Neural Networks and Deep Learning

Fully Convolutional

Agenda

- **Image Segmentation**
- **Semantic segmentation**
- **Datasets for semantic/instance segmentation**
- **Fully Convolutional**
- **U-Net**

Image and object segmentation

- **Image Segmentation**
  - Group pixels into regions that share some similar properties

Superpixels
(Ren ICCV 2003

- **Segmenting images into meaningful objects**
  - Object-level segmentation: accurate localization and recognition

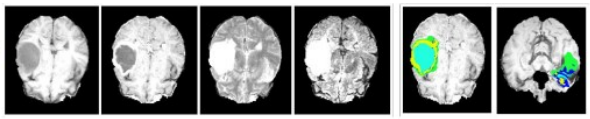
Object segmentation: applications

Image editing and composition (Xu, 2016)

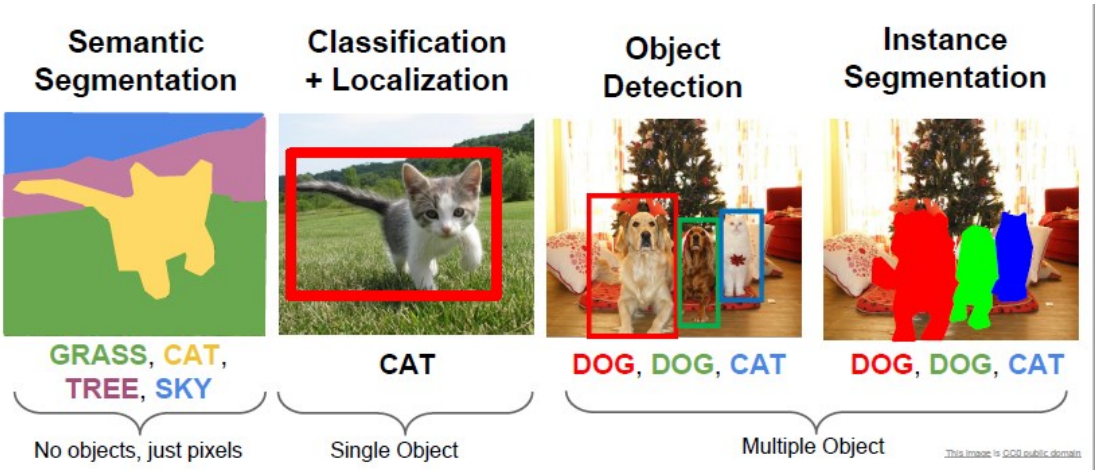
Robotics

Autonomous driving
(cordts, 2016)

Medical image analysis
(Casamitjana, 2017)



Segmentation, Localization, Detection



Semantic segmentation

- Label every pixel in the image with a category label: recognize the class of every pixel

- Do not differentiate instances , only care about pixels

Mottaghi et al, "The role of context for object detection and semantic segmentation in the wild", CVPR 2014

Instance segmentation

- Detect instances, categorize and label every pixel
- Labels are class-aware and instance-aware

Arnab,Torr "Pixelwise instance segmentation with a dynamically instantiated network", CVPR 2017

Datasets for semantic/instance segmentation

Pascal Visual Object Classes

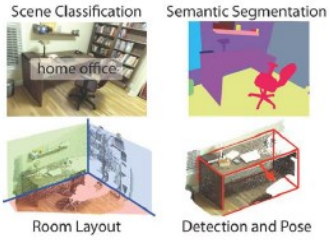
- 20 categories
- +10,000 images
- Semantic segmentation GT
- Instance segmentation GT

Pascal Context

- Real indoor & outdoor scenes
- 540 categories
- +10,000 images
- Dense annotations
- Semantic segmentation GT
- Objects + stuff

Datasets for semantic/instance segmentation

SUN RGB-D



- Real indoor scenes
- 10,000 images
- 58,658 3D bounding boxes
- Dense annotations
- Instances GT
- Semantic segmentation GT
- Objects + stuff

COCO Common Objects in Context

- Real indoor & outdoor scenes
- 80 categories
- +300,000 images
- 2M instances
- Partial annotations
- Semantic segmentation GT
- Instance segmentation GT
- Objects, but no stuff

Datasets for semantic/instance segmentation

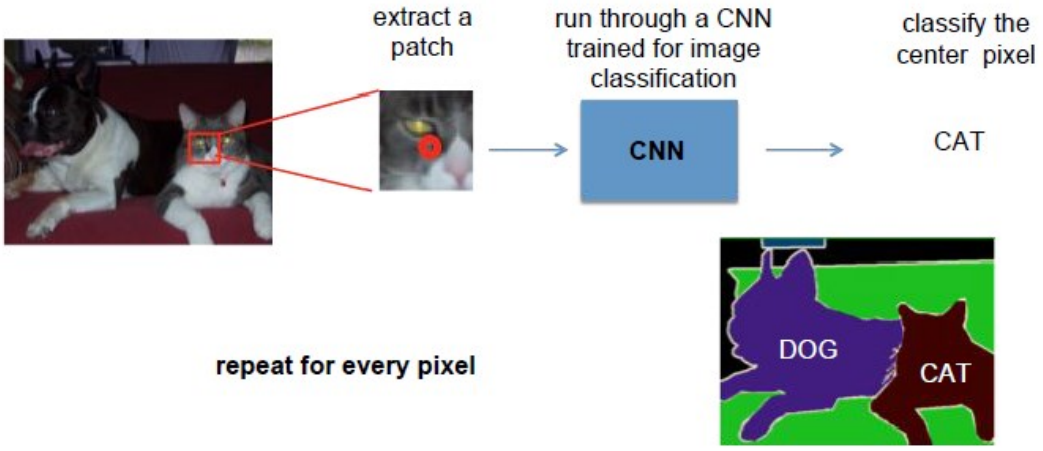
CityScapes

- Real driving scenes
- 30 categories
- +25,000 images
- 20,000 partial annotations
- 5,000 dense annotations
- Semantic segmentation GT
- Instance segmentation GT
- Depth, GPS and other metadata
- Objects and stuff

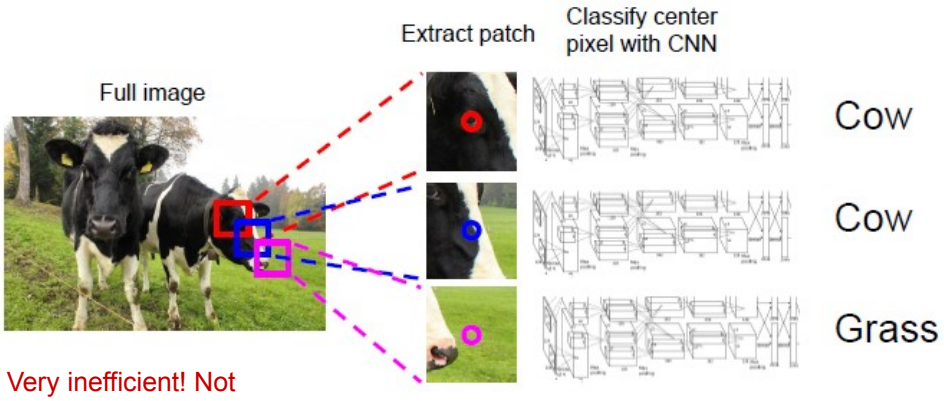
ADE20K

- Real general scenes
- +150 categories
- +22,000 images
- Semantic segmentation GT
- Instance + parts segmentation GT
- Objects and stuff

