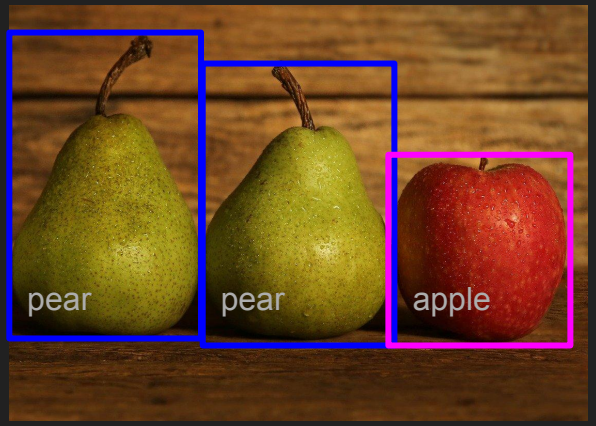
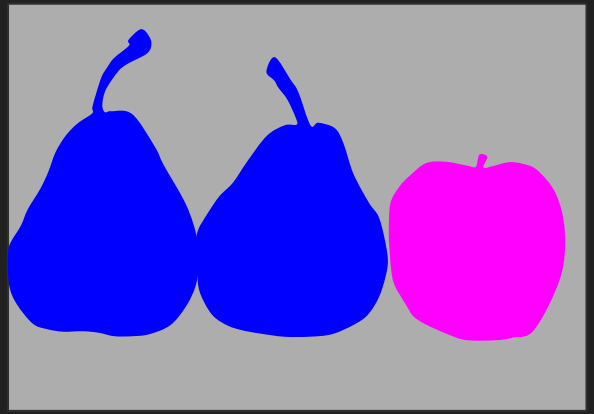
# Convolutional neural networks (convnets)

# Part 2 - Segmentation & Detection

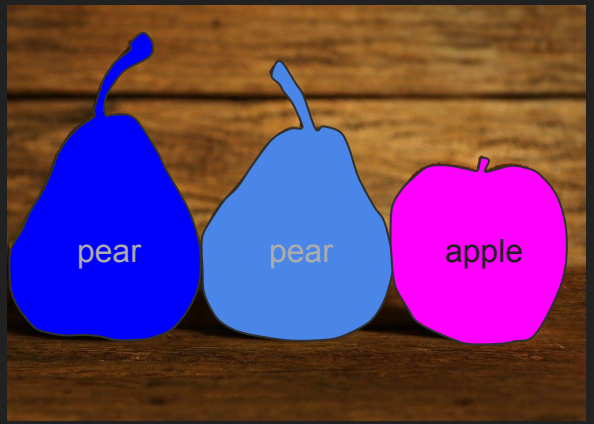
# Computer vision tasks




Object detection



Semantic segmentation
(e.g. pear, apple, background)

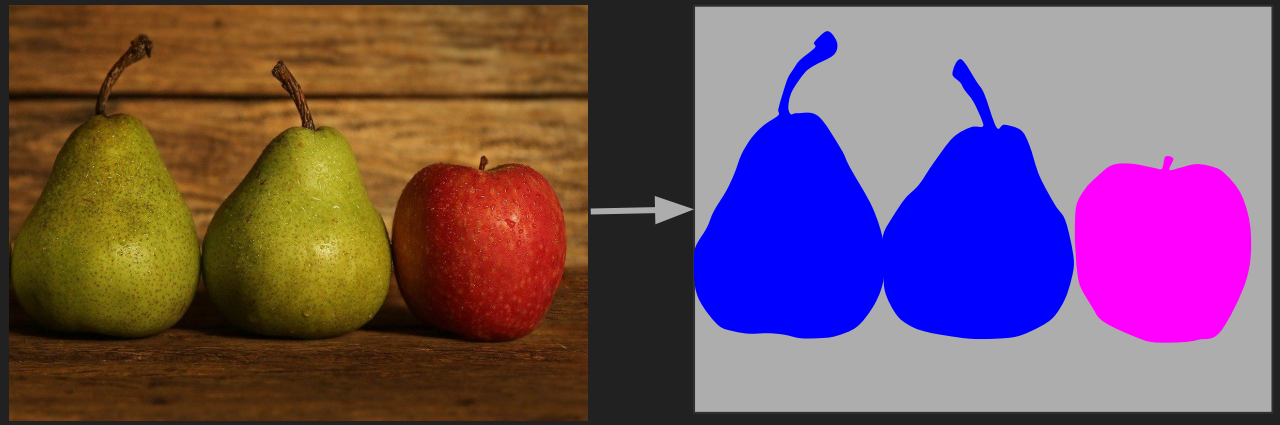



Instance
segmentation

Image by Tracy Lundgren from Pixabay

# Semantic segmentation

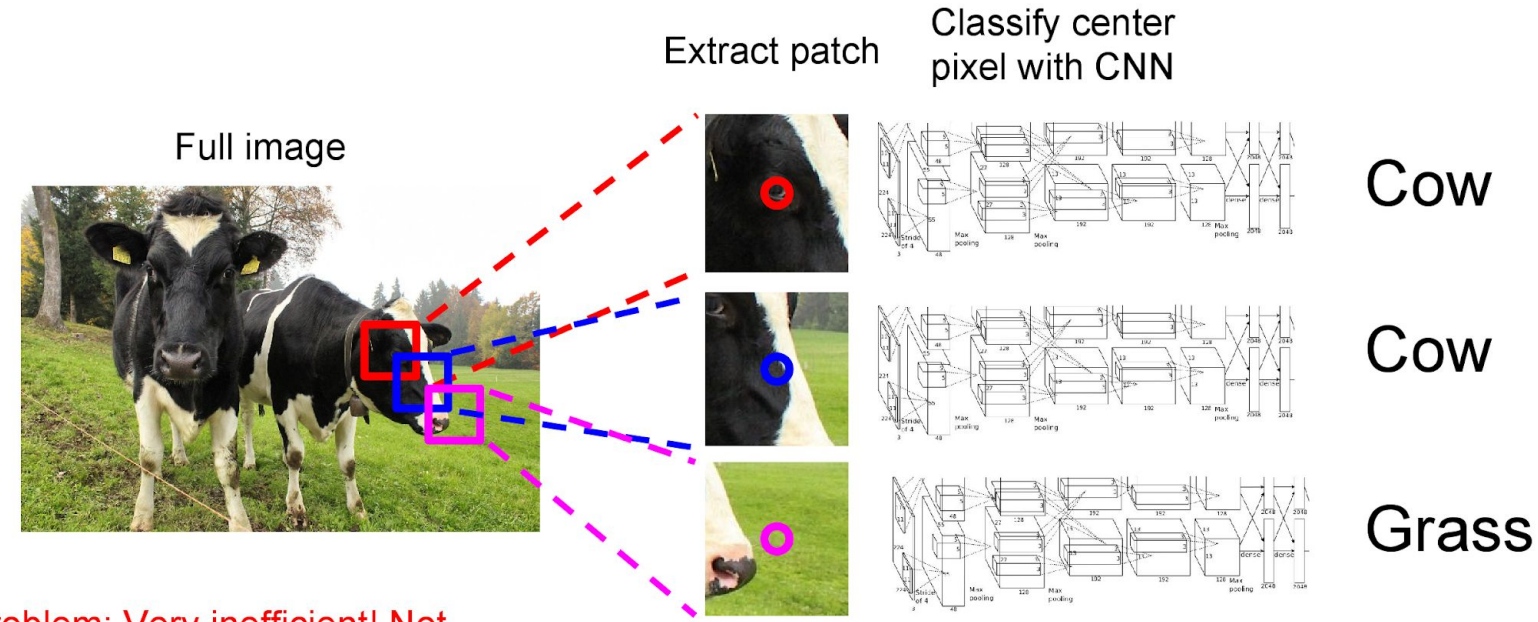
# Semantic segmentation



Semantic segmentation
(e.g. pear, apple, background)

- Assign a class to each pixel.
- No separation between instances of objects of same class.
- May be viewed as the classification problem on the pixel level.

# Semantic Segmentation Idea: Sliding Window

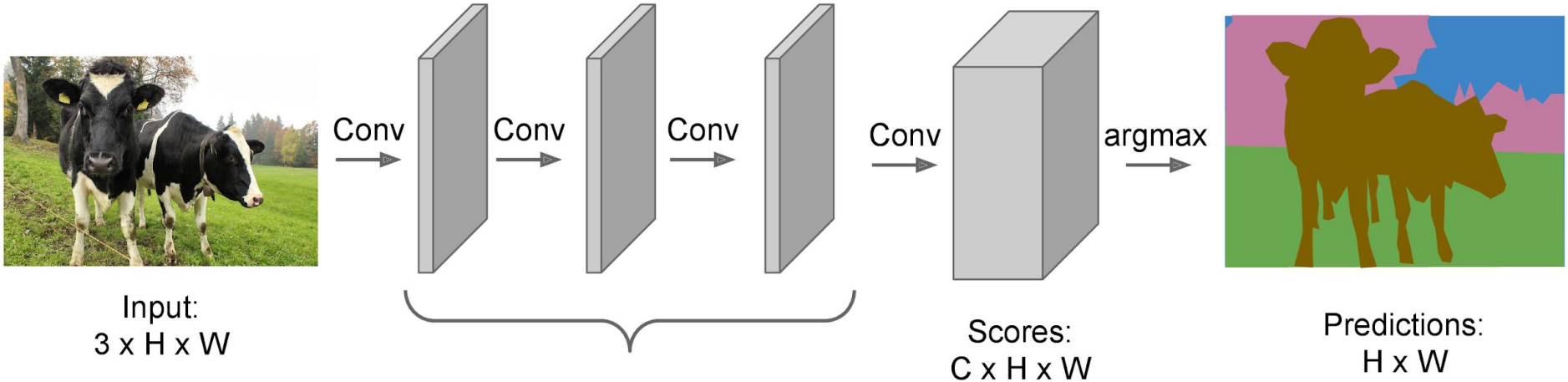



Problem: Very inefficient! Not reusing shared features between overlapping patches

