Neural Networks

Semantic Segmentation

(P-ITEEA-0011)

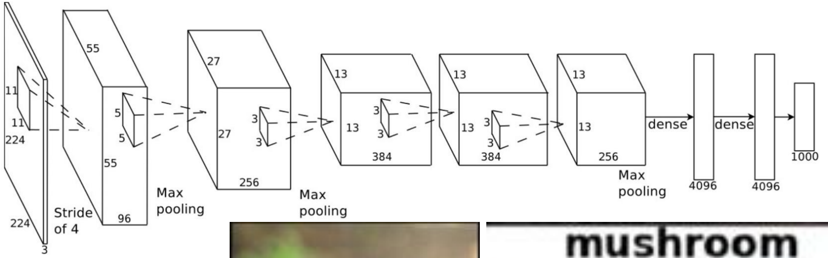
Akos Zarandy
Lecture 8
November 12, 2019

Announcement

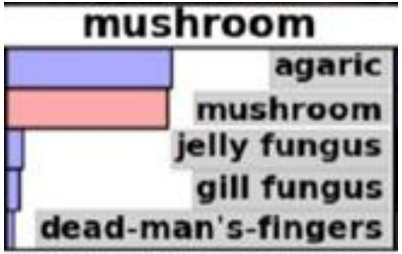
- Midterm project were taken by many people ☺
- Midterm project counts for those
  - Paper based test result is 5
    - Can get offered grade 4 or 5 based on the quality of the midterm project solution
  - Paper based test result is 5
    - Can get offered grade 3 only if the quality of the midterm project solution is satisfactory
    - One can go for better grade in exam period
- If somebody changes his/her mind about midterm project after this announcement, he or she has to write a letter to Soma Kontar **today!**

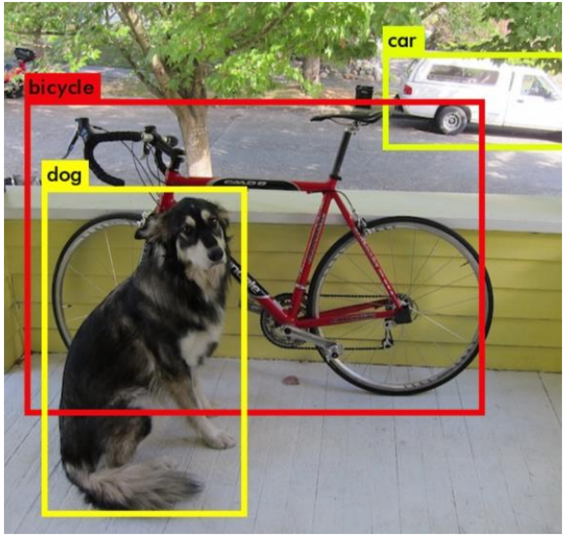
Short quiz 60% required!

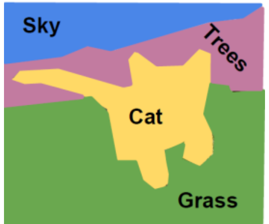
Recap



- Last Lecture we discussed
  - How to do image classification
    - Alexnet
    - One decision per image (*classification*)
  - Detection and Localization is more complex
    - Multiple (few) decision per image
      - *Regressions* for localization
      - *Classification* for detection
  - Pixel level Segmentation
    - Very high number of decisions (*classification*) per image





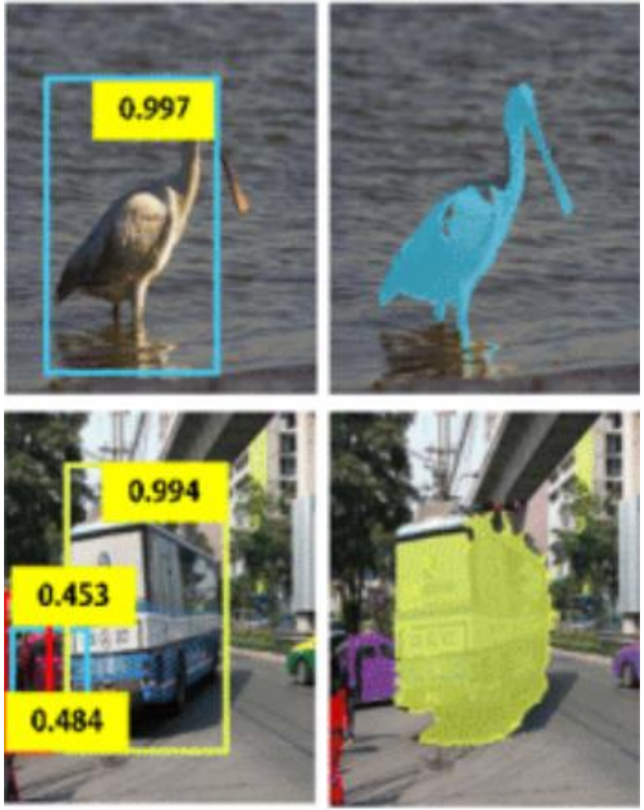


Contents

- Detection and Localization
  - PASCAL Database and Competion
  - R-CNN
    - Region proposal, Classification
    - Support Vector Machine (SVN), Bounding box refinement
  - Fast R-CNN
  - Faster R-CNN
- Semantic Image Segmentation
  - U-Net
  - DeConvNet
  - SegNet
    - Resolution controlling
    - Atrous convolutions, sub-pixel image combination

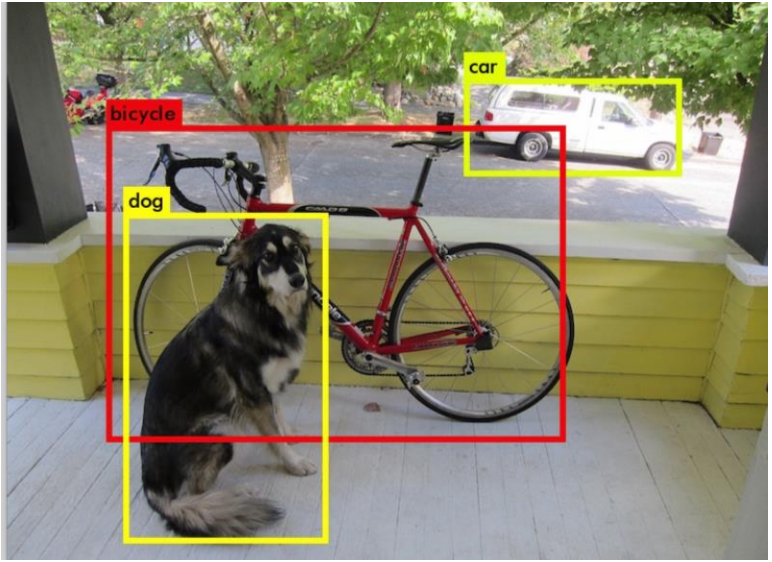
The PASCAL Object Recognition Database and Challange

- Annotated image database
  - Detection (squared objects)
  - Segmentation (segmented objects)
- Challenge
  - The PASCAL Visual Object Classes Challenge (PASCAL VOC)



Object detection/localization and classification

- Chicken and egg problem
  - You need to know that it is a bicycle before able to say that both a wheel part and a pipe segment belongs to the same object
  - You need to know that the red box contains an object before you can recognize it. (Cannot recognize a bicycle if you try it from separated parts)
- Our brain does it parallel
- How neural nets can solve it?
  - Detection by regression?
  - Detection by classification?



Detection as Regression?

(finding bounding box coordinates)

DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers

Detection as Regression?

(finding bounding box coordinates)

DOG, (x, y, w, h)
CAT, (x, y, w, h)

= 8 numbers

Detection as Regression?

(finding bounding box coordinates)

CAT, (x, y, w, h)
CAT, (x, y, w, h)
....
CAT (x, y, w, h)

= many numbers

Need variable sized outputs

Detection as Classification

(classify the content of each bounding boxes)

CAT? NO

DOG? NO

Detection as Classification

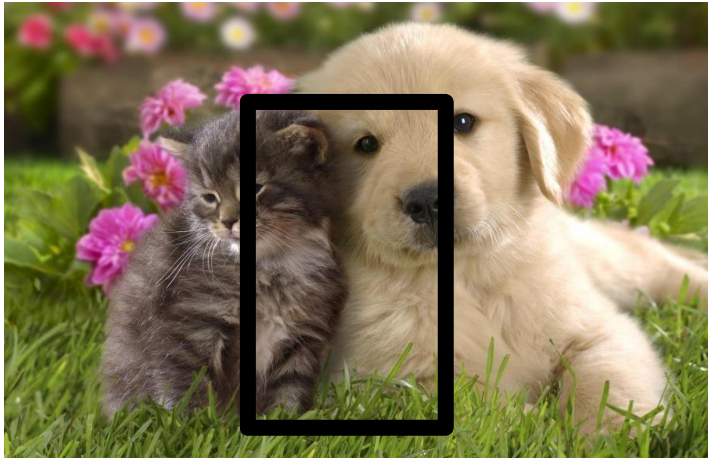
(classify the content of each bounding boxes)

CAT? YES!

DOG? NO

Detection as Classification

(classify the content of each bounding boxes)



CAT? NO

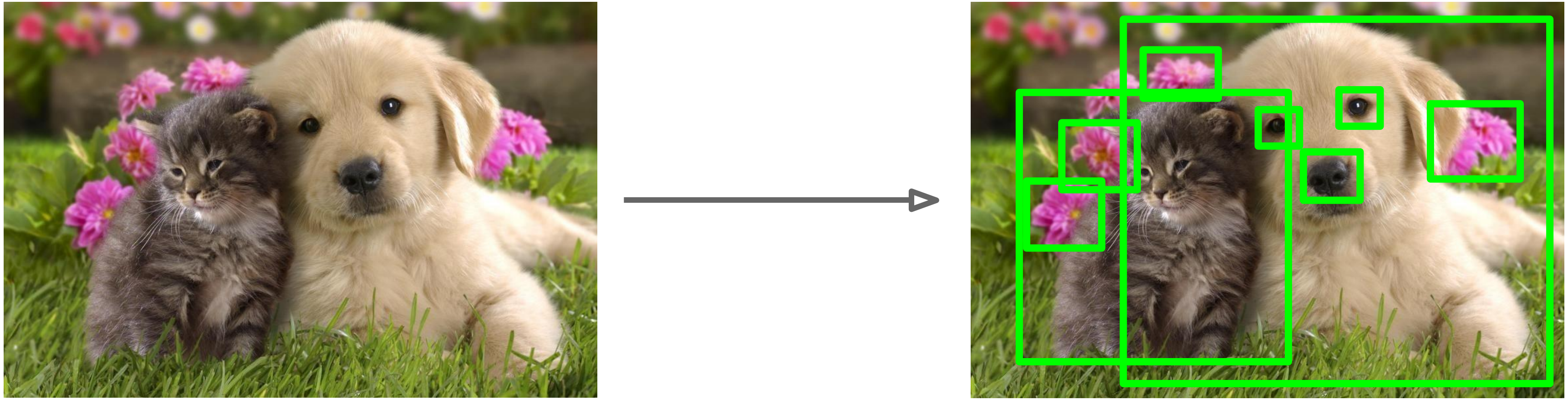
DOG? NO

Problem: Need to too test many positions and scales, and use a computationally demanding classifier (CNN)

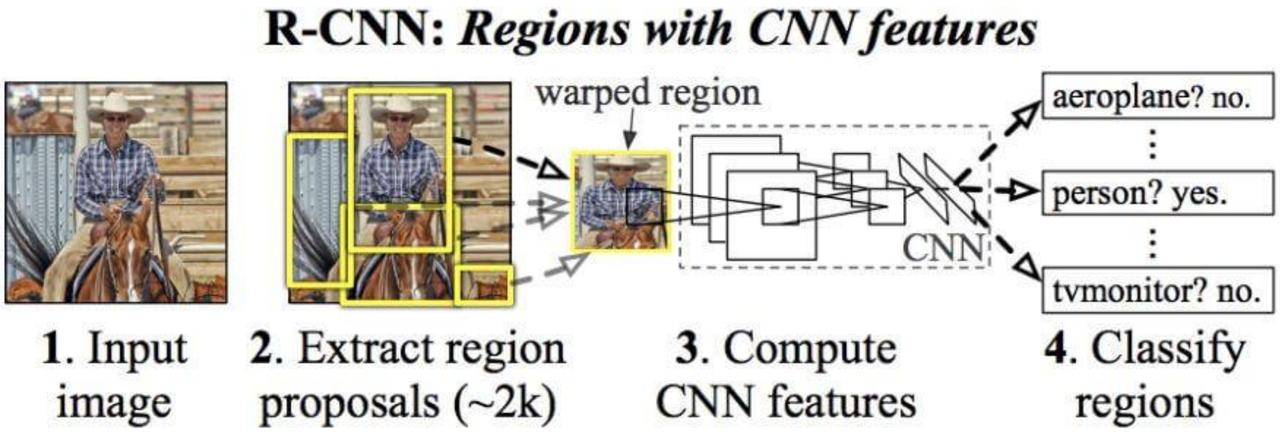
Solution: Only look at a tiny subset of possible positions

Region Proposals

- Find "blobby" image regions that are likely to contain objects
- "Class-agnostic" object detector
- Look for "blob-like" regions

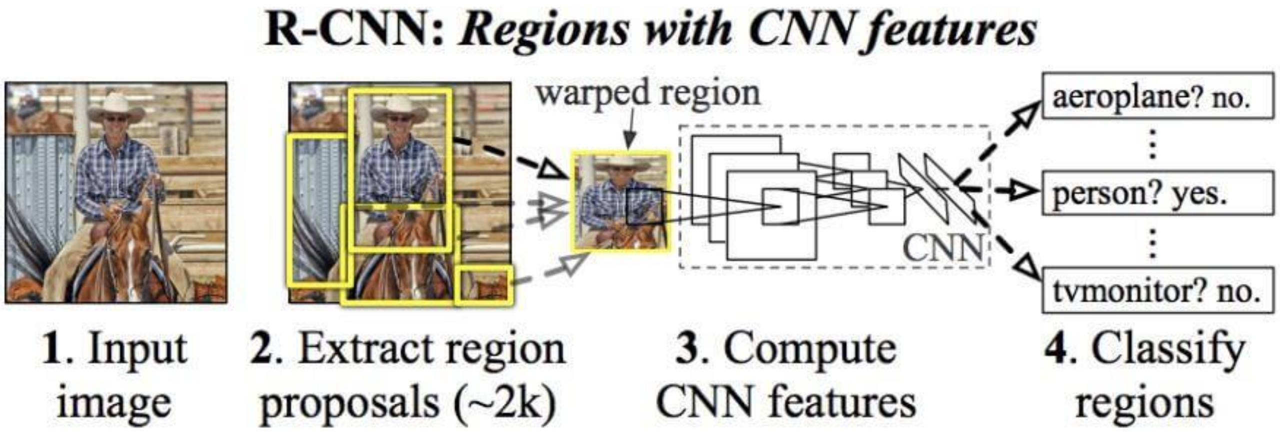


R-CNN in a Glance



1. Input image
2. Region proposals
3. Compute CNN features with warped images
4. Classification with Support Vector Machine (SVM)
5. Ranking/selecting/merging → detections
6. Bounding box regression

The R-CNN algorithm



1. Input image
2. Region proposals
3. Compute CNN features with warped images
4. Classification with Support Vector Machine (SVM)
5. Ranking/selecting/merging → detections
6. Bounding box regression

R-CNN: Region Proposal

- Requirements:
  - Propose a large number (up to 2000) of regions (boxes) with different sizes
  - Still much better than exhausting search with multi-scale sliding window (brute force)
  - Boxes should contain all the candidate objects with high probability
- R-CNN works with various Region proposal methods:
  - Objectness
  - Constrained Parametric Min-Cuts for Automatic Object Segmentation
  - Category Independent Object Proposals
  - Randomized Prim
  - Selective Search
- Selective Search is the fastest and provides best regions

R-CNN: Selective Search I

- Graph based segmentation (Felzenszwalb and Huttenlocher method)
  - cannot be used in this form, because one object is covered with multiple segments, moreover regions for occluded objects will not be covered
- Idea: oversegment it and apply scaled similarity based merging

