Introduction to Semantic Segmentation using Convolutional Neural Networks

RADOVIC MILOS, EVERSEEN
milos.radovic@everseen.com

Outline

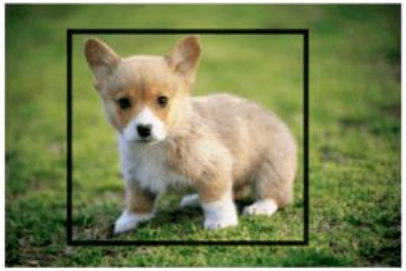
Introduction	Common computer vision tasks
	Semantic segmentation - problem definition
	Semantic segmentation vs instance segmentation
	Applications
How does it work?	Representing the semantic segmentation task
	Constructing an architecture
	Resolving computational burden
	Methods for upsampling
	Transposed convolution
	Defining a loss function
Network architectures	Fully Convolutional Networks for Semantic Segmentation (FCN)
	U-net
	The One Hundred Layers Tiramisu: Fully Convolutional DenseNets (FC-DenseNet) for Semantic Segmentation

Common computer vision tasks

Classification

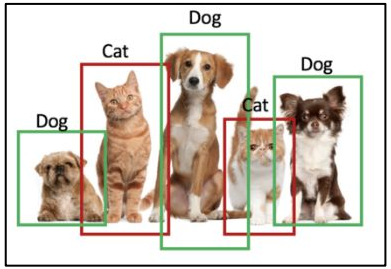
In classification we predict the class of an image.

Classification with localization



Here we predict the class label of an image as well as position of a bounding box surrounding the object.

Detection



Object detection involves localization of multiple objects (that doesn't have to belong to the same class).

Segmentation

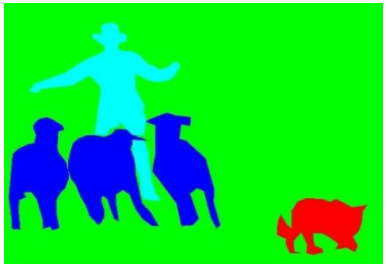


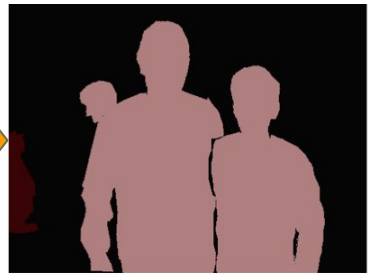
Image segmentation is the process of partitioning an image into multiple segments.

Semantic segmentation - problem definition

- The goal of semantic segmentation of an image is to label **each and every pixel** of an image with a corresponding class of what is being represented.
- Because we're predicting for every pixel in the image, this task is commonly referred to as **dense prediction**.

Input

Output



Input

Output

Semantic segmentation vs instance segmentation

- Semantic segmentation **does not separate instances of the same class**. It only predicts the category of each pixel.
- Instance segmentation is another approach for segmentation which does distinguish between separate objects of the same class (an example would be Mask R-CNN[1]).

Semantic segmentation

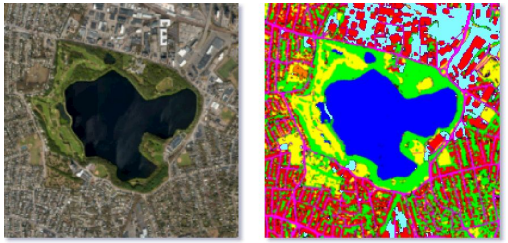
Instance segmentation

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, Mask R-CNN, CVPR 2017.

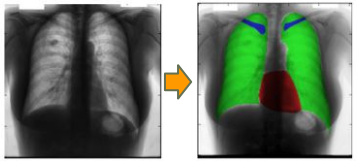
Applications

- Autonomous vehicles

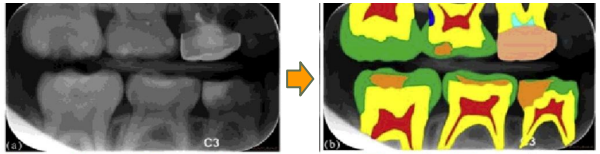
- Satellite (Or Aerial) Image Processing



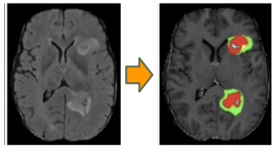
- Medicine



Organ segmentation



Substructure segmentation



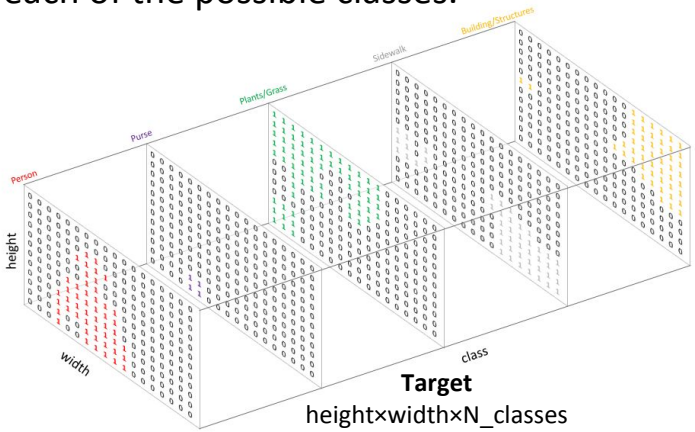
Lesion segmentation

- Fashion industry, scene understanding, etc...

