https://tinyurl.com/374wtvmy

Computer Vision 1

HC6a

Single Shot Detection, Network Architectures

Dr. Martin Oswald, Dr. Dimitris Tzionas, Dr. Arun Mukundan,
(m.r.oswald, d.tzionas, a.mukundan)@uva.nl

University of Amsterdam

computer vision lab

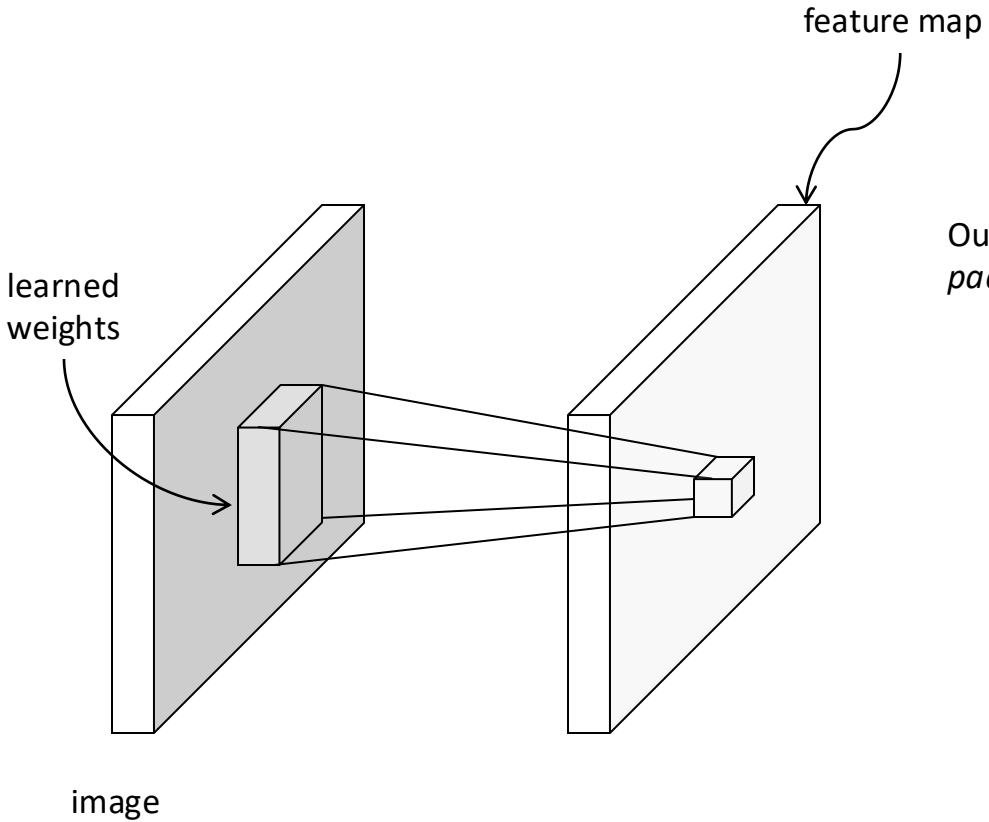
Today's Agenda

- Learning region proposals - Faster R-CNN
- Feature Pyramid Network (FPN)
- Single Shot Detectors (SSD) / Yolo
- Transfer learning
- Network architectures
  - VGG
  - ResNet
  - U-Net
  - Autoencoders

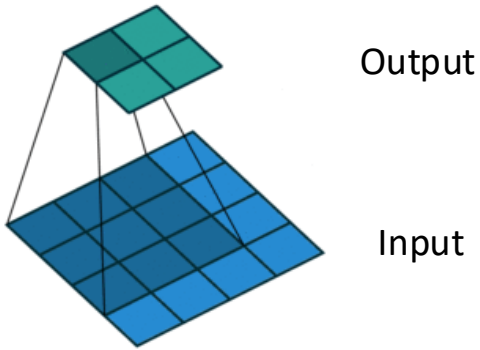
Last Lecture

Recap

Convolutional Architecture



Output feature map resolution depends on *padding* and *stride*

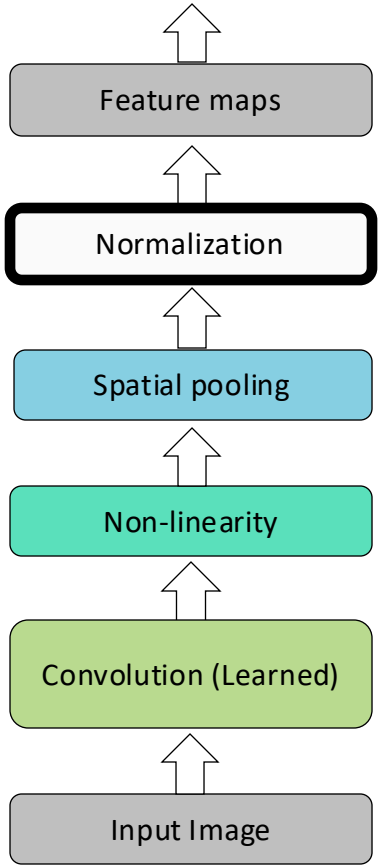


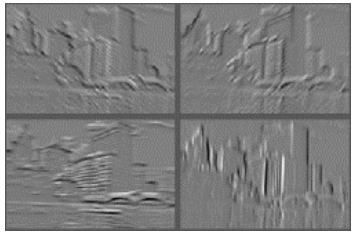
No padding, stride 1

[Slide credit: S. Lazebnik]

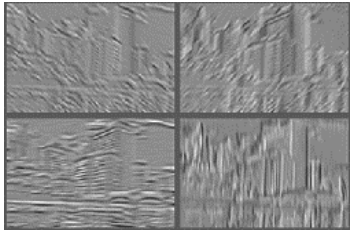
[Animation source]

Convolutional Neural Networks





Feature Maps



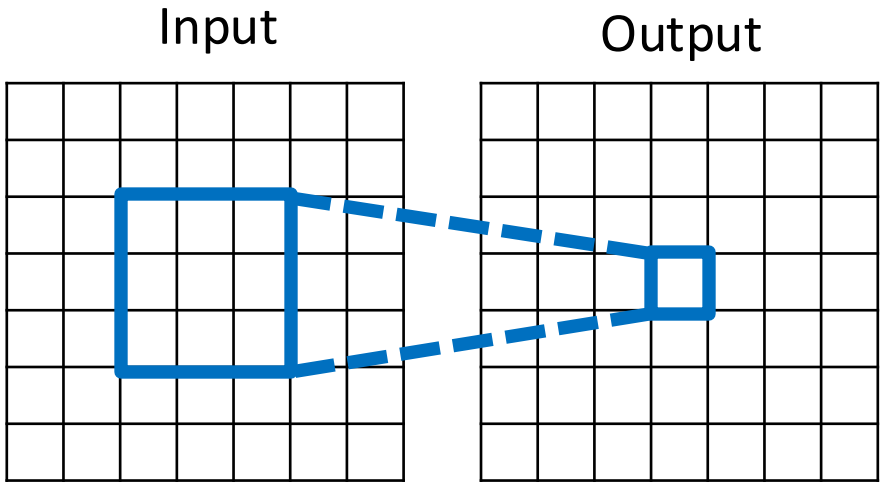
Feature Maps
After Contrast
Normalization

[Slide credit: S. Lazebnik]

Receptive Field

The *receptive field* of a unit is the region of the input feature map whose values contribute to the response of that unit (either in the previous layer or in the initial image)

3x3 convolutions, stride 1

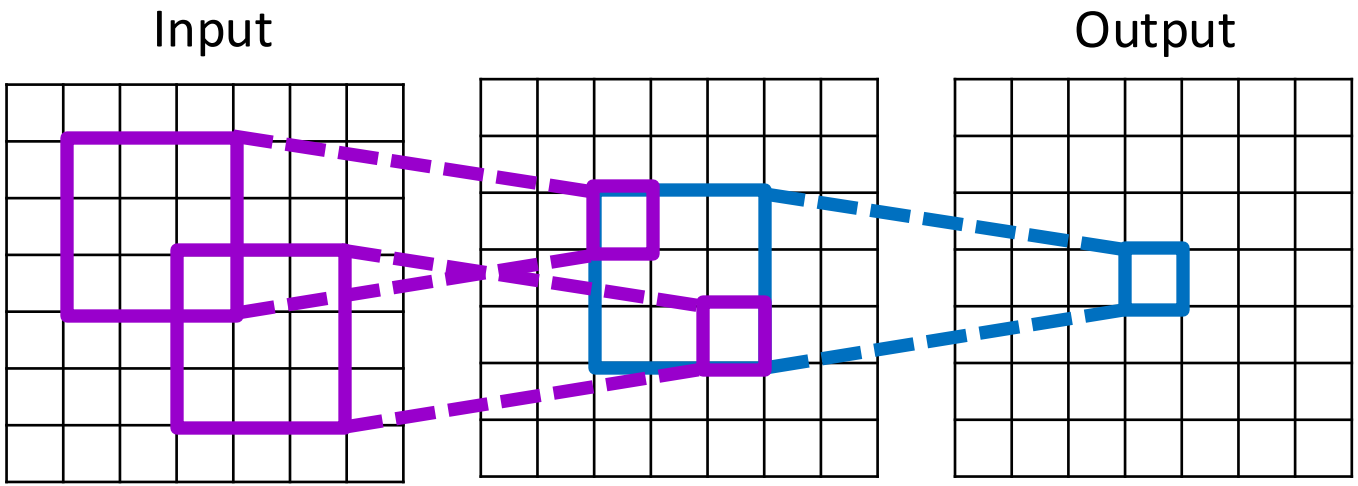


Receptive field size: 3

[Slide credit: S. Lazebnik]

Receptive Field

3x3 convolutions, stride 1

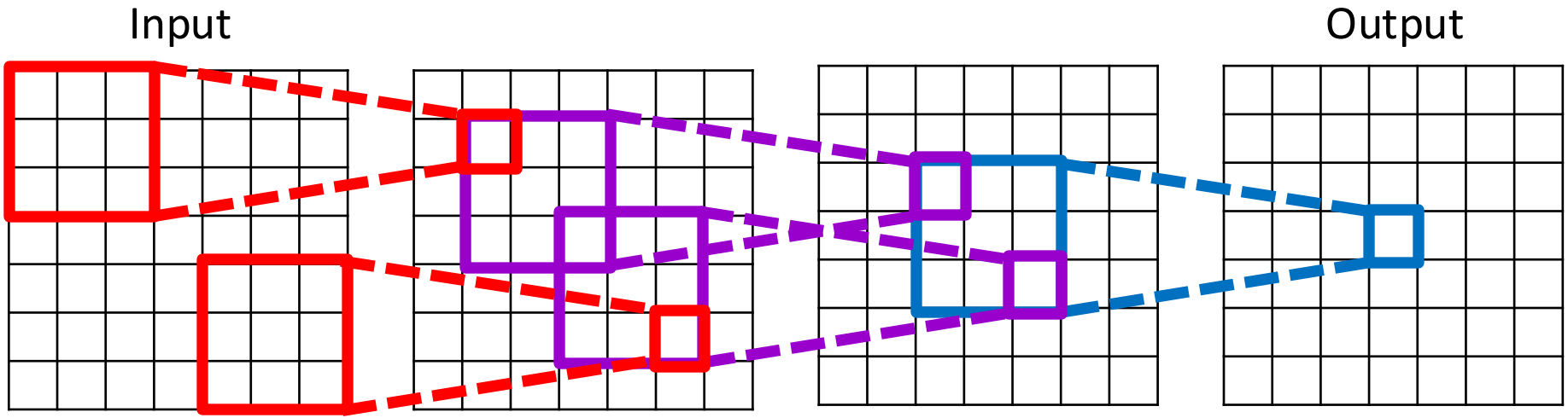


Receptive field size: 5

[Slide credit: S. Lazebnik]

Receptive Field

3x3 convolutions, stride 1



Receptive field size: 7

Each successive convolution adds $F - 1$ to the receptive field size

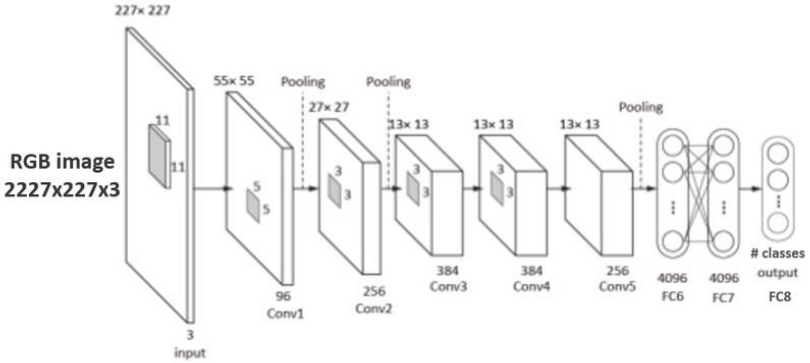
With L layers the receptive field size is $1 + L * (F - 1)$

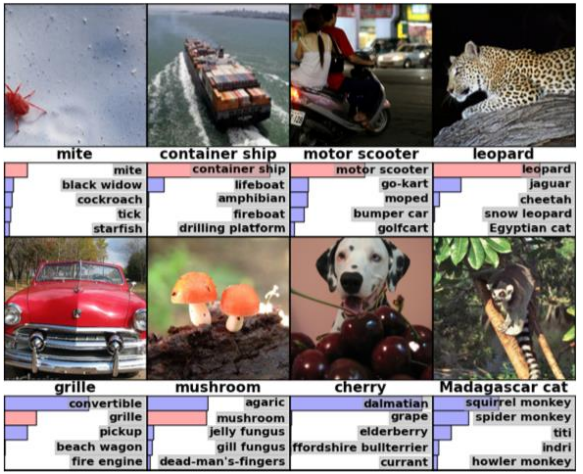
[Slide credit: S. Lazebnik]

ConvNet Architectures

AlexNet (Alex Krishevsky, 2012)

2012 ImageNet Large Scale Visual Recognition Challenge: 15% top 5 error





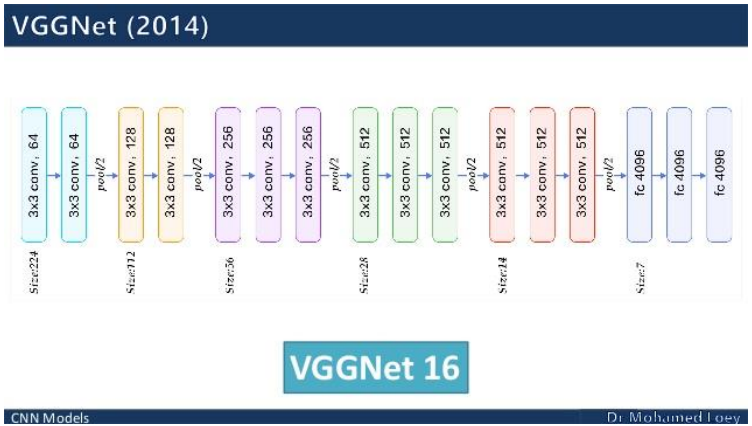
ConvNet Architecture:

- 5 convolutional layers and 3 fully connected layers (1000 classes)
- reduces spacial dimensions: 227x227 (image) → … → 13x13
- increases depth: 3 (image RGB depth) → … → 256 (nr of filters)
- use of Graphical Processing Unit (GPU)

ConvNet Architectures

VGGNet (Zimmerman, Simonyan, 2014)

The VGGNet achieved almost 92.7% top-5 test accuracy in 2014 ImageNet

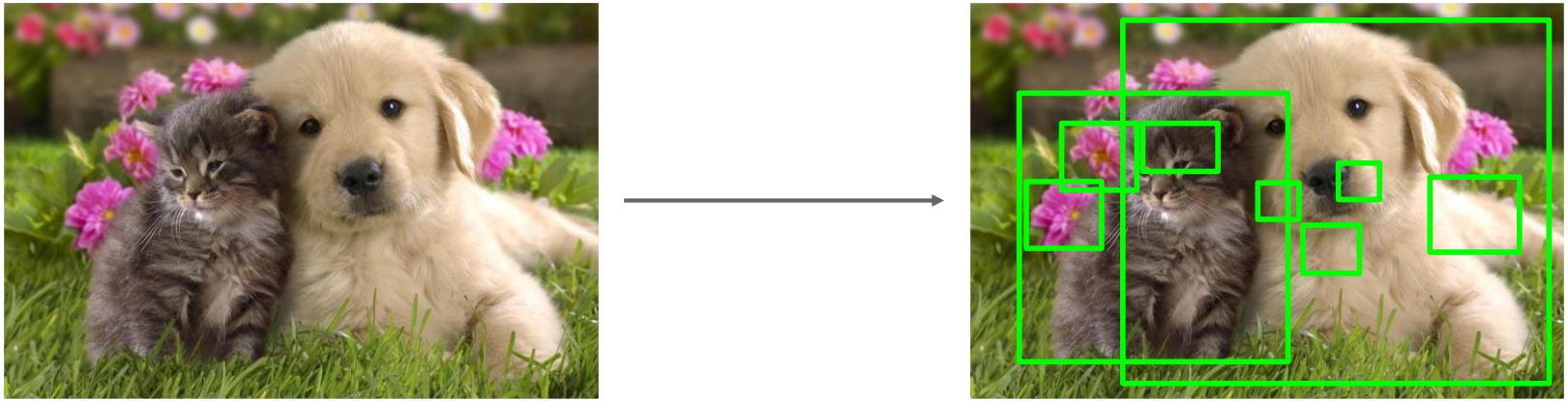


ConvNet Architecture:

- VGG16: 13 convolutional layers and 3 fully connected layers (1000 classes)
- reduces spacial dimensions: 224x224 (image) → … → 7x7
- increases depth: 3 (image RGB depth) → … → 512 (nr of filters)
- Large network. Even larger version: VGG19

Region Proposals

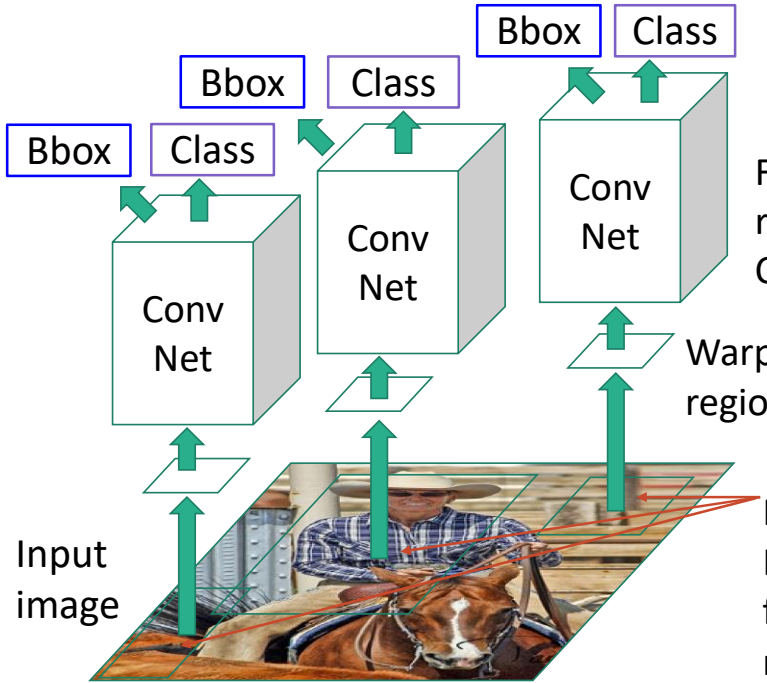
- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region
- proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

[Slide credit: Justin Johnson]

Region-based CNN (R-CNN)



Classify each region

Bounding box regression: Predict "transform" to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

Forward each region through ConvNet

Warped image regions (224x224)

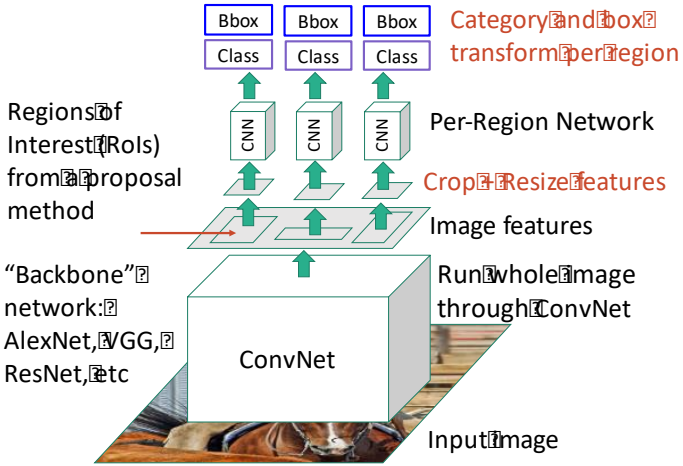
Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

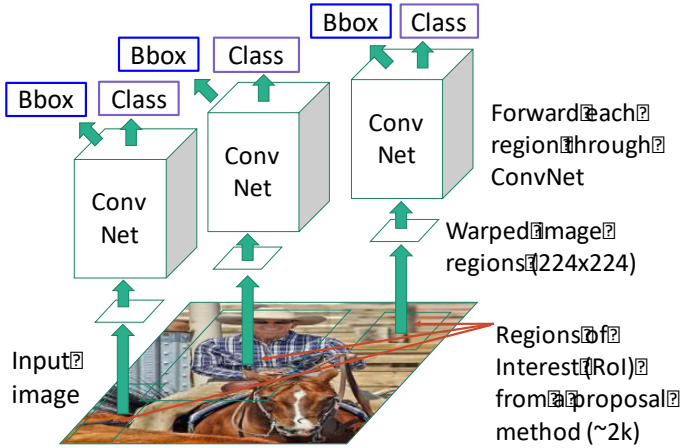
[Slide credit: Justin Johnson]

Fast R-CNN vs. (Slow) R-CNN

Fast R-CNN: Apply differentiable cropping to shared image features



"Slow" R-CNN: Apply differentiable cropping to shared image features

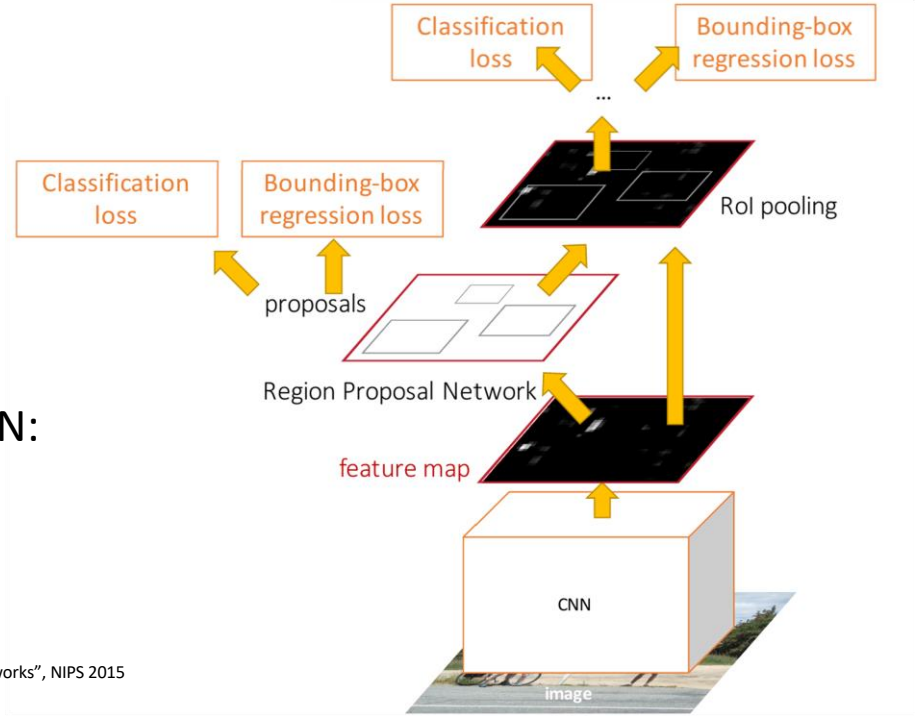


[Slide credit: Justin Johnson]

Faster R-CNN: Learnable Region Proposals

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one

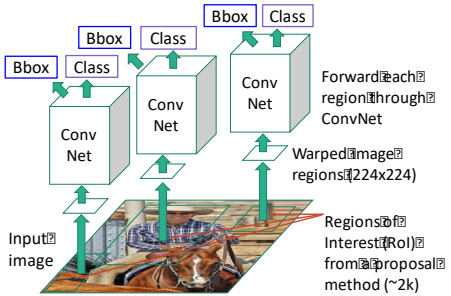


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

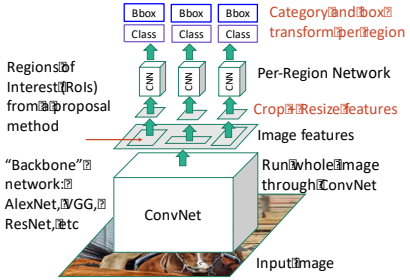
[Slide credit: Justin Johnson]

Overview: R-CNN Variants

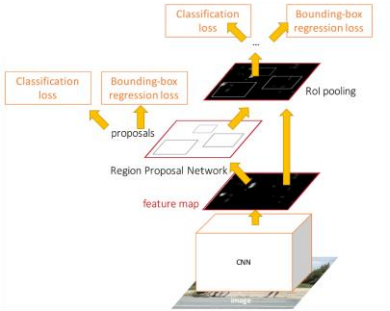
“Slow” R-CNN: Run CNN independently for each region



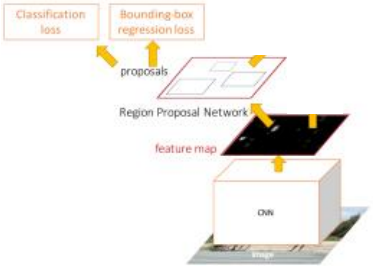
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN



Single-Stage: Fully convolutional detector



[Slide credit: Justin Johnson]

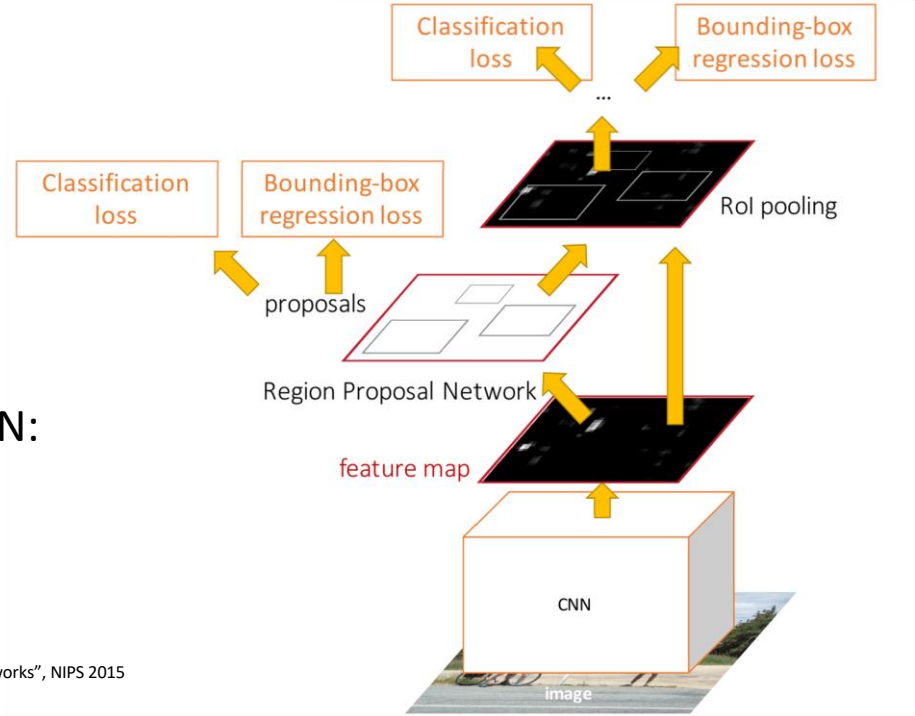
Today's Agenda

- Learning region proposals - Faster R-CNN
- Feature Pyramid Network (FPN)
- Single Shot Detectors (SSD) / Yolo
- Transfer learning
- Network architectures
  - VGG
  - ResNet
  - U-Net
  - Autoencoders

Faster R-CNN: Learnable Region Proposals

Insert **Region Proposal Network (RPN)** to predict proposals from features

Otherwise same as Fast R-CNN: Crop features for each proposal, classify each one

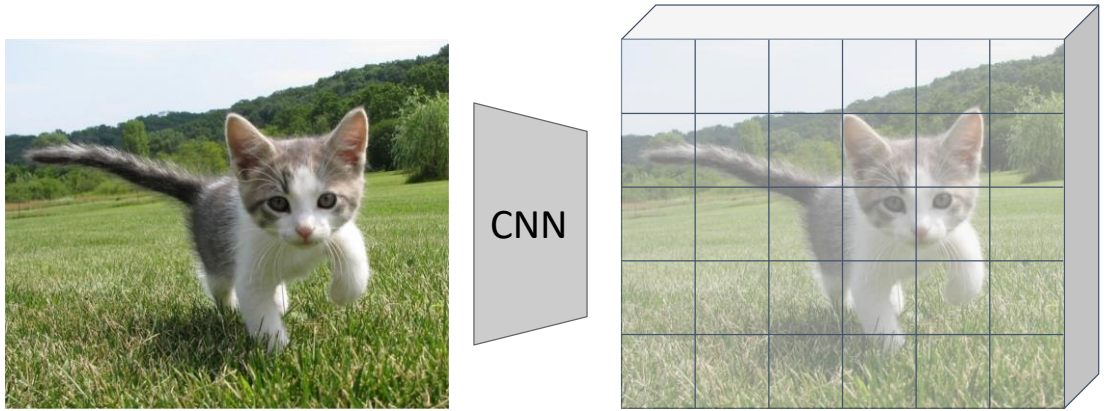


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get
features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

