EE 4211 Computer Vision

Lecture 11B: Semantic segmentation with deep learning models

Semester A, 2020-2021

Schedules

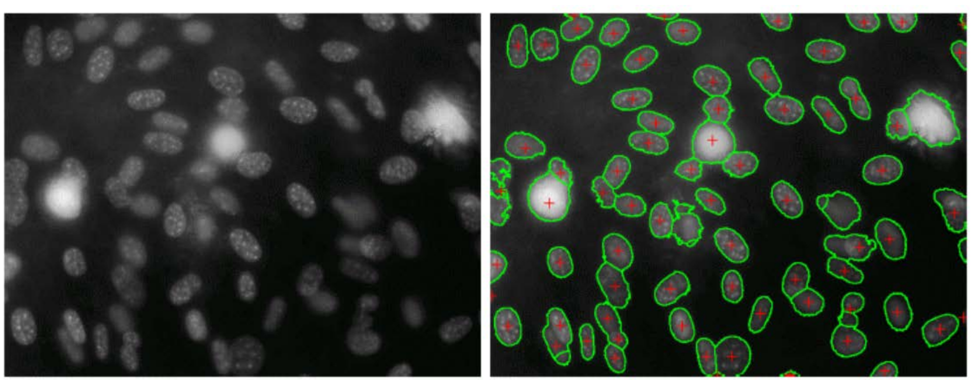
Week	Date	Topics
1	Sep. 4	Introduction/Imaging
2	Sep. 11	Image enhancement in spatial domain
3	Sep. 18	Image enhancement in frequency domain (HW1 out)
4	Sep. 25	Morphological processing
5	Oct. 2	Image restoration(HW1 due)
6	Oct. 9	Image restoration
7	Oct. 16	Midterm (no tutorials this week)
8	Oct. 23	Edge detection (HW2 out, illustrate the project)
9	Oct. 30	Image segmentation (HW2 due)
10	Nov. 6	Face recognition with PCA, LDA (tutorial on deep learning framework)
11	Nov. 13	Face recognition based on deep learning Image segmentation based on deep learning (tutorial on coding)
12	Nov. 20	Object detection with traditional methods (Quiz) Object detection based on deep learning
13	Nov. 27	Project presentation and summary
14	Dec. 4	Review and Summary

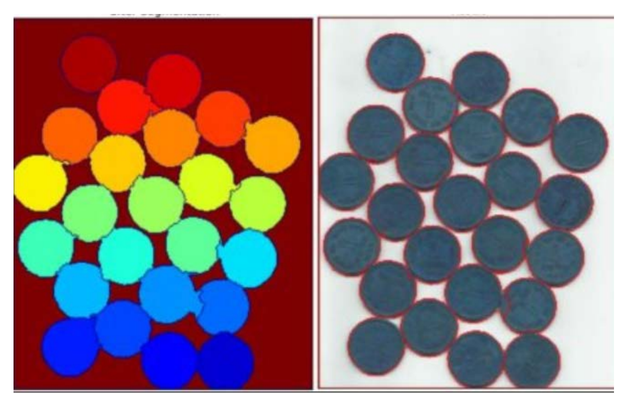
Lecture outline

- Introduction
- Different models
  - Fully Convolutional Network
  - DeconvNet, SegNet
  - U-Net
  - PSPNet
  - DeepLab v1, v2, v3, v3+
- Loss functions

Image Segmentation

- Segmentation:
  - Split/separate/subdivide an image into regions or objects
  - To facilitate recognition, understand region of interest
- Challenges of Segmentation:
  - The definition of a region/object is problem-dependent
  - One of the most difficult task in image processing
  - Accuracy determines success or failure of application





Introduction

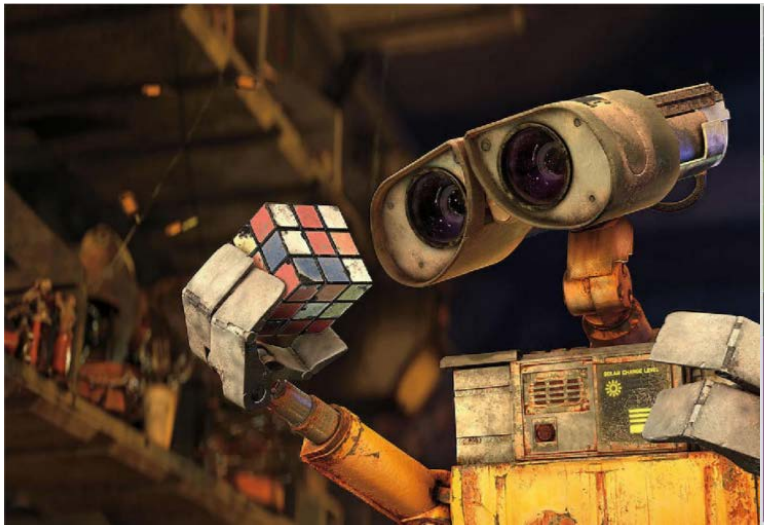
- Typically, image segmentation is applied to derive semantics (meaning) known as semantic segmentation.
  - This involves both classification and localization.
  - Localization is at pixel-level unlike object detection where bounding boxes are used.
- Instance segmentation: each instance or occurrence of an object is assigned a different label.

Semantic segmentation

Instance segmentation

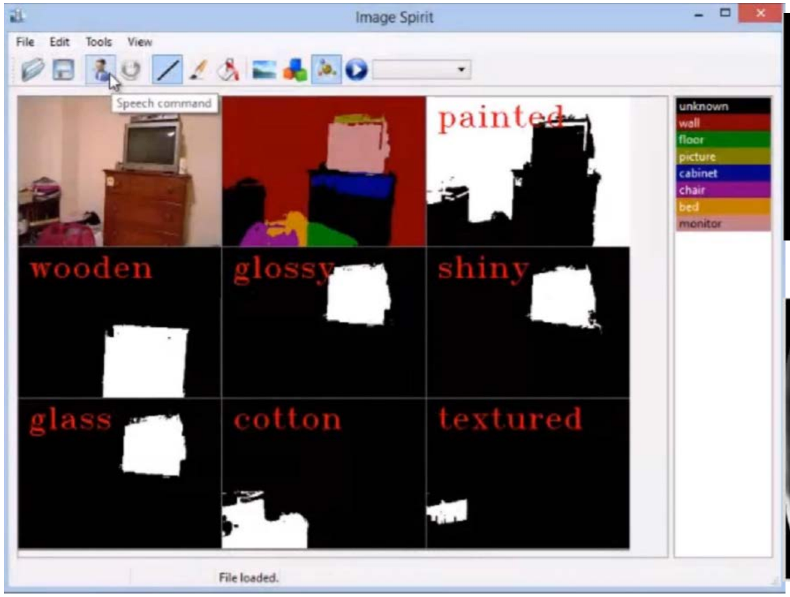
Applications

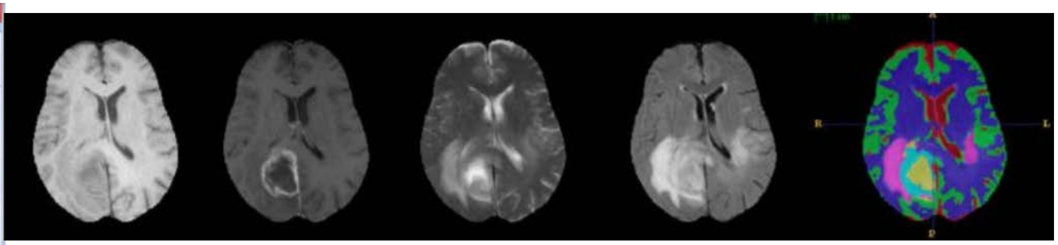
- To let robots segment objects so that they can grasp them
- Road scenes understanding; useful for autonomous navigation of cars and drones

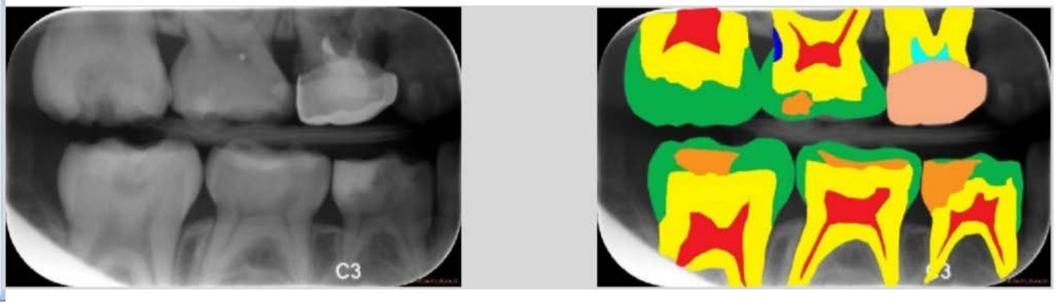



Applications

- Useful tool for editing images
- Medical purposes: e.g. segmenting tumors, dental cavities, ...





