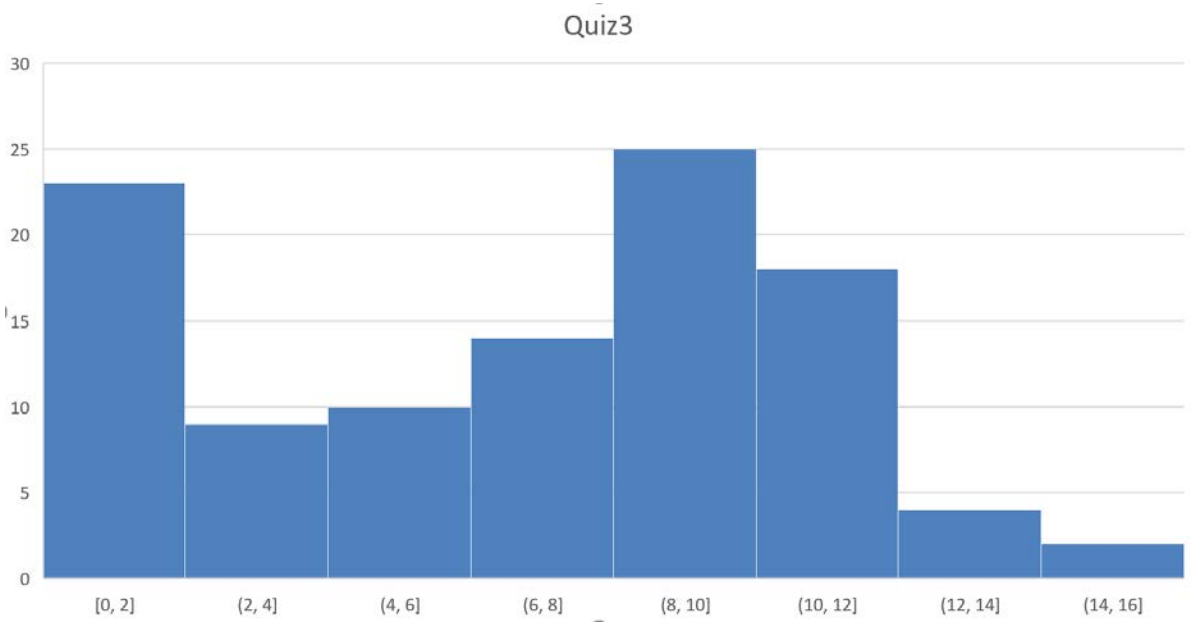
Lecture 9: CNNs in Computer Vision – Part I

Xuming He
SIST, ShanghaiTech
Fall, 2019

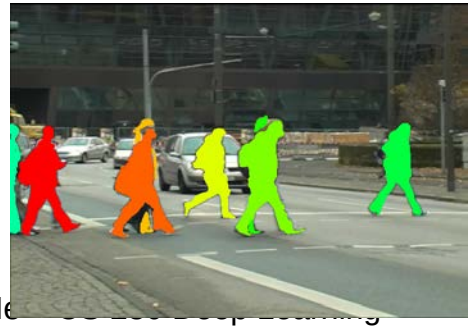
Administrative

- Homework 2 out
  - Get familiar with CNNs and Project pipeline
  - Think about your project proposals
- Quiz 4 results



Review

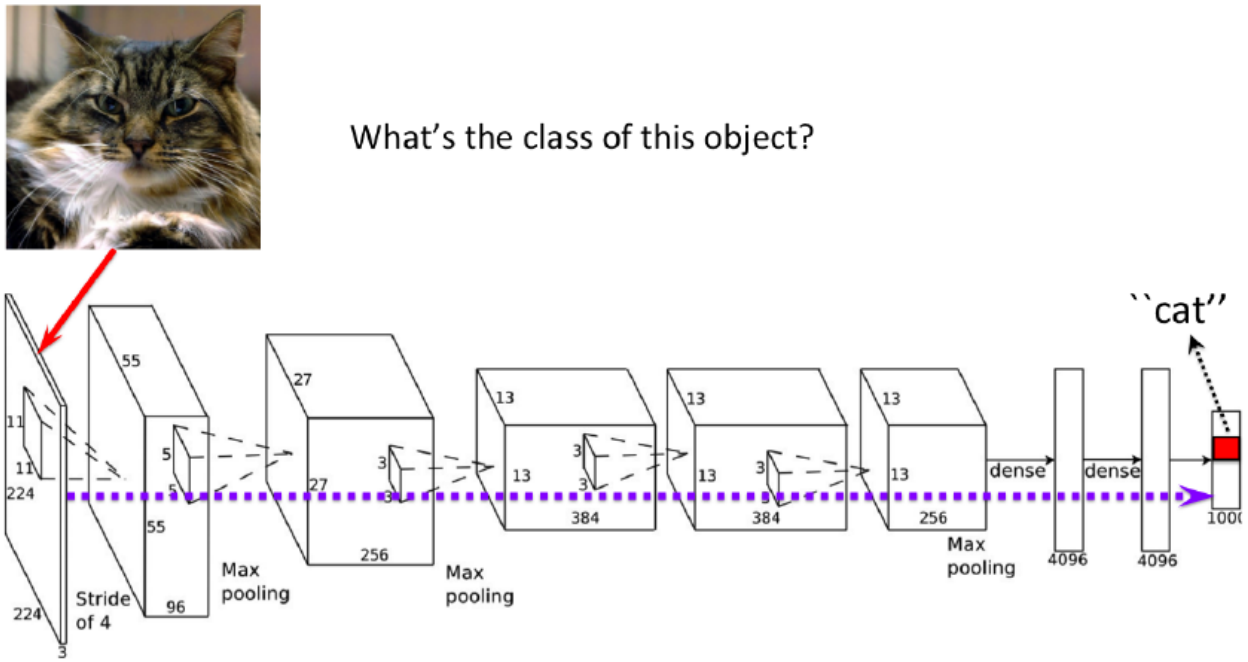
- In general, our goal is to learn a mapping from a signal to a ‘semantically meaningful’ representation.
  - Output can have many different forms:



red panda (*Ailurus fulgens*)

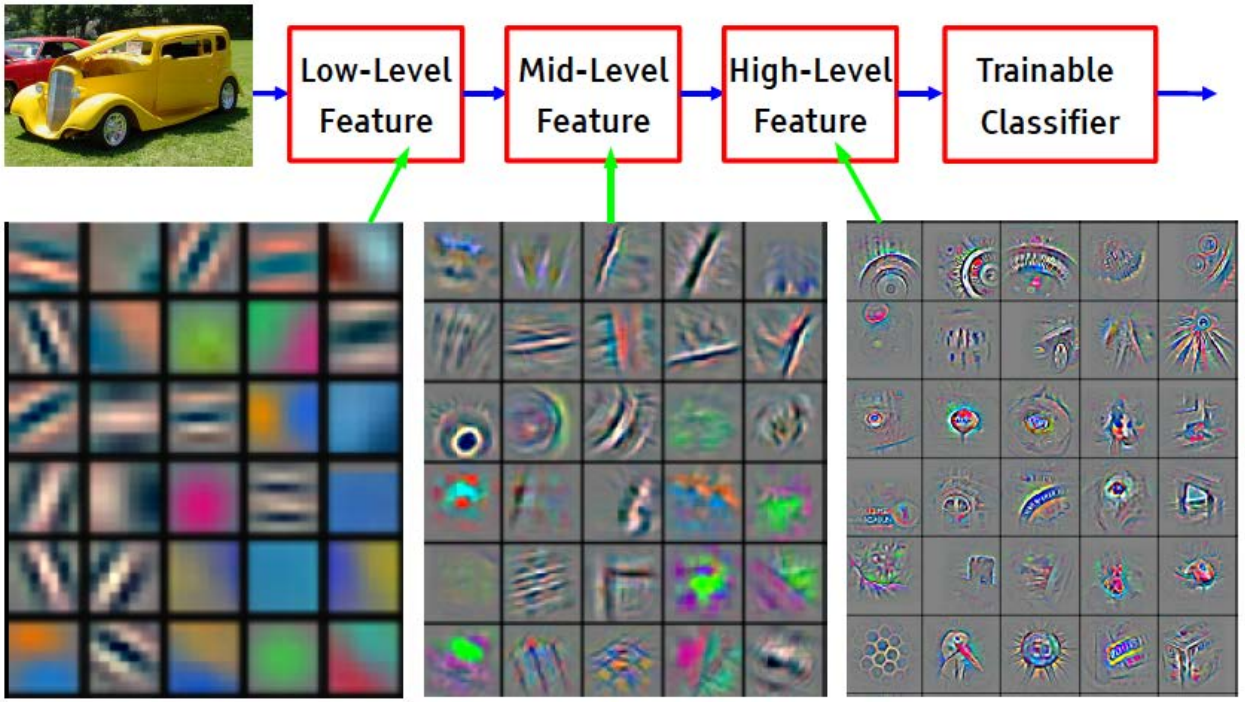
Previously on CNNs

- CNNs for image/object classification
  - AlexNet, VGGNet, GoogLeNet, ResNet, DenseNet
  - Trend towards deeper networks with more flexible network architecture
  - Better representation and more effective learning strategies



More on classification

- Why it works well for image classification?
  - Built-in translation and small deformation invariance
  - Hierarchical feature learning – shared representation
  - End-to-end training for the target problem



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

More on classification

- Is image classification a solved problem?
  - “(Super-)Human level” performance on some benchmarks
    - Face identification
    - ImageNet 1000 classes
- But compared to human vision…
  - Limitations in learning
    - We can learn new classes using one or two examples
    - We can also handle label noises
    - We can generalize to unfamiliar scenes
  - Limitation in prediction
    - We can also predict the uncertainty
    - We can easily handle adversarial examples
    - We are much more efficient in power consumption

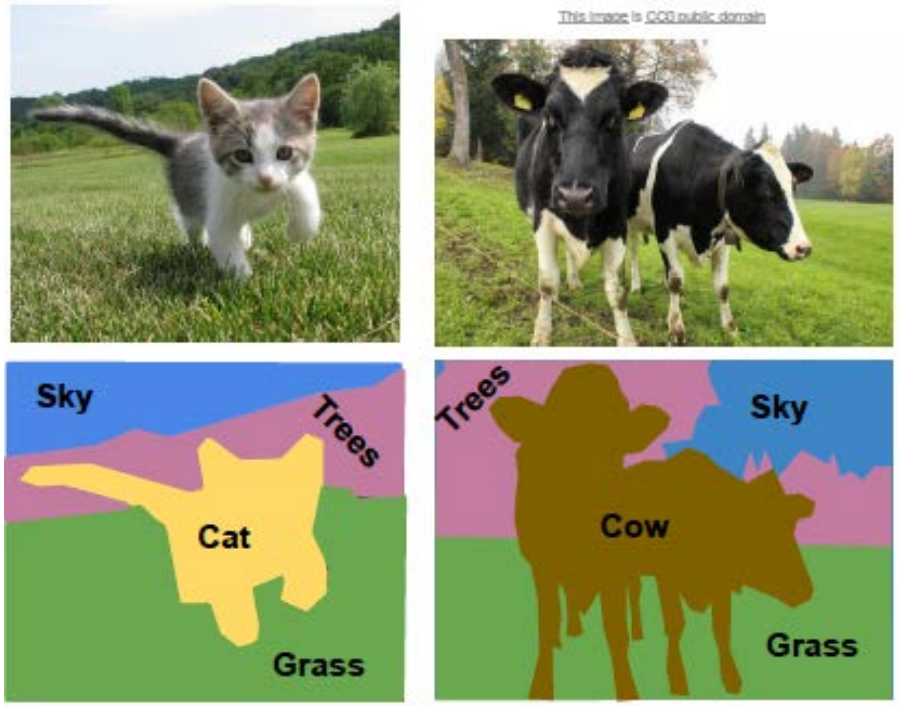
Outline

- What is semantic segmentation?
- Network architecture for semantic segmentation
  - Main idea for dense prediction
  - Upsampling operators
  - Modern network design
- Network training losses

