Analisi di Immagini e Video (Computer Vision)

Giuseppe Manco

Outline

- Segmentation
- Approcci classici
- Deep Learning for Segmentation

Crediti

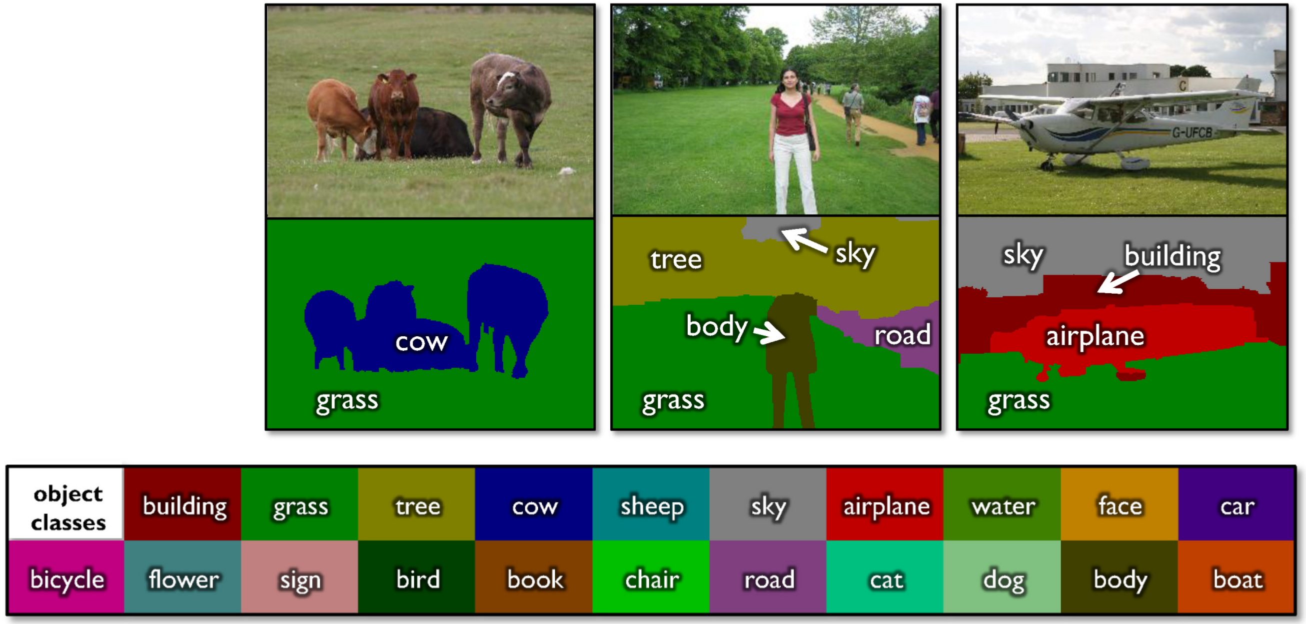
- Slides adattate da vari corsi e libri
  - Computational Visual Recognition (V. Ordonez), CS Virgina Edu
  - Computer Vision (S. Lazebnik), CS Illinois Edu

Approcci supervisionati

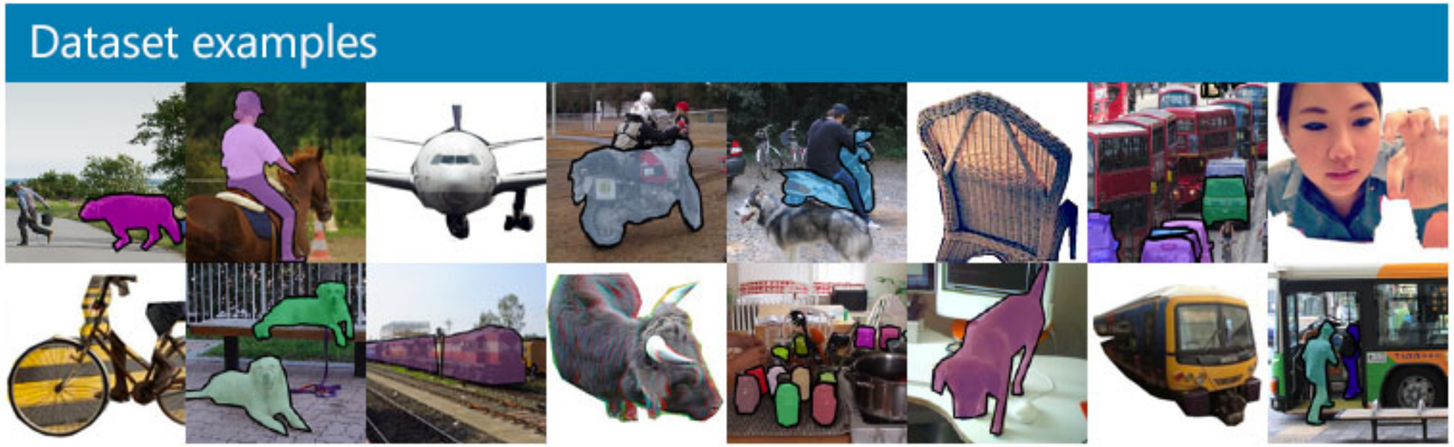
- L'approccio basato su CRF è semi-supervisionato
- Possiamo renderlo supervisionato?
  - Parametrizziamo gli unary e binary potentials
    - E.g., $p(y_i|x_i;\theta) = \frac{1}{Z}\exp\left(w_{y_i} \cdot F(x_i)\right)$
  - Apprendiamo i parametri che minimizzano l'energia media su tutti gli esempi

Semantic segmentation, object detection

- Problema
  - Etichettare ogni pixel con una classe
  - Multi-class problem
- Utilizzo di dati già etichettati
  - Pascal VOC
  - MS COCO



MS-COCO



- Large-scale dataset for object detection, segmentation and captioning
  - 330K images (>200K labeled)
  - 1.5 million object instances
  - 80 object categories
  - 91 stuff categories
  - 5 captions per image
  - 250,000 people with keypoints

Perché Deep Learning?

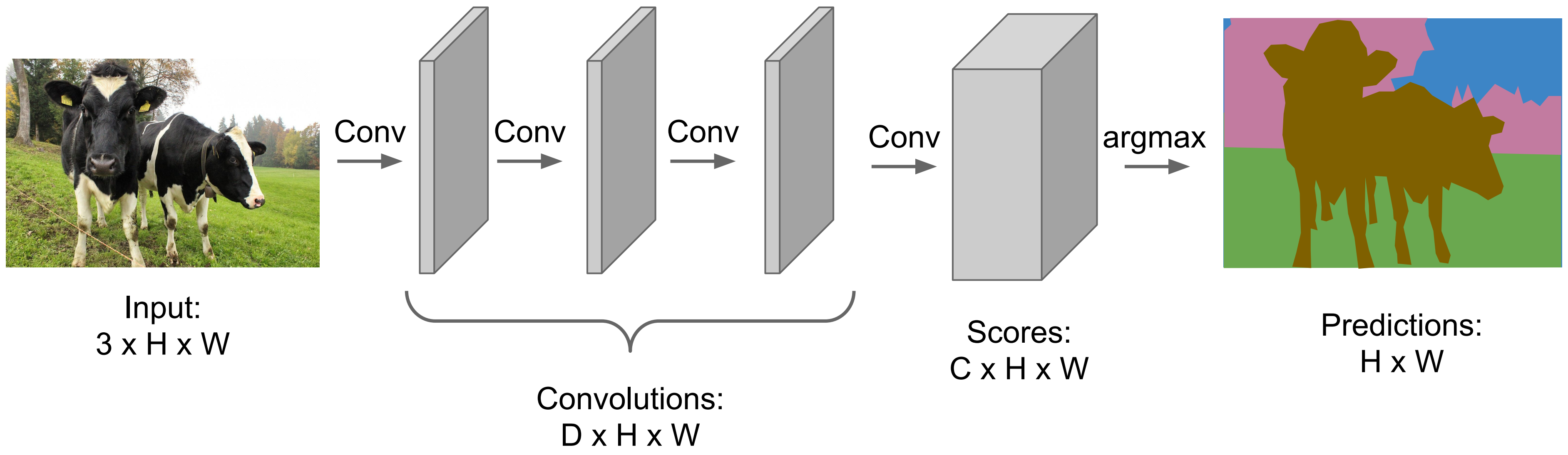
- Stesso principio dell’object detection
  - Convolutional features, learned from training data
- Accuratezza
- Velocità

Approcci

- Approcci downsampling-upsampling
- Metodi multi-scala

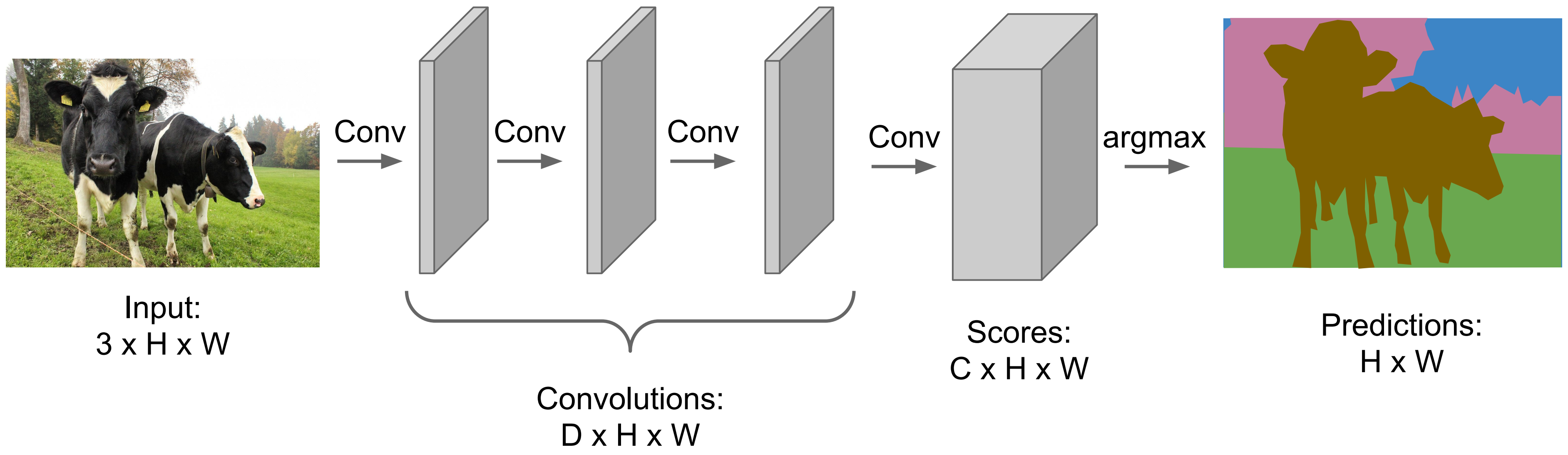
Fully Convolutional Networks

- Utilizziamo i layer convoluzionali per fare le predizioni sui vari pixel



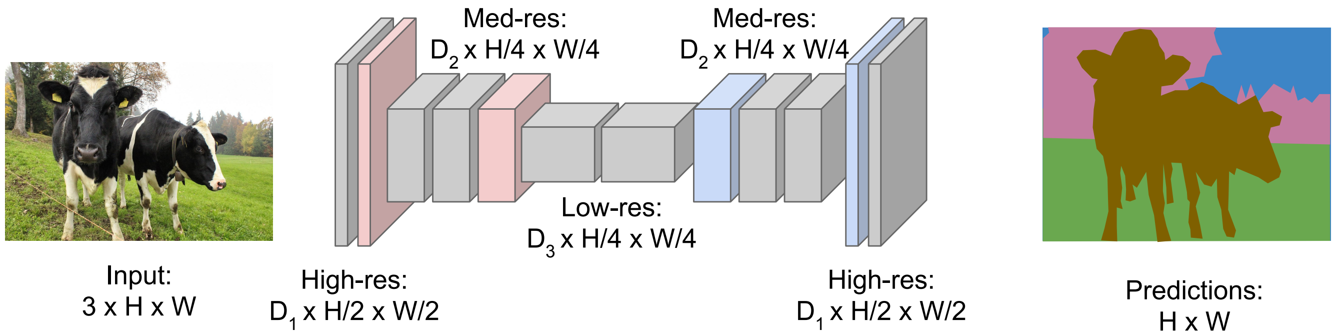
Fully Convolutional Networks

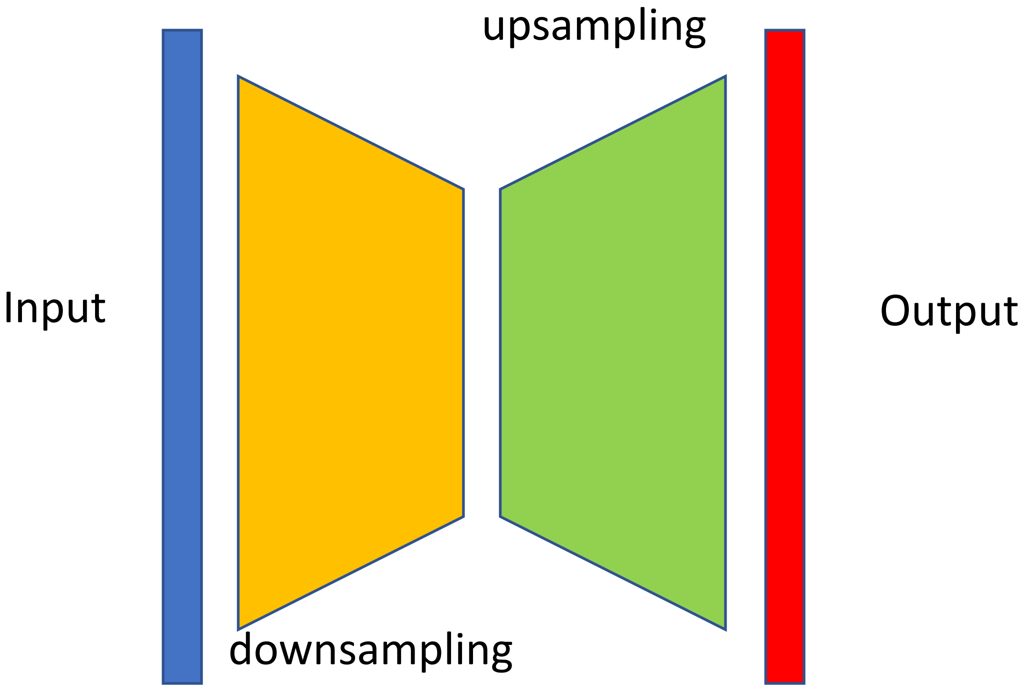
- Utilizziamo i layer convoluzionali per fare le predizioni sui vari pixel
  - Ma fare convoluzioni su feature map grandi è costoso



Fully Convolutional Networks

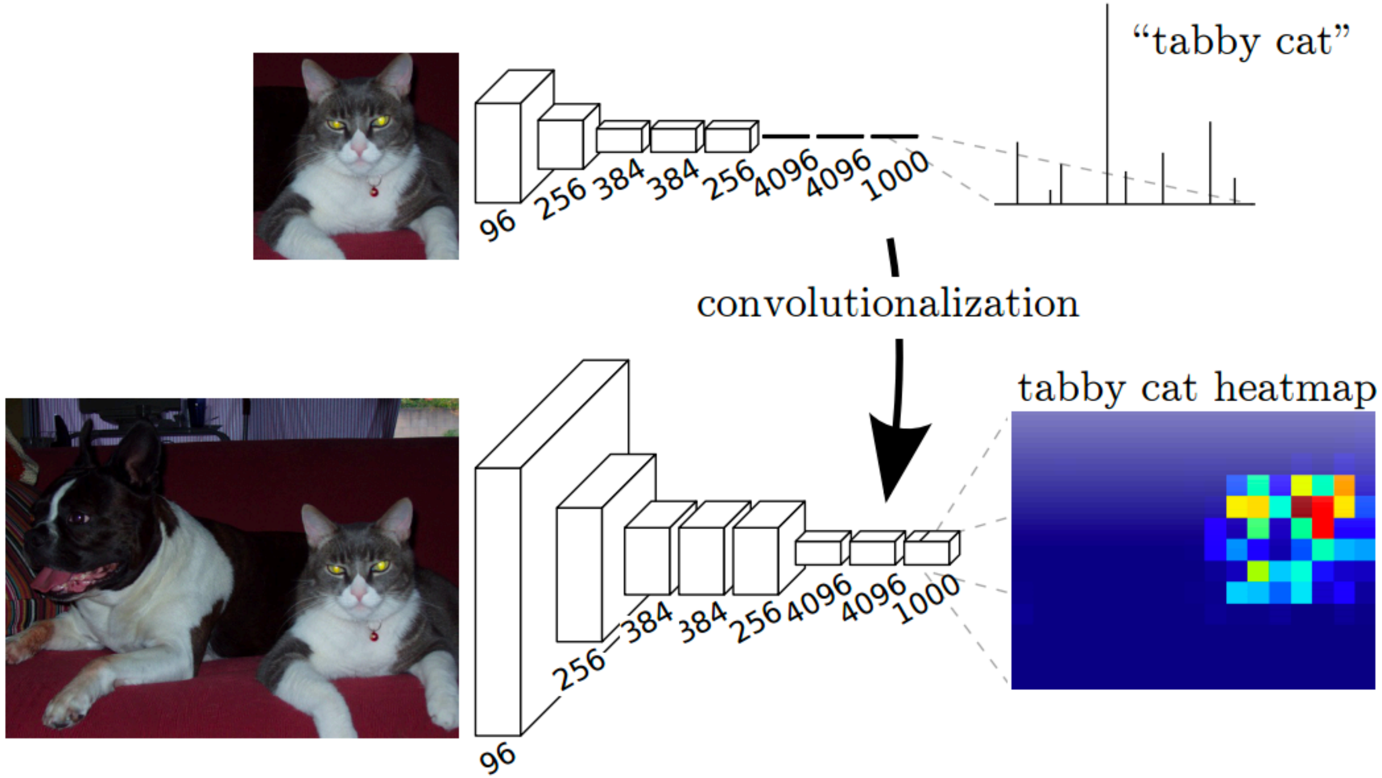
- Soluzione
  - Architettura Encoder-Decoder





