Classify regions with SVMs

Forward each region through ConvNet

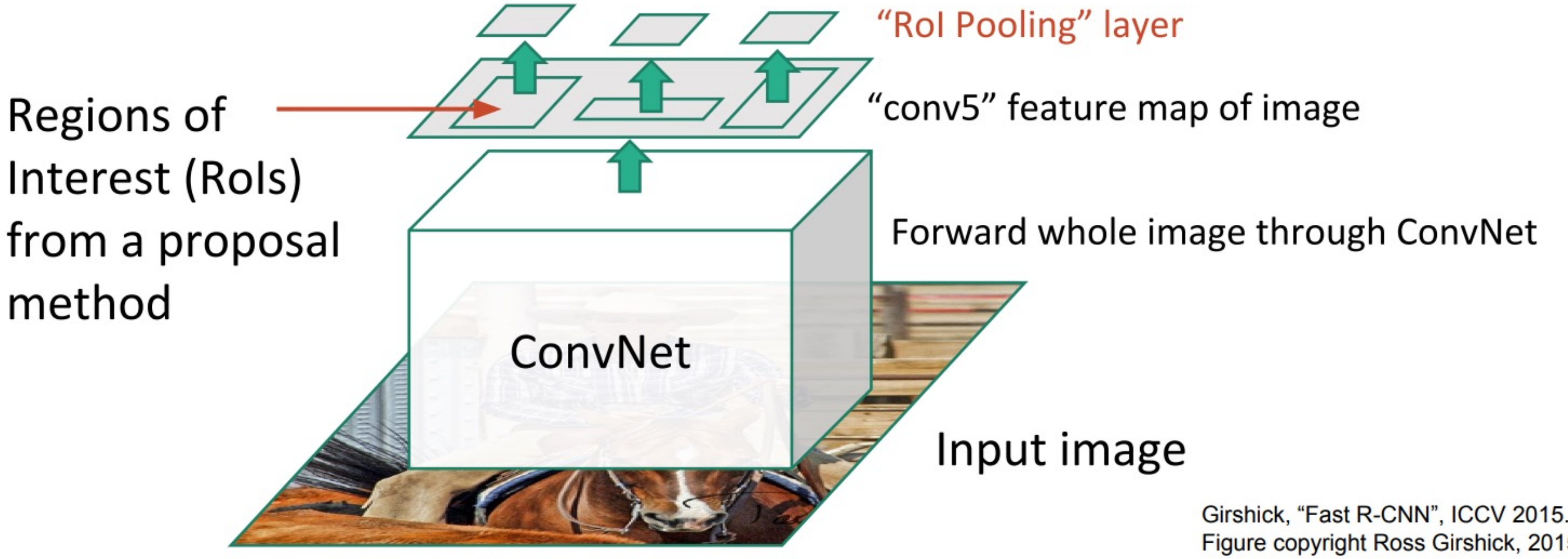
Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](). Reproduced with permission.

Stanford CS231n - http://cs231n.stanford.edu

Fast R-CNN

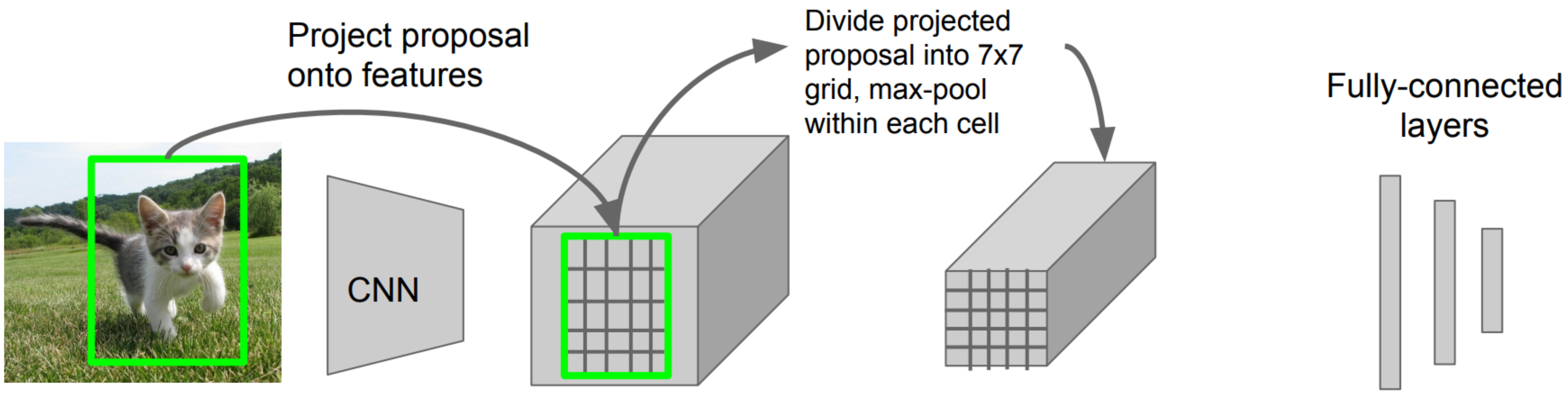


(Main features first, instead of repeating 2000X)

Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Stanford CS231n - http://cs231n.stanford.edu

Fast R-CNN: RoI Pooling



Hi-res input image:
3 x 640 x 480
with region
proposal

Hi-res conv features:
512 x 20 x 15;

Projected region
proposal is e.g.
512 x 18 x 8
(varies per proposal)

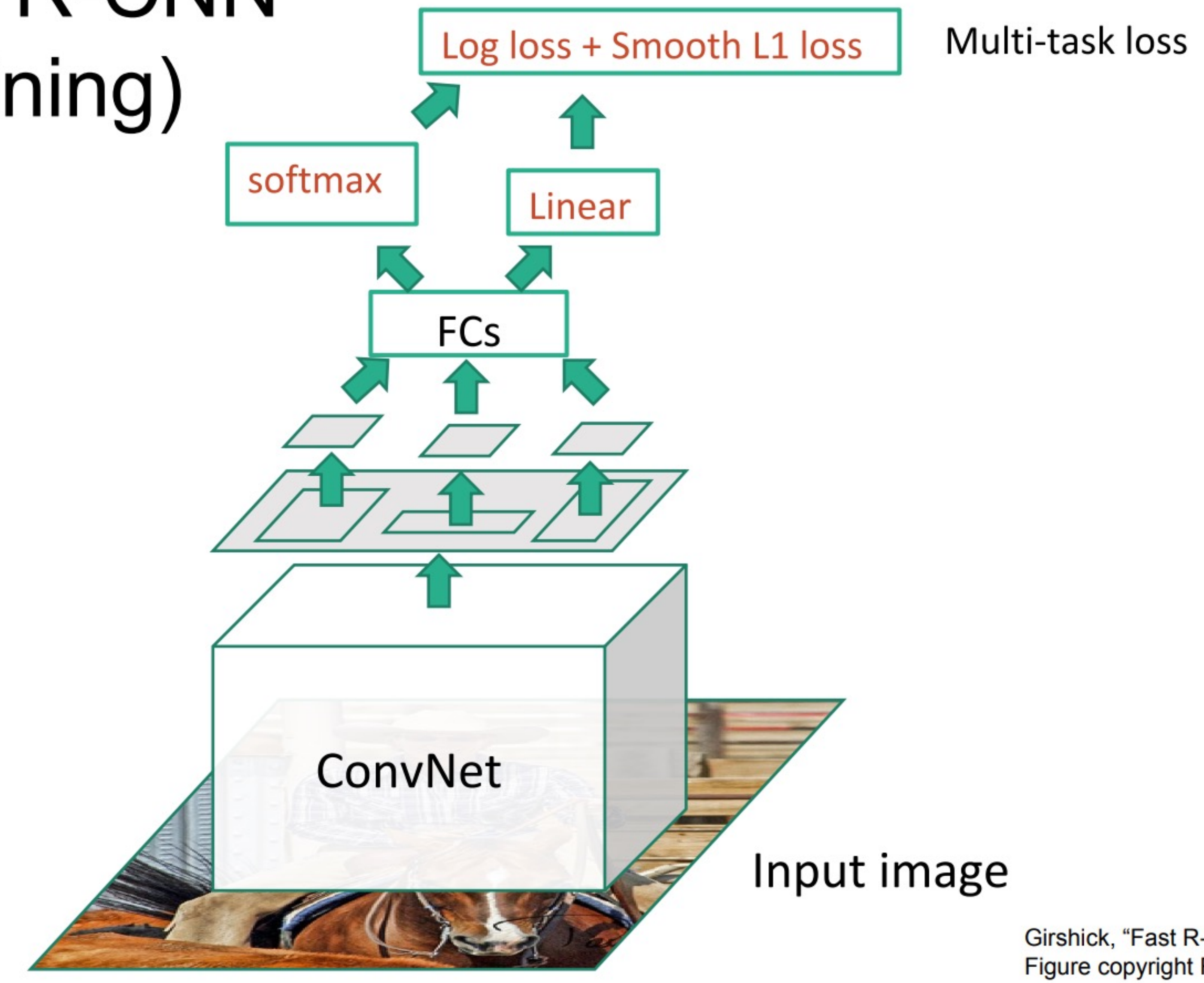
RoI conv features:
512 x 7 x 7
for region proposal

Fully-connected layers expect
low-res conv features:
512 x 7 x 7

Girshick, “Fast R-CNN”, ICCV 2015.

Stanford CS231n - http://cs231n.stanford.edu

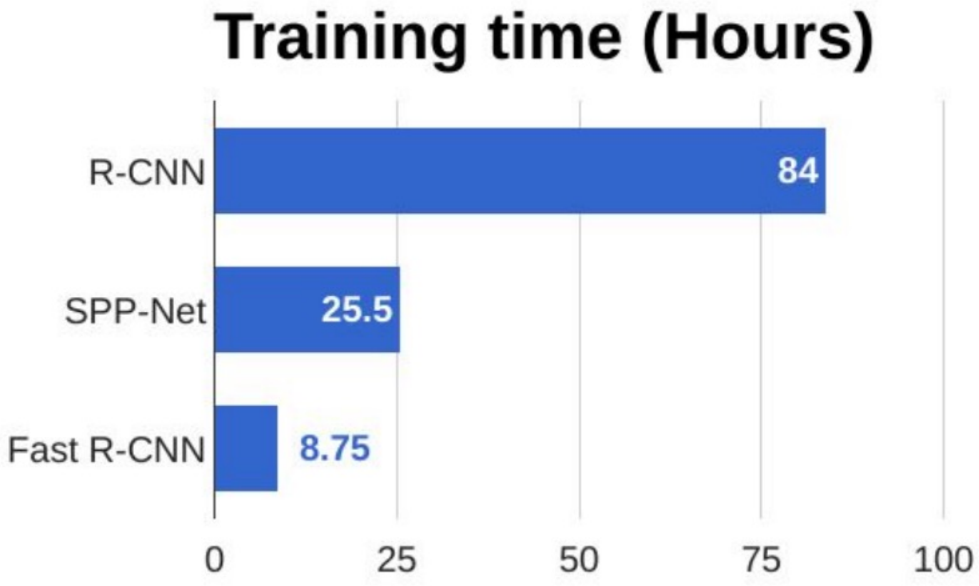
Fast R-CNN (Training)

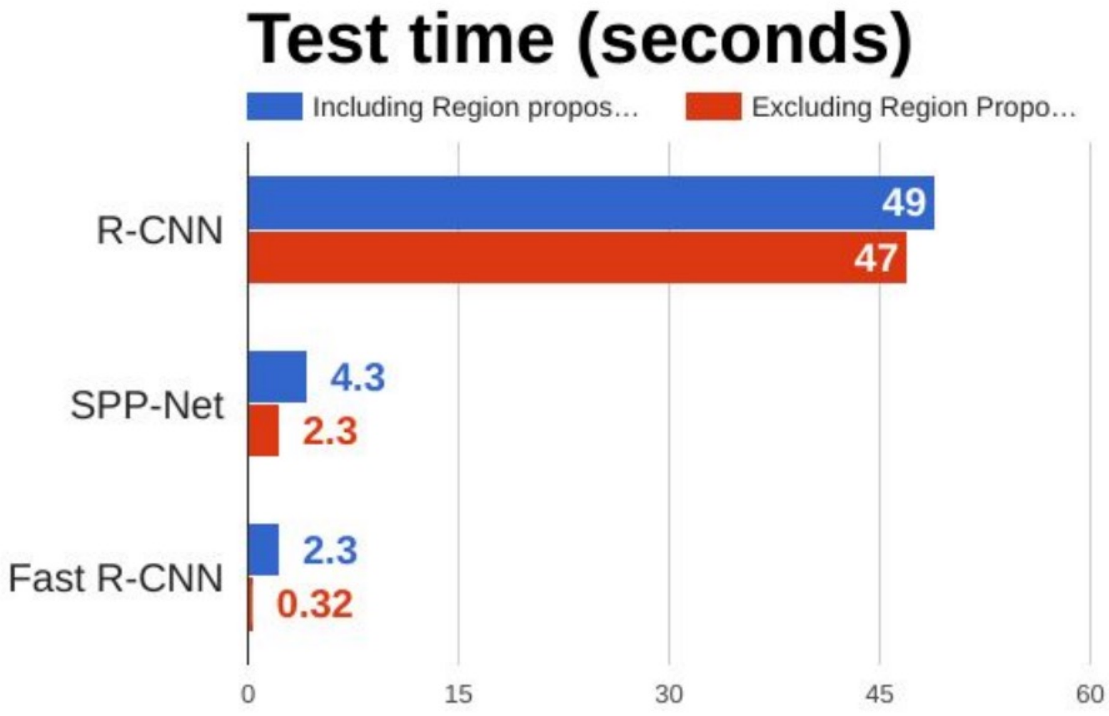


Girshick, "Fast R-CNN", ICCV 2015.