Acknowledgement: Feifei Li et al's cs231n notes

Semantic Segmentation

- Problem setup
  - Label each pixel in the image with an object category label
  - Do not differentiate object instances



Key to many applications

- Autonomous robots and cars

- Safety and security

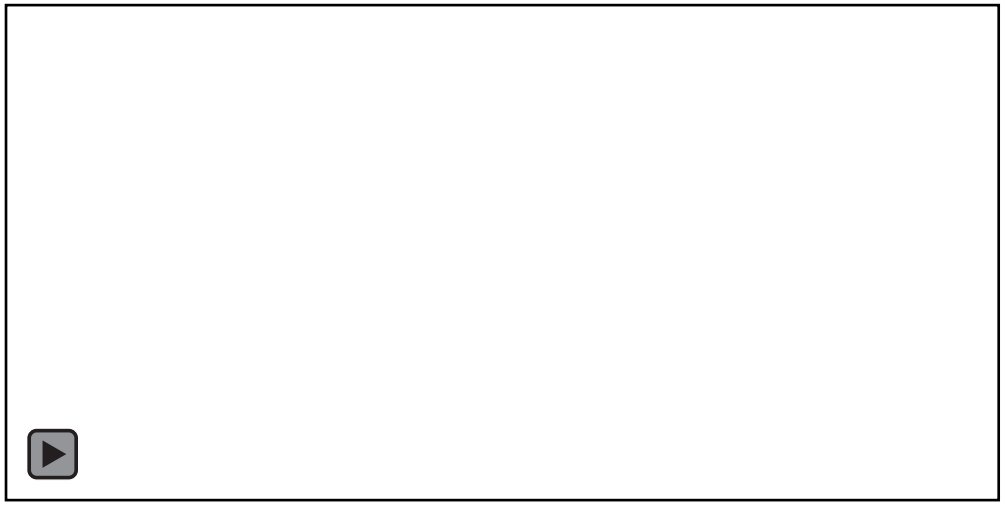
- Medical analysis and health

- etc…

Key to many applications

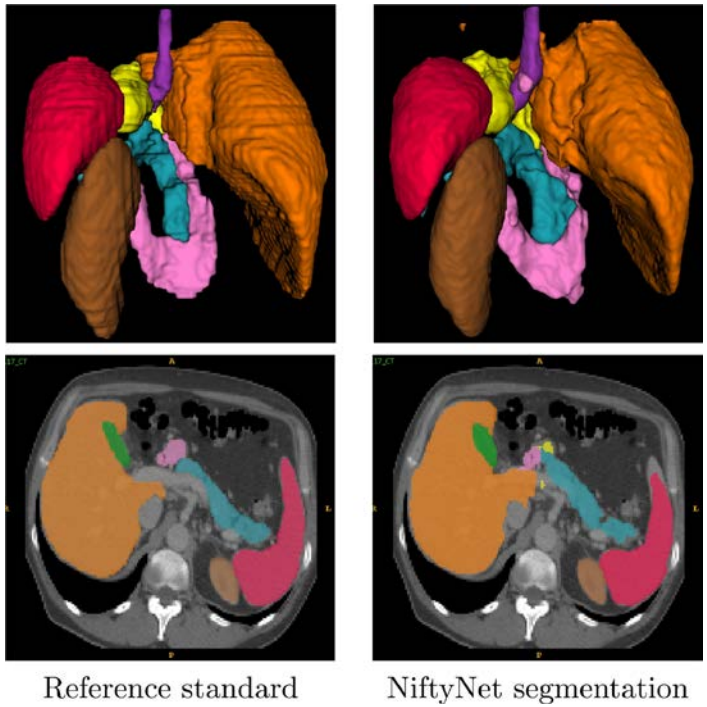
Autonomous driving

https://youtu.be/qWl9idsCuLQ



Multi-organ abdominal CT segmentation

https://doi.org/10.1016/j.cmpb.2018.01.025

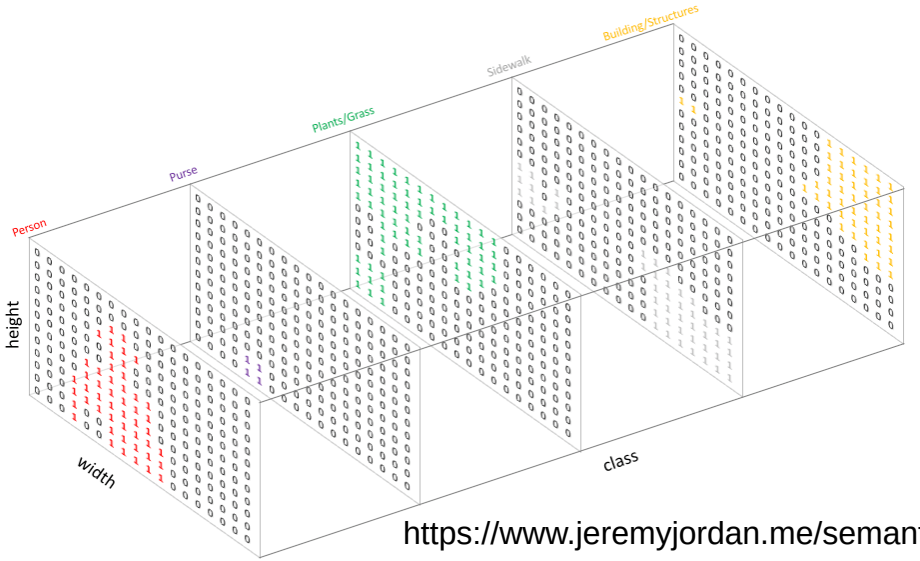


Reference standard NiftyNet segmentation

Semantic Segmentation

- Problem formulation
  - Pixel-wise object classification task

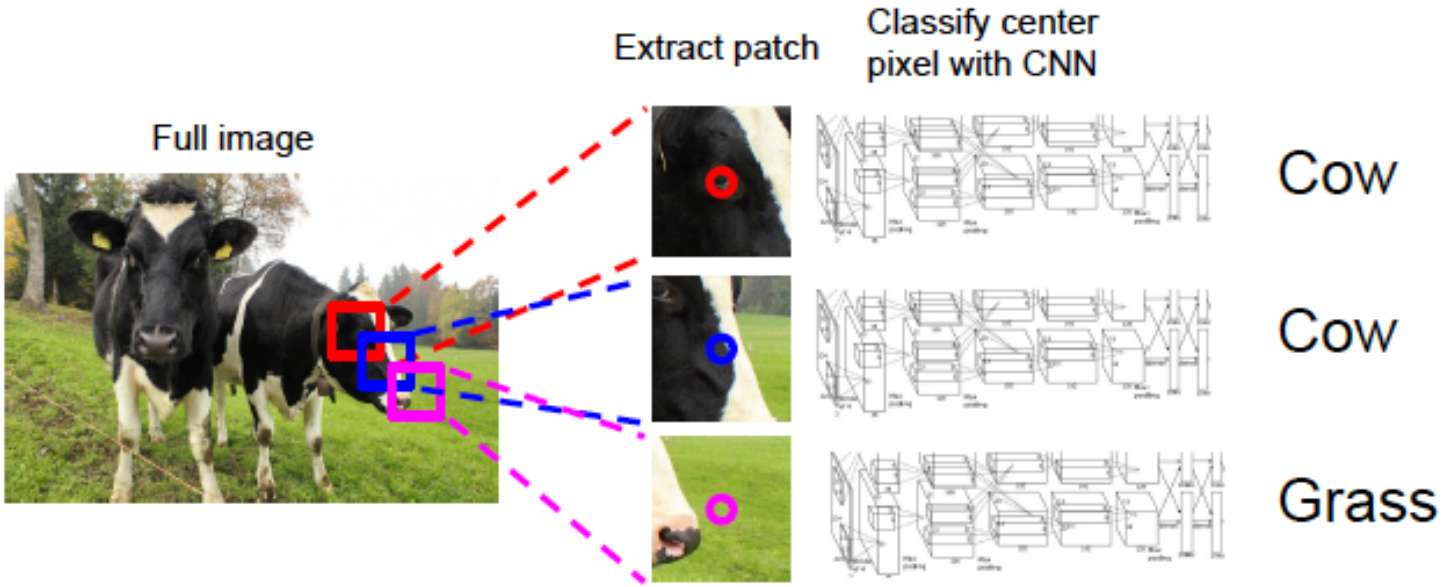
  - One-hot encoding



https://www.jeremyjordan.me/semantic-segmentation/

Why this is challenging?

- A naïve approach

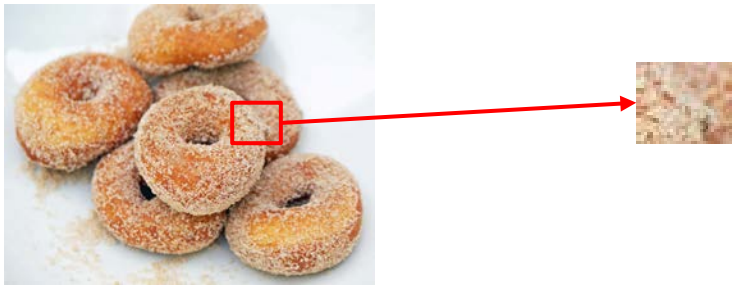


Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

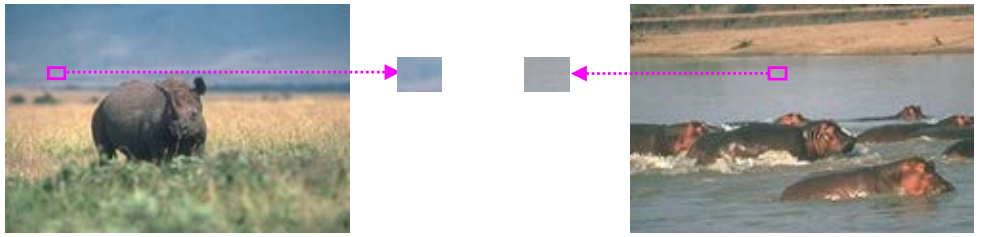
Problem: Very inefficient! Not reusing shared features between overlapping patches

Why this is challenging?