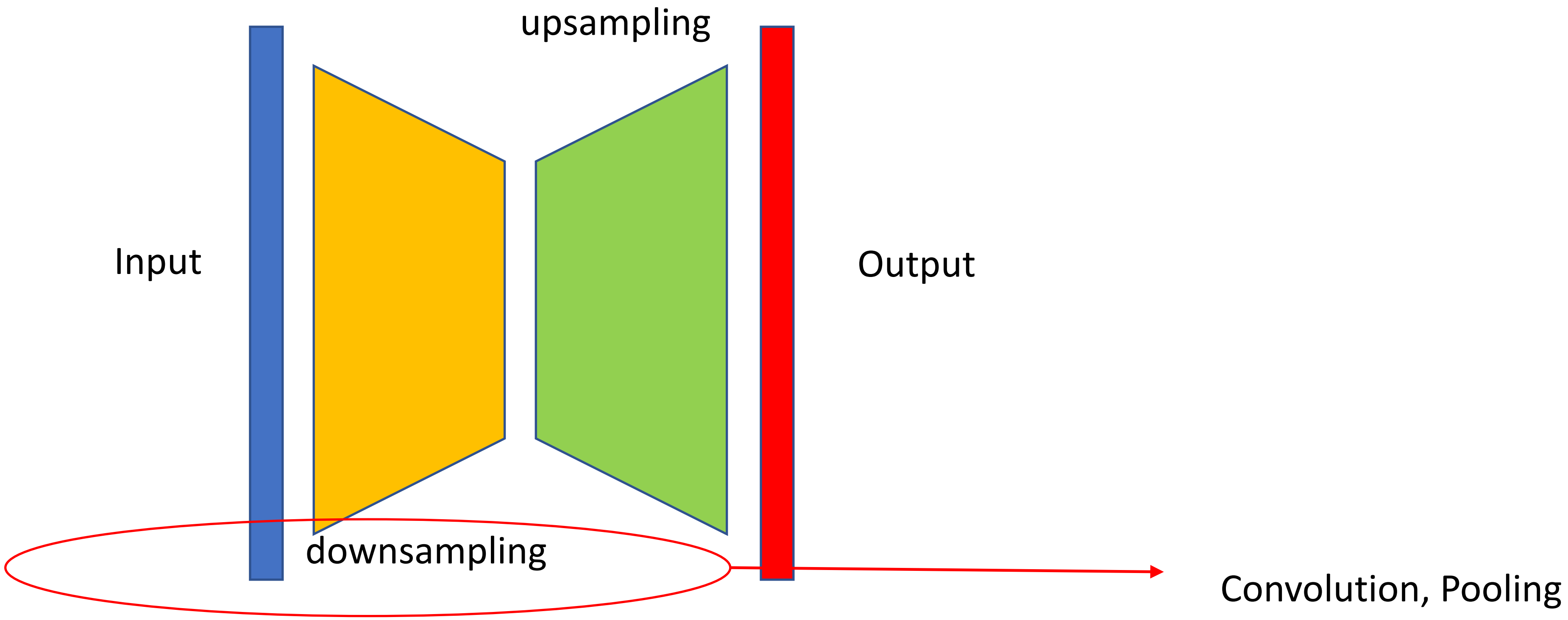
Convolutionalization

- Fully Convolutional Layers
- Faster-RCNN, SSD



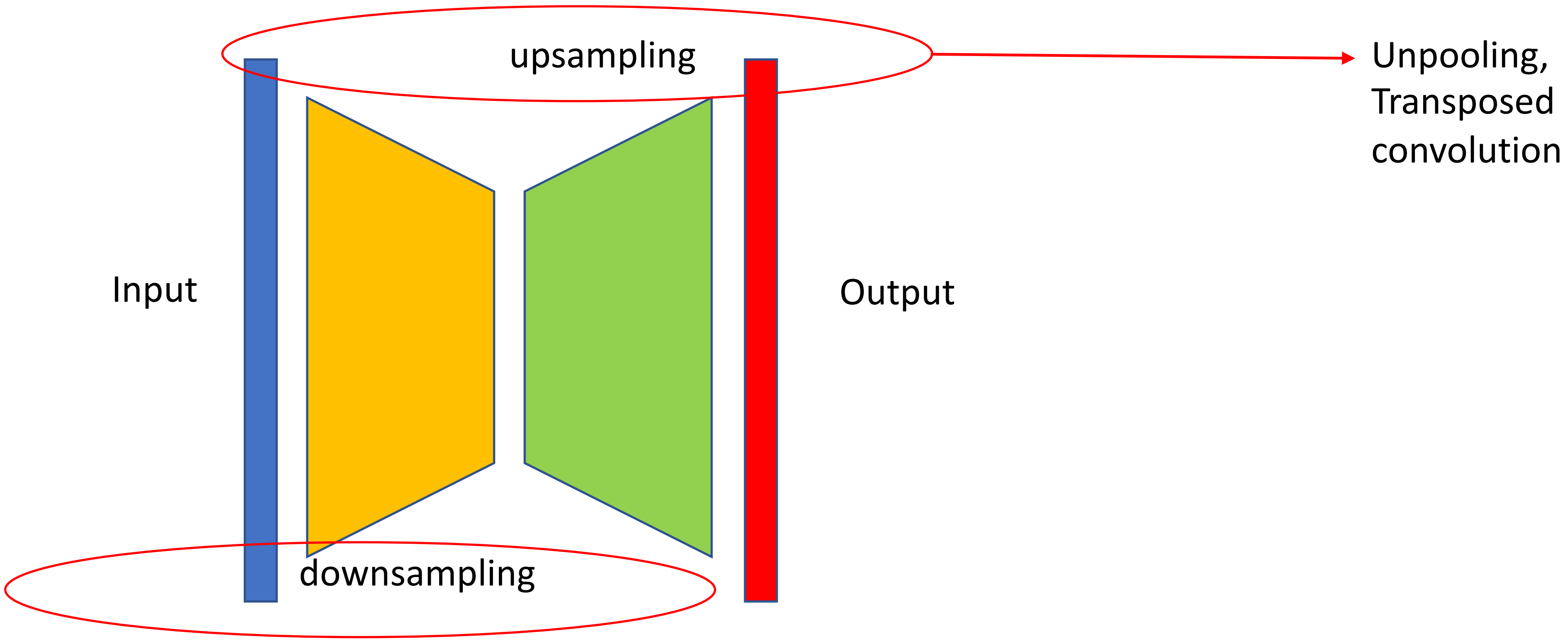
Fully Convolutional Networks

- Architettura Encoder-Decoder



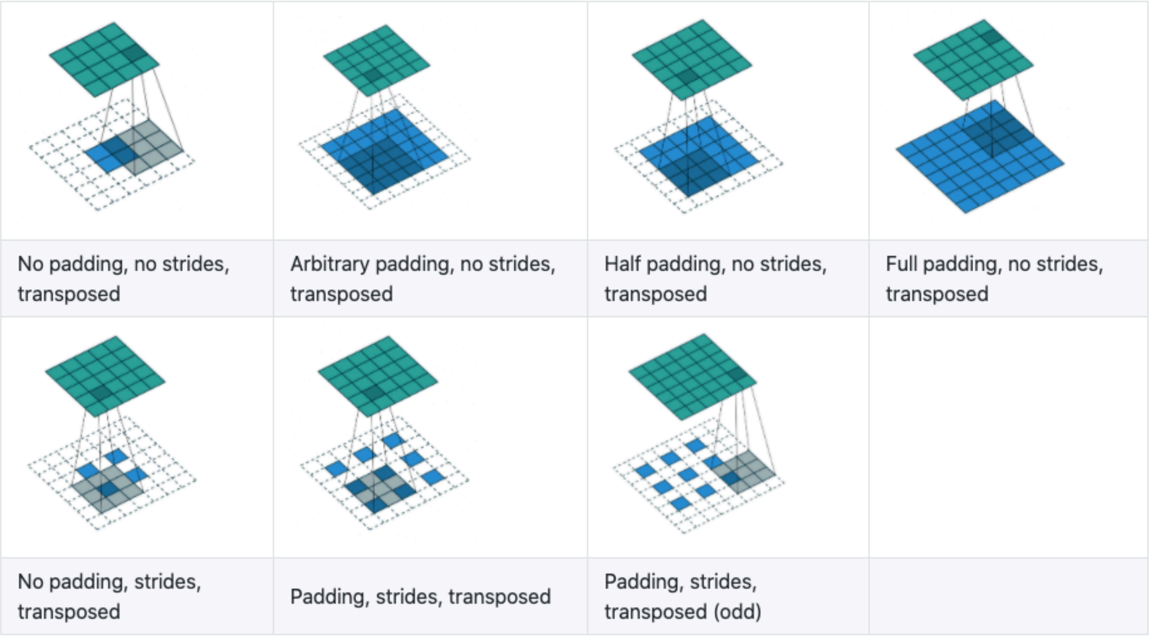
Fully Convolutional Networks

- Architettura Encoder-Decoder



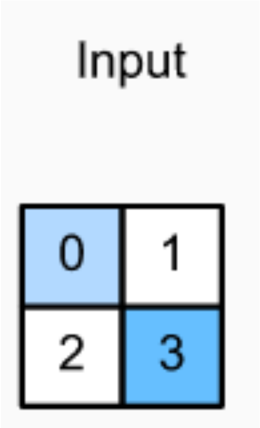
Up-sampling Convolutions

- Upsampling
  - Da un’input a bassa risoluzione si passa ad uno a più alta risoluzione
  - Transposed Convolution
    - Qual è la relazione?
      - Suggerimento: invertiamo le relazioni originarie

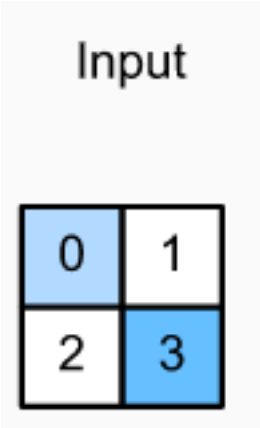


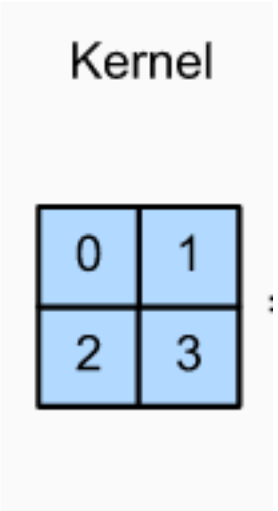
https://github.com/vdumoulin/conv_arithmetic