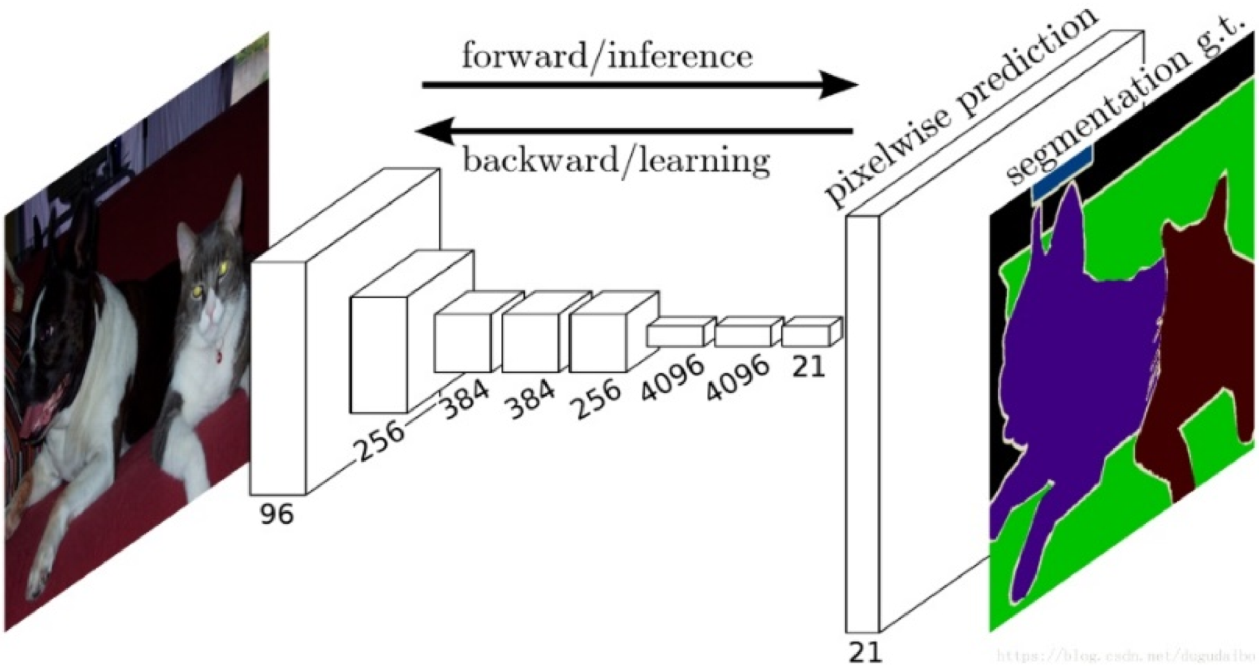


Segmentation in traditional methods

- Edge-based Segmentation
  - Finding boundary between adjacent regions
- Threshold-based Segmentation
  - Finding regions by grouping pixels of similar gray values
- Region-based Segmentation
  - Finding regions directly using growing or splitting
- Motion-based Segmentation
  - Finding regions by comparing successive frames of a video sequence to identify regions that correspond to moving objects

Segmentation with deep learning models

- Define CNN architecture
- Define a loss measuring performance (loss function)
  - In the training, loss values will be used to update the model parameters.
- Minimize the loss (optimizer)

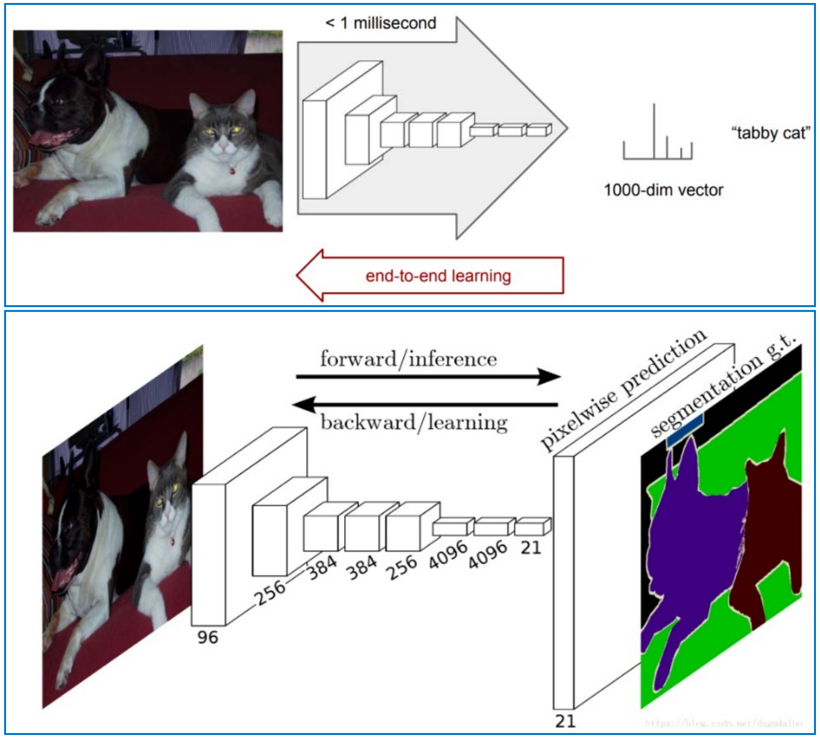


Lecture outline

- Introduction
- Different models
  - Fully Convolutional Network
  - DeconvNet, SegNet
  - U-Net
  - PSPNet
  - DeepLab v1, v2, v3, v3+
- Loss functions

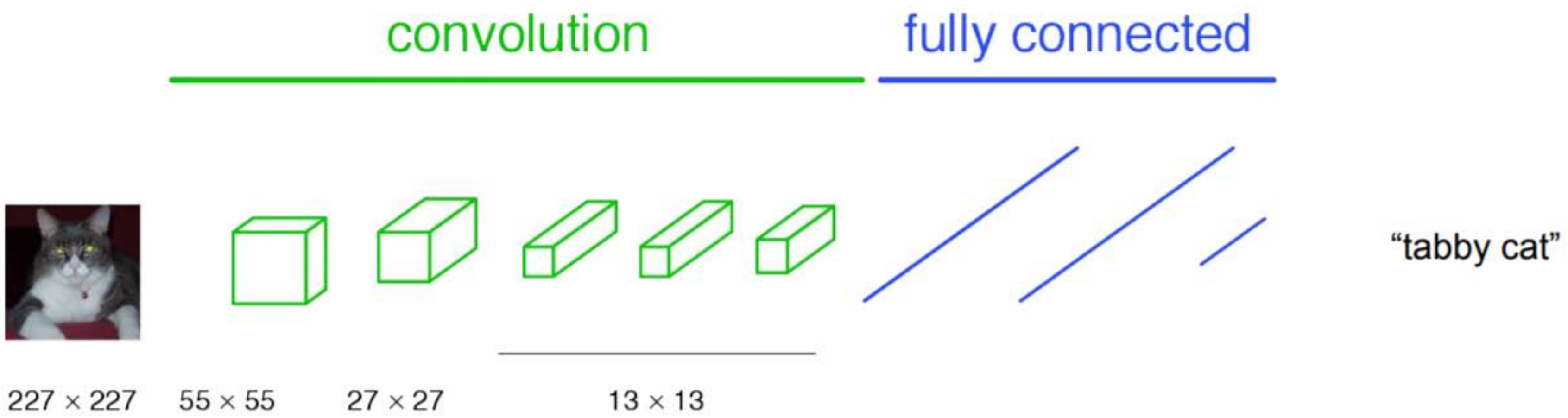
Fully Convolutional Network

- First segmentation architecture proposed using a Convolutional Neural Network (CNN).
- Key idea is to eliminate the use of fully connected layers and replace it with convolution layers.
- Hence called fully convolutional network (FCN)



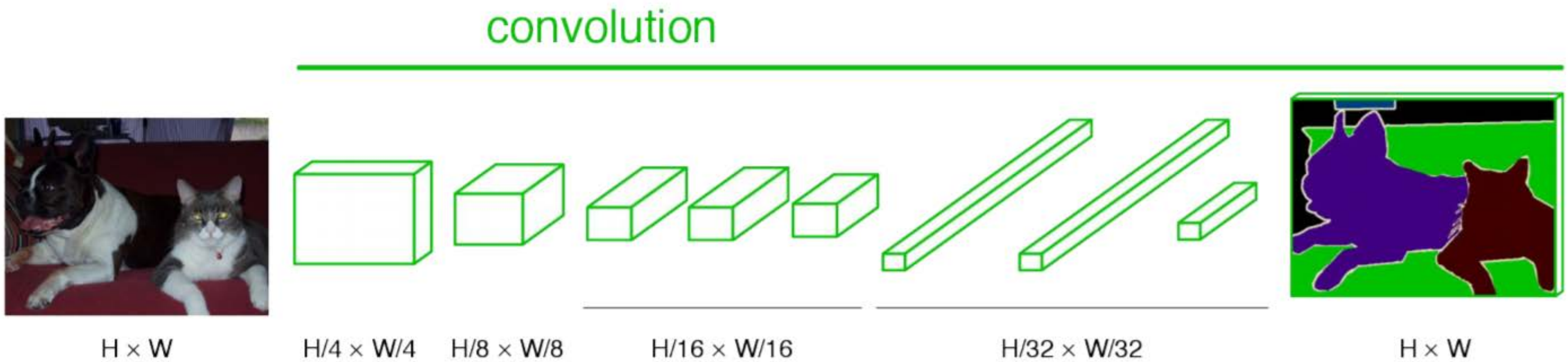
Fully Convolutional Network

- An example of a classification network.
- Convolution layers are typically followed by fully connected layers for classification.