Stanford CS231n - http://cs231n.stanford.edu

R-CNN vs SPP vs Fast R-CNN





Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Stanford CS231n - http://cs231n.stanford.edu

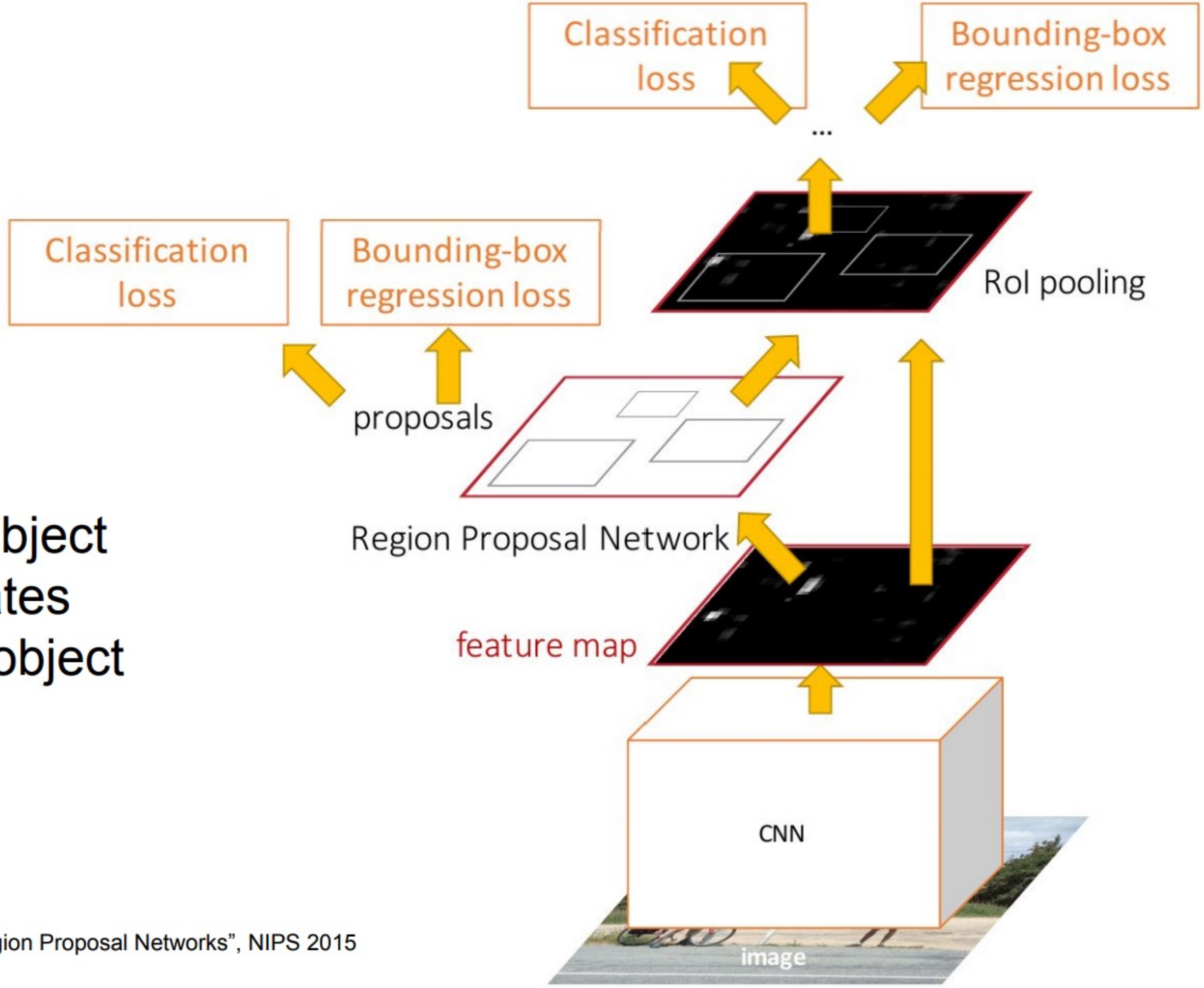
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



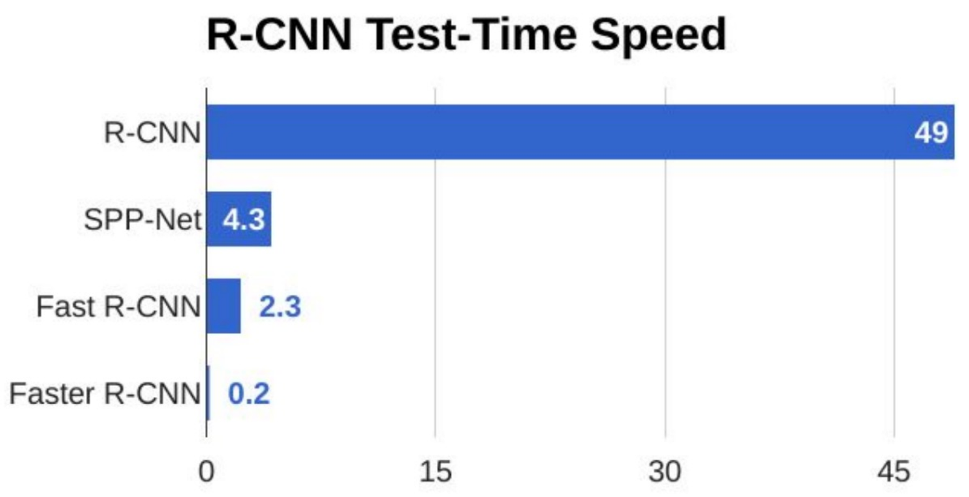
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

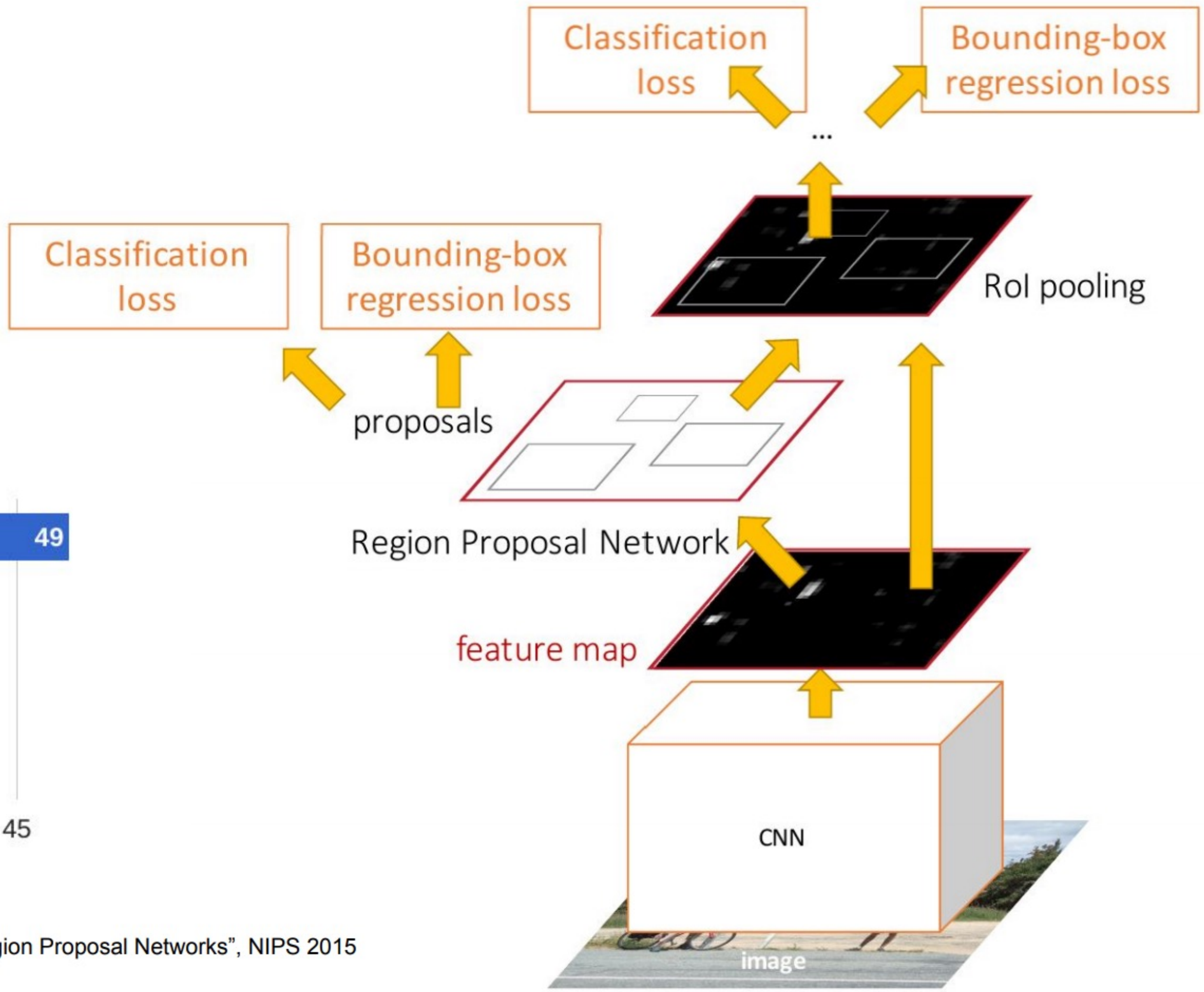
Stanford CS231n - http://cs231n.stanford.edu

Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features





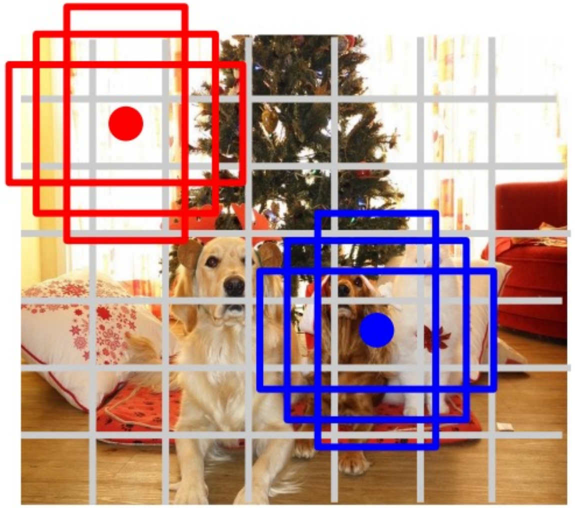
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Stanford CS231n - http://cs231n.stanford.edu

Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network!

