Fast RCNN

Object Detection vs Object Classification

- Object Detection is a much more complex task than object classification
- Complexity arises due to the problem of object localization – processing proposals individually and fine tuning them
- Solution compromises speed, accuracy and simplicity
- Current solutions are multistage approaches

Problems with RCNN

- Uses a multistage pipeline for training [ConvNet-SVM-BBoxReg]
- Training is slow because features are extracted from each proposal of each image for SVM and BBoxReg
- Also, these feature are written to disk which require high storage
- It is also slow at test time because each proposal is processed

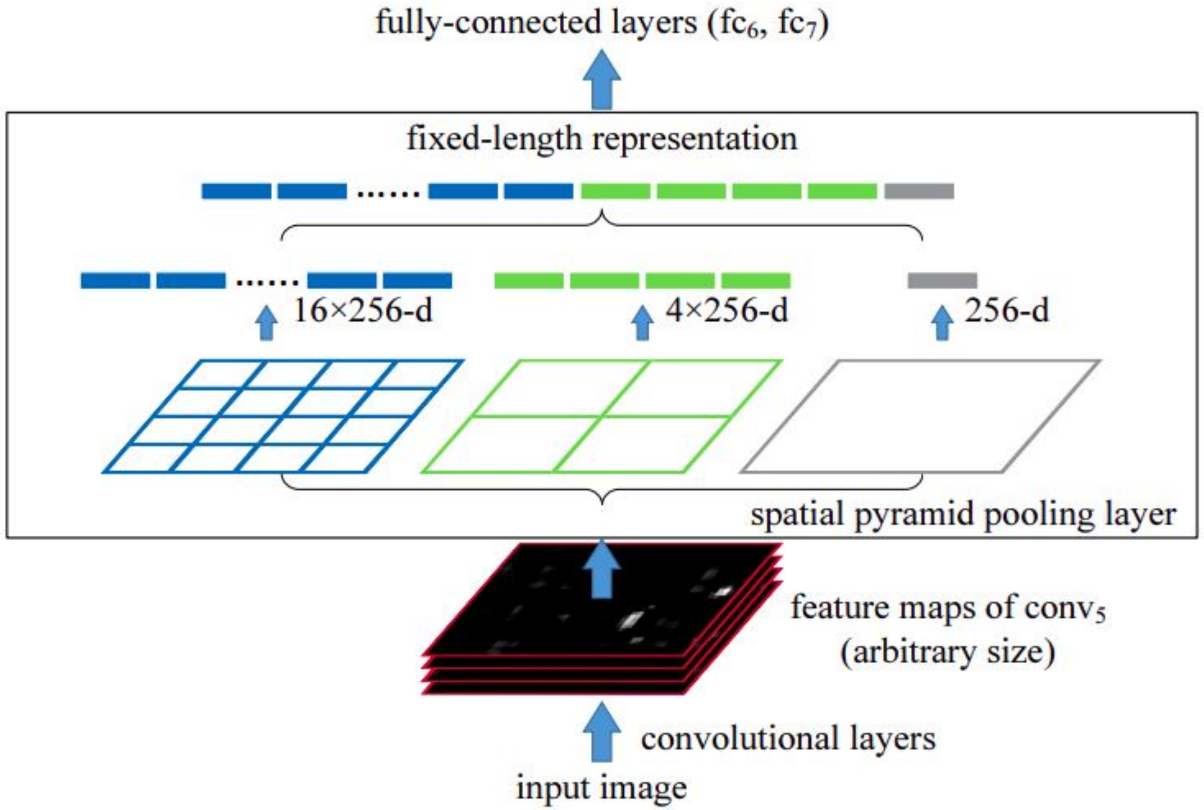
It is slow because it performs ConvNet forward pass for each proposal without sharing computation

Spatial Pyramid Pooling (SPPnet)