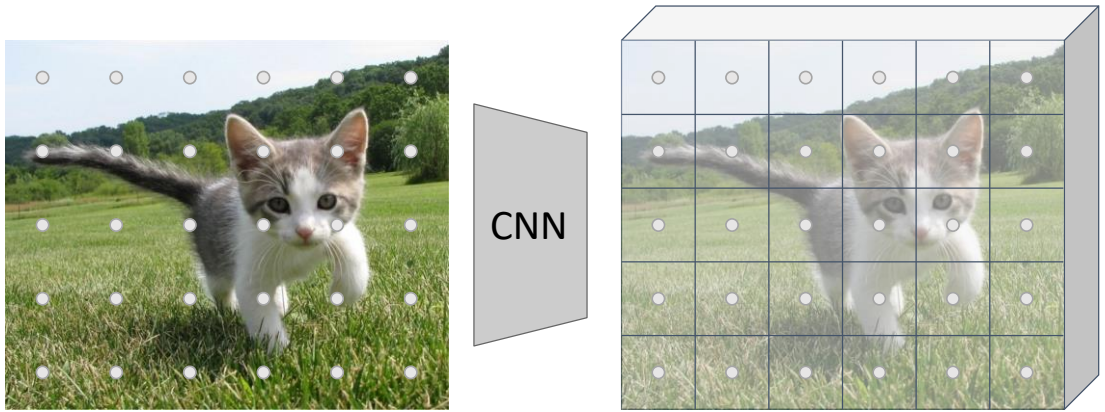
Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

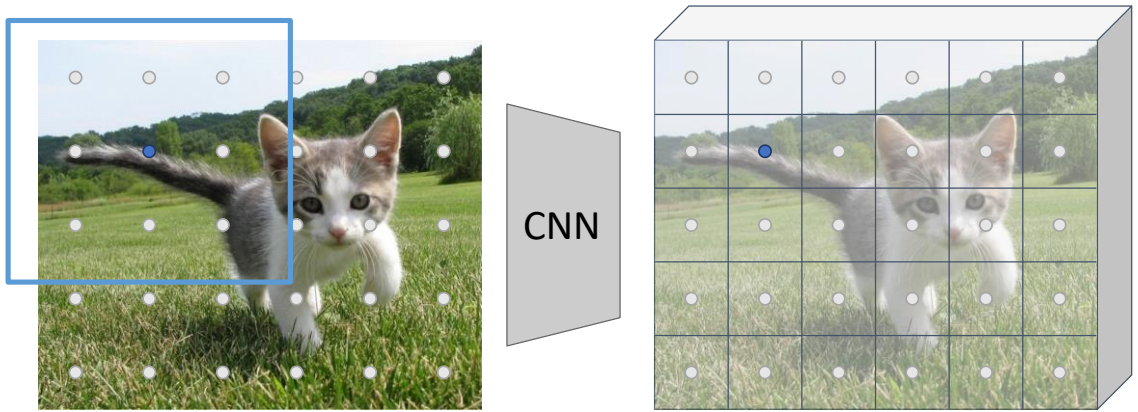
[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Imagine an **anchor box** of fixed size at each point in the feature map

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

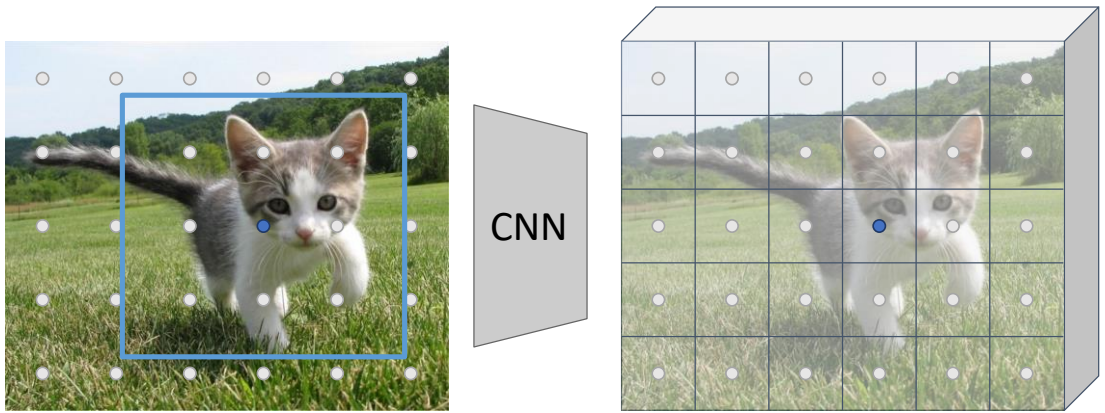
[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Imagine an **anchor box** of fixed size at each point in the feature map

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

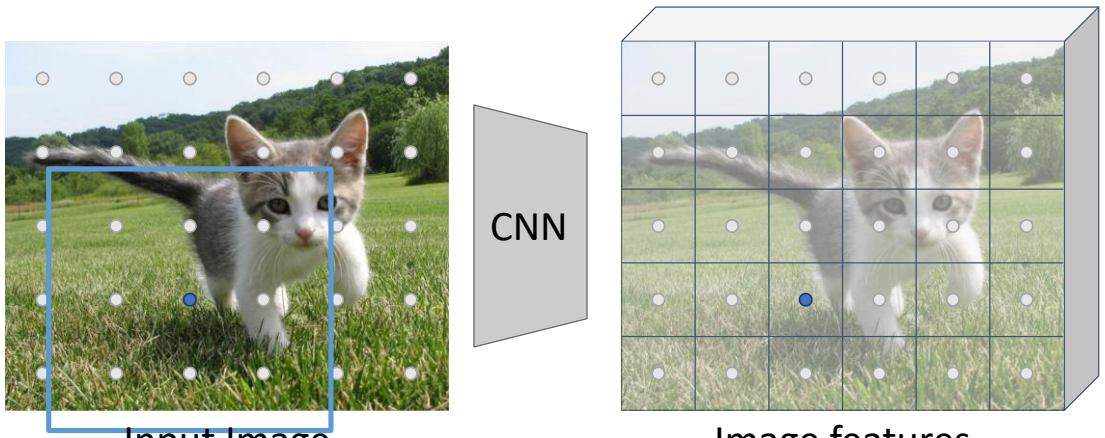
[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Imagine an **anchor box** of fixed size at each point in the feature map

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Input Image
(e.g. 3 x 640 x 480)

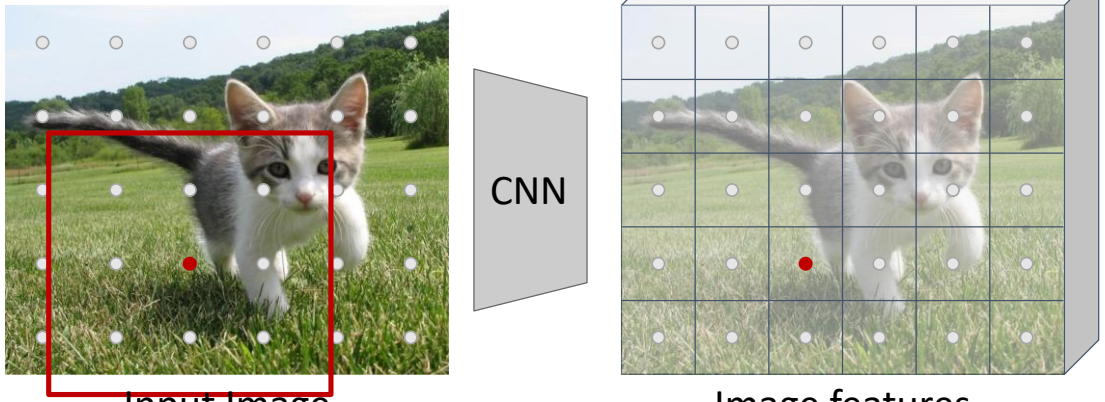
Image features
(e.g. 512 x 5 x 6)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

Each feature corresponds to a point in the input

Image features
(e.g. 512 x 5 x 6)

Imagine an **anchor box** of fixed size at each point in the feature map

Classify each anchor as positive (object) or negative (no object)

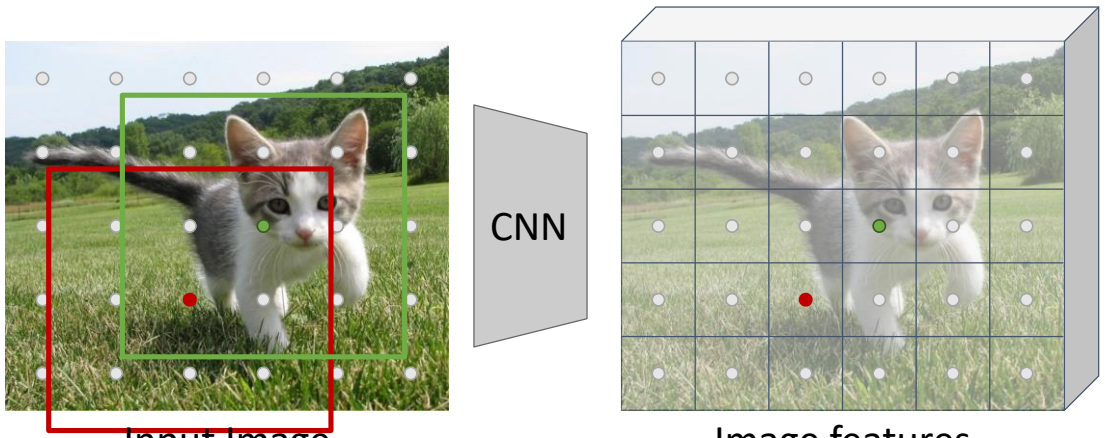
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

Imagine an **anchor box** of fixed size at each point in the feature map

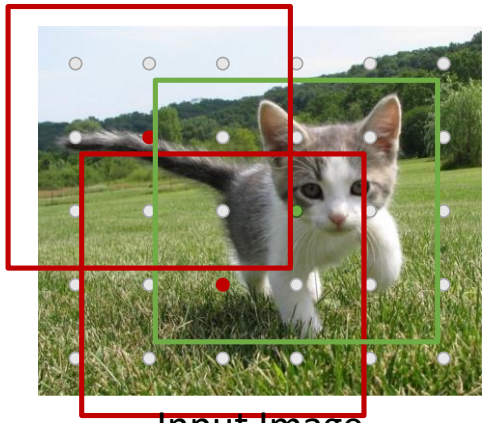
Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

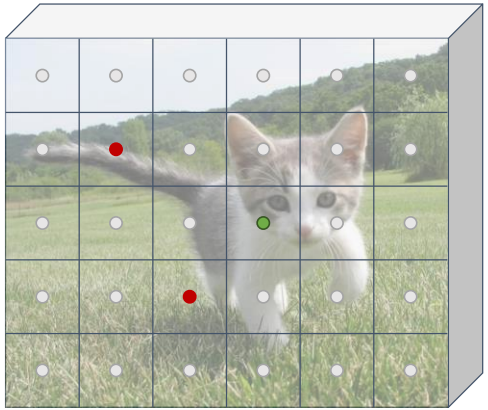


Image features
(e.g. 512 x 5 x 6)

Imagine an **anchor box** of fixed size at each point in the feature map

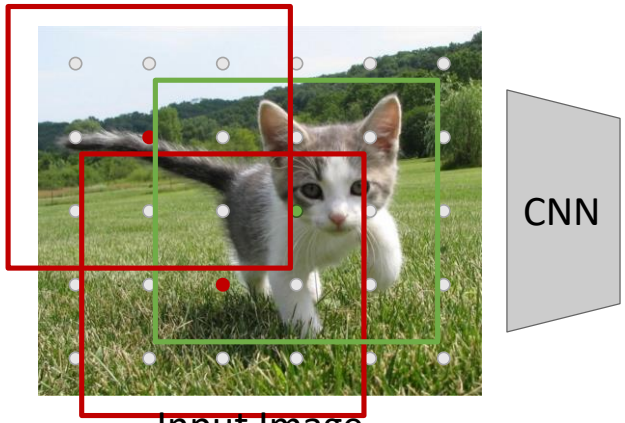
Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

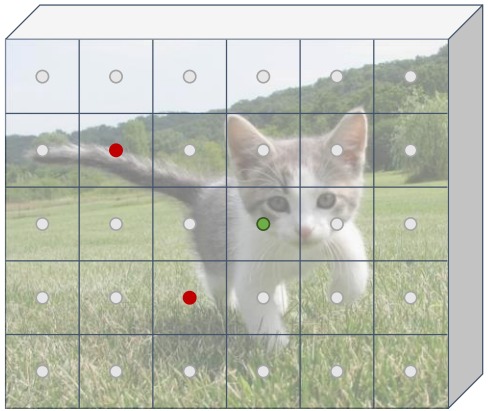


Image features
(e.g. 512 x 5 x 6)

Predict object vs not object scores for all anchors with a conv layer (512 input filters, 2 output filters)

Conv

Anchor is object?
2 x 5 x 6

Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

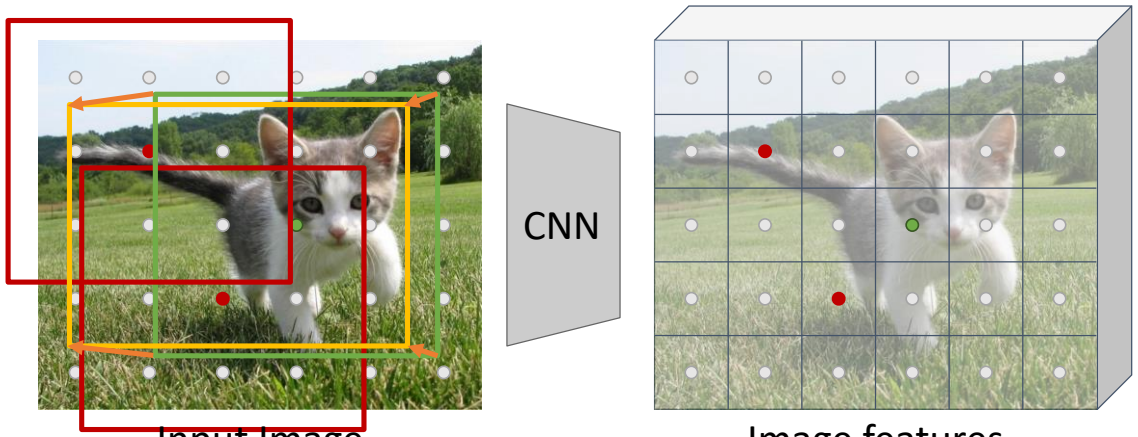
[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

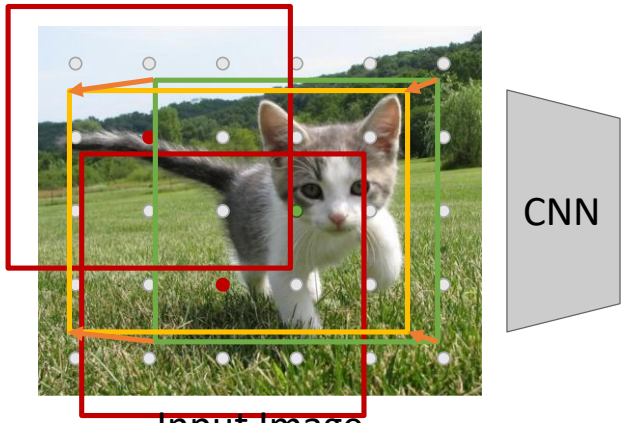
Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

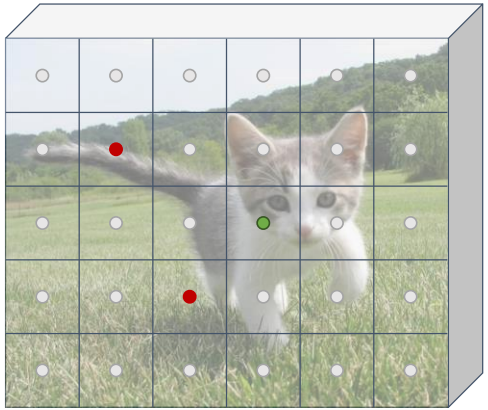


Image features
(e.g. 512 x 5 x 6)

Conv

For positive anchors, also predict a transform that converting the anchor to the GT box (like R-CNN)
Predict transforms with conv

Anchor is object?
2 x 5 x 6

Anchor transforms
4 x 5 x 6

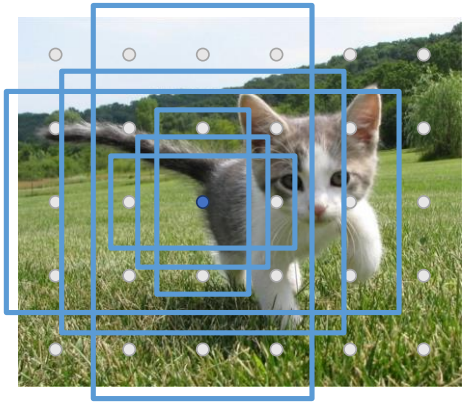
Classify each anchor as positive (object) or negative (no object)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

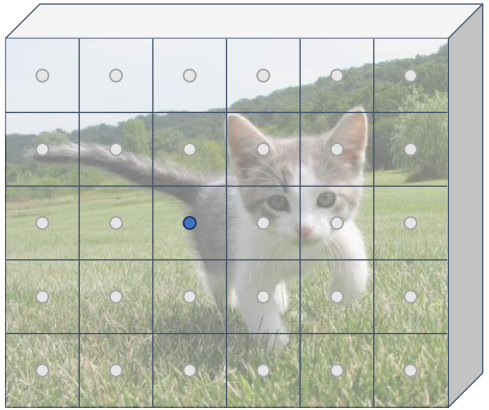


Image features
(e.g. 512 x 5 x 6)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Conv

→ Anchor is object?
2K x 5 x 6

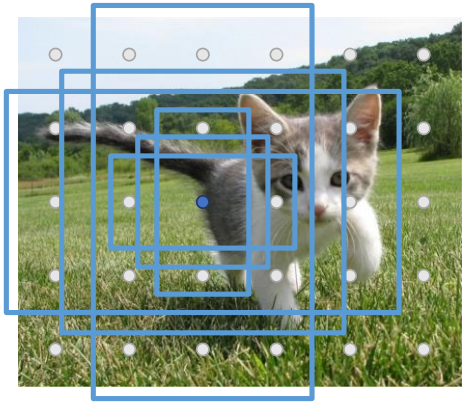
→ Anchor transforms
4K x 5 x 6

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

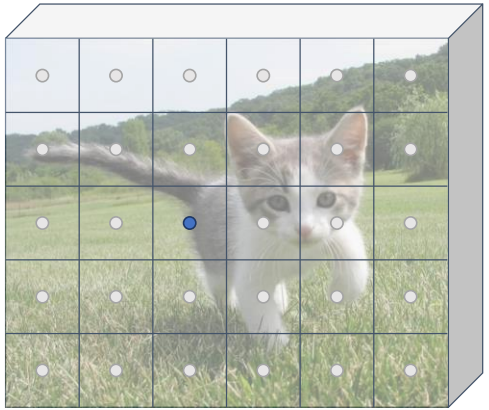


Image features
(e.g. 512 x 5 x 6)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Conv

→ Anchor is object?
2K x 5 x 6

→ Anchor transforms
4K x 5 x 6

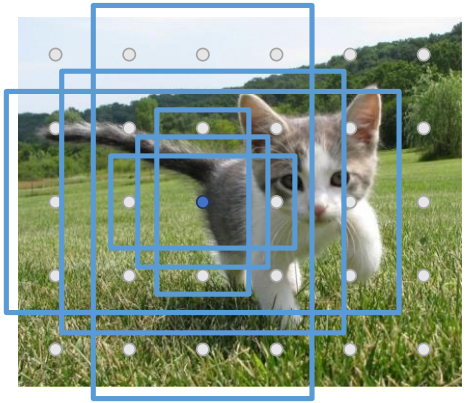
During training, supervised positive / negative anchors and box transforms like R-CNN

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

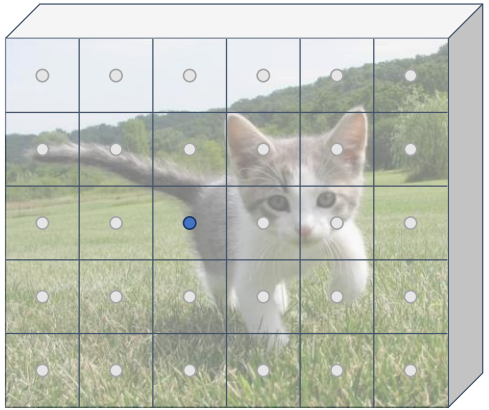


Image features
(e.g. 512 x 5 x 6)

Conv

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

→ Anchor is object?
2K x 5 x 6

→ Anchor transforms
4K x 5 x 6

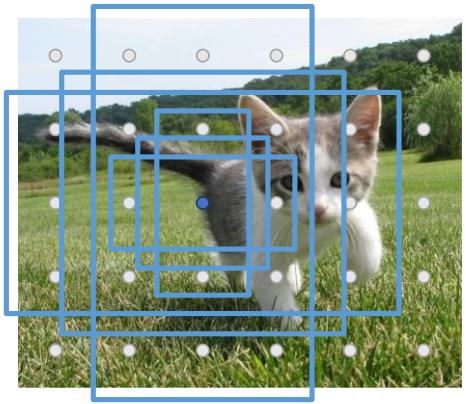
Positive anchors: >= 0.7 IoU with some GT box (plus highest IoU to each GT)

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

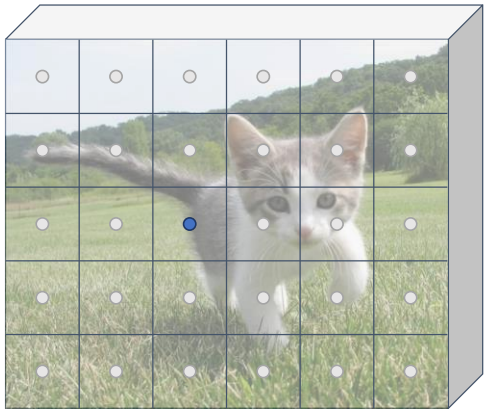


Image features
(e.g. 512 x 5 x 6)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Conv

→ Anchor is object?
2K x 5 x 6

→ Anchor transforms
4K x 5 x 6

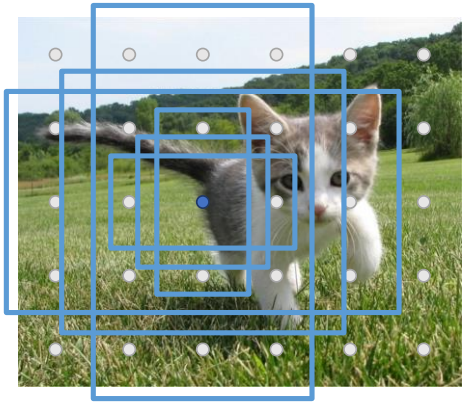
Negative anchors: < 0.3 IoU with all GT boxes. Don't supervised transforms for negative boxes.

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

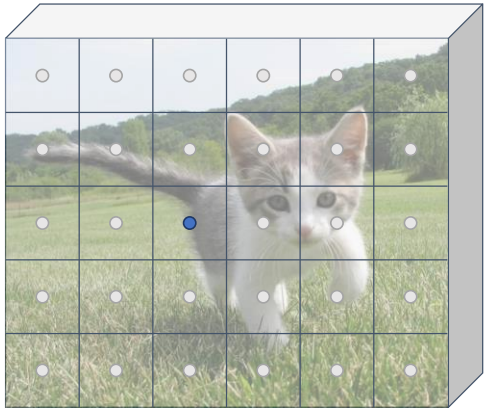


Image features
(e.g. 512 x 5 x 6)

Conv

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

→ Anchor is object?
2K x 5 x 6

→ Anchor transforms
4K x 5 x 6

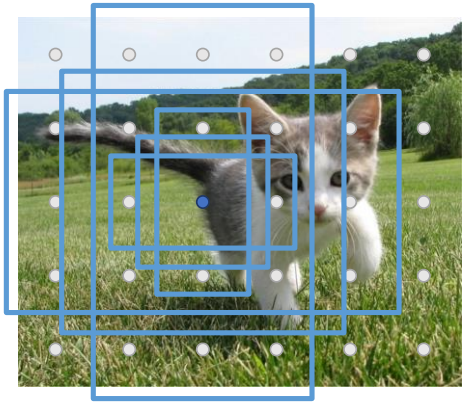
Neutral anchors: between 0.3 and 0.7 IoU with all GT boxes; ignored during training

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Region Proposal Network (RPN)

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

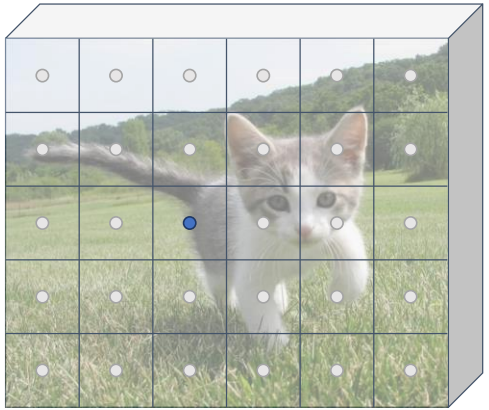


Image features
(e.g. 512 x 5 x 6)

In practice: Rather than using one anchor per point, instead consider K different anchors with different size and scale (here K = 6)

Conv

→ Anchor is object?
2K x 5 x 6

→ Anchor transforms
4K x 5 x 6

At test-time, sort all K*5*6 boxes by their positive score, take top 300 as our region proposals

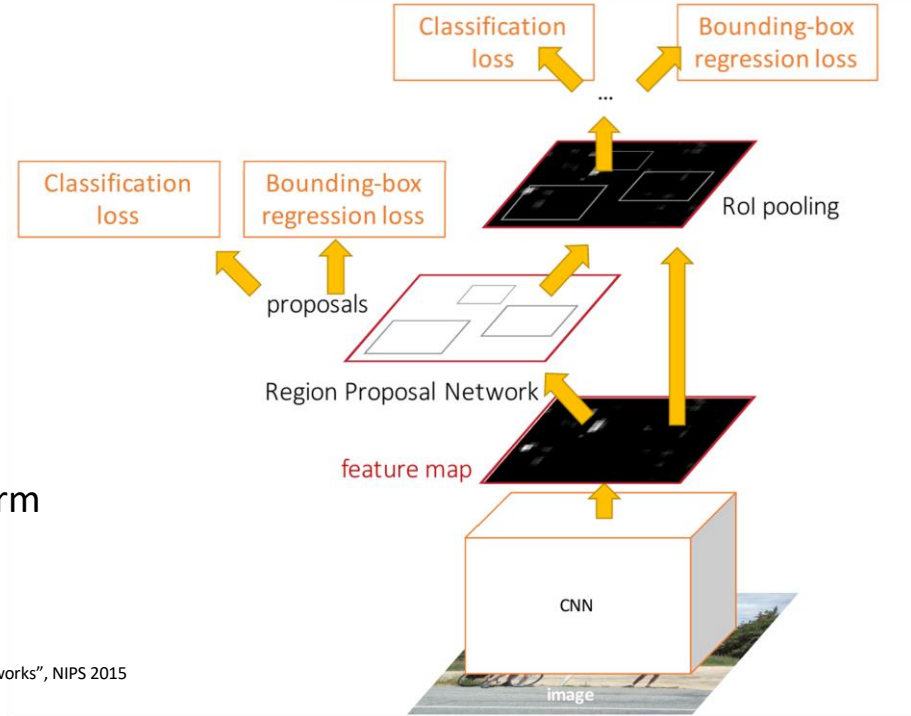
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

[Slide credit: Justin Johnson]

Faster R-CNN: Learnable Region Proposals

Jointly train with 4 losses:

1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

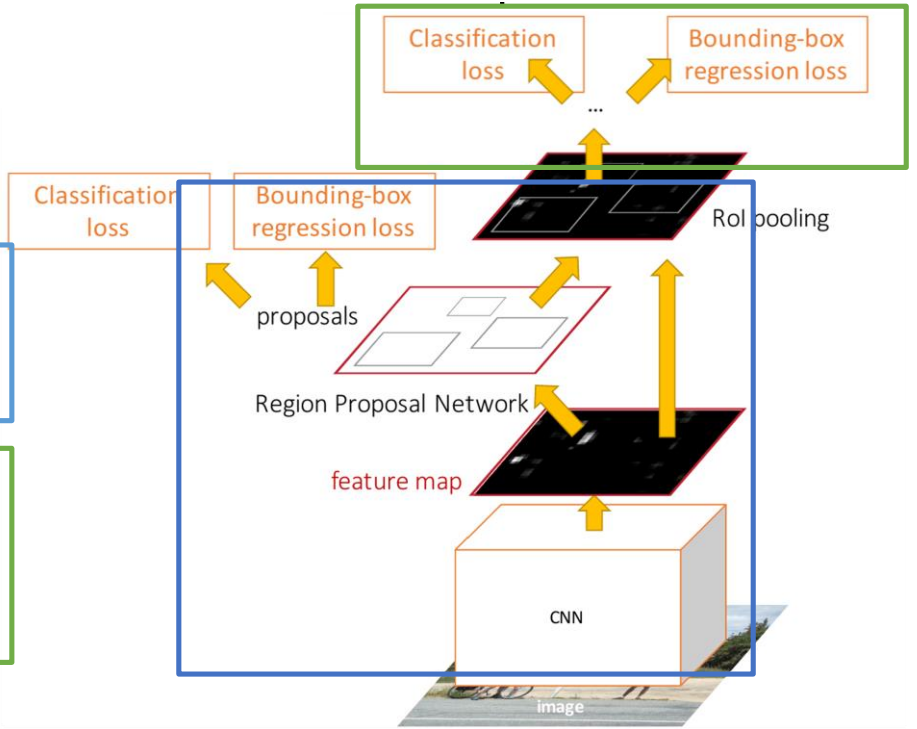
[Slide credit: Justin Johnson]

