Lecture 14b: Beyond classification – segmentation and autoencoders

Machine Learning and Imaging

BME 590L
Roarke Horstmeyer

Class project option – work with a new image dataset from Kenya!

- Collaboration with Dr. Wendy Prudhomme-O'Meara at the Duke Global Health Institue
- Certain species of mosquitos carry the malaria parasite
- Classification of different mosquitos into different species at different locations/villages can offer some really useful information!

Dataset:

- **4 species imaged: 1710** images identified as gambiae, **402** as unidentified, **107** images as funestus but **only 17** as demeilloni

- Each species imaged 4 times from 4 directions

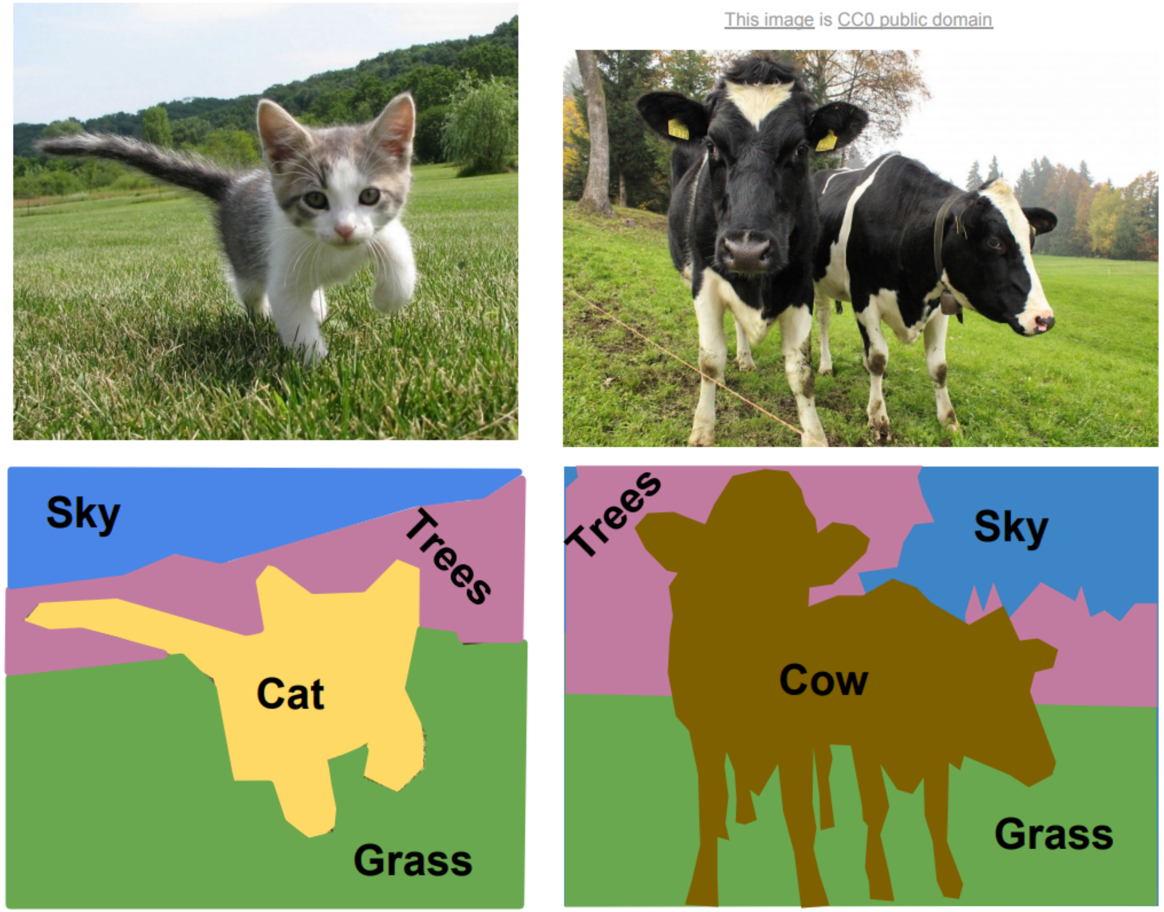
- In one of 4 states: unfed, blood-fed, gravid, half gravid

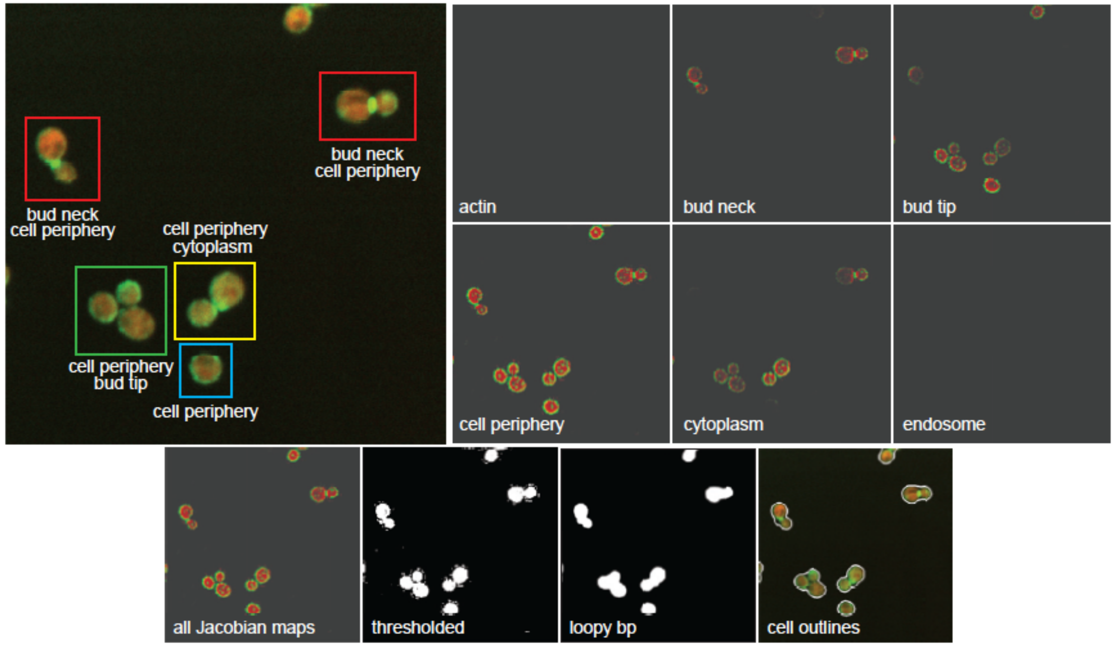
Task: Prep dataset, develop a classification (or other) network to establish some preliminary findings