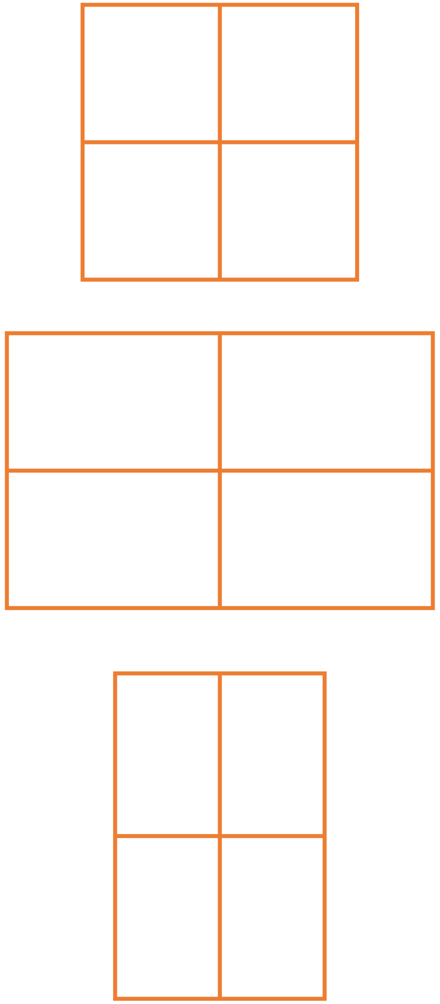
- Made the box prediction very fast
- The fixed size constraint of a CNN is not because of ConvNet forward pass but due to Fully Connected Layer at the end
- Problem was solved by replacing the last layer of ConvNet with a Spatial Pyramid Pooling Layer

Spatial pyramid pooling networks (SPPnets) were proposed to speed up R-CNN by sharing computation