Faster R-CNN: Learnable Region Proposals

Faster R-CNN is a
Two-stage object detector

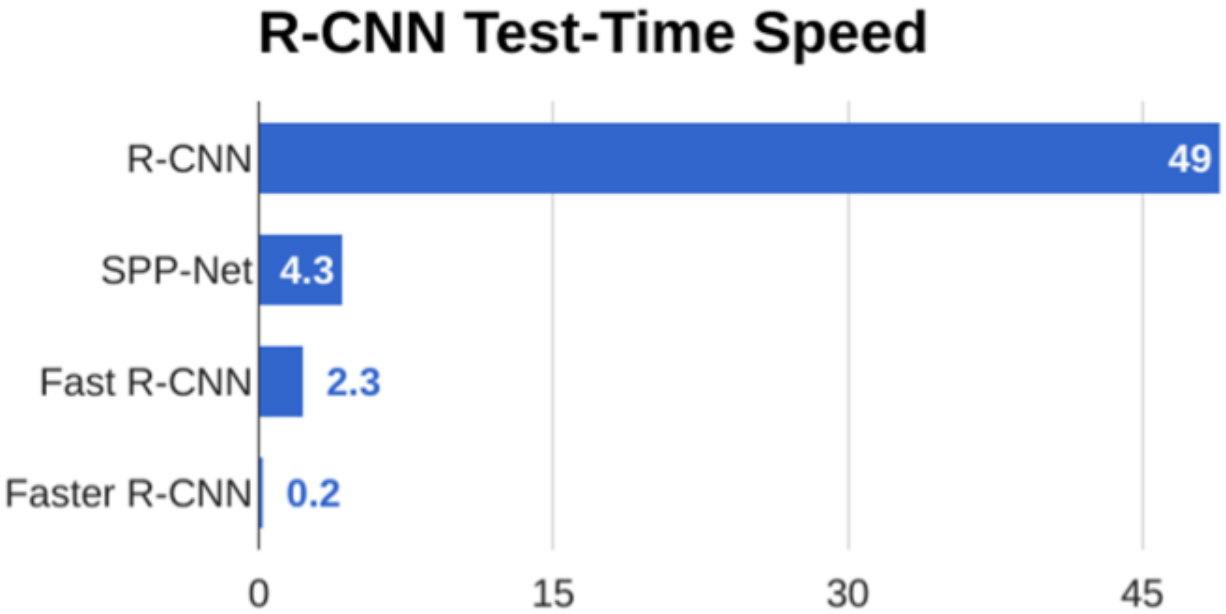
First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



[Slide credit: Justin Johnson]

Faster R-CNN: Learnable Region Proposals



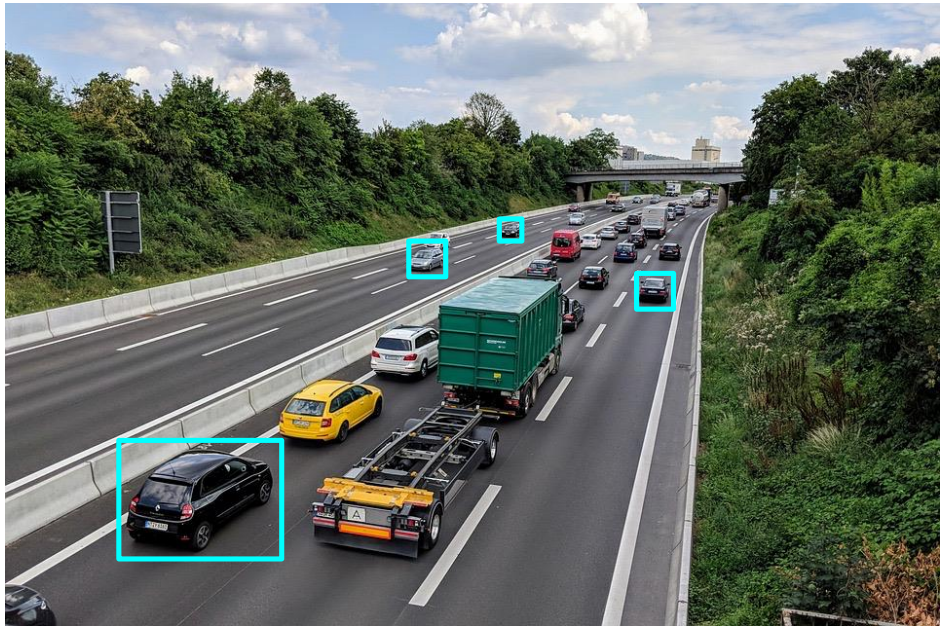
[Slide credit: Justin Johnson]

Today's Agenda

- Learning region proposals - Faster R-CNN
- Feature Pyramid Network (FPN)
- Single Shot Detectors (SSD) / Yolo
- Transfer learning
- Network architectures
  - VGG
  - ResNet
  - U-Net
  - Autoencoders

Dealing with Scale Variations

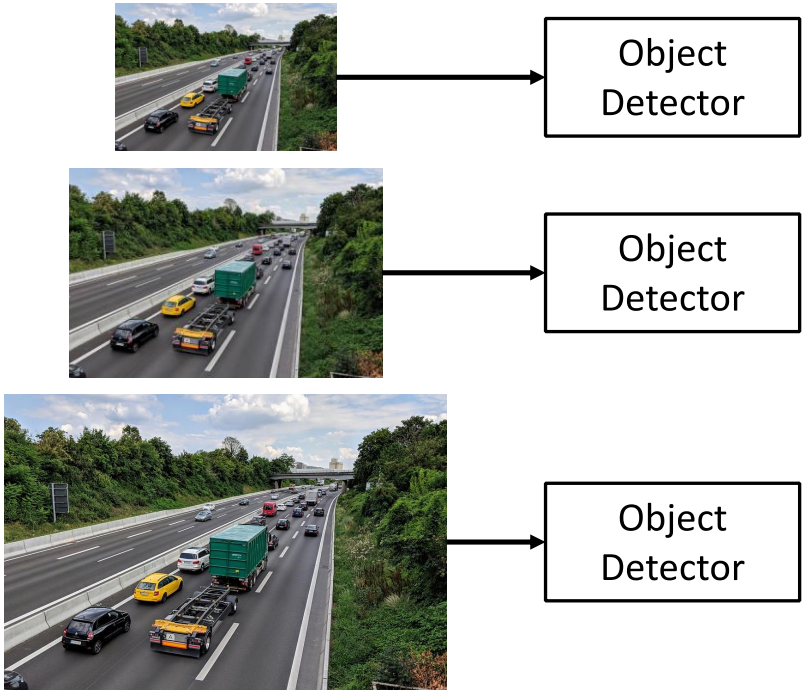
We need to detect objects of many different scales.
How to improve *scale invariance* of the detector?



[Slide credit: Justin Johnson]

Scale Variations: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.



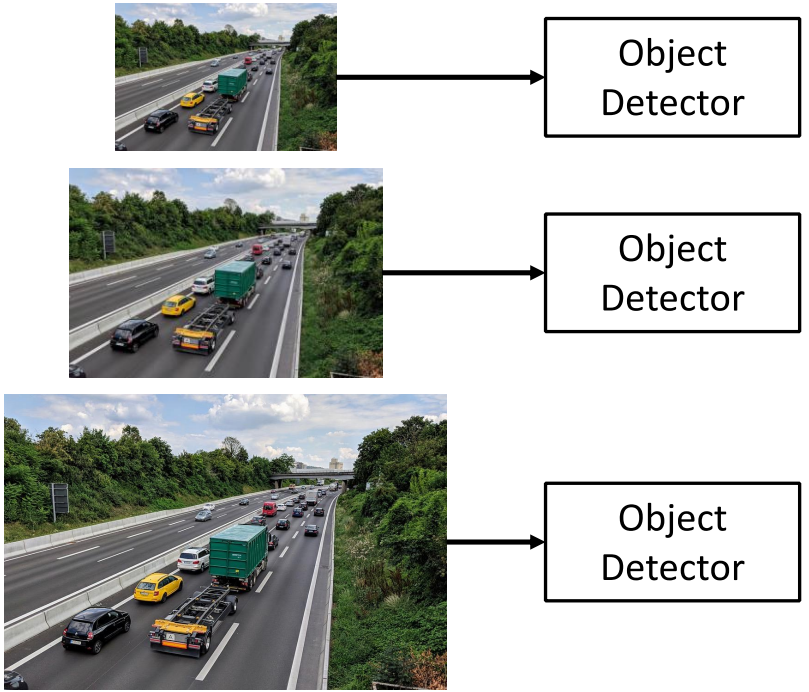
Lin et al, “Feature Pyramid Networks for Object Detection”, ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

Problem: Expensive! Don't share any computation between scales

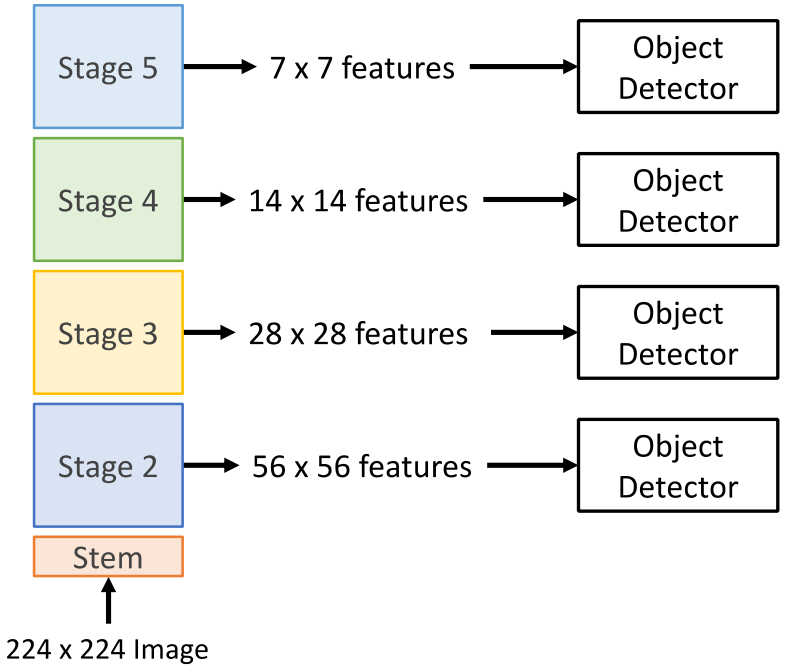


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Multiscale Features

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level



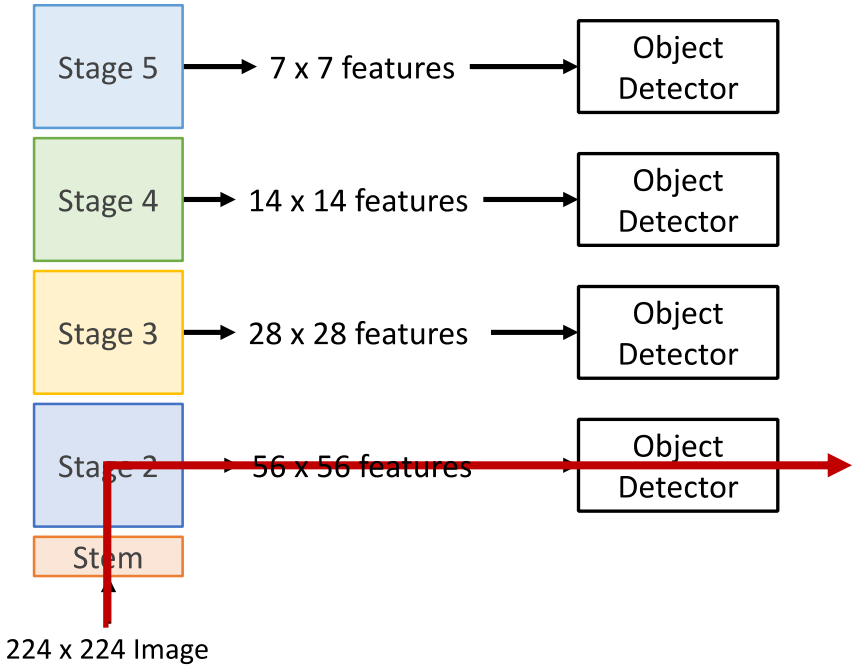
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Multiscale Features

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level

Problem: detector on early features doesn't make use of the entire backbone; doesn't get access to high-level features

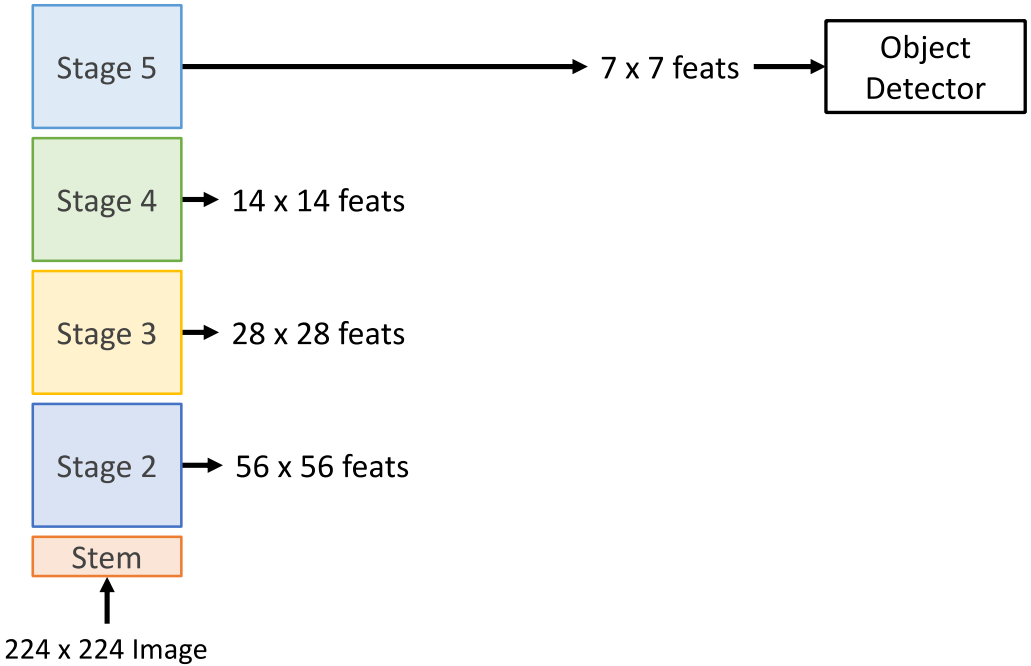


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

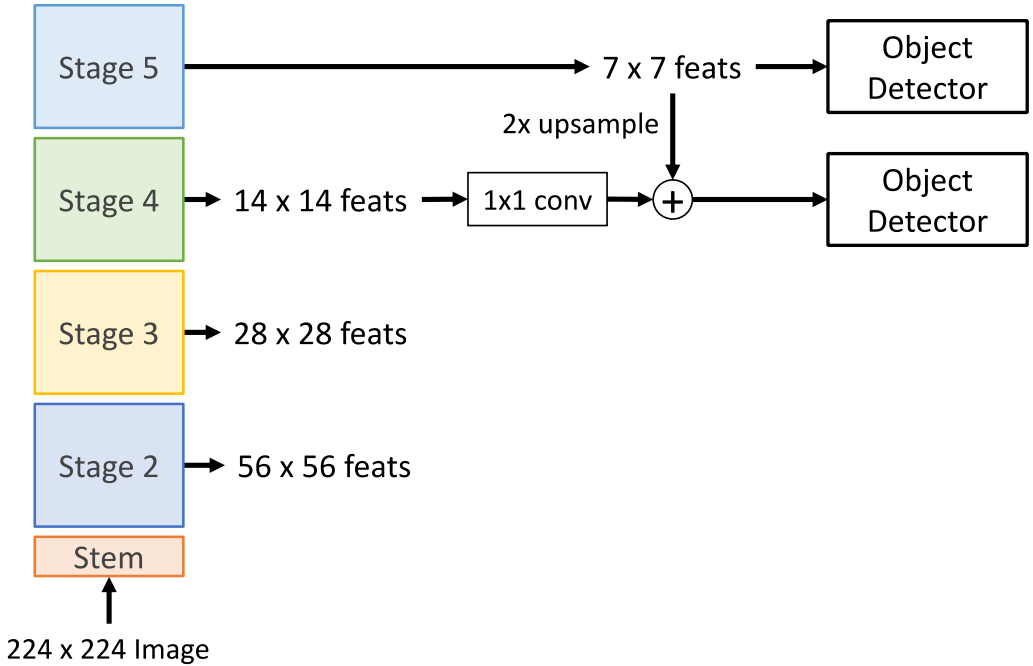


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

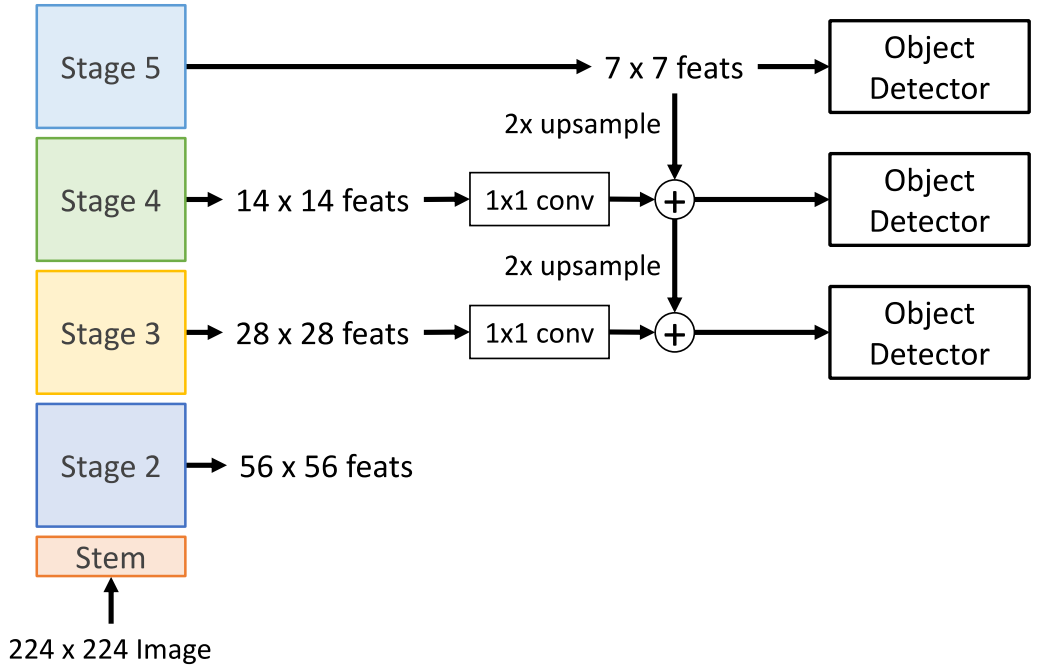


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

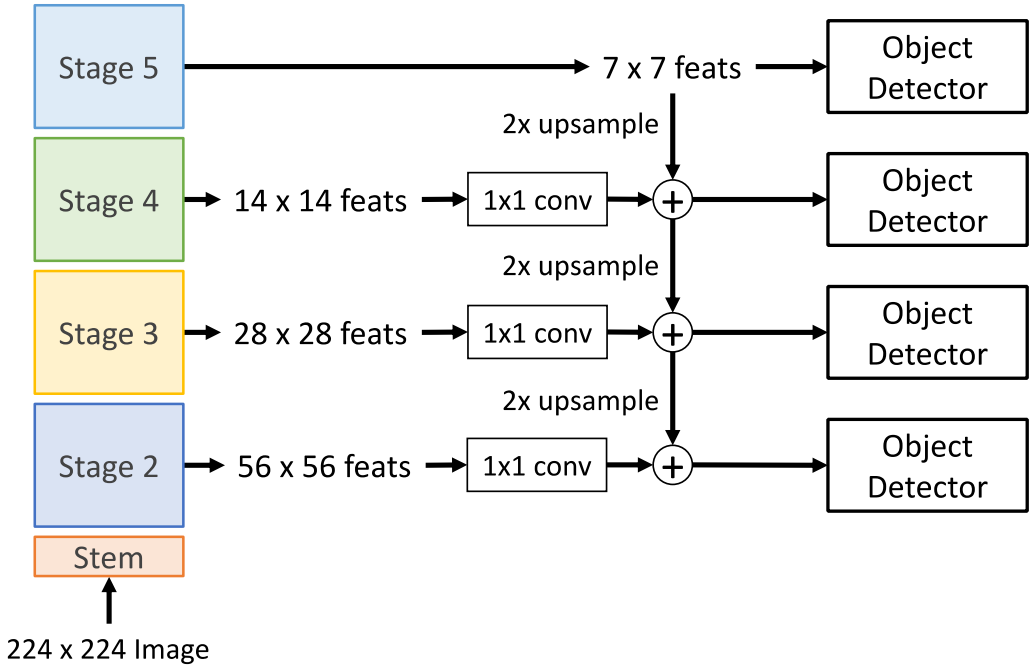


Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features



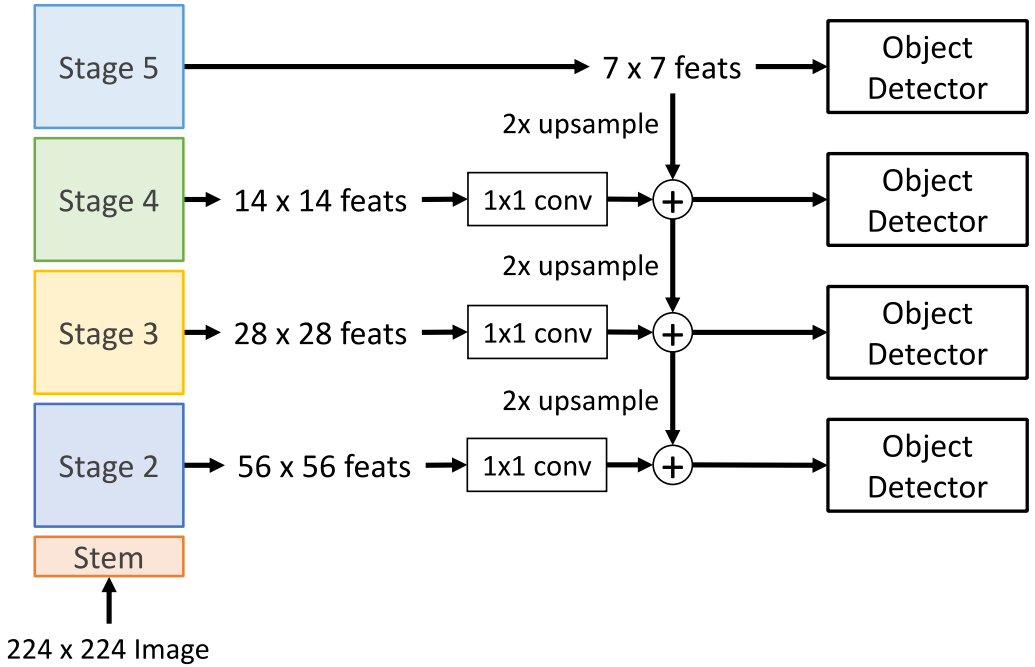
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

[Slide credit: Justin Johnson]

Scale Variations: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice



Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

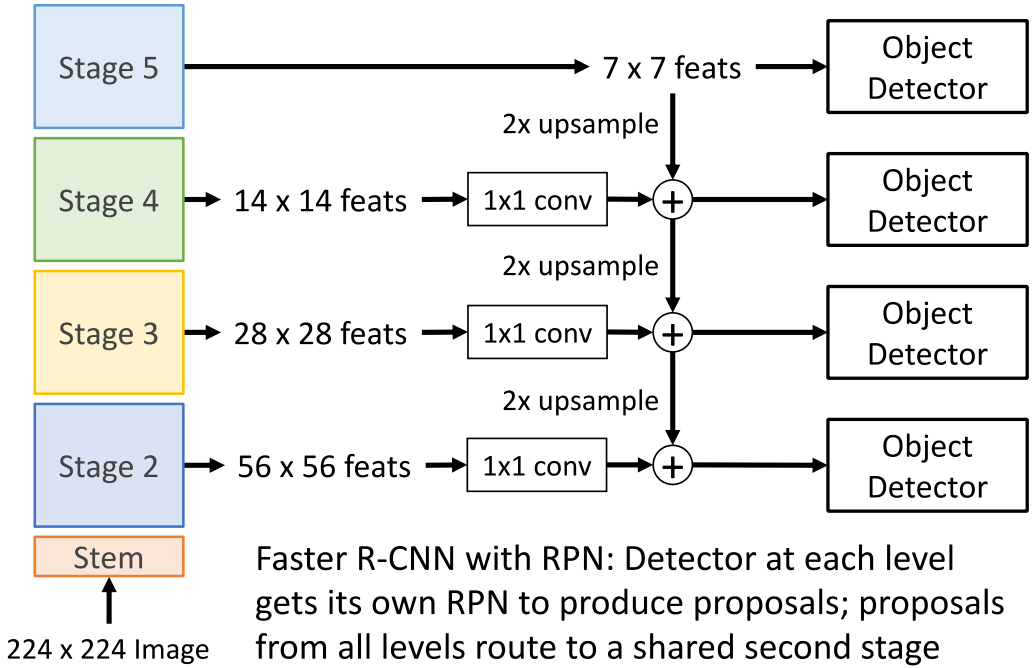
[Slide credit: Justin Johnson]

Scale Variations: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice

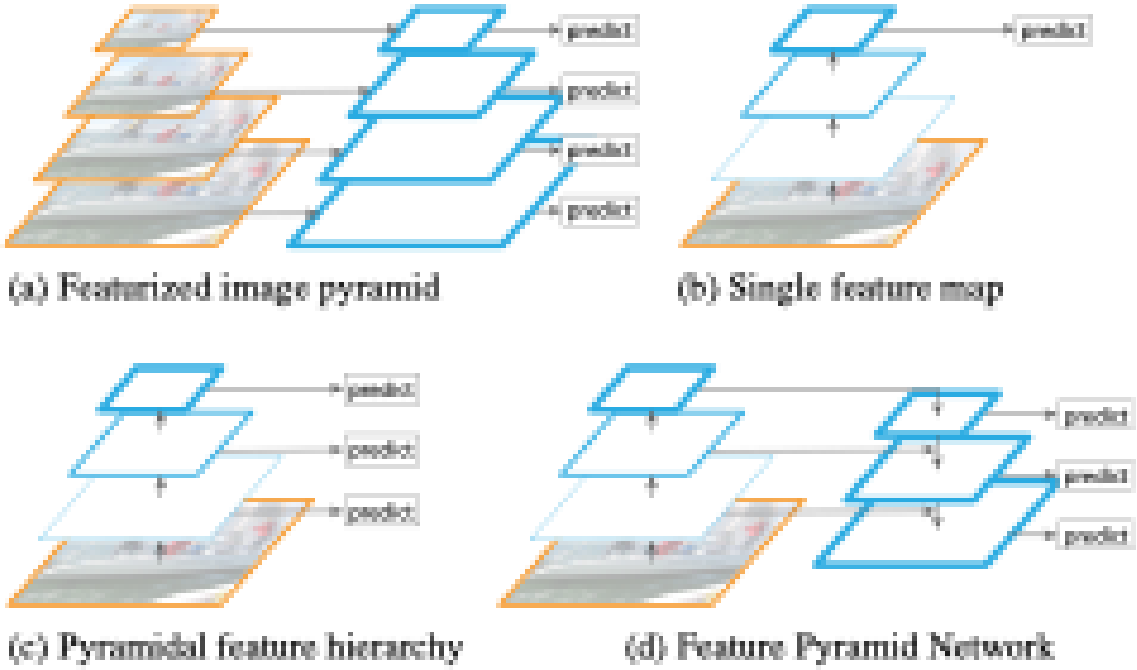
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017



Faster R-CNN with RPN: Detector at each level gets its own RPN to produce proposals; proposals from all levels route to a shared second stage

[Slide credit: Justin Johnson]

Scale Variations: Feature Pyramid Network



[Lin et al., "Feature Pyramid Networks for Object Detection", CVPR 2017]

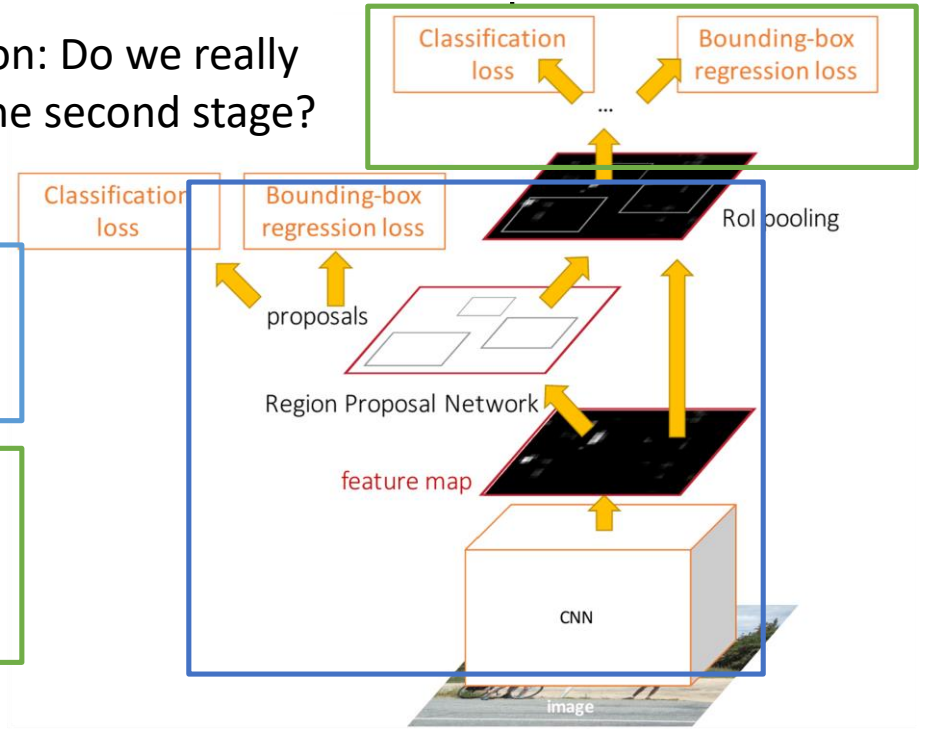
Faster R-CNN: Learnable Region Proposals

Question: Do we really need the second stage?