Transposed Convolution

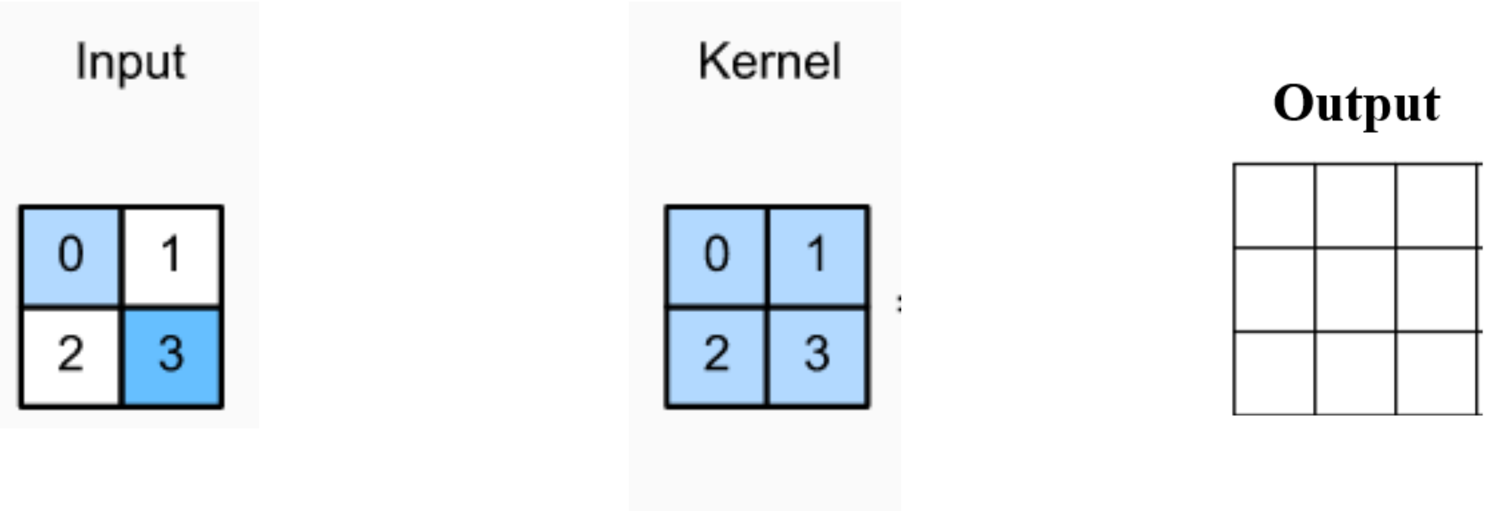


Transposed Convolution



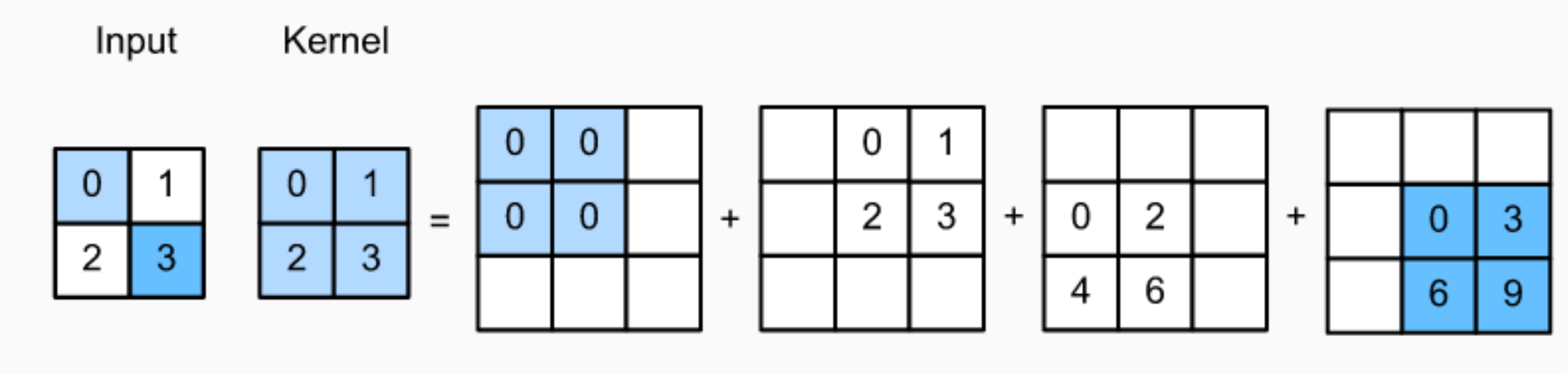


Transposed Convolution



Transposed Convolution

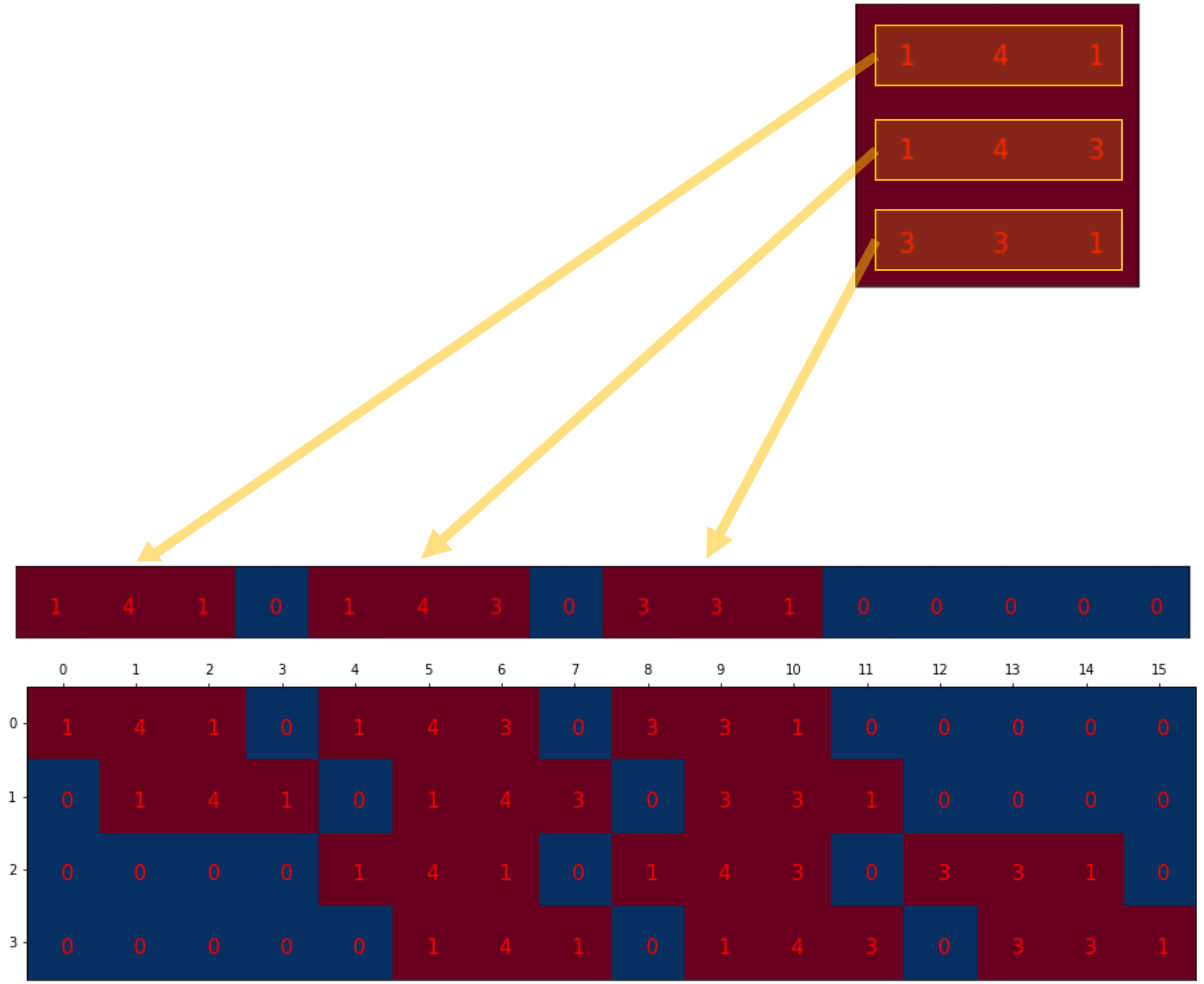
- Ogni valore si distribuisce su un intorno dell’output in base al kernel.



- La distribuzione viene guidata da padding e stride

Convolution e transposed convolution

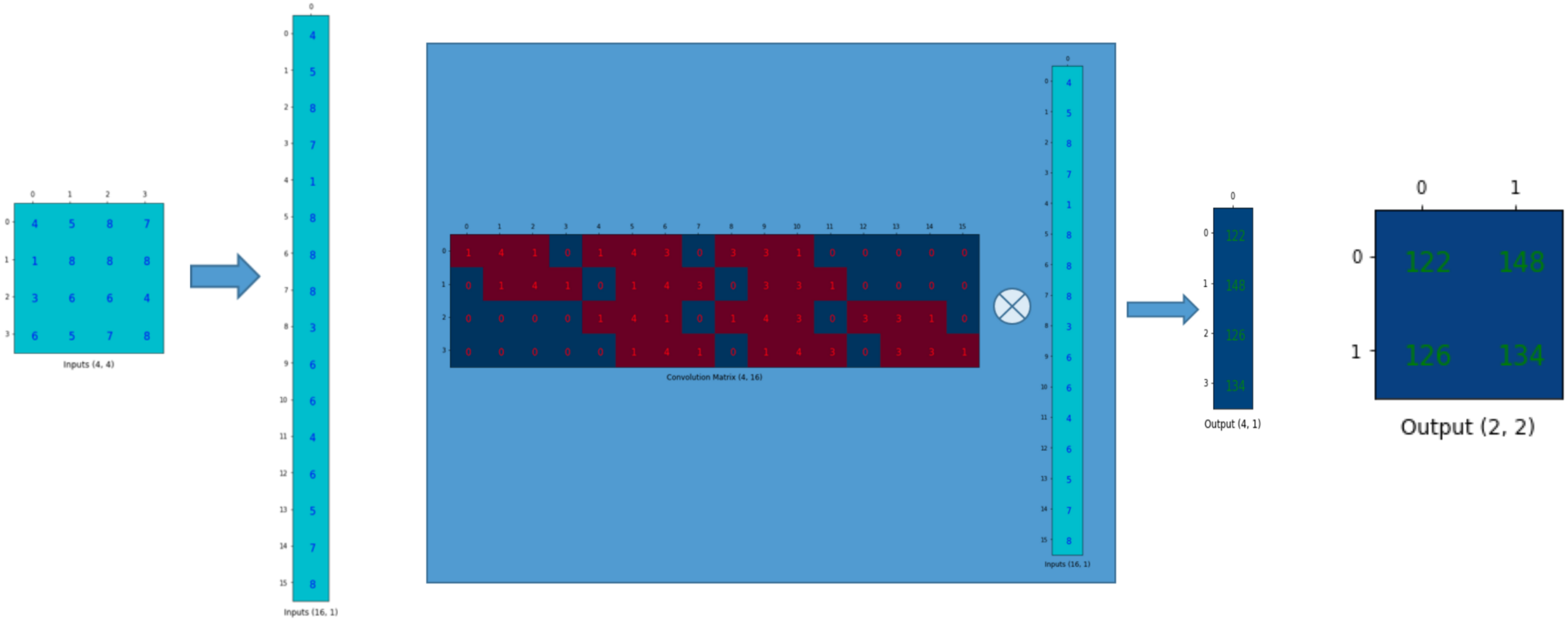
- Ogni riga definisce un'operazione di convoluzione
  - Filtro 3x3, input 4x4
    - No padding, no strides, no dilation



Convolution Matrix (4, 16)

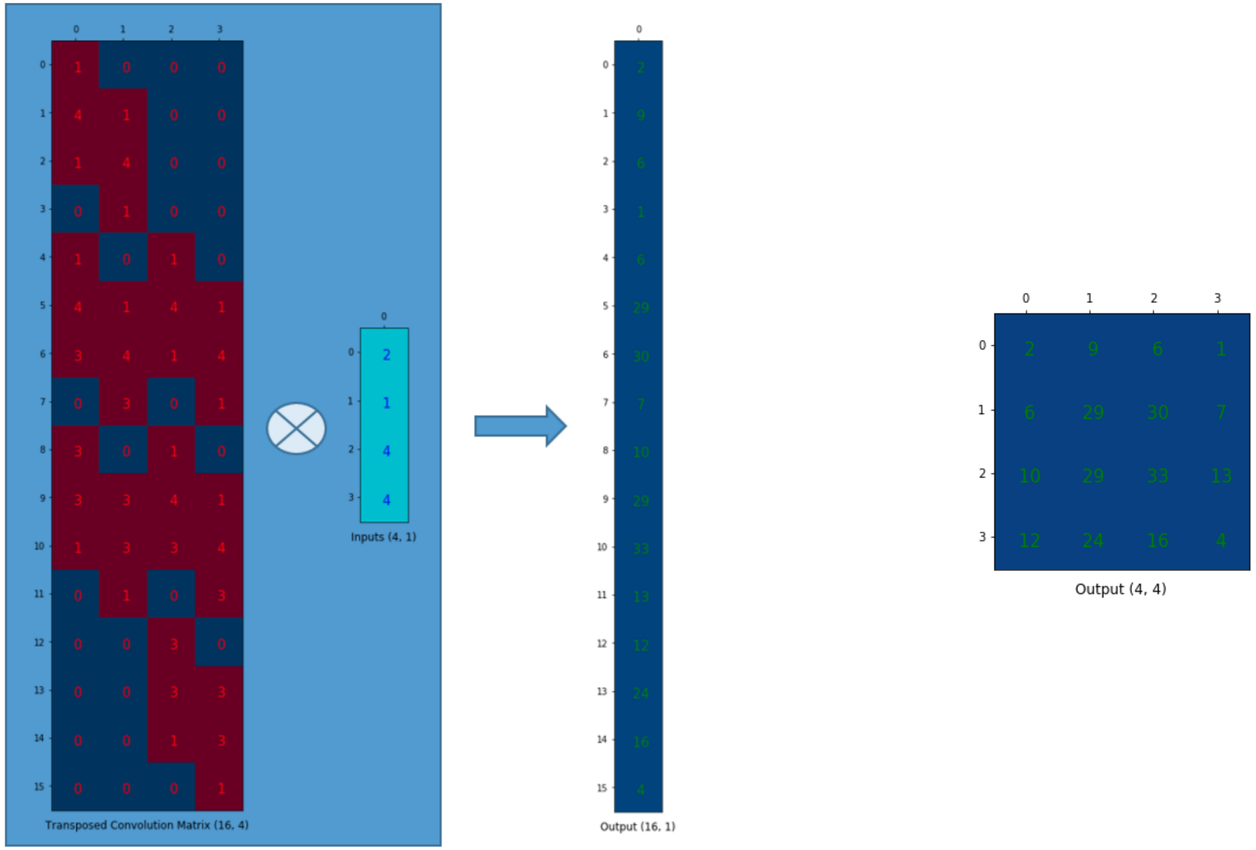
Convolution e transposed convolution

- Ogni riga definisce un’operazione di convoluzione



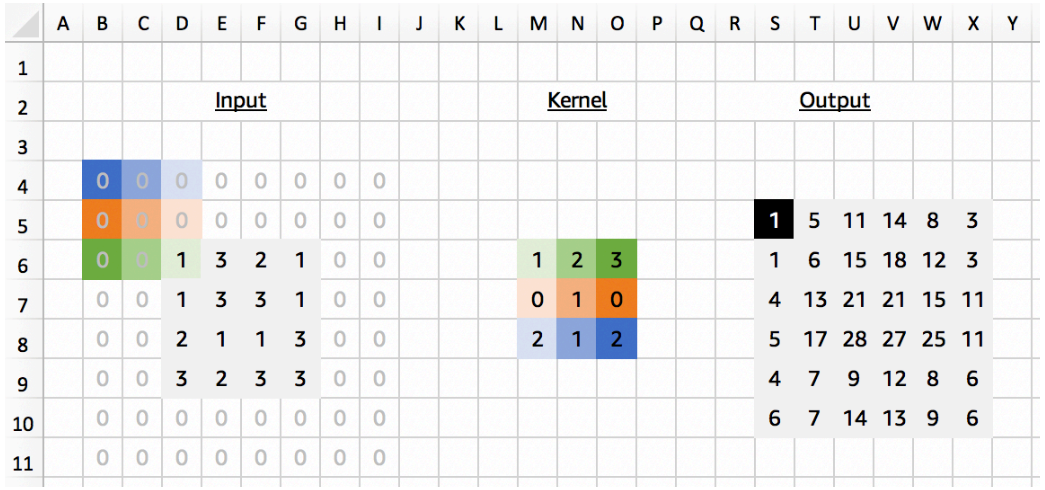
Convolution e transposed convolution

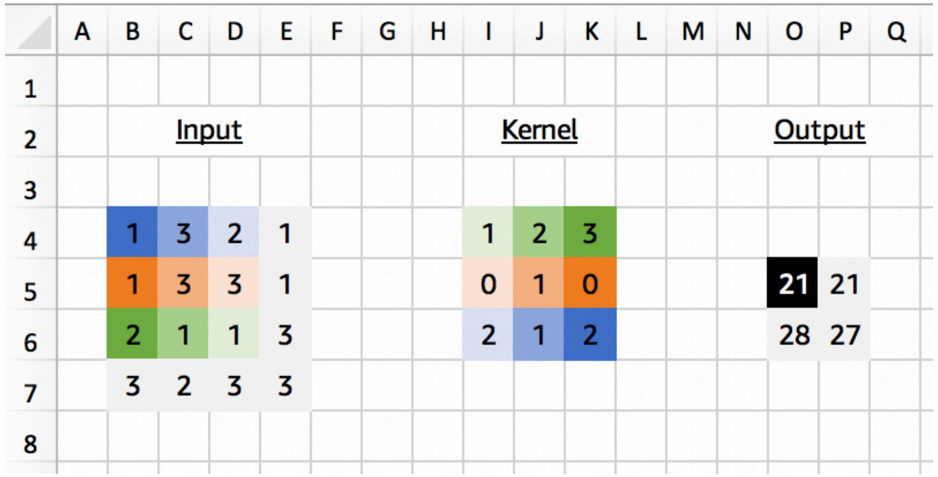
- Trasponendo la matrice di convoluzione, otteniamo l'operazione opposta



ConvTranspose, Padding

- Decrementa l’output della TD
  - Interpretazione: l’ammontare di padding che l’input richiede per completare l’output
  - Quale sarebbe l’output dell’esempio precedente?

