Object Detection and Segmentation

A. Maier, K. Breininger, L. Mill, N. Ravikumar, T. Würfl
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 27, 2018

Outline

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Introduction

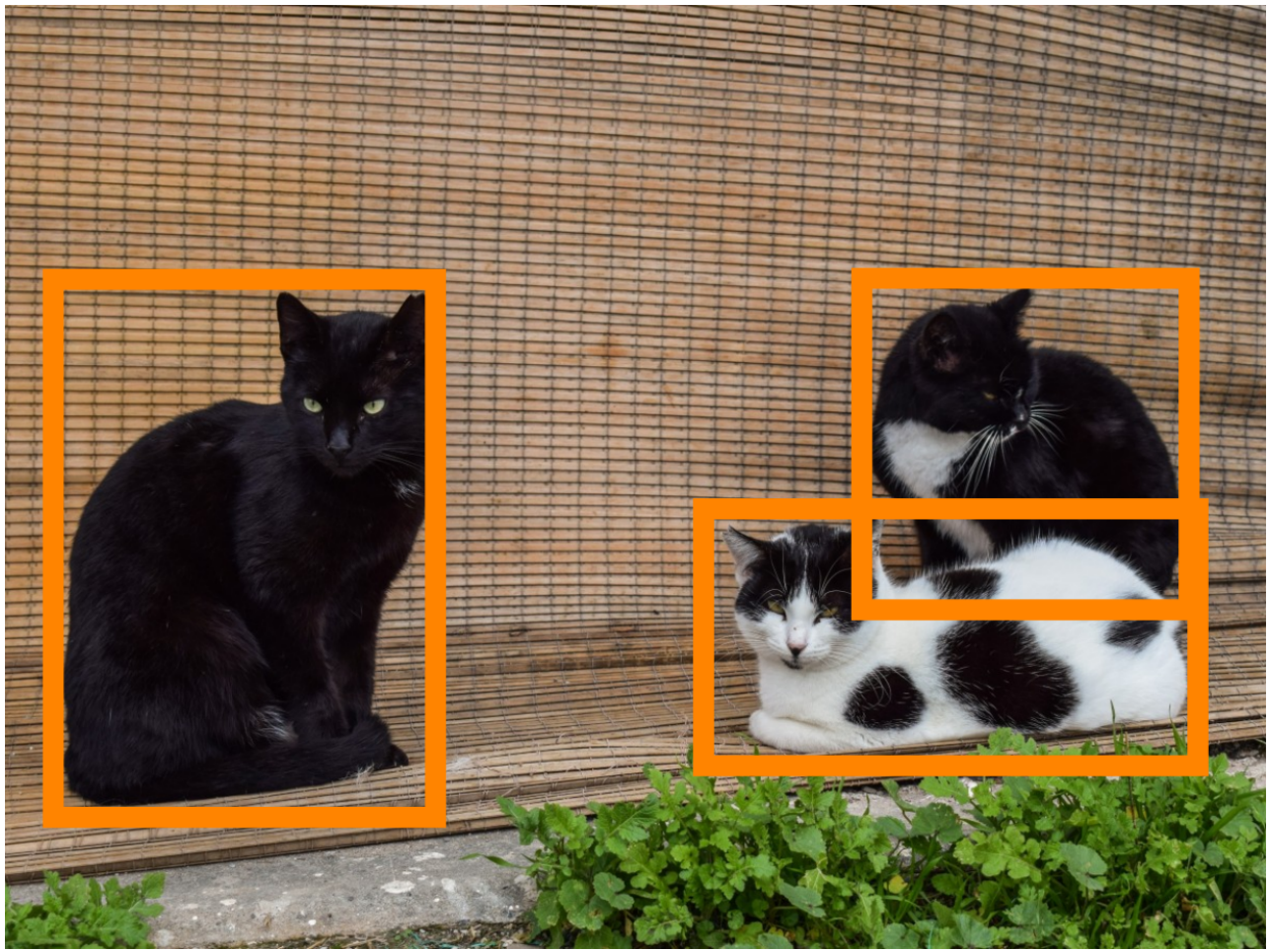
Today's Topics

So far: Classification ➔"Cat"

Today's Topics

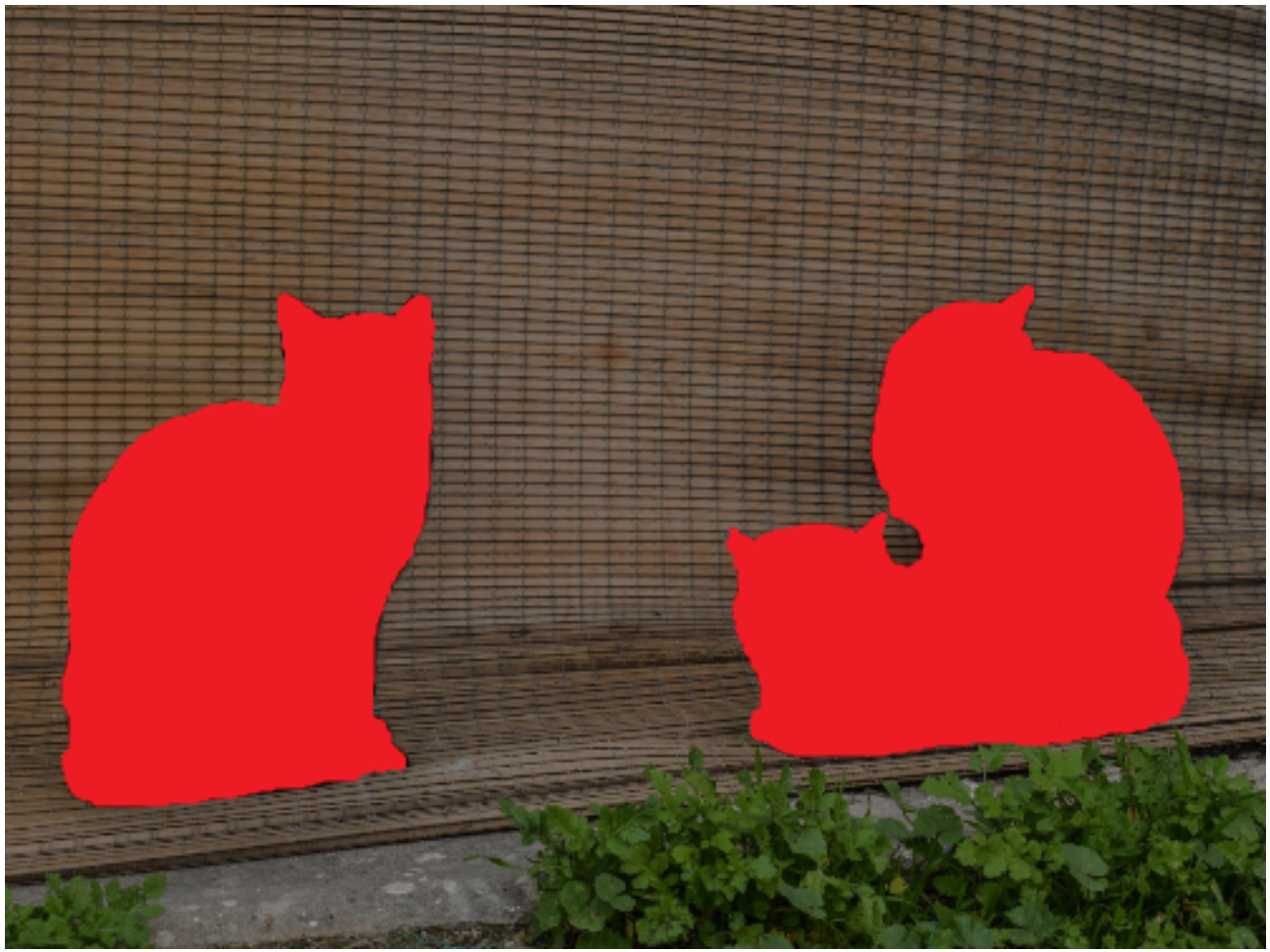
Object recognition + localization (no instance separation) – rarely used

Today's Topics



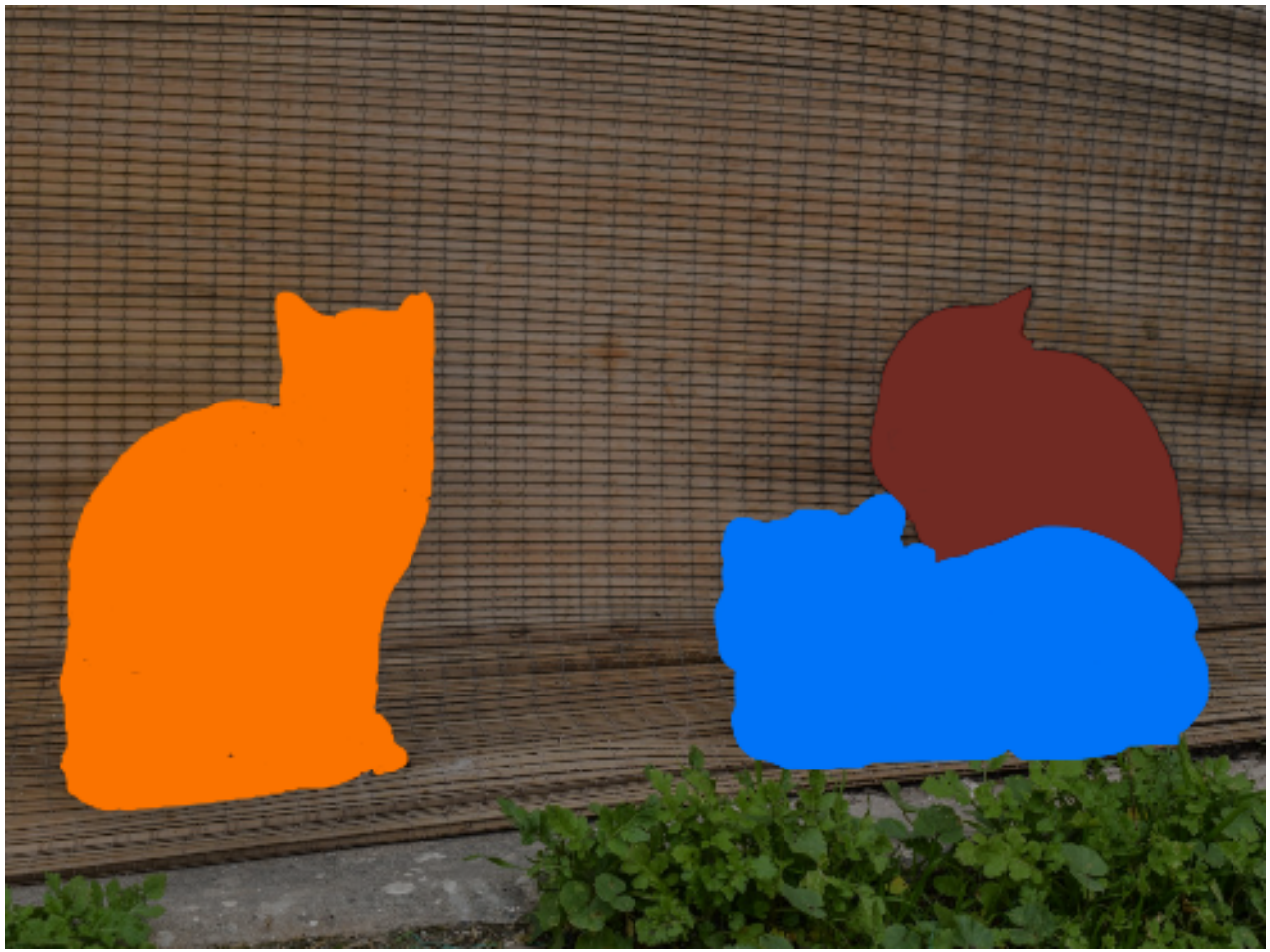
Today Part I: Object detection = instance recognition + localization

Today's Topics



Today Part II: Semantic segmentation

Today's Topics



Today Part III: Semantic instance segmentation

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

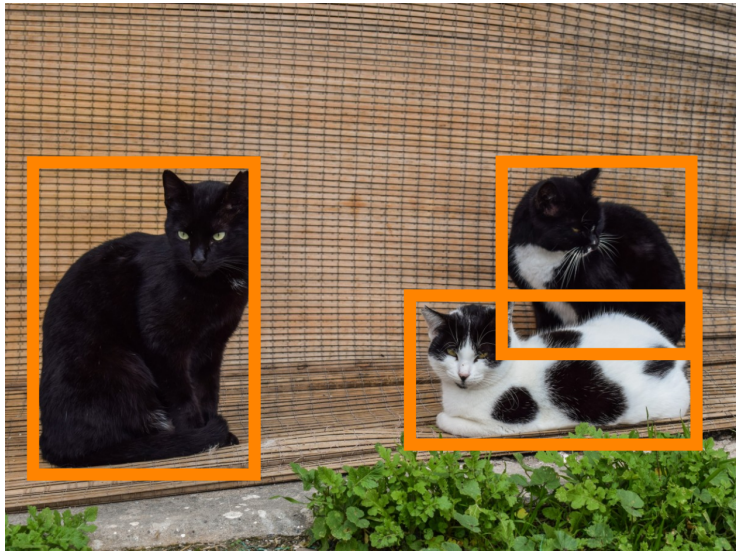
Object Detection

Motivation and Background

Object Detection – Goal

Two tasks:

- Object location
- Classification



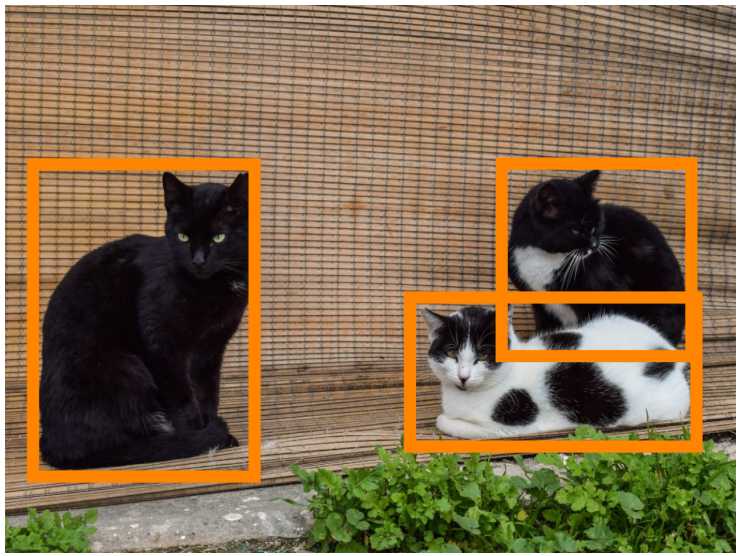
Object Detection – Goal

Two tasks:

- Object location
- Classification

Commonly solved by:

1. Hypothesize bounding boxes
2. Re-sample selected boxes
3. Apply classifier



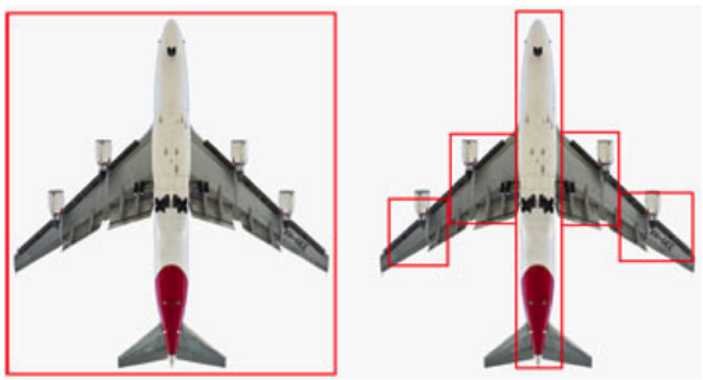
Main distinction: clever combination/replacement of these steps to achieve higher speed or accuracy

What are we looking for?

Bounding Box

- Smallest box by some measure that fully contains the object in question
- Typically defined by: top left corner (x, y), width w, height h

  + classifier confidence

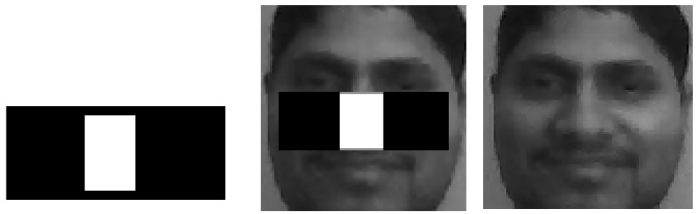


Source: https://github.com/rkerian/Launch-Academy/blob/master/bounding-box/bounding-box.md

Early approaches

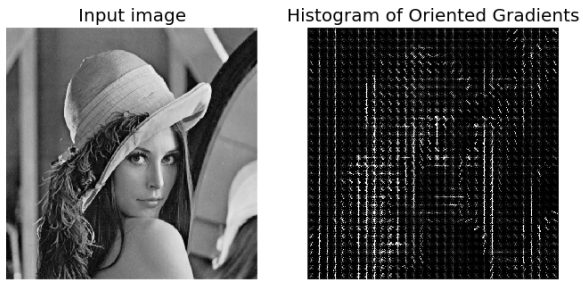
Viola & Jones ’01

- Mainly used for face detection
- Haar Features + Adaboost + cascading classifier for fast rejection
- First competitive real-time object detection



Dalal & Triggs ’05

- Histogram of Oriented Gradients (HOG)
- Support Vector Machines



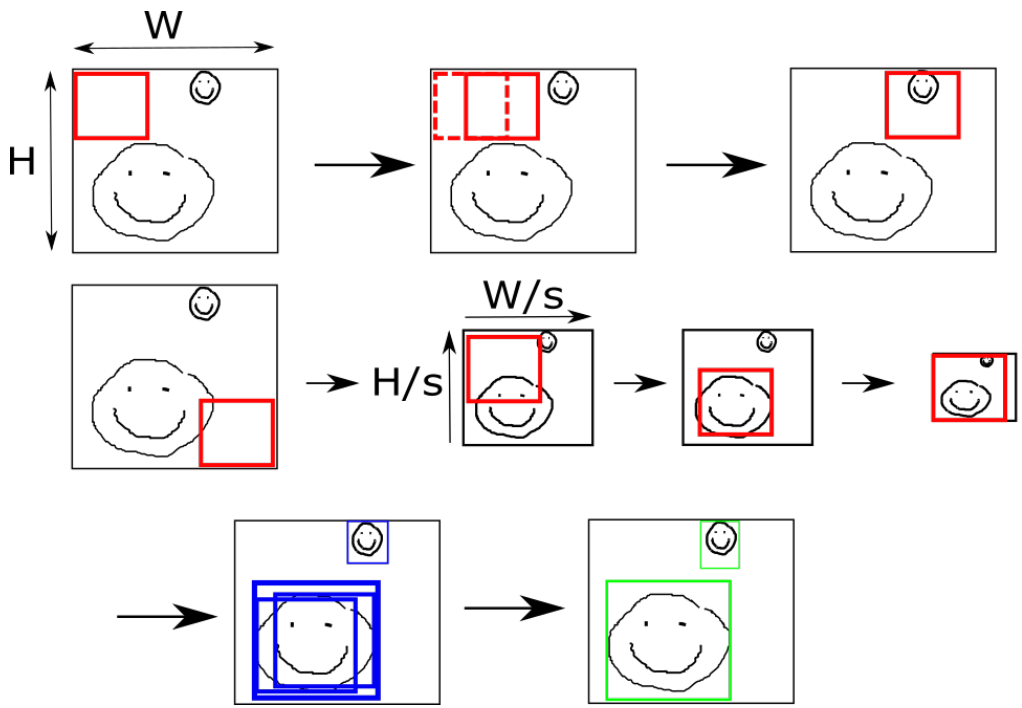
Source: https://en.wikipedia.org , www.sharky93.github.io/docs/gallery/

Modern Approaches

Based on neural networks

- **Sliding window**: classify each possible window by a CNN
- **Region proposal CNN (R-CNN)**: interesting image regions are singled out first, then classified by CNN
- **Single-Shot Detectors**: joint detection and classification
  - You Only Look Once (YOLO)
  - Single-Shot MultiBox Detector (SSD)