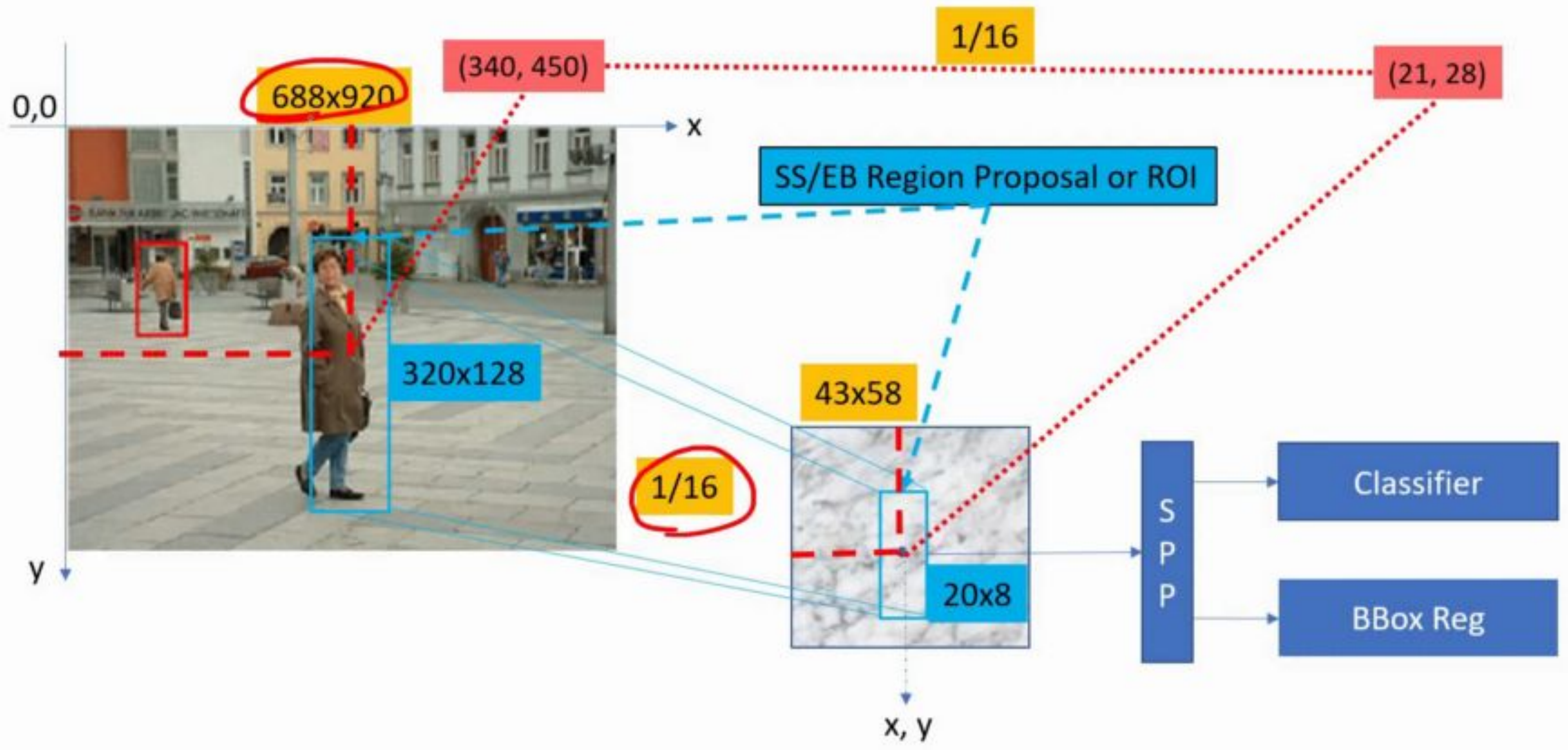
SPP Layer



fully-connected layers (fc_6, fc_7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of $conv_5$
(arbitrary size)
convolutional layers
input image



SPPnet in object detection

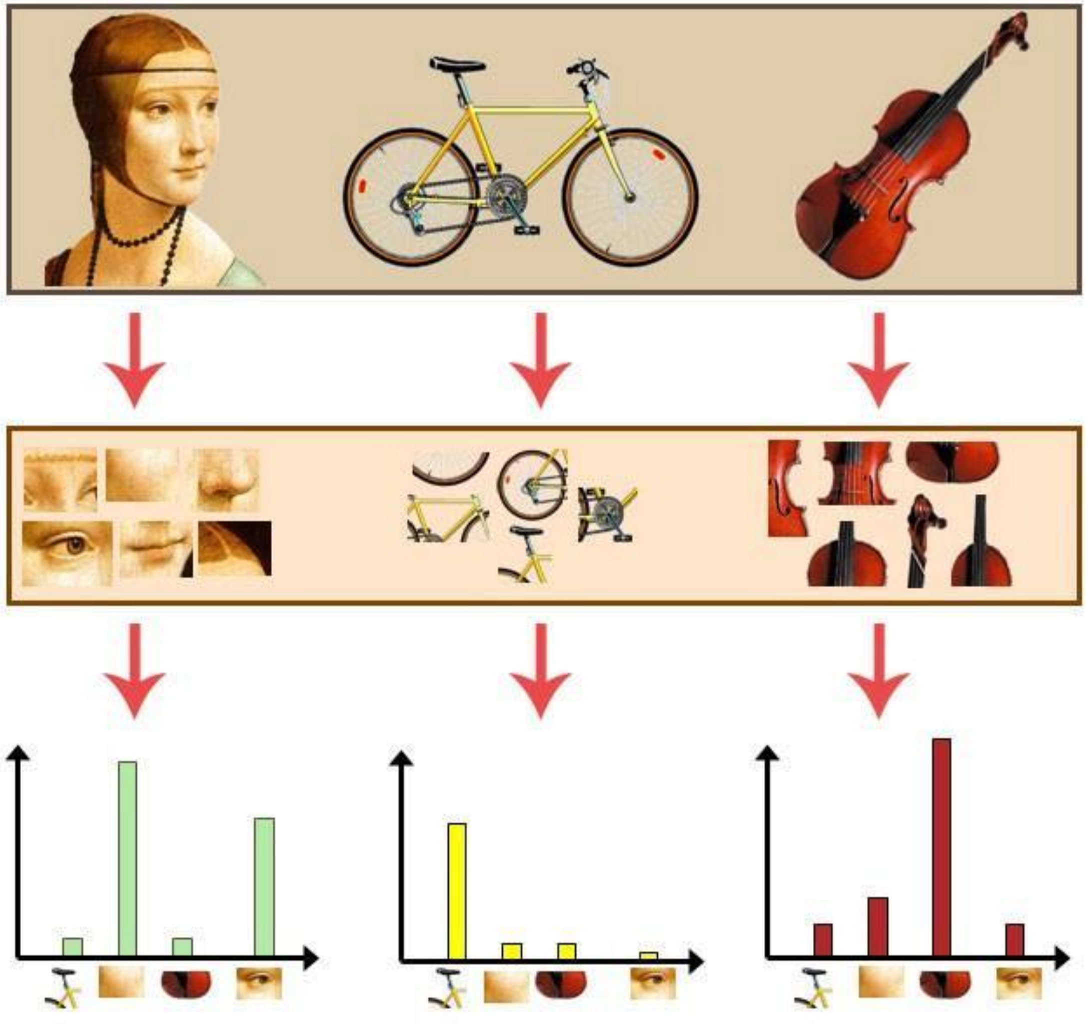


SPPnet Drawbacks

- Like R-CNN, training is a multi-stage pipeline that involves extracting features, fine-tuning a network with log loss, training SVMs, and finally fitting bounding-box regressors
- Features are also written to disk (high storage required)
- The fine-tuning algorithm cannot update the convolutional layers that precede the spatial pyramid pooling