Faster R-CNN is a
Two-stage object detector

First stage: Run once per image
- Backbone network
- Region proposal network

Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset



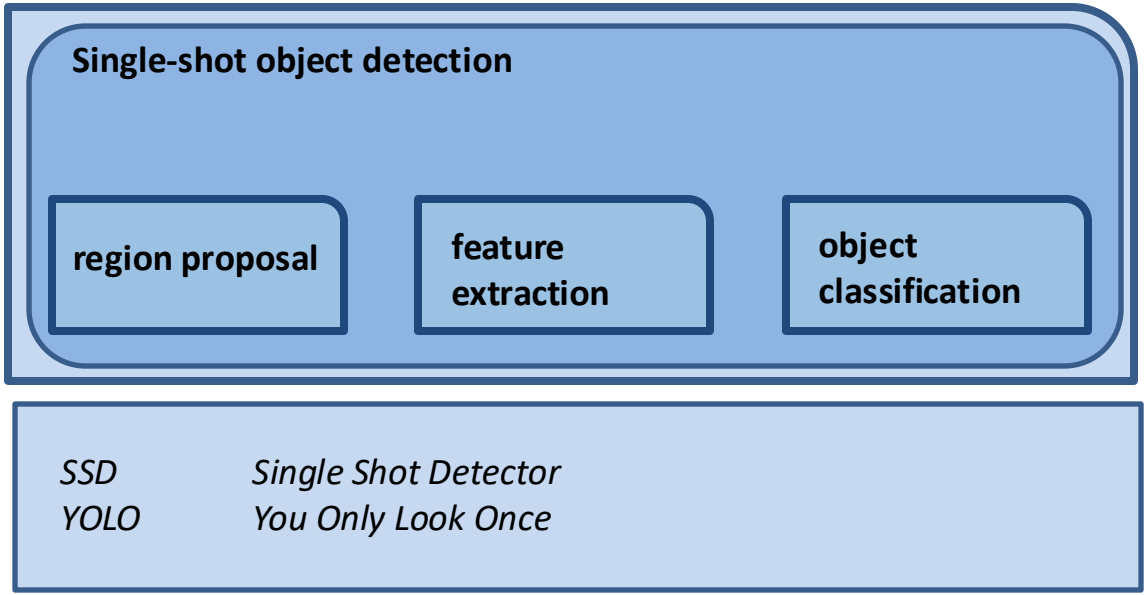
[Slide credit: Justin Johnson]

Today's Agenda

- Learning region proposals - Faster R-CNN
- Feature Pyramid Network (FPN)
- Single Shot Detectors (SSD) / Yolo
- Transfer learning
- Network architectures
  - VGG
  - ResNet
  - U-Net
  - Autoencoders

Single-Shot Detection

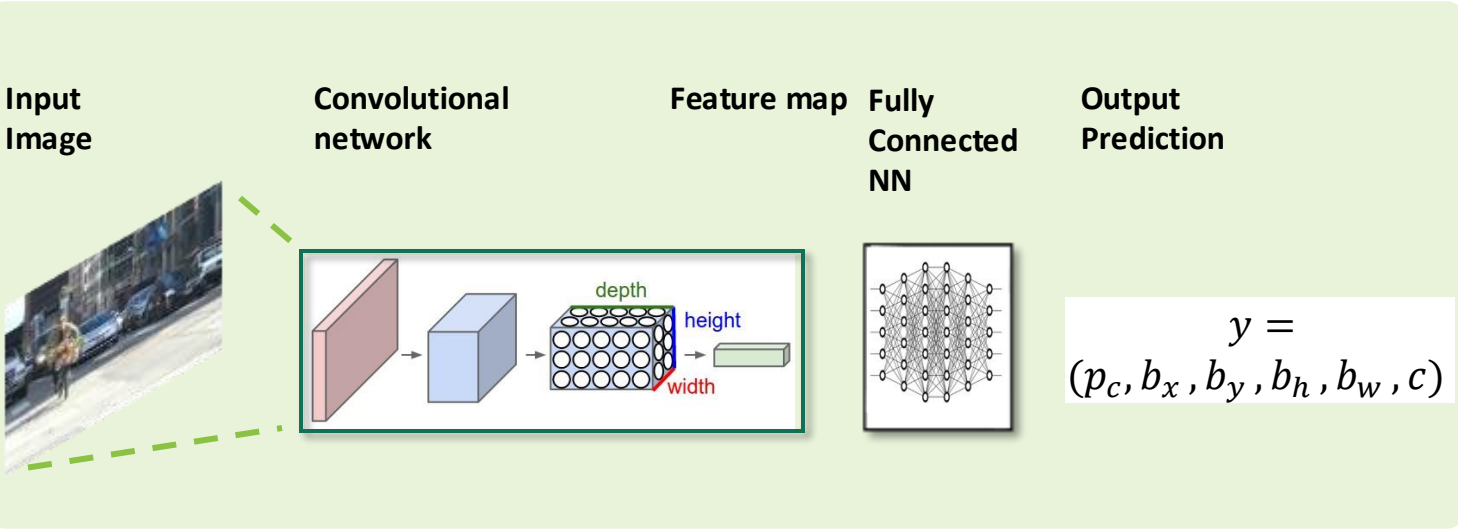
Single-Shot Detection methods do the region proposal, feature extraction and classification steps in one go



Single-Shot Detection

Single-Shot Detectors:

- Feature map is created for the whole image
- Prediction is a combination of bounding box coordinates and class confidence.



Single-Shot Detection

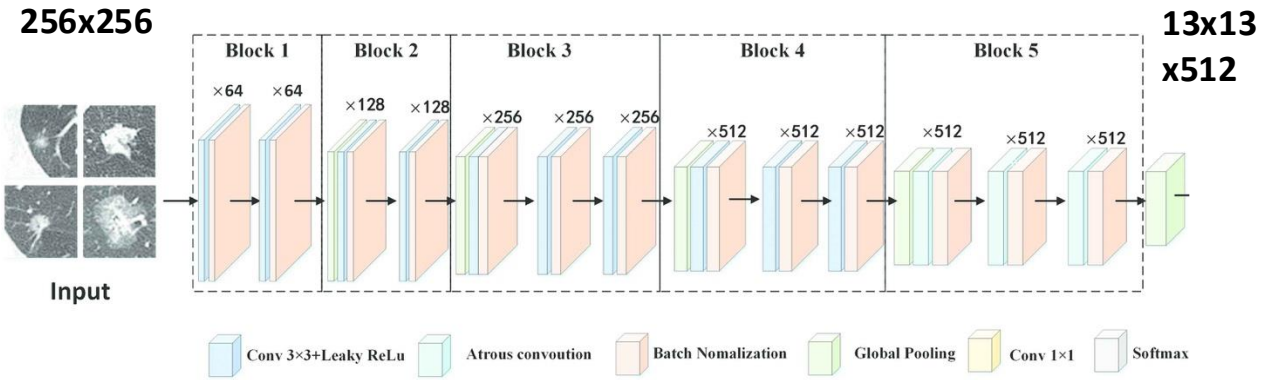
Anchor-based single-shot detectors work with following steps:

A. Divide image in grid cells
B. Pre-define anchor–boxes
C. Refine box coordinates
D. Train class confidence
E. Filter best predictions

Single-Shot Detection

A. Divide image in grid cells

Use a convolutional neural network to divide image in grid cells and extract convolutional features for each grid cell

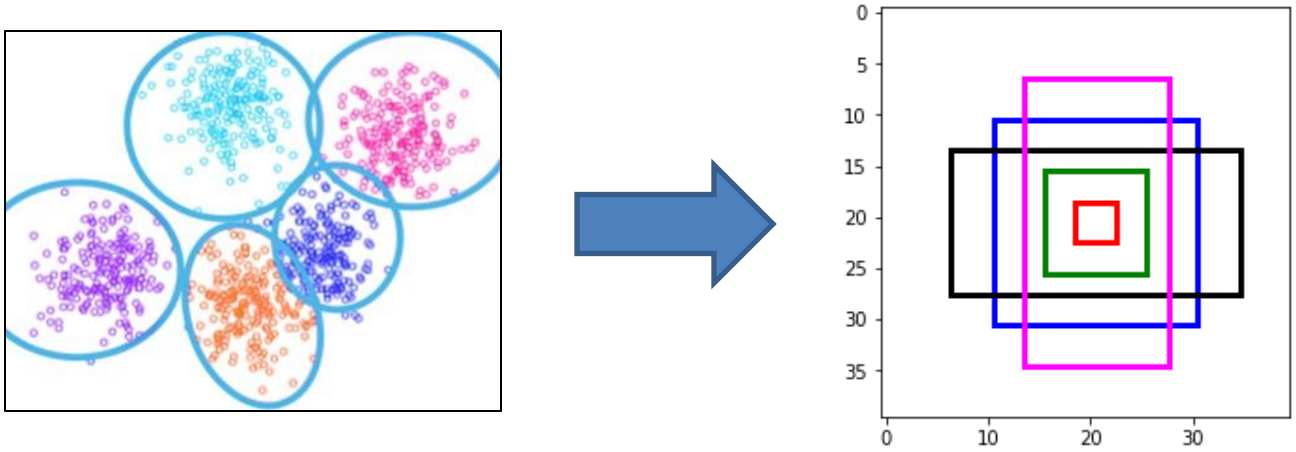


CNN typically reduces dimension in each convolutional block (by pooling)
This example CNN leads to an Feature Map with 512 features (filter outcomes) for each of 13x13 grid cells.

Single-Shot Detection

B. Pre-define anchor boxes

Define default anchor boxes with typical object sizes and aspect ratio’s



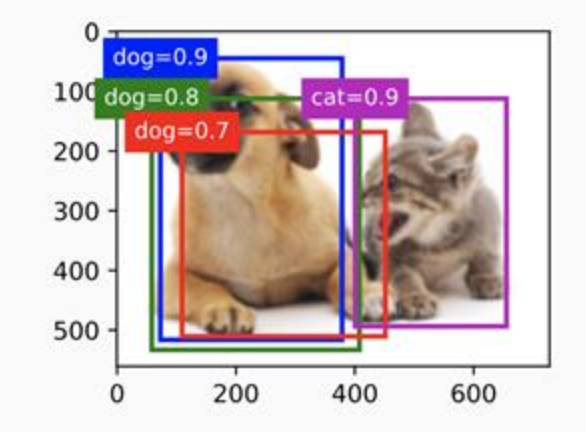
5 default anchor boxes are defined manually or based on k-means clustering on an existing image dataset. This step is usually done off-line (not included in CNN training)

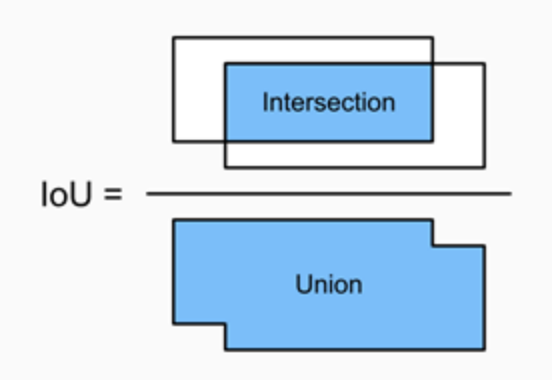
Single-Shot Detection

C. Refine box coordinates

For each grid cell, a number of box sizes can be learned by adjusting the pre-defined anchor boxes.

Learning is by minimizing Loss (Intersection over Union) between prediction and ground truth.

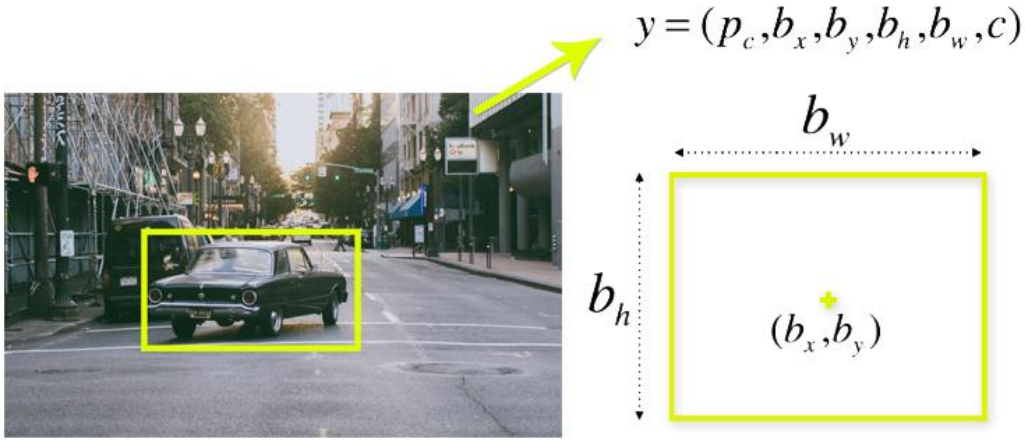




Single-Shot Detection

D. Train class confidence (jointly with step C)

Based on convolutional features for each box with an object, a class confidence can be learned.



Yolo example: class confidence is combination of c and pc

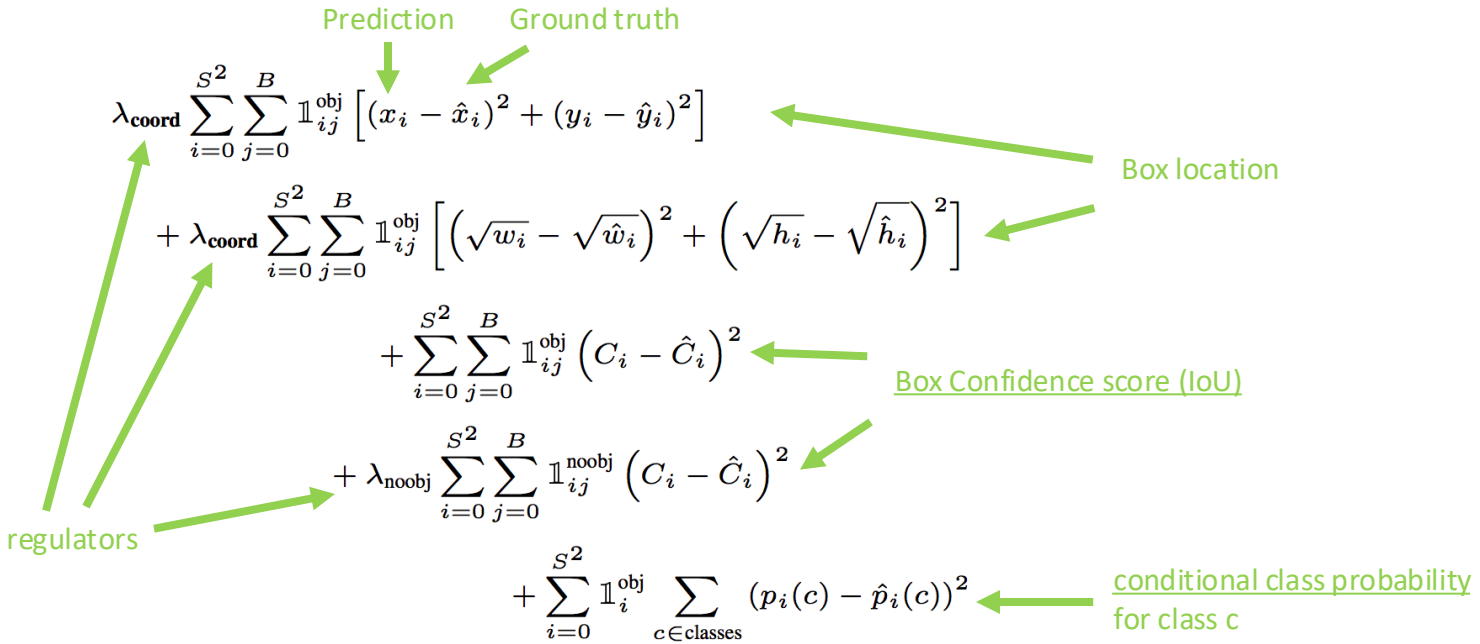
b_x, b_y, b_h, b_w box coordinates

c	box probability that there is an object in box
p_c	conditional class probability for each class c

Single-Shot Detection

D. Train class confidence (jointly with step C)

Location and class probability are trained simultaneously.
Loss function (regression) combined both.



Single-Shot Detection

E. Filter best predictions

Non-maximum suppression filters out predictions with low class confidence. Filter out boxes with confidence below threshold.

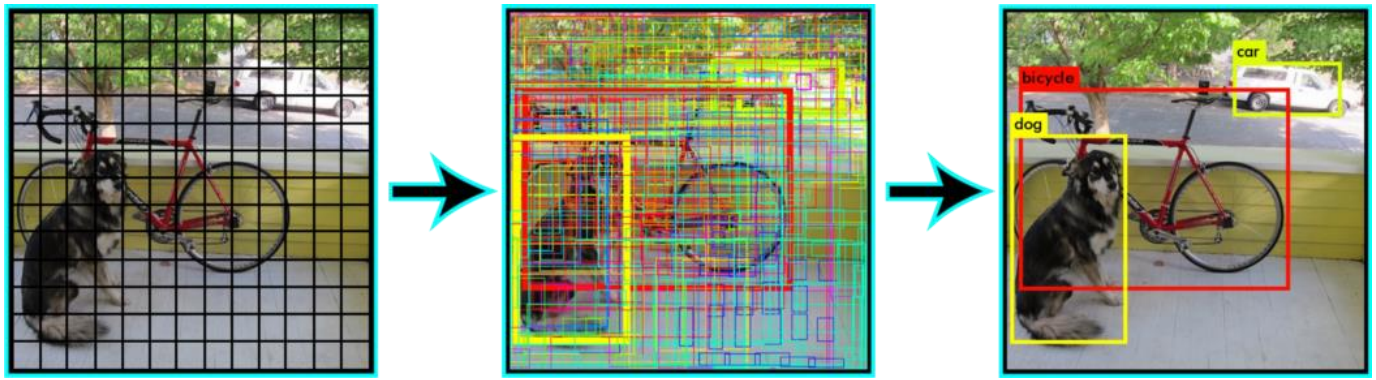


Image grid
S x S
S = 13

Prediction
Every grid cell B boxes
B = 5

So:
13 x 13 x 5 = 845 boxes

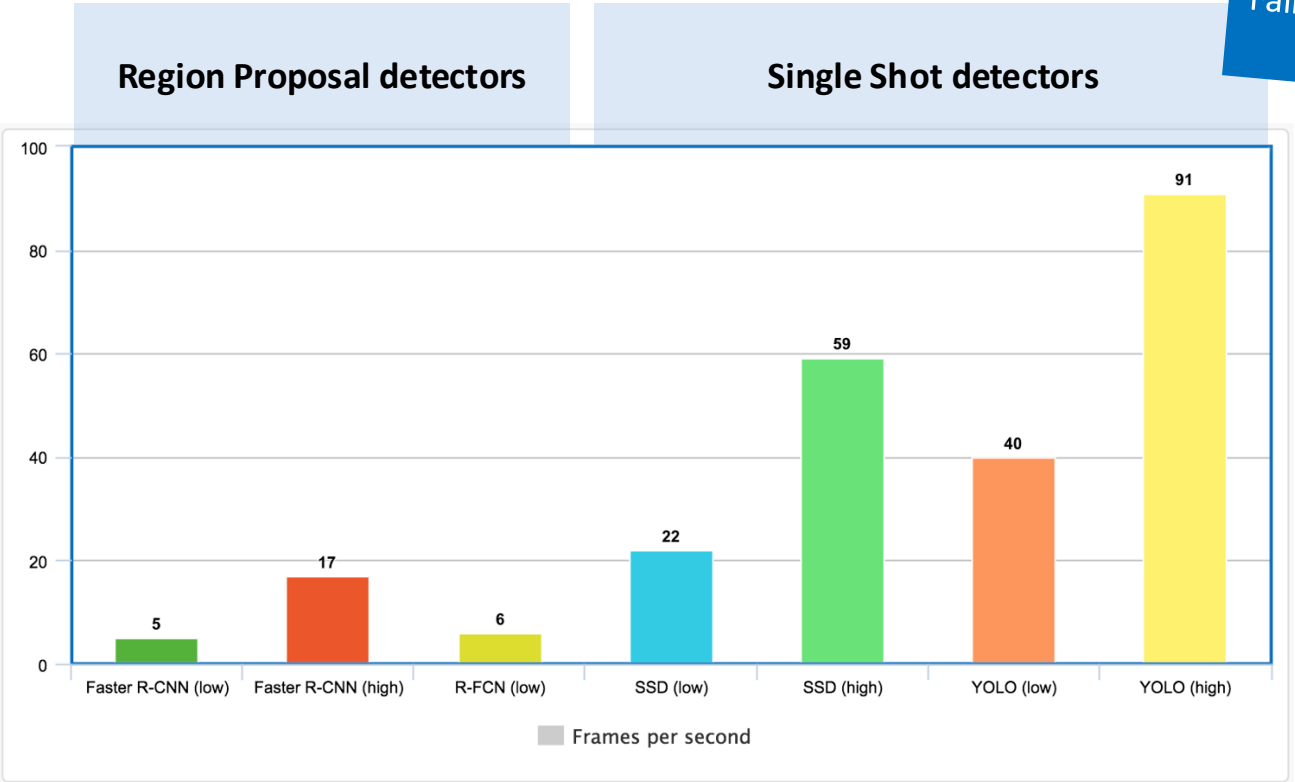
Thresholding:
Box confidence
score > 0.3

Non-Maximum suppression
Class predictions with highest class confidence score

Single-Shot Detection

SSD / YOLO vs. R-CNNs

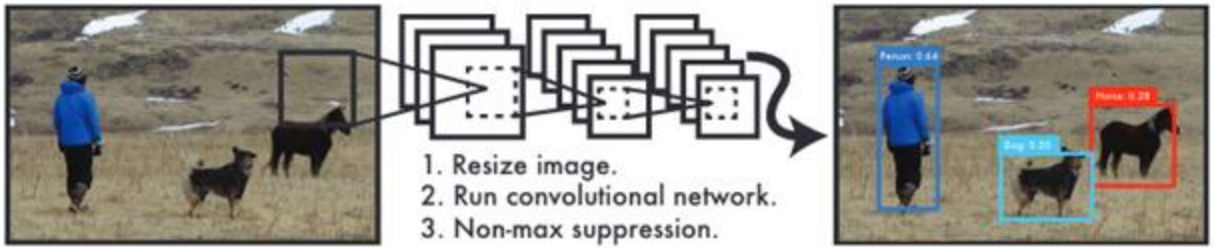
Warning:
Fair comparison
is difficult



[Source: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359]

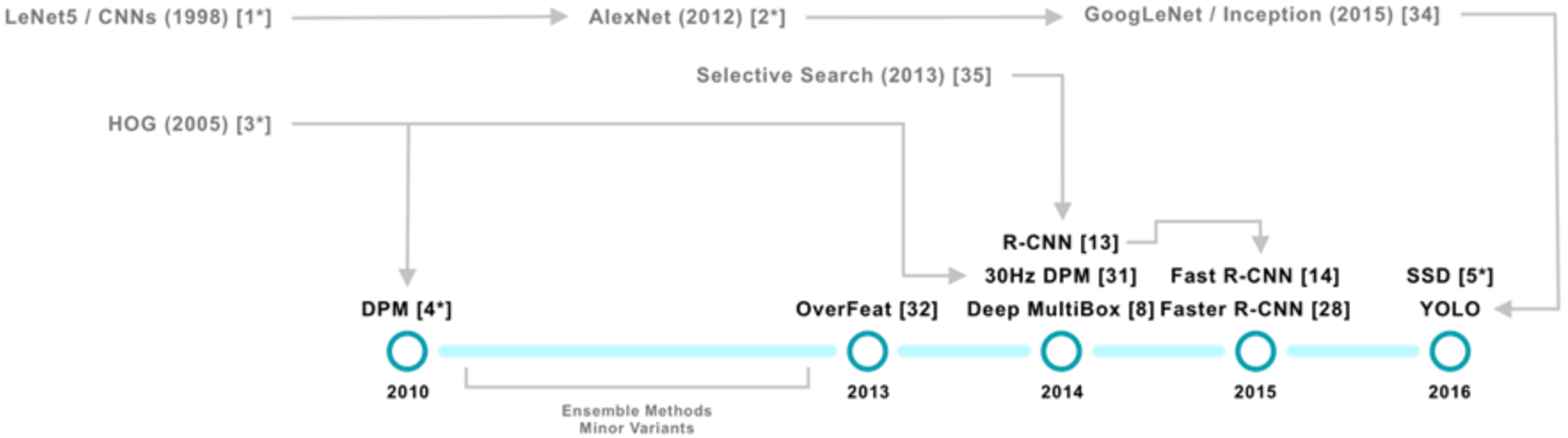
You Only Look Once (YOLO)

- Simple architecture
- Single regression problem, from pixels to bounding box coordinates and class probabilities: **simpler to optimize**
- **FAST:** 45 FPS for YOLO and 150 FPS for YOLO-Fast
- Sees the global context, not just patches



[Redmon et al. CVPR 2016]

You Only Look Once (YOLO)

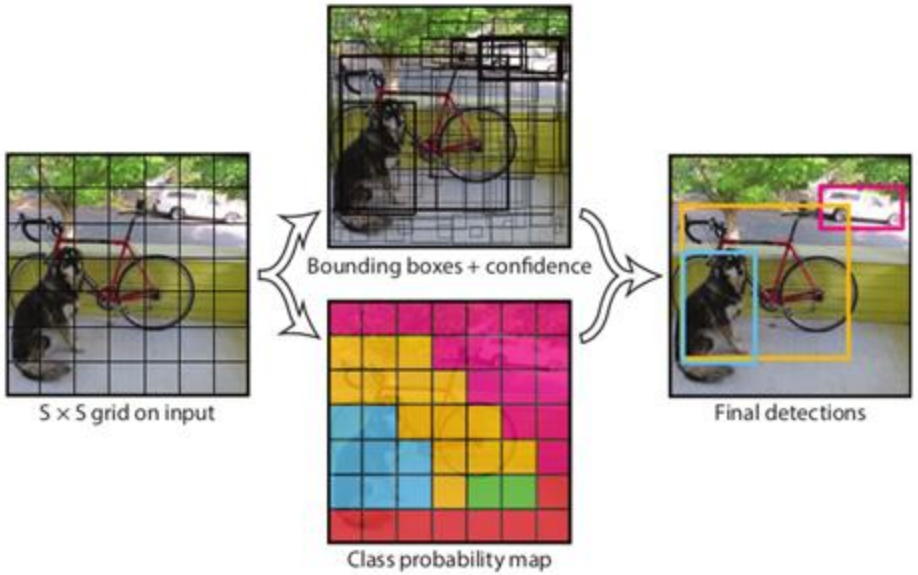


Developed and released a novel object detection approach that:

- Is 2.5 times faster than competing methods and 6.5 times faster than SoTA while sacrificing ~10% mAP
- Only consists of a single neural network which is easy to train
- Considers the global context

[Redmon et al. CVPR 2016]

YOLO – Unified Detection



- Divide image into **S x S** grid.
- Each cell predicts:
  - **B** bounding boxes and confidence scores (*Localization*)
  - **C** conditional class probabilities (*Classification*).
    for the object whose center it roughly contains.

[Redmon et al. CVPR 2016]

YOLO - Localization

e.g. for B=2

ground truth:
(x,y,w,h)
predicted:
(x1,y1,w1,h1,c1)
(x2,y2,w2,h2,c2)

B bounding boxes and confidence scores.

(x,y,w,h,c) → B x 5 outputs

- **x,y**: center of the b. box
- **w,h**: dimensions of b. box
- **c**: confidence

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

[Redmon et al. CVPR 2016]

YOLO - Localization

e.g. for B=2

ground truth:
(x,y,w,h)
predicted:
~~(x1,y1,w1,h1,c1)~~
(x2,y2,w2,h2,c2)

Only keep bbox with highest IoU:

B bounding boxes and confidence scores.

(x,y,w,h,c) → B x 5 outputs

- **x,y**: center of the b. box
- **w,h**: dimensions of b. box
- **c**: confidence

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

[Redmon et al. CVPR 2016]

YOLO - Localization

e.g. for B=2

ground truth:

-

predicted:

(x1,y1,w1,h1,**0**)

Ideally:

confidence = $\begin{cases} \text{conf=0} & \text{if object present} \\ \text{conf=IoU} & \text{if no object present} \end{cases}$

B bounding boxes and confidence scores.

(x,y,w,h,c) → B x 5 outputs

- **x,y**: center of the b. box
- **w,h**: dimensions of b. box
- **c**: confidence

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

[Redmon et al. CVPR 2016]

YOLO - Classification

$(p_{dog}p_{bike},p_{car})$ = (0.1,0.2,0.8)

C conditional class probabilities.