*Slide taken from CS231n@stanford*

# Semantic Segmentation Idea: Fully Convolutional

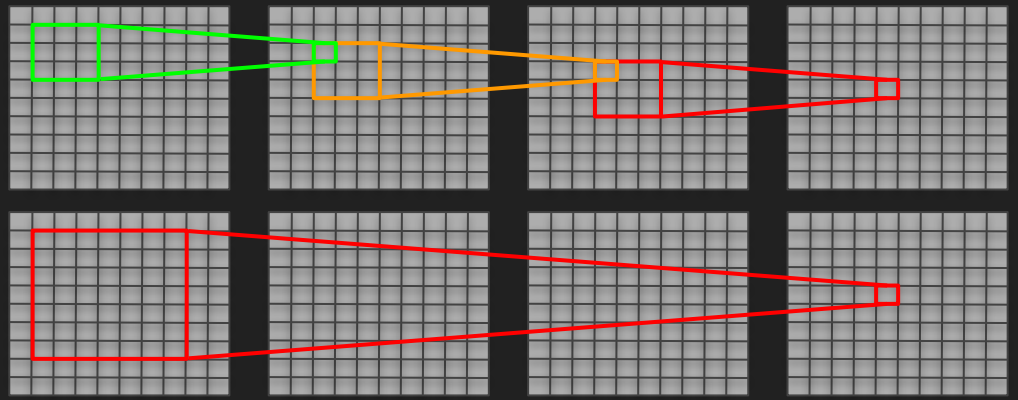
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



# Semantic segmentation - keeping same size

If wxh is unchanged:

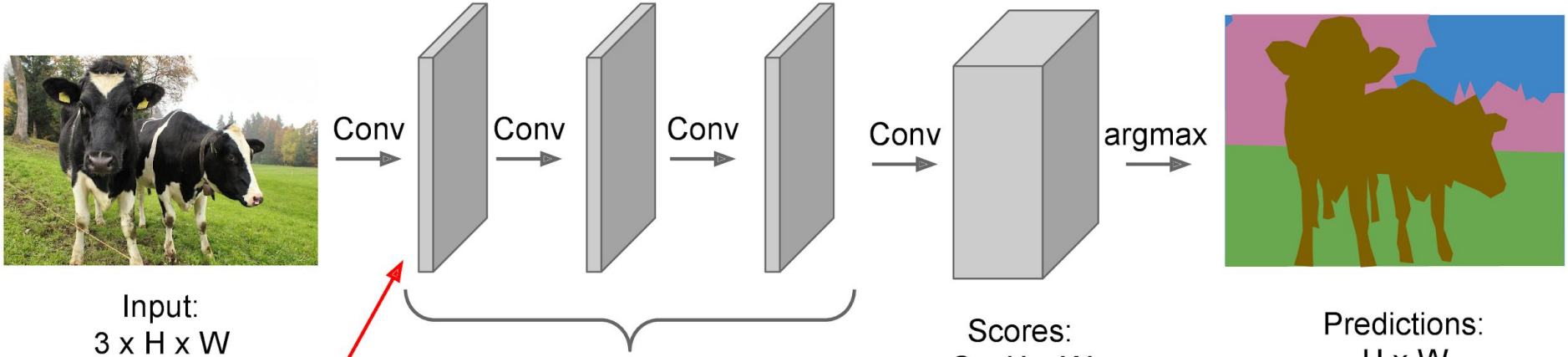
- Receptive field is small.
  - Note: dilation.



- If the number of filters(channels) is large, the memory and computation cost is large.

# Semantic Segmentation Idea: Fully Convolutional

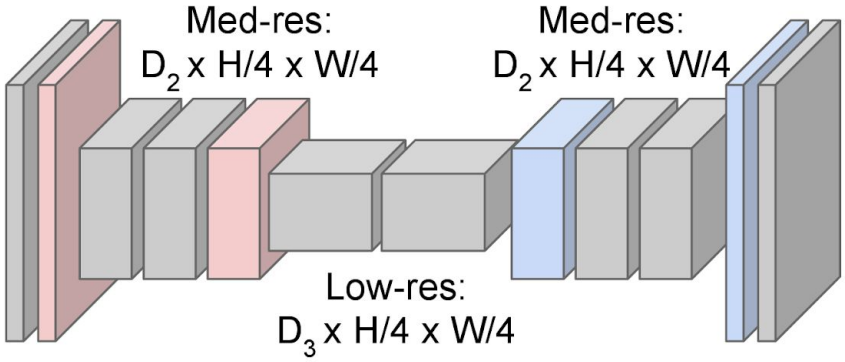
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



# Semantic Segmentation Idea: Fully Convolutional

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

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

*Slide taken from CS231n@stanford*

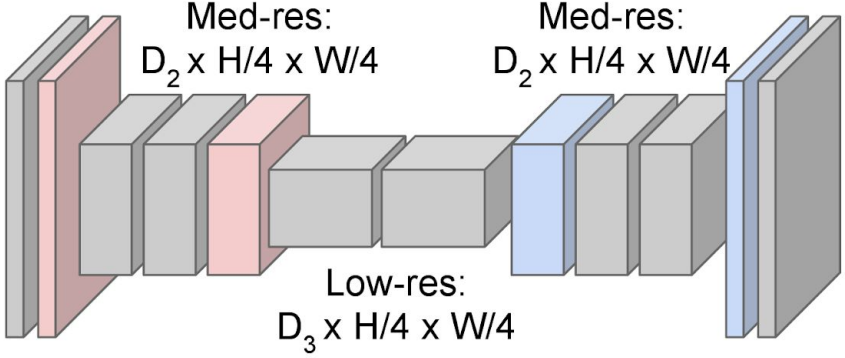
# Semantic Segmentation Idea: Fully Convolutional

**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

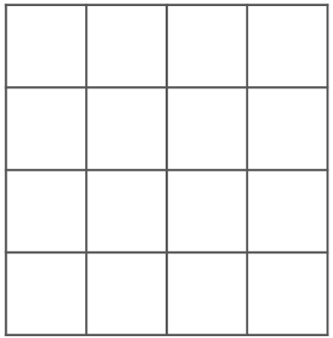
Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

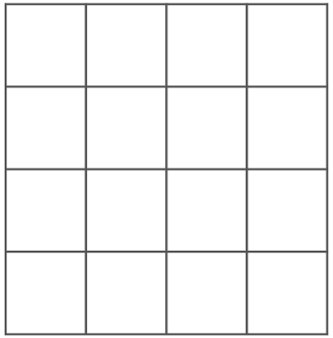
*Slide taken from CS231n@stanford*

# Learnable upsampling - explanation

Typical 3 x 3 convolution, stride 1 pad 1

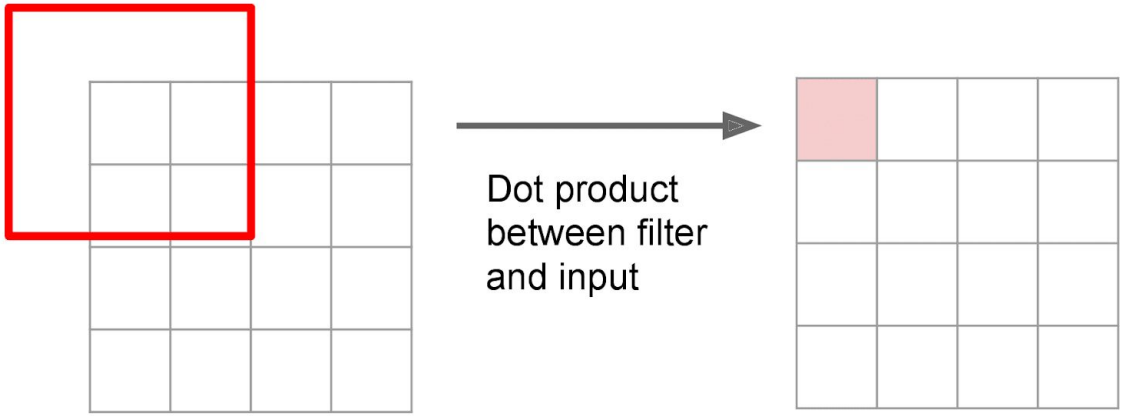


Input: 4 x 4

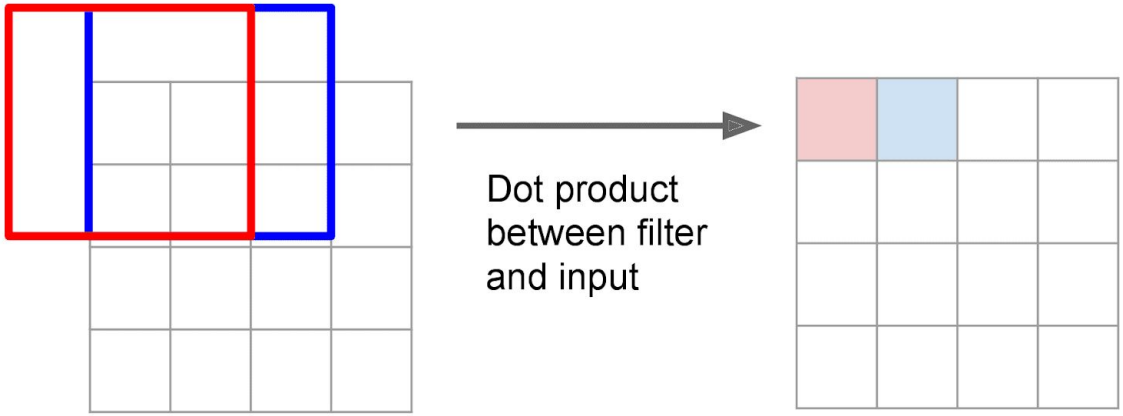


Output: 4 x 4

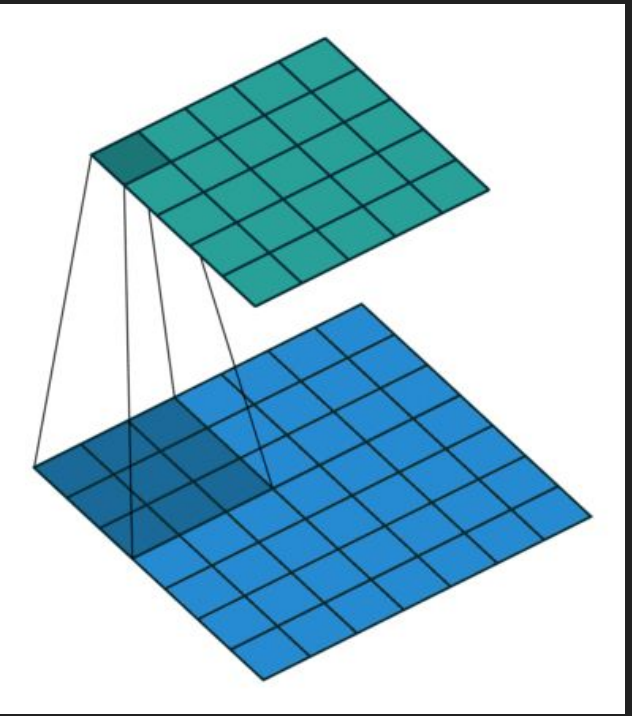
Typical 3 x 3 convolution, stride 1 pad 1

