

Analisi di Immagini e Video (Computer Vision)

Giuseppe Manco

Outline

- Object Detection
- Region Proposal Networks
- Single-Shot Detection
- Yolo

Crediti

- Slides adattate da vari corsi e libri
 - Computational Visual Recognition (V. Ordonez), CS Virginia Edu
 - Computer Vision (S. Lazebnik), CS Illinois Edu
 - Mohamed Elgendy [Elg20]
 - <https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection>
 - <https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/>

Image Classification



dog

Multi-label classification



dog

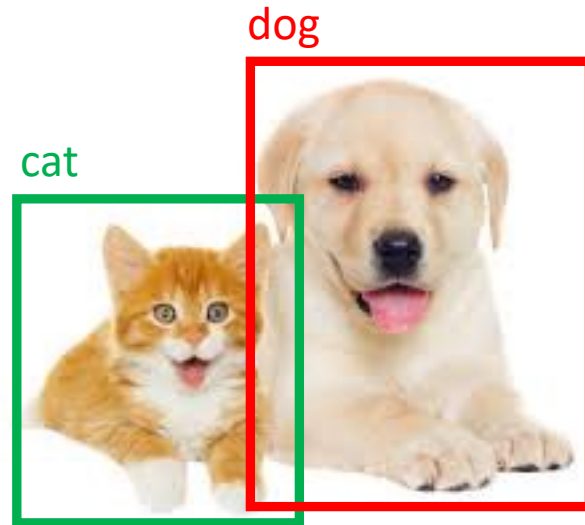


dog,cat

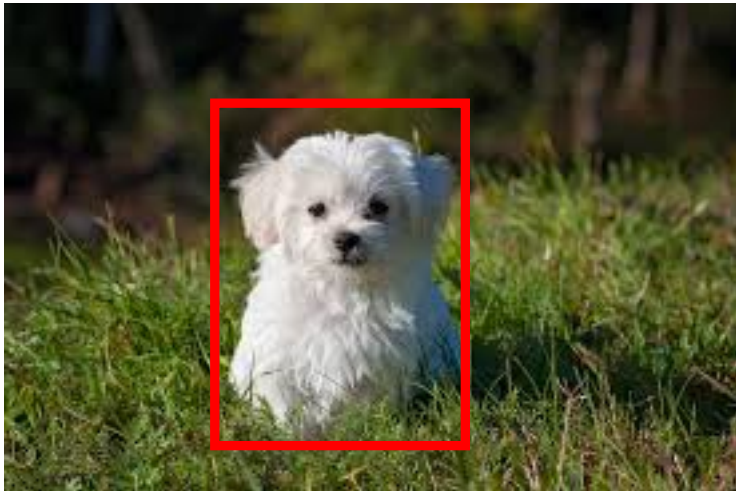
Single-Object Detection



Multi-Object Detection



Cosa caratterizza la posizione di un oggetto?



Cosa caratterizza la posizione di un oggetto?

- Boundary coordinates
 - $(x_1, y_1), (x_2, y_2)$

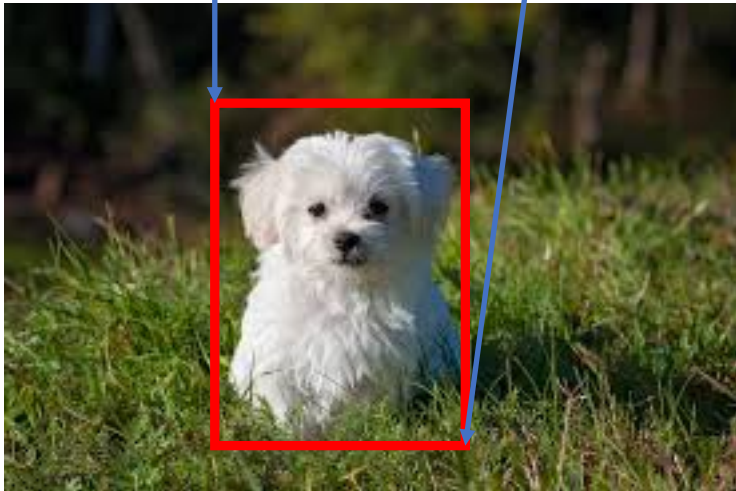


Image Classification, object detection

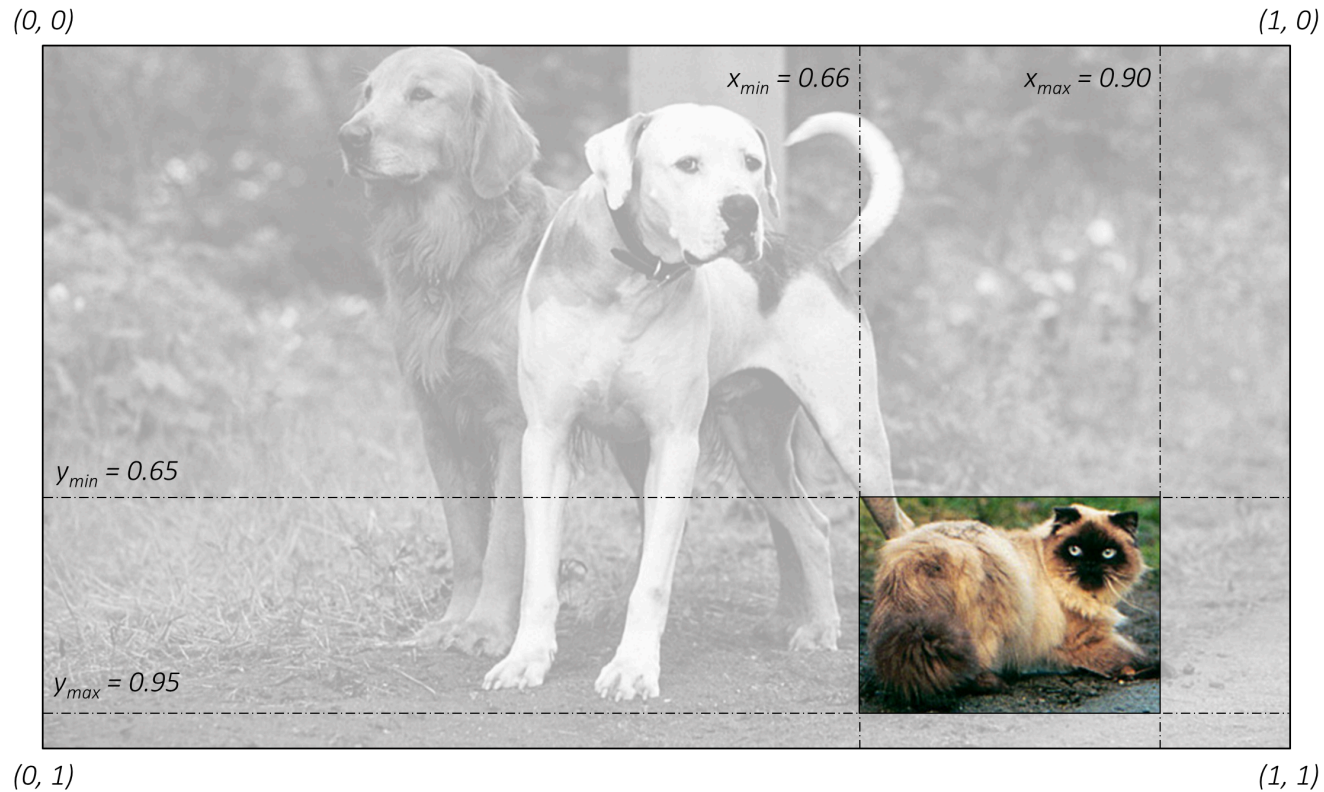
Image classification	Object detection
<ul style="list-style-type: none">- What?- Input: un'immagine con uno o più oggetti- Output: class label(s)	<ul style="list-style-type: none">- What? + Where?- input: un'immagine con uno o più oggetti- Output:<ul style="list-style-type: none">- uno o più bounding boxes- Un'etichetta di classe per ogni bounding box

Bounding boxes



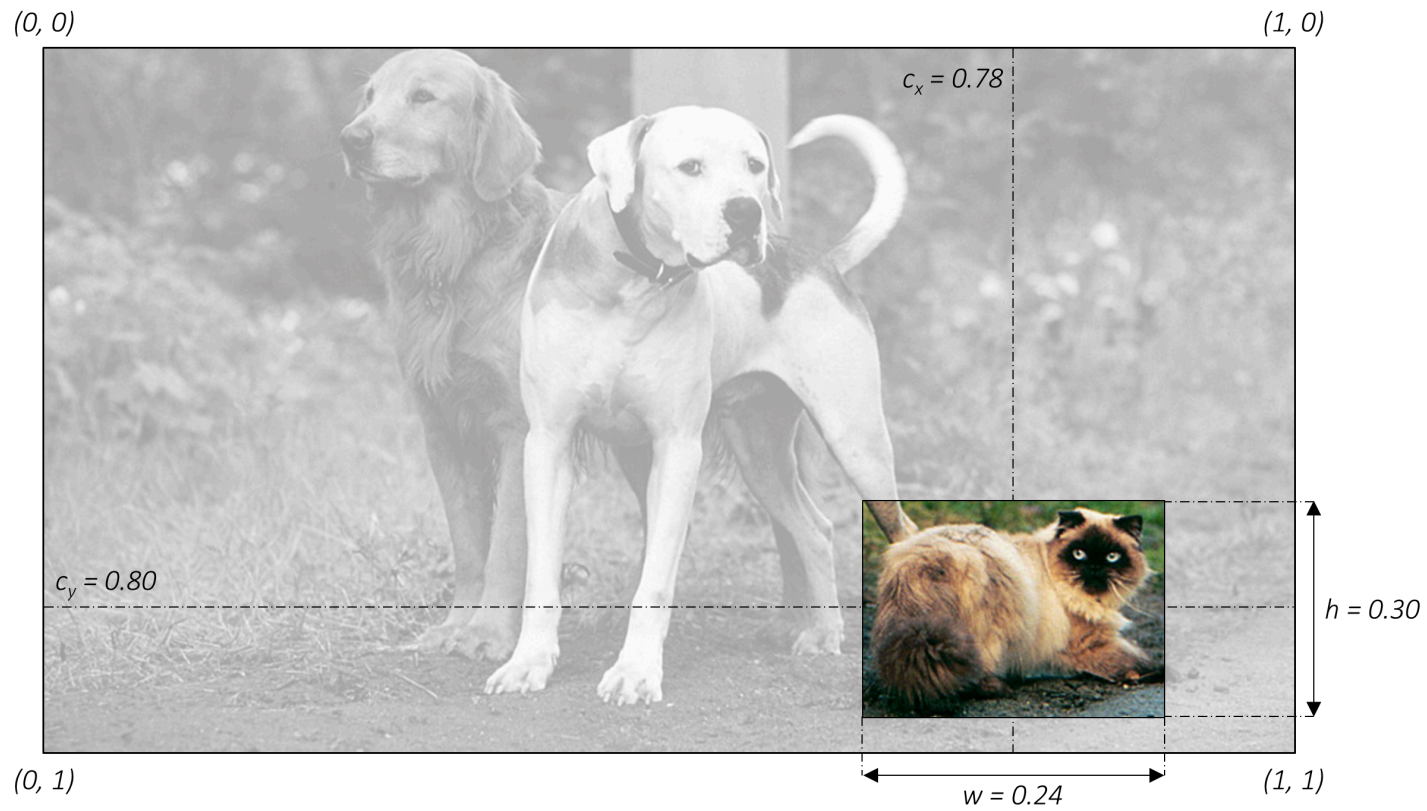
Boundary Coordinates $(x_{min}, y_{min}, x_{max}, y_{max}) = (640, 356, 870, 520)$

Bounding boxes



Boundary Coordinates $(x_{min}, y_{min}, x_{max}, y_{max}) = (0.66, 0.65, 0.90, 0.95)$

Bounding boxes



Center-Size Coordinates $(c_x, c_y, w, h) = (0.78, 0.8, 0.24, 0.30)$

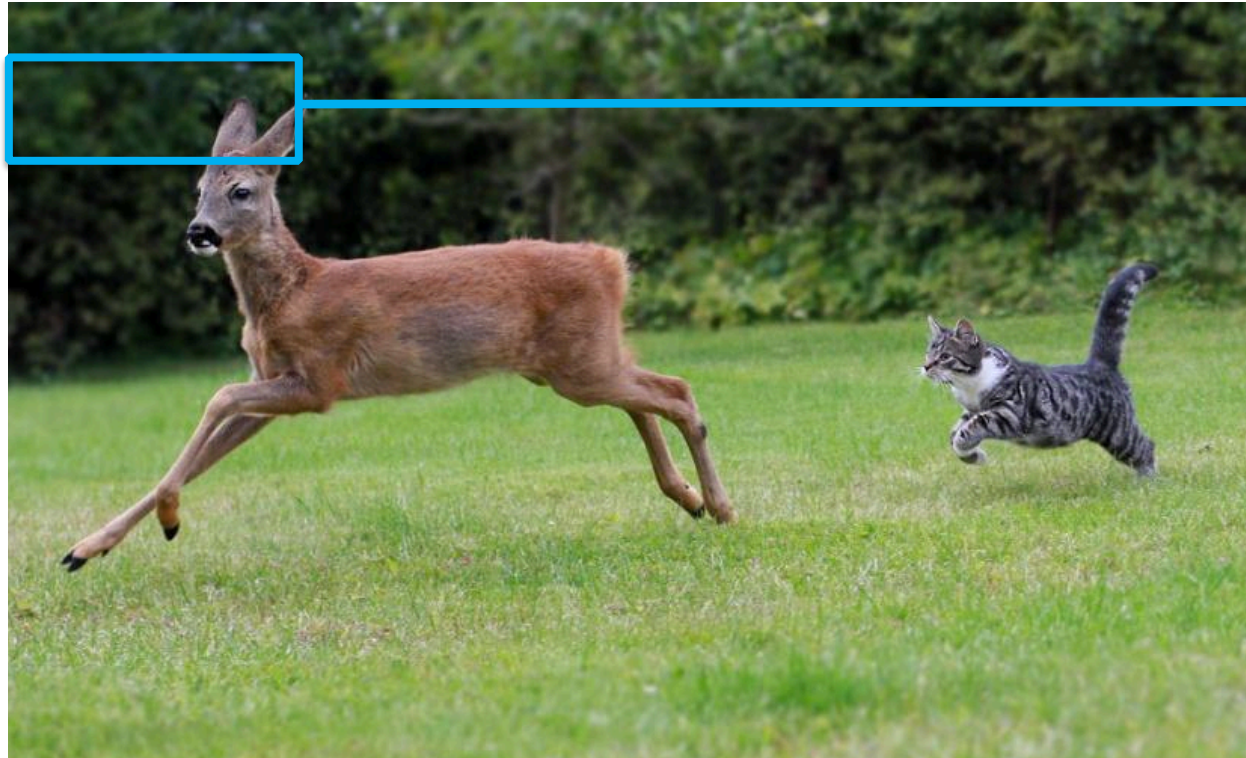
Image Classification, object detection

Image classification	Object detection
<ul style="list-style-type: none">- What?- Input: un'immagine con uno o più oggetti- Output: class label(s)	<ul style="list-style-type: none">- What? + Where?- input: un'immagine con uno o più oggetti- Output:<ul style="list-style-type: none">- uno o più bounding boxes<ul style="list-style-type: none">- (x, y, w, h)- Un'etichetta di classe per ogni bounding box

Approcci all'object detection

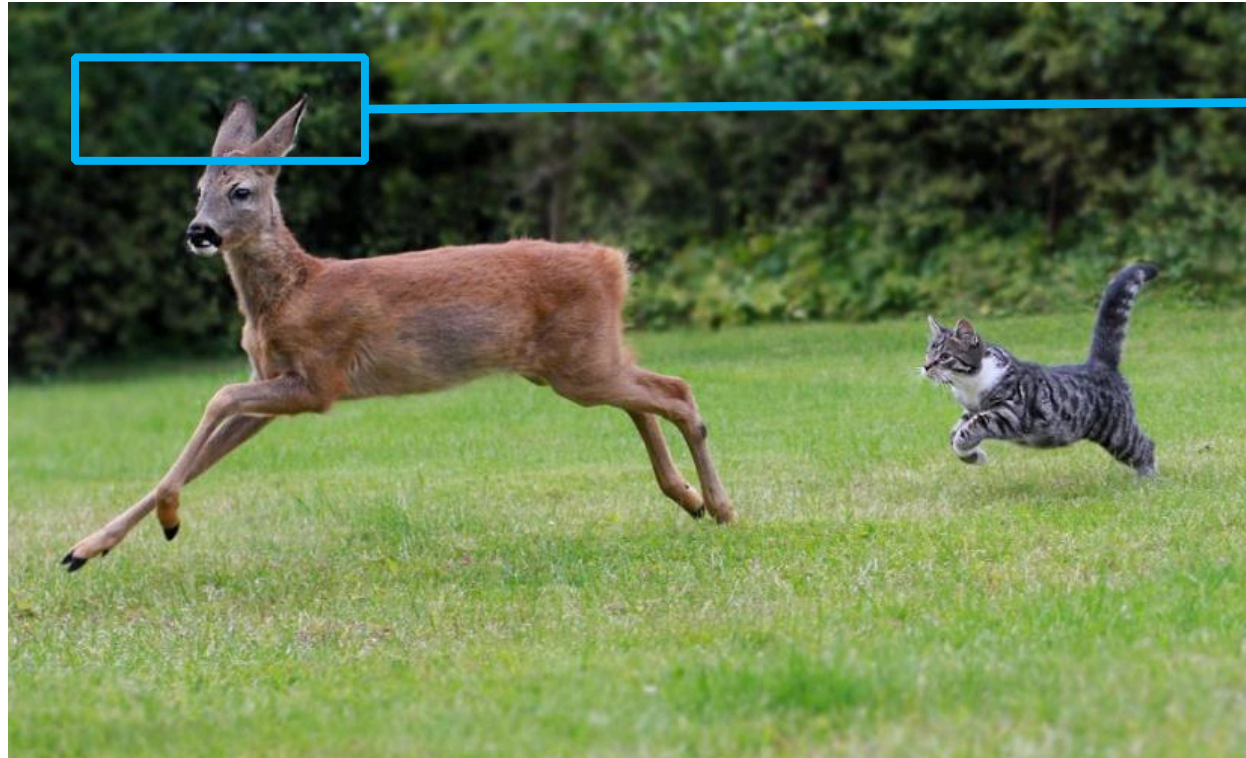
- Si esamina ogni posizione/scala
- Si utilizza qualche meccanismo propositivo per analizzare un insieme di regioni candidate

Sliding windows



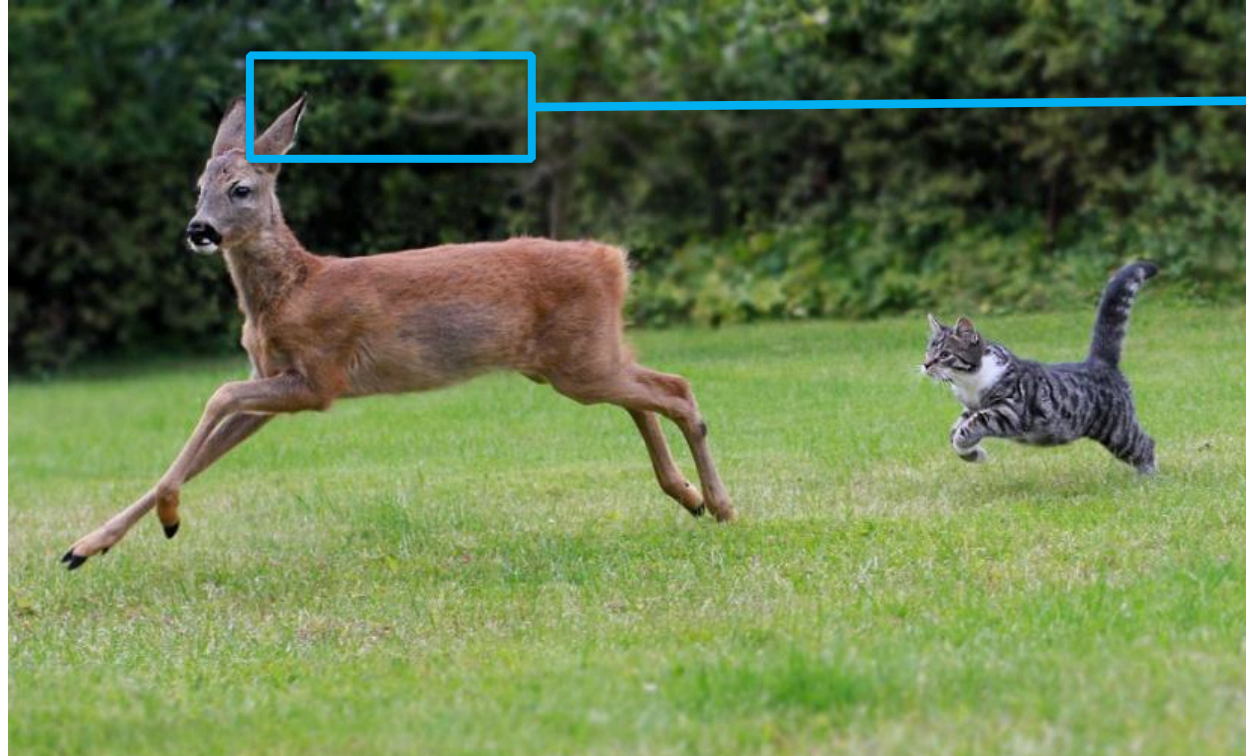
deer?
cat?
background?

Sliding windows



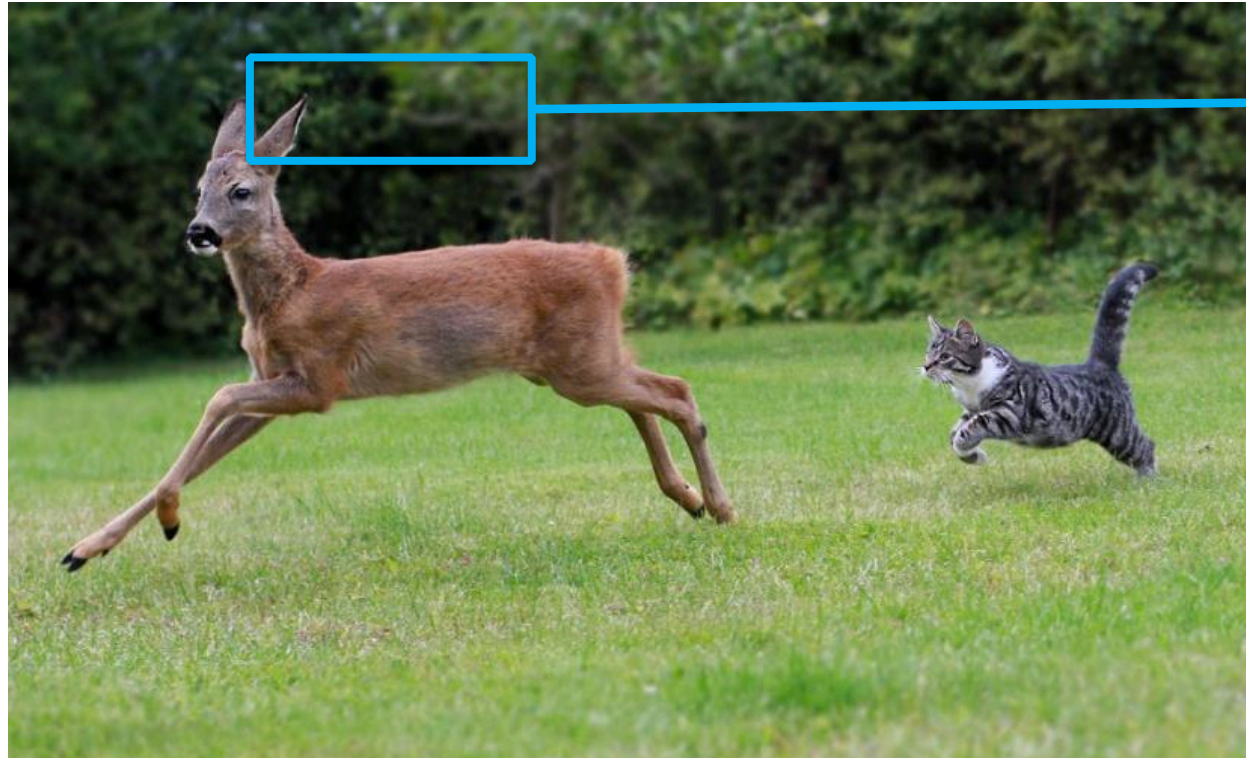
deer?
cat?
background?

Sliding windows



deer?
cat?
background?