Representing the semantic segmentation task

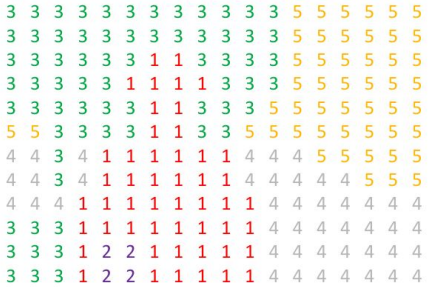
- The goal of semantic segmentation is to take an image as **input** and output a segmentation map where each pixel contains a class label represented as an integer.
- **Target** can be created by one-hot encoding the class labels - essentially creating an output channel for each of the possible classes.

Input
height×width×3 (RGB color image)
height×width×1 (grayscale image)



Target
height×width×N_classes

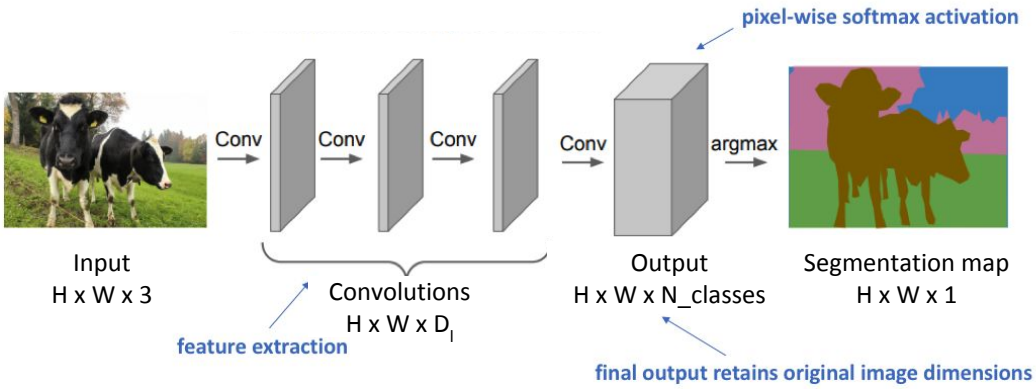
argmax



Segmentation map
height×width×1

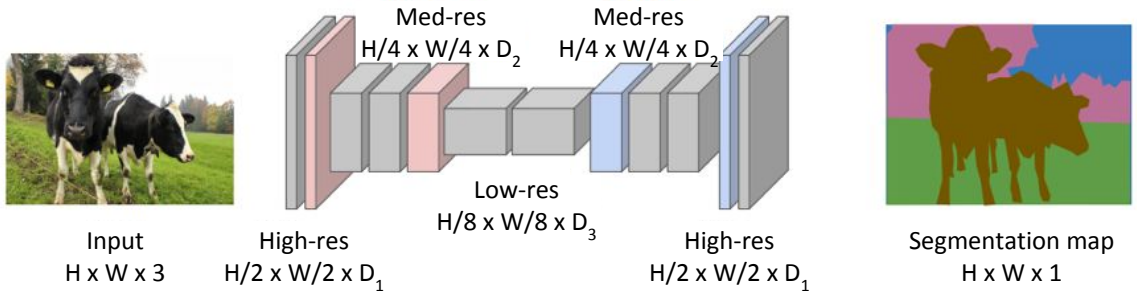
Constructing an architecture

- A naive approach: simply stack a number of convolutional layers (with same padding to preserve dimensions) and output a final segmentation map.
- This directly learns a mapping from the input image to its corresponding segmentation through the successive transformation of feature mappings; however, it's quite **computationally expensive** to preserve the full resolution throughout the network.



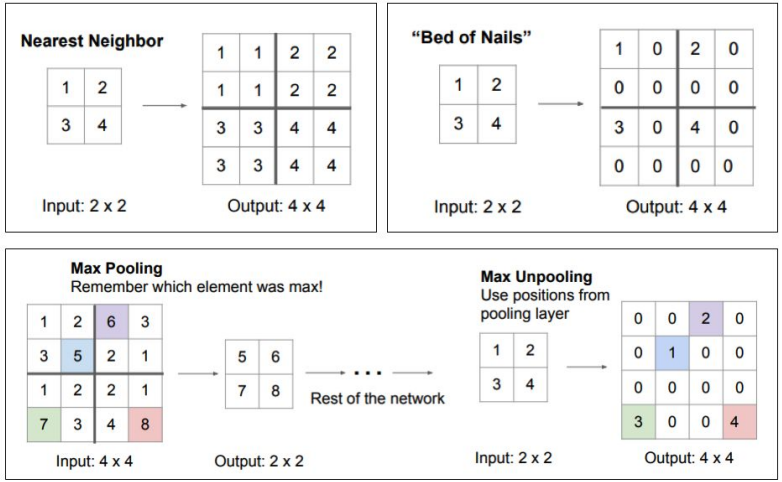
Resolving computational burden

- In order to maintain expressiveness, we typically need to reduce height and width of feature maps as we get deeper in the network. This is fine for classification but **not for semantic segmentation**!
- Resolution: **encoder/decoder** structure where we **downsample** the spatial resolution of the input, developing lower-resolution feature mappings which are learned to be highly efficient at discriminating between classes, and the **upsample** the feature representations into a full-resolution segmentation map.



