Designing, Visualizing and Understanding Deep Neural Networks

Lecture 8: Object Detection and Segmentation

CS 182/282A Spring 2019
David Chan

Slides originated from Canny, Chen, Chou, Li, Karpathy, Johnson, and Yang

Last Time: Batch Normalization

[Ioffe and Szegedy, 2015]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

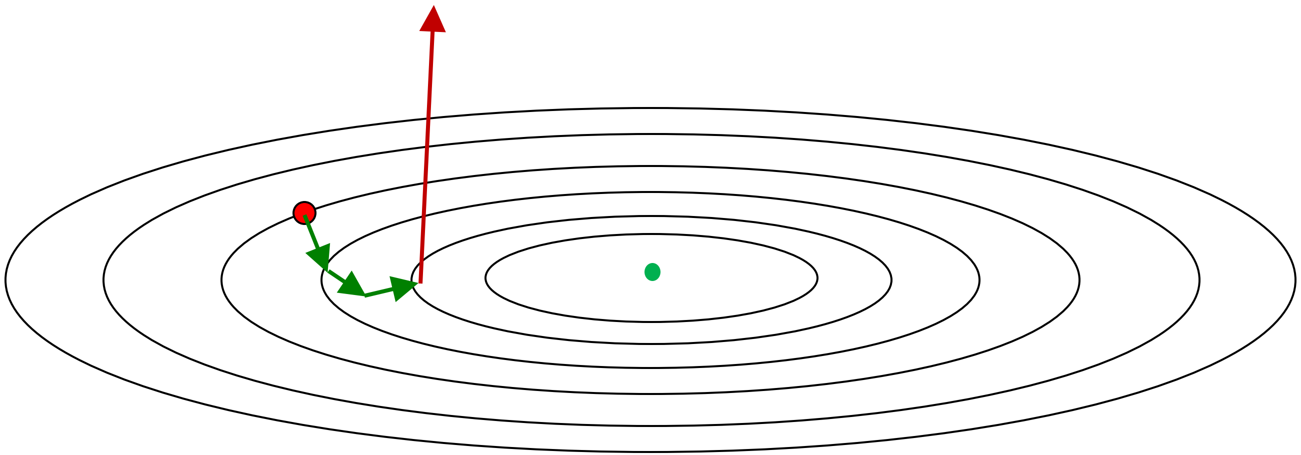
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$\boxed{y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}}$$

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Reduces need for dropout

Un-normalization!! Re-compute and apply the optimal scaling and bias for each neuron!
Learn γ and β (same dims as μ and σ^2).
It can (should?) learn the identity mapping!

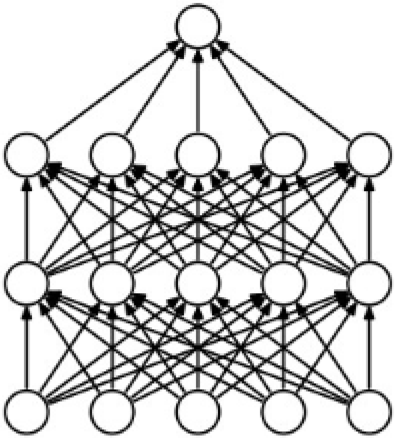
Last Time: Gradient Clipping by Value or Norm



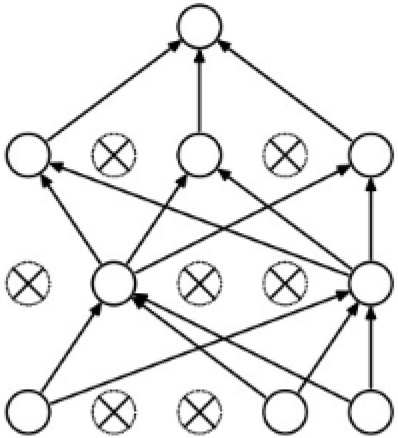
Last Time: Dropout

"randomly set some neurons to zero in the forward pass"

i.e. multiply by random bernoulli variables with parameter p.



(a) Standard Neural Net



(b) After applying dropout.

Note, p is the probability of keeping a neuron

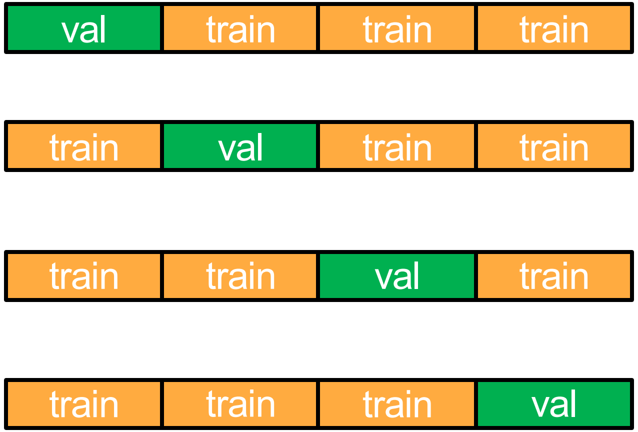
[Srivastava et al., 2014]

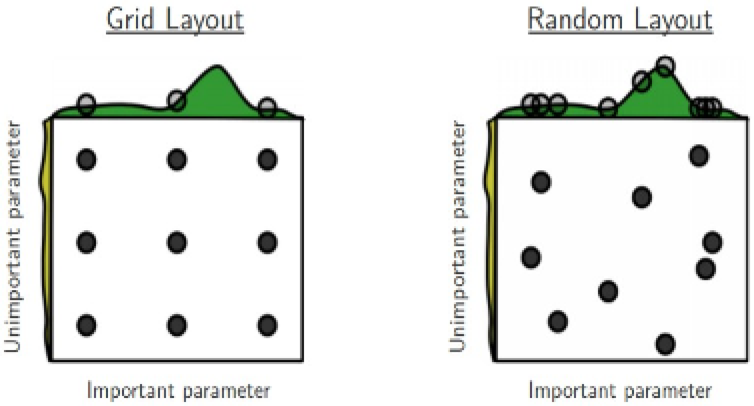
Last Time: Ensembles (VGGNet and CIFAR 10)

Model	Prediction method	Test Accuracy
Baseline (10 epochs)	Single model	0.837
True ensemble of 10 models	Average predictions	0.855
True ensemble of 10 models	Voting	0.851
Snapshots (25) over 10 epochs	Average predictions	0.865
Snapshots (25) over 10 epochs	Voting	0.861
Snapshots (25) over 10 epochs	Parameter averaging	0.864

Last Time: Hyperparameter Optimization

Use Validation blocks to compare hyper-parameter choices

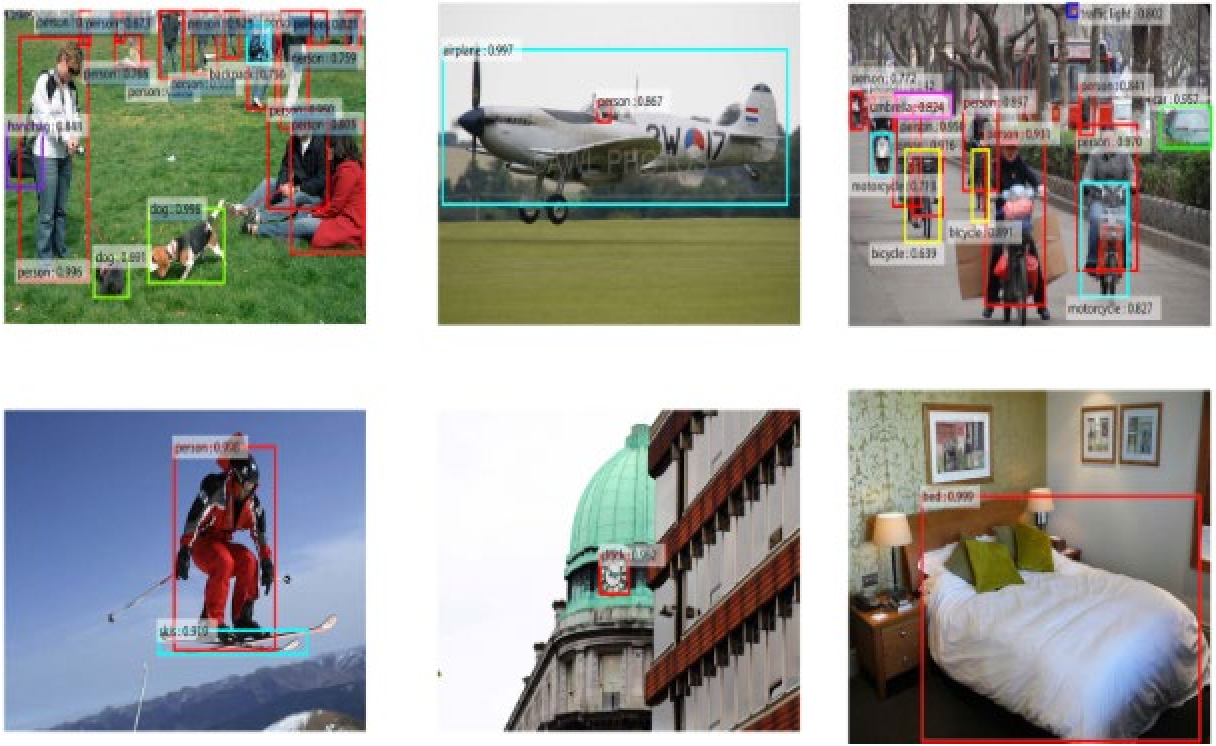




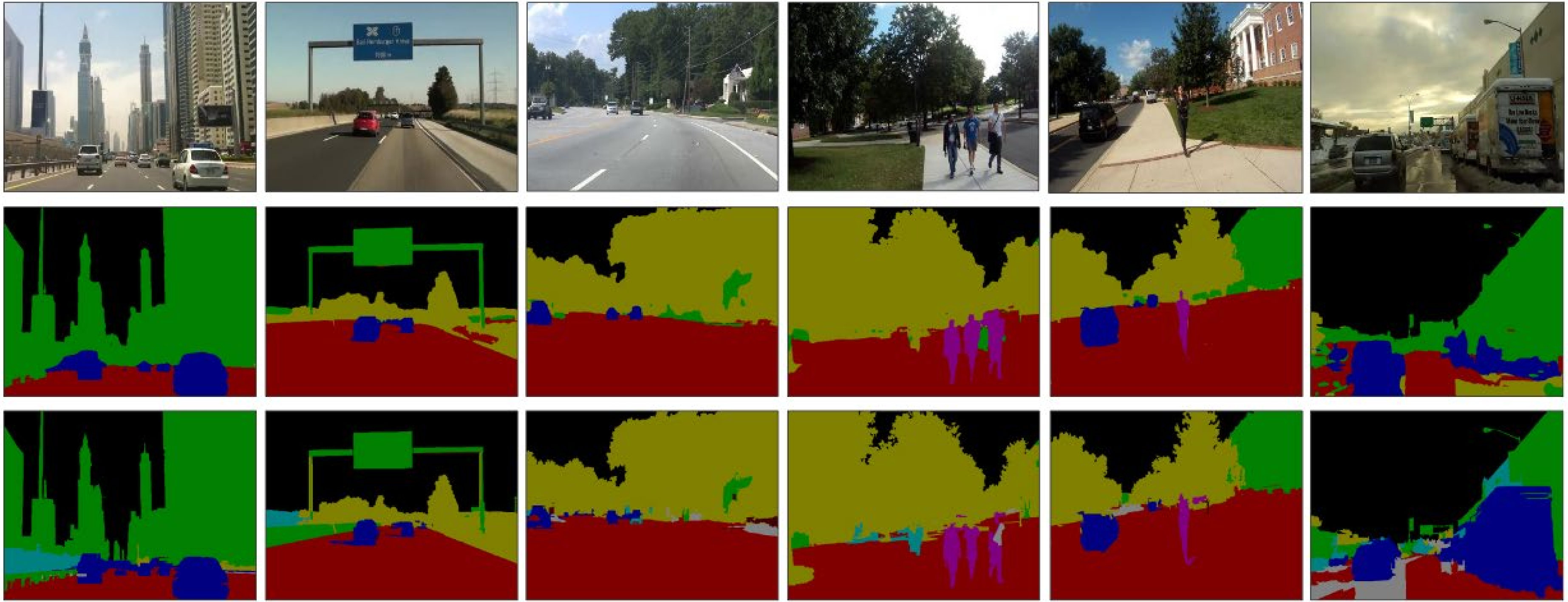
Course Updates/Logistics

- Project Proposals are due today
- Assignment 1 was due yesterday…

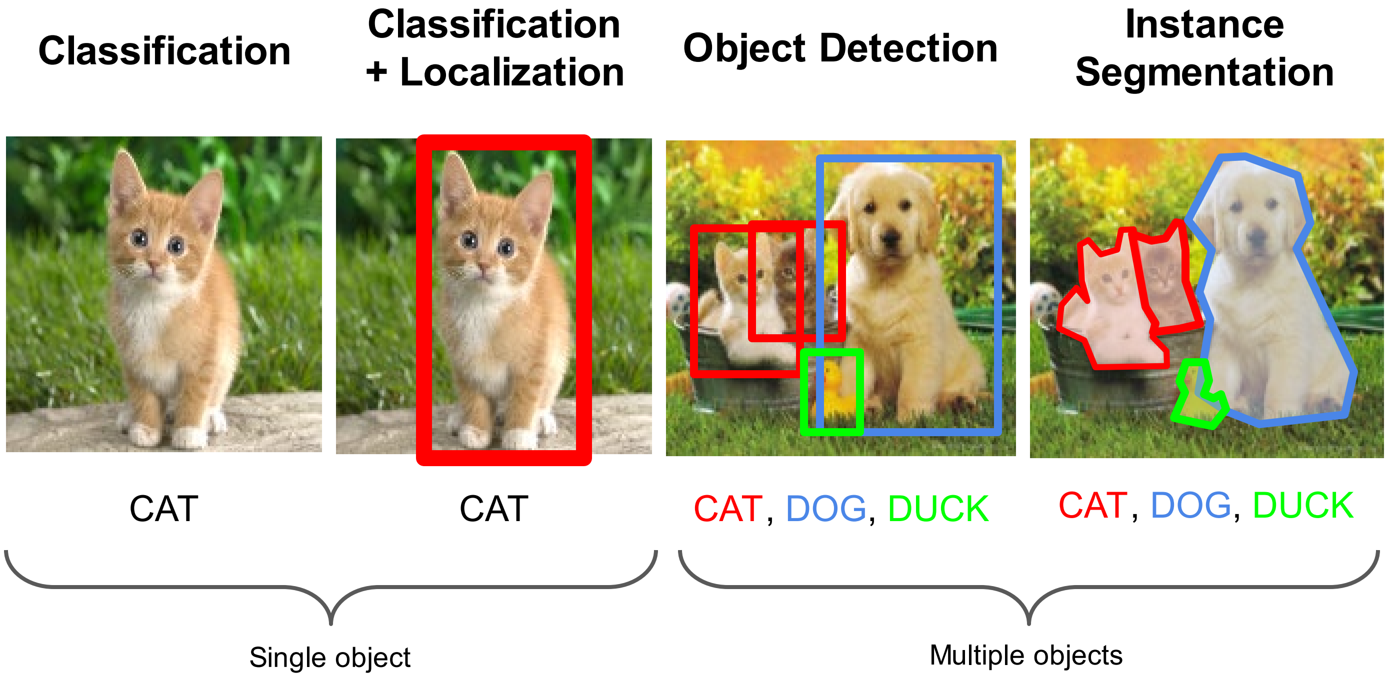
This Time: Localization and Detection



This Time: Localization and Detection



Computer Vision Tasks



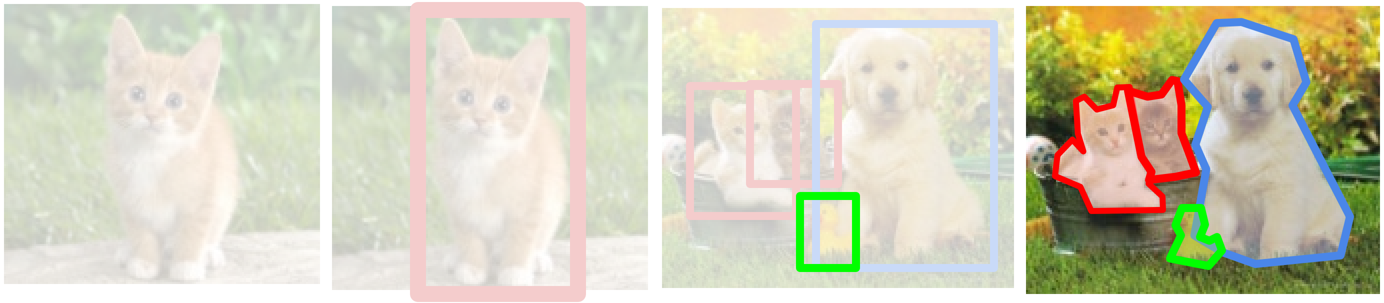
Computer Vision Tasks

Classification

Classification
+ Localization

Object Detection

Instance Segmentation



Semantic Segmentation

Label every pixel!

Don’t differentiate instances, only worry about pixels

Classic computer vision problem

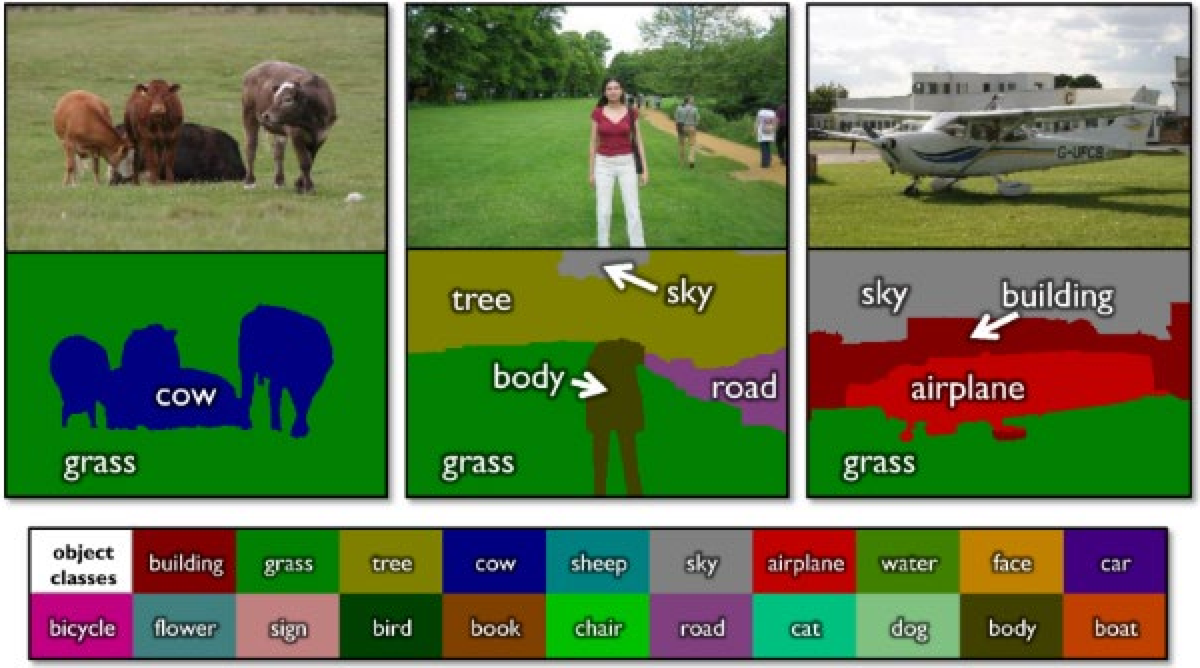
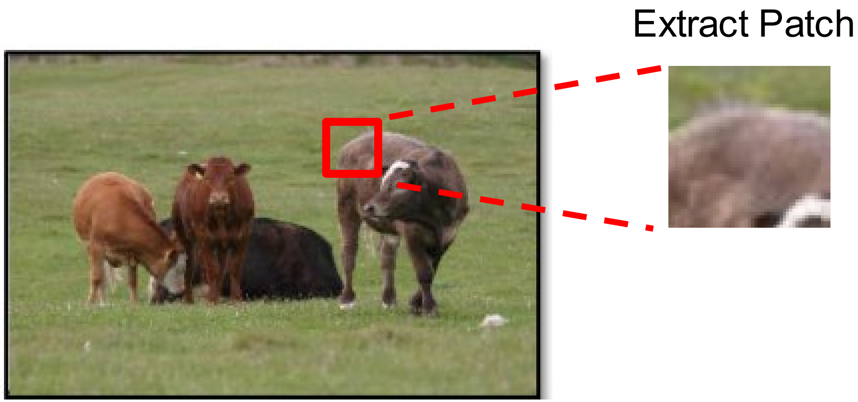