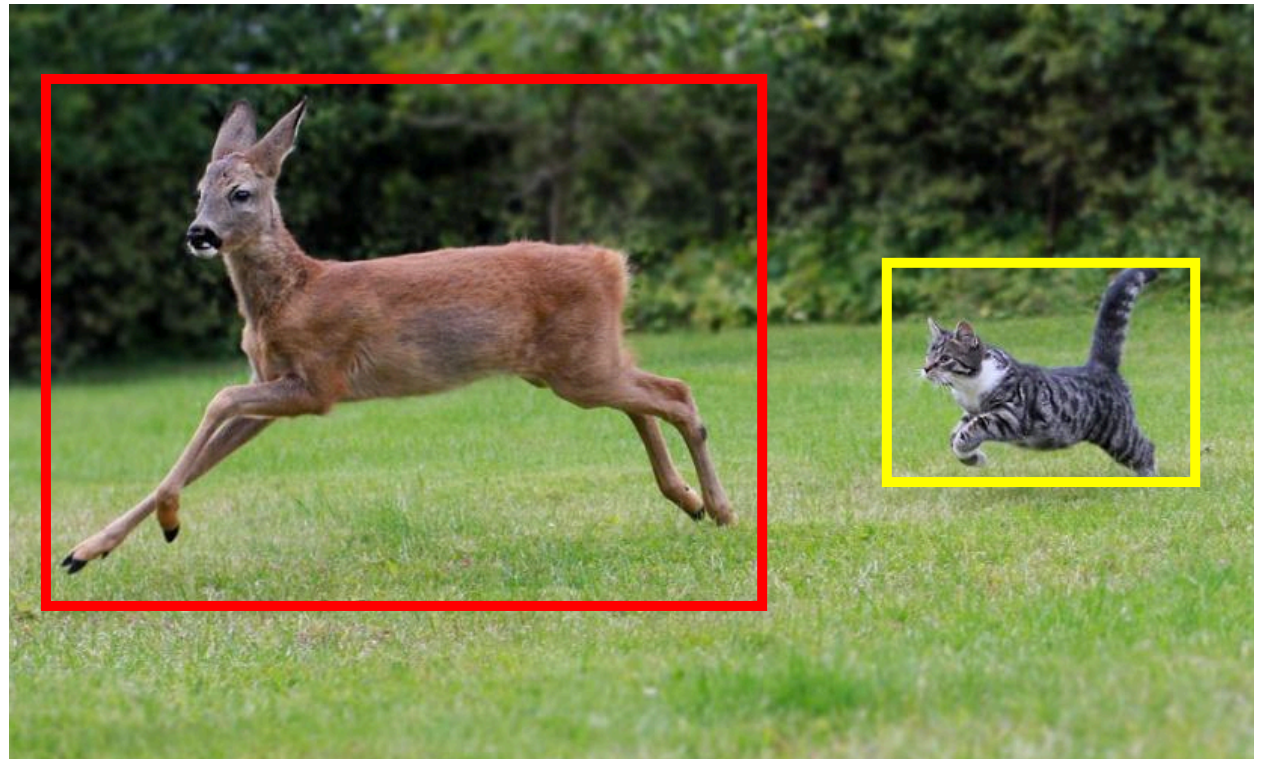
Sliding windows



deer?
cat?
background?

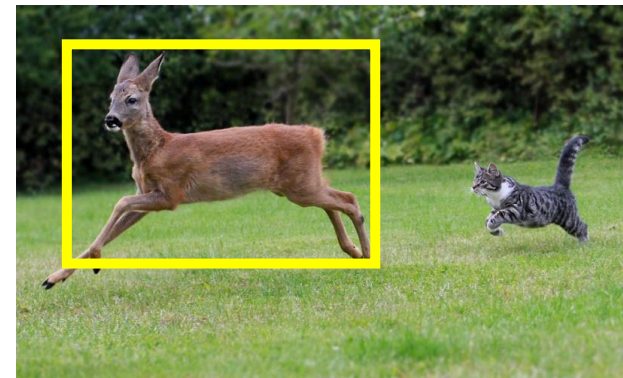
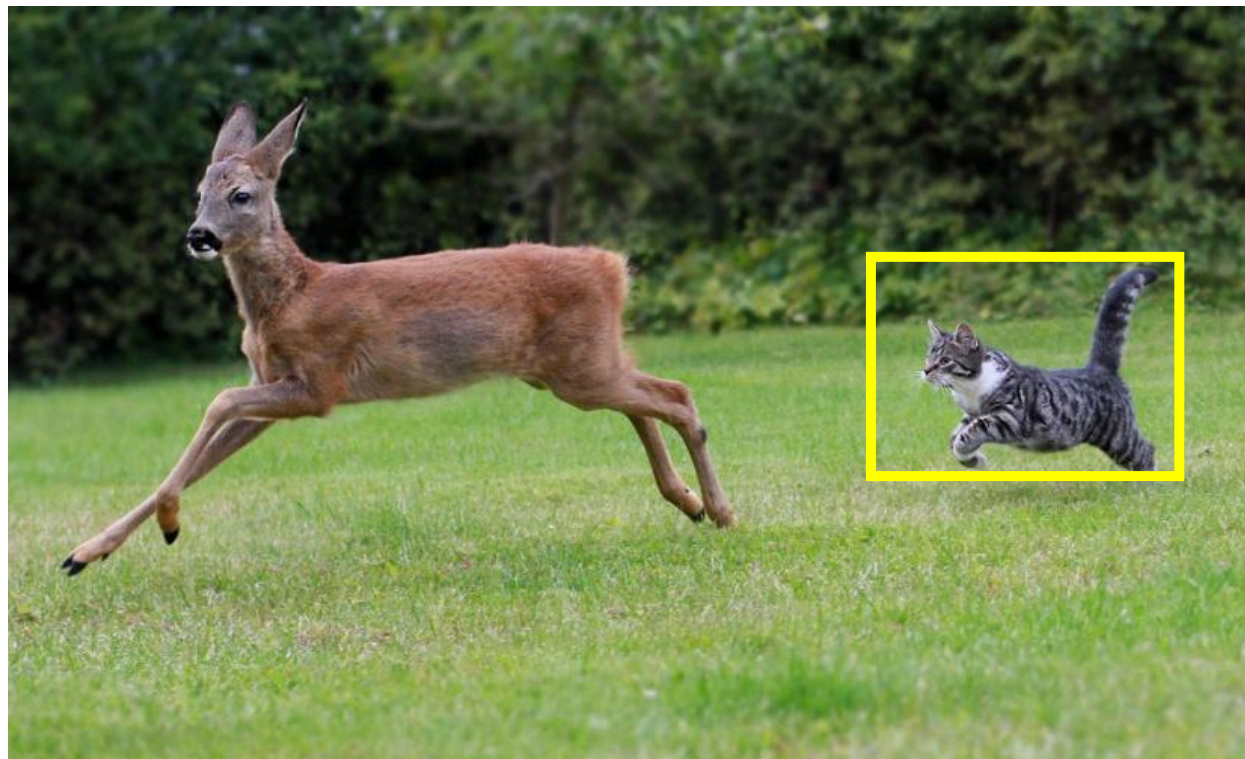
Sliding windows

- Esplosione combinatoria!
 - Dimensioni delle patch
 - Quantità di patch



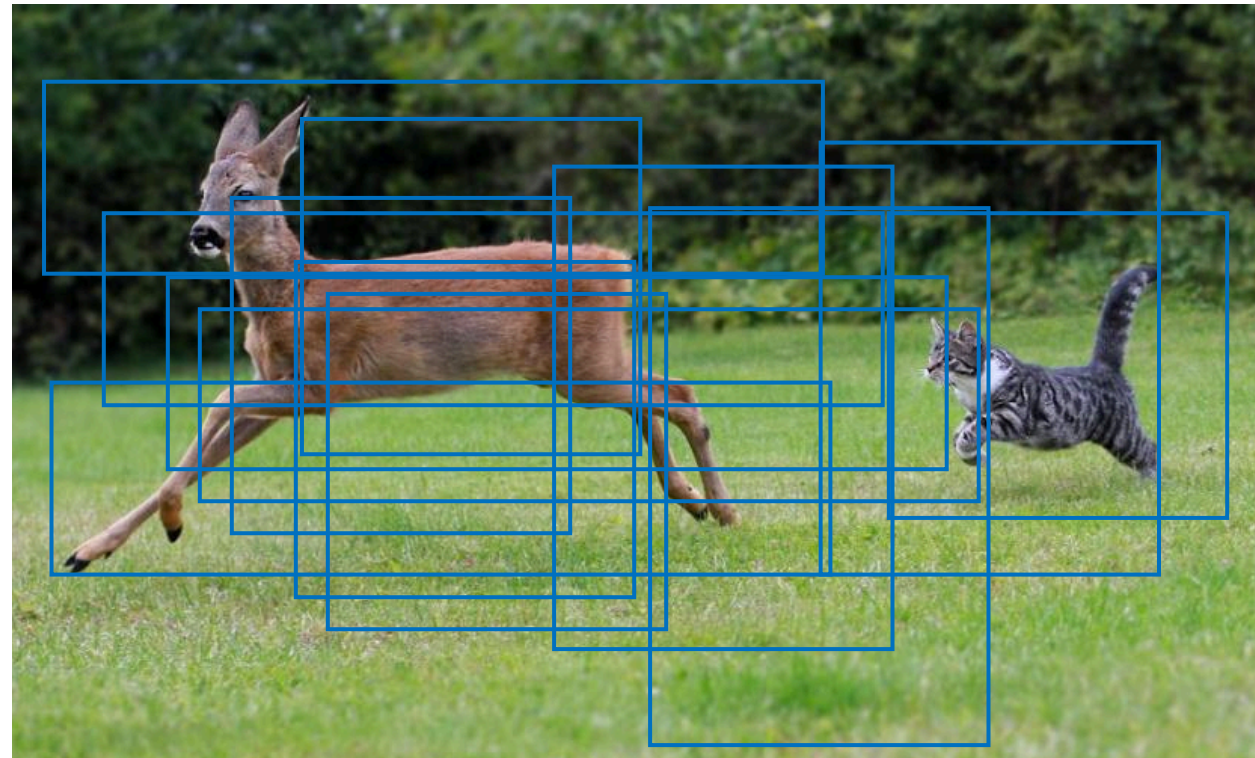
Approccio Naive

- Soluzione parziale: pyramids



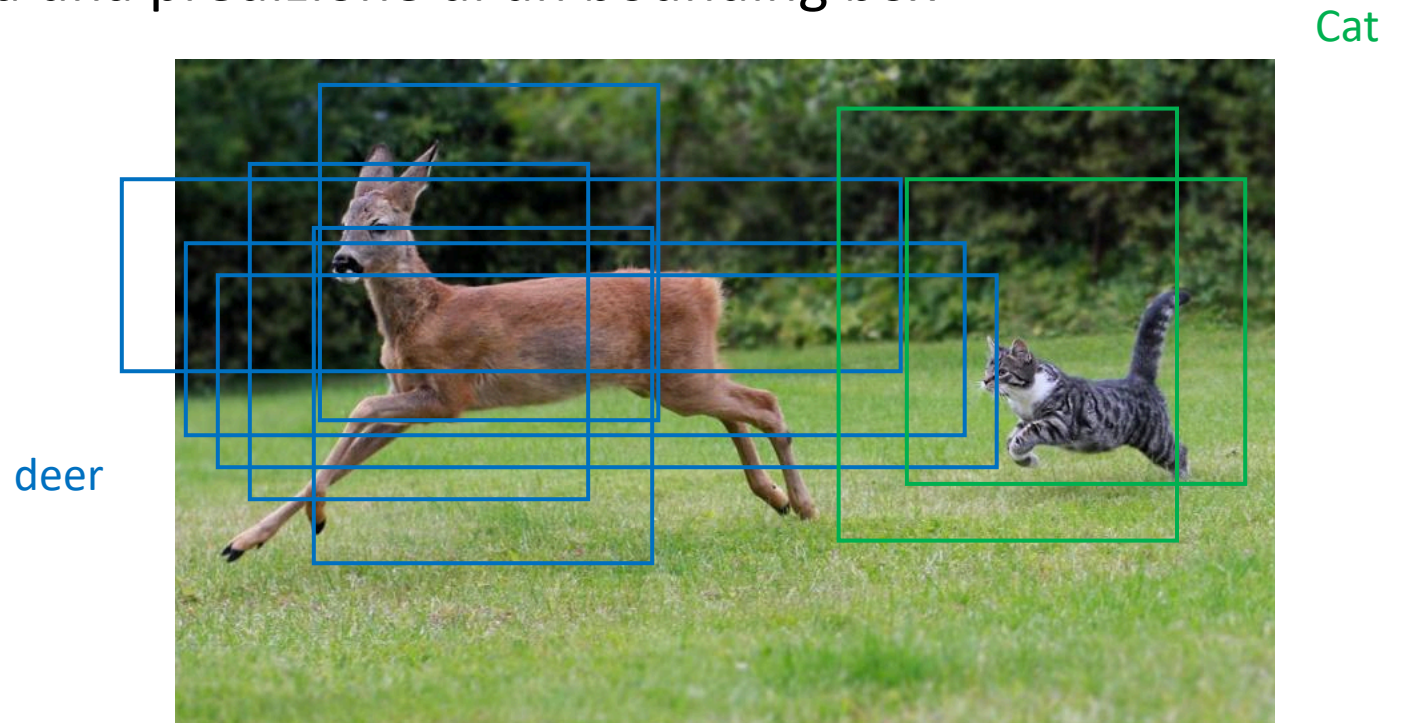
Region Proposals

- Si generano Regioni di interesse (ROI)



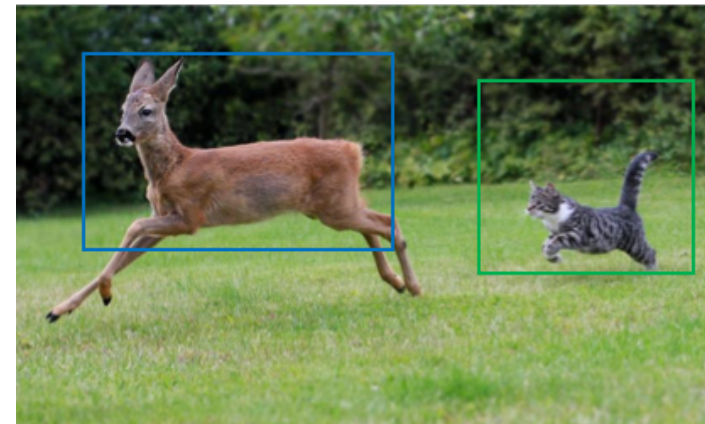
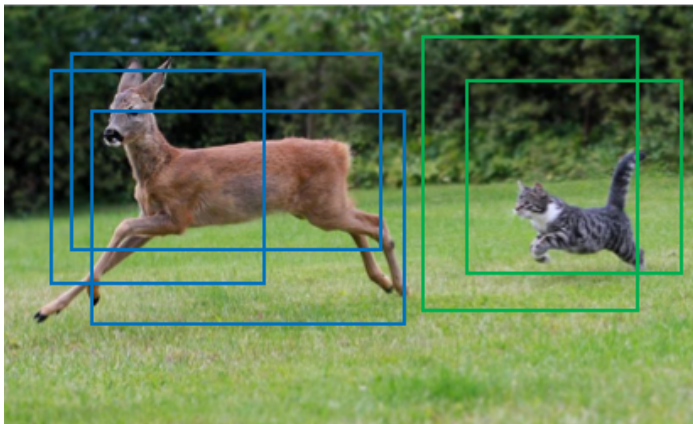
Region Proposals

- Si generano Regioni di interesse (ROI)
 - Algoritmo/modello di deep learning
- Feature extraction e predizioni sulle ROI
 - Su ogni ROI viene effettuata una predizione di un bounding box



Region Proposals

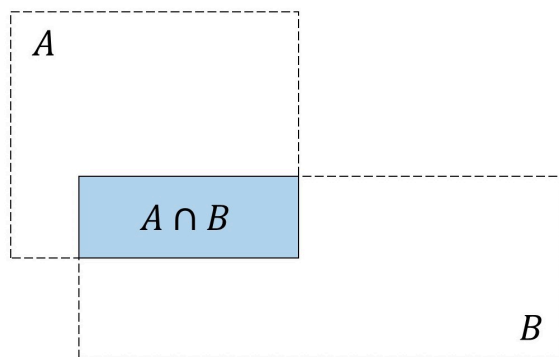
- Si generano Regioni di interesse (ROI)
 - Algoritmo/modello di deep learning
- Feature extraction e predizioni sulle ROI
 - Su ogni ROI viene effettuata una predizione di un bounding box
- Non-Maximum Suppression
 - I bounding box non ottimali vengono rimossi



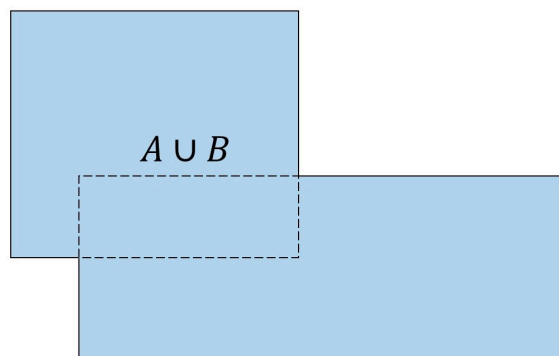
Qualità di un object detector

- FPS
 - frames per second (FPS)
 - Quante immagini al secondo si è in grado di analizzare
- Precision/Recall, mean Average Precision (mAP)
 - Intersection over Union

Intersection over Union

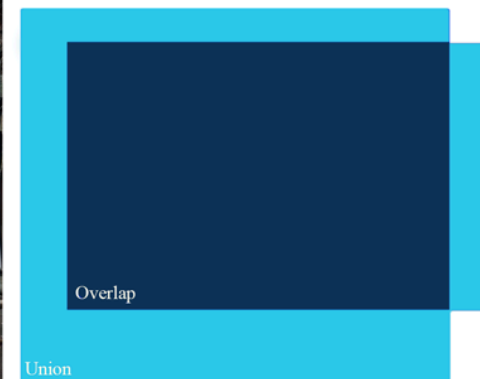


$$IoU = \frac{A \cap B}{A \cup B} =$$



-  Ground truth
-  Prediction

$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$



Precision, Recall

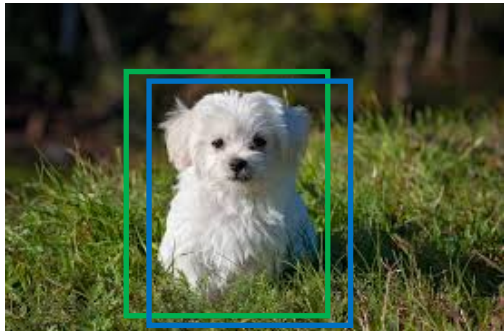
- True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN)

$$Prec = \frac{TP}{TP + FP}$$

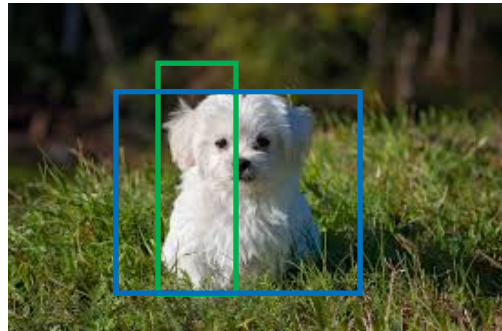
$$Rec = \frac{TP}{TP + FN}$$

- Riferito ad un match
 - Dato un bounding box predetto e uno «ground truth»
 - True Positive se $IoU > \theta$, con θ valore di soglia, altrimenti False Positive

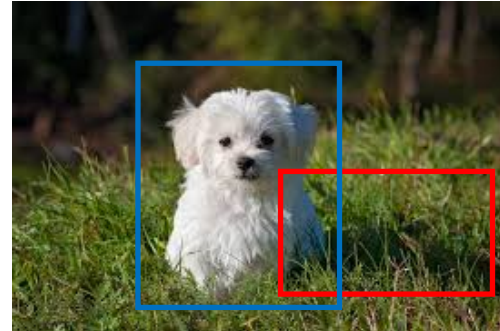
TP



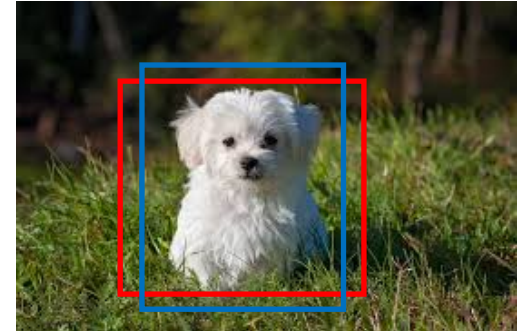
FP



TN



FN



Precision/Recall

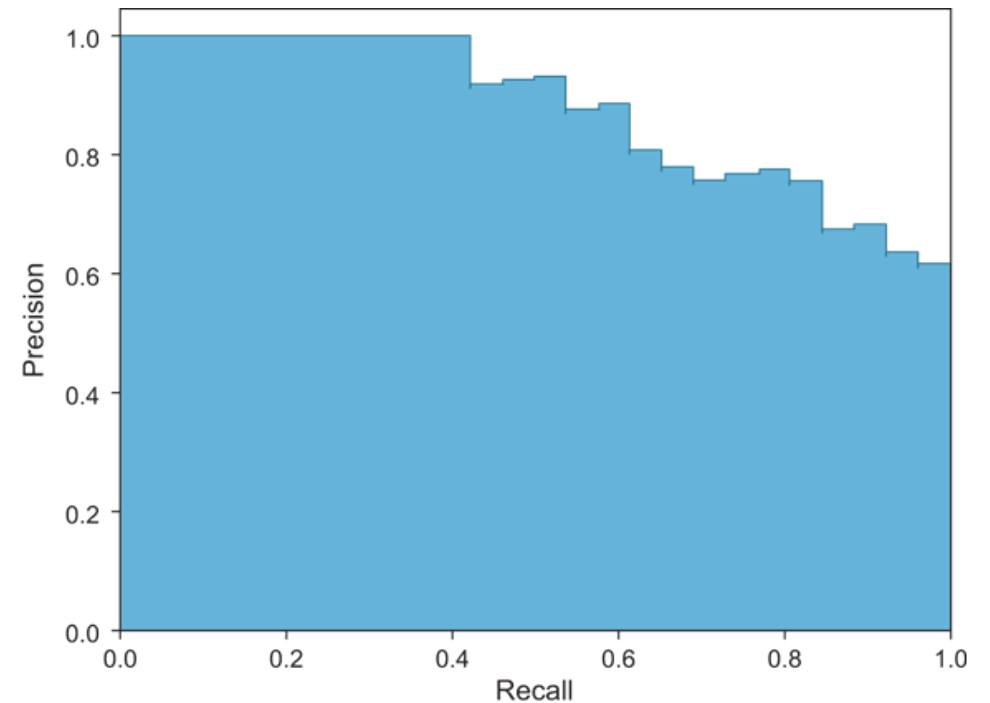
- Valori di Precision/recall al variare di θ , riferito agli oggetti di classe c

$$AP_c = \int_0^1 Prec_c(\theta) d\theta$$

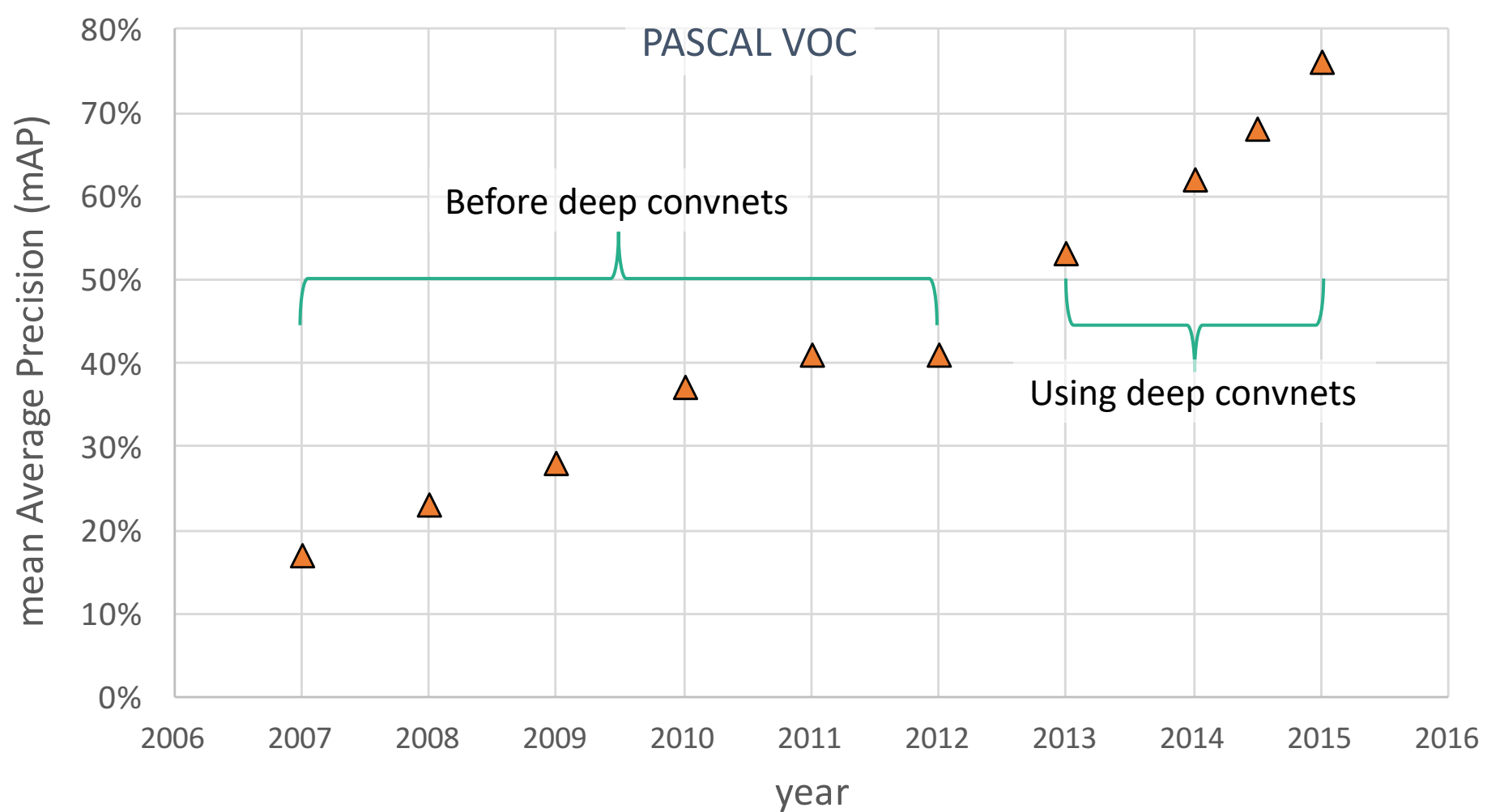
- In genere si può calcolare a gradoni, raccogliendo tutti i possibili valori di soglia

$$mAP = avg_c(AP_c)$$

$$mAP@t = avg_c(Prec_c(\theta))$$



Evoluzione

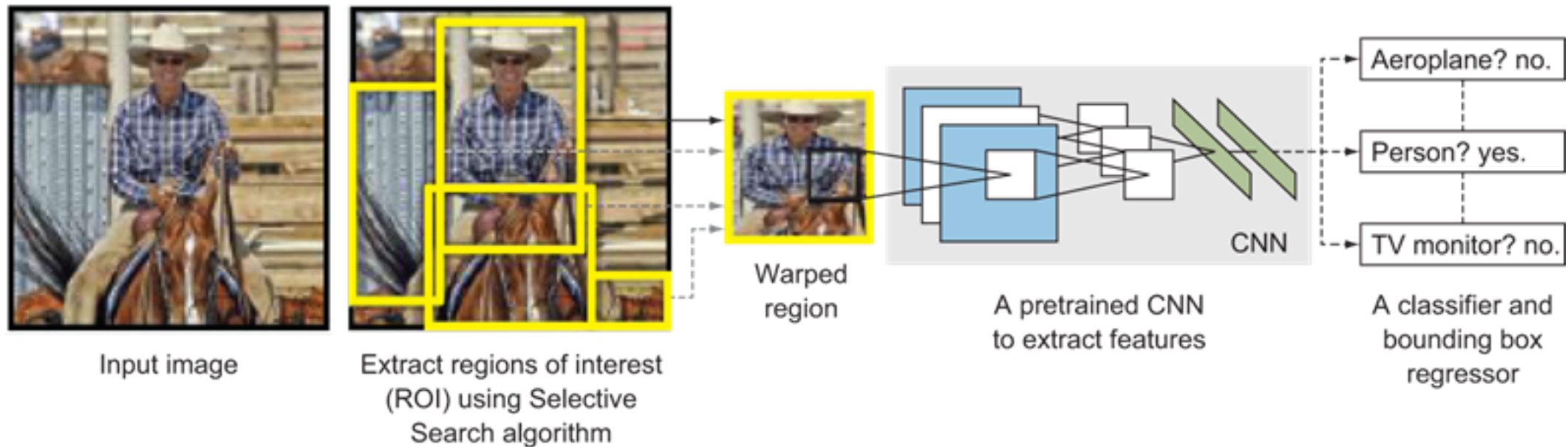


Approcci

- Multi-shot detectors
 - R-CNN
 - Fast R-CNN
 - Faster R-CNN
- Single-Shot detectors
 - SSD
 - YOLO