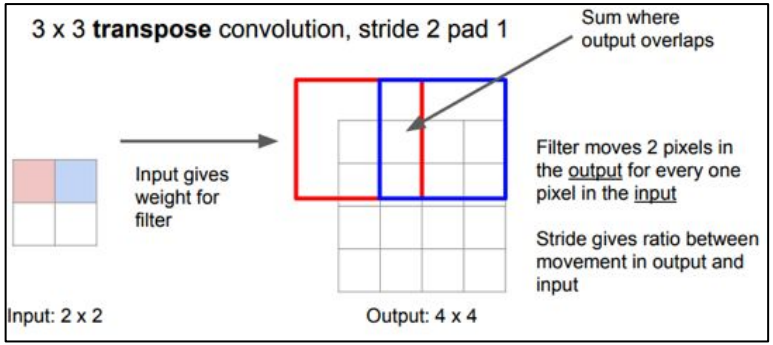
Methods for upsampling

- Whereas pooling operations downsample the resolution by summarizing a local area with a single value, "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.

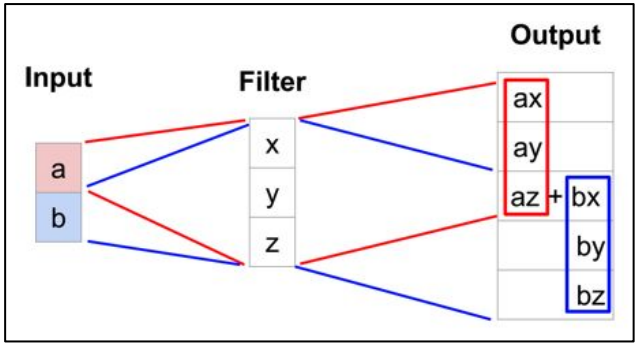


Transposed convolution

- Transposed convolution is by far the most popular approach as it allows us to develop a *learned upsampling*.
- In transposed convolution, we take a single value from the low-resolution feature map and multiply all of the weights in our filter by this value, projecting those weighted values into the output feature map.



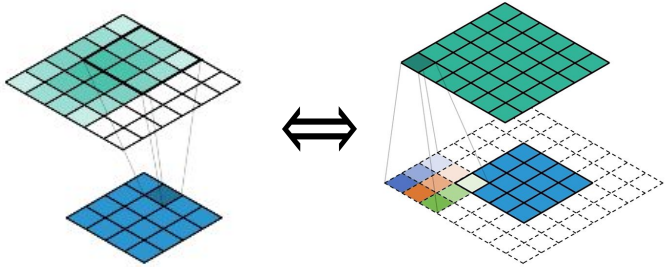
Transposed convolution - 2d example



Transposed convolution - 1d example

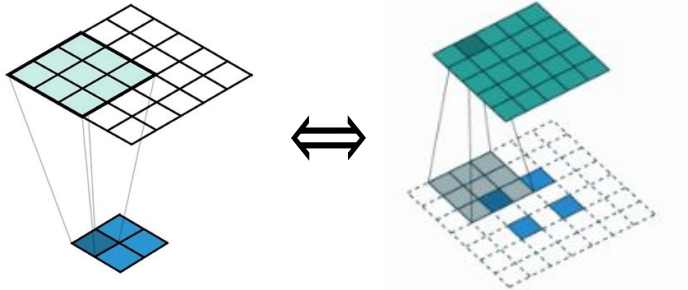
Transposed convolution

- Transposed convolution can be represented as a convolution with some modification of the input.



Conv2DTranspose with 3x3 kernel and no padding applied to a 4x4 input to give a 6x6 output.

Conv2D with stride 1 and 2x2 padding

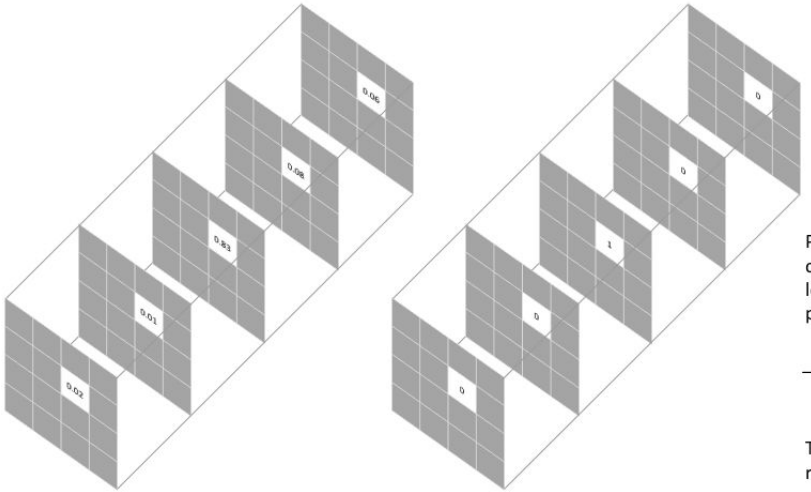


A Conv2DTranspose with 3x3 kernel, stride of 2x2 and no padding applied to a 2x2 input to give a 5x5 output.

Conv2D with stride 1 and 2x2 padding

Defining a loss function

- The most commonly used loss function for the task of image segmentation is a **pixel-wise cross entropy loss**.
- Problem: Model is biased towards the most prevalent class.