From classification to semantic segmentation



Semantic Segmentation Idea: Sliding Window



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

- A classification network becoming fully convolutional
  - Fully connected layers can also be viewed as convolutions with kernels that cover the entire input region

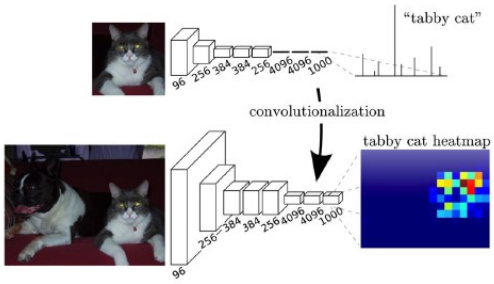
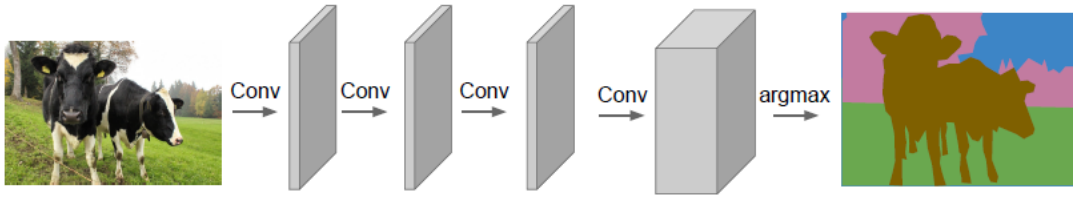


Fig. 2. Transforming fully connected layers into convolution layers enables a classification net to output a spatial map. Adding differentiable interpolation layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end pixelwise learning.

Shelhamer, Long, Darrell, Fully Convolutional Networks for Semantic Segmentation, 2014-2016

Semantic Segmentation Idea: Fully Convolutional

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



Input:
3 x H x W

Convolutions:
D x H x W

Scores:
C x H x W

Predictions:
H x W

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

#channels = #classes

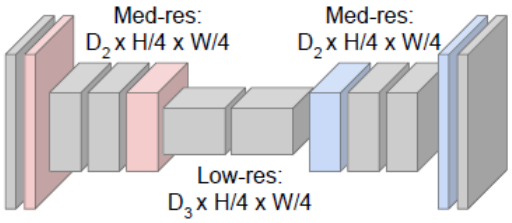
Semantic Segmentation Idea: Fully Convolutional

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

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"

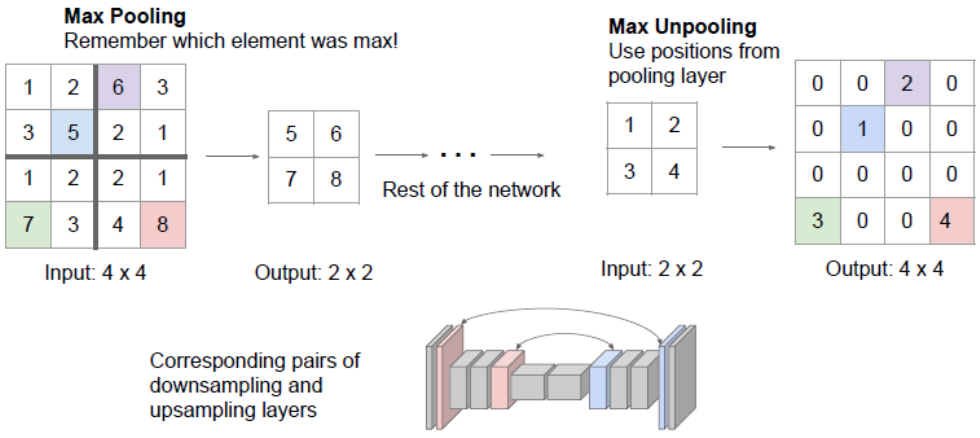
1	2
3	4

Input: 2 x 2

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

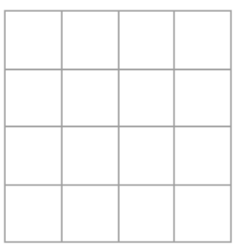
Output: 4 x 4

In-Network upsampling: “Max Unpooling”

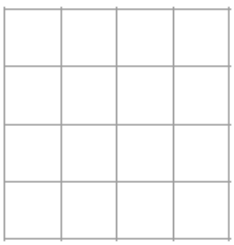


Learnable Upsampling: Transpose Convolution

Recall: Typical 3 x 3 convolution, stride 1 pad 1



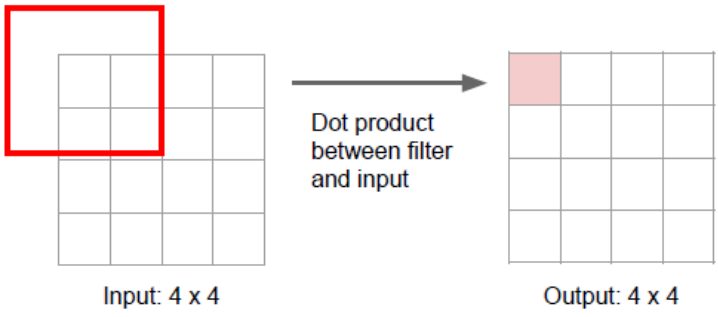
Input: 4 x 4



Output: 4 x 4

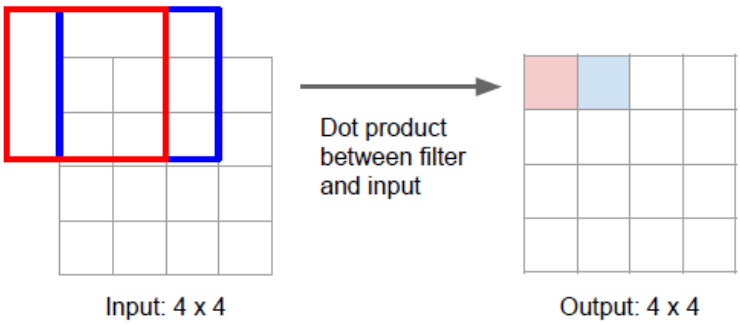
Learnable Upsampling: Transpose Convolution