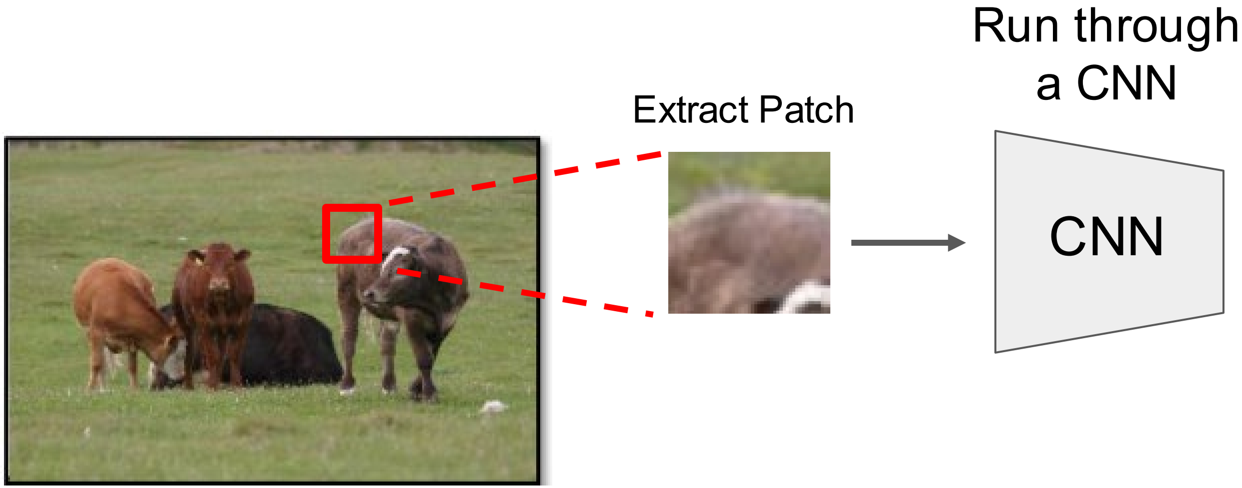
Figure credit: Shotton et al, “TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context”, IJCV 2007

Idea #1 – Classify Every Pixel

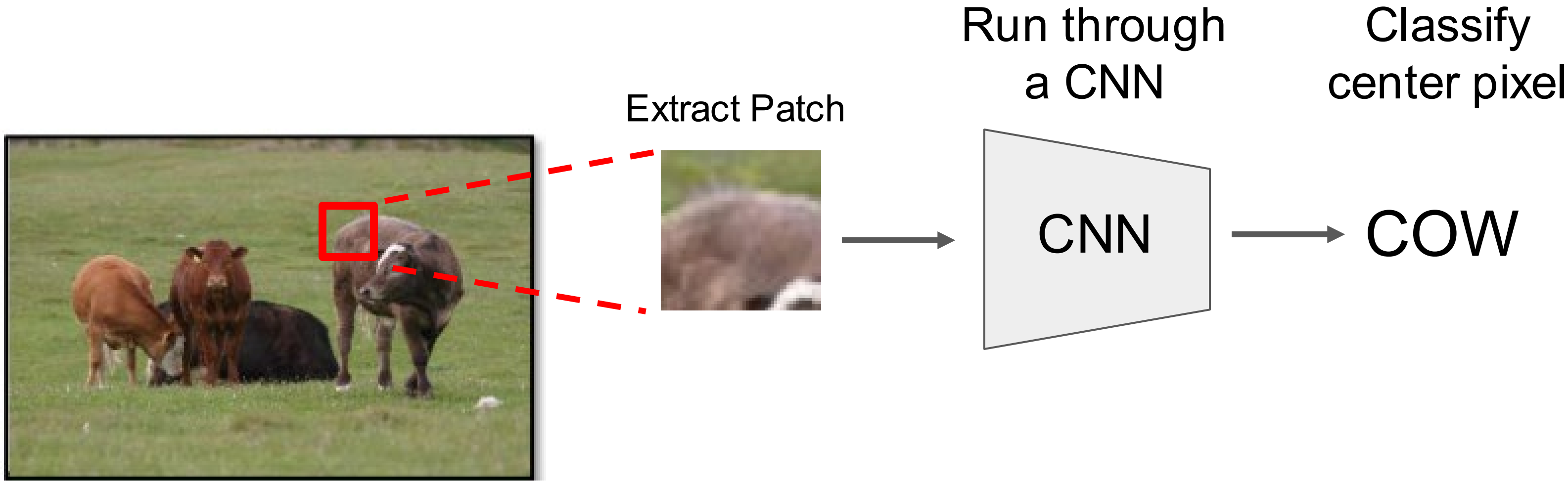
Idea #1 – Classify Every Pixel



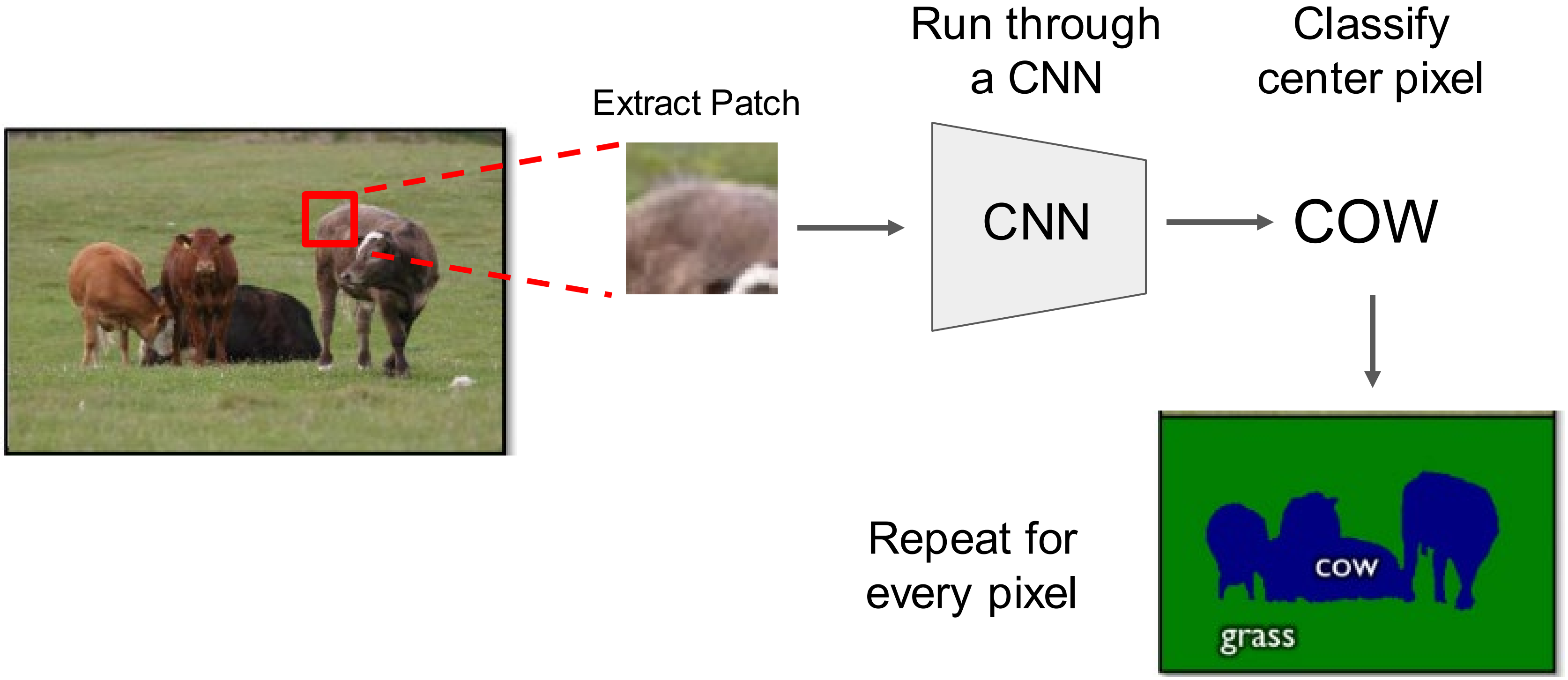
Idea #1 – Classify Every Pixel



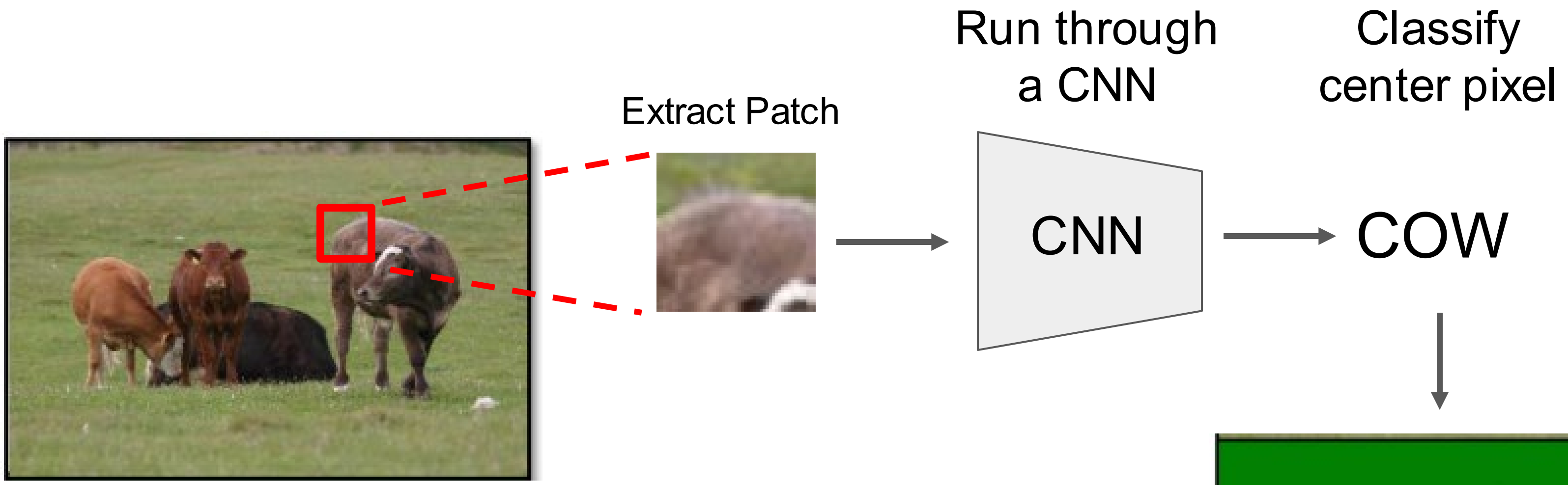
Idea #1 – Classify Every Pixel

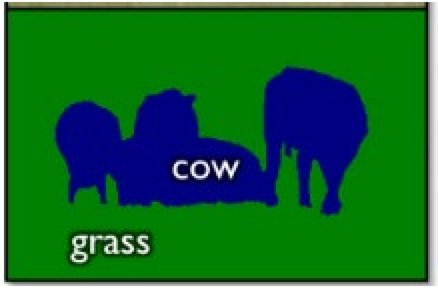


Idea #1 – Classify Every Pixel



Idea #1 – Classify Every Pixel





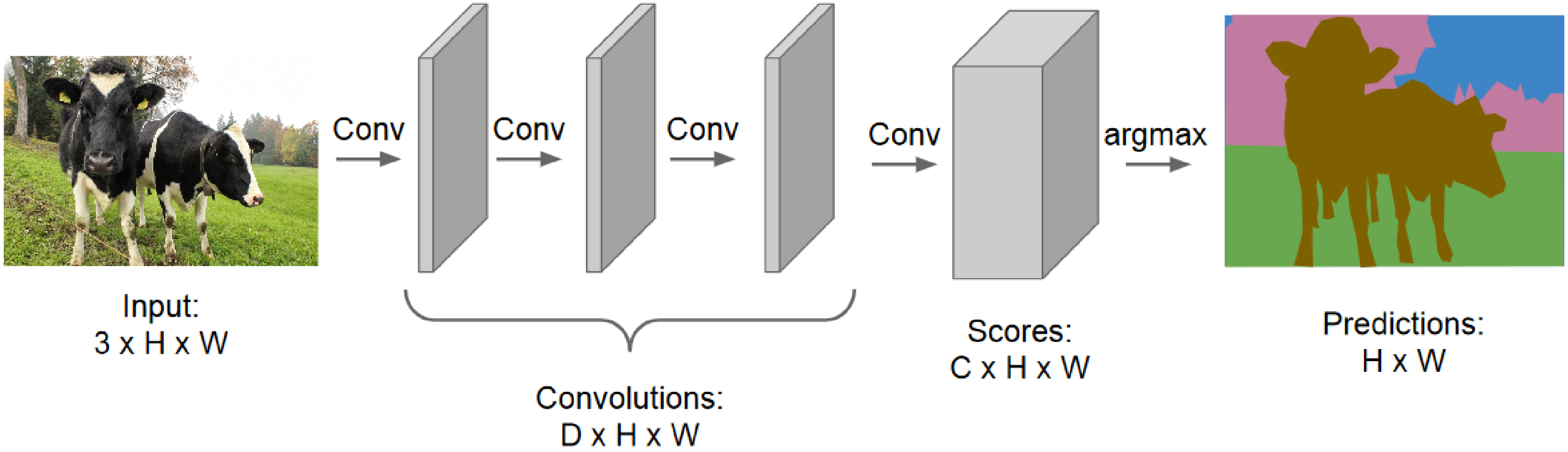
Repeat for every pixel

Issues:

- **Repeats the same computation multiple times**
- **Doesn't make use of global information**

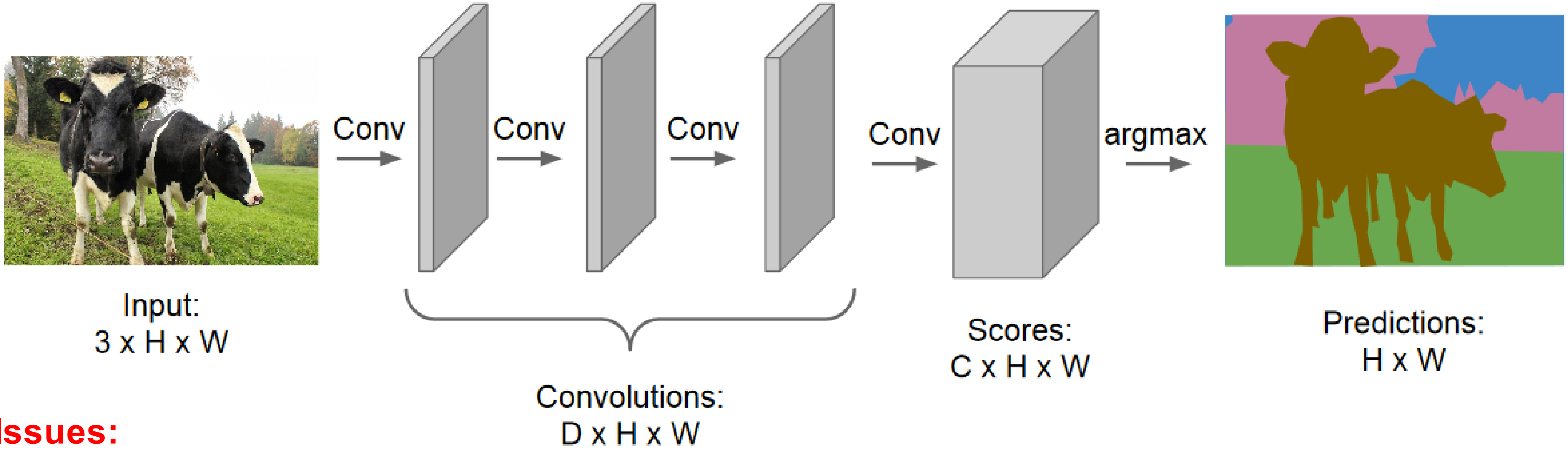
Idea #2 – Convolutional Network

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



Idea #2 – Convolutional Network

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

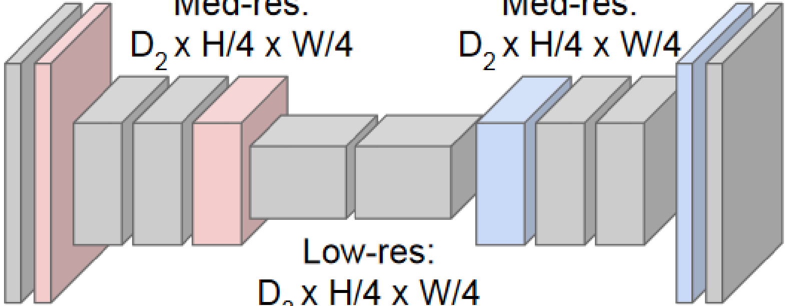


Issues:
- **Convolutions at full resolution can be expensive**

Idea #3 – Fully Connected CNN

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

Input:
$3 \times H \times W$



High-res:
$D_1 \times H/2 \times W/2$

Med-res:
$D_2 \times H/4 \times W/4$

Low-res:
$D_3 \times H/4 \times W/4$

Med-res:
$D_2 \times H/4 \times W/4$

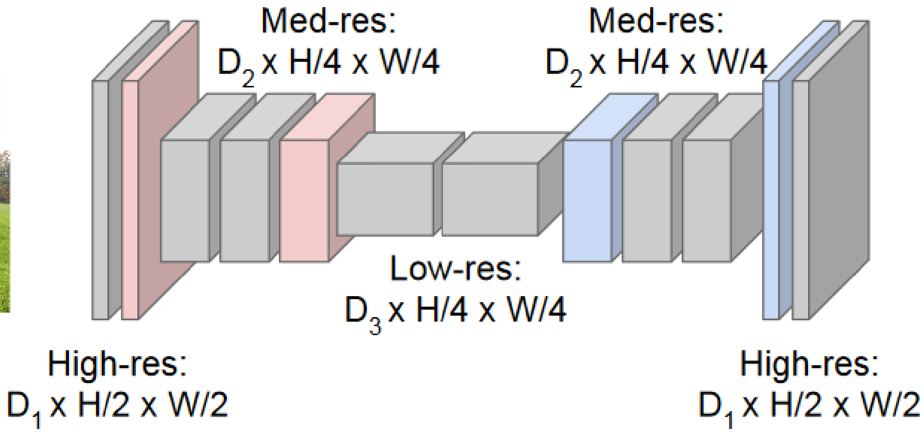
High-res:
$D_1 \times H/2 \times W/2$

Predictions:
$H \times W$

Idea #3 – Fully Connected CNN

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

Input:
3 x H x W



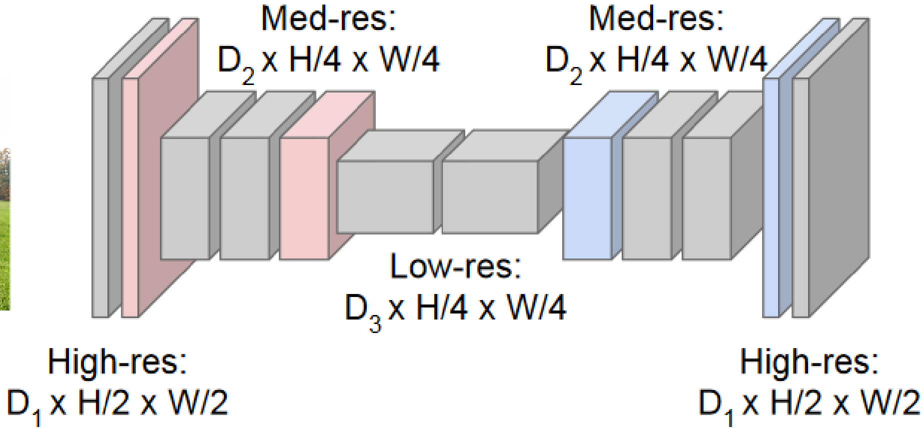
Predictions:
H x W

Downsampling:
Pooling, strided convolution

Idea #3 – Fully Connected CNN

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

Input:
$3 \times H \times W$



Predictions:
$H \times W$

Downsampling:
Pooling, strided convolution

Upsampling:
???