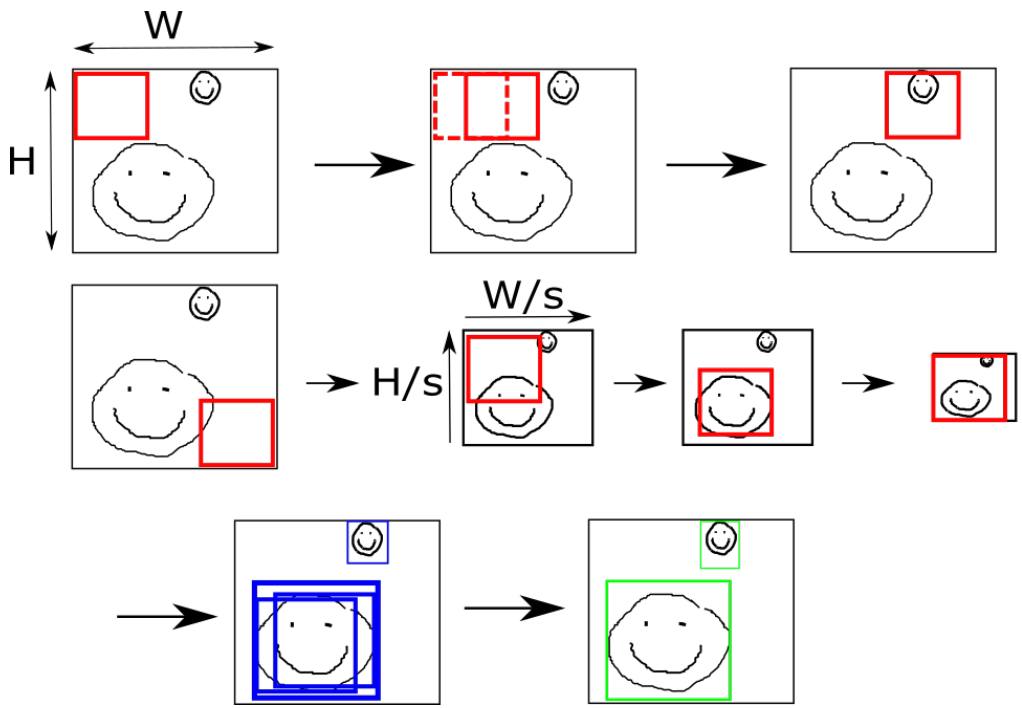
CNN with sliding window



- Slide fixed size window over image + apply trained CNN to each window
- Use multiple image scales to detect objects of different sizes

Source: https://www.semanticscholar.org/

CNN with sliding window



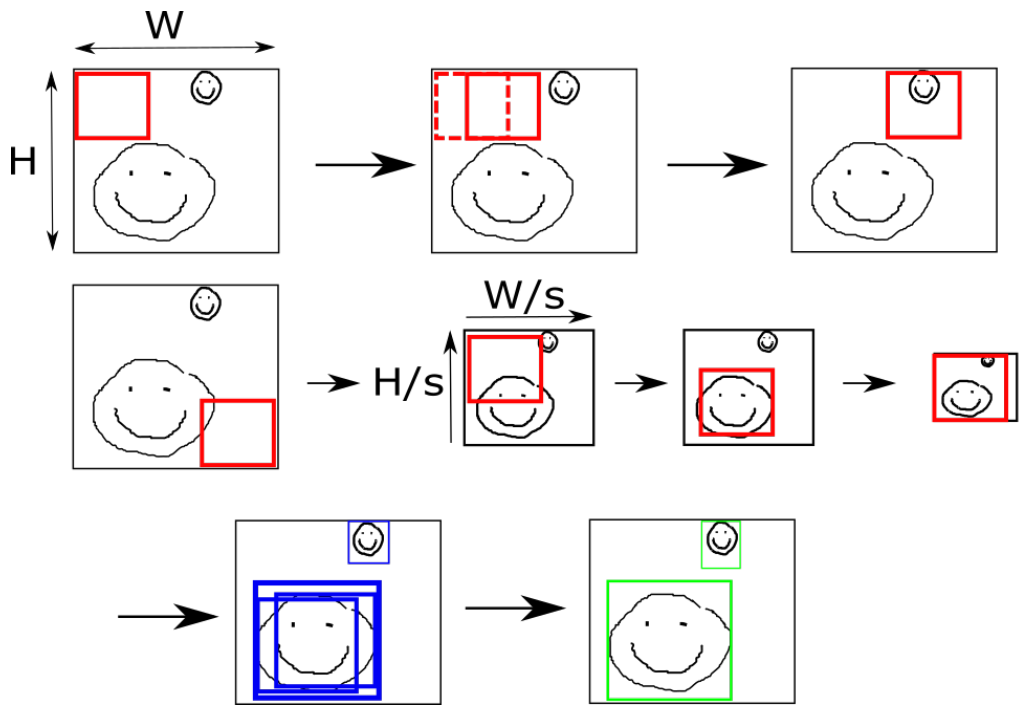
- Slide fixed size window over image + apply trained CNN to each window
- Use multiple image scales to detect objects of different sizes

➔ Large number of predictions, but only a few with high confidence

➔ Keep only multiple detected high confidence areas

Source: https://www.semanticscholar.org/

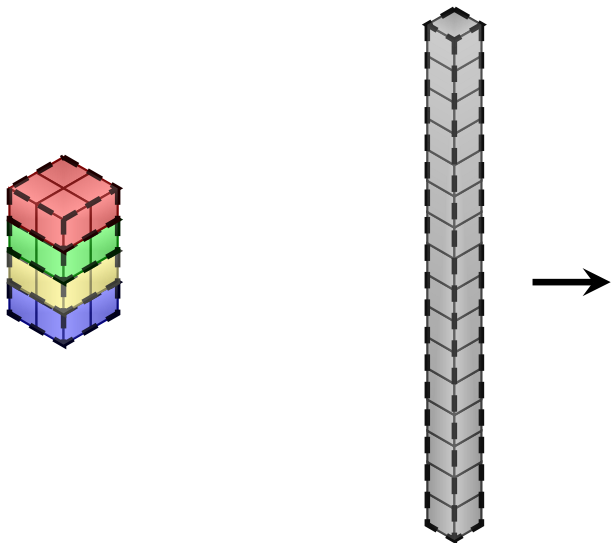
CNN with Sliding Window



+ Trained **classification network** can be used for object detection directly
− Computationally inefficient: One pass through the network for every patch!
➔ Improve speed by using Fully Convolutional Networks

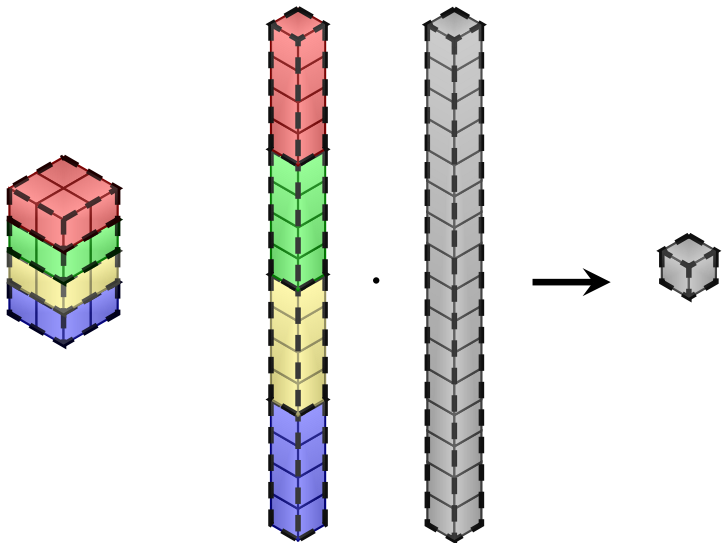
Source: https://www.semanticscholar.org/

Reminder: Fully Convolutional Networks



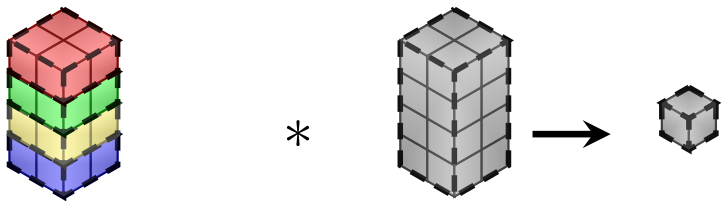
- How can we apply a **fully connected layer** to an arbitrary shaped tensor?

Reminder: Fully Convolutional Networks



- How can we apply a **fully connected layer** to an arbitrary shaped tensor?
- We can **flatten** the activations

Reminder: Fully Convolutional Networks

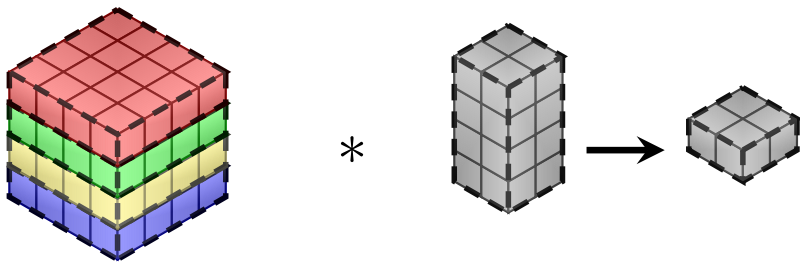


- How can we apply a **fully connected layer** to an arbitrary shaped tensor?
- We can **flatten** the activations

➔ but we can also **reshape the weights** and **convolve**

- This produces the **exact same result**

Reminder: Fully Convolutional Networks



- How can we apply a **fully connected layer** to an arbitrary shaped tensor?
- We can **flatten** the activations

➔ but we can also **reshape the weights** and **convolve**

- This produces the **exact same result**

➔ but can be calculated on **arbitrary spatial input sizes**

- Results can be **aggregated** using pooling

Reminder: Properties of Fully Convolutional Networks

- We **reuse results** ➔ much more efficient

➔ for **inference** fully convolutional is **superior**

Reminder: Properties of Fully Convolutional Networks

- We **reuse results** ➔ much more efficient

➔ for **inference** fully convolutional is **superior**

- But pay attention during **training**:
  - **Background patches** are usually **more frequent**
  - ➔ Resampling data more difficult compared to patches
  - Updates are more strongly correlated

Reminder: Properties of Fully Convolutional Networks

- We **reuse results** ➔ much more efficient

➔ for **inference** fully convolutional is **superior**

- But pay attention during **training**:
  - **Background patches** are usually **more frequent**
  - ➔ Resampling data more difficult compared to patches
  - Updates are more strongly correlated
- A possible remedy: **Train on patches**, use fully convolutional during inference

Region-based Detectors

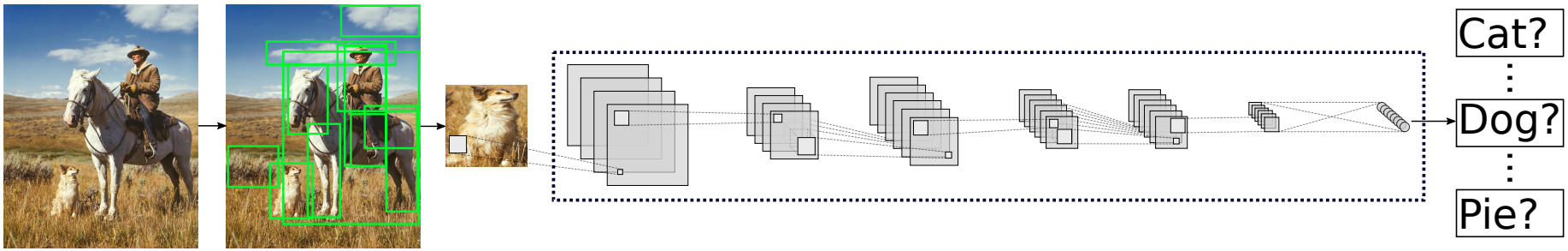
Region Proposals

- Efficiency can be improved using FCNs
- Approach is **still brute-force**
- We also have to **search** over **multiple scales**

Region Proposals

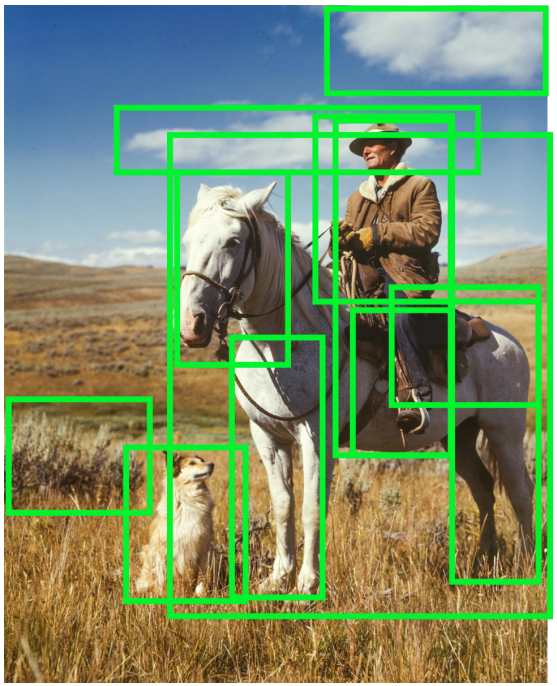
- Efficiency can be improved using FCNs
- Approach is **still brute-force**
- We also have to **search** over **multiple scales**
- Can't we **improve efficiency** by only **considering interesting regions**?

Region Proposals



- Efficiency can be improved using FCNs
- Approach is **still brute-force**
- We also have to **search** over **multiple scales**
- Can't we **improve efficiency** by only **considering interesting regions**?
- Limit classification to "region proposals" ➔ Regional CNN (R-CNN) [20]
- We need a method to generate the **hypothesis regions**

Selective Search [23]

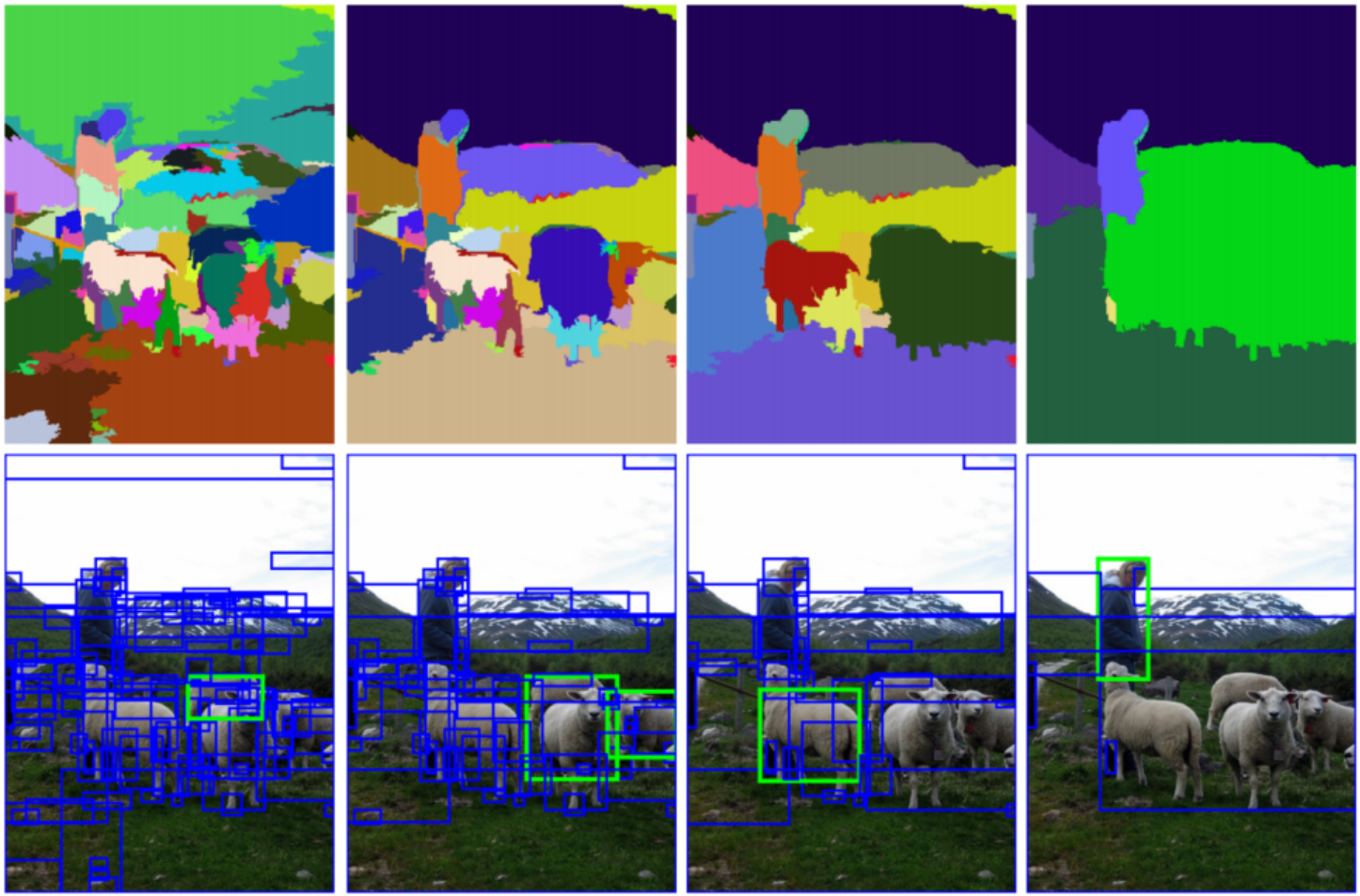


- Region proposal via **selective search**
- Candidate objects by grouping pixels of similar texture or color
- Apply for different sized windows

➔ Produce few thousand ($\approx$ 2k) object proposals per image $\ll$ number possible windows

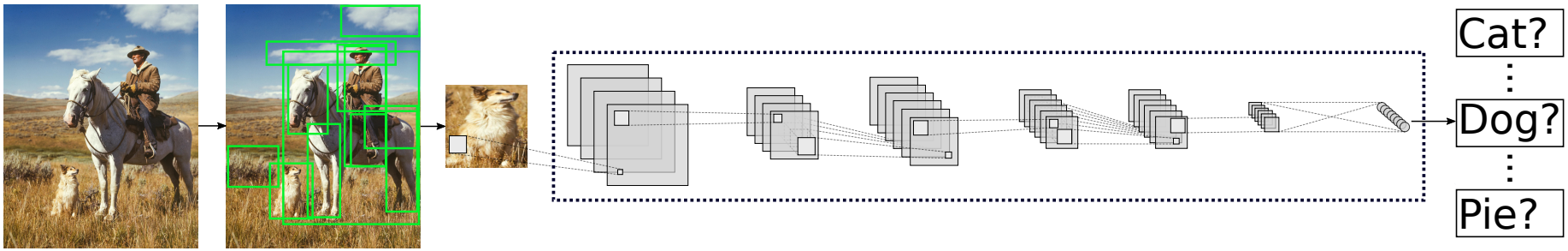
- Essentially a form of segmentation

Selective Search Output



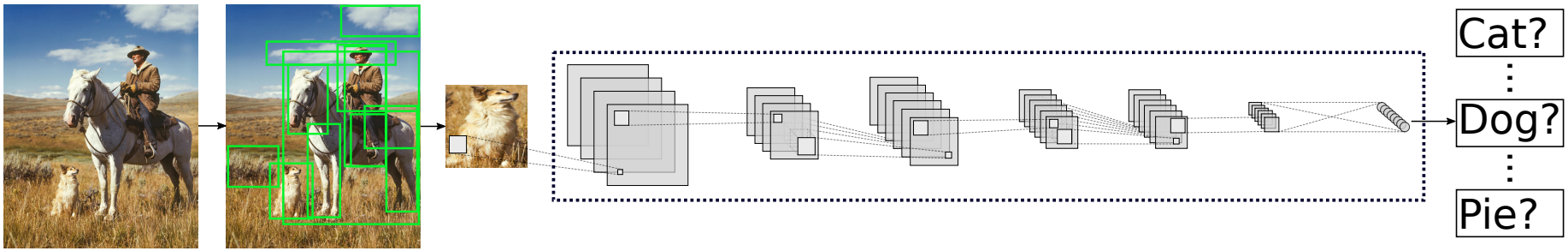
Source: [23]

Region-proposal CNNs [20]



- Limit classification to “region proposals” ➔Regional CNN (R-CNN)
- For each window
  - Warp to standard window size
  - Pre-trained CNN for feature extraction
  - Linear SVM for object classification
  - Linear regression for bounding box refinement

Region-proposal CNNs [20]



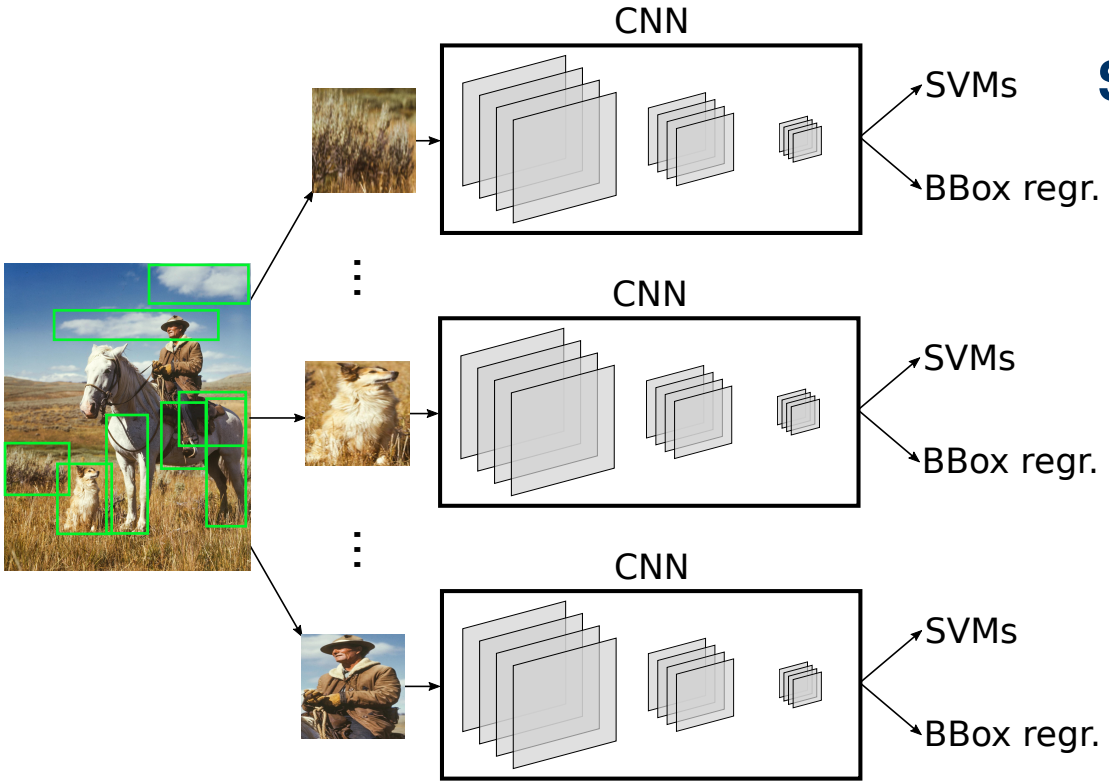
- Limit classification to “region proposals” ➔Regional CNN (R-CNN)
- For each window
  - Warp to standard window size
  - Pre-trained CNN for feature extraction
  - Linear SVM for object classification
  - Linear regression for bounding box refinement