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



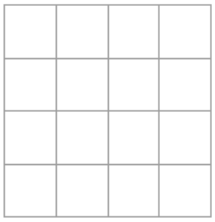
Learnable Upsampling: Transpose Convolution

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

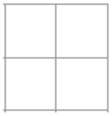


Learnable Upsampling: Transpose Convolution

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



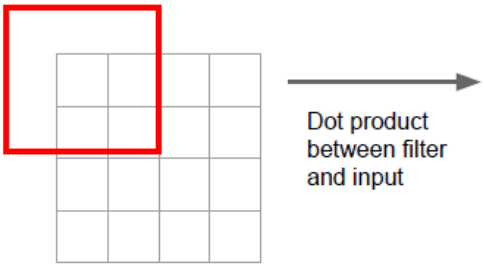
Input: 4 x 4



Output: 2 x 2

Learnable Upsampling: Transpose Convolution

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

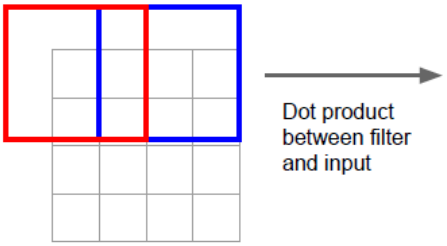


Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transpose Convolution

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



Dot product between filter and input

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

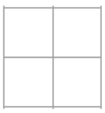
Stride gives ratio between movement in input and output

Input: 4 x 4

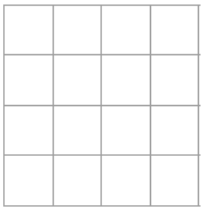
Output: 2 x 2

Learnable Upsampling: Transpose Convolution

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

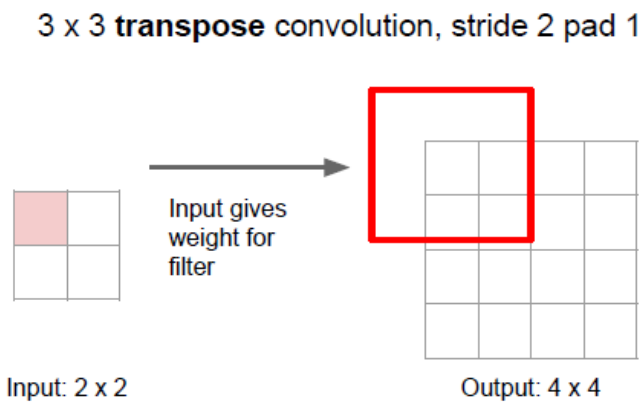


Input: 2 x 2



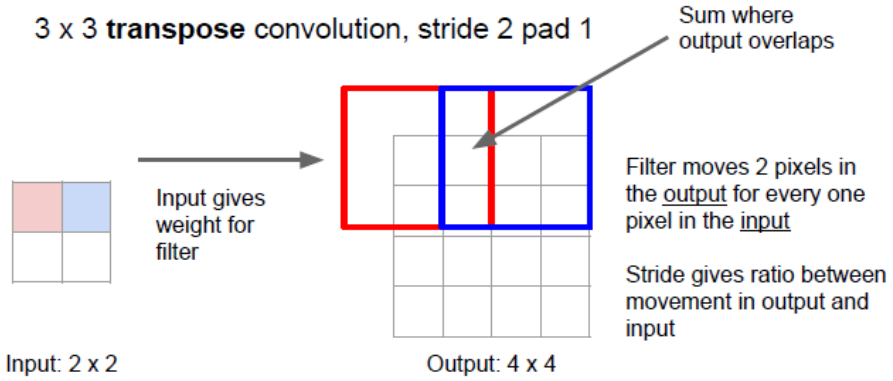
Output: 4 x 4

Learnable Upsampling: Transpose Convolution

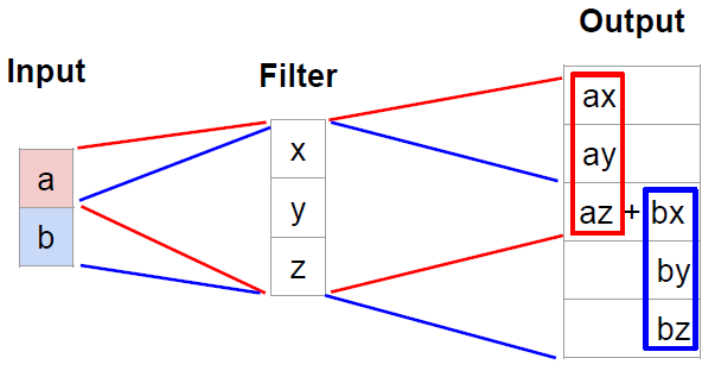


Learnable Upsampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution



Transpose Convolution: 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

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

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

Example: 1D conv, kernel size=3, 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!

Semantic Segmentation Idea: Fully Convolutional

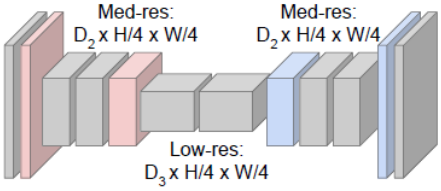
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: Unpooling or strided transpose convolution

Input: 3 x H x W



High-res: D1 x H/2 x W/2

High-res: D1 x H/2 x W/2

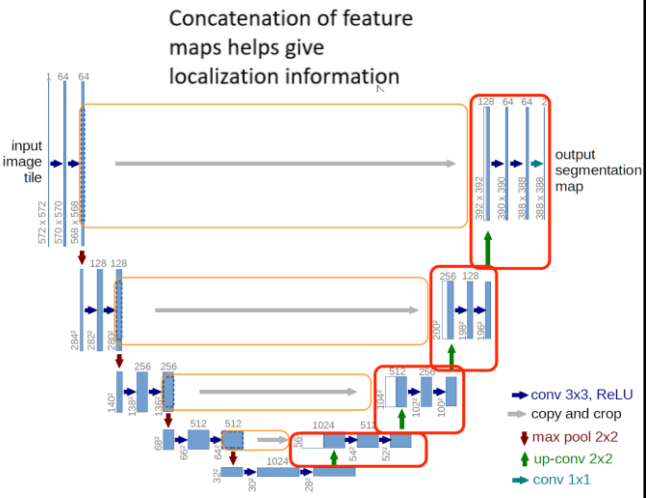
Predictions: H x W

Other architectures for Semantic segmentation

- **DeepLab:** 'atrous' convolutions + spatial pyramid + CRF (Chen, ICLR 2015)
- **CRF-RNN:** FCN + CRF as Recurrent NN (Zheng, ICCV 2015)
- **U-Net** (Ronnemberger, 2015)
- **Fully Convolutional DenseNets** (Jégou, 2016)
- **Dilated convolutions** (Yu, 2016)