Prediction for a selected pixel

Target for the corresponding pixel

Pixel-wise loss is calculated as the log loss, summed over all possible classes

$$-\sum_{classes} y_{true} \log\left(y_{pred}\right)$$

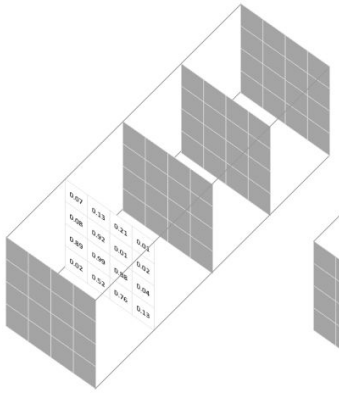
This scoring is repeated over all **pixels** and averaged

Soft Dice loss

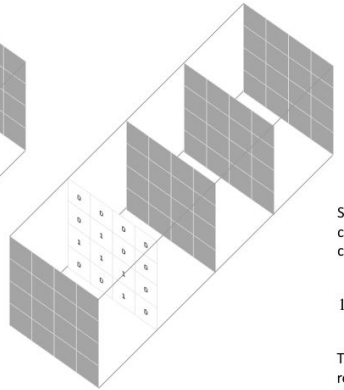
- Dice coefficient is essentially a measure of overlap between two samples.
- The soft Dice loss is normalized according to the size of the target mask so it does not struggle learning from classes with lesser spatial representation in an image.

$$Dice = \frac{2|A \cap B|}{|A| + |B|}$$

$$L = 1 - Dice$$



Prediction for a selected class



Target for the corresponding class

Soft Dice coefficient is calculated for each class mask

$$1 - \frac{2\sum_{pixels} y_{true} y_{pred}}{\sum_{pixels} y_{true}^2 + \sum_{pixels} y_{pred}^2}$$

This scoring is repeated over all **classes** and averaged

$$|A \cap B| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{element-wise multiply}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.41$$

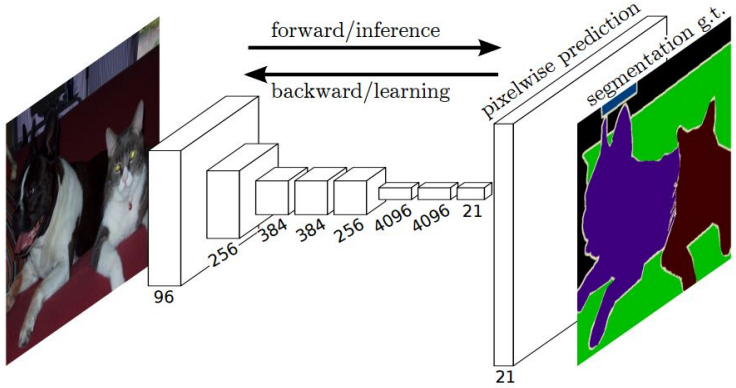
prediction target

$$|A| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix}^{2\ (optional)} \xrightarrow{\text{sum}} 7.82$$

$$|B| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{2\ (optional)} \xrightarrow{\text{sum}} 8$$

Fully Convolutional Networks for Semantic Segmentation (FCN)

- In FCN paper[2] authors utilized classification networks to serve as the **encoder** module of the network, appending a **decoder** module to upsample the coarse feature maps into a full-resolution segmentation map.
- Adding layers and a spatial loss (pixel-wise cross entropy loss) produces an efficient architecture for end-to-end dense learning.



The FCN end-to-end dense prediction pipeline

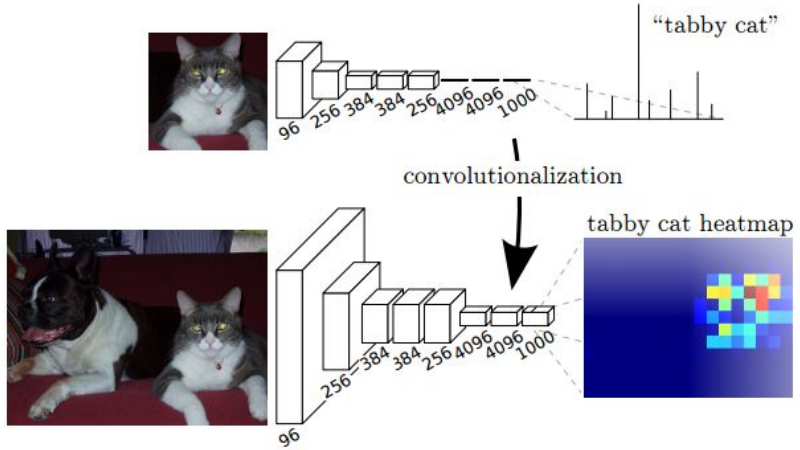
[2] Jonathan Long, Evan Shelhamer, Trevor Darrell, Fully Convolutional Models for Semantic Segmentation, CVPR 2015.

FCN: From classifier to dense prediction

- **Encoder**: AlexNet, VGG, and GoogLeNet classification networks.
- **Decoder**: Transposed convolution layers.