Semantic Segmentation

Label each pixel in the image with a category label

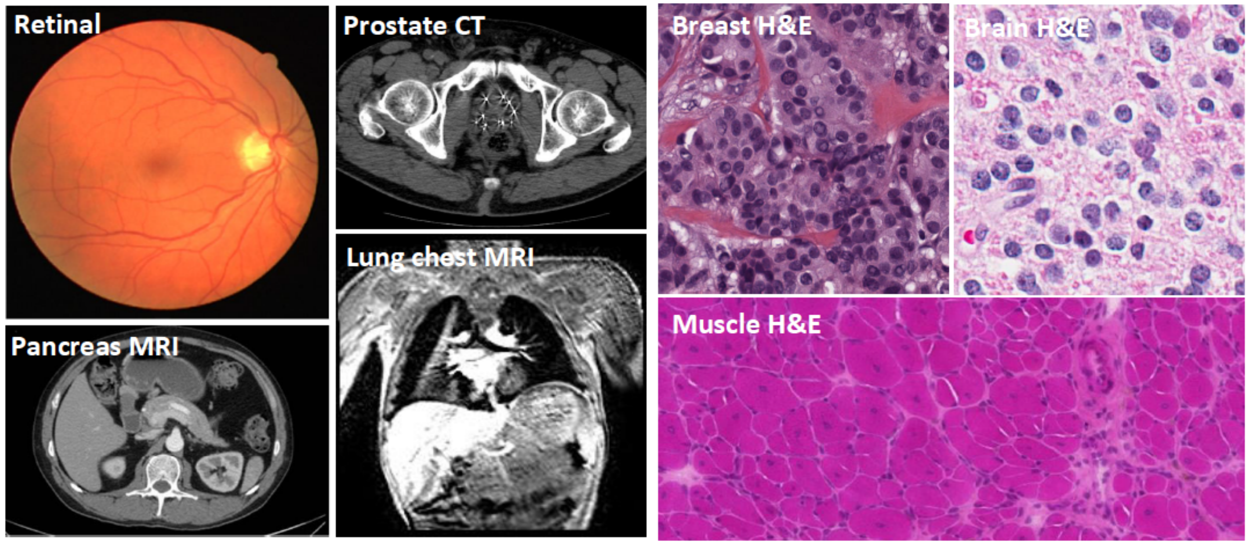
Don’t differentiate instances, only care about pixels

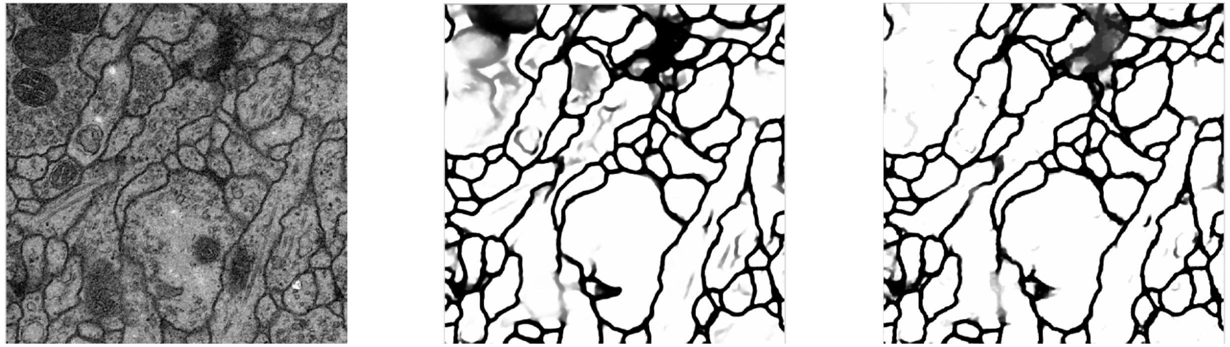




Oren Z. Kraus et al., "Classifying and Segmenting Microscopy Images Using Convolutional Multiple Instance Learning," arXiv 2015

Other possible examples:

