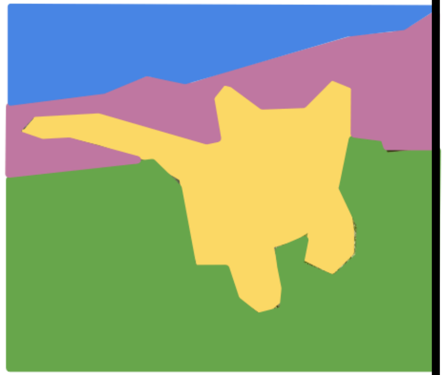
Lecture 16: Beyond classification – segmentation and autoencoders

Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

Other Computer Vision Tasks

Semantic Segmentation

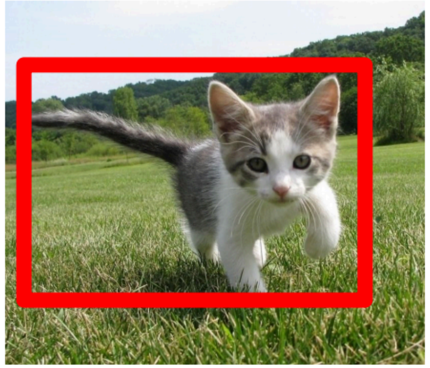


GRASS, CAT, TREE, SKY

No objects, just pixels

Balanced equation

Classification + Localization

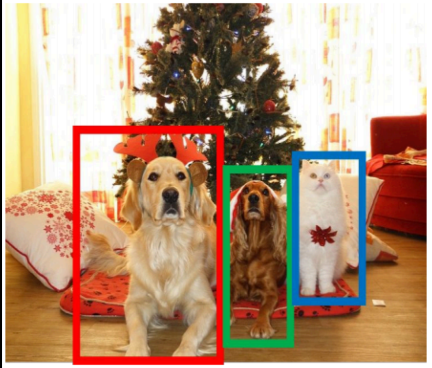


CAT

Single Object

Over-determined

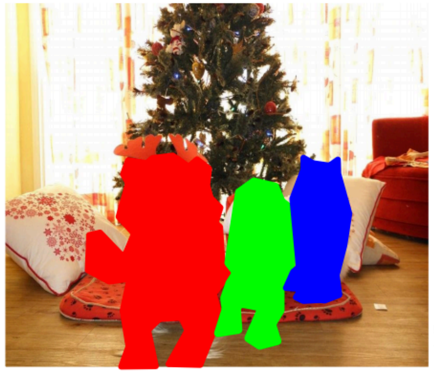
Object Detection



DOG, DOG, CAT

Over-determined

Instance Segmentation



DOG, DOG, CAT

Over-determined

Multiple Object

Super-resolution

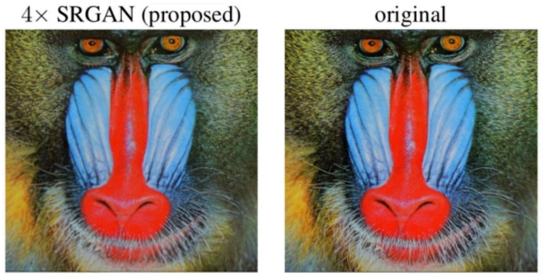


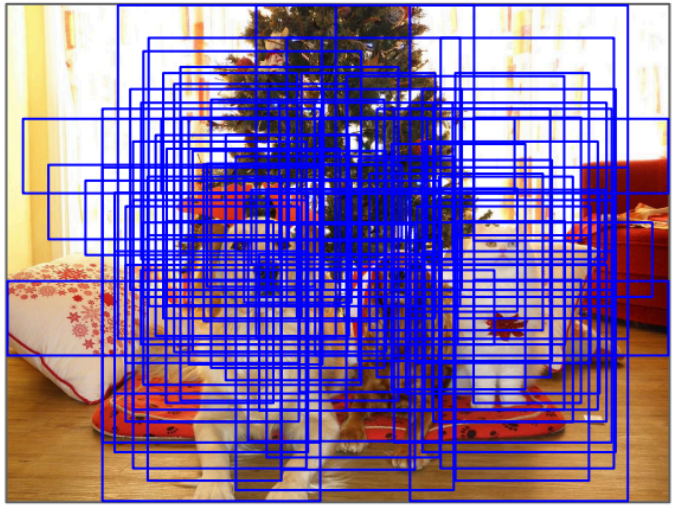
Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

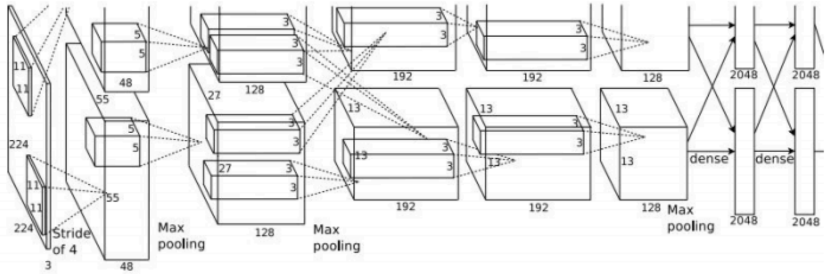
Under-determined

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

Object Detection as 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

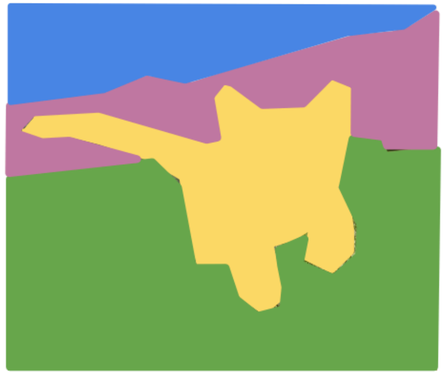
Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

More information provided in ~20 minute recording posted to:
https://deepimaging.github.io/lectures/

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

Other Computer Vision Tasks

Semantic Segmentation

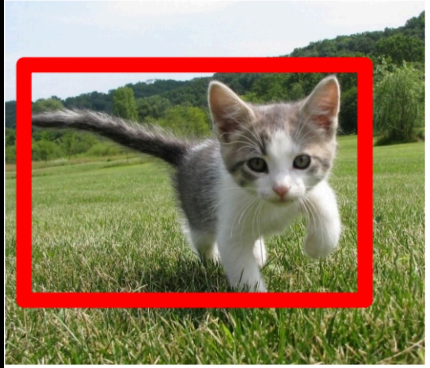


GRASS, CAT, TREE, SKY

No objects, just pixels

Balanced equation

Classification + Localization

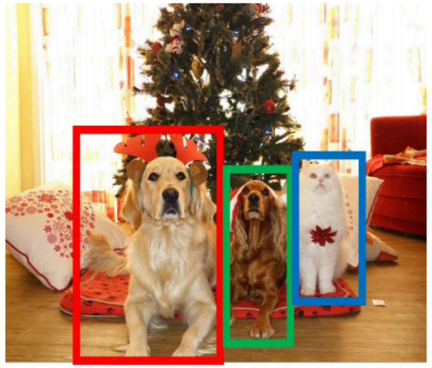


CAT

Single Object

Over-determined

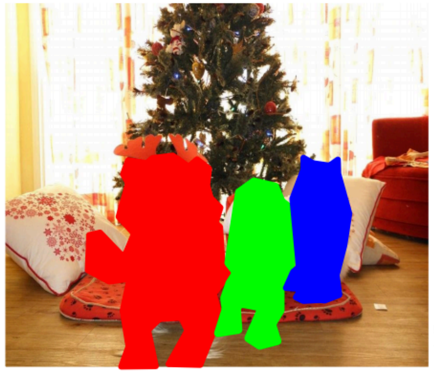
Object Detection



DOG, DOG, CAT

Over-determined

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Over-determined

Super-resolution

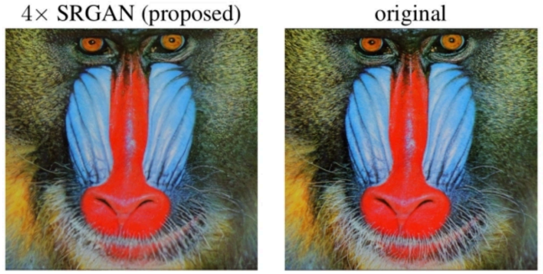


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Under-determined

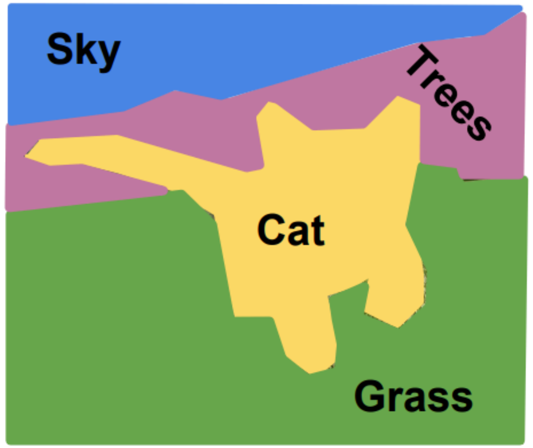
Stanford CS231n - http://cs231n.stanford.edu

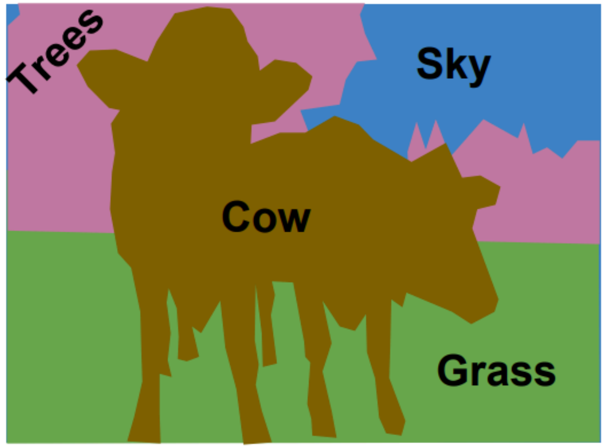
Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels

This image is CC0 public domain



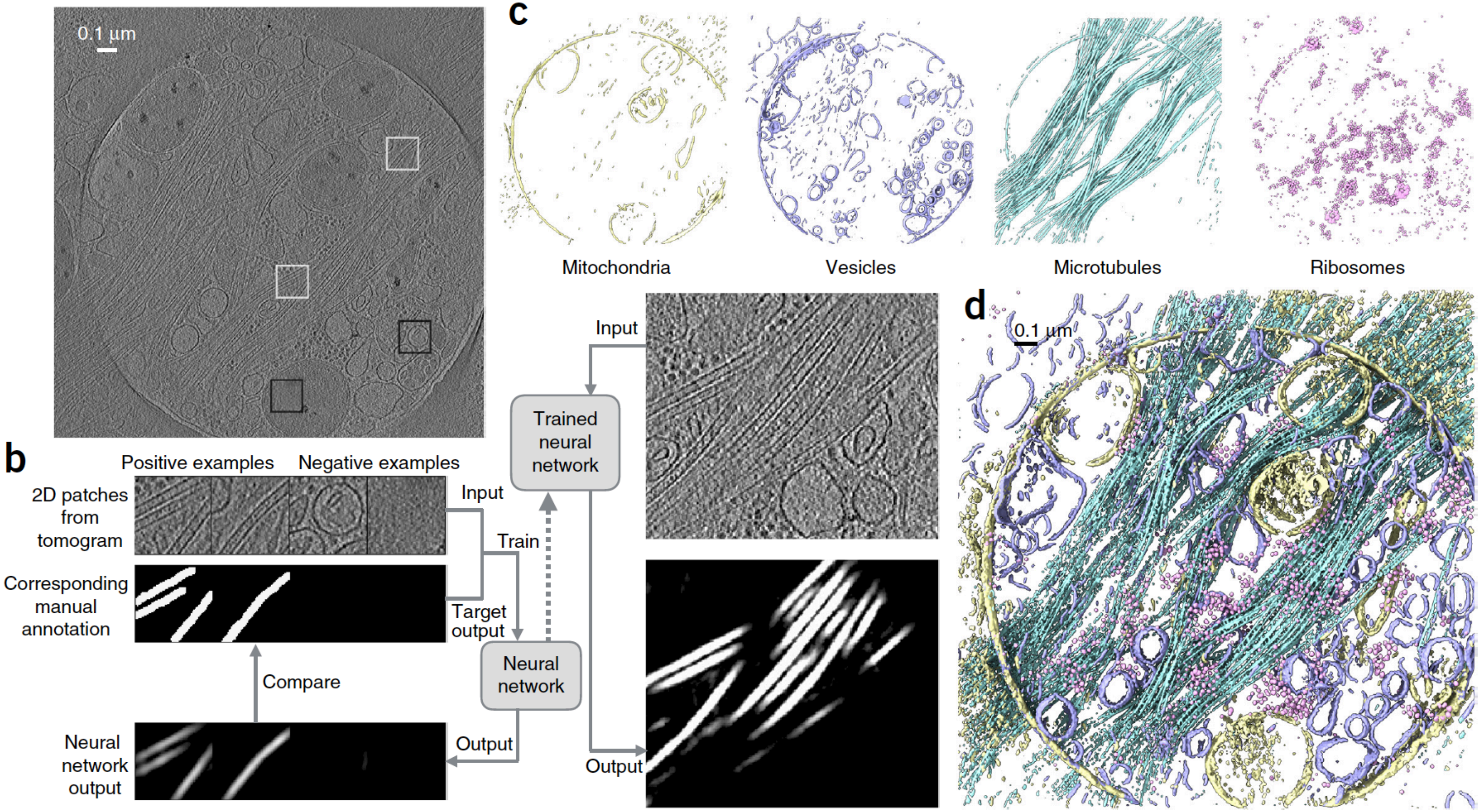


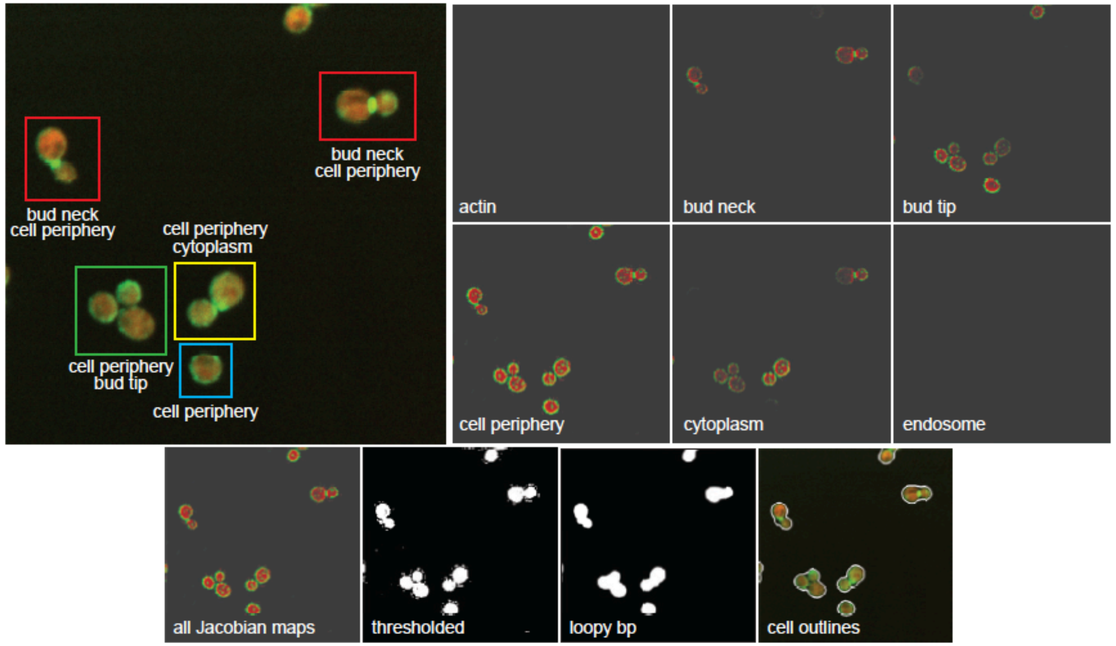
Stanford CS231n - http://cs231n.stanford.edu

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]

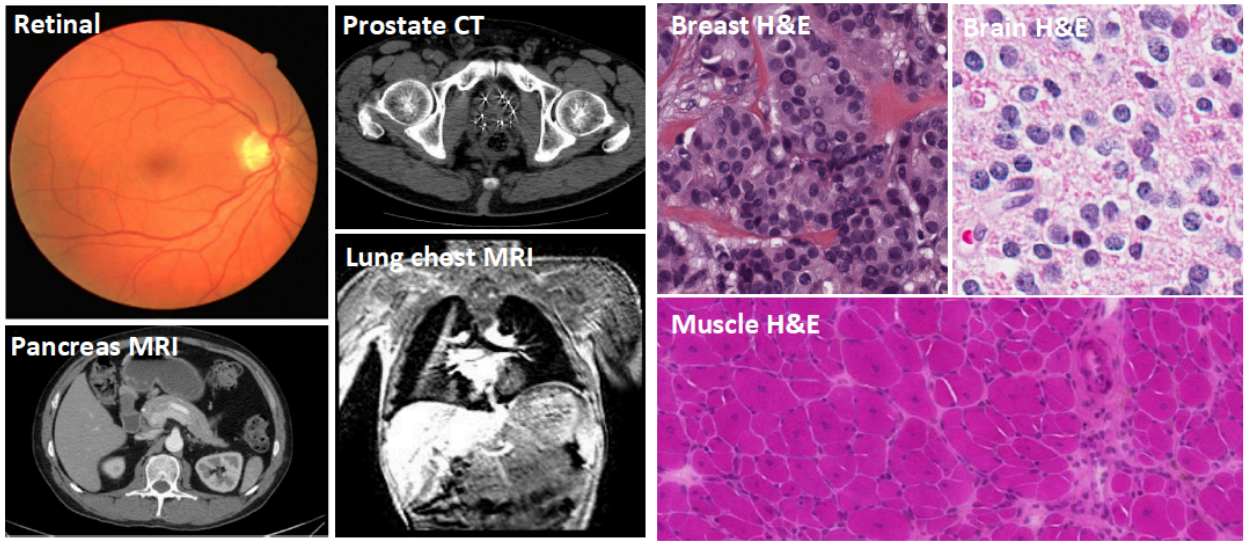
deep imaging

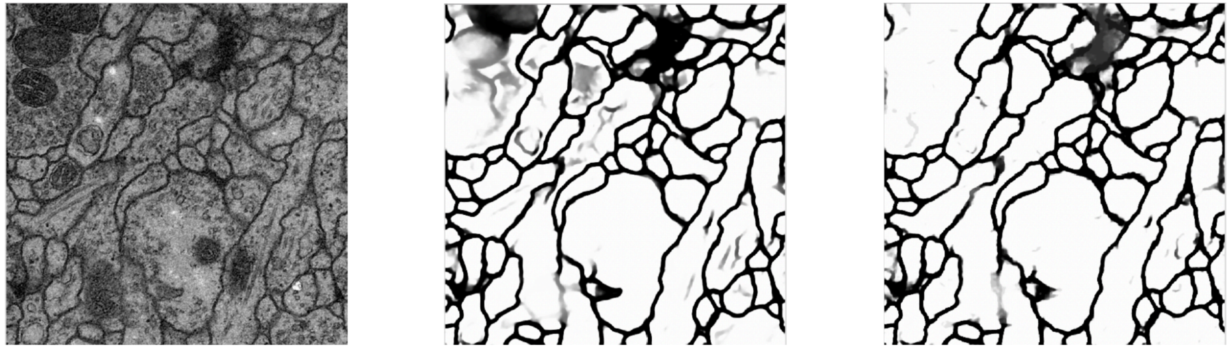




Oren Z. Kraus et al., "Classifying and Segmenting Microscopy Images Using Convolutional Multiple Instance Learning," arXiv 2015

Other possible examples:

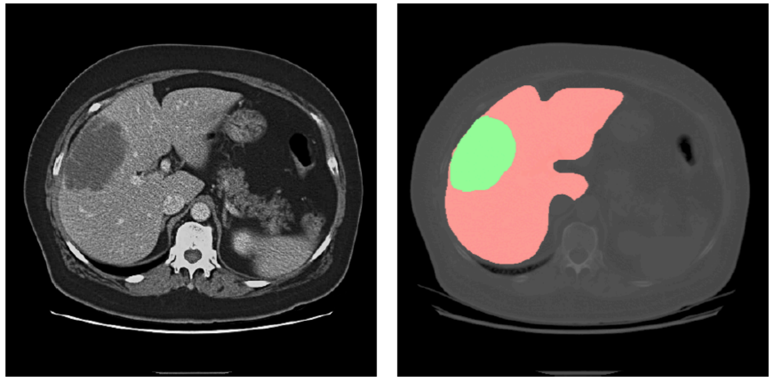




(a) Input image

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

(c) Segmentation result of our pipeline

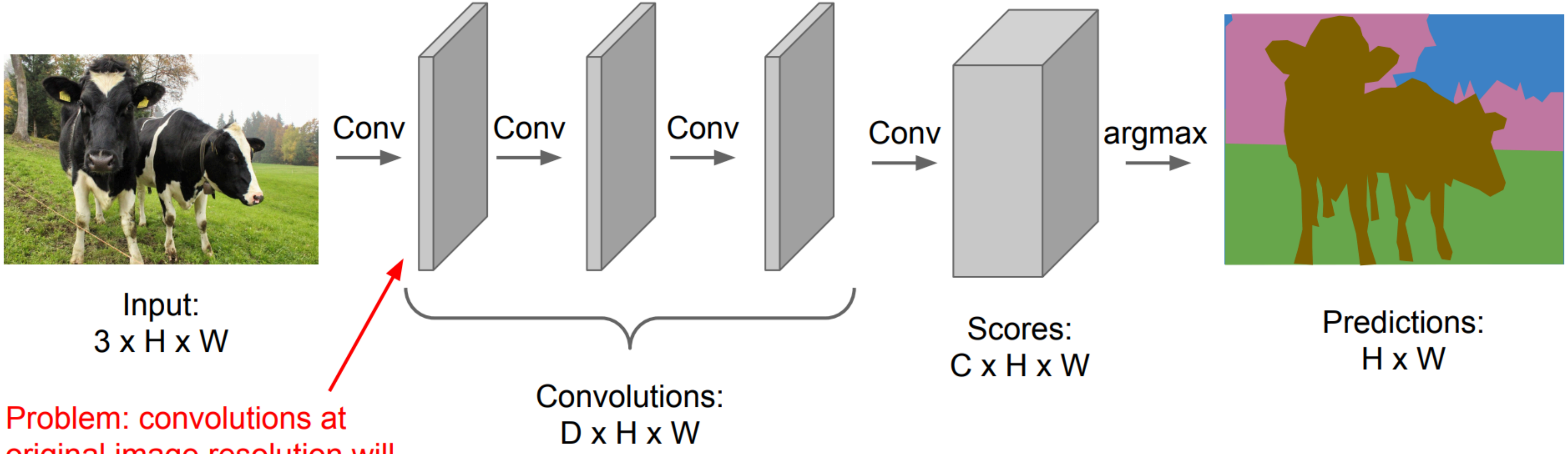


M. Drozdzal et al., Learning Normalized Inputs for Iterative Estimation in Medical Image Segmentation (2017)

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!