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

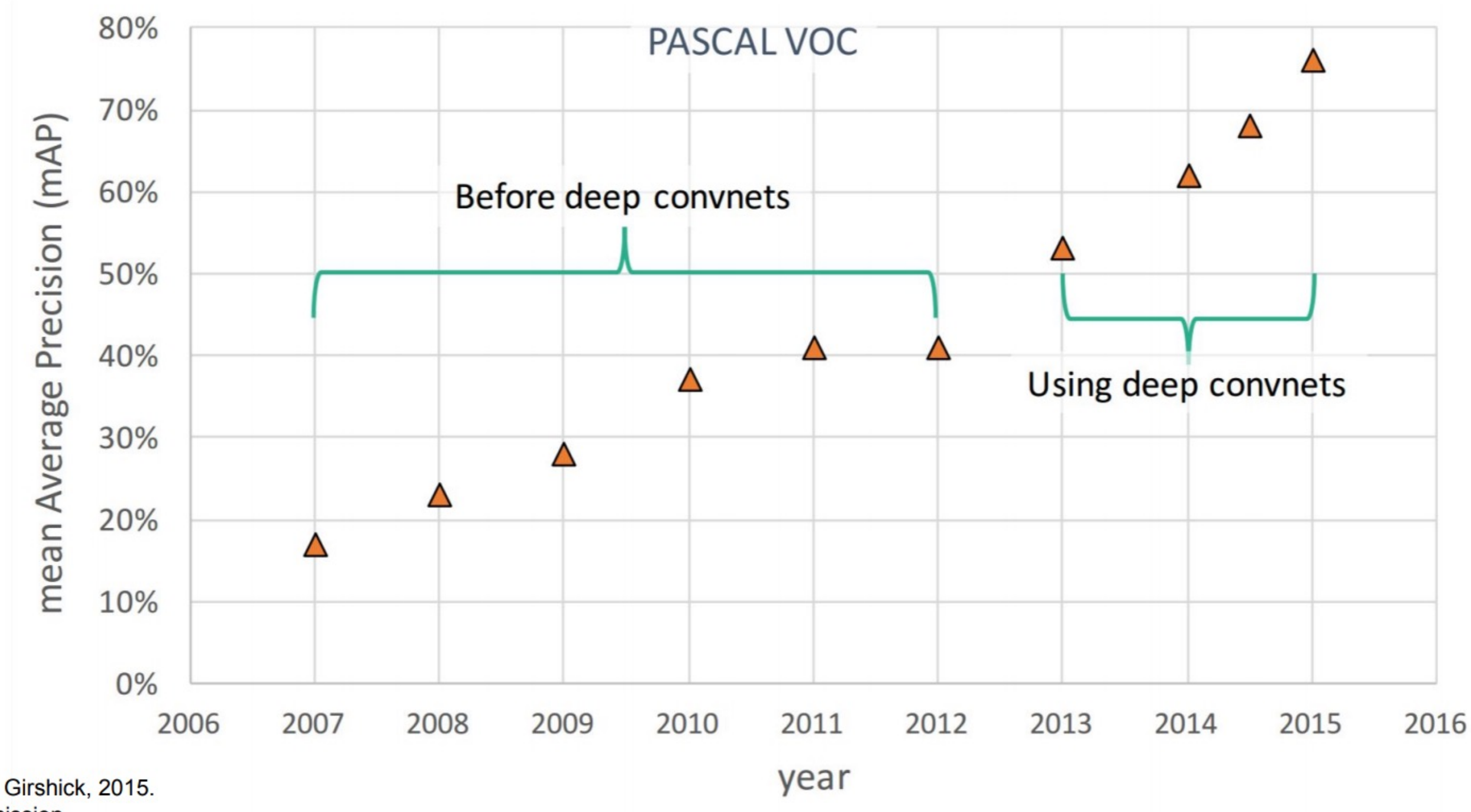
Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Stanford CS231n - http://cs231n.stanford.edu

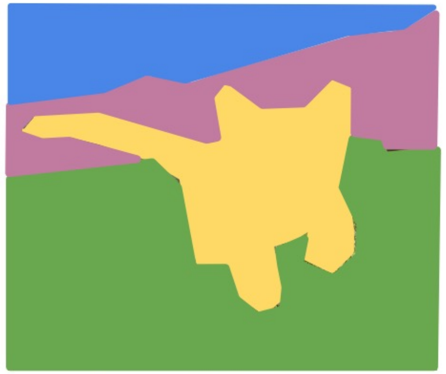
Object Detection: Impact of Deep Learning



