Economics 2355: Object Detection

Melissa Dell

Harvard University

February 2021

Outline

Class Projects

- Central to the course is the class project
- The aim is to allow you to apply whatever methods seem most relevant to your research to a concrete problem
- If this helps you with a concrete project you are working on, this is great, but it isn't necessary. The key thing is to get hands on experiencing designing and implementing a deep learning pipeline

Class Projects

- You are free to work in groups - with others in the course or outside the course - or on your own. If you are working in groups, though, I encourage each person to have a distinct part of the pipeline that they are responsible for implementing, so you don't lose out on gaining hands on experience
- You need to consider what compute you are able to access when designing the project. Cloud GPUs can be pricey. Azure provides $100 in free credit to students

Assignments

- Report 1
  - What raw data do you hope to process?
  - What is the desired output?
  - Initial thoughts on relevant methods?
- Report 2
  - Provide a more specific road map for what your data processing pipeline will look like
- Report 3
  - Report progress on implementing each of the steps
  - Any bottlenecks or challenges you'd like advice on?
- Final Report: submit final product, which might include:
  - A data appendix (modeled after a paper) or documentation (modeled after a github documentation page)
  - A github repository
  - If you are building an open source tool, a webpage for it (can be built through github) can be central to generating interest

Outline

Computer Vision Problems

At the core of solving almost all CV problems are the CNNs we discussed previously. Recall that we use spatial filters (weight matrices) to connect layers of the network

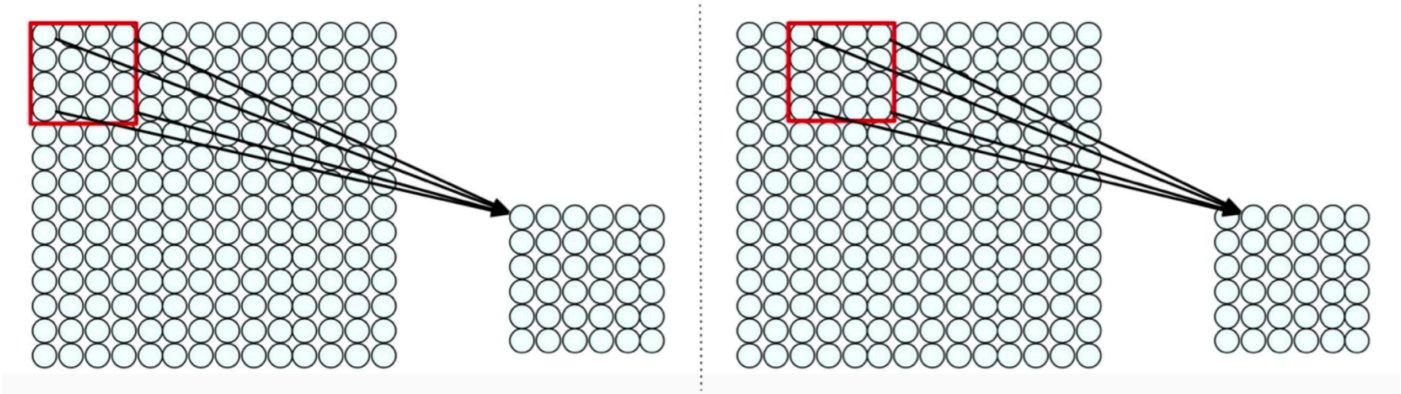
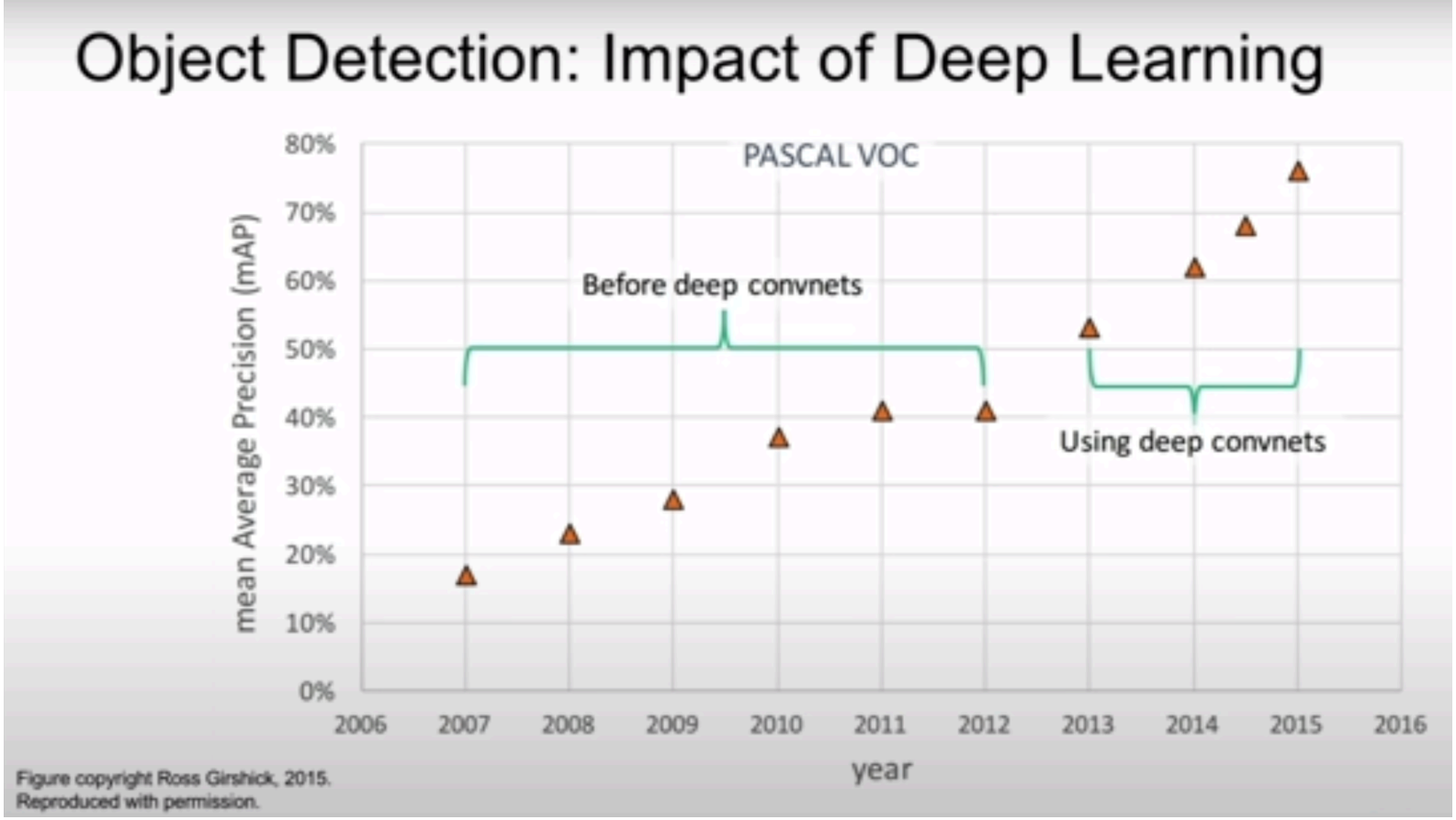


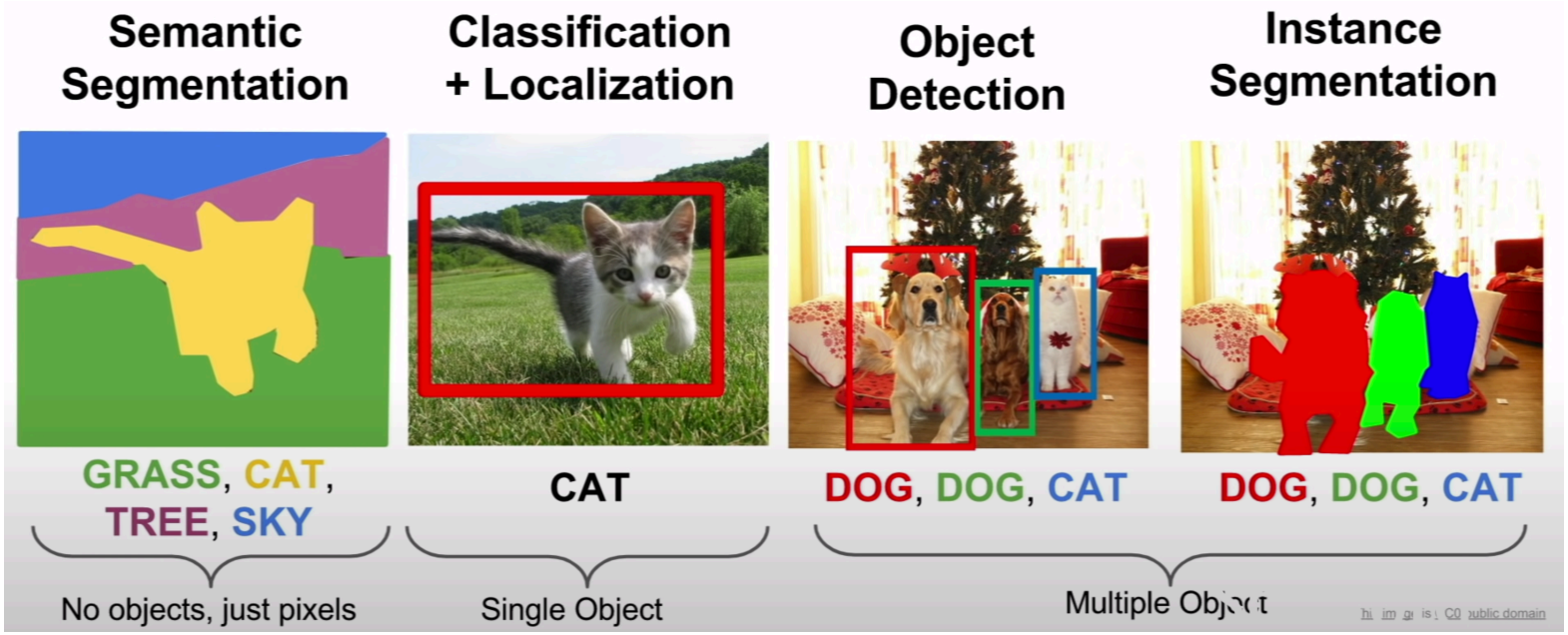
Image credit: Stanford CS 231n Lecture Notes

Object Detection and Deep Learning

As with other computer vision tasks, deep learning has transformed object detection and related CV problems