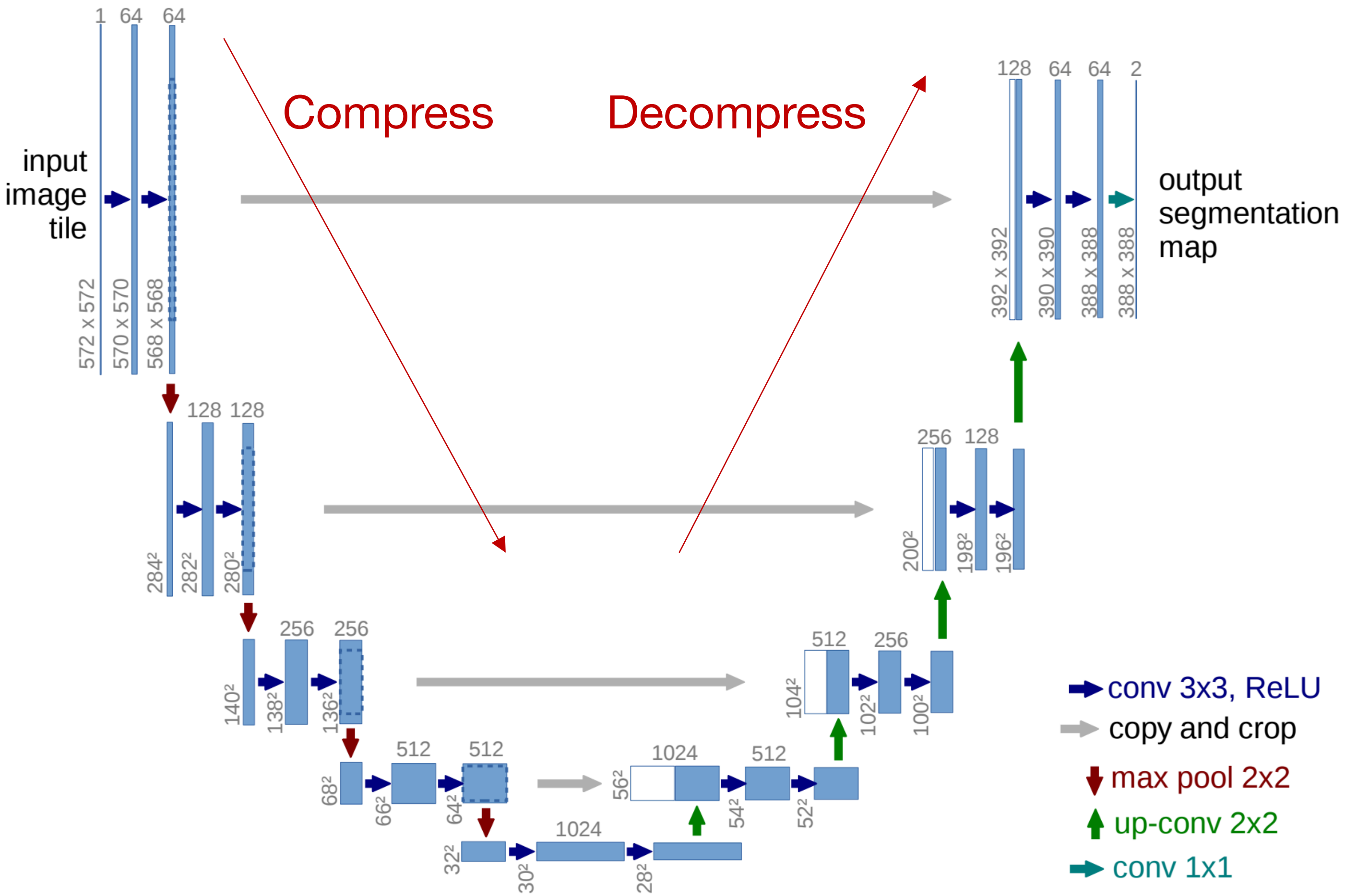
Problem: convolutions at original image resolution will be very expensive ...

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/

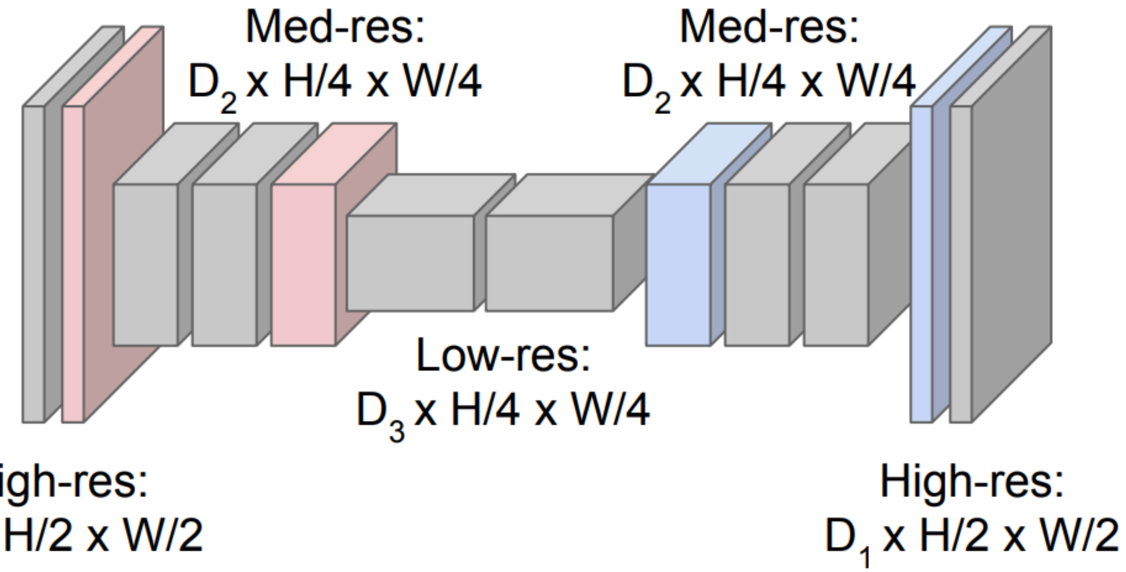
Semantic Segmentation Idea: Fully Convolutional

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



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

deep imaging

In-Network upsampling: “Unpooling”

Nearest Neighbor

Input: 2 x 2

1	2
3	4

Output: 4 x 4

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

“Bed of Nails”

Input: 2 x 2

1	2
3	4

Output: 4 x 4

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

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

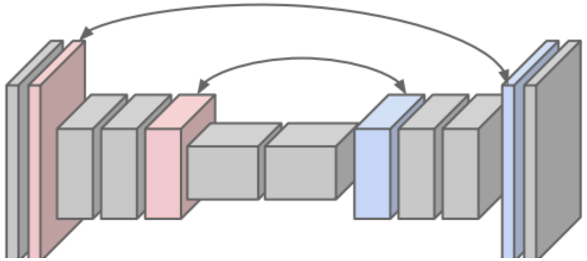
1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

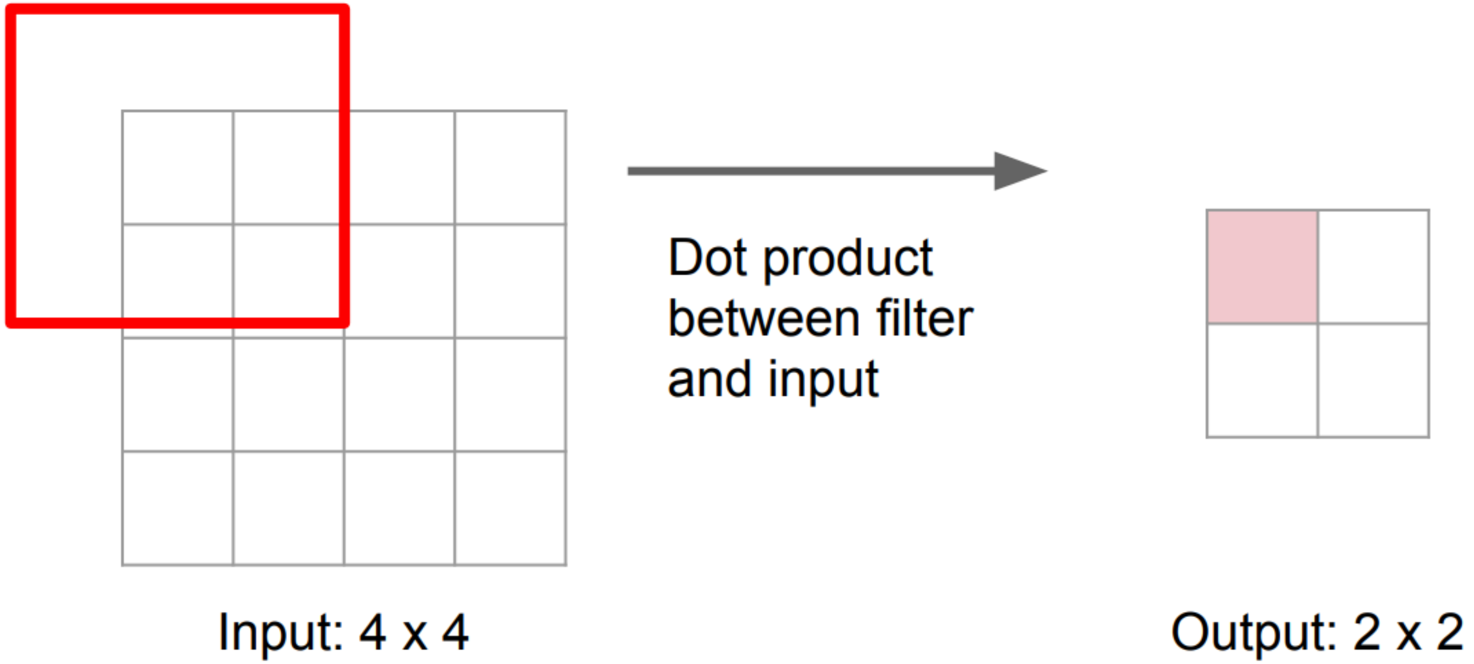
Corresponding pairs of downsampling and upsampling layers



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

Learnable Upsampling: Transpose Convolution

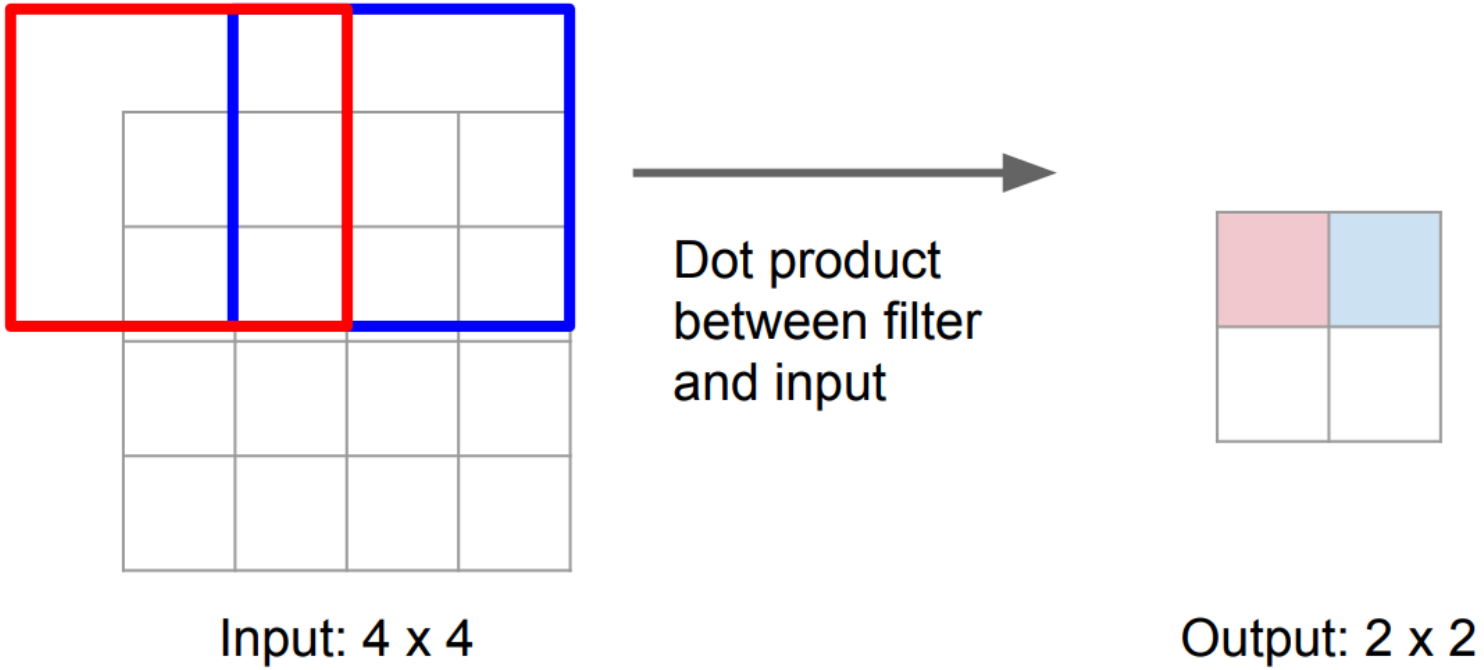
Recall: Normal 3 x 3 convolution, stride 2 pad 1