In-Network Upsampling

Nearest Neighbor

1	2
3	4

Input: 2 x 2

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

“Bed of Nails”

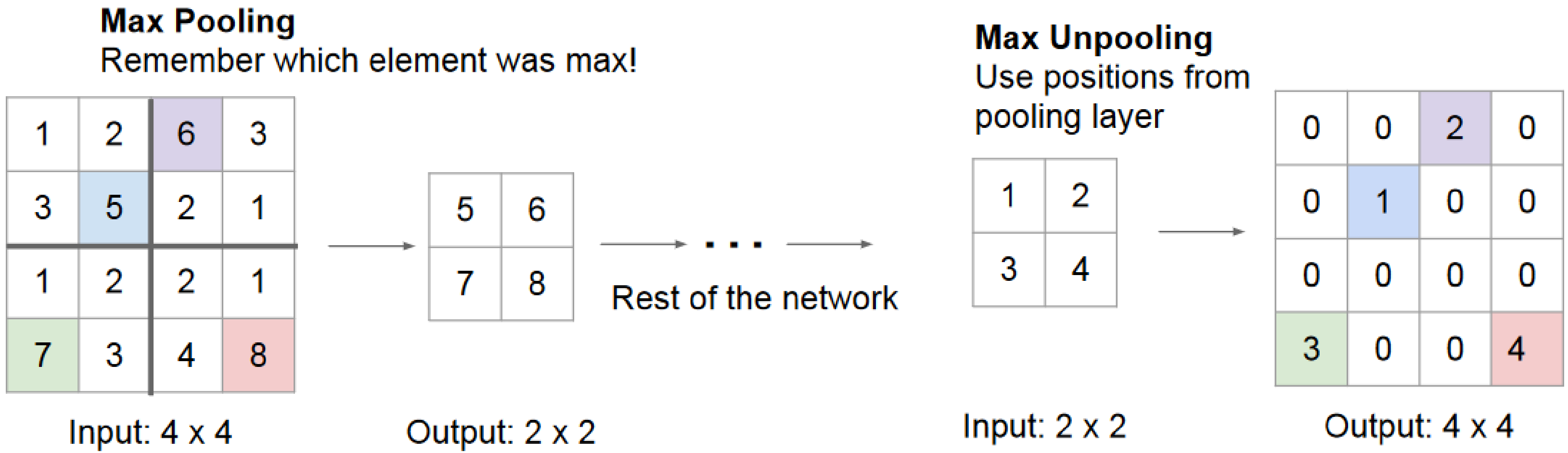
1	2
3	4

Input: 2 x 2

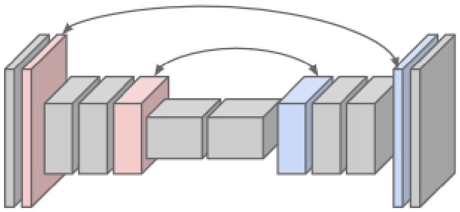
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

In-Network Upsampling: Max Unpooling

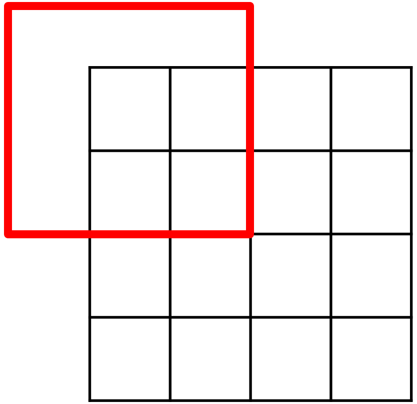


Corresponding pairs of downsampling and upsampling layers

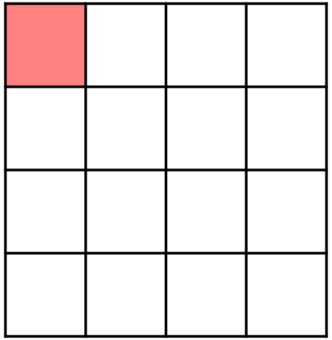


Learnable Upsampling: Transpose Convolution

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



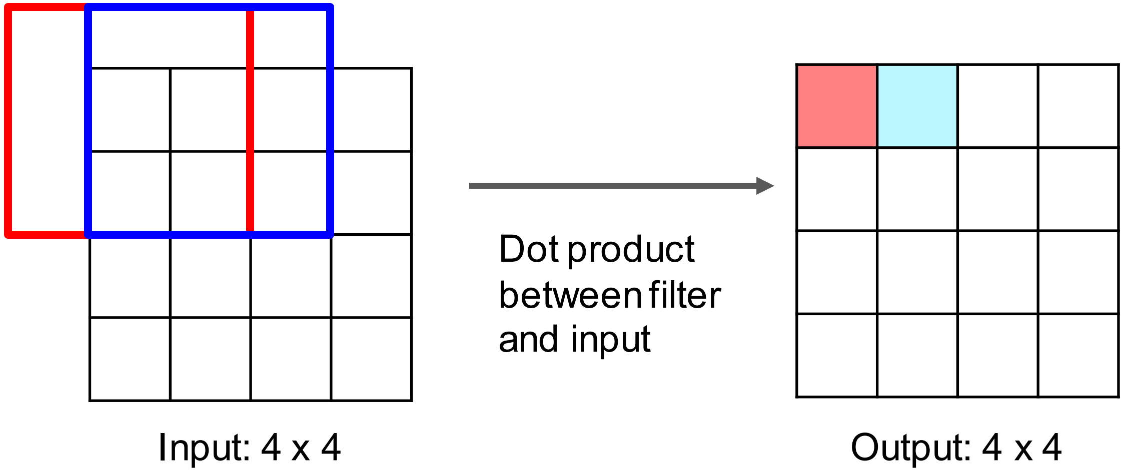
Input: 4 x 4



Output: 4 x 4

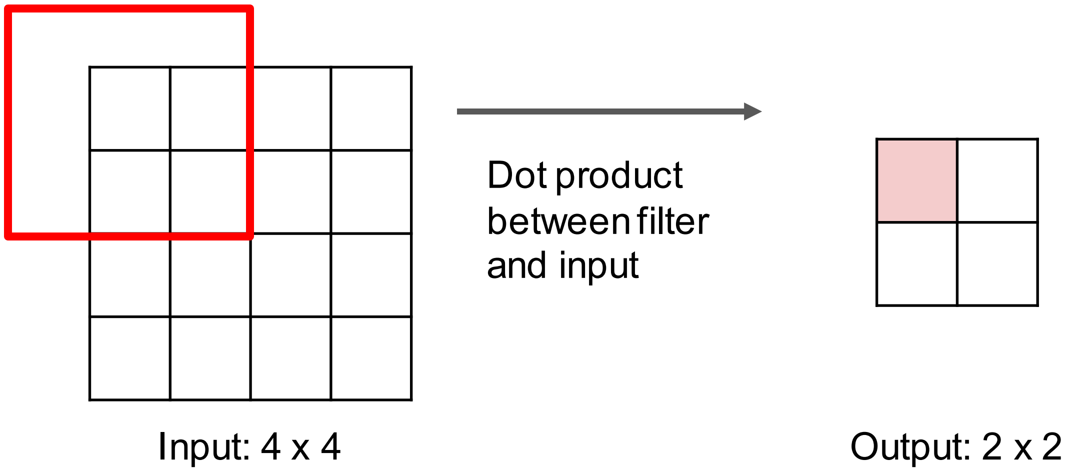
Learnable Upsampling: Transpose Convolution

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



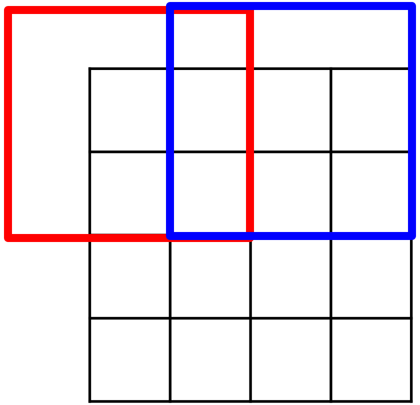
Learnable Upsampling: Transpose Convolution

Recall: Typical 3 x 3 convolution, **stride 2** pad 1



Learnable Upsampling: Transpose Convolution

Recall: Typical 3 x 3 convolution, **stride 2** pad 1

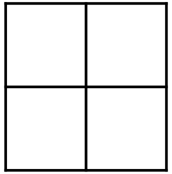


Dot product between filter and input

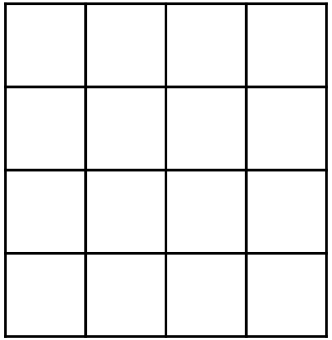
Input: 4 x 4

Output: 2 x 2

Learnable Upsampling: Transpose Convolution

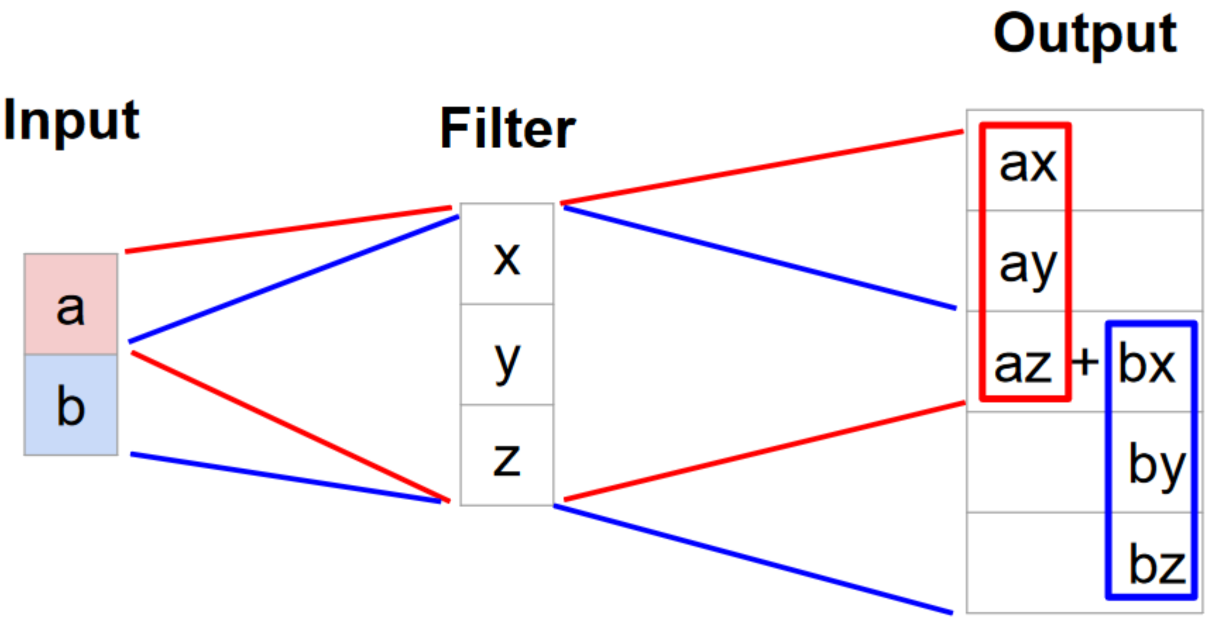


Input: 2 x 2



Output: 4 x 4

Learnable Upsampling: Transpose Convolution

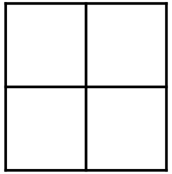


Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

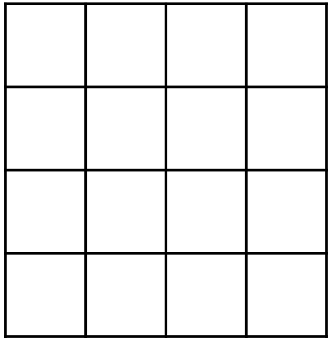
Need to crop one pixel from output to make output exactly 2x input

Learnable Upsampling: Transpose Convolution

Transpose Convolution: Typical 3 x 3 convolution, stride 2 pad 1



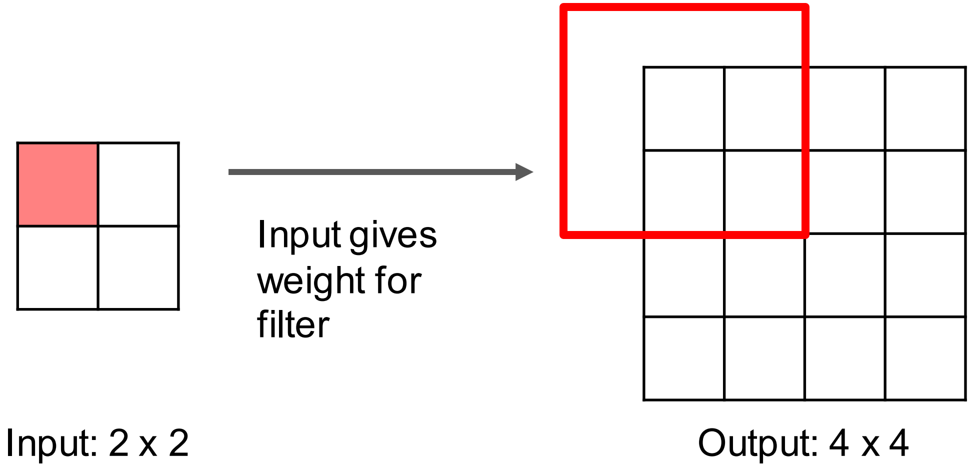
Input: 2 x 2



Output: 4 x 4

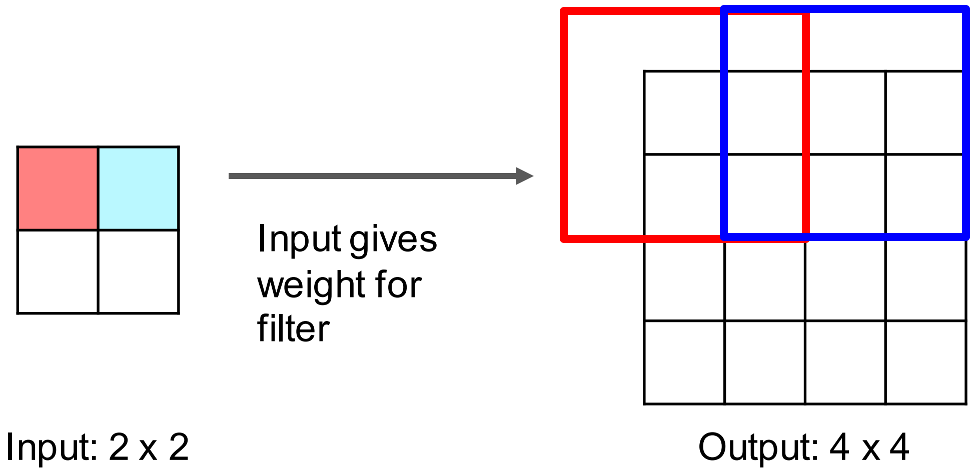
Learnable Upsampling: Transpose Convolution

Transpose Convolution: Typical 3 x 3 convolution, stride 2 pad 1



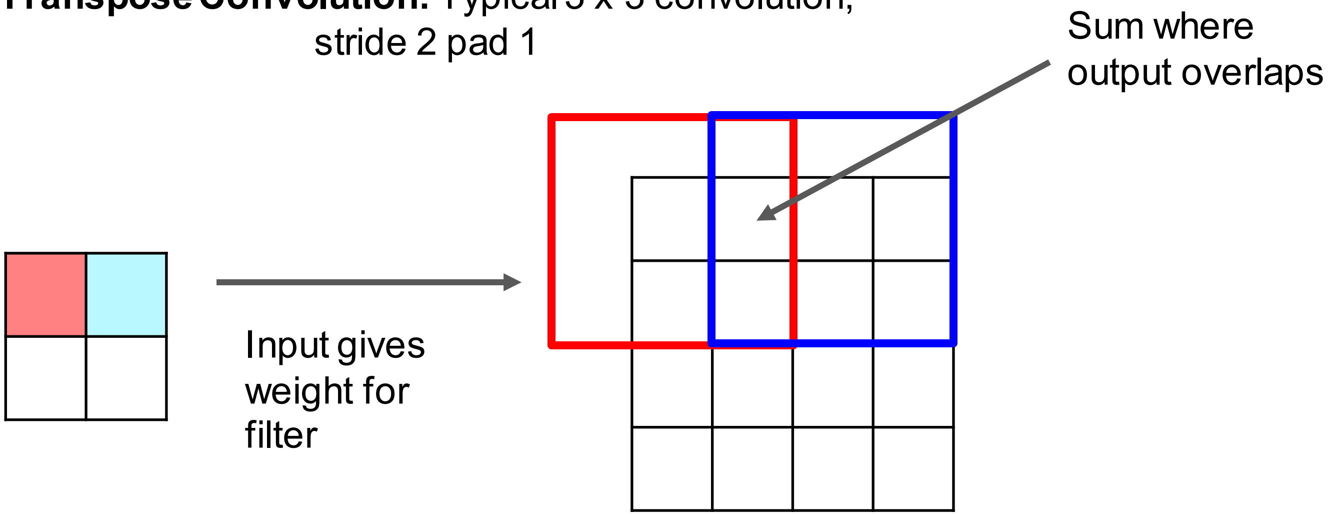
Learnable Upsampling: Transpose Convolution

Transpose Convolution: Typical 3 x 3 convolution, stride 2 pad 1



Learnable Upsampling: Transpose Convolution

Transpose Convolution: Typical 3 x 3 convolution, stride 2 pad 1

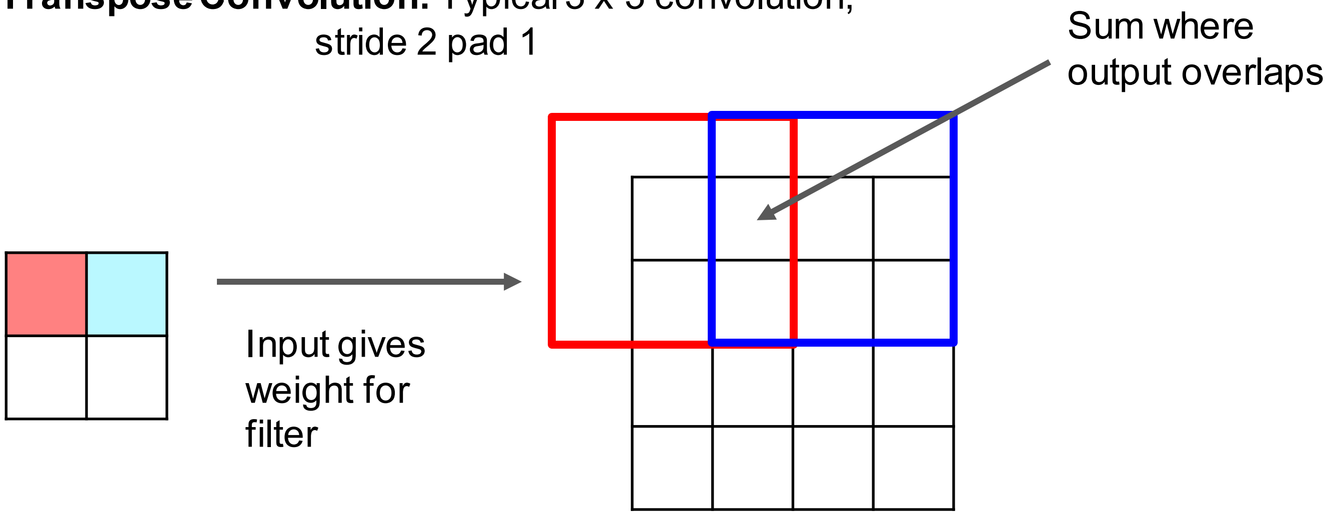


Input: 2 x 2

Output: 4 x 4

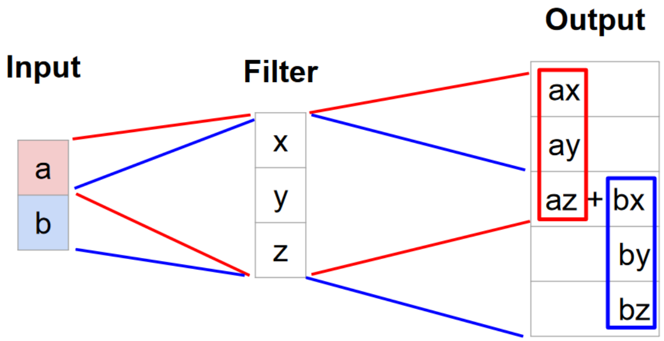
Learnable Upsampling: Transpose Convolution

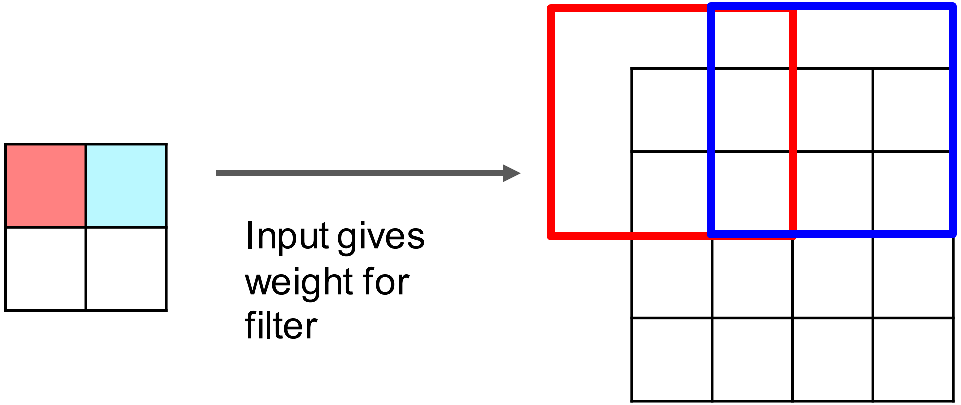
Transpose Convolution: Typical 3 x 3 convolution, stride 2 pad 1



Same as backward pass for normal convolution!