Bag Of Visual Words

- Inspired by Bag of Words (used in NLP) where a document is analyzed by calculating the frequency of words
- Similarly in computer vision, BOVW is used to represent an image as a set of features



Fast R-CNN [Details]

Fixes all disadvantages of RCNN & SPPnet. Improves accuracy and speed.

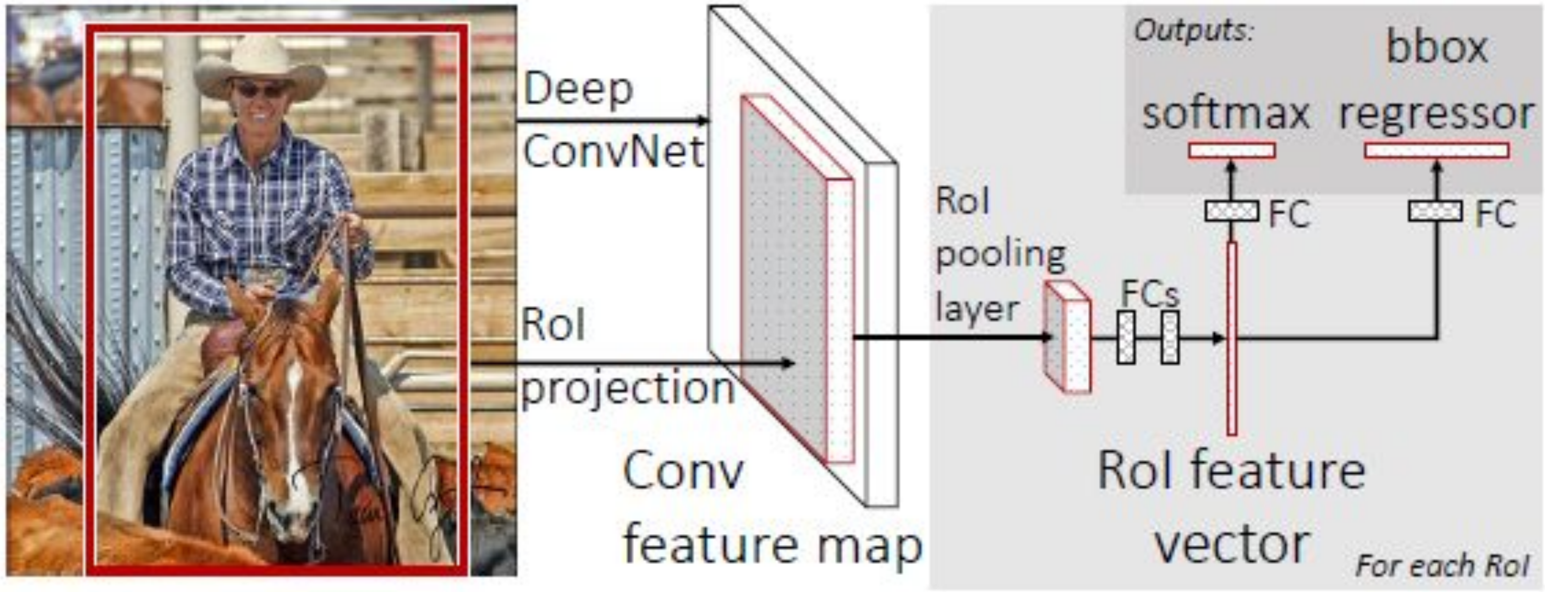
Spoilers ⚠️

- Higher detection quality
- Single stage training process
- Training can update all layers
- No disk storage required for feature caching