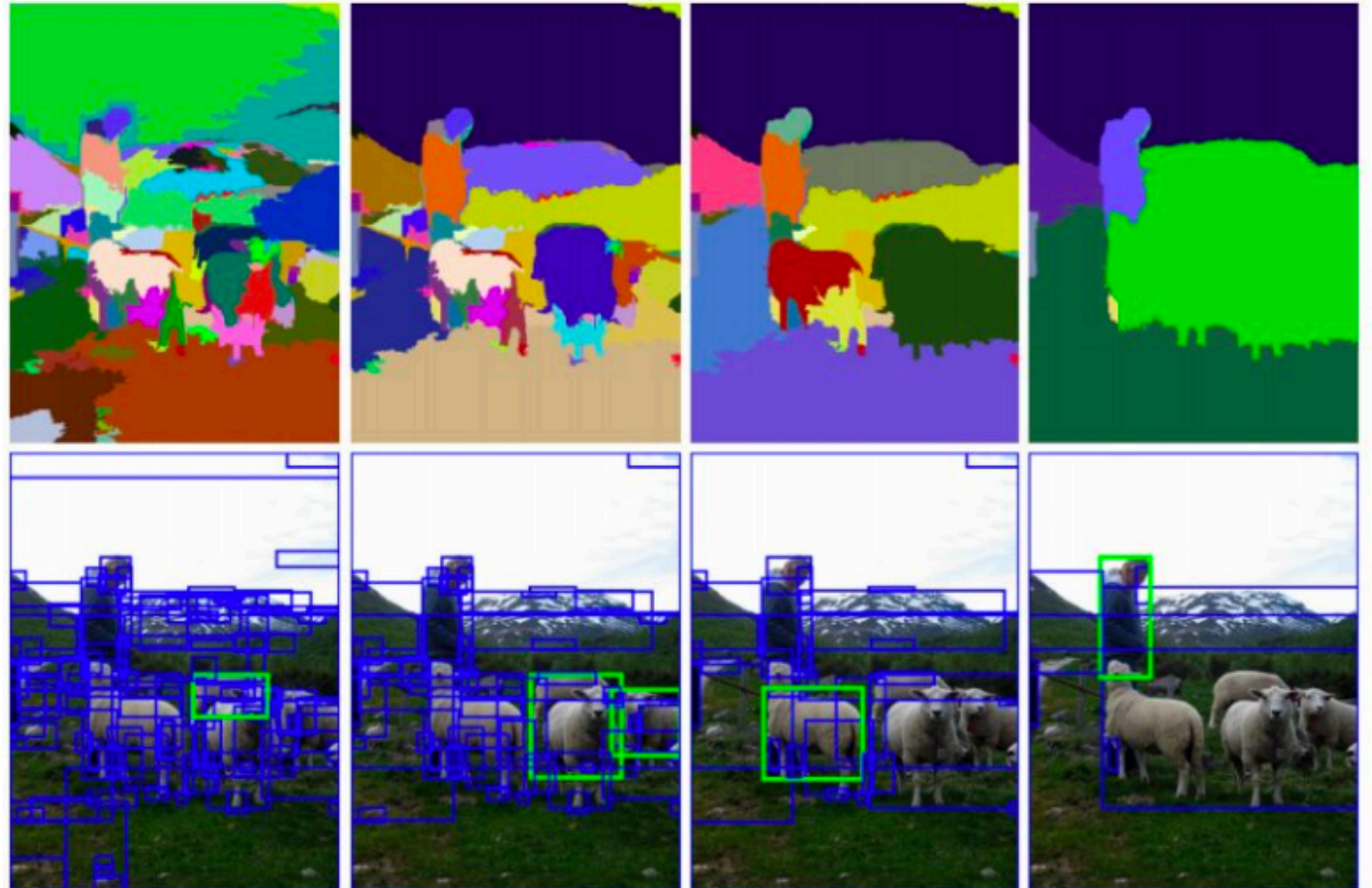
Region-Based Convolutional Neural Networks (R-CNN)

- Ross Girshick et al., 2014

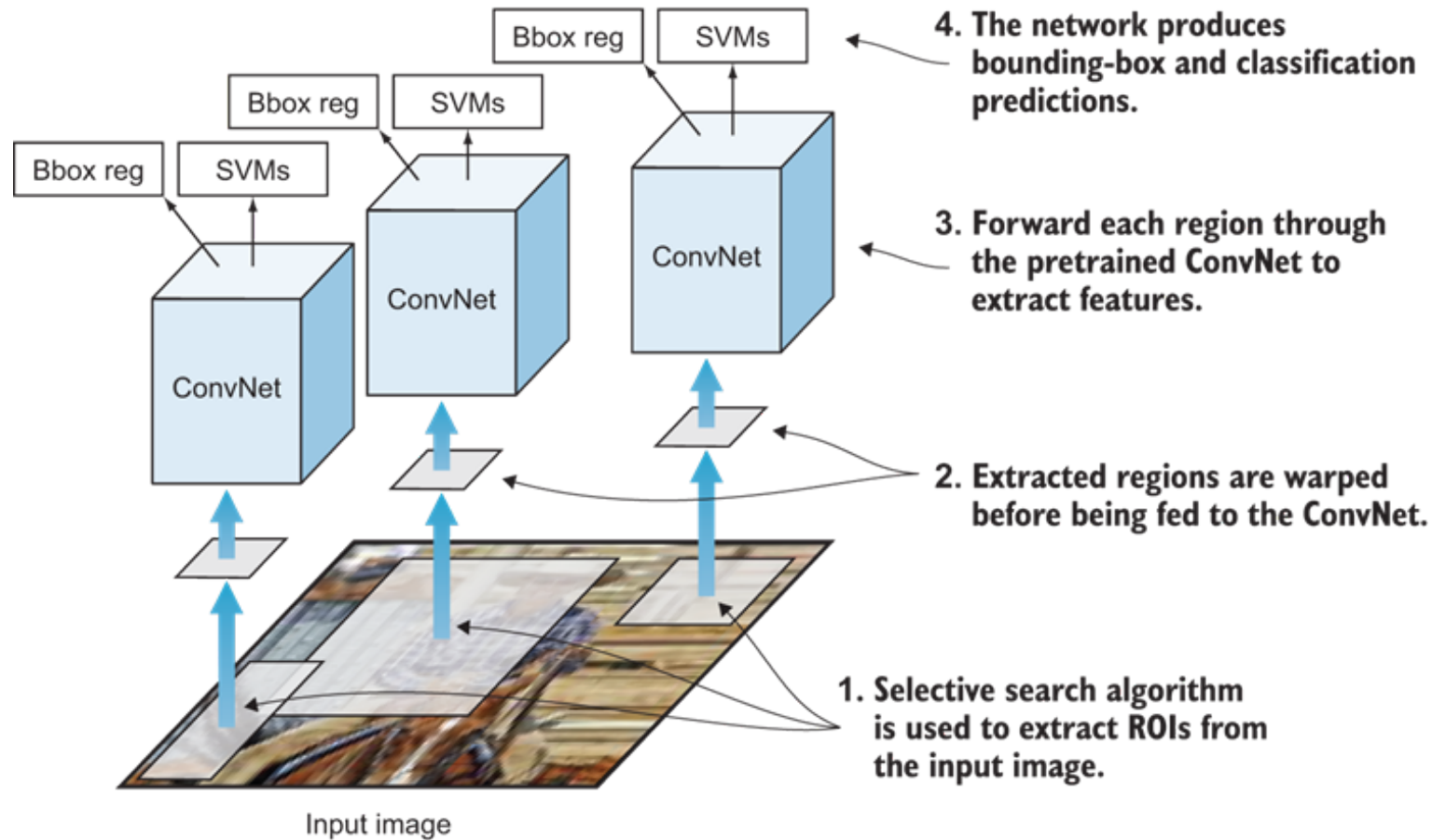


Selective Search

- Si segmenta l'immagine usando un algoritmo graph-based
 - Bounding box per ogni segmento
- Progressivamente si uniscono i bounding box sulla base della loro vicinanza e somiglianza
 - Colore, texture, ...



Feature extraction e predizione



Bounding Box Regression

- Le coordinate (c_x, c_y, w, h) possono essere espresse in termini di offset $(g_{c_x}, g_{c_y}, g_w, g_h)$ rispetto alle coordinate $(\hat{c}_x, \hat{c}_y, \hat{w}, \hat{h})$ della prior proposal region

$$g_{c_x} = \frac{c_x - \hat{c}_x}{\hat{w}} \quad g_{c_y} = \frac{c_y - \hat{c}_y}{\hat{h}}$$

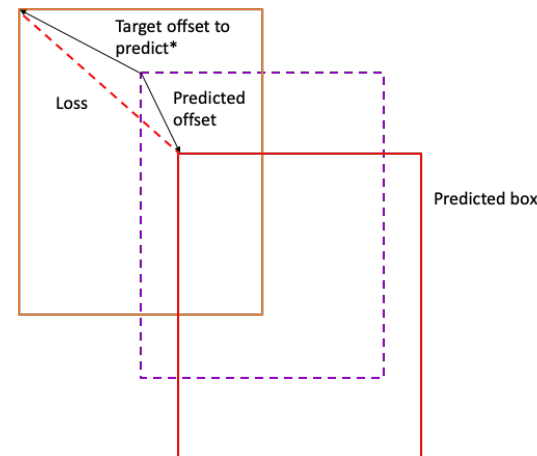
$$g_w = \log\left(\frac{w}{\hat{w}}\right) \quad g_h = \log\left(\frac{h}{\hat{h}}\right)$$

- Regressione su $(g_{c_x}, g_{c_y}, g_w, g_h)$

Prior with center-size coordinates $(\hat{c}_x, \hat{c}_y, \hat{w}, \hat{h})$



Bounding Box with center-size coordinates (c_x, c_y, w, h)

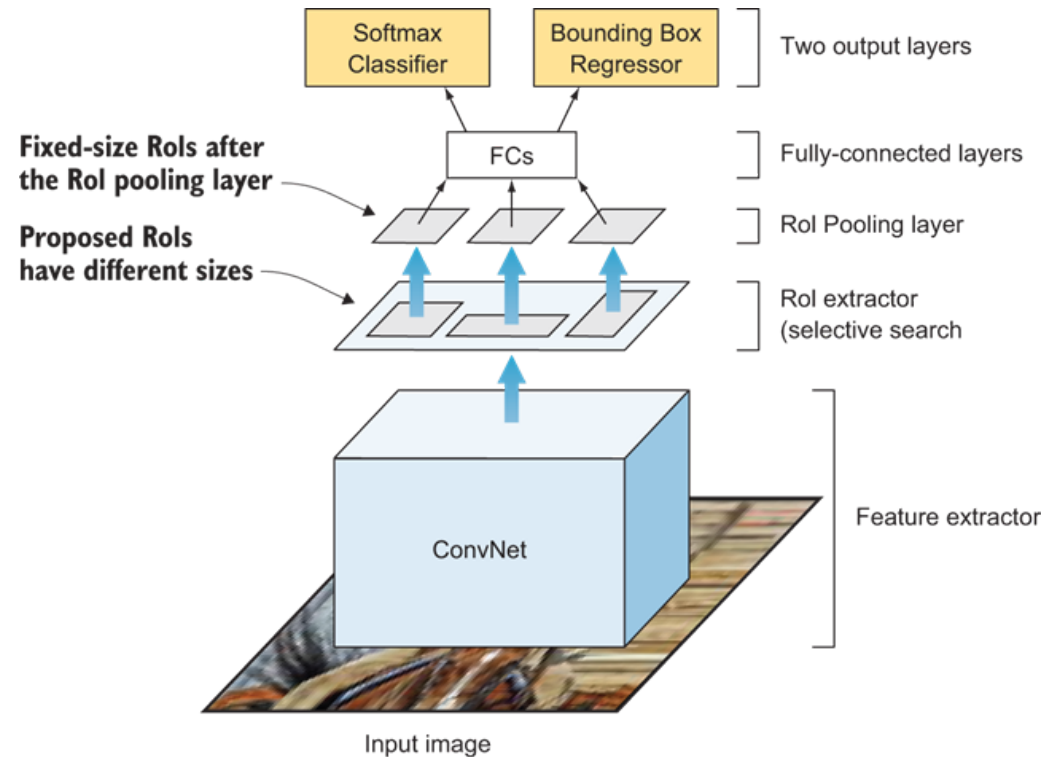


R-CNN

- Algoritmo lento
 - La fase SS genera 2000 blocchi candidati che devono essere passati al FE e al classificatore
 - $\sim 47\text{s}/\text{image} = 0.021\text{FPS}$
- Training multi-stage
 - Ogni modulo appreso separatamente
 - Estremamente lento (~ 84 ore)
 - Space complexity
- mAP 66%

Fast R-CNN

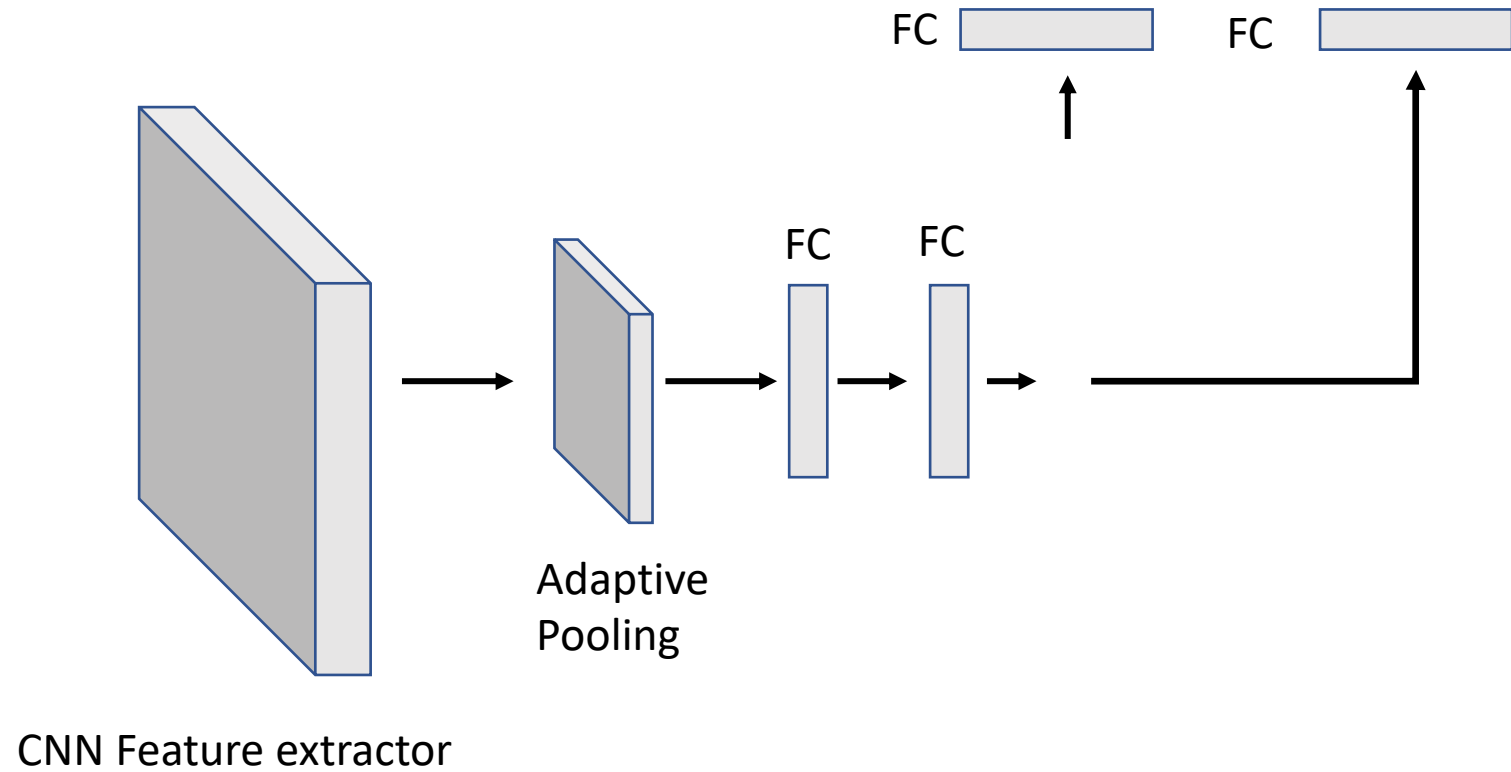
- Girshick, 2015
- Idea: utilizziamo un unico feature extractor e integriamo un modulo di classificazione