Input image

Segmented image

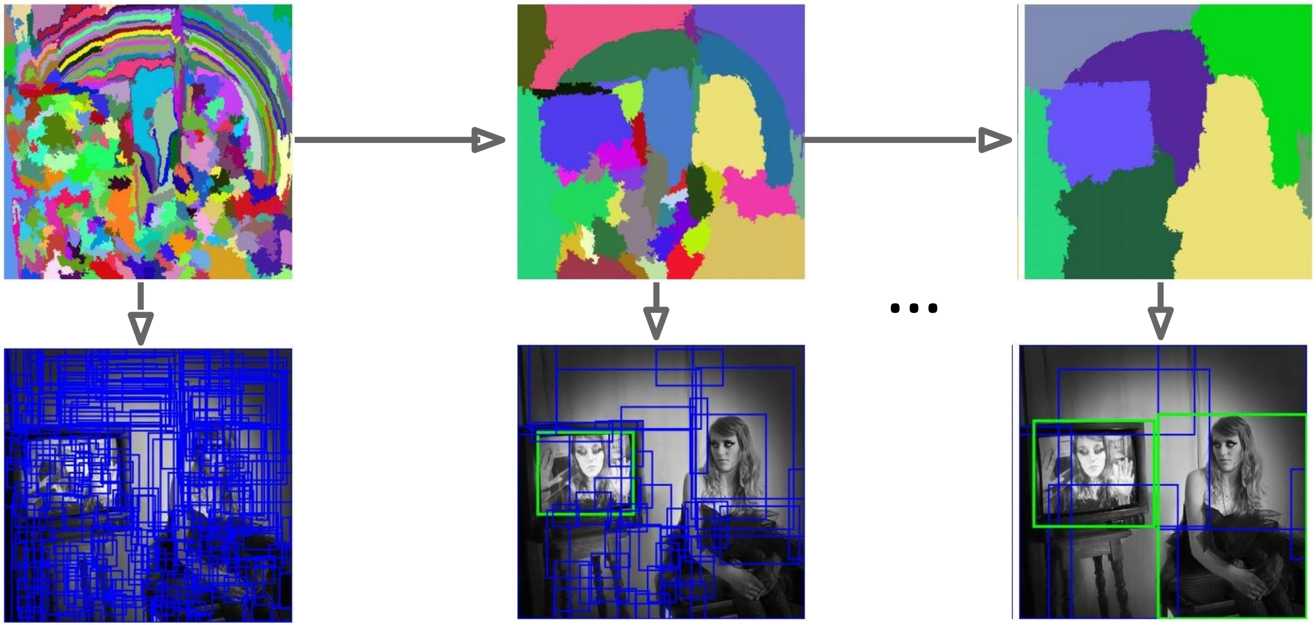
Oversegmented image

 https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/

R-CNN: Selective Search II

Step-by-step merging regions at multiple scales based on similarities

Convert regions to boxes



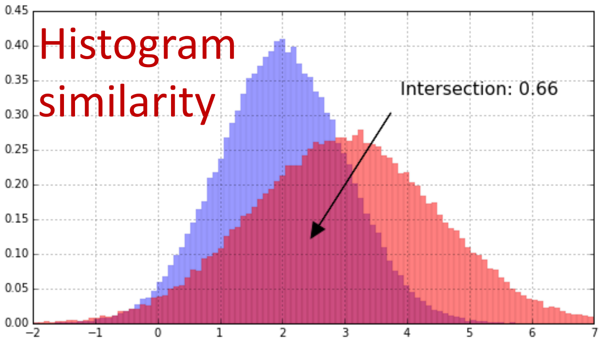
Original fine scale   Step one merging   Step *n* merging

Similarity measures I

Color Similarity

- Generate color histogram of each segment (descriptor)
  - 25 bins/ color channels
  - Descriptor vector (c_i^k)size: 3x25=75
- Calculate histogram similarity for each region pair

$$s_{color}(r_i, r_j) = \sum_{k=1}^{75} \min(c_i^k, c_j^k)$$



c_i^k is the histogram value for the k^{th} bin in color descriptor

Texture Similarity

- Texture features: Gaussian derivatives at 8 orientations in each pixel
  - 10 bins/color channels
  - Descriptor vector (t_i^k)size: 3x10x8=240
- Each region will have a texture histogram
- Calculate histogram similarity for each region pair

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{240} \min(t_i^k, t_j^k)$$

t_i^k is the histogram value for the k^{th} bin in texture descriptor

https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/

Similarity measures II

Size Similarity

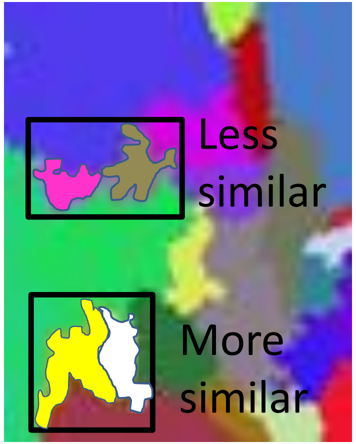
- Helps merging the smaller sized objects
- Since we do bottom up merging, the small segments will be merged first, because their size similarity score is higher

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(image)}$$

$size(image)$ is the size of the entire image in pixels

Shape Similarity

- Measures how well two regions are fit
  - How close they are
  - How large is the overlap



$$s_{fill}(r_i, r_j) = \\ = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(image)}$$

$size(BB_{ij})$ is the size of the bounding box of r_i and r_j

Similarity measures III

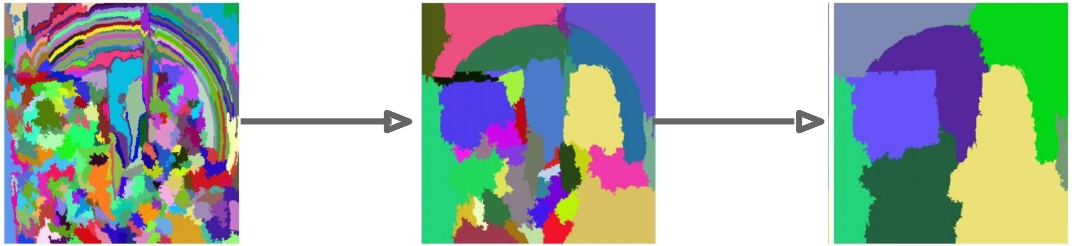
Final Similarity

- Linear combination of the four similarities

$$s_{final}(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{shap}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

List or proposed region

1. Initial oversegmentation
2. Calculation the similarities
3. Merge the similar regions
4. The formed regions are added to the region list (*this ensures that there will be smaller and larger regions in the list as well*)
5. Goto 2



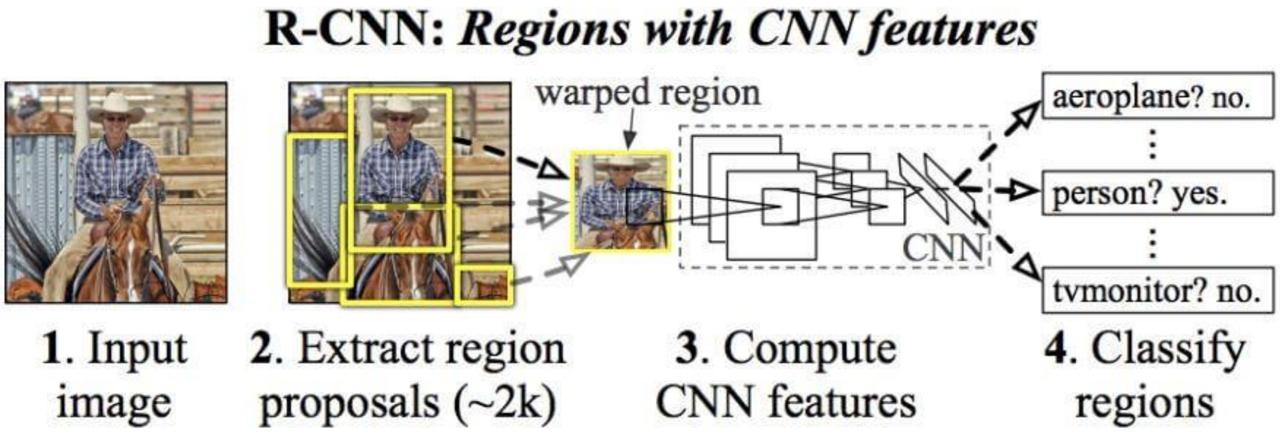
https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/

Proposed regions

- Few hundreds or few thousand boxes
- Includes all the objects with high probability
- Number of the boxes are much smaller than with brute force method
- C and python functions exist

https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/

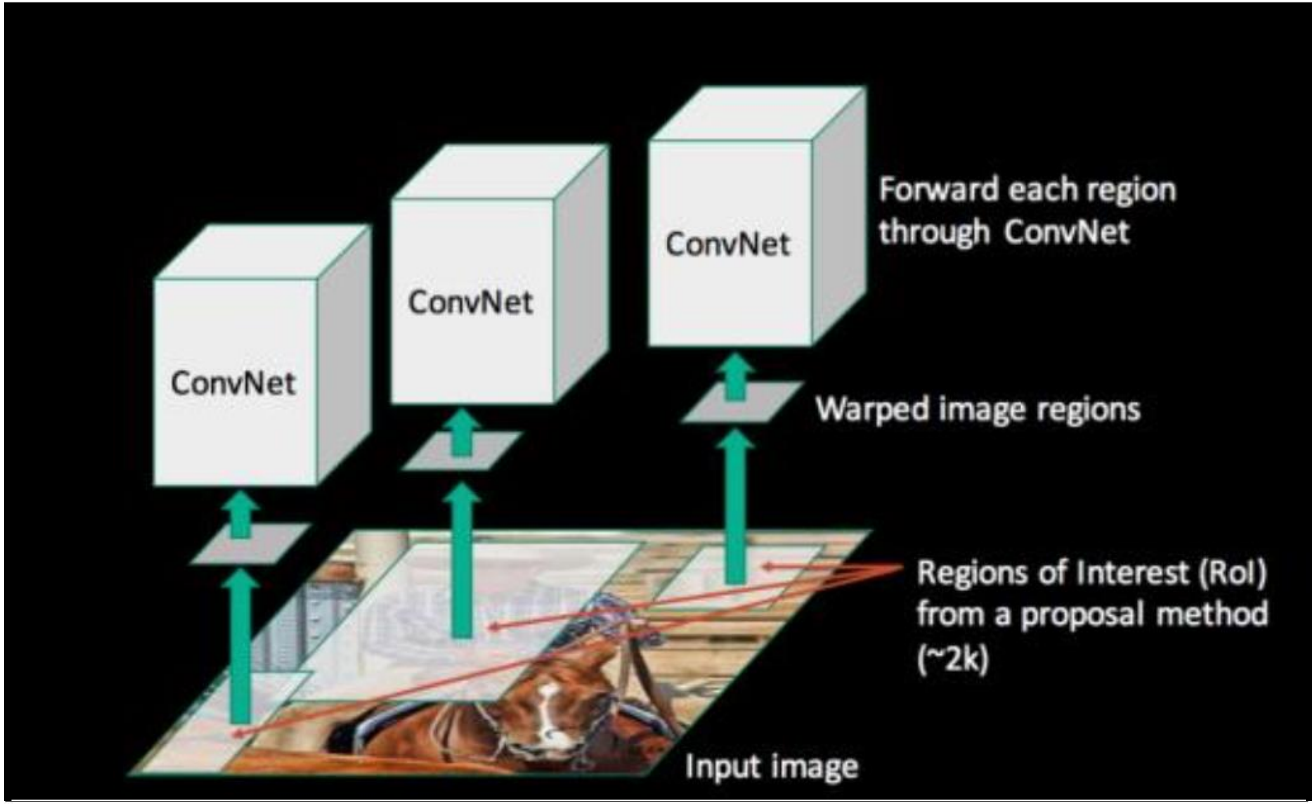
The R-CNN algorithm



1. Input image
2. Region proposals
3. Compute CNN features with warped images
4. Classification with Support Vector Machine (SVM)
5. Ranking/selecting/merging → detections
6. Bounding box regression

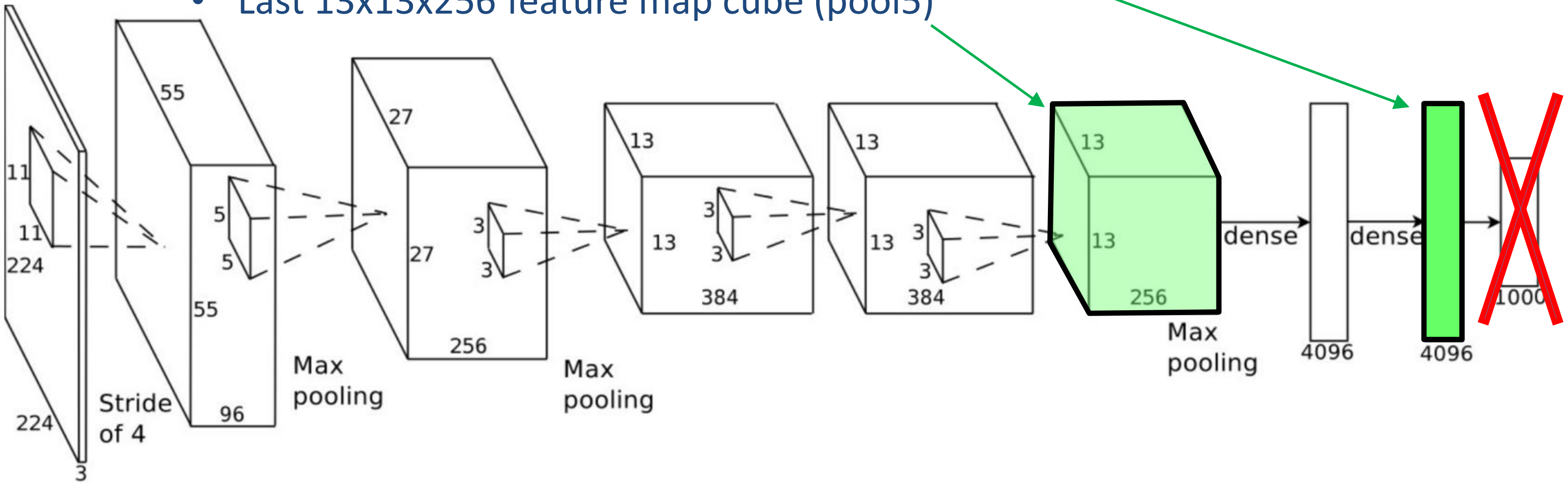
Computing the features of the regions

- Cut the regions one after the other
- Resize (warp) the regions to the input size of the ConvNet
- Calculate the features of the individual regions

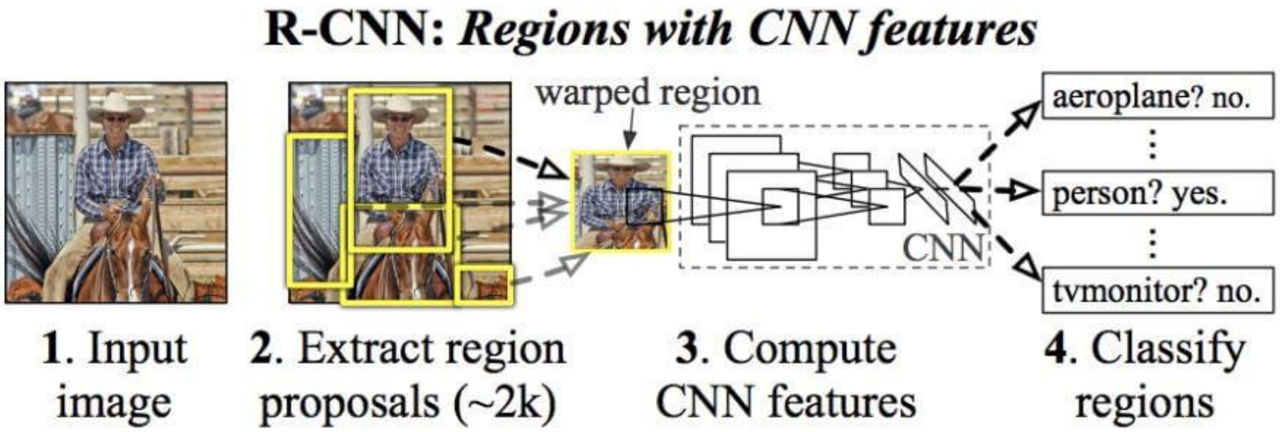


Convolution network

- Pre-trained AlexNet, later VGGNet
- The decision maker SoftMax layer was cut
  - Outputs:
    - 4096 long feature vectors from each region
    - Last 13x13x256 feature map cube (pool5)