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M

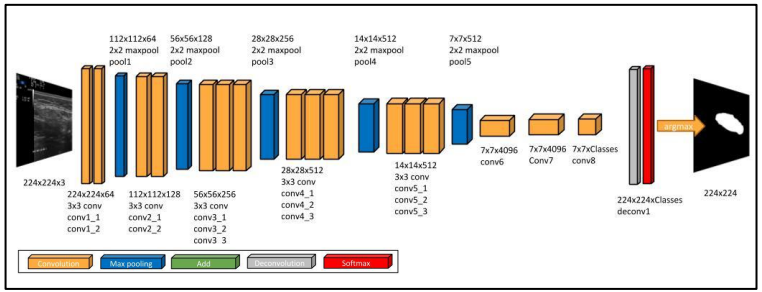
Results on the on the validation set of PASCAL VOC 2011[3]



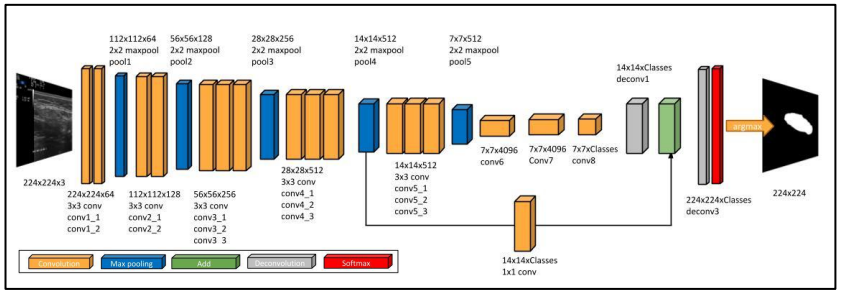
Transforming fully connected layers into convolution layers enables a classification net to output a heatmap

[3] http://host.robots.ox.ac.uk/pascal/VOC/voc2011/index.html

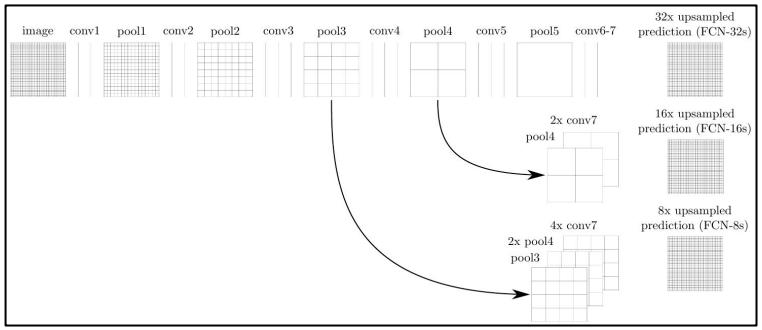
FCN: Combining what and where



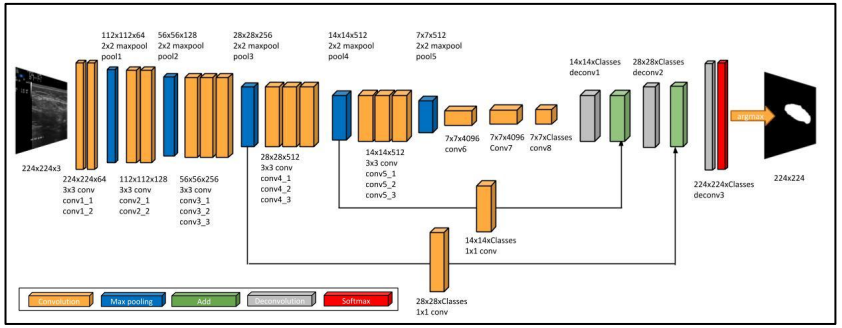
FCN-32s



FCN-16s



FCN architectures - Image from the original paper

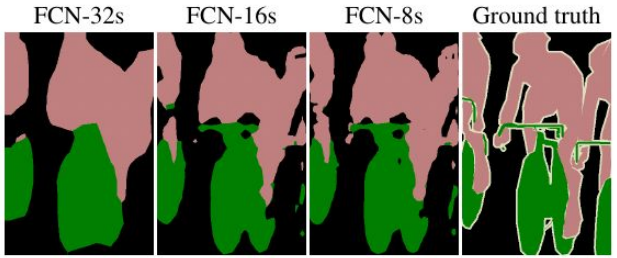


FCN-8s

FCN: results

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

Comparison of skip FCNs on a PASCAL VOC 2011 dataset.



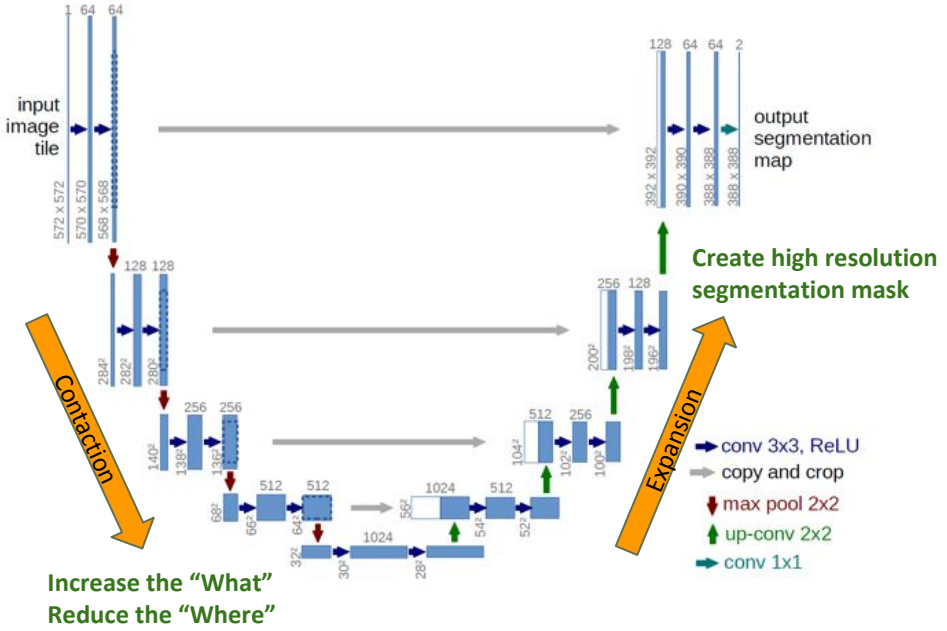
Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [12]	47.9	-	-
SDS [17]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	~ **175 ms**

Results on the PASCAL VOC 2011 and 2012 test sets.