\+ Improved retrieval rate at that time (2013) by more than 30%

– Slow

– Not end-to-end

Region Sampling



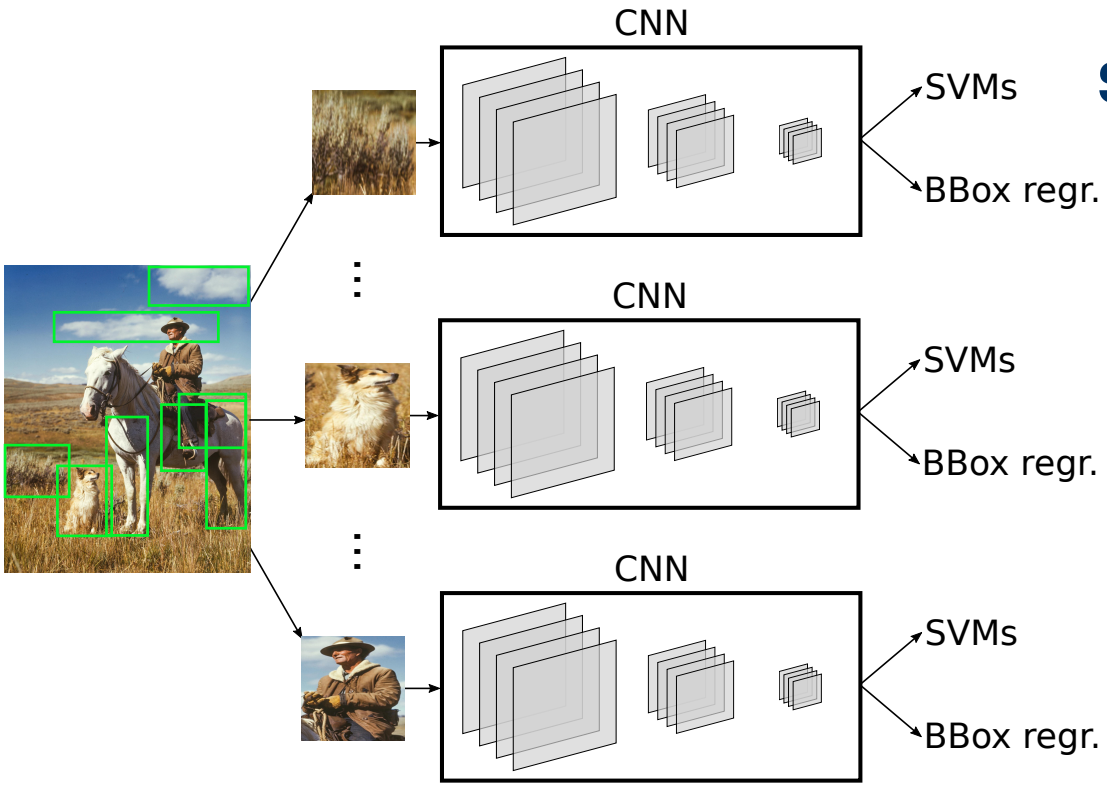
Slow R-CNN

- CNN trained and evaluated at each region proposal

➔ Training is slow

➔ Inference is slow

Region Sampling



Slow R-CNN

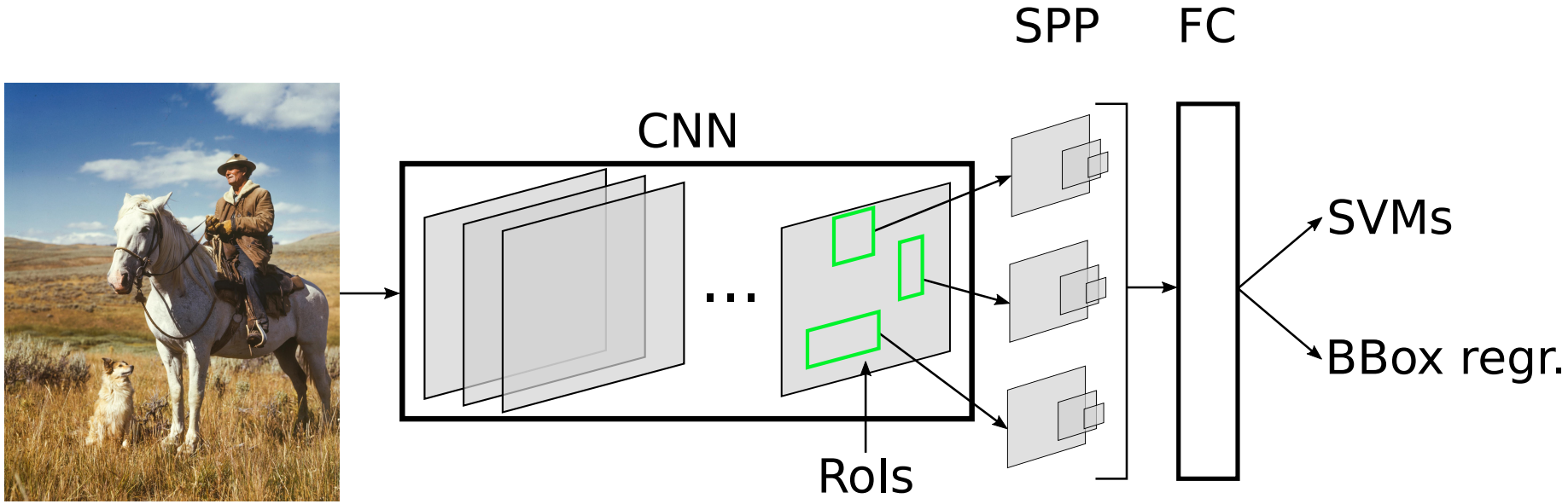
- CNN trained and evaluated at each region proposal

➔ Training is slow

➔ Inference is slow

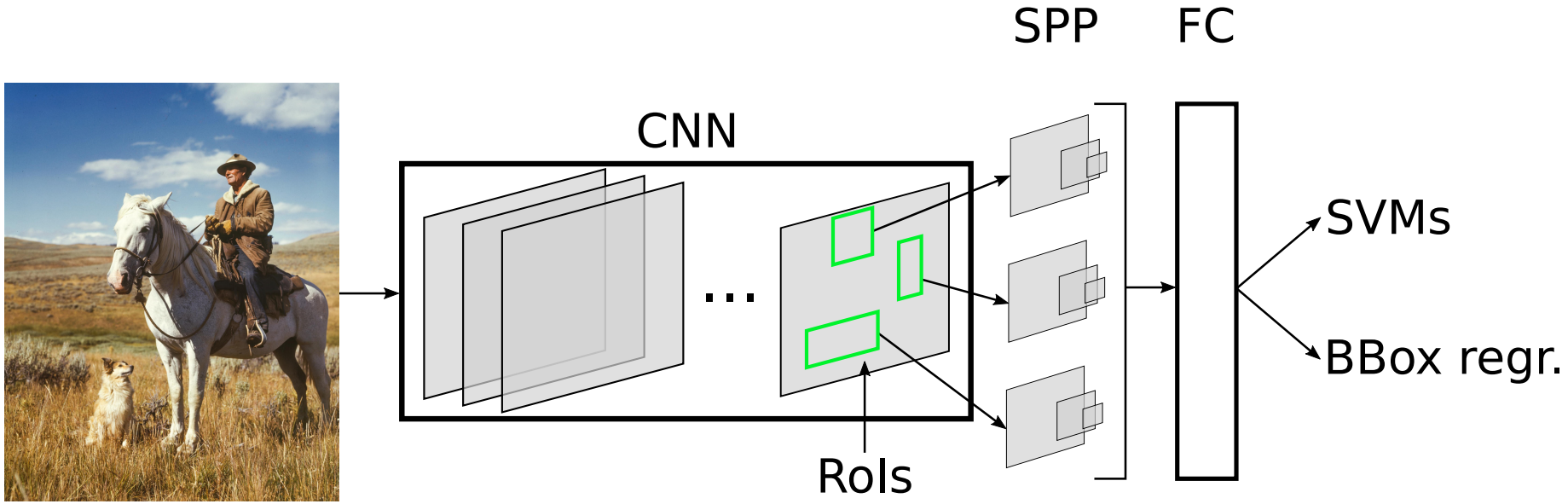
  ➔ Possible solution: spatial pyramid pooling

Spatial Pyramid Pooling Networks [22]



- One pass through the network
- Region proposals on last conv layer
- SPP layer pools to fixed length (3x max-pool w. different window size & stride)

Spatial Pyramid Pooling Networks [22]



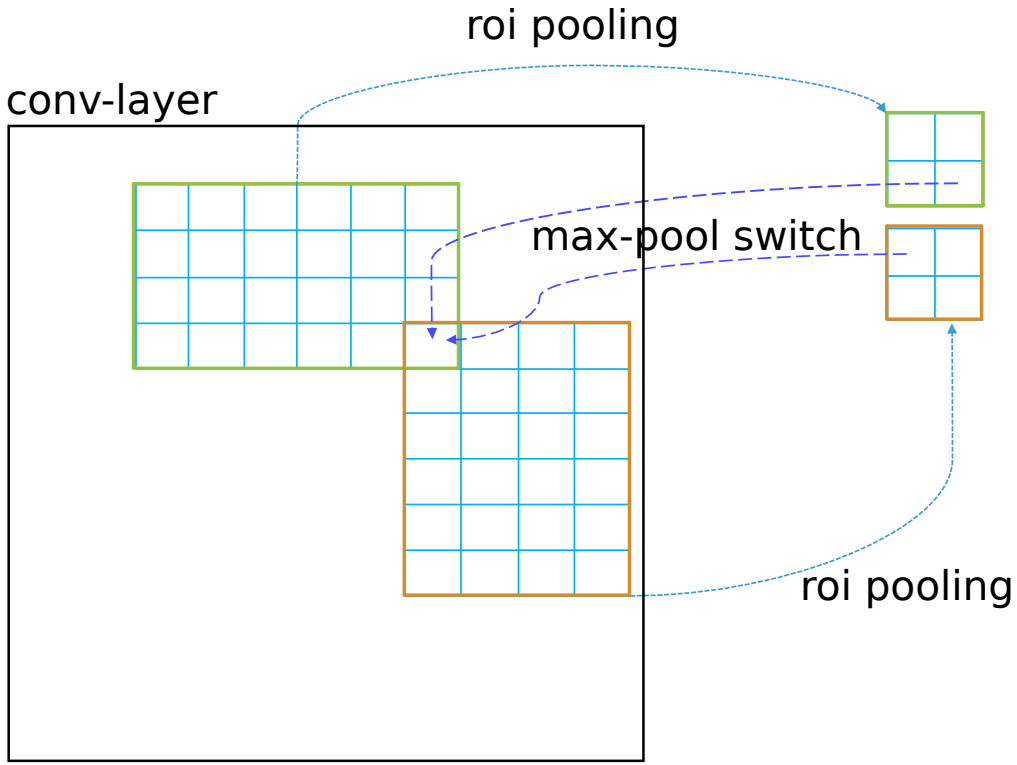
- One pass through the network
- Region proposals on last conv layer
- SPP layer pools to fixed length (3x max-pool w. different window size & stride)

\+ Image-wise computation shared ➔ faster inference

– R-CNN problems: slow training / not end-to-end

– New problem: cannot update parameters below SPP layer during training

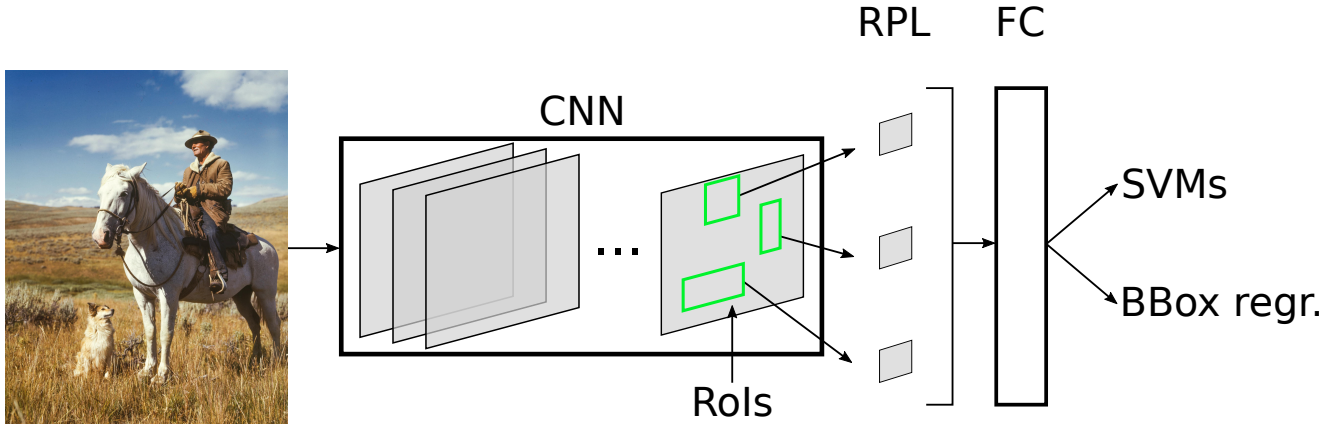
Region of Interest Pooling Layer



- Instead SPP use “region of interest pooling layer (RPL)”
- Max-pool to fixed size by using individual stride length for width/height
- Rest equal to max-pooling (store-max-pool switches)

➔ Trainable!

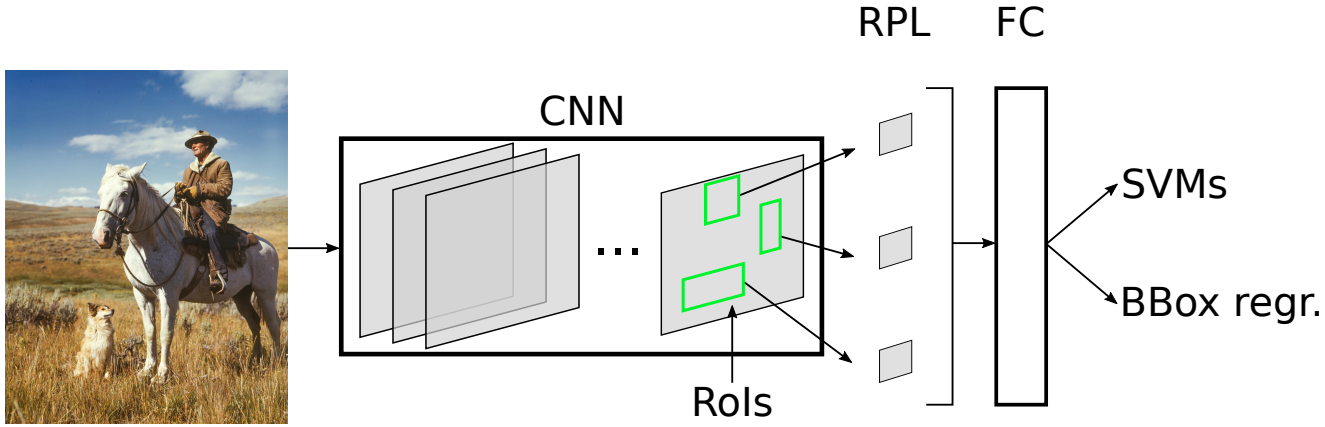
Training RPL: Improved Region Sampling



- Now trainable **but** region-wise sampling for mini-batches:
  - Sample 128 RoIs uniformly at random

  ➔ High variability in one mini-batch

Training RPL: Improved Region Sampling



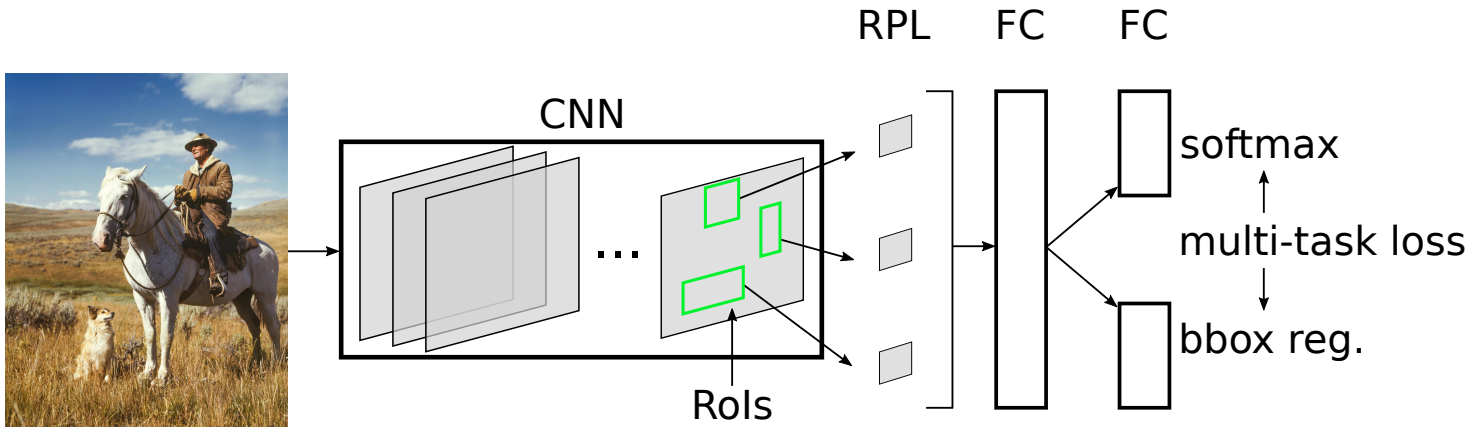
- Now trainable **but** region-wise sampling for mini-batches:
  - Sample 128 RoIs uniformly at random
  - ➔ High variability in one mini-batch

➔ Use **hierarchical sampling**

  - Sample small number of images (2)
  - Sample many examples from each image (64)

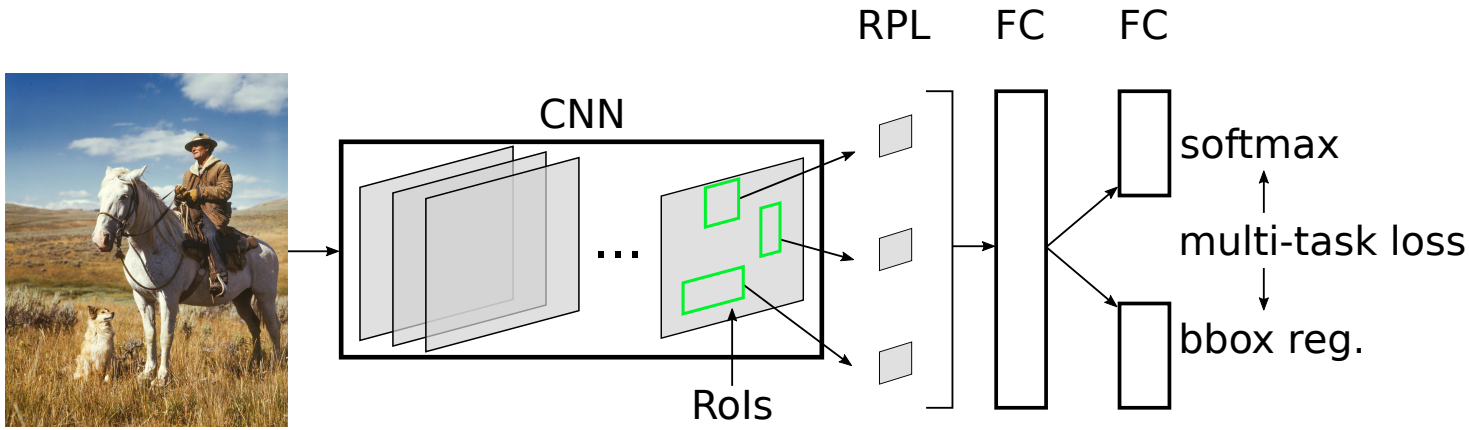
- Another trick: share computation between overlapping examples from the same image by taking union of RoIs' receptive fields