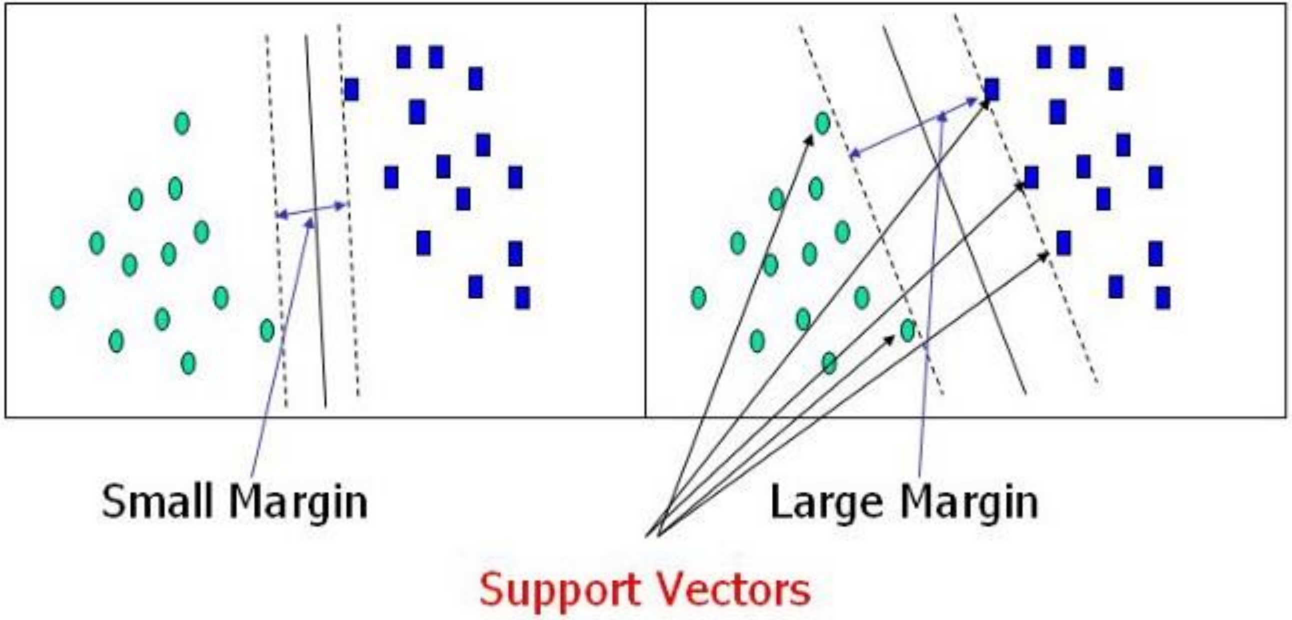
The R-CNN algorithm



1. Input image
2. Region proposals
3. Compute CNN features with warped images
4. Classification with Support Vector Machine (SVM)
5. Ranking/selecting/merging → detections
6. Bounding box regression

Linear Support Vector Machine

- **Idea:** Separate the data point in the data space with a boundary surface (hyperplane) with maximum margin
- Vectors pointing to the data points touching the margins are the support vectors
- The parameters of the optimal hyperplane is calculated with regression



- Similar to single layer perceptron, but optimized for maximum margin

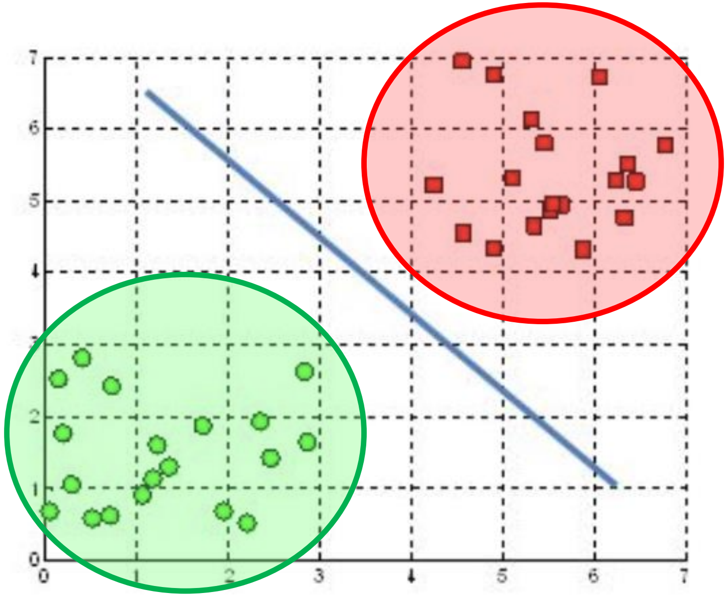
https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Why SVM?

- Why not use simple the classification output of the AlexNet?
- During the training, the AlexNet/VGGNet is not trained
- Only SVM is trained
- The number of category is much smaller
  - Designed for 20-200 categories rather than 1000

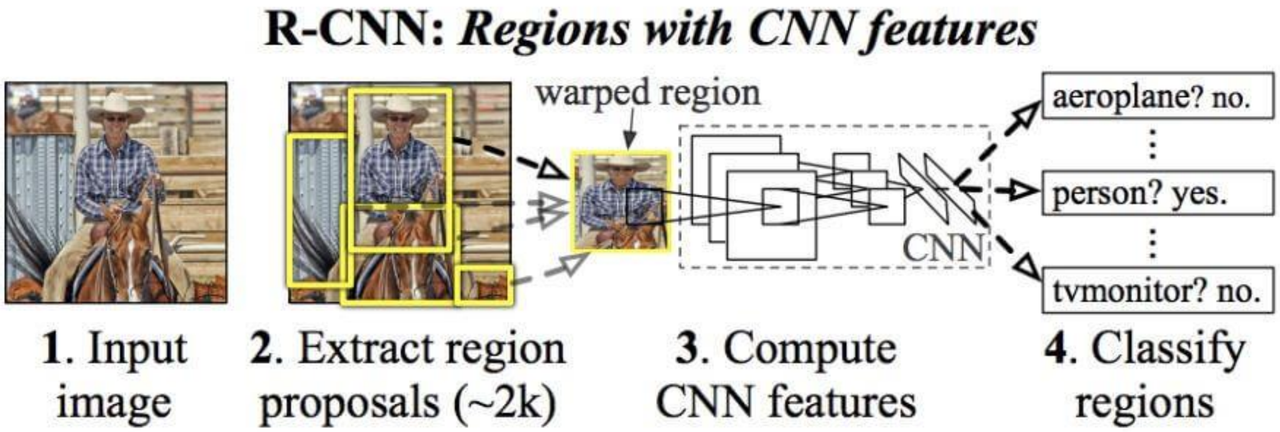
Decision with SVM

- As many separate SVM as many category we have



The result: Each region is categorized in every image classes.

The R-CNN algorithm

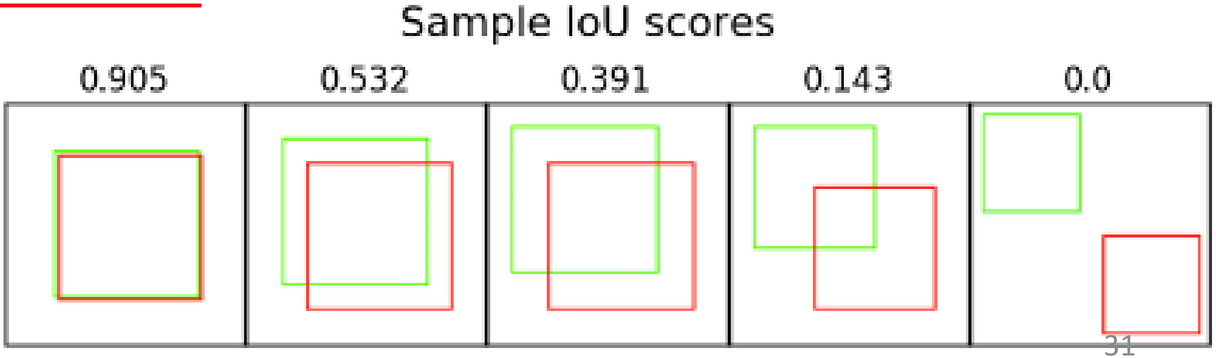


1. Input image
2. Region proposals
3. Compute CNN features with warped images
4. Classification with Support Vector Machine (SVM)
5. Ranking/selecting/merging → detections
6. Bounding box regression

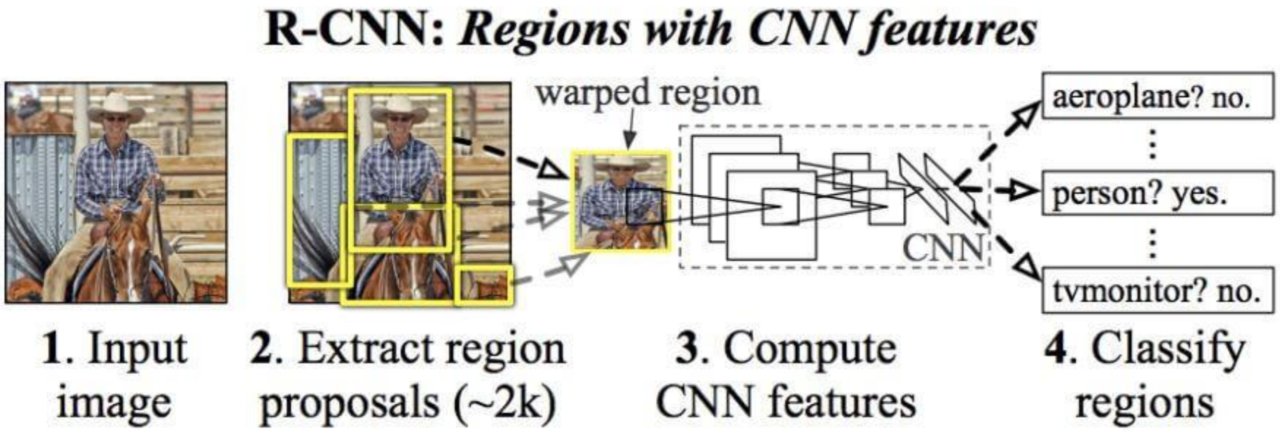
Ranking, selecting, merging

- Greedy non-maximum suppression
  - Regions with low classification probabilities are rejected
  - Regions with high Intersection over Union values (within the same category) are merged

- *Result:* **localized and classified object**

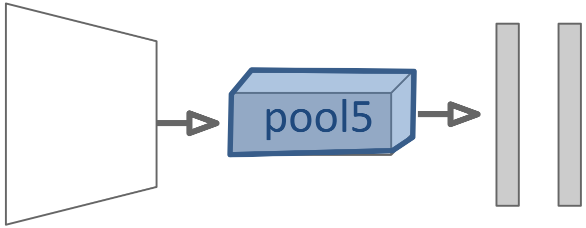


The R-CNN algorithm



1. Input image
2. Region proposals
3. Compute CNN features with warped images
4. Classification with Support Vector Machine (SVM)
5. Ranking/selecting/merging → detections
6. Bounding box regression

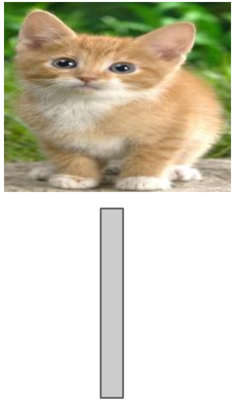
Bounding Box Regression

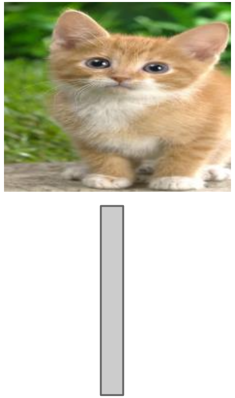


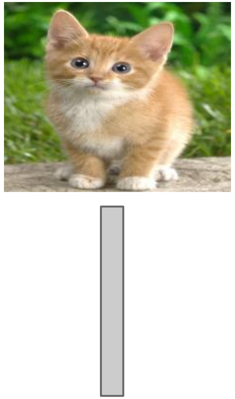
- Linear regression model
- One per object category
- Input: last feature map cube of the conv net (pool5)
- Output: size and position modification to the bounding box:
  - dx, dy, dw, dh

Training image regions

Input:
Cached feature map cube (pool5)







Regression targets:
(dx, dy, dw, dh)
(normalized)

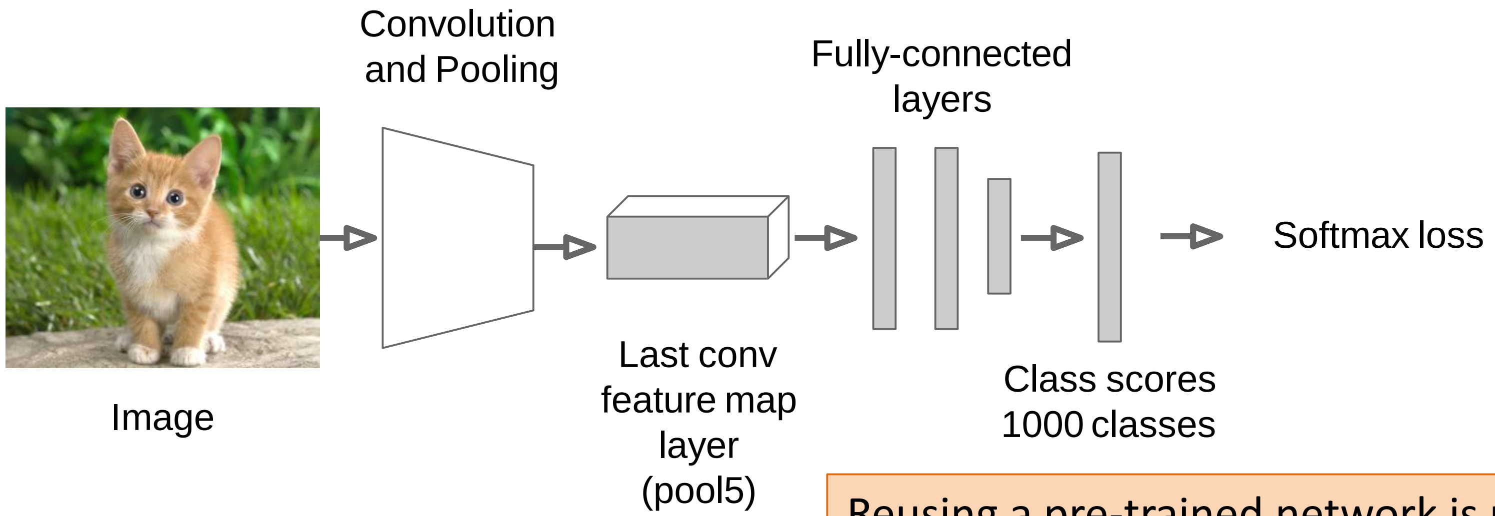
(0, 0, 0, 0)
Proposal is good

(.25, 0, 0, 0)
Proposal too far to left

(0, 0, -0.125, 0)
Proposal too wide

R-CNN Training

Step 1: Take a pretrained Convolutional Neural Network (e.g. AlexNet)



Reusing a pre-trained network is useful, if there is not enough data to train or if it provides good enough result. Fine tuning typically needed!

Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

R-CNN Training

Step 2: Extract features

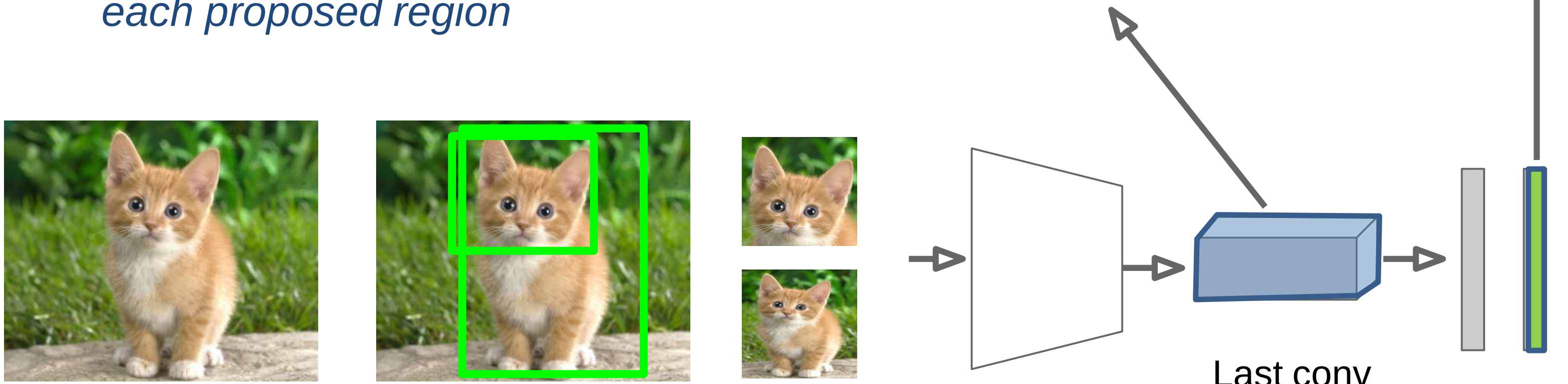
- *Go through the data base*
- *Use region proposal*
- *Calculate the features for each proposed region*

Save the feature cube to disk!

This feature cube describes the relative position information, and will be used for bounding box regression. (Sometimes this is used for classification as well.)

Save the feature vector to disk!