U-net

- Initially developed for the segmentation of biomedical images.
- The U-net[4] architecture consists of a **contracting path** to capture context and a symmetric **expanding path** that enables precise localization.
- Won the *segmenting and tracking moving cells* challenge at ISBI 2015.



[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015.
[5] ISBI 2015 : International Symposium on Biomedical Imaging, https://biomedicalimaging.org/2015/.

U-net: loss function

- Authors used **weighted** pixel-wise cross entropy loss which force the network to learn the border pixels.

$$p_k(x) = exp(a_k(x)) / \sum_{k'=1}^{K} exp(a_{k'}(x))$$

$$E = \sum_{x \in \Omega} w(x) log(p_{l(x)}(x))$$

$$w(x) = w_c(x) + w_0 \cdot exp\left(- \frac{(d_1(x) + d_2(x))^2}{2\sigma^2} \right)$$

$a_k(x)$ - activation in feature channel *k* at the pixel position *x*

K - number of classes

$l(x)$ - ground truth label at position x

$w(x)$ - total weight at the pixel position *x*

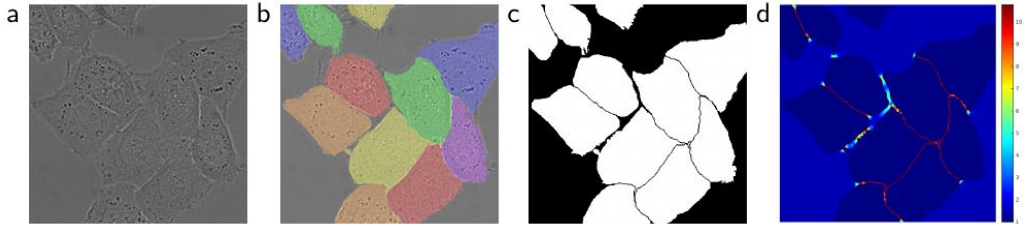
$w_c(x)$ - weight value to balance the class frequencies

d_1 - distance to the border of the nearest cell

d_2 - distance to the border of the second nearest cell

$w_0 = 10$

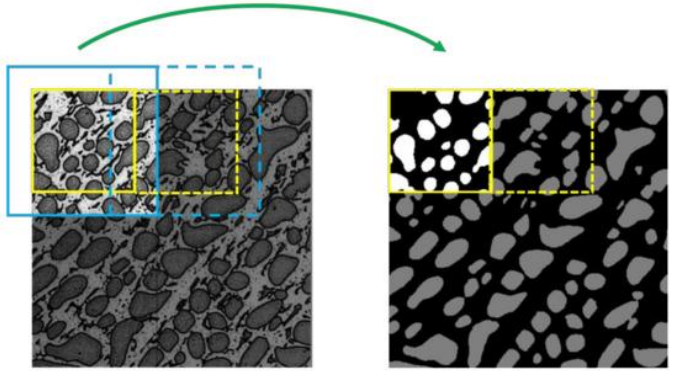
$\sigma \approx 5$



(a) raw image (b) overlay with ground truth segmentation (c) mask for semantic segmentation (d) weight map

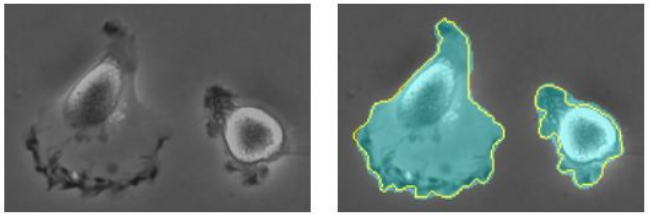
U-net: overlap-tile strategy for seamless segmentation

- Overlap-tile strategy allows the seamless segmentation of arbitrarily large images by splitting input images into tiles (this may help us to overcome GPU memory limitations).
- To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image.

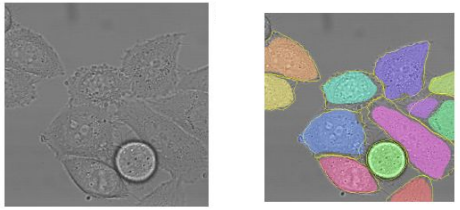


Result of segmentation within the yellow lines requires image data within the blue lines as input. Solid lines refer to the first segmentation and dashed line refers to the second segmentation. By repeating this process, an entire segmentation mask will be obtained for each slice.

U-net: results



Part of an input image of the “PhC-U373” data set (left). Segmentation result (cyan mask) with manual ground truth (yellow border) - right.