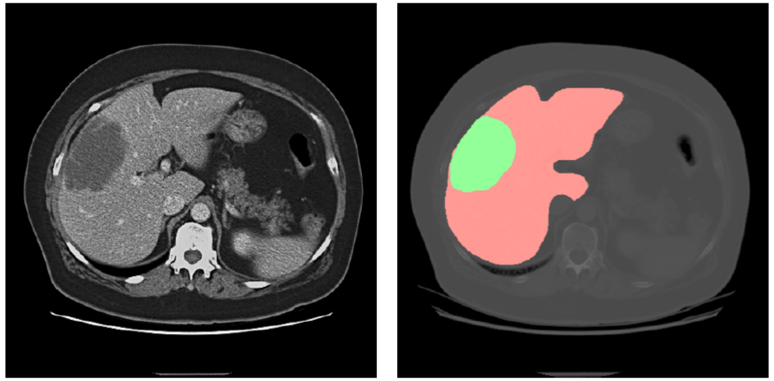




(a) Input image

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

(c) Segmentation result of our pipeline

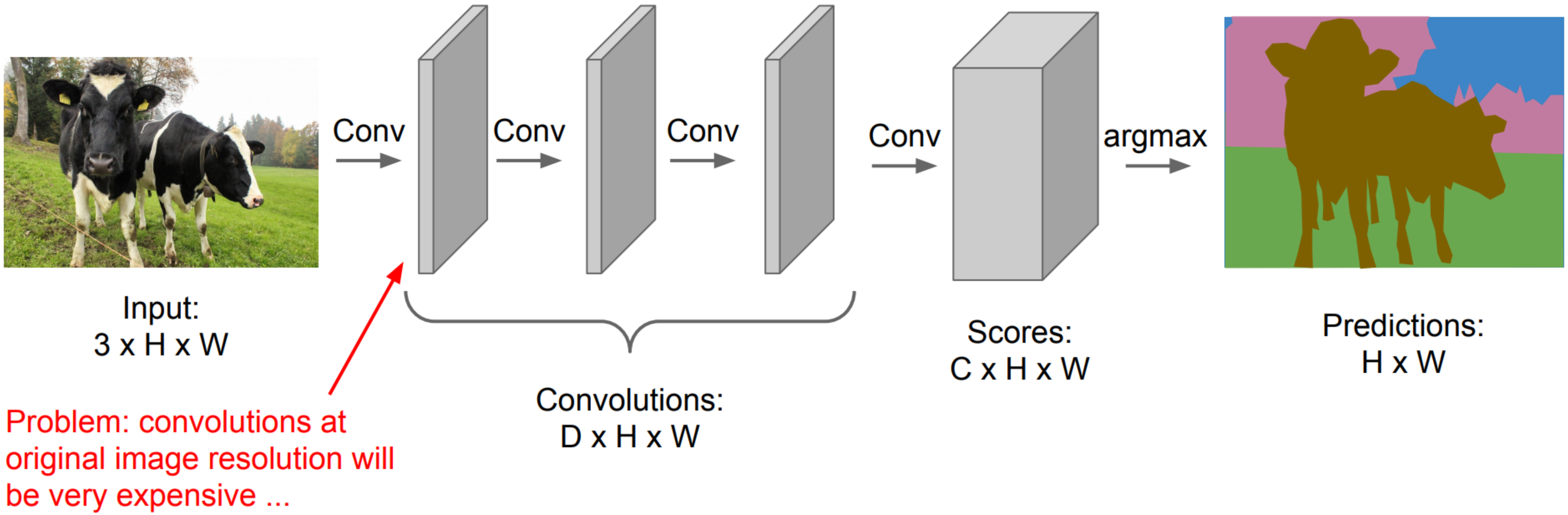


M. Drozdzal et al., Learning Normalized Inputs for Iterative Estimation in Medical Image Segmentation (2017)

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

Semantic Segmentation Idea: Fully Convolutional？

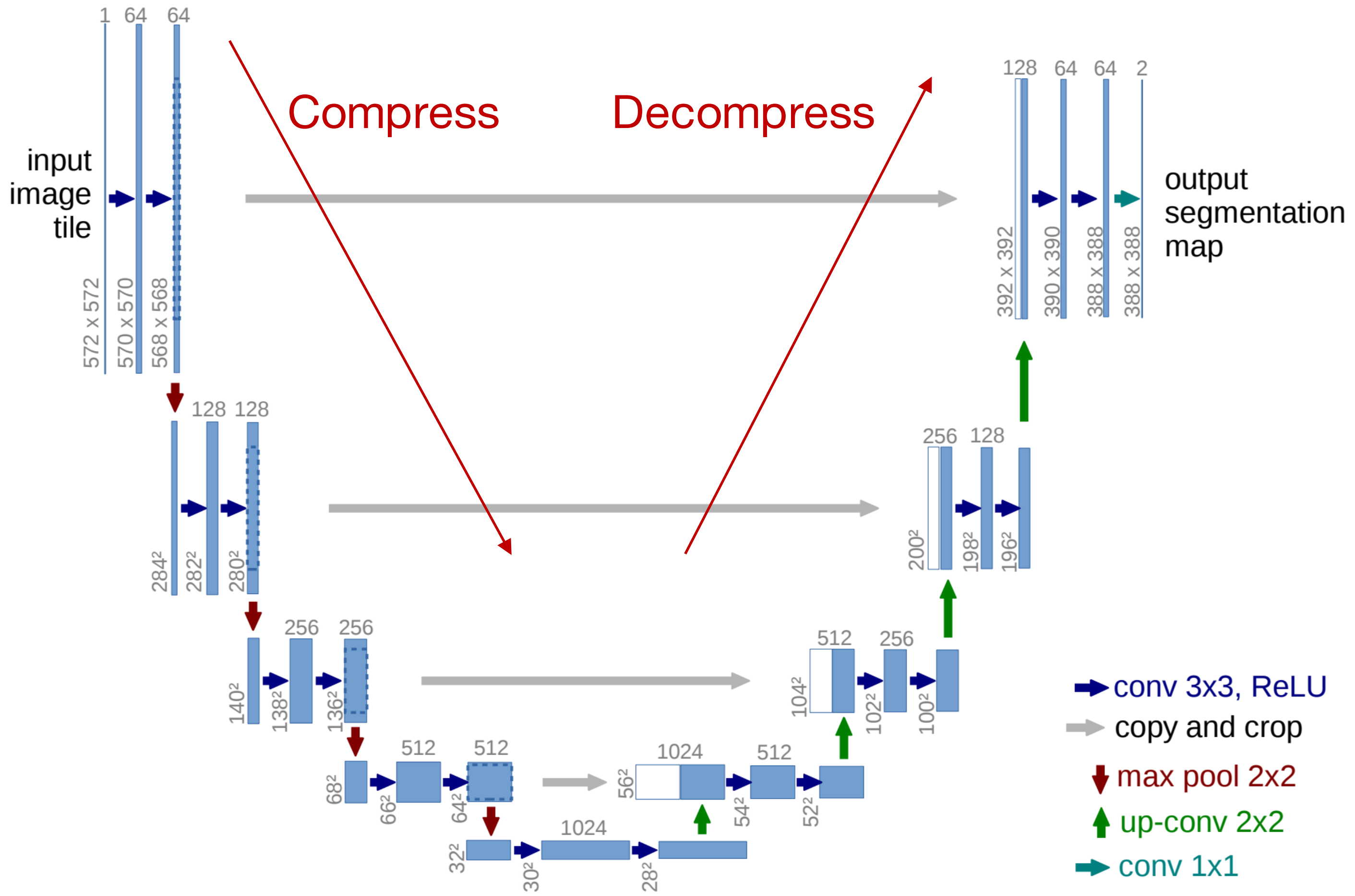
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



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/

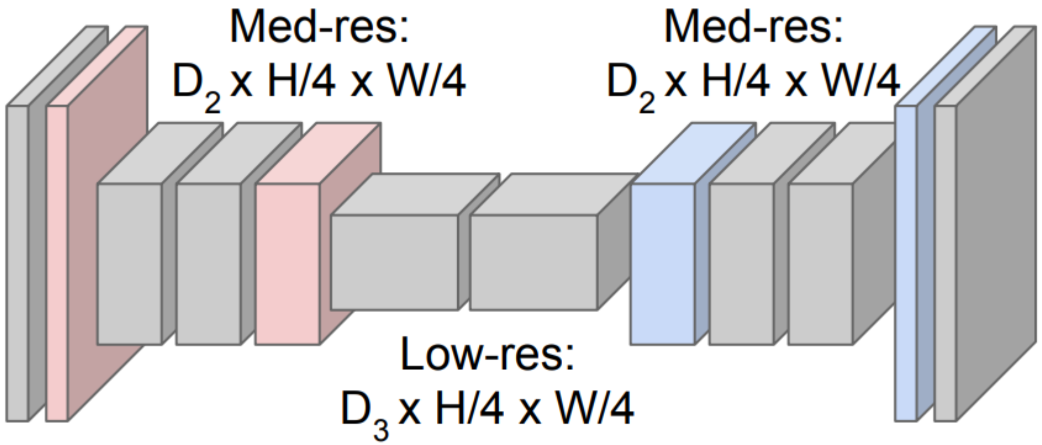
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: ???

Input: $3 \times H \times W$



High-res: $D_1 \times H/2 \times W/2$

High-res: $D_1 \times H/2 \times W/2$

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

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

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

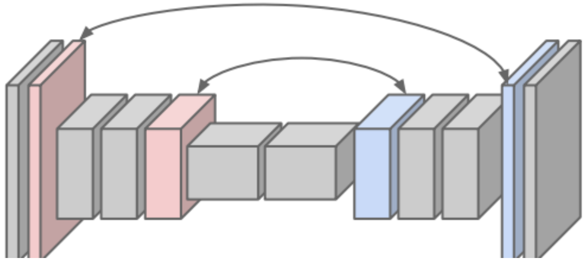
1	2
3	4

Input: 2 x 2

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

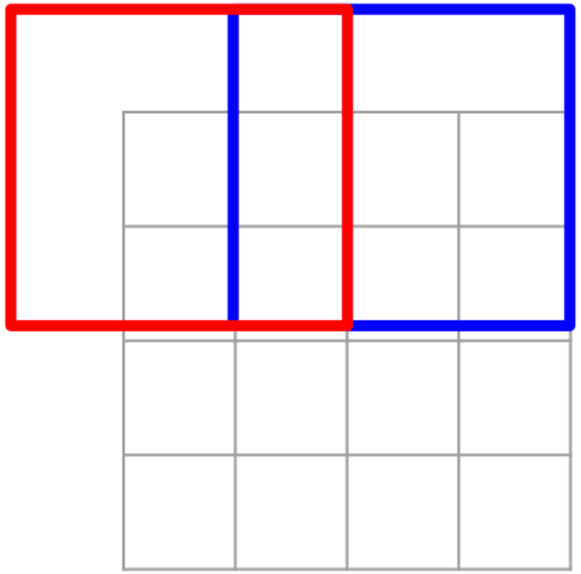
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