Fast RCNN Architecture

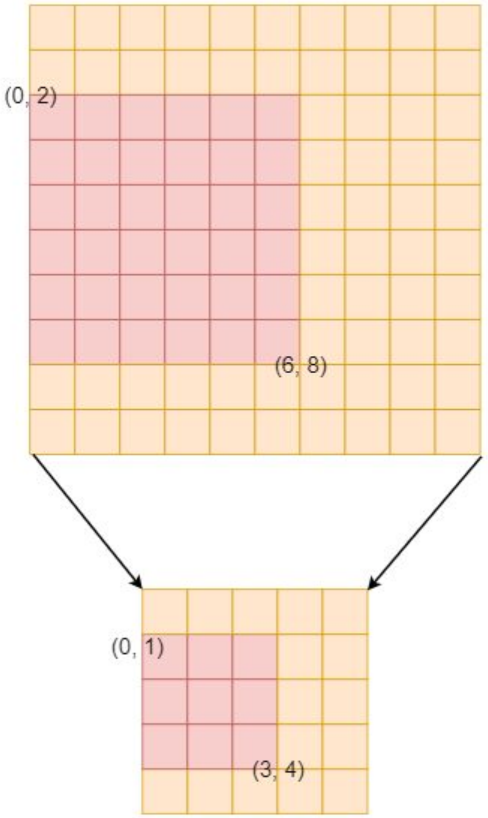
- Takes an image and set of proposals as input
- Produces a conv feature map out of image
- Region of Interest (RoI) pooling layer extracts fixed-length feature vector corresponding to each proposal
- Feature vector is fed into a sequence of fully connected layers
- This branches into two sibling output layers:-
  1. Softmax layer for K classes and 1 background class
  2. Bounding box position for each of K classes

Fast RCNN Architecture

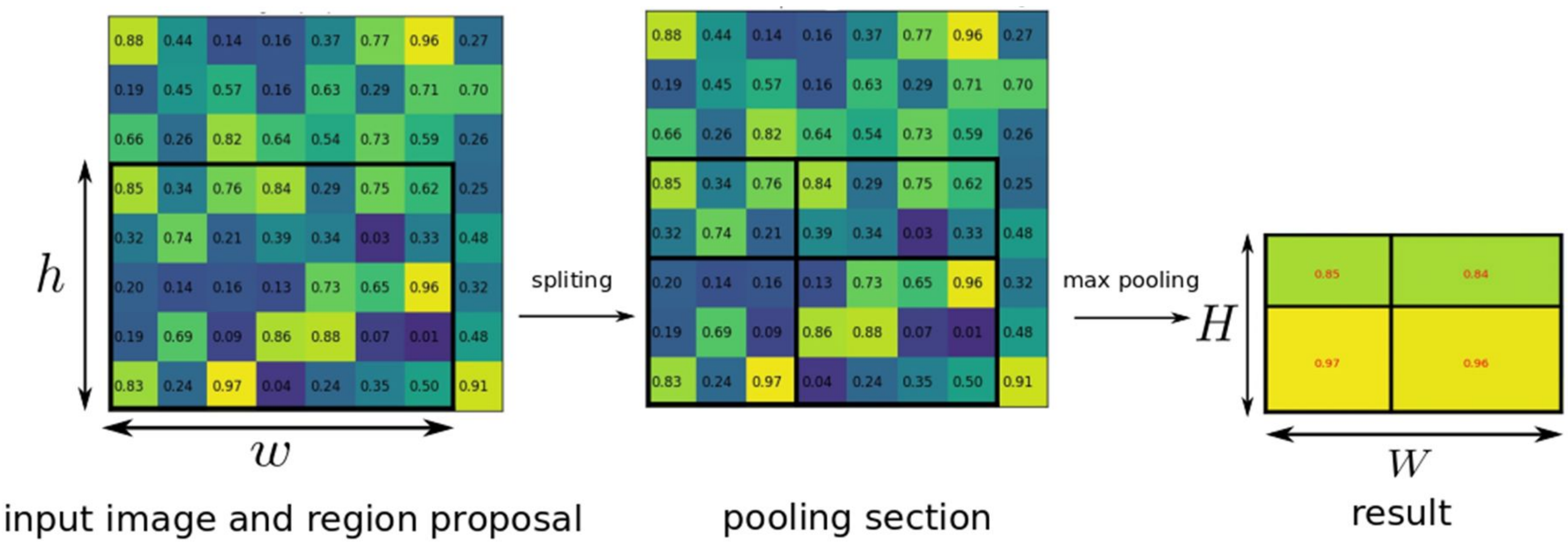


RoI pooling layer

- Uses max pooling to convert features inside a region of interest into a small feature map of fixed size (HxW)
- Region of Interest is a rectangular window in conv feature map
- RoI max pooling works by dividing the RoI window (hxw) into an grid of sub-windows of approximate h/H x w/W size and then max-pooling the values in each sub-window into the corresponding output grid cell.