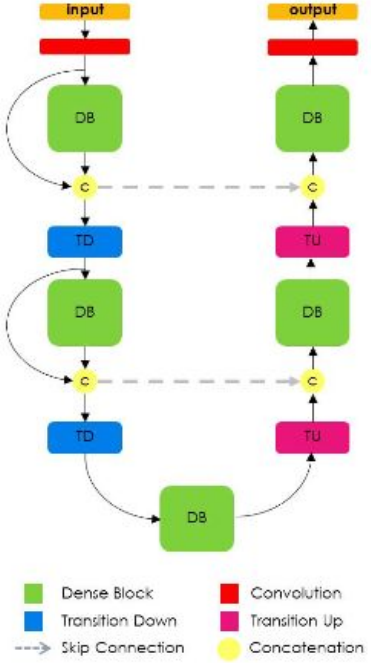
Input image of the “DIC-HeLa” data set (left). Segmentation result (random colored masks) with manual ground truth (yellow border) - right.

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**

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

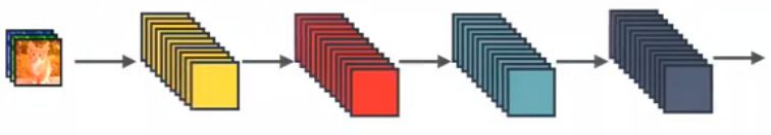
The One Hundred Layers Tiramisu: Fully Convolutional DenseNets (FC-DenseNet) for Semantic Segmentation

- In this paper[6] authors utilized ideas from the DenseNets[7] to deal with the problem of semantic segmentation.
- The network is composed of a **downsampling path** responsible for extracting coarse semantic features, (consisting from convolution, transition down and dense blocks) and an **upsampling path** trained to recover the input image resolution (consisting from convolution, transition up and dense blocks).

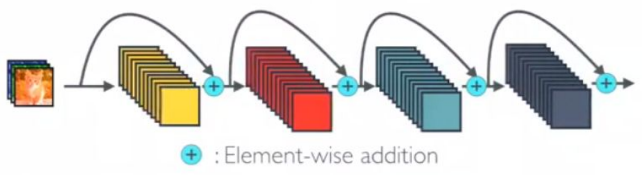


[6] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, Yoshua Bengio, The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation, CVPR 2017
[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, Densely Connected Convolutional Networks, CVPR 2017

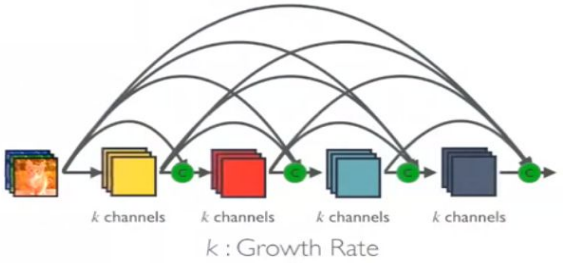
FC-DenseNet: from standard convolution to dense block



Standard ConvNet Concept



ResNet Concept



Dense Block in DenseNet with Growth Rate k Concept

FC-DenseNet: Dense block

Let x_l be the output of the l^{th} layer.

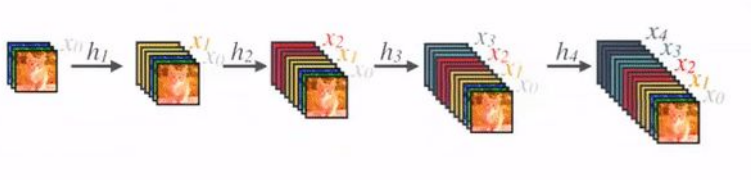
In a standard CNN, x_l is computed by applying a non-linear transformation **H** (usually convolution followed by a ReLU and often dropout) to the output of the previous layer x_{l-1}:

$$x_l = H_l(x_{l-1})$$

Pushing this idea further, DenseNets design a more sophisticated connectivity pattern that iteratively concatenates all feature outputs in a feedforward fashion. Thus, the output of the l^{th} layer is defined as:

$$x_l = H_l([x_{l-1}, x_{l-2}, \ldots, x_0])$$

where [...] represents the concatenation operation. In this case, **H** is defined as batch normalization, followed by ReLU, a convolution and dropout.



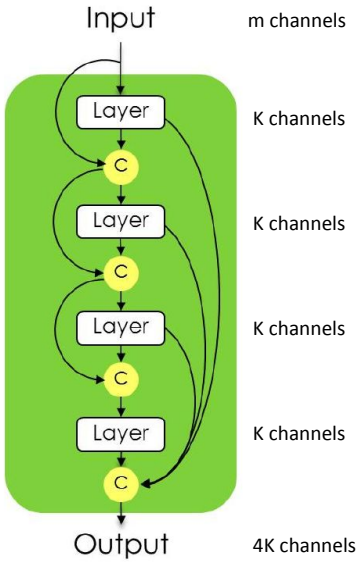
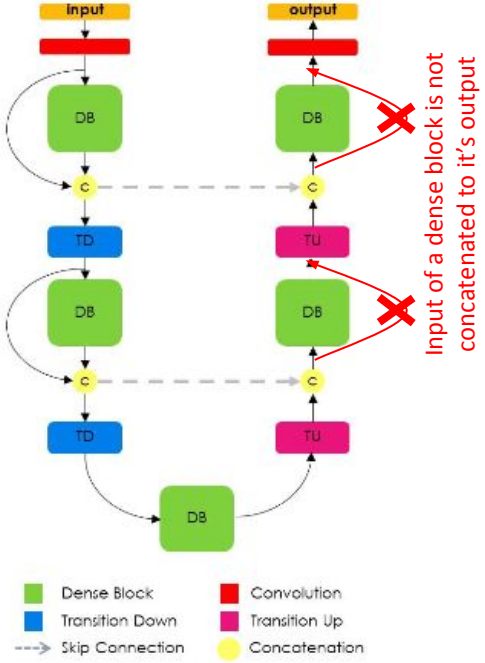


Diagram of a dense block of 4 layers.