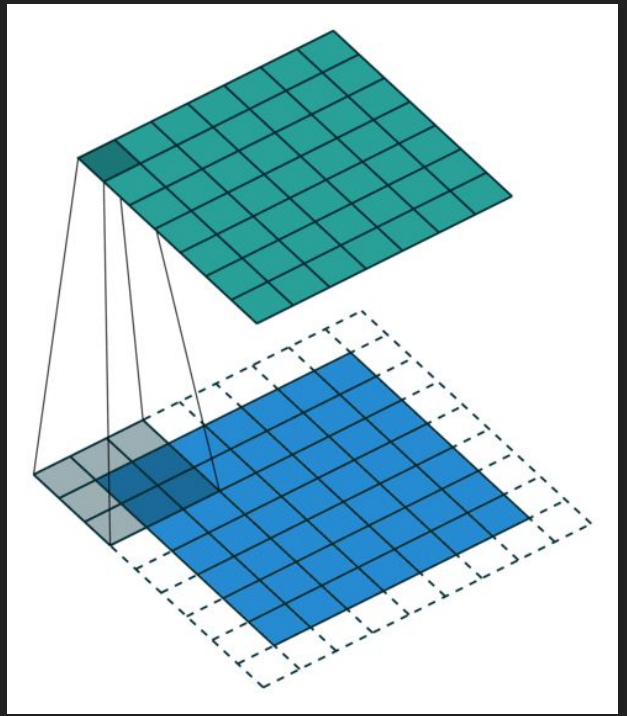


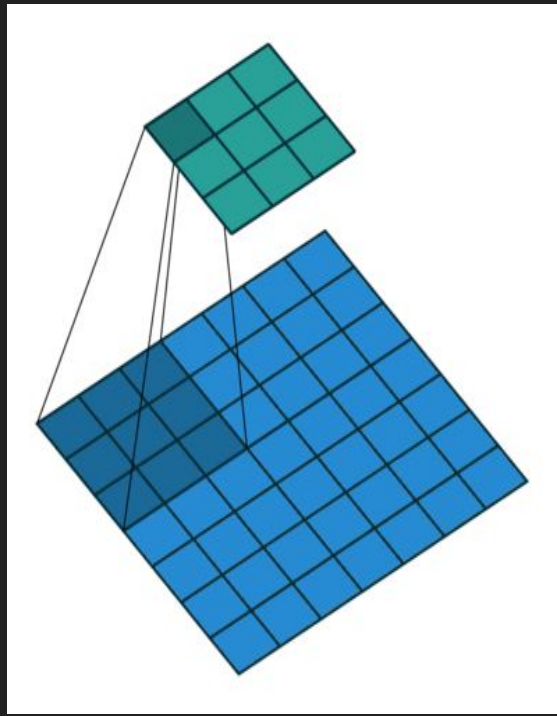
Typical 3 x 3 convolution, stride 1 pad 1



# Regular convolution

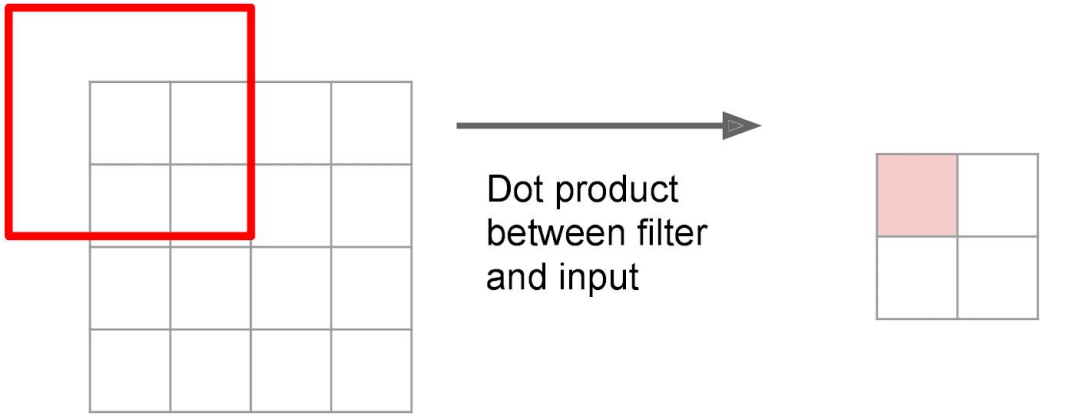






*Images generated by using* https://github.com/vdumoulin/conv_arithmetic

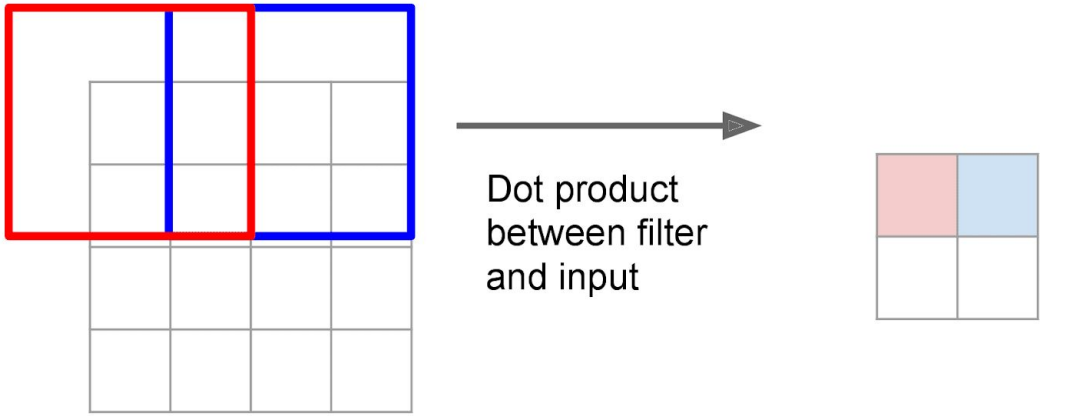
Typical 3 x 3 convolution, stride 2 pad 1




Input: 4 x 4

Output: 2 x 2

Typical 3 x 3 convolution, stride 2 pad 1

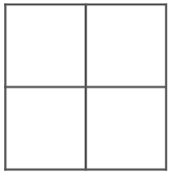



Input: 4 x 4

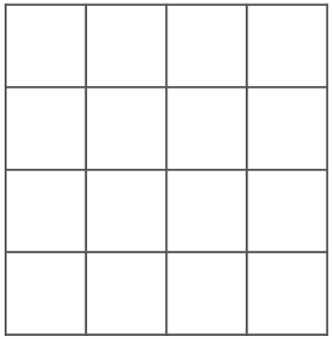
Output: 2 x 2

# Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1



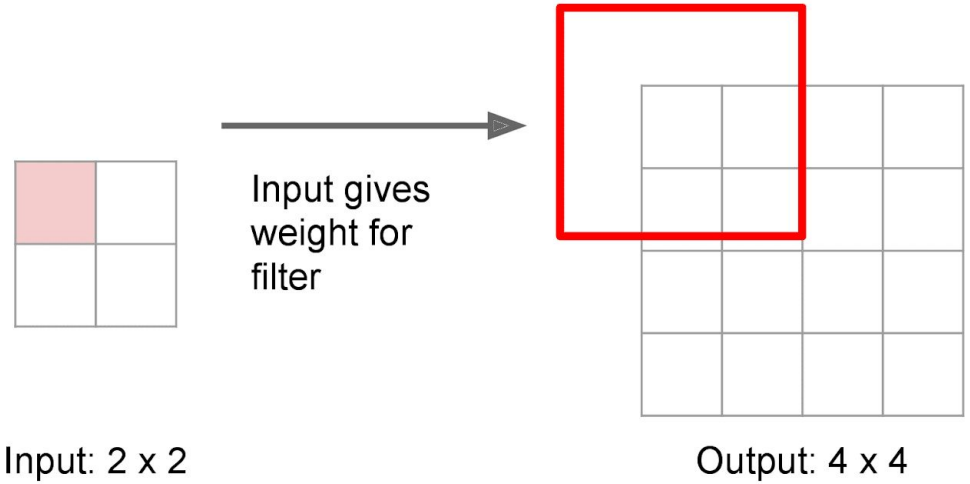
Input: 2 x 2



Output: 4 x 4

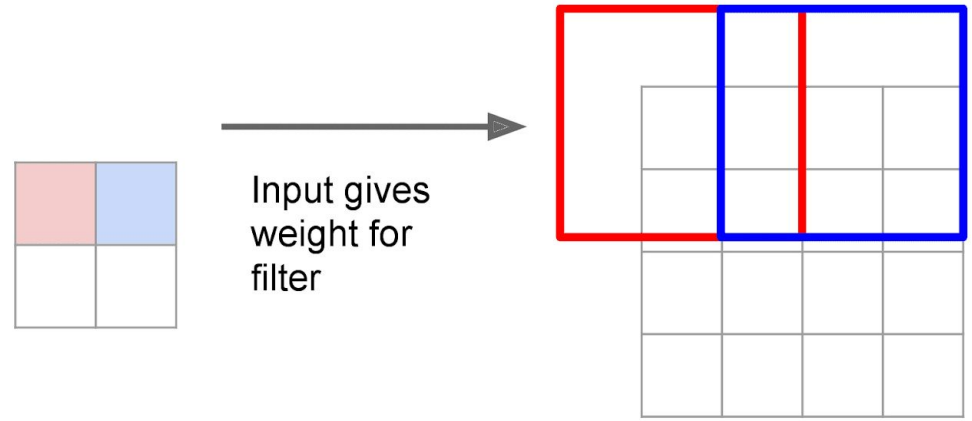
# Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1