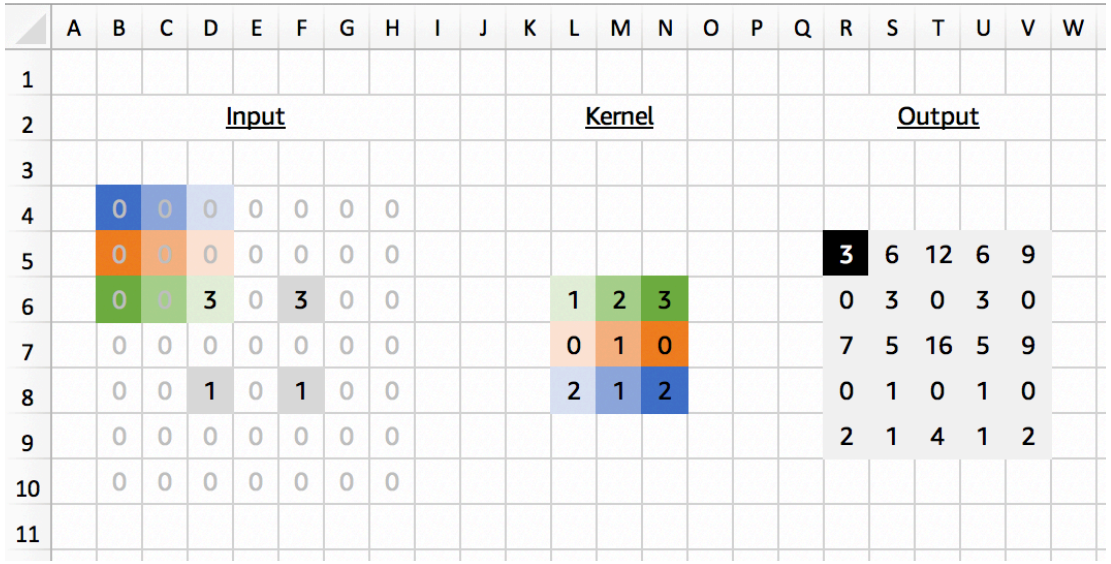


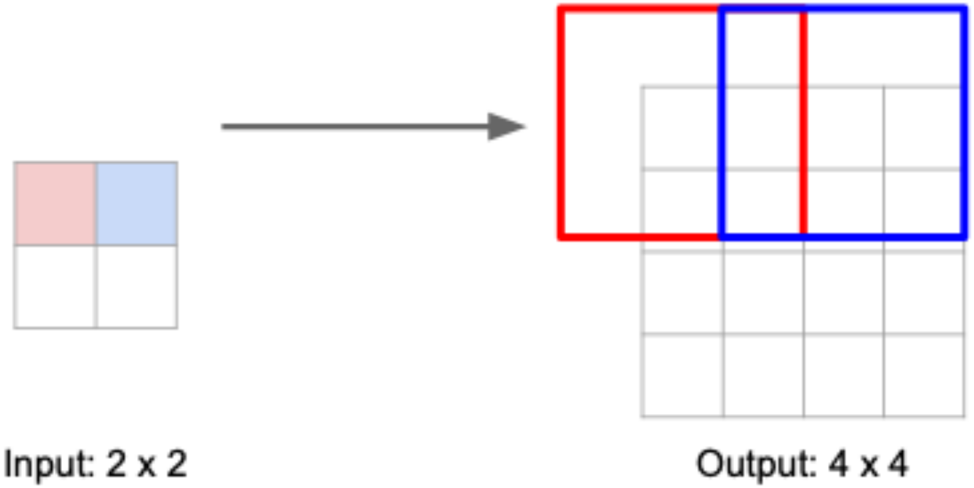


ConvTranspose, Stride

- Espande l’output
  - Di conseguenza «fraziona» l’input aggiungendo spazi

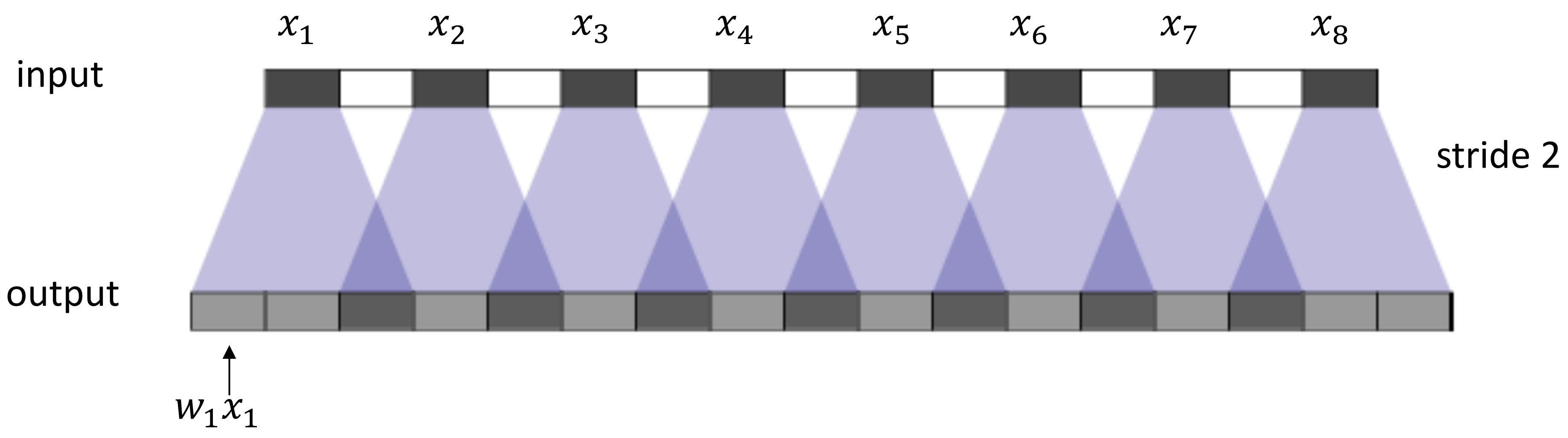
Il filtro si muove di 2 pixel sull’output





ConvTranspose, checkerboarding

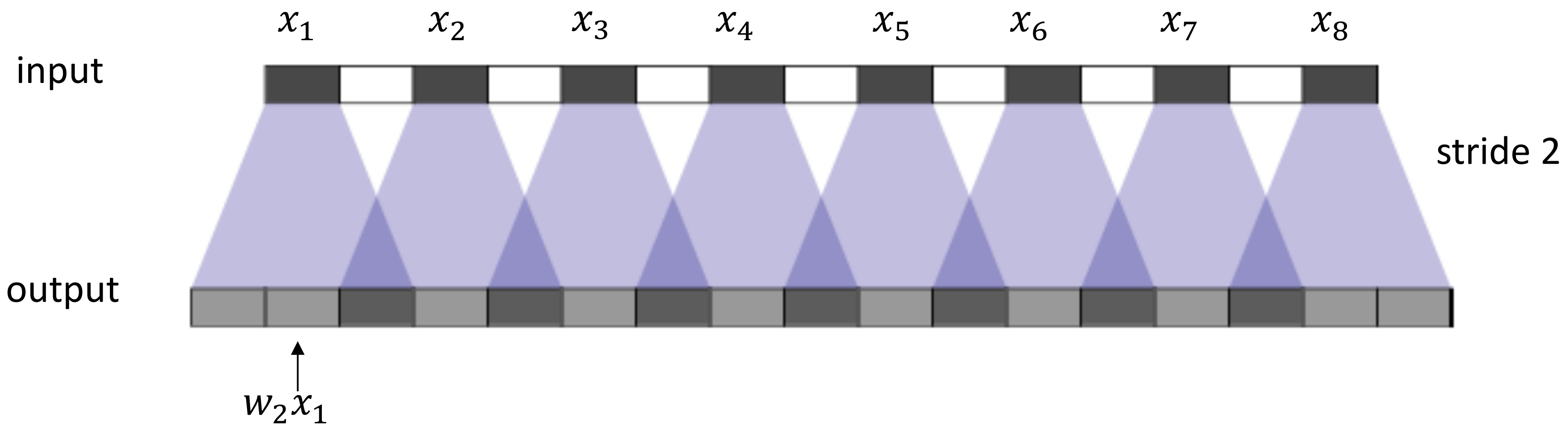
- Filter $[w_1, w_2, w_3]$, *output stride* = 2