FC-DenseNet: resolving computational burden

- **Problem**: Since the upsampling path increases the feature maps spatial resolution, the linear growth in the number of features would be too memory demanding, especially for the full resolution features in the pre-softmax layer.
- **Resolution**: In order to overcome this limitation, the input of a dense block is not concatenated with its output. Thus, the transposed convolution is applied only to the feature maps obtained by *the last dense block and not to all feature maps concatenated so far*.



FC-DenseNet: network architecture

Architecture	
Input, $m = 3$	
3×3 Convolution, $m = 48$	Downsampling path
DB (4 layers) + TD, $m = 112$	Downsampling path
DB (5 layers) + TD, $m = 192$	Downsampling path
DB (7 layers) + TD, $m = 304$	Downsampling path
DB (10 layers) + TD, $m = 464$	Downsampling path
DB (12 layers) + TD, $m = 656$	Downsampling path
Bottleneck ⇨ DB (15 layers), $m = 896$	
TU + DB (12 layers), $m = 1088$	Upsampling path
TU + DB (10 layers), $m = 816$	Upsampling path
TU + DB (7 layers), $m = 578$	Upsampling path
TU + DB (5 layers), $m = 384$	Upsampling path
TU + DB (4 layers), $m = 256$	Upsampling path
1×1 Convolution, $m = c$	Upsampling path
Softmax	

Architecture details of FC-DenseNet103 model

Layer
Batch Normalization
ReLU
3×3 Convolution
Dropout $p = 0.2$

Transition Down (TD)
Batch Normalization
ReLU
1×1 Convolution
Dropout $p = 0.2$
2×2 Max Pooling

Transition Up (TU)
3×3 Transposed Convolution $stride = 2$

Building blocks of fully convolutional DenseNets

FC-DenseNet: results

Model	Pretrained	# parameters (M)	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Cyclist	Mean IoU	Global accuracy
SegNet [1]	✓	29.5	68.7	52.0	87.0	58.5	13.4	86.2	25.3	17.9	16.0	60.5	24.8	46.4	62.5
Bayesian SegNet [15]	✓	29.5	n/a											63.1	86.9
DeconvNet [21]	✓	252	n/a											48.9	85.9
Visin et al. [36]	✓	32.3	n/a											58.8	88.7
FCN8 [20]	✓	134.5	77.8	71.0	88.7	76.1	32.7	91.2	41.7	24.4	19.9	72.7	31.0	57.0	88.0
DeepLab-LFOV [5]	✓	37.3	81.5	74.6	89.0	82.2	42.3	92.2	48.4	27.2	14.3	75.4	50.1	61.6	–
Dilation8 [37]	✓	140.8	82.6	76.2	89.0	84.0	46.9	92.2	56.3	35.8	23.4	75.3	55.5	65.3	79.0
Dilation8 + FSO [17]	✓	140.8	**84.0**	77.2	91.3	**85.6**	**49.9**	92.5	59.1	**37.6**	16.9	76.0	**57.2**	66.1	88.3
Classic Upsampling	✗	20	73.5	72.2	92.4	66.2	26.9	90.0	37.7	22.7	30.8	69.6	25.1	55.2	86.8
FC-DenseNet56 (k=12)	✗	1.5	77.6	72.0	92.4	73.2	31.8	92.8	37.9	26.2	32.6	79.9	31.1	58.9	88.9
FC-DenseNet67 (k=16)	✗	3.5	80.2	75.4	93.0	78.2	40.9	**94.7**	58.4	30.7	**38.4**	81.9	52.1	65.8	90.8
FC-DenseNet103 (k=16)	✗	9.4	83.0	**77.3**	**93.0**	77.3	43.9	94.5	**59.6**	37.1	37.8	**82.2**	50.5	**66.9**	**91.5**

Results on **CamVid**[8] dataset (11 classes, 367 training frames, 101 validation frames, 233 testing frames, 360x480 resolution). Architectures: (1) 56 layers (**FC-DenseNet56**), with 4 layers per dense block and a growth rate of 12; (2) 67 layers (**FC-DenseNet67)** with 5 layers per dense block and a growth rate of 16; (3) 103 layers (**FC-DenseNet103**) with a growth rate k = 16; (4) **Classic Upsampling**, an architecture using standard convolutions in the upsampling path instead of dense blocks.

Model	Acc.
2D models (no time)	
2D-V2V-from scratch [34]	55.7
FC-DenseNet103	**79.4**
3D models (incorporate time)	
3D-V2V-from scratch [34]	66.7
3D-V2V-pretrained [34]	76.0

Results on **Gatech**[9] dataset (63 videos for training/validation and 38 for testing with 190 frames per video on average. There are 8 classes in the dataset.)

[7] http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/
[8] http://www.cc.gatech.edu/cpl/projects/videogeometriccontext/

Thanks!

milos.radovic@everseen.com