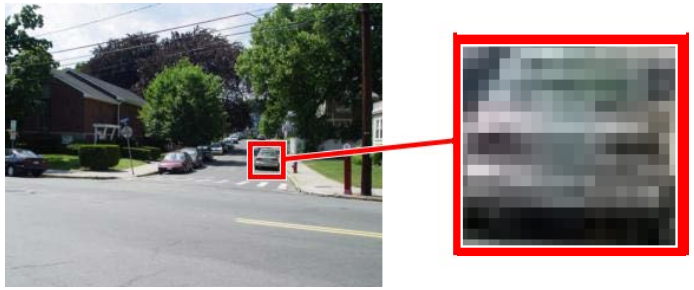
- Ambiguities in local image cues
  - Contours



  - Regions



  - Objects



- Inspiration from human vision
  - To use contextual information from a scene

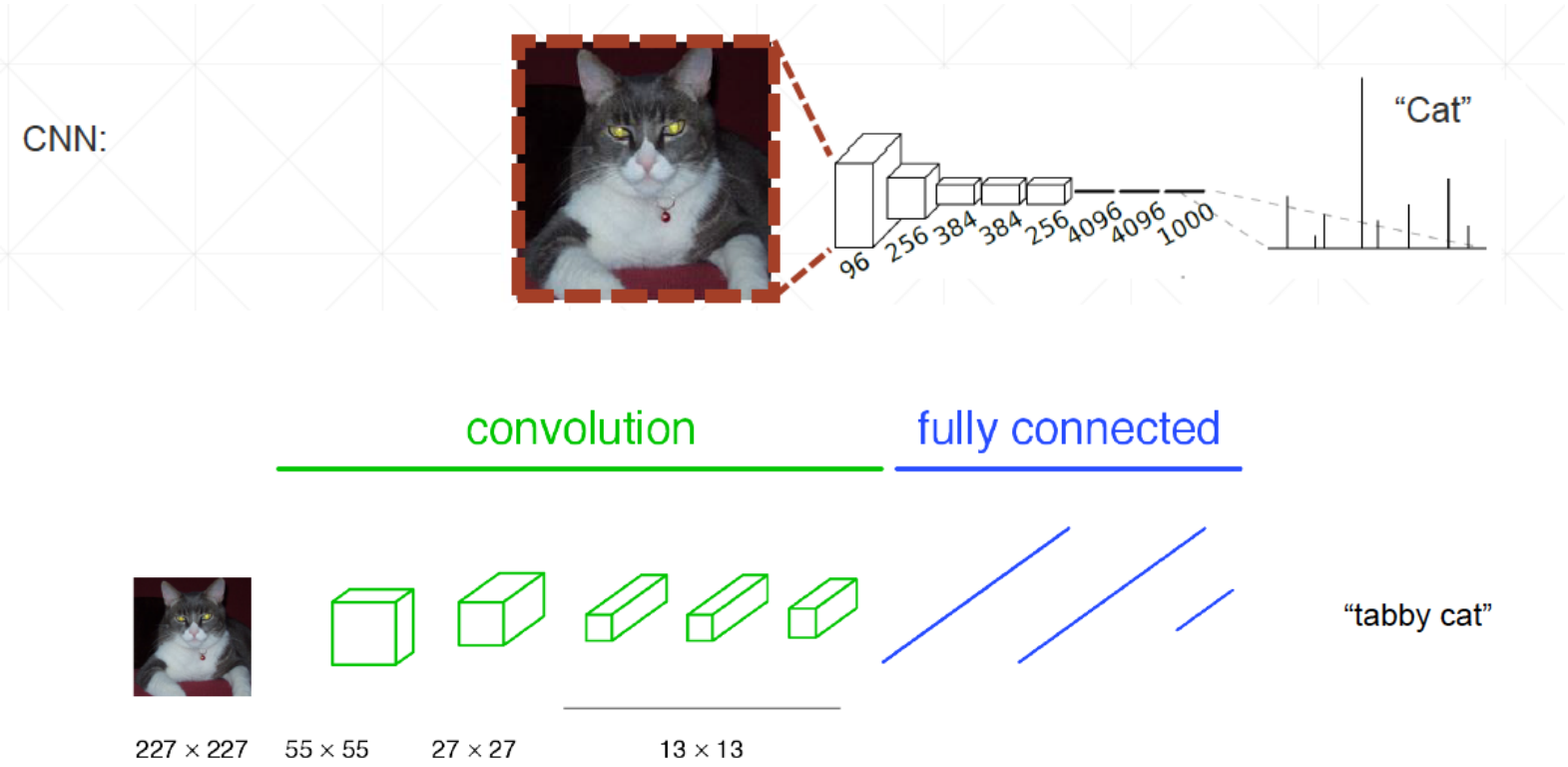
Outline

- What is semantic segmentation?
- Network architecture for semantic segmentation
  - Main idea for dense prediction
  - Upsampling operators
  - Modern network design
- Network training losses

Acknowledgement: Feifei Li et al's cs231n notes

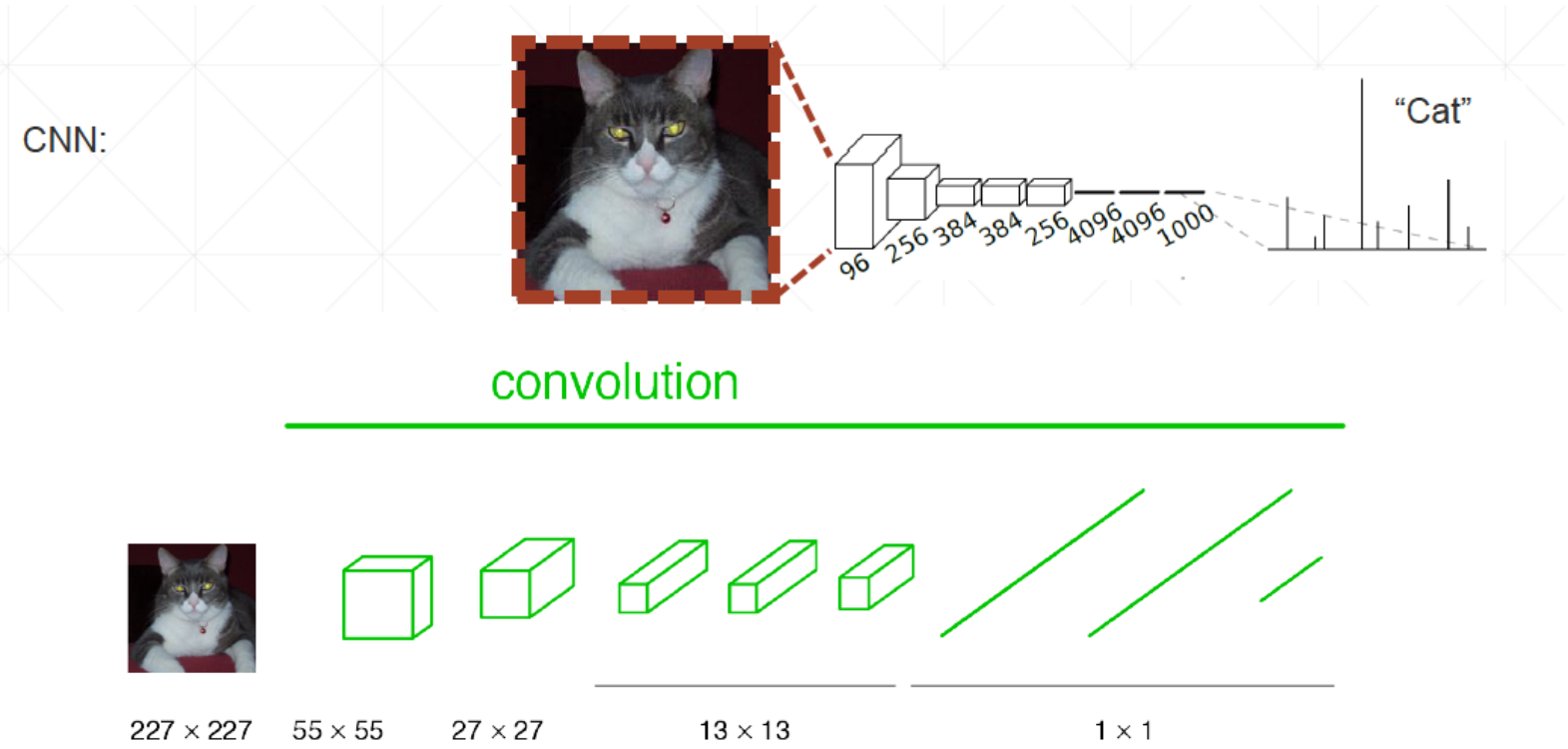
Network Design I: Efficiency

- Starting from a classification network



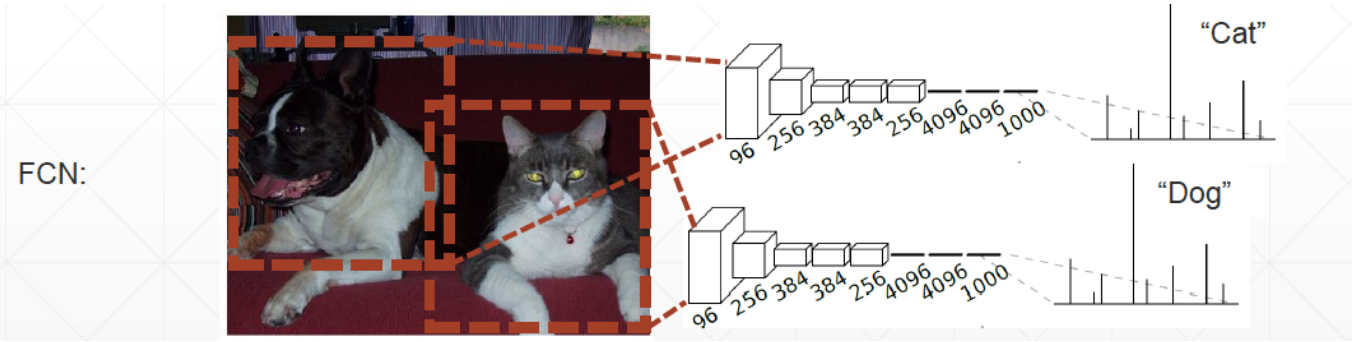
Network Design I: Efficiency

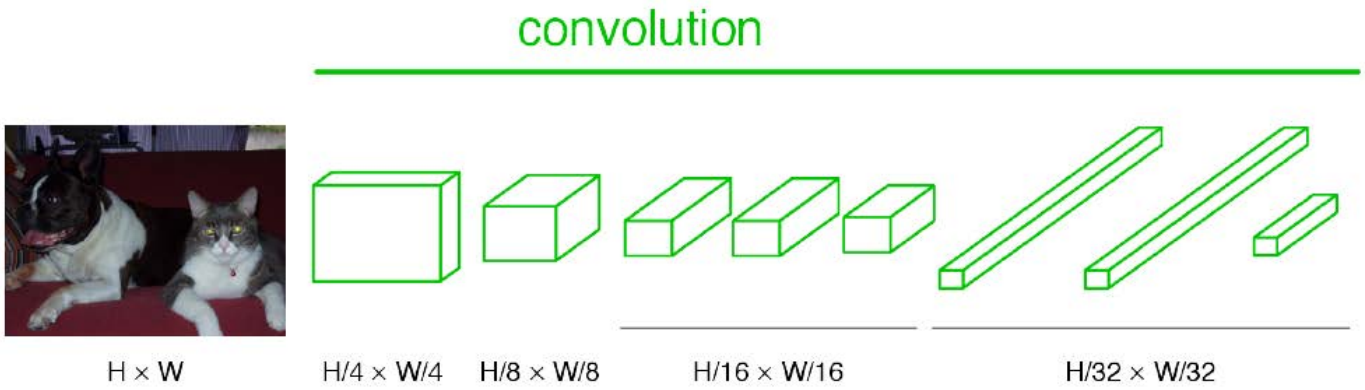
- Interpreting fully connected layers as 1x1 convolution