Fast R-CNN



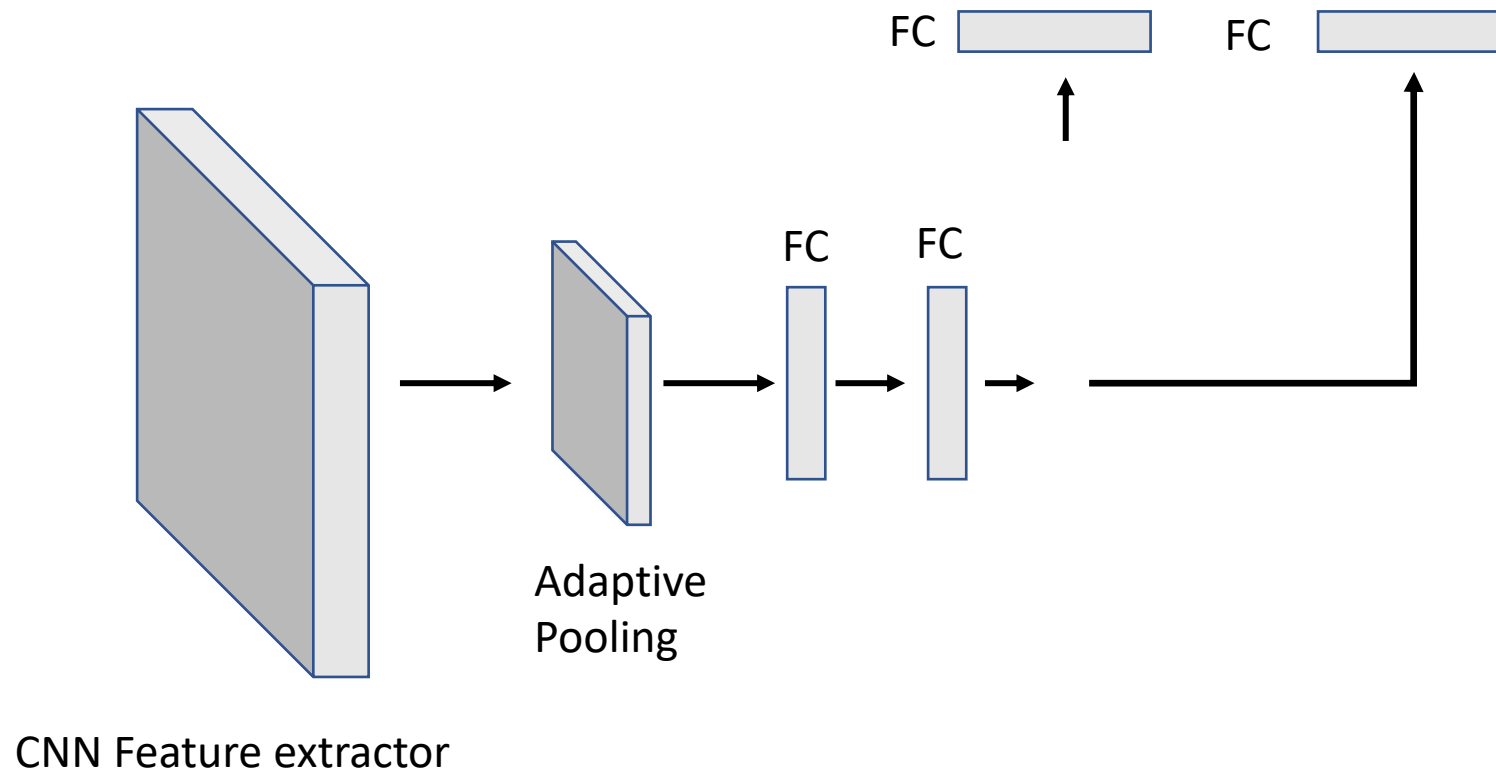
Input image



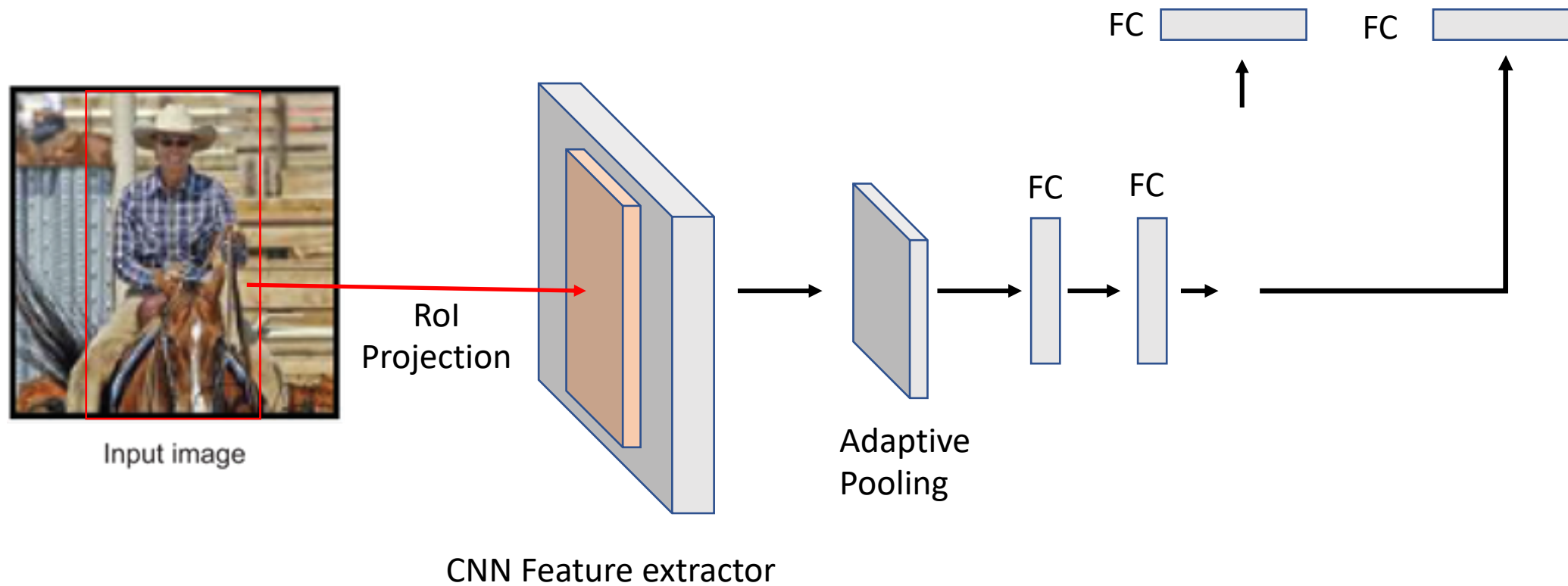
Fast R-CNN



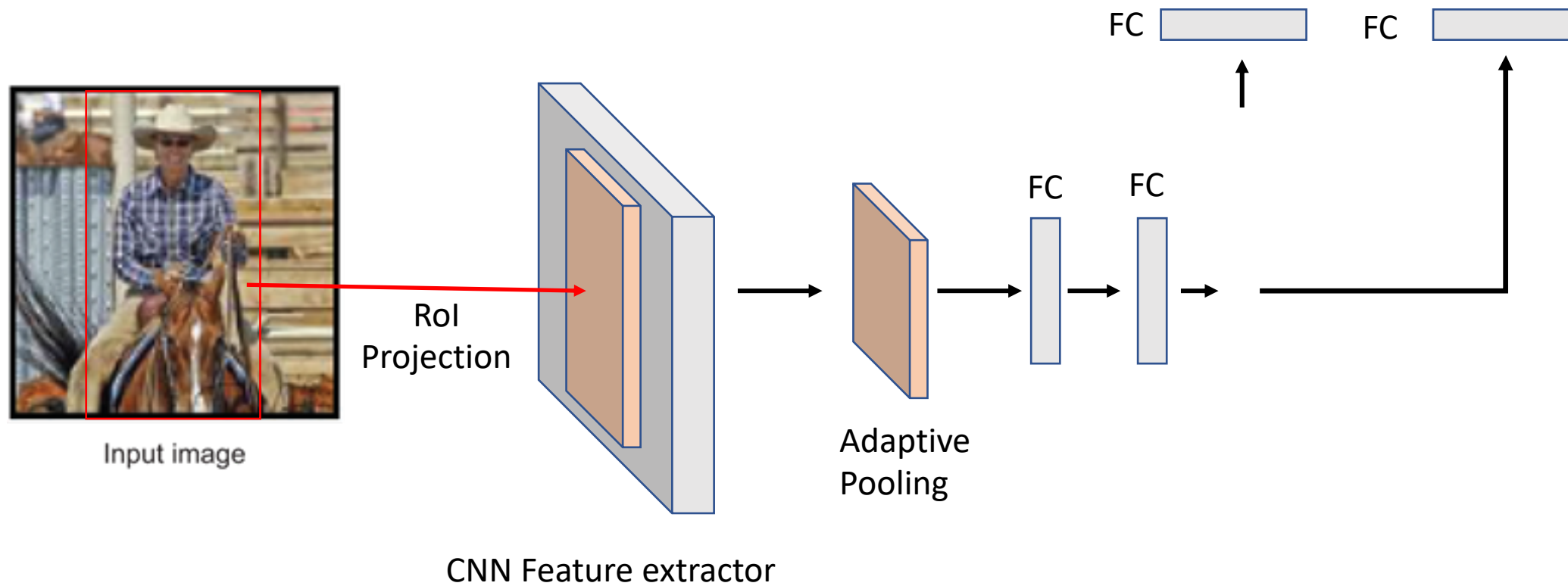
Input image



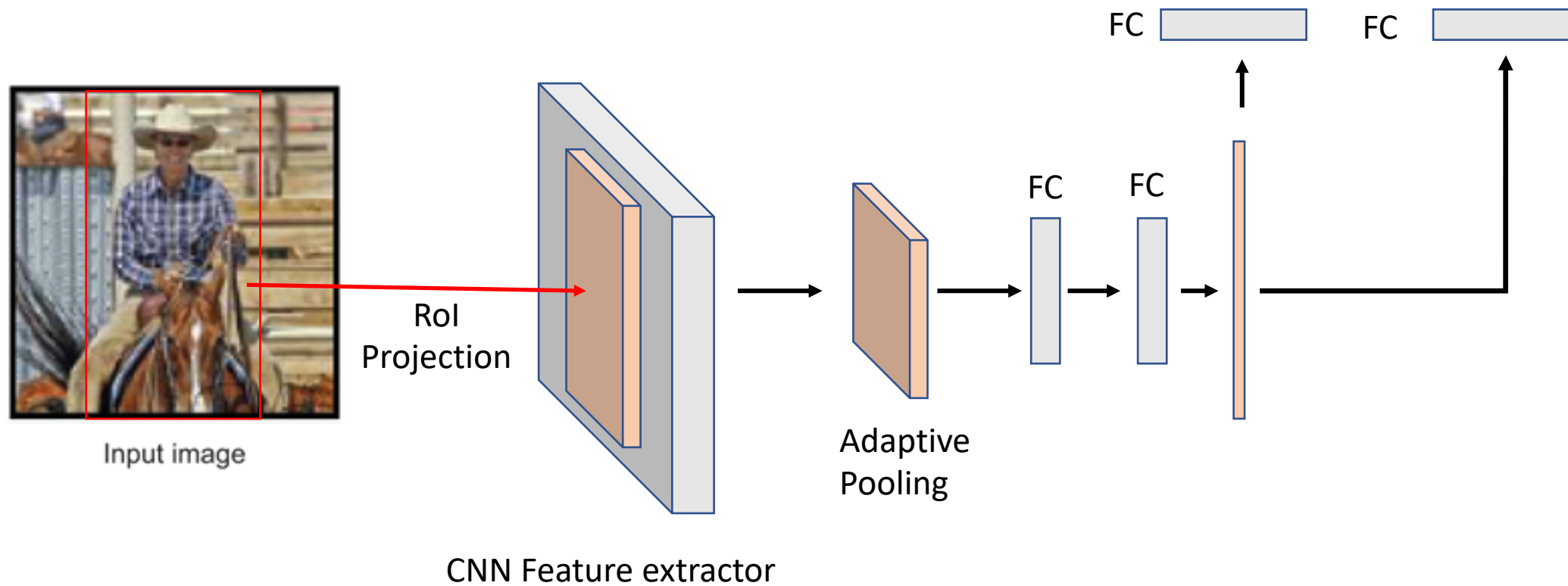
Fast R-CNN



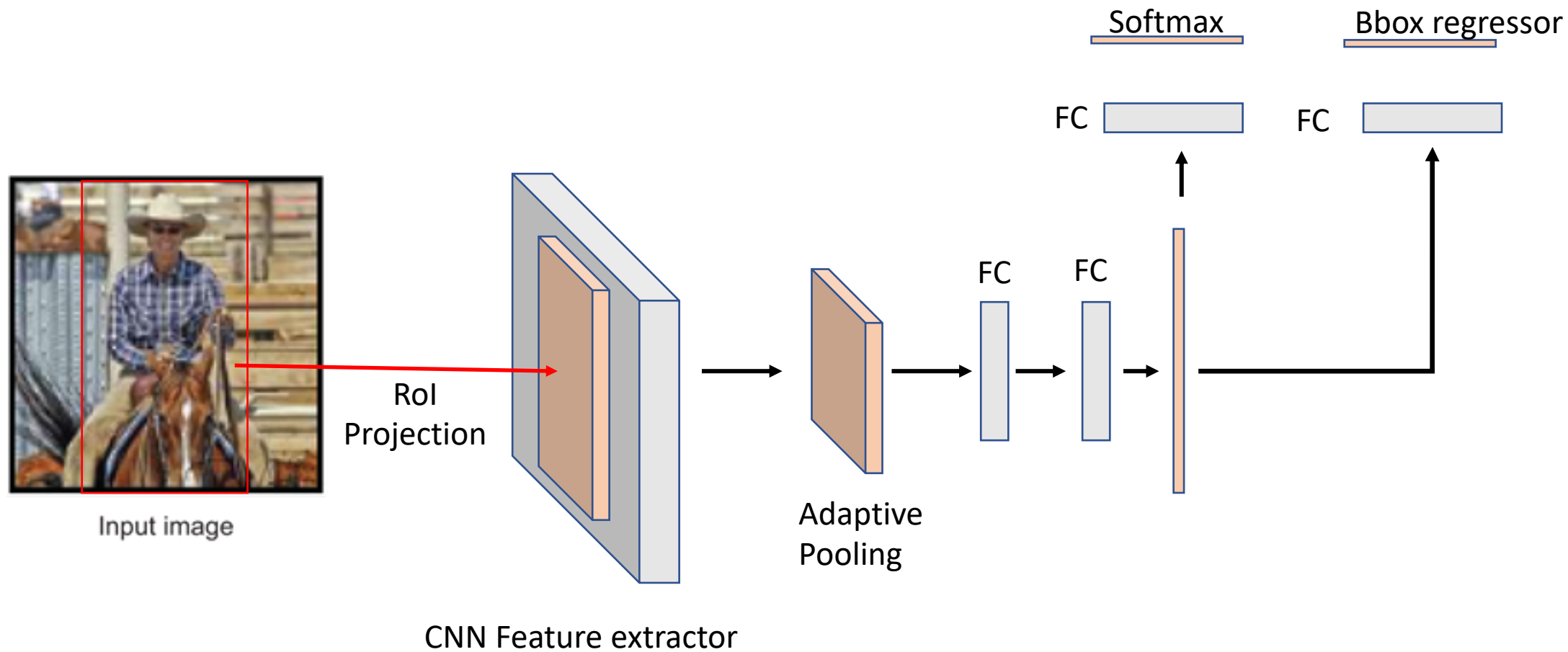
Fast R-CNN



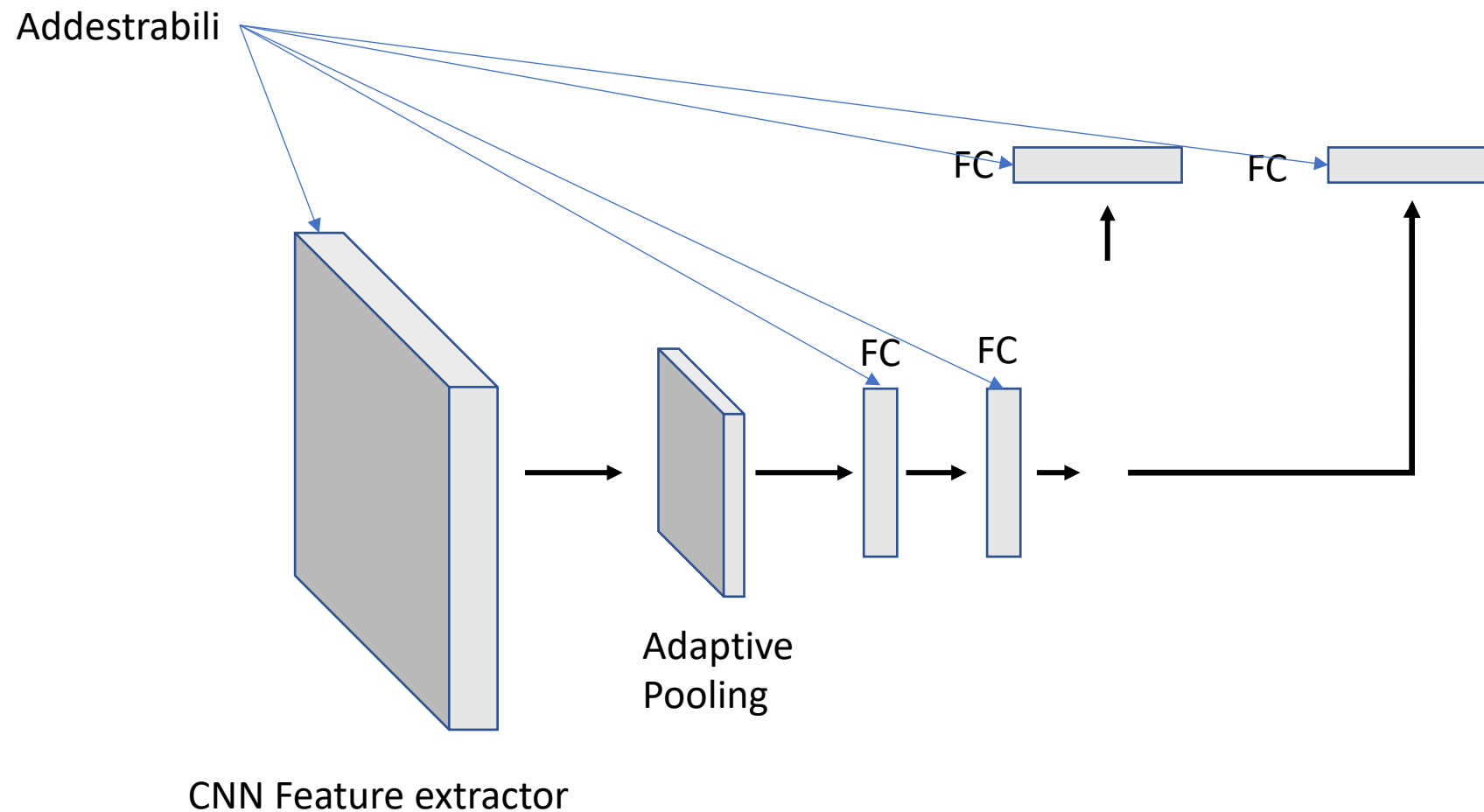
Fast R-CNN



Fast R-CNN

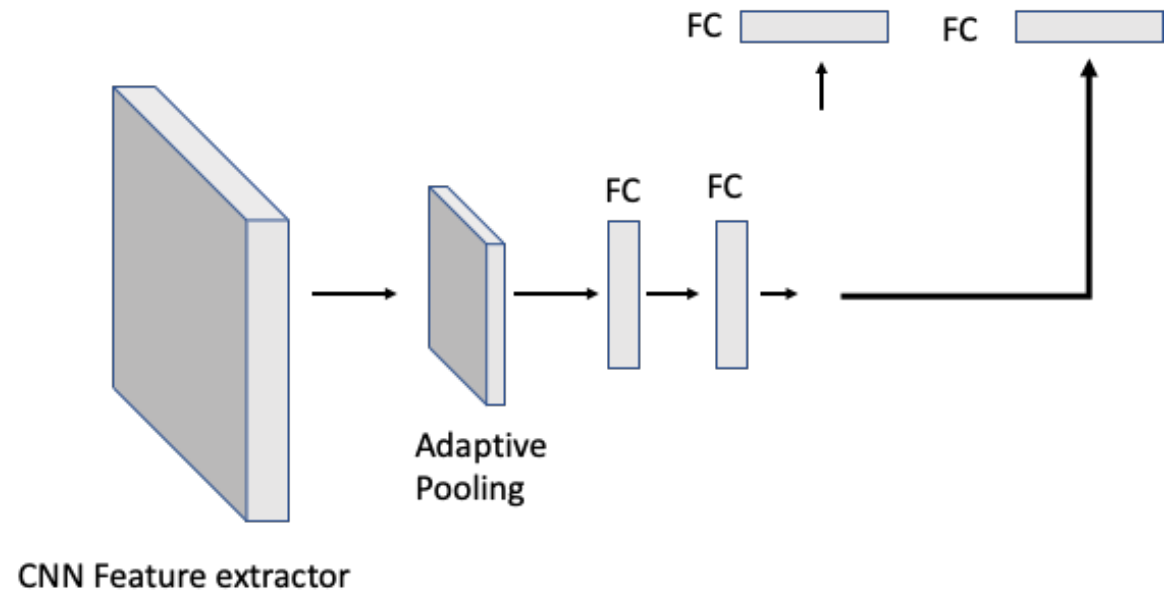


Fast R-CNN



Fast R-CNN

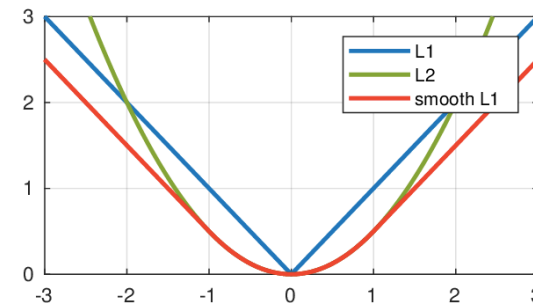
- Un oggetto $o_i = (y_i, c_i)$
 - $c_i = (c_{i,x}, c_{i,y}, c_{i,w}, c_{i,h})$
- Un box $b_i = (\hat{p}_i, \hat{c}_i)$



$$loss(o_i) = L_{CE}(\hat{p}_i, y_i) + [y_i > 0]L_{smooth}(c_i, \hat{c}_i)$$

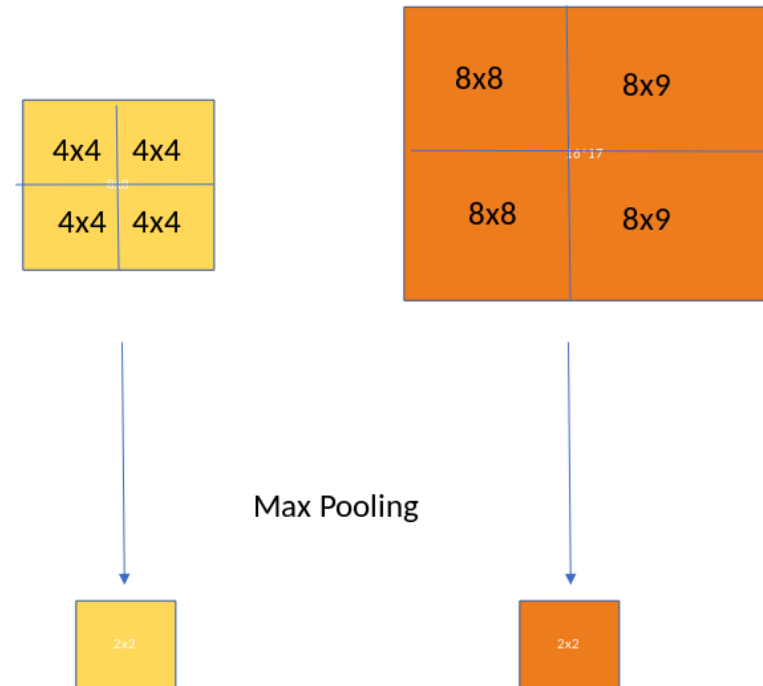
$$L_{smooth}(c_i, \hat{c}_i) = \sum_{j \in \{x, y, w, h\}} smooth_{L_1}(c_{ij} - \hat{c}_{ij})$$

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & o.w. \end{cases}$$

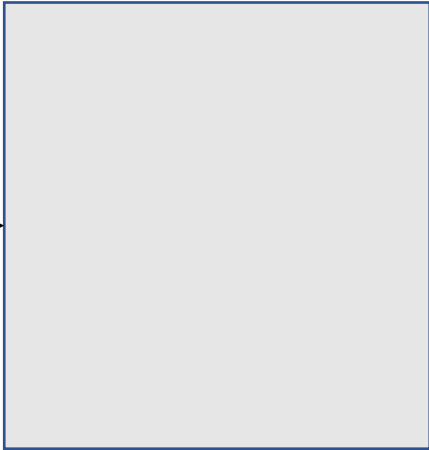
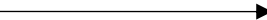


Adaptive Pooling

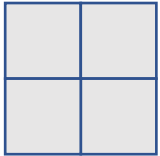
- Adatta la size sull'input per produrre lo stesso output
- **Indipendente dalla dimensione della RoI**
- Differenziabile
 - Ricordate la regola del gradient router