Animation: https://distill.pub/2016/deconv-checkerboard/

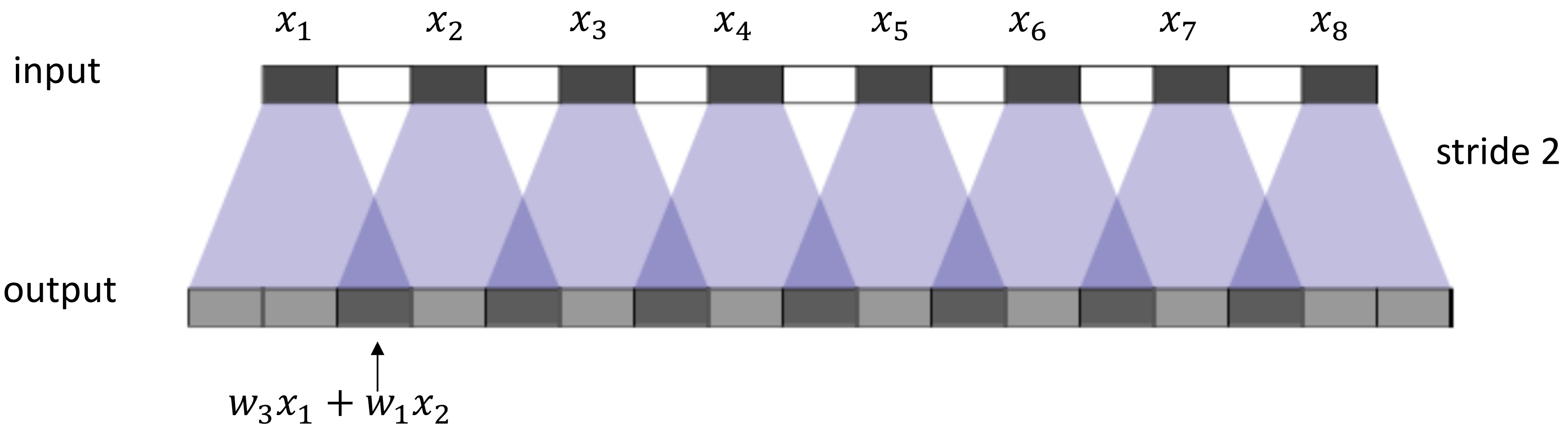
ConvTranspose, checkerboarding

- Filter $[w_1, w_2, w_3]$, *output stride* = 2



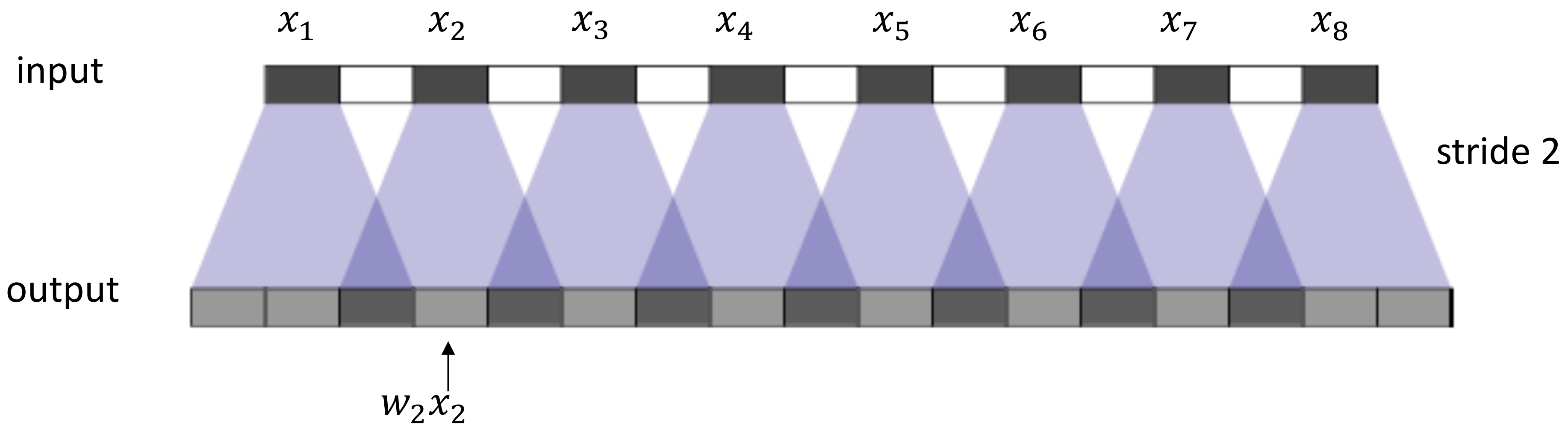
ConvTranspose, checkerboarding

- Filter $[w_1, w_2, w_3]$, *output stride* = 2



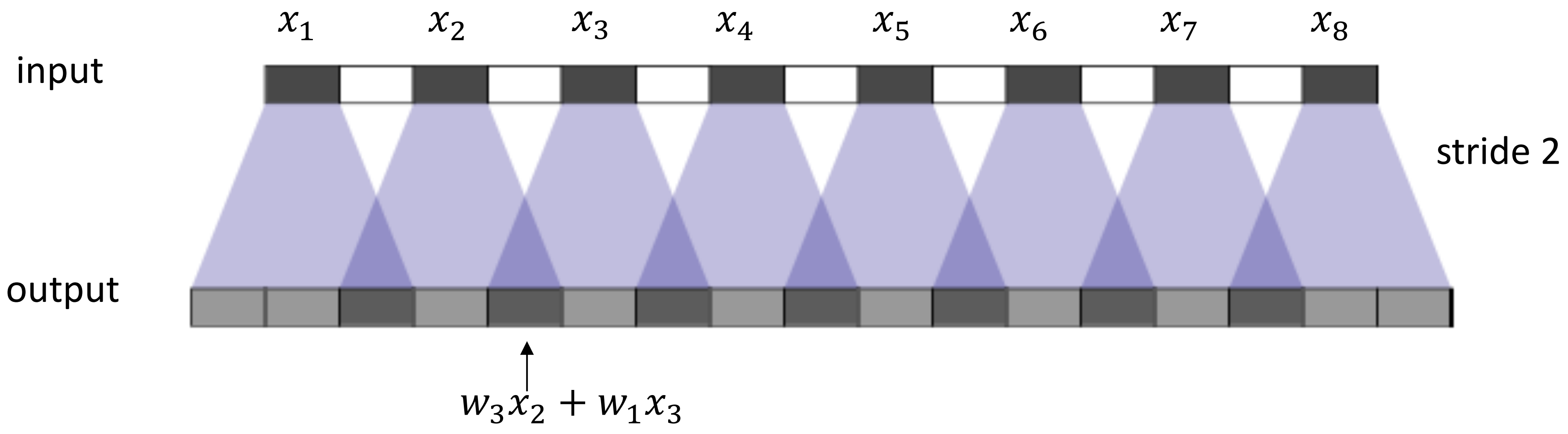
ConvTranspose, Checkerboarding

- Filter $[w_1, w_2, w_3]$, *output stride* = 2



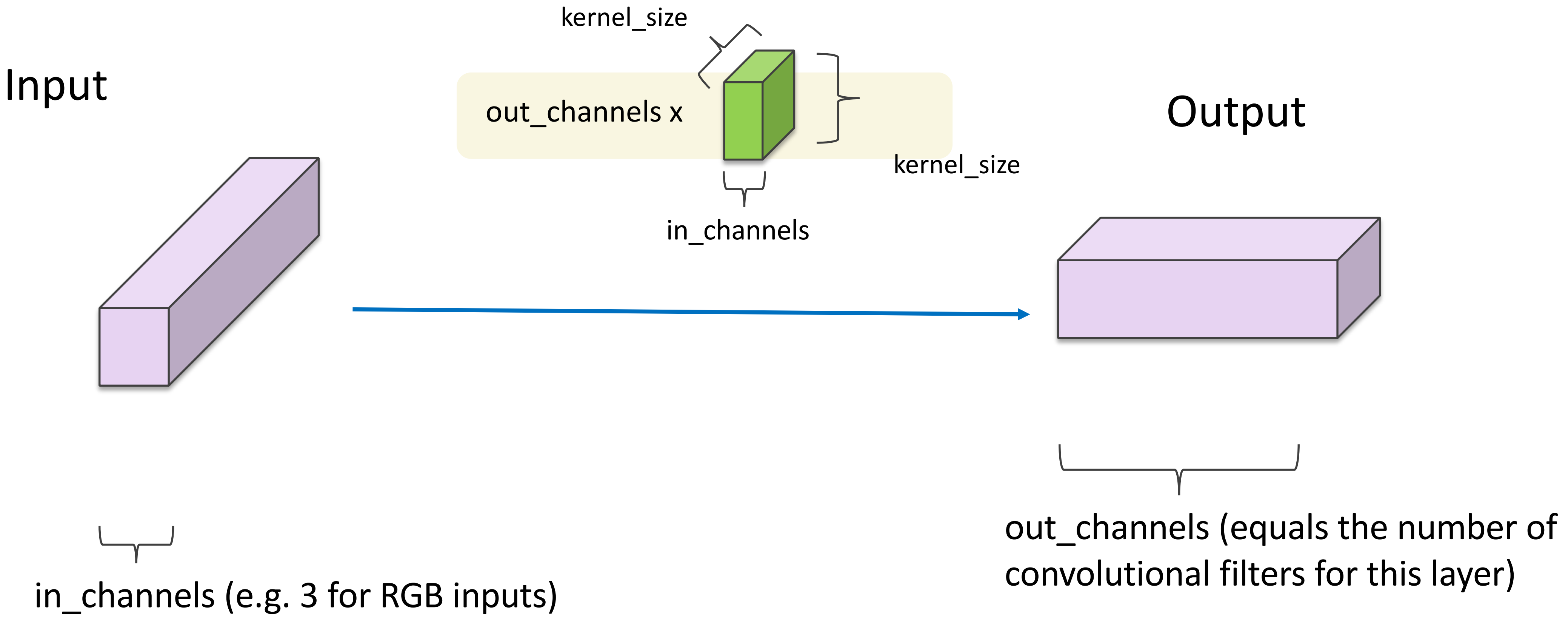
ConvTranspose, Checkerboarding

- Filter $[w_1, w_2, w_3]$, *output stride* = 2



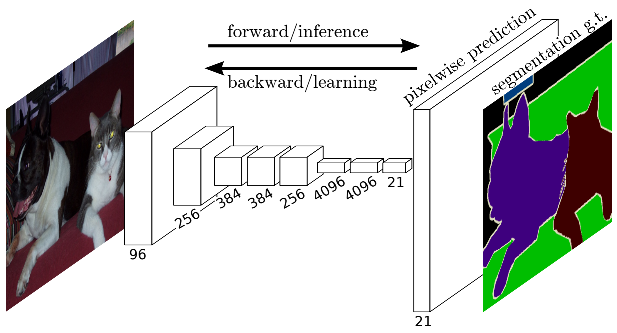
Transposed Convolution in Pytorch

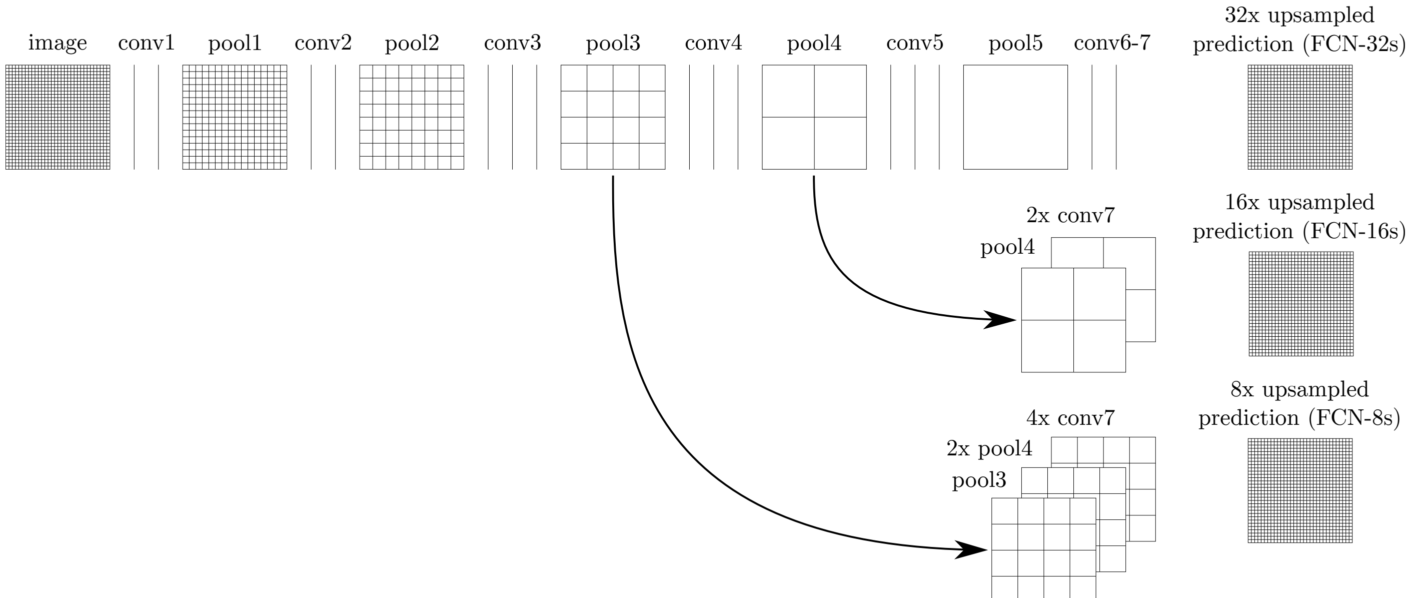
CLASS `torch.nn.ConvTranspose2d(in_channels, out_channels, kernel_size, stride=1, padding=0, output_padding=0, groups=1, bias=True, dilation=1, padding_mode='zeros')`



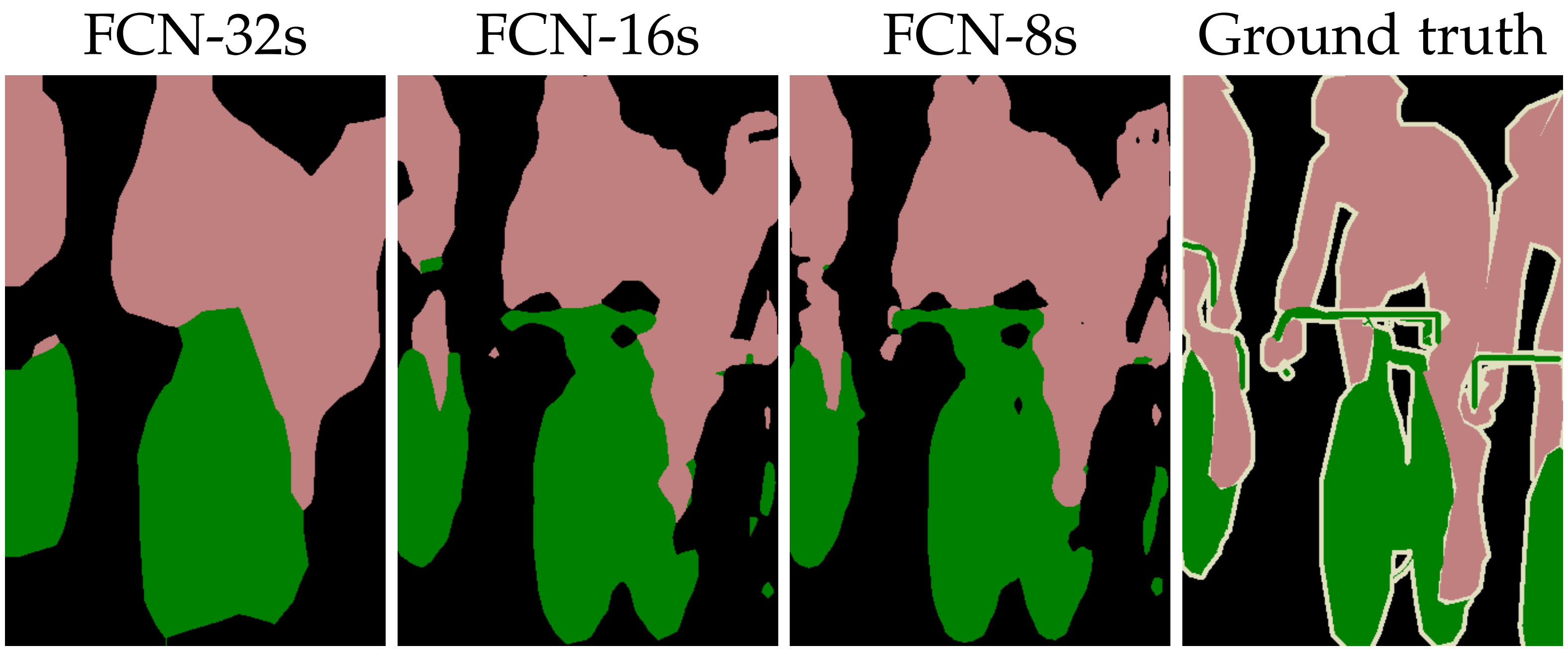
FCN: architettura

- Principio
  - Riduciamo la dimensione, facciamo upsampling
- Tre varianti
  - Coarse upsampling
  - Combined upsampling, skip connections (tramite somma)





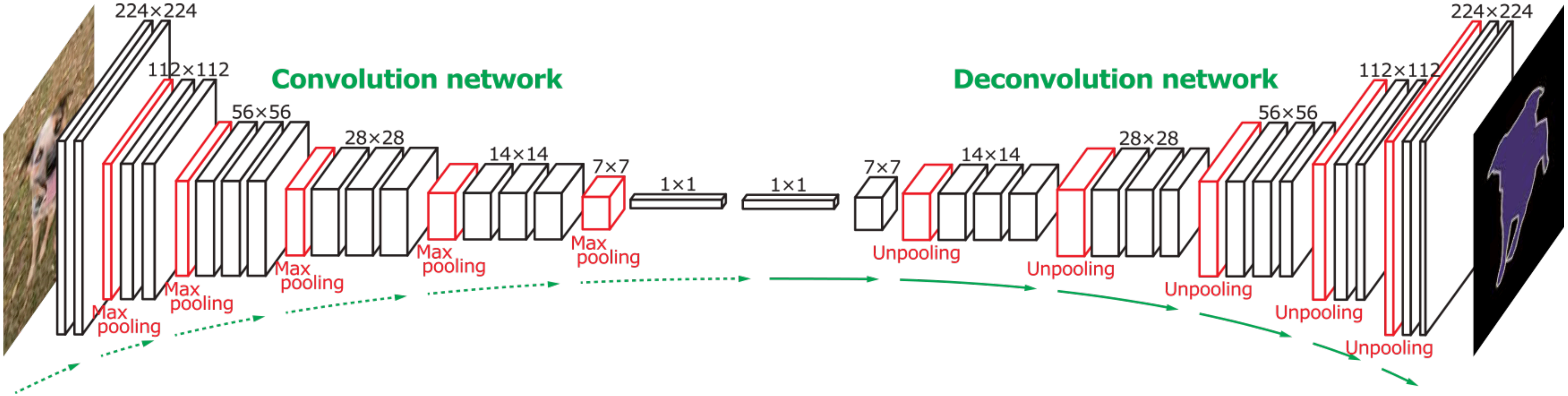
FCN



- L’utilizzo di ConvTranspose con stride di grandi dimensioni causa la presenza di artefatti
- Scarsa risoluzione ai bordi
  - L’encoding causa perdita di informazione

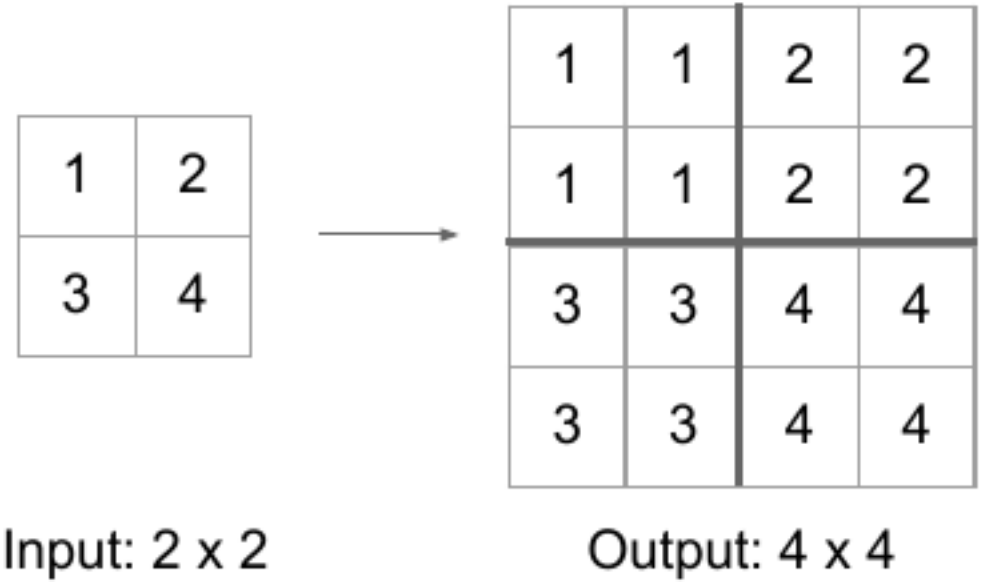
DeconvNet
Up-sampling Convolutions or ”Deconvolutions”

- Backbone: VGG



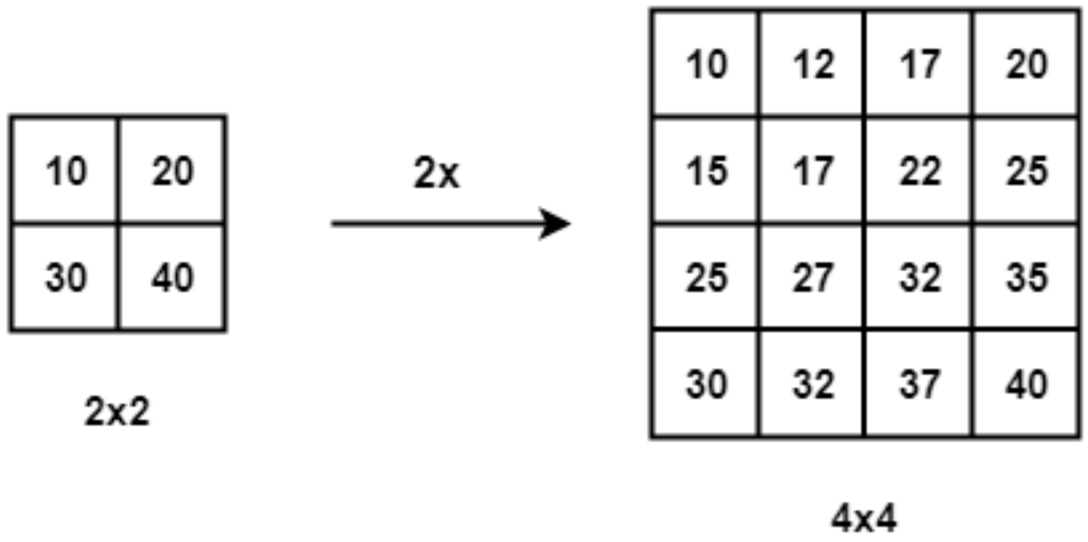
http://cvlab.postech.ac.kr/research/deconvnet/