Computer Vision Problems



Stanford CS231n

Outline

Segmentation Overview

Semantic Segmentation

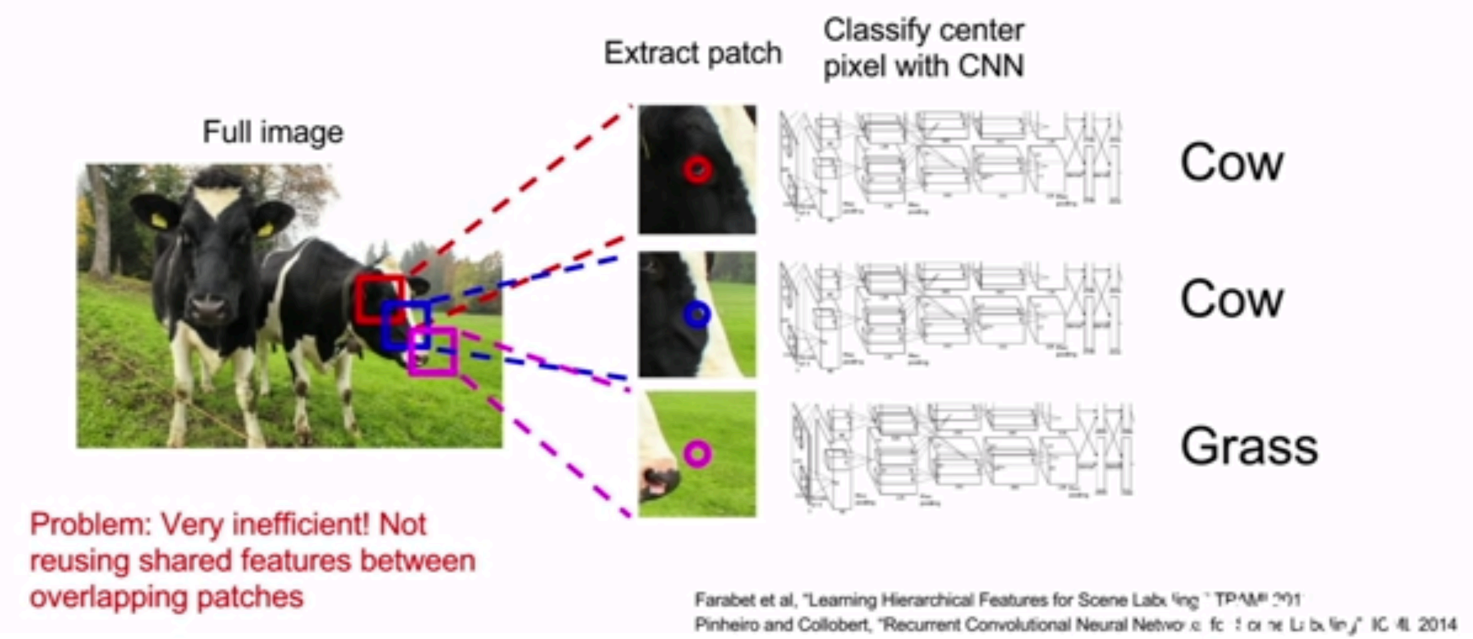
- Classify all pixels
- Fully convolutional models: downsample than upsample
- Learnable upsampling (transpose convolution)

Instance Segmentation

- Detect instance (individual object), generate mask
- Similar pipeline to object detection

Semantic Segmentation: Initial Approach

This is a quite old problem in computer vision. Initially, semantic segmentation used a sliding window approach to treat semantic segmentation as a classification problem:

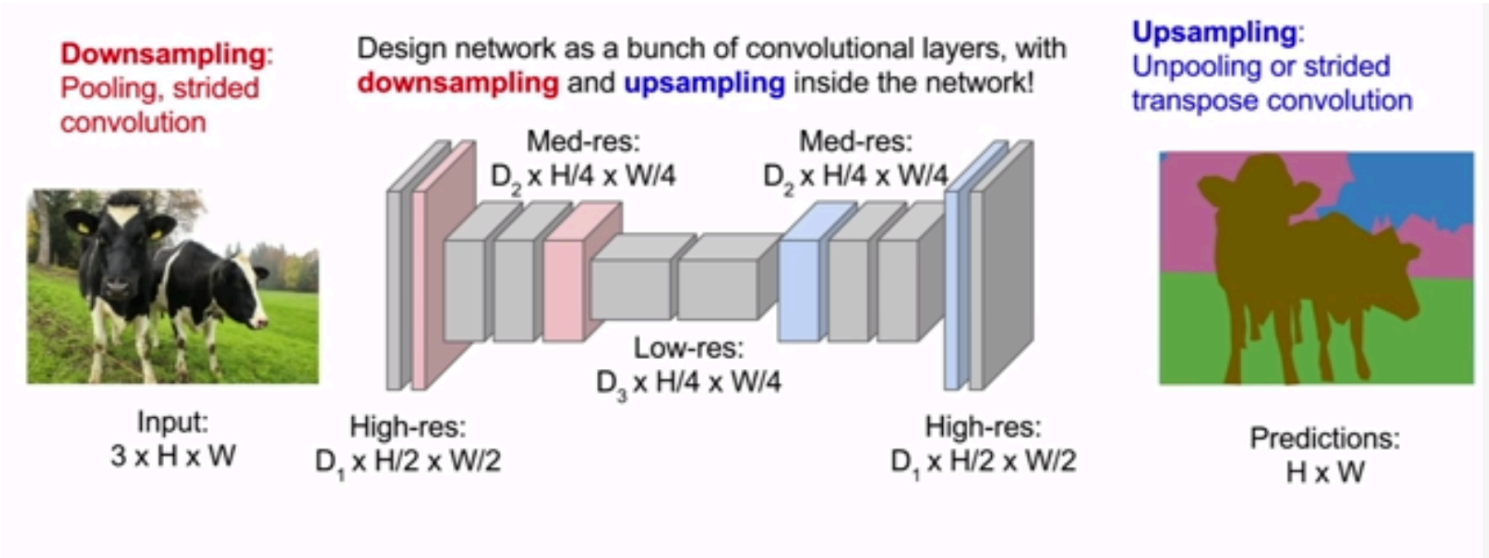


Stanford CS231n

Super costly; have to run each of the patches through the ConvNet classifier

Semantic Segmentation

Breakthrough: Stack a reverse CNN with a regular CNN (VGG, ResNet, etc). The regular CNN extracts the semantically strong features, the CNN in reverse scales this back up to the higher resolution to predict the mask (pixel-by-pixel classification loss).



Stanford CS231n

Regular Convolution

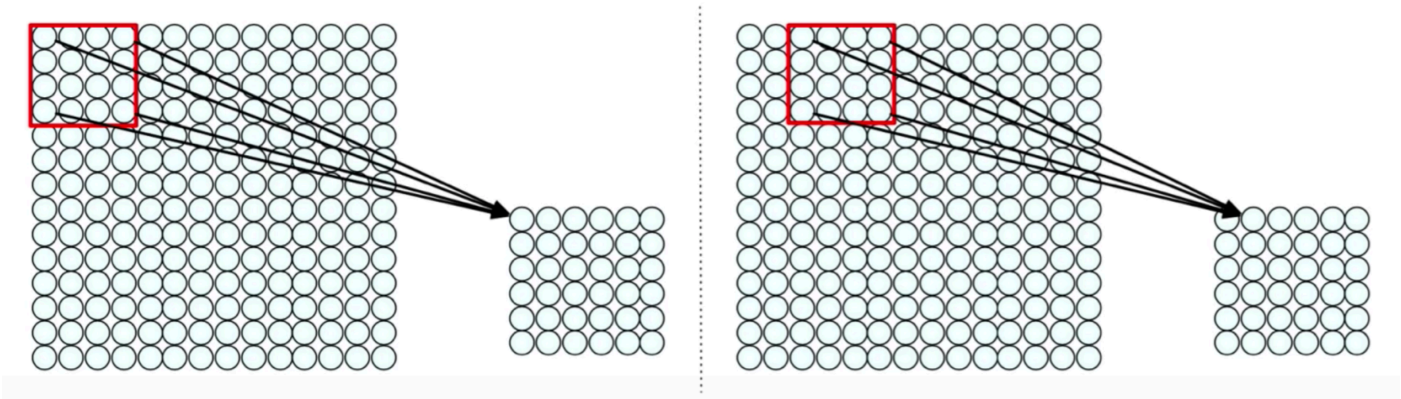
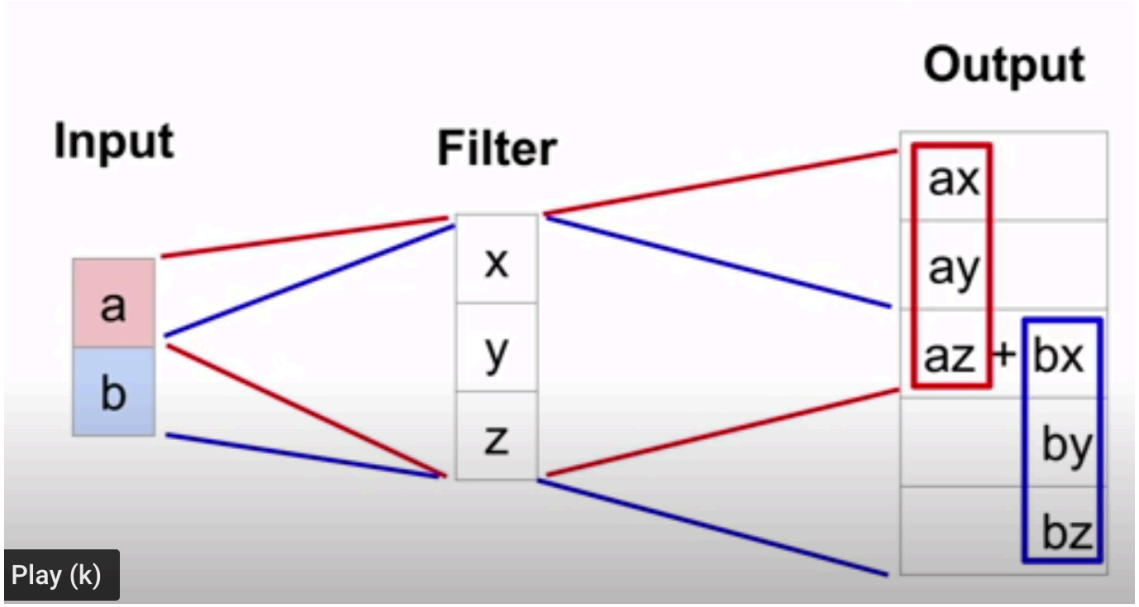


Image credit: Stanford CS 231n Lecture Notes

Transpose Convolution



Stanford CS231n

Often called fractionally strided convolution or deconvolution in the literature (but note deconvolution means something different in signal processing)

Outline

Localization

- We know that there is **only one** object per image, which we want to classify and determine its bounding box coordinates
- This problem doesn't seem that likely to arise for us in practice, but forms a foundation for how we approach more realistic problems
- Draws heavily on the machinery we already developed for classification, but add a fully connected layer at the end to predict object bounding box coordinates

Regression

In machine learning, we have two types of problems:

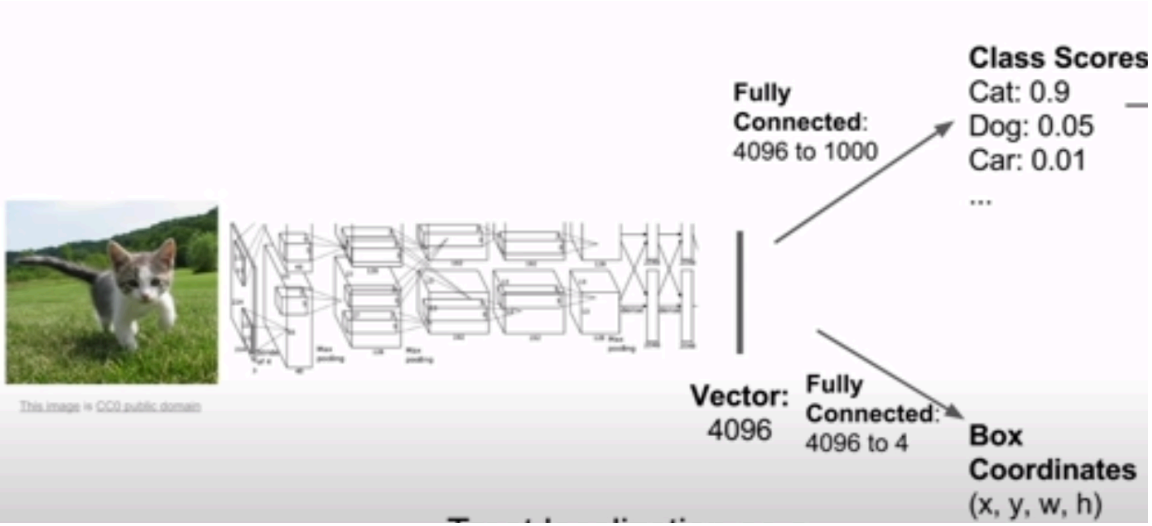
- Classification (discrete outputs)
- Regression (continuous outputs)