Network Design I: Efficiency

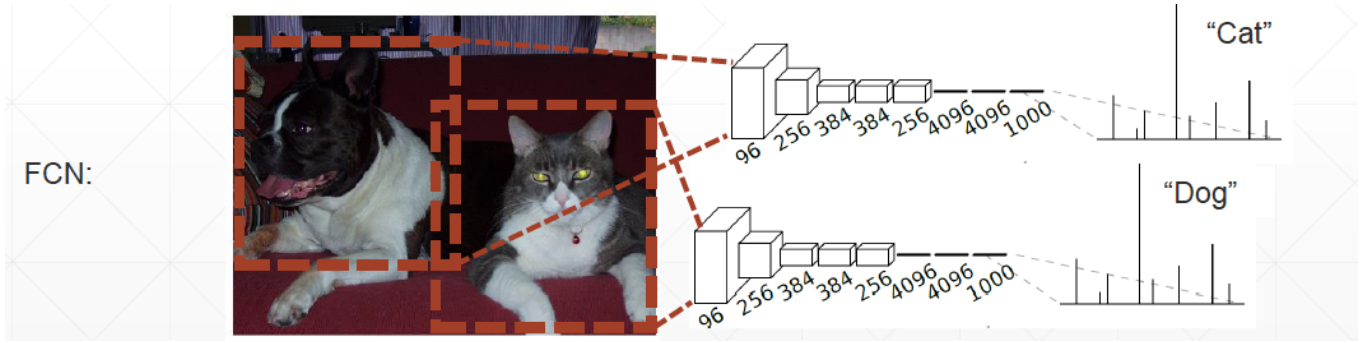
- Extending to a complete image

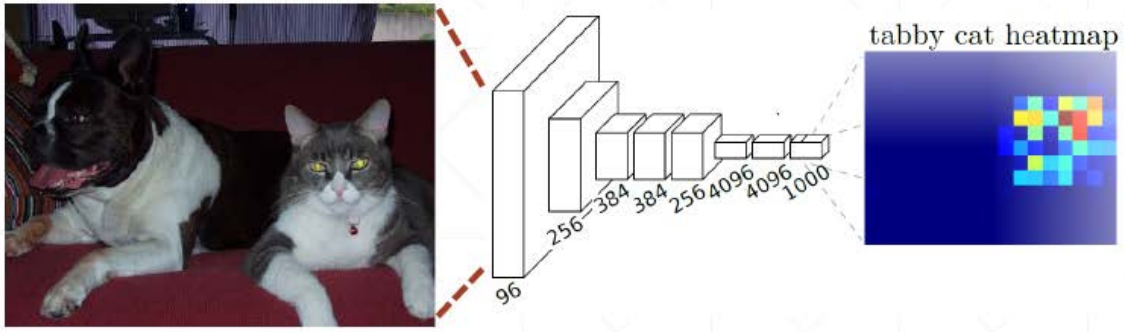




Network Design I: Efficiency

- Extending to a complete image

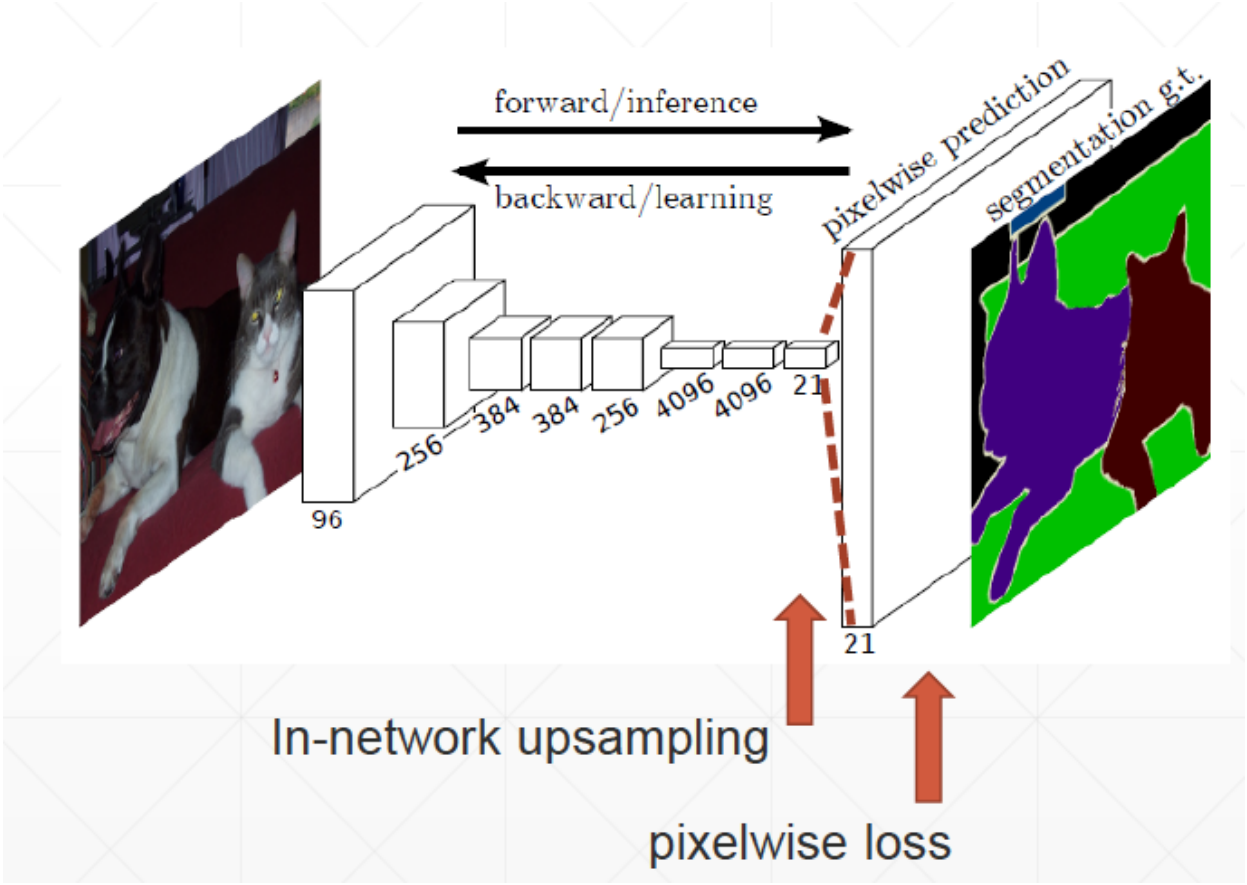




- Keep kernel sizes and strides
- Replace dense layer with convolution

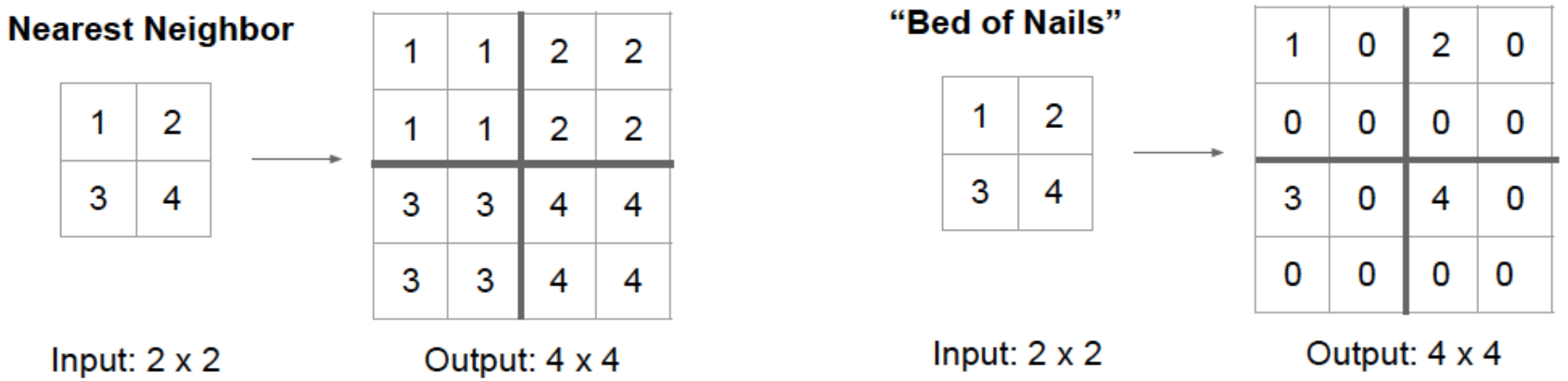
Network Design I: Efficiency

- Main idea: CNN + Upsampling operation in network



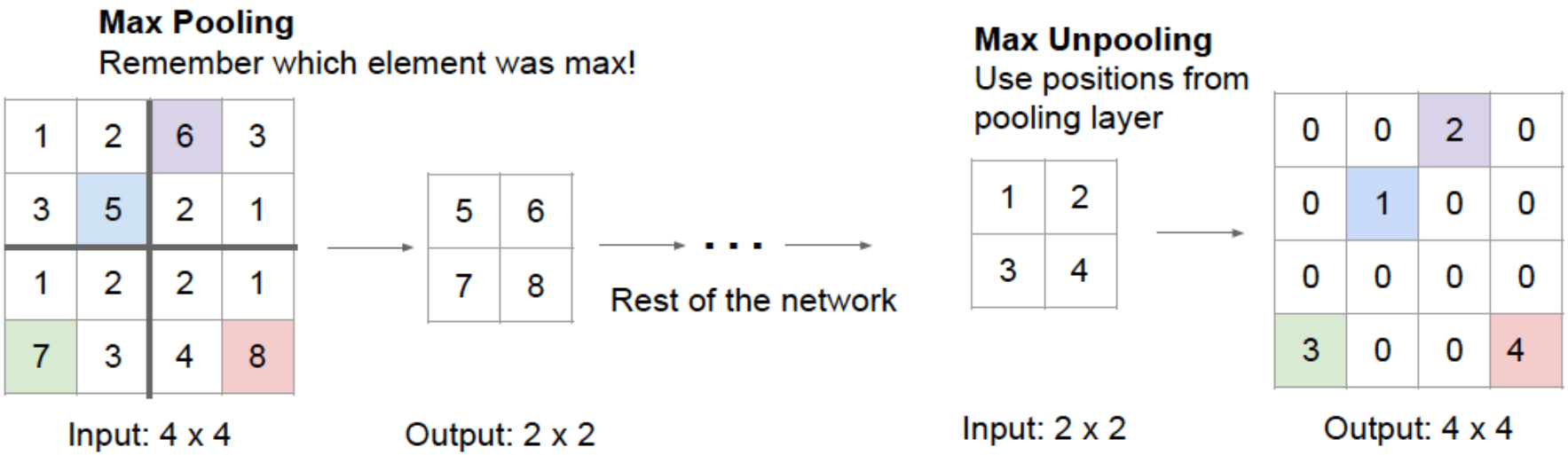
In-Network upsampling

- Unpooling

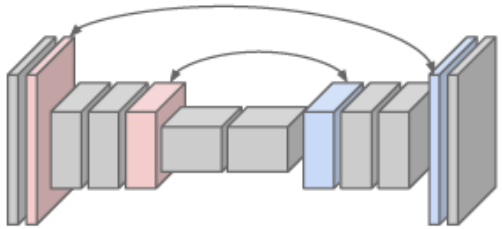


In-Network upsampling

- Max Unpooling



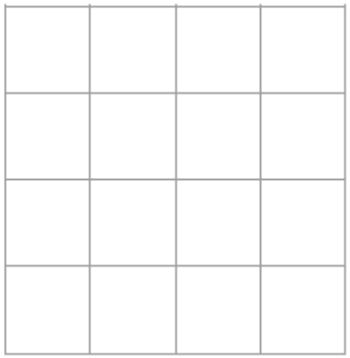
Corresponding pairs of downsampling and upsampling layers