Fully Convolutional Network

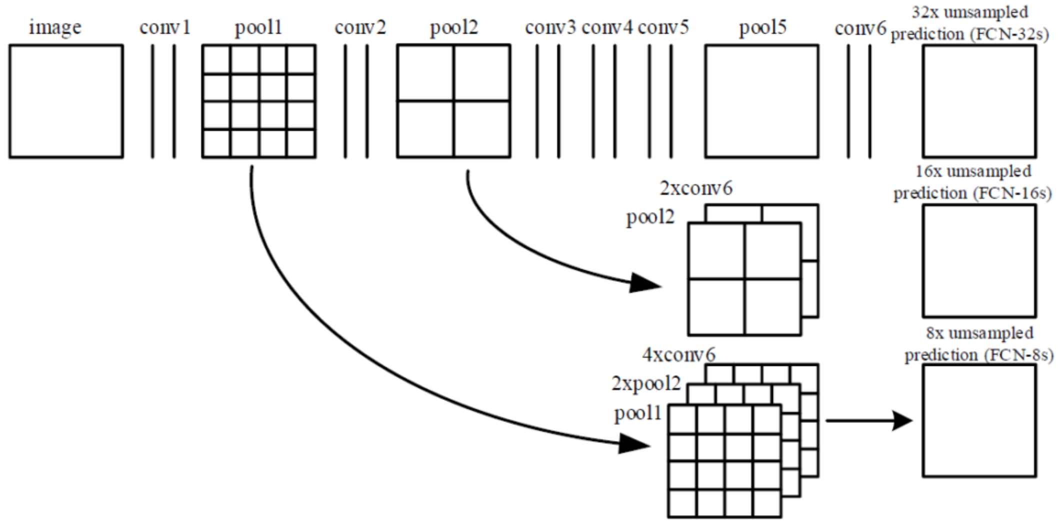
- FCN transfers knowledge from VGG16 to perform semantic segmentation.
- The fully connected layers of VGG16 is replaced by convolution layers.
- Upsample the last layer using deconvolution layer to produce output of same size as input.
- This is FCN-32s, as a stride of 32 is used by **deconvolution** kernel.



https://github.com/shelhamer/fcn.berkeleyvision.org

Fully Convolutional Network

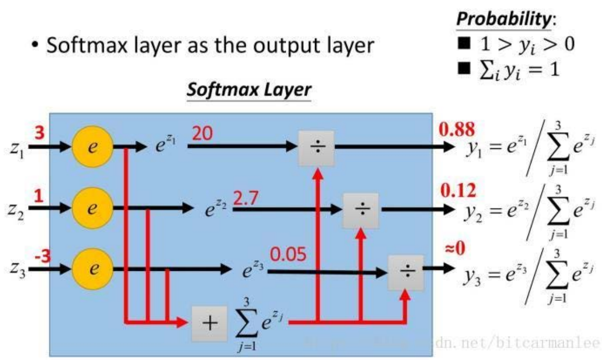
- Other variants do exist such as FCN-8s, FCN-16s
- The deconvolution at the last layer can lose a lot of resolution
- One option is adding "skip" connections from earlier higher-resolution layers. For these variants the last convolution layers are upsampled and fused with earlier pooling layers to produce finer results (via summation).

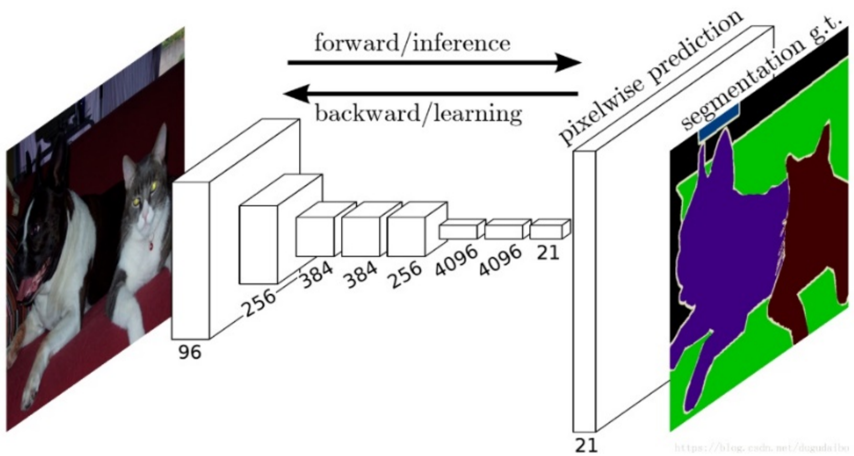


Fully Convolutional Network

- The output of the last layer is a tensor of depth k, where k is the number of classes.
- Softmax is applied such that values are stored between 0 and 1.
- Ground truth mask is stored as one-hot encoding.
- One-hot encoding is a representation of categorical values such that the active class is assigned “1” whereas the rest are “0”.

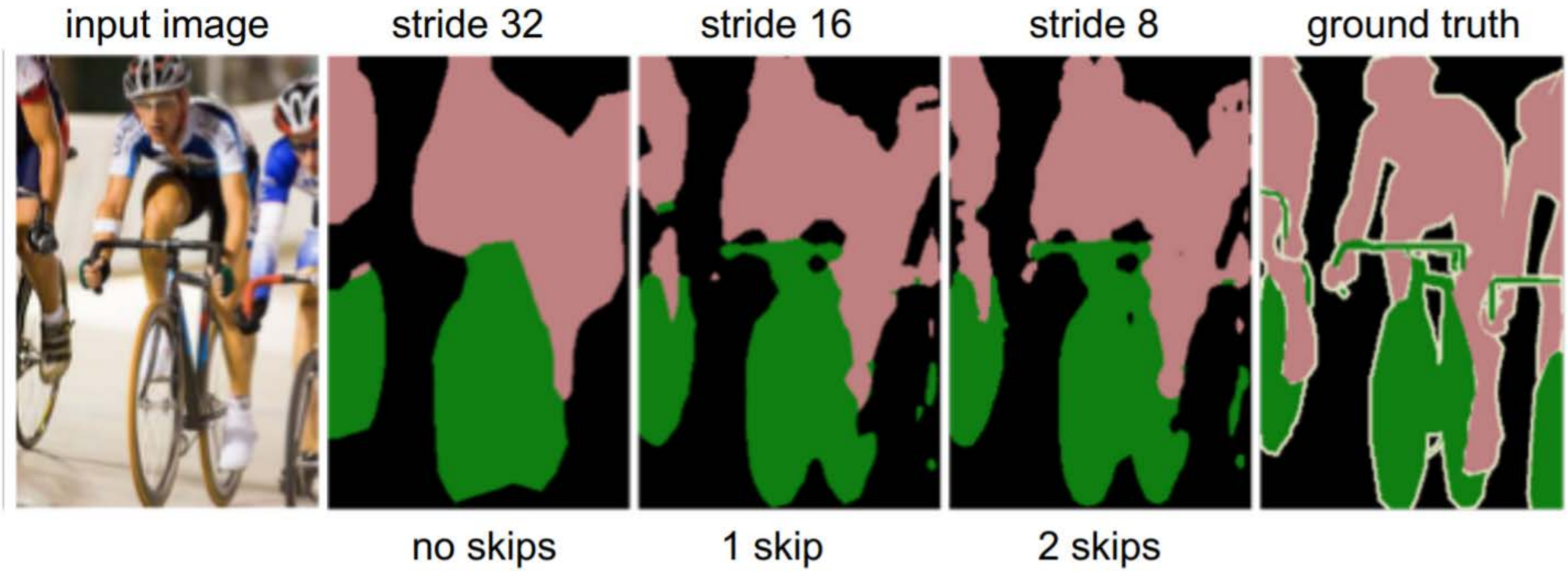
Label	Car	Person	Building
1	1	0	0
2	0	1	0
3	0	0	1





Fully Convolutional Network

- Comparison with different network baselines

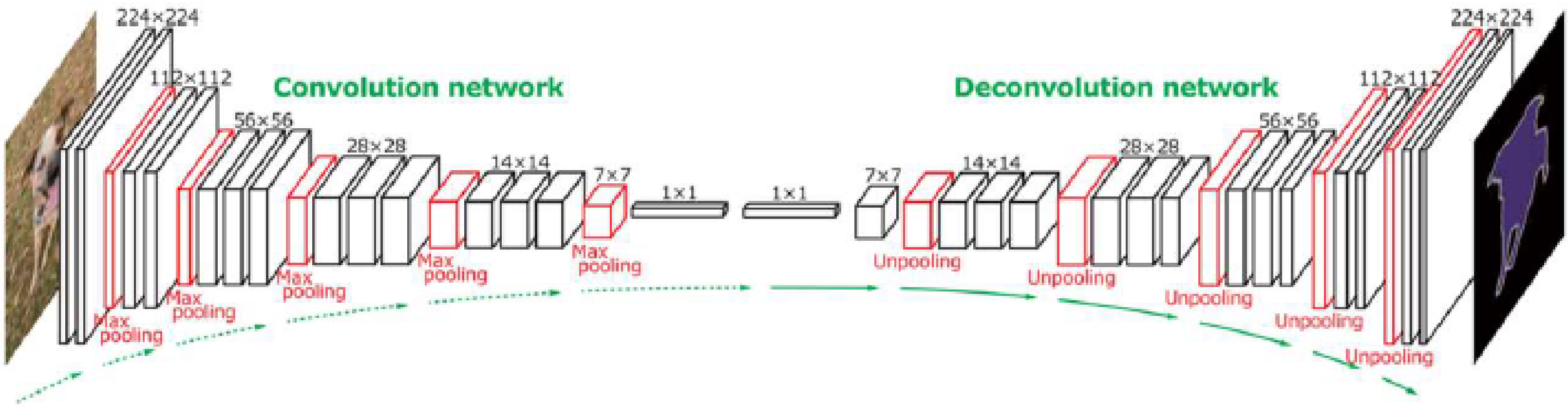


Fully Convolutional Network

- Significantly improved the state of the art in semantic segmentation.
- Poor object delineation: e.g. spatial consistency neglected.