Learnable Upsampling: Transpose Convolution

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



Input: 4 x 4

Dot product between filter and input

Output: 2 x 2

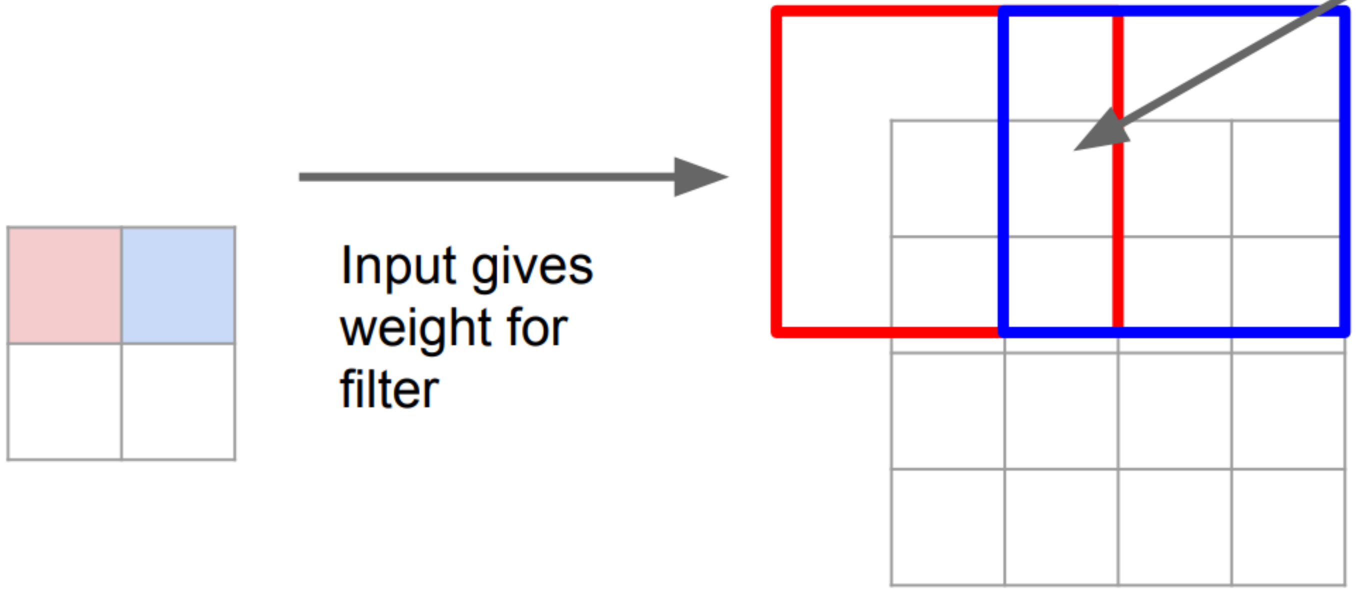
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Learnable Upsampling: Transpose Convolution