Learnable Upsampling: Transpose Convolution





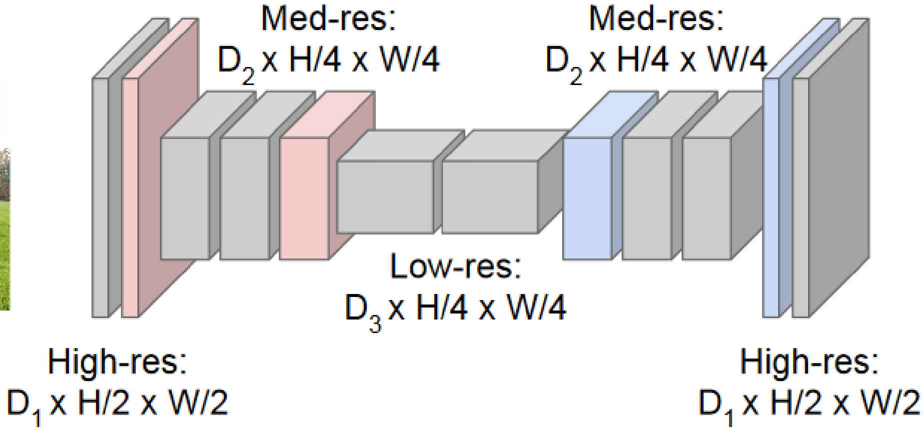
Input: 2 x 2

Output: 4 x 4

Idea #3 – Fully Connected CNN

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

Input:
3 x H x W

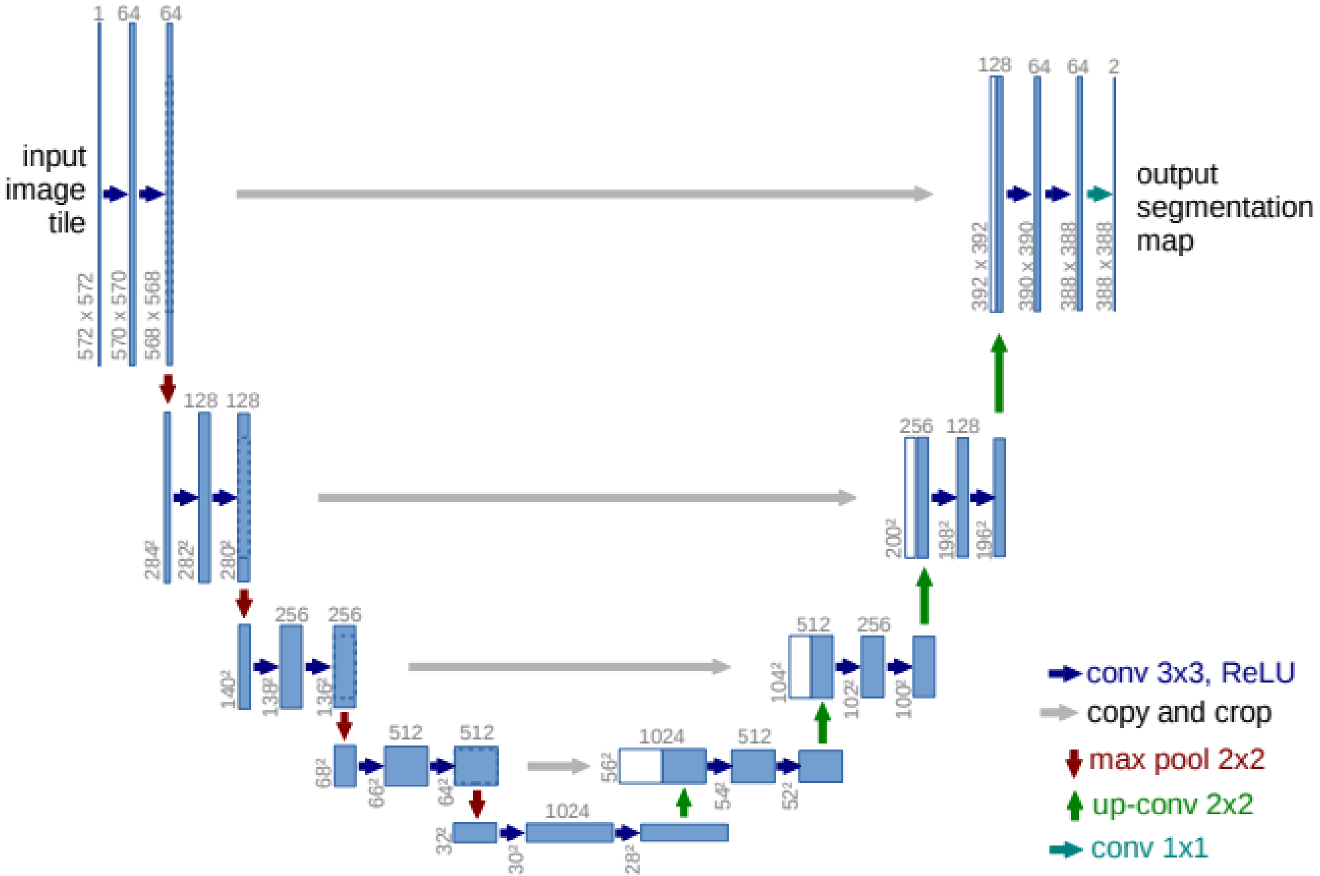


Predictions:
H x W

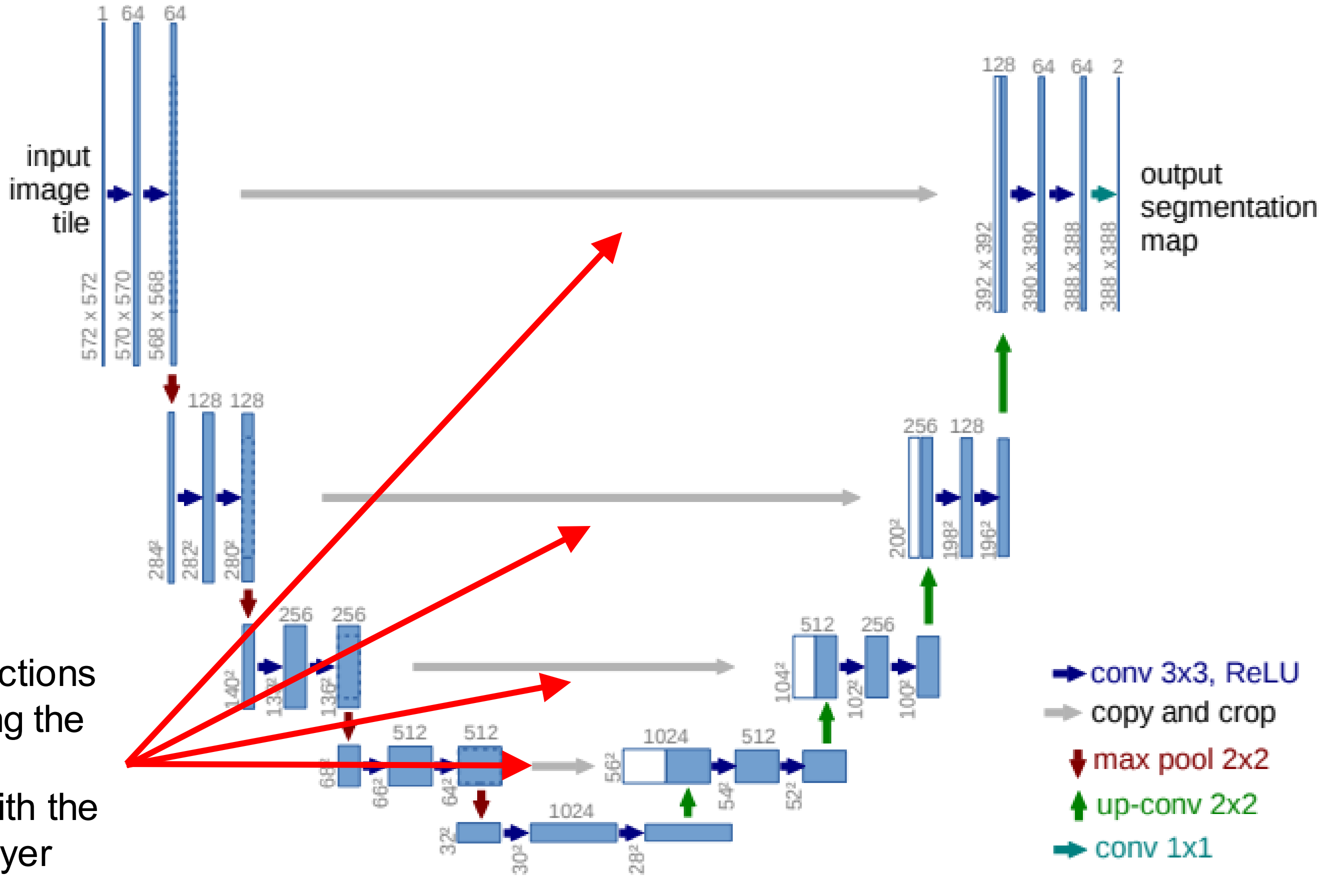
Downsampling:
Pooling, strided convolution

Upsampling:
Strided Transpose convolution

Idea #4 – UNet (FCNN + Residuals)



Idea #4 – UNet (FCNN + Residuals)



Residual Connections Made by copying the input, and concatenating with the upsampled layer

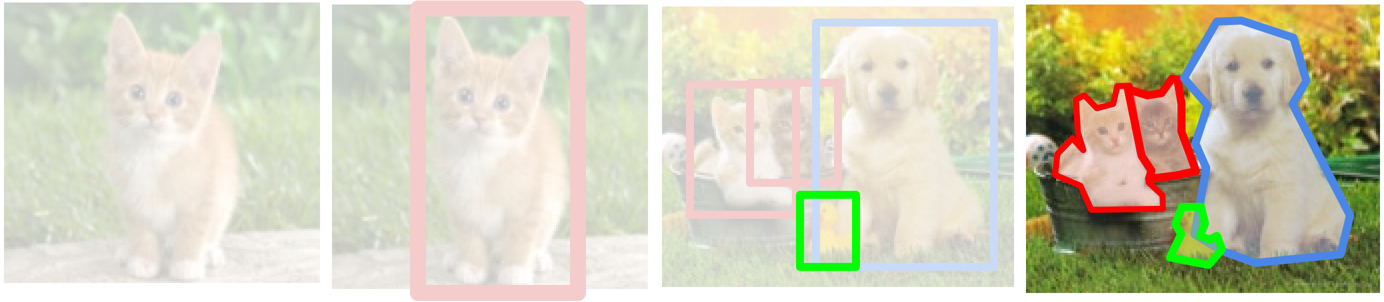
Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation



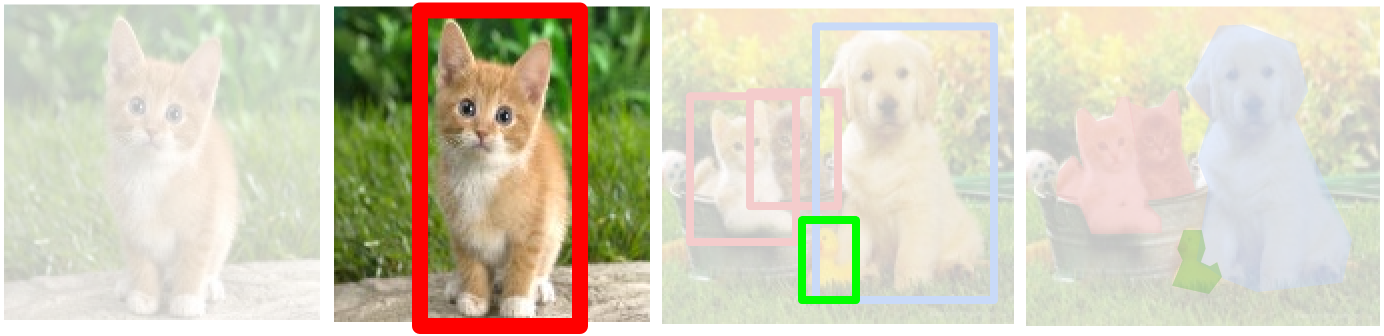
Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation



Classification + Localization

Classification

Input: Image
Output: Class Label
Evaluation Metric: Class Accuracy

Cat

Classification + Localization

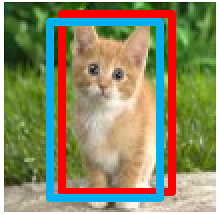
Classification

Input: Image
Output: Class Label
Evaluation Metric: Class Accuracy

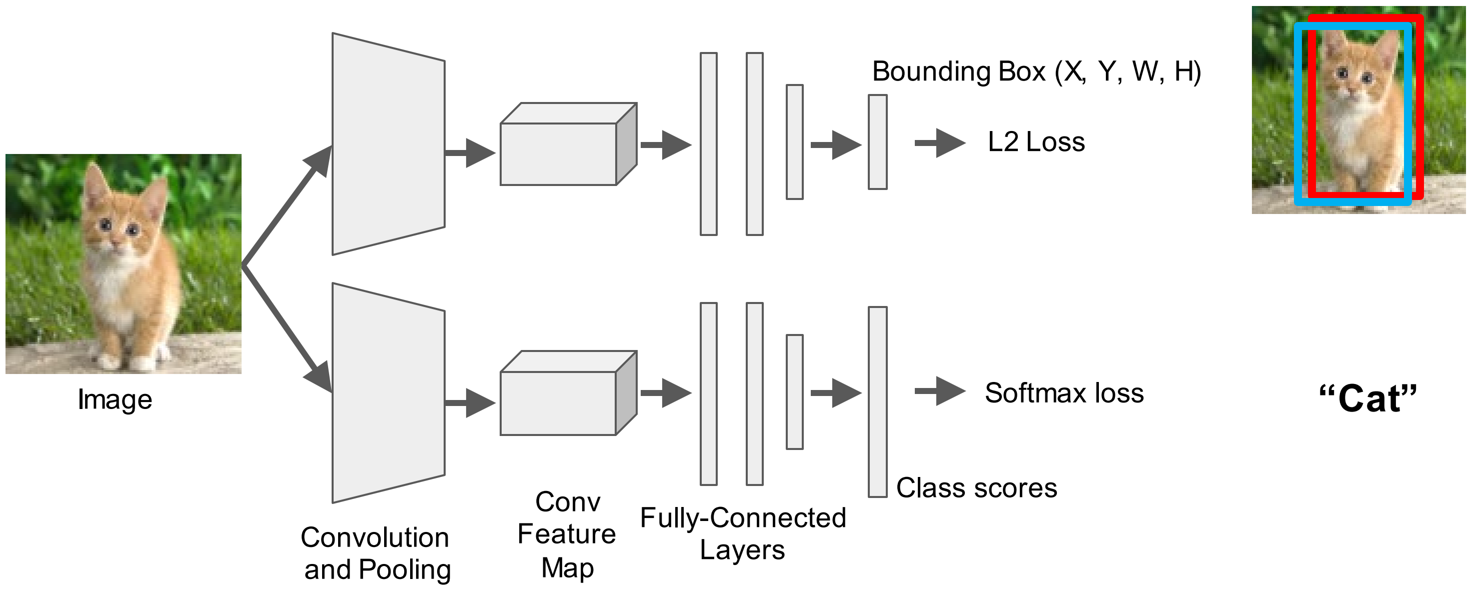
Cat

Localization

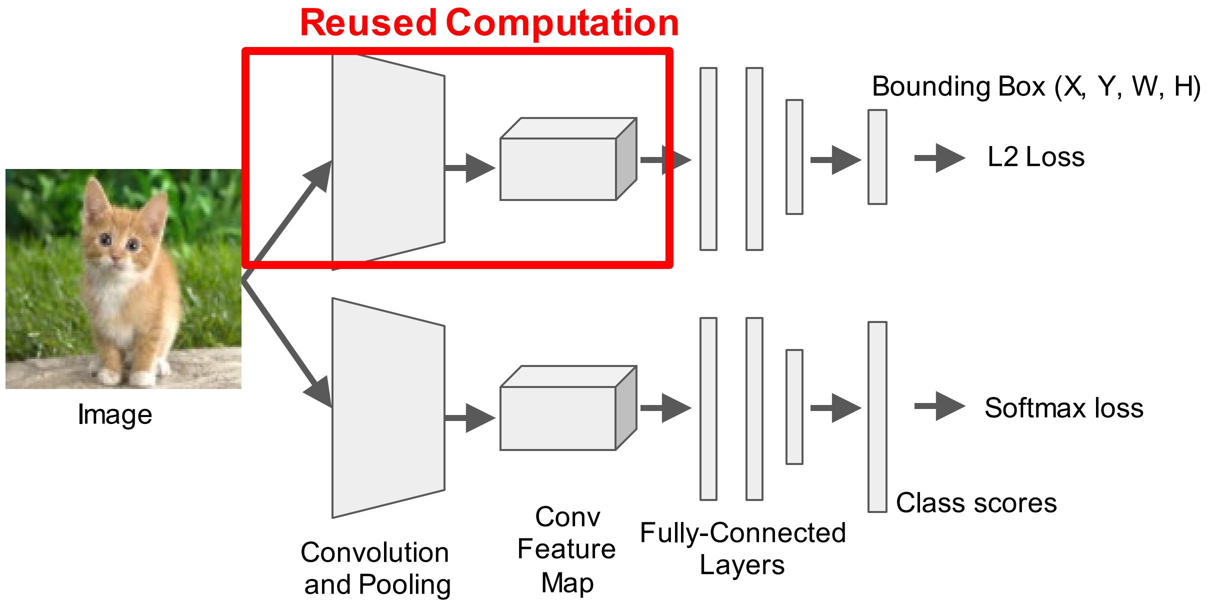
Input: Image
Output: Bounding Box
Evaluation Metric: Intersection Over Union (IoU)