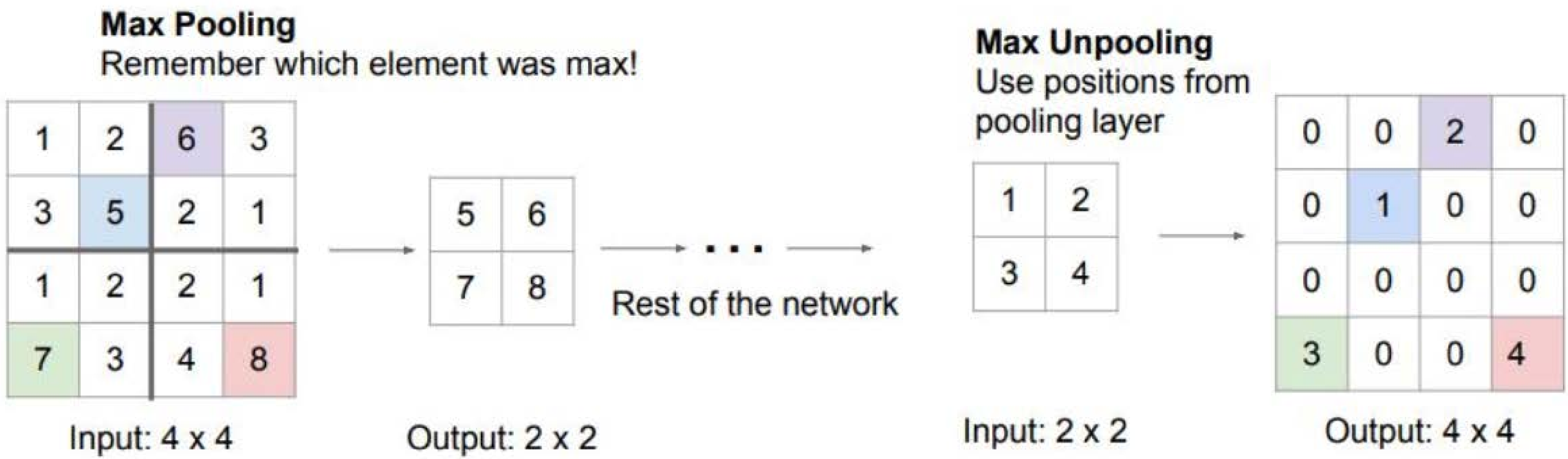
Deconvolutional Network

- Has an Encoder-Decoder (Convolution-Deconvolution) structure.
- Decoder side has a stack of unpooling layer and deconvolution layers.
- Compared to FCN, finer details are preserved in the output.



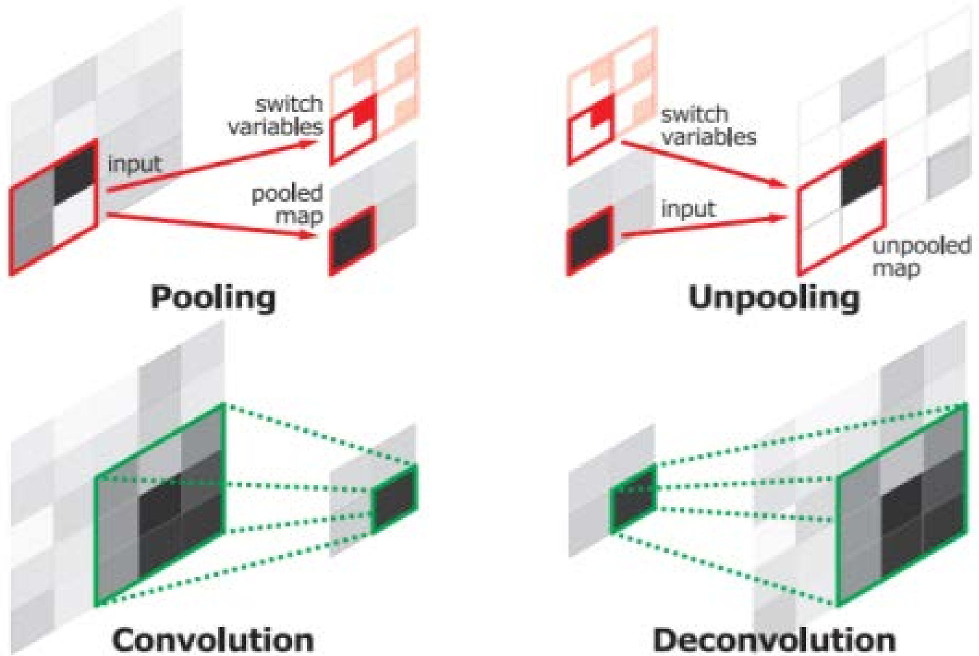
Deconvolutional Network

- Pooling operation (max): selects the max value across a window. Hence, downsamples the input.
- Unpooling operation (max):
  - Stores the indices of max value from the encoder side.
  - On the decoder side, the unpooling layer places back each output back to its pooled location.
  - Due to this operation, the output will be sparse.



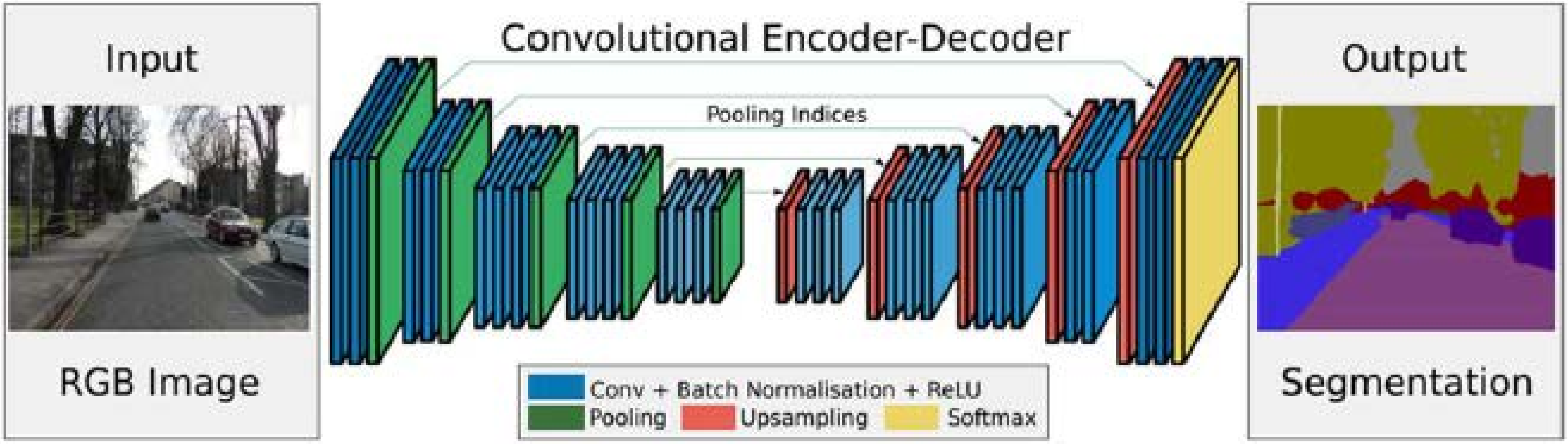
Deconvolutional Network

- The unpooling layer produces sparse output.
- To have dense predictions, deconvolution layers are applied.
  - Convolution - multiple inputs with a filter to produce single output.
  - Deconvolution: single input with a filter to produce multiple outputs.
- Note that deconvolution is called transposed convolution in practice.