Fast R-CNN [6]



- Region proposal Layer (trained via improved region sampling)
- Replace SVM + regression by: softmax layer for object classification + regression layer for bounding box finetuning ➔ trained by multi-task loss

Fast R-CNN [6]

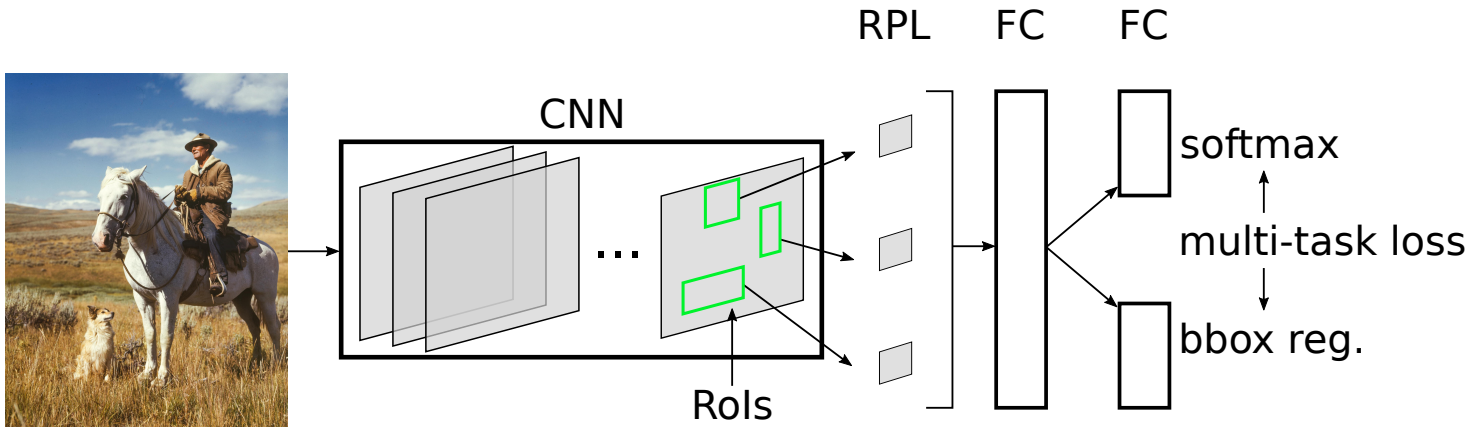


- Region proposal Layer (trained via improved region sampling)
- Replace SVM + regression by: softmax layer for object classification + regression layer for bounding box finetuning ➔ trained by multi-task loss

\+ 9× faster than R-CNN

− Not real-time

Fast R-CNN [6]



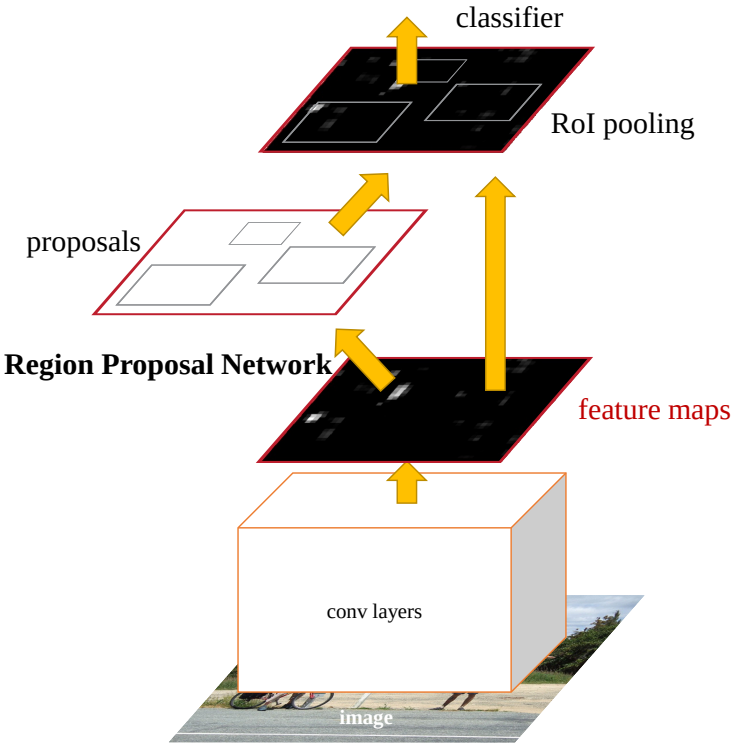
- Region proposal Layer (trained via improved region sampling)
- Replace SVM + regression by: softmax layer for object classification + regression layer for bounding box finetuning ➔ trained by multi-task loss

\+ 9× faster than R-CNN

− Not real-time

/ Nearly end-to-end **apart** from RoI proposals on conv layer

Faster R-CNN [5]

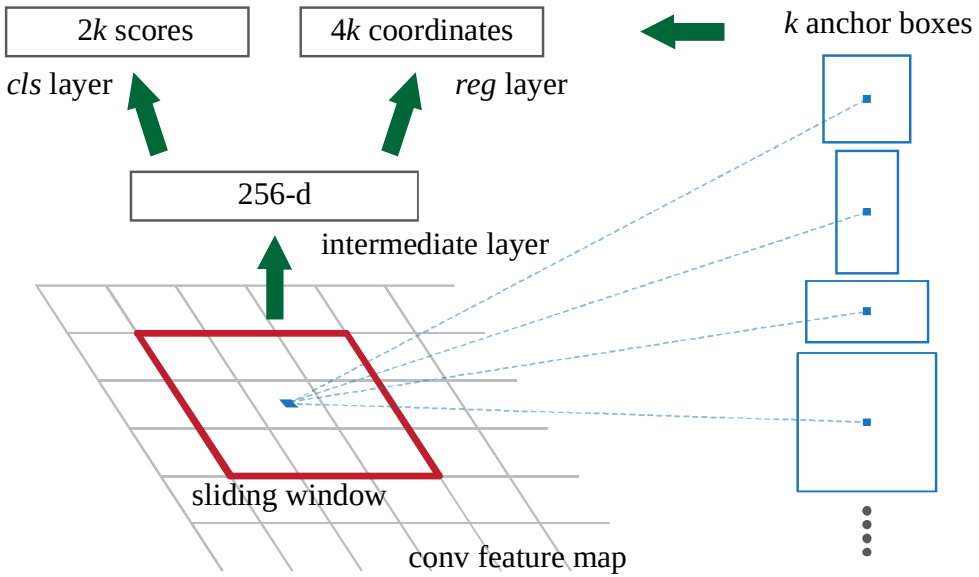


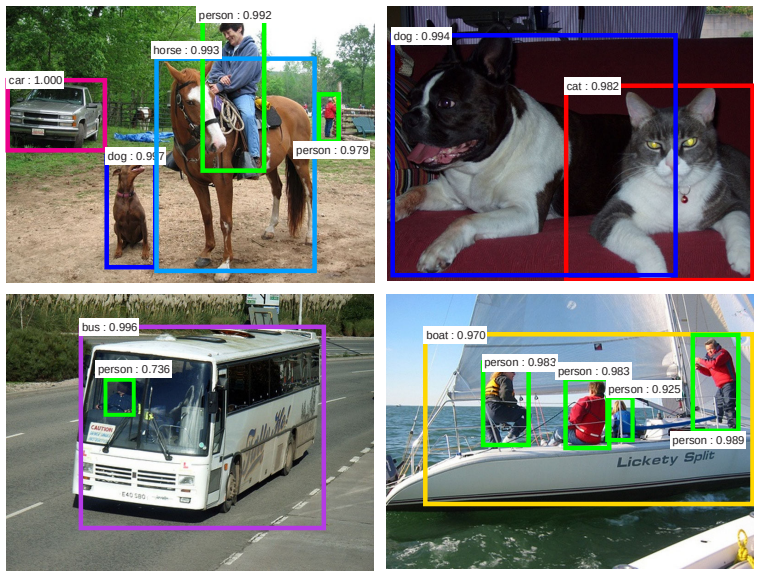
- Key idea: Add region proposal network
- Input: generated feature maps
- Output: region propsals

➔ Integrate into “Fast R-CNN”

Source: [5]

Faster R-CNN [5]





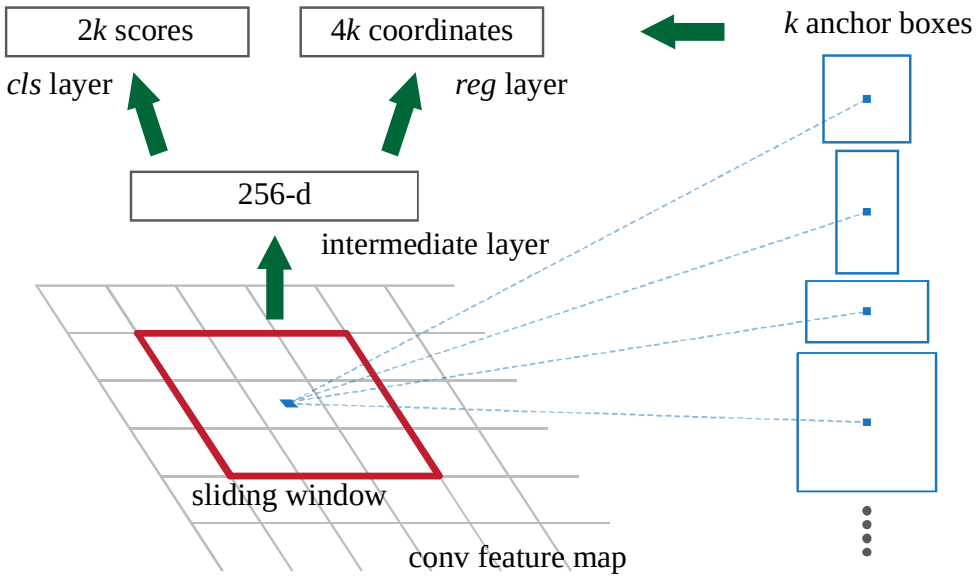
- defines k anchor boxes associated with different scale and aspect ratios (typically: 3 scales / 3 aspect ratios ➔ $k = 9$)

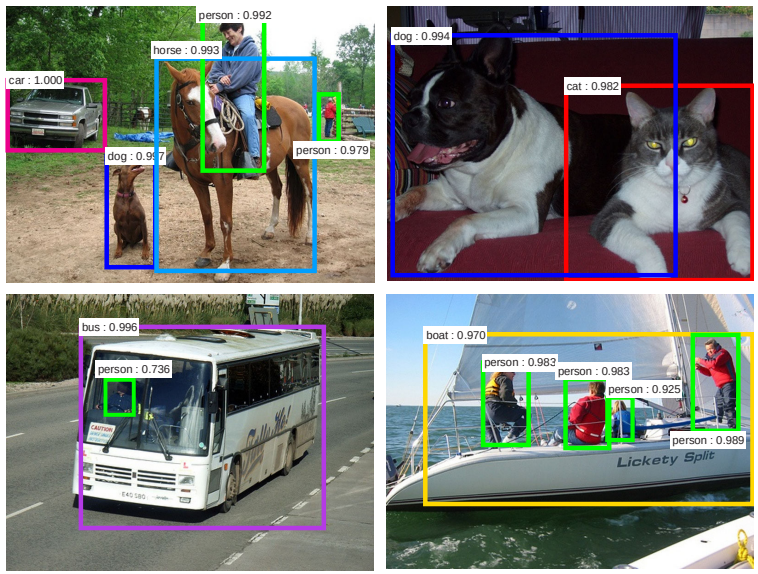
➔ Efficient multi-scale ability (neither image nor filter sizes need to change)

➔ $4k$ box parameters + $2k$ scores (softmax for object/not object)

Source: [5]

Faster R-CNN [5]





- defines k anchor boxes associated with different scale and aspect ratios (typically: 3 scales / 3 aspect ratios ➔ $k = 9$)

➔ Efficient multi-scale ability (neither image nor filter sizes need to change)

➔ $4k$ box parameters + $2k$ scores (softmax for object/not object)

\+ End-to-end system

– Not real-time (5-17 fps)

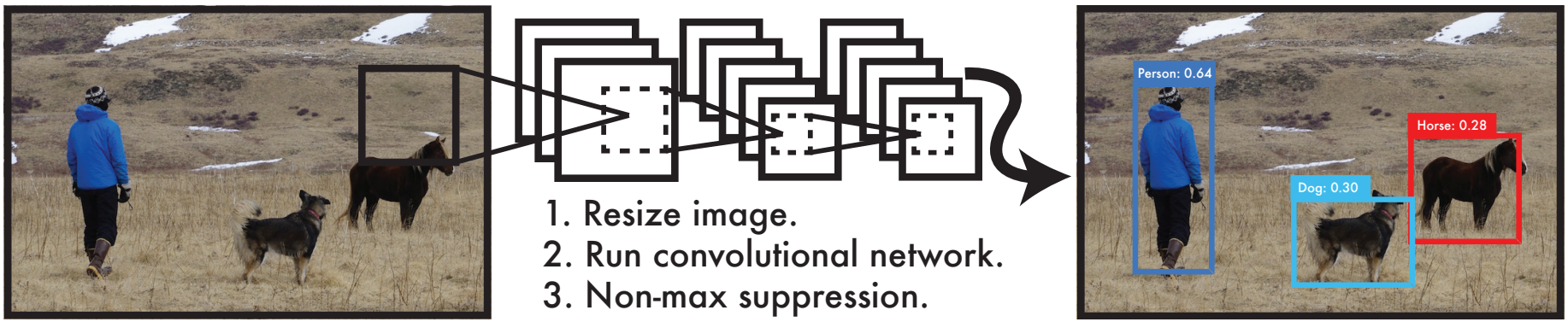
Source: [5]

Single-Shot Detectors

Can't we just use the region proposal network as detector?

In a YOLO fashion?

YOLO [26]



- R-CNN / Fast R-CNN produces many object candidate suggestions that are passed to the CNN

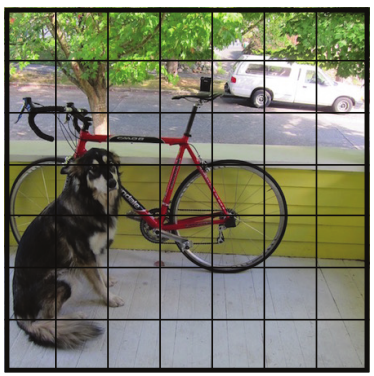
➔ Slow and cumbersome

➔ Single-shot detectors replace the many forward-passes by only one

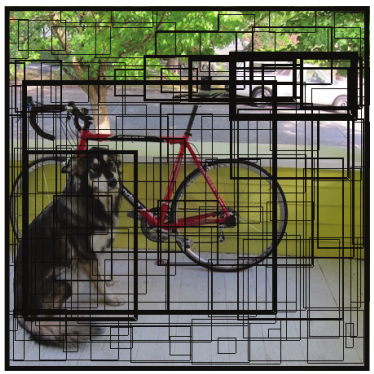
- **You Only Look Once** (YOLO) algorithm
- Combined bounding box prediction and classification in one network

Source: [26]

YOLO [26]



$S \times S$ cells



bound. boxes + confidence

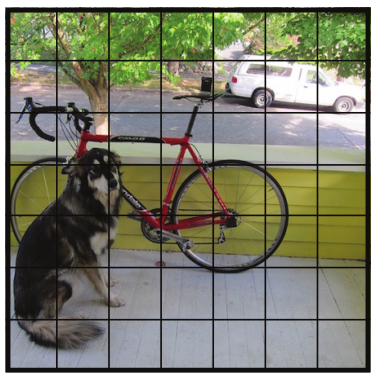
class probability map

- Each cell predicts
  - B bounding boxes + confidence score
  - Class confidence

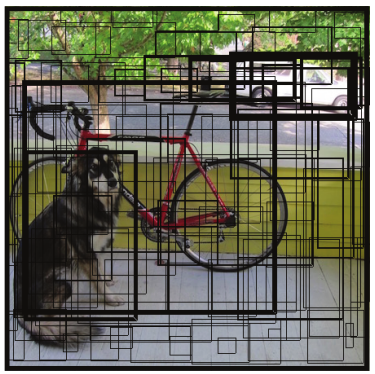
➔ CNN predicts:
$S \times S \times (5B + C)$
values for C classes

Source: [26]

YOLO [26]

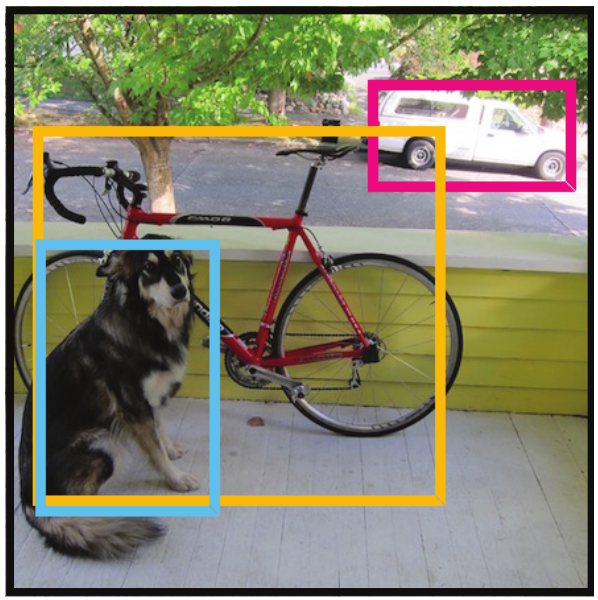


$S \times S$ cells



bound. boxes + confidence

class probability map



Adjust output according to aggregated score

Source: [26]

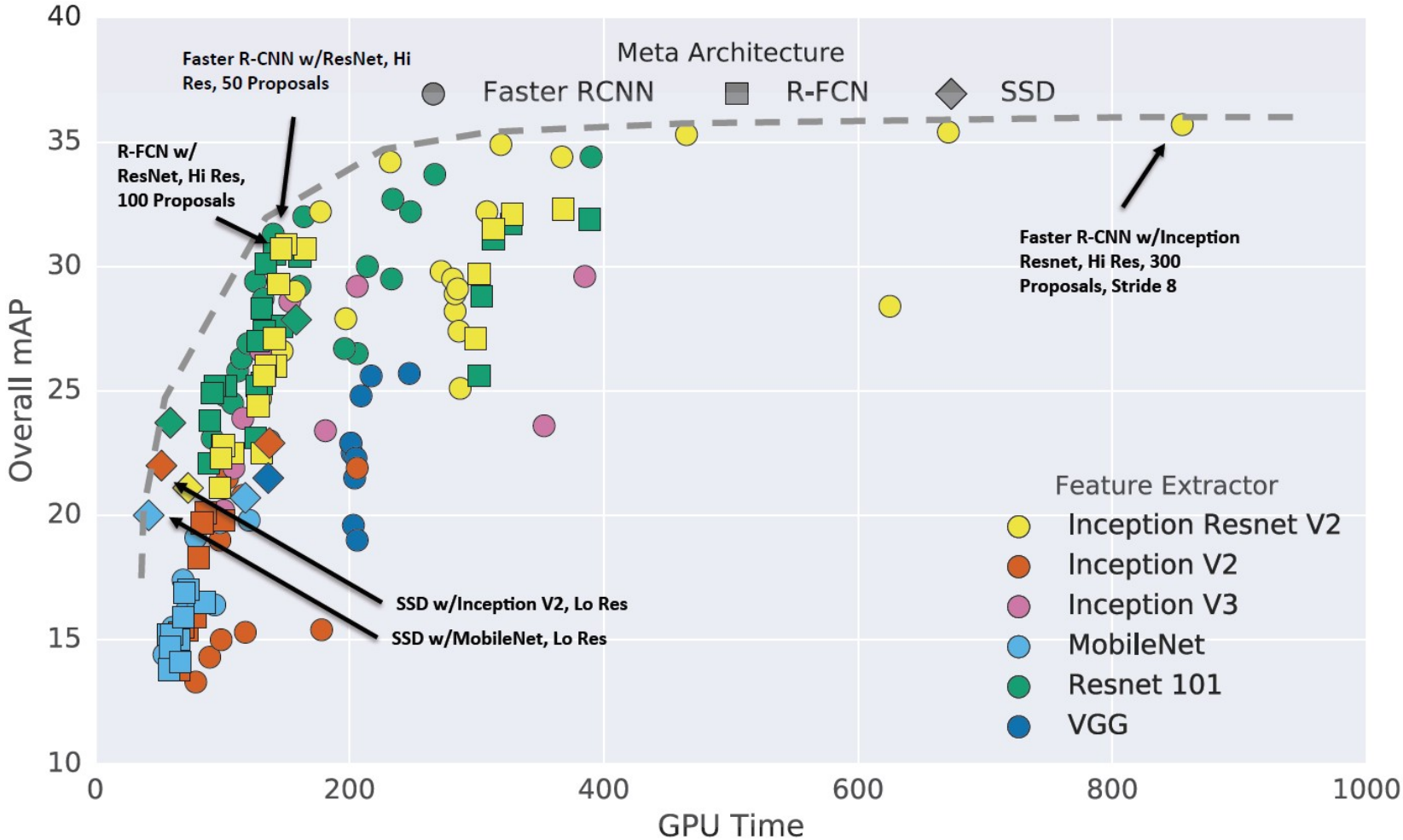
YOLO9000 [27]

- YOLO9000 is an improved version of YOLO advertised as Better, Faster, Stronger
- Better
  - Batch normalization and use of a high-res classifier improve mAP by up to 6%
  - Anchor boxes found by clustering over the training data improves recall by 7%
  - Training over multiple scales allows YOLO9000 to detect objects at different resolutions more easily
- Faster
  - Using a different CNN architecture speeds up the forward pass
- Stronger
  - Hierarchical prediction on a tree allows to combine different object detection datasets
- All this allows YOLO9000 to detect up to 9000 classes in real-time or faster

The Single-Shot Multi-Box Detector[24]

- A popular alternative to YOLO:
  - **Single-Shot**: Like YOLO, only one forward pass through the CNN
  - **MultiBox**: Name of the bounding box regression technique which SSD is based on [15]
  - **Detector**: It's (obviously) an object detector
- Differs from YOLO in several aspects but shares the same core idea

Tradeoff: Speed-Accuracy



Source: Speed/accuracy trade-offs for modern convolutional object detectors [12]

Class Imbalance to Tackle the Speed-Accuracy Tradeoff

- All single shot detectors evaluate **many hypothesis locations**

➔ but **most** of them are **easy negatives**

- This **imbalance** is **not addressed** by current training
- In classical methods this is dealt with by **hard-negative mining**
- Can we **change the loss functions** to pay **less attention to easy examples**?