Adaptive Pooling



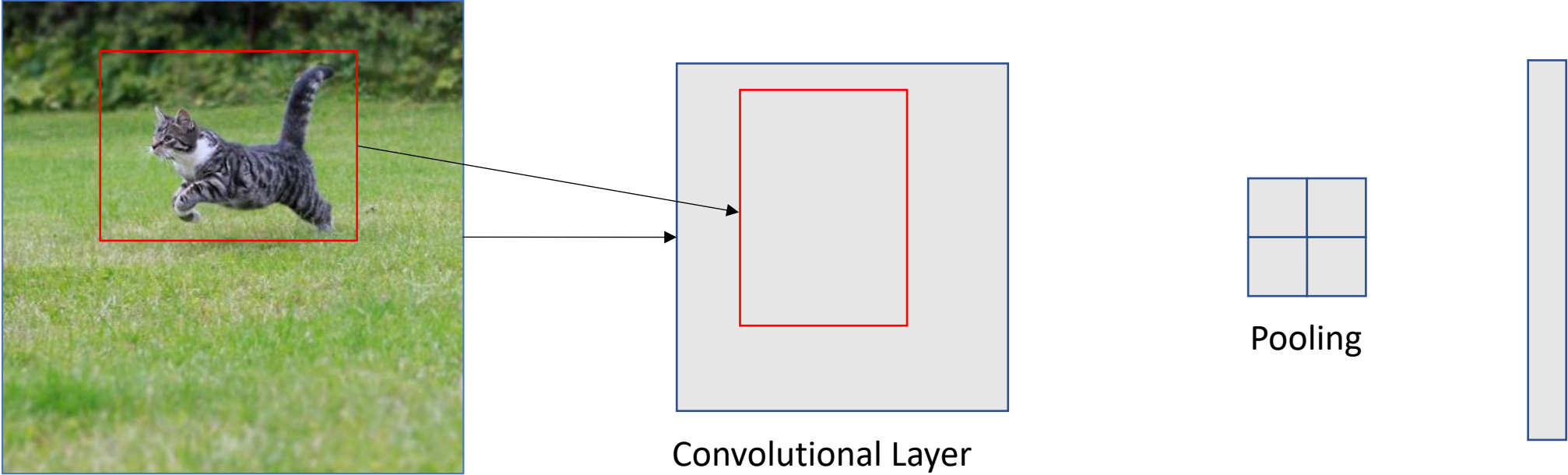
Convolutional Layer



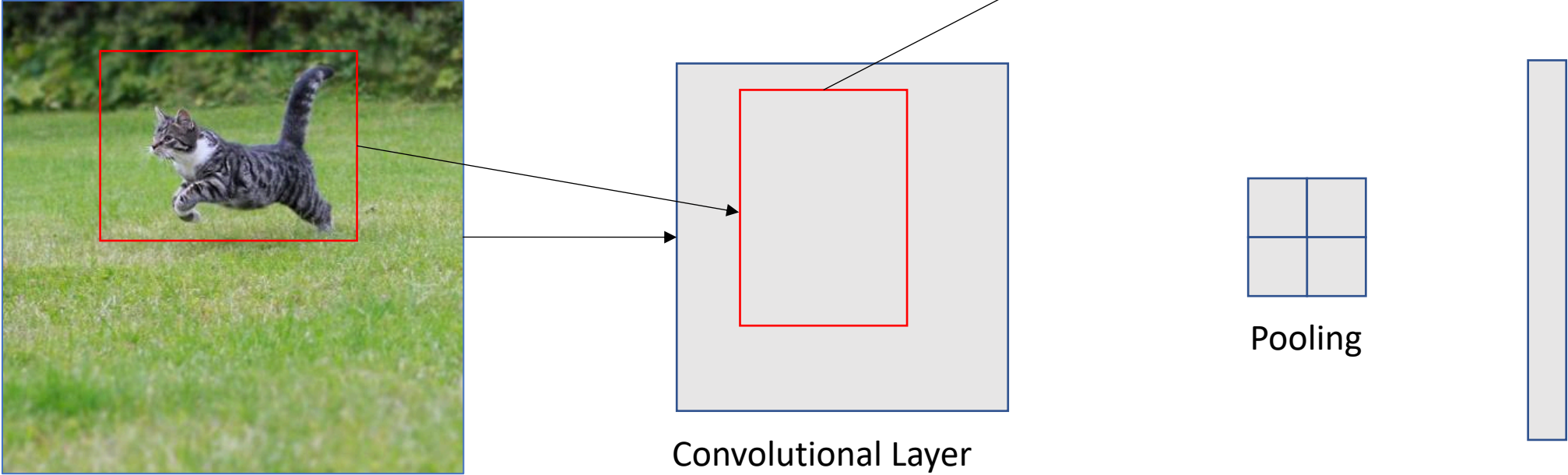
Pooling



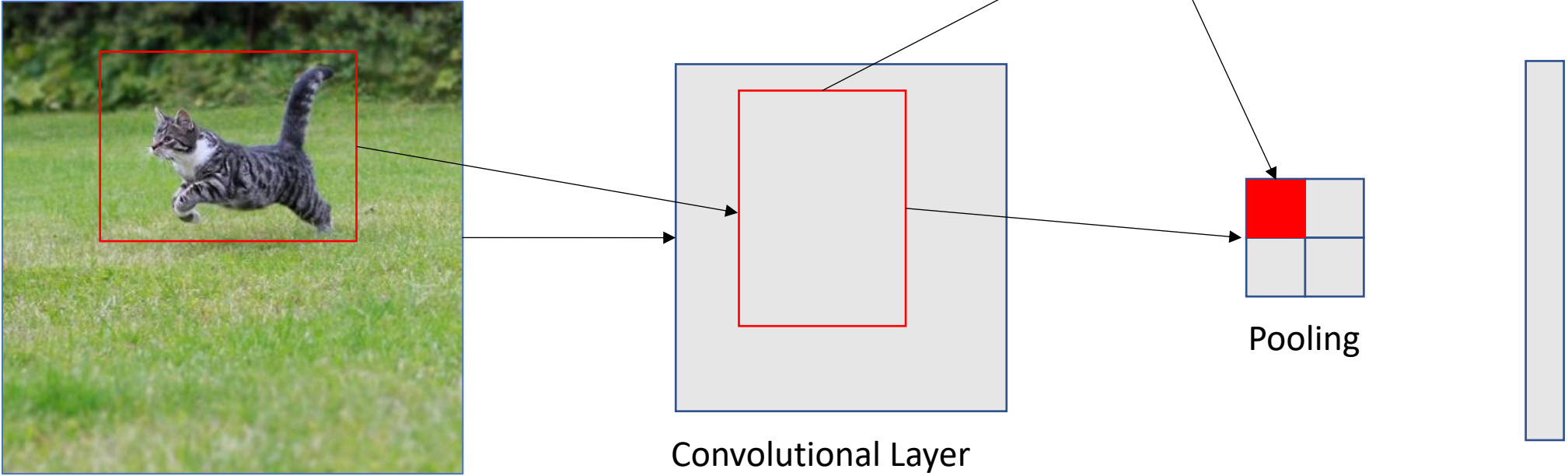
Adaptive Pooling



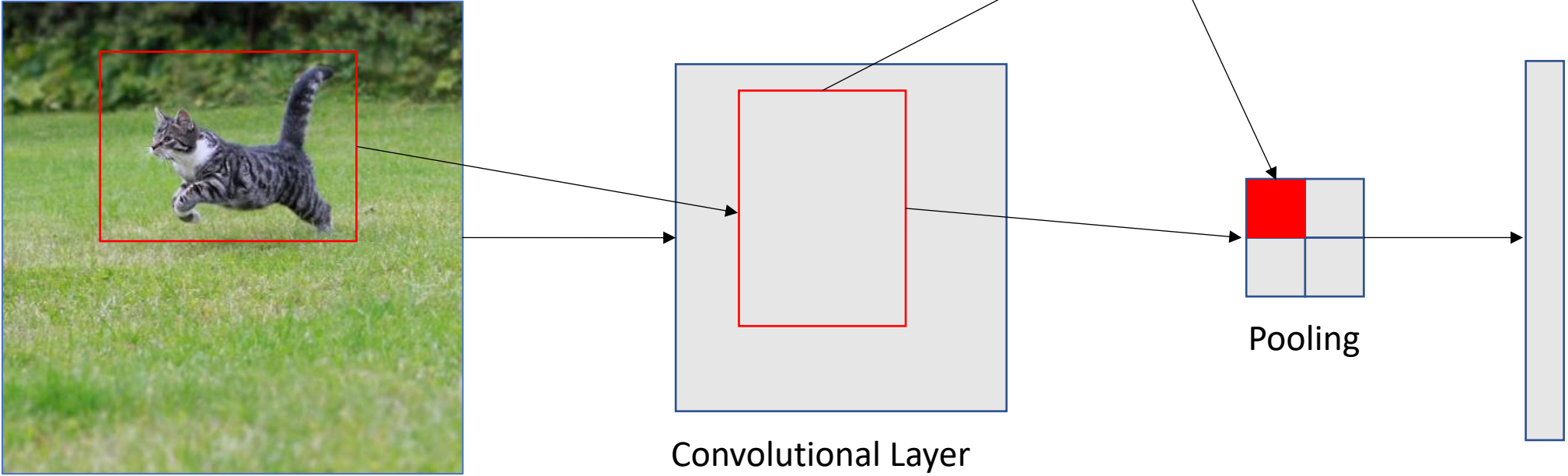
Adaptive Pooling



Adaptive Pooling



Adaptive Pooling

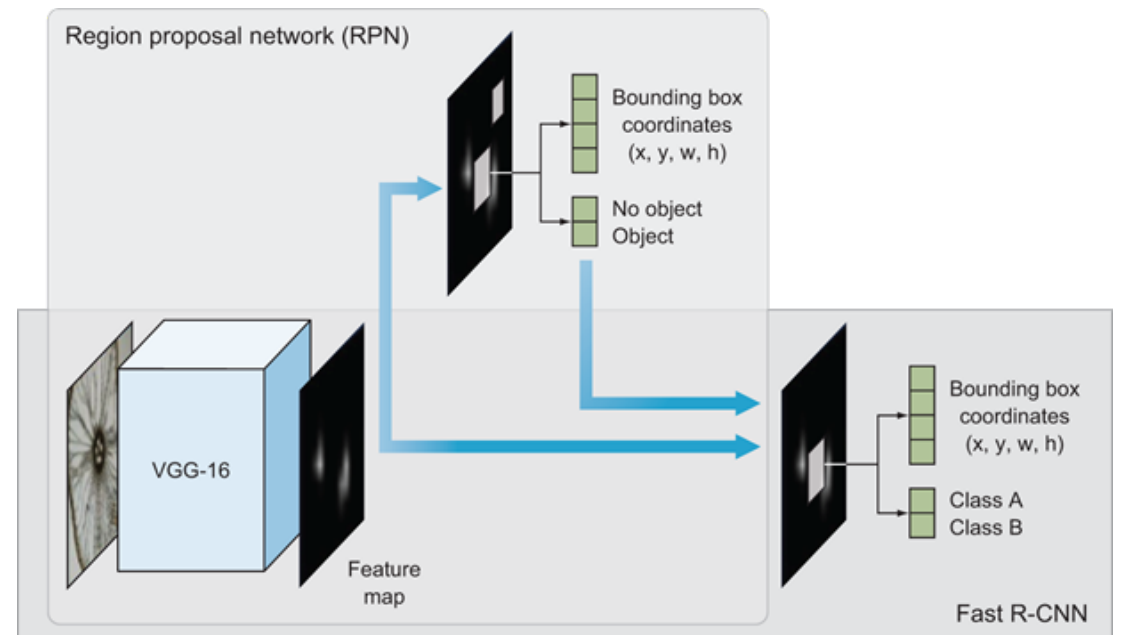


Fast R-CNN

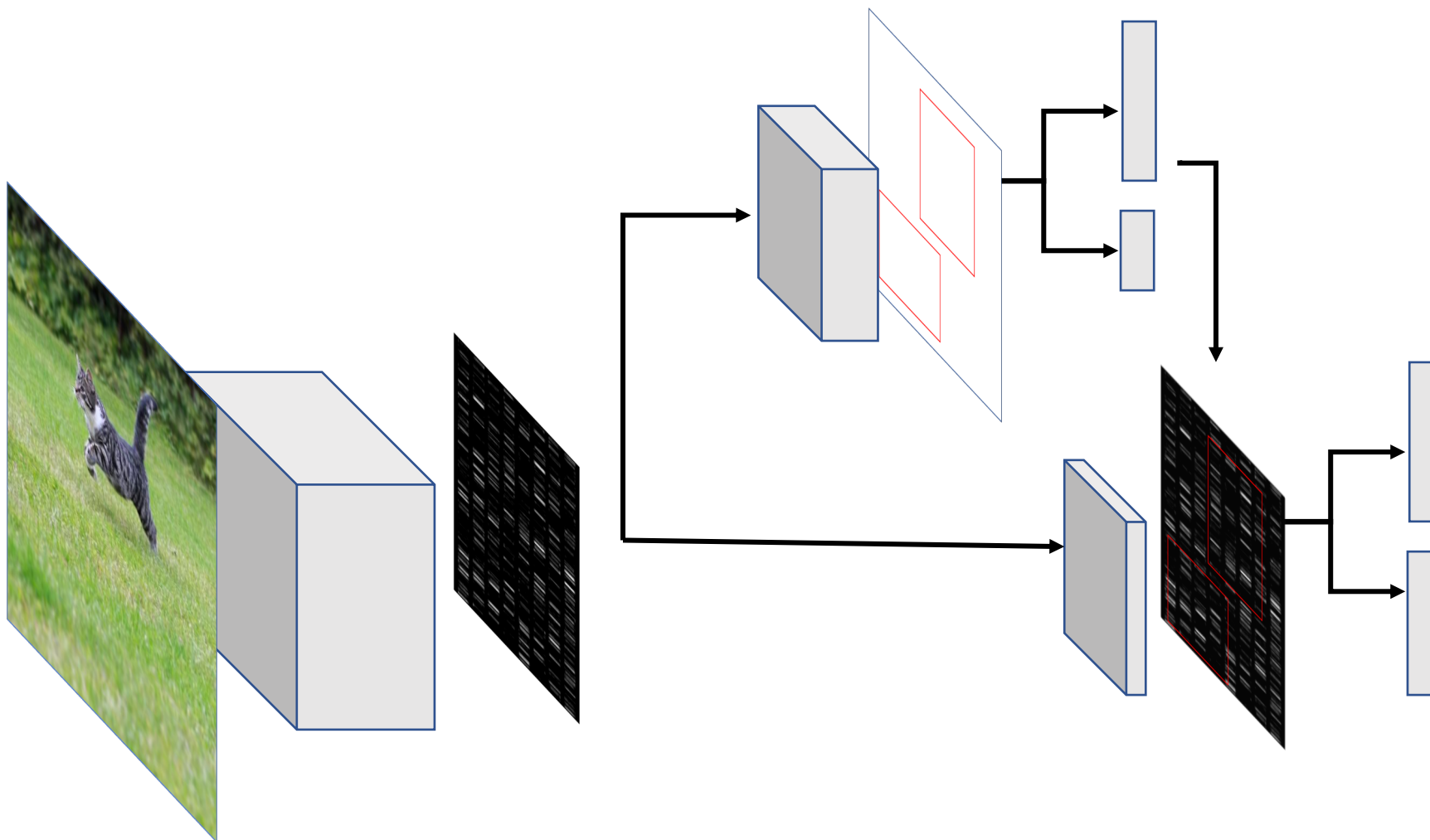
- Vantaggi
 - Architettura integrata
 - Training veloce: speedup 8.8x
 - Predizione veloce
 - 0.32s/image ~ 3FPS
 - mAP 66.9%
- Problemi
 - Le region proposals dipendono ancora dal Selective Search
 - Estremamente lento: 2s/image

Faster R-CNN

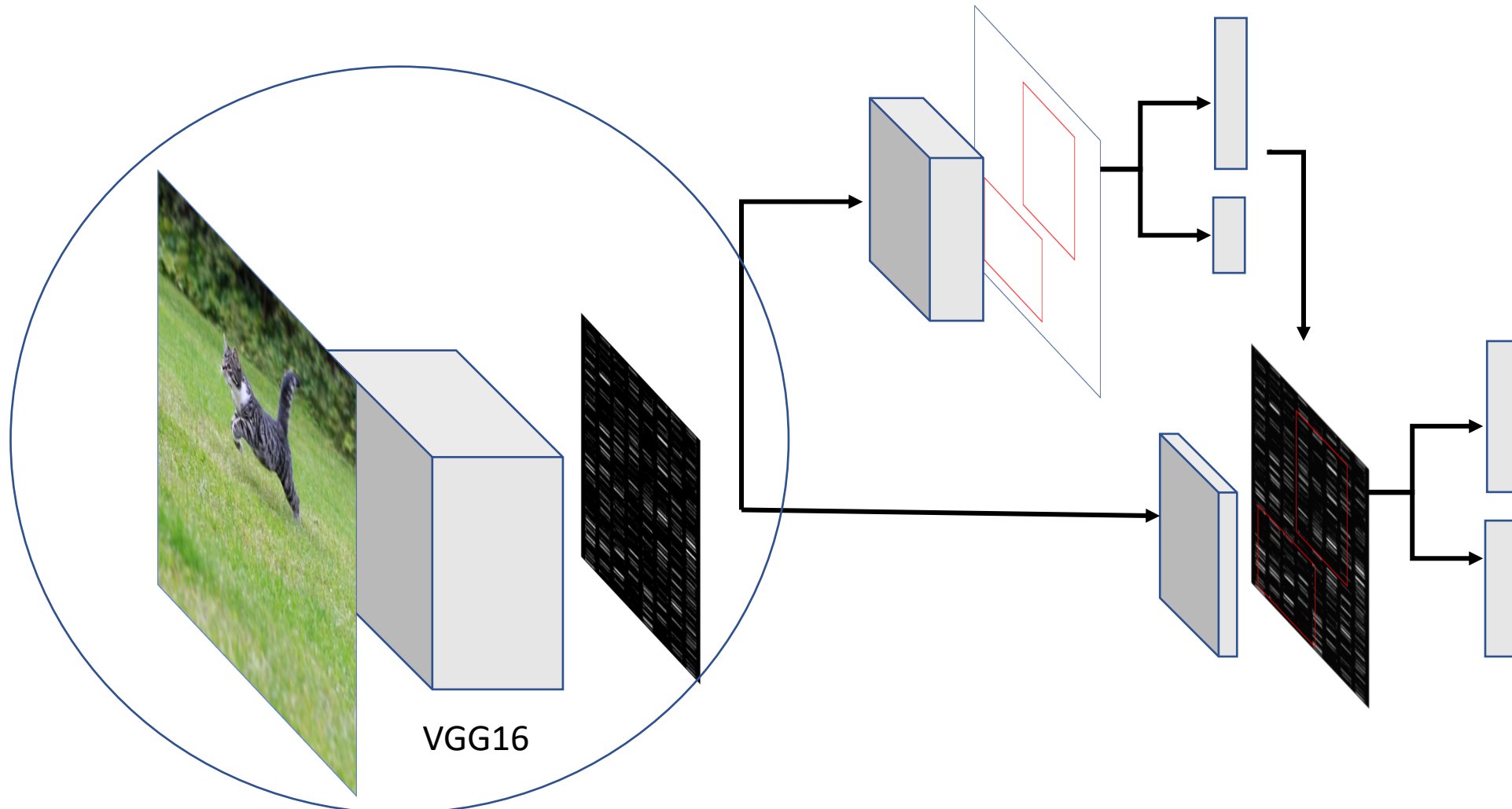
- Girshick et. al, 2016
- Idea: facciamo generare le proposal regions dalla rete stessa
- Due reti a cascata
 - La prima per generare le proposal regions
 - la seconda per la detection



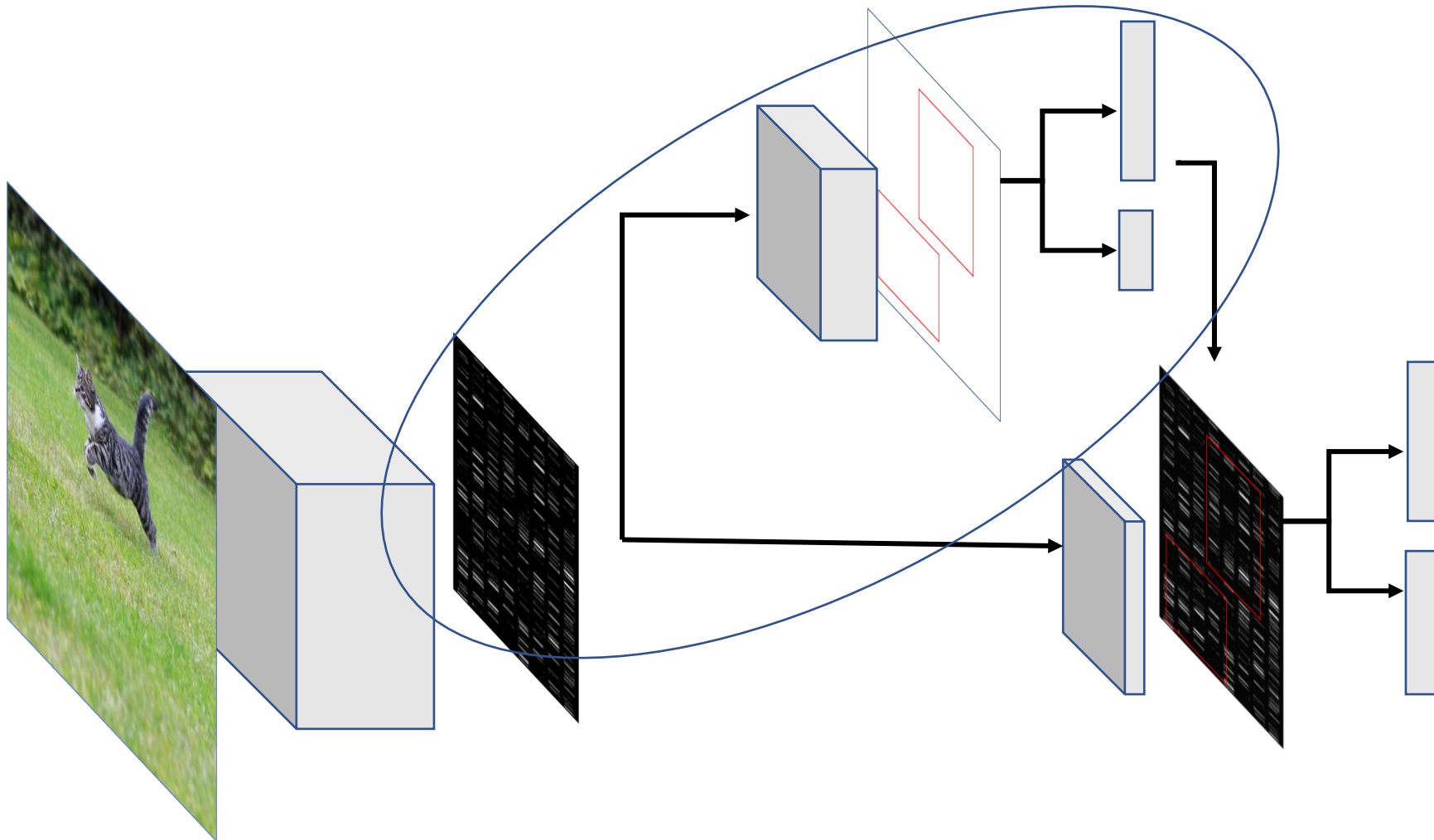
Faster R-CNN



Fase 1: feature extraction



Fase 2: RPN Network



Come vengono generate le Region Proposals?

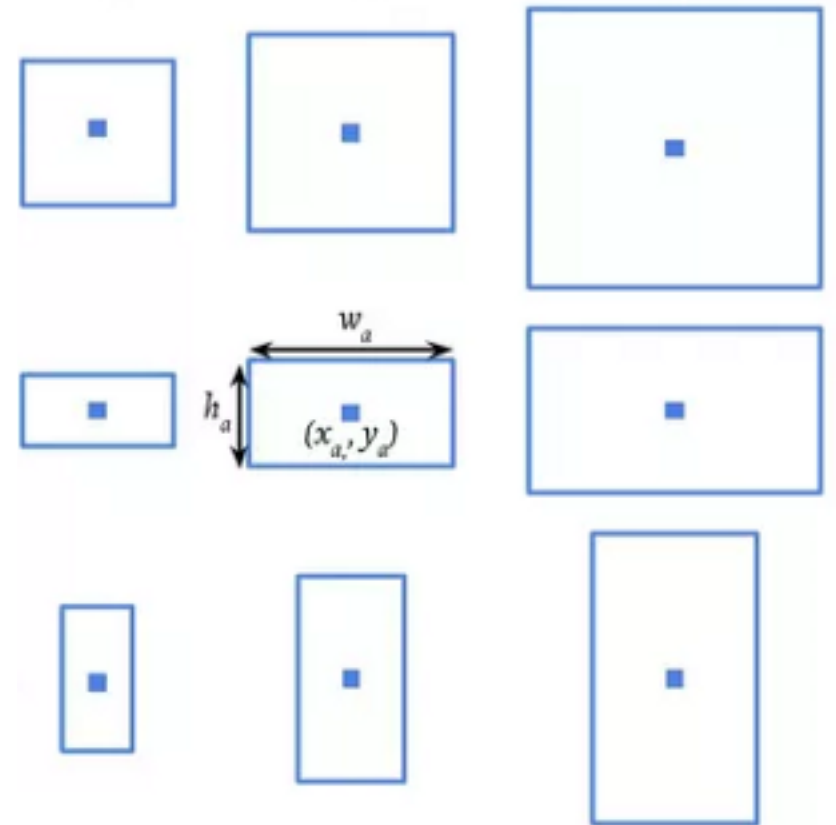
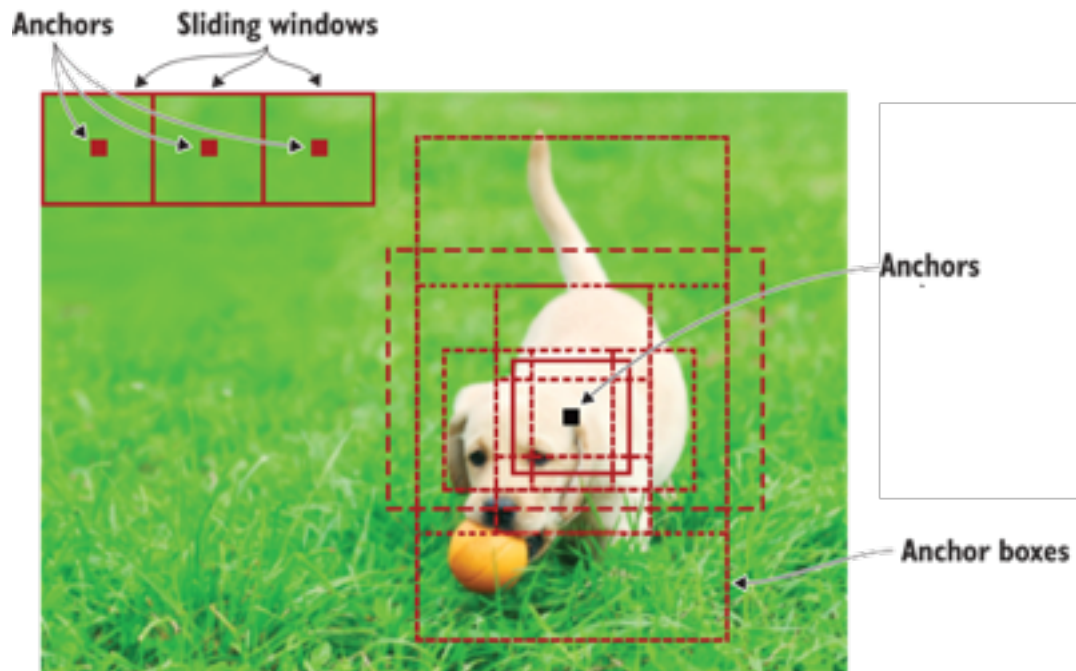
- Anchor boxes (priors)
 - Box di dimensione fissata e precalcolata
- Tre elementi
 - Posizione x, y
 - Scala s
 - Rapporto (aspect ratio) a
- Ogni punto genera una quadrupla (x, y, w, h)

$$w = s \cdot a^2$$

$$h = \frac{s}{a}$$

Come vengono generate le Region Proposals?