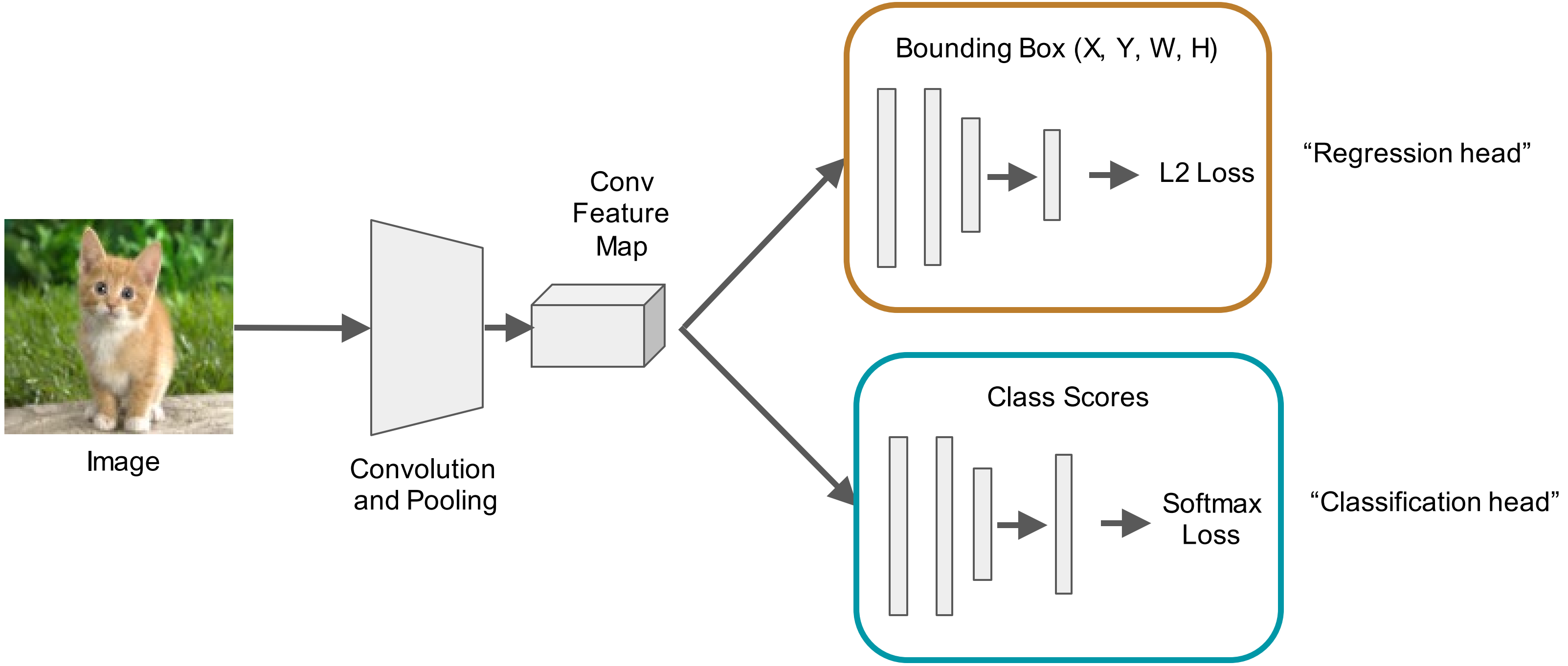
Simple Classification + Localization



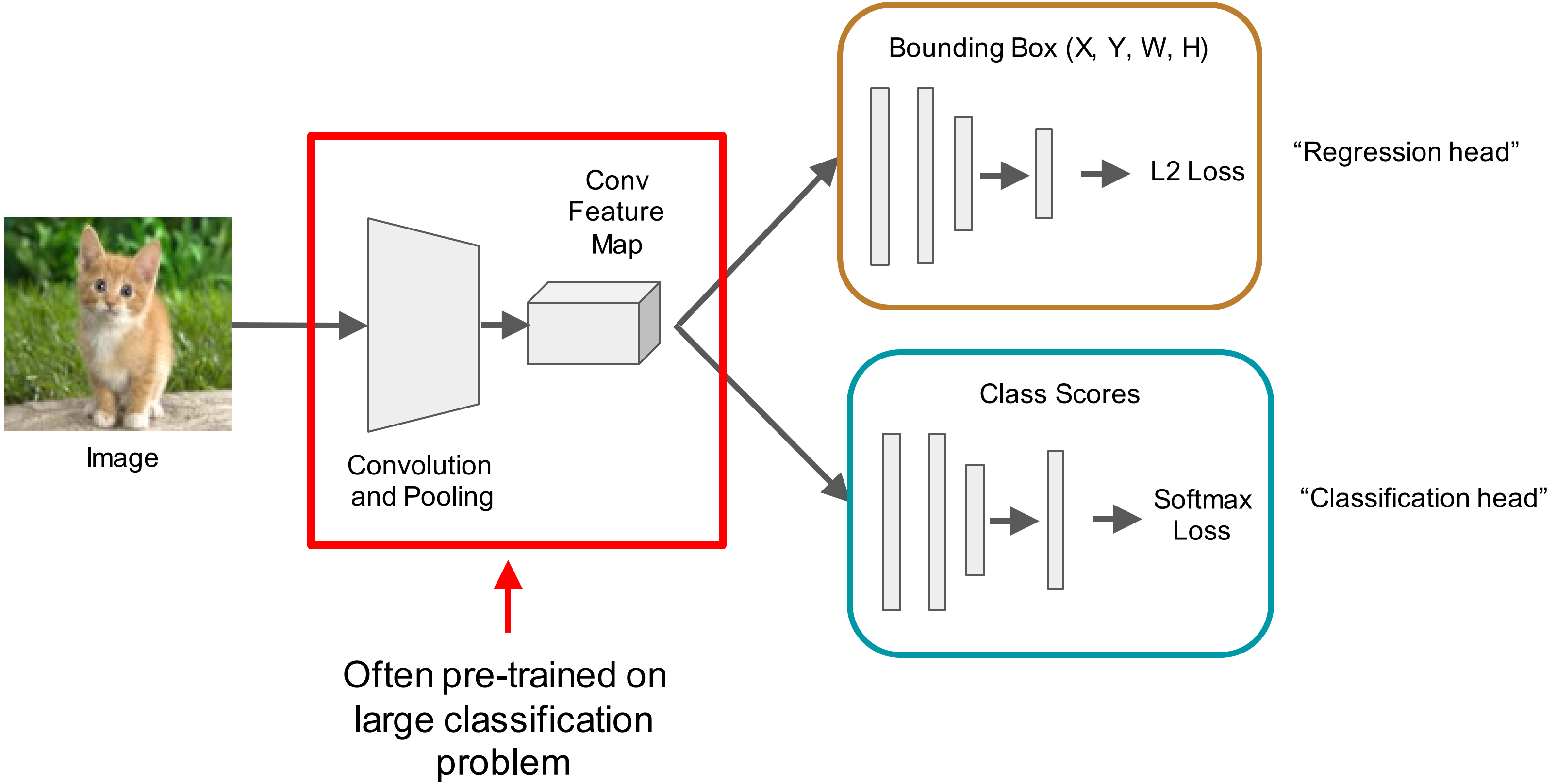
Simple Classification + Localization



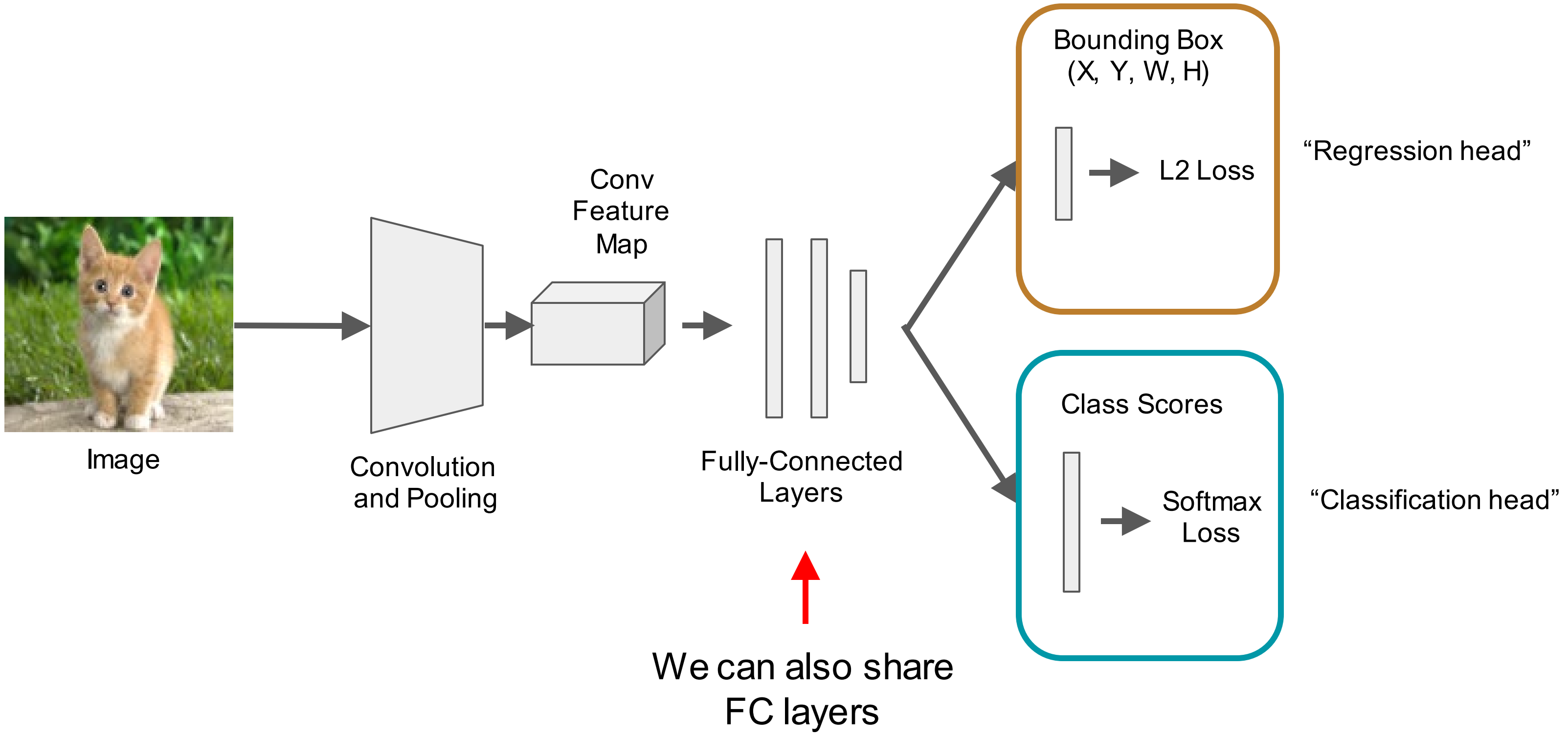
Simple Classification + Localization



Simple Classification + Localization

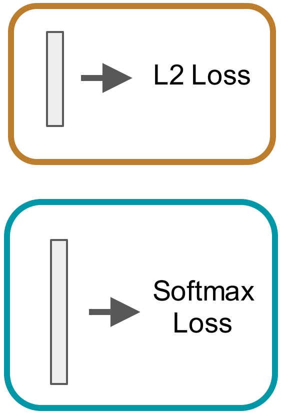


Simple Classification + Localization



Per-Class vs. Class Agnostic Regression

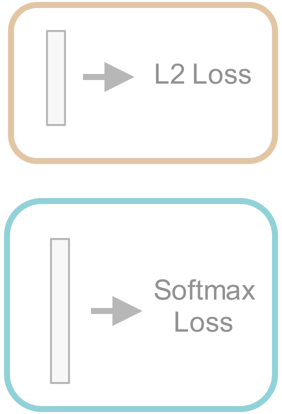
Class Agnostic



Bounding Box
(X, Y, W, H)
4 Numbers

Class Scores
1 Class selected
(# of Classes output)

Per-Class

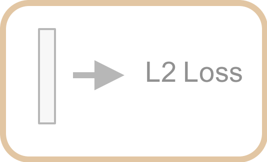


One bounding box for **EACH** class
(# of Classes) x 4 outputs

Class Scores
1 Class selected
(# of Classes output)

Per-Class vs. Class Agnostic Regression

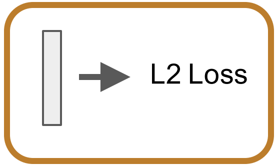
Class Agnostic



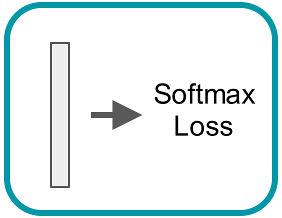
Bounding Box
(X, Y, W, H)
4 Numbers

Class Scores
1 Class selected
(# of Classes output)

Per-Class

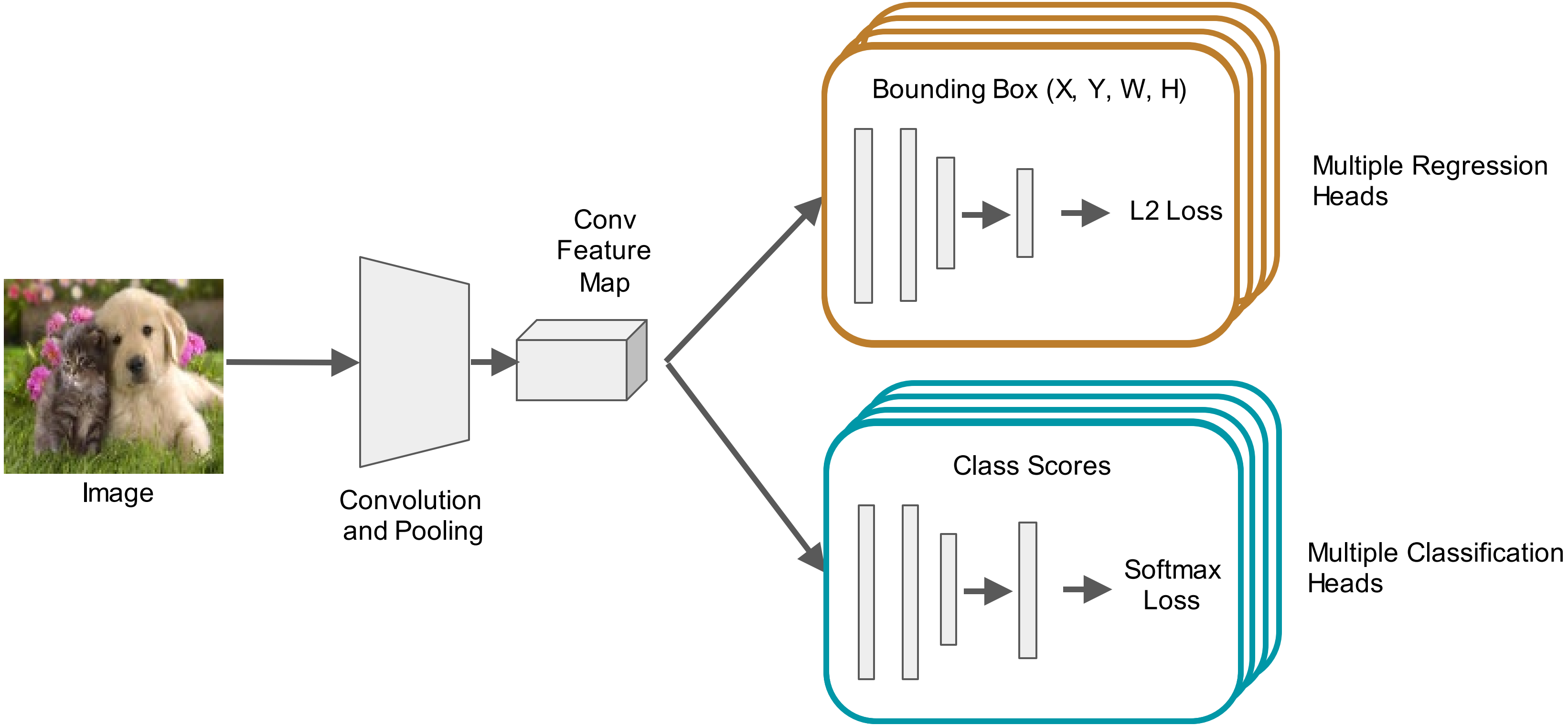


One bounding box for
EACH class
(# of Classes) x 4 outputs



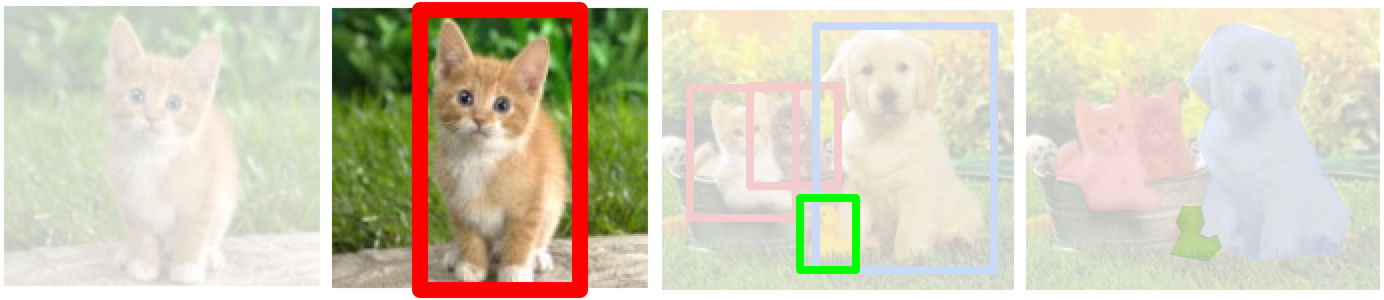
Class Scores
1 Class selected
(# of Classes output)

Aside: Localizing Multiple Objects



Computer Vision Tasks

Classification | Classification + Localization | Object Detection | Instance Segmentation



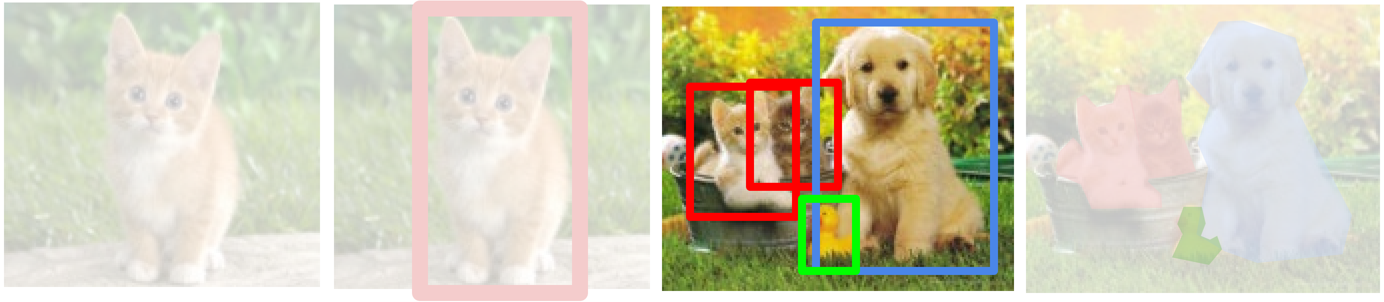
Computer Vision Tasks

Classification

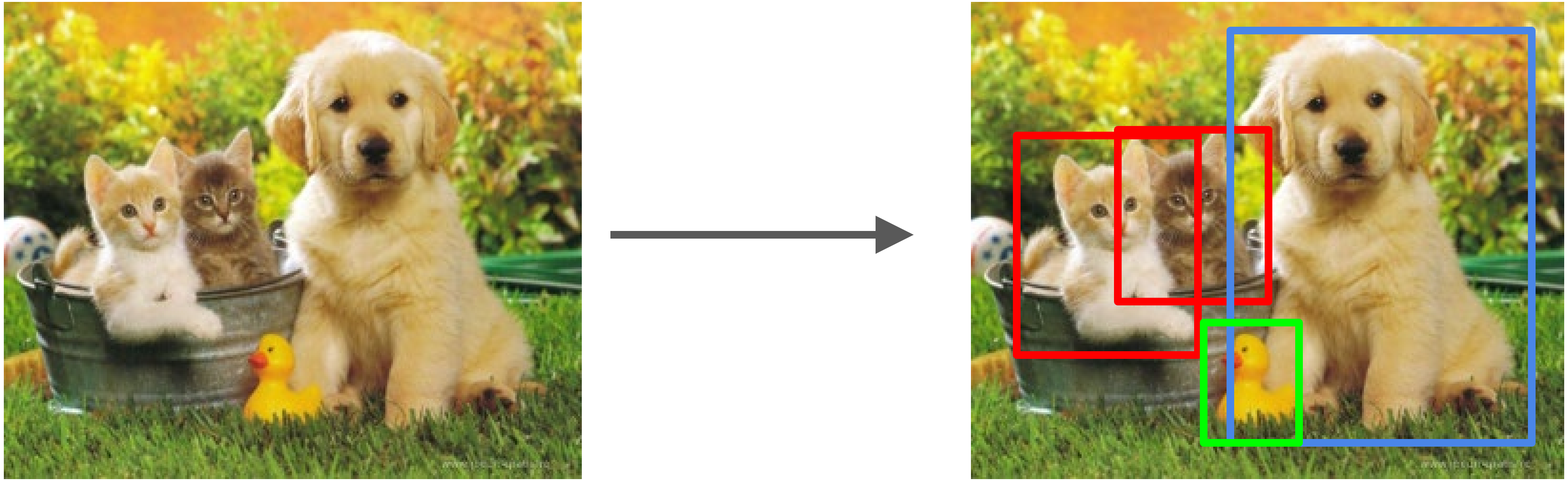
Classification
+ Localization

Object Detection

Instance
Segmentation

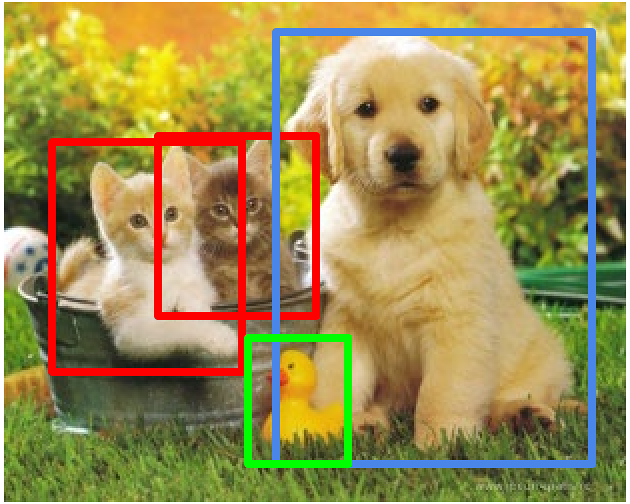


Object Detection



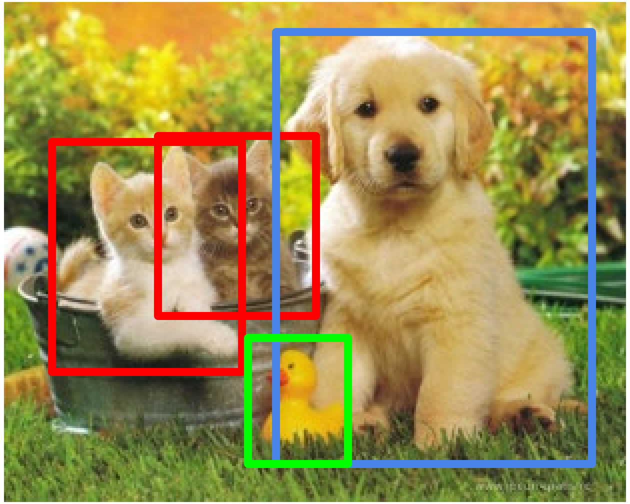
Object Detection

- We use a metric called “mean average precision” (mAP)