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

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, <u>stride 2</u> pad 1



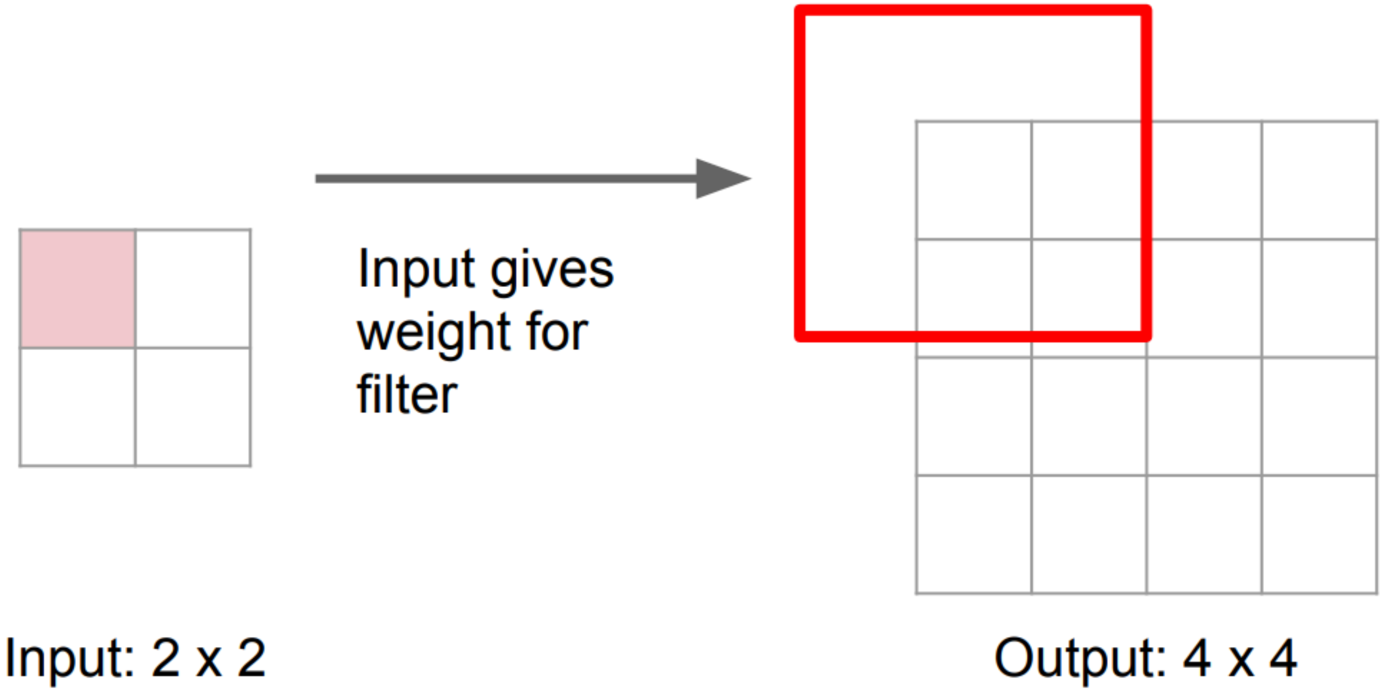
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

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

Learnable Upsampling: Transpose Convolution

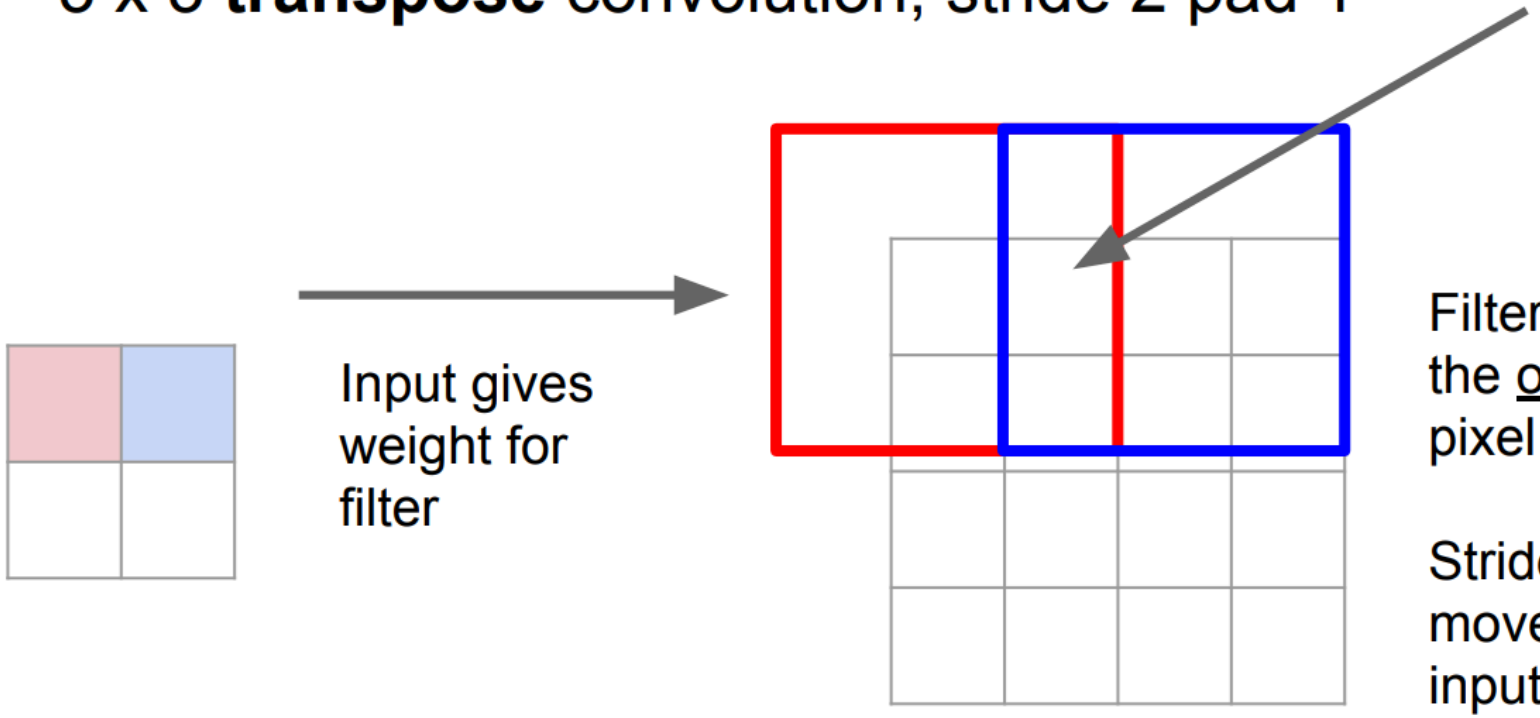
3 x 3 **transpose** convolution, stride 2 pad 1



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

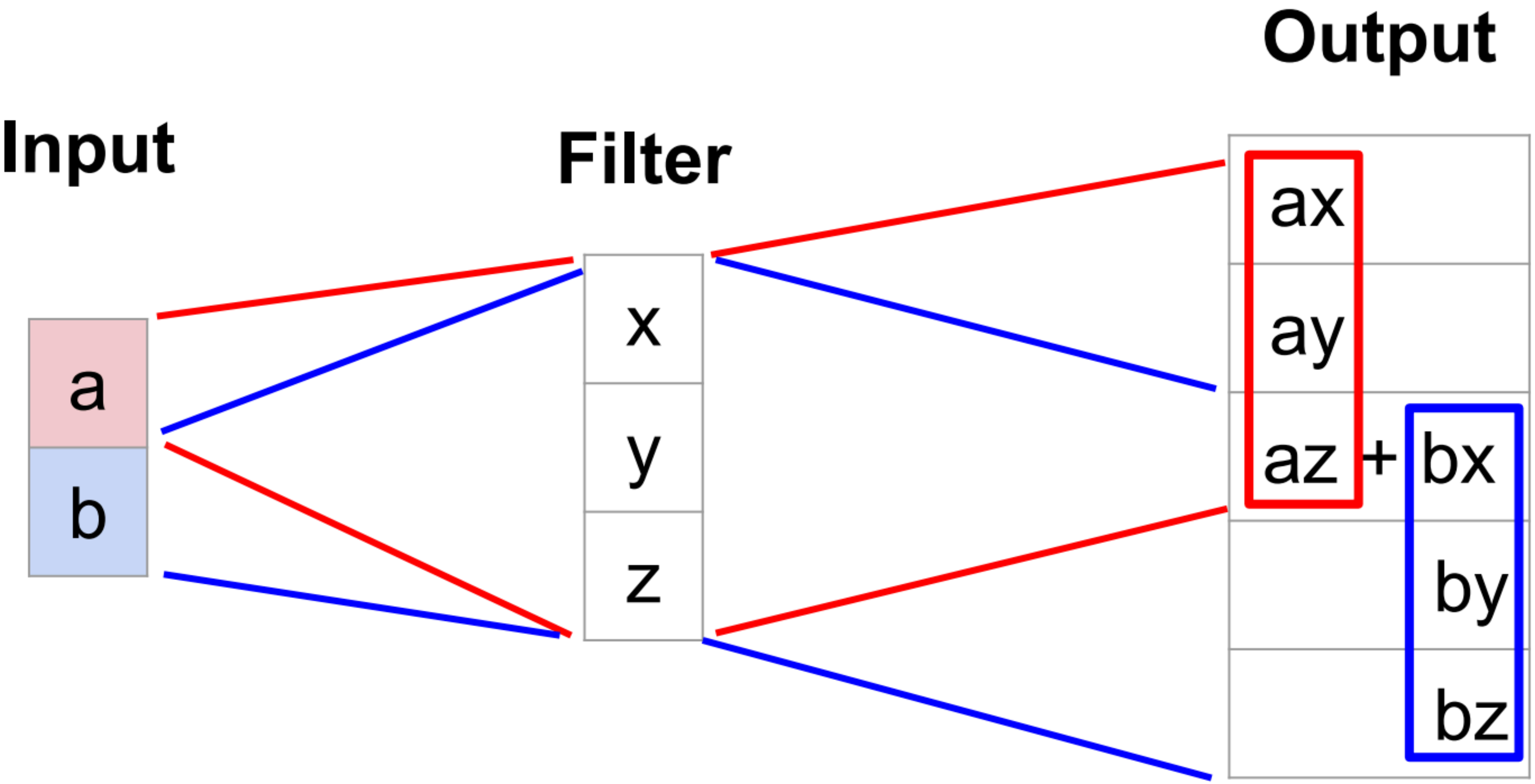
Stride gives ratio between movement in output and input

Input: 2 x 2

Output: 4 x 4

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, stride=2, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