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

Sum where output overlaps



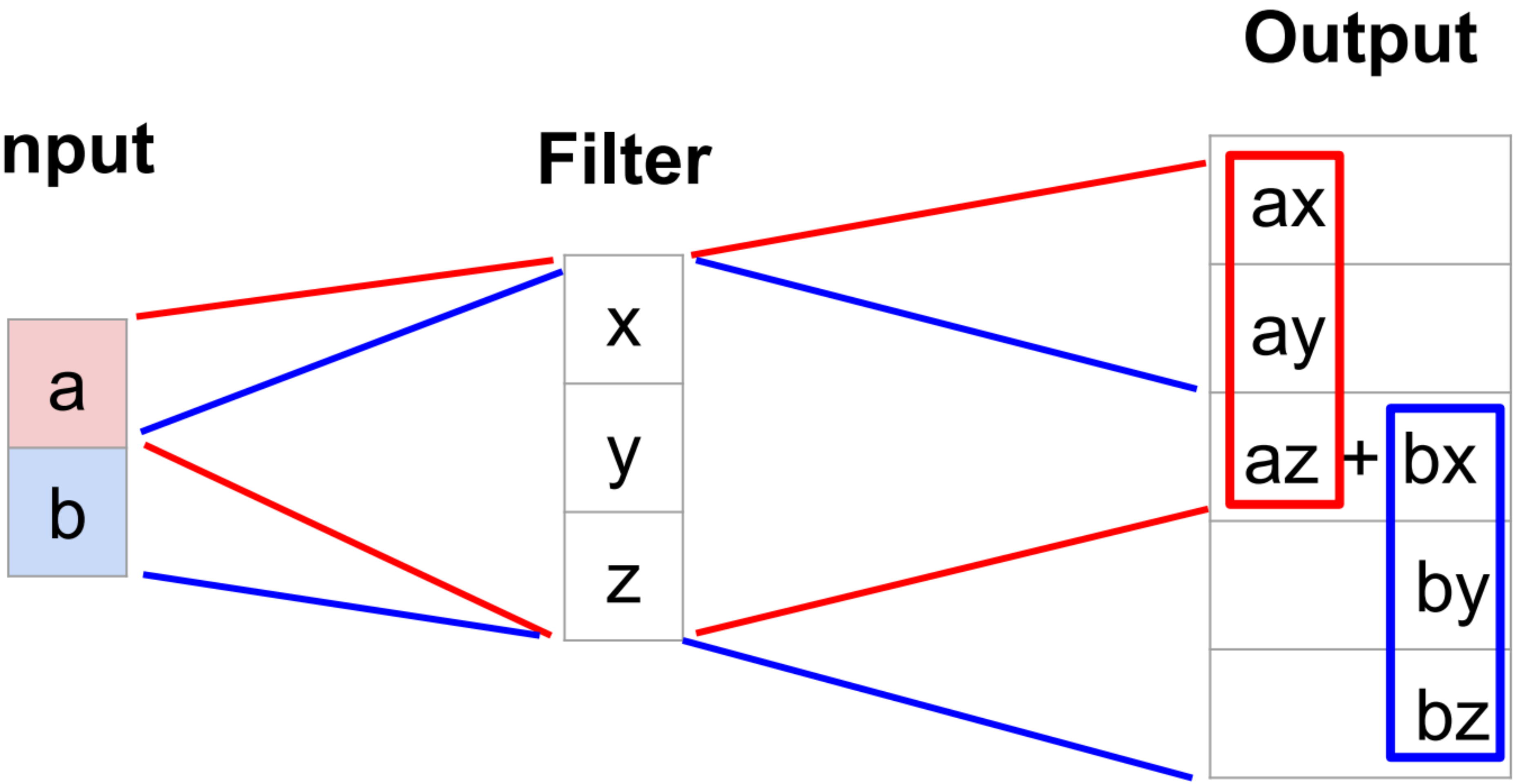
Input: 2 x 2

Output: 4 x 4

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

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

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

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

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

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

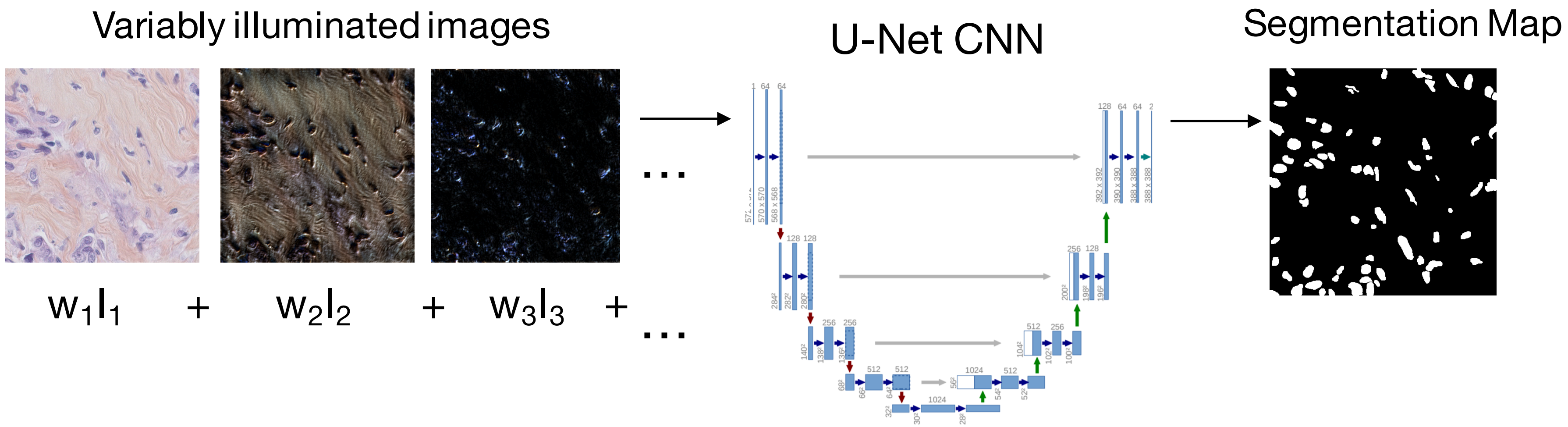
Convolution transpose multiplies by the transpose of the same matrix:

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

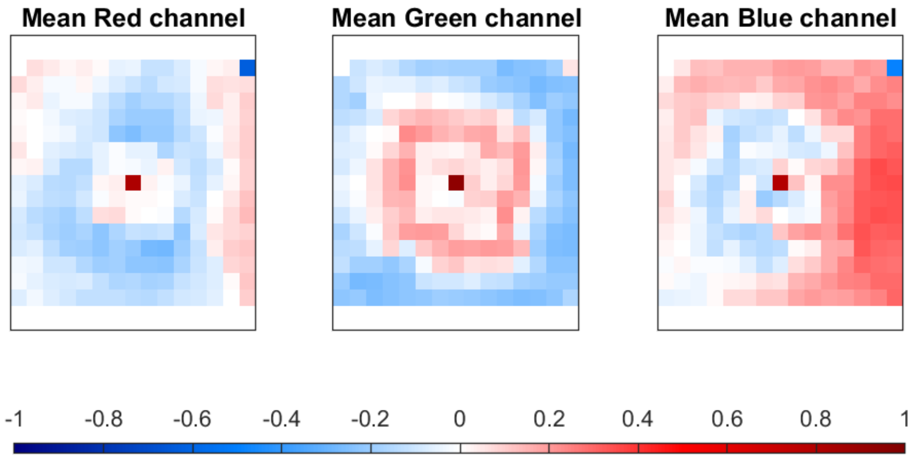
$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