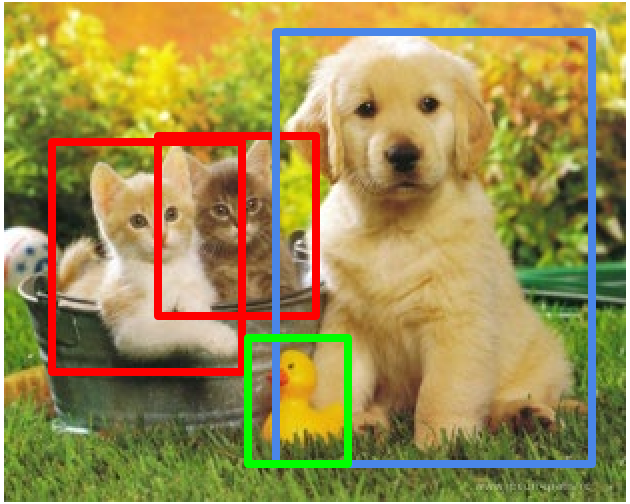
Object Detection

- We use a metric called “mean average precision” (mAP)
- Compute average precision (AP) separately for each class, then average over classes



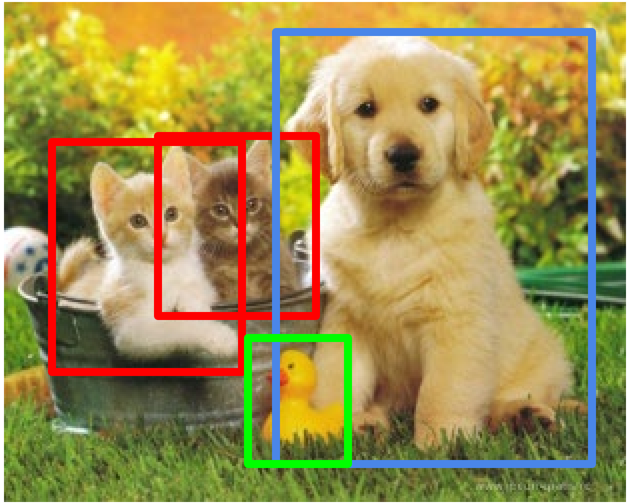
Object Detection

- We use a metric called "mean average precision" (mAP)
- Compute average precision (AP) separately for each class, then average over classes
- A detection is a true positive if it has IoU (Intersection over Union) with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)



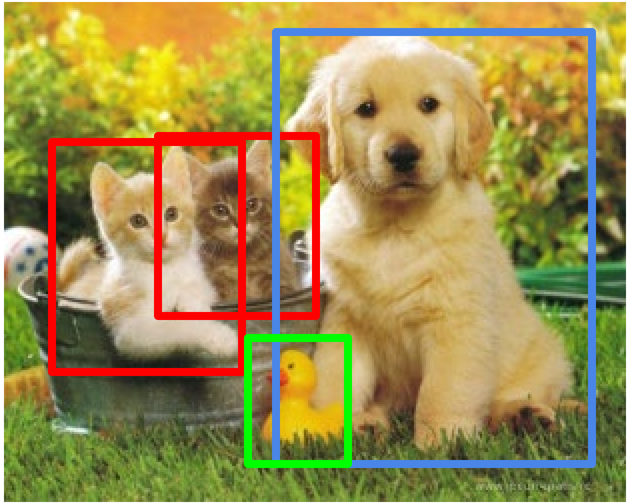
Object Detection

- We use a metric called "mean average precision" (mAP)
- Compute average precision (AP) separately for each class, then average over classes
- A detection is a true positive if it has IoU (Intersection over Union) with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)
- Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve



Object Detection

- We use a metric called “mean average precision” (mAP)
- Compute average precision (AP) separately for each class, then average over classes
- A detection is a true positive if it has IoU (Intersection over Union) with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)
- Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve
- TL;DR mAP is a number from 0 to 100; high is good



Detection using Regression?

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

= 16 numbers

Detection using Regression?

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

= 16 numbers

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

= 8 numbers

Detection using Regression?

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

= 16 numbers

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

= 8 numbers

CAT, (x, y, w, h)
CAT, (x, y, w, h)
…

= Many Numbers

Detection using Regression?

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

= 16 numbers

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

= 8 numbers

CAT, (x, y, w, h)
CAT, (x, y, w, h)
…

= Many Numbers

Need variable sized outputs

Detection using Classification?

CAT? NO

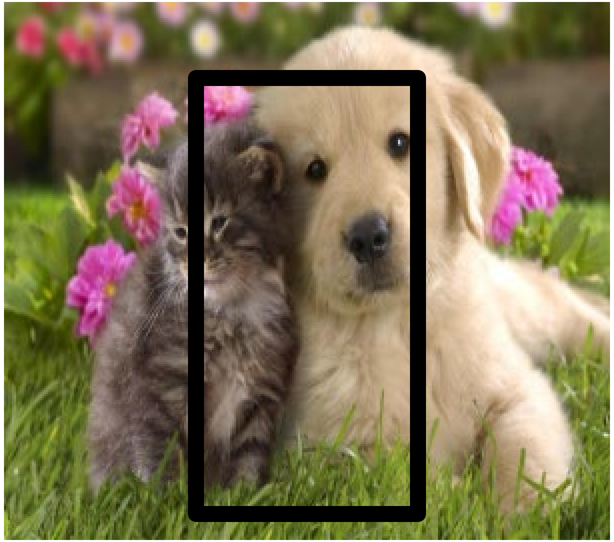
DOG? NO

Detection using Classification?

CAT? YES!

DOG? NO

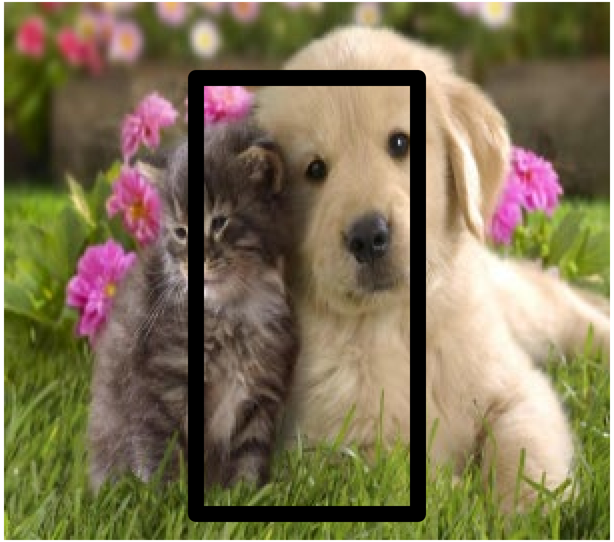
Detection using Classification?



CAT? NO

DOG? NO

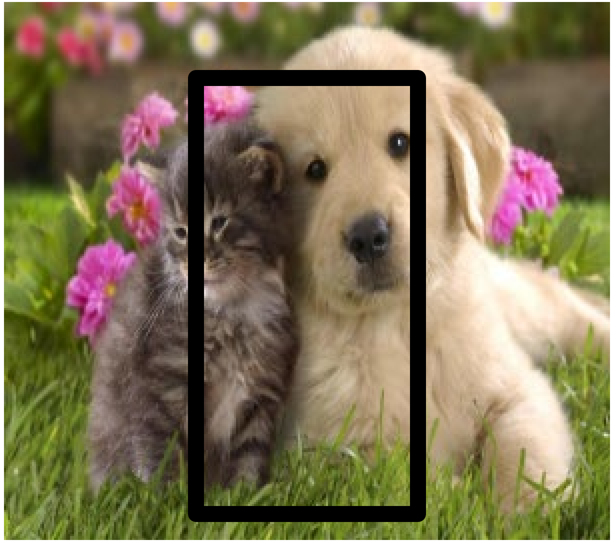
Detection using Classification?



Problem: Need to test many positions and scales

Solution: If your classifier is fast enough, just do it

Detection using Classification?



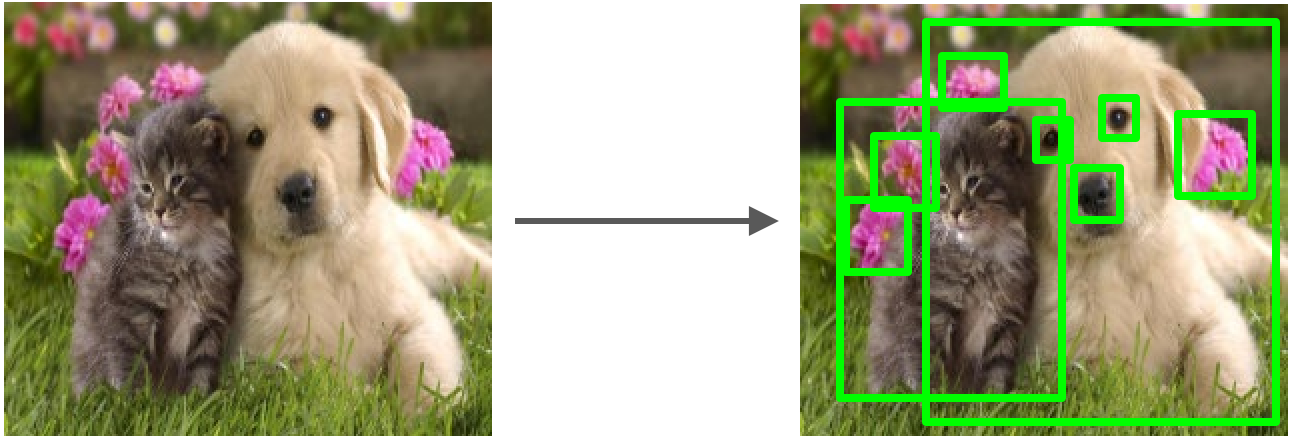
Problem: Need to test many positions and scales

~~**Solution:** If your classifier is fast enough, just do it~~

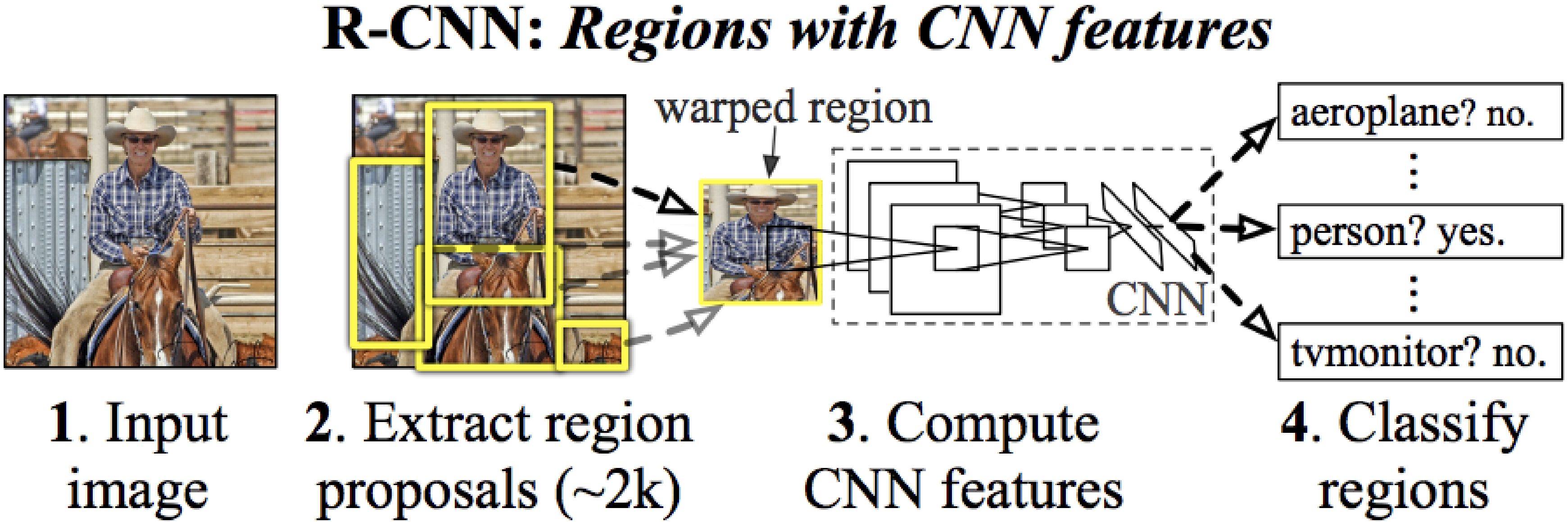
Solution: Only look at promising regions of the image

Region Proposals

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



Classification + Region Proposals: R-CNN

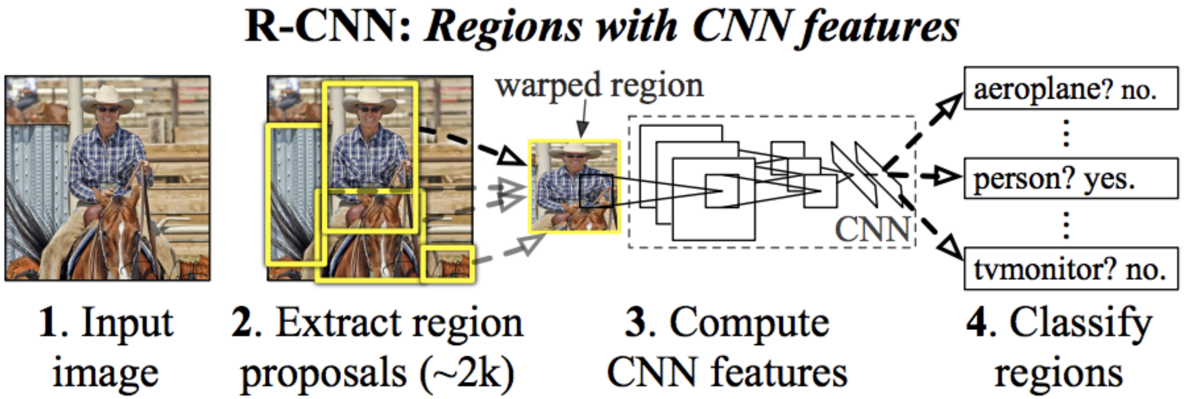


Girshick et. al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. CVPR 2014

Classification + Region Proposals: R-CNN

Issues

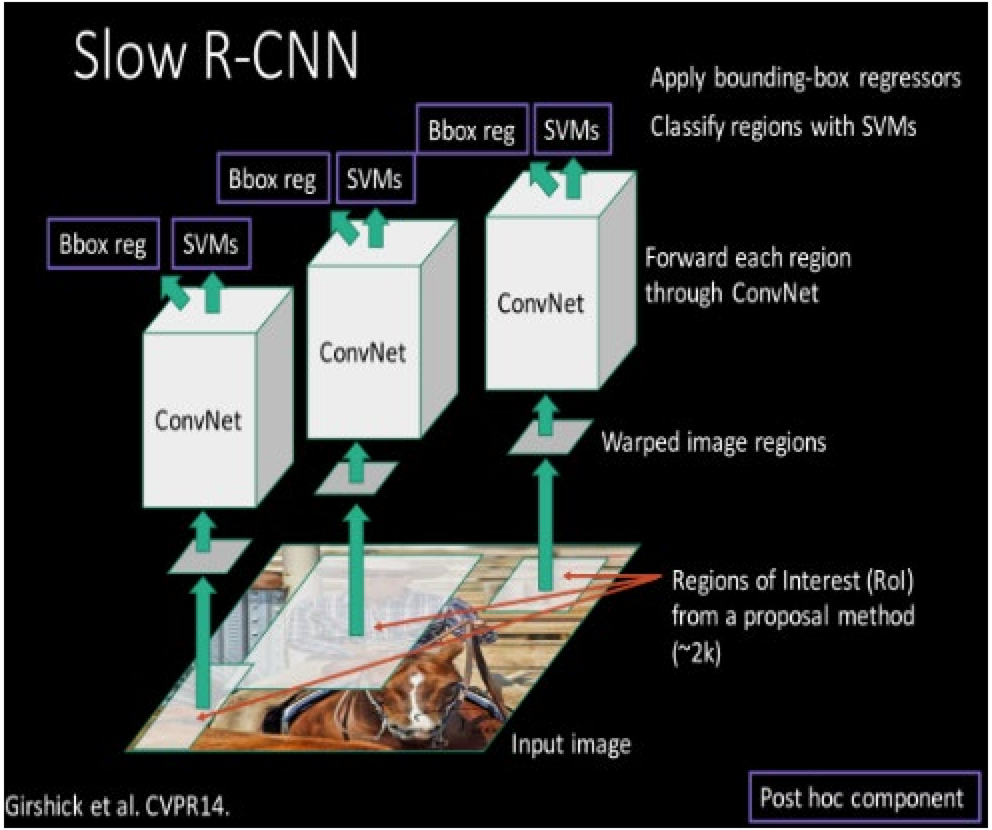
- Finding region proposals can be hard/time consuming
- Classifying each part of the image is time/space consuming