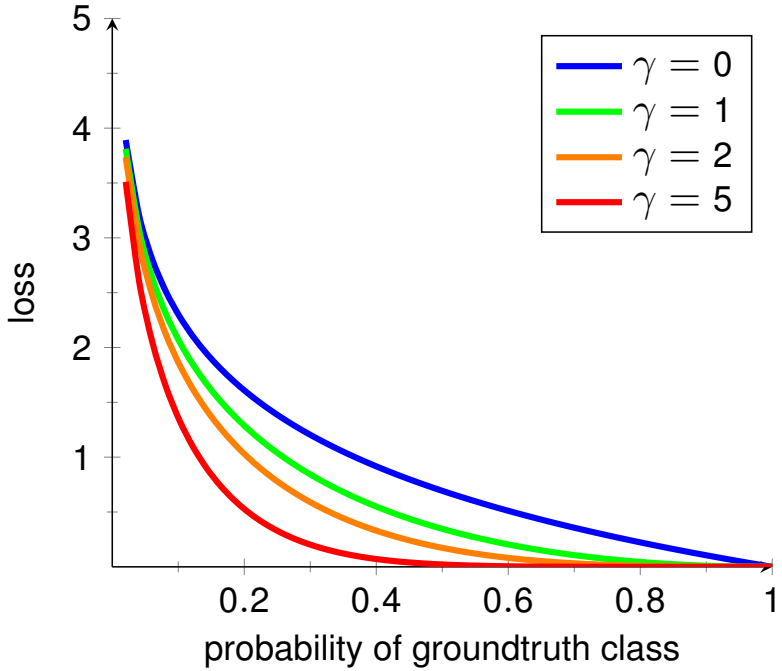
Focal Loss[7]

- Objectness is binary
- Model as Bernoulli distributed:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}$$

- The usual loss function is cross-entropy: $\mathbf{CE}(p_t) = -\log(p_t)$

Focal Loss[7]



- Objectness is binary
- Model as Bernoulli distributed:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}$$

- The usual loss function is cross-entropy: $\mathbf{CE}(p_t) = -\log(p_t)$
- We modify this to:

$$\mathbf{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \cdot \log(p_t)$$

- α_t are **imbalance weights** calculated as inverse class frequency
- γ is a hyperparameter **decreasing** the **influence of easy examples**

Summary: Object detection

- Main tasks: Bounding boxes + classification
- Sliding window approach extremely inefficient
- Region proposal networks reduce number of candidates
- Single-shot detectors (YOLO, SSD) avoid additional step
- Object detector concepts can be combined with arbitrary feature extraction/classification network
- Speed/accuracy tradeoff!

Segmentation

Motivation

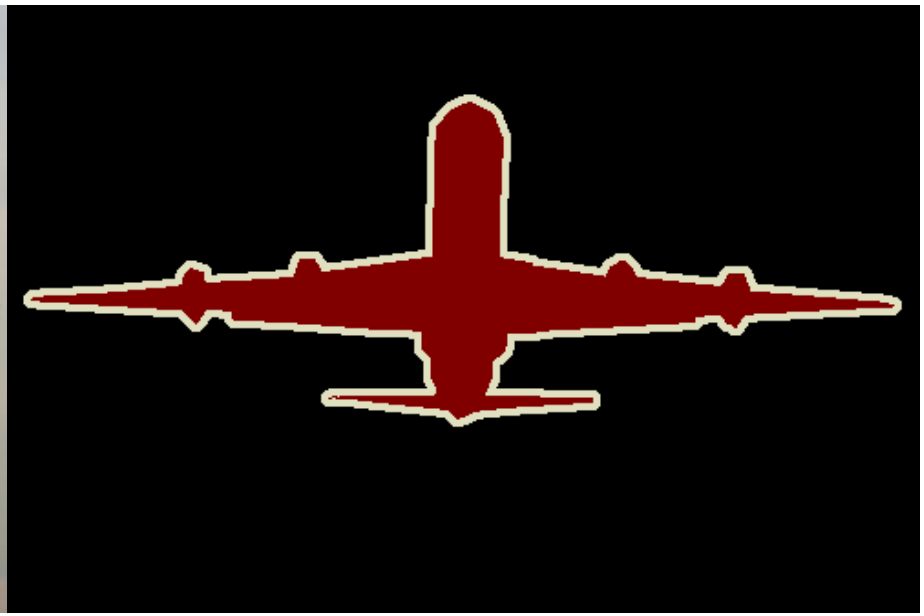
What is Semantic Segmentation?

Source: Pascal VOC 2012

- Goal: partition images into **different segments**
- Segments are regions that delineate **meaningful objects**
- Label regions with an **object category label**
- Each pixel gets a semantic class
  ➔ pixel-wise dense classification
- Concepts can be applied to other signals, e.g., sound

What is Semantic Segmentation?

Source: Pascal VOC 2012



Source: Pascal VOC 2012

Applications of Segmentation

- Medical Imaging
  - Organs
  - Vessels
  - Cells
- Autonomous driving
  - Traffic signs
  - Pedestrian detection
  - Street detection
- Aerial imaging, robotics, image editing etc.

Source: Cityscapes dataset

Evaluation Metrics

- Usefulness of segmentation depends on many factors:
- Execution time, memory footprint and quality
- Quality of a method can be assessed by different metrics
- Main problem: Classes are not equally distributed ➔ have to account for that

Assumptions:

- $k + 1$ classes including void or background
- p_{ij} is the amount of pixels of class i inferred to belong to class j

➔ p_{ii} represents the number of true positive

Evaluation Metrics

1. **Pixel Accuracy (PA):** Ratio between the amount of correctly classified pixels and the total number of pixels

$$PA = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}}$$

2. **Mean Pixel Accuracy (MPA):** Average ratio of correctly classified pixels per-class basis

$$PA = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij}}$$

Evaluation Metrics

3. **Mean Intersection over Union (MIoU):** Ratio ratio between the intersection and the union of two sets

$$PA = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}$$

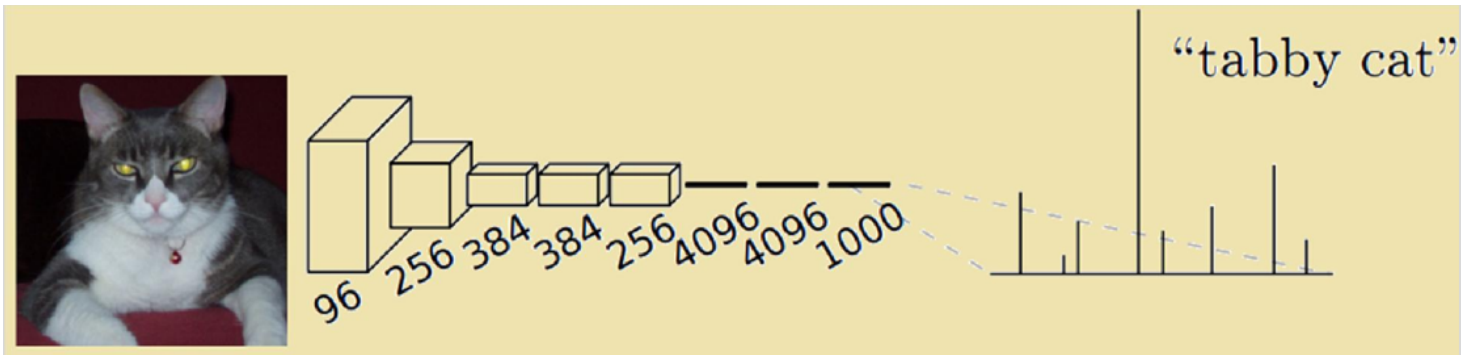
4. **Frequency Weighted Intersection over Union (FWIoU):** balanced MIoU – weights each class depending on its frequency

$$PA = \frac{1}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}} \sum_{i=0}^{k} \frac{\sum_{j=0}^{k} p_{ij} p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}$$

Fully Convolutional Networks for Segmentation

Reminder: Fully Convolutional Networks

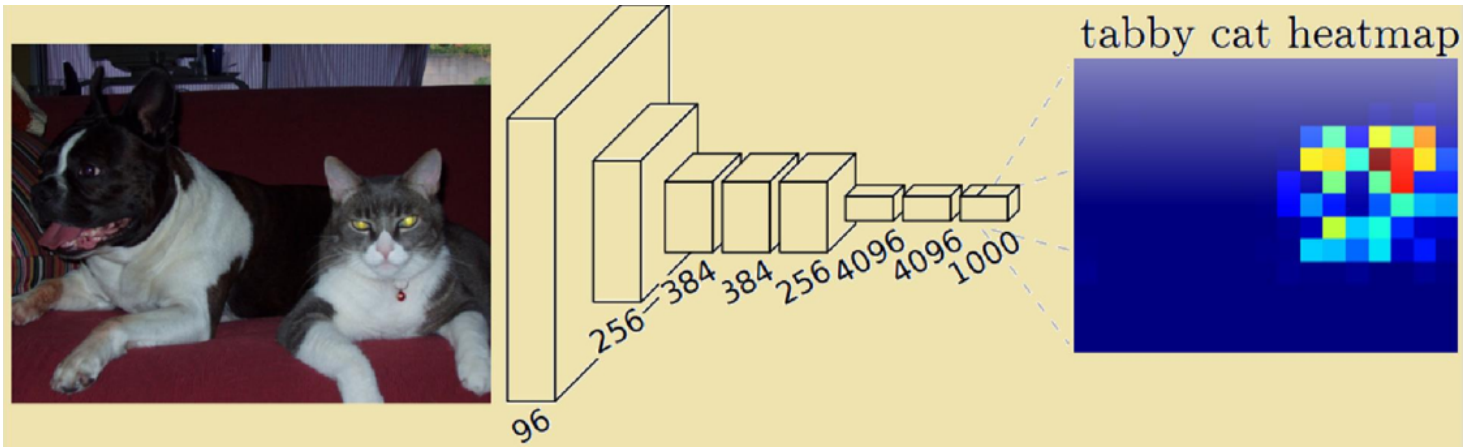
- Convolutional neural networks with **fully connected layers** used for classification:
  - CNN layers for feature extraction
  - Output dimensions reduced because of subsampling
  - **Fully connected layers** have fixed inputs and remove spatial information
  - Output: vector encoding class probabilities



Source: Long et al., 2015

Fully Convolutional Networks (cont.)

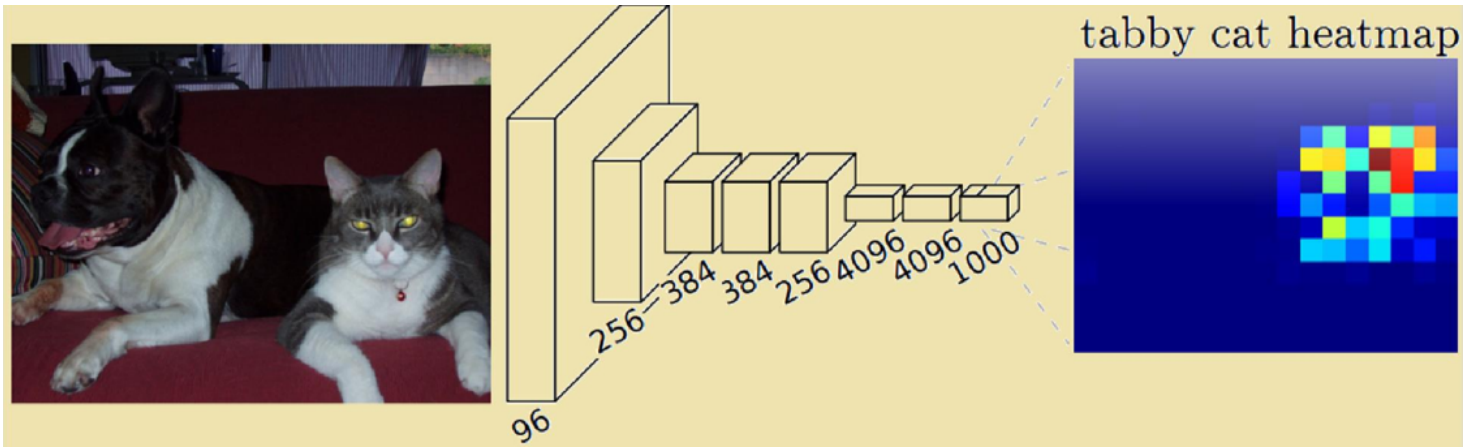
- Fully convolutional networks for segmentation [13]
  - Transform fully connected layers to **convolutional** layers
  - Output independent of size ➔ heat map
  - Pooling operations coarsen the output of the network



Source: Long et al., 2015

Fully Convolutional Networks (cont.)

- Fully convolutional networks for segmentation [13]
  - Transform fully connected layers to **convolutional** layers
  - Output independent of size ➔ heat map
  - Pooling operations coarsen the output of the network



➔ Main problem: Segmentation is coarse (compared to input resolution)

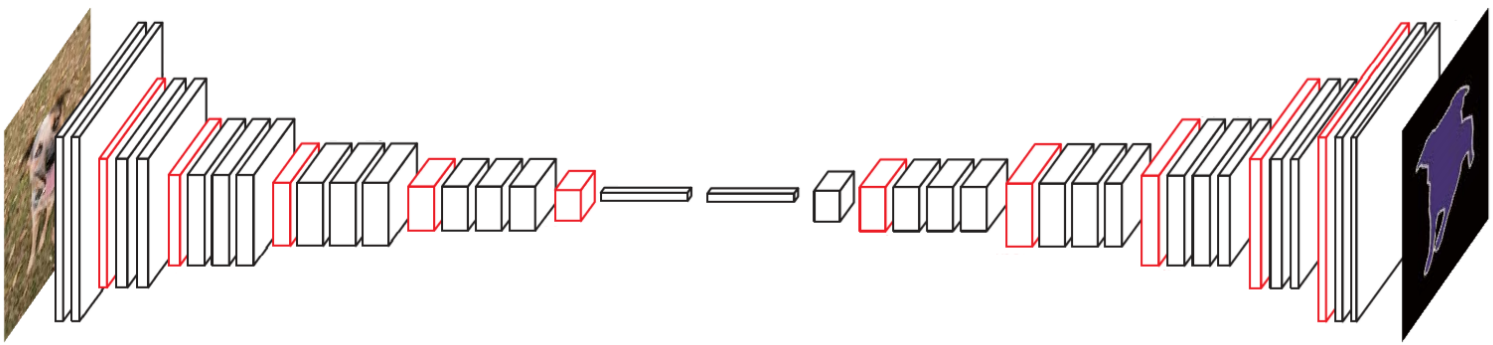
➔ Main question: How can we increase the resolution and keep global context?

Source: Long et al., 2015

Encoder and Decoder

- Most methods use a classification network as basis

➔ If it can pickup the class, it should be able to learn the correct features



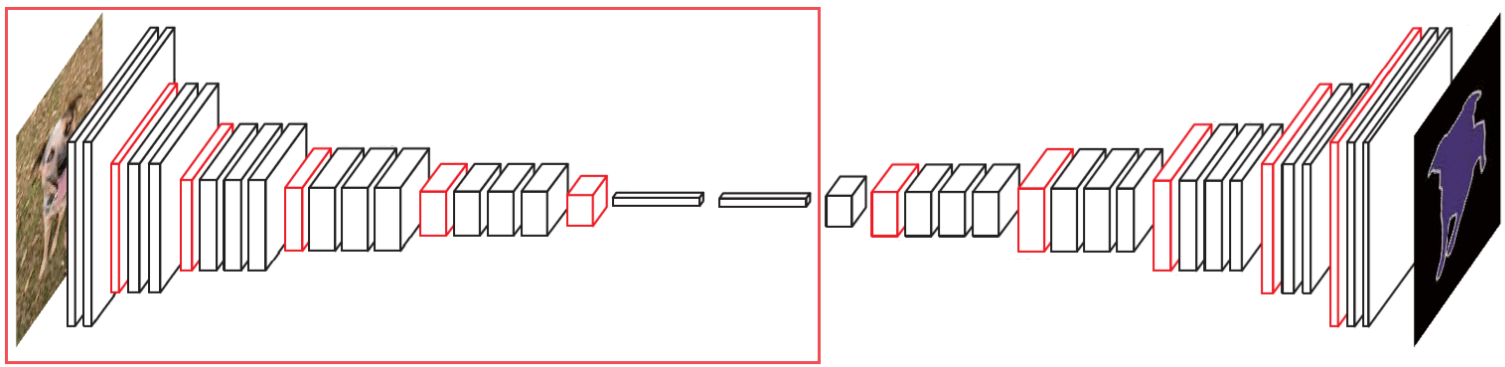
Source: modified from: Long et al. 2015

Encoder and Decoder

- Most methods use a classification network as basis

➔ If it can pickup the class, it should be able to learn the correct features

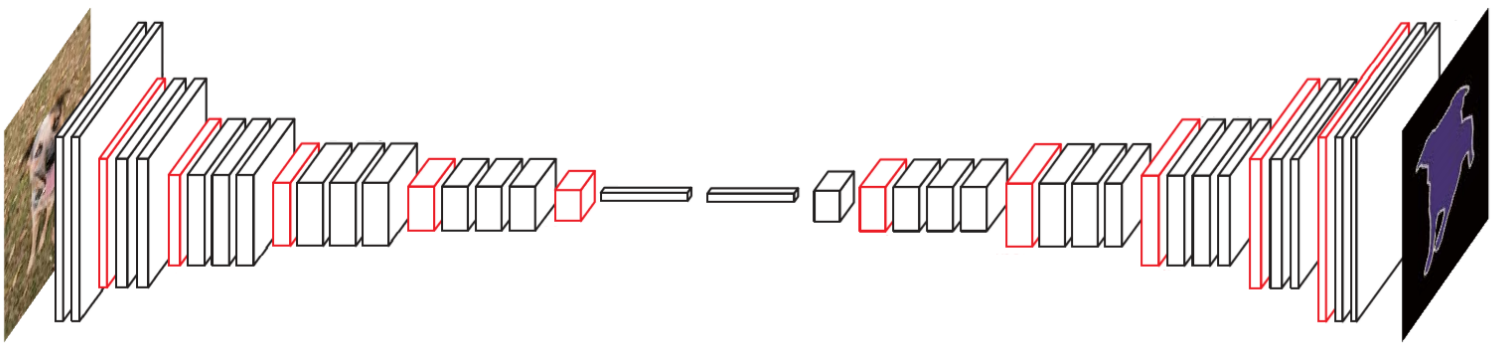
- Encoder part of the network



Source: modified from: Long et al. 2015

Encoder and Decoder (cont.)

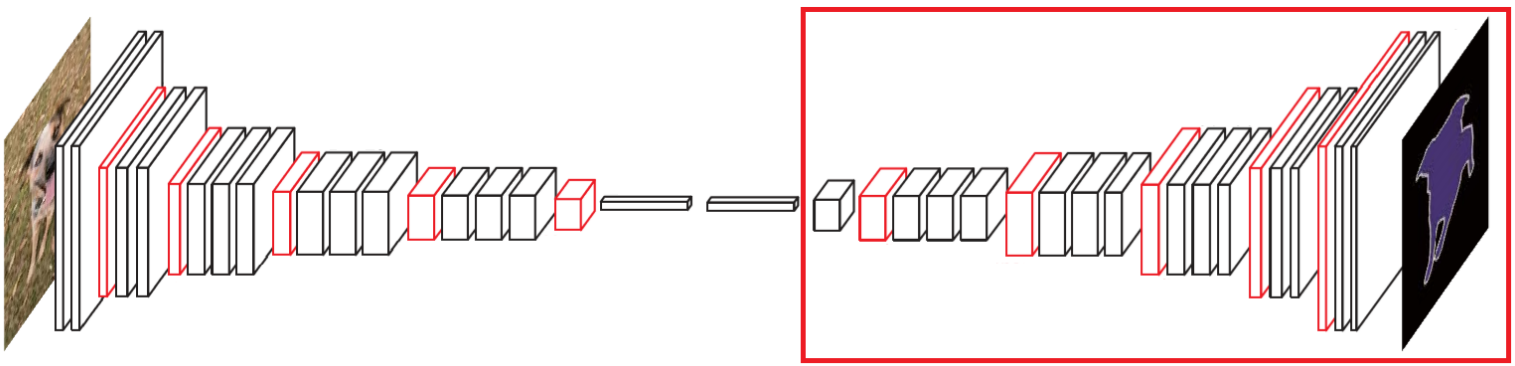
- Network has to learn to **decode** (map) the output of the encoder to pixel-wise predictions



Source: modified from: Long et al. 2015

Encoder and Decoder (cont.)