Stanford CS231n - http://cs231n.stanford.edu

Other Computer Vision Tasks

Semantic Segmentation

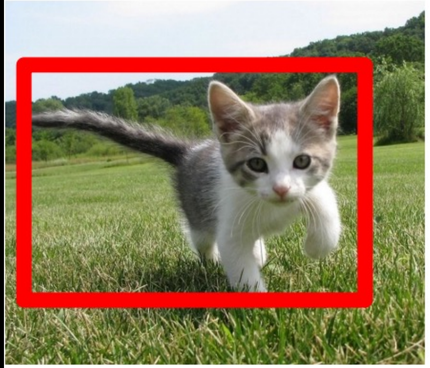


GRASS, CAT, TREE, SKY

No objects, just pixels

Balanced equation

Classification + Localization

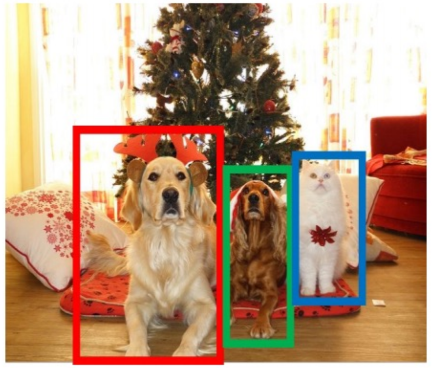


CAT

Single Object

Over-determined

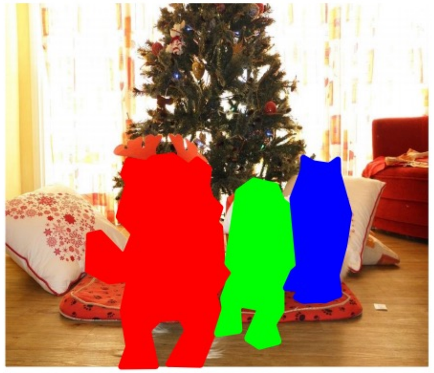
Object Detection



DOG, DOG, CAT

Over-determined

Instance Segmentation



DOG, DOG, CAT

Multiple Object

This image is CC0 public domain

Over-determined

Super-resolution

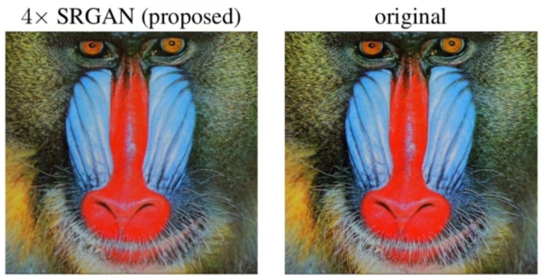


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

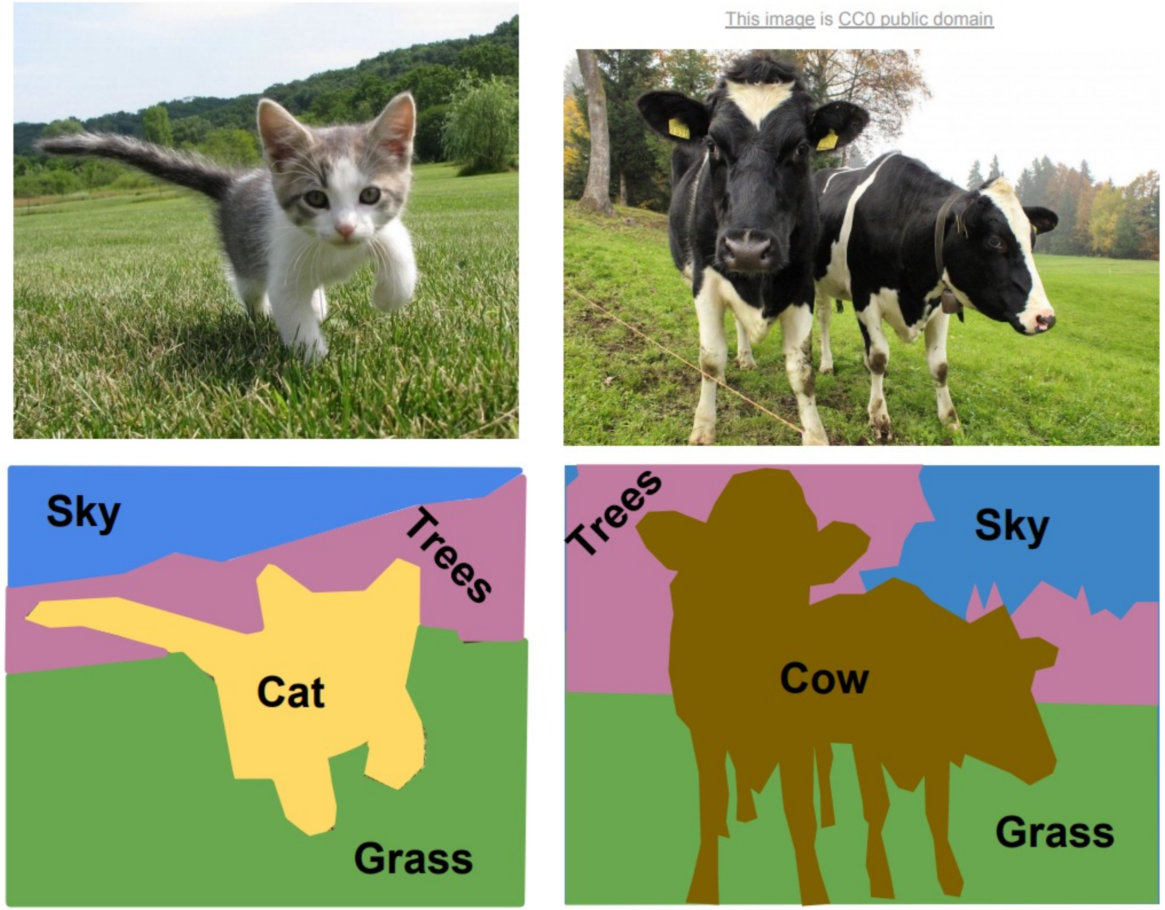
Under-determined

Stanford CS231n - http://cs231n.stanford.edu

Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels

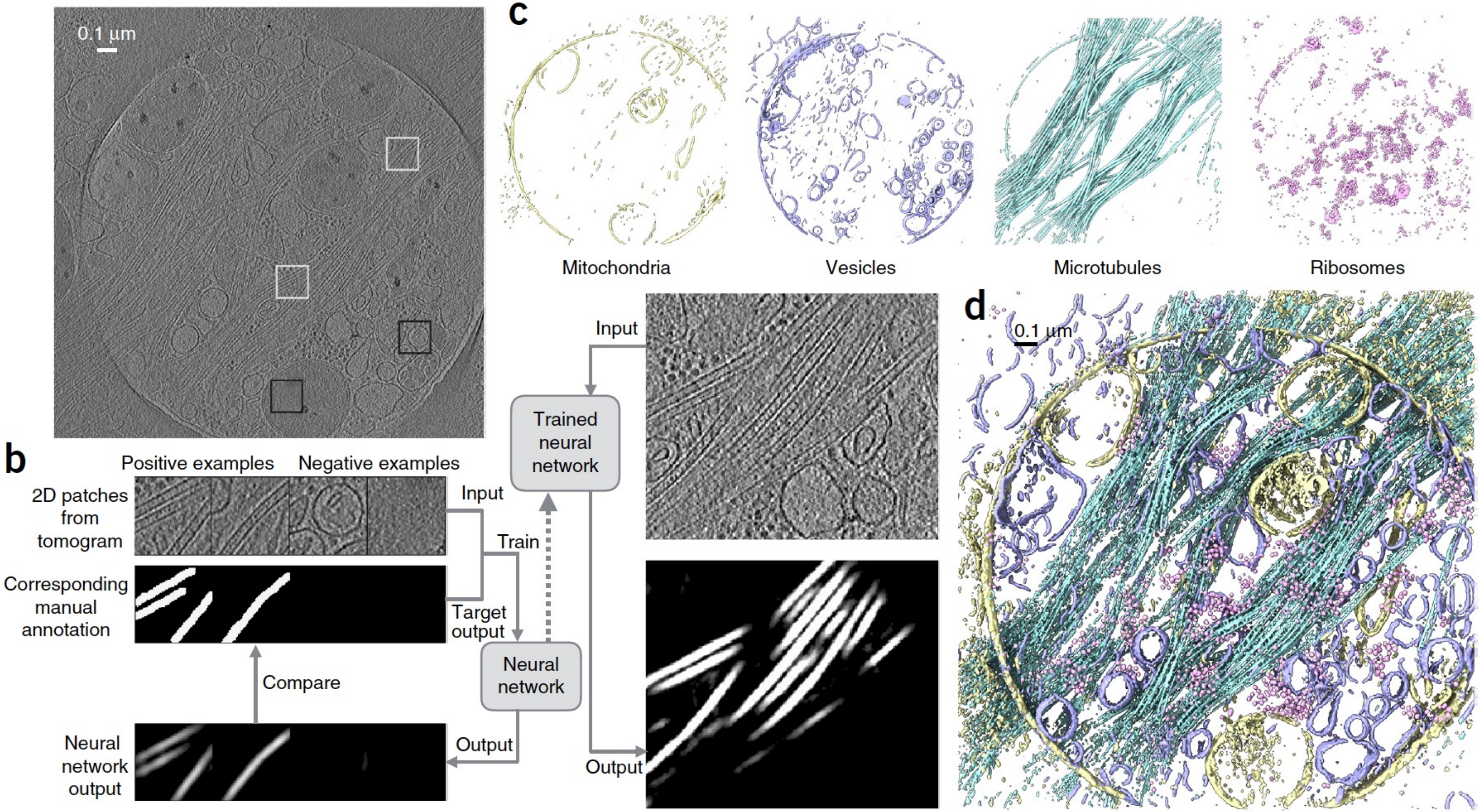


Stanford CS231n - http://cs231n.stanford.edu

Convolutional neural networks for automated annotation of cellular cryo-electron tomograms

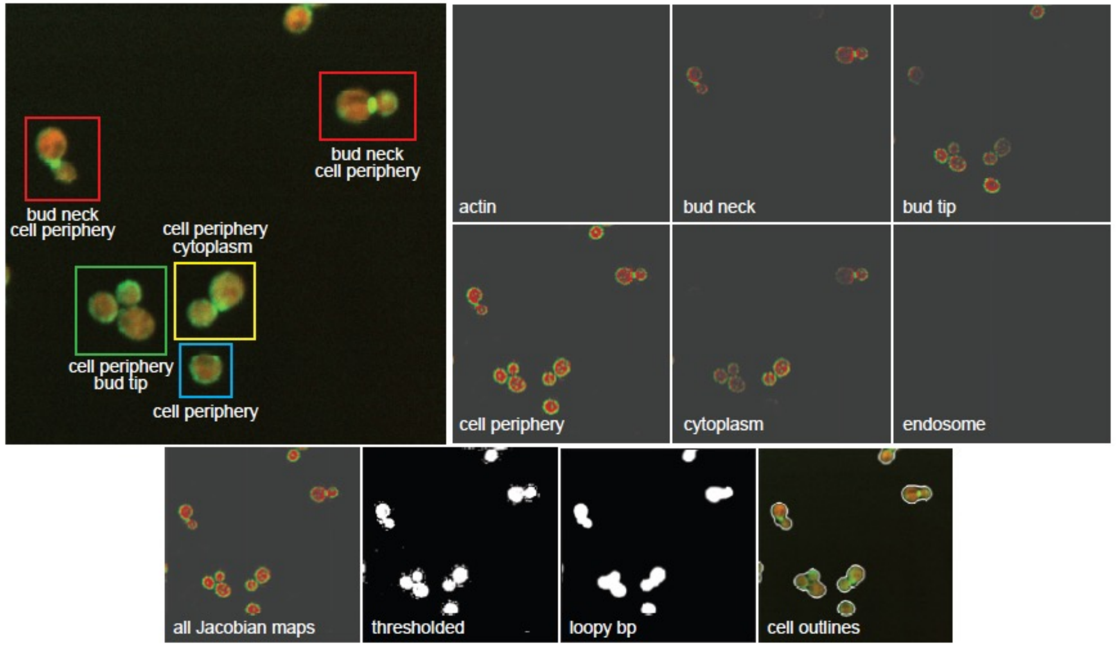
Muyuan Chen[1,2], Wei Dai[2,4], Stella Y Sun[2], Darius Jonasch[2], Cynthia Y He[3], Michael F Schmid[2], Wah Chiu[2] & Steven J Ludtke[2]

deep imaging

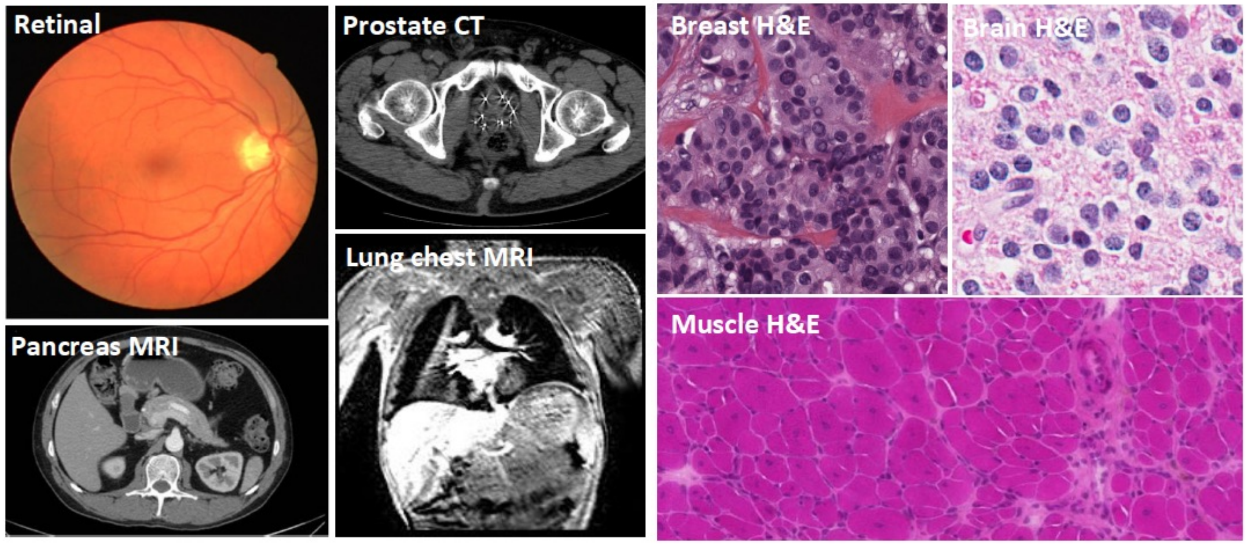


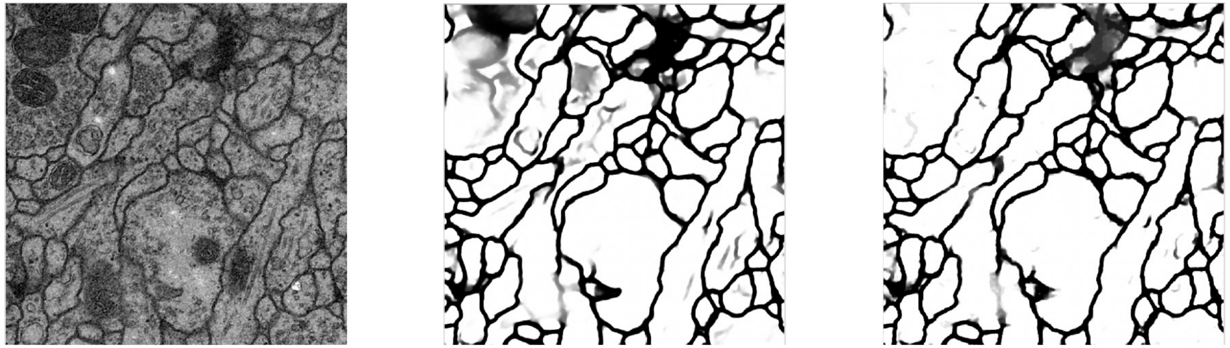
deep imaging

Other possible examples:



Oren Z. Kraus et al., “Classifying and Segmenting Microscopy Images Using Convolutional Multiple Instance Learning,” arXiv 2015

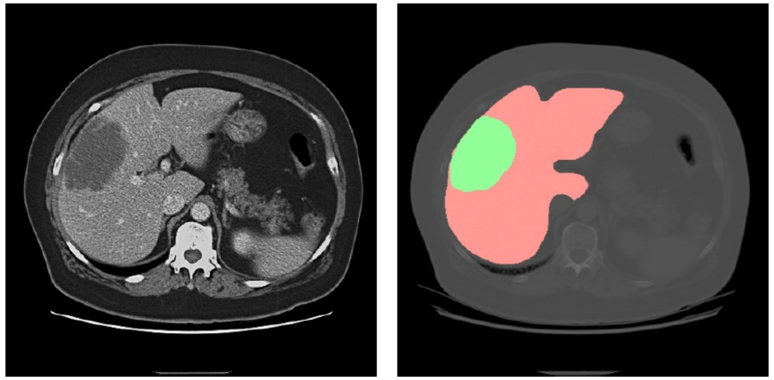




(a) Input image

(b) FC-ResNet with dropout at test time [17]

(c) Segmentation result of our pipeline

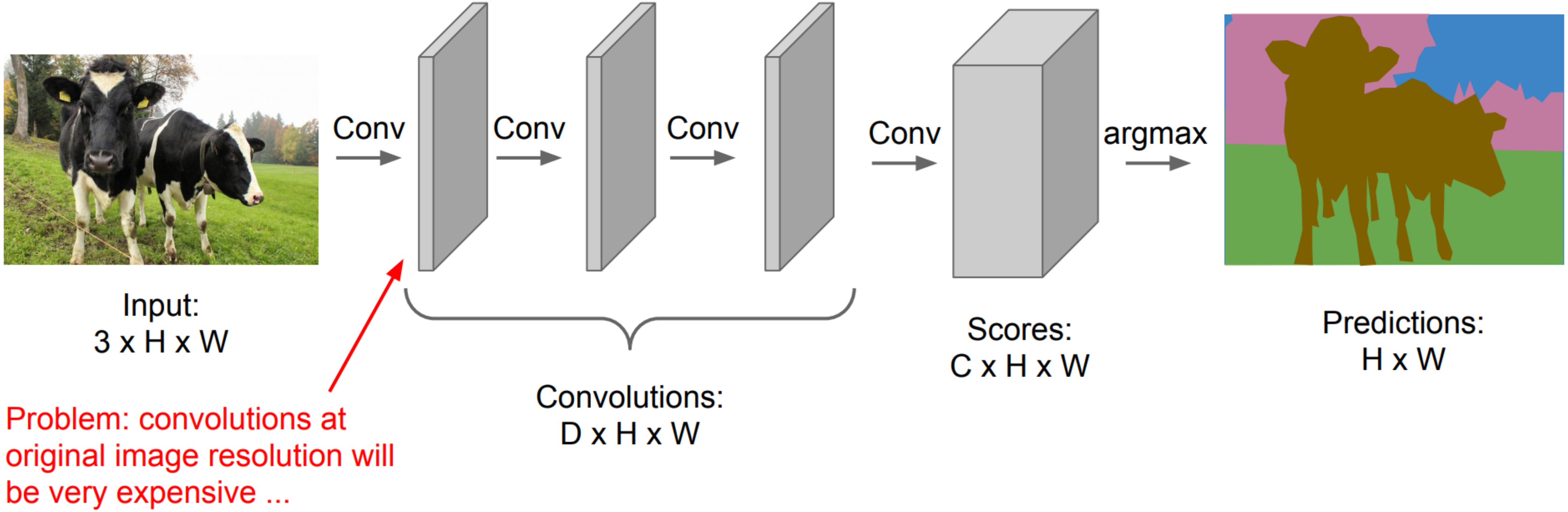


M. Drozdzal et al., Learning Normalized Inputs for Iterative Estimation in Medical Image Segmentation (2017)

Z. Zhang et al., Recent Advances in the Applications of Convolutional Neural Networks to Medical Image Contour Detection (2017)