Girshick et. al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. CVPR 2014

Fast R-CNN

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

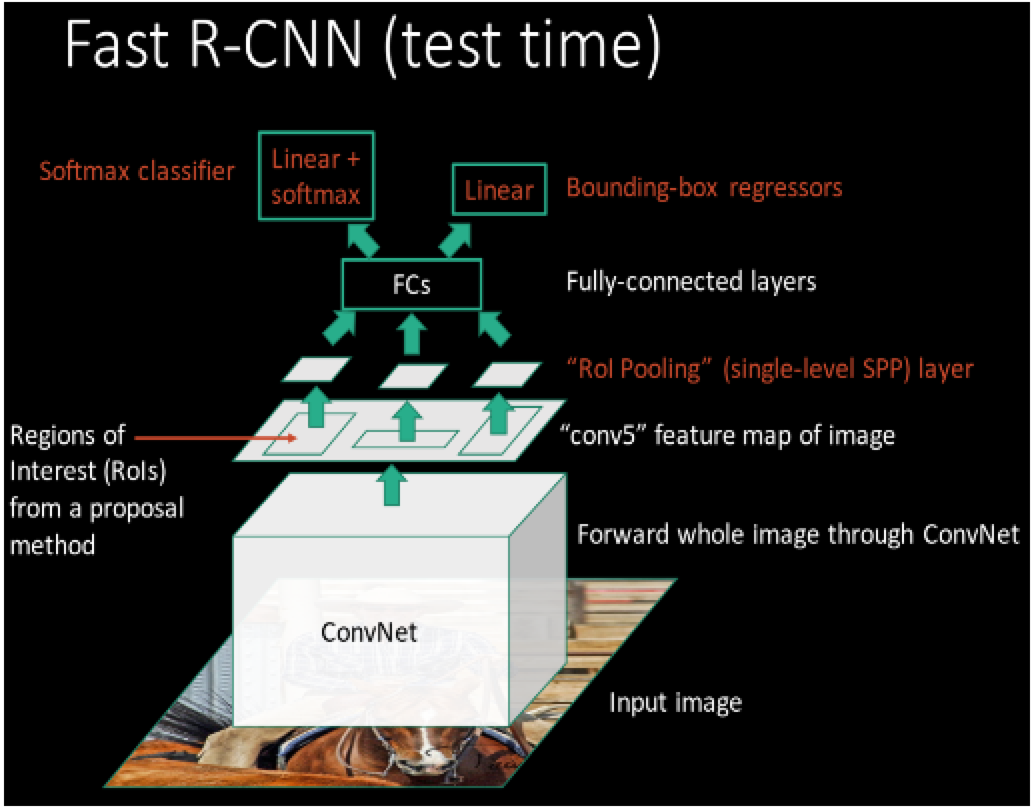


Girshick, "Fast R-CNN", ICCV 2015. Slide credit: Ross Girschick

Fast R-CNN

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



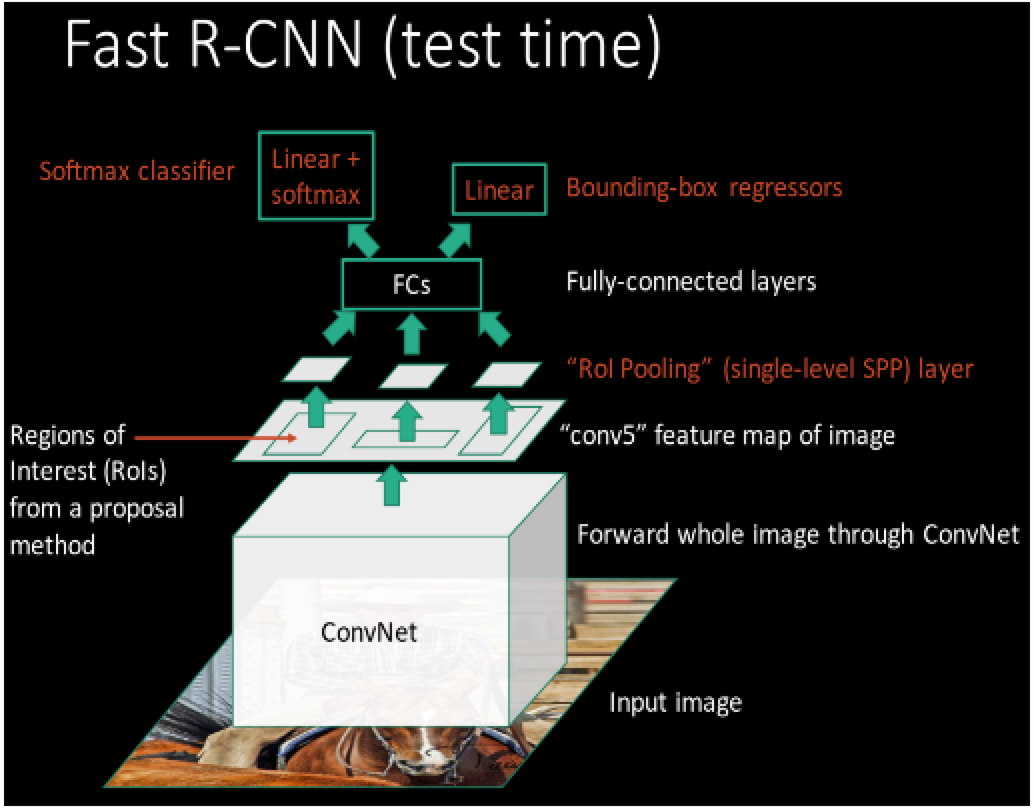
Girschick, "Fast R-CNN", ICCV 2015. Slide credit: Ross Girschick

Fast R-CNN

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

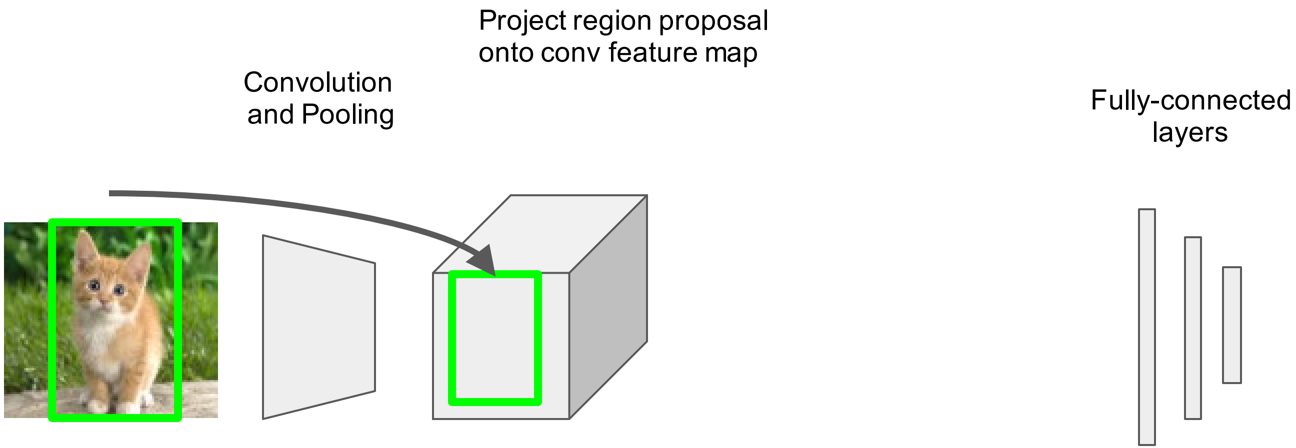
R-CNN Problem #3:
Complex training pipeline

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



Girschick, "Fast R-CNN", ICCV 2015. Slide credit: Ross Girschick

Fast R-CNN

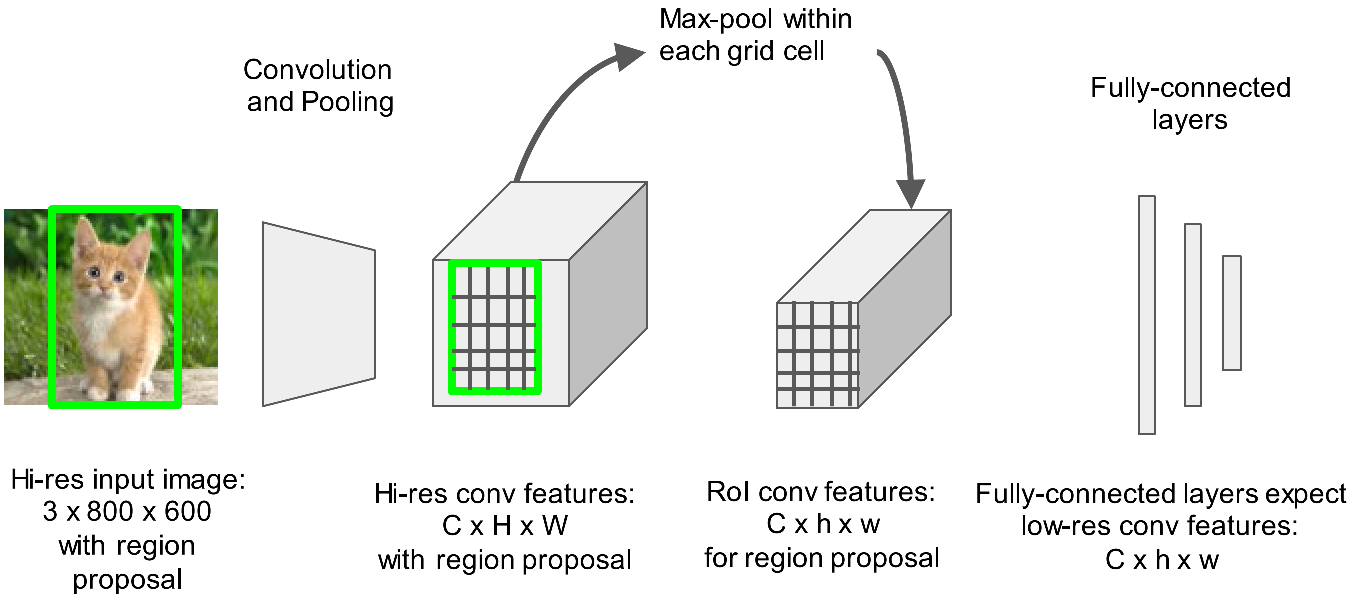


Hi-res input image:
3 x 800 x 600
with region
proposal

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


Max-pool within each grid cell
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
RoI conv features: C x h x w for region proposal
Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN

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

Region proposals are still slow!

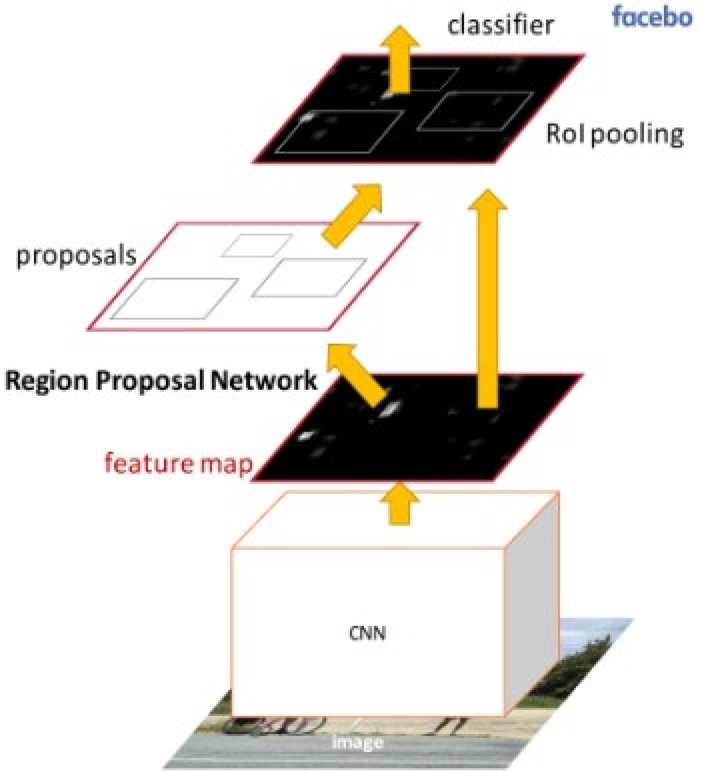
Using VGG-16 CNN on Pascal VOC 2007 dataset

Faster R-CNN

Insert a **Region Proposal Network (RPN)** after the last convolutional layer

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



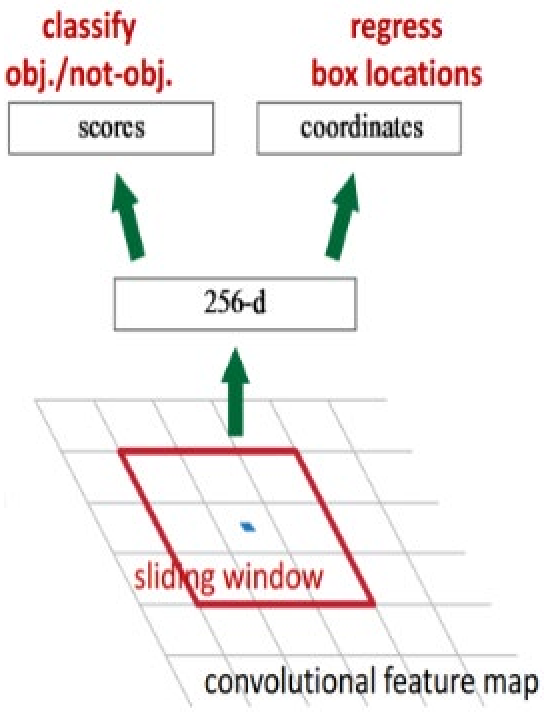
Faster R-CNN

Slide a small window on the feature map

Build a small network for:
• classifying object or not-object, and
• 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



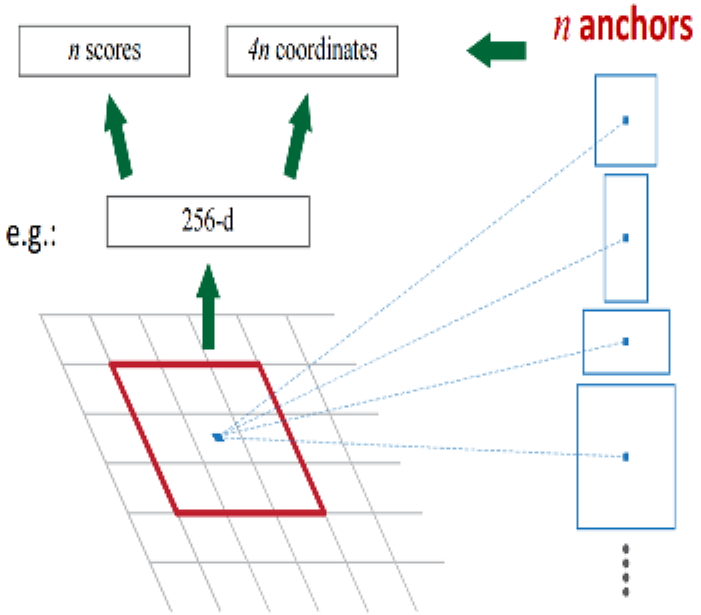
Faster R-CNN

Use **N anchor boxes** at each location

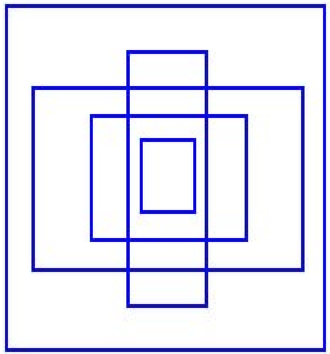
Anchors are **translation invariant**: use the same ones at every location

Regression gives offsets from anchor boxes

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

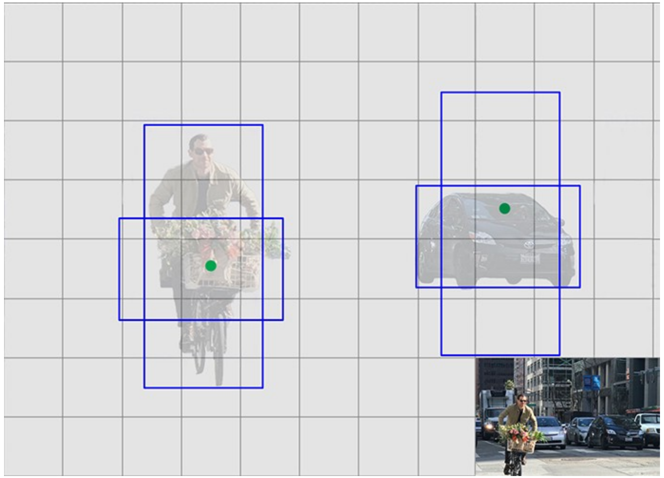


Faster R-CNN



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

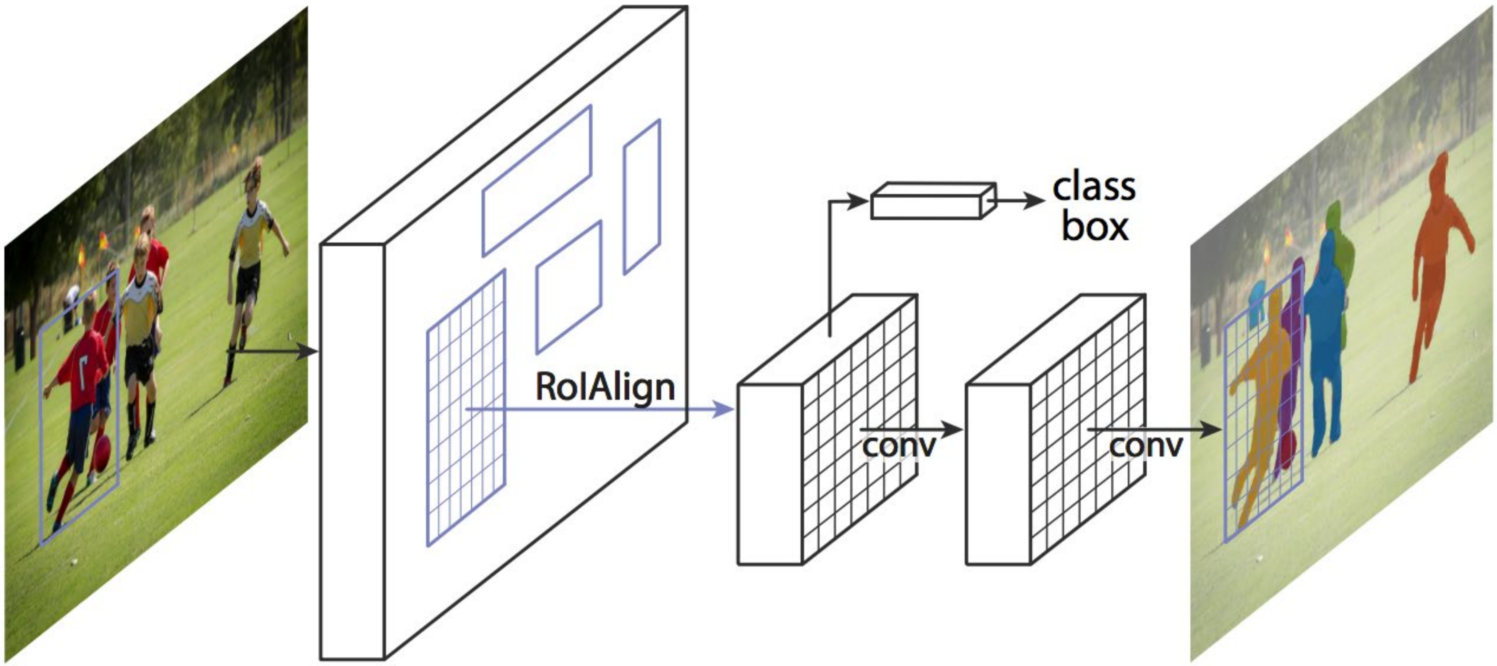
Image Credit: https://medium.com/@andersasac/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9

Faster R-CNN

	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**

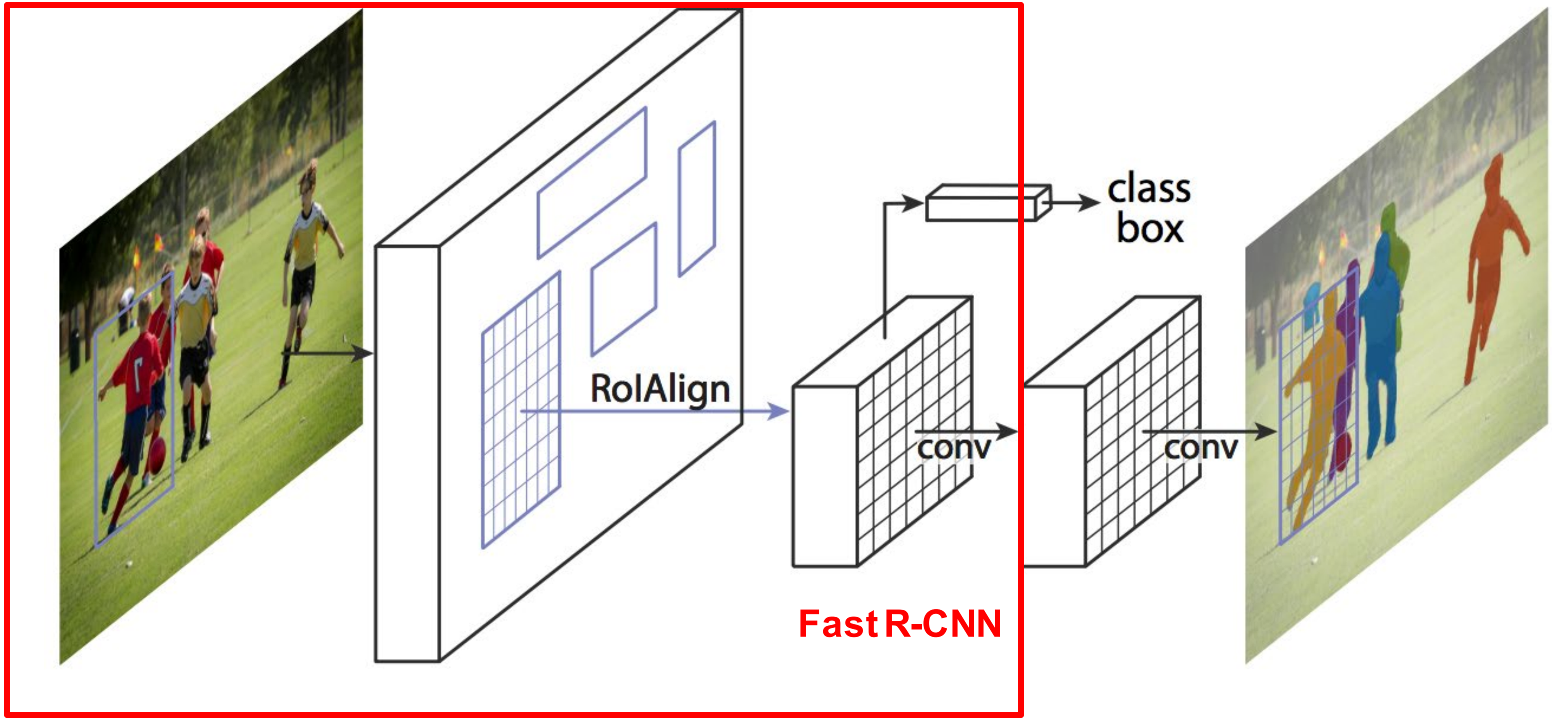
Slide credit: Kaiming He

Aside: Mask R-CNN for Semantic Segmentation



He et al, "Mask R-CNN", arXiv 2018

Aside: Mask R-CNN



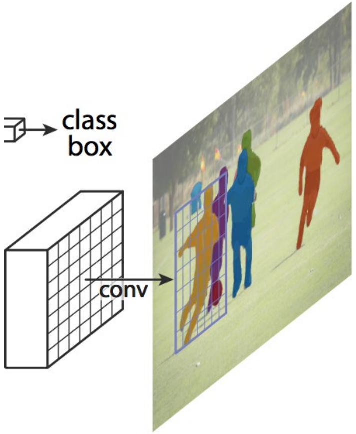
He et al, "Mask R-CNN", arXiv 2018

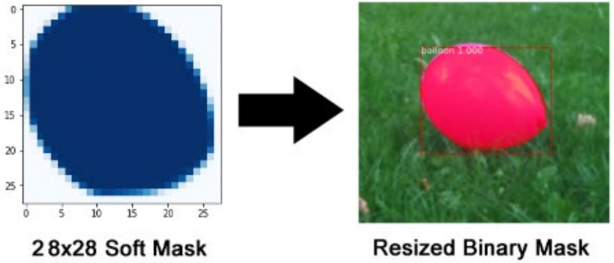
Aside: Mask R-CNN

Mask branch is a CNN that takes positive regions selected by the ROI classifier, and generates soft binary masks for them.