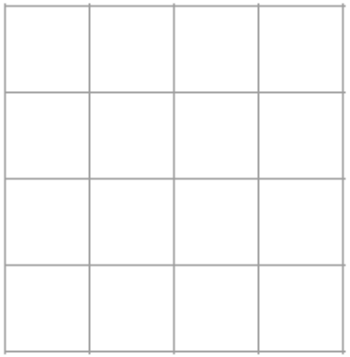
In-Network upsampling

- Learnable Upsampling: Transpose convolution

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



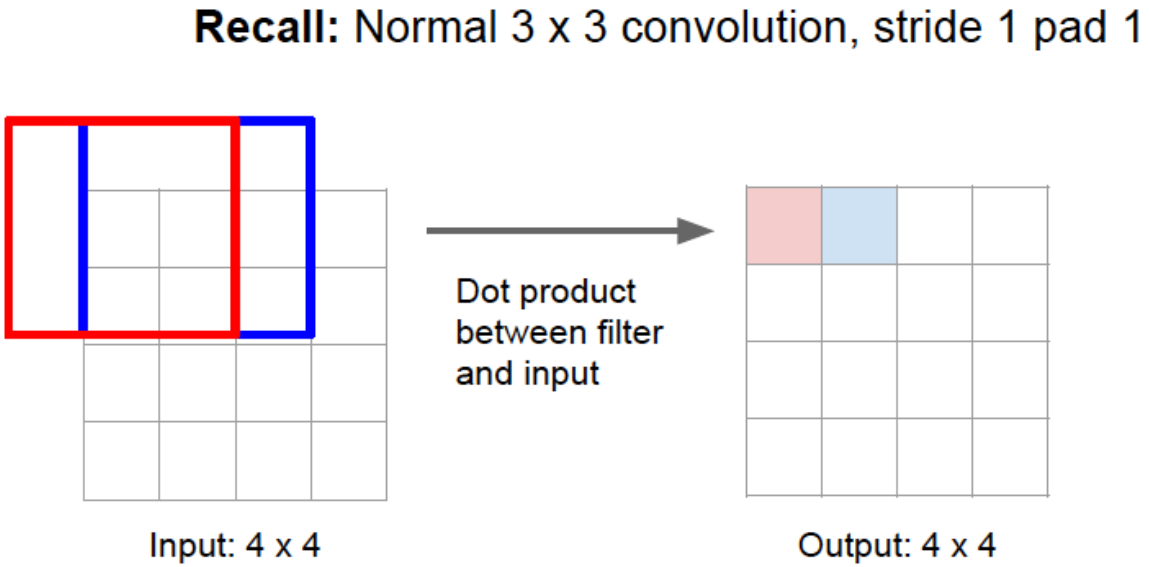
Input: 4 x 4



Output: 4 x 4

In-Network upsampling

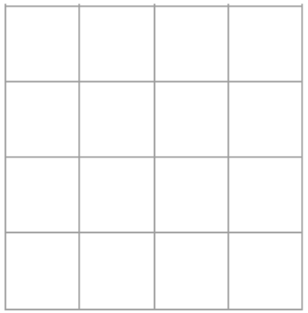
- Learnable Upsampling: Transpose convolution



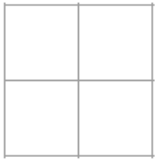
In-Network upsampling

- Learnable Upsampling: Transpose convolution

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



Input: 4 x 4

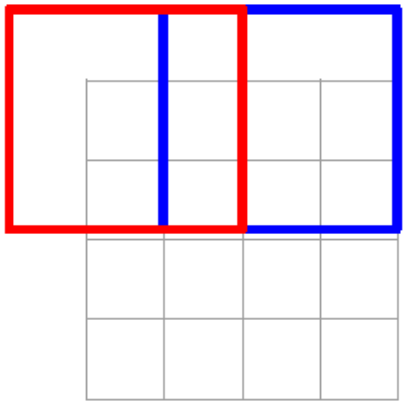


Output: 2 x 2

In-Network upsampling

- Learnable Upsampling: Transpose convolution

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



Input: 4 x 4

Dot product between filter and input

Output: 2 x 2

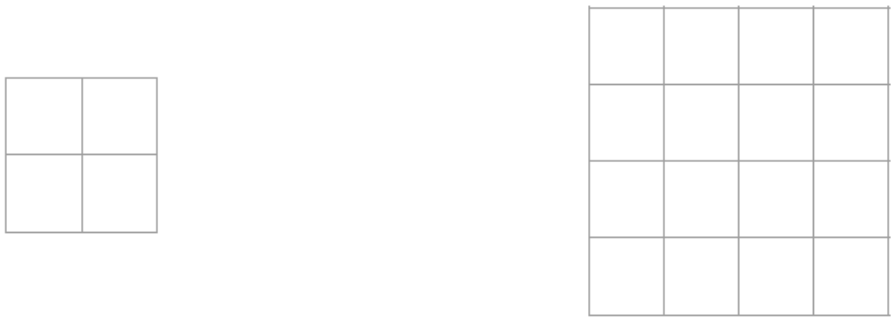
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

In-Network upsampling

- Learnable Upsampling: Transpose convolution

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

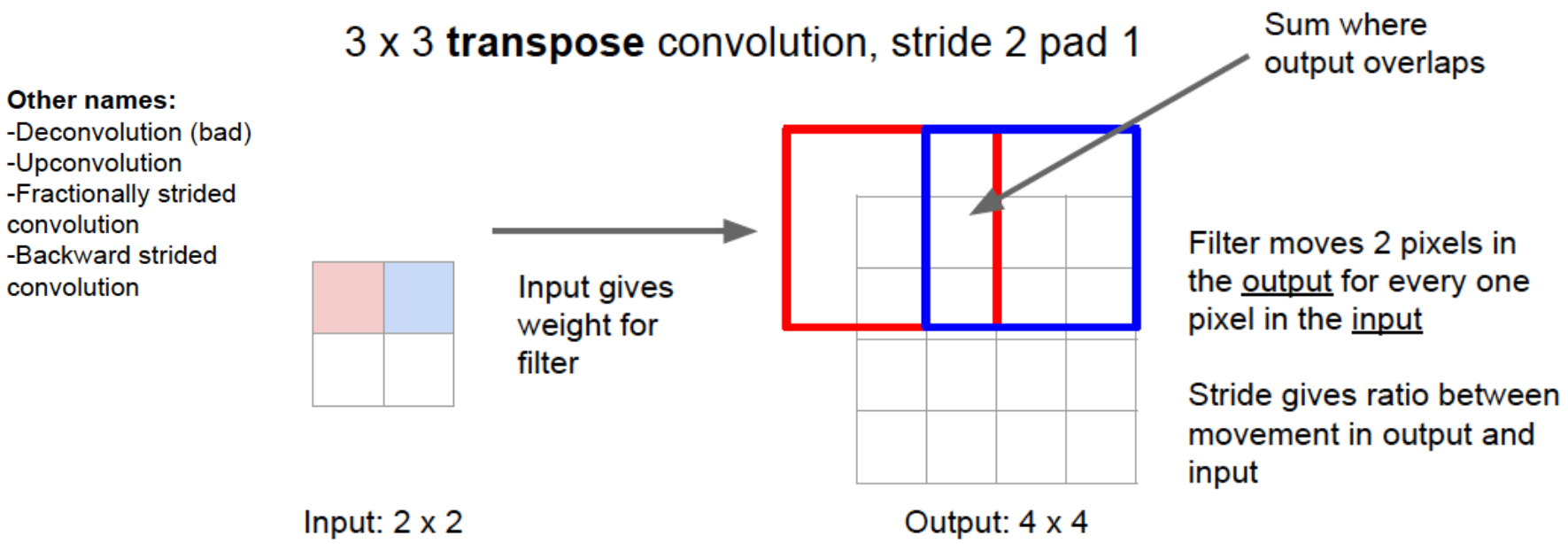


Input: 2 x 2

Output: 4 x 4

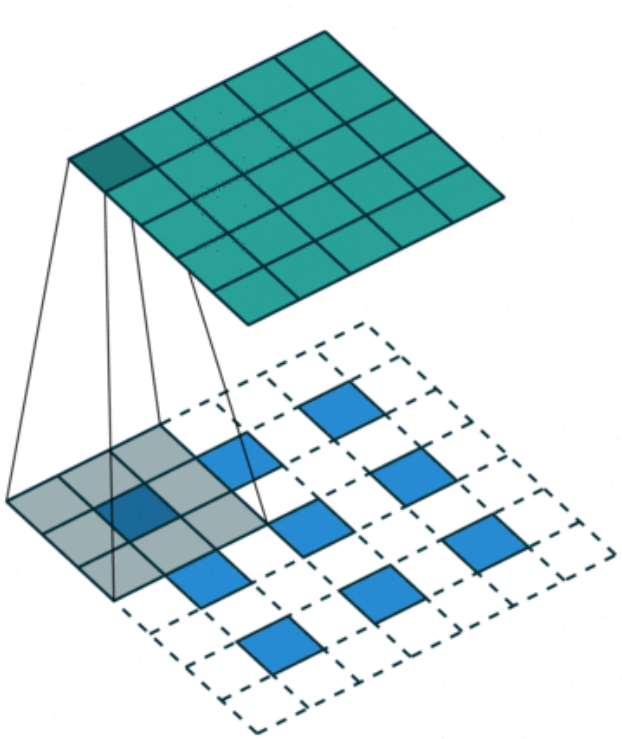
In-Network upsampling

- Learnable Upsampling: Transpose convolution



In-Network upsampling

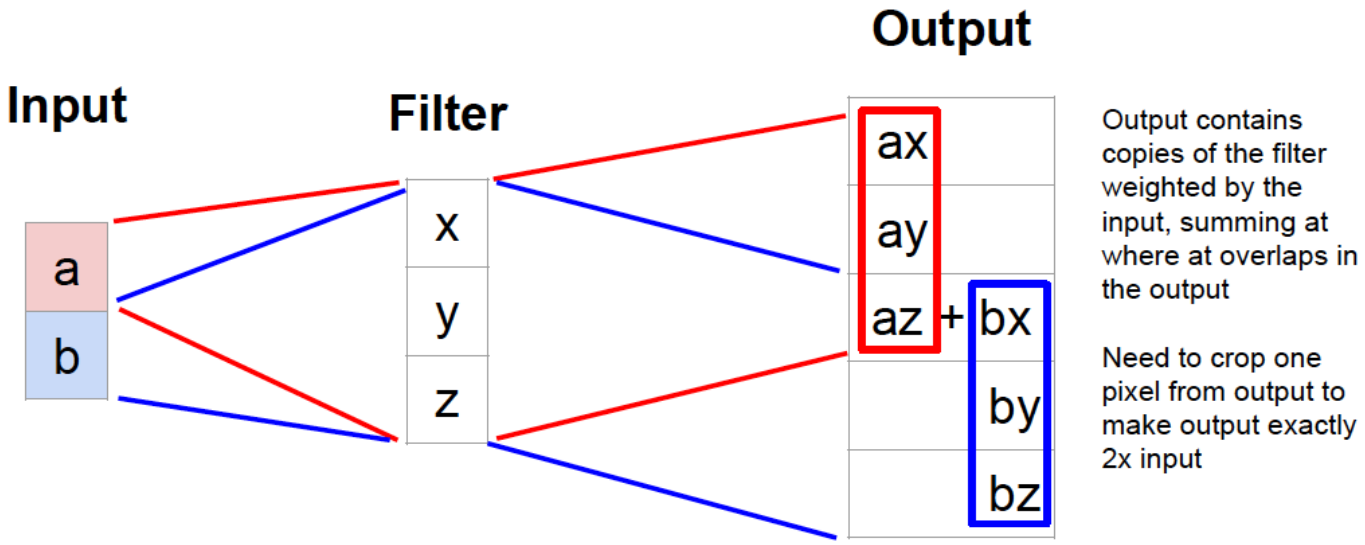
- 2D animation



https://github.com/vdumoulin/conv_arithmetic

In-Network upsampling

- Learnable Upsampling: Transpose convolution
  - 1D example



In-Network upsampling

- Learnable Upsampling: Transpose convolution
  - 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:

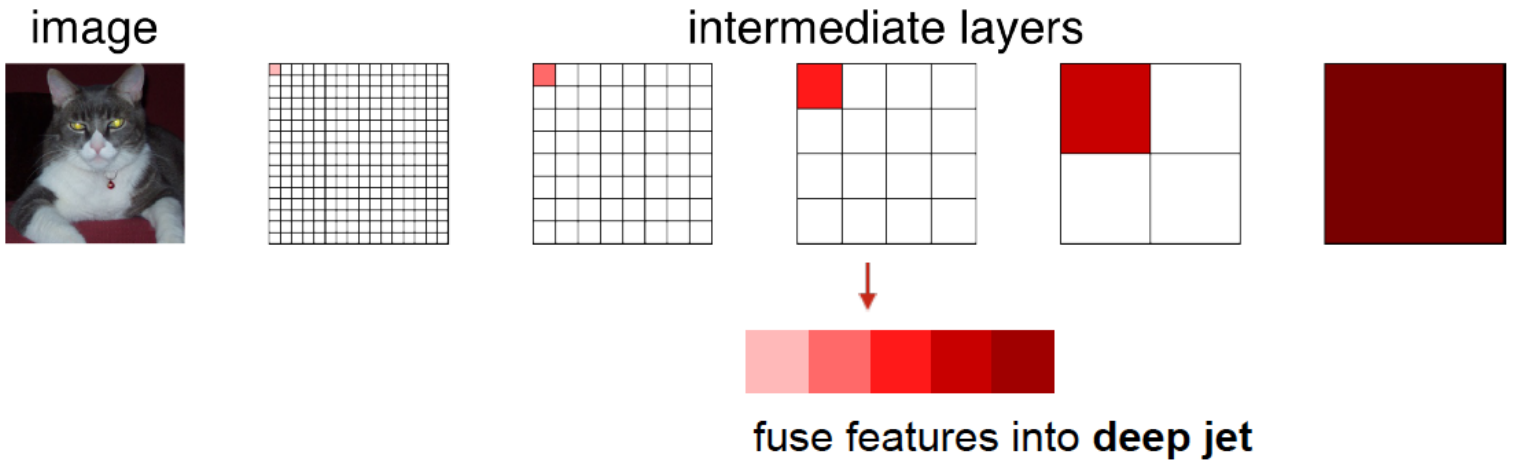
$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

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

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

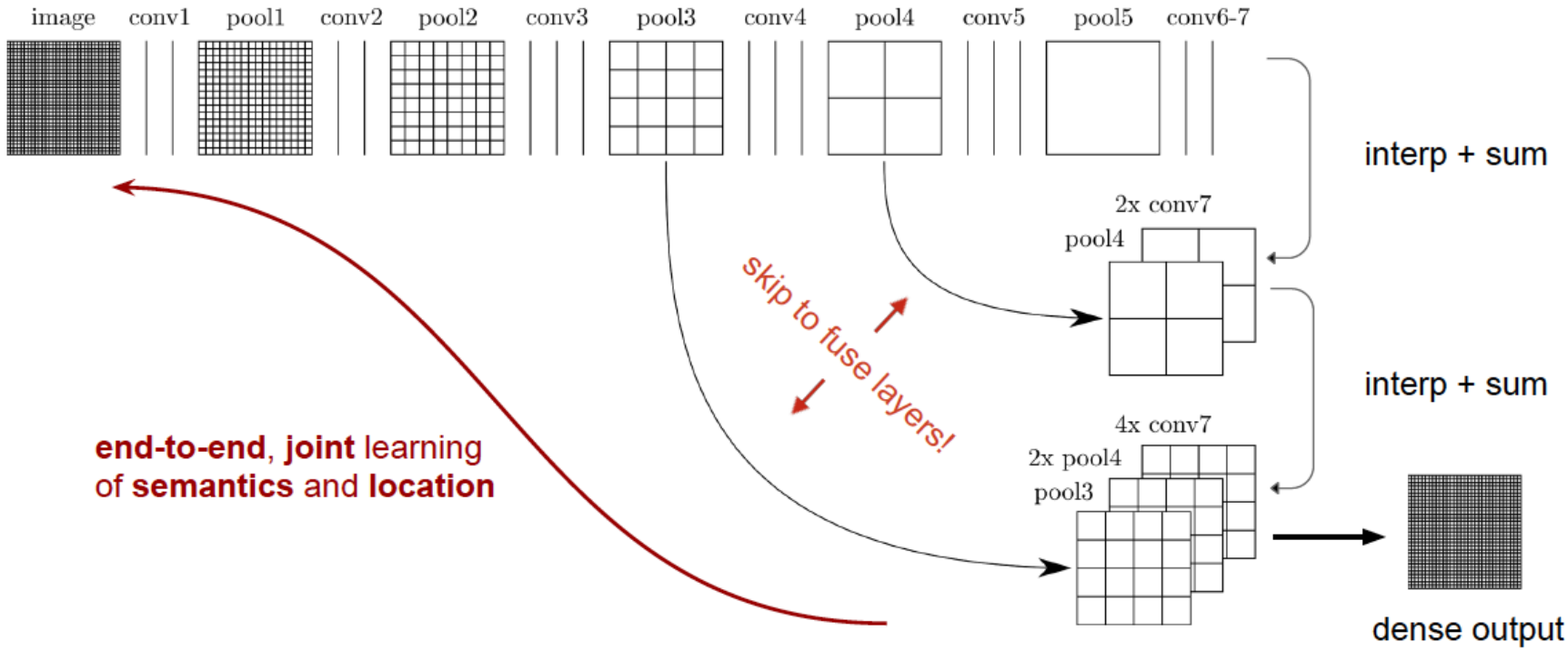
Network Design II: Spatial resolution

- Fully Convolutional Network [Long et al, CVPR 2015]
  - Upsampling: low-resolution, lack spatial details
  - Combining *where (local, shallow)* with *what (global, deep)*



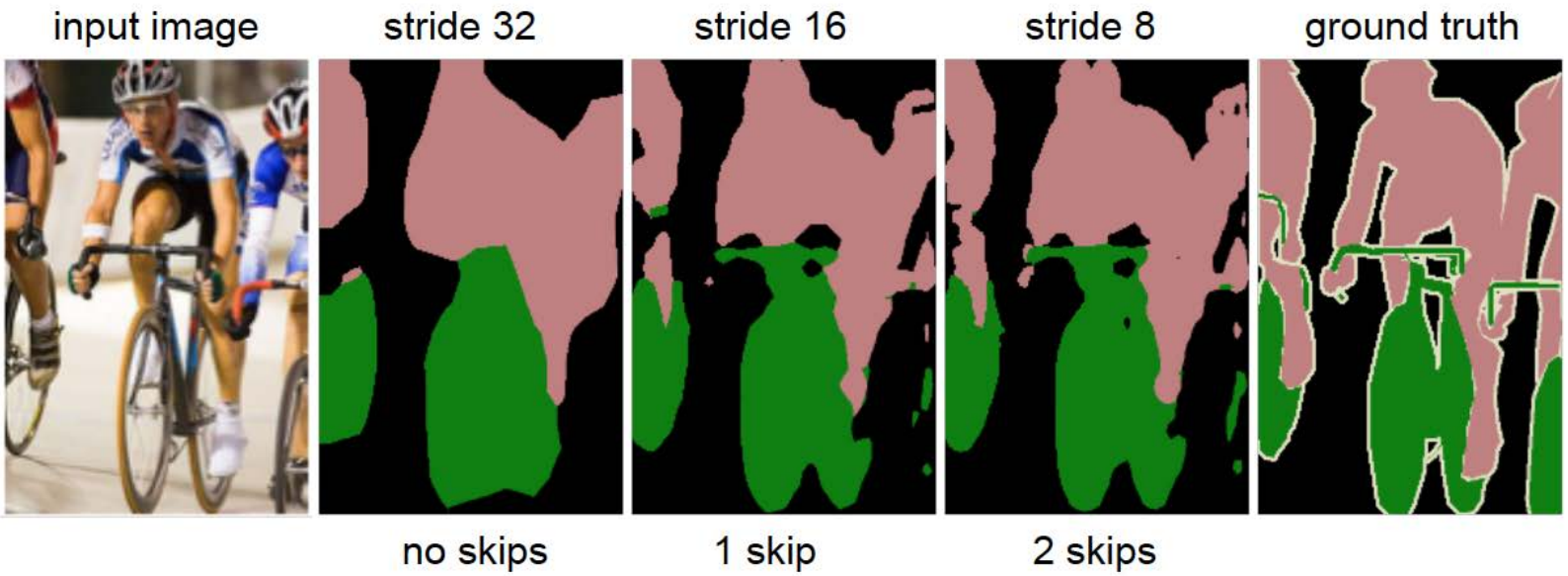
Network Design II: Spatial resolution

- Fully Convolutional Network [Long et al, CVPR 2015]
  - Upsampling: low-resolution, lack spatial details
  - Introducing ***skip layers***



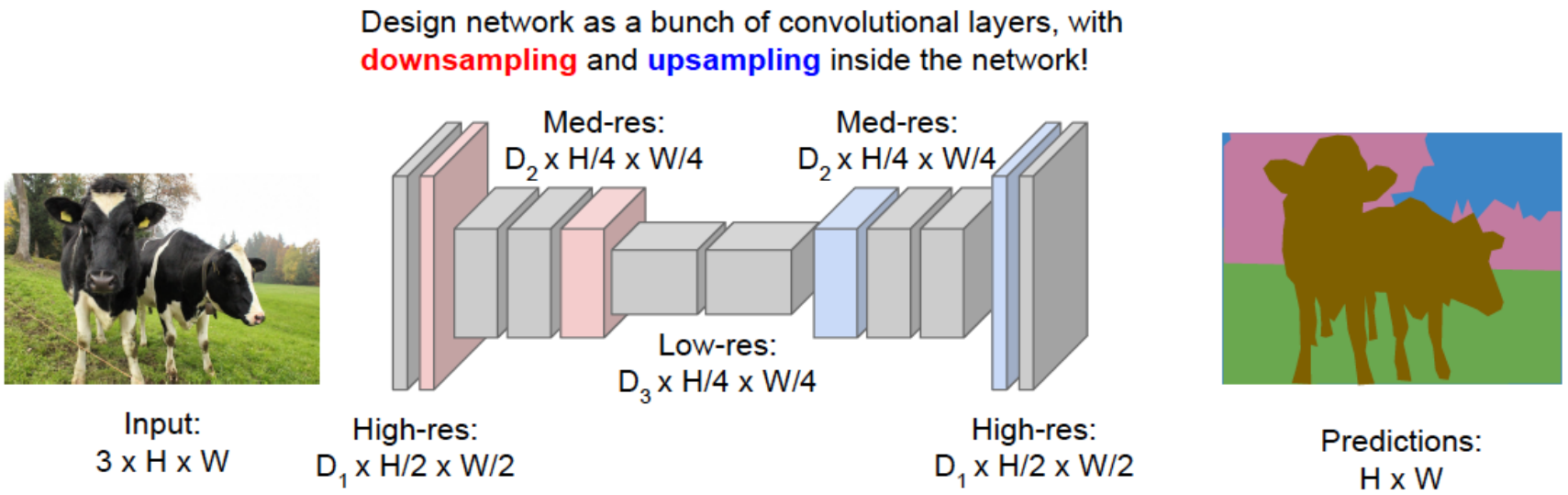
Network Design II: Spatial resolution

- Fully Convolutional Network [Long et al, CVPR 2015]
  - Upsampling: low-resolution, lack spatial details
  - Skip layer refinement