RoI pooling example

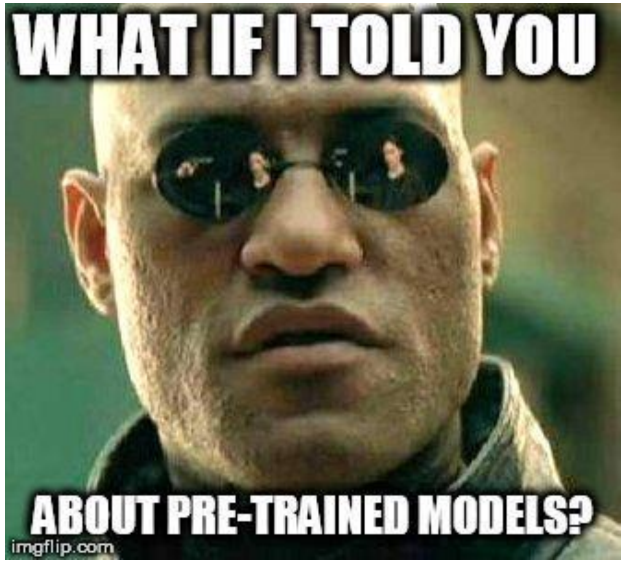


Using Pre-trained Network

- Last max pooling layer is replaced with RoI pooling layer
- Last fully connected layer and sofmax layer is replaced with two sibling layers
- Network is modified to take 2 data inputs – input image and RoIs

Used 3 pretrained networks of different sizes

AlexNet > VGG-CNN-M-1024 > VGG16



Fine tuning

- Backpropagation through SPPnet is highly inefficient when each RoI during training comes from a different image
- To solve this issue, fast RCNN uses a hierarchical sampling method
- Fast RCNN uses single stage training process with the help of multitask loss function

Multi-task loss

- First sibling layer outputs a probability distribution over K+1 classes

$$p = (p_0, \ldots, p_k)$$

- Second sibling layer outputs a bounding-box regression offset,

$$t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$$

- Each RoI is labelled with a ground truth class u and a ground truth bounding-box regression target v
- Multi-task loss L can be used to train both simultaneously

Loss function

λ controls the balance between two losses. In this paper $\lambda=1$

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v),$$

Log loss for true class *u*

$$L_{\text{cls}}(p, u) = -\log p_u$$

Catch-all background class is labelled as u=0. This will ensure that BBox loss is included only when actual class is not background

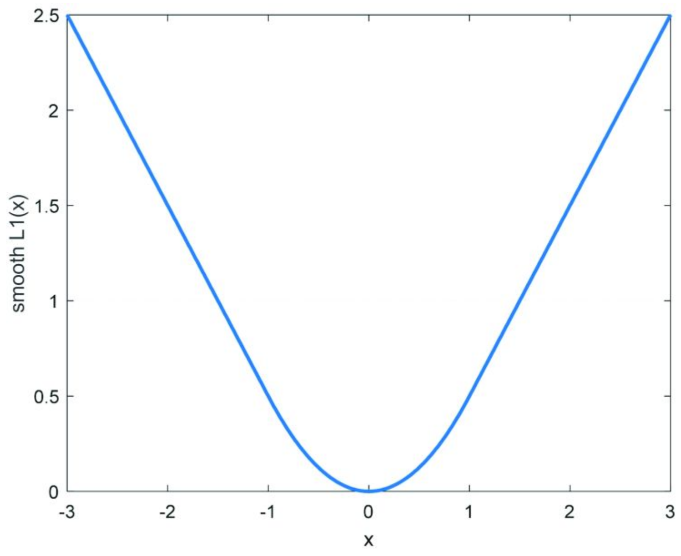
Defined over a tuple of true bounding-box regression targets for class *u*, $v = (v_x; v_y; v_w; v_h)$, and a predicted tuple $t^u = (t^u_x, t^u_y, t^u_W, t^u_h)$, again for class u.