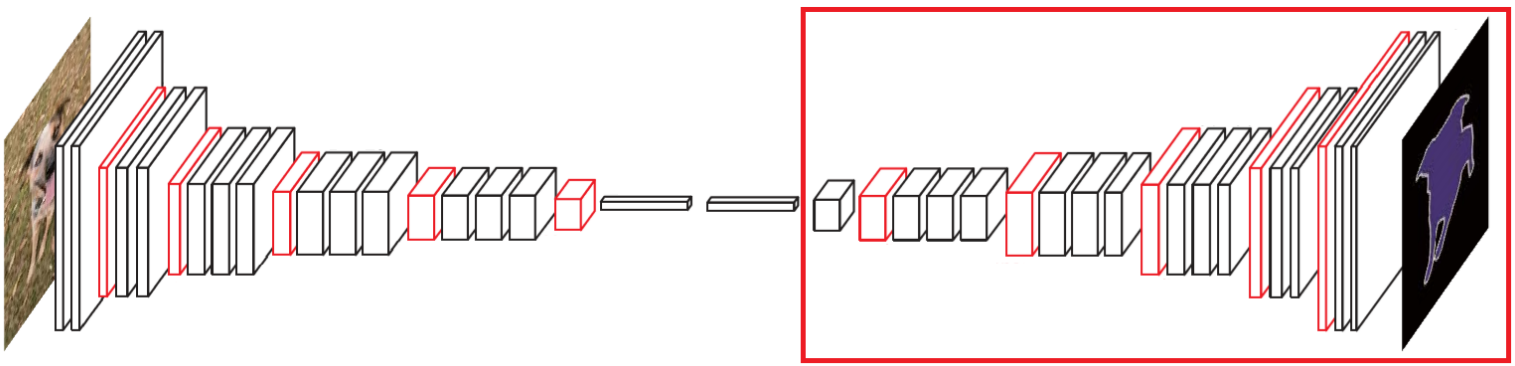
- Network has to learn to **decode** (map) the output of the encoder to pixel-wise predictions
- Decoder part of the network



Source: modified from: Long et al. 2015

Encoder and Decoder (cont.)

- Network has to learn to **decode** (map) the output of the encoder to pixel-wise predictions
- Decoder part of the network



Example networks with different decoders:

- Long et al.'s Fully Convolutional Networks [13]
- SegNet [1]
- U-net [21]

Source: modified from: Long et al. 2015

Upsampling

Upsampling Techniques

- Decoder requires upsampling to enable pixel-wise prediction

Upsampling Techniques

- Decoder requires upsampling to enable pixel-wise prediction
- Different options possible:

Upsampling Techniques

- Decoder requires upsampling to enable pixel-wise prediction
- Different options possible:
- Unpooling
  - Nearest Neighbor unpooling
  - Bed of Nails
  - Using max pooling indices

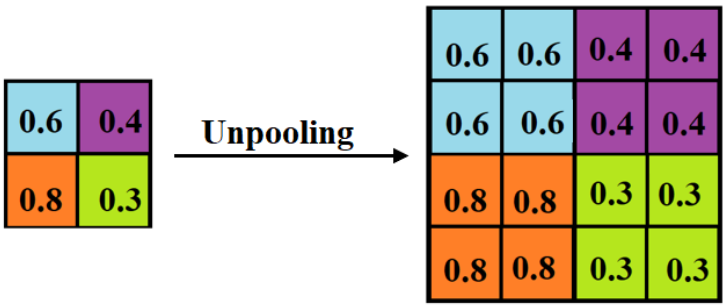
Upsampling Techniques

- Decoder requires upsampling to enable pixel-wise prediction
- Different options possible:
- Unpooling
  - Nearest Neighbor unpooling
  - Bed of Nails
  - Using max pooling indices
- Transpose convolution

Upsampling Techniques (cont.)

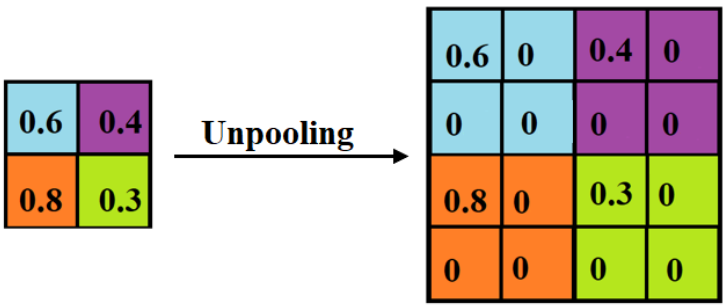
Nearest neighbor unpooling

Duplicate the value of the element for the region



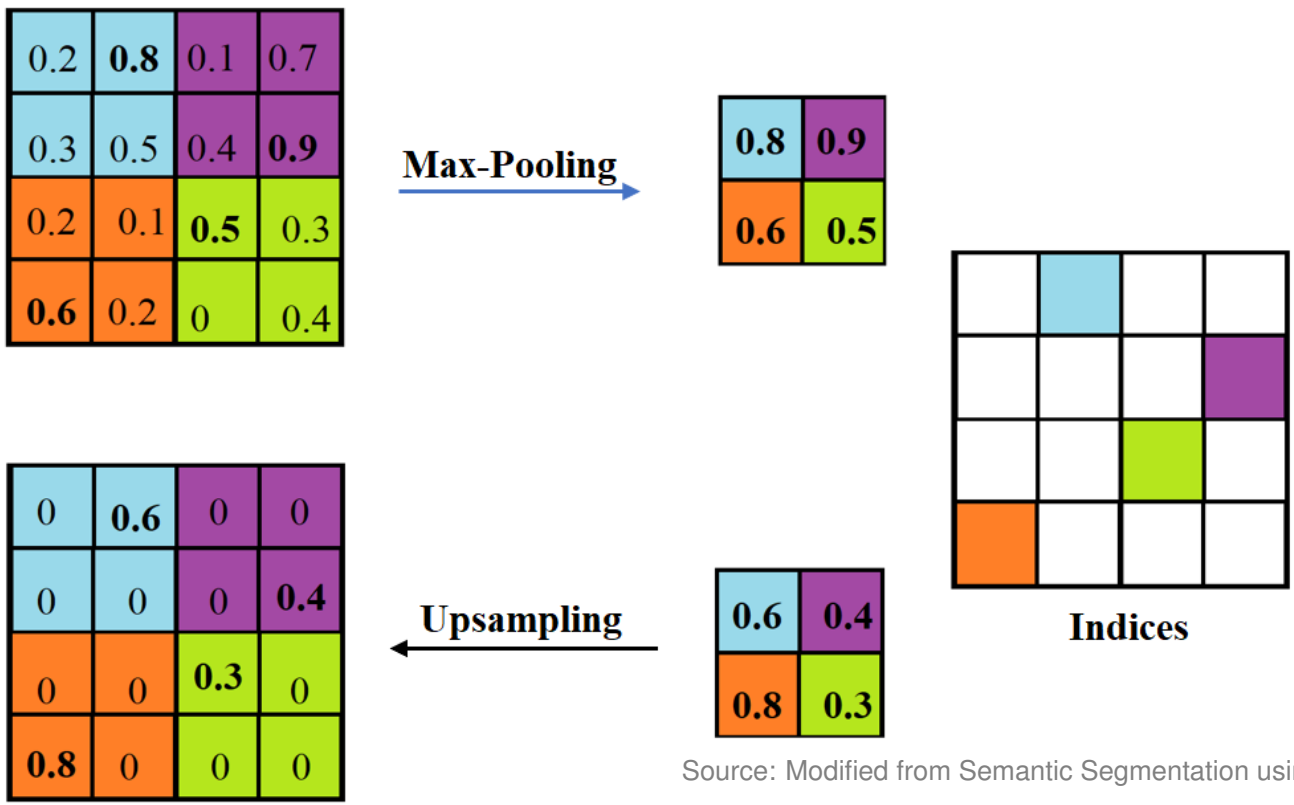
Bed of Nails

Take the one value and make the others zero



Upsampling Techniques

Using max pooling indices



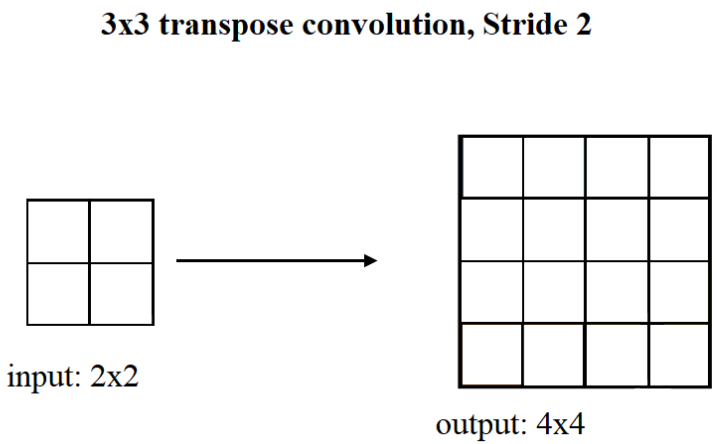
Source: Modified from Semantic Segmentation using FCN over the years

Upsampling Techniques: Transpose convolution

- Learnable upsampling (sometimes wrongly called "Deconvolution")

Upsampling Techniques: Transpose convolution

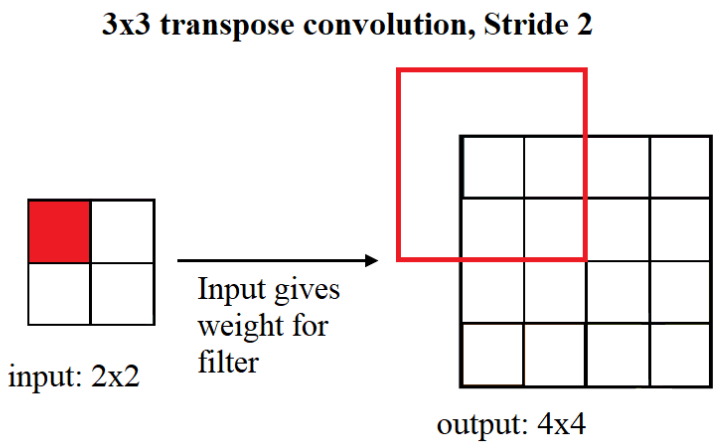
- Learnable upsampling (sometimes wrongly called "Deconvolution")
- Filter moves 2 pixel in the output for every 1 pixel in the input
- Higher upsampling can be controlled with stride



Upsampling Techniques: Transpose convolution

- Learnable upsampling (sometimes wrongly called "Deconvolution")
- Filter moves 2 pixel in the output for every 1 pixel in the input
- Higher upsampling can be controlled with stride
- Transpose convolution is backward pass of normal convolution

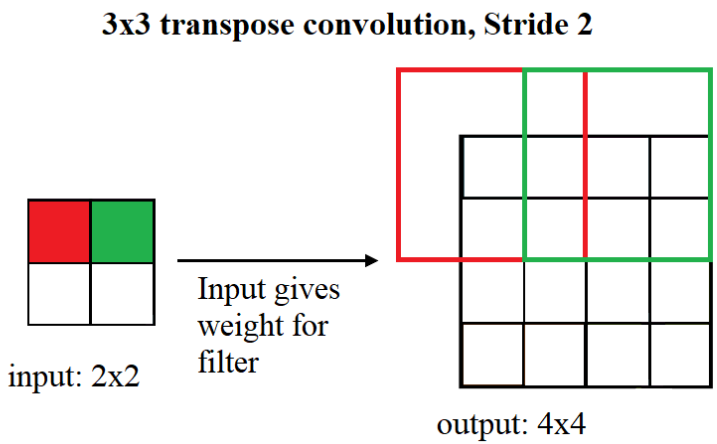
➔ Stride results in upsampling (“fractionally strided convolution”)



Upsampling Techniques: Transpose convolution

- Learnable upsampling (sometimes wrongly called "Deconvolution")
- Filter moves 2 pixel in the output for every 1 pixel in the input
- Higher upsampling can be controlled with stride
- Transpose convolution is backward pass of normal convolution

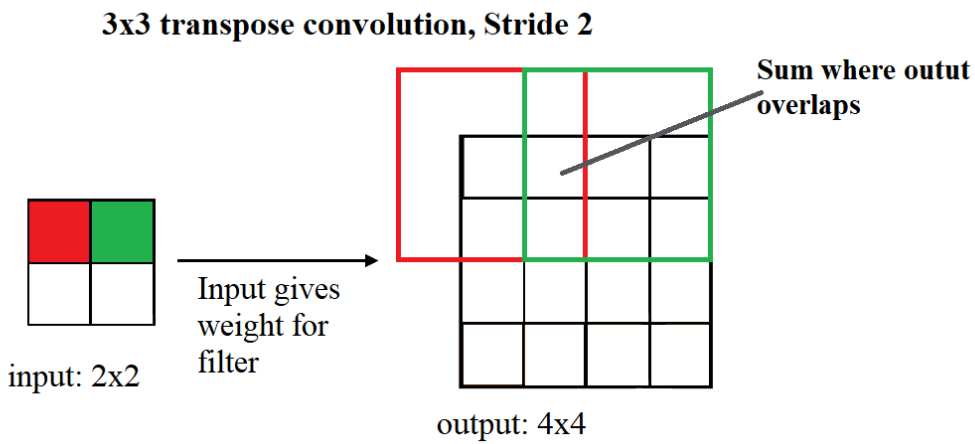
➔ Stride results in upsampling (“fractionally strided convolution”)



Upsampling Techniques: Transpose convolution

- Learnable upsampling (sometimes wrongly called "Deconvolution")
- Filter moves 2 pixel in the output for every 1 pixel in the input
- Higher upsampling can be controlled with stride
- Transpose convolution is backward pass of normal convolution

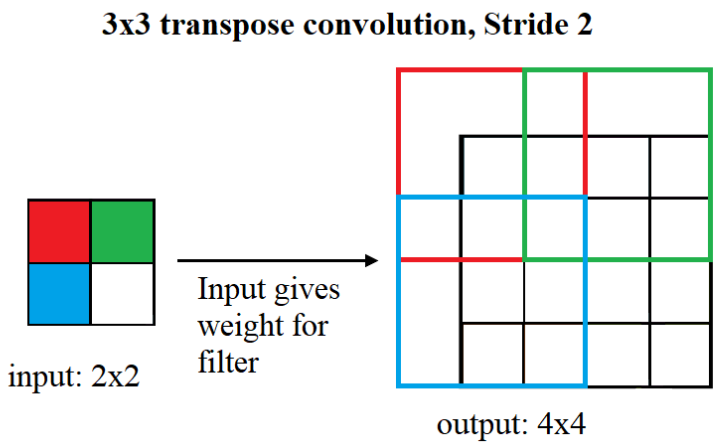
➔ Stride results in upsampling (“fractionally strided convolution”)



Upsampling Techniques: Transpose convolution

- Learnable upsampling (sometimes wrongly called "Deconvolution")
- Filter moves 2 pixel in the output for every 1 pixel in the input
- Higher upsampling can be controlled with stride
- Transpose convolution is backward pass of normal convolution

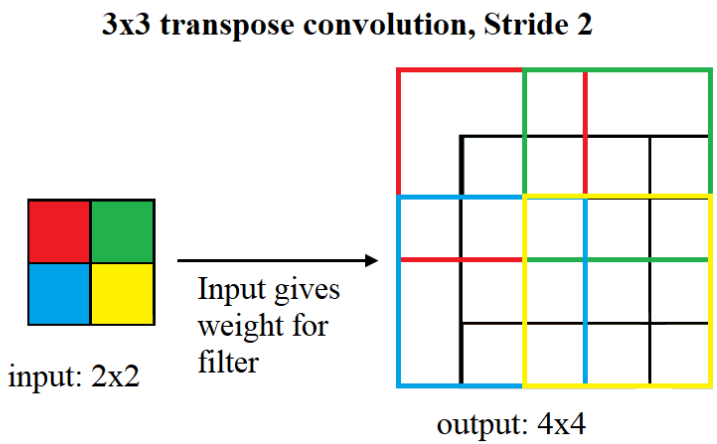
➔ Stride results in upsampling (“fractionally strided convolution”)



Upsampling Techniques: Transpose convolution

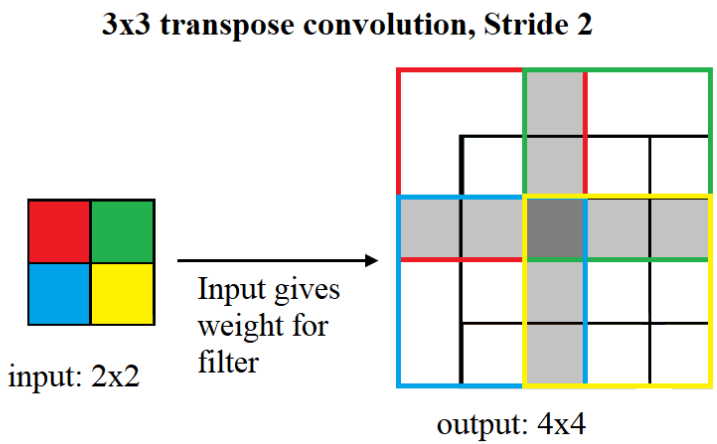
- Learnable upsampling (sometimes wrongly called "Deconvolution")
- Filter moves 2 pixel in the output for every 1 pixel in the input
- Higher upsampling can be controlled with stride
- Transpose convolution is backward pass of normal convolution

➔ Stride results in upsampling (“fractionally strided convolution”)



Upsampling Techniques: Checkerboard Artifacts

- Transpose convolution has uneven overlap when the kernel size is not divisible by the stride
- The uneven overlaps on the two axes multiply, creating a characteristic checkerboard artifact
- In principle, network could learn weights to avoid this but ...
- ... in practice, networks struggle to avoid it completely



Checkerboard Artifacts

How to avoid

- Choose appropriate kernel size: use a kernel size that is divisible by the stride, avoiding the overlap issue
- Separate upsampling from convolution to compute features
- For example:
  - Resize the image (e.g., using NN or bilinear interpolation)
  - Add a convolutional layer

Integrating Context Knowledge

Integrating Context Knowledge: Combining Where and What

- Semantic segmentation requires integration of information from various spatial scales

➔ Need to balance local and global information

- Local information crucial to achieve good pixel-level accuracy
- Global context of the image enables to resolve local ambiguities
- CNNs struggle with this balance

➔ Different approaches to integrate local & global information

Integrating Context Knowledge (cont.)

Long et al.'s Fully Convolutional Networks [13]

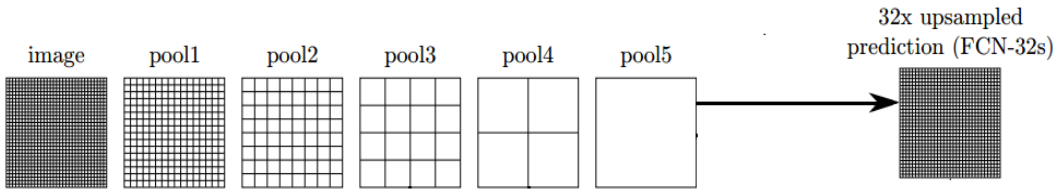
- Upsampling using learnable transposed convolutions
- Idea: Add links combining the final prediction and previous (lower) layers with finer strides
- Additional 1×1 convolution after pooling layer, predictions are added up

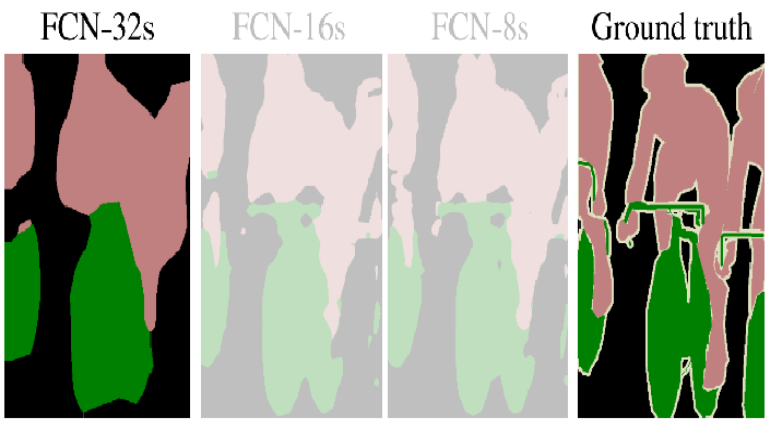
➔ Make local predictions with global structure

- Network topology is a directed acyclic graph (DAG), with **"skip connections"** from lower to higher layers
- Refinement of coarse segmentation

Integrating Context Knowledge (cont.)