SegNet

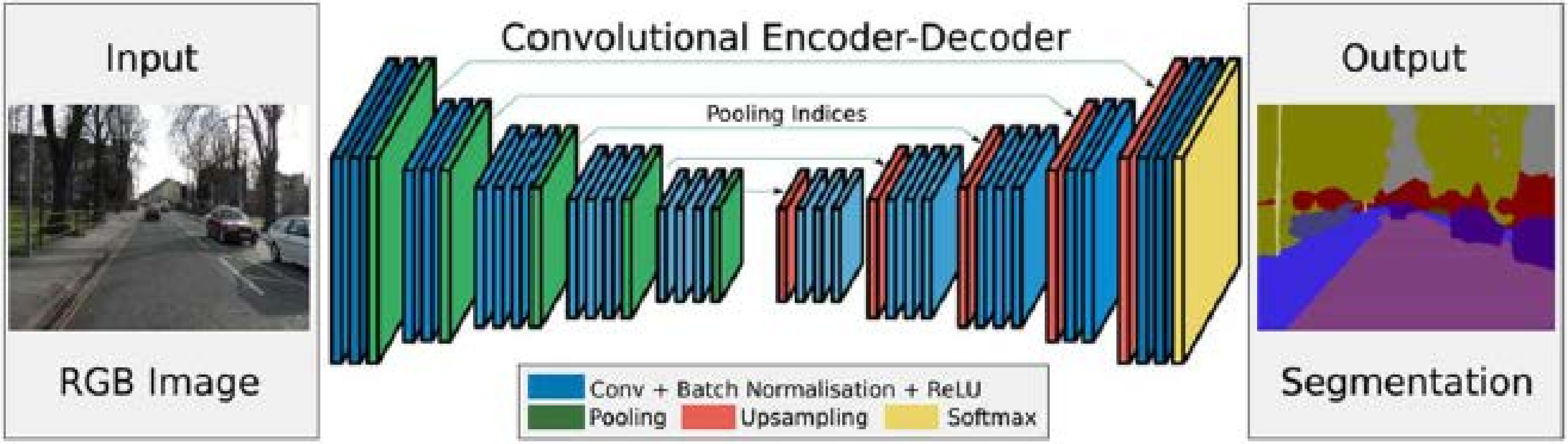
- Similar to deconvolutional network but without fully-connected layers.
- Far less parameters as no fully-connected layers are used.
- Better performance than deconvolutional network.



http://mi.eng.cam.ac.uk/projects/segnet/

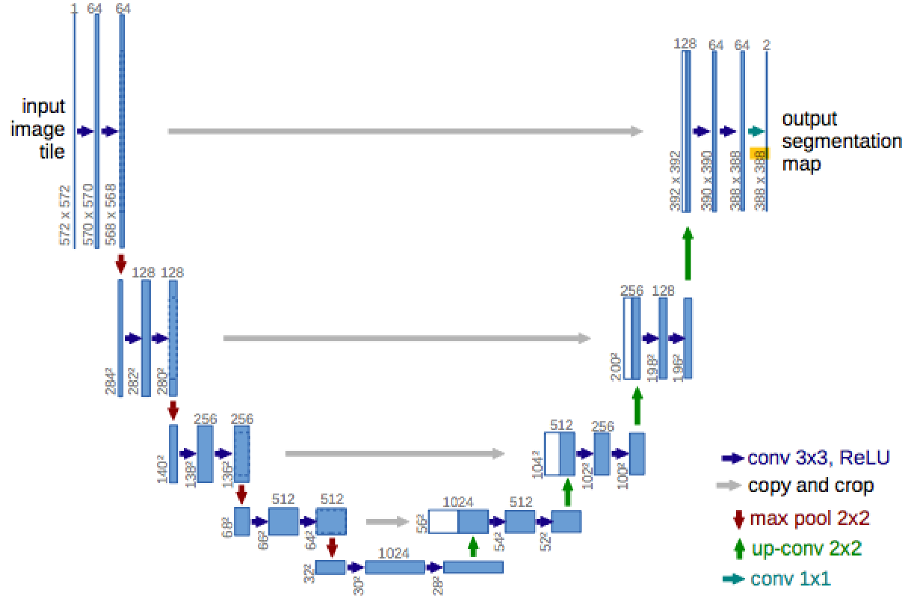
SegNet

- Each encoder: one or more convolutional layers with batch normalization and a ReLU non-linearity, followed by non-overlapping maxpooling
- Use max-pooling indices in the decoders to perform upsampling of low resolution feature maps
- Retain high frequency details and also reduce the total number of trainable parameters in the decoders
- Tend to be smooth even without a Conditional Random Field based post-processing.(a kind of post-processing model)



U-Net

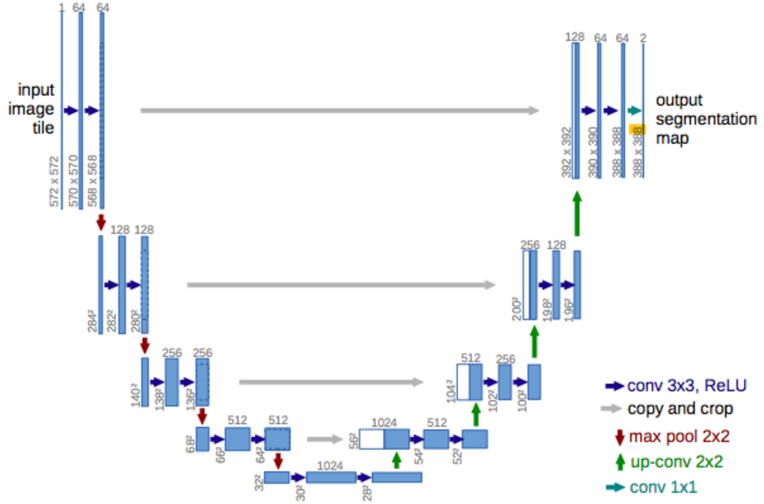
- Introduced for ISBI challenge of segmentation neuronal structures.
- Has “U” structure as the name implies.
- At every three layers, outputs are cropped.
- On the decoder side, the network concatenates the outputs from encoder and then upsamples it.
- There is also a “W” net for segmentation!



https://github.com/milesial/Pytorch-UNet

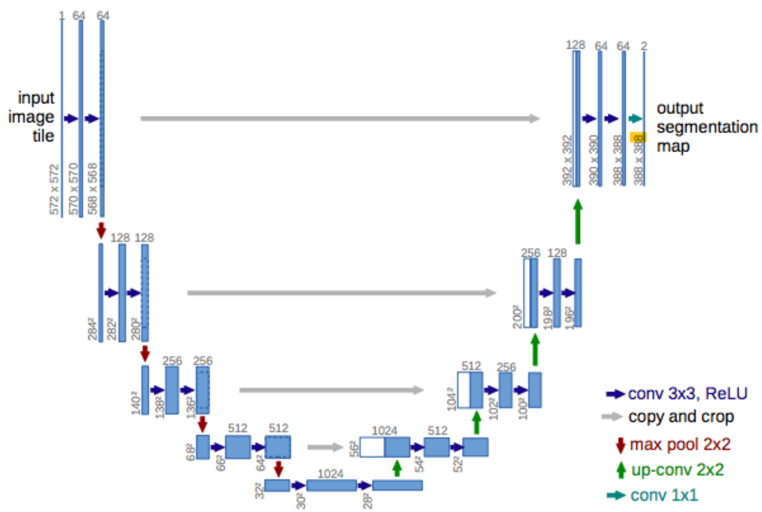
U-Net

- Contracting/downsampling path with 4 blocks
  - 3x3 Convolution Layer + activation function (with batch normalization)
  - 3x3 Convolution Layer + activation function (with batch normalization)
  - 2x2 Max Pooling
  - Note that the number of feature maps doubles at each pooling, starting with 64 feature maps for the first block, 128 for the second, and so on.
  - Purpose: capture the context of the input image in order to be able to do segmentation. This coarse contextual information will then be transfered to the upsampling path by means of skip connections.



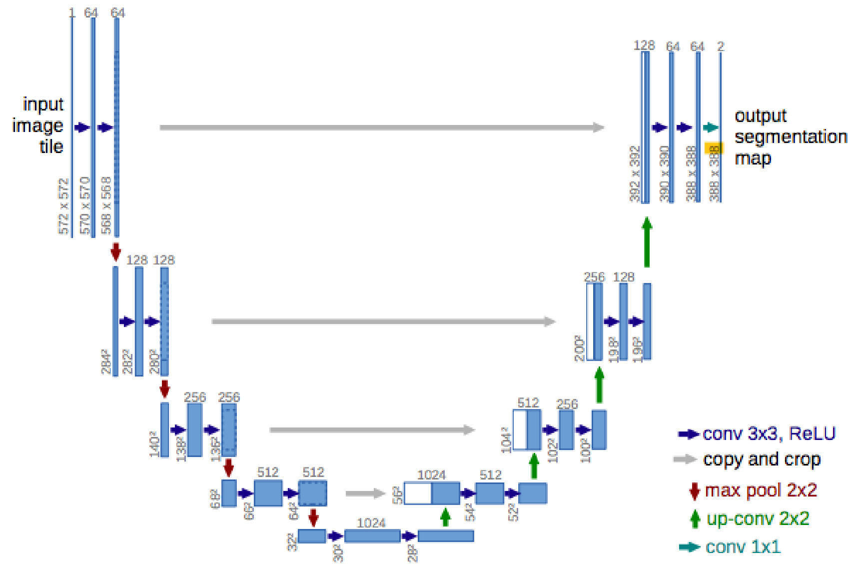
U-Net

- Expanding/upsampling path with 4 blocks.
  - Deconvolution layer with stride 2
  - Concatenation with the corresponding cropped feature map from the contracting path
  - 3x3 Convolution layer + activation function (with batch normalization)
  - 3x3 Convolution layer + activation function (with batch normalization)
- Purpose: enable precise localization combined with contextual information from the contracting path.