$(p_1,\ldots,p_C)$ → C outputs

- p_i:

$$\Pr(\text{Class}_i|\text{Object})$$

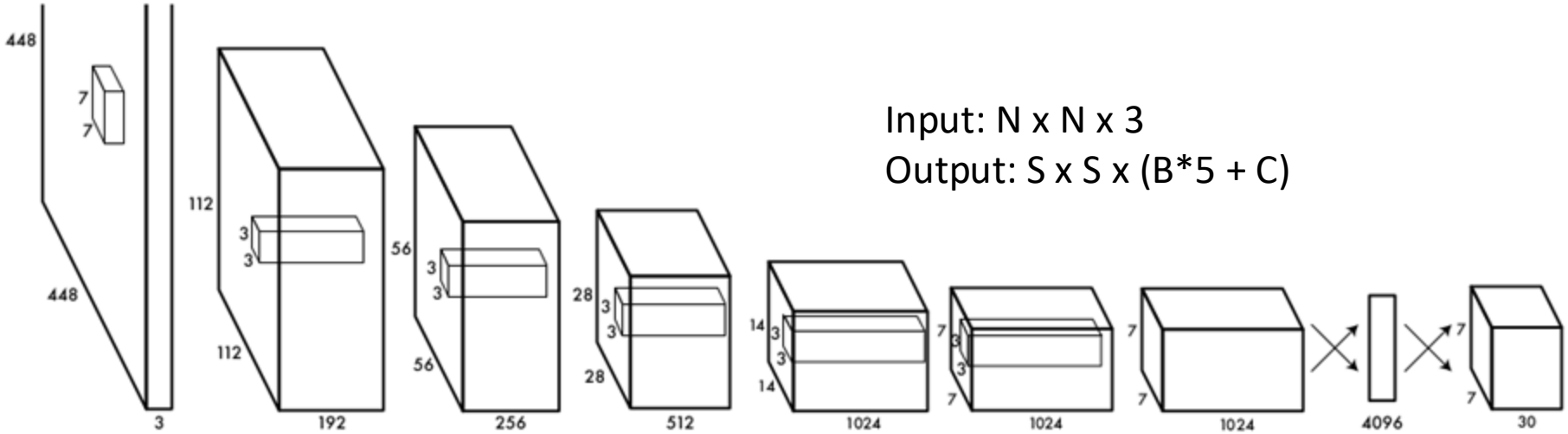
[Redmon et al. CVPR 2016]

YOLO - Confidence Score

$$\underbrace{\Pr(\text{Class}_i|\text{Object})}_{\text{Classification output}} * \underbrace{\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}}_{\text{Localization output}} = \underbrace{\Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}}_{\text{Class-specific confidence score}}$$

[Redmon et al. CVPR 2016]

YOLO - Architecture



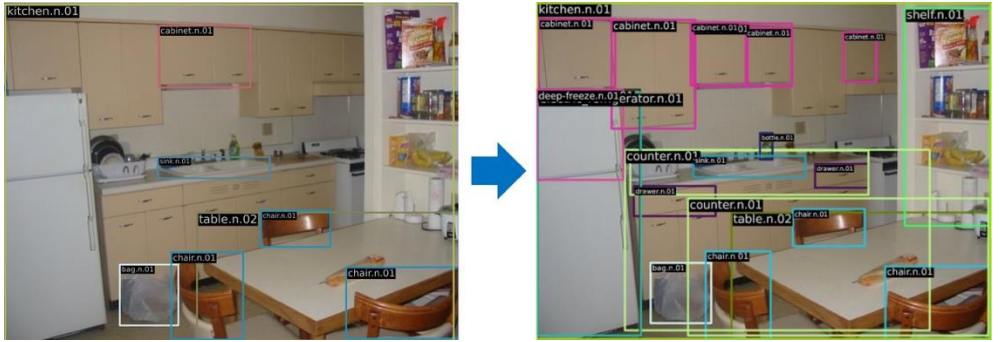
[Redmon et al. CVPR 2016]

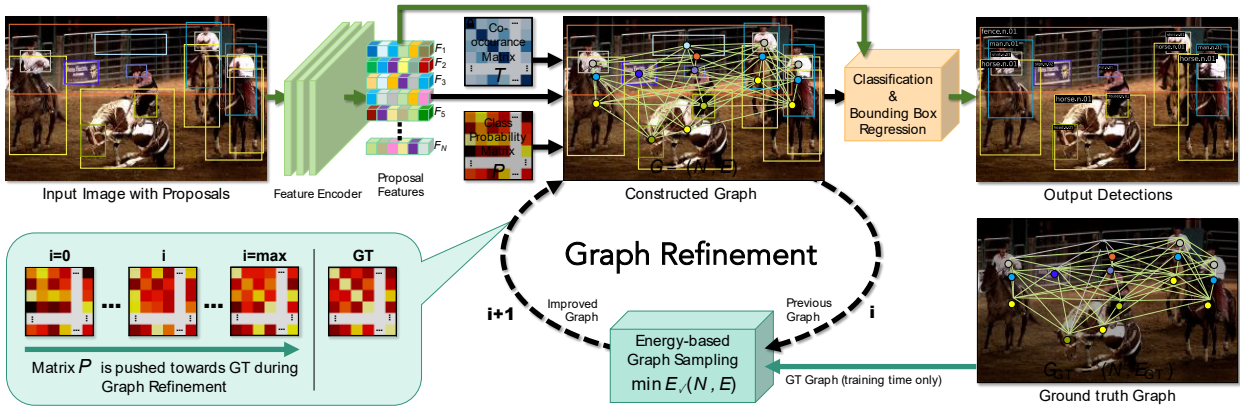
YOLO – Architecture Timeline

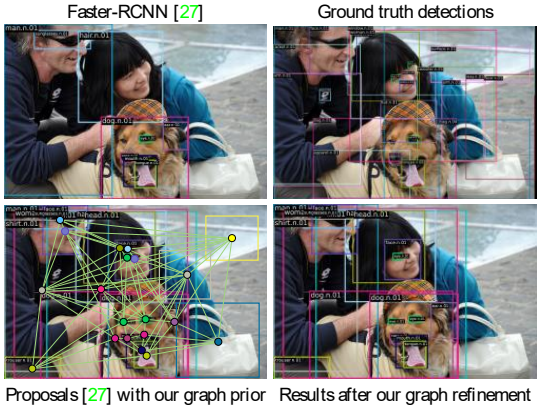
Recent Work: Object Detection @UvA

Key ideas:

- Leverage context information about surrounding objects
- Context encoded via graph network of object proposals
- Learn & use co-occurance statistics







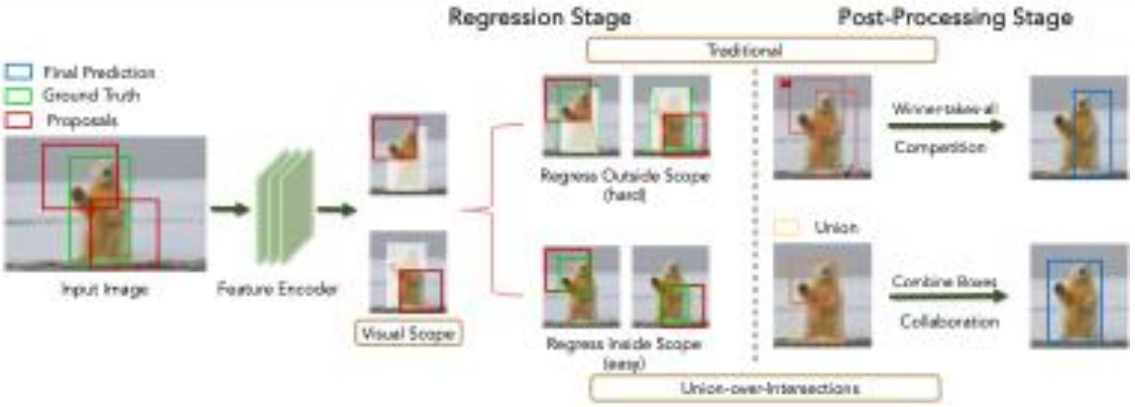
Recent Work: Object Detection @UvA

Key ideas:

- Regress bbox size only within scope
- Combine resized bboxes for final result rather than selecting only one with NMS

Consistent improvements for various two stage detectors:

Method	Backbone	mAP↑	AP_{50}↑	AP_{75}↑	AP_S↑	AP_M↑	AP_L↑
Faster R-CNN (Ren et al., 2015)	R-50-fpn	37.4	58.1	40.4	21.2	41.0	48.1
Faster R-CNN w/ ***UoI***	R-50-fpn	**38.1**	**58.7**	**40.9**	**21.7**	**41.8**	**49.5**
Faster R-CNN (Ren et al., 2015)	R-101-fpn	39.4	60.1	43.1	22.4	43.7	51.1
Faster R-CNN w/ ***UoI***	R-101-fpn	**40.3**	**60.8**	**43.5**	**23.1**	**44.5**	**52.8**
Mask R-CNN (He et al., 2017)	R-50-fpn	38.2	58.8	41.4	21.9	40.9	49.5
Mask R-CNN w/ ***UoI***	R-50-fpn	**38.8**	**59.6**	**41.8**	**22.2**	**41.6**	**50.9**
Mask R-CNN (He et al., 2017)	R-101-fpn	39.8	60.3	43.4	23.1	43.8	52.5
Mask R-CNN w/ ***UoI***	R-101-fpn	**40.9**	**61.1**	**44.0**	**23.5**	**44.7**	**54.6**
Cascade R-CNN (Cai & Vasconcelos, 2018)	R-101-fpn	42.5	60.7	46.4	23.5	46.5	56.4
Cascade R-CNN w/ ***UoI***	R-101-fpn	**43.1**	**61.9**	**46.8**	**24.0**	**47.2**	**57.3**
YOLOv3 (Redmon & Farhadi, 2018)	DarkNet-53	33.7	56.6	35.3	19.4	36.8	44.3
YOLOv3 w/ ***UoI***	DarkNet-53	**34.5**	**57.5**	**35.9**	**19.9**	**37.3**	**45.2**
Def-DETR (Zhu et al., 2021)	R-50-fpn	44.3	63.2	48.6	26.8	47.7	58.8
Def-DETR w/ ***UoI***	R-50-fpn	**44.8**	**63.9**	**49.1**	**27.2**	**48.3**	**59.8**

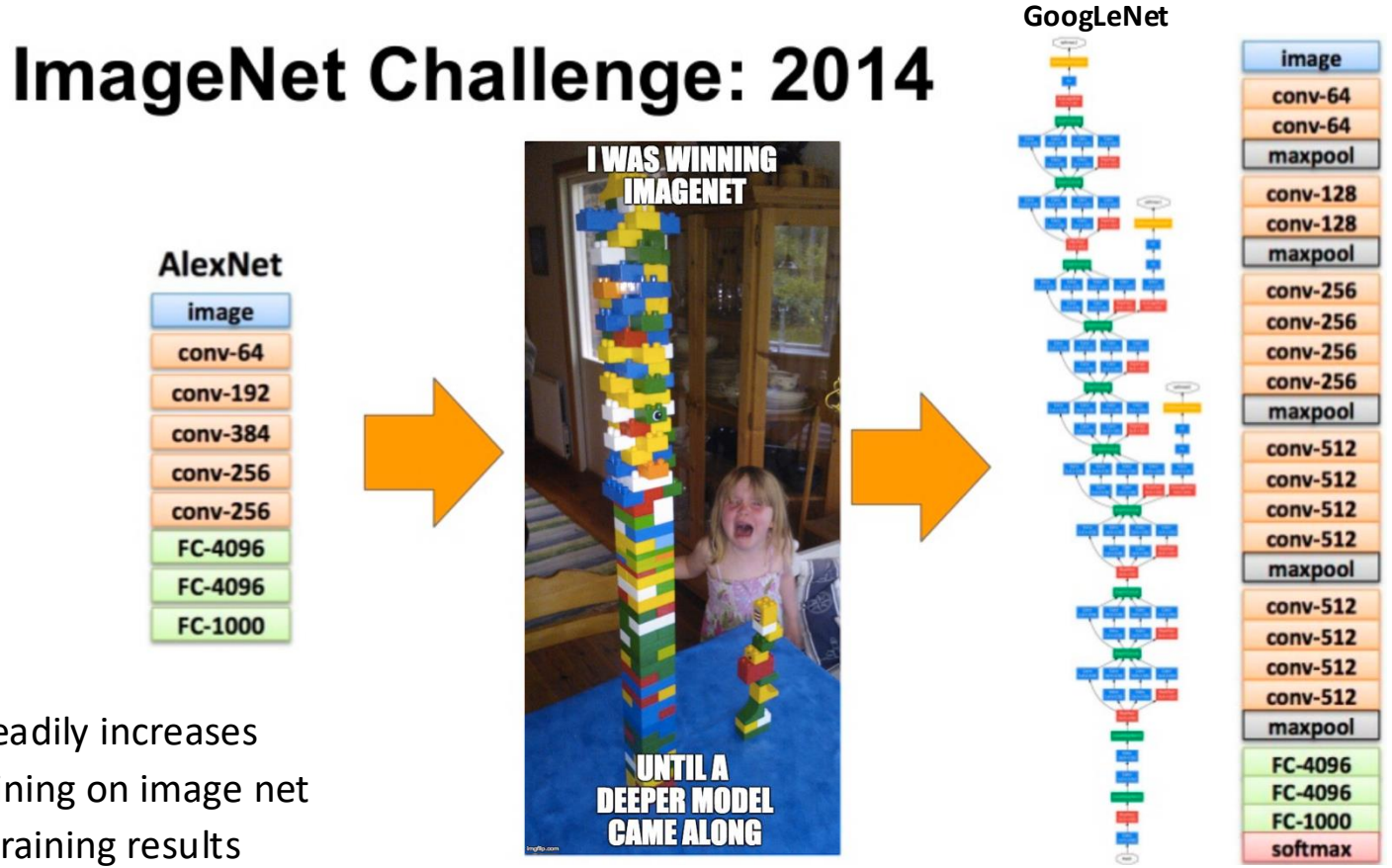


[Bhowmik, Mettes, Oswald, Snoek, "Union-over-intersections: Object detection beyond winner-takes-all", ICLR 2025]

Today's Agenda

- Learning region proposals - Faster R-CNN
- Feature Pyramid Network (FPN)
- Single Shot Detectors (SSD) / Yolo
- Transfer learning
- Network architectures
  - VGG
  - ResNet
  - U-Net
  - Autoencoders

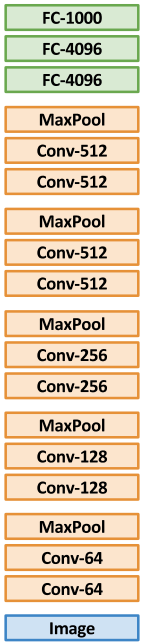
Classification Model Size



- Model size steadily increases
- Expensive training on image net
- Idea: Re-use training results

Transfer Learning

1. Train on ImageNet

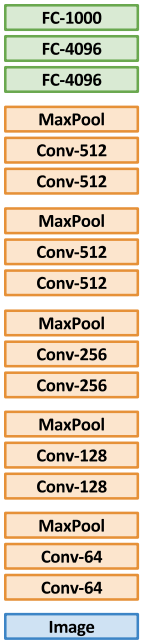


[Slide credit: Justin Johnson]

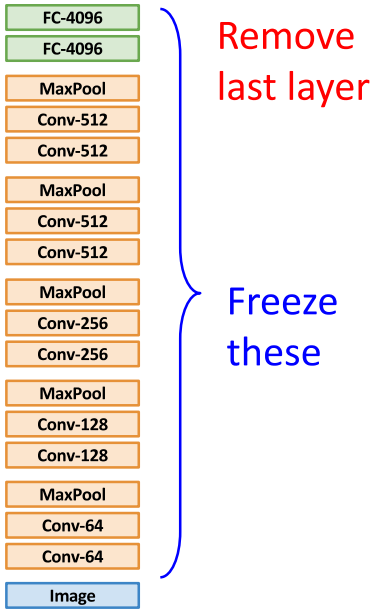
Transfer Learning

Transfer Learning: Feature Extraction

1. Train on ImageNet



2. Extract features with CNN, train linear model



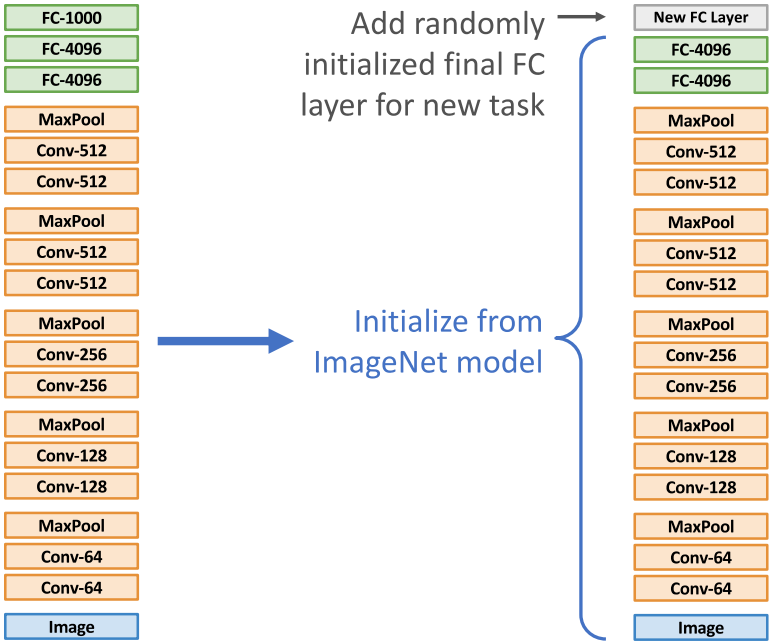
Remove last layer

Freeze these

[Slide credit: Justin Johnson]

Transfer Learning: Fine-Tuning

1. Train on ImageNet

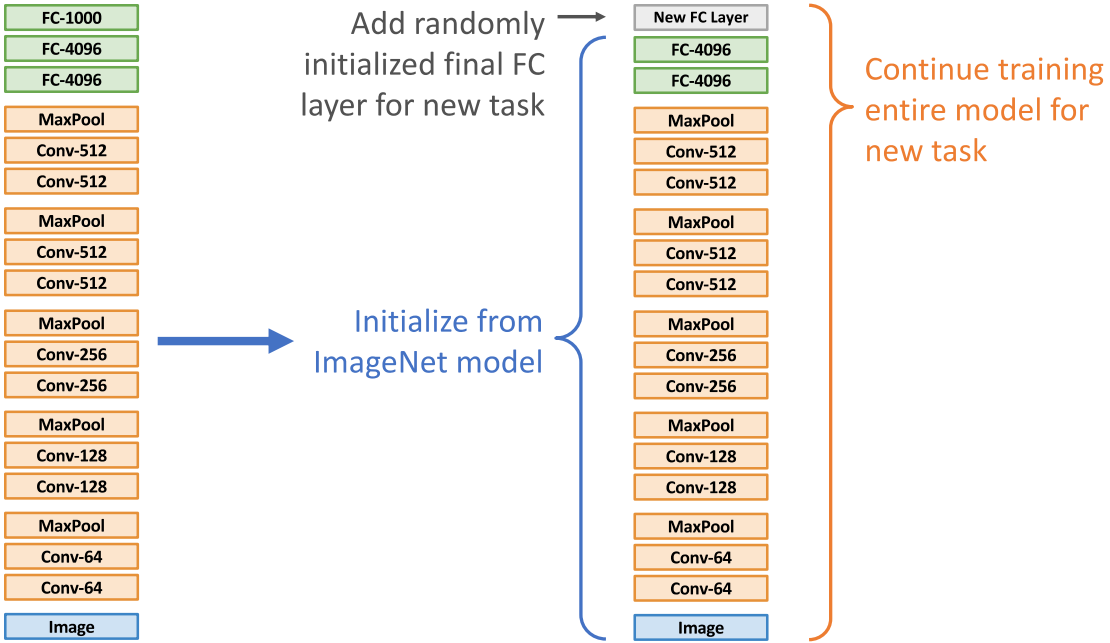


[Slide credit: Justin Johnson]

Transfer Learning

Transfer Learning: Fine-Tuning

1. Train on ImageNet

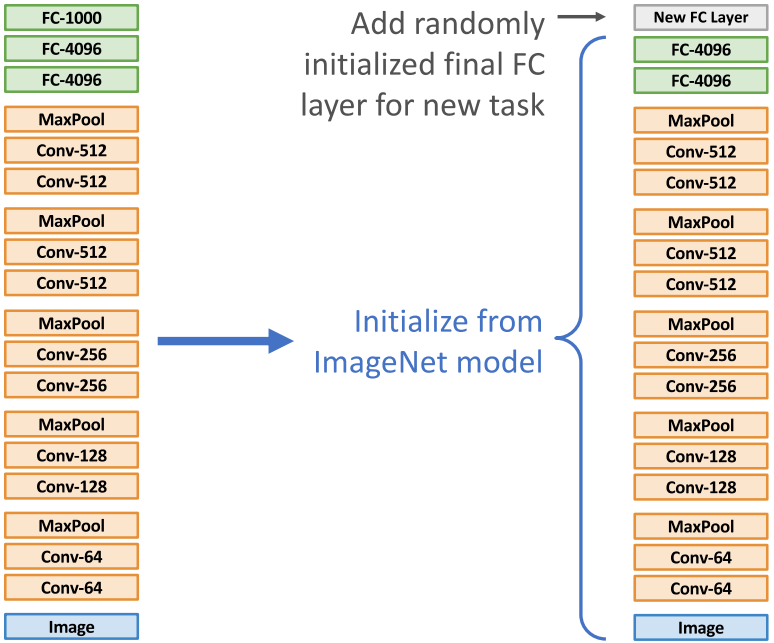


[Slide credit: Justin Johnson]

Transfer Learning

Transfer Learning: Fine-Tuning

1. Train on ImageNet



Some tricks:

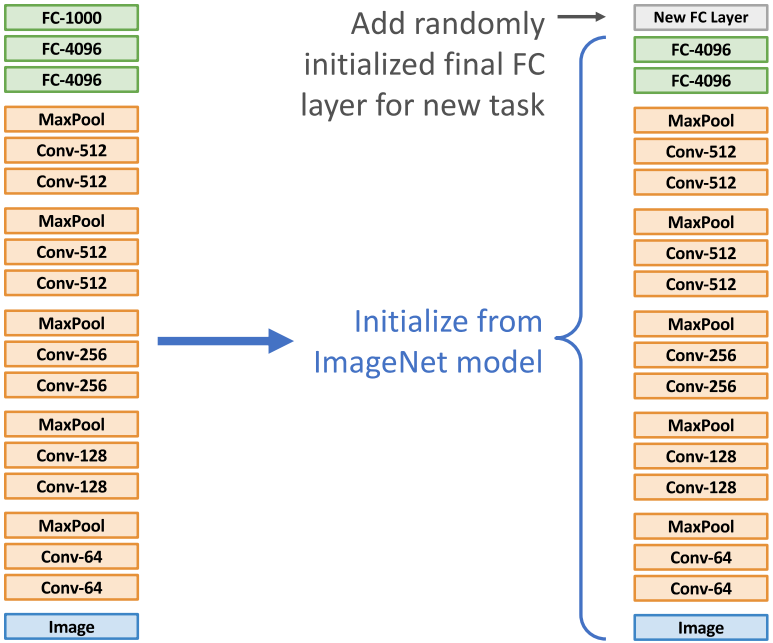
- Train with feature extraction first before fine-tuning
- Lower the learning rate: use ~1/10 of LR used in original training
- Sometimes freeze lower layers to save computation
- Train with BatchNorm in “test” mode

[Slide credit: Justin Johnson]

Transfer Learning

Transfer Learning: Fine-Tuning

1. Train on ImageNet



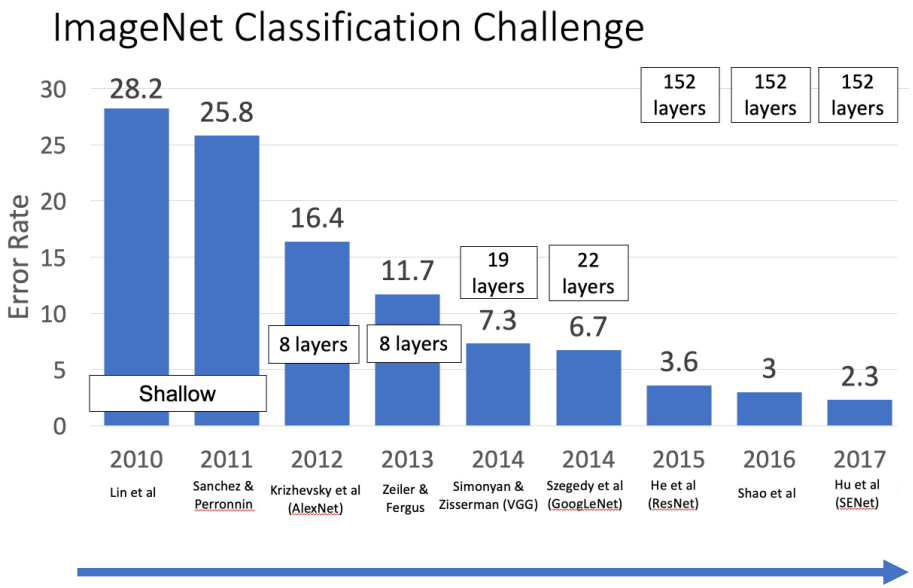
Compared with Feature Extraction, Fine-Tuning:

- Requires more data
- Is more computationally expensive
- Can give higher accuracies

[Slide credit: Justin Johnson]

Transfer Learning

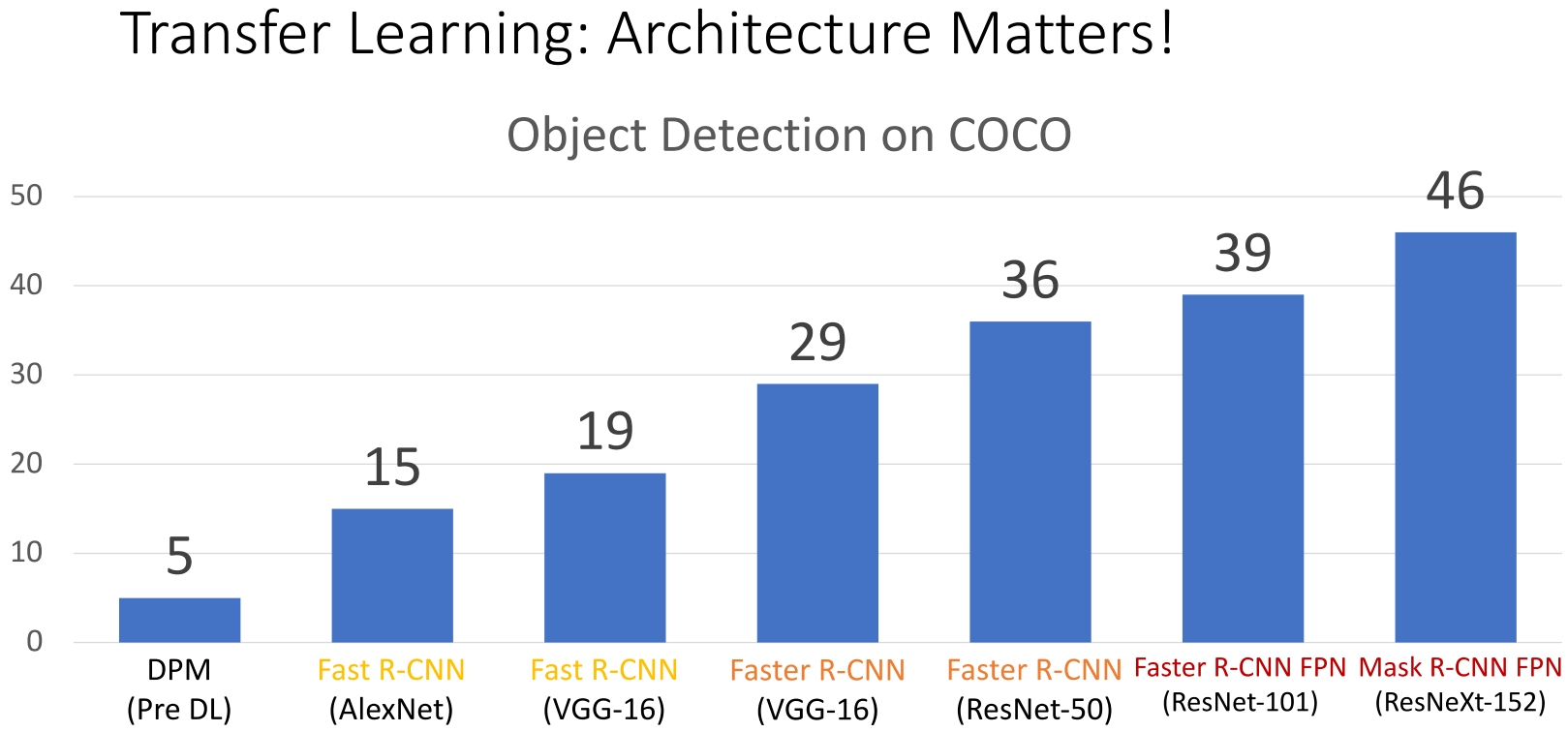
Transfer Learning: Architecture Matters!



Improvements in CNN architectures lead to improvements in many downstream tasks thanks to transfer learning!

[Slide credit: Justin Johnson]

Transfer Learning



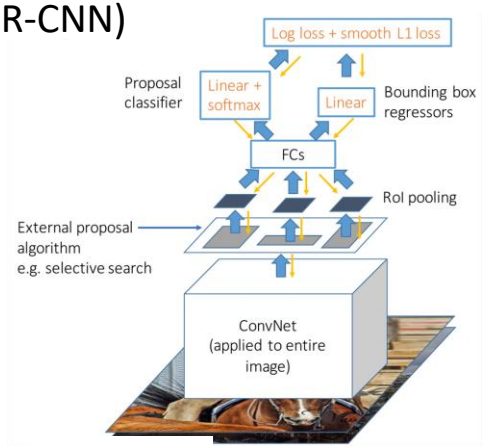
Ross Girshick, "The Generalized R-CNN Framework for Object Detection", ICCV 2017 Tutorial on Instance-Level Visual Recognition

[Slide credit: Justin Johnson]

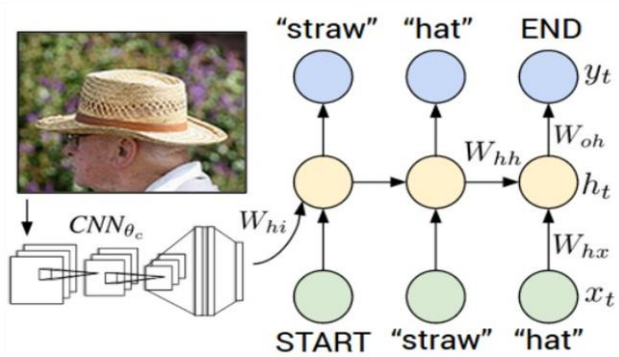
Transfer Learning

Transfer learning is pervasive!
It's the norm, not the exception

Object Detection
(Fast R-CNN)



Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.



Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

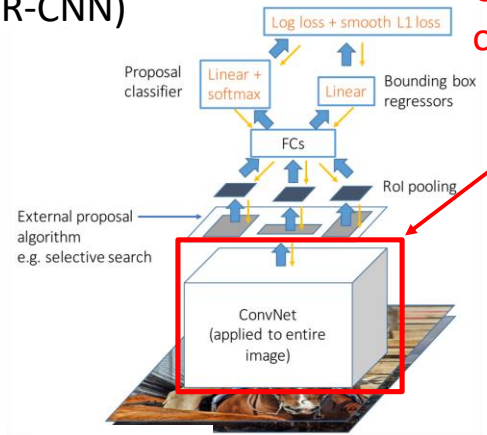
[Slide credit: Justin Johnson]

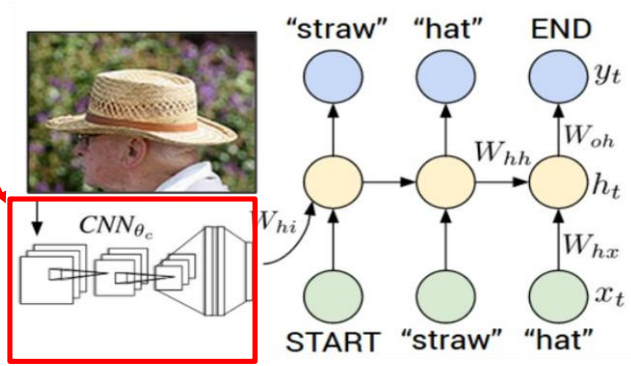
Transfer Learning

Transfer learning is pervasive!
It's the norm, not the exception

Object Detection
(Fast R-CNN)

CNN pretrained
on ImageNet





Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

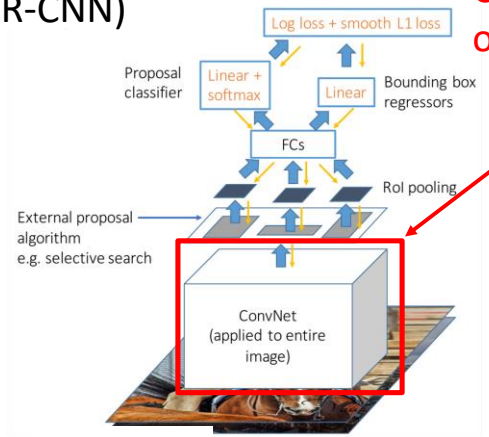
Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

[Slide credit: Justin Johnson]

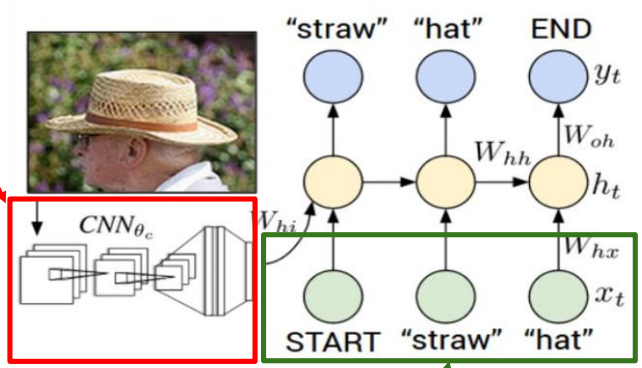
Transfer Learning

Transfer learning is pervasive!
It's the norm, not the exception

Object Detection
(Fast R-CNN)



CNN pretrained
on ImageNet



Word vectors pretrained
with word2vec

Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

[Slide credit: Justin Johnson]

Today's Agenda

- Learning region proposals - Faster R-CNN
- Feature Pyramid Network (FPN)
- Single Shot Detectors (SSD) / Yolo
- Transfer learning
- Network architectures
  - VGG
  - ResNet
  - U-Net
  - Autoencoders

VGG Network (from the Visual Geometry Group Oxford)

VGG: Deeper Networks, Regular Design

VGG Design rules:

All conv are 3x3 stride 1 pad 1

All max pool are 2x2 stride 2

After pool, double #channels

Network has 5 convolutional **stages**:

Stage 1: conv-conv-pool

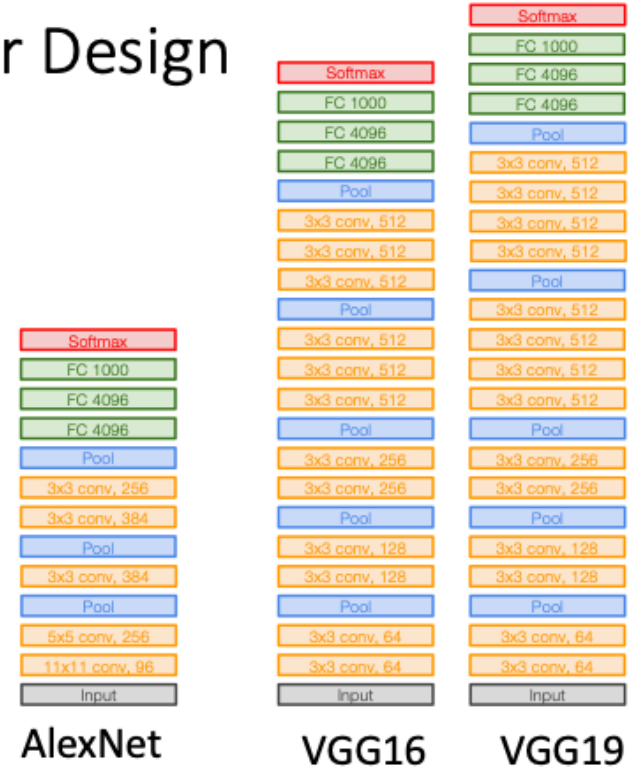
Stage 2: conv-conv-pool

Stage 3: conv-conv-pool

Stage 4: conv-conv-conv-[conv]-pool

Stage 5: conv-conv-conv-[conv]-pool

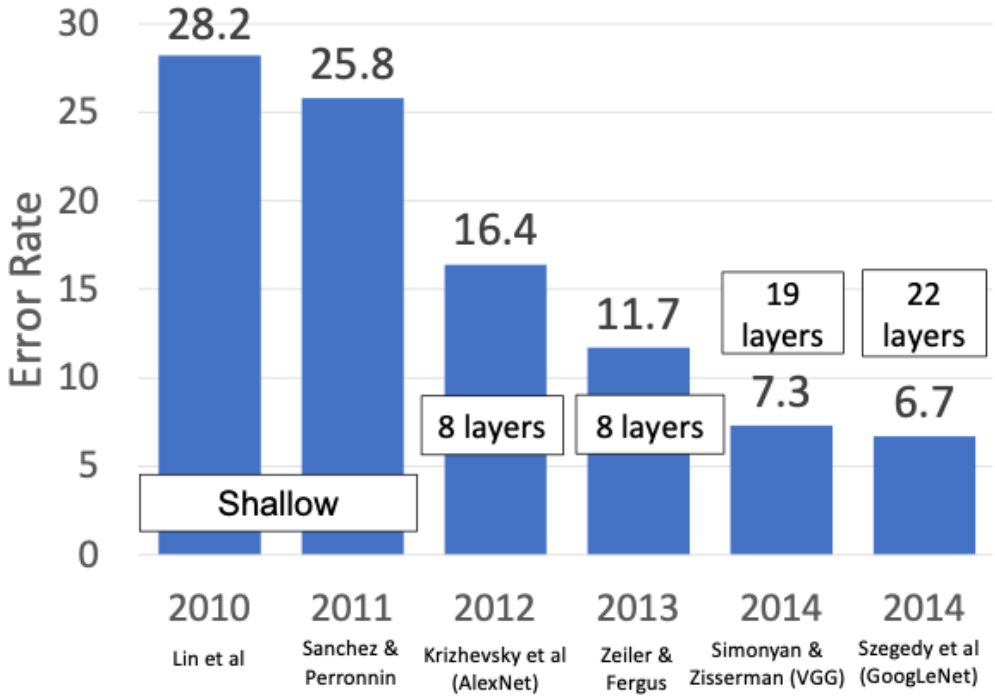
(VGG-19 has 4 conv in stages 4 and 5)



[Slide credit: Justin Johnson]

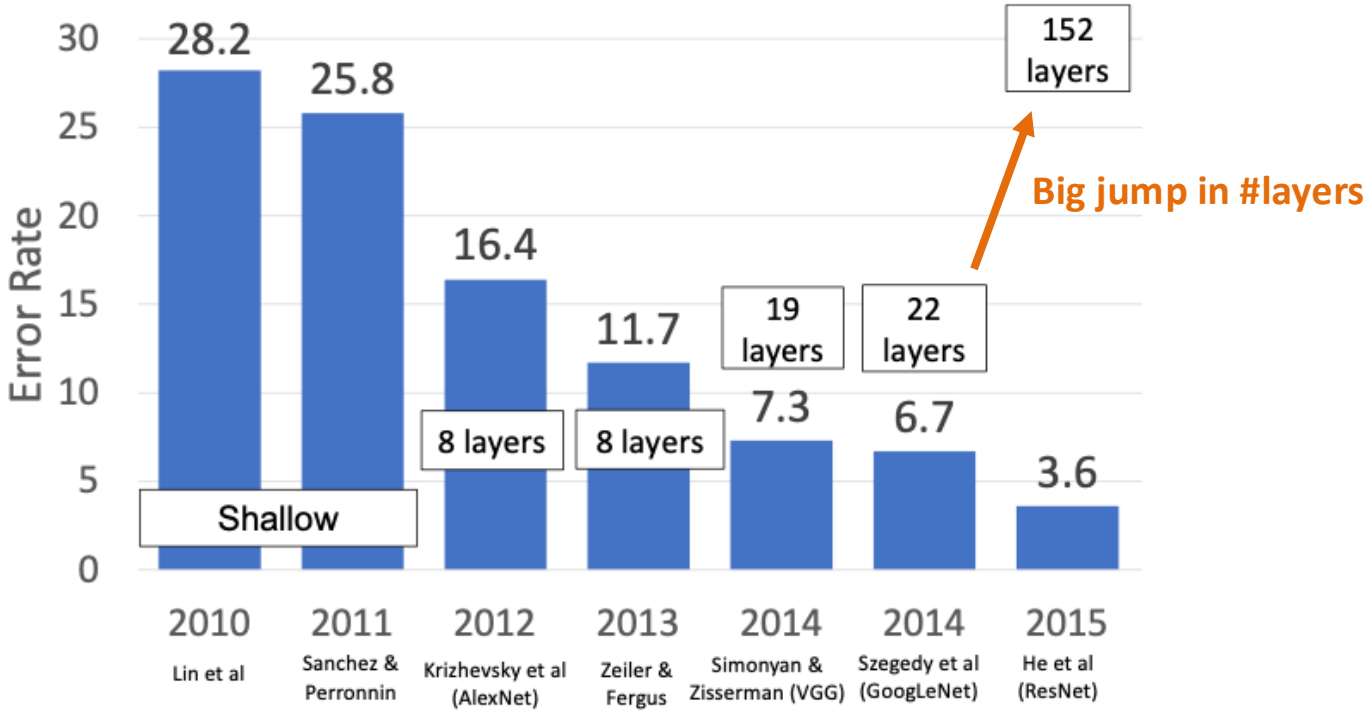
[Simonyan and Zissermann, "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR 2015]

ImageNet Classification Challenge



[Slide credit: Justin Johnson]

ImageNet Classification Challenge



[Slide credit: Justin Johnson]

ResNet (ILSVRC 2015 winner)

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

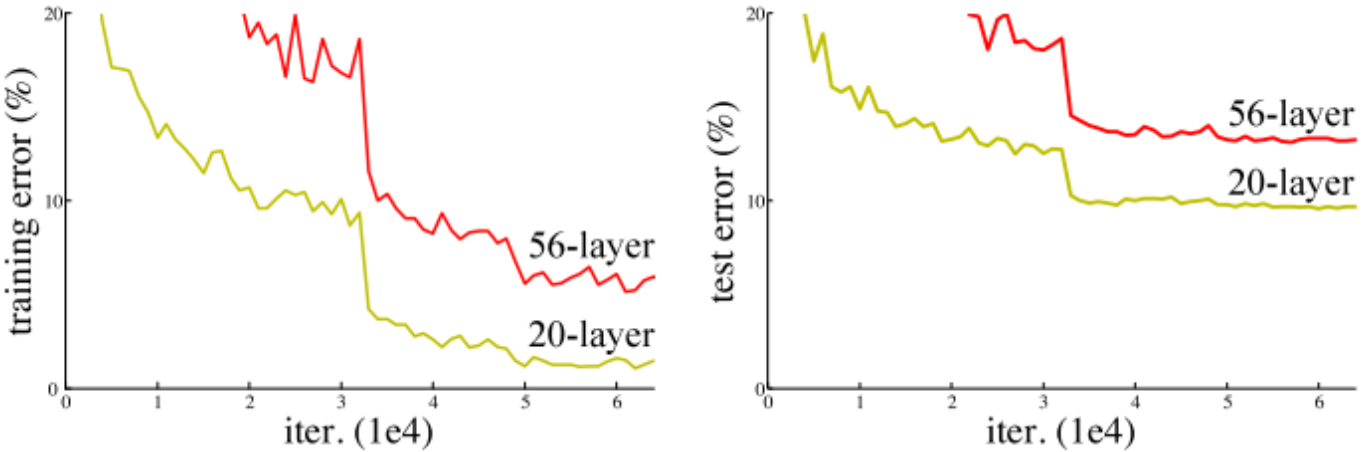
ResNet, 152 layers
(ILSVRC 2015)

[Slide credit: S. Lazebnik]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet – Motivation: Vanishing Gradients

Deeper networks tend to saturate. Problem comes from vanishing gradients.
Deeper networks were underfitting rather than overfitting due to limitations of the optimization.



[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

- A deeper model can emulate a shallower model: copy layers from shallower model, set extra layers to identity.

  Thus deeper models should do at least as good as shallow models.

- **Hypothesis**: This is an optimization problem. Deeper models are harder to optimize, and in particular don't learn identity functions to emulate shallow models.

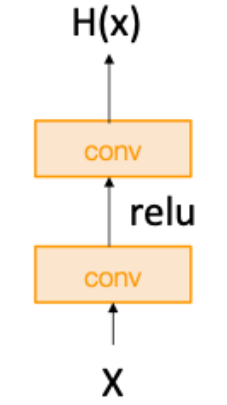
- **Solution**: Change the network so learning identity functions with extra layers is easy!

[Slide credit: Justin Johnson]

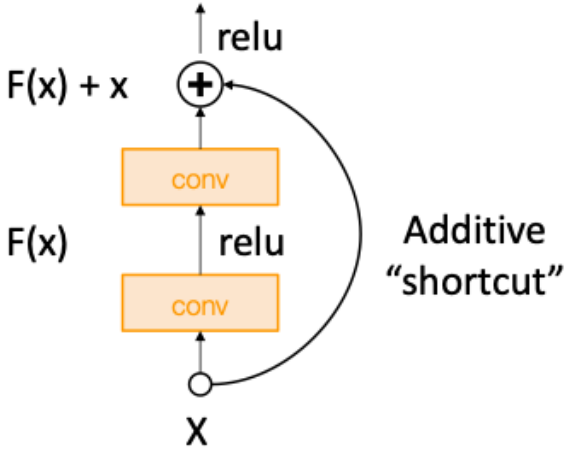
[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

Solution: Change the network so learning identity functions with extra layers is easy!



"Plain" block



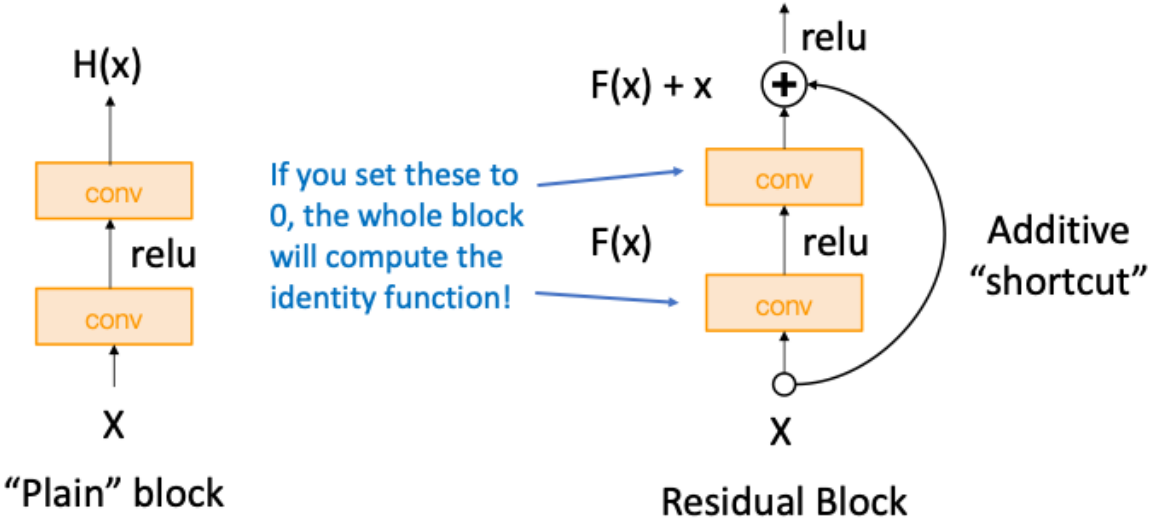
Residual Block

[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

Solution: Change the network so learning identity functions with extra layers is easy!



[Slide credit: Justin Johnson]

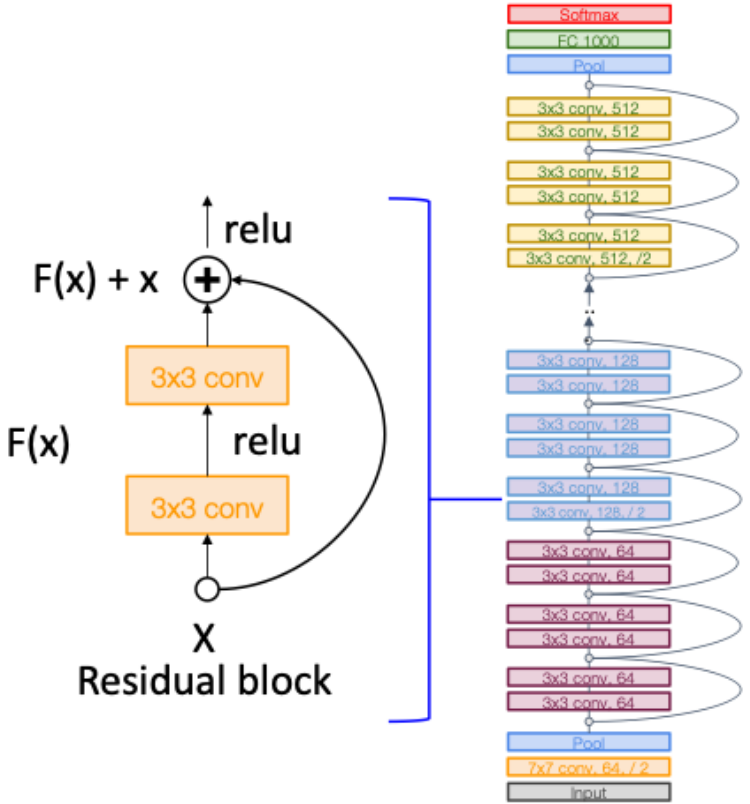
[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

A residual network is a stack of many residual blocks

Regular design, like VGG: each residual block has two 3x3 conv

Network is divided into **stages**: the first block of each stage halves the resolution (with stride-2 conv) and doubles the number of channels



[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

ResNet-18:

Stem: 1 conv layer

Stage 1 (C=64): 2 res. block = 4 conv

Stage 2 (C=128): 2 res. block = 4 conv

Stage 3 (C=256): 2 res. block = 4 conv

Stage 4 (C=512): 2 res. block = 4 conv

Linear

ImageNet top-5 error: 10.92

GFLOP: 1.8

ResNet-34:

Stem: 1 conv layer

Stage 1: 3 res. block = 6 conv

Stage 2: 4 res. block = 8 conv

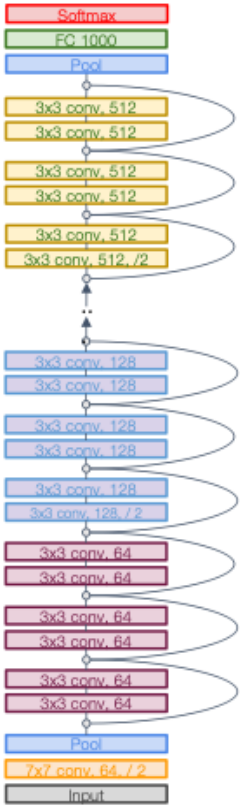
Stage 3: 6 res. block = 12 conv

Stage 4: 3 res. block = 6 conv

Linear

ImageNet top-5 error: 8.58

GFLOP: 3.6

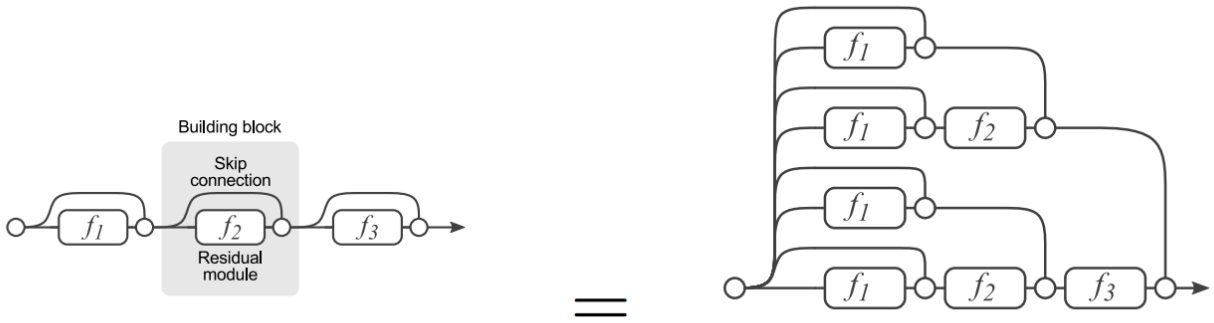


[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

- ResNets are collections of many paths of different length, and shorter paths predominantly contribute to training



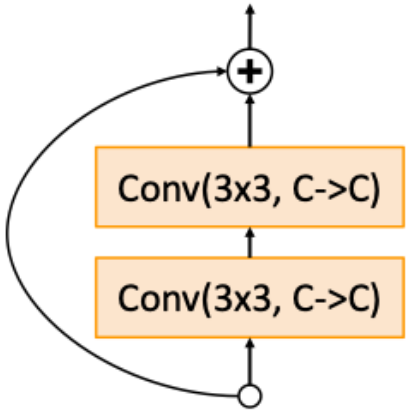
(a) Conventional 3-block residual network (b) Unraveled view of (a)

Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.

[Slide credit: S. Lazebnik]

[A. Veit, M. Wilber, S. Belongie, Residual Networks Behave Like Ensembles of Relatively Shallow Networks, NIPS 2016]

ResNet

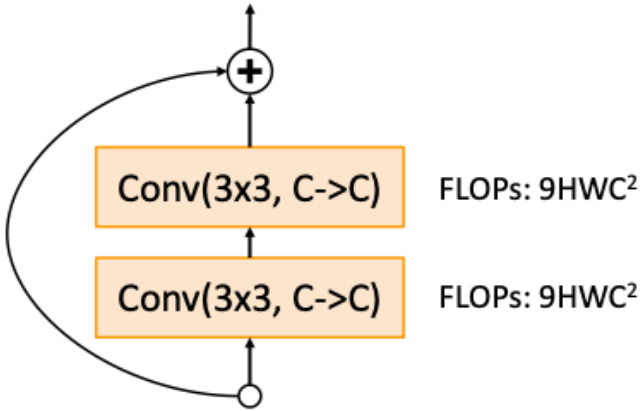


"Basic"
Residual block

[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

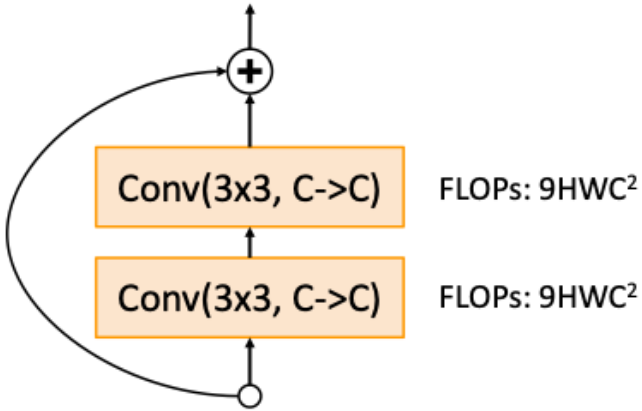
ResNet



[Slide credit: Justin Johnson]

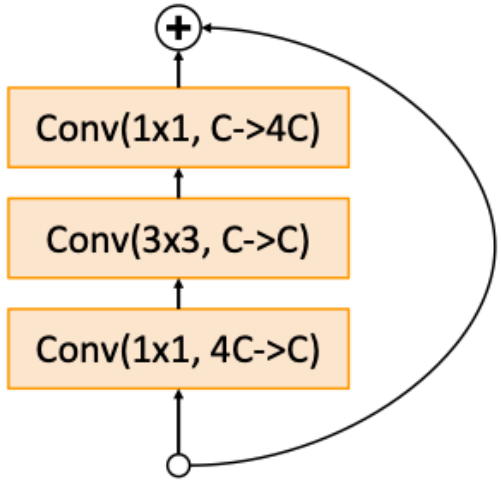
[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet



"Basic"
Residual block

Total FLOPs:
18HWC²



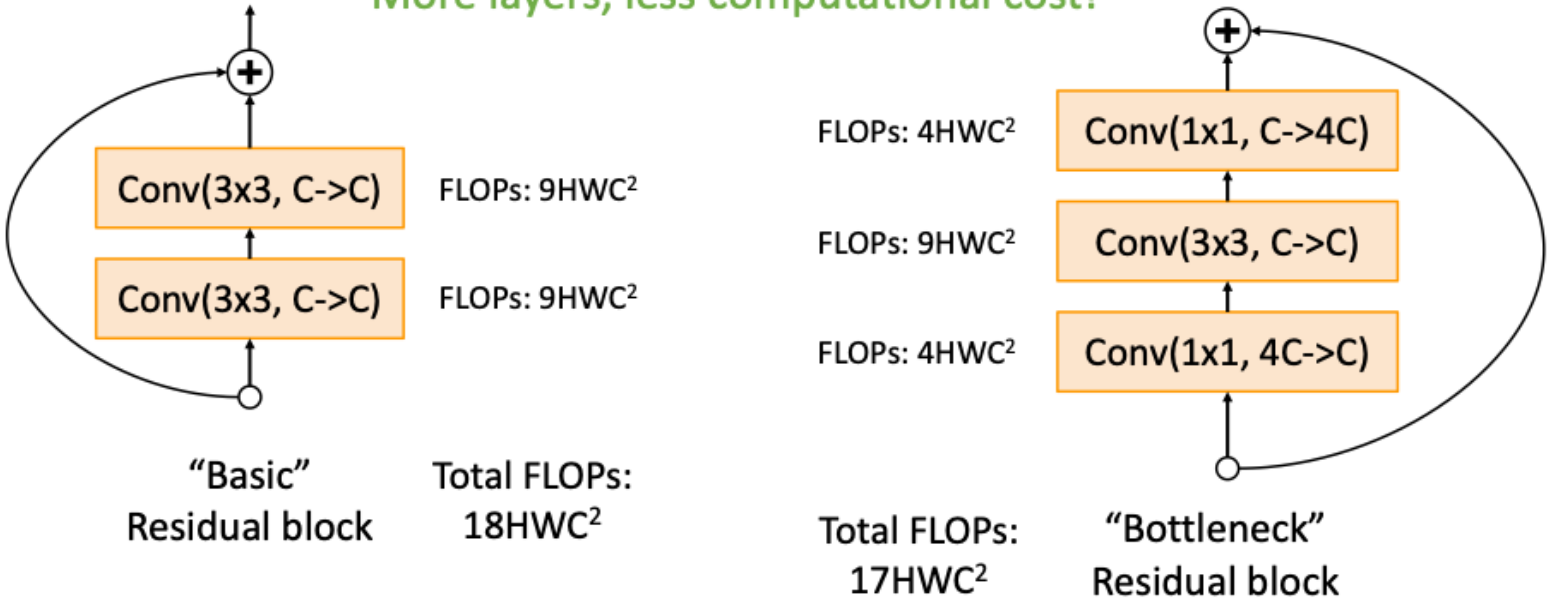
"Bottleneck"
Residual block

[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

More layers, less computational cost!



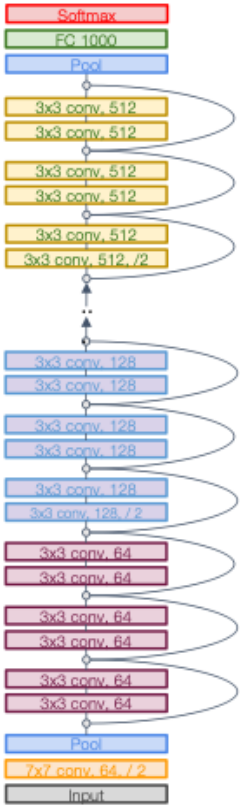
[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

ResNet

Deeper ResNet-101 and ResNet-152 models are more accurate, but also more computationally heavy

			Stage 1		Stage 2		Stage 3		Stage 4				
	Block type	Stem layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	Blocks	Layers	FC layers	GFLOP	ImageNet top-5 error
ResNet-18	Basic	1	2	4	2	4	2	4	2	4	1	1.8	10.92
ResNet-34	Basic	1	3	6	4	8	6	12	3	6	1	3.6	8.58
ResNet-50	Bottle	1	3	9	4	12	6	18	3	9	1	3.8	7.13
ResNet-101	Bottle	1	3	9	4	12	23	69	3	9	1	7.6	6.44
ResNet-152	Bottle	1	3	9	8	24	36	108	3	9	1	11.3	5.94



[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

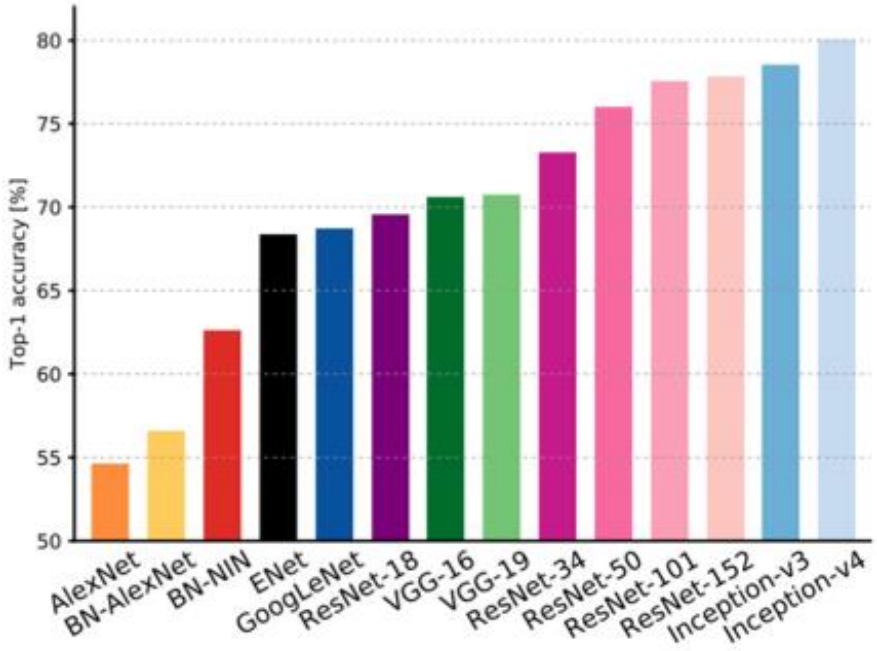
ResNet

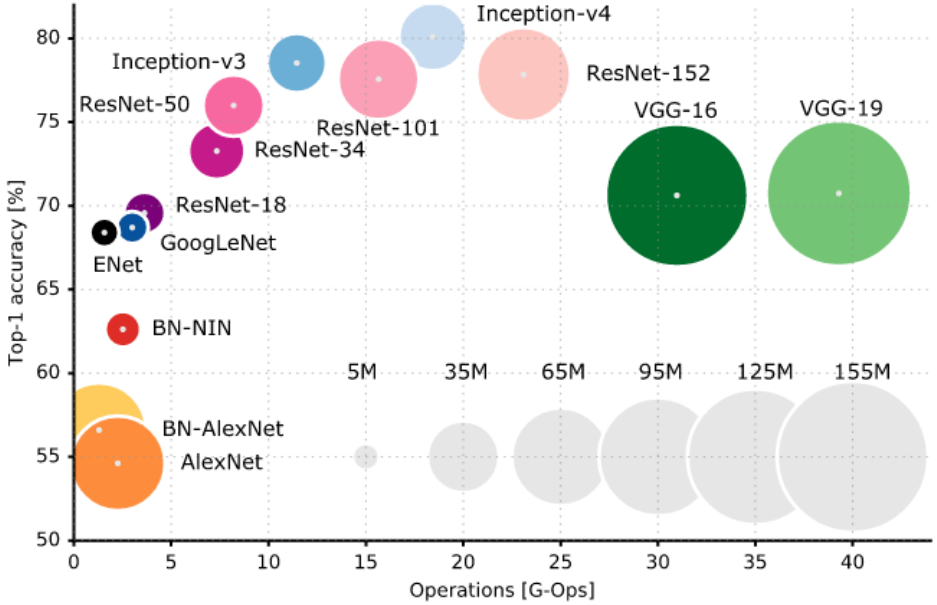
- Able to train very deep networks
- Deeper networks do better than shallow networks (as expected)
- Swept 1st place in all ILSVRC and COCO 2015 competitions
- Still widely used today!