Semantic Segmentation Idea: Fully Convolutional？

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

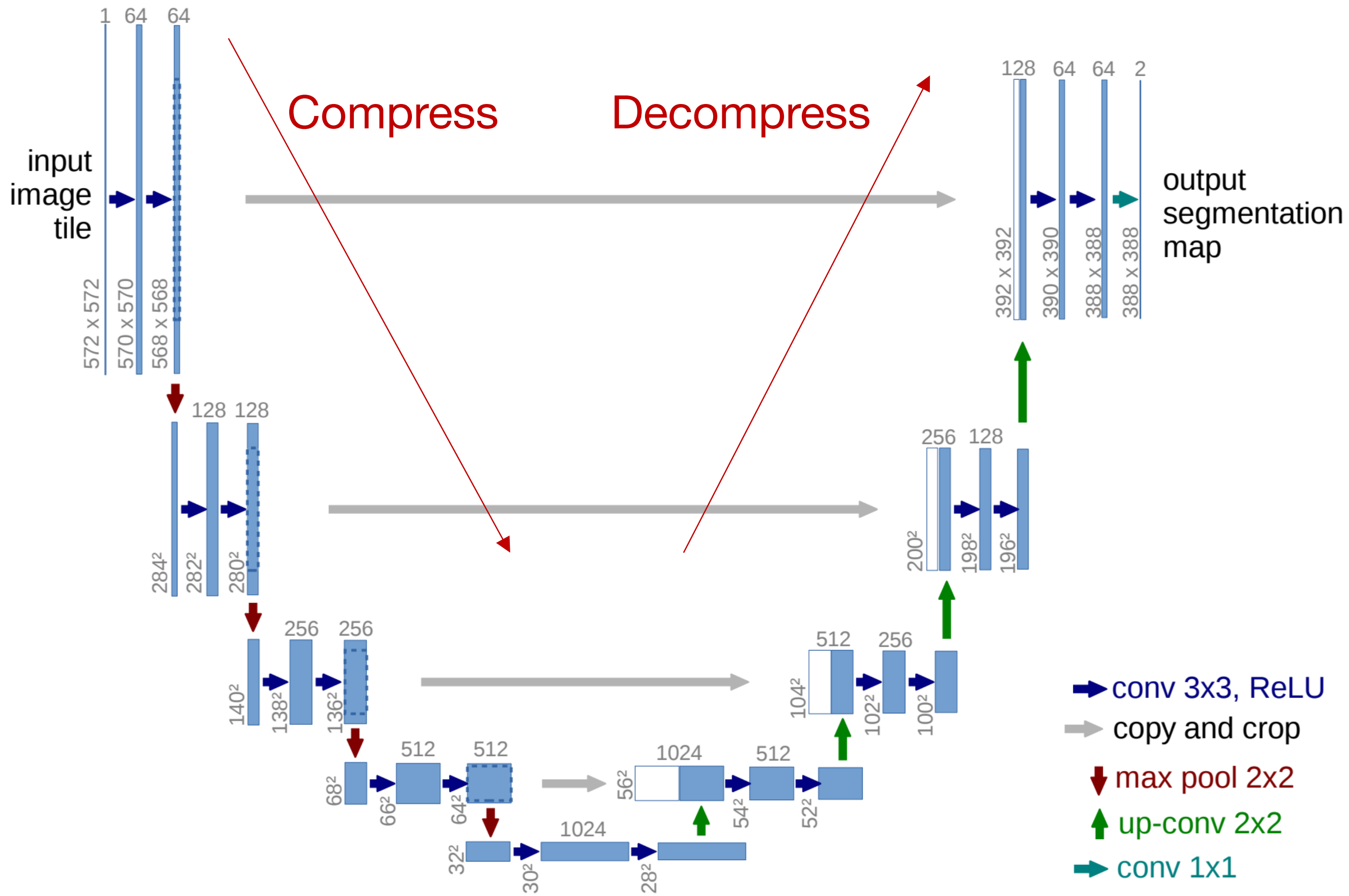


Stanford CS231n - http://cs231n.stanford.edu

Instead, *compress x-y dimensions of* input image

- Compress spatial features into learned filters
- Then, decompress learned filters back into same spatial dimensions

U-Net Architecture



U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
`ronneber@informatik.uni-freiburg.de`,
WWW home page: `http://lmb.informatik.uni-freiburg.de/`

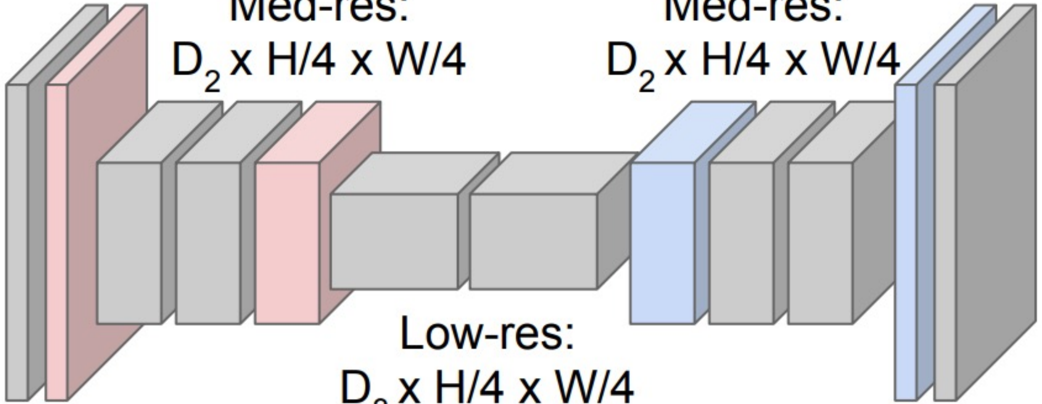
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: ???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

Med-res:
D_2 x H/4 x W/4

Low-res:
D_3 x H/4 x W/4

Med-res:
D_2 x H/4 x W/4

High-res:
D_1 x H/2 x W/2

Predictions:
H x 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

1	2
3	4

Input: 2 x 2

→

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

Output: 4 x 4

“Bed of Nails”

1	2
3	4

Input: 2 x 2

→

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

Output: 4 x 4

Stanford CS231n - http://cs231n.stanford.edu

In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

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

Input: 4 x 4

5	6
7	8

Output: 2 x 2

…

Rest of the network

Max Unpooling
Use positions from pooling layer

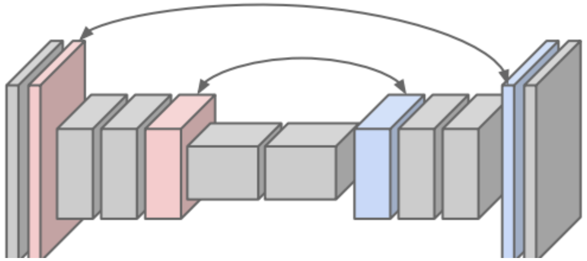
1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

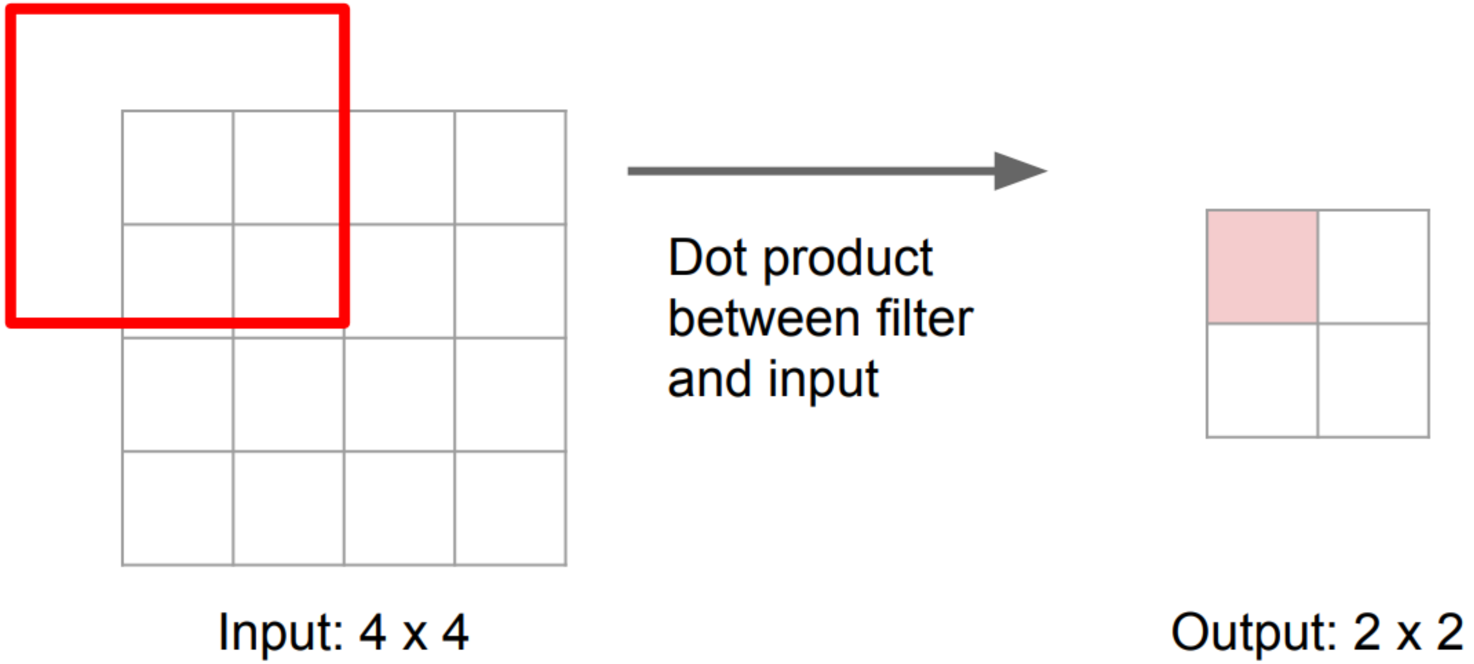
Corresponding pairs of downsampling and upsampling layers



Stanford CS231n - http://cs231n.stanford.edu

Learnable Upsampling: Transpose Convolution

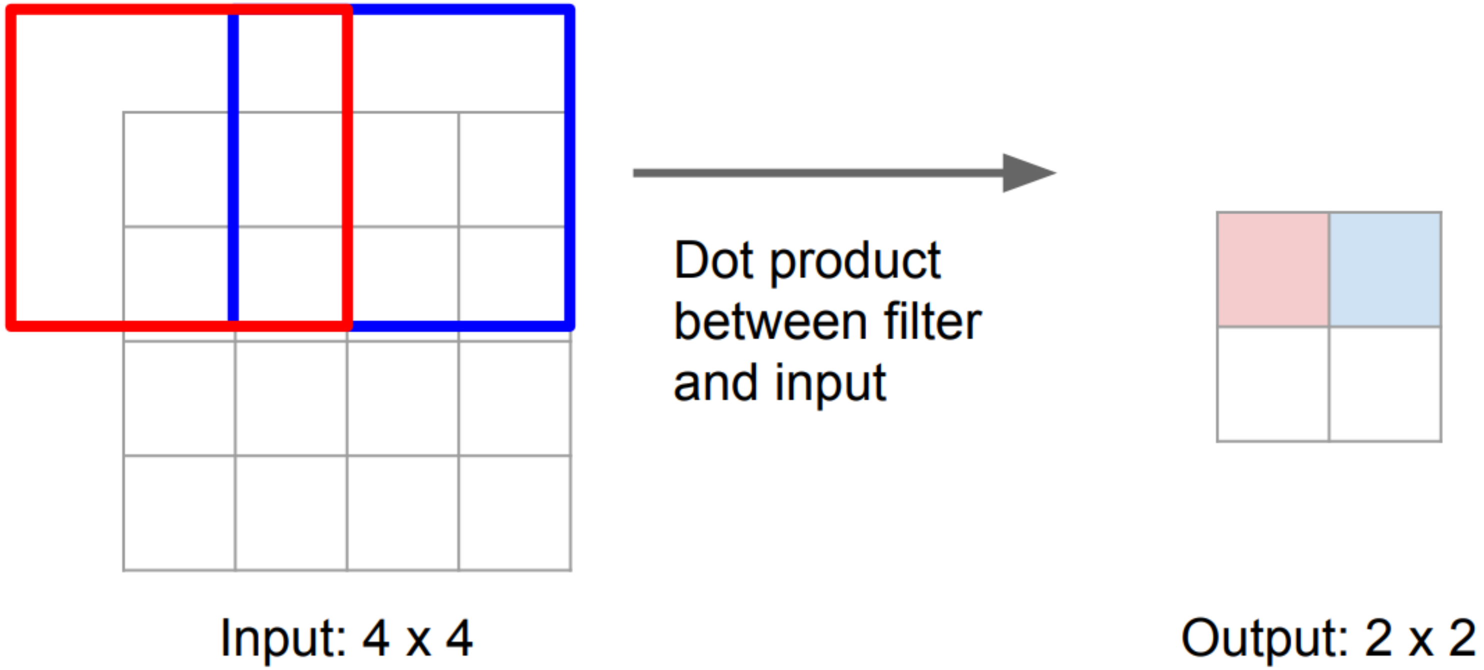
Recall: Normal 3 x 3 convolution, stride 2 pad 1