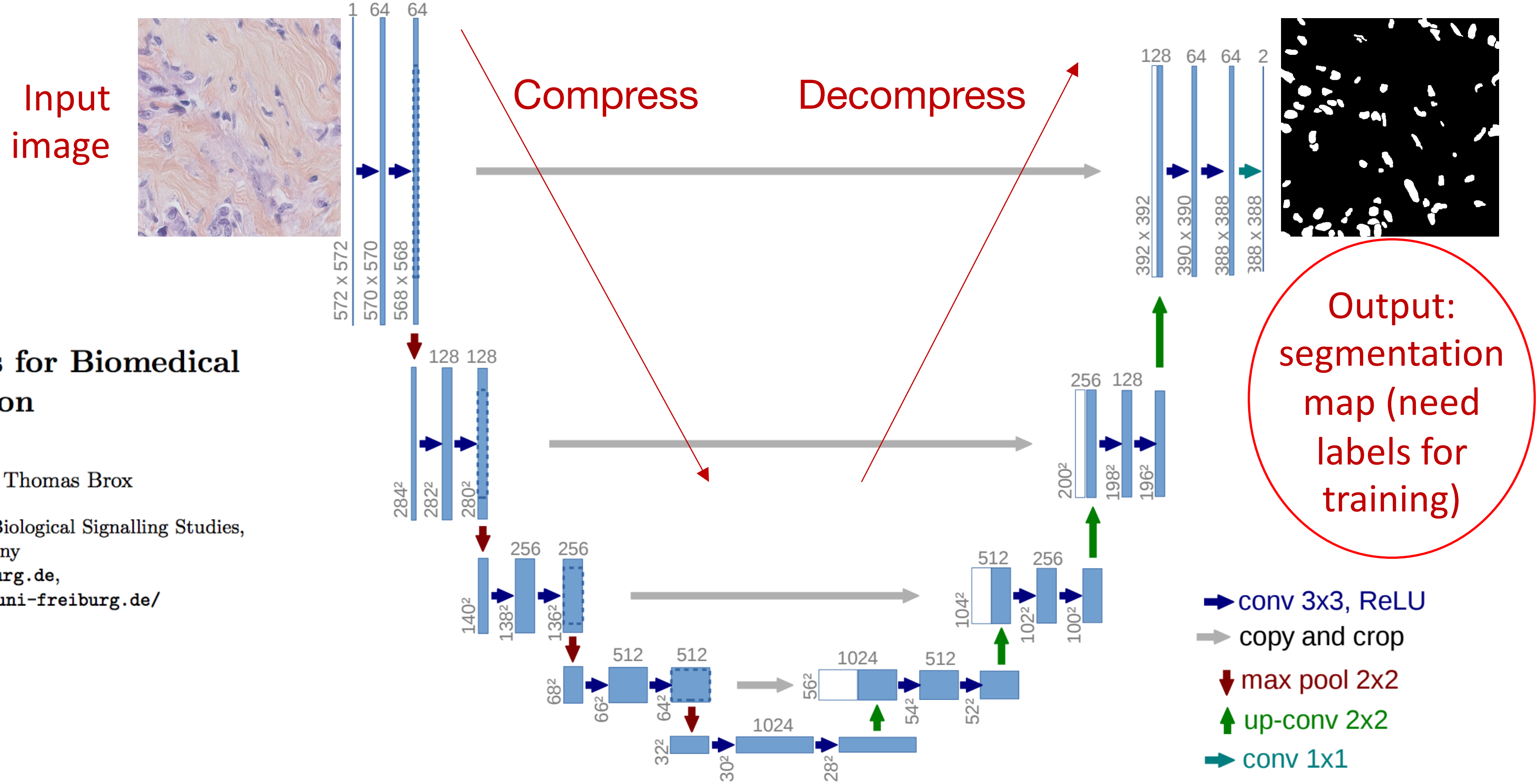
$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

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

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

Segmentation: need a map of classes for label

U-Net Architecture



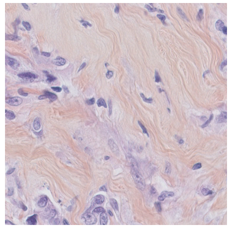
U-Net: Convolutional Networks for Biomedical Image Segmentation

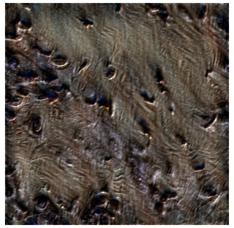
Olaf Ronneberger, Philipp Fischer, and Thomas Brox

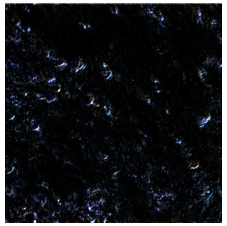
Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Learned sensing for improved image segmentation

Variably illuminated images from different LEDs



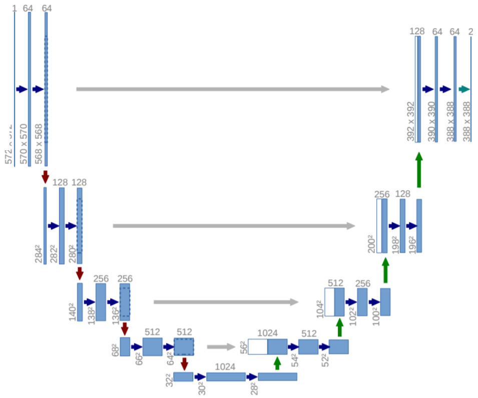




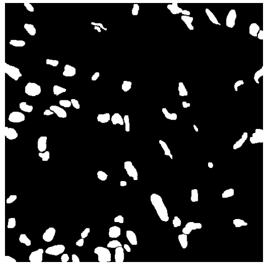
…

$w_1 I_1 \quad + \quad w_2 I_2 \quad + \quad w_3 I_3 \quad + \; \ldots$

U-Net CNN



Segmentation Map

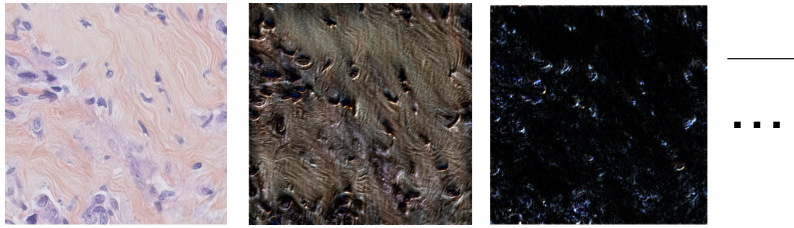


*If we allow w's here to be trainable weights, then we can find ideal brightnesses for the different LEDs!

Learned sensing for improved image segmentation

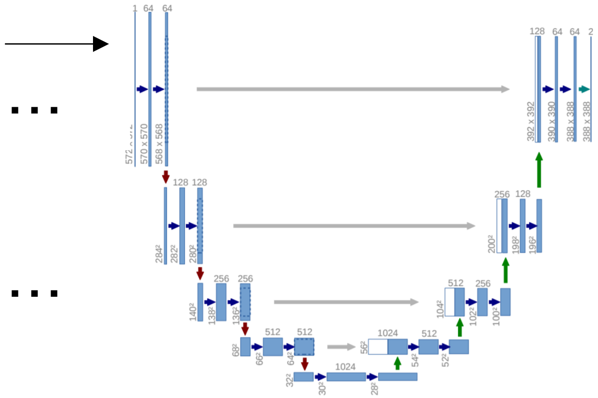
deep imaging

Variably illuminated images

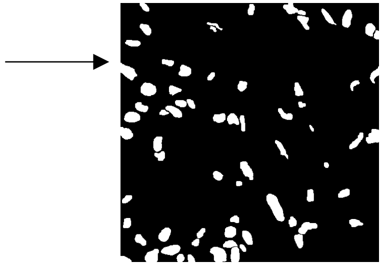


$w_1 I_1$ + $w_2 I_2$ + $w_3 I_3$ + …

U-Net CNN

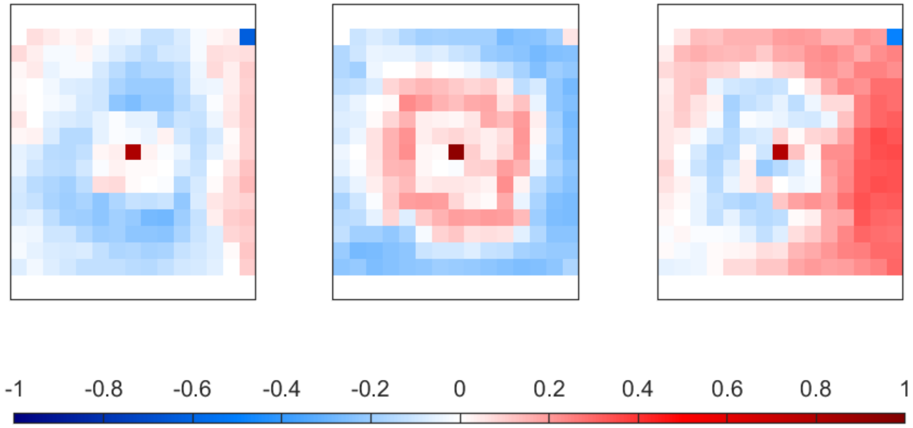


Segmentation Map

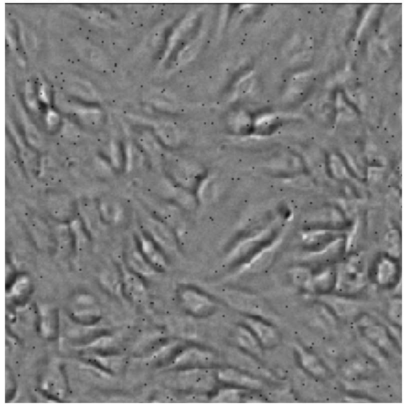


Optimized illumination for nuclei segmentation

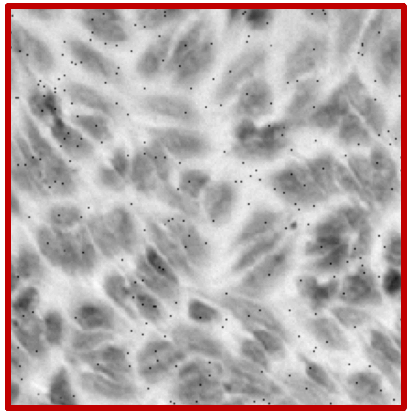
Mean Red channel   Mean Green channel   Mean Blue channel



Standard illumination



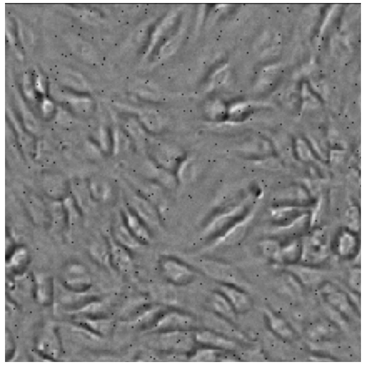
Learned illumination



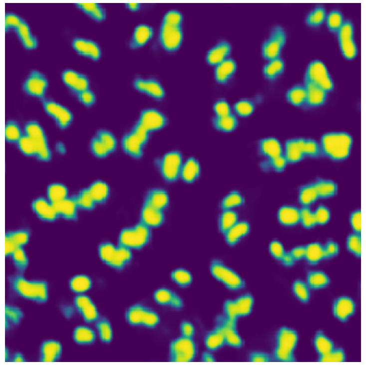
+5-10% accuracy

Image segmentation –current workflow

Capture: BF images



Capture: Fluorescence



(e.g., DAPI-stained nuclei)