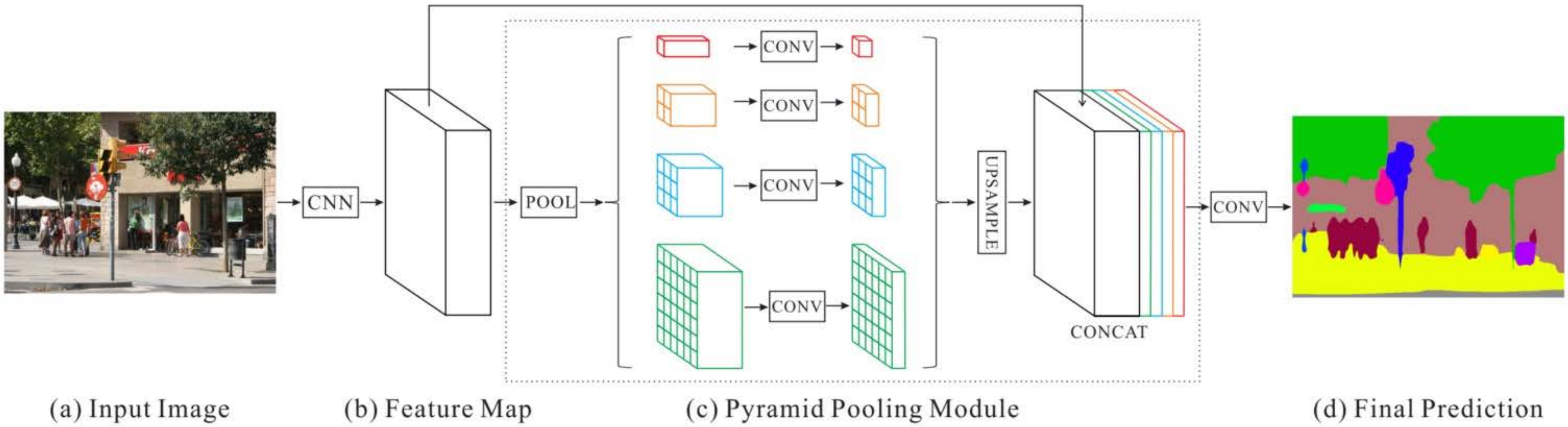
Unet

- Bottleneck
  - This part of the network is between the contracting and expanding paths. The bottleneck is built from simply 2 convolutional layers (with batch normalization), with dropout.
- The U-Net combines the location information from the downsampling path with the contextual information in the upsampling path to finally obtain a general information combining localization and context



Pyramid Scene Parsing Network (PSPNet)

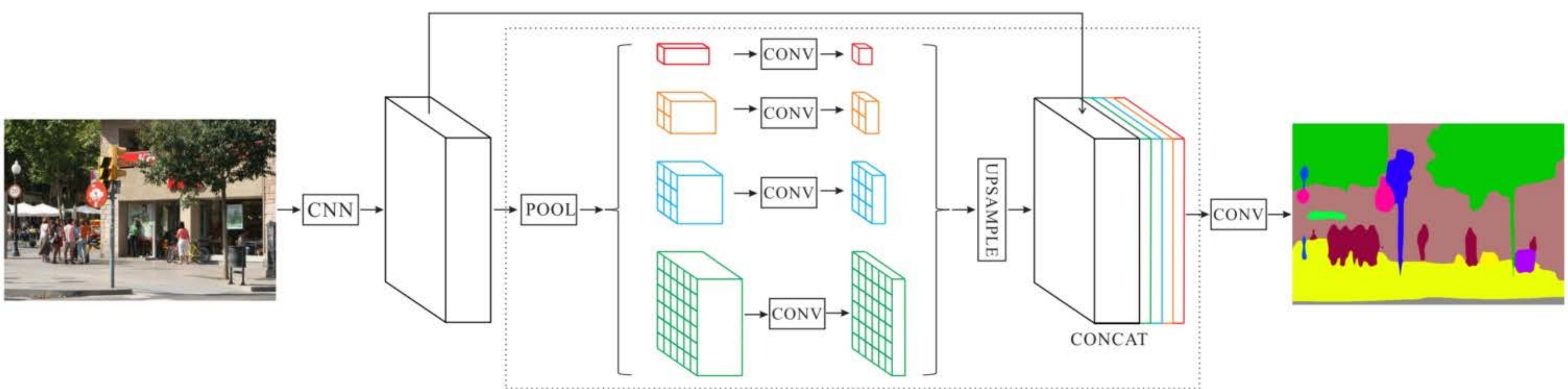
- Motivation: Errors occur because of contextual relations between of objects.
- Eg: car on the road (not in the sky!)
- To take context into account, Pyramid pooling module is proposed.



https://hszhao.github.io/projects/pspnet/

Pyramid Scene Parsing Network (PSPNet)

- Take output from last layer of a CNN,
  - Pool it at different levels (**global average pooling, 2*2,3*3,6*6**).
  - Convolution followed by each pooling
  - Upsample all the outputs and concat the results.
  - Convolution to produce the final outputs.



(a) Input Image (b) Feature Map (c) Pyramid Pooling Module (d) Final Prediction

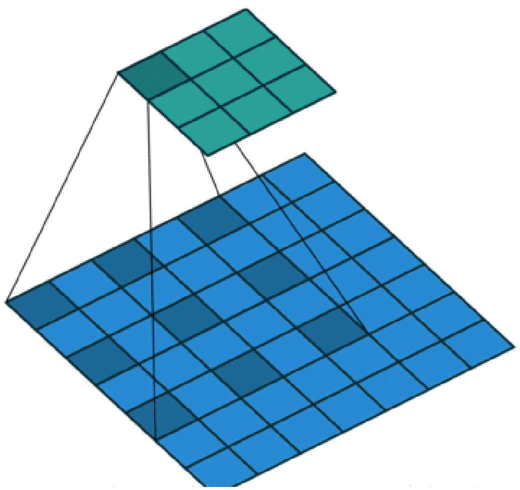
DeepLab

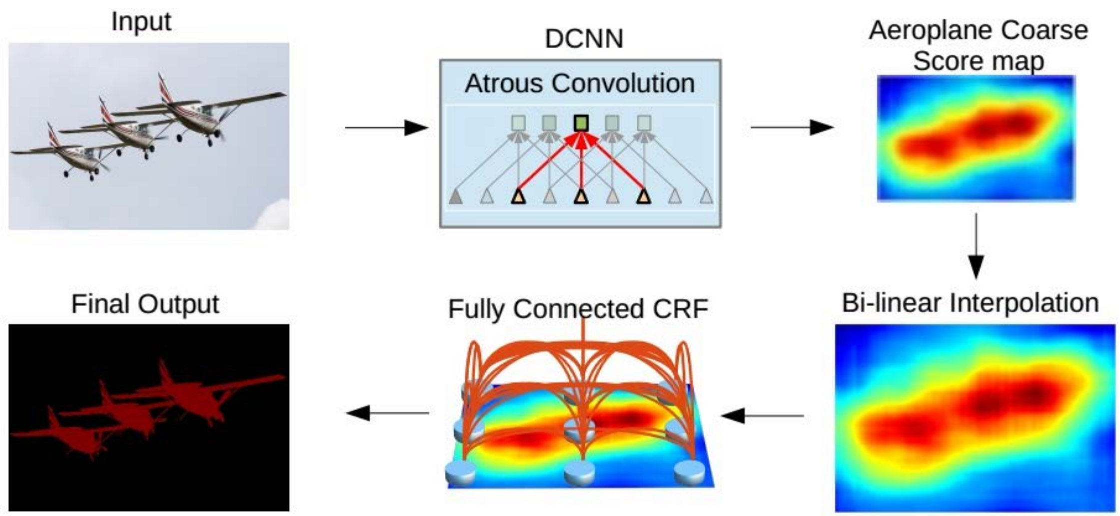
- DeepLab is a state-of-art deep learning model for semantic image segmentation, where the goal is to assign semantic labels (e.g. person, dog, cat) to every pixel in the input image.
- DeepLabv1, DeepLabv2, DeepLabv3, DeepLabv3+,