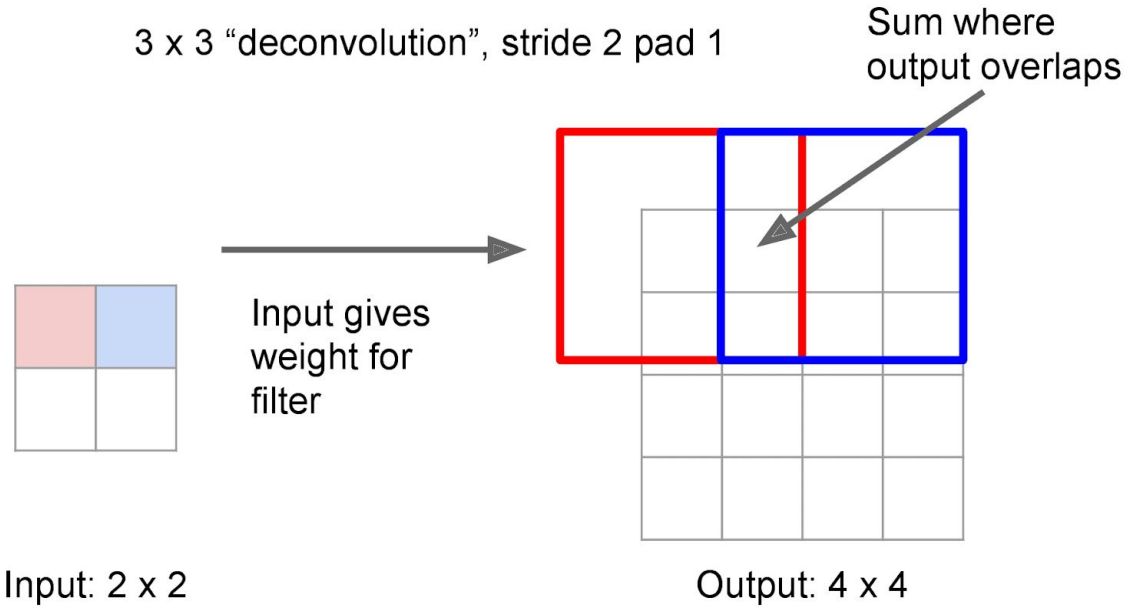


# Learnable Upsampling: “Deconvolution”

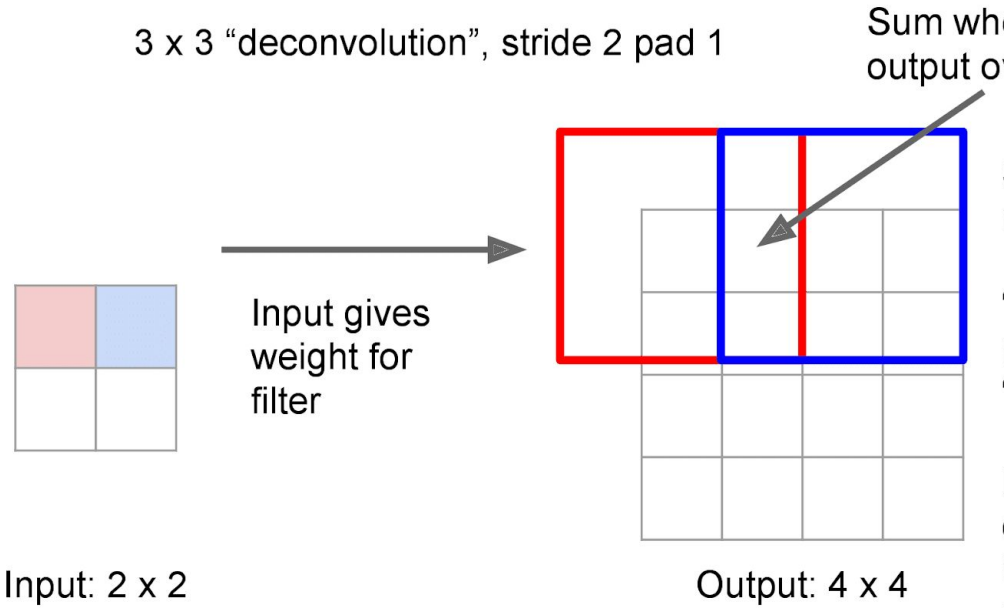
3 x 3 “deconvolution”, stride 2 pad 1



# Learnable Upsampling: “Deconvolution”



# Learnable Upsampling: “Deconvolution”




Same as backward pass for normal convolution!

“Deconvolution” is a bad name, already defined as “inverse of convolution”

**Better names:**
convolution transpose,
backward strided convolution,
1/2 strided convolution,
upconvolution

*Slide taken from CS231n@stanford*

slido

# How many input pixel an output pixel depends on?

ⓘ Start presenting to display the poll results on this slide.

# Learnable upsampling, alternative view

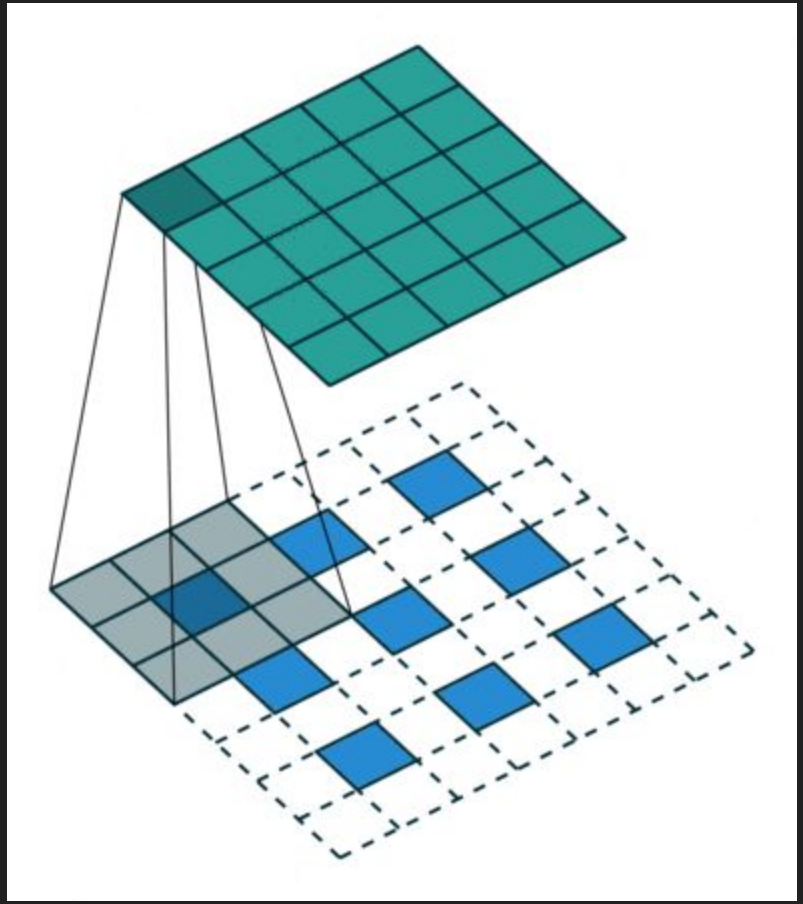
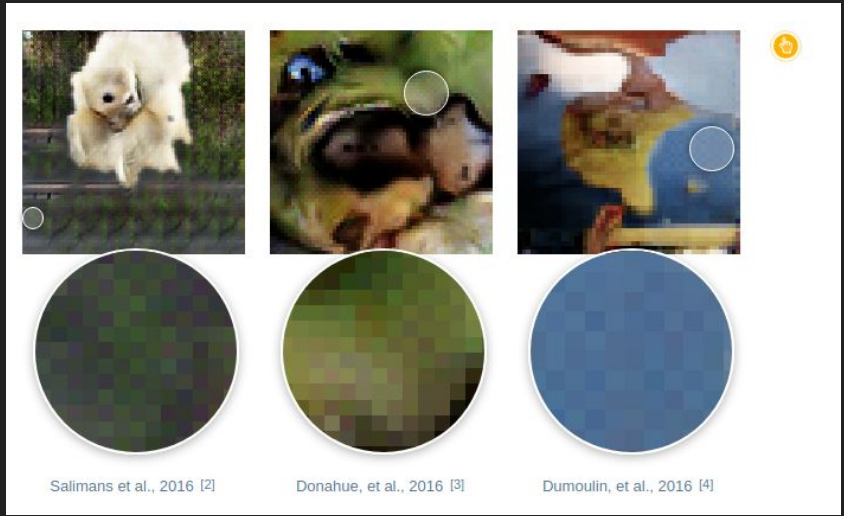
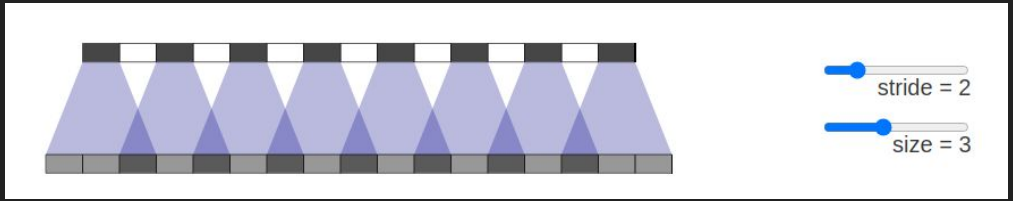
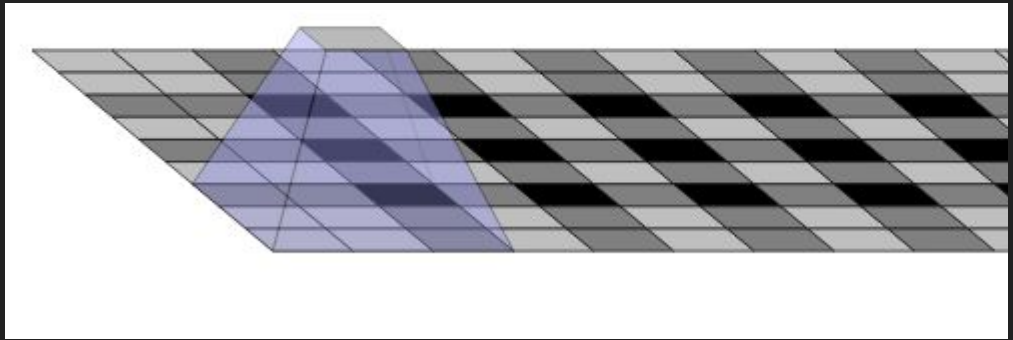