Regression is used in the same sense that we use it in economics, with a Euclidean (L2) loss: i.e. $\sum(\hat{x} - x)^2$ (sometimes people use other regression losses for continuous outputs, i.e. L1 or smoothed L1 but analogous)

Treat Localization as Regression

- Recall our desired output from localization is:
  - A class (i.e. picture of a cow, car, plane, etc)
  - Location (coordinates) of the single object that appears in the image
- The first is an image classification problem, that we already know how to solve
- The second is a regression problem (predicting four continuous numbers)

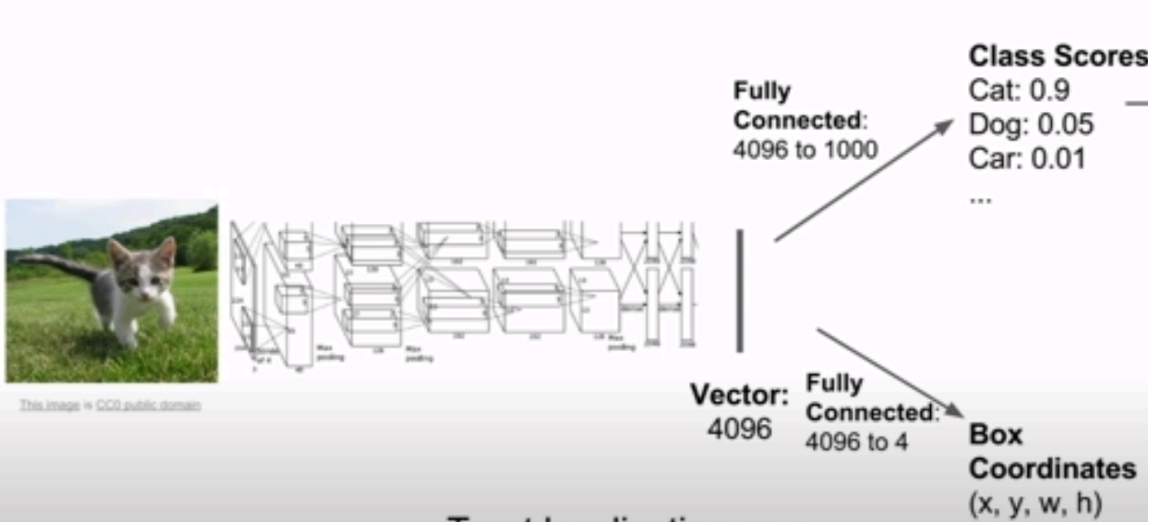
Treat Localization as a Regression Problem



Stanford CS231n

Recall our neural nets are legos analogy. We can add a regression head to the end of the CNN features extractor. The network now has two branches with fully connected layers, for classification and regression

Treat Localization as a Regression Problem



Stanford CS231n

The classification head will have fully connected weights that predict the N class probabilities from the features maps.

The regression head will predict 4 coordinates for that object. (Sometimes predict coordinates for all object classes but only use the ones for the correct class in the loss).

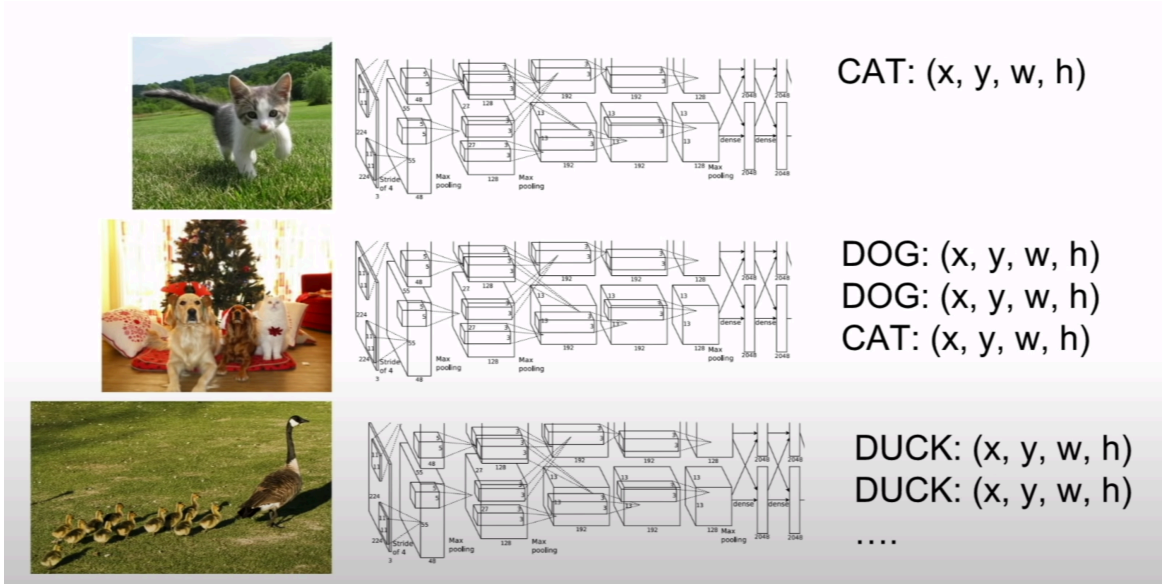
Regression/classification heads can be attached after a FC layer or after the last conv layer

Training

- Combine two loss functions into a multi-task loss
  - Classification loss (i.e. softmax)
  - Regression loss (i.e. L2): sometimes predict box for all categories but only apply to correct one
- Have a hyperparameter that tells how to weight the sum of two losses (changes value of loss so can't evaluate this hyperparameter using the value of the loss)

Outline

Can we approach object detection as localization?



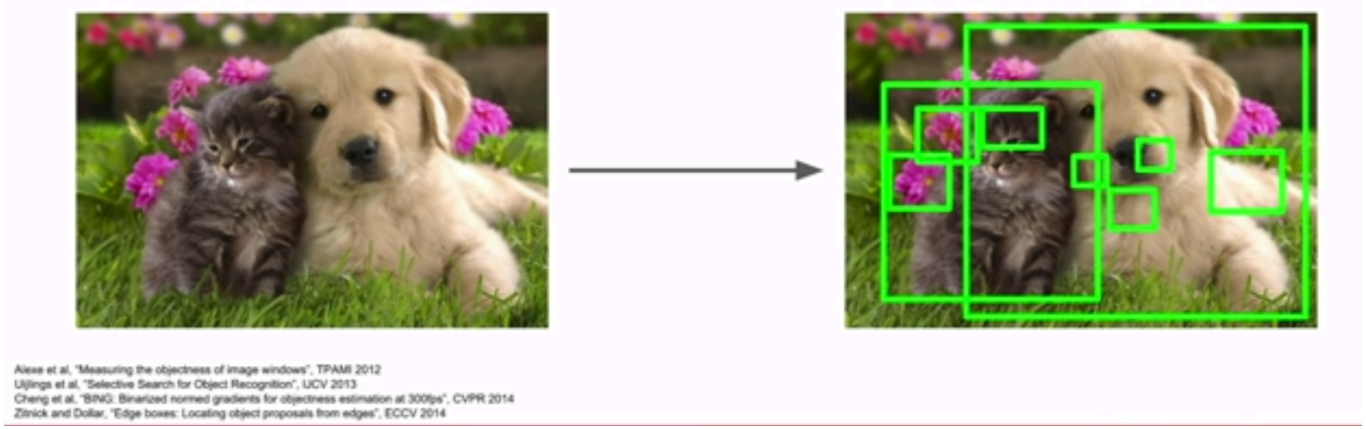
Stanford CS231n

Problem: our regression head outputs a fixed number. But how many coordinates we need to predict depends on how many objects are in the image, which is unknown ex ante.

Object Detection as Classification?

- One approach would be to apply a CNN to many different crops of the same image using a brute force sliding window
- Classify each crop as object class or background, treating as a localization problem
- Need to apply CNN to huge number of locations, aspect ratios, and scales - unfeasible to compute with a deep CNN

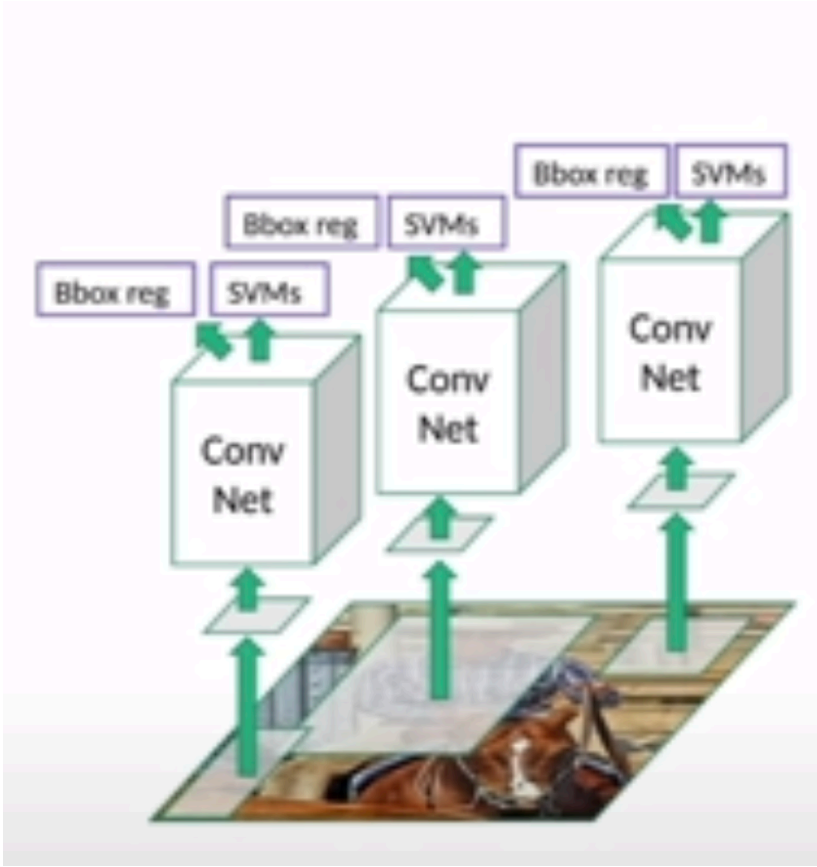
Region Proposal Via Selective Search



Stanford CS231n

Instead, what people did initially was to use traditional image processing tools (not deep learning) to propose i.e. 2,000 regions where objects might be present. Look for blobby regions. Fast-ish to run (a couple of seconds per image).