This feature vector describes the content, and will be used for classification



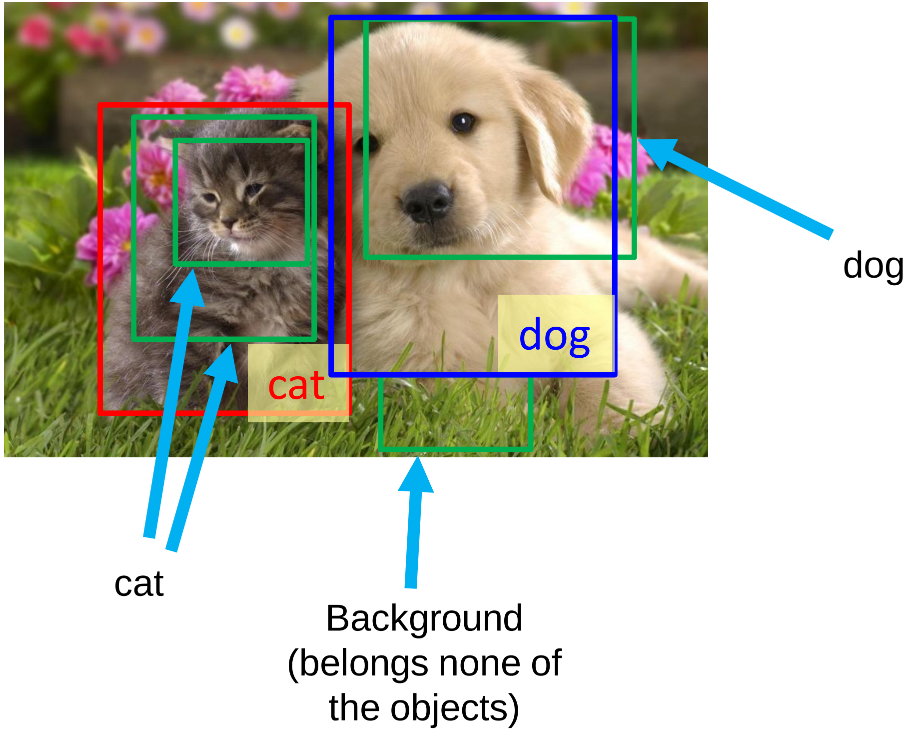
Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

R-CNN Training

Step 3: Identify which proposed region belongs to which object class

Based on the annotated image

Proposed region overlaps with the annotated image segment? (IoU)

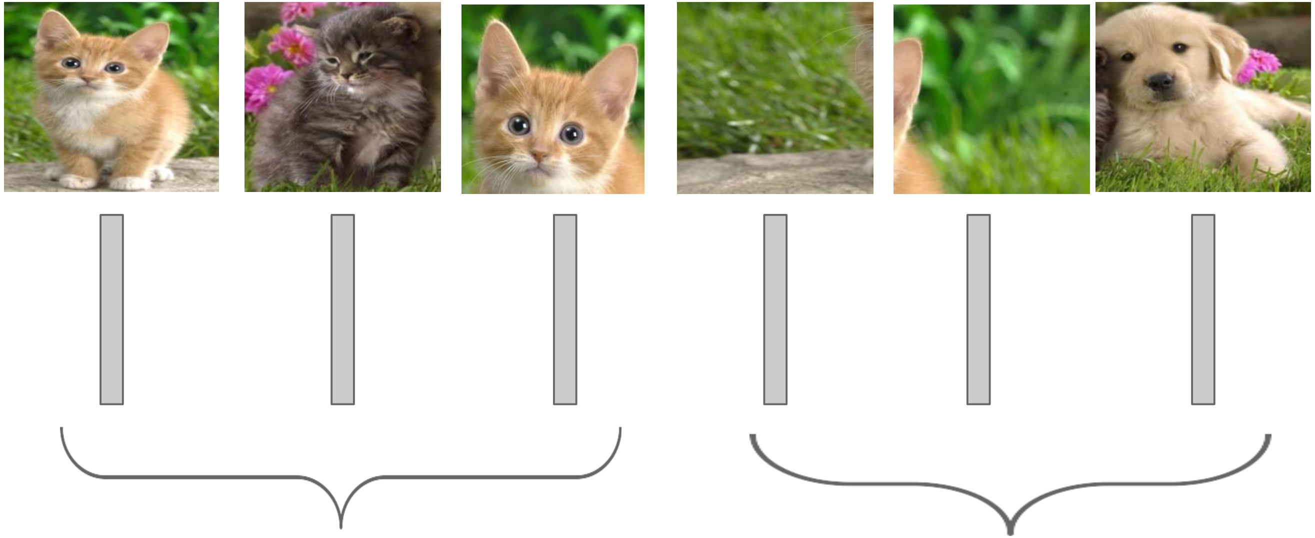


R-CNN Training

Step 4: Train one SVM per class to classify region features

Training image regions

Cached region features vectors



Positive samples for cat SVM

Negative samples for cat SVM

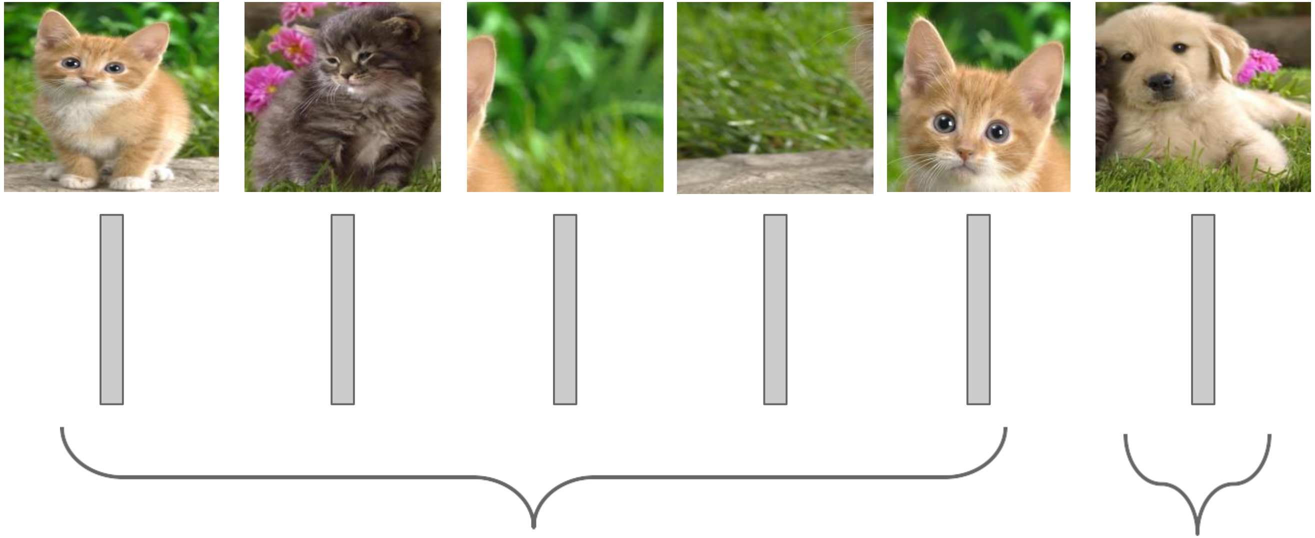
Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

R-CNN Training

Step 4: Train one SVM per class to classify region features

Training image regions

Cached region features vectors



Negative samples for dog SVM

Positive samples for dog SVM

Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

R-CNN Training

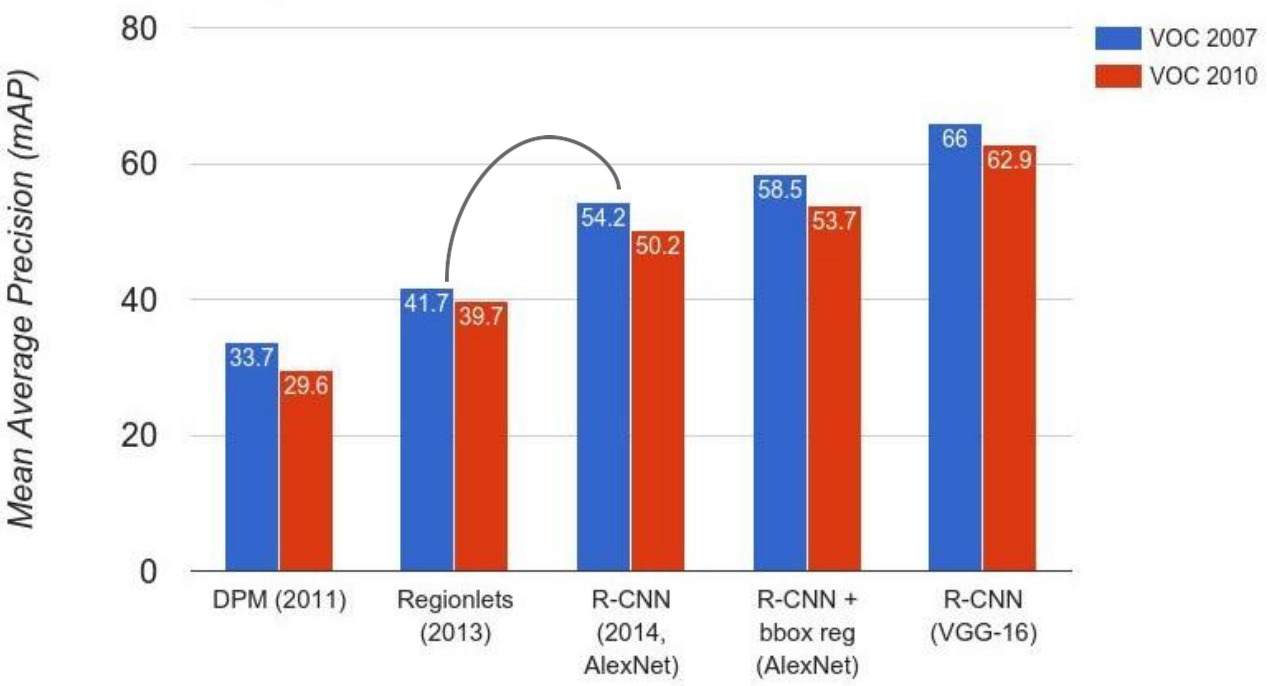
Step 5 (bbox regression): For each class, train a linear regression model to map from cached features cubes to offsets/size of the boxes to fix “slightly wrong” position proposals

Training image regions			
Cached region feature cube (pool5)			
Regression targets (dx, dy, dw, dh) Normalized coordinates	(0, 0, 0, 0) Proposal is good	(.25, 0, 0, 0) Proposal too far to left	(0, 0, -0.125, 0) Proposal too wide

Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

R-CNN Results

Big improvement (~25%)
compared to pre-CNN methods

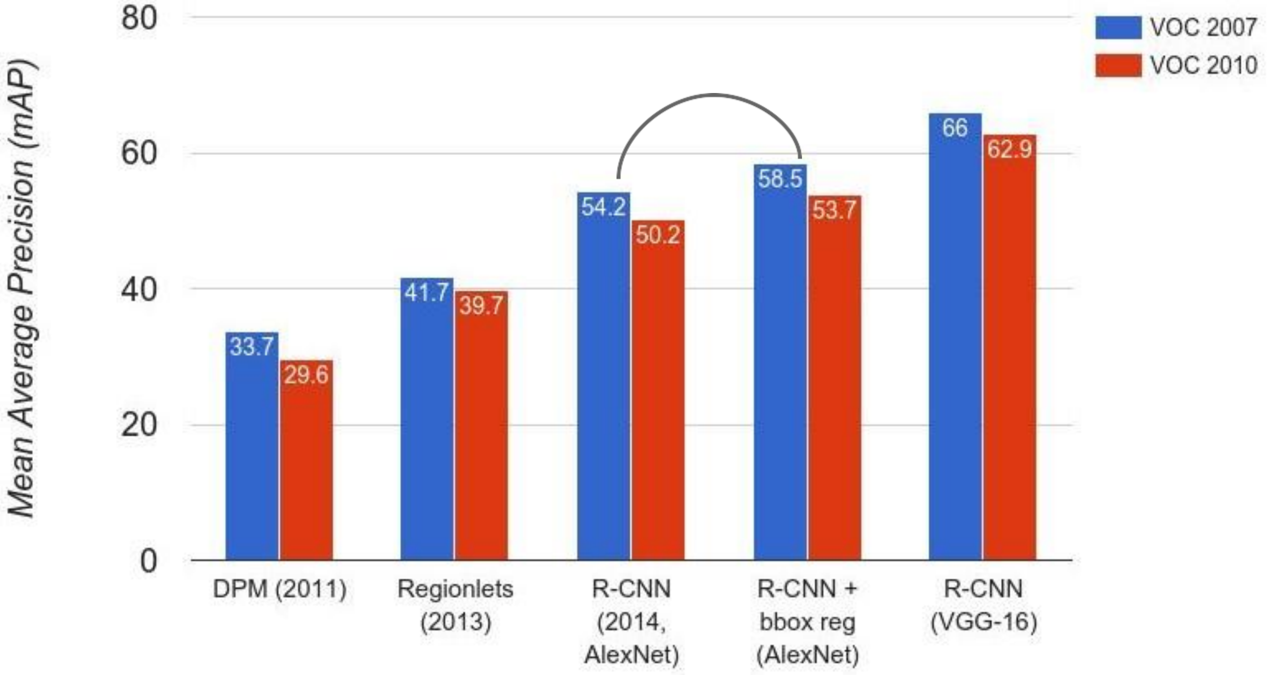


Wang et al, “Regionlets for Generic Object Detection”, ICCV 2013

Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

R-CNN Results

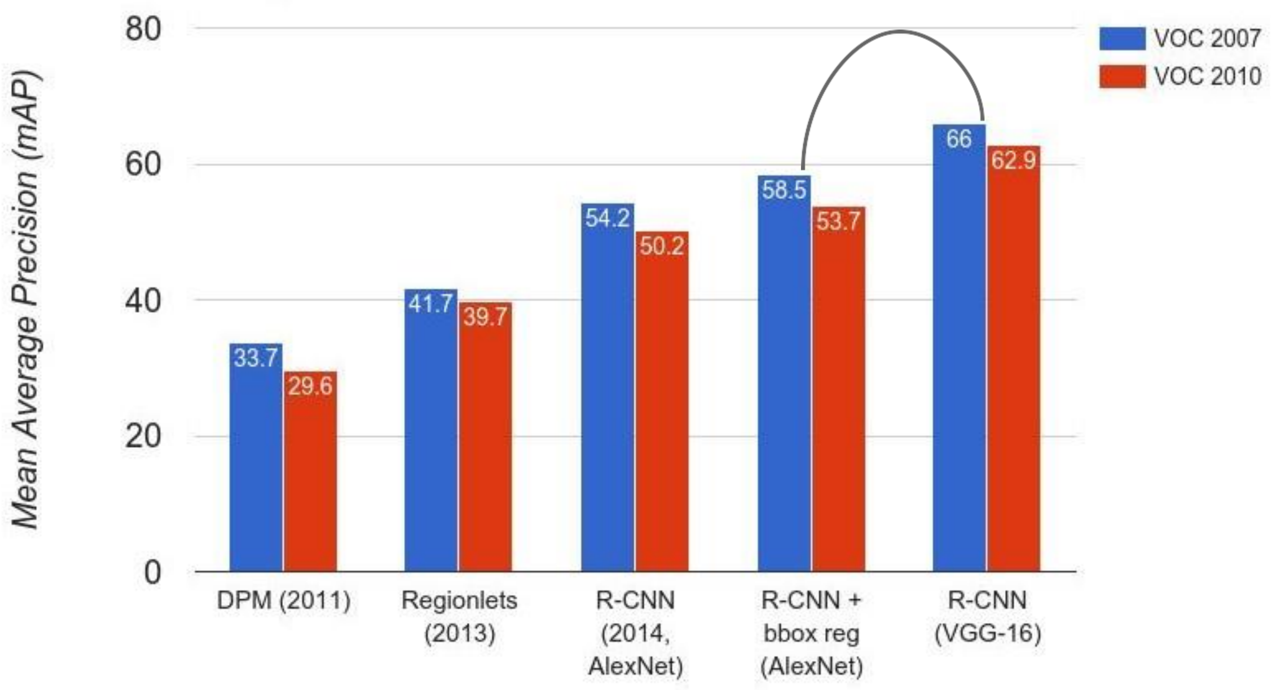
Bounding box regression helps a bit



Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

R-CNN Results

Features from a deeper network help a lot



Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li

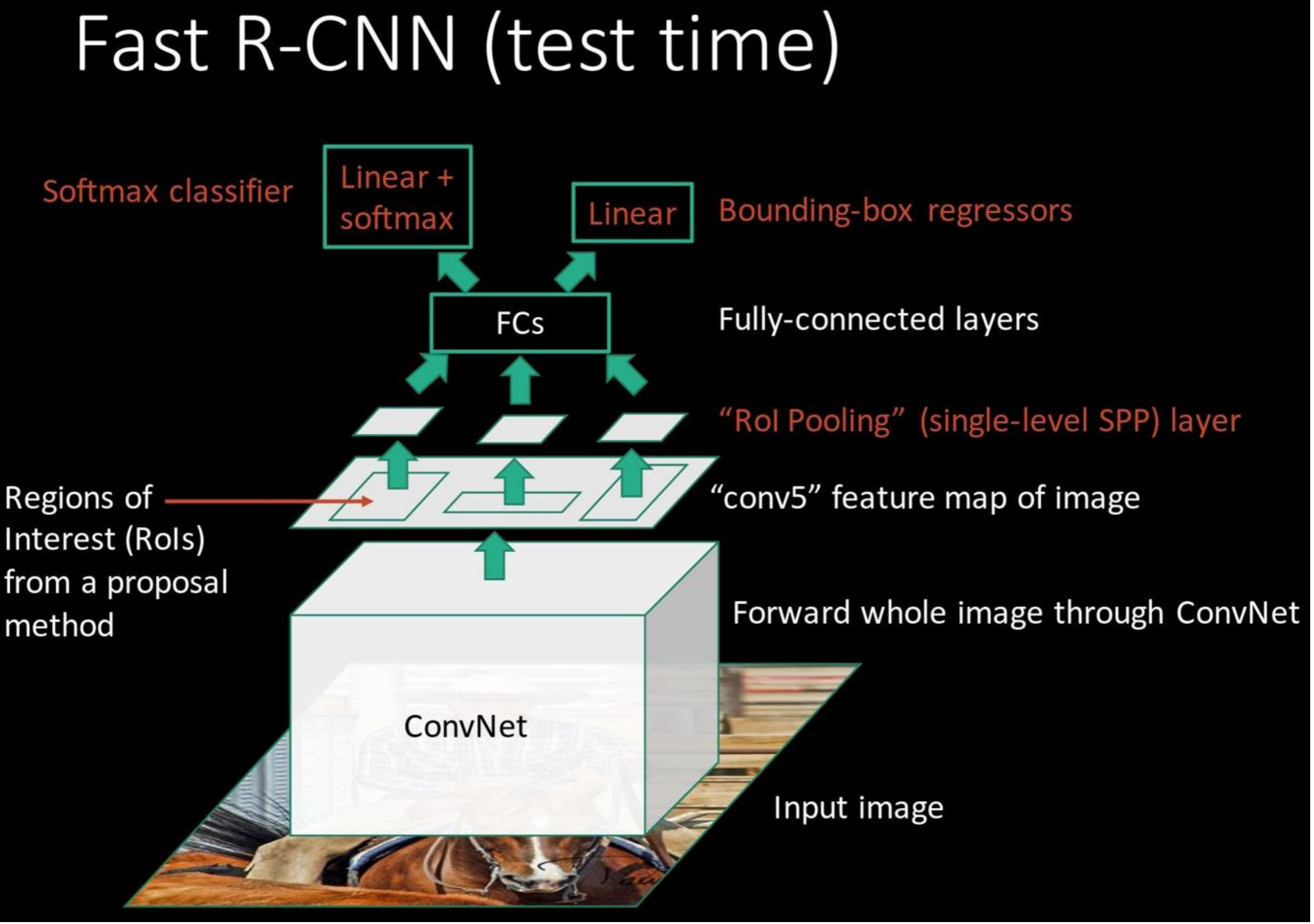
R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal
   - Recalculate the features again-and-again in the overlapping regions

2. SVMs and bbox regressors are post-hoc:
   - CNN features not updated in response to SVMs and regressors

3. Complex multistage training pipeline
   - Calculate the features for all the regions for all the training image first
   - Then train for SVM and bbox regressor separately

Slide Credits: Justin Johnson, Andrej Karpathy, Fei-Fei Li



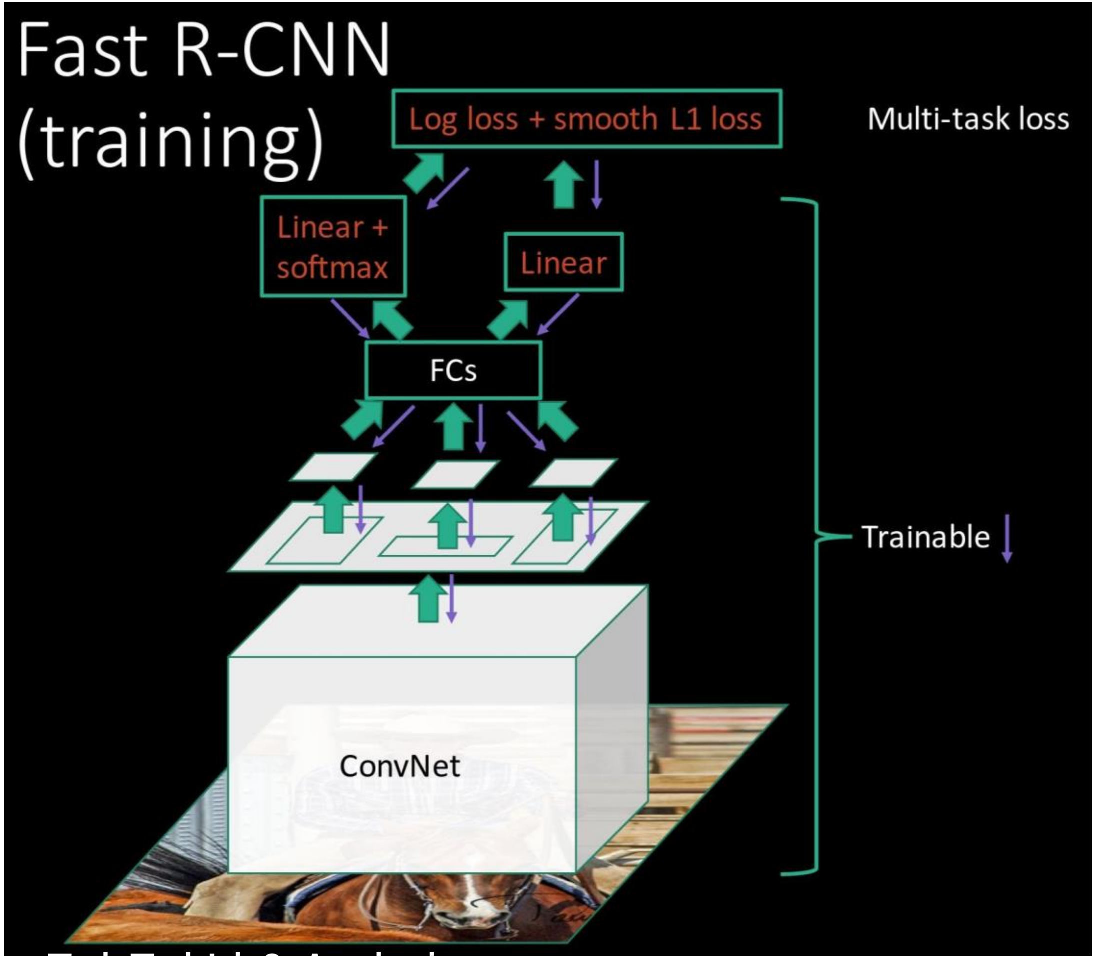
R-CNN Problem #1: Slow at test-time due to independent forward passes of the CNN

Solution: Share computation of convolutional layers between proposals for an image

Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46



R-CNN Problem #2:
Post-hoc training: CNN not updated in response to final classifiers and regressors

R-CNN Problem #3:
Complex training pipeline

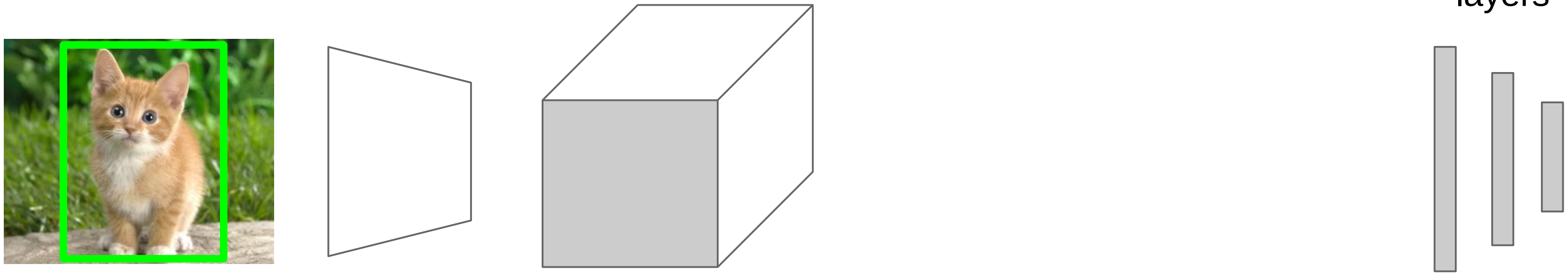
Solution:
Just train the whole system end-to-end all at once!

Slide credit: Ross Girschick

Fast R-CNN: Region of Interest Pooling

Convolution and Pooling

Fully-connected layers

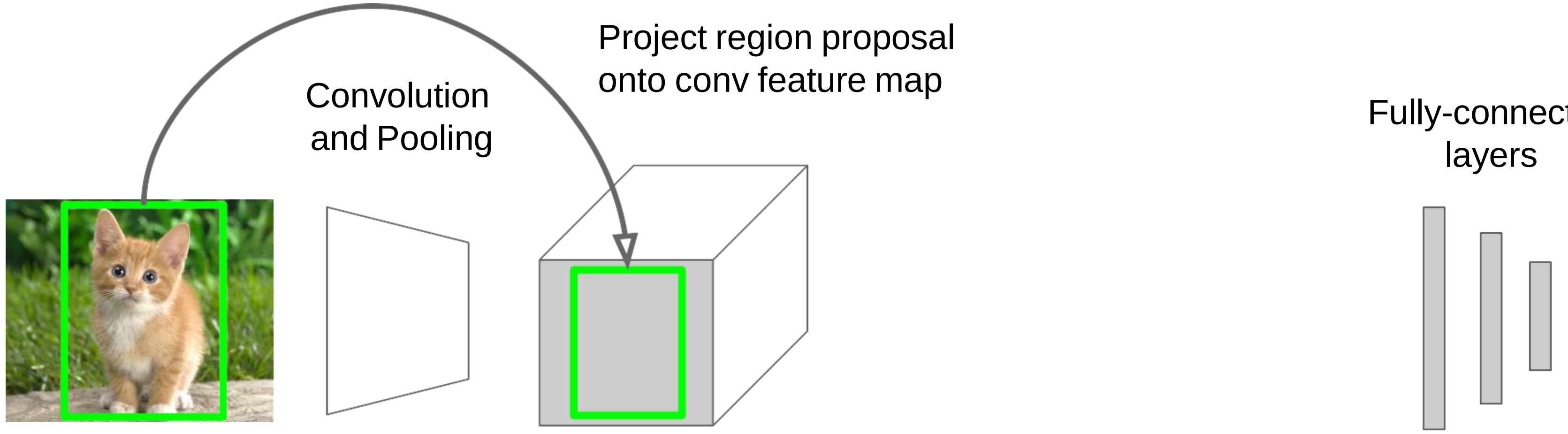


Hi-res input image:
3 x 800 x 600
with region
proposal

Hi-res conv features:
C x H x W
with region proposal

Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling

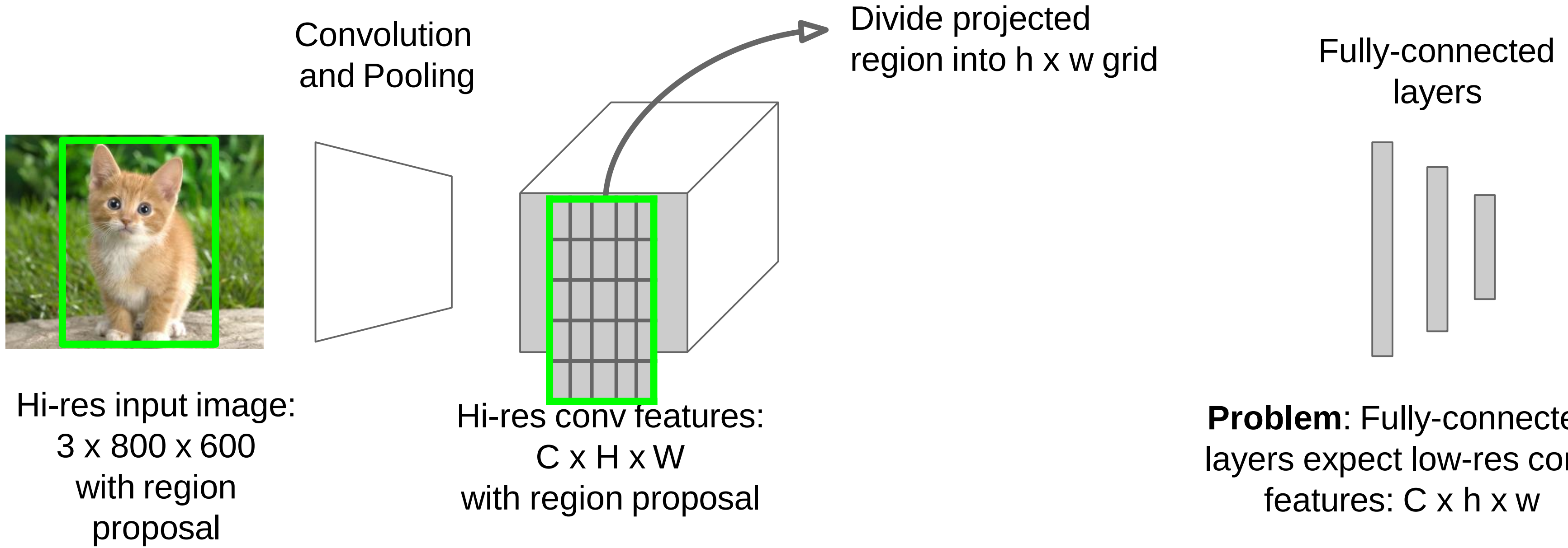


Hi-res input image:
3 x 800 x 600
with region
proposal

Hi-res conv features:
C x H x W
with region proposal

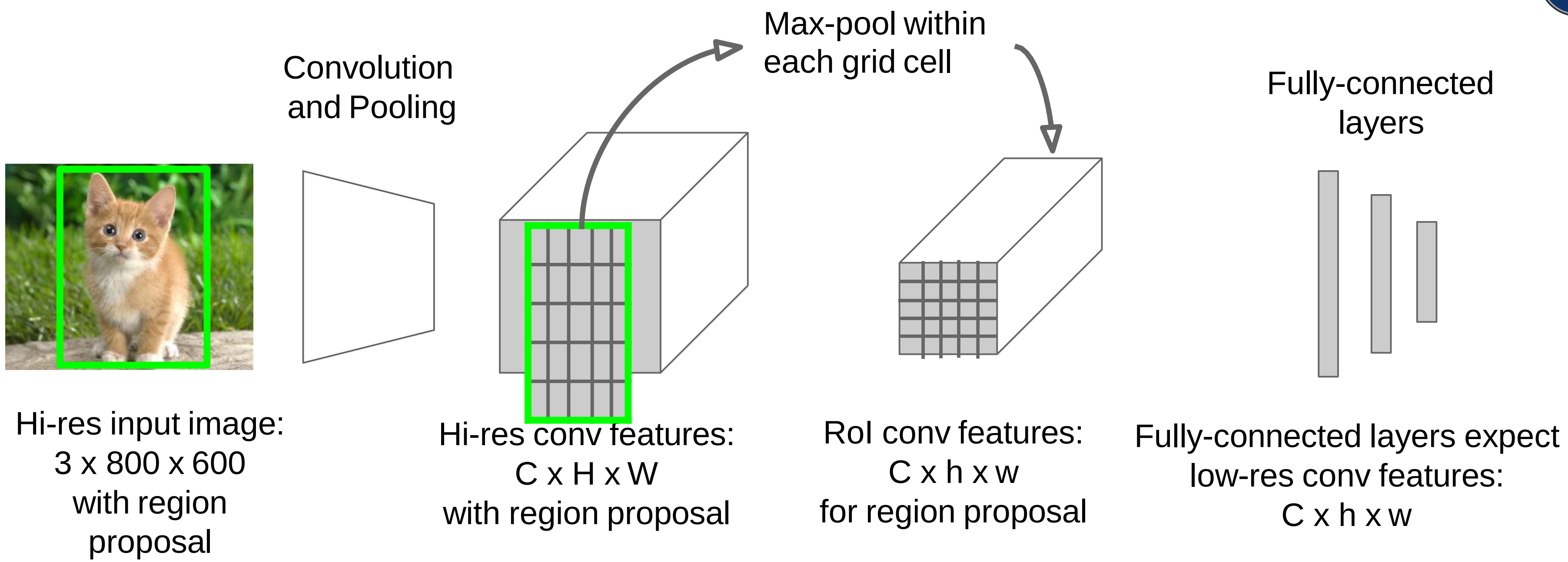
Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling

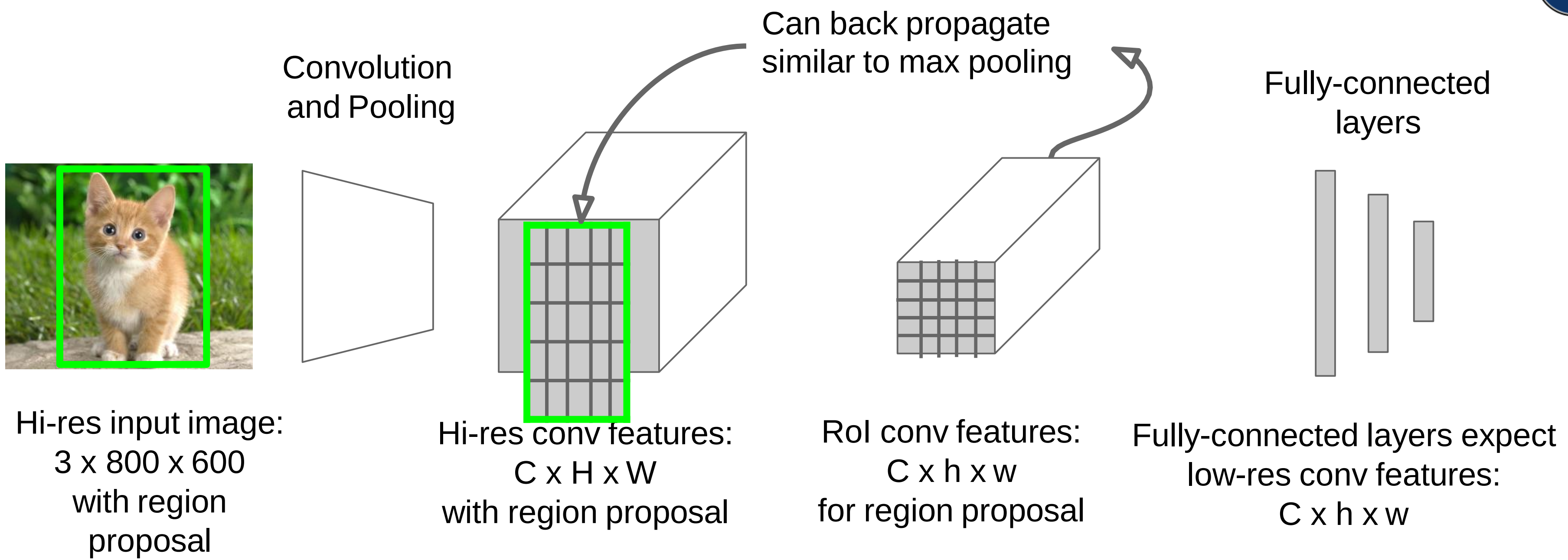


Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling



Fast R-CNN: Region of Interest Pooling



Instead of SVM, a SoftMax layer makes the decision at Fast R-CNN.

Fast R-CNN Results

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	**9.5 hours**
	(Speedup)	1x	**8.8x**
FASTER!	Test time per image	47 seconds	**0.32 seconds**
	(Speedup)	1x	**146x**
Better!	mAP (VOC 2007)	66.0	**66.9**

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Results

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	**9.5 hours**
	(Speedup)	1x	**8.8x**
FASTER!	Test time per image	47 seconds	**0.32 seconds**
	(Speedup)	1x	**146x**
Better!	mAP (VOC 2007)	66.0	**66.9**

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Problem:

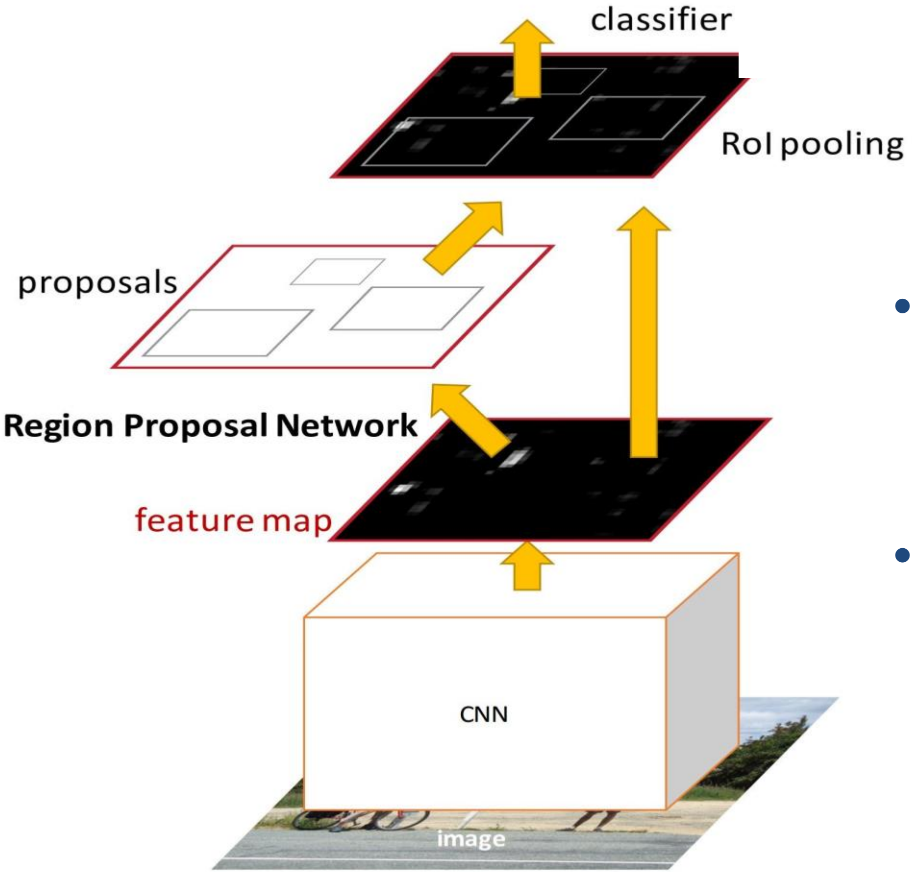
	R-CNN	**Fast R-CNN**
Test time per image without **Region Proposals**	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with **Region Proposals**	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

Fast R-CNN Problem Solution:

Test-time speeds don't include region proposals
Just make the CNN do region proposals too!

	R-CNN	Fast R-CNN
Test time per image without **Region Proposals**	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with **Region Proposals**	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

Faster R-CNN:



- Insert a **Region Proposal Network (RPN)** after the last convolutional layer
  - Reuse the CNN computation

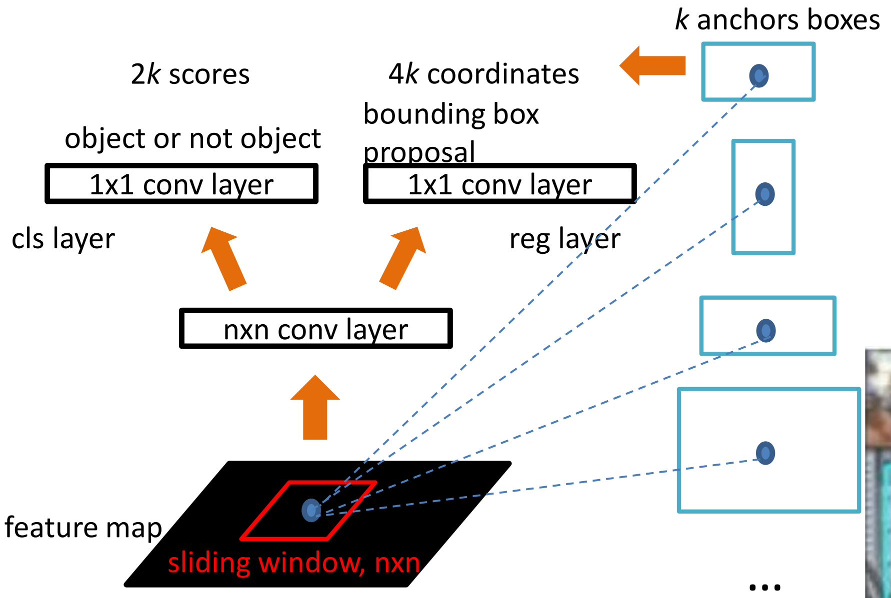
- RPN trained to produce region proposals directly; no need for external region proposals!

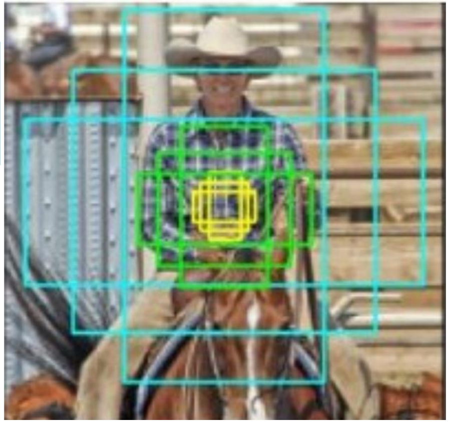
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN

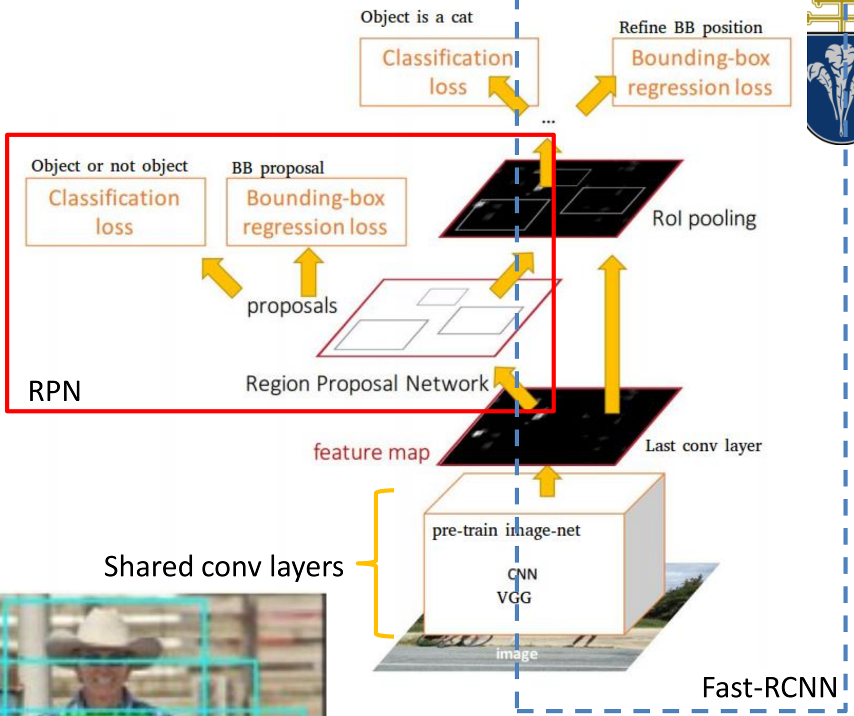
https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4

Faster-RCNN

Region Proposal Networks:







Anchors:
three rectangle
in three scales.

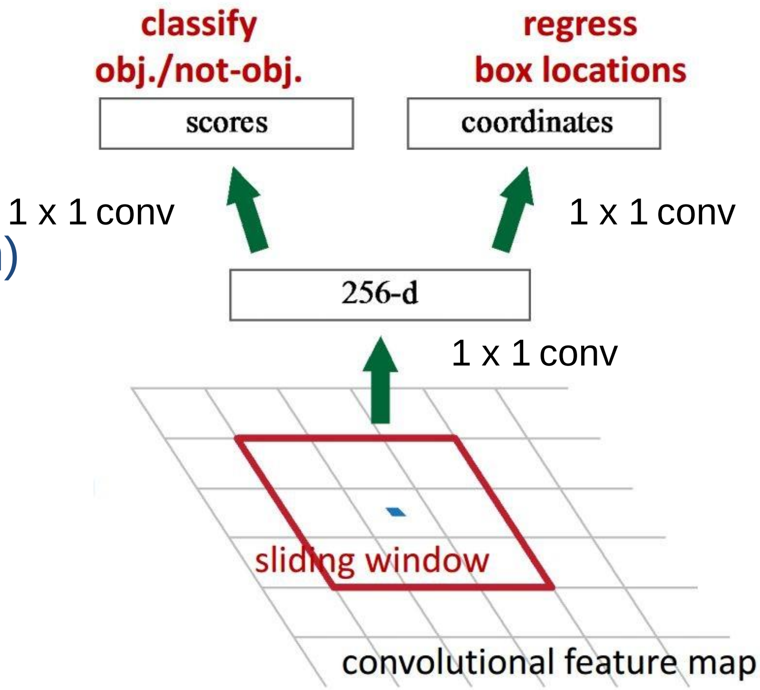
Faster R-CNN: Region Proposal Network

Slide a small window on the feature map
(very small computational effort per position)

Build a small network for:
- classifying object or not-object, (Binary decision)
- regressing bbox locations

Position of the sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



Slide credit: Kaiming He

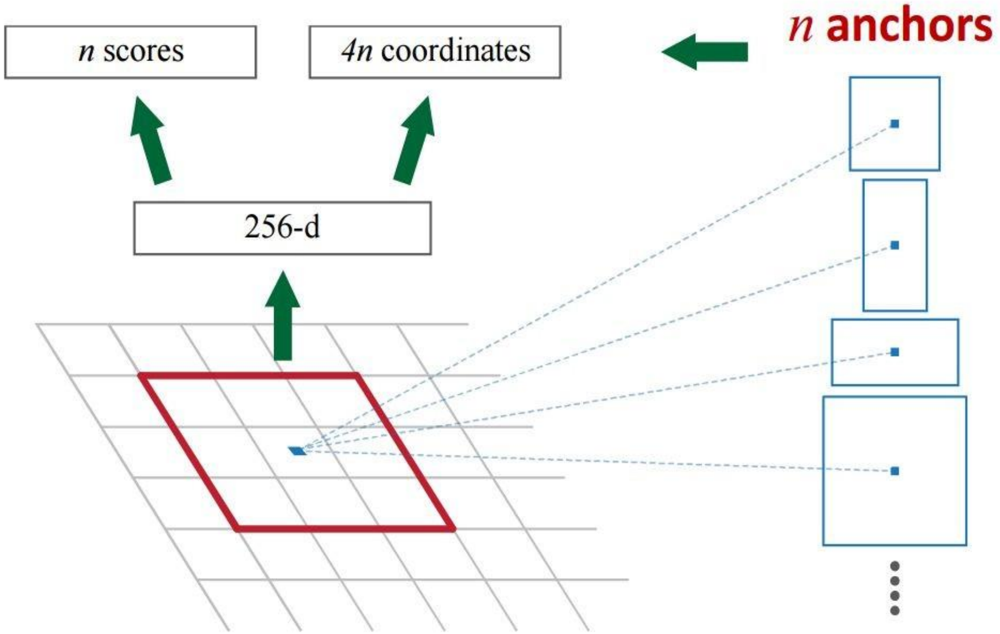
Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

Regression gives offsets from anchor boxes

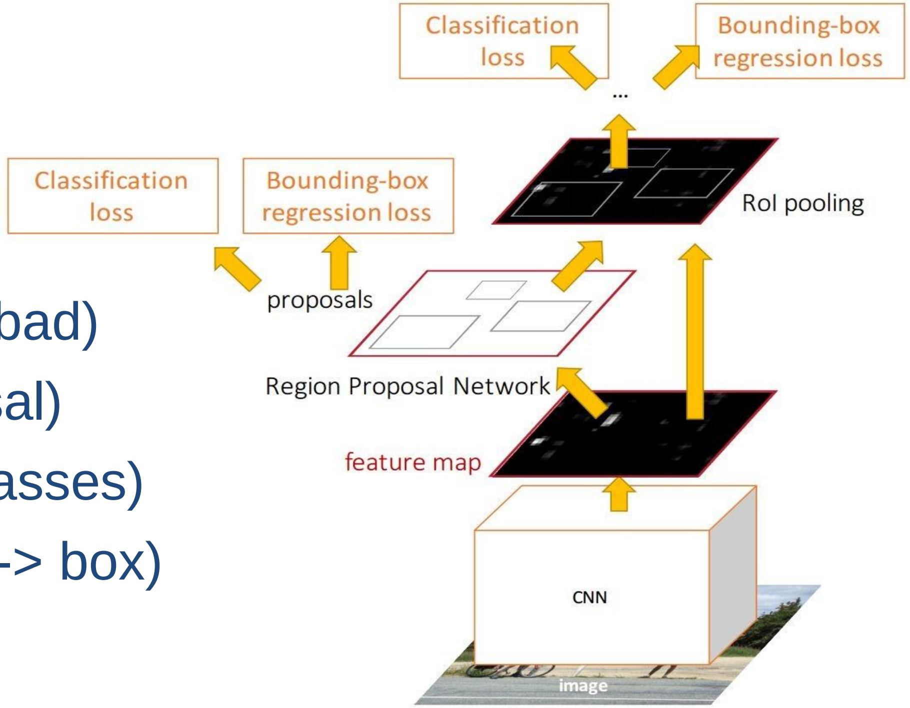
Classification gives the probability that each (regressed) anchor shows an object