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

Learned sensing for improved image segmentation



Optimized illumination for nuclei segmentation



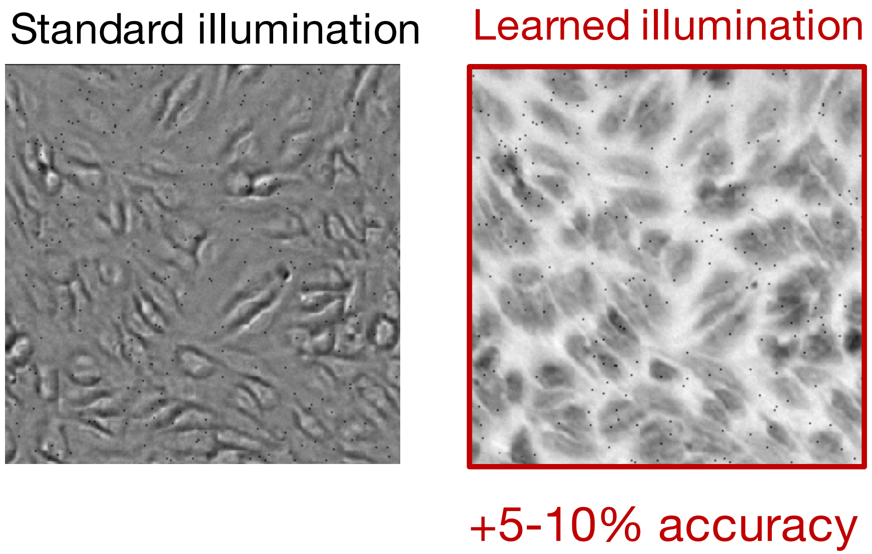
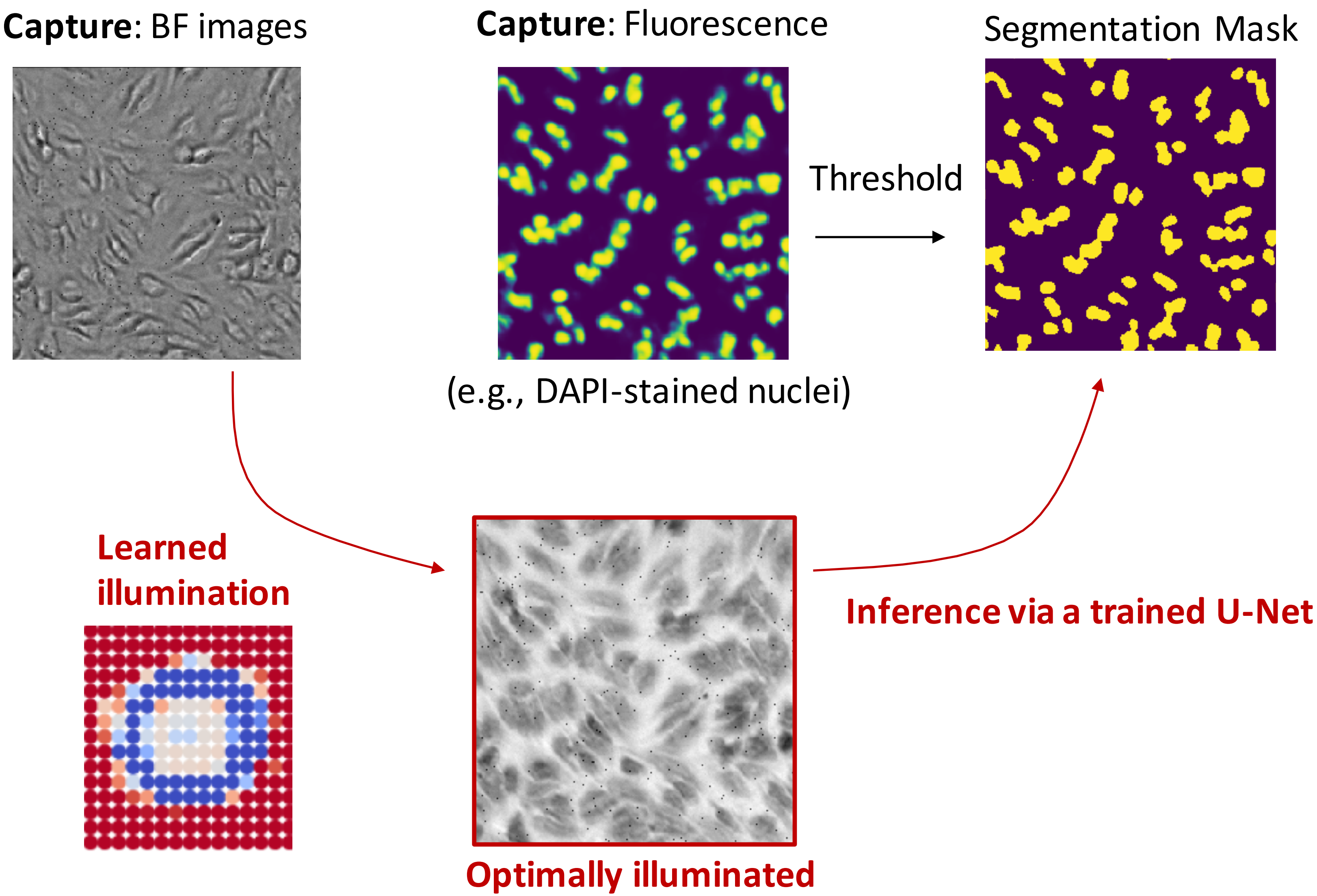


Image segmentation –current workflow

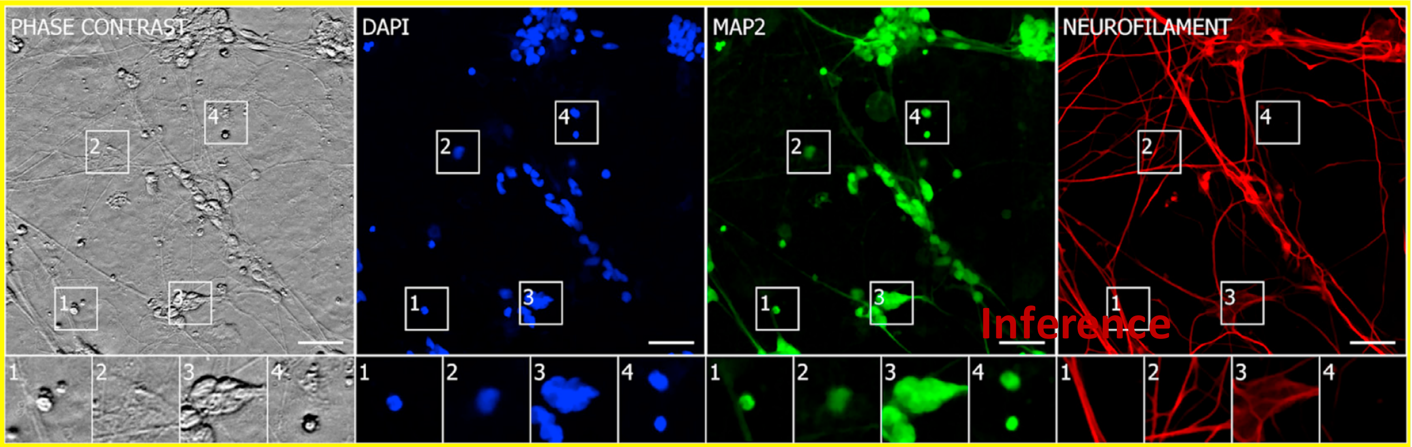


in silico labeling: fluorescence image inference from bright-field data

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

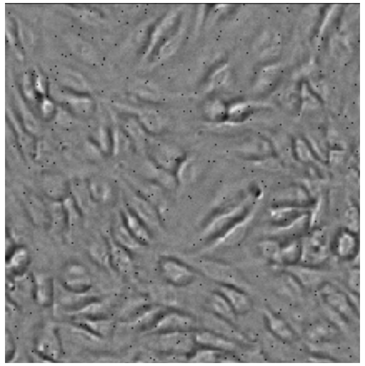


BF Focal stack (26+ images)

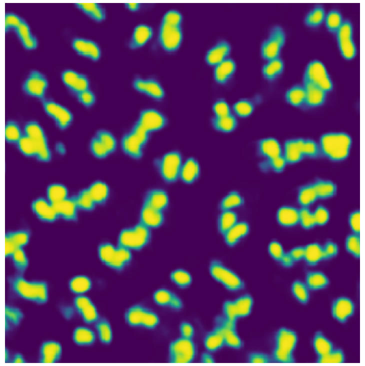
Task: bright-field to fluorescence image Inference

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

Capture: Bright field



Capture: Fluorescence



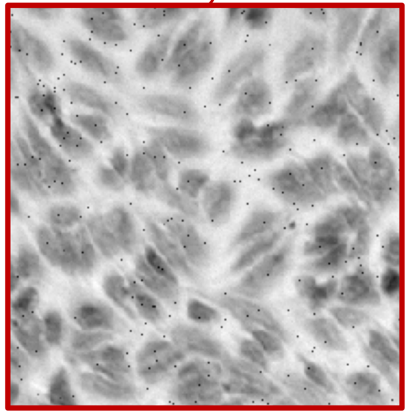
Threshold

Segmentation Mask

We can just Inference the fluorescence image itself...

Learned illumination for fluorescent image inference?

?