Region CNN (Girshick et al., 2014)



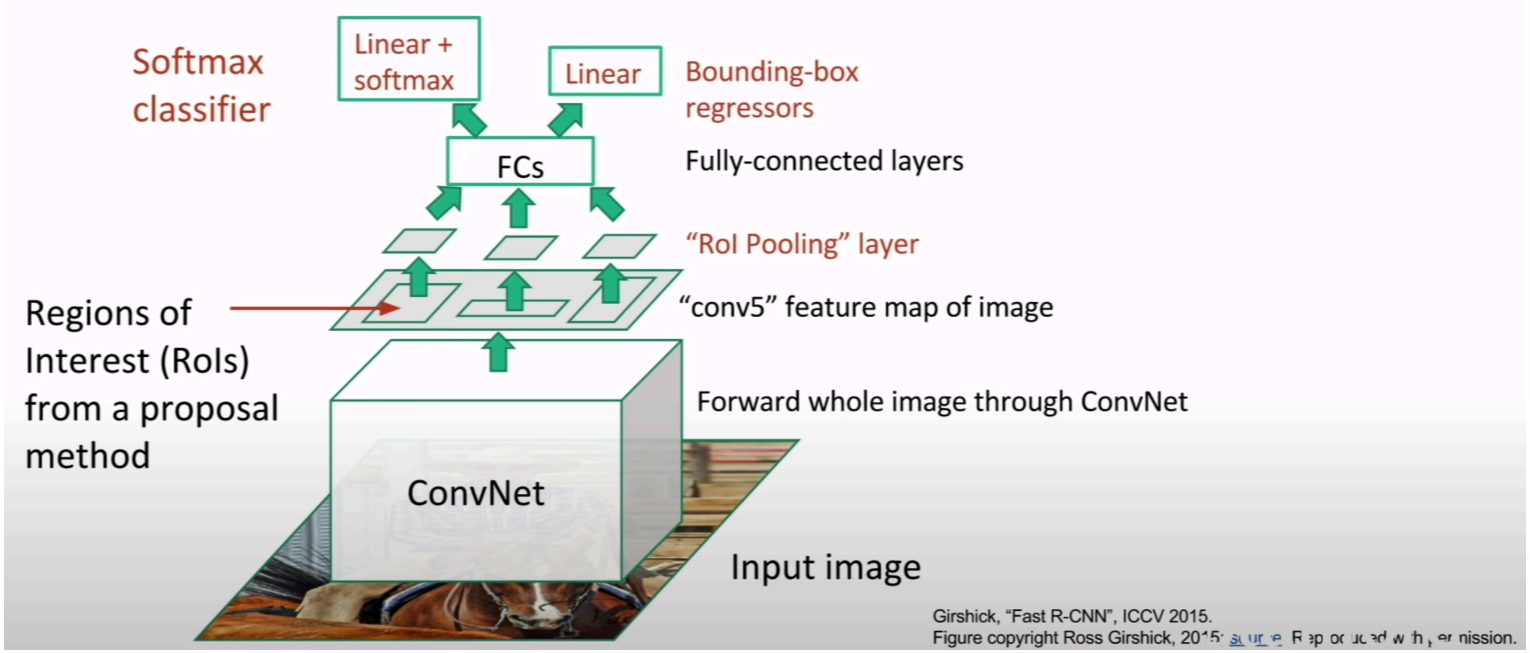
Crop, warp, run through CNN, use an SVM classifier to predict category (add a background class) and a regression to predict correction to the bounding box (proposals won't be perfect)

Problems

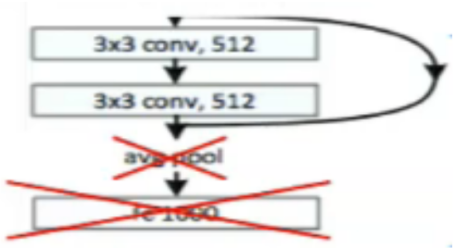
- Slow to train (84 hours)
- Takes a lot of disk space because saves the region proposals to disk
- Ad hoc training: fine-tune CNN backbone with a softmax classifier and then train post-hoc SVM for classification and regression for bounding boxes
- Inference slow (47s/image): thousands of forward passes for each region proposal

Solution to This Challenge (Fast R-CNN)

Rather than running every potential region proposal separately through the ConvNet, reuse the convolutional features map (expensive to compute) across the entire image



Using a CNN as a features extractor

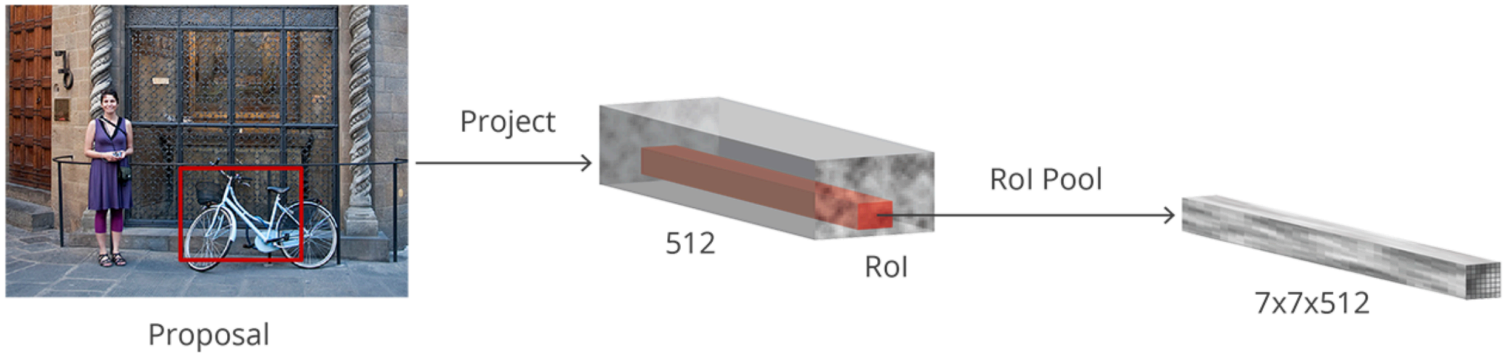


Remove the classification layers so as to use it only as a features extractor. This makes the network fully convolutional, which means that it can take in any input size (very important in detection).

Region of Interest Pooling

- Fixed size feature maps are required to classify proposal regions into a fixed number of classes (FC layers required fixed input sizes)
- Crop the convolutional feature map using each proposal
- Then resize each crop to a fixed sized (i.e. $14 \times 14 \times \textit{convdepth}$) using interpolation (usually bilinear). After cropping, max pooling with a 2x2 kernel is used to get a final (i.e. $7 \times 7 \times \textit{convdepth}$) feature map for each proposal.

RoI Pooling



Central to being able to reuse the conv feature map across proposals

Region-based CNN

- After extracting the convolutional features for each proposal, Region-based CNN mimic the final stages of classification CNNs where a fully-connected layer is used to output a score for each possible object class.
- Two objectives:
  - Classify proposals into one of the pre-defined classes, plus a background class (for removing bad proposals): use a fully connected layer with $N + 1$ units
  - Adjust the bounding box for the proposal according to the predicted class: use a fully connected layer with $4N$ units
- Essentially turned this into a localization problem on each region proposal

Training the R-CNN

- Proposals above an IoU threshold with any ground truth box get assigned to the ground truth; those below a threshold assigned to the background
- Targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the threshold
- Randomly sample a balanced (on foreground/background) mini-batch of proposals
- Multi-task loss (weighted sum of classification and bounding box loss). For the bounding boxes, only take into account loss for the correct class

Post-Processing

- Going to get overlap between predictions
- Non-Maximum Suppression (NMS) takes the list of proposals sorted by score and iterates over the sorted list, discarding those proposals that have an IoU larger than some predefined threshold with a proposal that has a higher score
- Have to be very careful with this threshold: too low and you'll fail to detect objects altogether, too high and you'll end up with too many proposals for the same object
- For our final list of objects, we also can set a score threshold

Evaluation

- Evaluate using mAP
- mAP penalizes you when you miss a box that you should have detected, as well as when you detect something that does not exist or detect the same thing multiple times
- In many papers in this lit, you'll see mAP in the 0.5 range and people argue this is good. In our experience, for documents, it needs to be more like > 0.8 (will depend on specific features of the image)

Mean Average Precision (mAP)

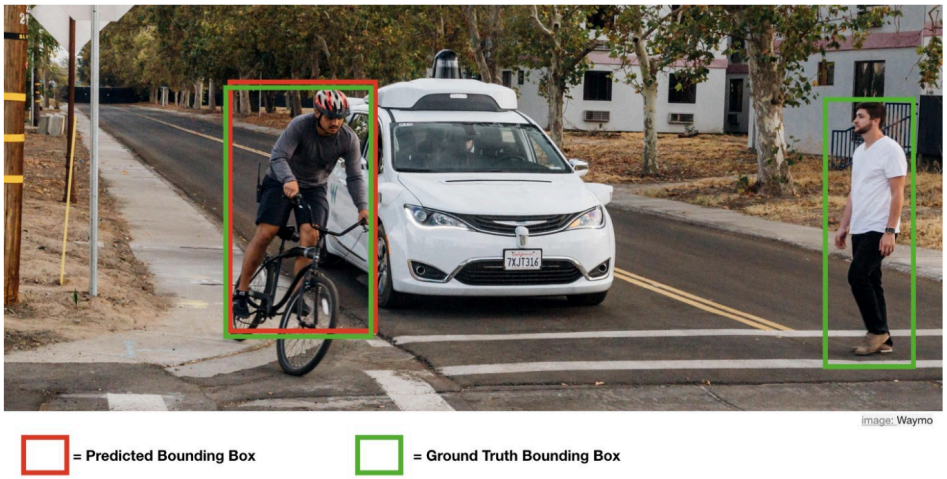
Despite the name, average precision is not the average of precision

Precision:

$$TP = \frac{TP}{TP + FP} \tag{1}$$

TP (true positives) and FP (false positivies)

Suppose mAP was average of precision

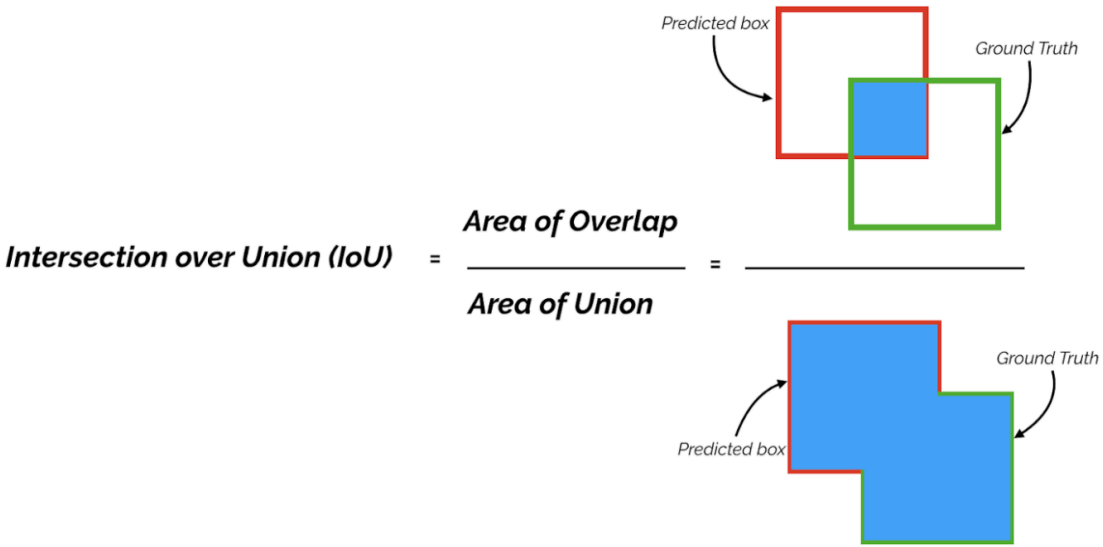


https:

//towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

TP is 1 FP is 0 Precision is 1

Intersection over Union

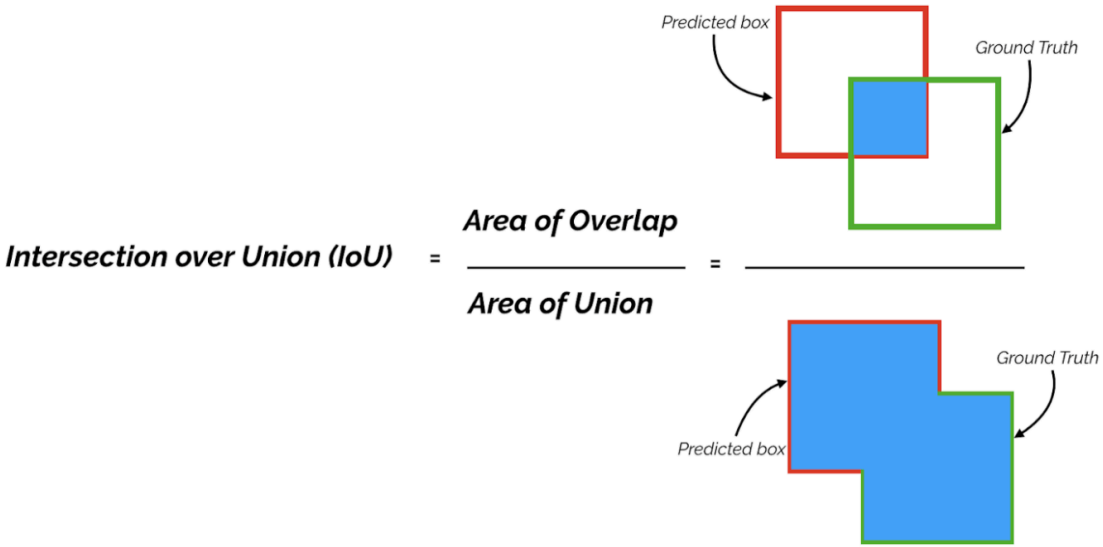


https:

//towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

IoU

If the IoU threshold is 0.5, and the IoU value for a prediction is 0.7, then we classify the prediction as a True Positive (TP). On the other hand, if IoU is 0.3, we classify it as a False Positive (FP)



https:

//towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2

Recall

How well you find true positives; i.e. we can find 80% of objects

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Average Precision

Suppose we have a dataset that has five apples, and there are ten predictions:

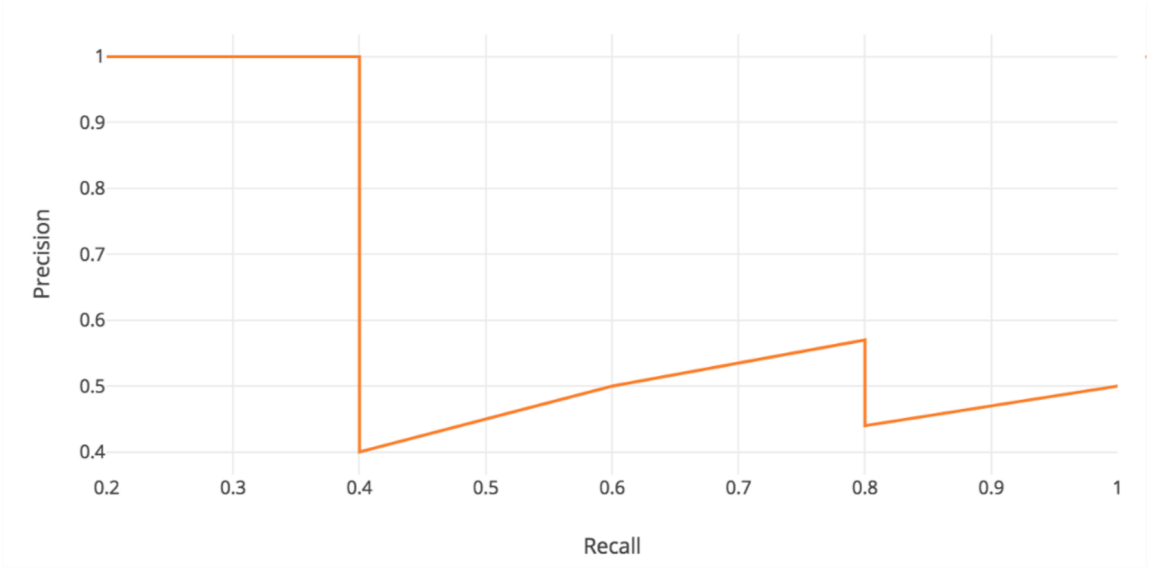
Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

Precision is proportion of TP; recall is proportion of TP out of possible positives (5). Recall increases; precision falls with FP and increases with TP. If multiple detections of the same object are detected, it counts the first one as a positive while the rest as negatives.

Average Precision

Average precision is the area under the precision-recall curve:



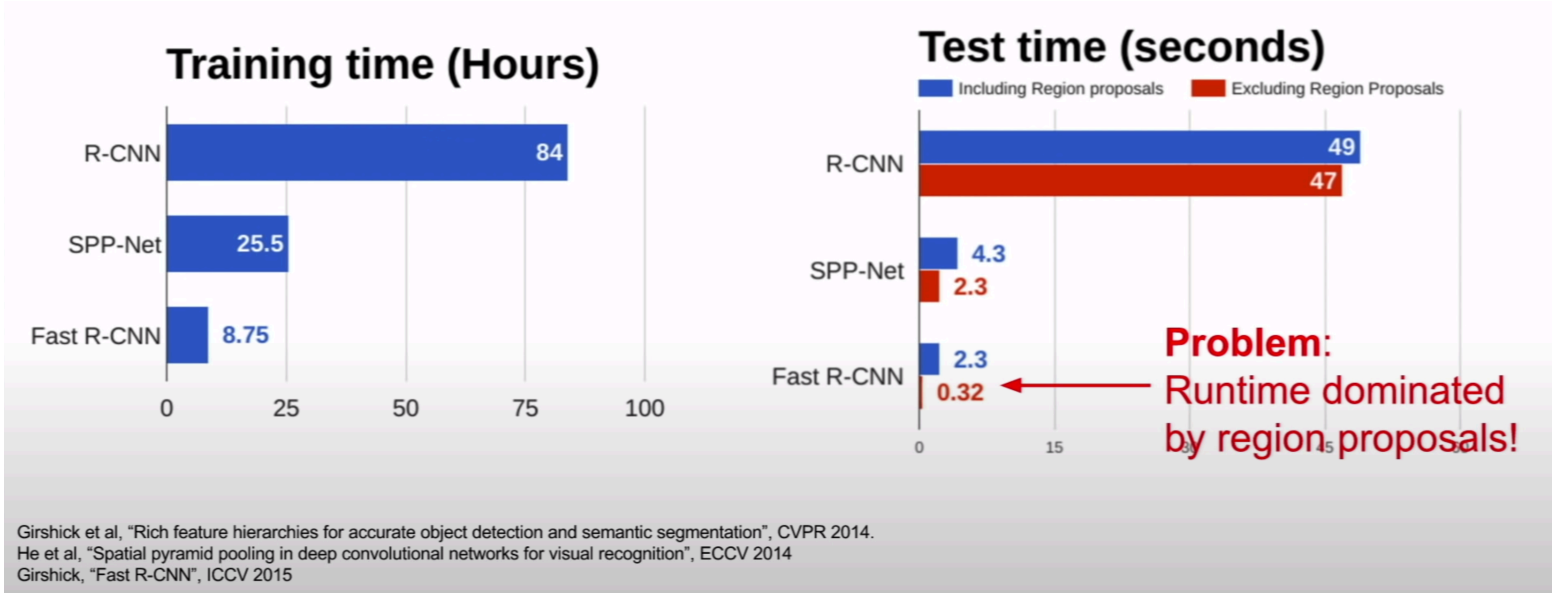
Precision-recall curve

https:

//jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

usually smooth curve in practice; since precision and recall are between 0 and 1, so is average precision

Problem with Fast R-CNN



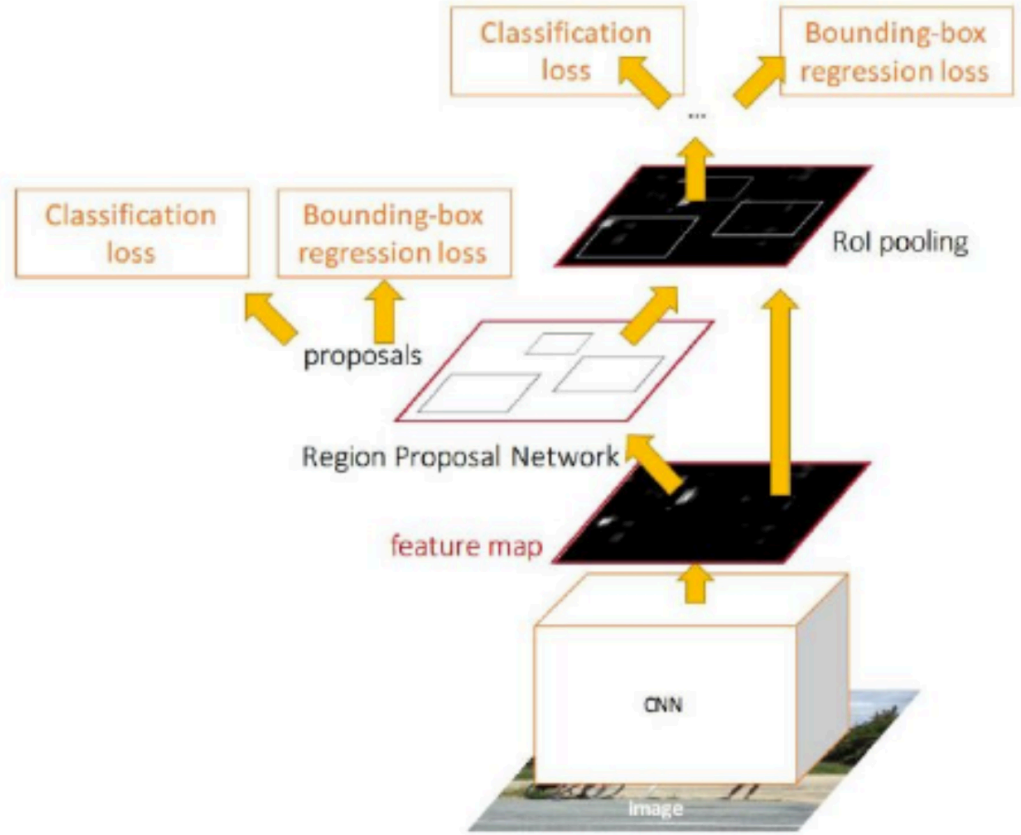
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Stanford CS231n