- $s = [0.5, 1, 2]$
- $a = [1:1, 1:2, 2:1]$

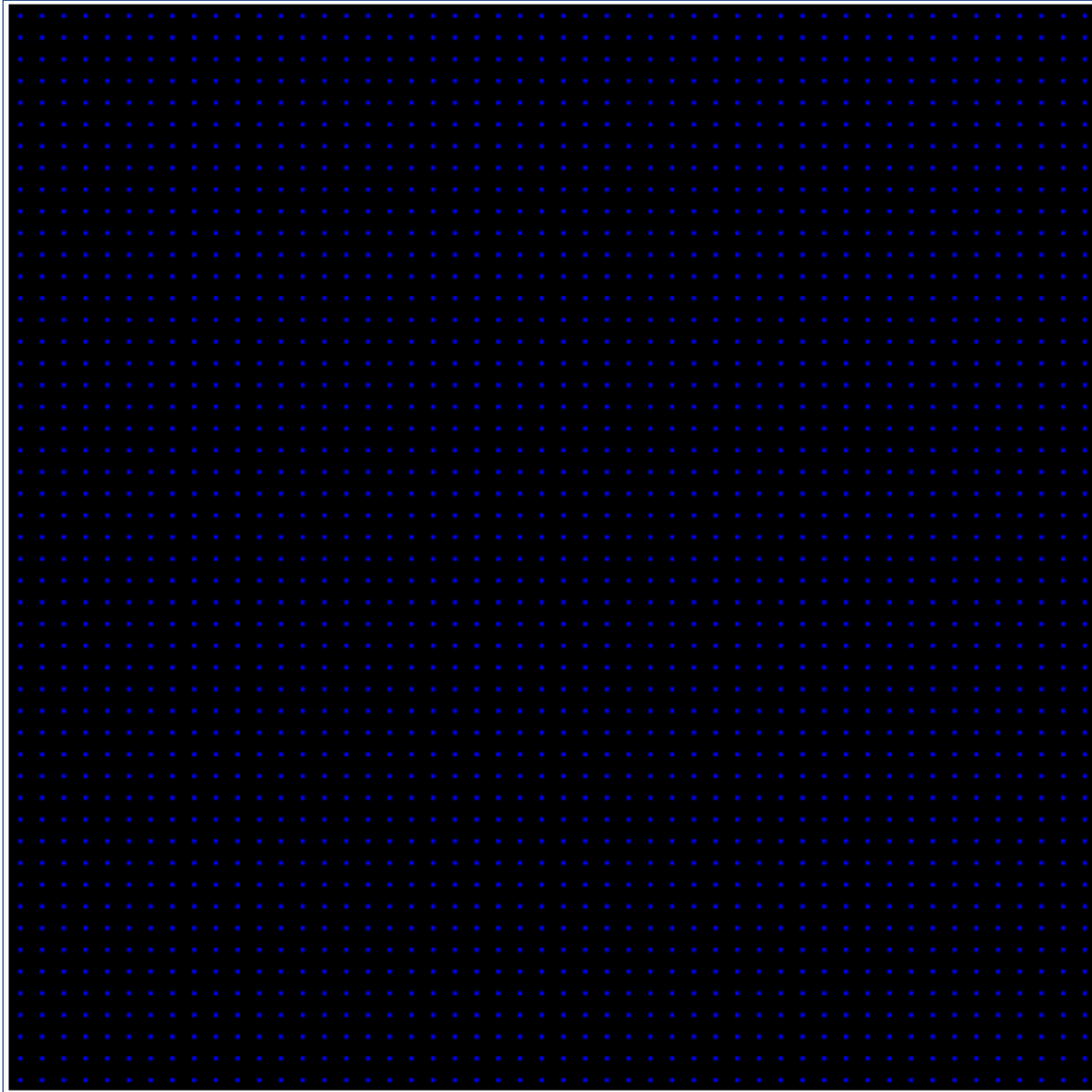


800x800



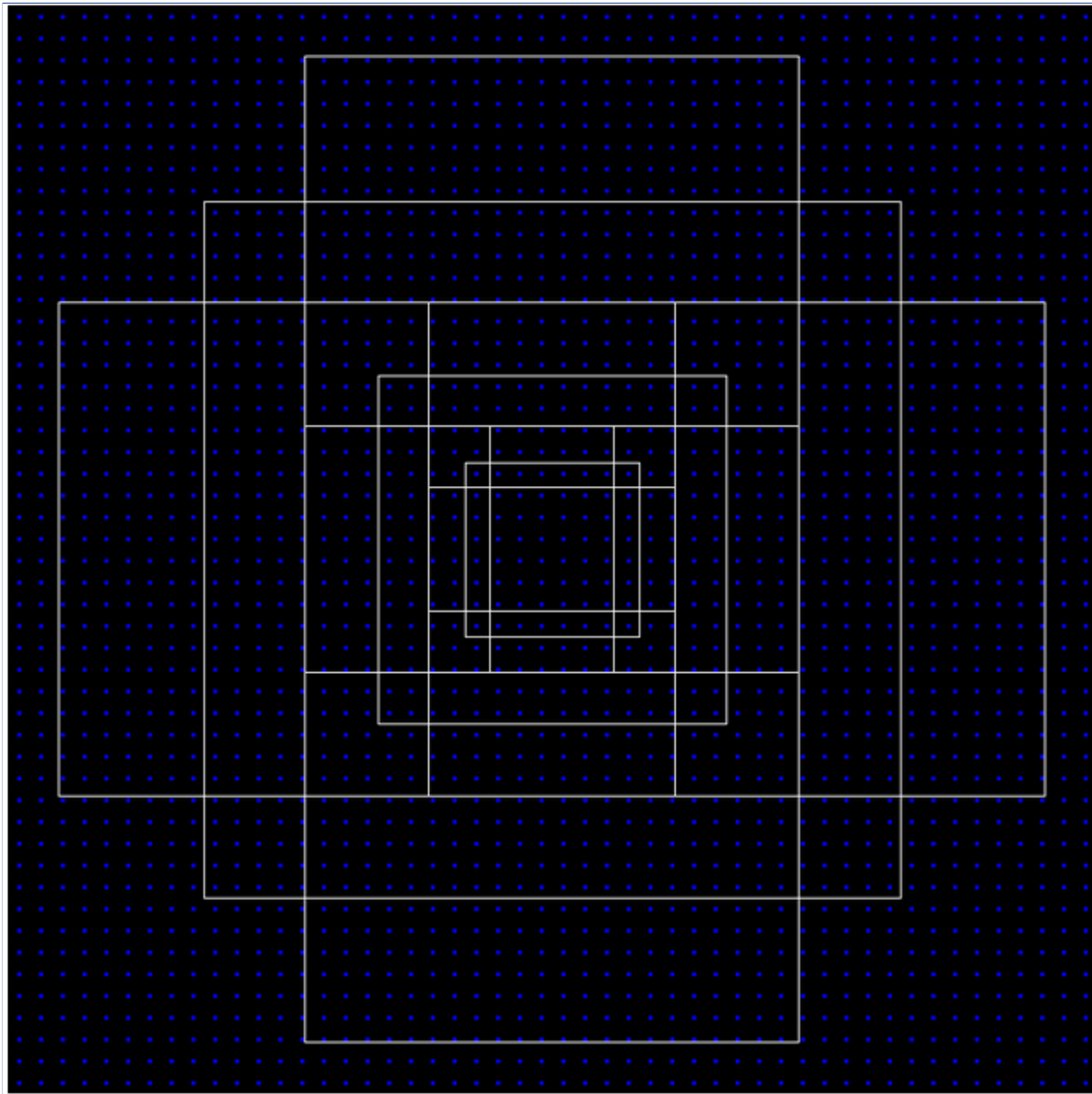
Subsampling ratio: 16

50x50



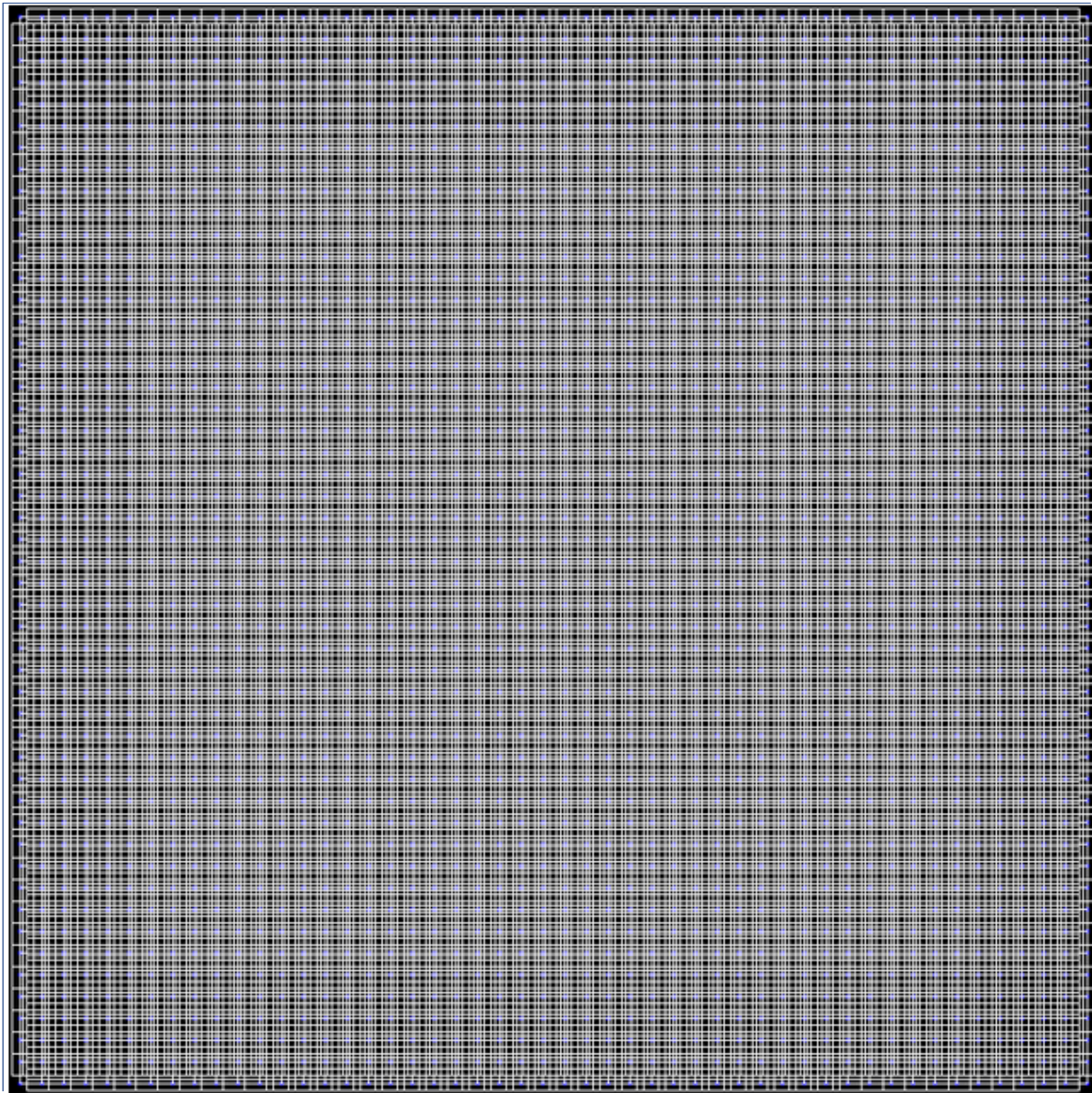
Subsampling ratio: 16





Subsampling ratio: 16



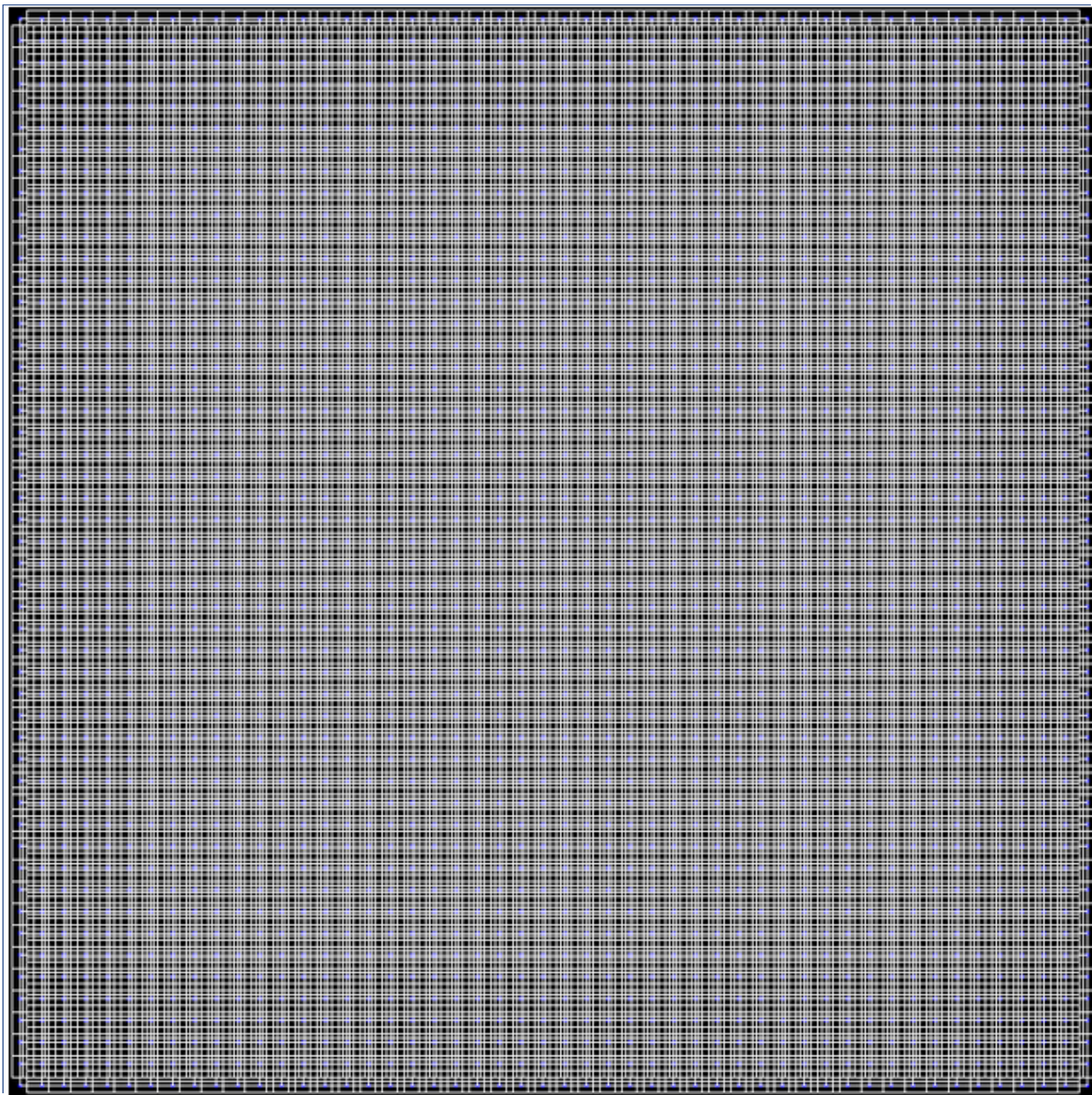


Subsampling ratio: 16

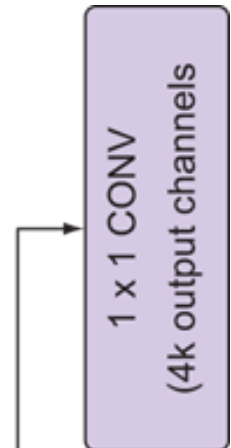
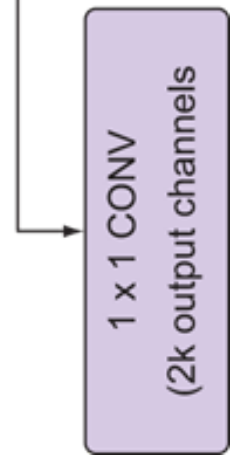
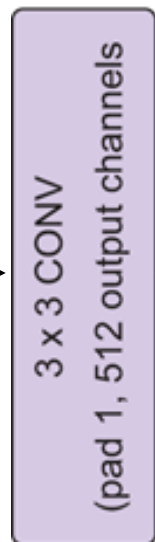


50x50

- Ogni cella della feature map individua k anchor boxes sull'immagine originale
- Totale anchor boxes: $k \cdot W \cdot H$
 - dove W, H è la size della feature map



Subsampling ratio: 16



Classification layer

Regression layer

RPN in Pytorch

```
class RPN(nn.Module):
    def __init__(self, in_channels = 512, mid_channels = 512, n_anchor = 9):
        super(RegioProposalNetwork, self).__init__()

        conv1 = nn.Conv2d(in_channels, mid_channels, 3, 1, 1)
        nn.init.normal(conv1.weight, mean=0, std=0.001)
        nn.init.zeros(conv1.bias)

        reg_layer = nn.Conv2d(mid_channels, n_anchor * 4, 1, 1, 0)
        nn.init.normal(reg_layer.weight, mean=0, std=0.001)
        nn.init.zeros(reg_layer.bias)

        cls_layer = nn.Conv2d(mid_channels, n_anchor * 2, 1, 1, 0)
        nn.init.normal(cls_layer.weight, mean=0, std=0.001)
        nn.init.zeros(cls_layer.bias)

    def forward(self, x):
        x = conv1(x)
        pred_anchor_locs = reg_layer(x)
        pred_cls_scores = cls_layer(x)

        pred_anchor_locs = pred_anchor_locs.permute(0, 2, 3, 1).contiguous().view(1, -1, 4)
        pred_cls_scores = pred_cls_scores.permute(0, 2, 3, 1).contiguous()

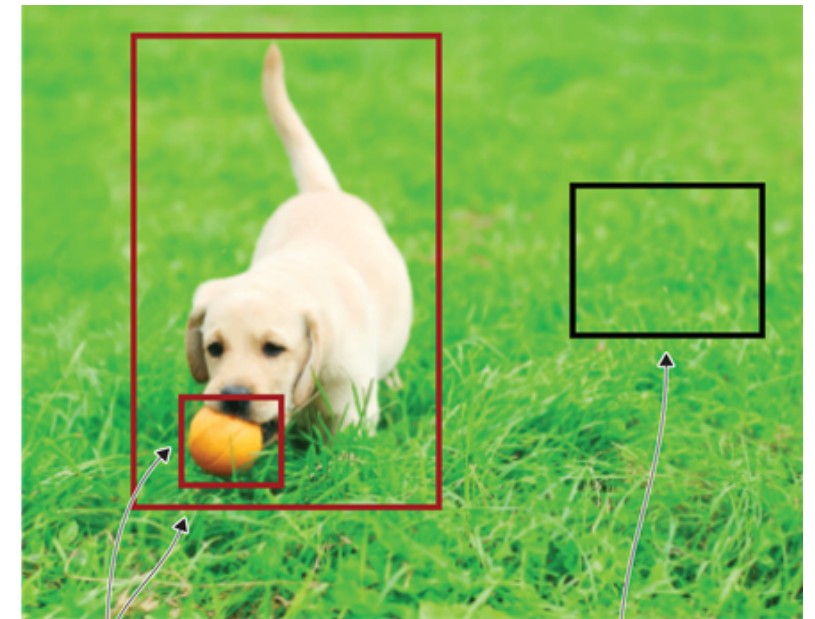
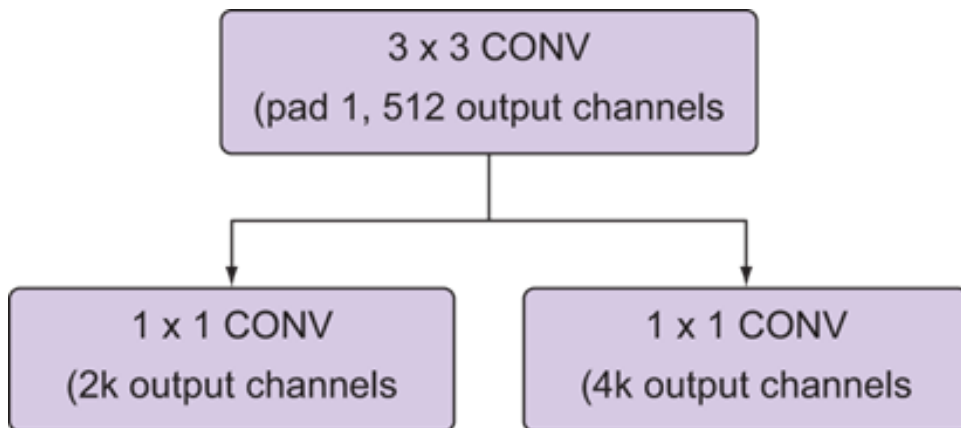
        pred_cls_scores = pred_cls_scores.view(pred_cls_scores.shape[0], -1, 2)

        return pred_anchor_locs, pred_cls_scores
```

Classification, regression

- Un anchor box è positivo se ha un $IoU > 0.7$ con un oggetto
 - e negativo se ha un $IoU < 0.3$

$$loss = \sum_i L_{cls}(\hat{p}_i, y_i) + \sum_i L_{reg}(\hat{c}_i, c_i)$$

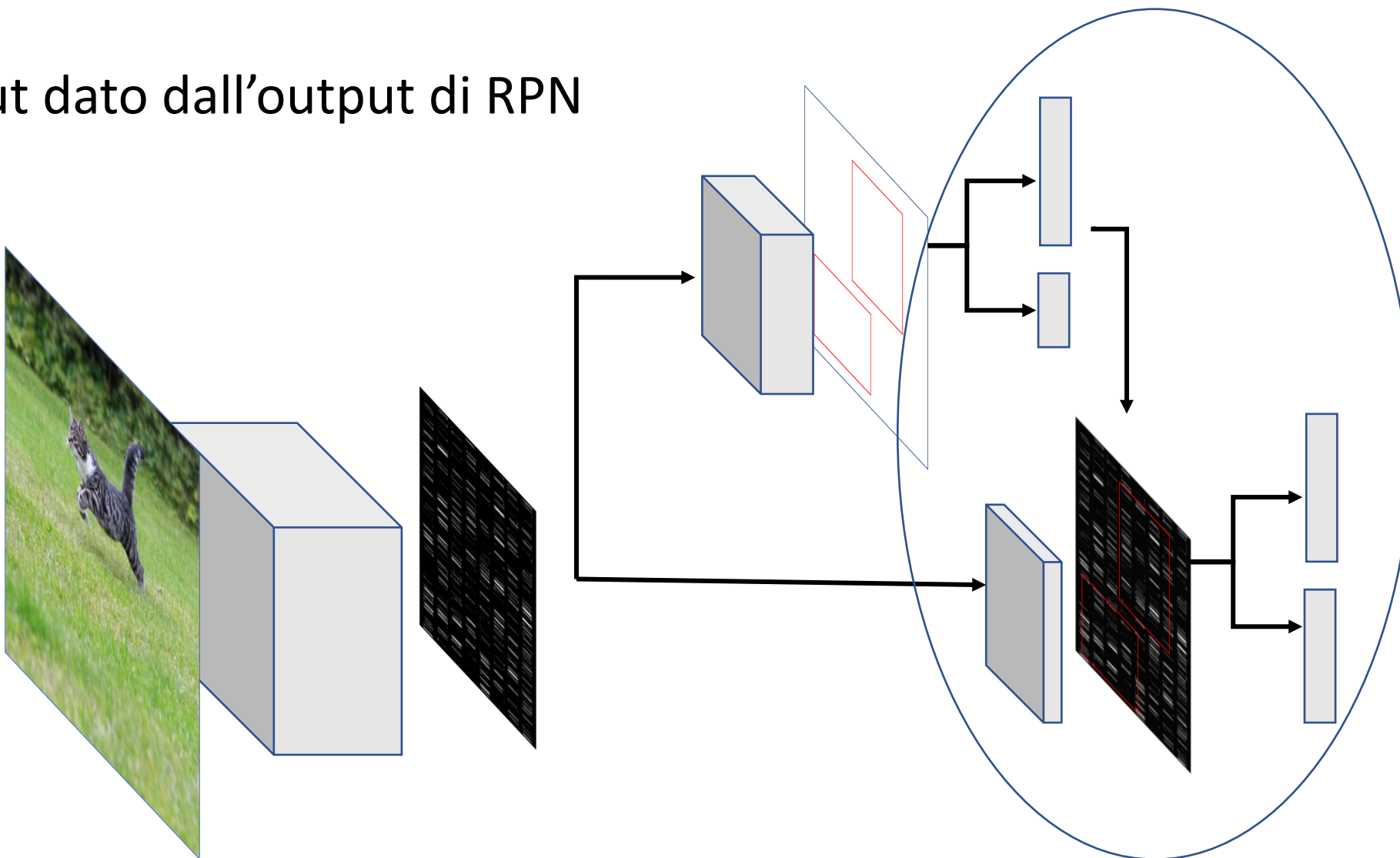


High objectness score
(foreground)

Low objectness score
(foreground)

Fase 3: Fast R-CNN

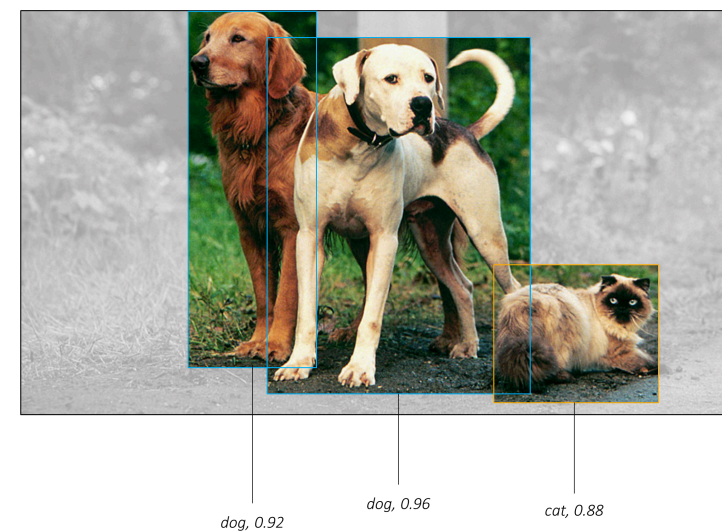
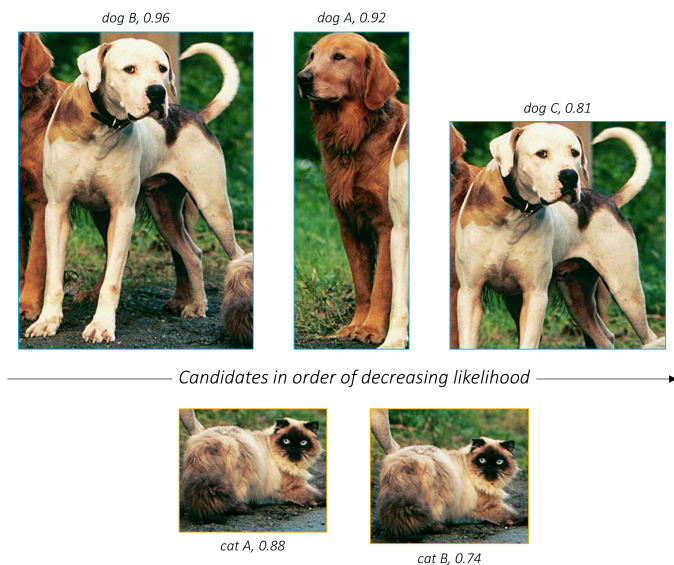
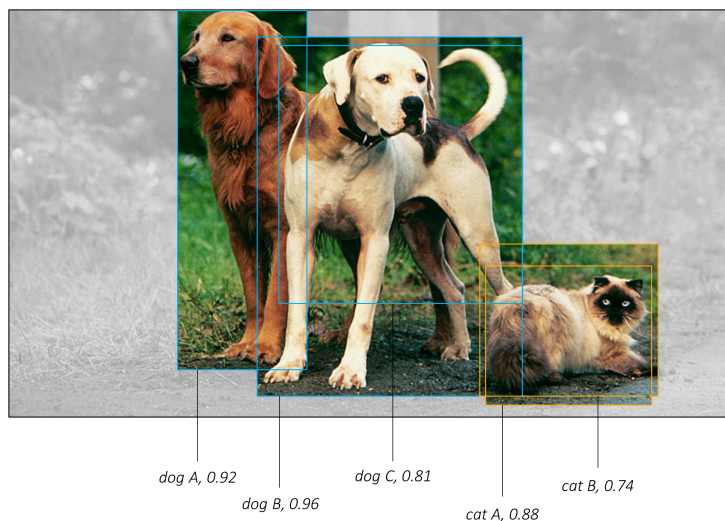
- Input dato dall'output di RPN



Postprocessing: Non-Maximum Suppression

- Si ordinano le regioni per score di classificazione
- Ogni box rimuove tutti i box della stessa classe che lo seguono e che hanno $IoU > 0.5$

There are usually multiple predictions for the same object



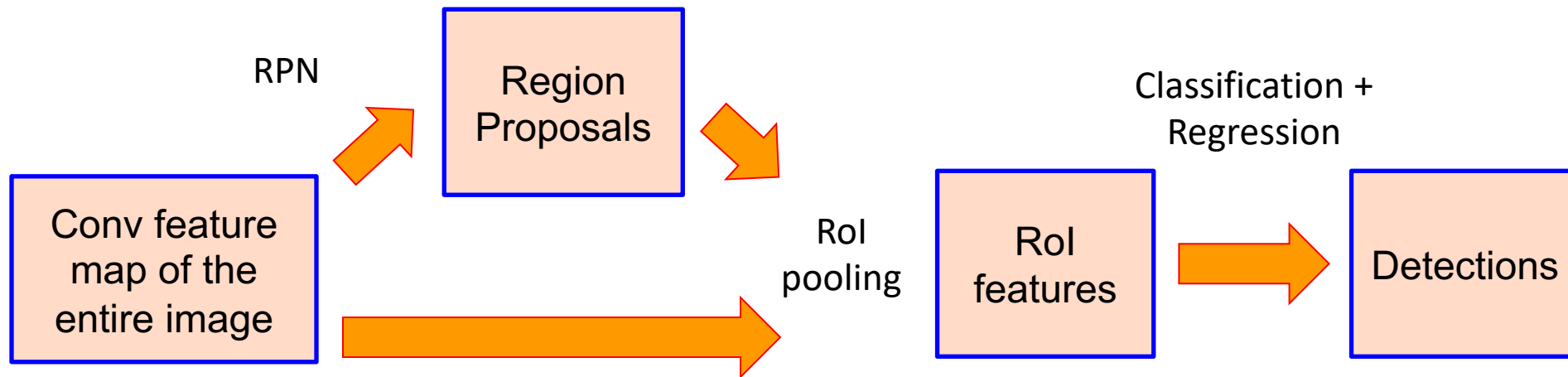
Riassunto

Metodo	Sec/image (FPS)	Speedup	mAP
R-CNN	~50s (0,02)	1x	66%
Fast R-CNN	~2s (0,5)	25x	66,9%
Faster R-CNN	~0,2s (5-7)	250x	69,9%