Unpooling



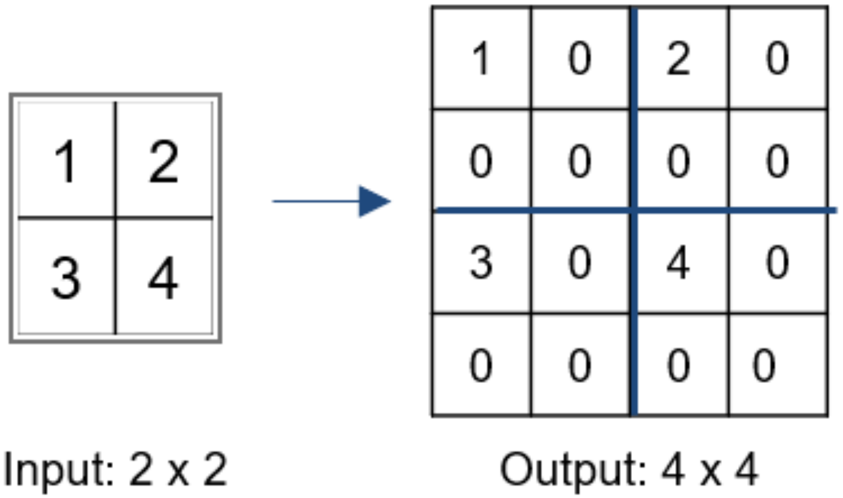
Unpooling

- Bilinear interpolation



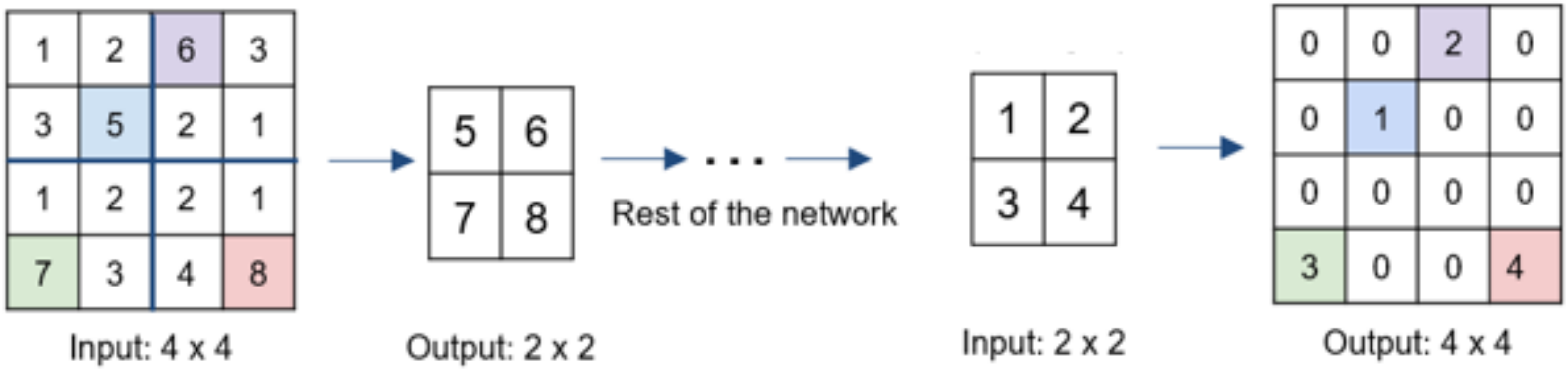
Unpooling

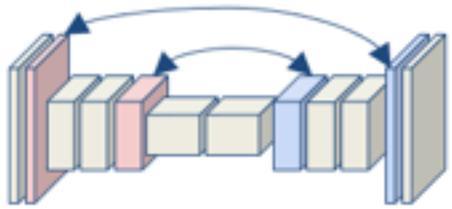
- Bed of nails



Unpooling

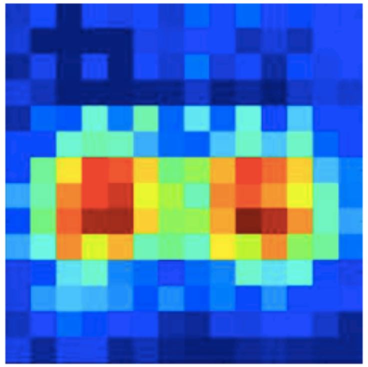
- Max unpooling

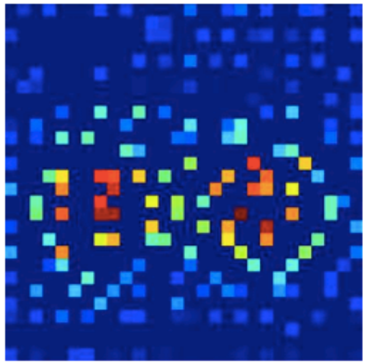


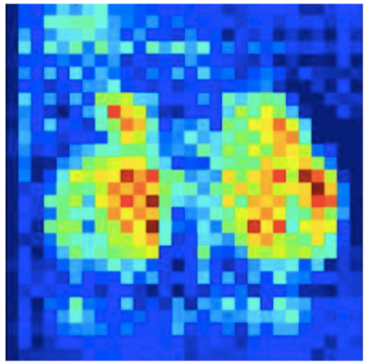


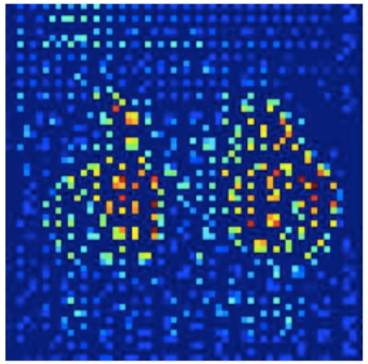
DeconvNet

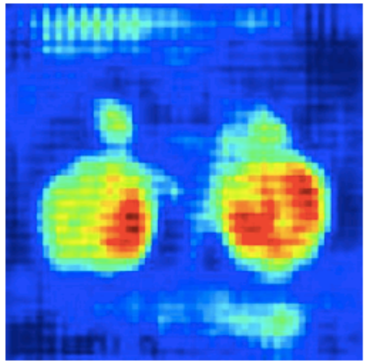

Original image

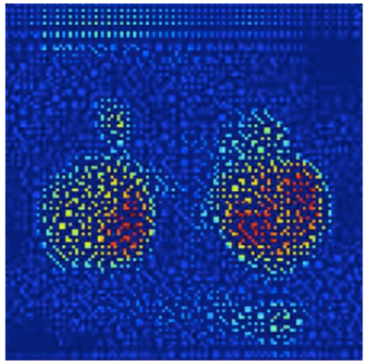

14x14 deconv

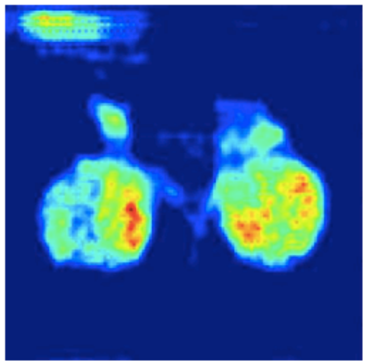

28x28 unpooling

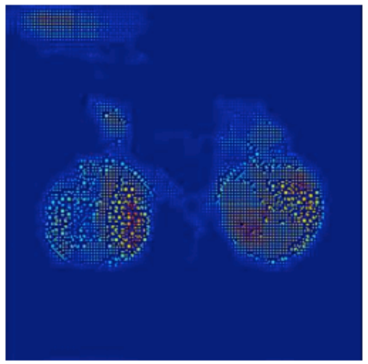

28x28 deconv

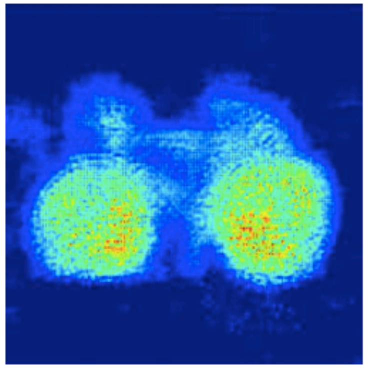

54x54 unpooling


54x54 deconv

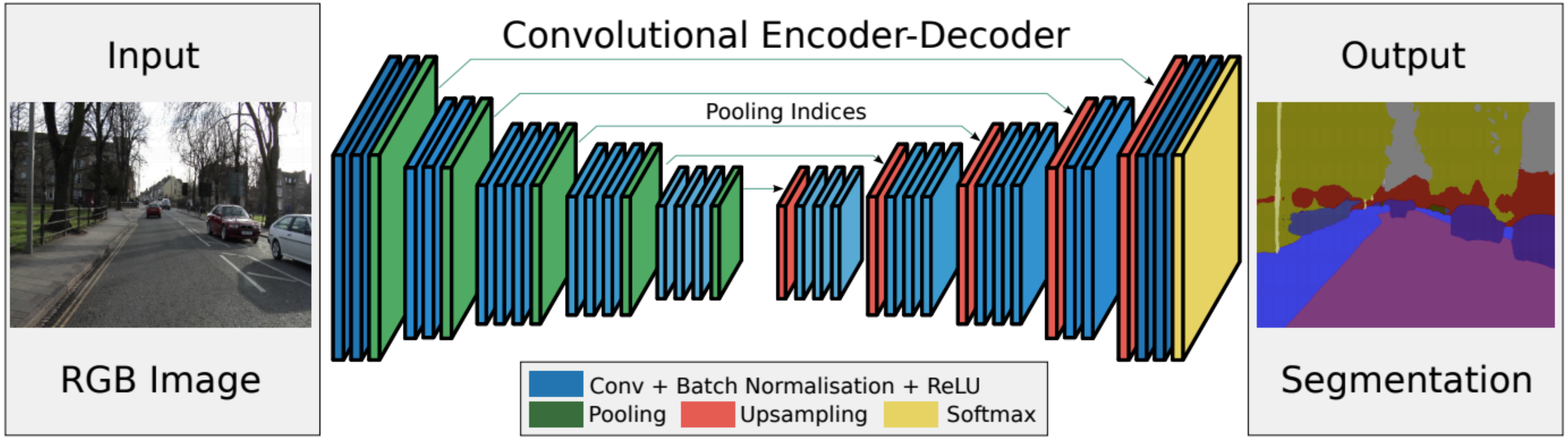

112x112 unpooling


112x112 deconv


224x224 unpooling


224x224 deconv

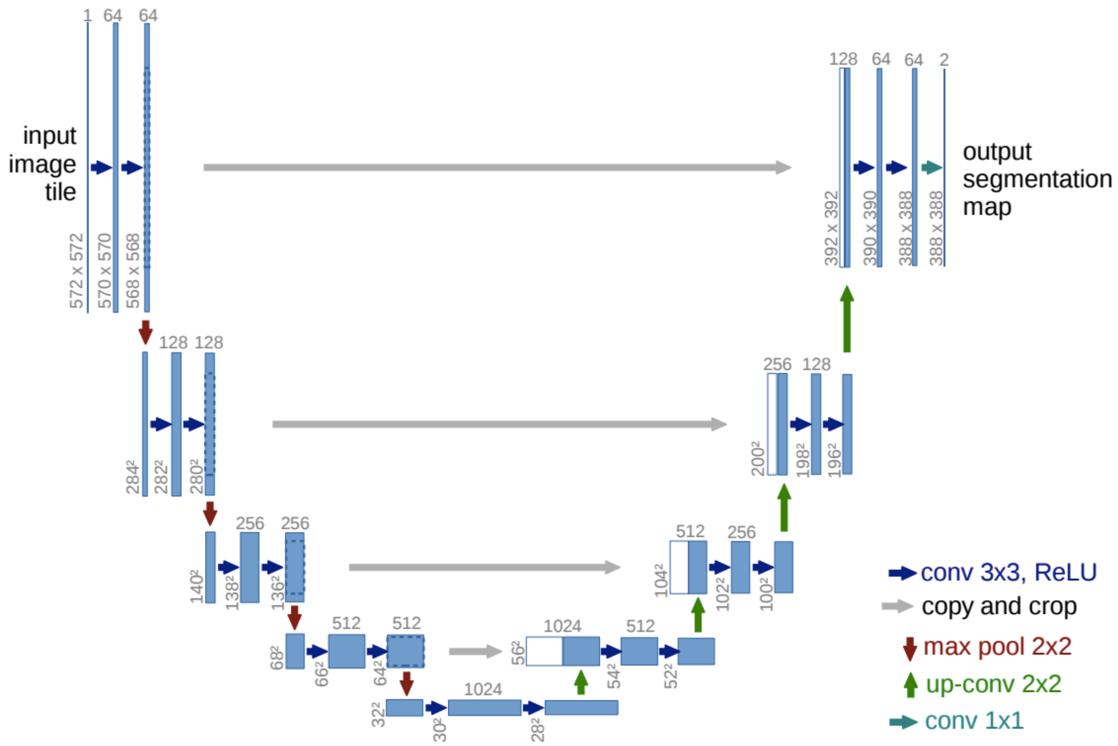
SegNet



Eliminando i FC layer, porta a risultati migliori

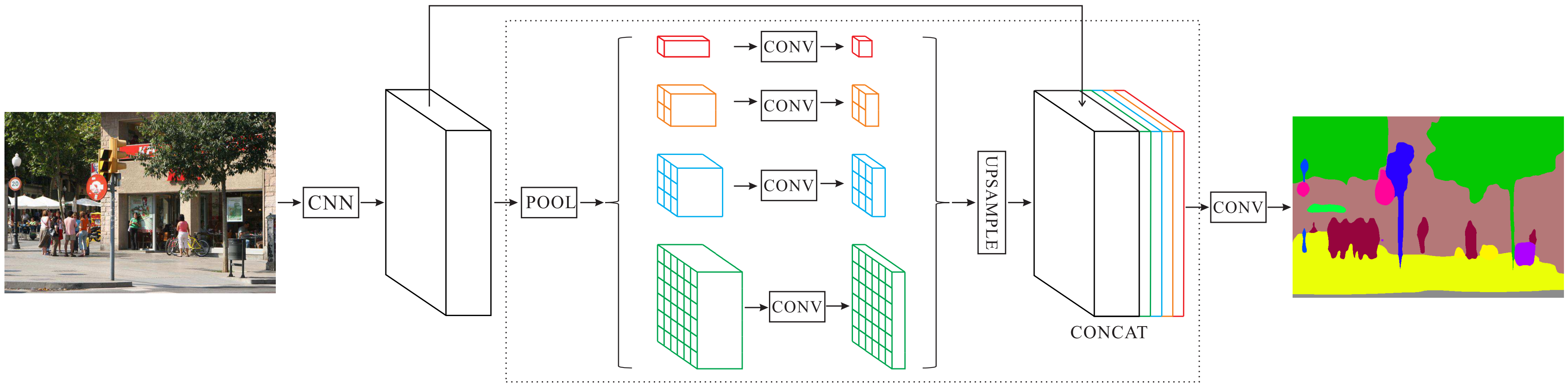
U-Net

- Usa le skip connections per combinare le feature maps
- La combinazione viene effettuata per concatenazione



Metodi Multi-scala

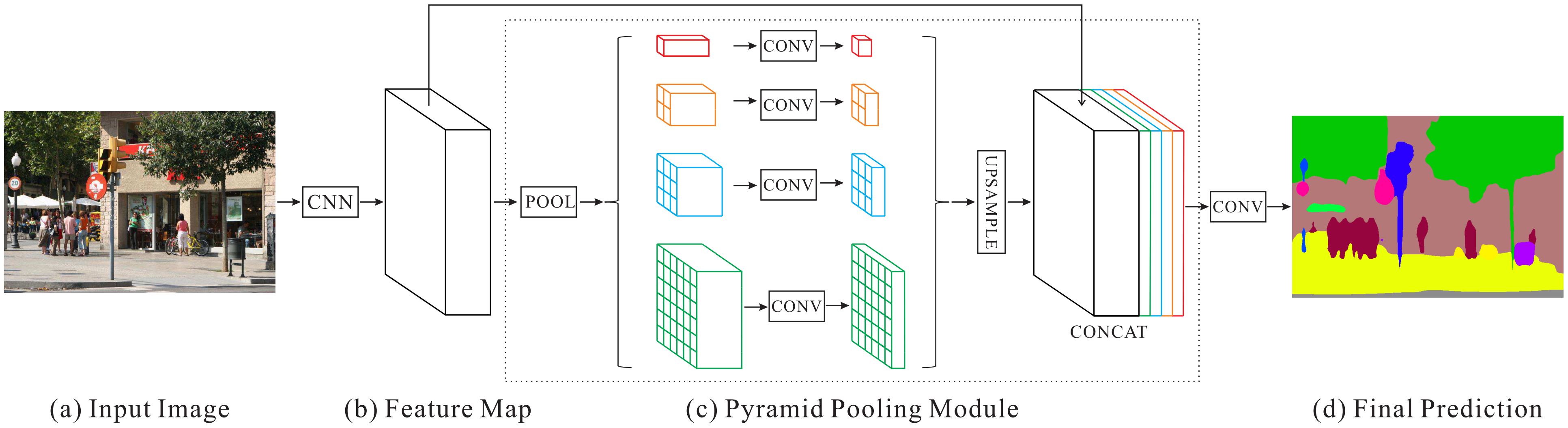
- Idea generale
  - Otteniamo una feature map utilizzando un’architettura standard (ResNet)
  - Applichiamo una serie di convoluzioni con filtri di dimensioni diverse per ottenere risoluzioni diverse
  - Encoding delle varie scale
  - Upsampling e combinazione dei risultati



(a) Input Image (b) Feature Map (c) Pyramid Pooling Module (d) Final Prediction

Metodi Multi-scala

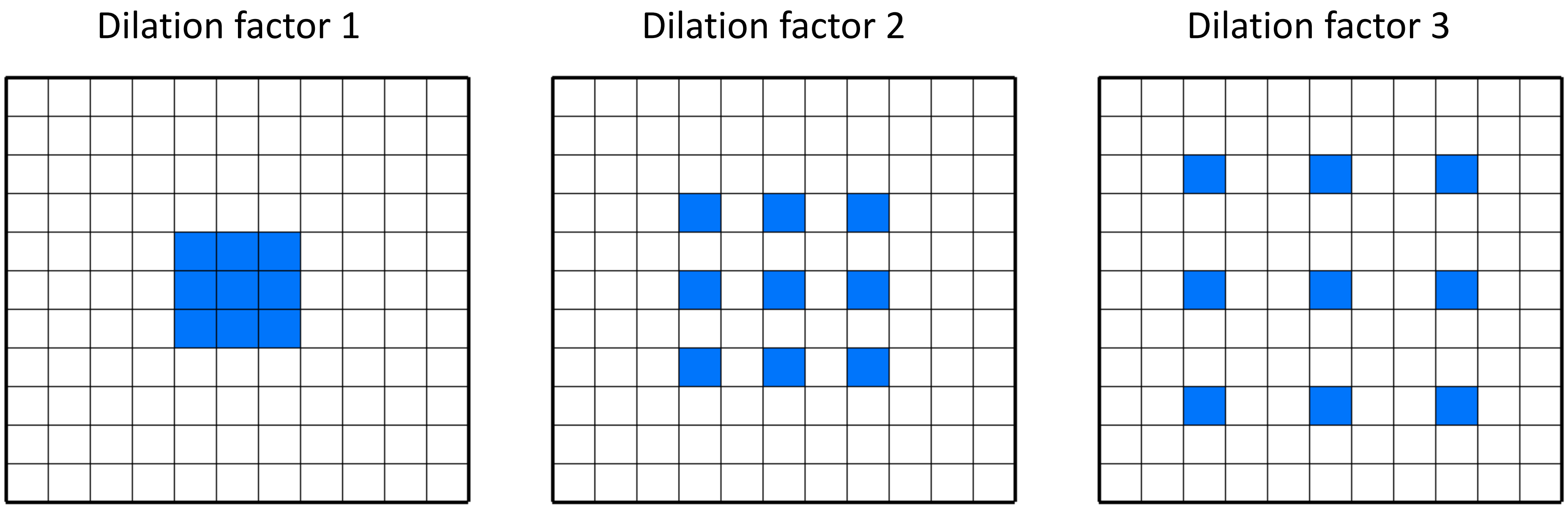
- Esplosione combinatoria del numero di parametri