Bounding box regression loss

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

- L1 loss is less sensitive to outliers than L2 loss
- Training with L2 loss can require careful tuning of learning rates in order to prevent exploding gradients



Mini batch sampling

- A hierarchical sampling technique is employed
- First N images are sampled and then R/N RoIs are sampled in each mini-batch
- Here N=2 and R=128, 64 RoIs per image were included in a mini-batch, which consisted of only 2 images

Mini batch sampling

- 25% of the RoIs were taken from object proposals that have intersection over union (IoU) overlap with a ground truth bounding box of at least 0.5 [u>=0]
- The remaining RoIs are sampled from object proposals that have a maximum IoU with ground truth in the interval [0.1, 0.5) [u=0]
- During training, images are horizontally flipped with probability 0.5
- No other data augmentation is used

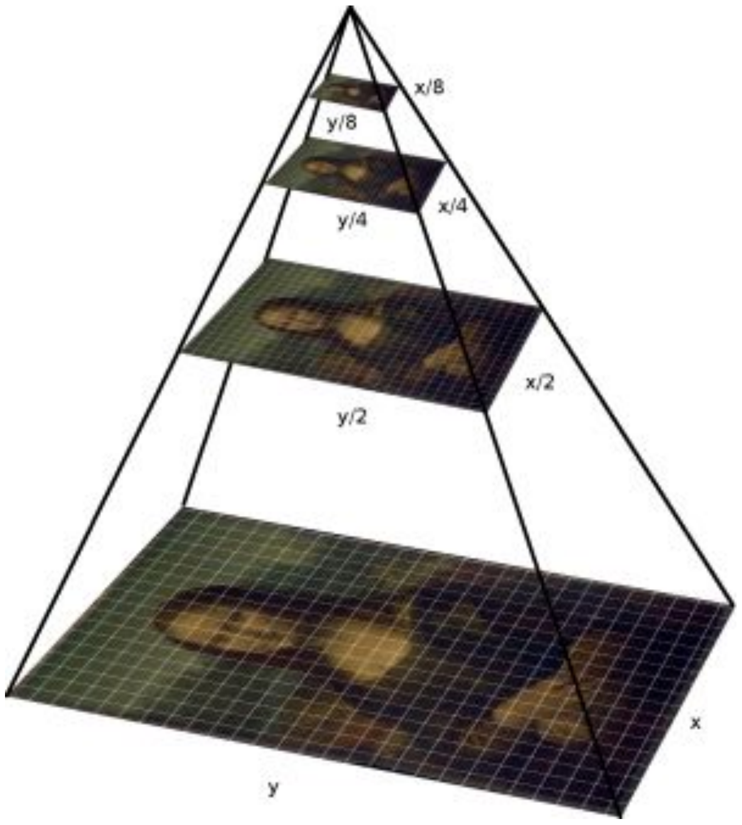
Backpropagation

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j \left[i = i^*(r,j) \right] \frac{\partial L}{\partial y_{rj}}.$$

SGD Hyper-parameters

- FC layers were initialized from gaussian distribution and bias=0
- Global learning rate was 0.001 for first 30k minibatches, reduced to 0.0001 for next 10k minibatches
- Momentum = 0.9, Parameter Decay = 0.0005

Scale Invariance



- 2 ways of achieving scale invariance:
  - Bruteforce – process for fixed size
  - Multiscale approach – use image pyramid
- At test time, image pyramid is used to approximately scale-normalize proposals
- At training time, a random scale is sampled whenever required like augmentation

Fast RCNN Detection

- Network takes an image or image pyramid (as a list) and a list of R(typically around 2000) object proposals
- When using an image pyramid, each RoI is assigned to the scale such that the scaled RoI is closest to 224^2 pixels in area
- For each test ROI, a probability distribution (p) and a set of predicted bounding-box offsets are predicted (for each class k)
- Detection confidence is assigned for each class
- Non-maximum suppression is performed for each class (like RCNN)