Faster R-CNN: Training

One network, four losses

- RPN classification (anchor good / bad)
- RPN regression (anchor -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)



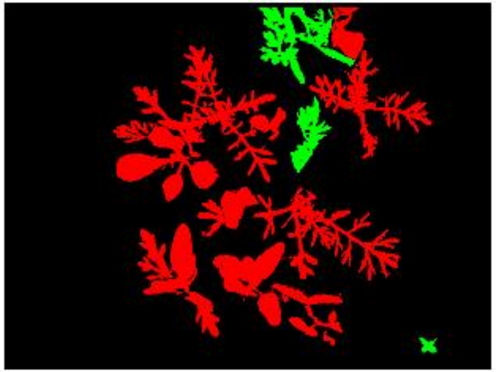
Slide credit: Ross Girschick

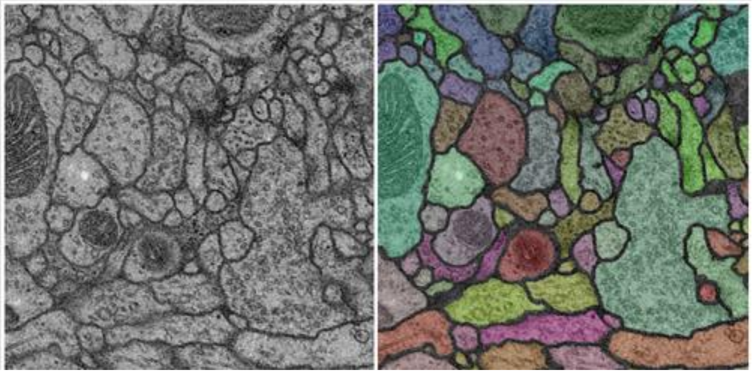
Faster R-CNN: Results

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Segmentation

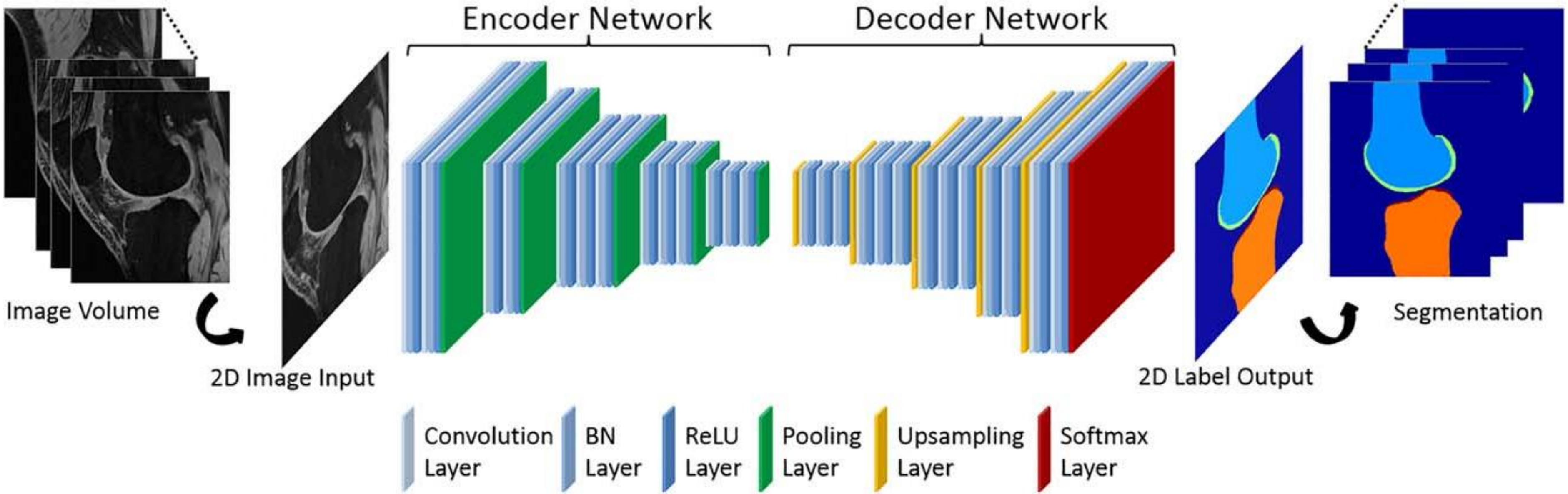
- Pixel-wise classification
  - Scene understanding
    - Autonomous driving
  - Medical imaging
  - Precision agriculture





Segmentation Architecture in General

- Same resolution is needed at the end

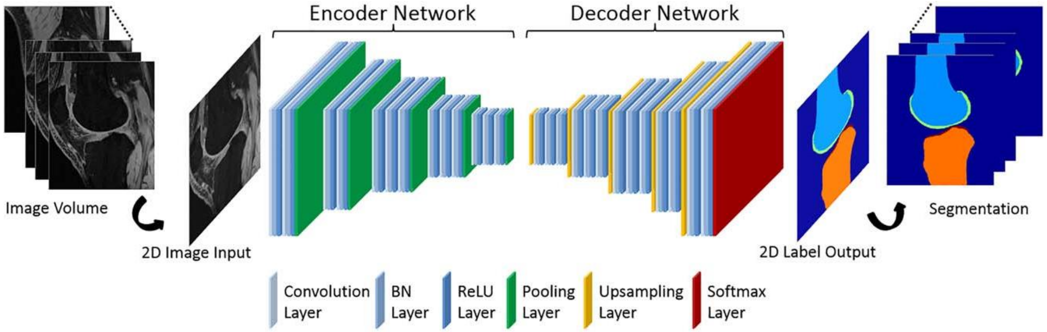


Contraction side

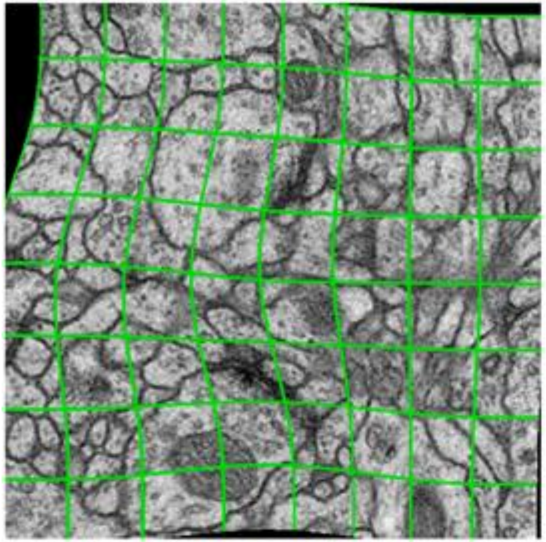
Expansion side

Segmentation Architecture in General

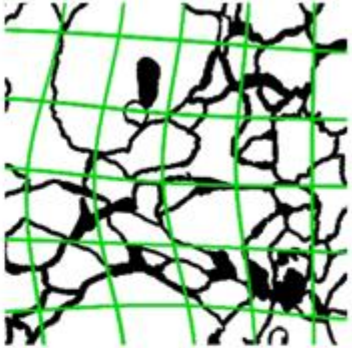
- Encoder Network: extract image features using deep convolutional network
  - Each layer: bank of trainable convolutional filters, followed by
  - ReLUs and max-pooling to downsample image features
- Decoder Network: upsamples feature map back to image resolution with final output having same number of channels as there are pixel classes
  - Deconvolution
  - Network mirrors encoder network
- Pixel-wise softmax over final feature map and cross-entropy loss function for training using some kind of SGD.



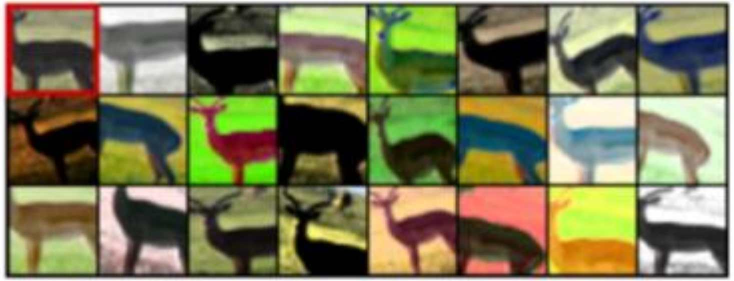
U-Net



resulting deformed image
(for visualization: no rotation, no shift, no extrapolation)



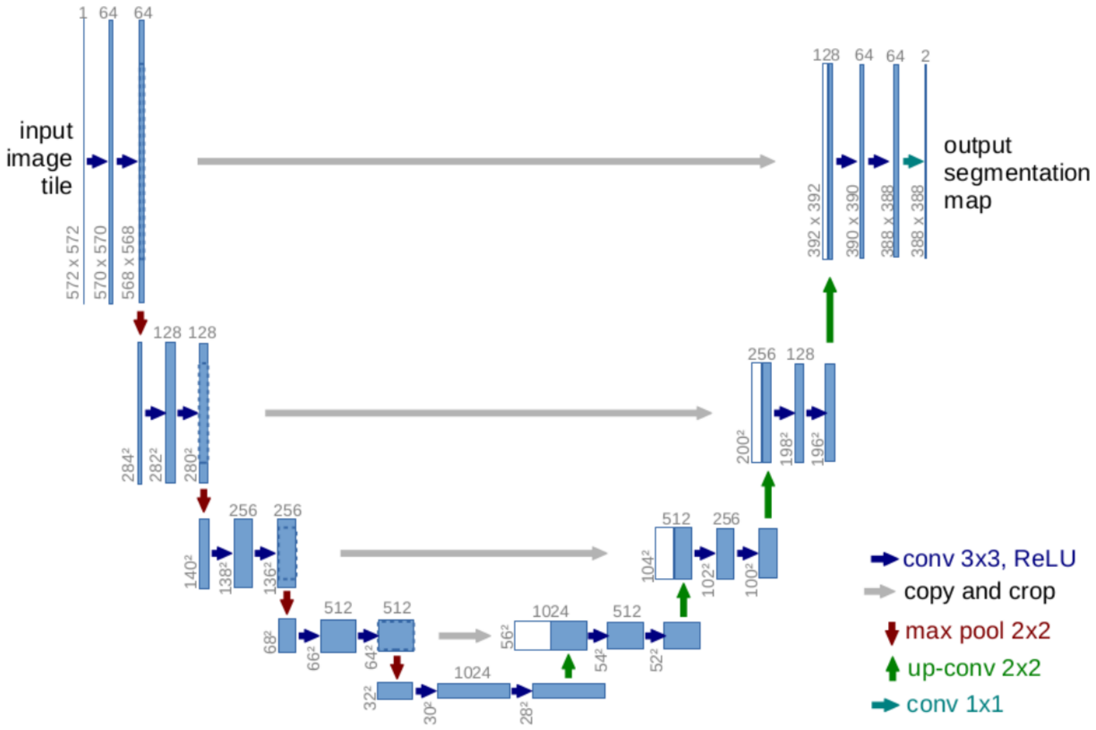
correspondingly deformed
manual labels



From [3]

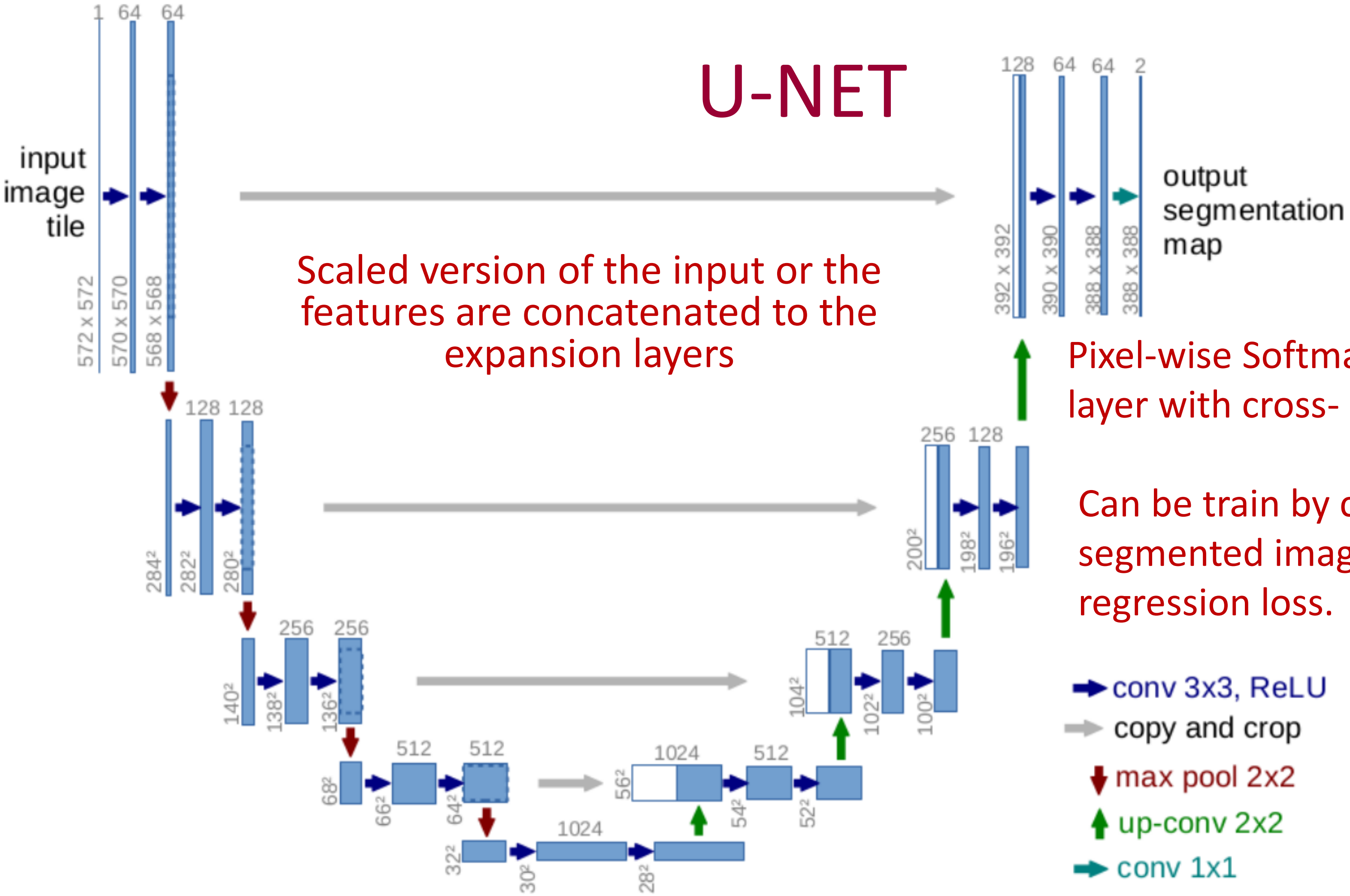
- Designed for biomedical image processing: cell segmentation
- Data augmentation via applying elastic deformations,
  - Natural since deformation is a common variation of tissue
  - Smaller dataset is enough

U-Net



- Concatenate features from encoder network
  - instead of reusing pooling indices
- Relatively shallow network with low computational demand
  - 3x3 convolution kernel size only
  - 2x2 max pooling
- No fully connected layer in the middle

U-NET



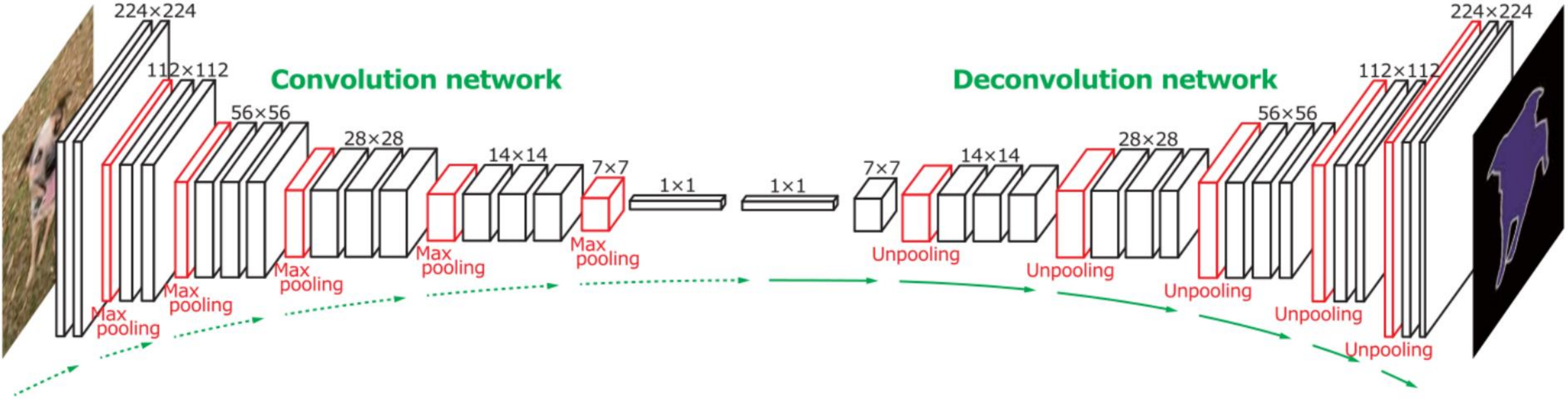
Scaled version of the input or the features are concatenated to the expansion layers

Pixel-wise Softmax at the last layer with cross- entropy loss.