U-Net

- A contracting path and an expansive path
- Adds convolutions in the upsampling path ("symmetric" net)
- Skip connections: concatenation of feature maps



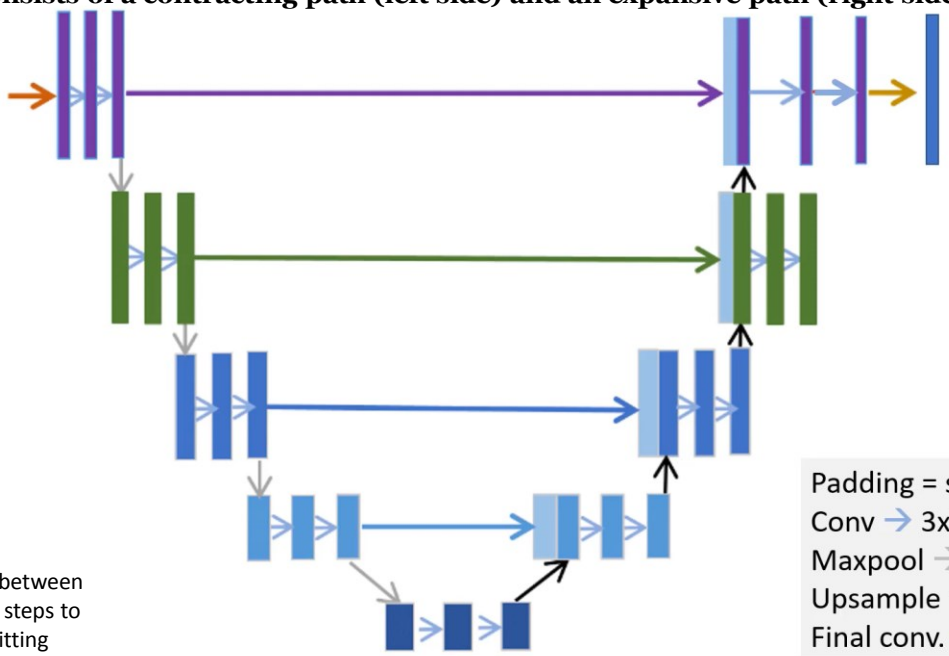
Winner of
CAD Caries challenge ISBI 2015
Cell tracking challenge ISBI 2015

Ronneberger et al, "U-Net: Convolutional Networks for Biomedical Image Segmentation", arXiv 2015

U-Net

- U-Net takes its name from the architecture, which when visualized, appears similar to the letter *U.*
- Input images are obtained as a segmented output map.
- The most special aspect of the architecture in the second half.
- The network does not have a fully-connected layer. Only the convolution layers are used.
- Each standard convolution process is activated by a ReLU activation function.

U-Net consists of a contracting path (left side) and an expansive path (right side)!



Also dropout between convolutional steps to prevent overfitting

Padding = same
Conv → 3x3 ReLU
Maxpool → 2x2 (Stride=2)
Upsample → 2x2
Final conv. → 1x1

Contraction path (downsampling)

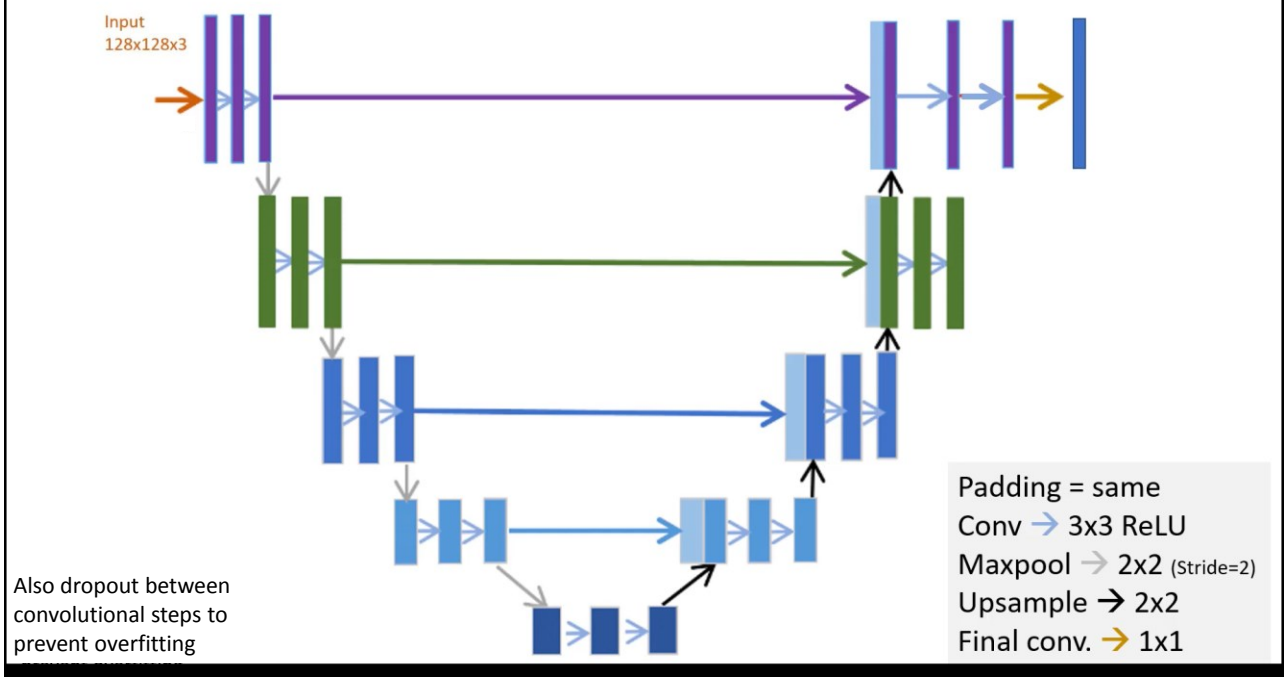
- Look like a typical CNN architecture, by consecutive stacking two 3x3 convolutions (blue arrow) followed by a 2x2 max pooling (red arrow) for downsampling. At each downsampling step, the number of channels is doubled.