Long et al.'s Fully Convolutional Networks [13]

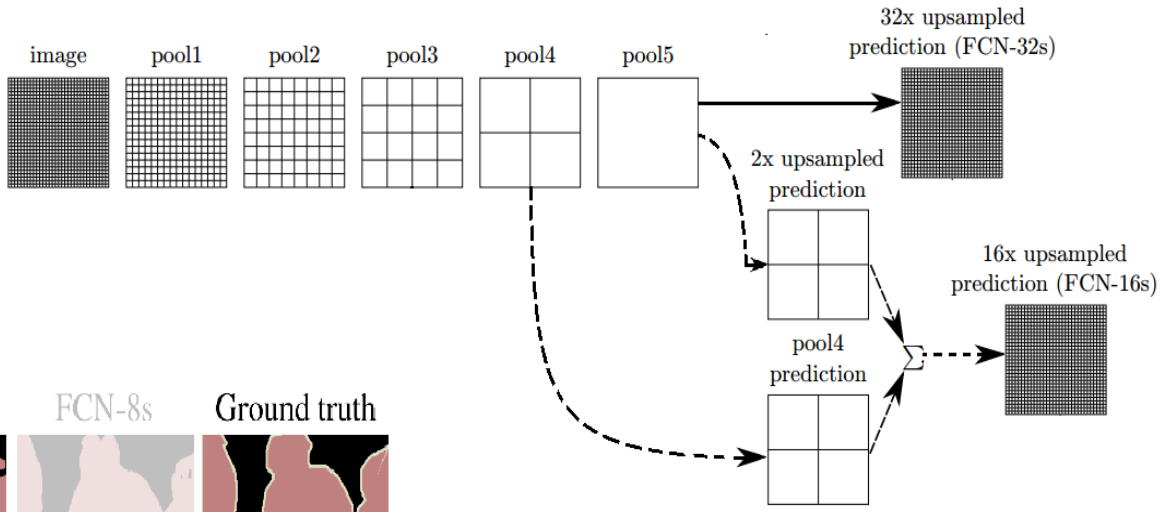


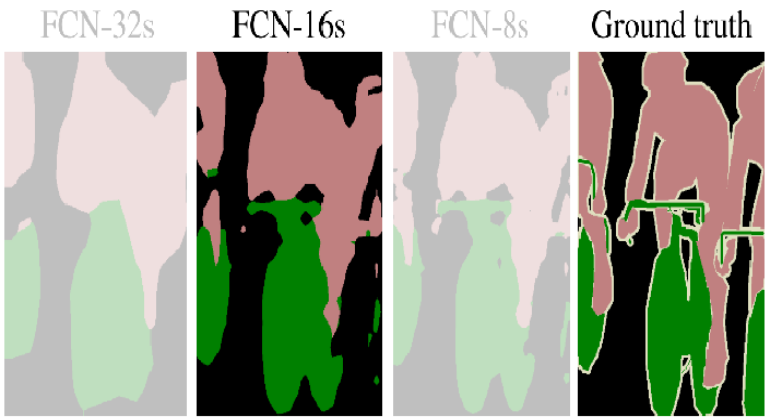


Source: Long et al. 2015

Integrating Context Knowledge (cont.)

Long et al.'s Fully Convolutional Networks [13]

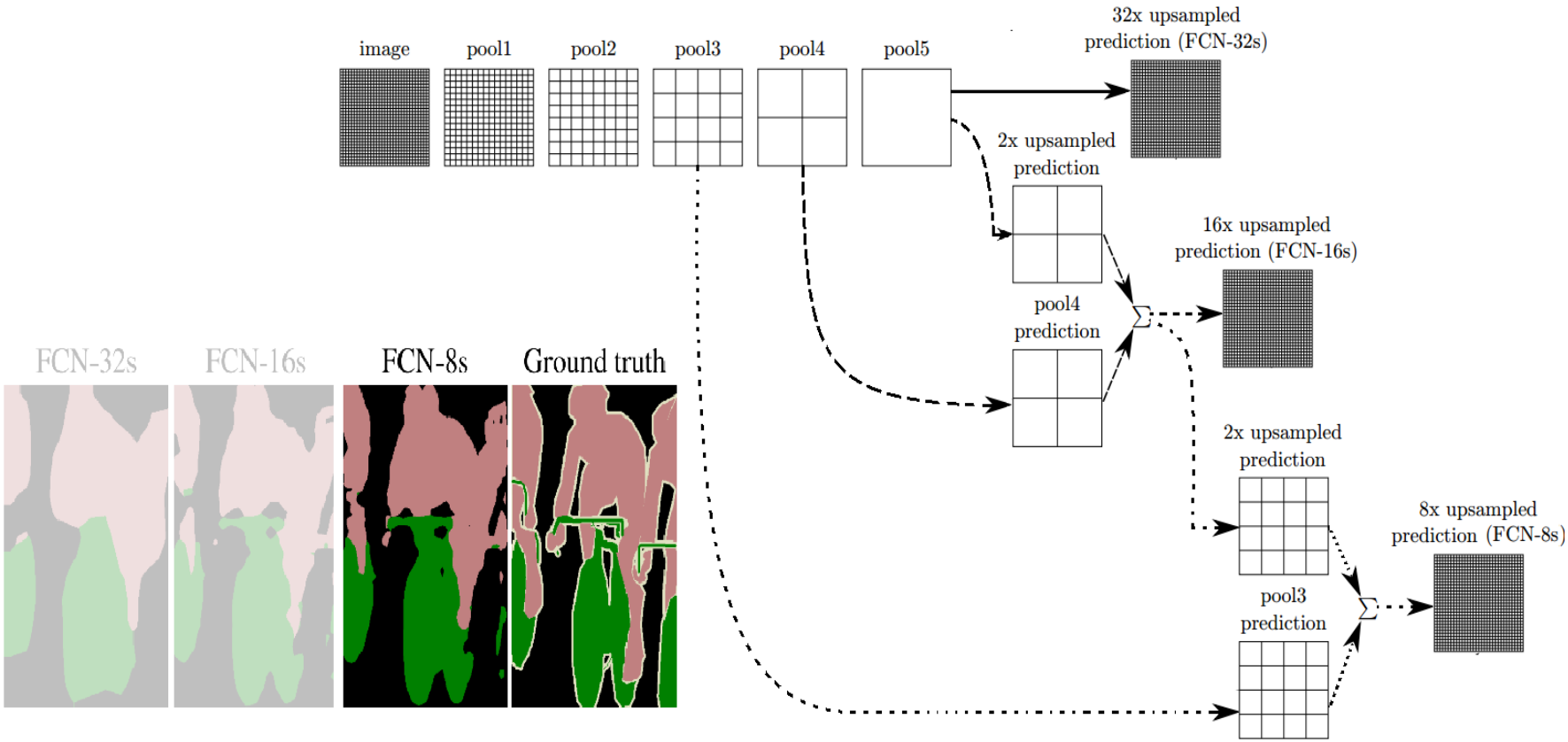




Source: Long et al. 2015

Integrating Context Knowledge (cont.)

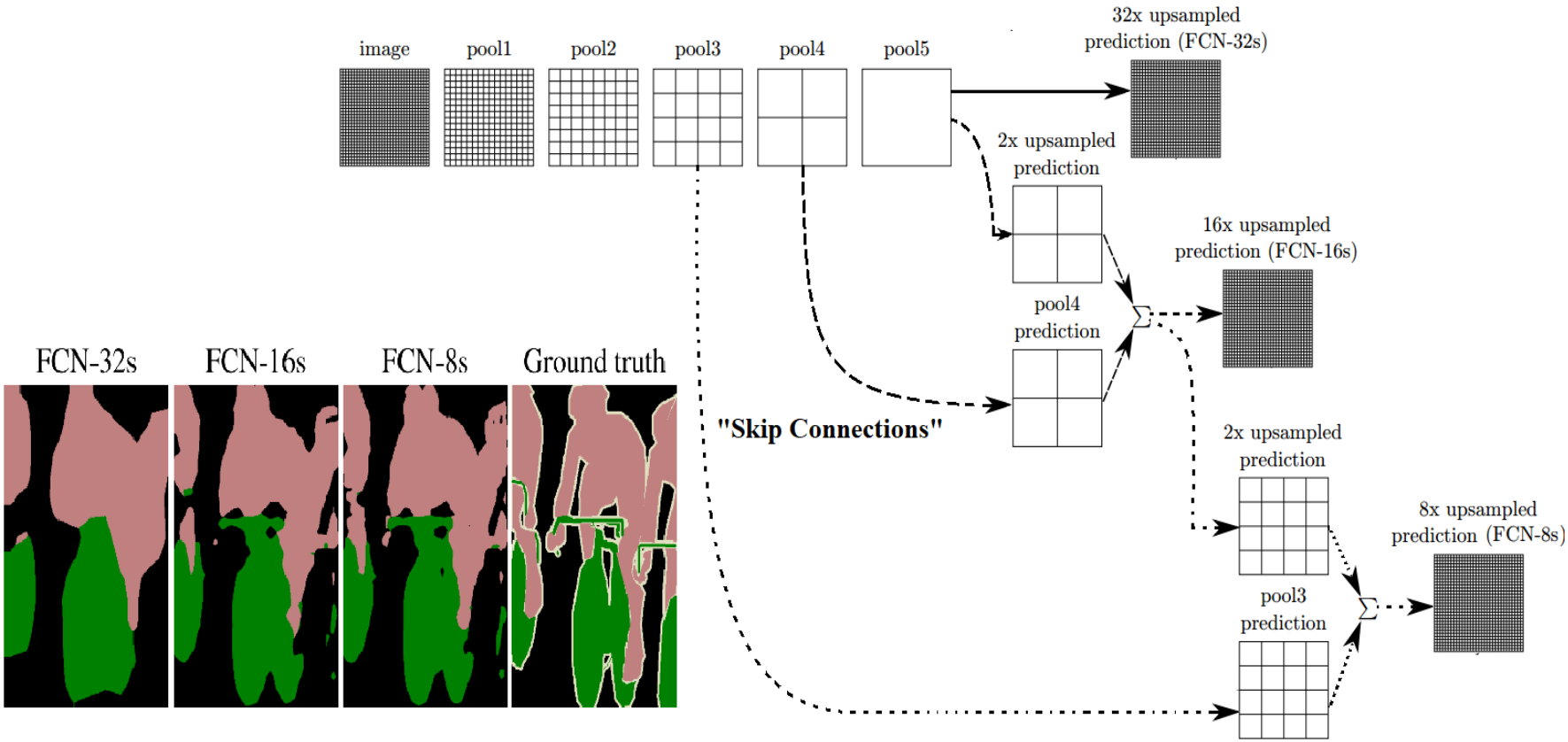
Long et al.'s Fully Convolutional Networks [13]



Source: Long et al. 2015

Integrating Context Knowledge (cont.)

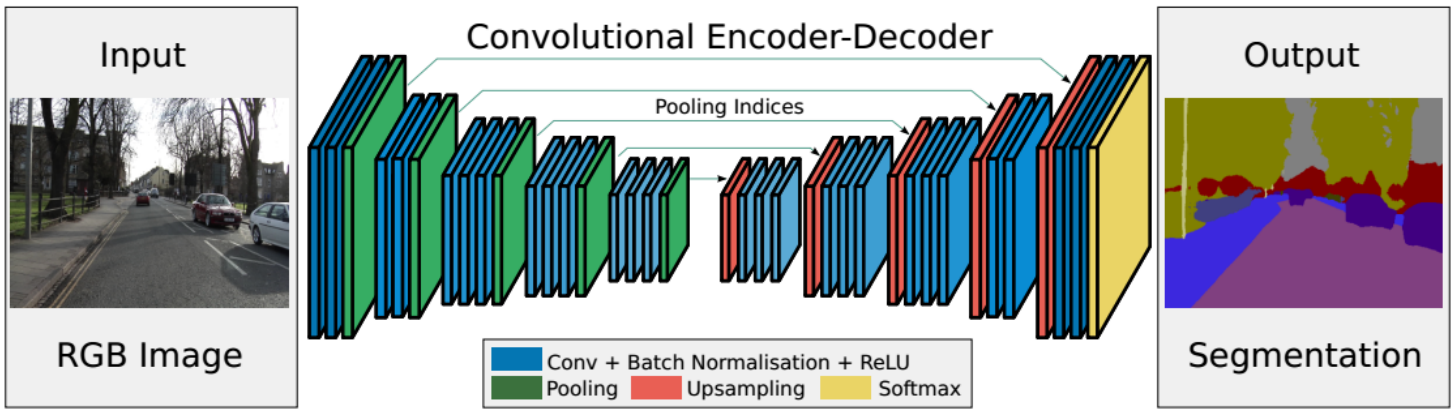
Long et al.'s Fully Convolutional Networks [13]



Source: Long et al. 2015

Integrating Context Knowledge (cont.)

SegNet [1]:



- Decoder: Upsampling & convolution layers followed by softmax
- Each upsampling layer corresponds to a **max-pooling layer** in encoder
- Upsampling reuses max-pooling indices from feature maps in encoder

➔ provides context information

Source: Badrinarayanan et al. 2015

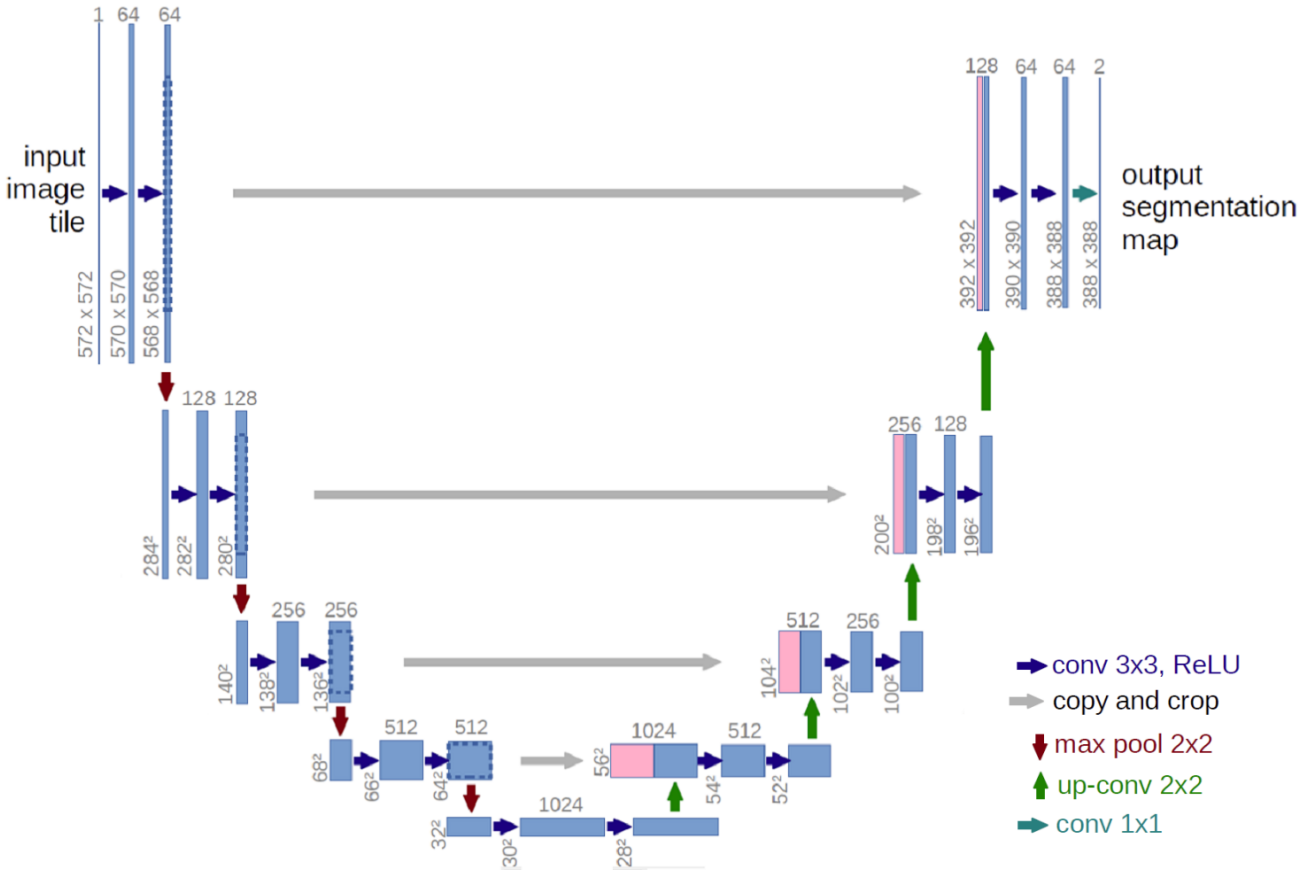
Integrating Context Knowledge (cont.)

U-Net [21]

- The network consists of:
  - Encoder: A contracting path to capture context
  - Decoder: A **symmetric** expansion path for localization (decoder)
- Encoder follows typical architecture of a CNN
- Decoder (expansion path) consists of
  - **Upsampling of feature maps** followed by a 2×2 convolution that halves the number of feature channels.
  - **Concatenation** with the corresponding cropped feature map from the contracting path
- The training strategy relies on use of data augmentation
  - Non rigid deformation
  - Rotation & translation

Integrating Context Knowledge (cont.)

U-Net [21]



Source: Ronneberger et al. 2015

Additional Approaches

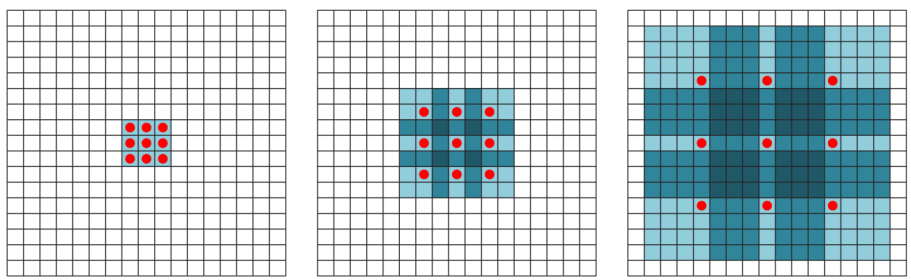
So far: Combination of feature maps from different levels + upsampling

Alternatives & extensions:

- Dilated convolutions
- Multi-scale networks
- Defer context modeling to another network, e.g. an RNN
- Refinement as a post-processing step with Conditional Random Fields (CRFs)

Additional Approaches: Dilated convolutions

- Also named a-trous convolutions
- Support exponentially expanding receptive fields without losing resolution
- The dilation rate L controls the upsampling factor
- Stacking $L-$dilated convolutions makes the receptive fields grow exponentially while the number of parameters for the filters grows linear



(a) 1-dilated (b) 2-dilated (c) 3-dilated

Source: Fisher and Vladlen, 2015

Additional Approaches: Dilated convolutions (cont.)