Faster R-CNN

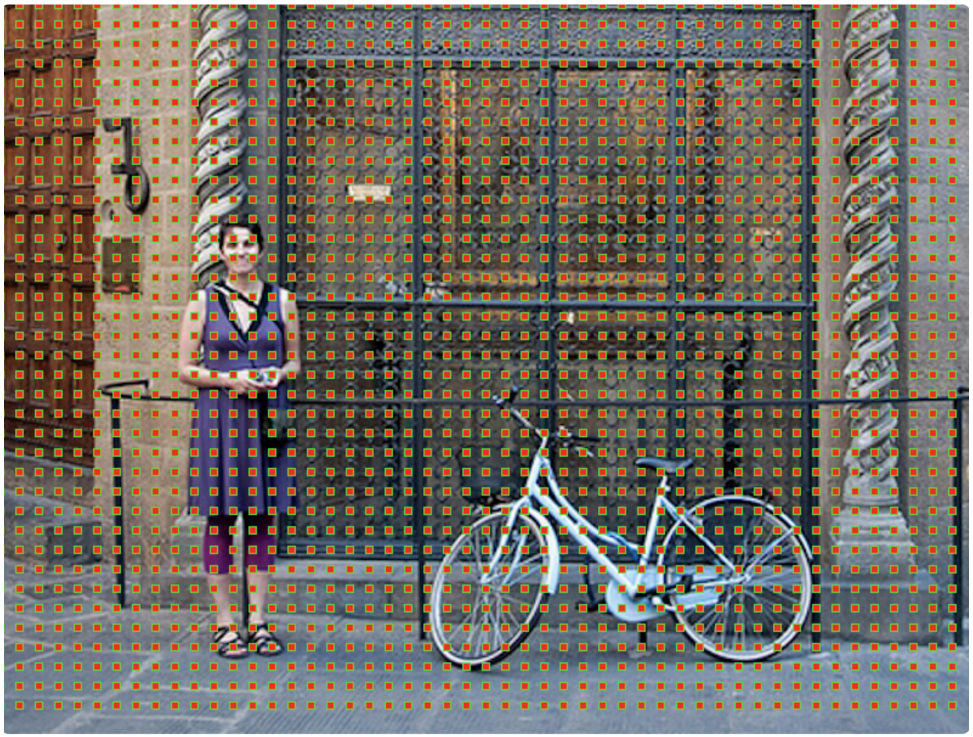
- Innovation: integrate the region proposal network as a learnable part of the model, following the convolutional backbone
- The RPN uses a sliding window over the features maps to get relevant anchor boxes - fixed sized bounding boxes of varying sizes that are placed throughout the image and represent the approximate bbox predictions
- The RPN has a binary classification for whether the bounding box has an object (background or foreground) and a regression head to refine the bounding boxes
- It classifies whether or not the box has an object by looking at its intersection-over-union with the ground truth in that region

Faster R-CNN



Significant computational saving because use the conv feature maps for everything

Anchors



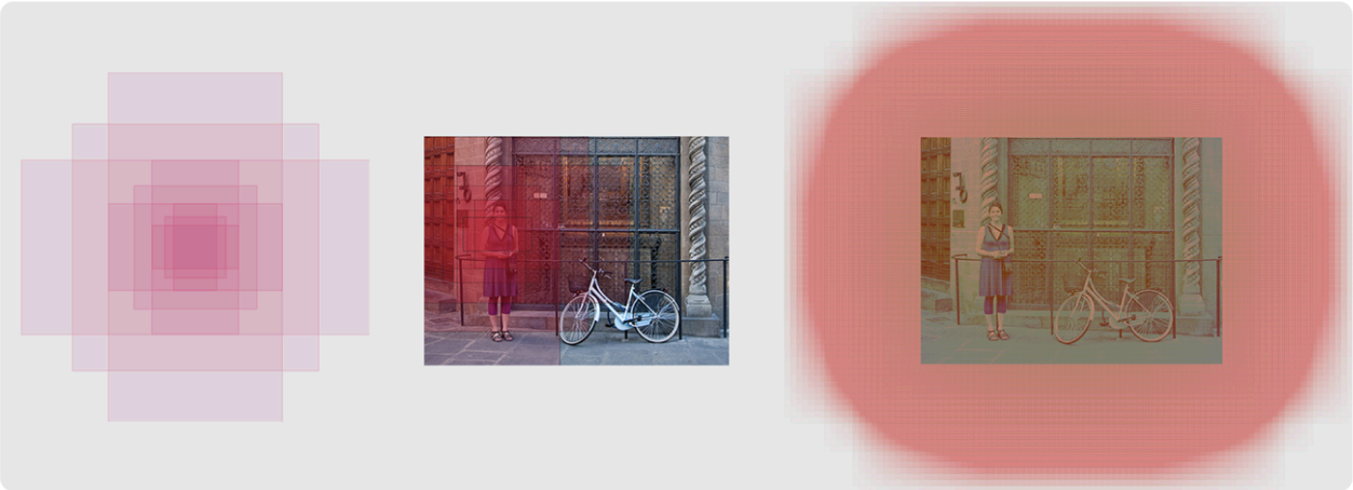
https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

If the image is $w \times h$, the feature map will end up $w/r \times h/r$ where r is the subsampling ratio (i.e. this is 16 in VGG). If we define one anchor per spatial position of the feature map, the final image will have anchors separated by r pixels.

Anchors

- In order to choose the set of anchors we define a set of sizes (e.g. 64px, 128px, 256px) and a set of ratios between width and height of boxes (e.g. 0.5, 1, 1.5) and take all possible combinations
- The Faster R-CNN paper has 9 anchor boxes
- In practice these anchor boxes are hyperparameters that need to be set appropriately

Anchors



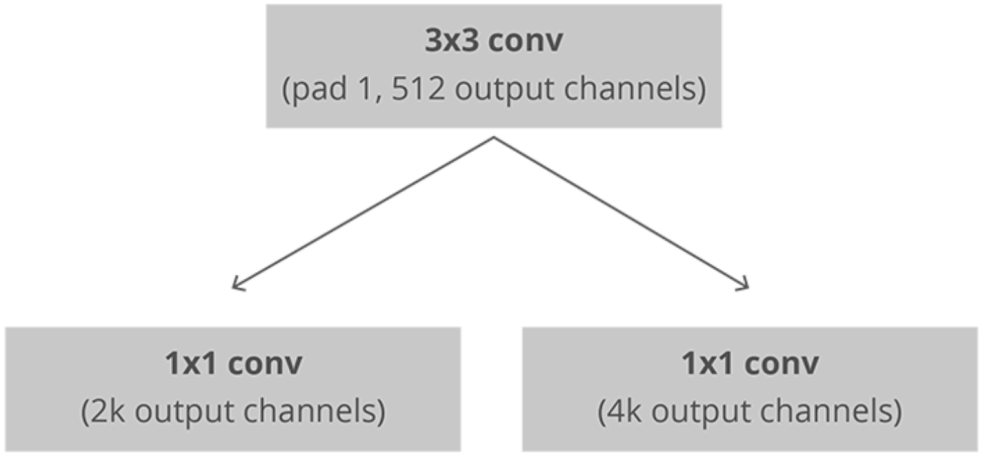
Left: Anchors, Center: Anchor for a single point, Right: All anchors

https://tryolabs.com/blog/2018/01/18/

faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

Drop anchors that extend off the image

Convolution Implementation of RPN



Convolutional implementation of an RPN architecture, where k is the number of anchors.

For each pixel, apply a 3 x 3 conv with 512 layers to the output features map from the backbone then apply parallel 1 x 1 convs whose depth depends on the number of anchors k (to get appropriate depth to connect the conv layer to the FC classifier and regression heads). Weights learn if foreground or background and adjustments to bbox coordinates.

RPN Outputs

- Classifier head: $2k$ probabilities for object/non-object for each point (k is number of anchors)
- Regression head: $4k$ predictions - the delta on the x and y coordinates, width, and height of the bounding box proposal. Apply these to anchors to get final proposals.

Training the RPN

- For training, we take all the anchors and put them into two different categories, based on whether they have significant overlap or minimal overlap with the ground truth boxes (foreground and background)
- Randomly sample those anchors to form a mini-batch size of 256, maintaining a balanced ratio between background and foreground anchors
- Use all anchors from the mini-batch to calculate the classification loss using cross-entropy
- Use only minibatch anchors classified as foreground to calculate the regression loss, using foreground anchor and ground truth boxes

Post-Processing Region Proposals

- Since anchors overlap, proposals end up also overlapping over the same object
- Non-Maximum Suppression (NMS) is applied, just as in the R-CNN
- After applying NMS, we keep the top N proposals sorted by score (again, pay attention to this parameter)

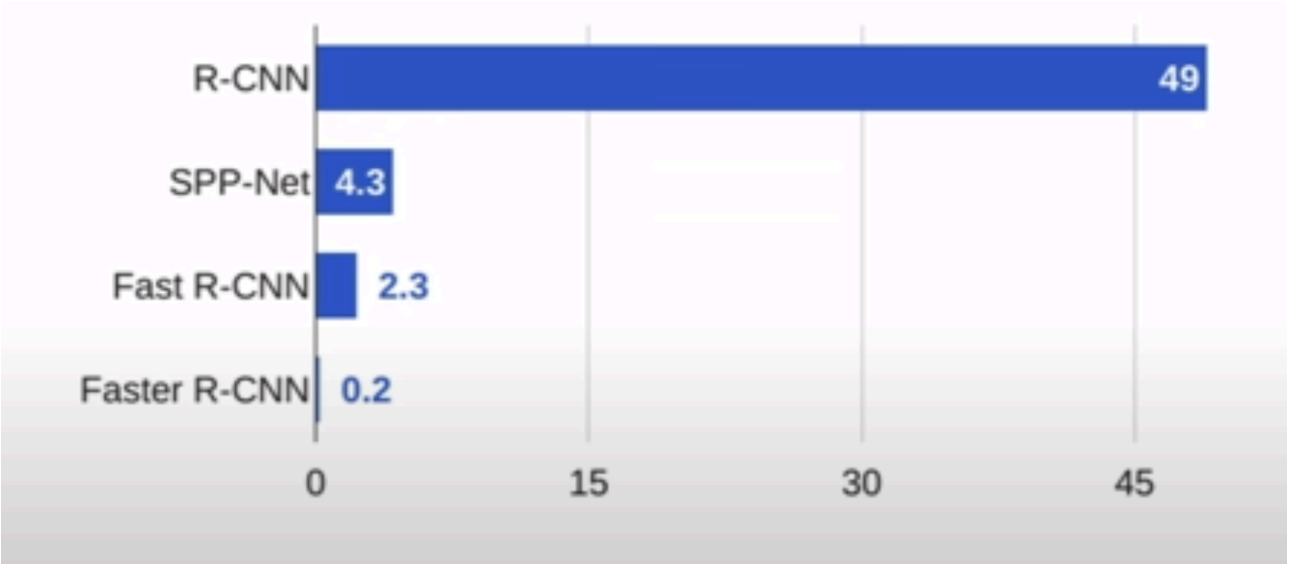
The rest of the model is Fast R-CNN

- Next are the same RoI and R-CNN layers that we saw with Fast R-CNN
- That is, Faster R-CNN is Fast R-CNN with region proposal integrated into the network