https://www.tensorflow.org/lite/models/segmentation/overview

DeepLab v1

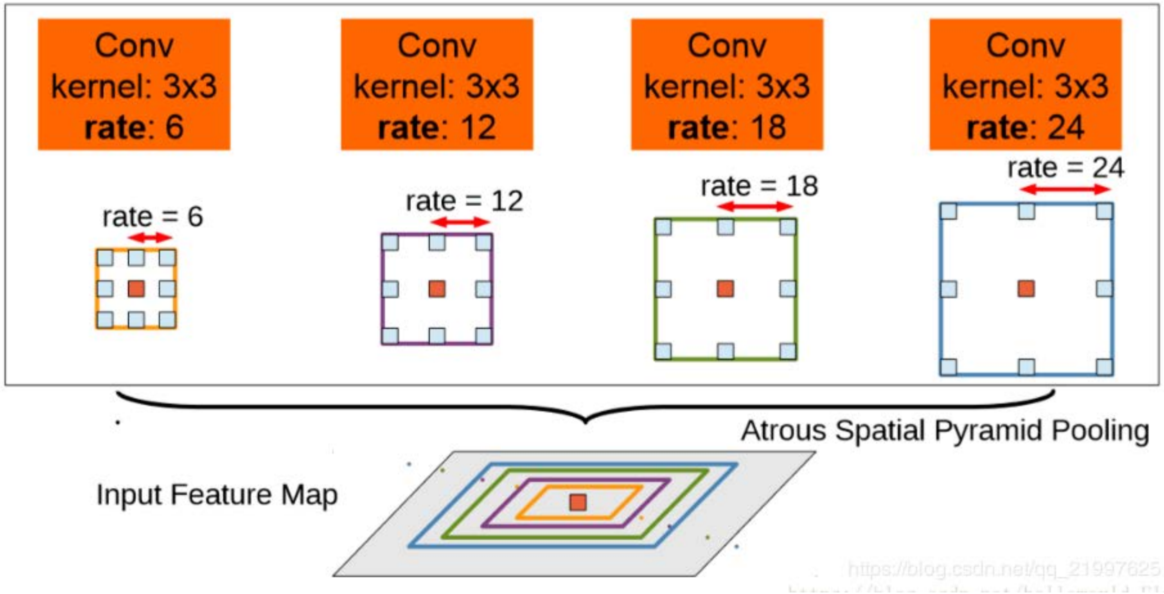
- DeepLab v1
  - Use pretrained VGG-16 to generate coarse map.
  - Atrous convolution enlarges the field of view of filters to incorporate larger context without increasing the number of parameters or the amount of computation.
  - Bilinear interpolation to upsample result.
  - Apply Conditional Random Field (CRF) to refine the result.





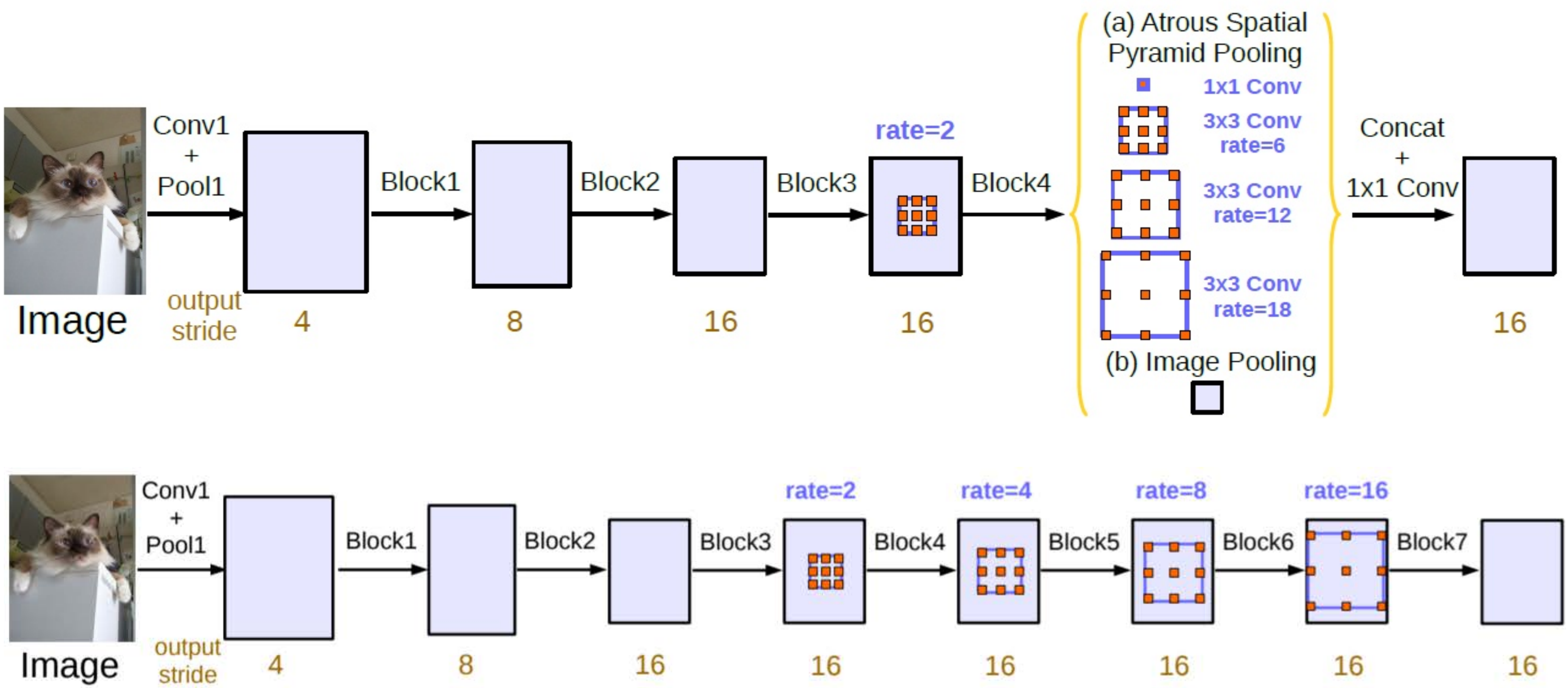
DeepLab v2

- DeepLab v2
- Motivation: existence of object at multiple scales.
  - Use ResNet instead of VGG.
  - Introduce Atrous Spatial Pyramid Pooling (ASPP): the outputs at the last convolution layer is dilated at different rates in parallel branches and fused together. ASPP helps to account for different object scales which can improve the accuracy.



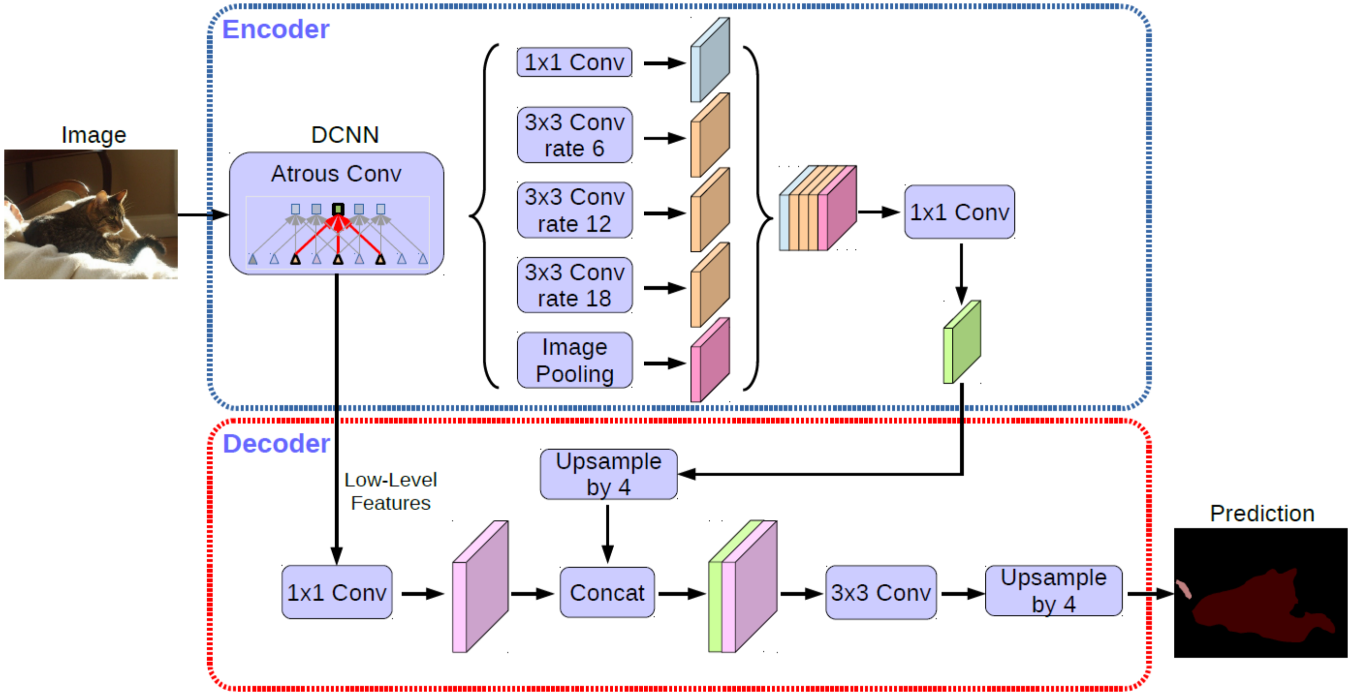
DeepLab v3

- Motivation: capture sharper object boundaries
- Can be used for different framework
- Design modules which employ atrous convolution in cascade or in parallel to capture multi-scale context by adopting multiple atrous rates
- Different Atrous Spatial Pyramid Pooling: image pooling and 1*1 conv