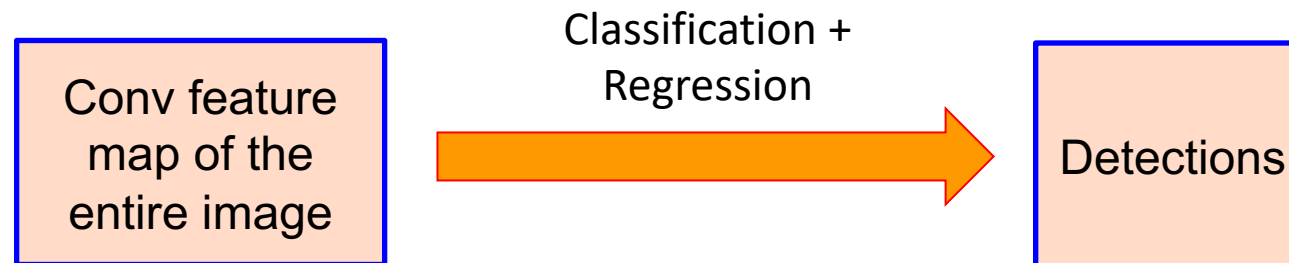
- Addestramento troppo lento
- Fasi multiple
- Può sopportare real-time object detection?
 - No
 - Troppo lenti

Evoluzione

- Multi-stage object detectors



- Single-shot object detectors

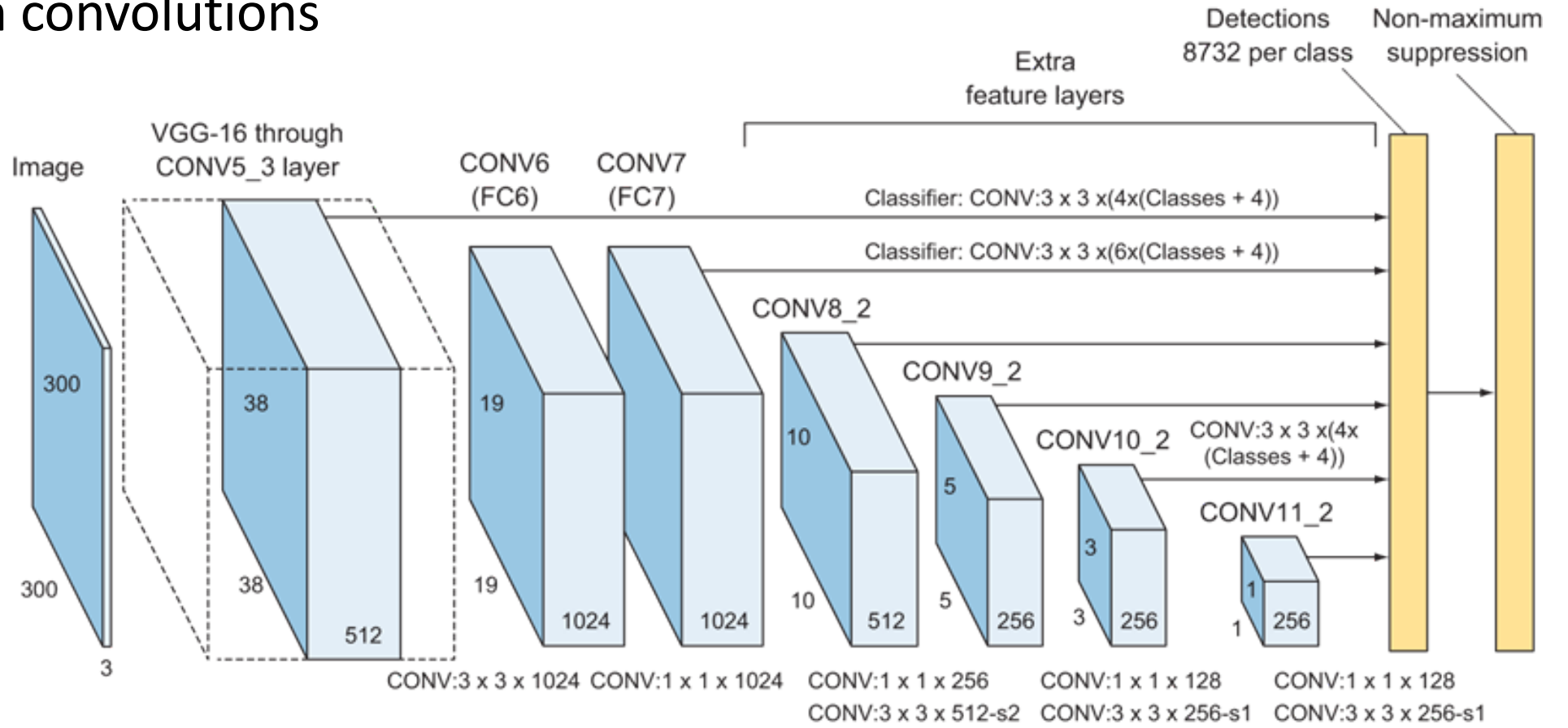


SSD: Single-Shot MultiBox Detector

- Szegedy et al., 2016
- 74% mAP, 59 FPS
- Principali caratteristiche
 - Eliminazione delle region proposals
 - Rete di base (preaddestrata) per estrarre le feature maps.
 - Multi-scale feature layers
 - Una serie di filtri convoluzionali in cascata
 - Riducono la dimensione e abilitano la detection su scale multiple
 - Non-maximum suppression

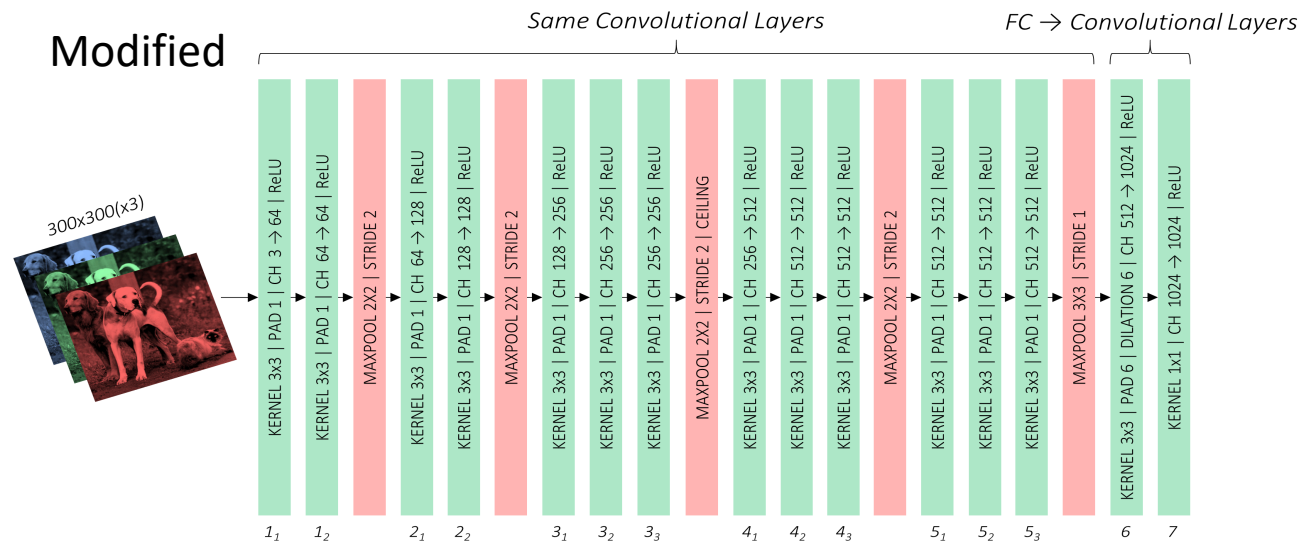
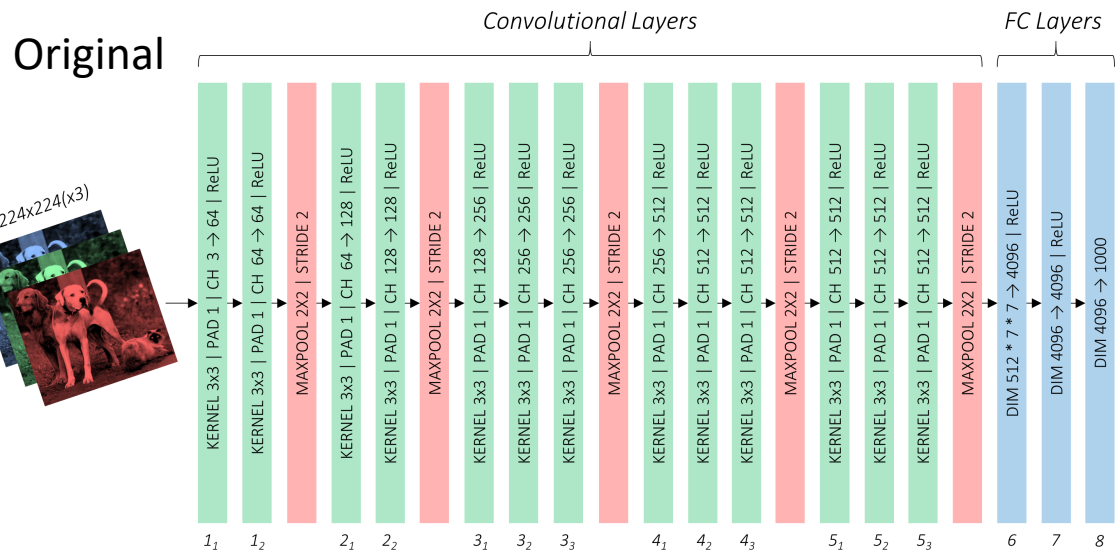
SSD

- Tre componenti
 - Base convolutions
 - Auxiliary convolutions
 - Prediction convolutions



SSD – Base convolutions

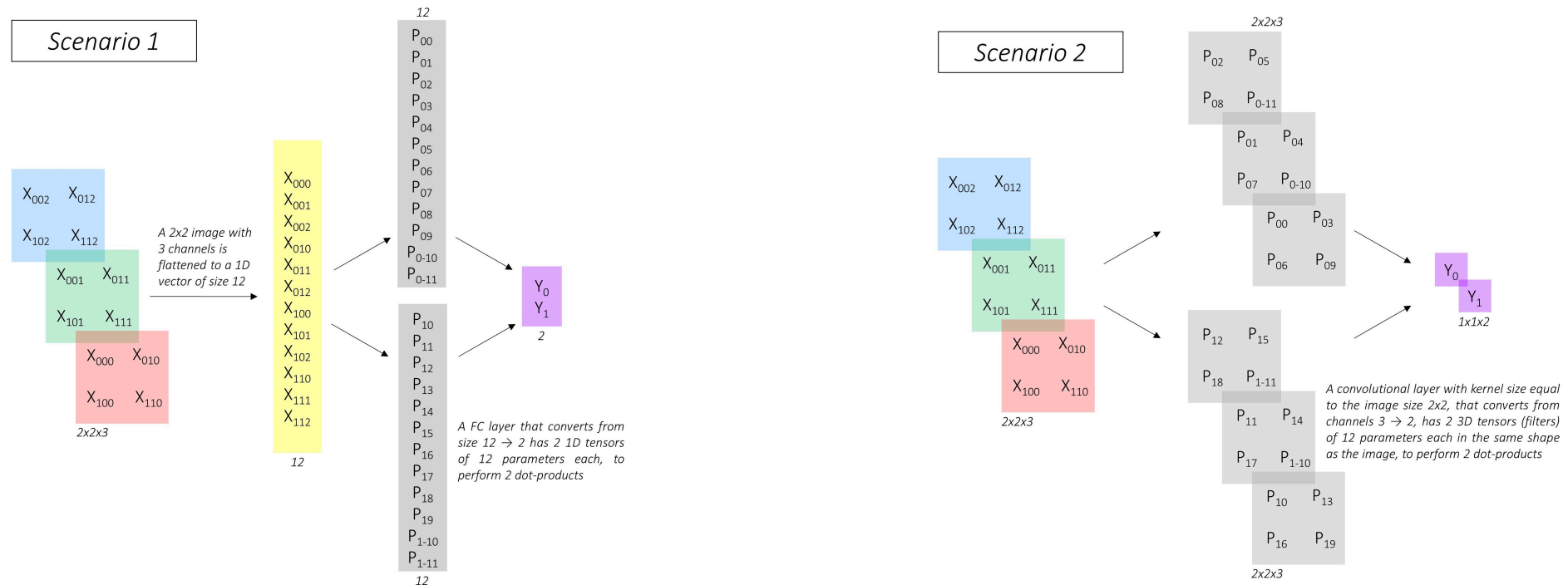
- Modified VGG-16
 - Input size: 300x300
 - Modifichiamo il quinto pooling layer
 - Rimuoviamo FC8, ristrutturiamo FC6, FC7 (convolutionize, reduce channels, subsample->decimazione/dilation)



SSD – Base convolutions

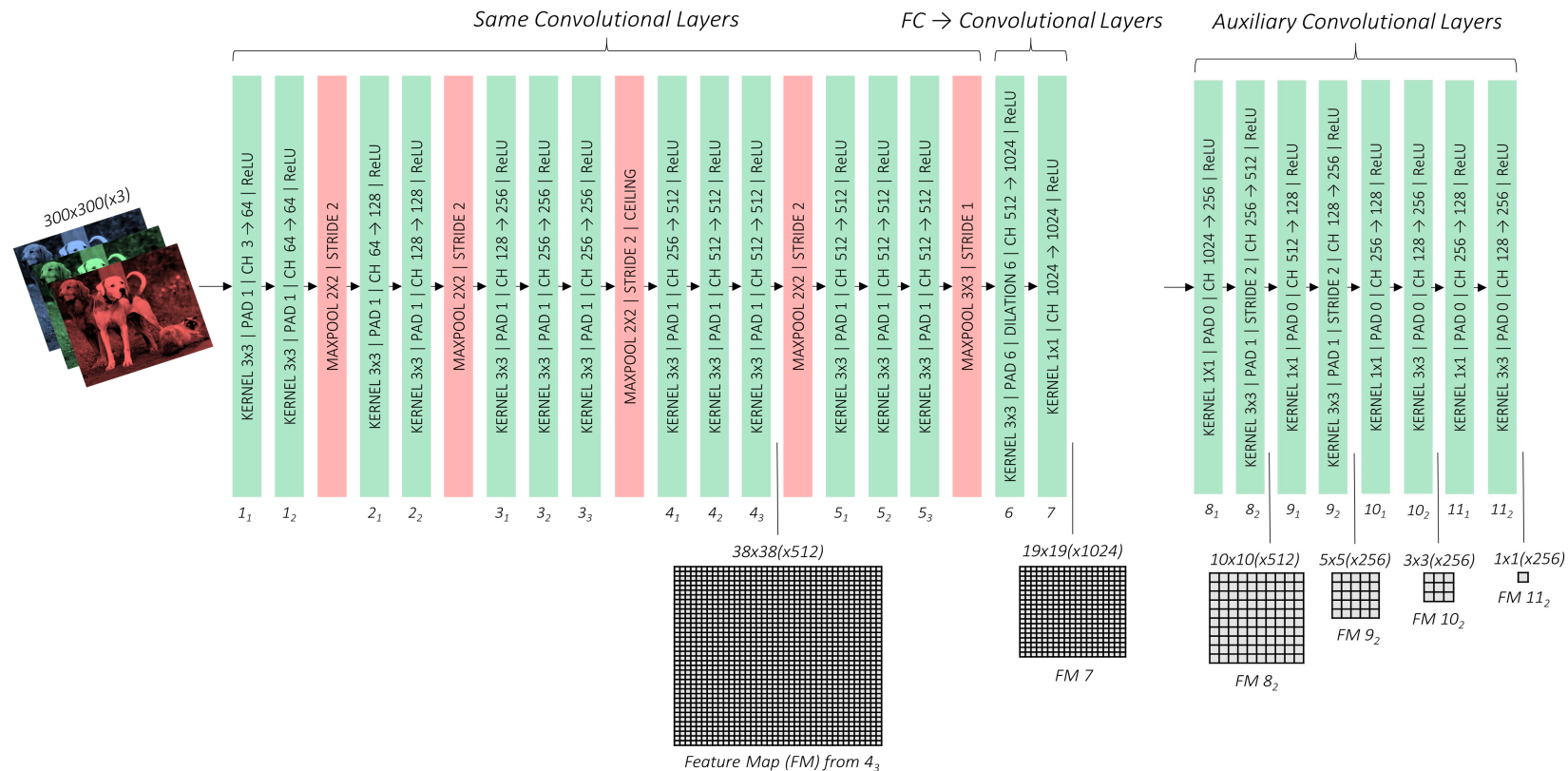
- Reshaping

- Su un'immagine di dimensione H, W su I canali, un FC con output N equivale a un CONV con kernel $H \times W$ e N canali di output



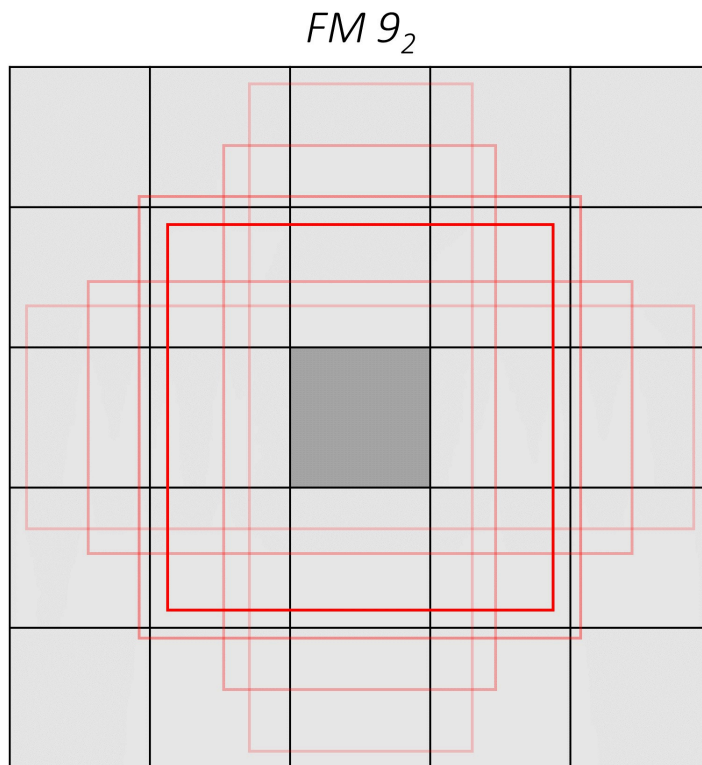
SSD – Auxiliary convolutions

- Quattro nuovi blocchi
 - Ogni blocco produce una Feature Map di output



SSD – Prediction outputs

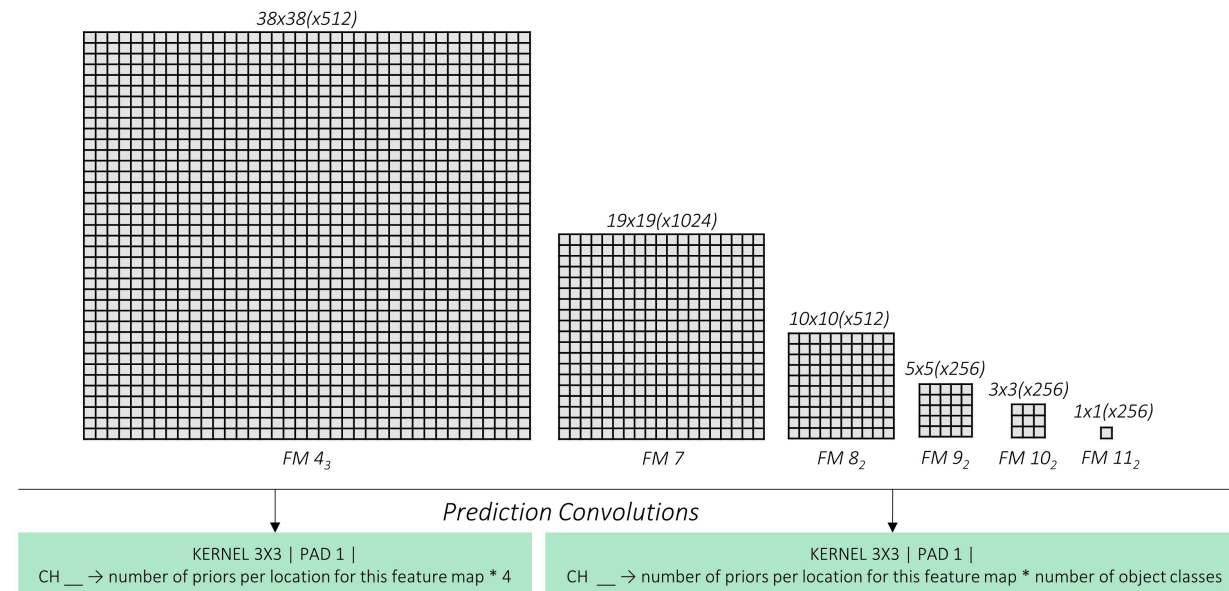
- Cosa rappresentano gli FM di output?
 - Celle su cui definire i priors



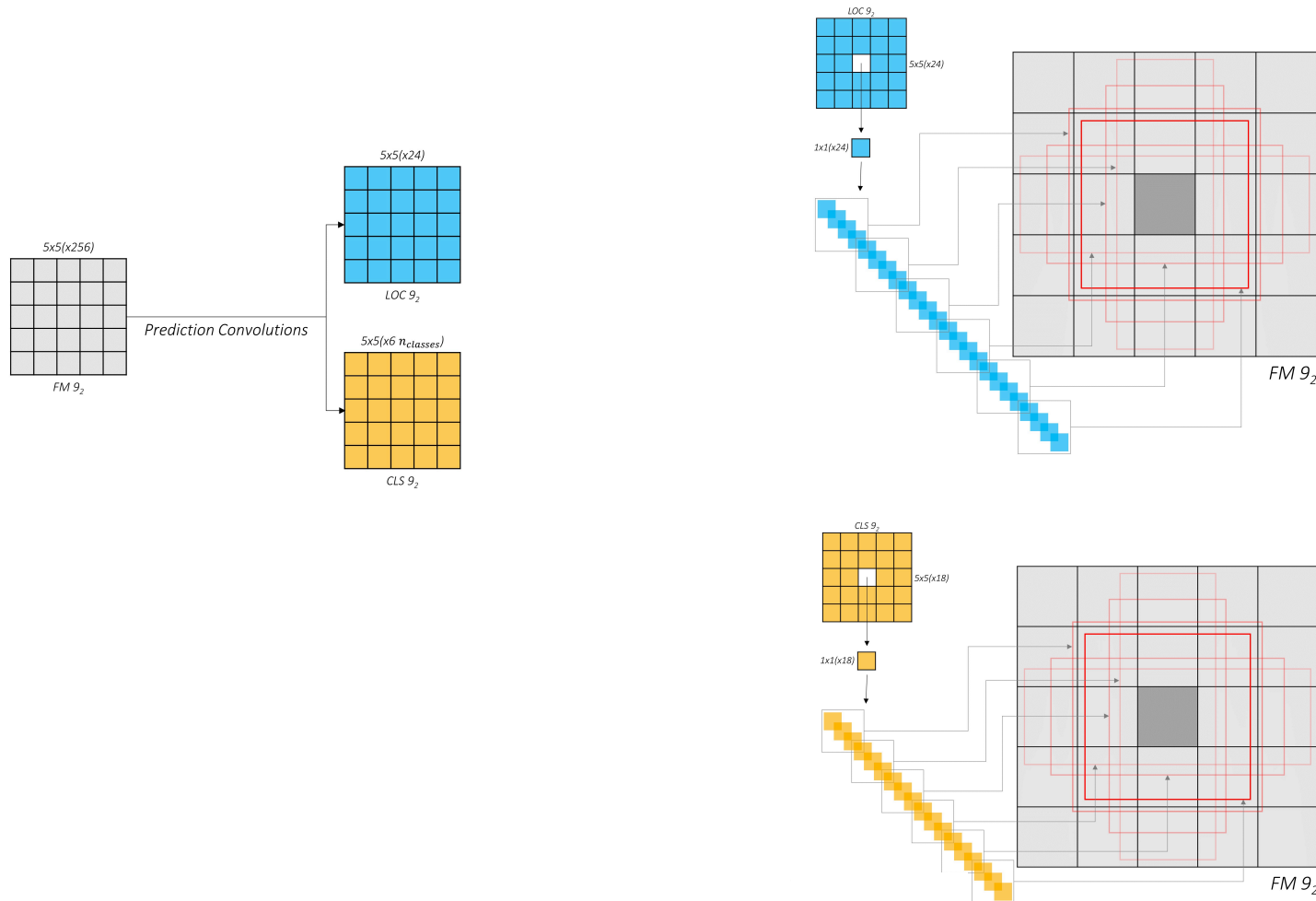
Feature Map From	Feature Map Dimensions	Prior Scale	Aspect Ratios	Number of Priors per Position	Total Number of Priors on this Feature Map
conv4_3	38, 38	0.1	1:1, 2:1, 1:2 + an extra prior	4	5776
conv7	19, 19	0.2	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	2166
conv8_2	10, 10	0.375	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	600
conv9_2	5, 5	0.55	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	150
conv10_2	3, 3	0.725	1:1, 2:1, 1:2 + an extra prior	4	36
conv11_2	1, 1	0.9	1:1, 2:1, 1:2 + an extra prior	4	4
Grand Total	-	-	-	-	8732 priors