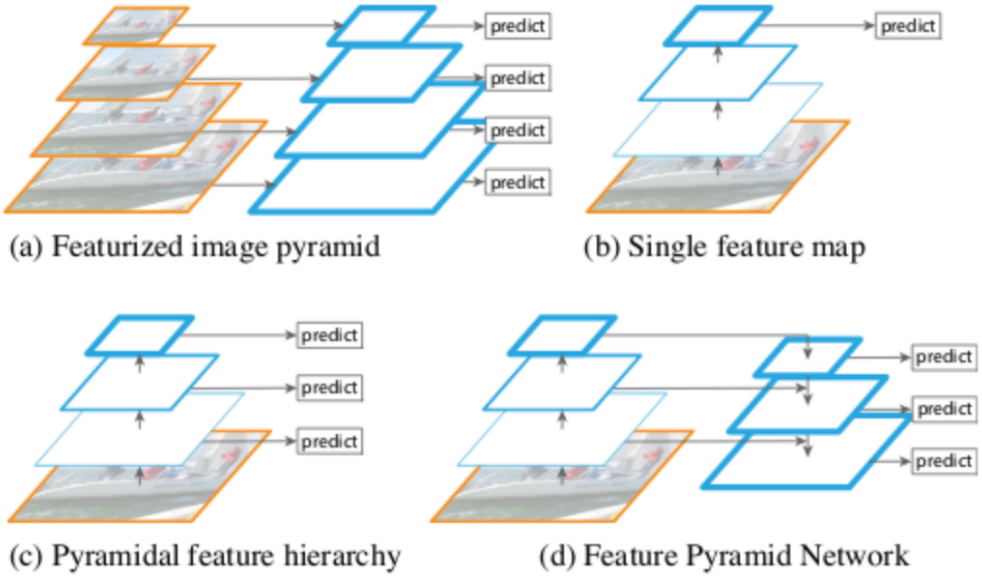
Training Faster R-CNN

- In the original paper, Faster R-CNN was trained using a multi-step approach, but it has since been shown that joint training leads to better results
- 4 different losses, two for the RPN and two for R-CNN. We have the trainable layers in RPN and R-CNN, and we also have the backbone CNN, which we can fine-tune or not
- The four different losses are combined using a weighted sum into a multi-task loss. Can also add regularization to the RPN, the R-CNN, and potentially the backbone

R-CNN Test Speed



Feature Pyramids



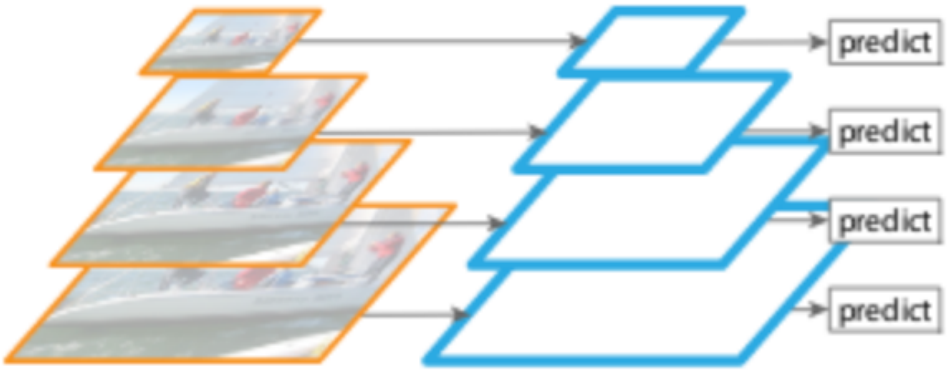
(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

Lin et al., 2017

Feature pyramids have a long history in computer vision

Problem with an Image Pyramid

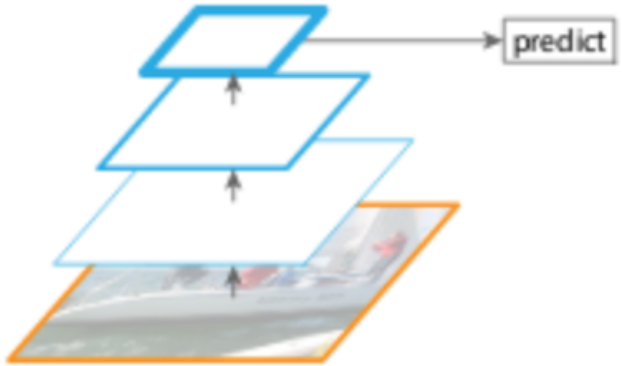


(a) Featurized image pyramid

Lin et al., 2017

Features are computed on each of the image scales independently. Way too computationally costly with a CNN backbone. Mostly used in pre-deep learning days.

Problem With a Single Feature Map



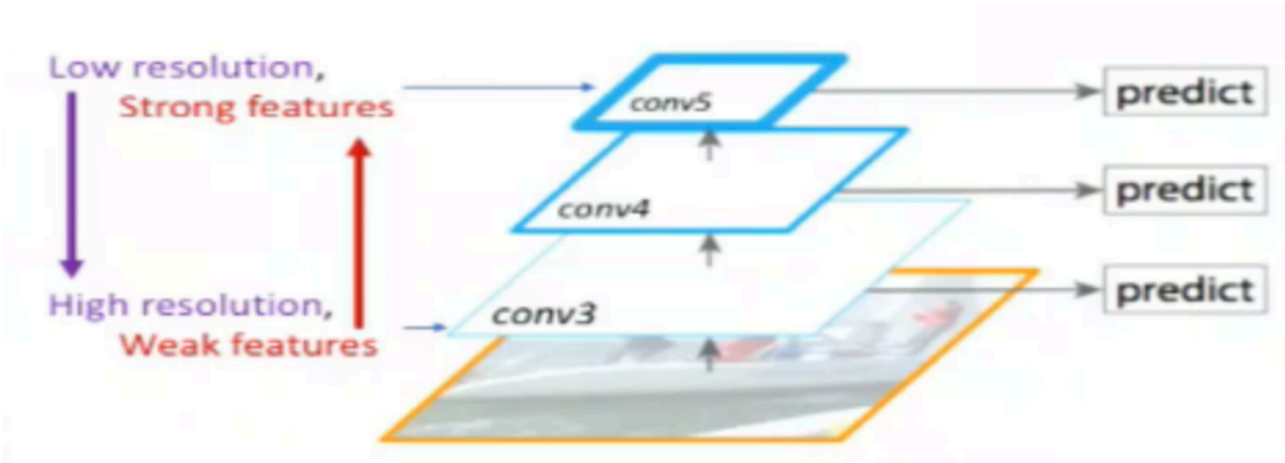
(b) Single feature map

Lin et al., 2017

The norm for object detection before the feature pyramid network. Makes it difficult to recognize objects of different scales, particularly small objects. Going to be a problem with documents, which often have many more small objects than a natural image.

Pyramidal Feature Hierarchy

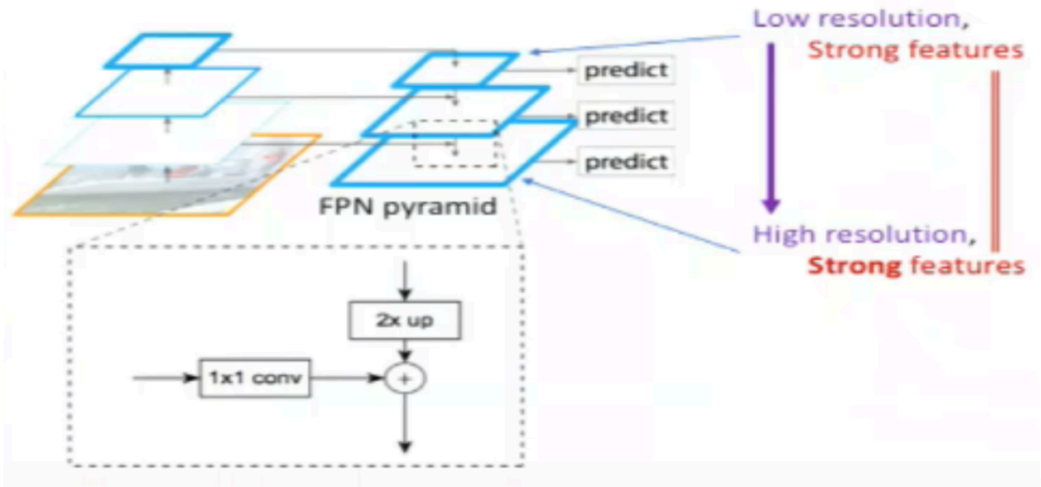
Reuse the pyramidal feature hierarchy from the conv net, as if it were a featurized image pyramid



Lin et al., 2017

Recall that in a conv net, the initial layers extract the low level features from the pixel inputs, and assemble them into higher level features as you progress through the network. Going to be a problem to detect objects from these low level features maps.

Feature Pyramid Network



Lin et al., 2017

Solves problem of weak features at later layers for multi-scale detection. Take strong features and propagate them to the high resolution feature maps. Lateral connections are added at each level of the pyramid.

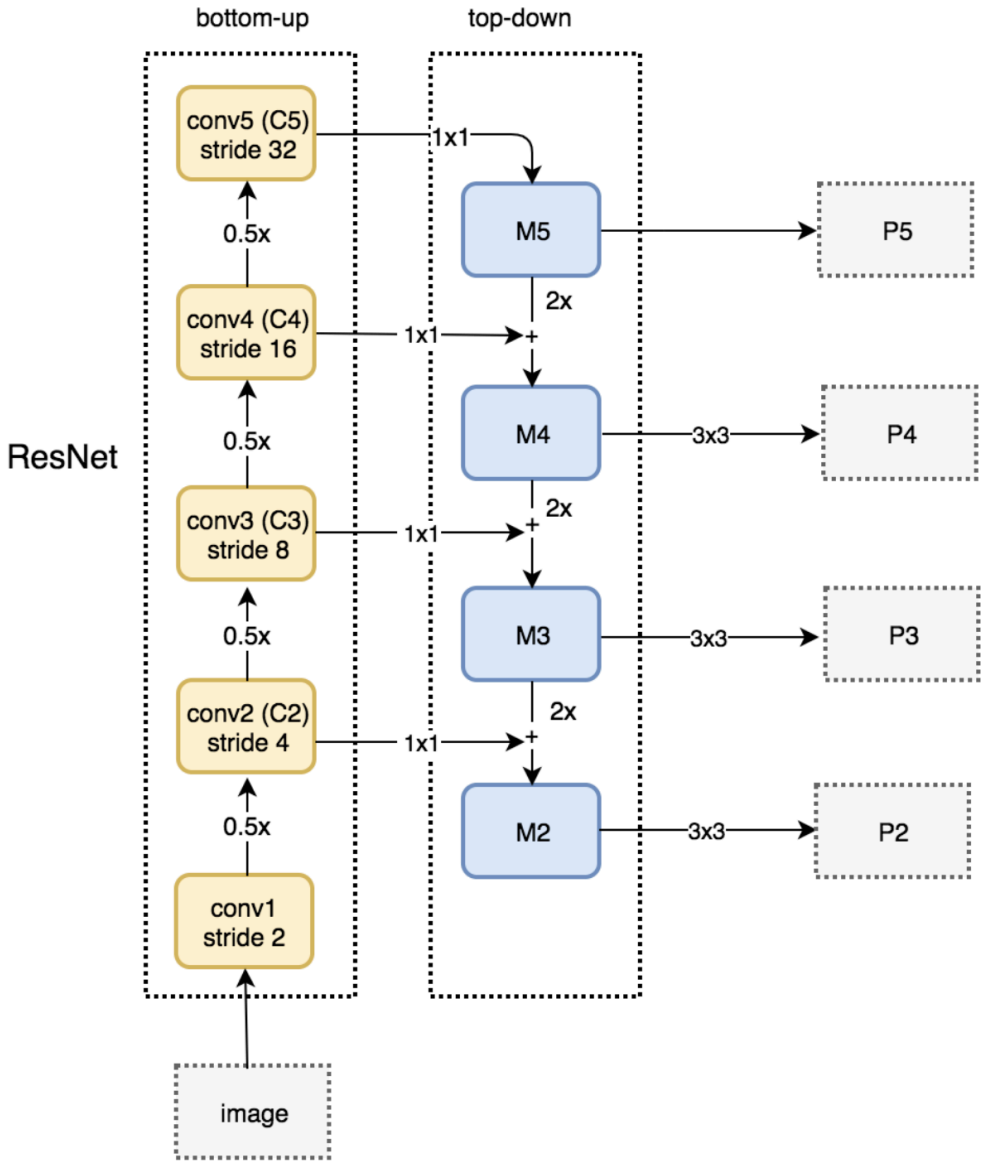
Bottom Up Pathway

- ResNet
- Consists of many convolution layers. As we move up, the spatial dimension is reduced by 1/2 (i.e. double the stride)