Generated masks are low-resolution.

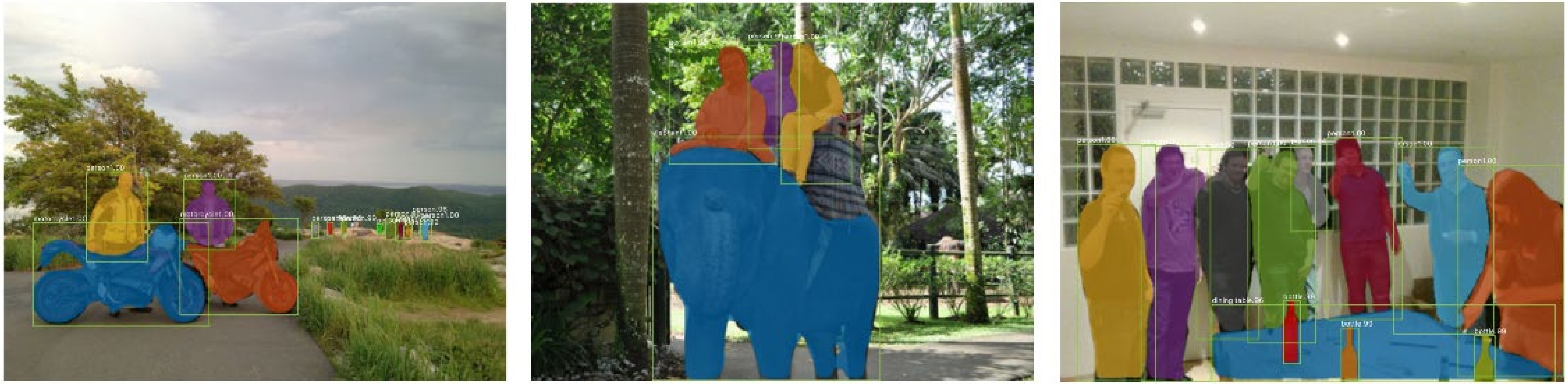
During training, we scale down to compute the loss, and during inference, we scale up to compute the mask.





He et al, "Mask R-CNN", arXiv 2018

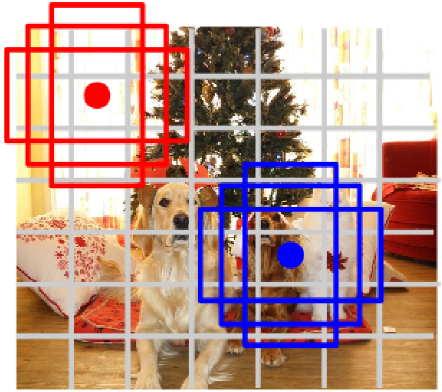
Aside: Mask R-CNN



He et al, “Mask R-CNN”, arXiv 2018

State of the art Models: Single Shot Detection

Input image
3 x H x W



Divide image into grid
7 x 7

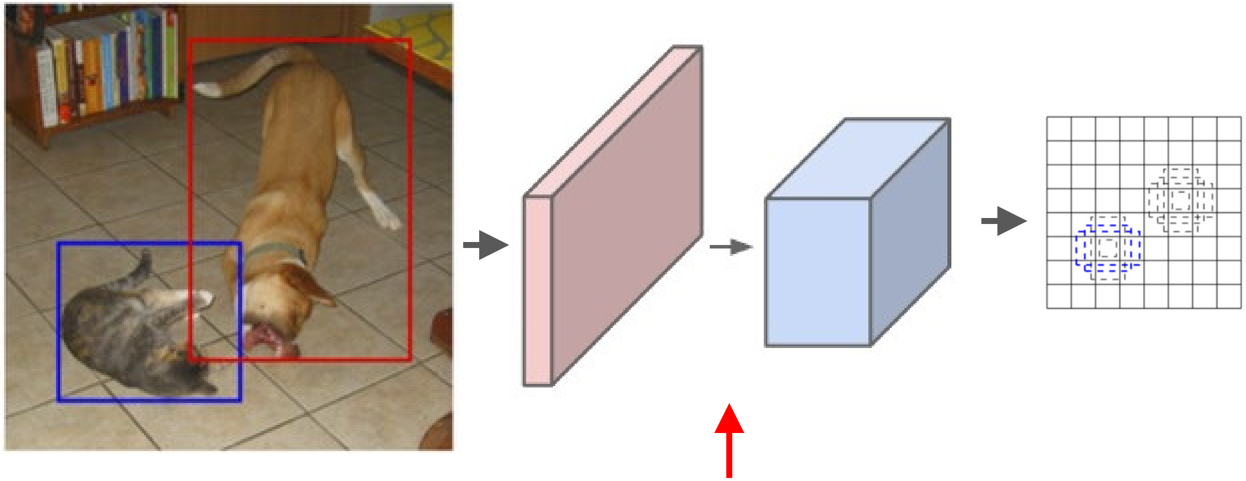
Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

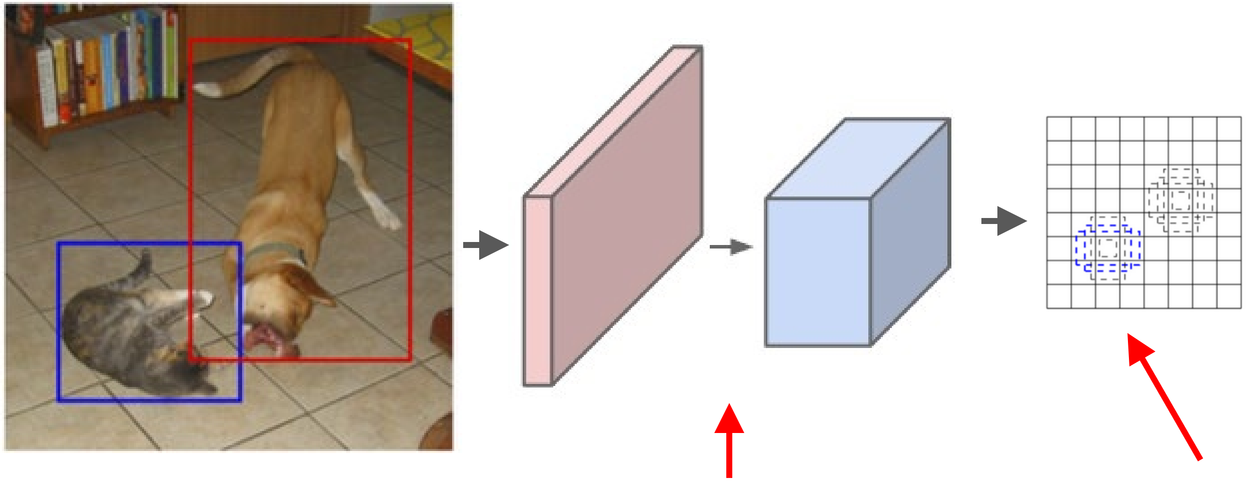
Redmon et al, “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016
Liu et al, “SSD: Single-Shot MultiBox Detector”, ECCV 2016

State of the art Models: Single Shot Detection



Extract Features with
Convolutional Network

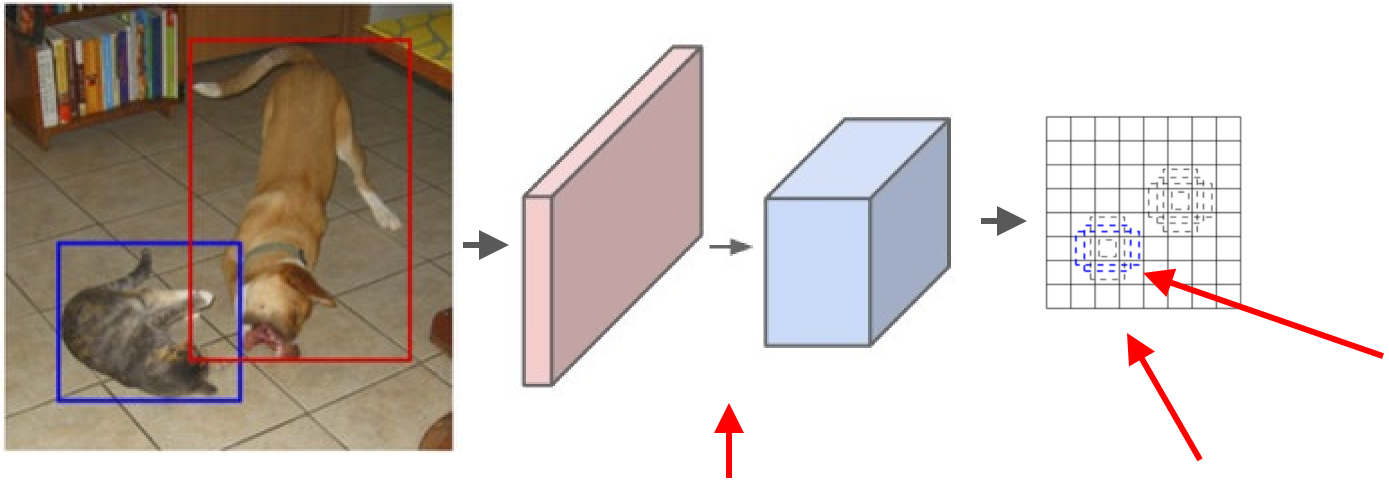
State of the art Models: Single Shot Detection



Extract Features with Convolutional Network

Split the features into equal grids. Predict if an element is in each anchor box

State of the art Models: Single Shot Detection

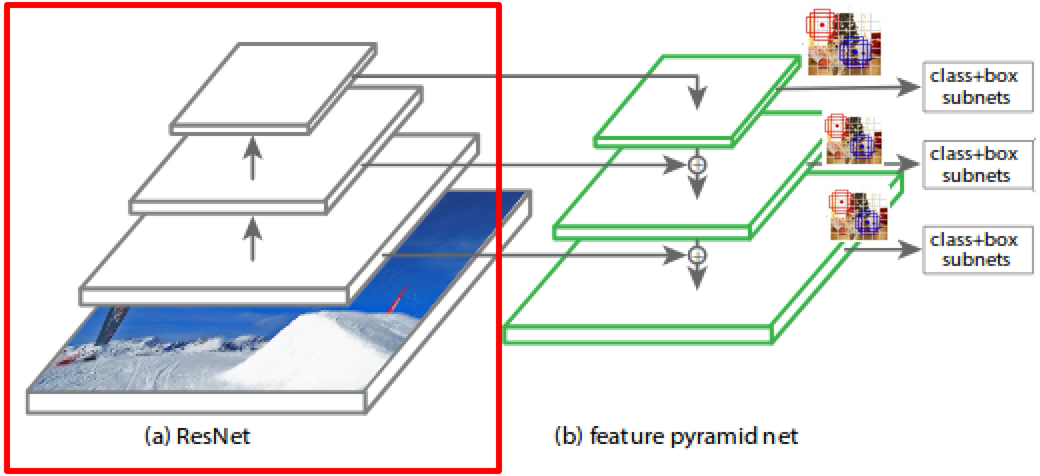


Extract Features with Convolutional Network

Split the features into equal grids. Predict if an element is in each anchor box

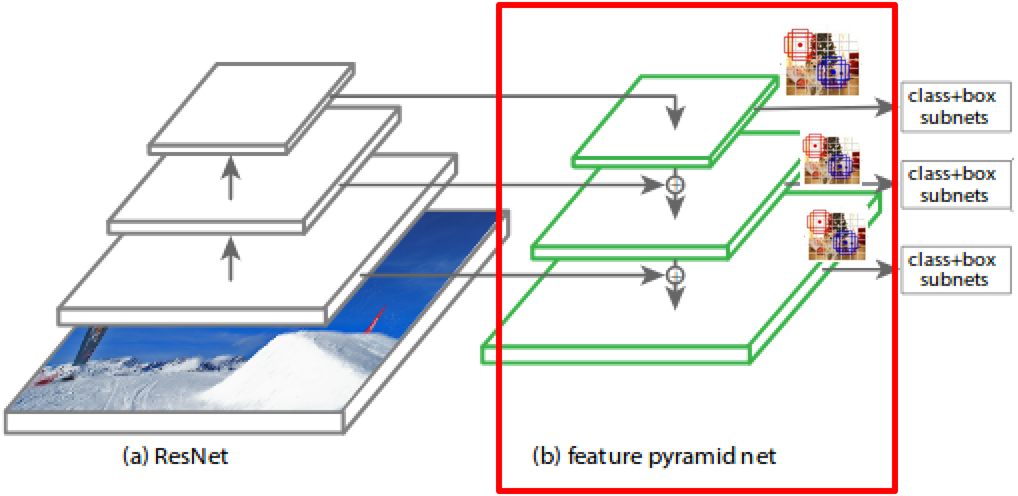
Predict a class for each anchor box

State of the art Models: RetinaNet



Step 1: Run the forward pass of a ResNet/Convolutional model

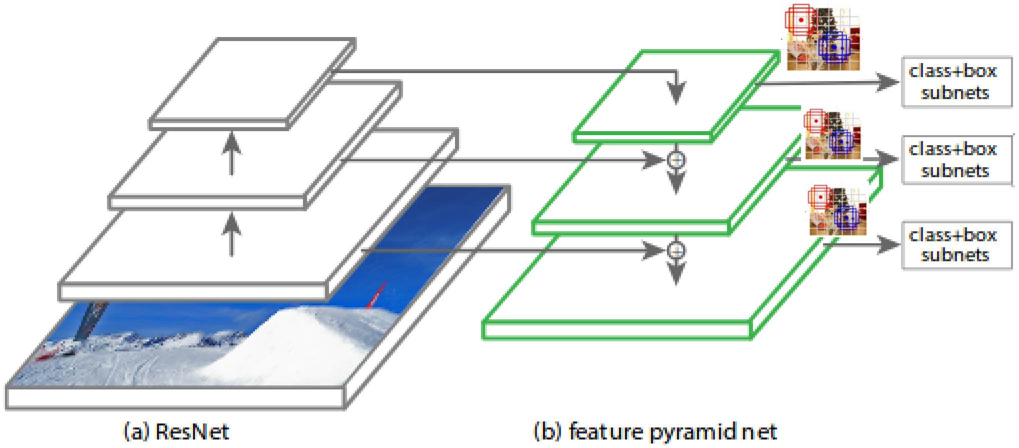
State of the art Models: RetinaNet

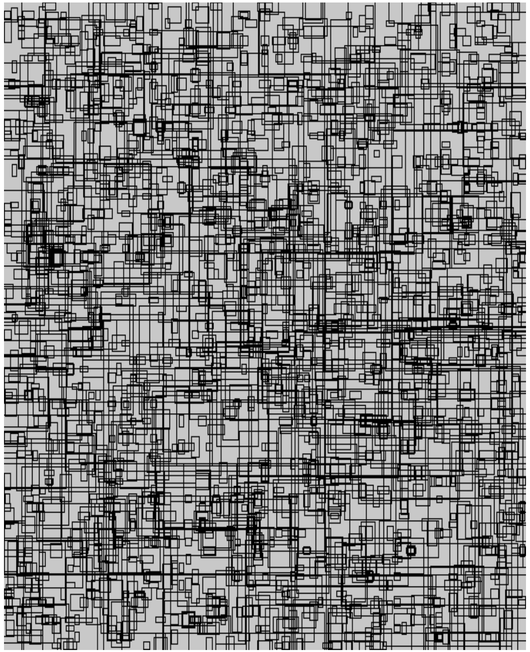


Step 1: Run the forward pass of a ResNet/Convolutional model

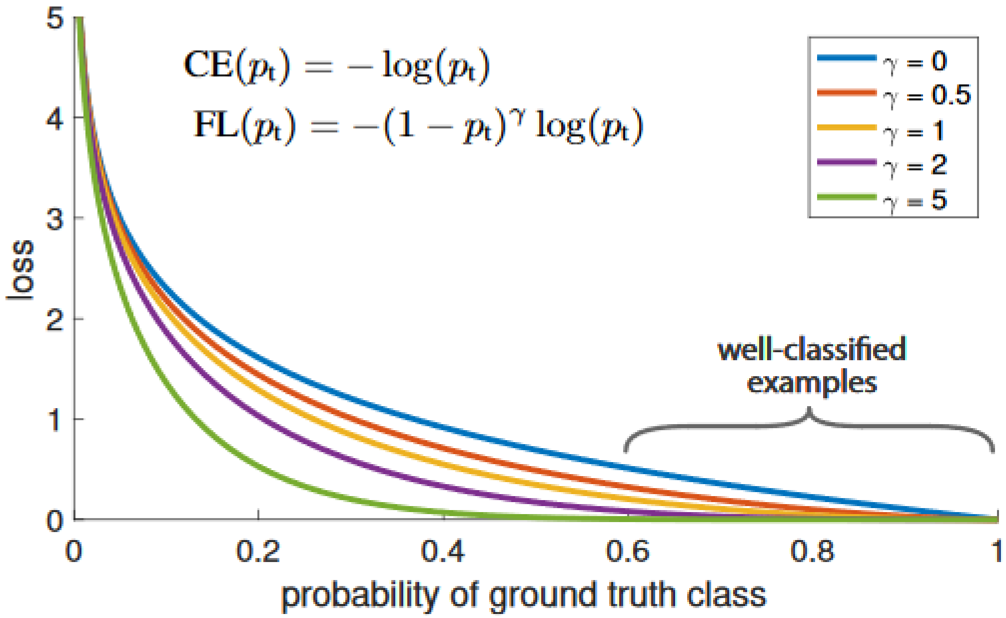
Step 2: At each level of down-sampling, do single-shot detection.

State of the art Models: RetinaNet





State of the art Models: RetinaNet (Focal Loss)



Lin et al, "Focal Loss for Dense Object Detection", arXiv 2018

State of the art Models

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Redmon et al, “You Only Look Once: Unified, Real-Time Object Detection”, arXiv 2015

Recap

Semantic Segmentation

- Classify at a pixel level
- Ignores "objectness" focuses on semantics
- Mask-RCNN/UNet for pixel-level semantics

Recap

Semantic Segmentation

- Classify at a pixel level
- Ignores “objectness” focuses on semantics
- Mask-RCNN/UNet for pixel-level semantics

Localization:

- Find a fixed number of objects (one or many)
- L2 regression from CNN features to box coordinates
- Much simpler than detection; consider it for your projects!
- Overfeat: Regression + efficient sliding window with FC -> conv conversion
- Deeper networks do better

Recap

Semantic Segmentation
- Classify at a pixel level
- Ignores "objectness" focuses on semantics
- Mask-RCNN/UNet for pixel-level semantics

Localization:
- Find a fixed number of objects (one or many)
- L2 regression from CNN features to box coordinates
- Much simpler than detection; consider it for your projects!
- Overfeat: Regression + efficient sliding window with FC -> conv conversion
- Deeper networks do better

Object Detection:
- Find a variable number of objects by classifying image regions
- Before CNNs: dense multiscale sliding window (HoG, DPM)
- Avoid dense sliding window with region proposals
- R-CNN: Selective Search + CNN classification / regression
- Fast R-CNN: Swap order of convolutions and region extraction
- Faster R-CNN: Compute region proposals within the network
- Deeper networks do better