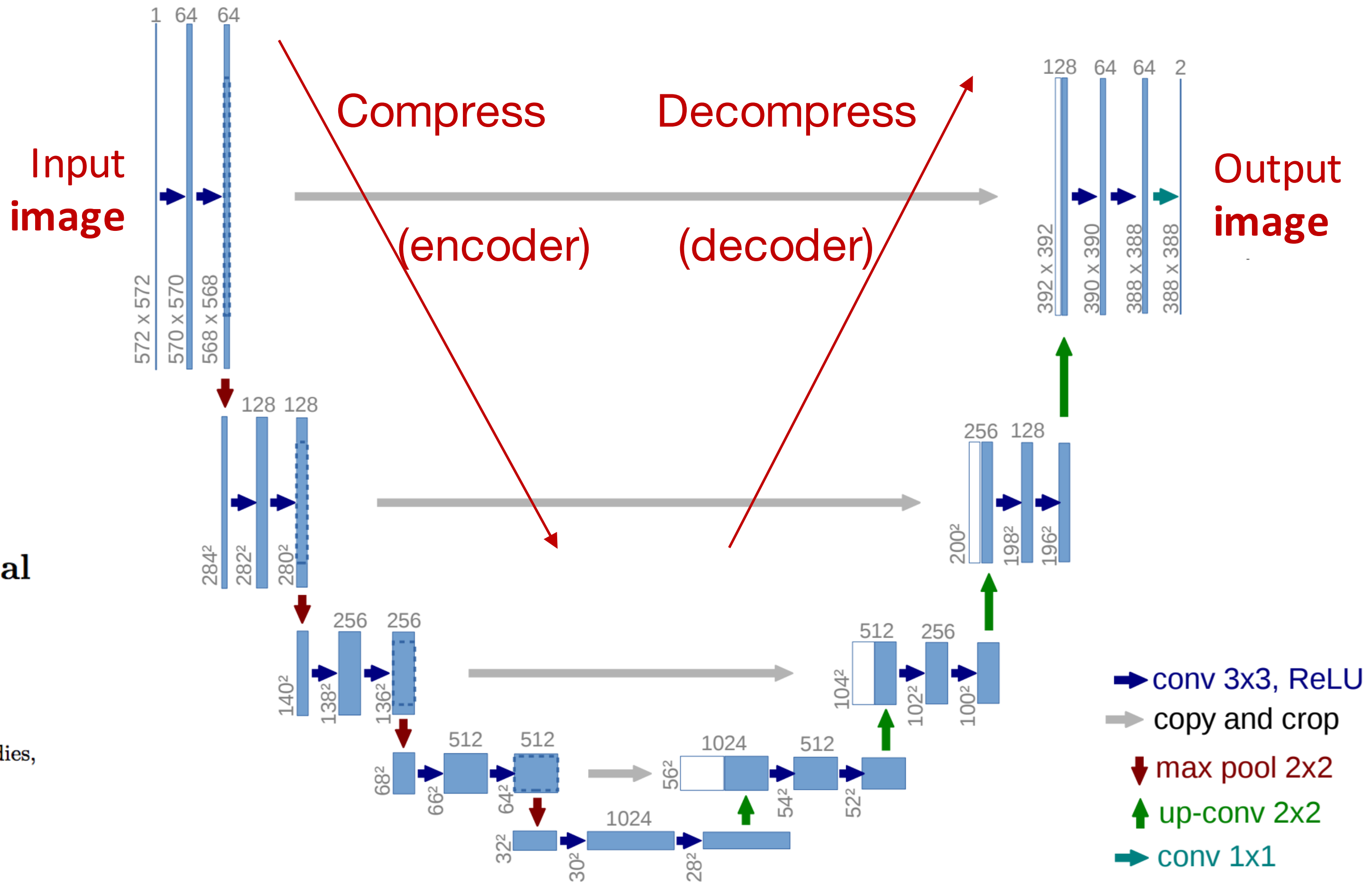


Optimally illuminated

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

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

U-Net Architecture



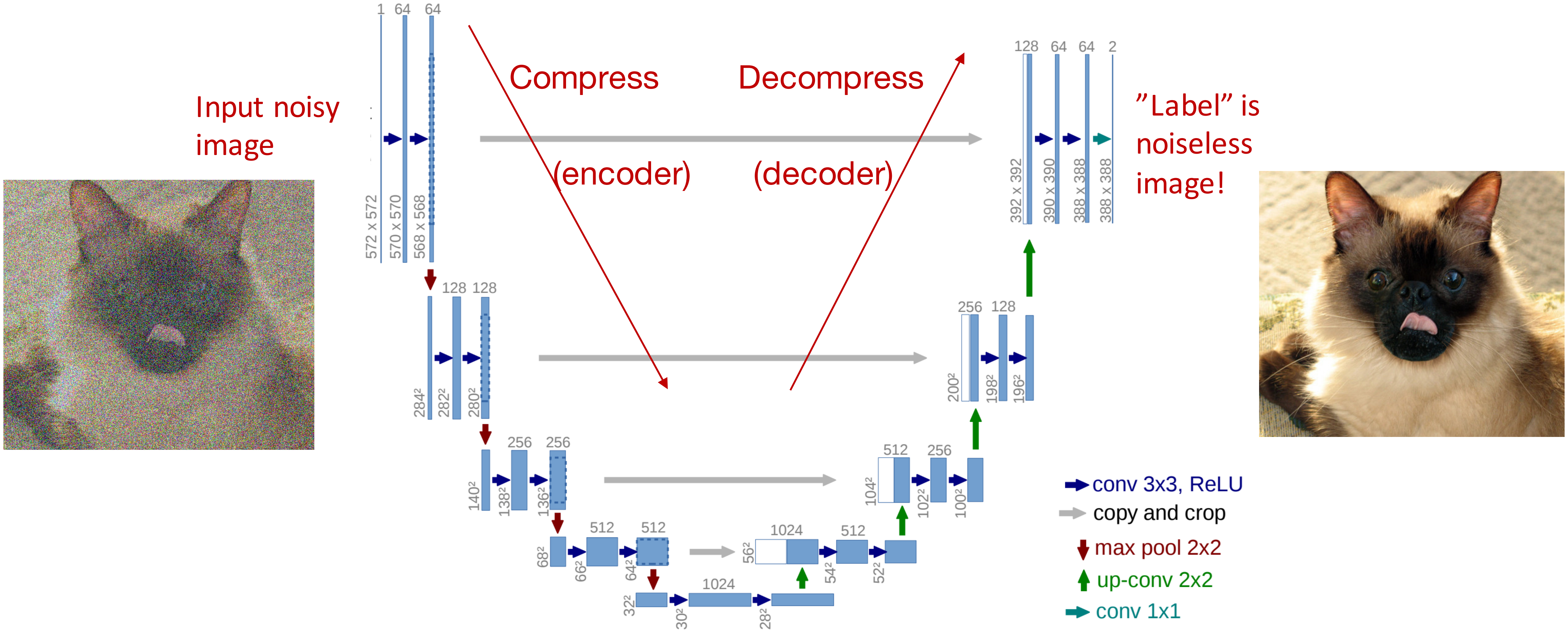
U-Net: Convolutional Networks for Biomedical Image Segmentation