*Image generated by using https://github.com/vdumoulin/conv_arithmetic*

# Checkerboard pattern in images generated by convolution transpose



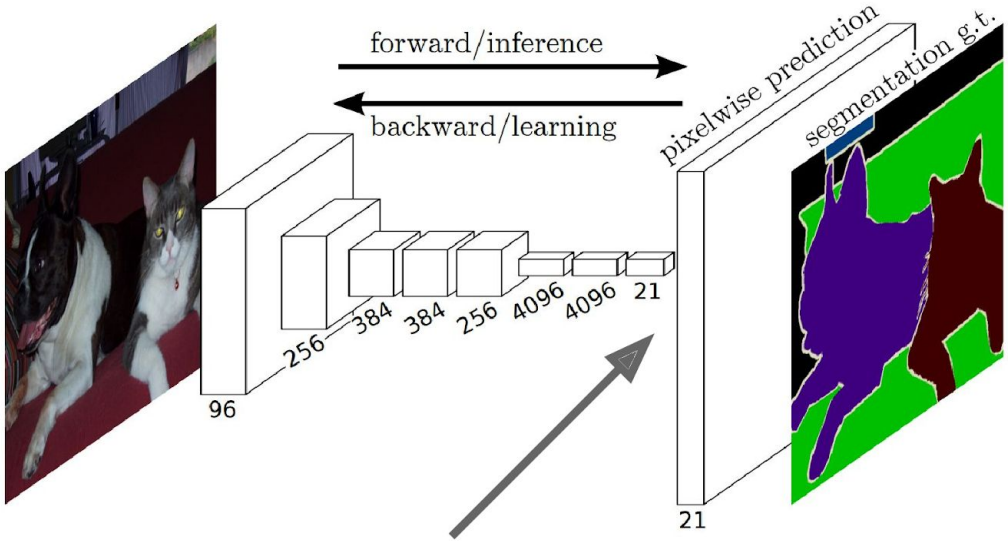



*https://distill.pub/2016/deconv-checkerboard/*

# Learnable upsampling - explanation end

# Semantic Segmentation: Upsampling

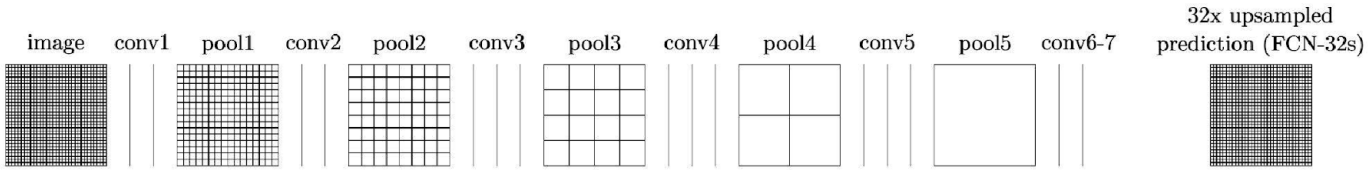



Learnable upsampling!