SSD - Prediction convolutions

- Ogni prior su ogni posizione di ogni feature map produce una predizione
- Due layer convoluzionali
 - Localization layer $3 \times 3 \times 4$
 - Regressione su $(g_{c_x}, g_{c_y}, g_w, g_h)$
 - Classification layer $3 \times 3 \times n_{classes}$



SSD – Prediction Convolutions



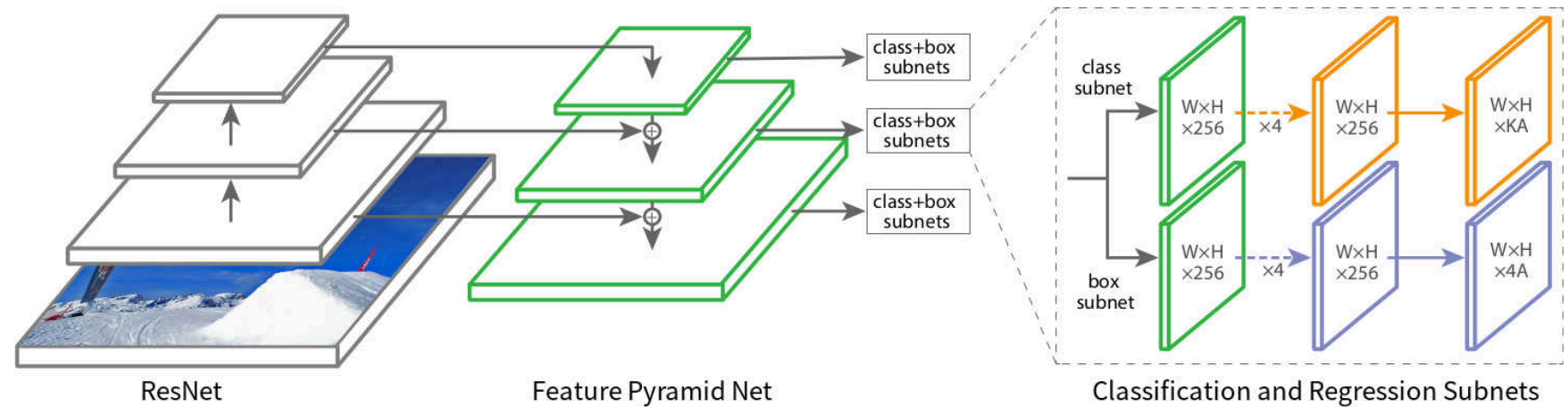
SSD - Riassunto

- Training tips
 - Hard negative mining: rapporto positivi/negativi 1 a 3
 - Data augmentation
 - Transfer learning: Atrous convolution: decimazione/dilation dei kernel su FC6
- Due varianti
 - SSD300
 - SSD512

Metodo	FPS	mAP	Priors
Faster R-CNN	5-7	69,9%	~6000
SSD300	46	74.3%	8732
SSD512	19	76,8%	24564

RetinaNet

- Feature Pyramids
 - Estende il concetto di multi-scale anchor boxes
- Focal loss



RetinaNet: architettura

- Tre componenti
 - ResNet come architettura base
 - Feature Pyramid Network per combinare i risultati
 - Una sottorete per BB Regression e Classification

Feature Pyramid Network

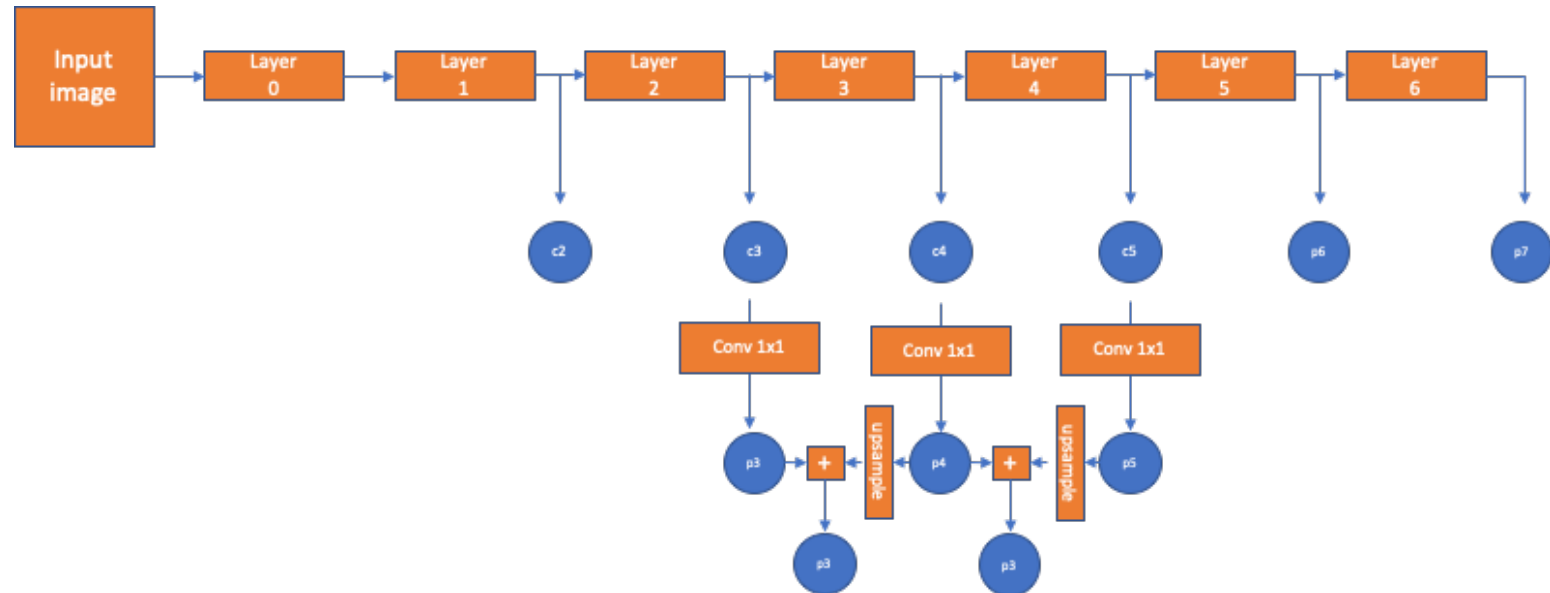
- In una rete convoluzionale, l'ultima feature map è la più significativa
- Idea: combiniamo l'informazione di tale feature map con quella dei layer inferiori
 - Upsample e combinazione
 - Predizione su ogni layer: le feature maps possono essere usate indipendentemente e rendono il modello scale-invariant



Architettura finale

- Anchor points
 - SSD ~10K
 - RetinaNet ~ 100K

```
def forward(self, x):  
    # Bottom-up  
    c1 = F.relu(self.bn1(self.conv1(x)))  
    c1 = F.max_pool2d(c1, kernel_size=3, stride=2, padding=1)  
    c2 = self.layer1(c1)  
    c3 = self.layer2(c2)  
    c4 = self.layer3(c3)  
    c5 = self.layer4(c4)  
    p6 = self.conv6(c5)  
    p7 = self.conv7(F.relu(p6))  
    # Top-down  
    p5 = self.latlayer1(c5)  
    p4 = self._upsample_add(p5, self.latlayer2(c4))  
    p4 = self.toplayer1(p4)  
    p3 = self._upsample_add(p4, self.latlayer3(c3))  
    p3 = self.toplayer2(p3)  
    return p3, p4, p5, p6, p7
```

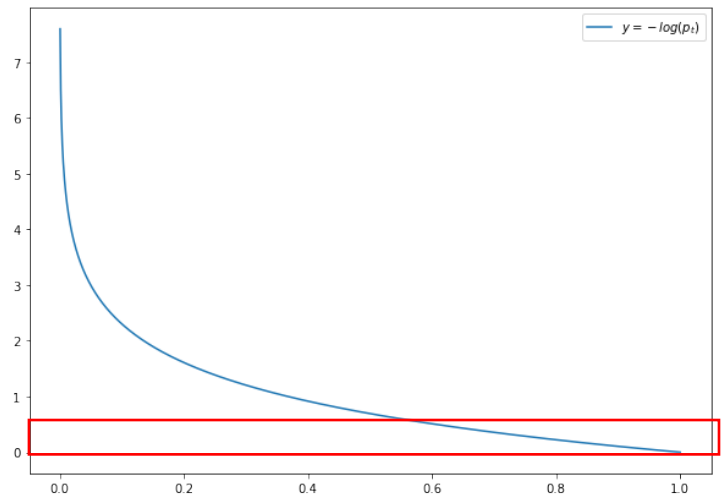


Focal Loss

- Con un numero così alto di anchor point, il numero di box negativi sarà incredibilmente alto
- Perché è un problema?
 - La classification loss sarà dominata dalle componenti negative

$$\text{CE}(p_t, y_t) = y_t \log p_t + (1 - y_t) \log 1 - p_t$$

- y_t rappresenta la ground truth sul box t
- Anche con $\text{CE}(p_t, y_t) \gg 0.5$ il contributo è non nullo

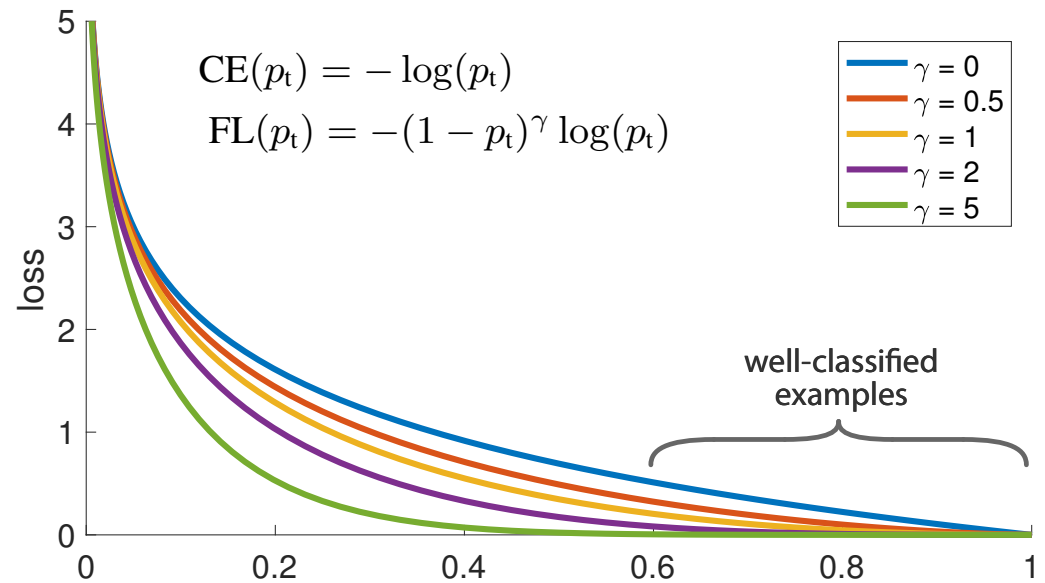


Focal loss

- Focal loss

$$\text{FL}(p_t) = -w_t(1 - p_t)^\gamma \log(1 - p_t)$$

- Ribilancia il contributo dei true negative
 - Il gradiente è dominata dall'incertezza (sui positivi)



YOLO – You Only Look Once

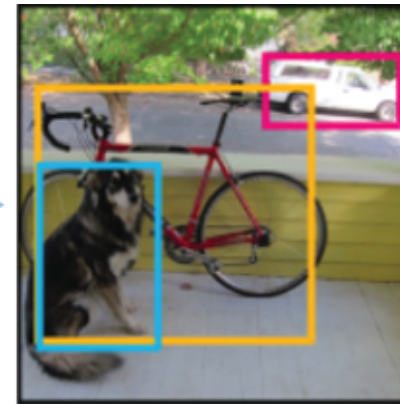
- Ideato da J. Redmon (<https://pjreddie.com/>)
- Tre versioni
 - V1, 2016; V2, 2017; - v3, 2018
 - Fast real time object detection
- Idea: limitiamo i bounding boxes utilizzando una griglia prefissata



Split the image into grids



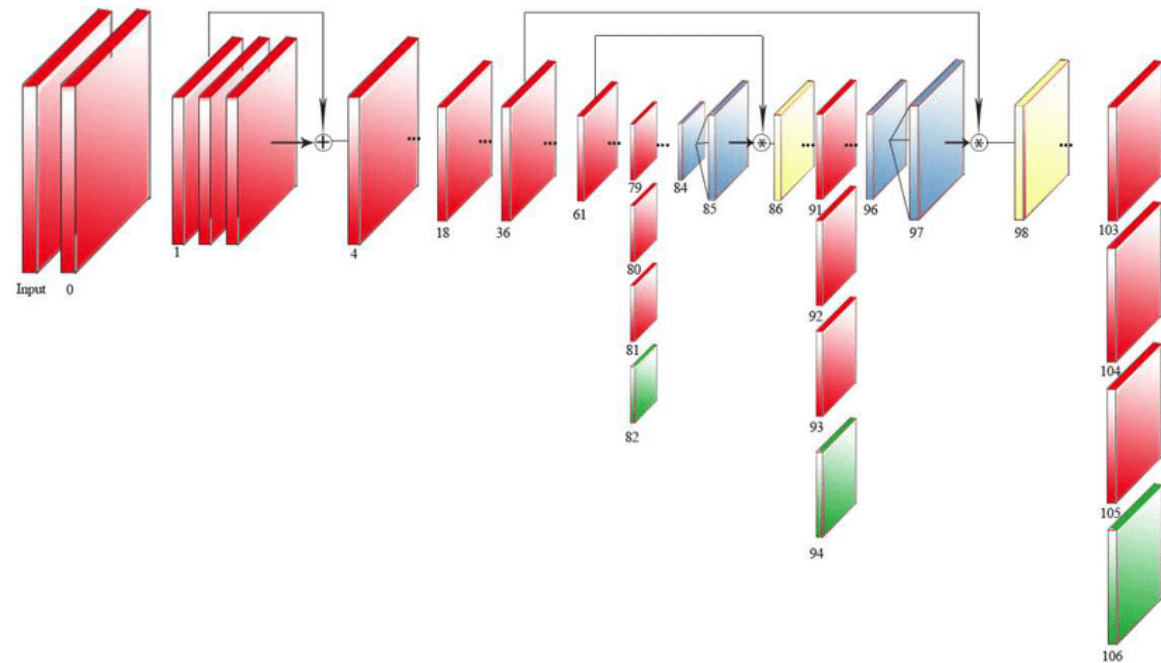
Predict bounding boxes
and classifications



Final predictions after
non-maximum suppression

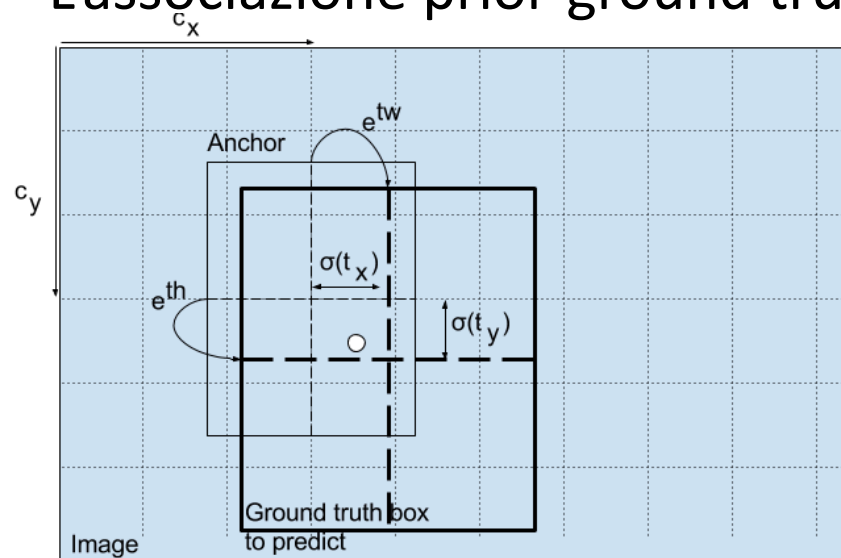
YOLOv3

- Fully Convolutional Networks
 - Simile a SSD
 - 75 convolutional layers
 - skip connections
 - upsampling layers



YOLOv3 - Principi

- Ogni cella definisce B Priors
- La cella è responsabile della predizione per l'oggetto il cui centro ricade in essa
- La predizione è effettuata con riferimento alla prior
 - L'associazione prior-ground truth è fatta sulla base dell'IoU

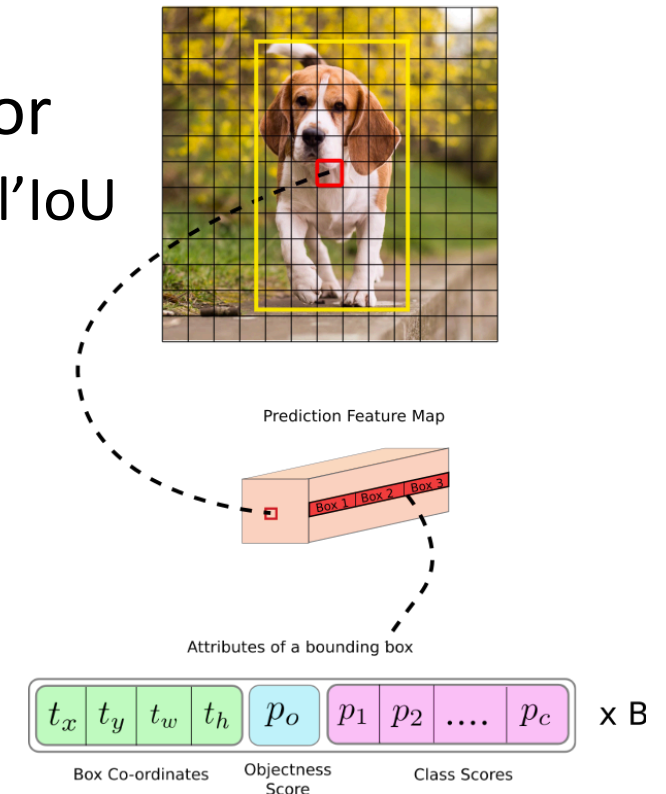


$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

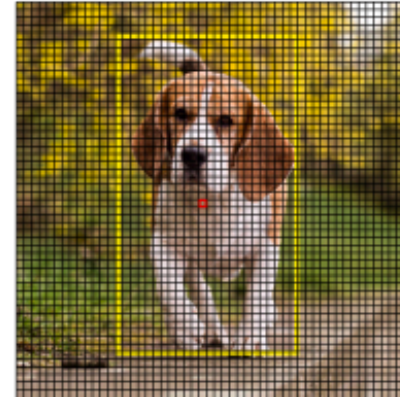
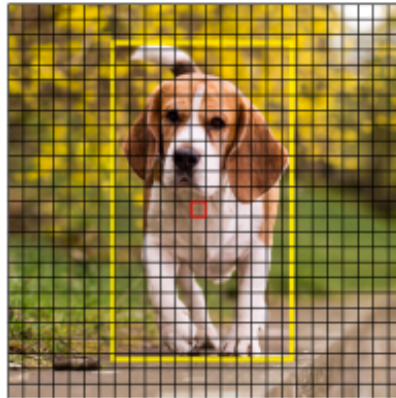
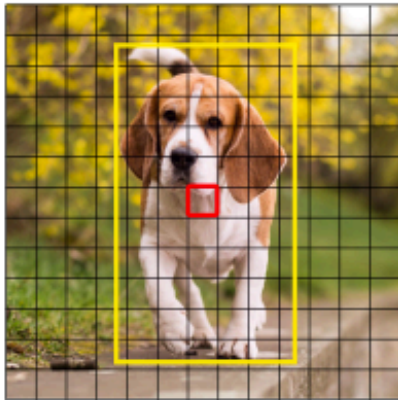
$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$



YOLOv3 - Principi

- Predizione su scale multiple
- Tre griglie con stride 32, 16, 8
 - Su un'immagine 416 x 416, la detection è fatta su griglie 13 x 13, 26 x 26 e 52 x 52



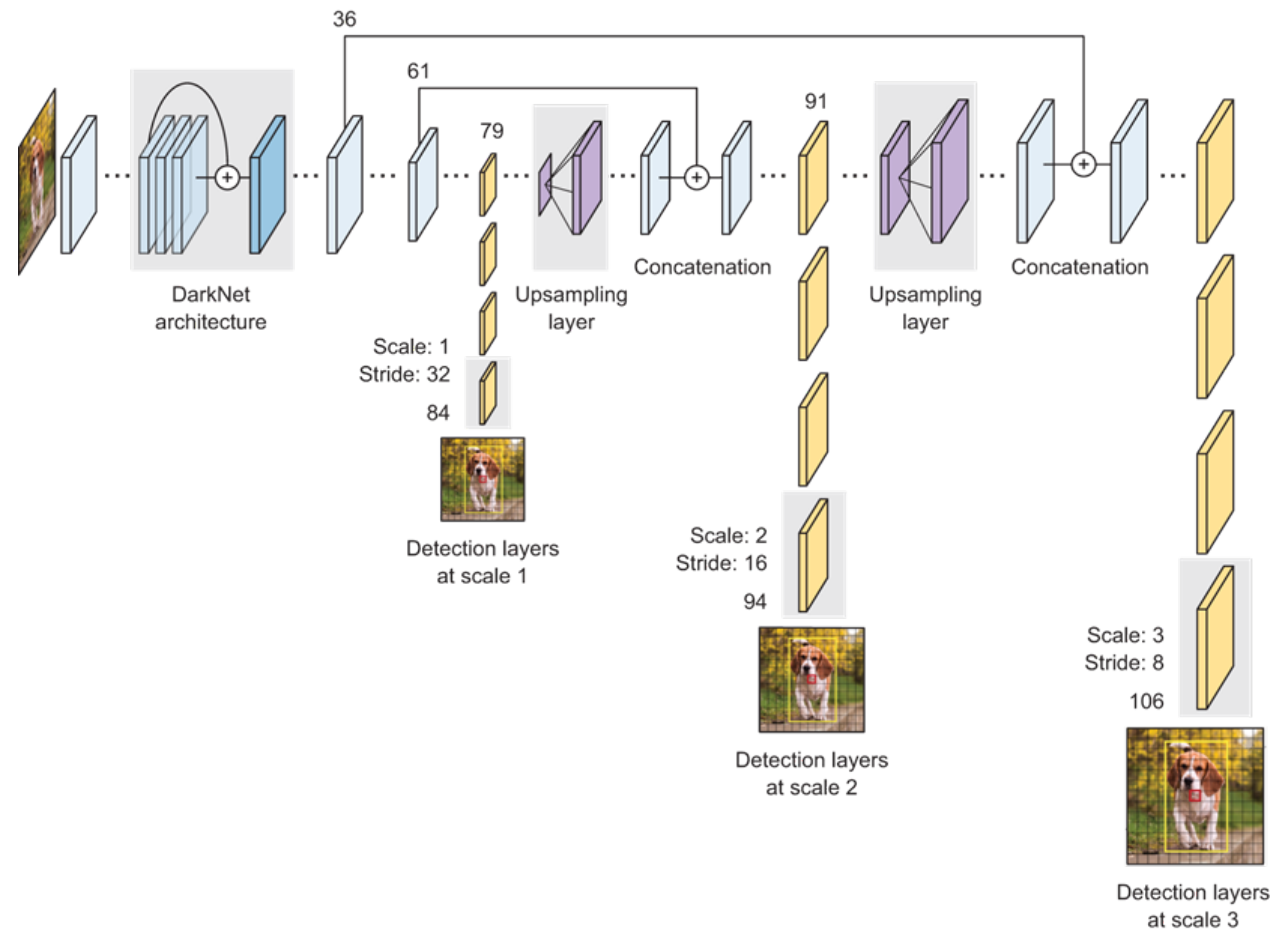
- Risultato: $3549 * B$ predizioni
 - Con $B=3$, 10647 predizioni

YOLOv3 - Architettura

- 1x1 CONV seguiti da 3x3 CONV
- Residual Blocks
- Feature extractor basato su Darknet-53
 - 106 fully convolutional layers

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

YOLOv3 - Architettura



YOLO - Loss

$$\begin{aligned} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\ & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\ & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

Regression

Object/no object confidence

Class prediction

YOLO - Loss

$$\begin{aligned}
 & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
 \end{aligned}$$

Cell i contains object, predictor j is responsible for it

Small deviations matter less for larger boxes than for smaller boxes

Confidence for object

Confidence for no object

Down-weight loss from boxes that don't contain objects ($\lambda_{\text{noobj}} = 0.5$)

Class probability

YOLOv3 - Loss

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$


$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

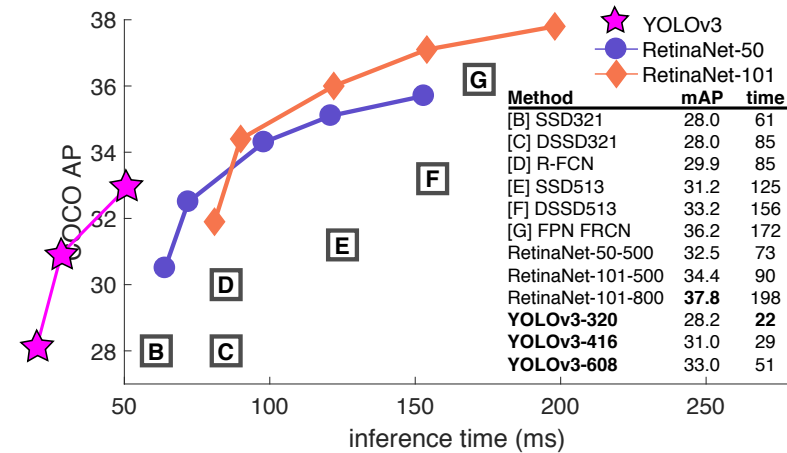
$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Rimpiazzati da
Cross-Entropy/Focal Loss



Sommario

- Estremamente veloce
- Accuratezza comparabile



Metodo	FPS	mAP	Priors
Faster R-CNN	5-7	69,9%	~6000
SSD300	46	74.3%	8732
SSD512	19	76,8%	24564
YOLOv3	~100	57,9%	10647