Threshold

Segmentation Mask

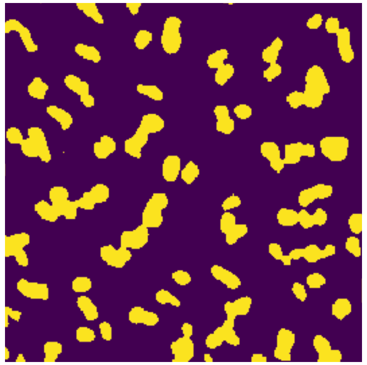
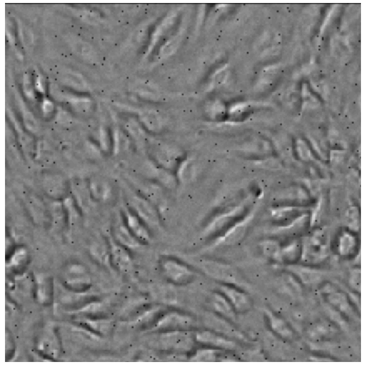
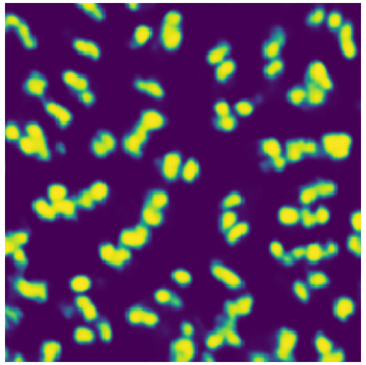


Image segmentation –current workflow

Capture: BF images



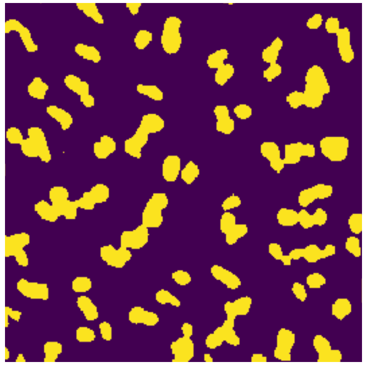
Capture: Fluorescence



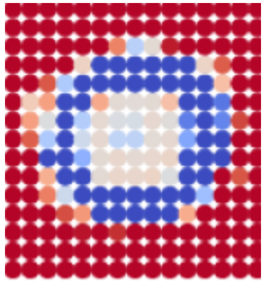
(e.g., DAPI-stained nuclei)

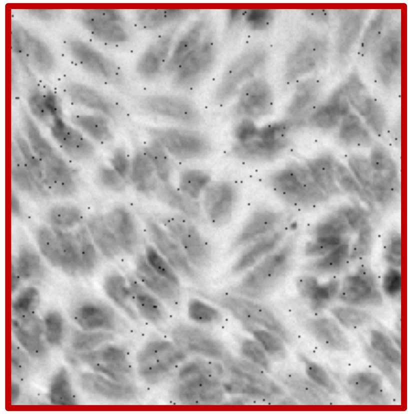
Threshold

Segmentation Mask



Learned illumination





Optimally illuminated

Inference via a trained U-Net

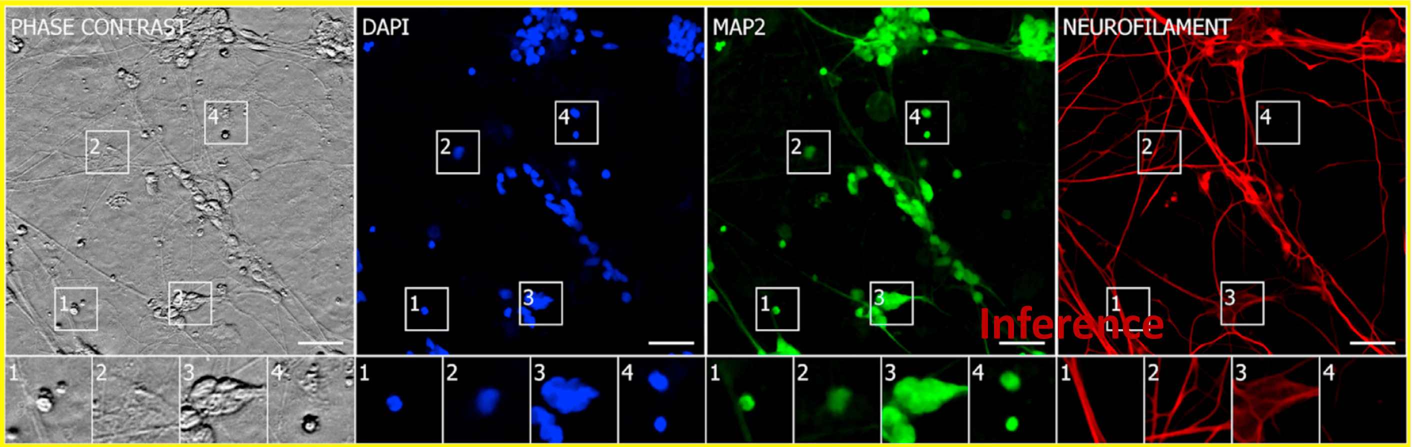
in silico labeling: fluorescence image inference from bright-field data

deep imaging

In Silico Labeling: Predicting Fluorescent Labels in Unlabeled Images

Eric M. Christiansen,[1,11,*] Samuel J. Yang,[1] D. Michael Ando,[1,9] Ashkan Javaherian,[2,9] Gaia Skibinski,[2,9] Scott Lipnick,[3,4,8,9] Elliot Mount,[2,10] Alison O'Neil,[3,10] Kevan Shah,[2,10] Alicia K. Lee,[2,10] Piyush Goyal,[2,10] William Fedus,[1,6,10] Ryan Poplin,[1,10] Andre Esteva,[1,7] Marc Berndl,[1] Lee L. Rubin,[3] Philip Nelson,[1,*] and Steven Finkbeiner[2,5,*]

[1]Google, Inc., Mountain View, CA 94043, USA
[2]Taube/Koret Center for Neurodegenerative Disease Research and DaedalusBio, Gladstone Institutes, San Francisco, CA 94158, USA
[3]Department of Stem Cell and Regenerative Biology, Harvard University, Cambridge, MA 02138, USA
[4]Department of Biomedical Informatics, Harvard Medical School, Boston, MA 02115, USA
[5]Departments of Neurology and Physiology, University of California, San Francisco, 94158, USA
[6]Montreal Institute of Learning Algorithms, University of Montreal, Montreal, QC, Canada
[7]Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA
[8]Center for Assessment Technology and Continuous Health, Massachusetts General Hospital, Boston, MA 02114, USA



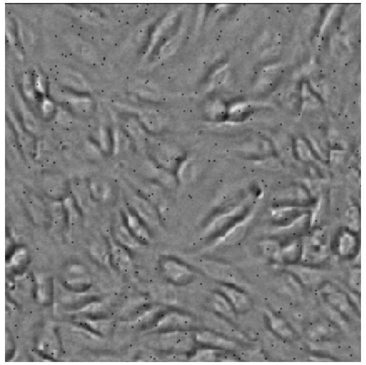
BF Focal stack (26+ images)

Task: bright-field to fluorescence image Inference

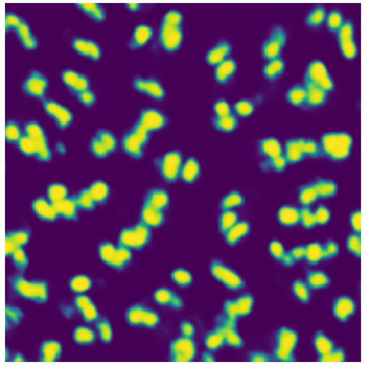
Image segmentation versus *in silico* labeling (fluorescence inference)

deep imaging

Capture: Bright field



Capture: Fluorescence



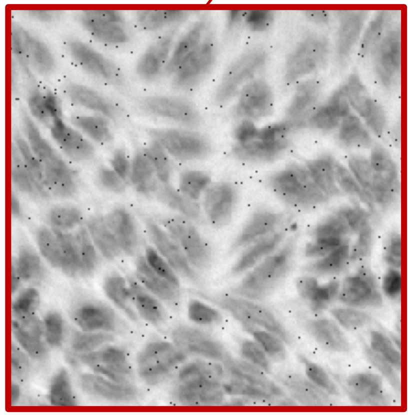
Threshold

Segmentation Mask

Directly infer the fluorescence image

Learned illumination for fluorescent image inference?

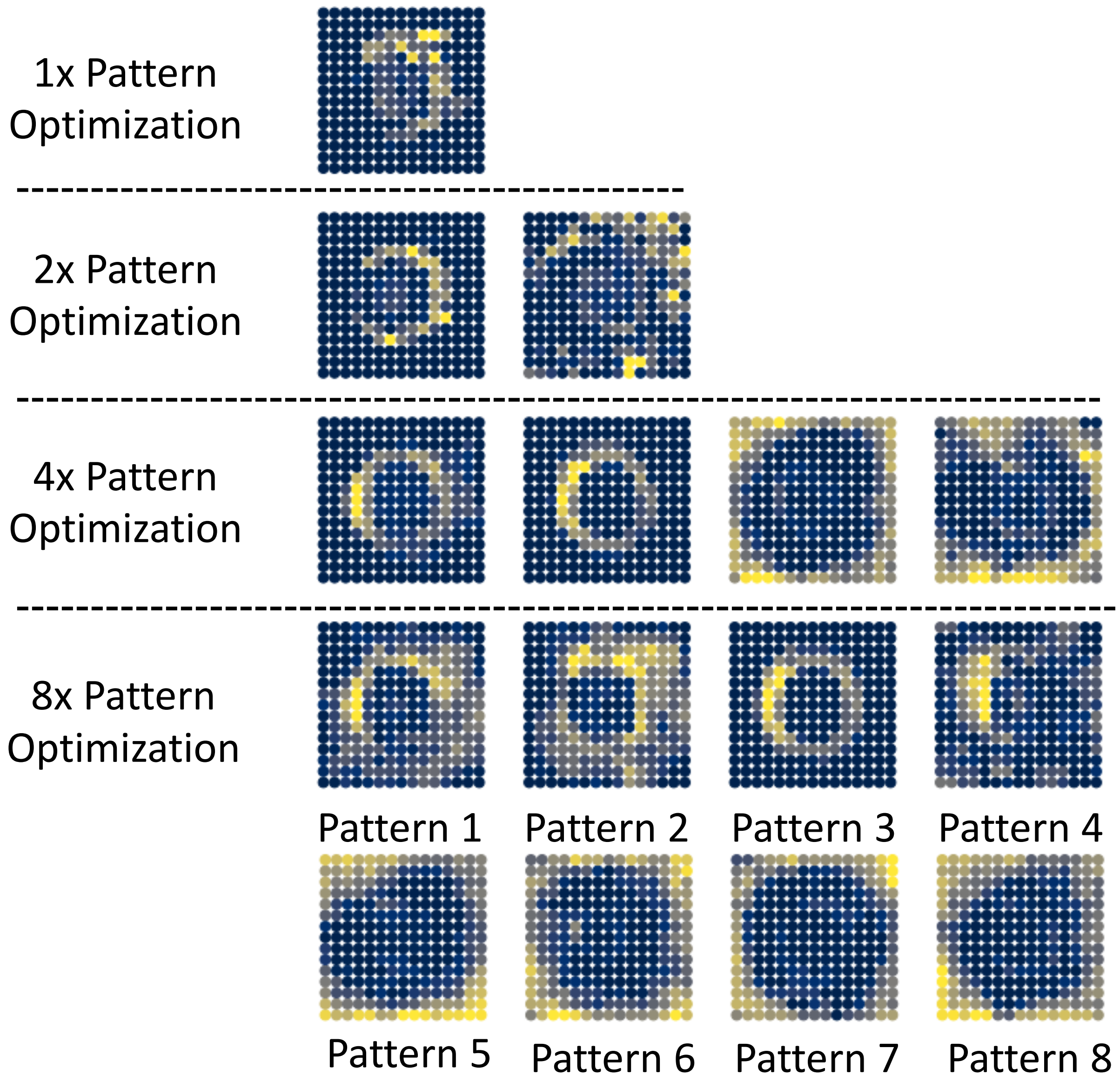
?