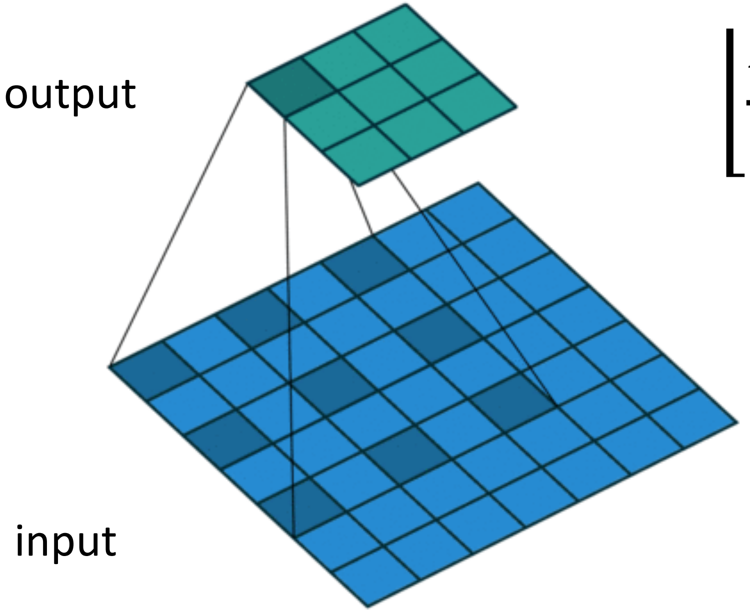
- Soluzione: Dilated convolutions

Dilated convolutions

- Invece di ridurre la risoluzione spaziale delle feature maps, utilizziamo un filtro sparso

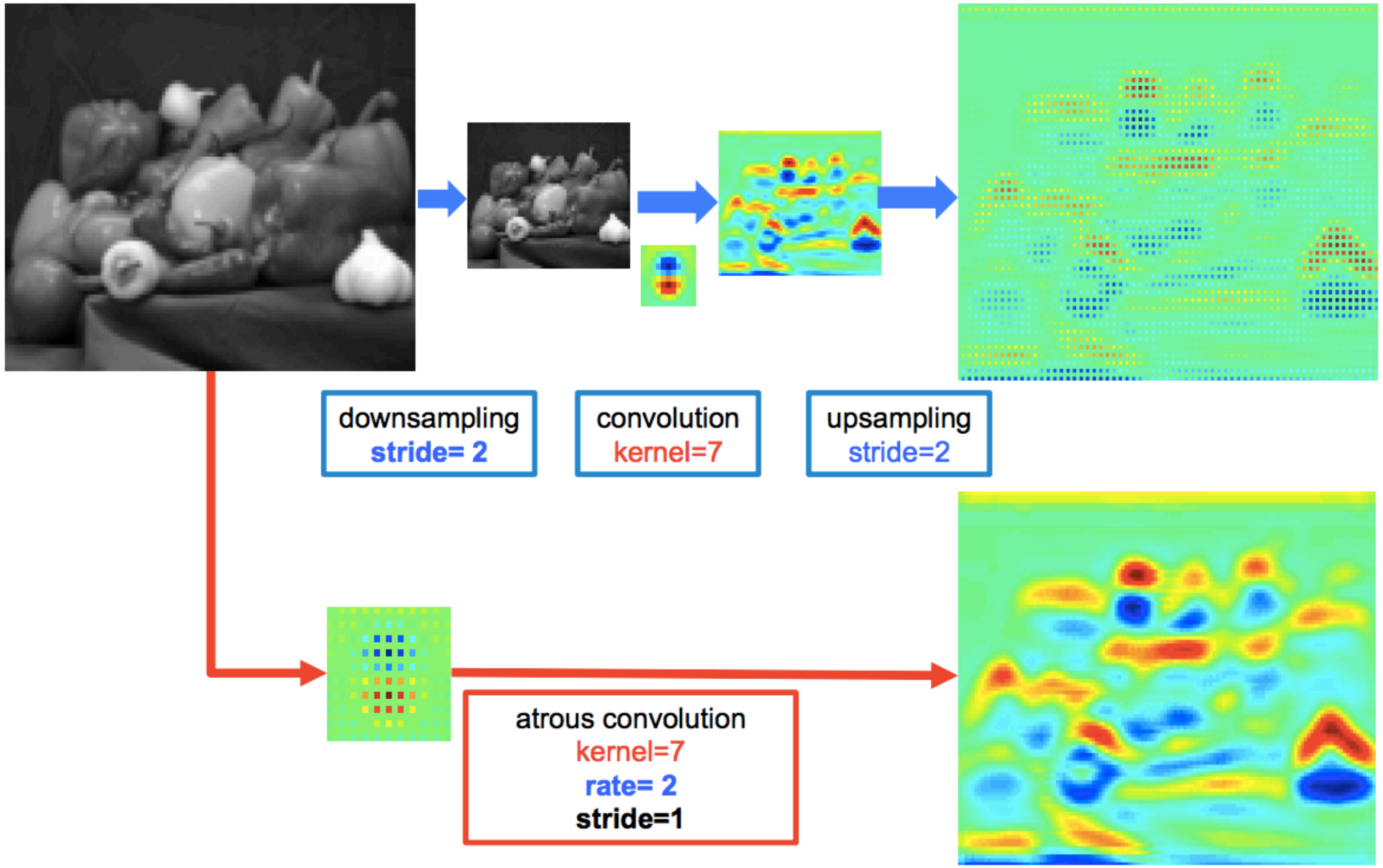


Dilated convolutions



$$\left\lfloor \frac{I - K - (K-1)(D-1) + 2P}{S} \right\rfloor + 1$$

Dilated convolutions



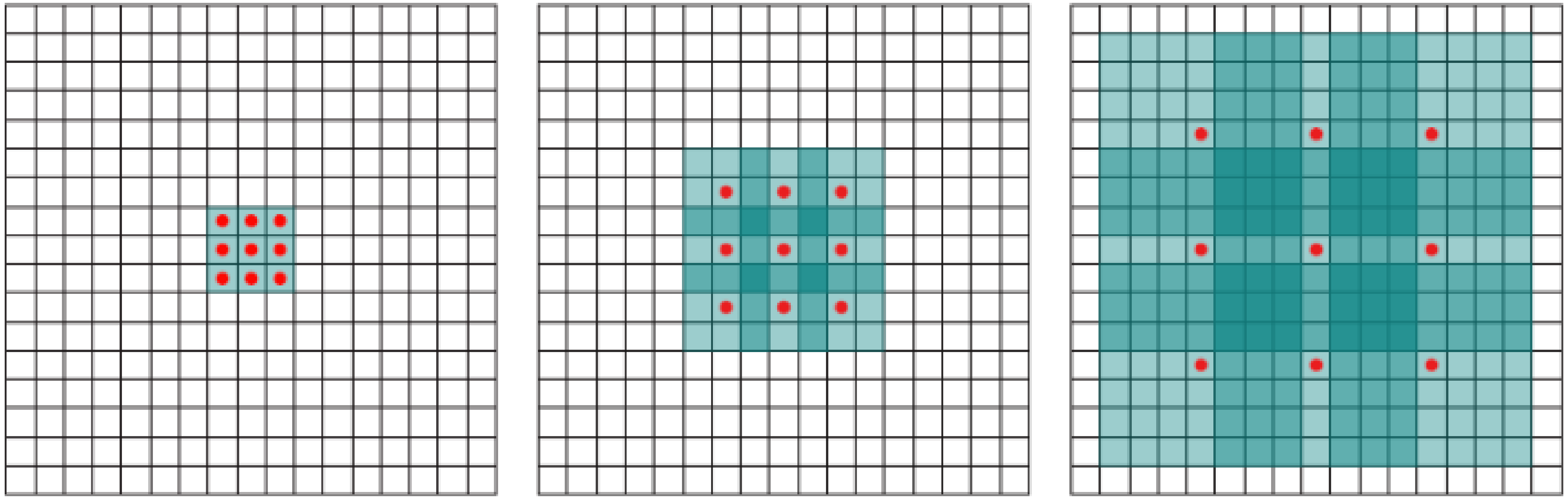
Dilated convolutions

- La dimensione del receptive field cresce esponenzialmente ma il numero di parametri è lineare

F1 calcolata da F0 con 1-dilated convolution

F2 calcolata da F1 con 2-dilated convolution

F3 calcolata da F2 4-dilated convolution

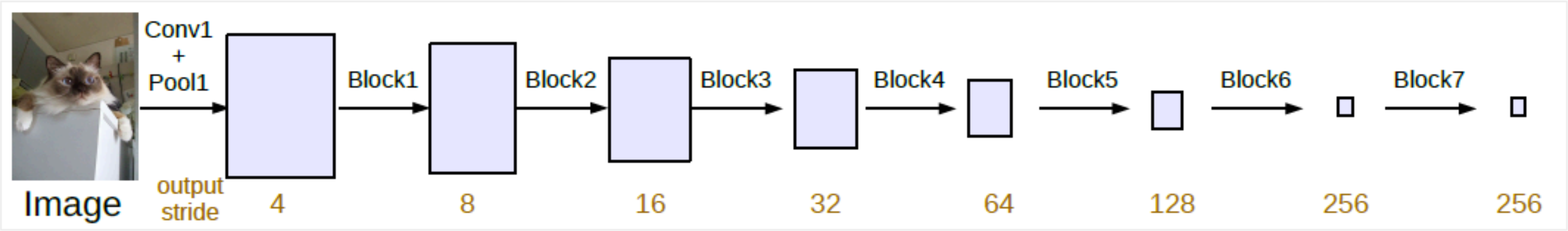


Receptive field: 3x3

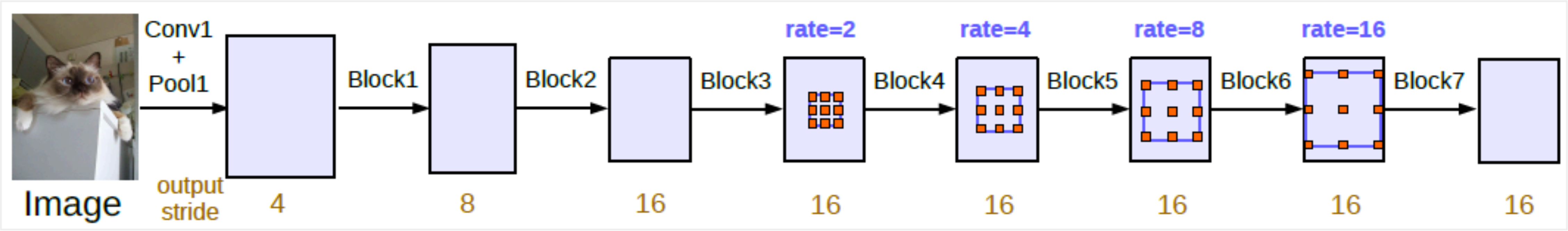
Receptive field: 7x7

Receptive field: 15x15

Vantaggi

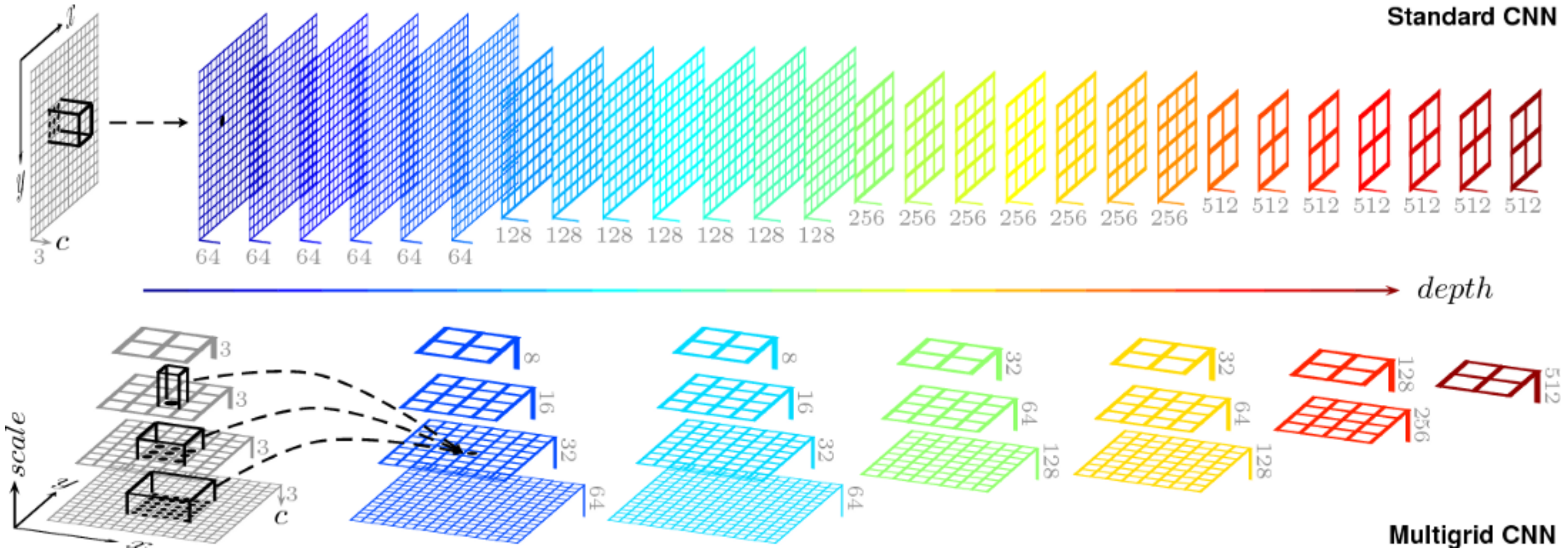


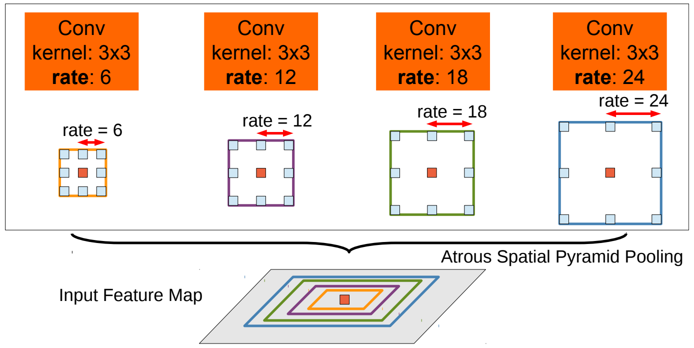
(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