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/

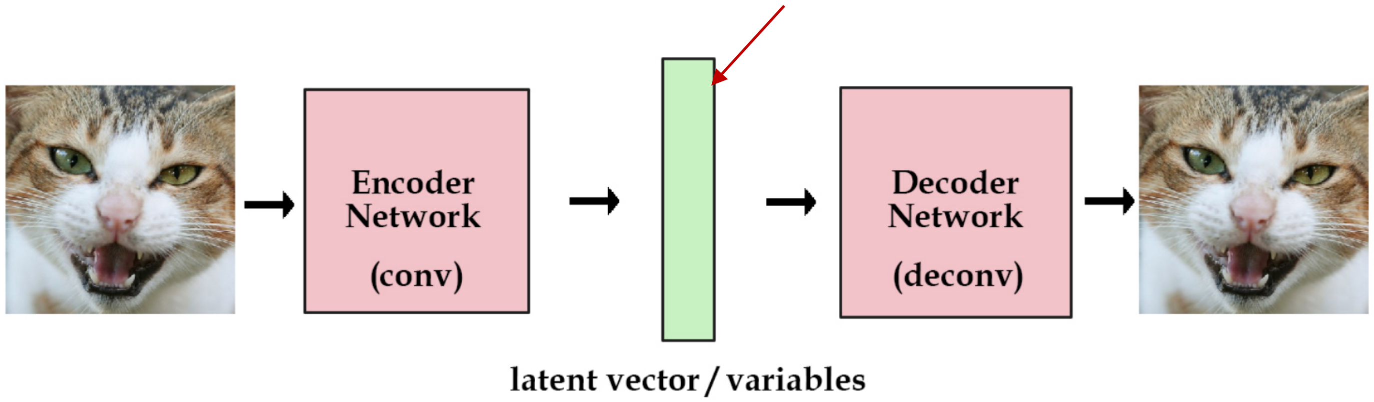
Another example: Denoising Autoencoder

U-Net Architecture



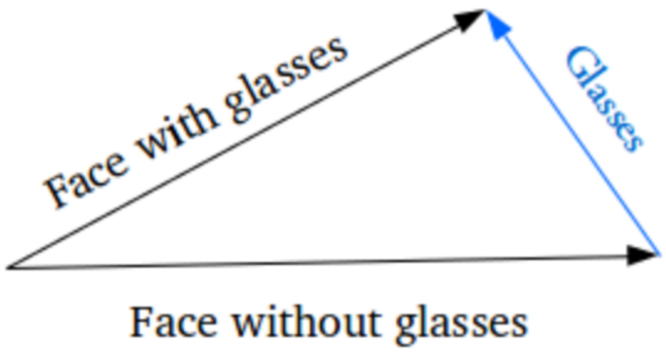
Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF



Minimize (KL) distance between latent vector and Gaussian normal

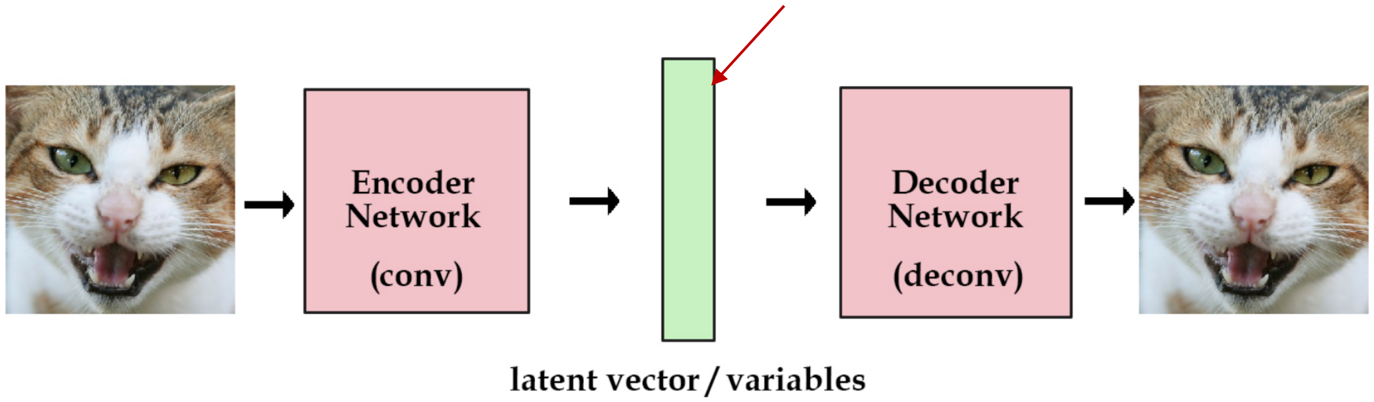
- With Gaussian PDF, can start to add/subtract latent vector in a normalized vector space



Adding new features to samples

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

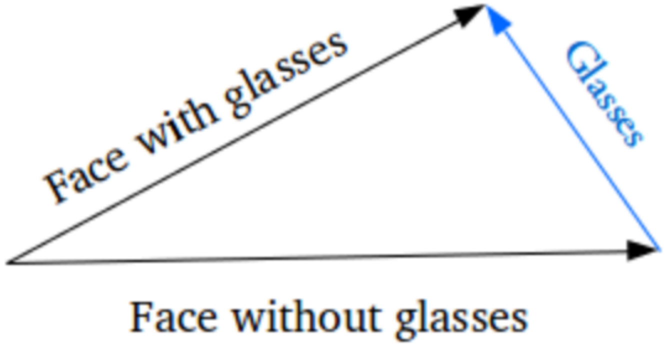


Minimize (KL) distance between latent vector and Gaussian normal

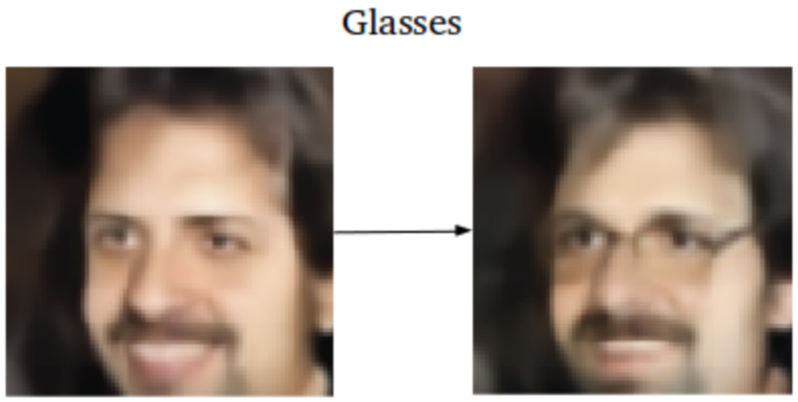
Generative Example (once trained):

- Encode image with glasses, obtain latent vector PDF $\mathbf{P_g}$
- Encode image without glasses, obtain PDF $\mathbf{P_{ng}}$
- Compute **diff** = $\mathbf{P_g} - \mathbf{P_{ng}}$
- 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



Exploring a specific variation of input data[1]

Code review: See the following:

Jupyter Notebook: A simple Autoencoder in Tensorflow/Keras

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