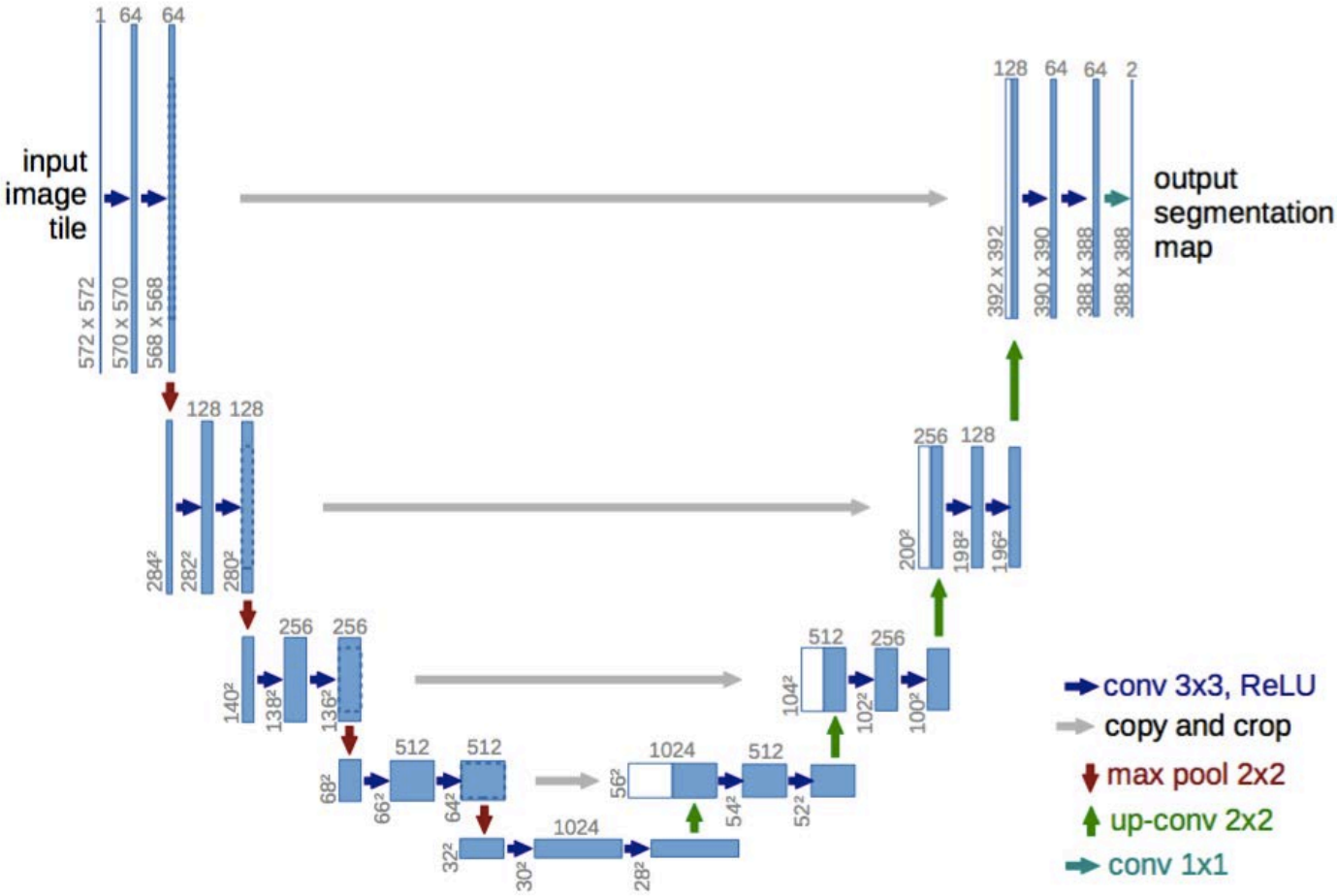
Network Design II: Spatial resolution

- General encoder-decoder architecture



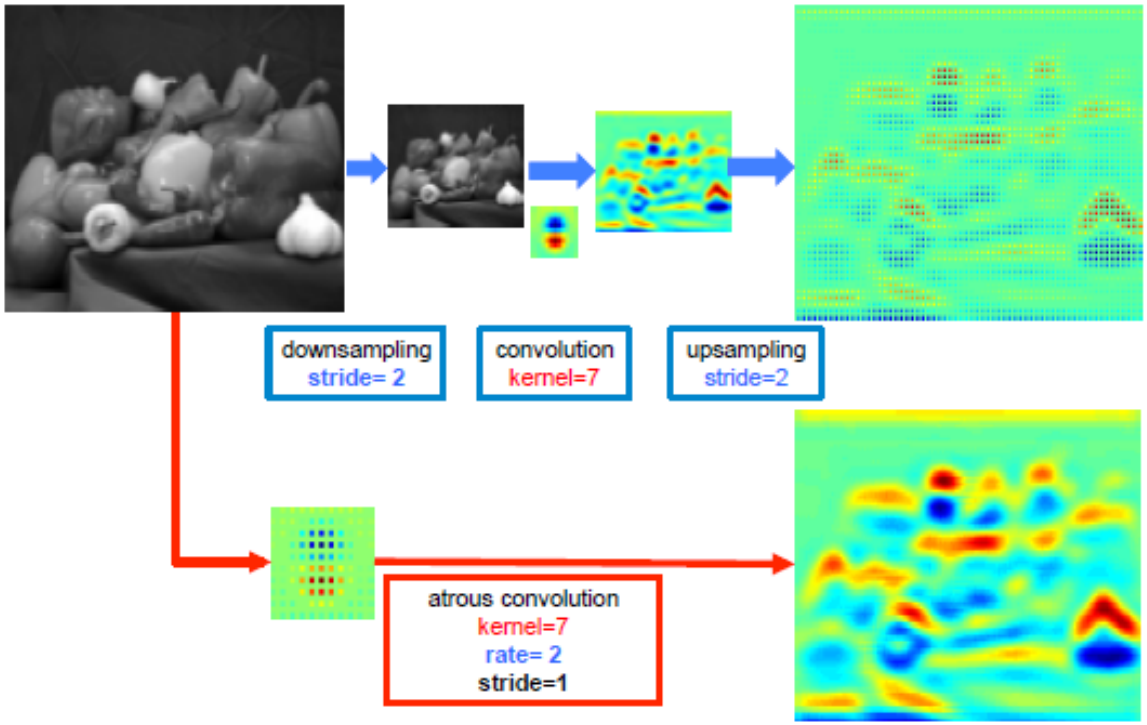
Network Design II: Spatial resolution

- U-Net [Ronneberger et al, MICCAI 2015]



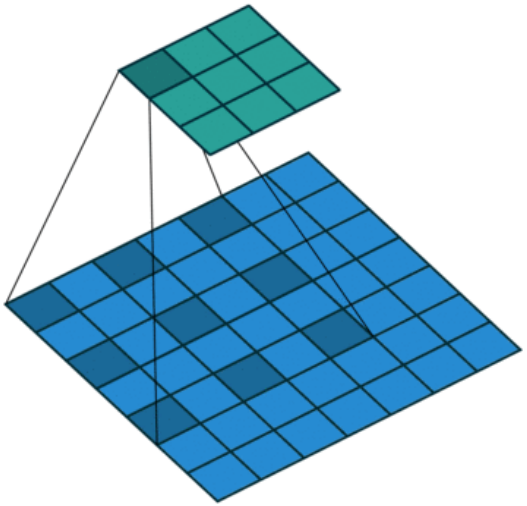
Network Design II: Spatial resolution

- Dilated Convolutional Network [Yu and Koltun, ICLR 2016]
  - Dense feature map without upsampling
  - ***Dilated (or Atrous) convolution***



Network Design II: Spatial resolution

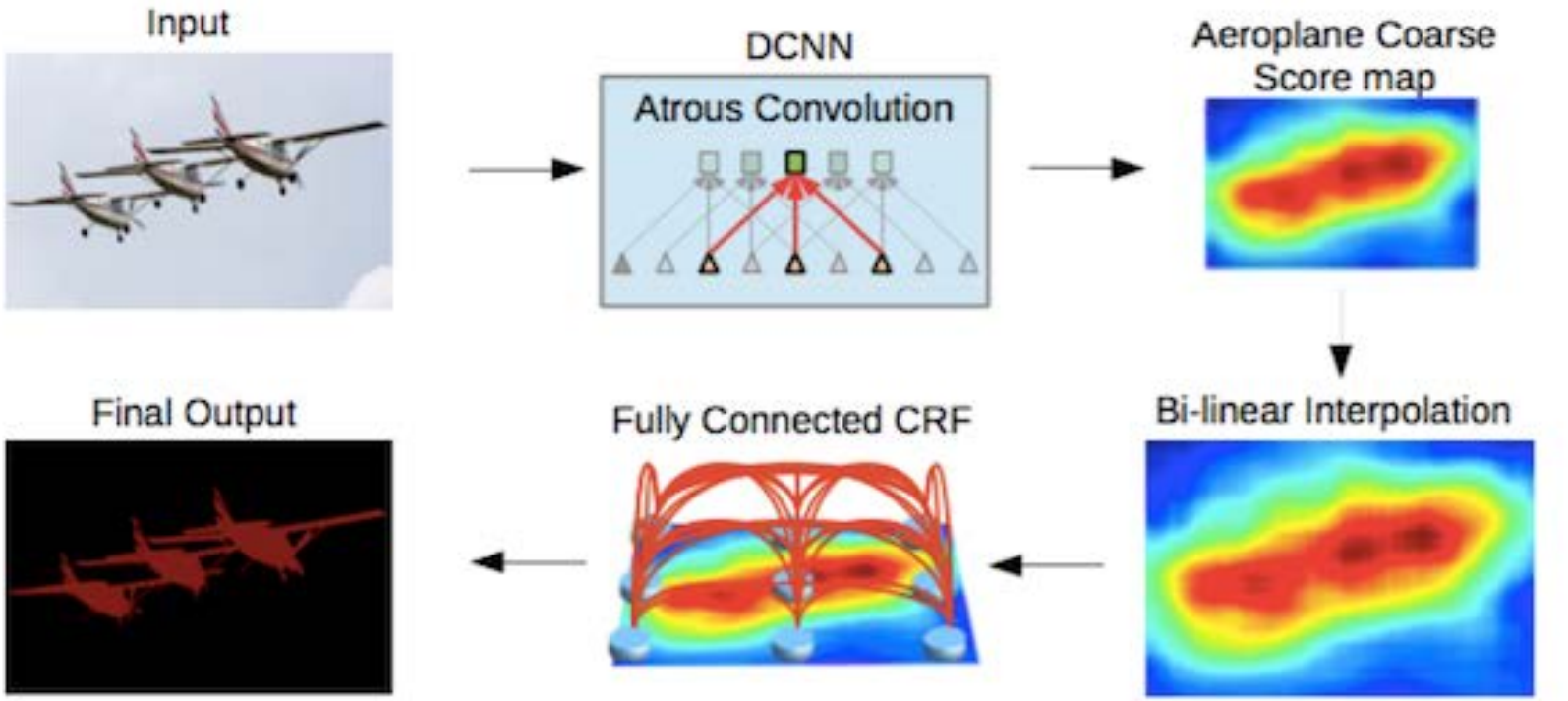
- Dilated Convolutional Network [Yu and Koltun, ICLR 2016]
  - Dense feature map without upsampling
  - ***Dilated (or Atrous) convolution***



$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k] w[k].$$

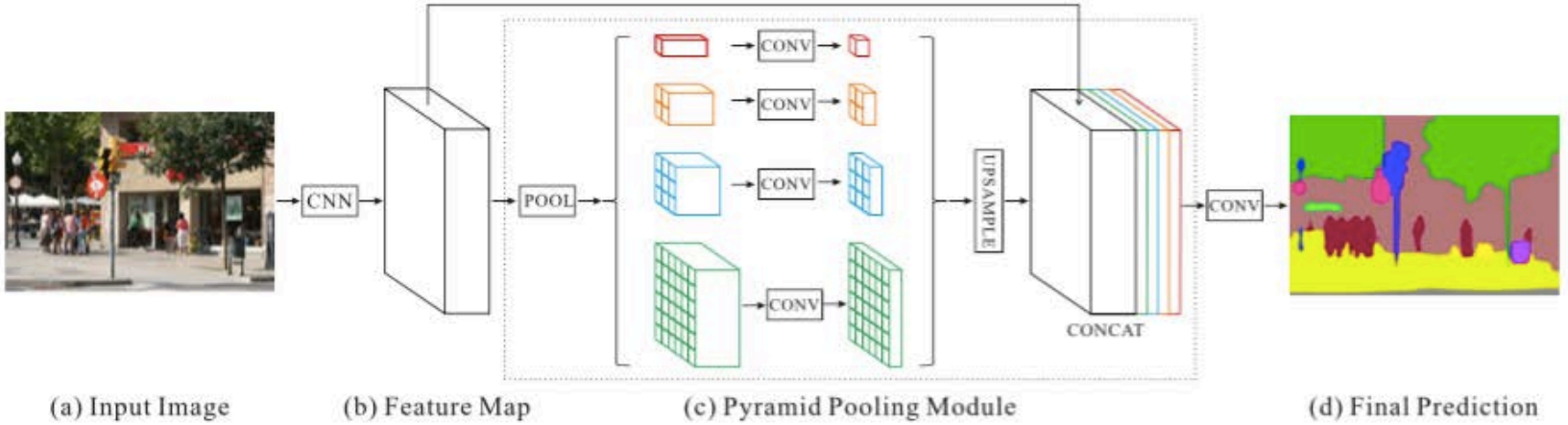
Network Design III: Multi-scale context

- DeepLab v1&v2
  - Post-processing with dense CRFs.