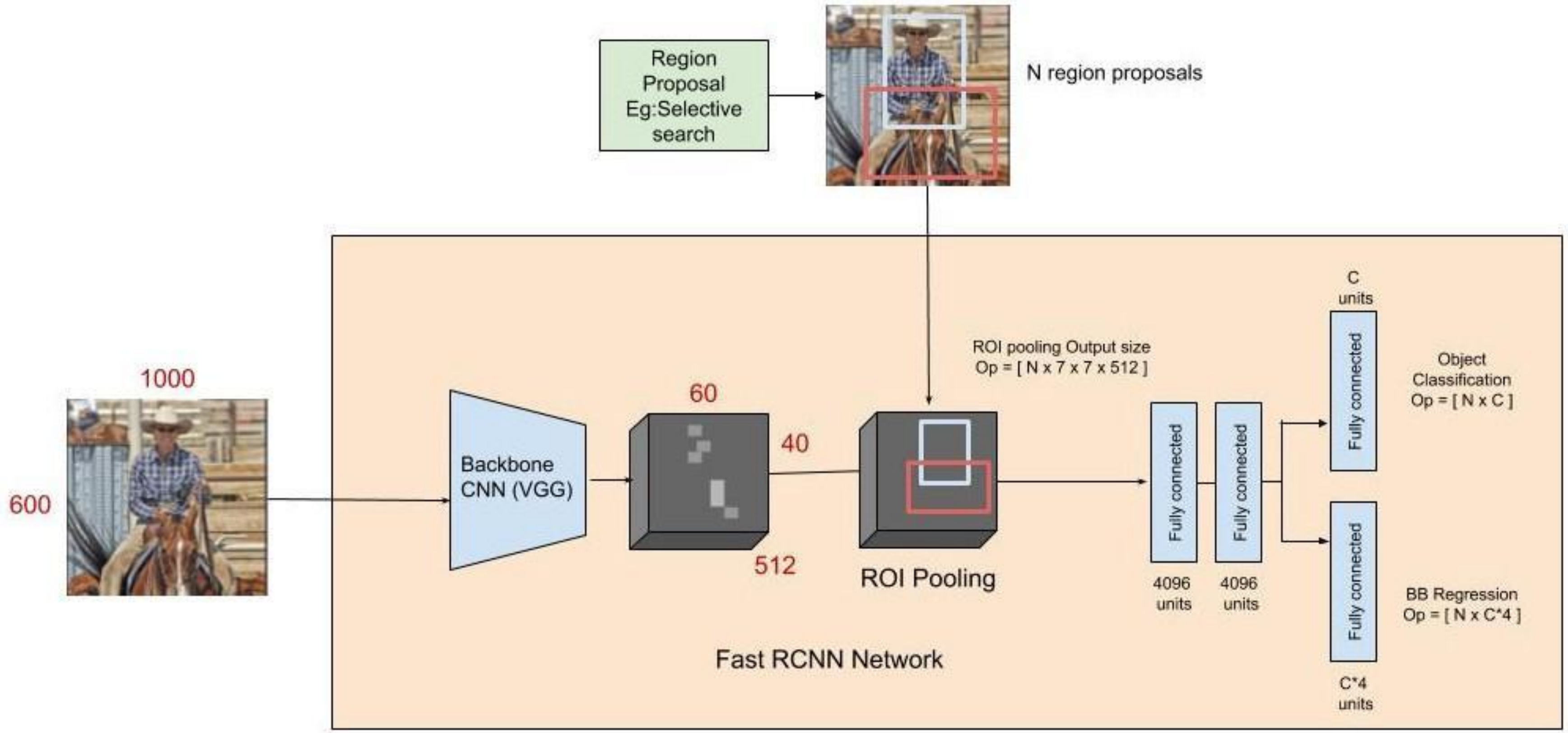
Truncated SVD

$$W \approx U \Sigma_t V^T$$

[u x v] = [u x t][t x t][t x v]

- Large fully connected layers are easily accelerated by compressing them with truncated SVD
- Truncated SVD reduces the parameter count from uv to t(u + v), which can be significant if t is much smaller than min(u, v)
- To compress a network, the single fully connected layer corresponding to W is replaced by two fully connected layers, without a non-linearity between them. The first of these layers uses the weight matrix $\Sigma_t V^T$ and the second uses U

Summary



Region Proposal Eg:Selective search
N region proposals
1000
600
Backbone CNN (VGG)
60
40
512
ROI Pooling
ROI pooling Output size
Op = [N x 7 x 7 x 512]
Fully connected
Fully connected
4096 units
4096 units
C units
Fully connected
Object Classification
Op = [N x C]
Fully connected
C*4 units
BB Regression
Op = [N x C*4]
Fast RCNN Network