Expansion path (up-convolution)

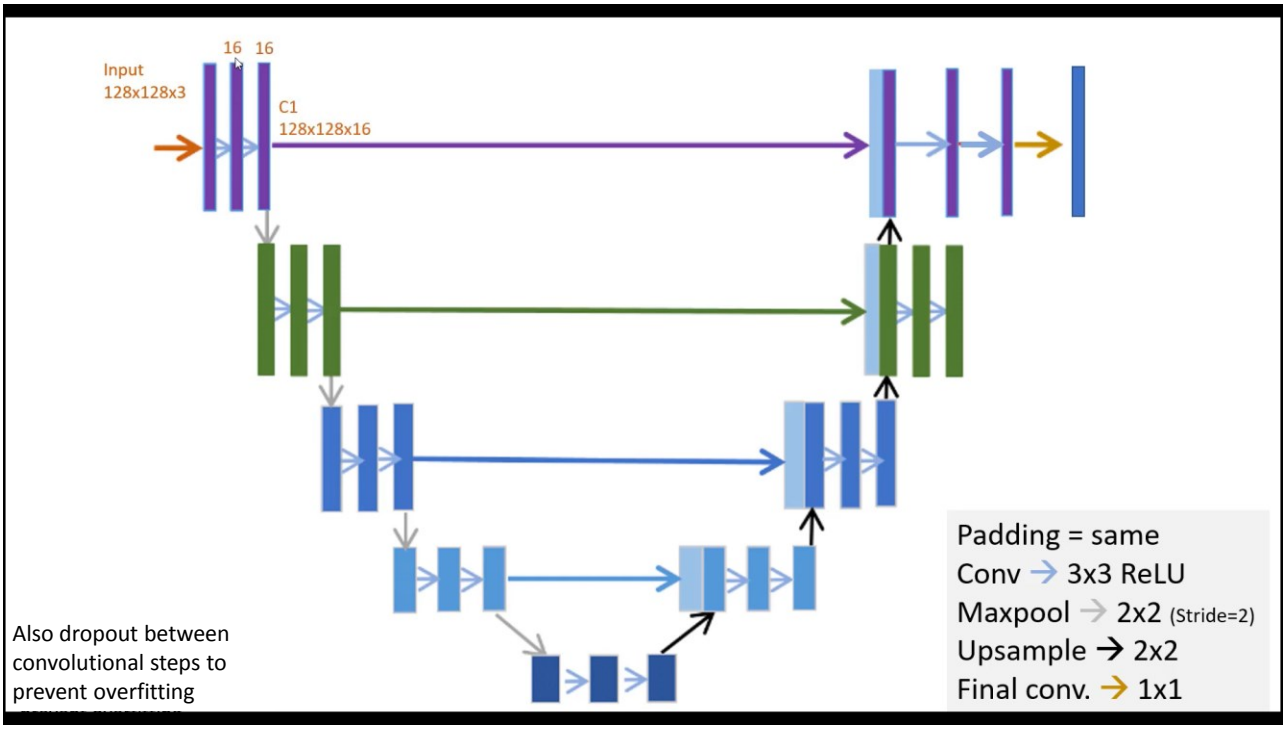
- A 2x2 up-convolution (green arrow) for upsampling and two 3x3 convolutions (blue arrow). At each upsampling step, the number of channels is halved.
- After each 2x2 up-convolution, a concatenation of feature maps with correspondingly layer from the contracting path (grey arrows), to provide localization information from contraction path to expansion path, due to the loss of border pixels in every convolution.