Stanford CS231n - http://cs231n.stanford.edu

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



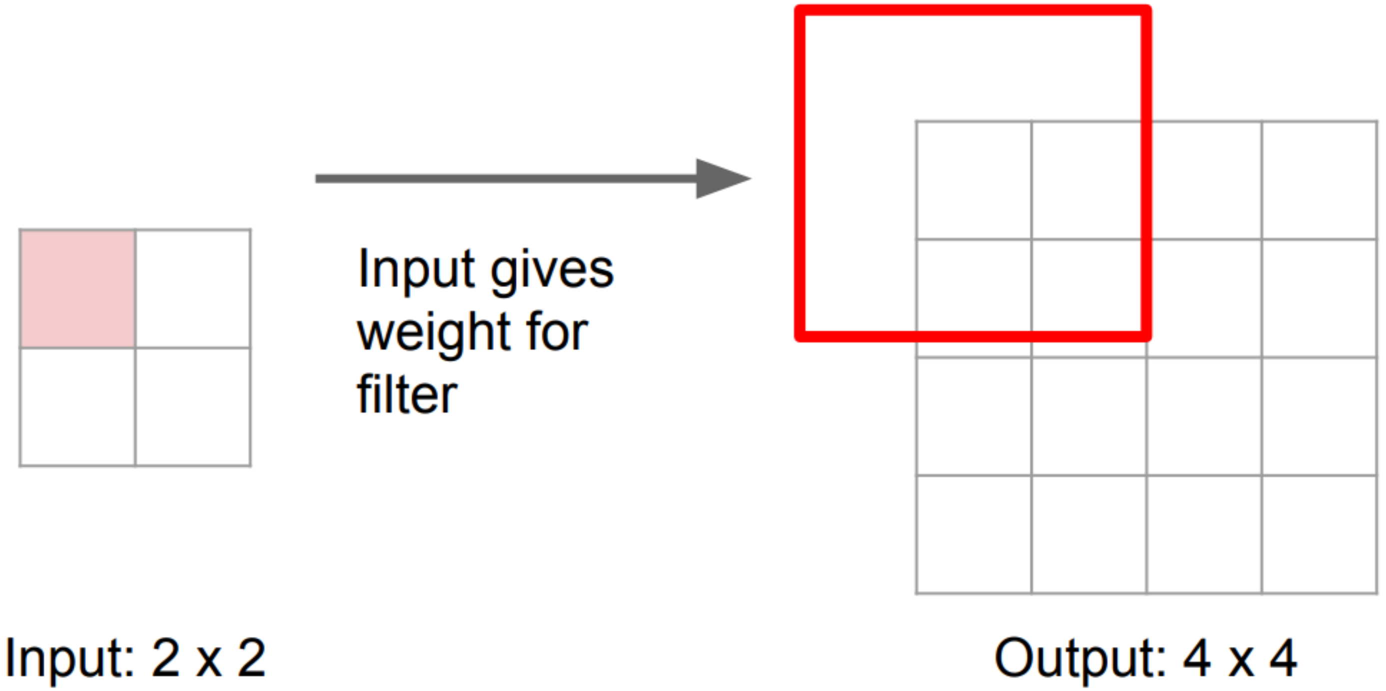
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Stanford CS231n - http://cs231n.stanford.edu

Learnable Upsampling: Transpose Convolution

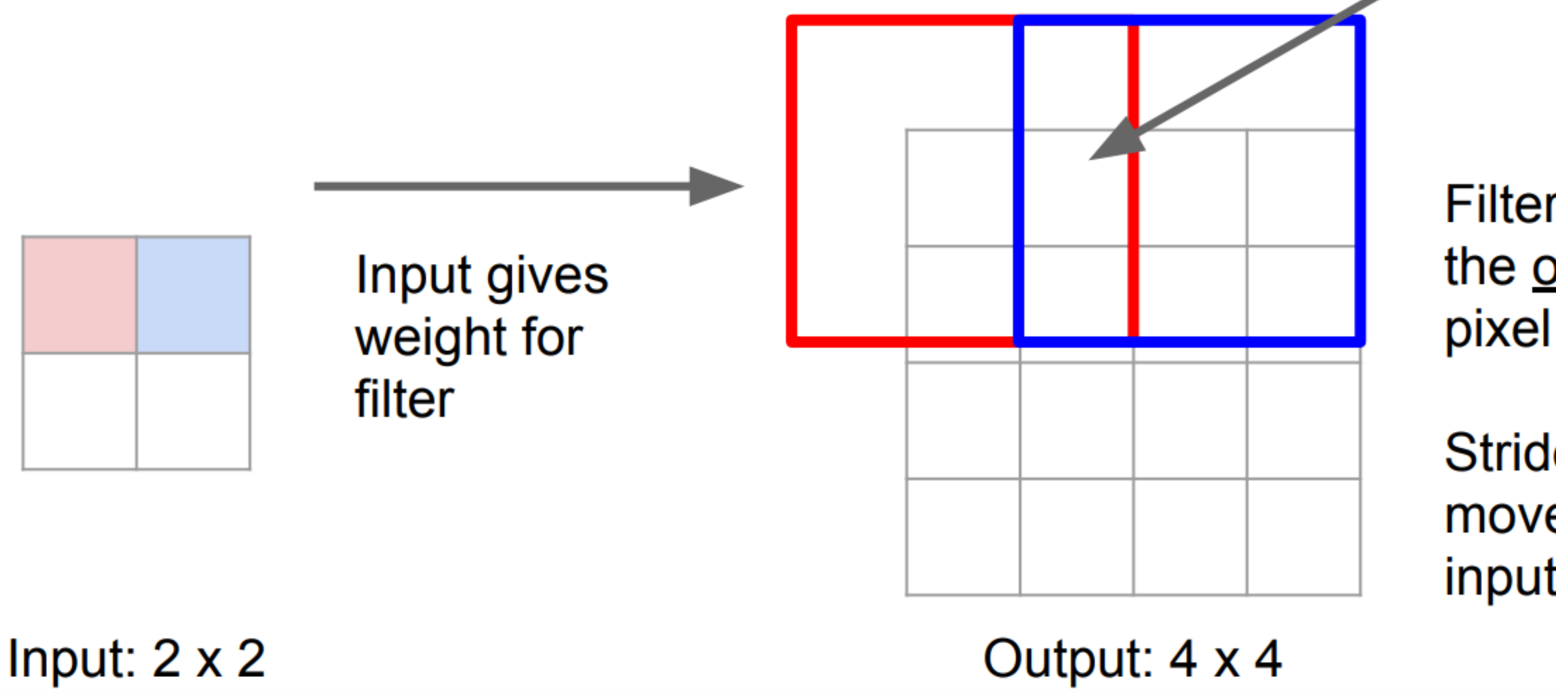
3 x 3 **transpose** convolution, stride 2 pad 1



Stanford CS231n - http://cs231n.stanford.edu

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1

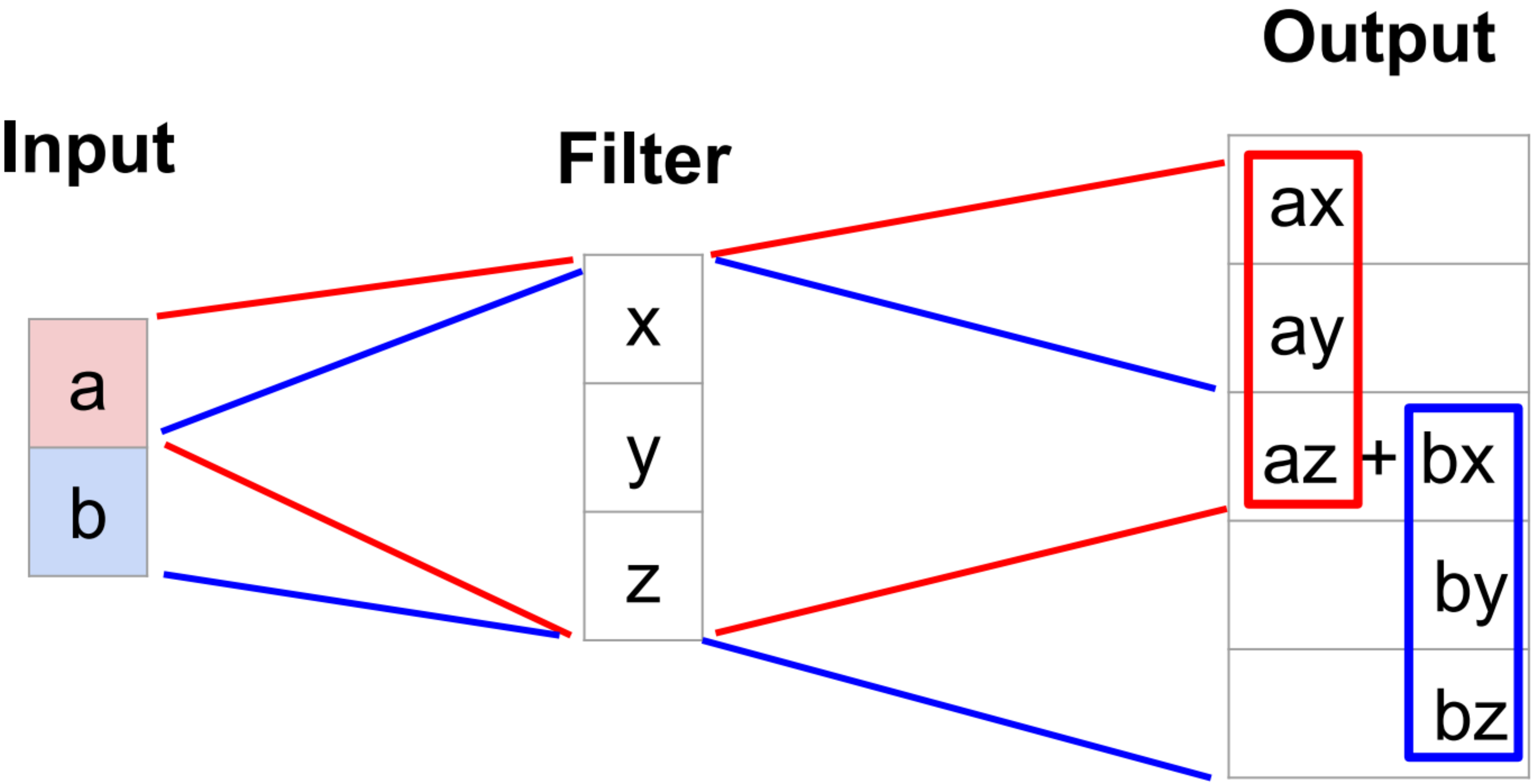


Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Stanford CS231n - http://cs231n.stanford.edu

Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

Stanford CS231n - http://cs231n.stanford.edu

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

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

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

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

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

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

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & 0 & x & y & x & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, <u>stride=2</u>, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, <u>stride=2</u>, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

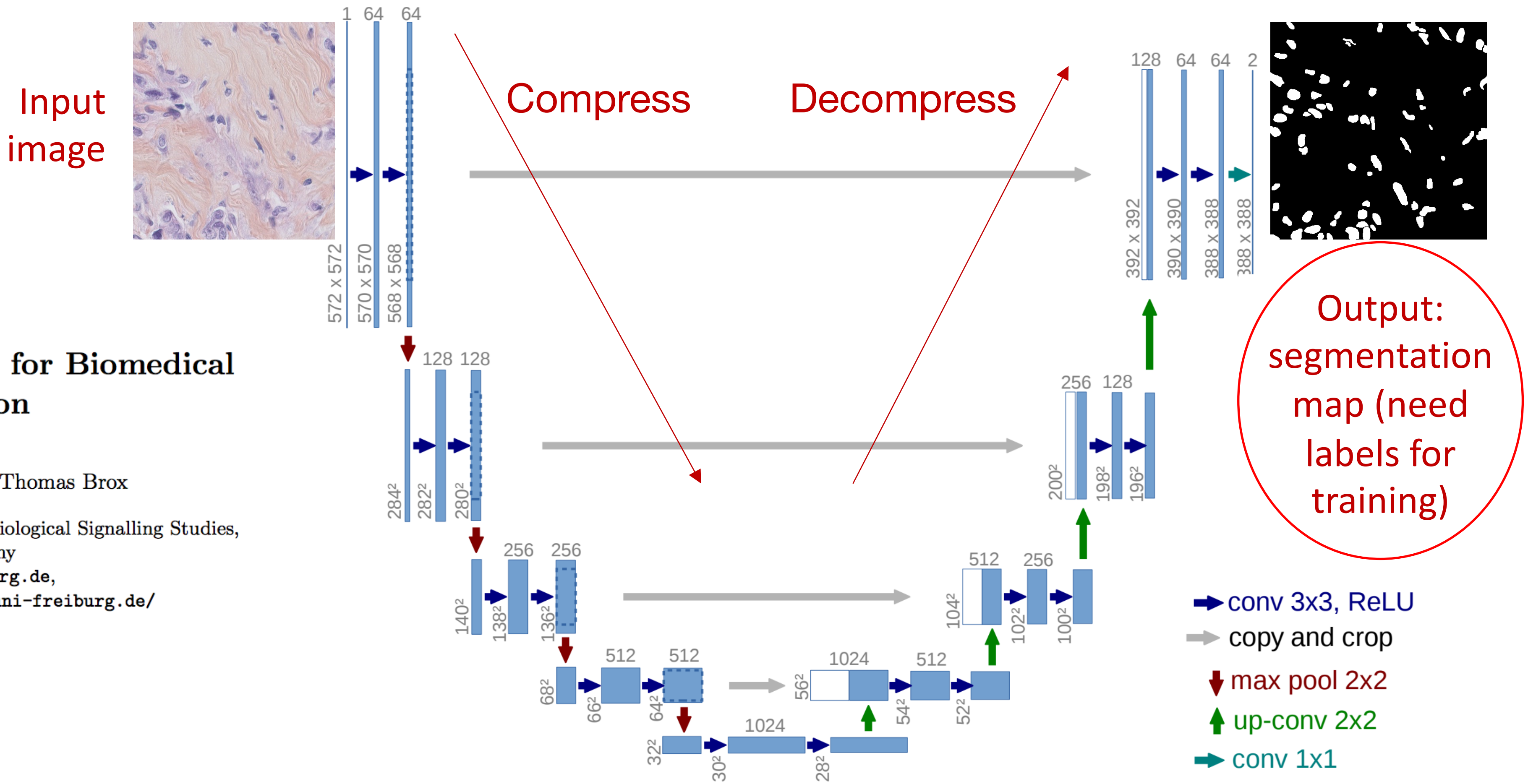
$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, convolution transpose is no longer a normal convolution!

Segmentation: need a map of classes for label

deep imaging

U-Net Architecture

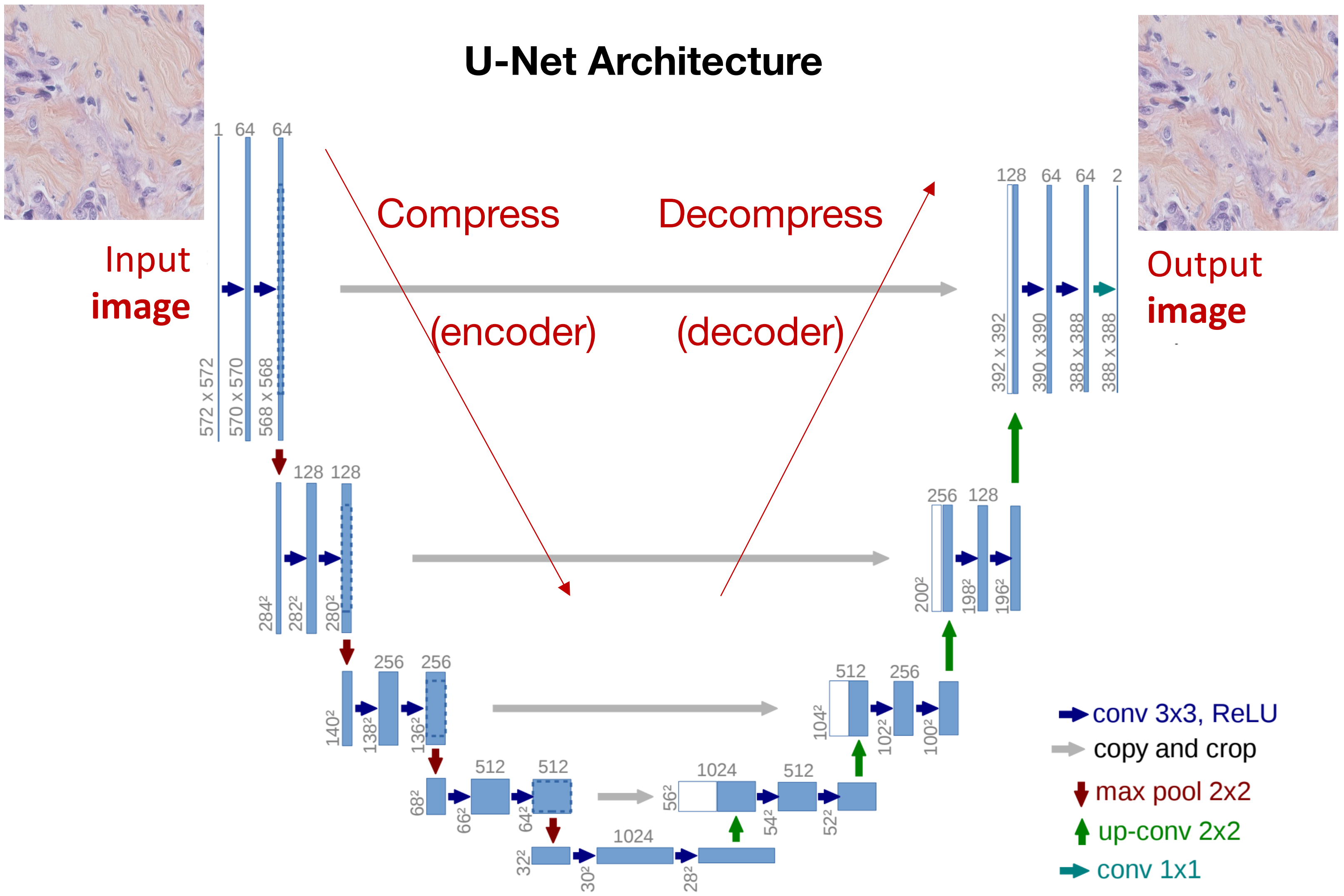


U-Net: Convolutional Networks for Biomedical Image Segmentation

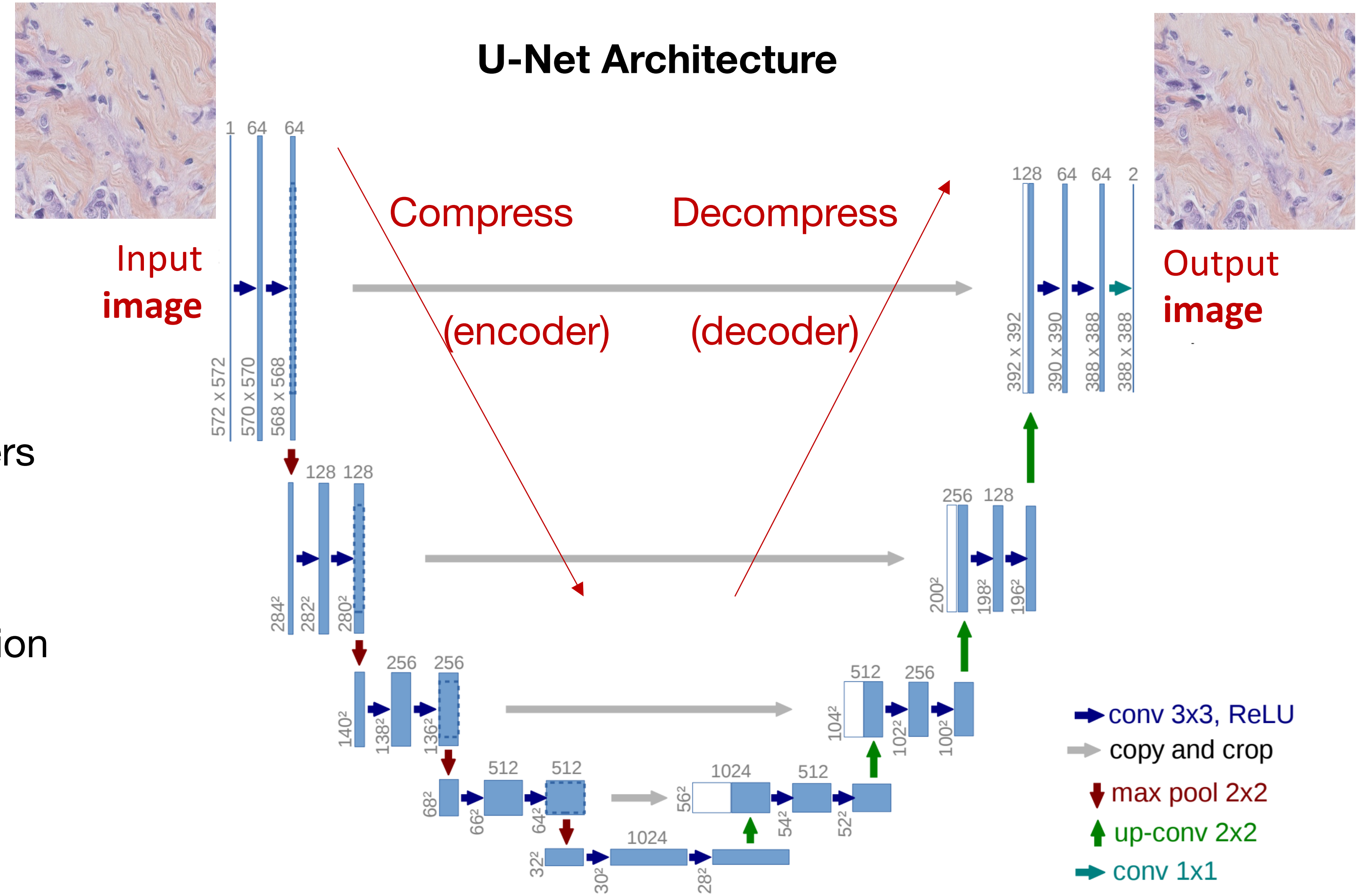
Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Instead, *compress x-y dimensions of* input image