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

*Slide taken from CS231n@stanford*

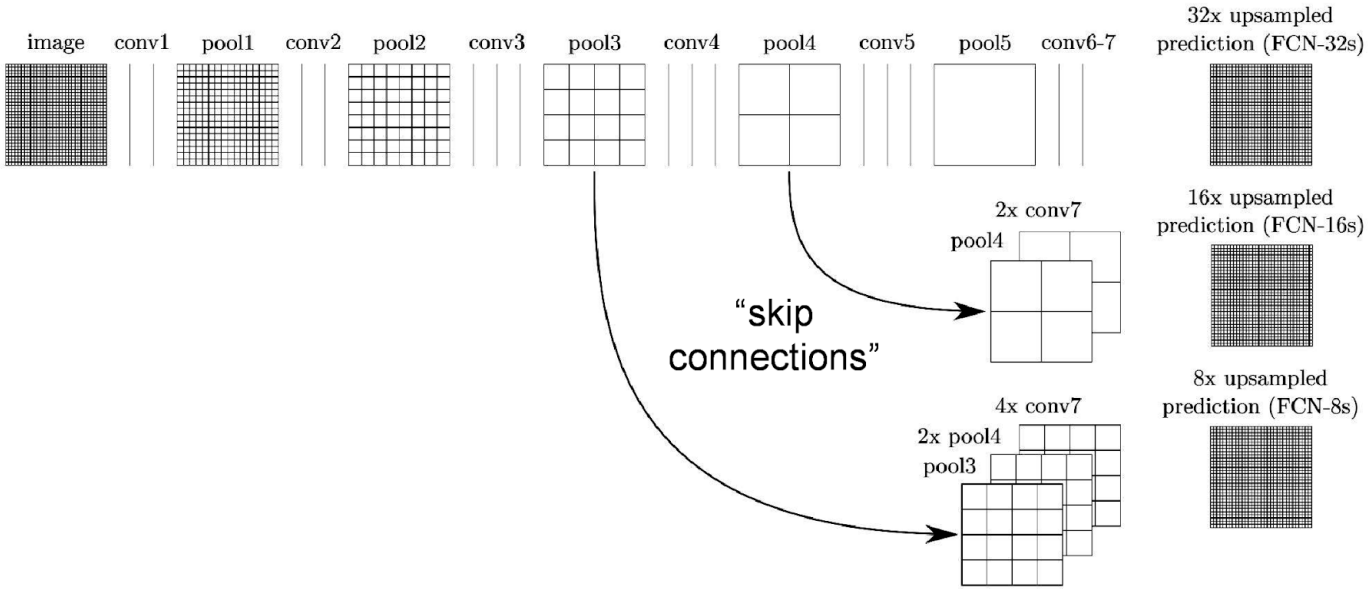
# Semantic Segmentation: Upsampling




Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

*Slide taken from CS231n@stanford*

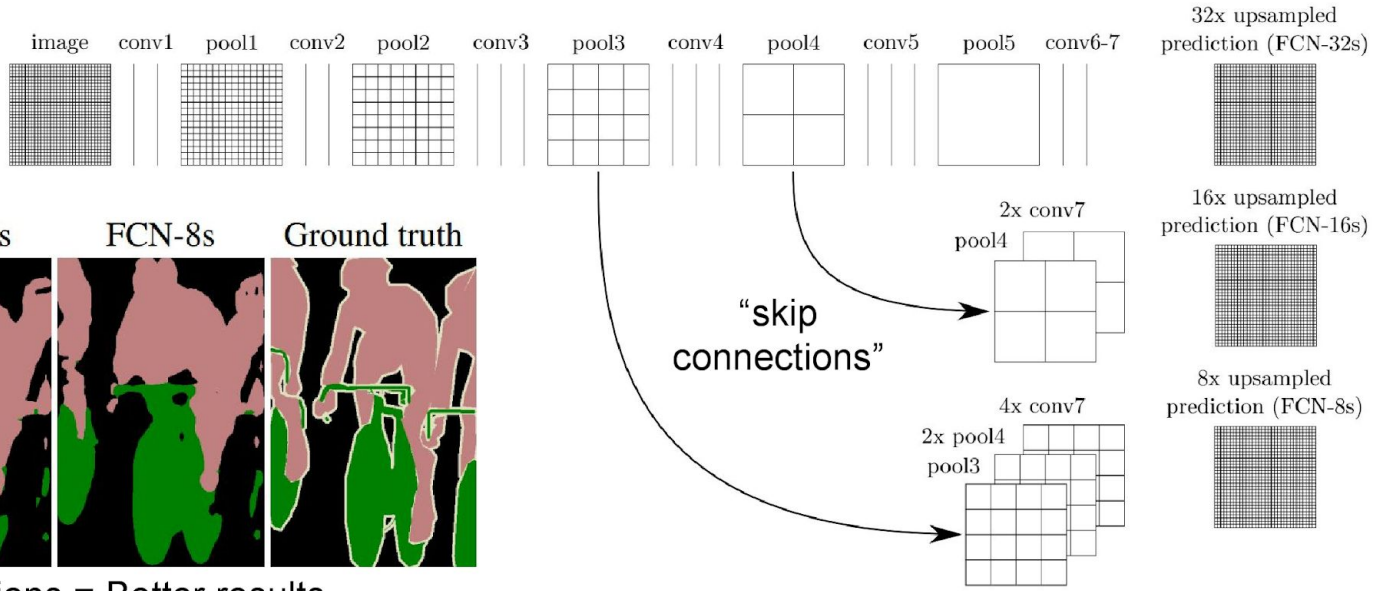
# Semantic Segmentation: Upsampling




Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

*Slide taken from CS231n@stanford*

# Semantic Segmentation: Upsampling





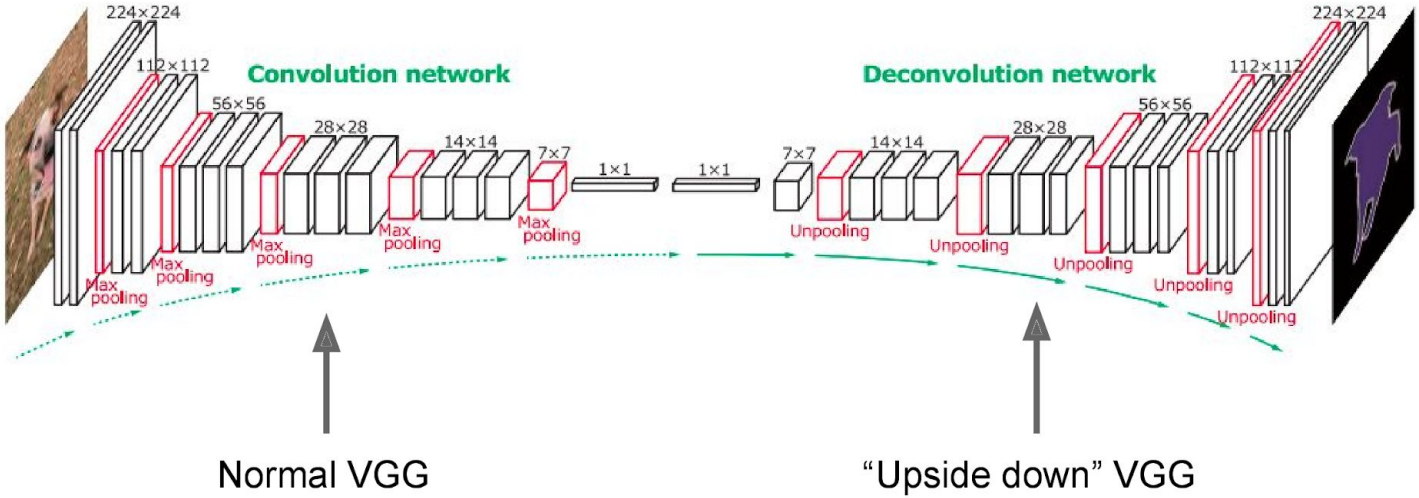


Skip connections = Better results

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

*Slide taken from CS231n@stanford*

# Semantic Segmentation: Upsampling



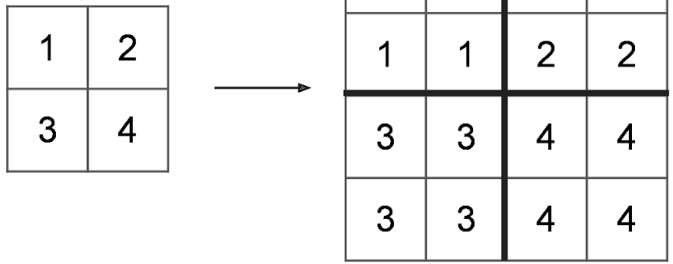

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

6 days of training on Titan X…

*Slide taken from CS231n@stanford*

# In-Network upsampling: “Unpooling”

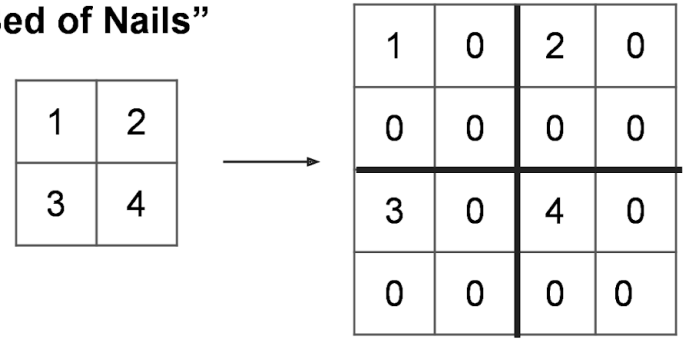
**Nearest Neighbor**




Input: 2 x 2

Output: 4 x 4

**“Bed of Nails”**




Input: 2 x 2

Output: 4 x 4

*Slide taken from CS231n@stanford*

# In-Network upsampling: “Max Unpooling”

**Max Pooling**
Remember which element was max!

| | | | |
|---|---|---|---|
| 1 | 2 | 6 | 3 |
| 3 | 5 | 2 | 1 |
| 1 | 2 | 2 | 1 |
| 7 | 3 | 4 | 8 |

Input: 4 x 4

| | |
|---|---|
| 5 | 6 |
| 7 | 8 |

Output: 2 x 2

. . .

Rest of the network

**Max Unpooling**
Use positions from pooling layer

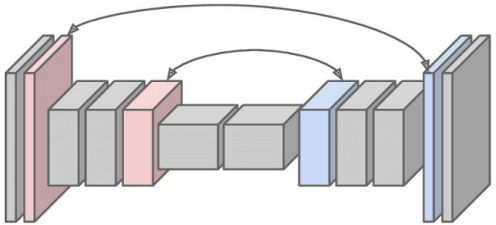
| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

Input: 2 x 2

| | | | |
|---|---|---|---|
| 0 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 4 |

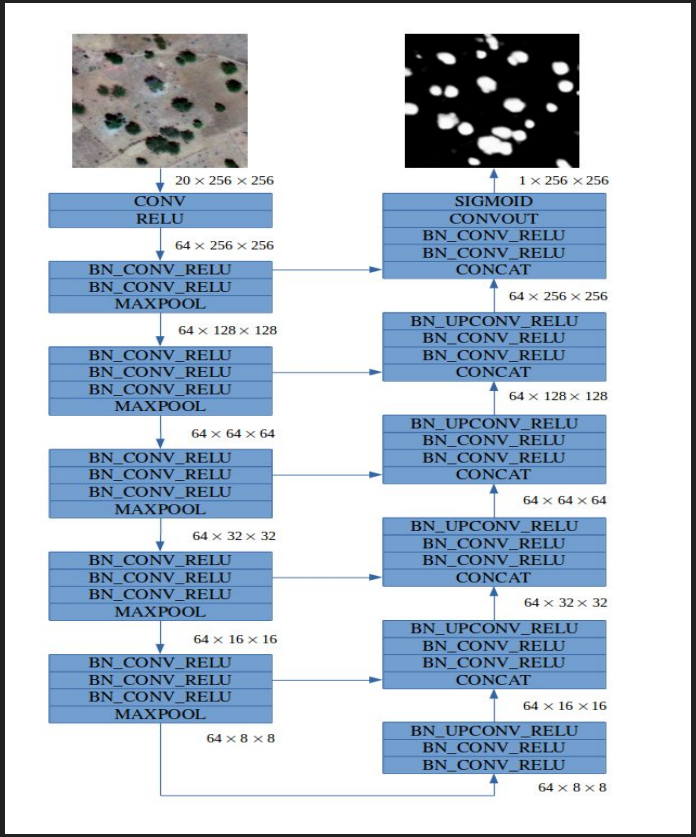
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



*Slide taken from CS231n@stanford*

# Semantic segmentation, take-home-message

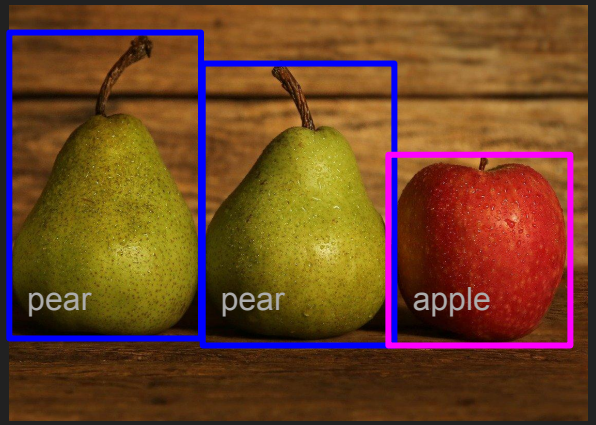



- Use U-net as a default starting point for semantic segmentation (sample architecture on the left)
- Fully convolutional solution (no region proposals)

*https://deepsense.io/deep-learning-for-satellite-imagery-via-image-segmentation/*

# Object detection

# Object detection

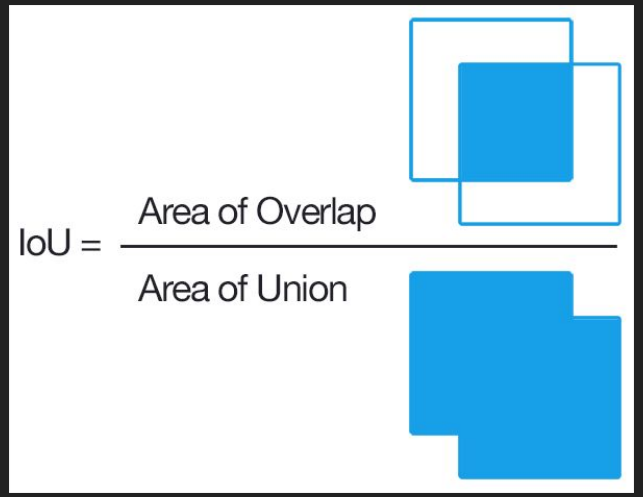


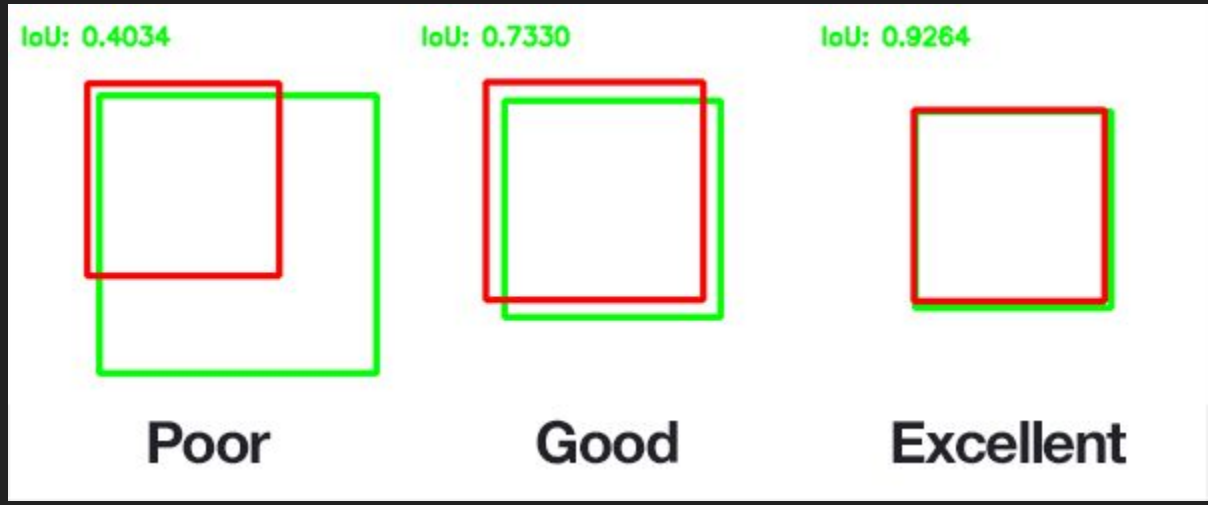

Object detection

- Localize all objects with bounding boxes
- Axis aligned
- Class for each box.