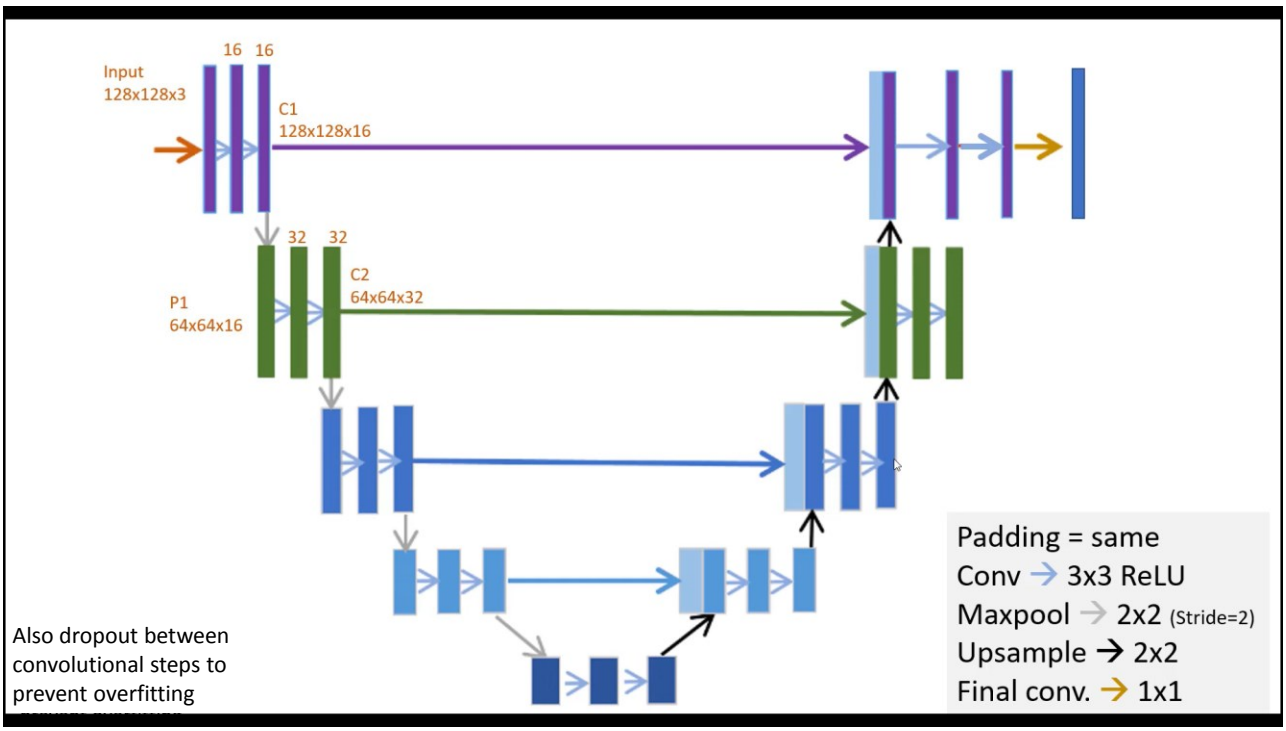
Final layer

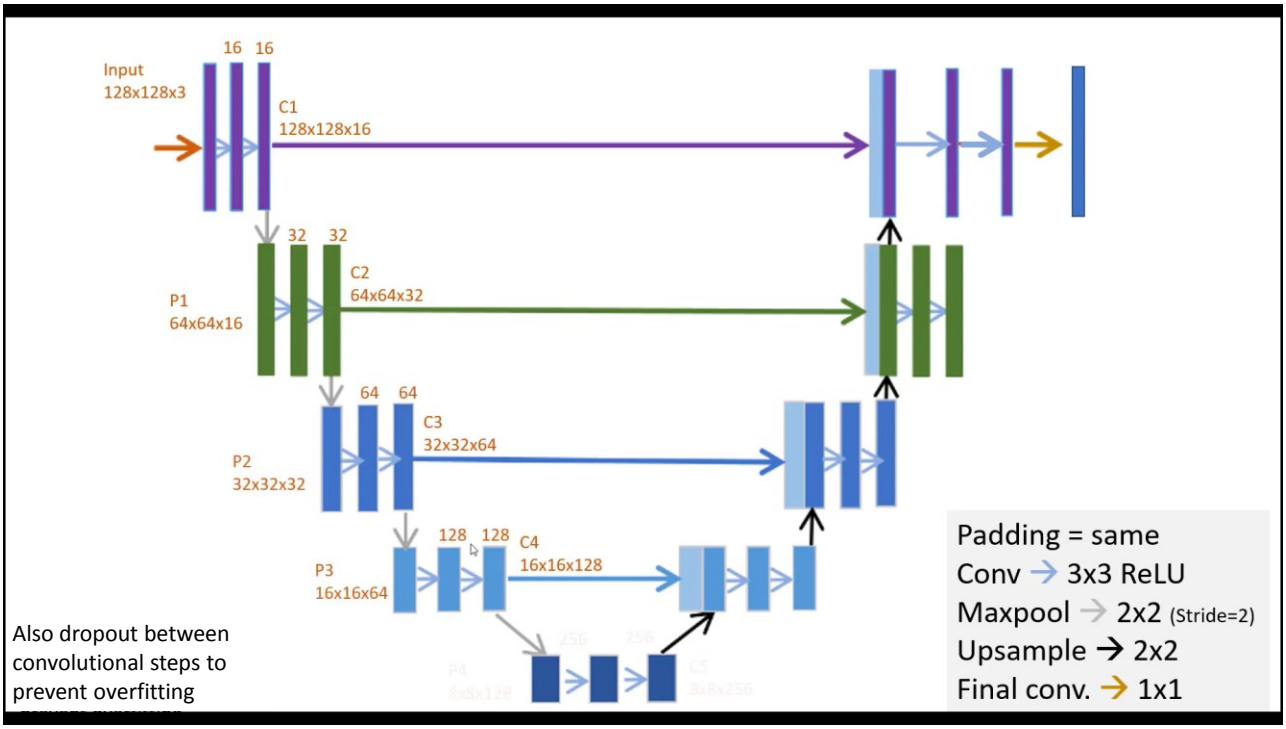
- A 1x1 convolution to map the feature map to the desired number of classes.



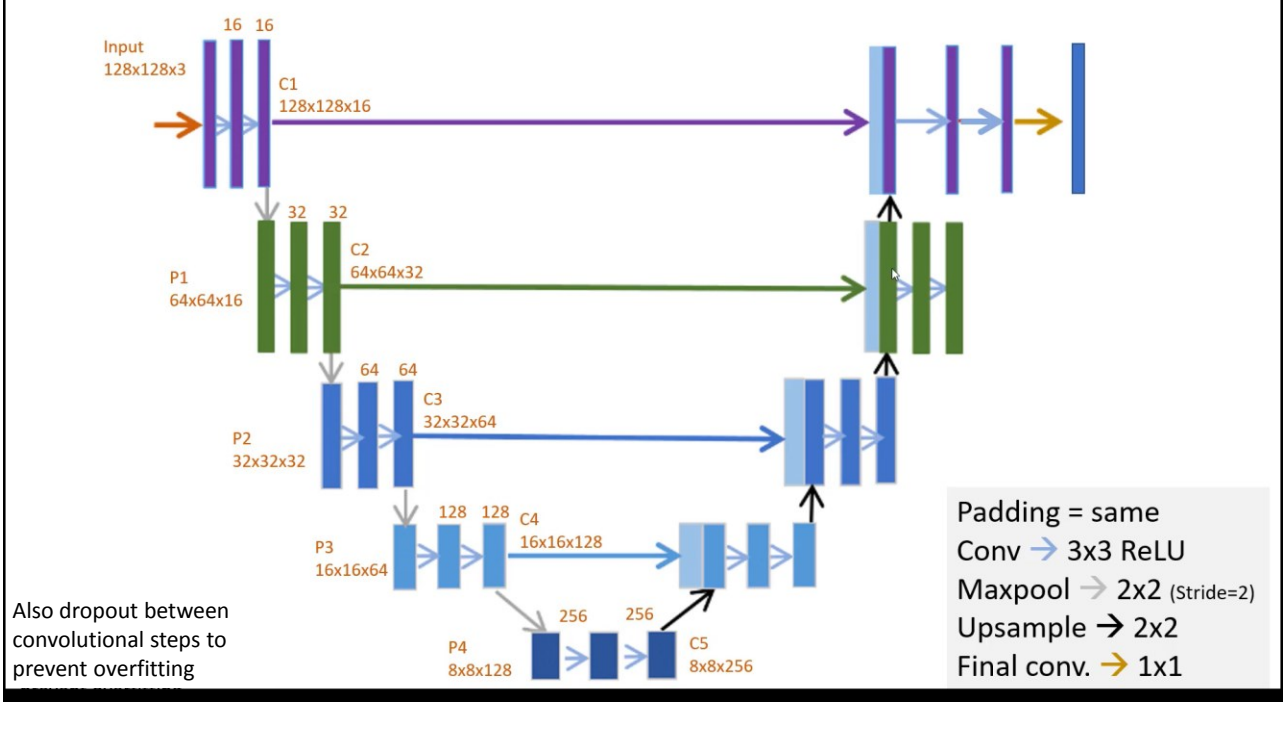
Also dropout between convolutional steps to prevent overfitting


16 16
Input
128x128x3
C1
128x128x16
Padding = same
Conv → 3x3 ReLU
Maxpool → 2x2 (Stride=2)
Upsample → 2x2
Final conv. → 1x1
Also dropout between convolutional steps to prevent overfitting