[Slide credit: Justin Johnson]

[K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)]

Comparing Architectures





[Slide credit: S. Lazebnik]

[https://culurciello.github.io/tech/2016/06/04/nets.html]

Encoder-Decoder Architecture

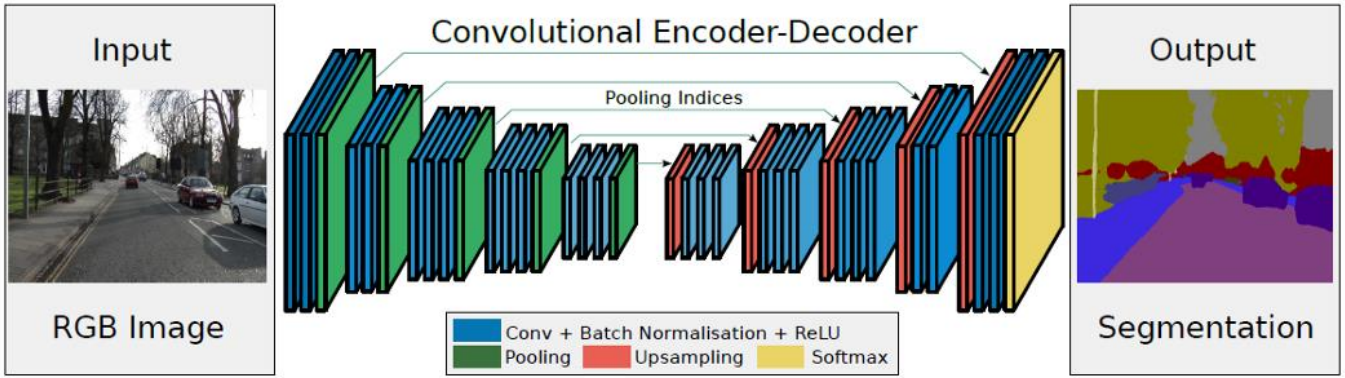
Two common use-cases:

- Pixel wise predictions - output is of same dimension of input
  - semantic segmentation (e.g. Segnet, U-Net)

- Auto-encoder for
  - unsupervised feature learning
  - unsupervised pre-training

Encoder-Decoder Architecture - SegNet

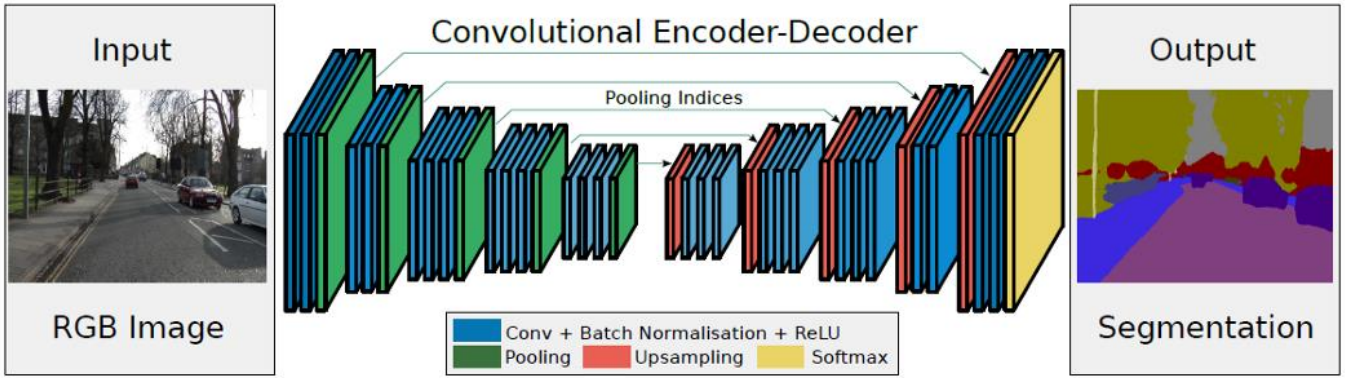
Segnet uses an encoder-decoder architecture for segmantic segmentation.



[Badrinarayanan et al., Segnet]

Encoder-Decoder Architecture - SegNet

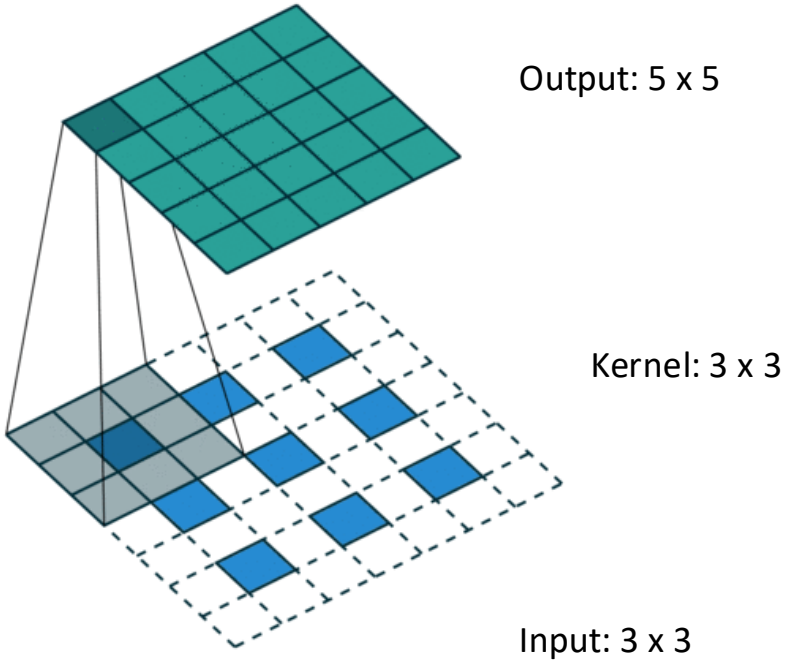
- Encoder is ‘normal’ convolution + pooling.
- Decoder uses upsampling + convolution
- (Learned) convolution is to refine the upsampling



[Badrinarayanan et al., Segnet]

SegNet: Transposed Convolution

- upsampling: unpooling
- convolution

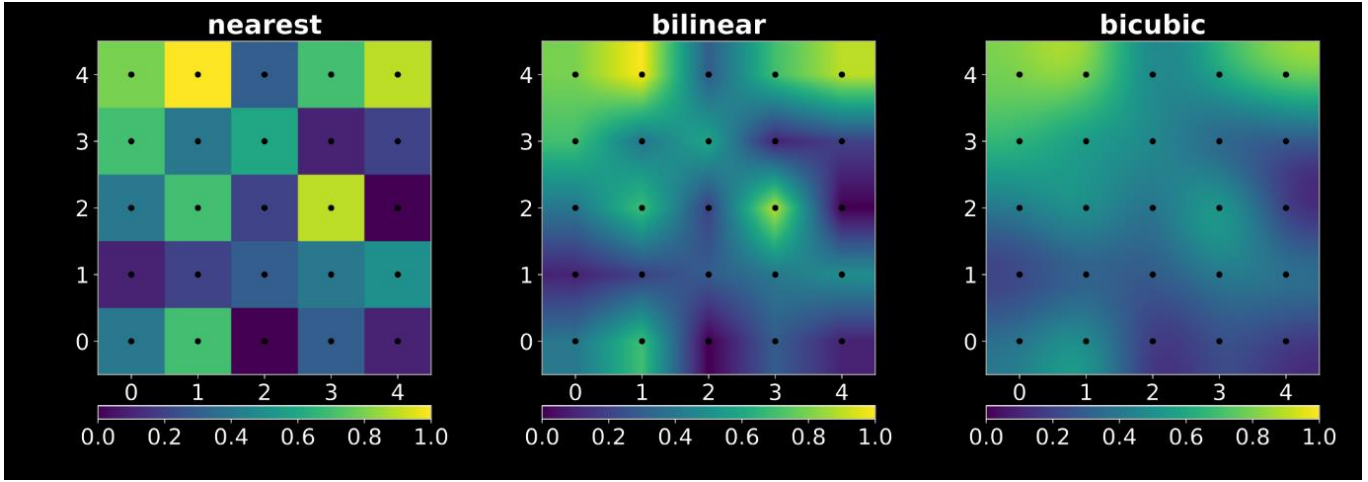


also: up-convolution,

not ~~deconvolution~~

Upsampling Methods

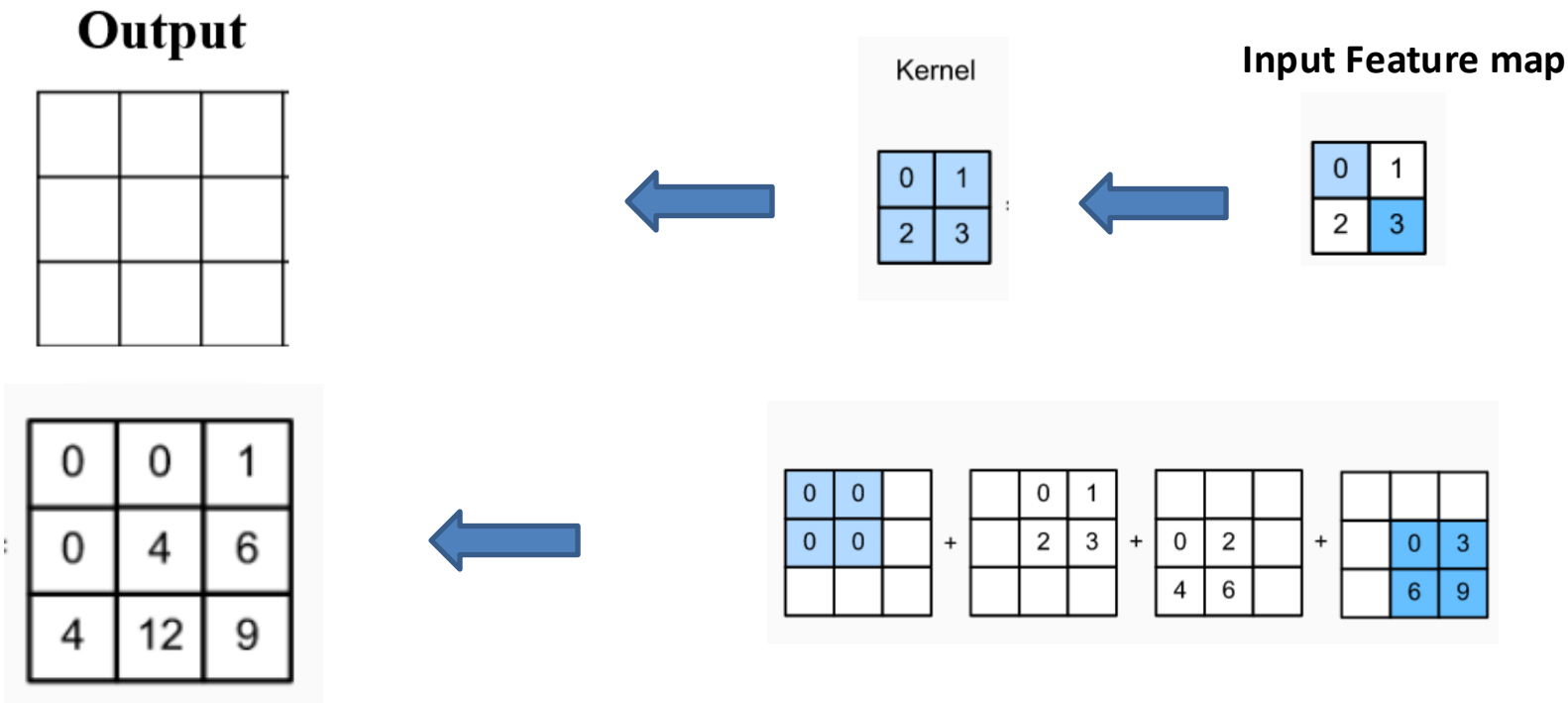
Recall: as alternative for convolution, different interpolation methods exist.



[Source: https://artoriuz.github.io/blog/super_resolution.html]

Upsampling Methods

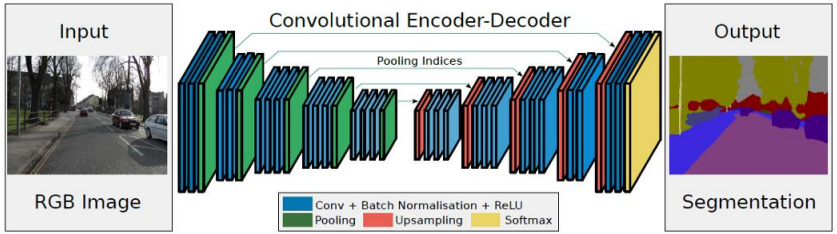
Use learned network – kernels and feature map to ‘deconvolve’ to original pixel space.

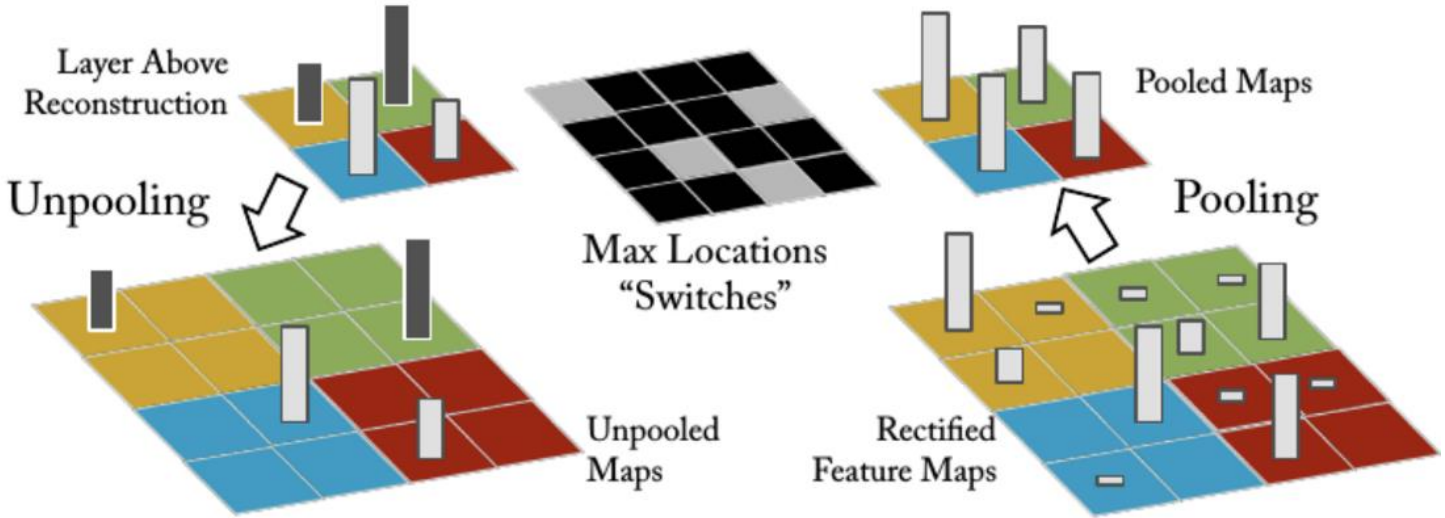


[Source: https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba]

Segnet: Alternative Unpooling operator

The unpooling uses the max-locations ('switches') that have been stored during pooling.



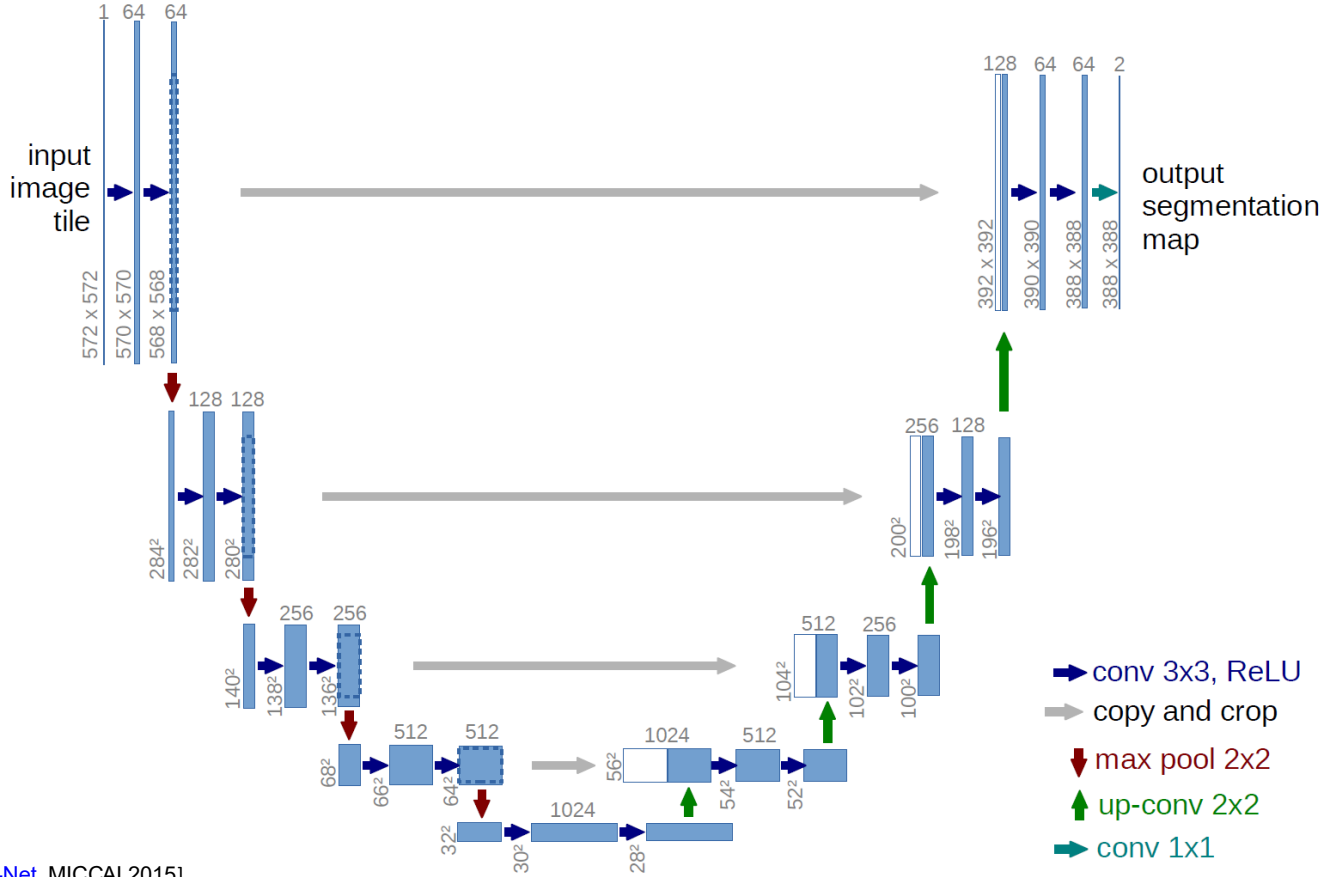


[Source: https://towardsdatascience.com/neural-style-transfer-and-visualization-of-convolutional-networks-7362f6cf4b9b]

Segnet - Results

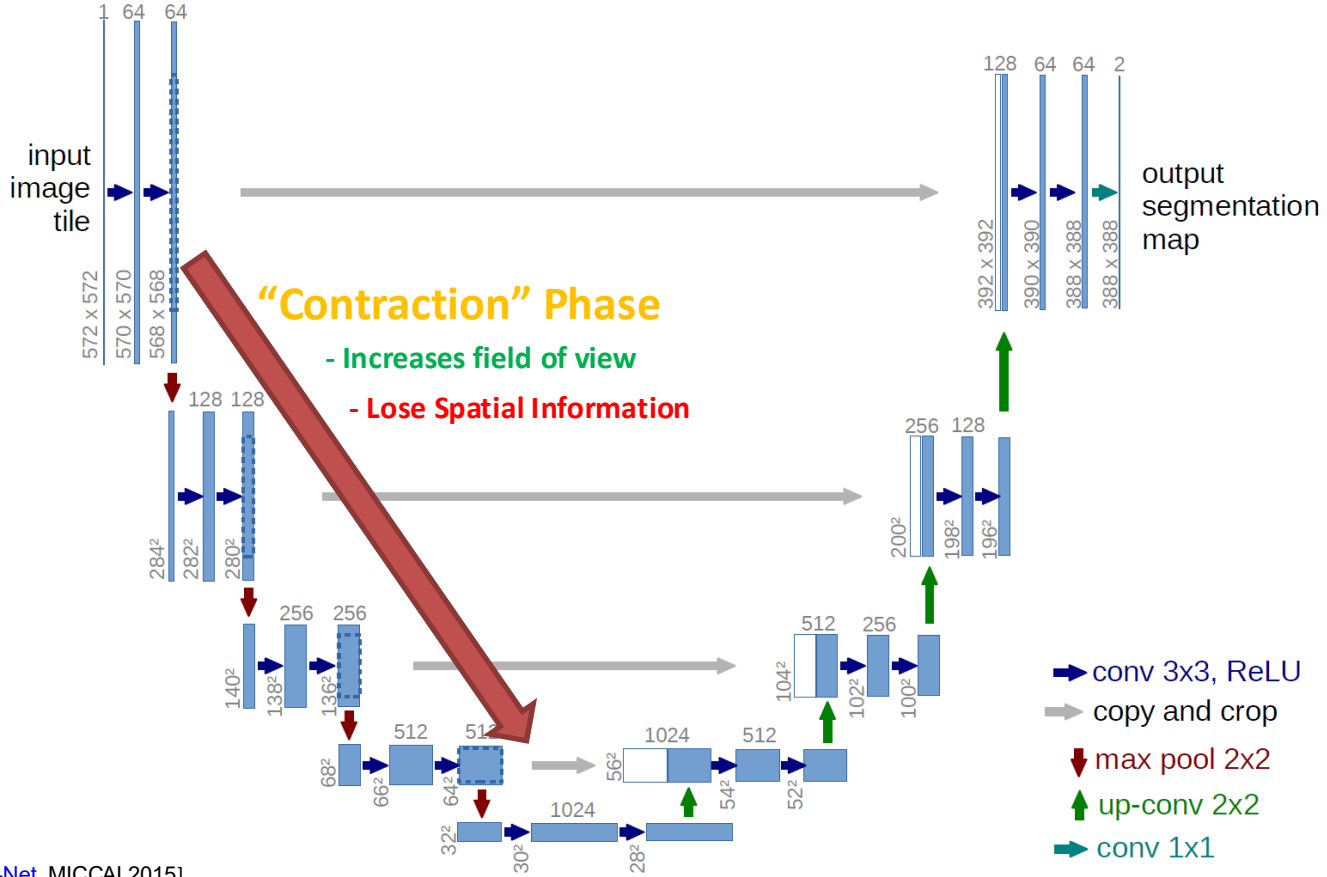
[Badrinarayanan et al., Segnet]

U-Net Architecture



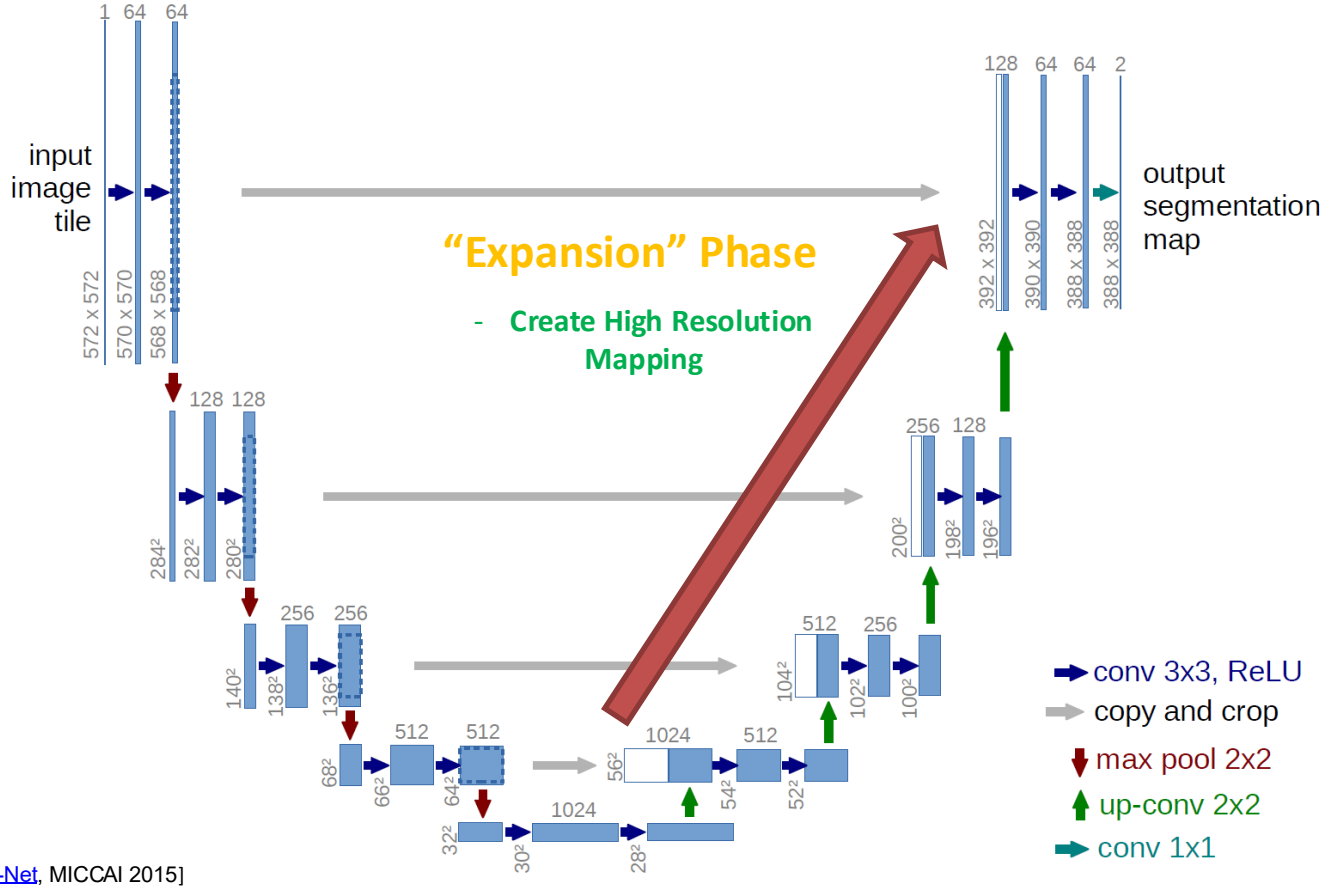
[Ronneberger, Fischer, Brox: U-Net, MICCAI 2015]