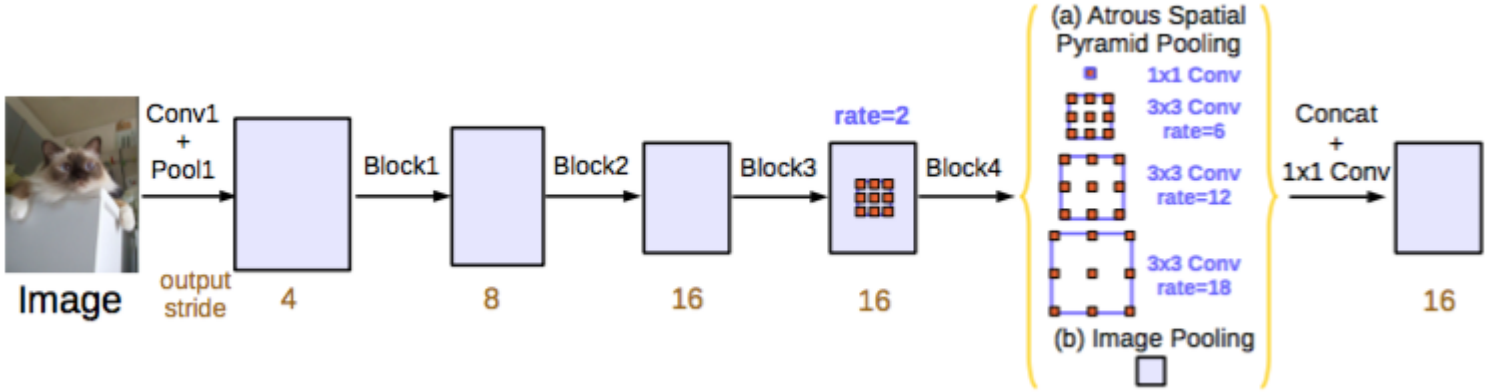


Network Design III: Multi-scale context

- PSPNet



- DeepLab v3



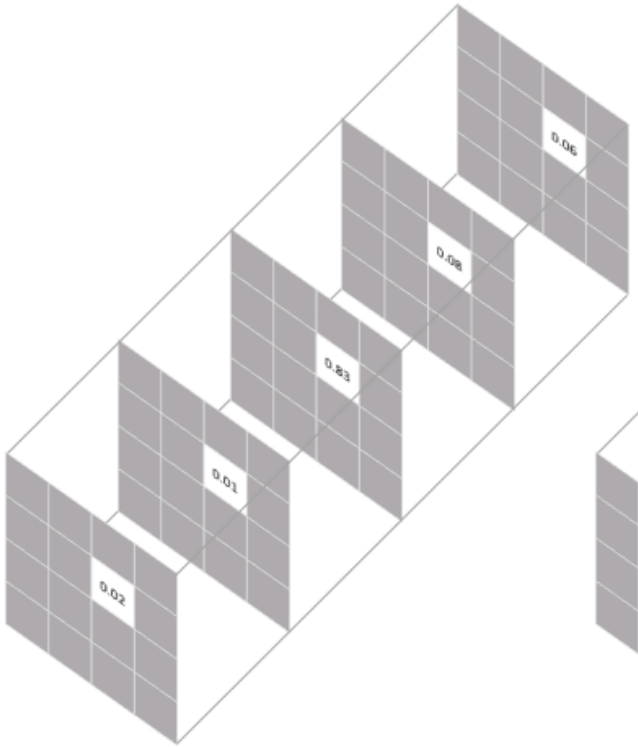
Outline

- What is semantic segmentation?
- Network architecture for semantic segmentation
  - Main idea for dense prediction
  - Upsampling operators
  - Modern network design
- Network training losses

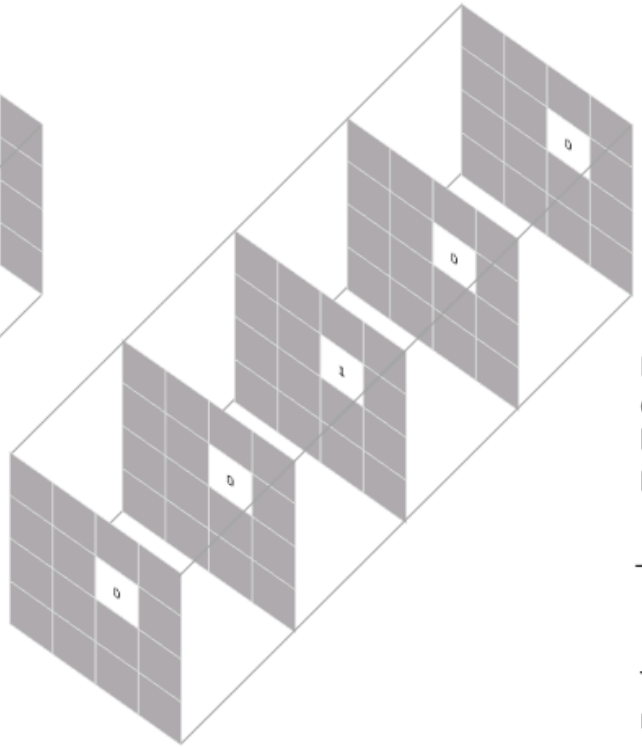
Acknowledgement: Feifei Li et al's cs231n notes

Semantic segmentation: loss function

- Pixel-wise loss



Prediction for a selected pixel



Target for the corresponding pixel

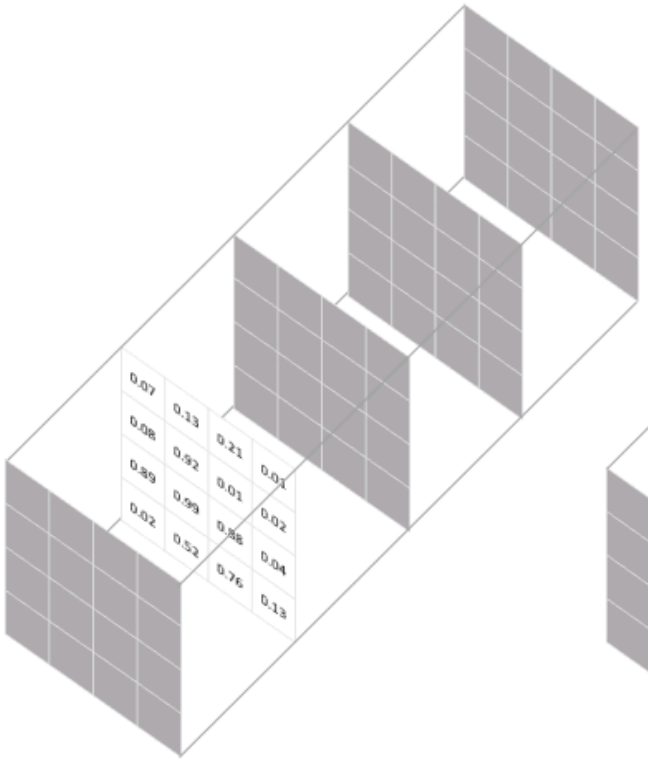
Pixel-wise loss is calculated as the log loss, summed over all possible classes

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

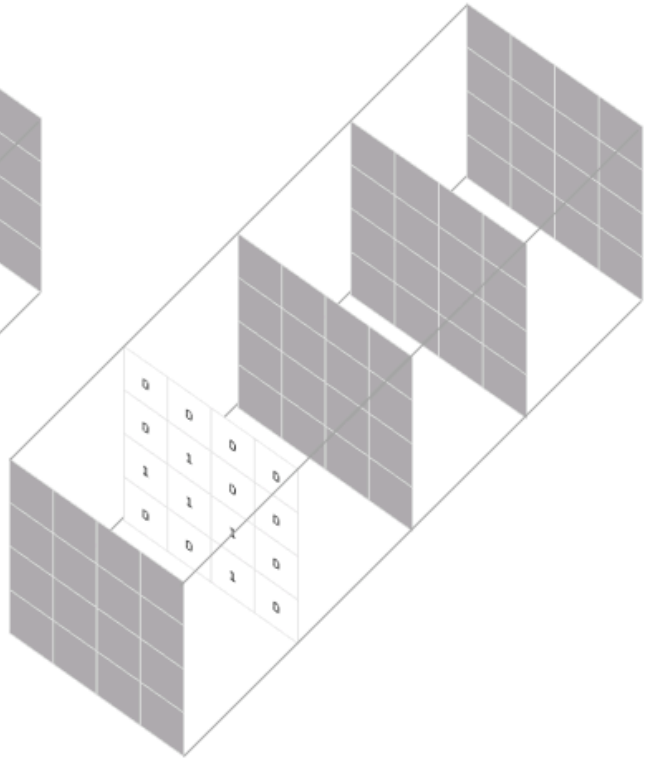
This scoring is repeated over all **pixels** and averaged

Semantic segmentation: loss function

- Region-based loss



Prediction for a selected class



Target for the corresponding class

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

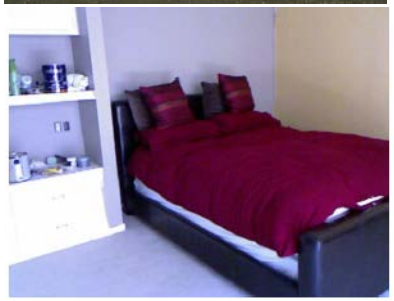
Soft Dice coefficient is calculated for each class mask

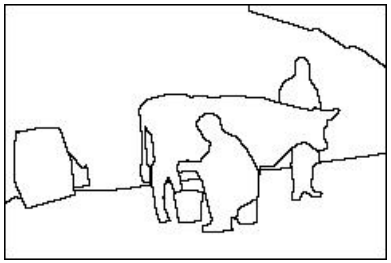
$$1 - \frac{2\sum_{pixels} y_{true}\, y_{pred}}{\sum_{pixels} y_{true}^2 + \sum_{pixels} y_{pred}^2}$$

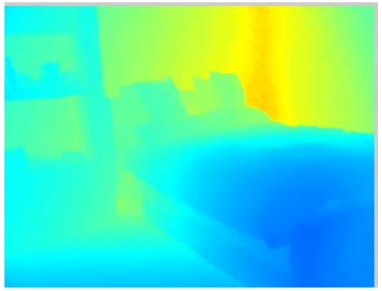
This scoring is repeated over all **classes** and averaged

Semantic Segmentation: Summary

- Pixel-wise annotation of images
  - An instance of scene understanding







Boundary

Depth

- Many questions remain unanswered
  - Training data?
  - Things vs. Stuff?
  - Boundary vs Region?

Semantic Segmentation: Summary

- Other research topics (not discussed)
  - *Low-level vision: superresolution, deblurring, inpainting, depth*
  - *Video: optical flow, action and activity recognition and detection*
  - *Volumetric/Multimodality: RGB-D images, medical imaging, etc.*

- Next time:
  - Instance detection and segmentation