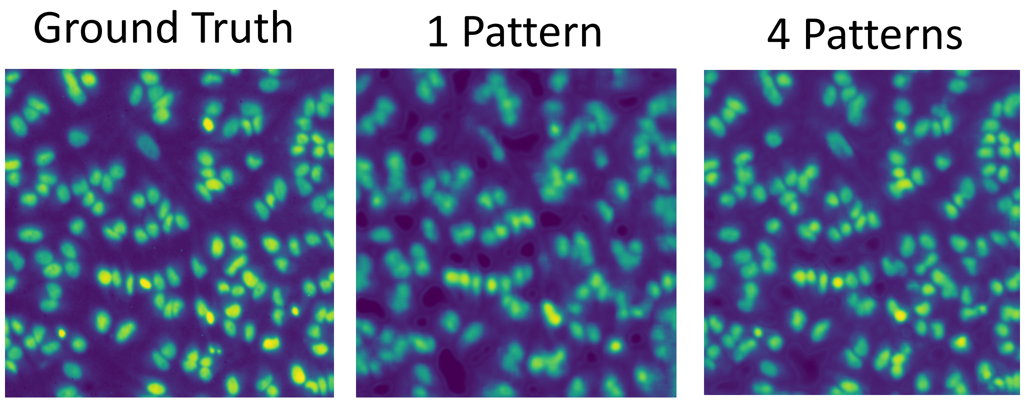


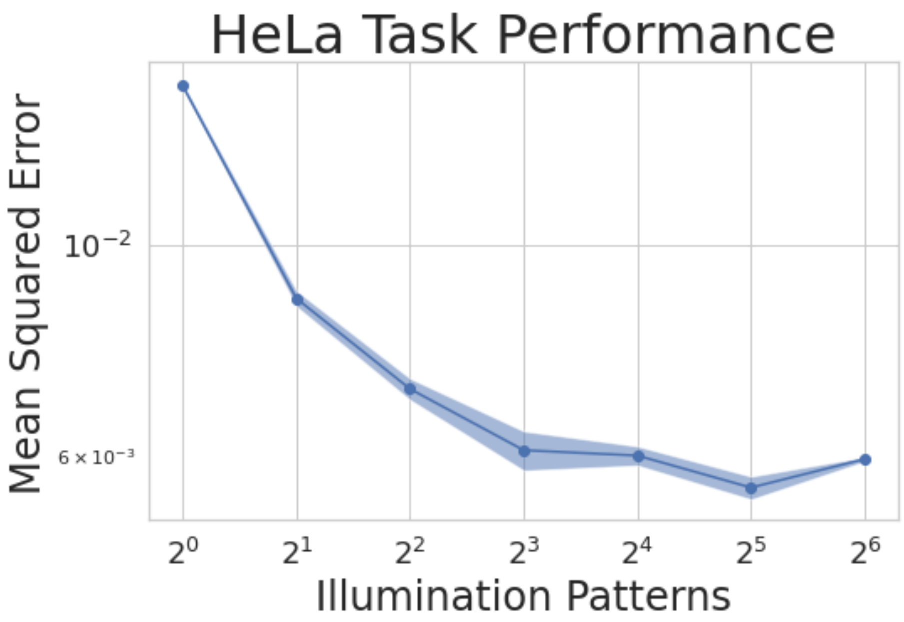
Optimally illuminated

Multiple Patterns for Fluorescence image inference

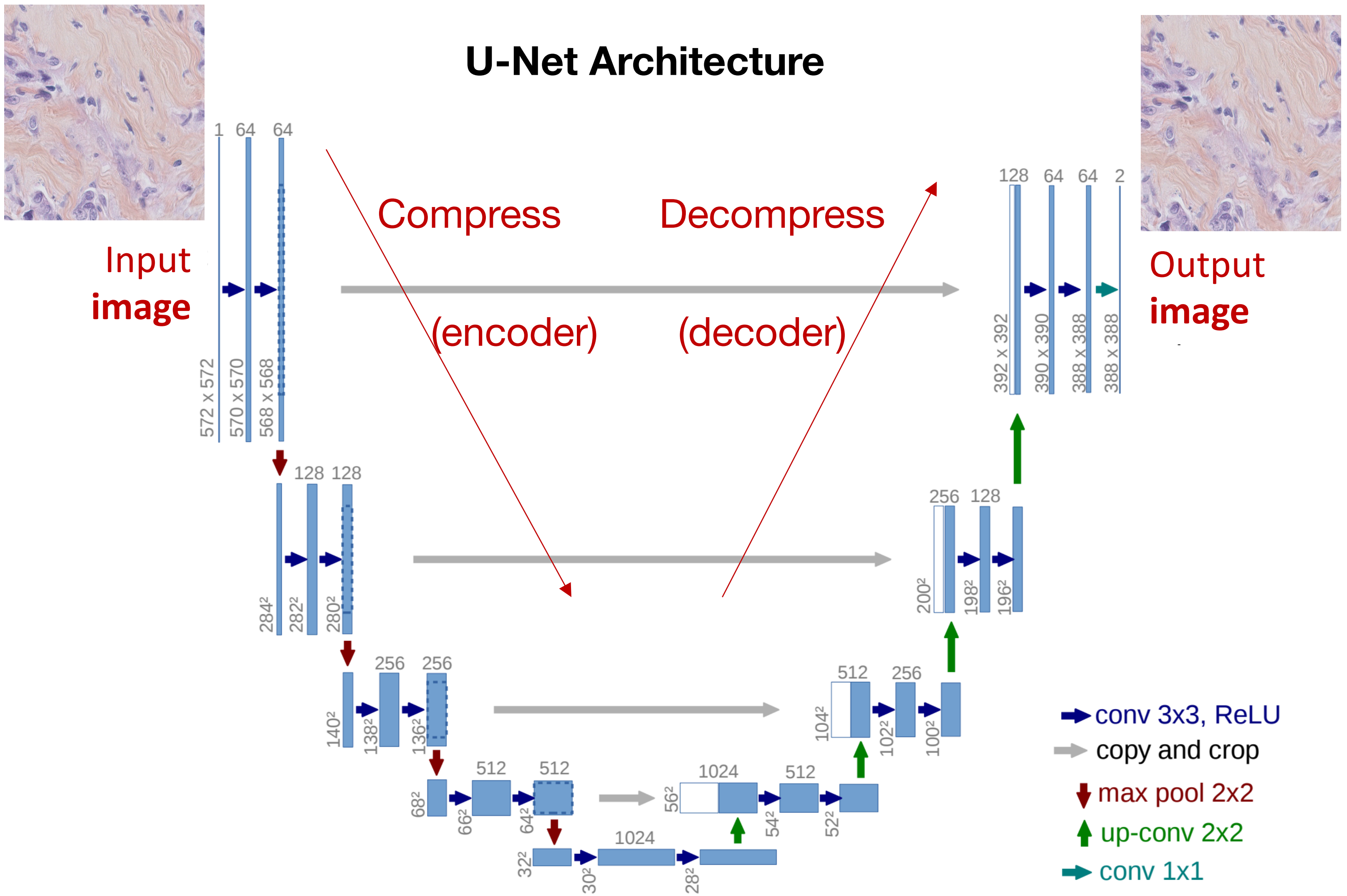
deep imaging



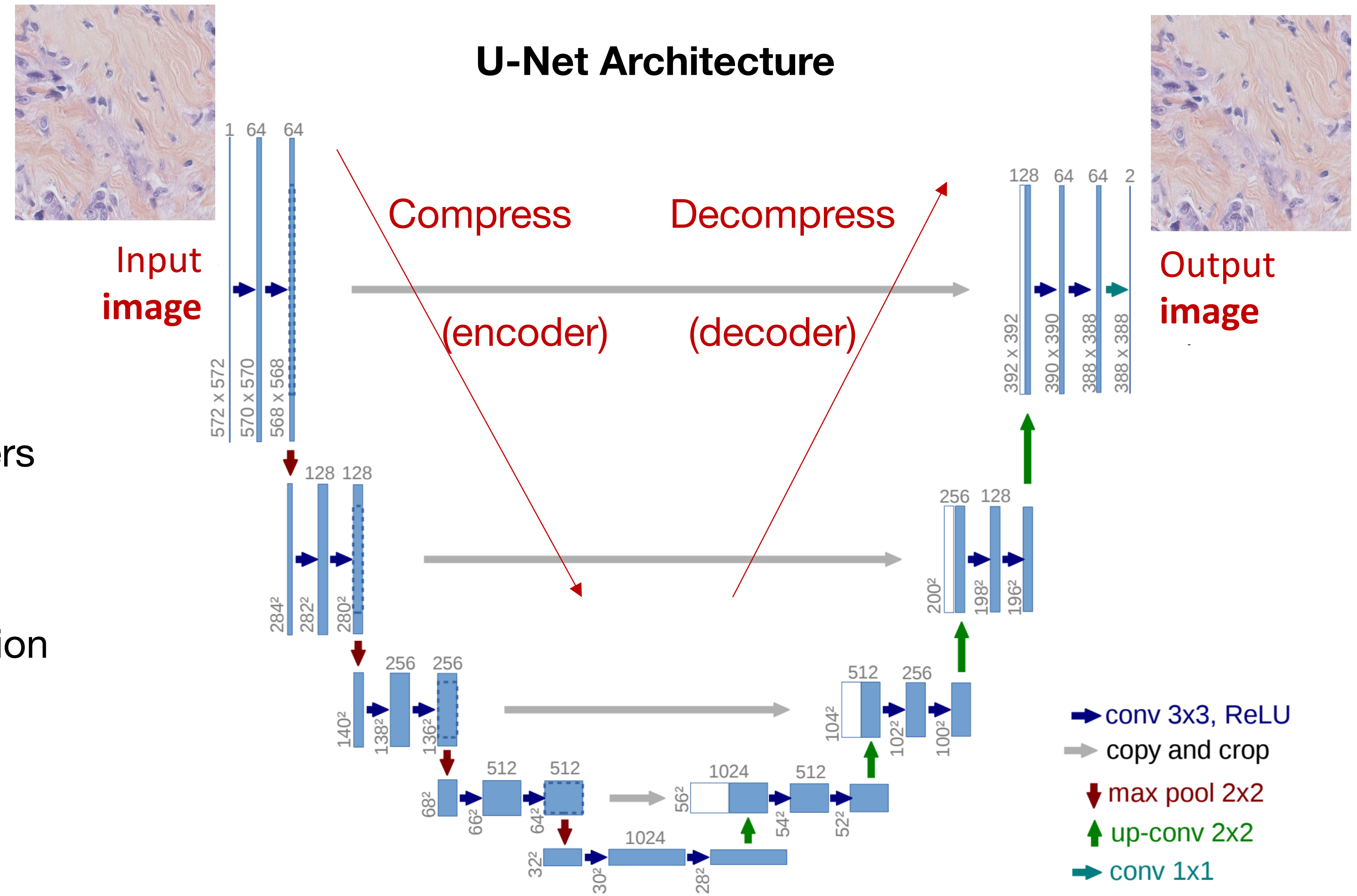




Instead, *compress x-y dimensions of* input image



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

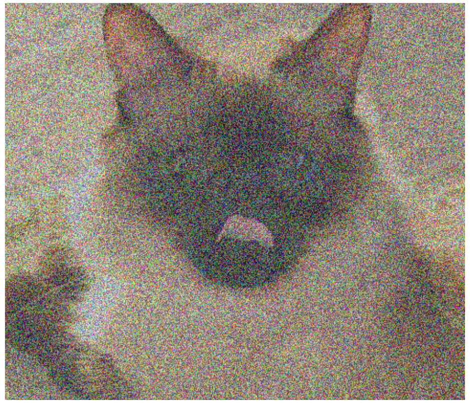


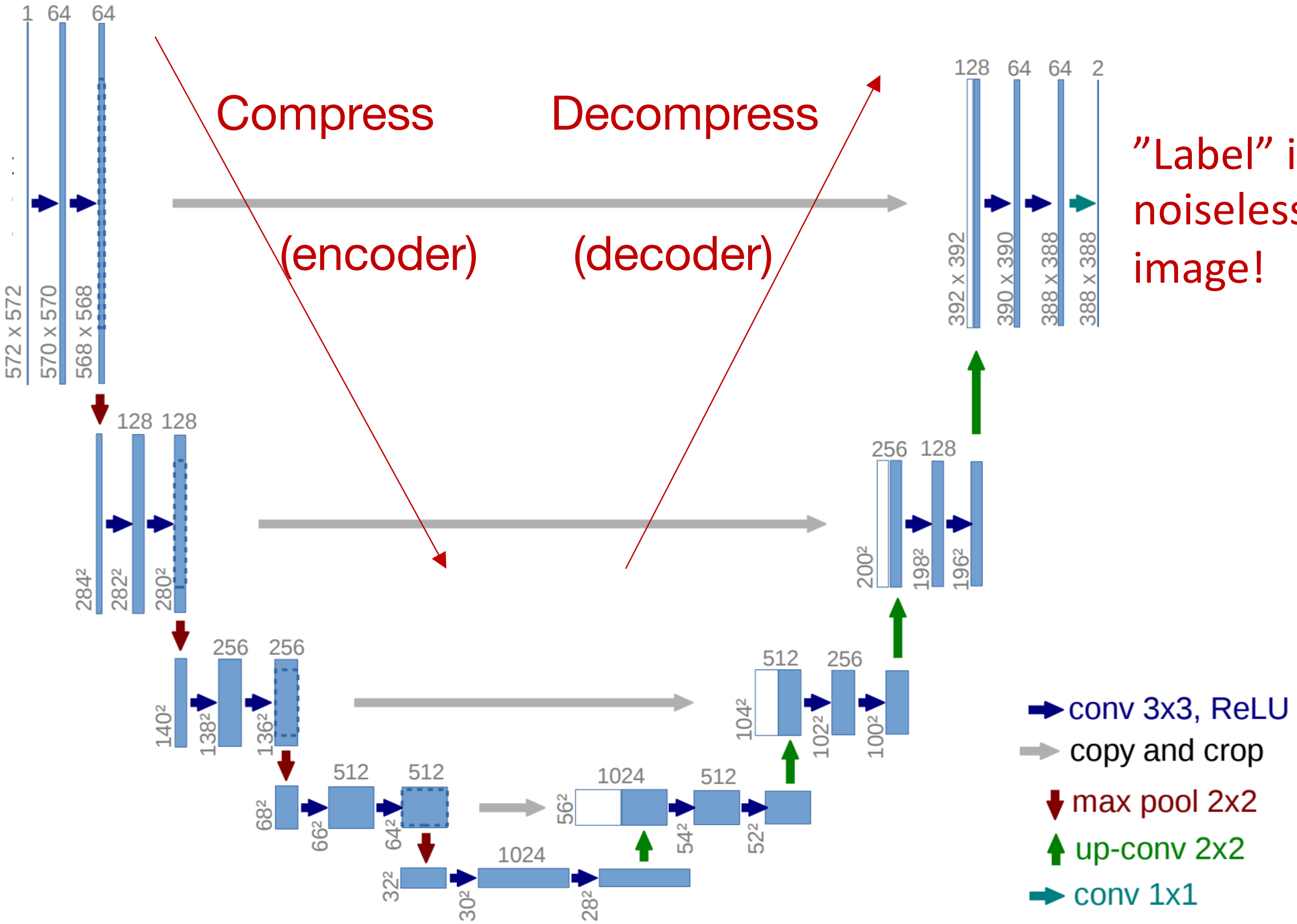
- Compress spatial features into learned filters
- Then, decompress learned filters back into same spatial dimensions
- **Termed an autoencoder**
- Analogous to image compression
- A pretty powerful idea…

Another example: Denoising Autoencoder

U-Net Architecture

Input noisy image



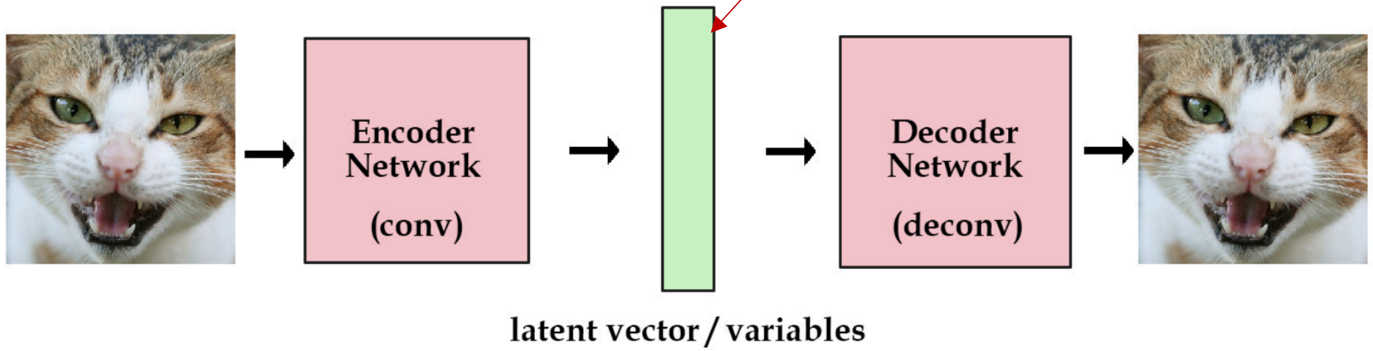


”Label” is noiseless image!

deep imaging

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF



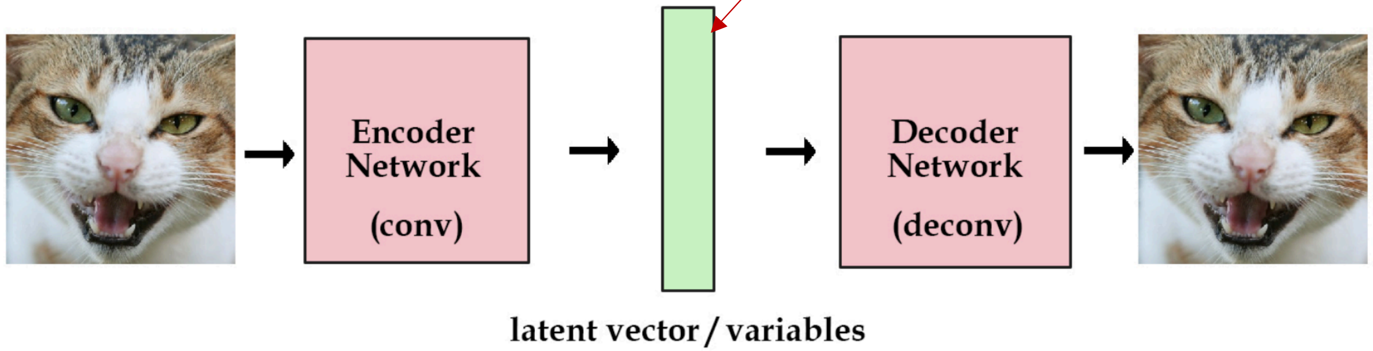
Minimize (KL) distance between latent vector and Gaussian normal

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