Image by Tracy Lundgren from Pixabay

# Intersection over union (IoU)




https://en.wikipedia.org/wiki/Jaccard_index

# Single object case

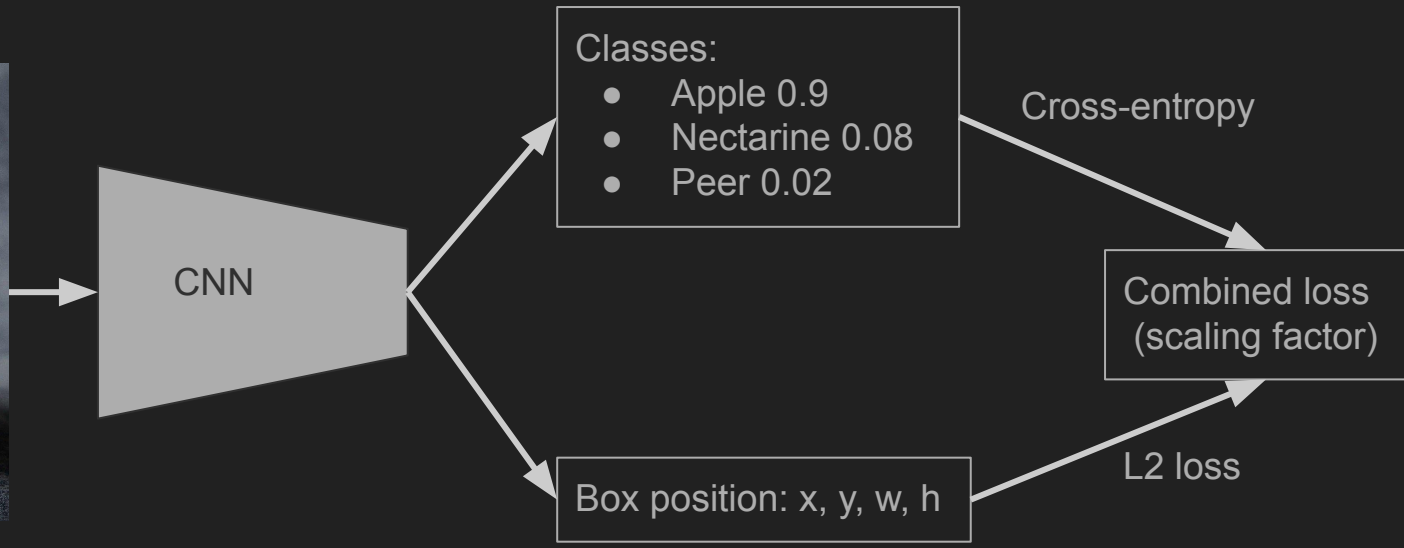
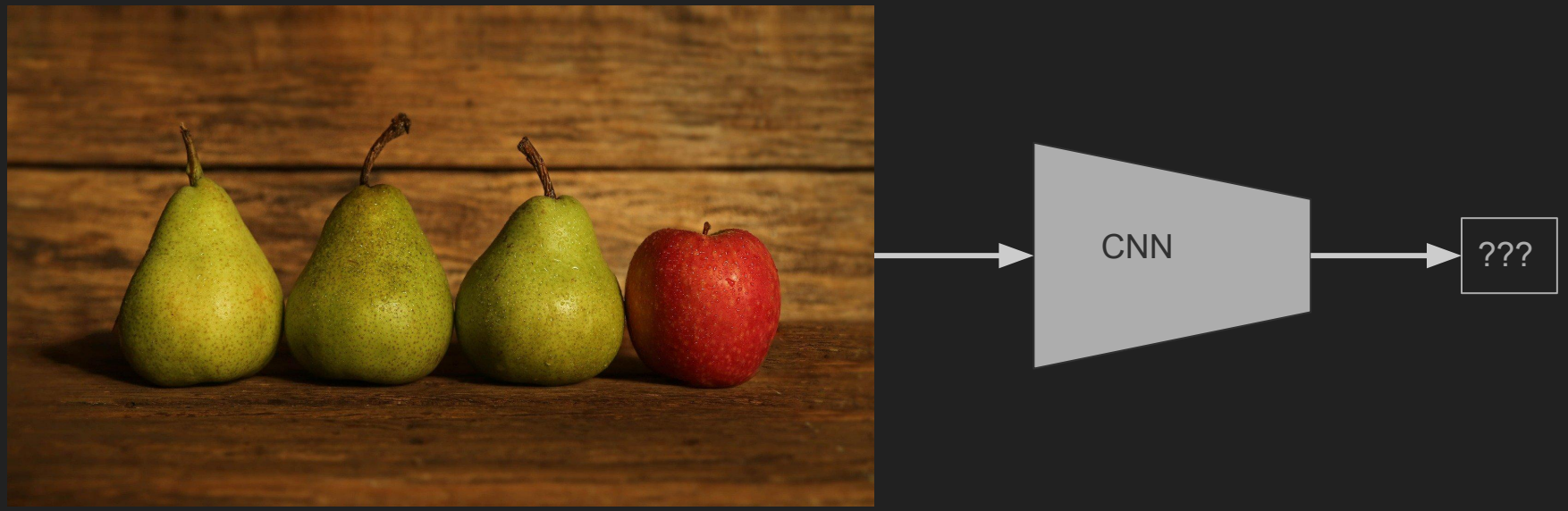
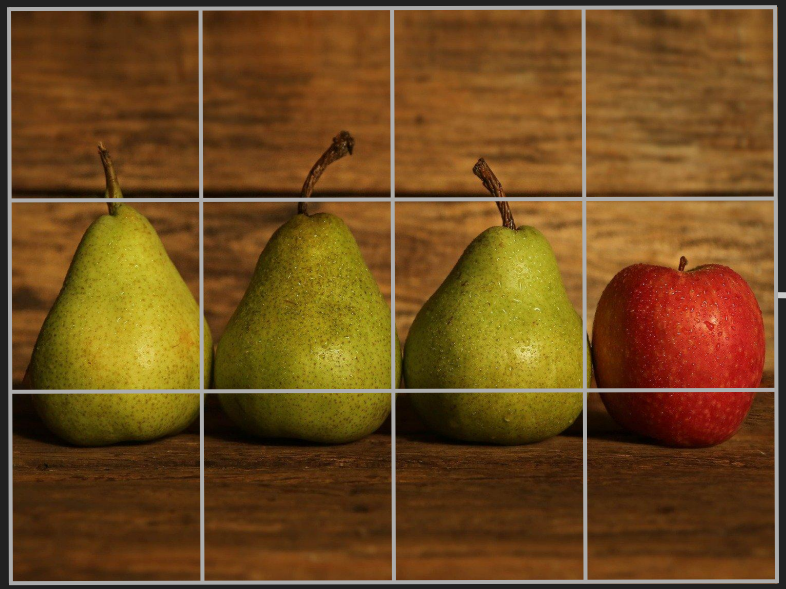



*Image by Michal Jarmoluk from Pixabay*

# Multi object case



# Idea 1: grid of predictions

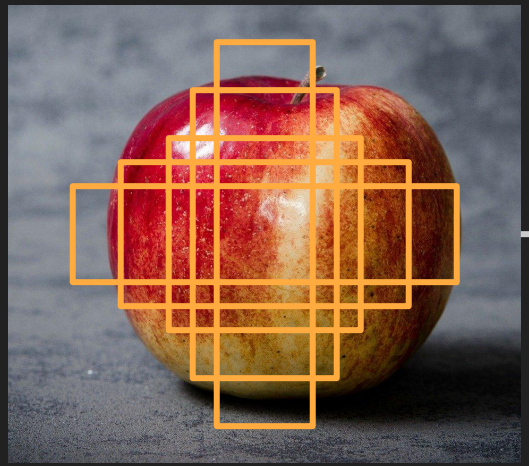


CNN

| | | | |
|---|---|---|---|
| Class, Box | Class, Box | Class, Box | Class, Box |
| Class, Box | Class, Box | Class, Box | Class, Box |
| Class, Box | Class, Box | Class, Box | Class, Box |

# Idea 2: Anchor boxes

- Consider a set of predefined rectangles: **anchor boxes** (thousands or more).
- Always the same set of boxes, regardless of the image.

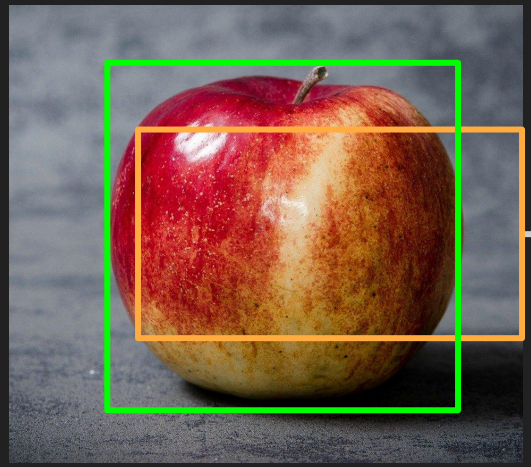


CNN

For each anchor box:

- Classify into C+1 classes (C classes + background)
- Refinement: scale+offset

# Idea 2: anchor boxes

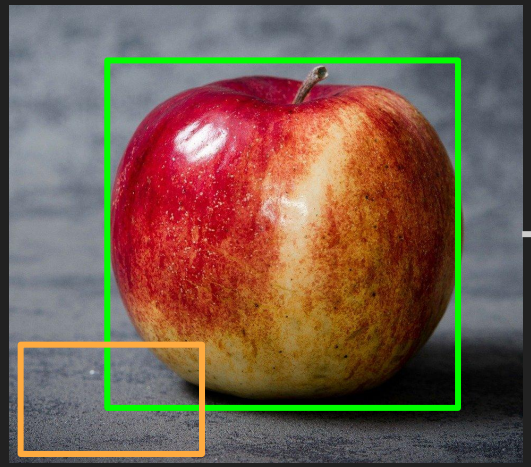


CNN

Class: apple

Refinement:

- Scale width = 0.915
- Scale height = 1.66
- Offset x = -0.12
- Offset y = 0.02



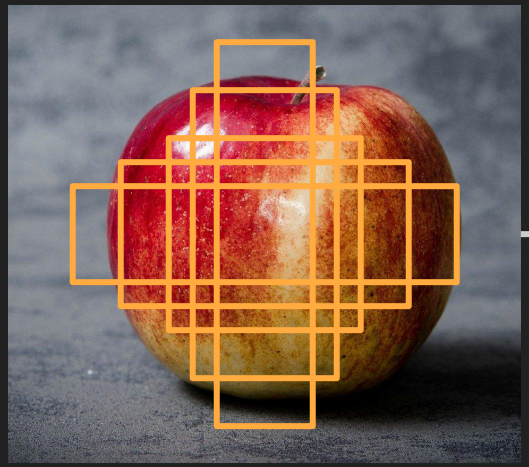
CNN

Class: background
(because IoU < threshold)