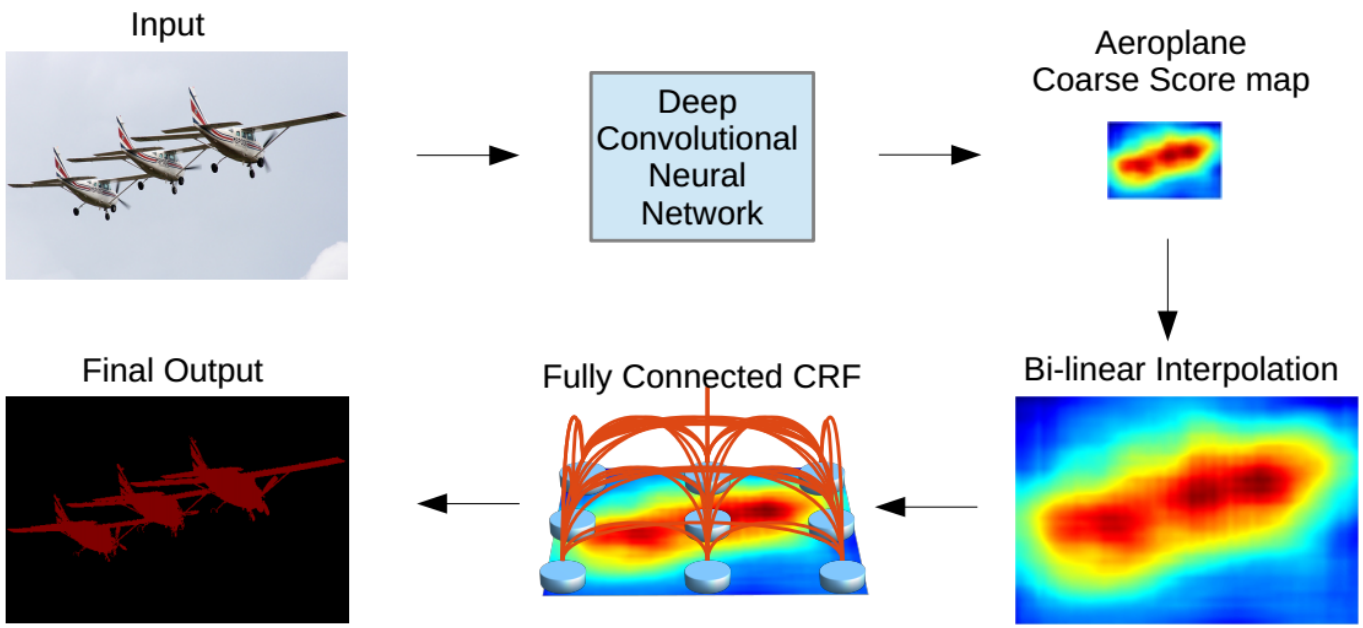
Examples

- DeepLab (improved version) [4]
- ENet [17]
- Multi scale context aggregation module [28]

Main issue: No efficient implementation available, benefit somewhat unclear

Additional Approaches: Conditional Random Fields

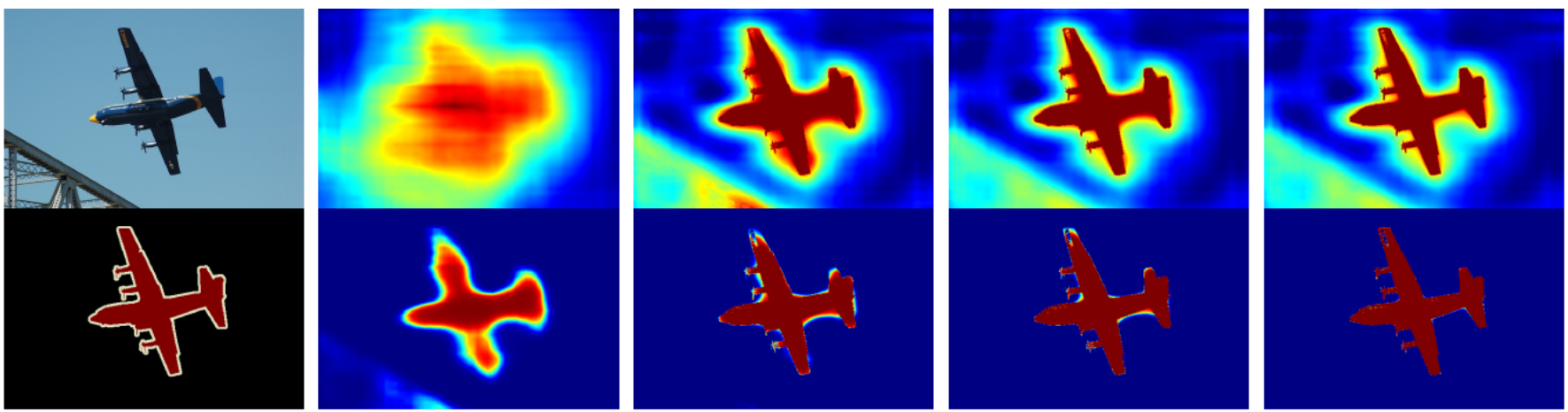
- Idea: Refine coarse CNN output using Conditional Random Field (CRF)
- Each pixel is modeled as a node in the random field, pairwise term between pixels
- Can capture long-range dependencies and fine local information



Source: Chen et al., 2014

Additional Approaches: Conditional Random Fields

- Example: DeepLab [3]
- Iterative CRF refinement of segmentation
- Improvement with extensions (e.g., atrous convolutions) in [4]
- Also possible: Modeling CRF with RNNs [29] ➔ end-to-end training

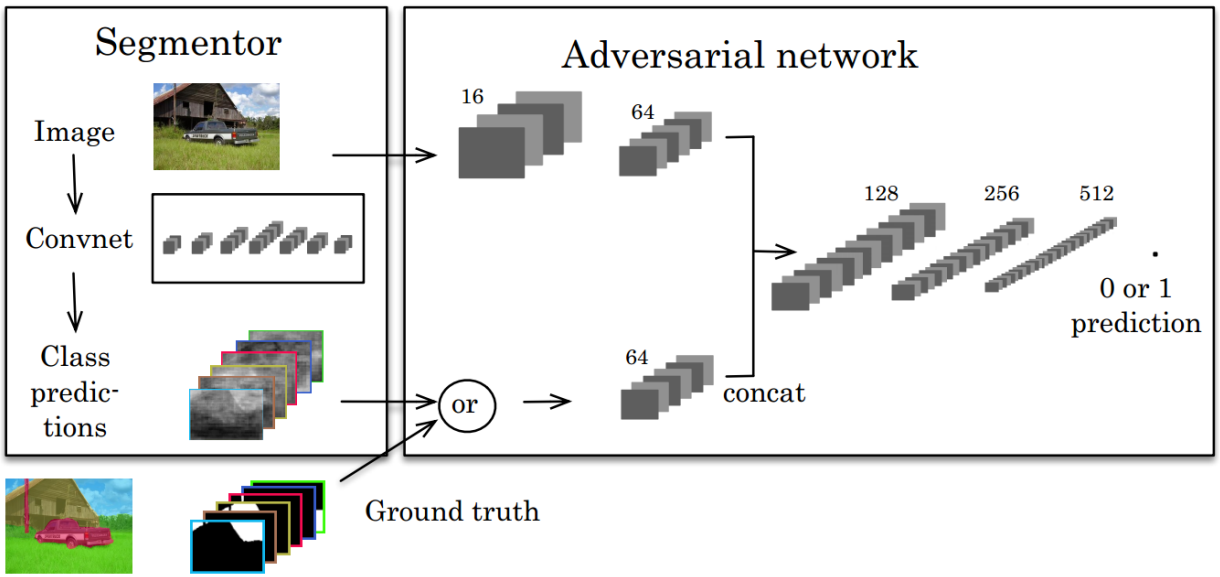


First row: inputs before softmax function, second row: output after softmax function.

Source: Chen et al., 2014

Advanced Topics

Adversarial Networks for Segmentation [14]



Source: Luc et al., 2016

Optimize Segmentor with hybrid loss function with two terms:

1.) Usual pixel-wise multi-class cross-entropy for semantic segmentation

2.) Loss based on min-max game with Discriminator

Adversarial Networks for Segmentation [14] (cont.)

- Given a data set of N training images $\mathbf{x}_n$ and a corresponding label maps $\mathbf{y}_n$ the loss function defined as:

$$l(\boldsymbol{\theta}_s, \boldsymbol{\theta}_a) = \sum_{n=1}^{N} l_{mce}(s(\mathbf{x}_n), \mathbf{y}_n) - \lambda \left[l_{bce}(a(\mathbf{x}_n, \mathbf{y}_n), 1) + l_{bce}(a(\mathbf{x}_n, s(\mathbf{x}_n)), 0) \right]$$

- $(\mathbf{x}_n, \mathbf{y}_n), n \in \{1, \ldots, N\}$: Data set with ground truth labels
- $\boldsymbol{\theta}_s$: Parameters of the Segmentor $s(\mathbf{x}, \mathbf{y})$
- $\boldsymbol{\theta}_a$: Parameters of the Discriminator $a(\mathbf{x}, \mathbf{y})$

Adversarial Networks for Segmentation [14] (cont.)

- Given a data set of N training images $\mathbf{x}_n$ and a corresponding label maps $\mathbf{y}_n$ the loss function defined as:

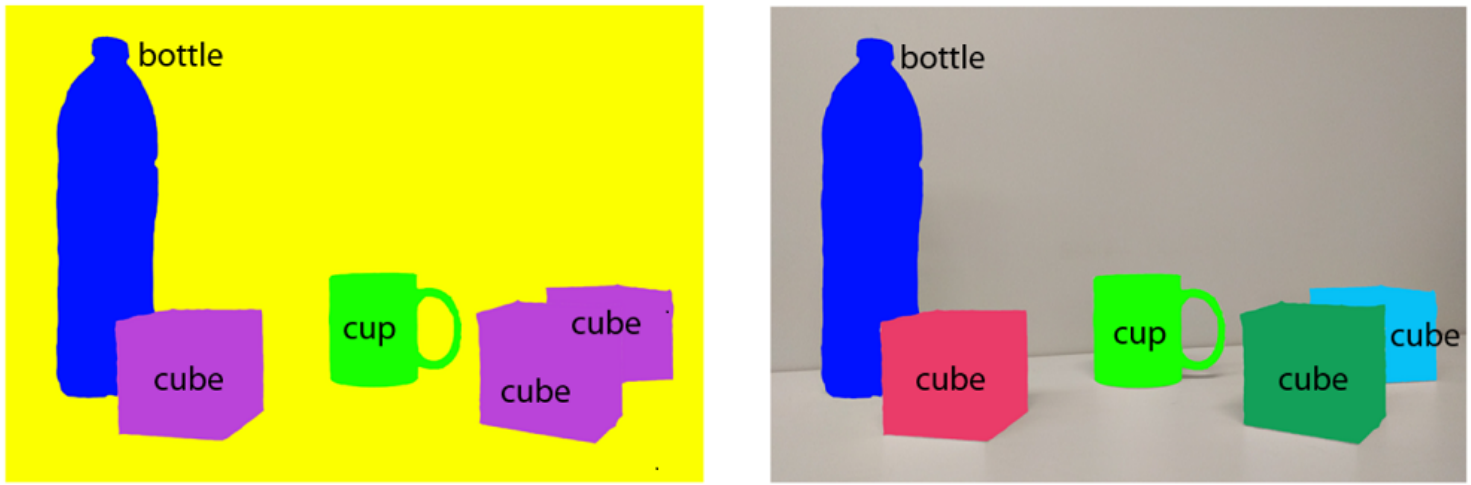
$$l(\theta_s, \theta_a) = \sum_{n=1}^{N} \underbrace{l_{mce}(s(\mathbf{x}_n), \mathbf{y}_n)}_{\text{multi-class cross-entropy}} - \lambda \underbrace{\left[l_{bce}(a(\mathbf{x}_n, \mathbf{y}_n), 1) + l_{bce}(a(\mathbf{x}_n, s(\mathbf{x}_n)), 0)\right]}_{\text{adverserial loss} \rightarrow \text{binary classification}}$$

- $(\mathbf{x}_n, \mathbf{y}_n), n \in \{1, \ldots, N\}$: Data set with ground truth labels
- θ_s: Parameters of the Segmentor $s(\mathbf{x}, \mathbf{y})$
- θ_a: Parameters of the Discriminator $a(\mathbf{x}, \mathbf{y})$

➔ Segmentor is trained both on the segmentation and on fooling the discriminator

➔ **Multi-task learning** with adversarial task

Instance Segmentation



Semantic segmentation

Instance segmentation

Source: Garcia et al. 2017

- Next step after semantic segmentation
- Main goal: detect **different instances** of the same class
- Number of instances initially unknown, pixel-wise prediction as in semantic segmentation not sufficient

➔ Combination of **object detection** and **semantic segmentation**

Instance Segmentation

Examples for potential applications:

- Information about occlusion
- Counting number of elements belonging to the same class
- Detecting object boundaries, e.g., for gripping objects in robotics

Examples in literature:

- Simultaneous Detection and Segmentation (SDS) [9]
- DeepMask [18]
- SharpMask [19]
- Mask R-CNN [10]

Instance Segmentation Example: Mask R-CNN [10]

- Back to the “start”: Combine object detection (R-CNN) with segmentation
- Object detection solves instance separation, segmentation refines bounding boxes per instance

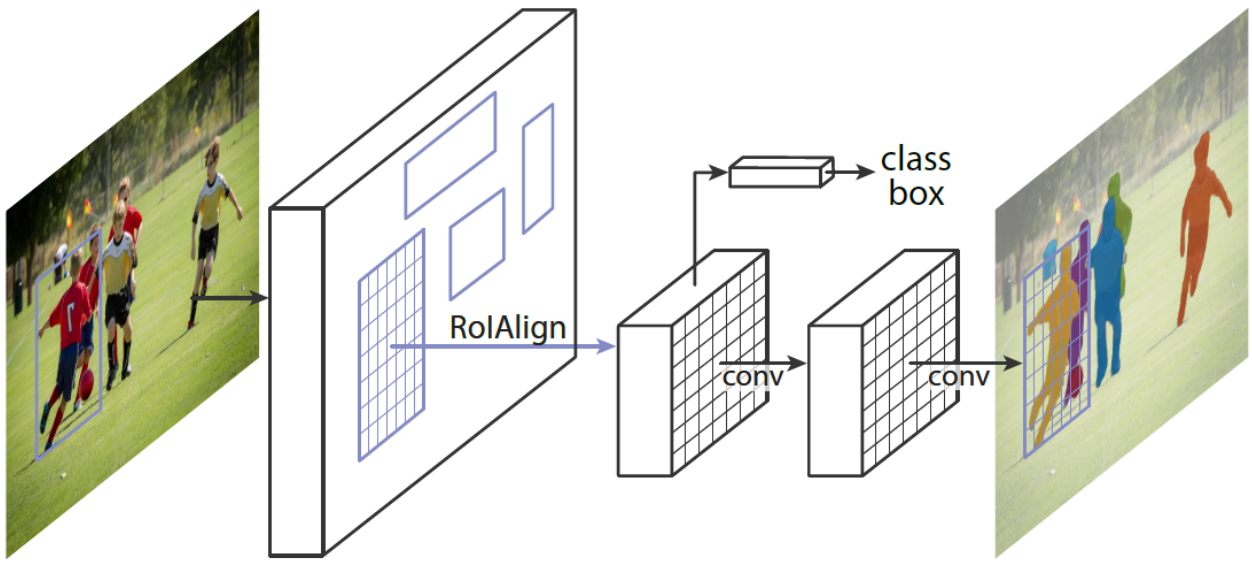
Source: He et al., 2017 [10]

Instance Segmentation Example: Mask R-CNN [10] (cont.)

Workflow: Two-stage procedure

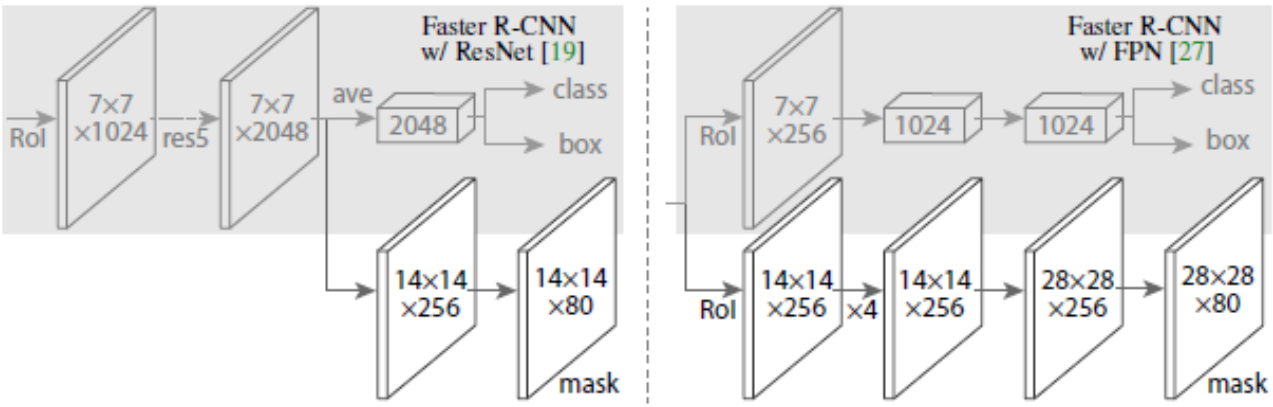
1. Region proposal: proposes candidate object bounding boxes
2. Classification, bounding-box regression **and** segmentation in parallel

➔ **Multi-task loss**: $L = L_{cls} + L_{box} + L_{class}$



Source: He et al., 2017 [10]

Instance Segmentation Example: Mask R-CNN [10] (cont.)



Options for stage 2 networks

- Segmentation is independent of classification ➔ only binary loss computed for correct class
- Second-stage networks can be arbitrarily coupled

➔ **Multi-task loss**: $L = L_{cls} + L_{box} + L_{class}$

Source: He et al., 2017 [10]

Instance Segmentation Example: Mask R-CNN [10] (cont.)

Examples for instance segmentation

Summary

- Fully convolutional networks preserve spatial layout and enable arbitrary input sizes combined with pooling
- Object detectors can be implemented as a sequence of region proposals and classification. Examples are the R-CNN family of networks.
- Alternatively one can go all YOLO and perform single-shot object detection with the region proposals networks
- Segmentation is commonly solved by architectures analyzing the image and subsequently refining coarse results
- Object detection and segmentation are closely related and combinations are common

NEXT TIME

ON DEEP LEARNING

Coming Up

- Methods to relieve the burden of labeling
- Integration of known operators

Comprehensive Questions

- What is the difference between semantic and instance segmentation and what is the connection to object detection?
- How can we construct a network which accepts arbitrary input sizes?
- What is ROI pooling?
- How can we perform backpropagation through a ROI pooling layer?
- What are typical measures for evaluation of segmentations?
- What similarities do typical autoencoders share with typical segmentation networks?
- Explain a method for instance segmentation.

Further Reading

- Link - Joseph Redmons awesome website including the DarkNet library and material about YOLO
- Link - Joseph Redmons CV - have a look at it - will definitely jumpstart your career!

References

References I

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: arXiv preprint arXiv:1511.00561 (2015). arXiv: 1311.2524.

[2] Xiao Bian, Ser Nam Lim, and Ning Zhou. “Multiscale fully convolutional network with application to industrial inspection”. In: Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on. IEEE. 2016, pp. 1–8.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”. In: CoRR abs/1412.7062 (2014). arXiv: 1412.7062.

References II

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: arXiv preprint arXiv:1606.00915 (2016).

[5] S. Ren, K. He, R. Girshick, et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: vol. 39. 6. June 2017, pp. 1137–1149.

[6] R. Girshick. “Fast R-CNN”. In: 2015 IEEE International Conference on Computer Vision (ICCV). Dec. 2015, pp. 1440–1448.

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, et al. “Focal loss for dense object detection”. In: arXiv preprint arXiv:1708.02002 (2017).

References III

[8] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, et al. “A Review on Deep Learning Techniques Applied to Semantic Segmentation”. In: arXiv preprint arXiv:1704.06857 (2017).

[9] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, et al. “Simultaneous detection and segmentation”. In: European Conference on Computer Vision. Springer. 2014, pp. 297–312.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, et al. “Mask R-CNN”. In: CoRR abs/1703.06870 (2017). arXiv: 1703.06870.

[11] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Rec Vol. 1. June 2005, 886–893 vol. 1.

References IV

[12] Jonathan Huang, Vivek Rathod, Chen Sun, et al. “Speed/accuracy trade-offs for modern convolutional object detectors”. In: CoRR abs/1611.10012 (2016). arXiv: 1611.10012.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recogni 2015, pp. 3431–3440.

[14] Pauline Luc, Camille Couprie, Soumith Chintala, et al. “Semantic segmentation using adversarial networks”. In: arXiv preprint arXiv:1611.08408 (2016).

[15] Christian Szegedy, Scott E. Reed, Dumitru Erhan, et al. “Scalable, High-Quality Object Detection”. In: CoRR abs/1412.1441 (2014). arXiv: 1412.1441.

References V

[16] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning deconvolution network for semantic segmentation”. In: Proceedings of the IEEE International Conference on Computer Vision. 2015, pp. 1520–1528.

[17] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, et al. “Enet: A deep neural network architecture for real-time semantic segmentation”. In: arXiv preprint arXiv:1606.02147 (2016).

[18] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. “Learning to segment object candidates”. In: Advances in Neural Information Processing Systems. 2015, pp. 1990–1998.

[19] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, et al. “Learning to refine object segments”. In: European Conference on Computer Vision. Springer. 2016, pp. 75–91.

References VI

[20] Ross B. Girshick, Jeff Donahue, Trevor Darrell, et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: CoRR abs/1311.2524 (2013). arXiv: 1311.2524.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: MICCAI. Springer. 2015, pp. 234–241.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: Computer Vision – ECCV 2014. Cham: Springer International Publishing, 2014, pp. 346–361.

[23] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, et al. “Selective Search for Object Recognition”. In: International Journal of Computer Vision 104.2 (Sept. 2013), pp. 154–171.

References VII

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al. “SSD: Single Shot MultiBox Detector”. In: Computer Vision – ECCV 2016. Cham: Springer International Publishing, 2016, pp. 21–37.

[25] P. Viola and M. Jones. “Rapid object detection using a boosted cascade of simple features”. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vis Vol. 1. 2001, pp. 511–518.

[26] J. Redmon, S. Divvala, R. Girshick, et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2016, pp. 779–788.

[27] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: CoRR abs/1612.08242 (2016). arXiv: 1612.08242.

References VIII

[28] Fisher Yu and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions”. In: arXiv preprint arXiv:1511.07122 (2015).

[29] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, et al. “Conditional Random Fields as Recurrent Neural Networks”. In: CoRR abs/1502.03240 (2015). arXiv: 1502.03240.