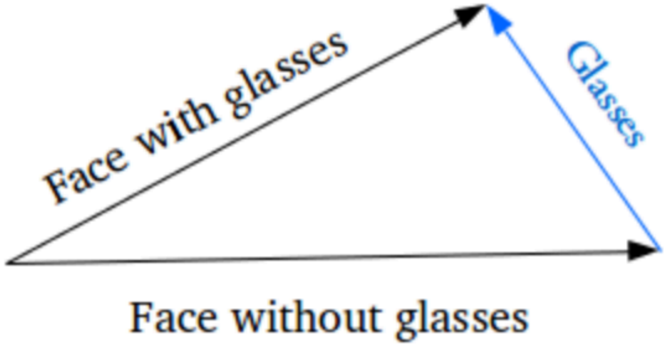
Example: Variational Autoencoder (VAE)

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

Force this vector to follow a Gaussian PDF



Minimize (KL) distance between latent vector and Gaussian normal

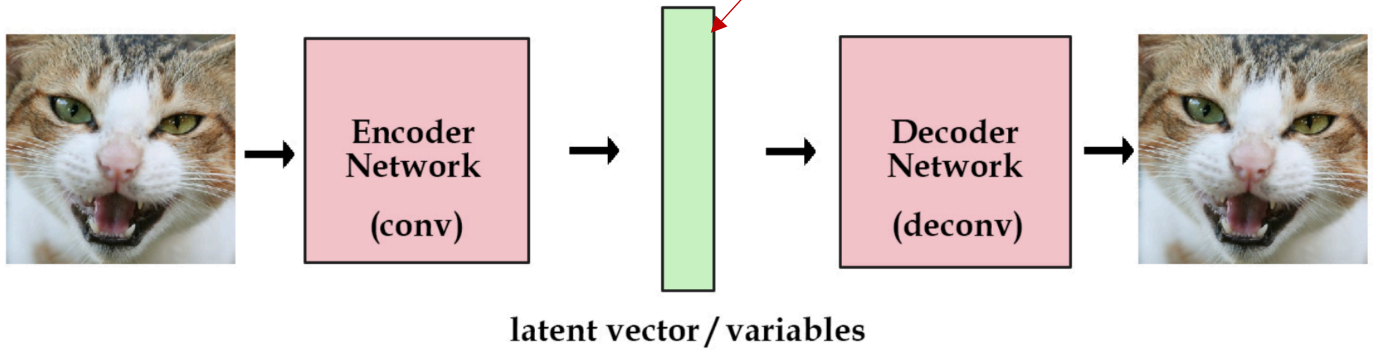


Adding new features to samples

Example: Variational Autoencoder (VAE)

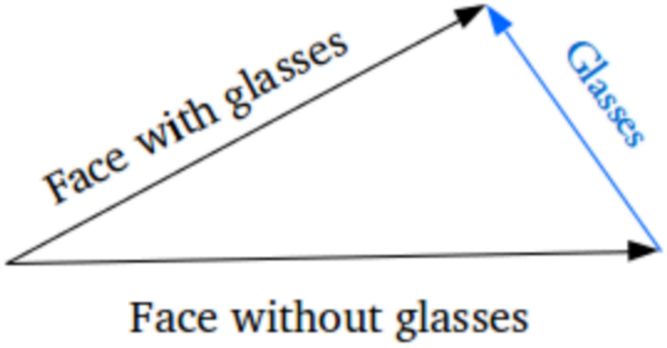
deep imaging

Force this vector to follow a Gaussian PDF

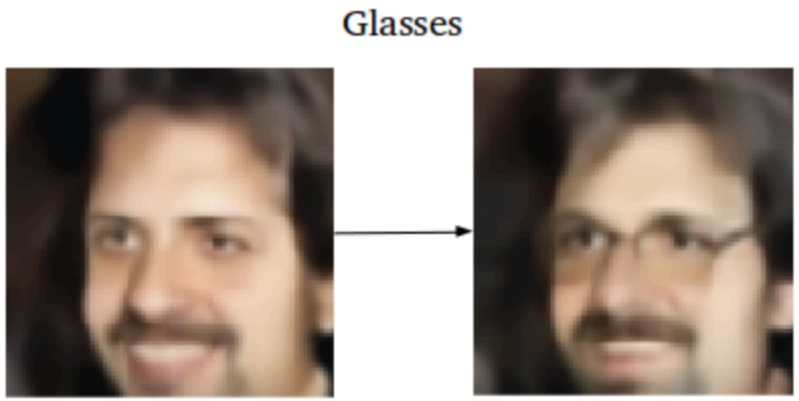


Minimize (KL) distance between latent vector and Gaussian normal

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



Adding new features to samples



Exploring a specific variation of input data[1]

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** = $\mathbf{P_g}$- $\mathbf{P_{ng}}$
- Encode new image to obtain $\mathbf{P_{new}}$ **,** add in **diff**
- Decode $\mathbf{P_{new}}$ **+ diff** to get guy with glasses!

Code review: See the following:

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

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