Instead, *compress x-y dimensions of* input image

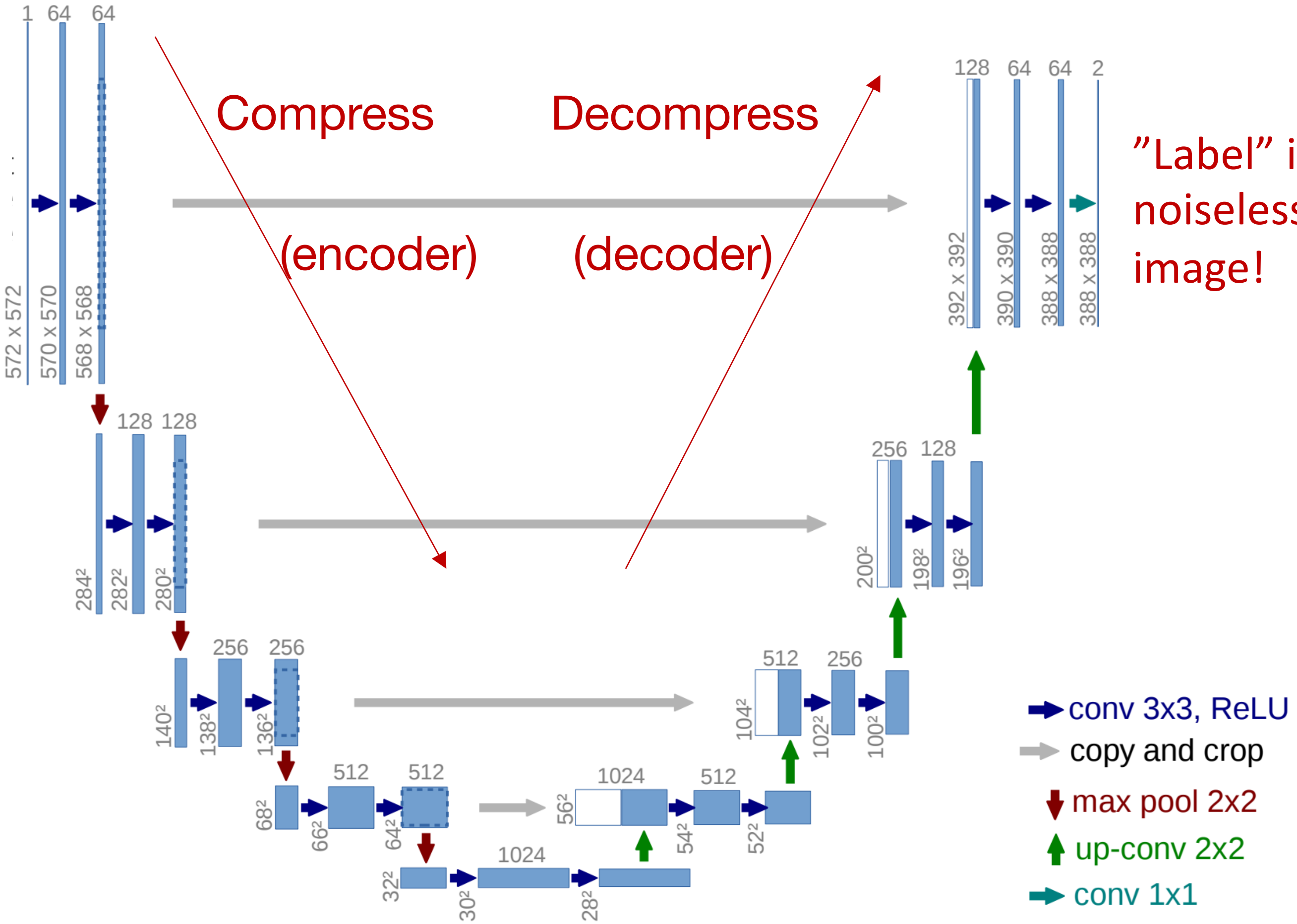


- Compress spatial features into learned filters
- Then, decompress learned filters back into same spatial dimensions
- **Termed an autoencoder**
- Analogous to image compression
- A pretty powerful idea…

Another example: Denoising Autoencoder

U-Net Architecture

Input noisy image



”Label” is noiseless image!

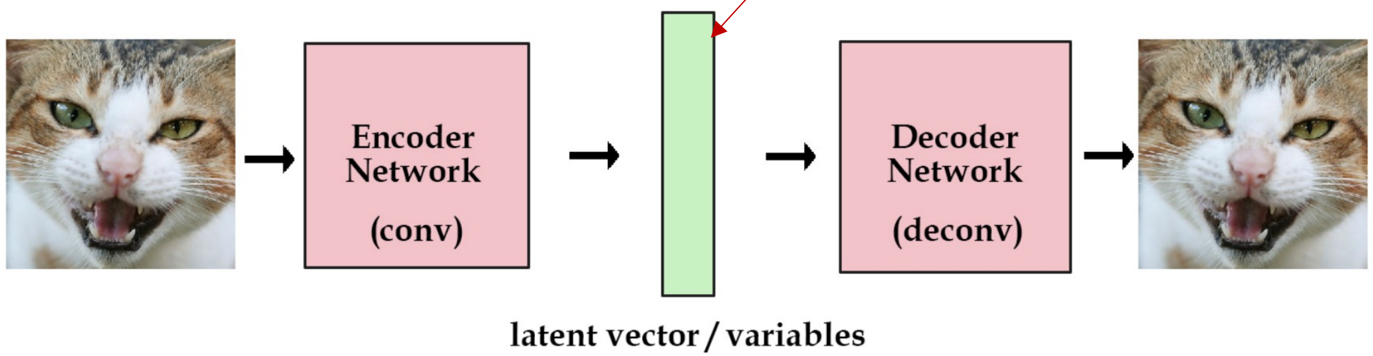
Code review: See the following:

[Jupyter Notebook: A simple Autoencoder in Tensorflow/Keras]

https://deepimaging.github.io/lectures/

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

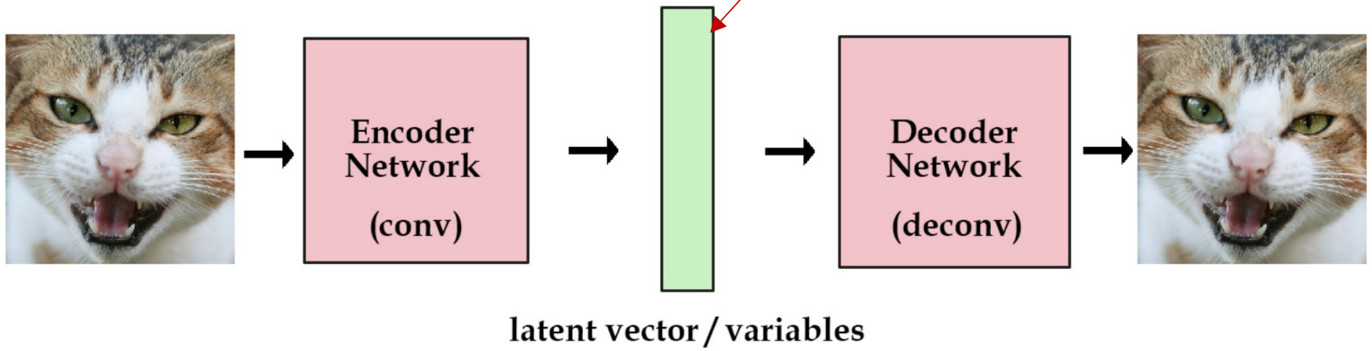


Minimize (KL) distance between latent vector and Gaussian normal

- Good *generative model*
- Have a clean probability distribution to select from to generate new examples

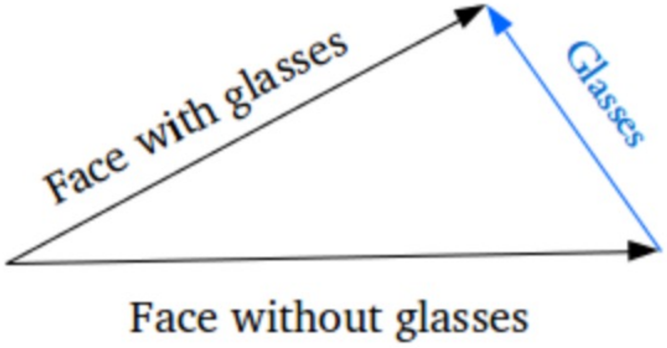
Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF



Minimize (KL) distance between latent vector and Gaussian normal

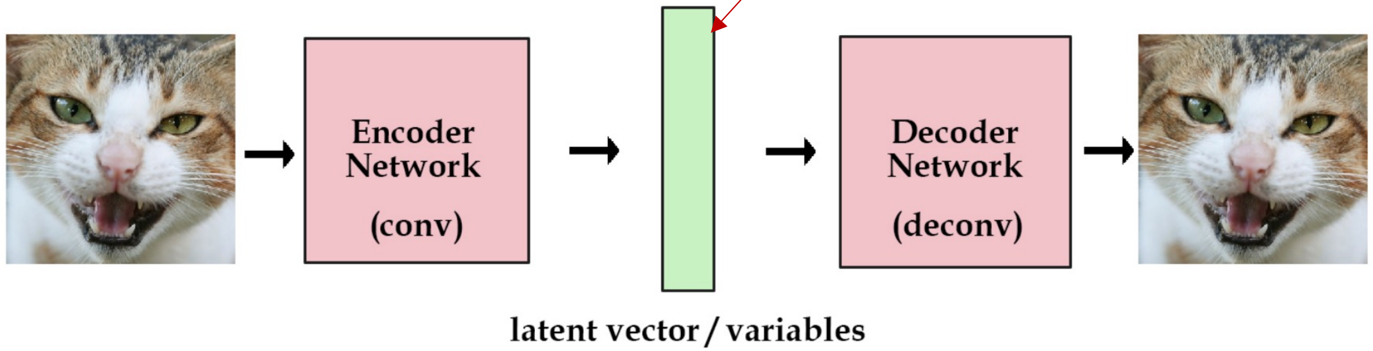
- With Gaussian PDF, can start to add/subtract latent vector in a normalized vector space



Adding new features to samples

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

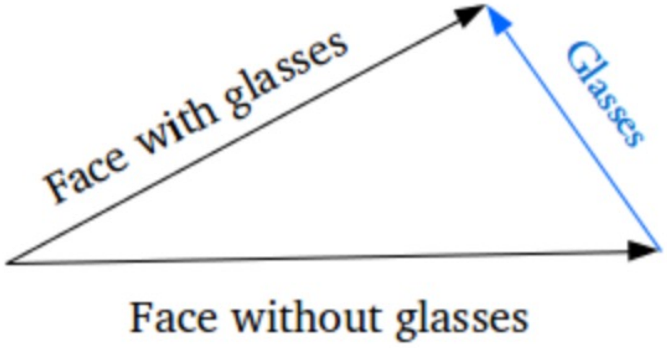


Minimize (KL) distance between latent vector and Gaussian normal

Generative Example (once trained):

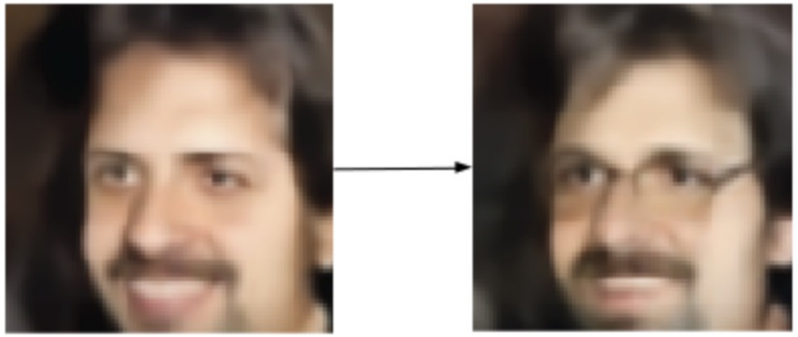
- Encode image with glasses, obtain latent vector PDF $\mathbf{P_g}$
- Encode image without glasses, obtain PDF $\mathbf{P_{ng}}$
- Compute **diff** = $\mathbf{P_g - P_{ng}}$
- Encode new image to obtain $\mathbf{P_{new}}$, add in **diff**
- Decode $\mathbf{P_{new}}$ + **diff** to get guy with glasses!

- With Gaussian PDF, can start to add/subtract latent vector in a normalized vector space



Adding new features to samples

Glasses



Exploring a specific variation of input data[1]

deep imaging