Refinement:

- Scale width = None
- Scale height = None
- Offset x = None
- Offset y = None

# Idea 2: Anchor boxes

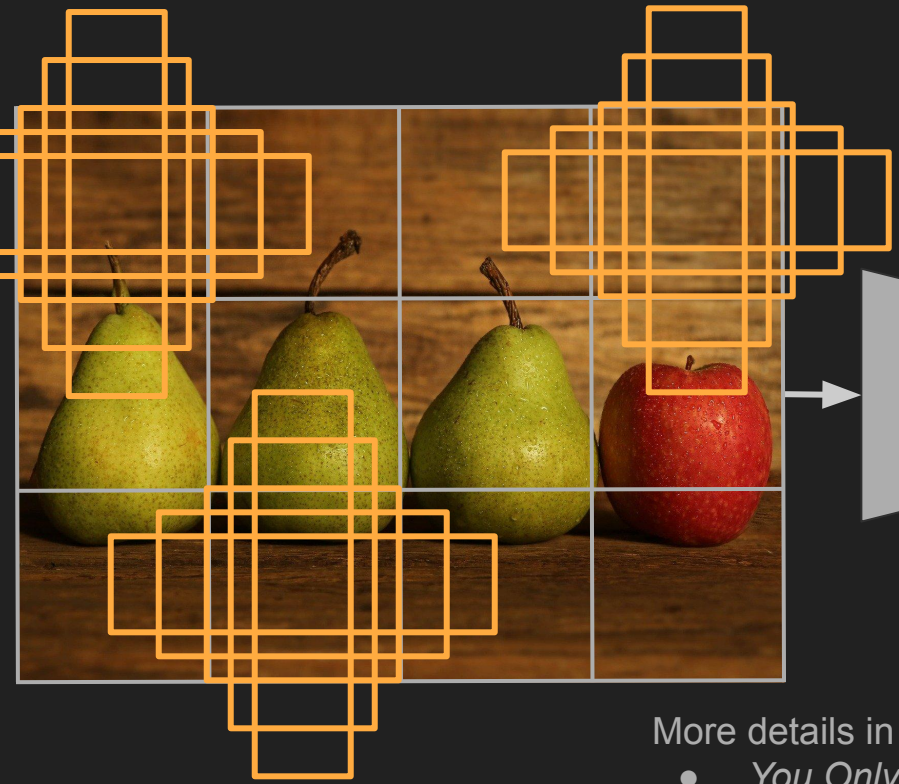


CNN

For each anchor box:

- Classify into C+1 classes (C classes + background)
- Refinement: scale+offset

# Idea 1 + Idea 2 = Single Stage Detector



CNN

| | | | |
|---|---|---|---|
| For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform |
| For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform |
| For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform | For each anchor:<br>• class,<br>• transform |

More details in articles:

- *You Only Look Once: Unified, Real-Time Object Detection*
- *SSD: Single Shot MultiBox Detector*
- *Focal Loss for Dense Object Detection*

slido

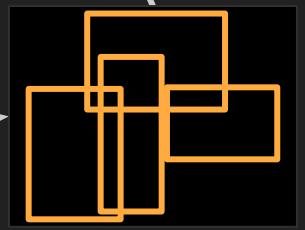
**Assume:5x5 grid,10 anchor boxes,C=3 classes (apples, pears, oranges)What is the size of the output tensor for a single input image? Remember about the class representing background (not detection).**

ⓘ Start presenting to display the poll results on this slide.

# 2-stage detection (sketch)

Region of interest pooling

Region proposal network



Proposals

CNN: Classify & refine

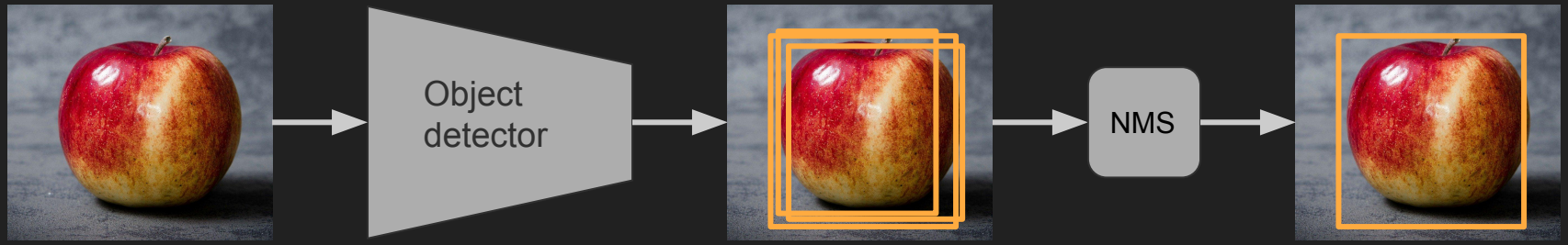
For each proposal:

- Classify: (C classes + background)
- Refinement: scale+offset

# 2-stage detection (sketch)

- Stage one: generate region proposals (object detection without classification)
- For each proposal run the network again on the patch:
  - Optimization: instead of running on the patch of the original image, run on a patch of features generated by the backbone.

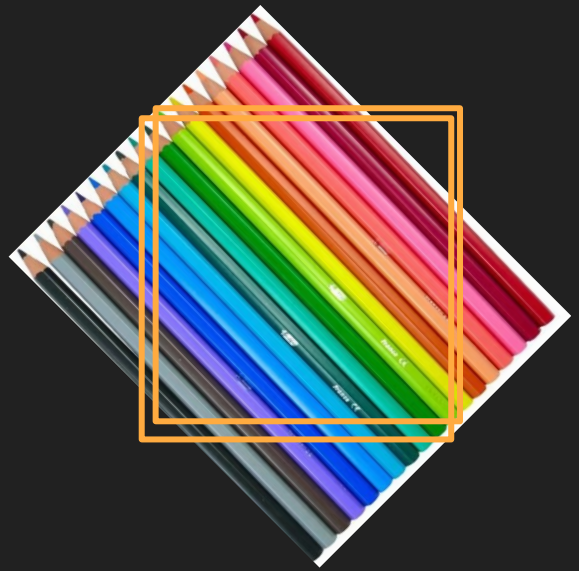
# Non-maximum suppression




NMS:

- Sort detection by decreasing prediction scores.
- Process detected boxes:
  - If a given object has IoU > threshold with some unprocessed candidate, remove the candidate.
- In what scenario NMS is likely to remove valid predictions?

# Non-maximum suppression



In what scenario NMS is likely to remove valid predictions?

- Bounding boxes of two disjoint objects have big IoU.

# Anchor free detectors: pixel level box predictions

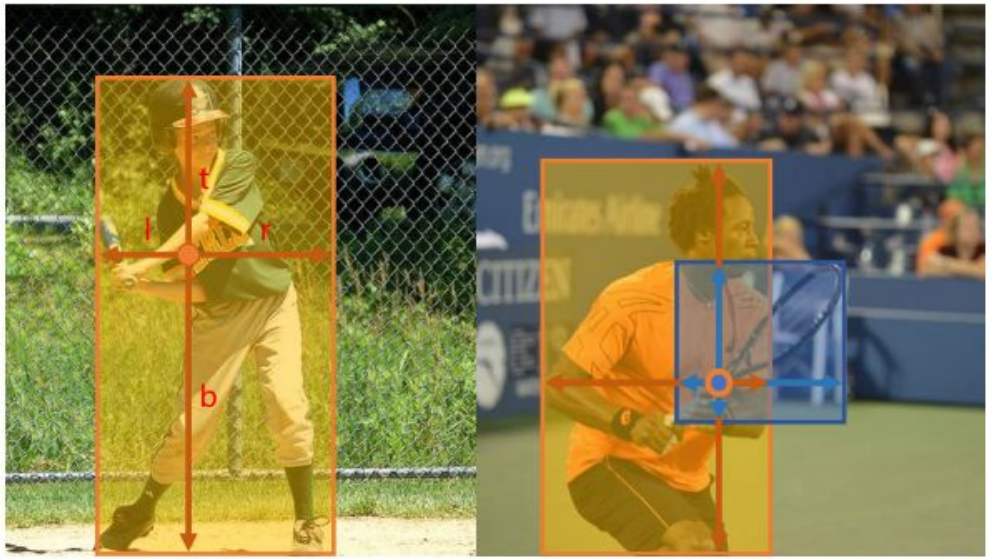



Fig. 1. **Overall concept of FCOS**. As shown in the left image, FCOS works by predicting a 4D vector $(l, t, r, b)$ encoding the location of a bounding box at each foreground pixel (supervised by ground-truth bounding box information during training). The right plot shows that when a location residing in multiple bounding boxes, it can be ambiguous in terms of which bounding box this location should regress.

FCOS paper

# Anchor free detectors: pixel level box predictions

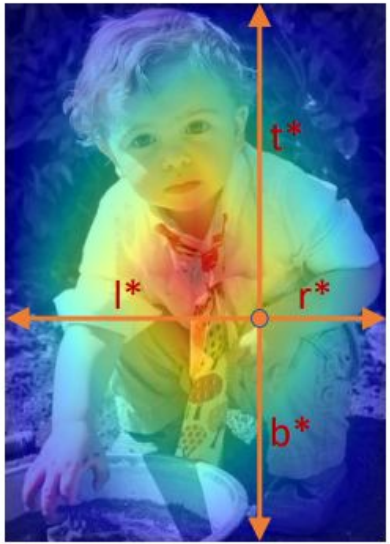



Fig. 4. **Center-ness.** Red, blue, and other colors denote 1, 0 and the values between them, respectively. Center-ness is computed using Eq. (3) and decays from 1 to 0 as the location deviates from the center of the object. During testing, the center-ness predicted by the network is multiplied with the classification score for NMS, thus being able to down-weight the low-quality bounding boxes predicted by a location far from the center of an object.

FCOS paper

# Object detection architectures - summary

- One stage detectors:
  - fast, less precise.
- Two stage detectors:
  - often give best results,
  - typically more computationally intensive.
- Anchor-free detectors (FCOS, CornerNet, CenterNet).

# Feedback is a gift

https://tinyurl.com/dnn-2025-11-12