Can be train by colored segmented image with regression loss.

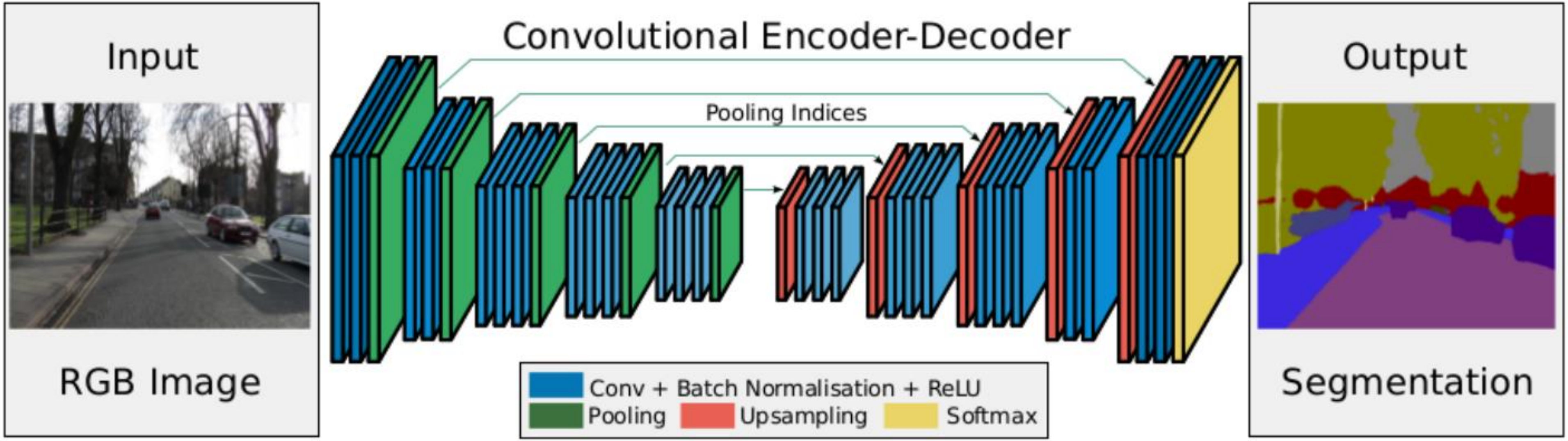
DeconvNet

- Instance-wise segmentation
- Two-stage training:
  - train on easy examples (cropped bounding boxes centered on a single object) first and
  - then more difficult examples



SegNet

- 13 convolutional layers from VGG-16
  - The original fully connected layers are discarded
- Max pooling indices (locations) are stored and sent to decoder
- Scene understanding



 https://towardsdatascience.com/review-segnet-semantic-segmentation-e66f2e30fb96

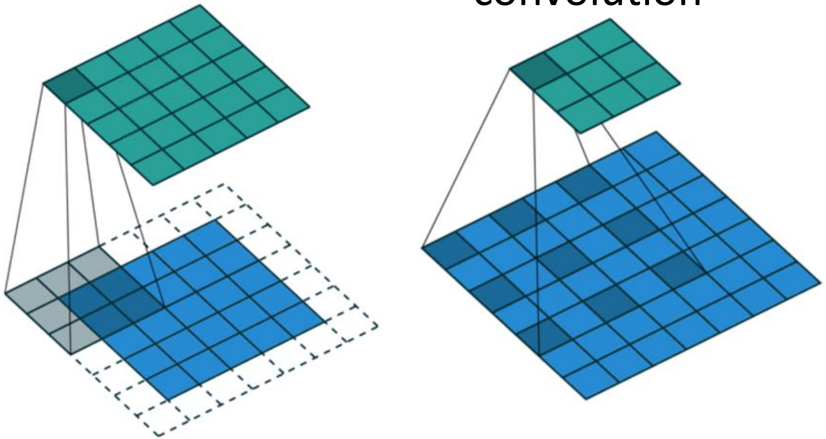
Avoiding resolution loss but no high computational load:

Atrous convolution

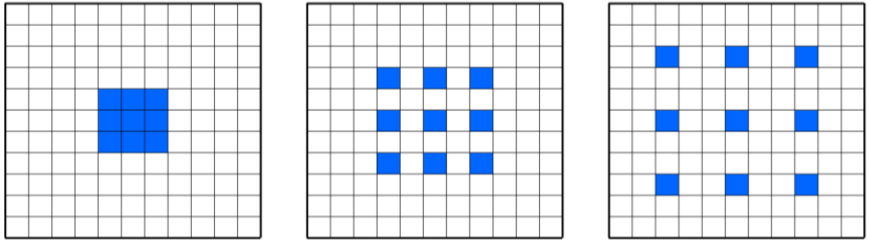
- How it works?
  - Blows up the kernel
  - Filling up the holes with zeros
    - Atrous means very dark (like the wholes between the values)
- Properties
  - Not doing downsampling
  - Not increasing computational load
  - But reaches larger neighborhood
  - Combines information from larger neighborhood

Normal convolution

Atrous (dilated) convolution

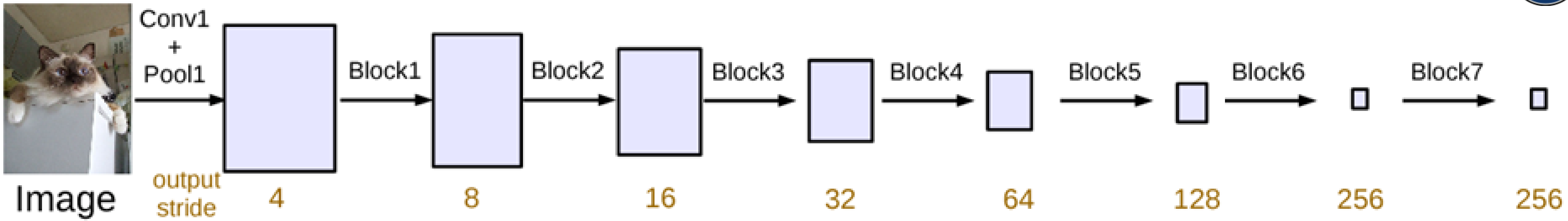


kernel

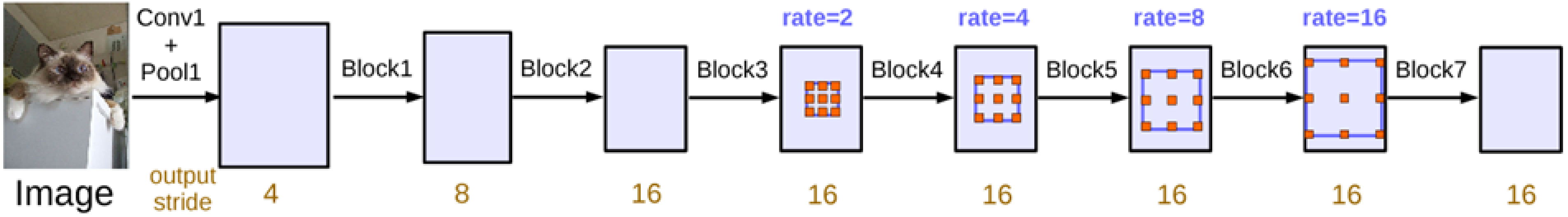


rate=1 rate=2 rate=3

Depth-to-Space



Normal convolution goes deeper with reducing resolution



Atrous convolution goes deeper without further reducing resolution

Filter size considerations

- **Small** field-of-view → accurate **localization**
- **Large** field-of-view → context **assimilation**
- Effective filter size increases (enlarge the field-of-view of filter)

$$n_o\colon\ k \times k \quad \rightarrow \quad n_a\colon \big(k + (k-1)(r-1)\big) \times \big(k + (k-1)(r-1)\big)$$

n_o : original convolution kernel size

n_a : atrous convolution kernel size

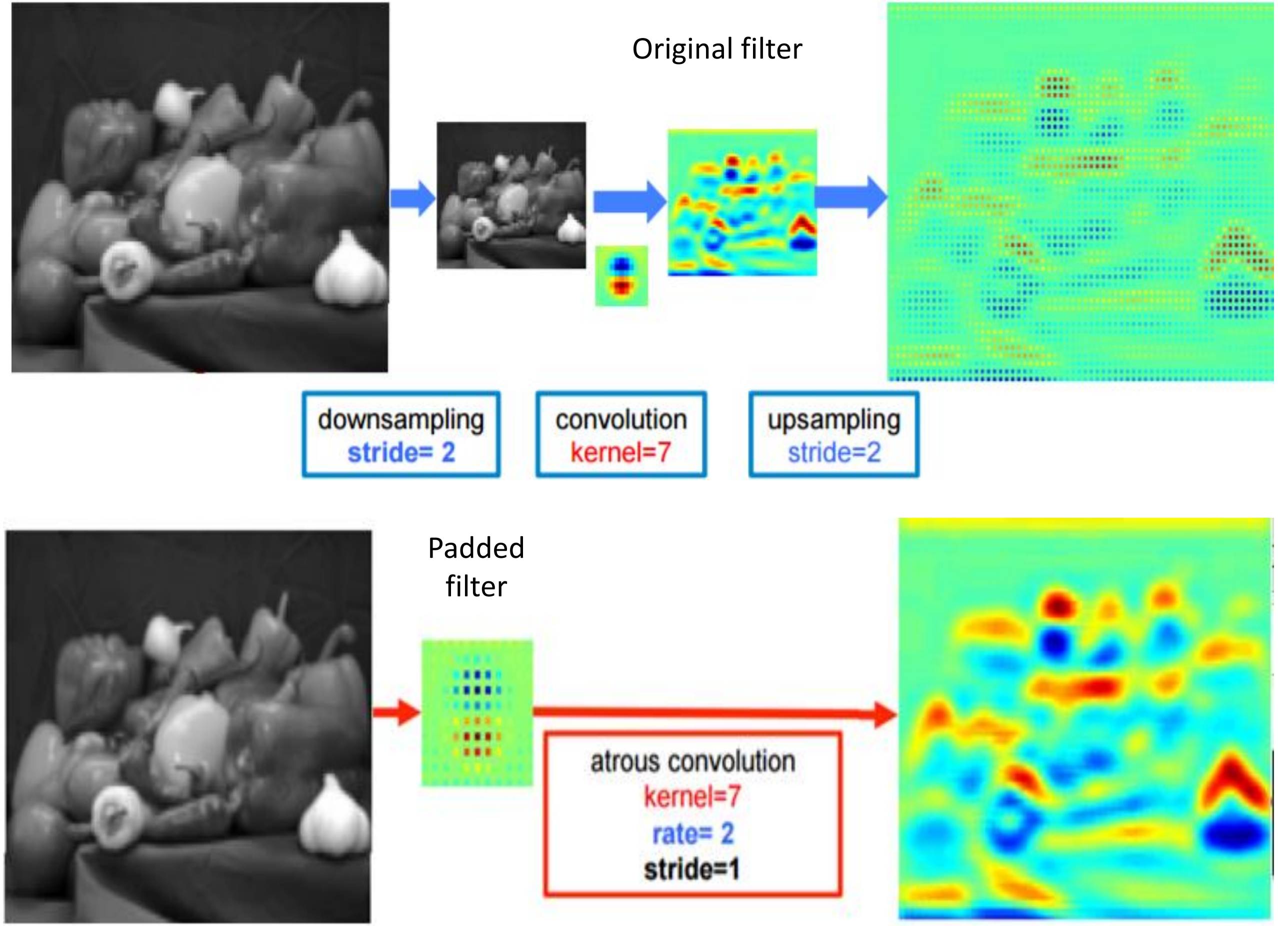
r: rate

- However, we take into account only the non-zero filter values:
  - Number of filter parameters is the same
  - Number of operations per position is the same

Visualizing atrous convolution

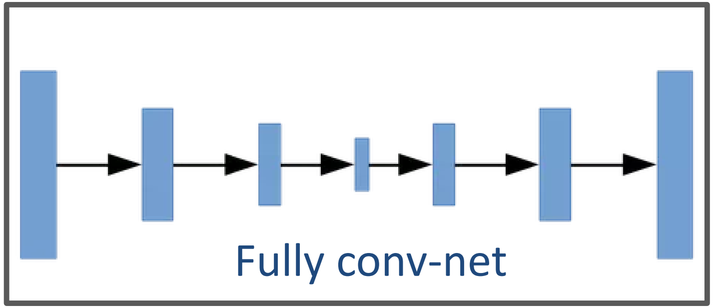
Standard convolution

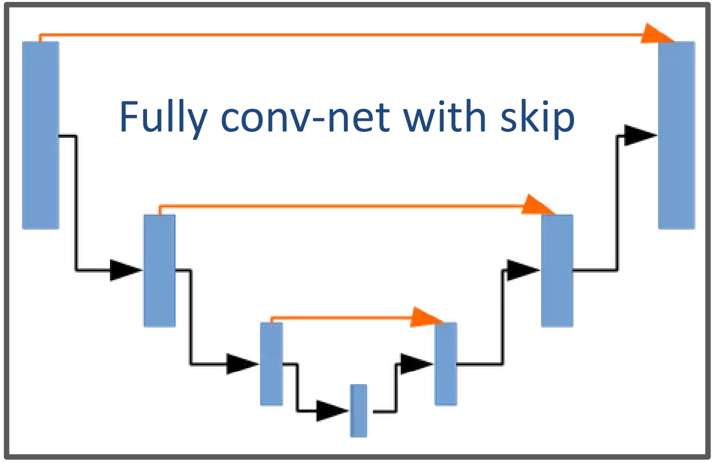
Atrous convolution

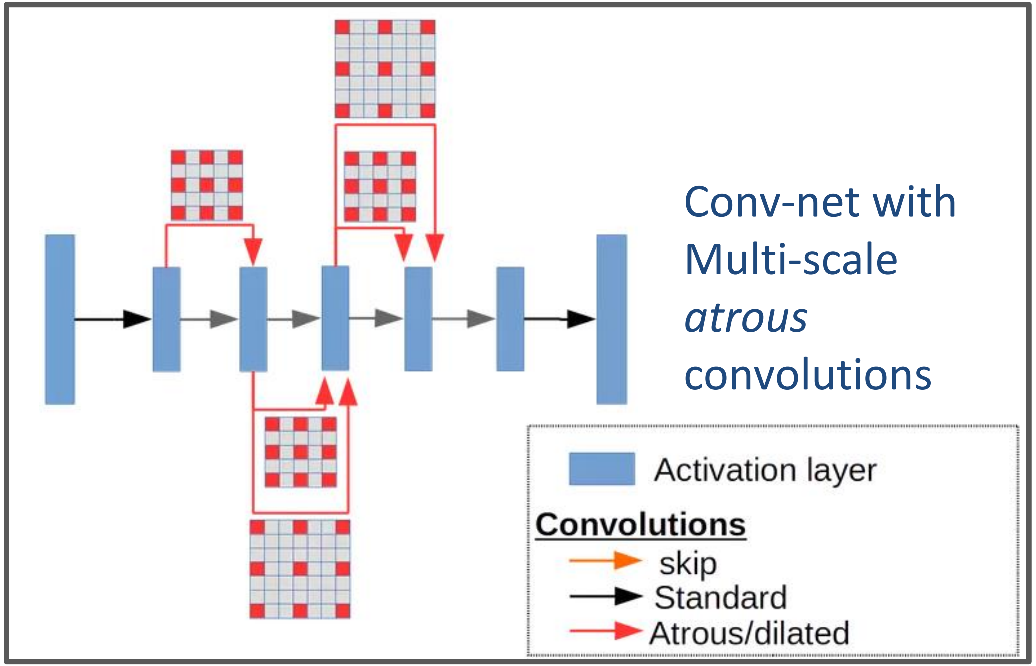


Chen, Liang-Chieh, et al. "[DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs]." *arXiv preprint arXiv:1606.00915* (2016).

Semantic segmentation CNN arrangements







- How to solve reduced resolution?
  - Do not downsample !!!
    - Convolution on large images ⇒ Small FOV Enlarge kernel
    - Size O(n^2) more parameters ⇒ getting close to fully
    - Connected layer, slow training, overfitting
  - Atrous Convolution.
    - Large FOV with little parameters → Kill two birds with one stone!