U-Net Architecture



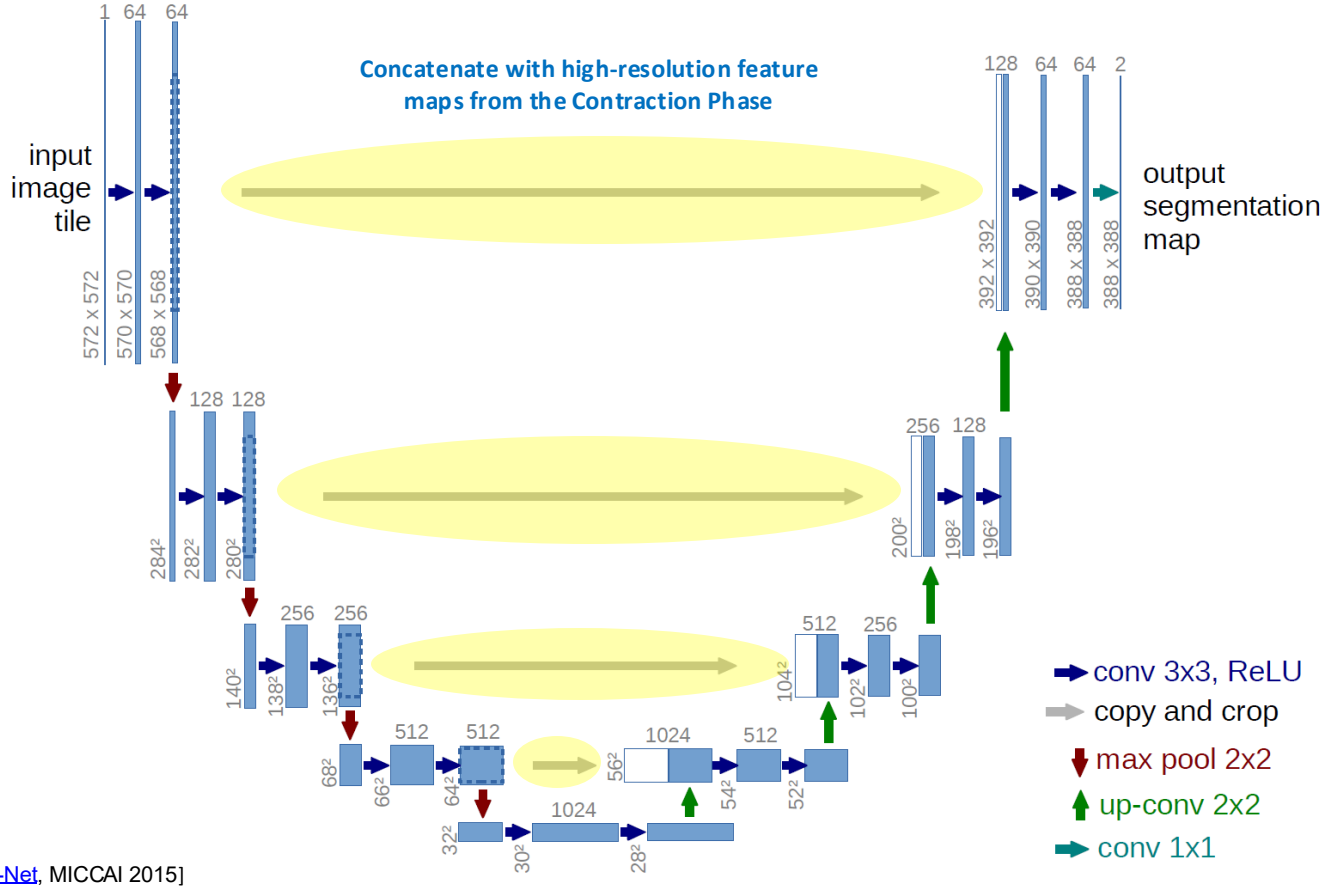
[Ronneberger, Fischer, Brox: U-Net, MICCAI 2015]

U-Net Architecture



[Ronneberger, Fischer, Brox: U-Net, MICCAI 2015]

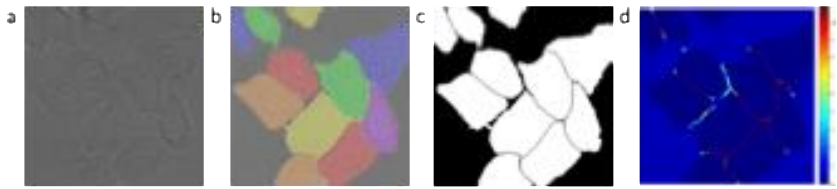
U-Net Architecture

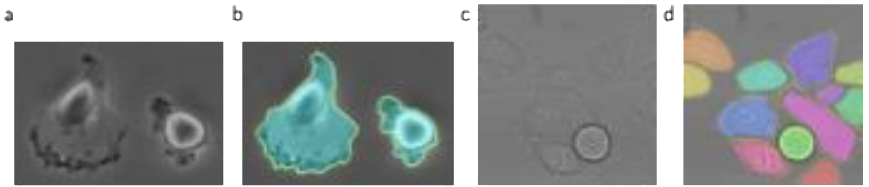


[Ronneberger, Fischer, Brox: U-Net, MICCAI 2015]

U-Net Architecture

- ConvNet architecture: inductive bias for translation equivariance
- Image-to-Image predictions: input resolution = output resolution
  - Segmentation
  - Superresolution
  - Denoising
  - Inpainting
  - Style transfer
  - etc.
- Encoder-decoder architecture: contracting & expanding path
- Combines the idea of ResNet with coarse-to-fine / pyramid networks: skip connections across hierarchy levels
- widely used today:
- originally developed for medical image segmentation:





[Ronneberger, Fischer, Brox: U-Net, MICCAI 2015]

Autoencoders

So far:

Supervised learning:

- training with large number of labeled images

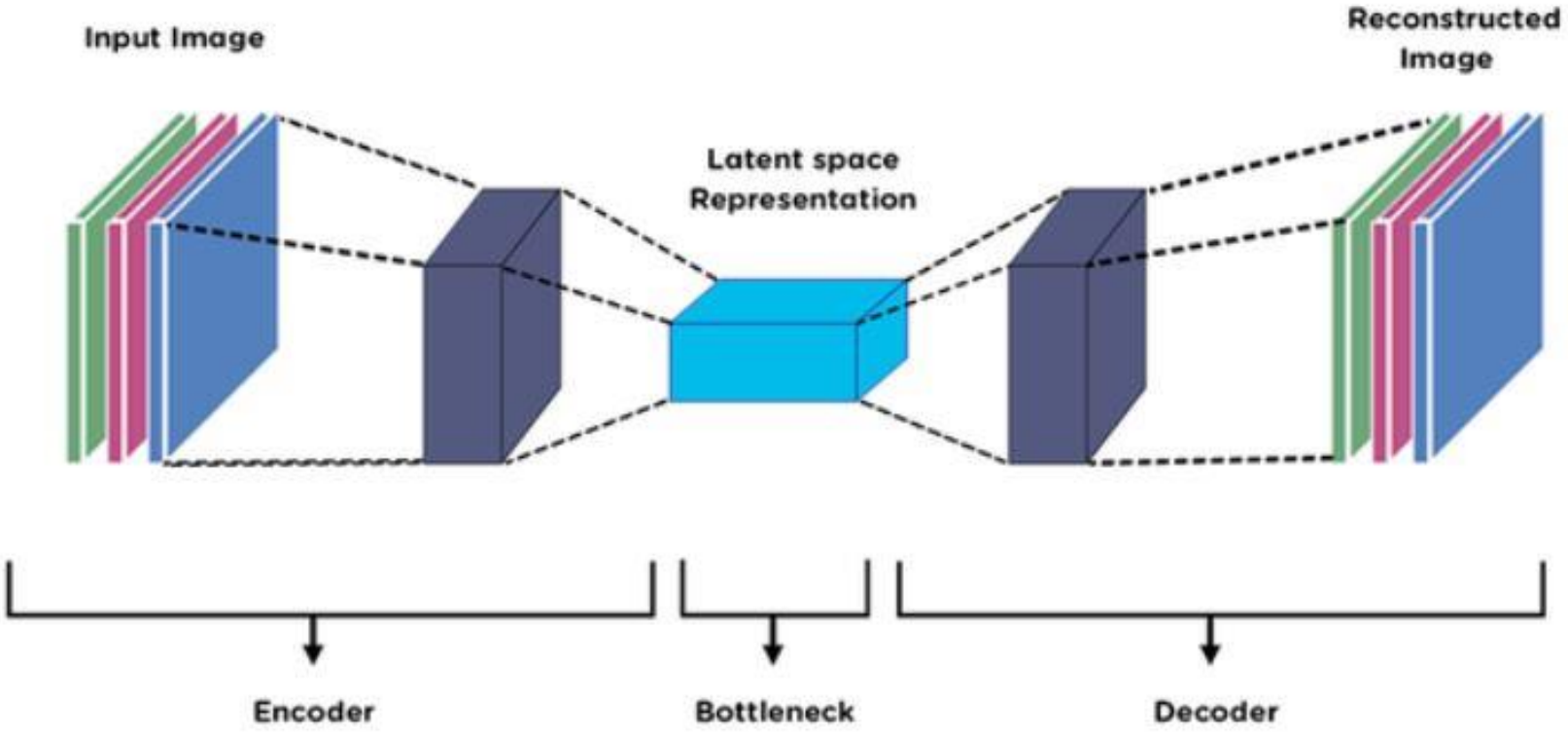
Now:

Unsupervised approach:

- to learn lower dimensional feature representations from unlabeled training data.

Autoencoders

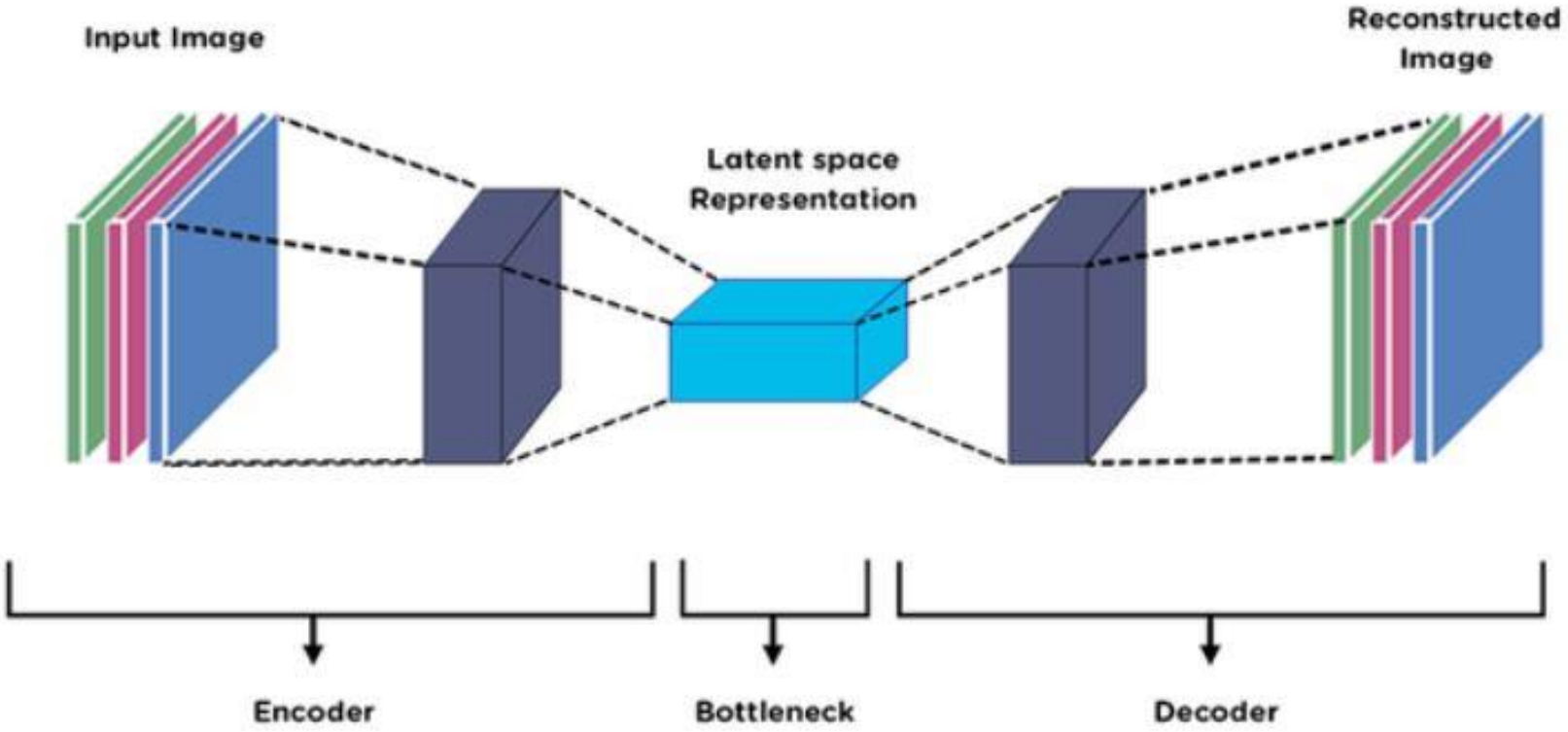
Encoder – decoder. Network is trained to minimize difference between Reconstructed Image and Input Image (e.g. LSE loss).



Autoencoders

Latent space is of lower dimension than input.

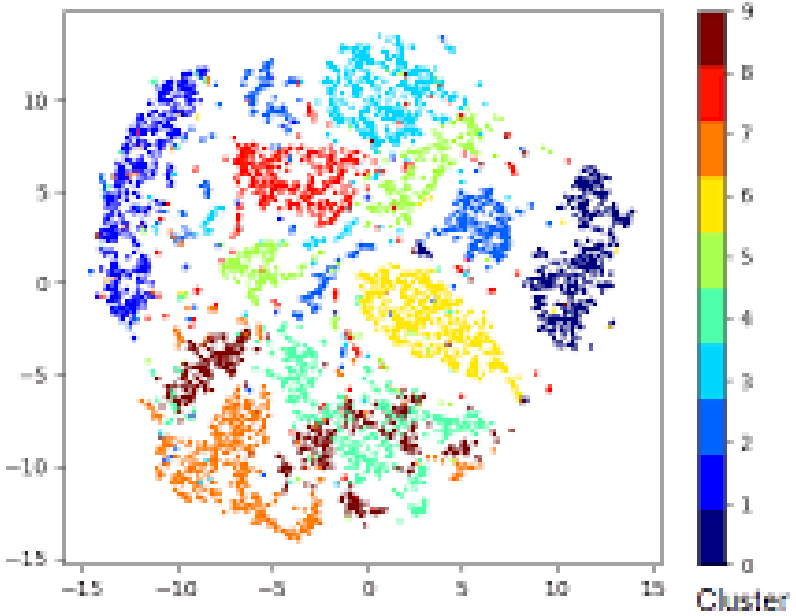
So reconstruction typical not perfect.



Autoencoders – Use case: Clustering

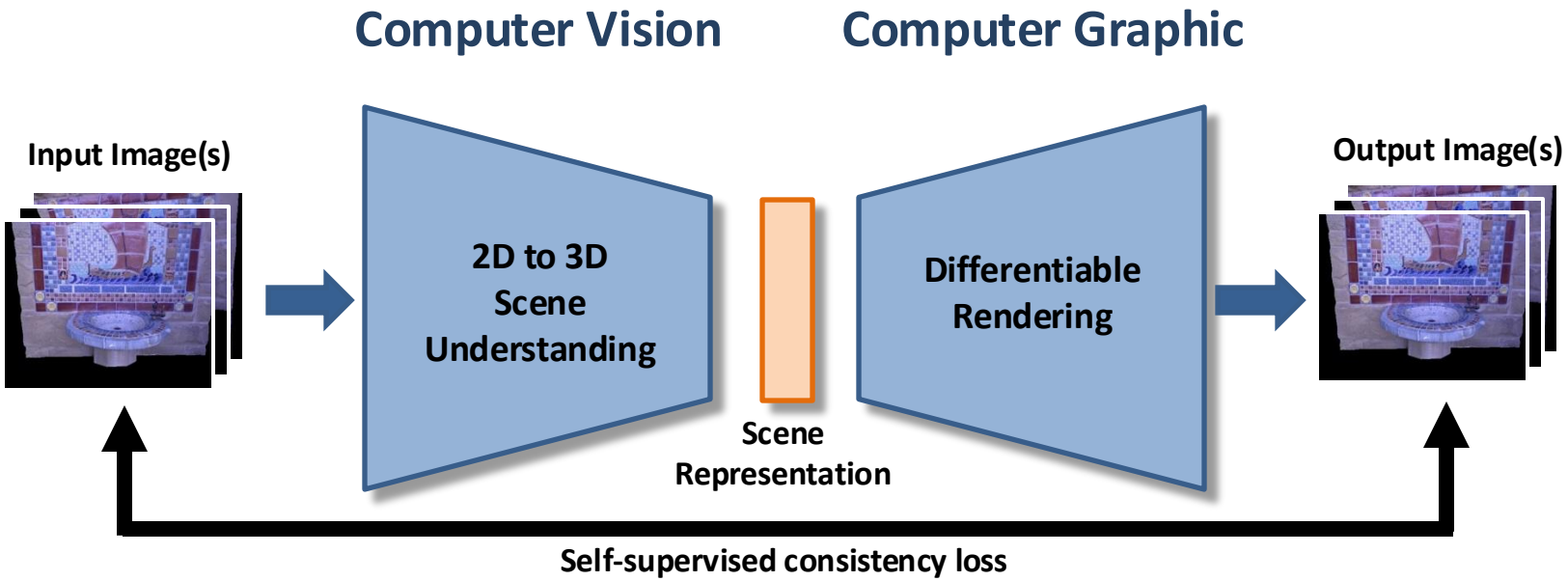
Embedding of MNIST dataset

MNIST



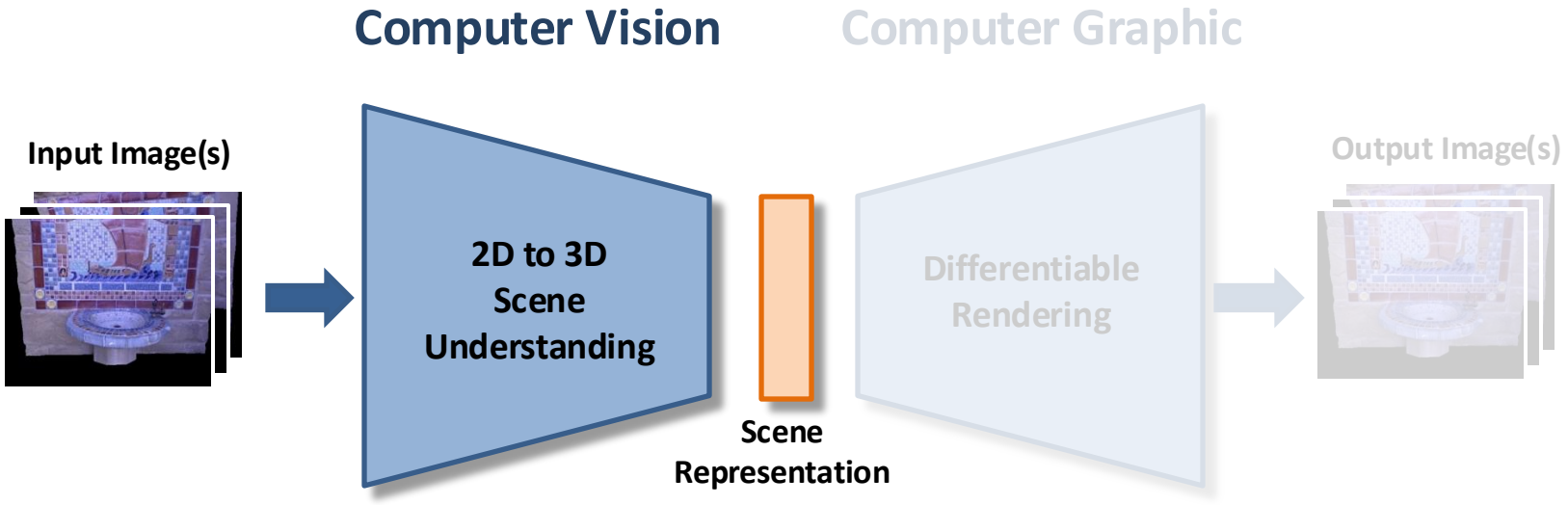
[Example resources: 1, 2]

Autoencoders – Self-Supervised Learning



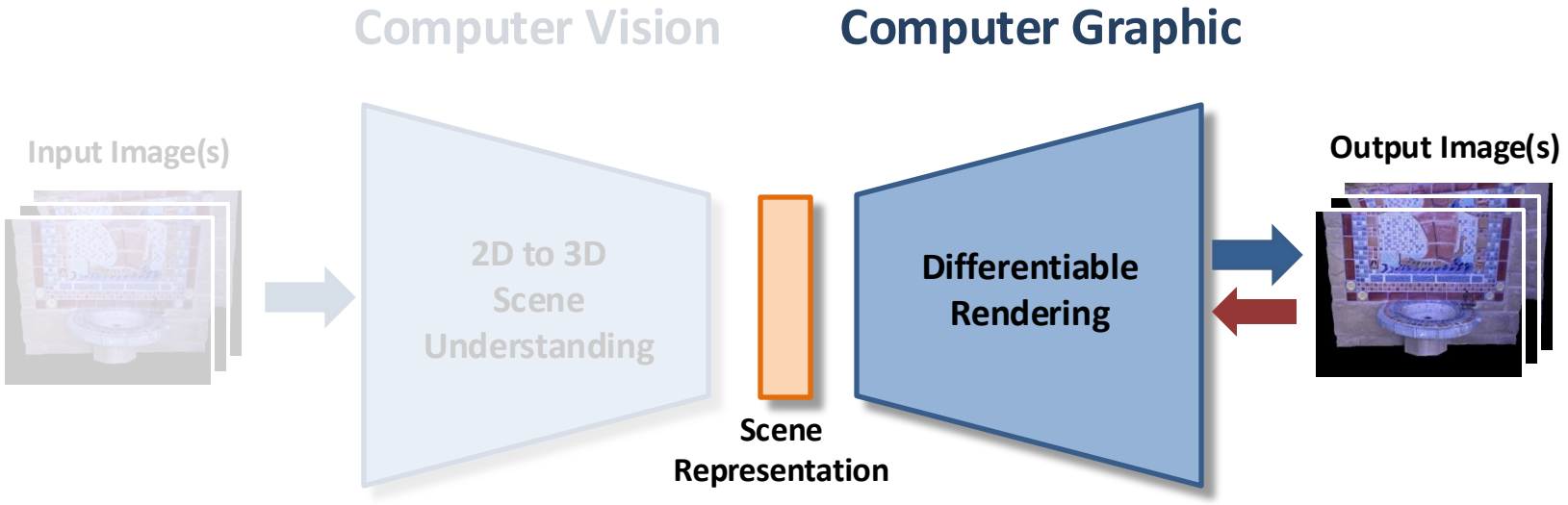
Training-Time

Autoencoders – Self-Supervised Learning



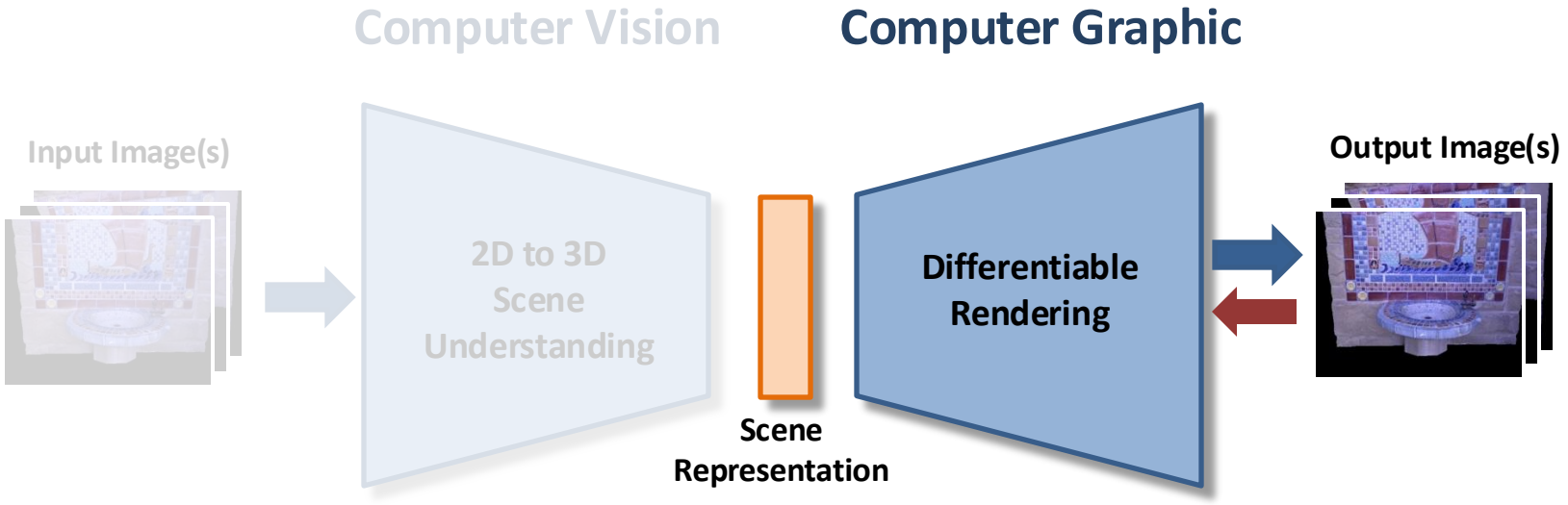
Test-Time

Autoencoders – Self-Supervised Learning



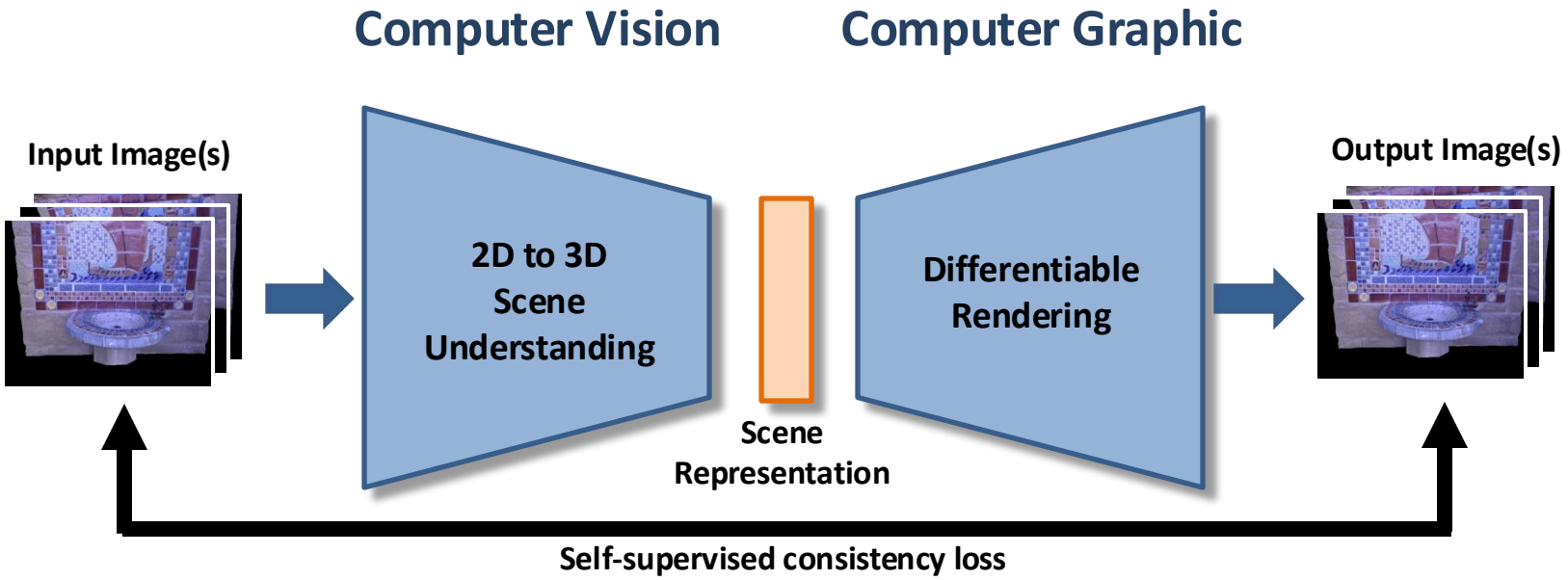
Test-Time

Autoencoders – Self-Supervised Learning



Test-Time

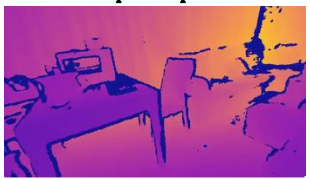
Autoencoders – Self-Supervised Learning

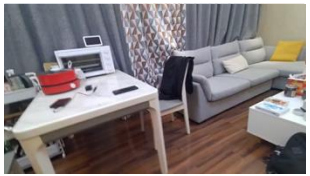


- This principle is the foundation of many recent works, e.g. Neural radiance fields (NeRF, Gaussian Splatting, etc.), 3D reconstruction, 3D localization (Dust3R, Mast3R, etc.), Human pose estimation, …
- Enables scalable learning without human labeling: only images/videos required

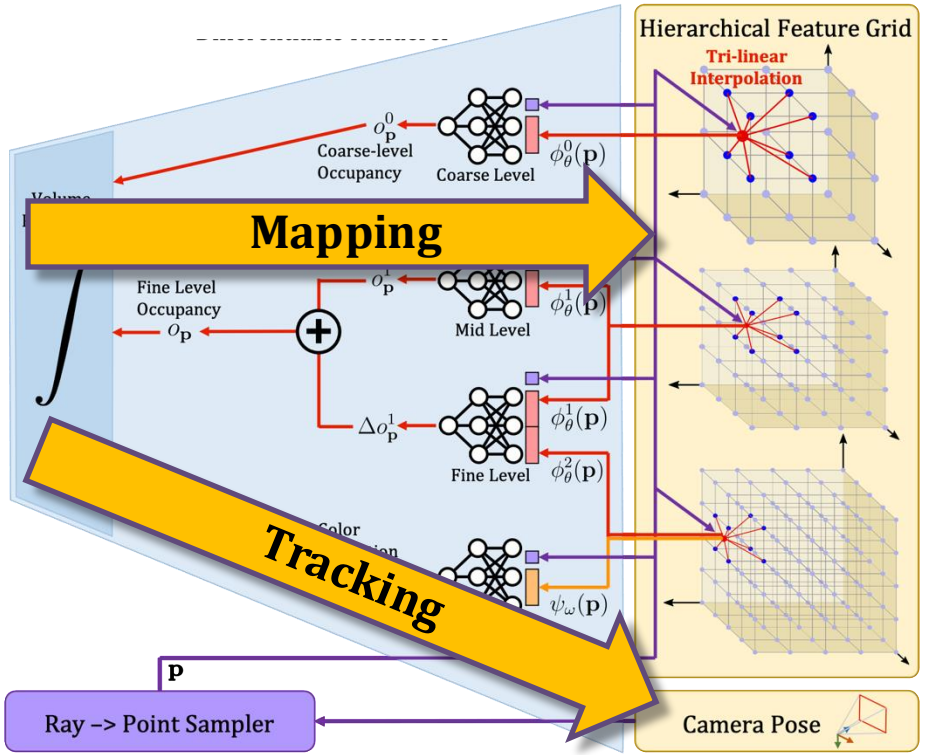
Example: NICE-SLAM

Input Depth





Input RGB



[NICE-SLAM, Zhu, Peng, Larsson, Xu, Bao, Cui, Oswald, Pollefeys, CVPR 2022]

Questions?

Disclaimer

Many of the slides used here are obtained from online resources (including many open lecture materials) without appropriate acknowledgement. They are used here for the sole purpose of classroom teaching. All the credit and all the copyrights belong to the original authors. You should not copy it, redistribute it, put it online, or use it for any other purposes than for this course.