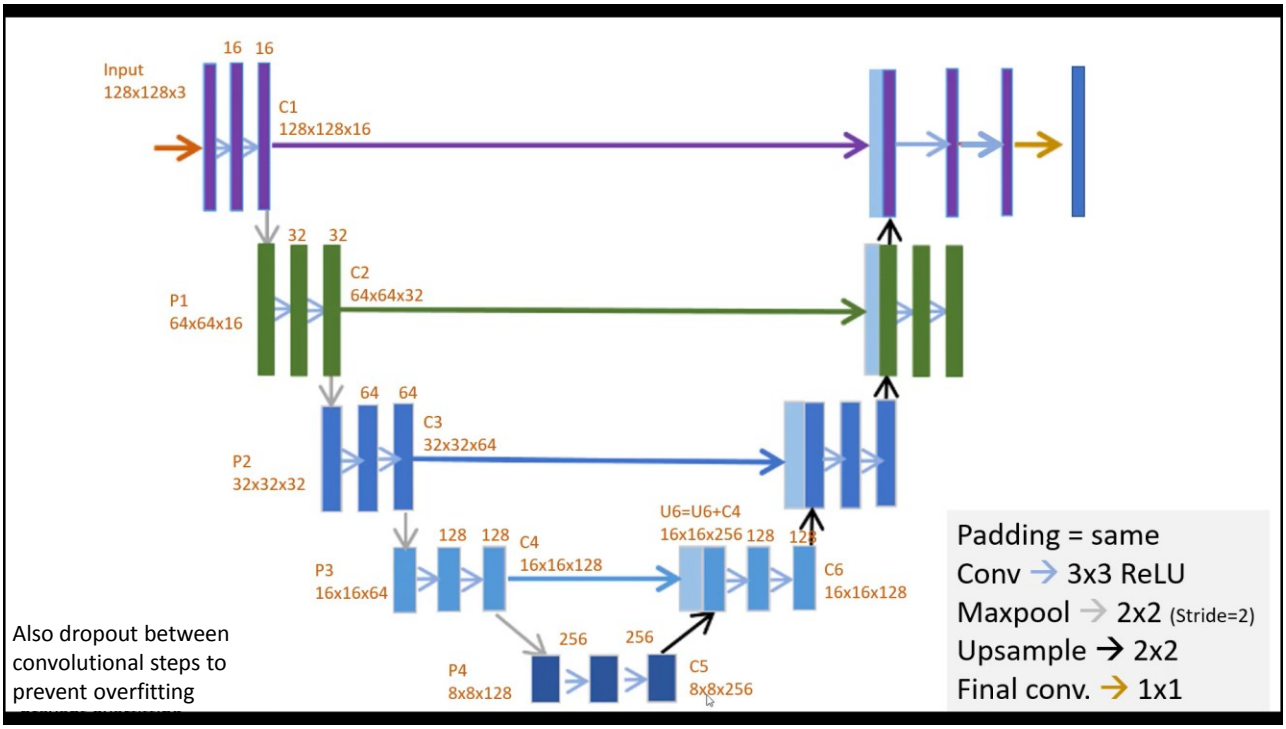

16 16
Input
128x128x3
C1
128x128x16
32 32
C2
64x64x32
P1
64x64x16
Padding = same
Conv → 3x3 ReLU
Maxpool → 2x2 (Stride=2)
Upsample → 2x2
Final conv. → 1x1
Also dropout between convolutional steps to prevent overfitting

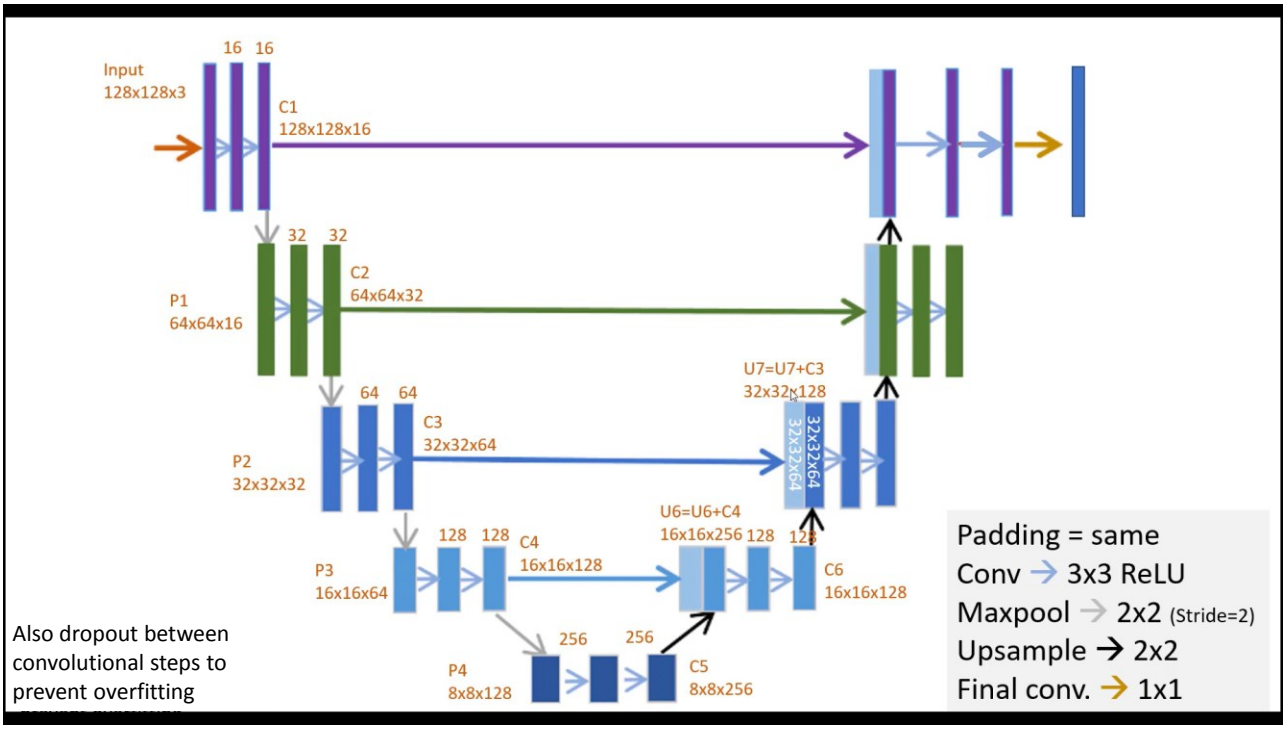


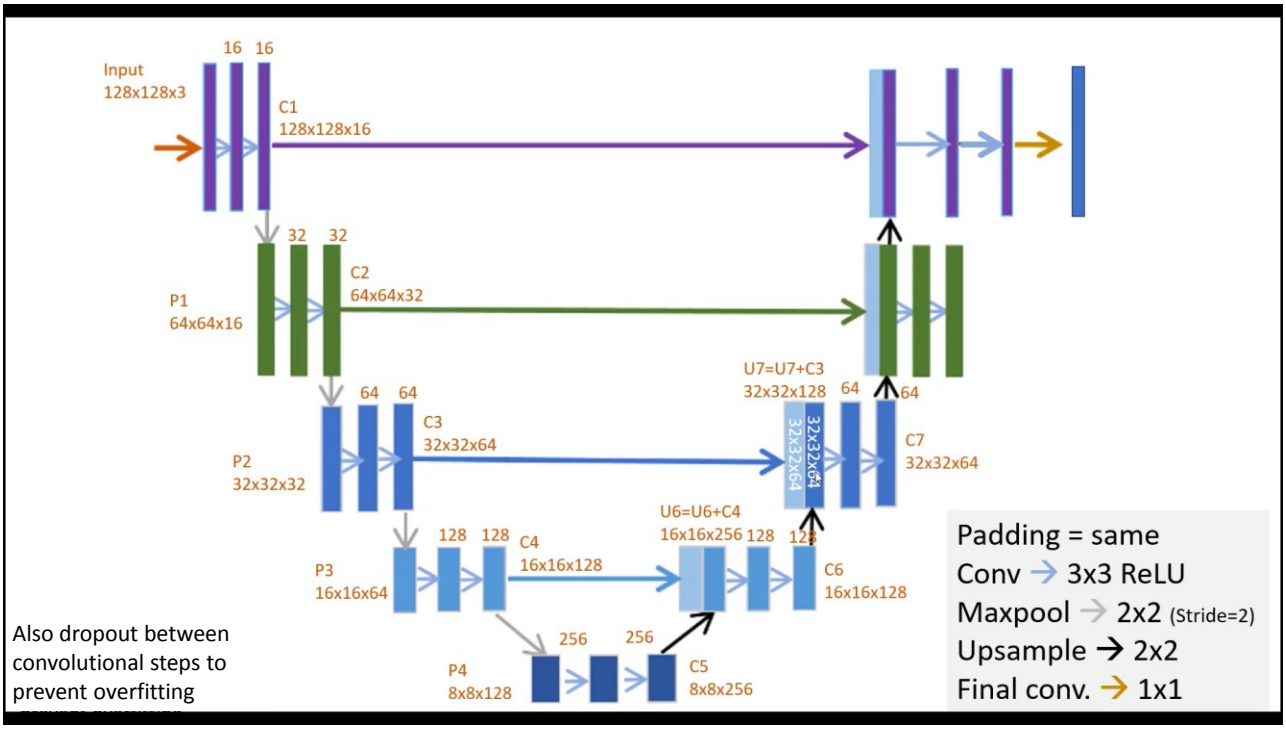
Also dropout between convolutional steps to prevent overfitting