DeepLab v3+

- Add a simple yet effective decoder module to further refine the segmentation results especially along object boundaries.
- DeepLab v3 + Decoder = DeepLab v3+

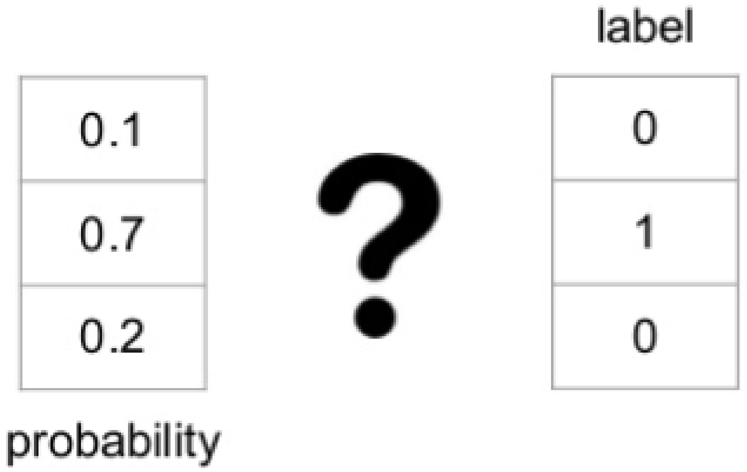


Lecture outline

- Introduction
- Different models
  - Fully Convolutional Network
  - DeconvNet, SegNet
  - U-Net
  - PSPNet
  - DeepLab v1, v2, v3, v3+
- Loss functions

Loss functions for segmentation

- In a supervised deep learning, the loss functions measure the quality of a particular set of parameters based on how well the output of the network agrees with the ground truth labels in the training data.
- Loss functions are used to guide the training process in order to find a set of parameters that reduce the value of the loss function.



Loss functions

- Definition
- Let $\mathbf{P}(Y = 0) = p$ and $\mathbf{P}(Y = 1) = 1 - p$
- The predictions are given by the logistic/sigmoid function

$$\mathbf{P}(\hat{Y} = 0) = \frac{1}{1+e^{-x}} = \hat{p} \text{ and } \mathbf{P}(\hat{Y} = 1) = 1 - \frac{1}{1+e^{-x}} = 1 - \hat{p}$$

- **Cross entropy (CE)** can be defined as follows

$$\mathrm{CE}\,(p, \hat{p}) = -\left(p \log(\hat{p}) + (1 - p)\log(1 - \hat{p})\right)$$

- This loss examines each pixel individually, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector.

Weighted cross entropy

- Weighted cross entropy is a variant of CE where all positive examples get weighted by some coefficients.
- It is used in the case of class imbalance.
- For example, when you have an image with 10% black pixels and 90% white pixels, regular CE won't work very well.
- It is defined as follows

$$\mathrm{WCE}\left(p, \hat{p}\right) = -\left(\beta p \log(\hat{p}) + (1-p)\log(1-\hat{p})\right)$$

Balanced cross entropy

- Balanced cross entropy (BCE) is similar to WCE. The only difference is that we weight also the negative examples.
- BCE can be defined as follows:
- $$\mathrm{BCE}\left(p, \hat{p}\right) = -\left(\beta p \log(\hat{p}) + (1-\beta)(1-p)\log(1-\hat{p})\right)$$

Focal loss

- Easily classified negatives comprise the majority of the loss and dominate the gradient. While α balances the importance of positive/negative examples, it does not differentiate between easy/hard examples.
- Focal loss (FL) tries to down-weight the contribution of easy examples, so that the CNN focuses more on hard examples.
- FL can be defined as follows:

$$\mathrm{FL}\left(p, \hat{p}\right) = -\left(\alpha(1-\hat{p})^{\gamma} p \log(\hat{p}) + (1-\alpha)\hat{p}^{\gamma}(1-p)\log(1-\hat{p})\right)$$

Focal Loss for Dense Object Detection
https://github.com/facebookresearch/Detectron

Dice Loss / F1 score

- The Dice coefficient is similar to the Jaccard Index (Intersection over Union, IoU):

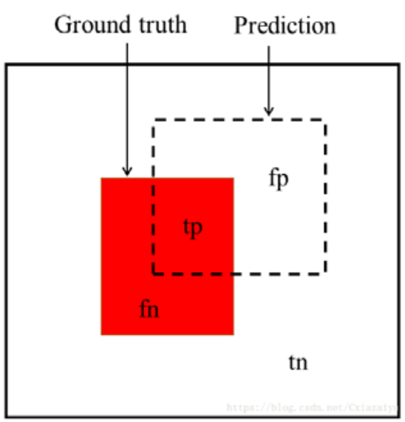
$$\mathrm{DC} = \frac{2TP}{2TP + FP + FN} = \frac{2|X \cap Y|}{|X| + |Y|}$$

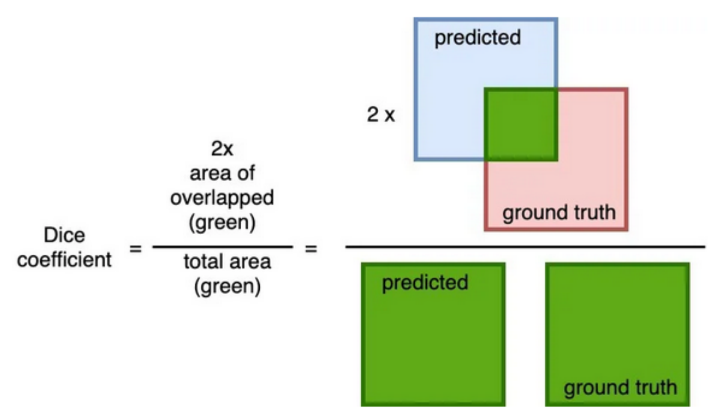
$$\mathrm{IoU} = \frac{TP}{TP + FP + FN} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

- The dice coefficient can also be defined as a loss function:

$$\mathrm{DL}\left(p, \hat{p}\right) = 1 - \frac{2p\hat{p} + 1}{p + \hat{p} + 1}$$

where $p \in \{0, 1\}$ and $0 \leq \hat{p} \leq 1$.

- 



Dice

- Example

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

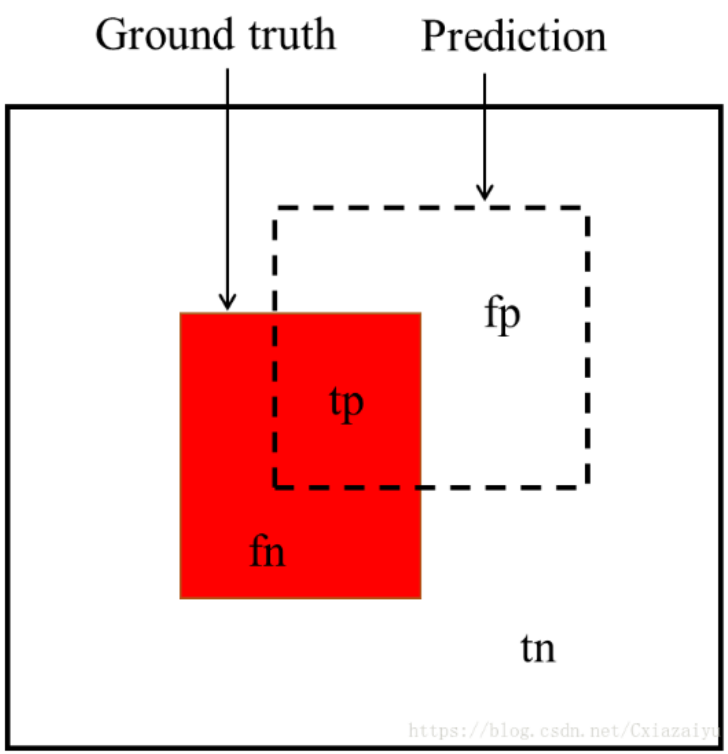
prediction target

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

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

Tversky loss

- Tversky index (TI) is a generalization of Dice's coefficient. TI adds a weight to FP (false positives) and FN (false negatives).
- To give FNs higher weights than FPs in training our network for highly imbalanced data, where detecting small lesions is crucial.



Tversky loss

- Tversky index (TI) is a generalization of Dice's coefficient. TI adds a weight to FP (false positives) and FN (false negatives).

$$\mathrm{TI}\left(p,\hat{p}\right)=\frac{p\hat{p}}{p\hat{p}+\beta(1-p)\hat{p}+(1-\beta)p(1-\hat{p})}$$

- Let $\beta=\frac{1}{2}$

- which is just the regular Dice coefficient.

$$\mathrm{TI}\left(p,\hat{p}\right)=\frac{2p\hat{p}}{2p\hat{p}+(1-p)\hat{p}+p(1-\hat{p})}=\frac{2p\hat{p}}{\hat{p}+p}$$

Tversky loss function for image segmentation using 3D fully convolutional deep networks, 2017