Top Down Pathway

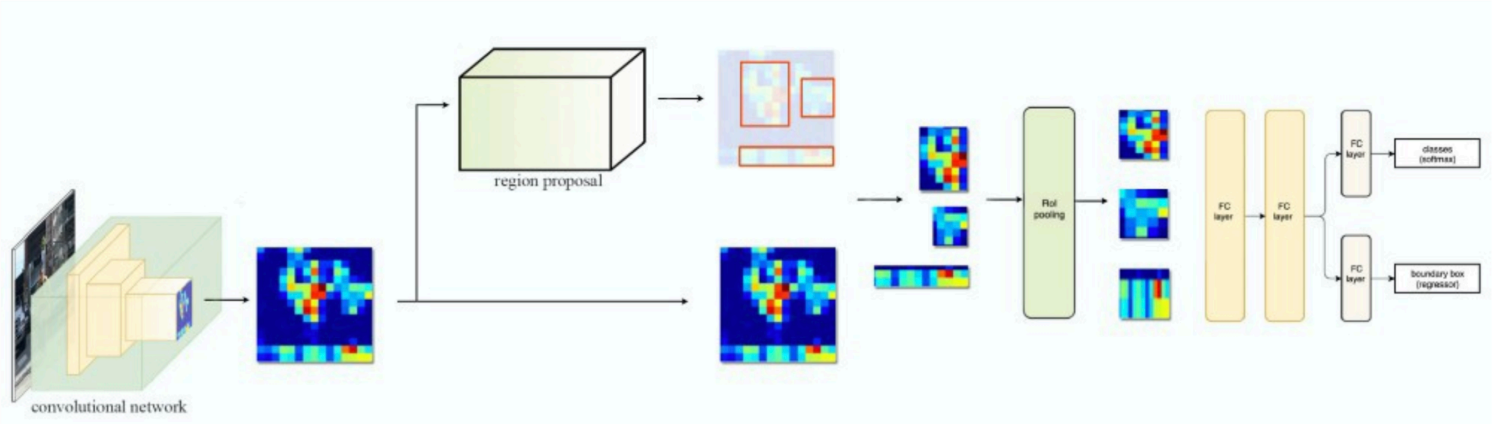
- Upsample the previous layer by 2 using nearest neighbors upsampling
- While the reconstructed layers are semantically strong, the locations of objects are not precise after all the downsampling and upsampling
- The model adds lateral connections between reconstructed layers and the corresponding feature maps to help the detector to predict the locations better
- 1 x 1 convs applied to the features maps from the bottom up pathway. All pyramid feature maps need a depth of 256, because they will share the same FC classifier and box regression head

Feature Pyramid Network



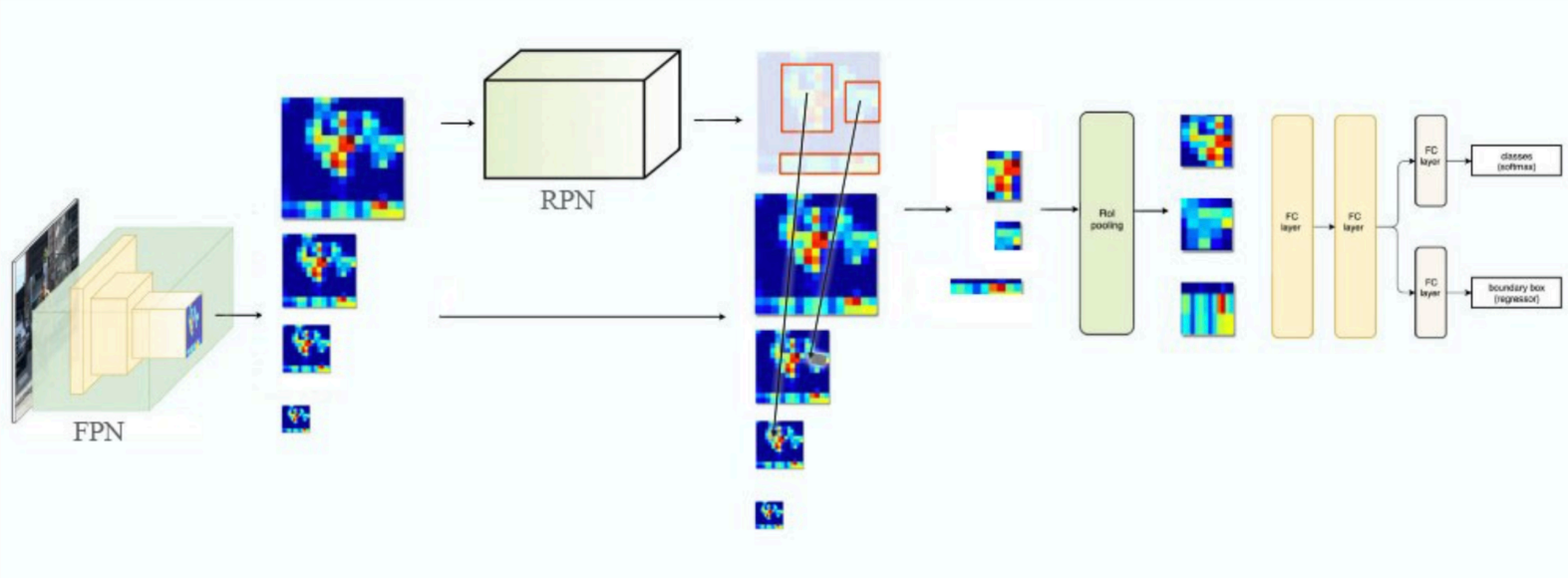
Vanilla Faster R-CNN

Recall vanilla Faster R-CNN



FPN R-CNN

Recall vanilla Faster R-CNN



The formula to pick the feature maps is based on the width w and height h of the ROI

Outline

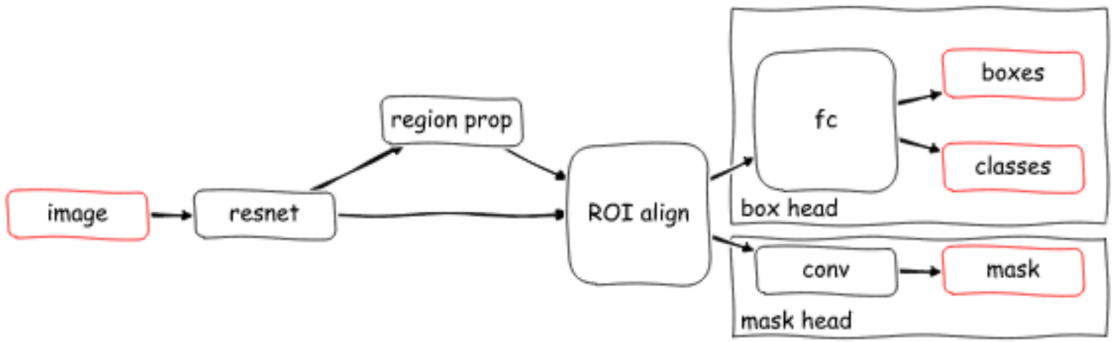
Object Segmentation

He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

Mask R-CNN

- The successor to Faster R-CNN - Mask R-CNN - essentially added a fully-connected branch to Faster R-CNN to predict masks
- Semantic segmentation problem inside each region proposal
- The mask loss is computed by taking the cross-entropy loss between the predicted mask and the ground truth, for each pixel
- It also incorporated Feature Pyramid Networks
- RoI pooling is modified slightly using a method called RoI Align, which allows for the finer spatial localization required to predict masks
- Trained on MS COCO 200K images, average of 5-6 instances per image

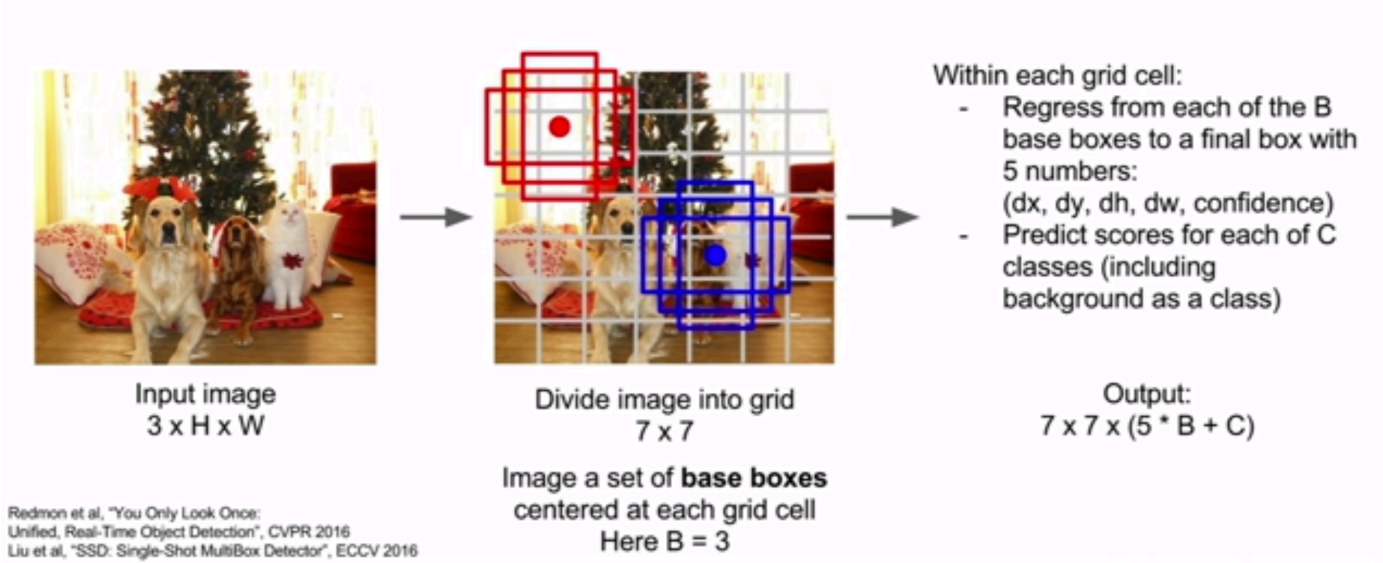
Mask R-CNN



https://kharshit.github.io/blog/2019/08/23/quick-intro-to-instance-segmentation

Outline

YOLO (You Only Look Once)



Stanford CS 231n

Treat object detection as a regression problem

YOLO/Single Shot Detector

- Treat like a regression problem. Divide input images into 7 x 7 grid; base bounding boxes within each grid; For each of these, predict offset and classification scores
- Essentially analogous to just using the RPN, except with all C classes plus background, rather than using the region proposal network (background/non-background and initial bbox predictions) and then further refining this (specific classes, additional bbox refinement)
- Intuitive that this is faster but the accuracy is not as high
- In our experience, it was not accurate enough to use on documents, where errors can easily be catastrophic for downstream analyses