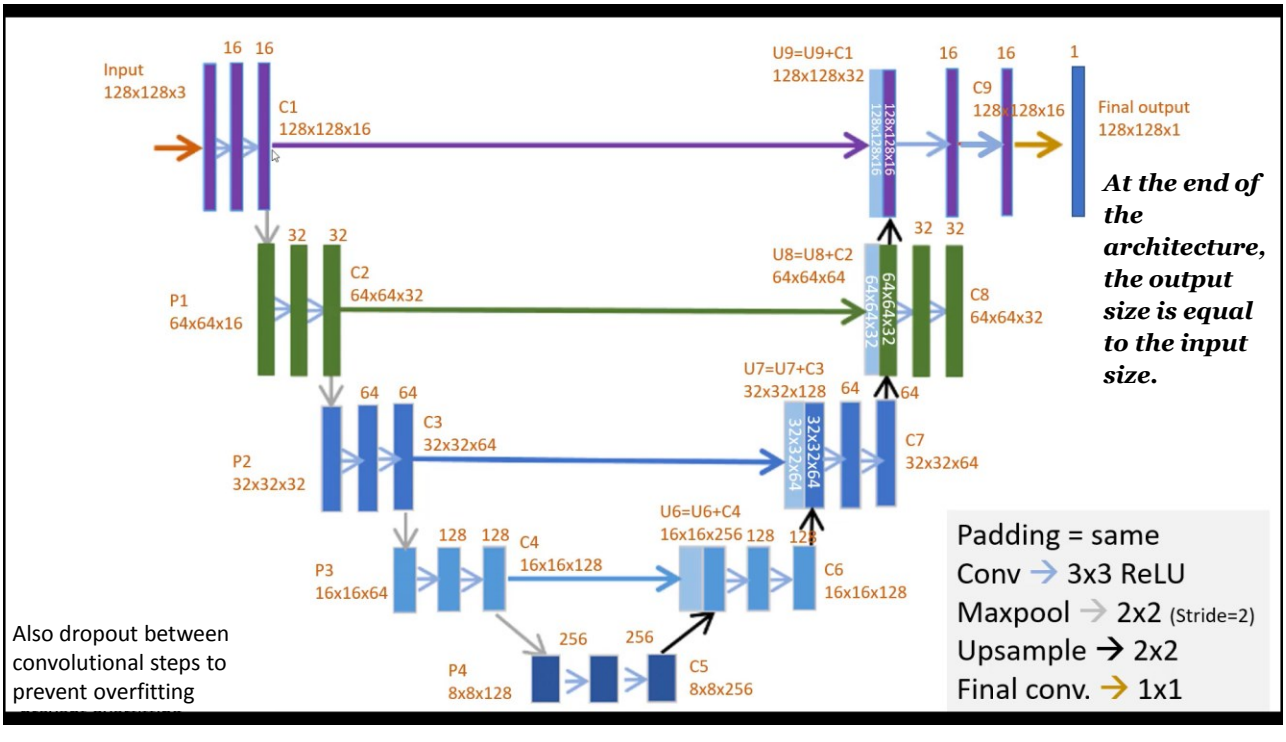
Also dropout between convolutional steps to prevent overfitting


Input
128x128x3
16 16
C1
128x128x16
32 32
C2
64x64x32
P1
64x64x16
64 64
C3
32x32x64
P2
32x32x32
U6=U6+C4
16x16x256
128 128
C4
16x16x128
P3
16x16x64
C6
16x16x128
256 256
C5
8x8x256
P4
8x8x128
Padding = same
Conv → 3x3 ReLU
Maxpool → 2x2 (Stride=2)
Upsample → 2x2
Final conv. → 1x1
Also dropout between convolutional steps to prevent overfitting


Input
128x128x3
16 16
C1
128x128x16
32 32
C2
64x64x32
P1
64x64x16
64 64
C3
32x32x64
P2
32x32x32
U7=U7+C3
32x32x128
32x32x64
32x32x64
U6=U6+C4
16x16x256
128 128
C4
16x16x128
P3
16x16x64
C6
16x16x128
256 256
C5
8x8x256
P4
8x8x128
Padding = same
Conv → 3x3 ReLU
Maxpool → 2x2 (Stride=2)
Upsample → 2x2
Final conv. → 1x1
Also dropout between convolutional steps to prevent overfitting

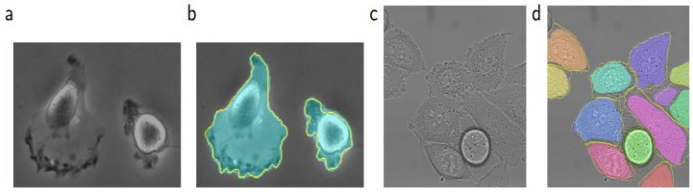


Also dropout between convolutional steps to prevent overfitting



Also dropout between convolutional steps to prevent overfitting

Result on the ISBI cell tracking challenge



(a) Image from PhC-U373 dataset

(b) Segmentation result (cyan mask) with manual ground truth (yellow border)

(c) Image from d DIC-HeLa dataset

(d) Segmentation result (random colored masks) with manual ground truth (yellow border)

Segmentation results (IOU) on the ISBI cell tracking challenge 2015

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

Resources

- http://bit.ly/dlcv2017
- Related Lecture from CS231n @ Stanford. http://cs231n.stanford.edu/

Thank You