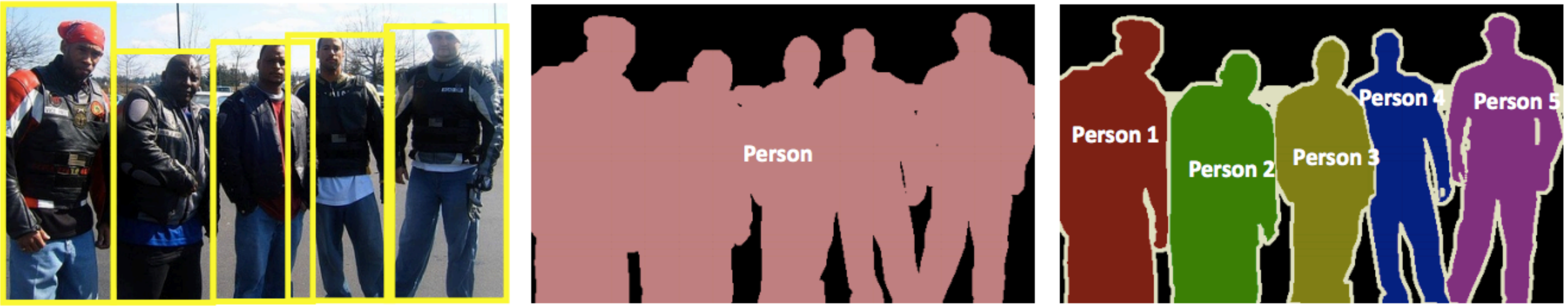
Multigrid CNN





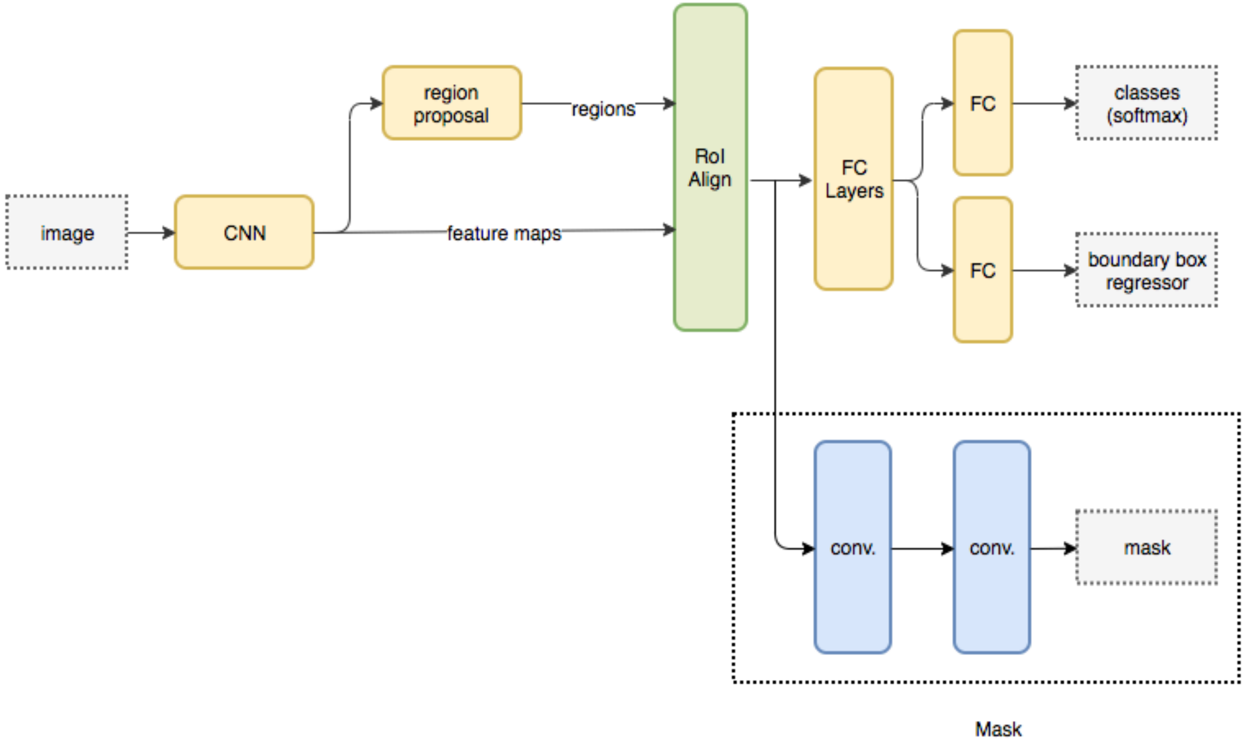
Instance Segmentation

- Obiettivo
  - Individuare non solo la segmentazione, ma anche l’istanza



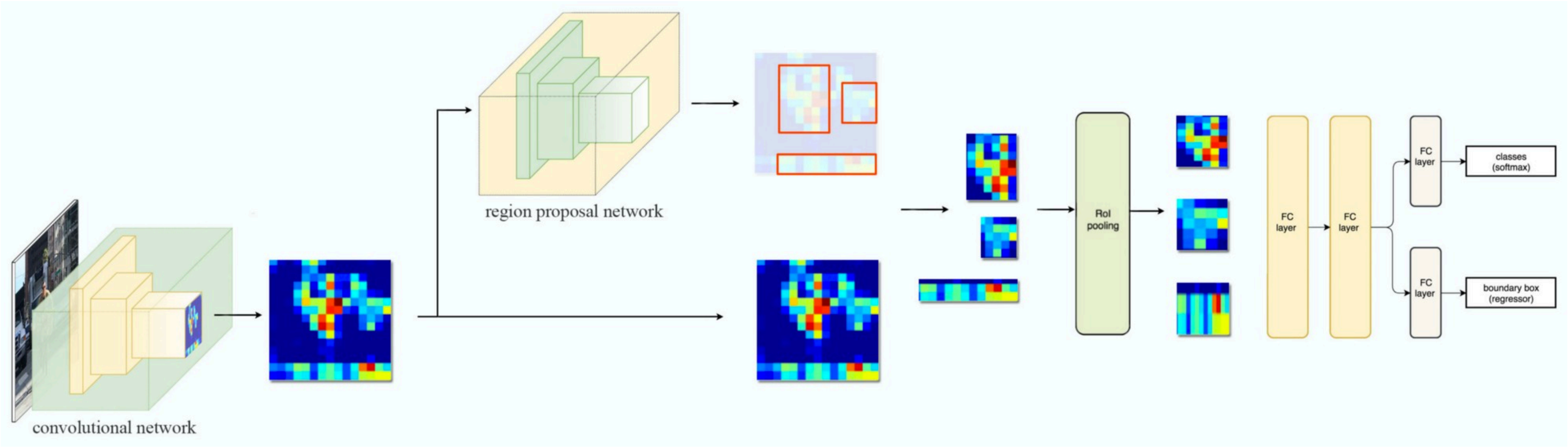
Mask R-CNN

- Mask R-CNN = Faster R-CNN + FCN sui RoIs

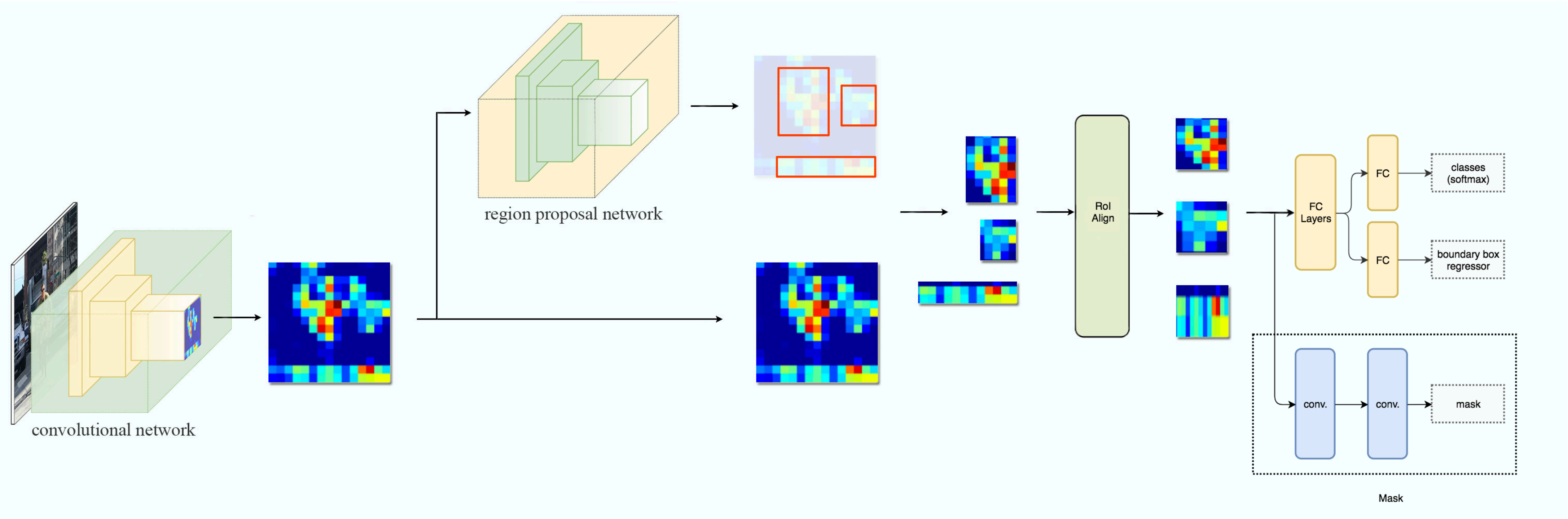


Loss: $L_{cs} + L_{box} + L_{mask}$

Recap: Faster R-CNN

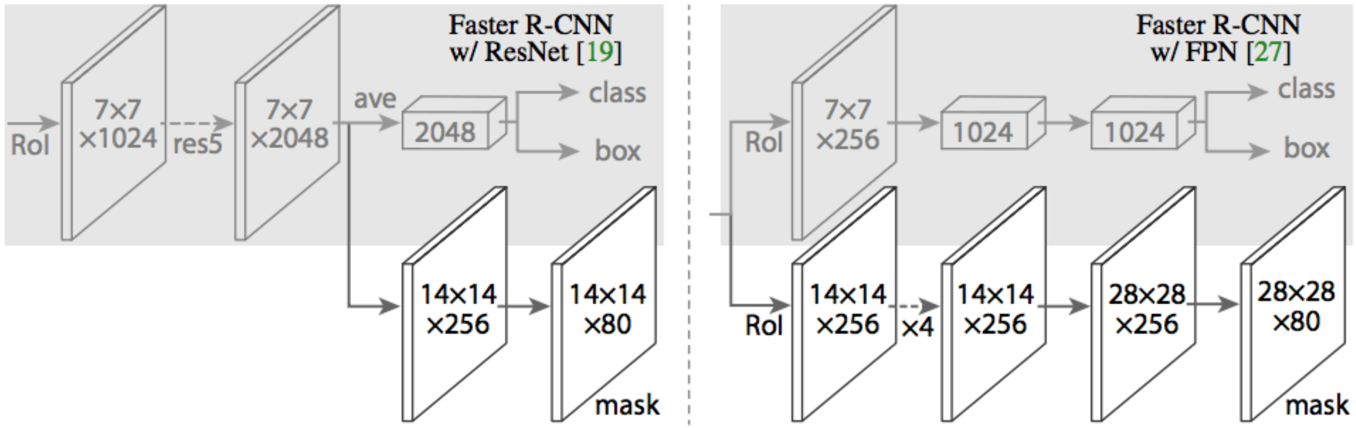


Da Faster R-CNN a Mask R-CNN



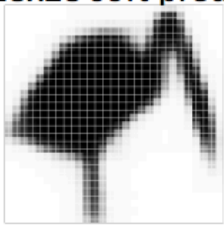
Mask R-CNN: Mask

- $K \cdot m{\times}m$
  - Una maschera di dimensione $m{\times}m$ per ognuna delle K classi
  - Ogni pixel è regolato da una sigmoide
  - Loss
    - Su una RoI associata alla classe k, L_{mask}è la binary cross-entropy relativa alla maschera m_k associata
      - Le altre maschere non contribuiscono alla loss



Mask R-CNN

28x28 soft prediction

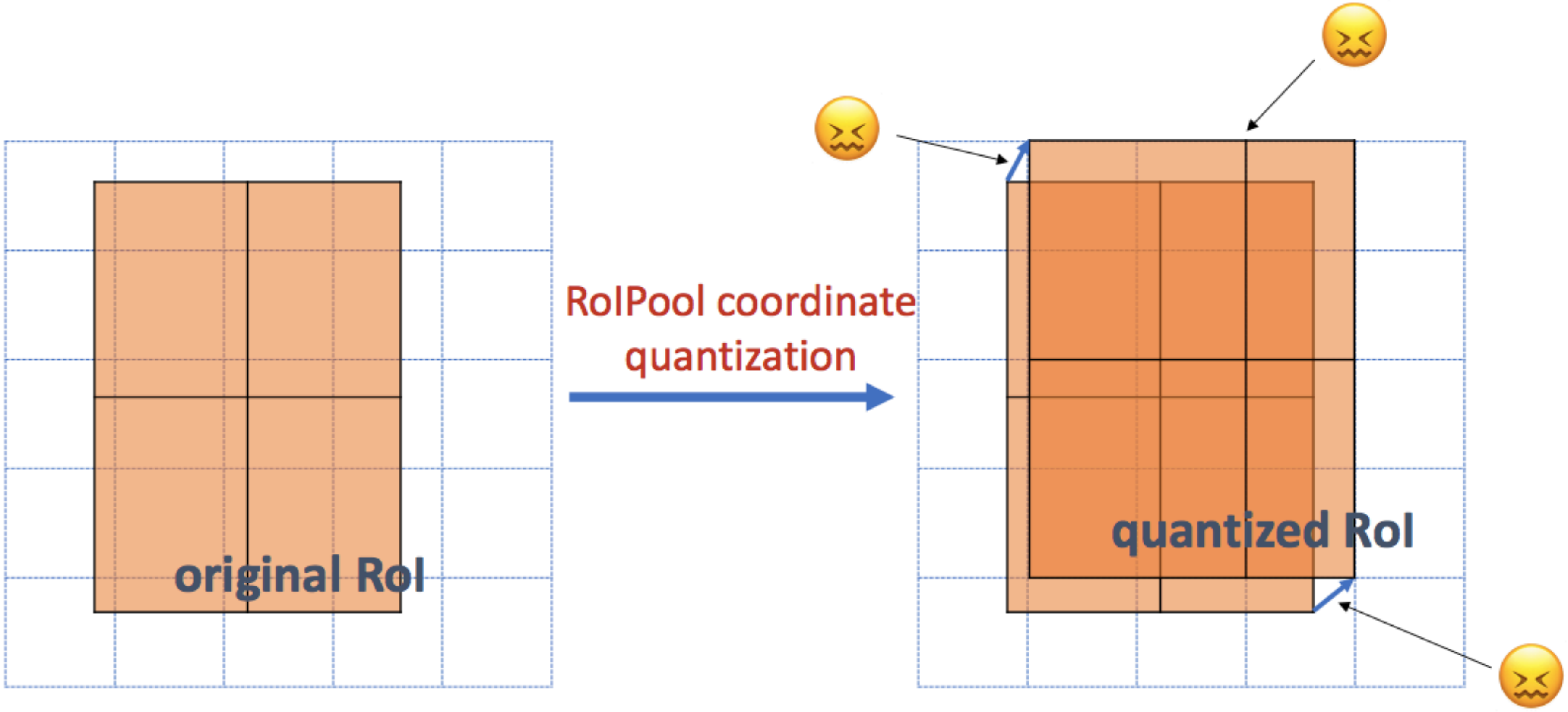


Resized Soft prediction

Final mask

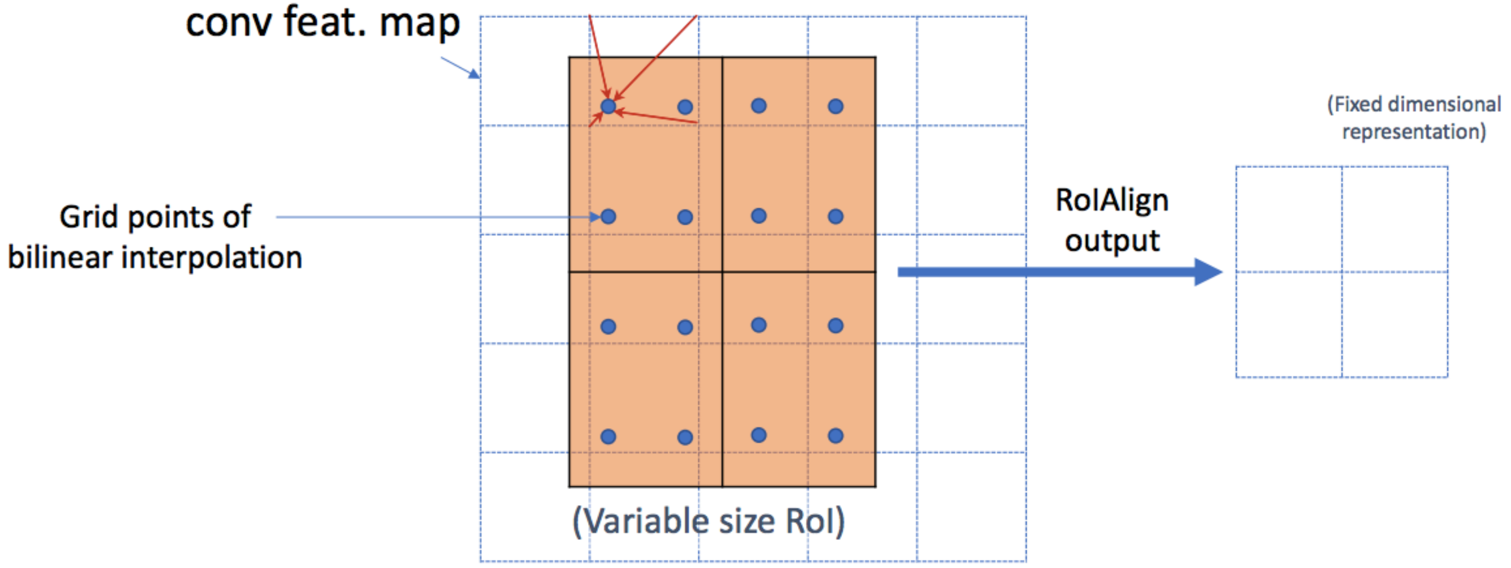
RoIAlign

- Il mapping di una regione sulla feature map con RoIPooling causa un riallineamento



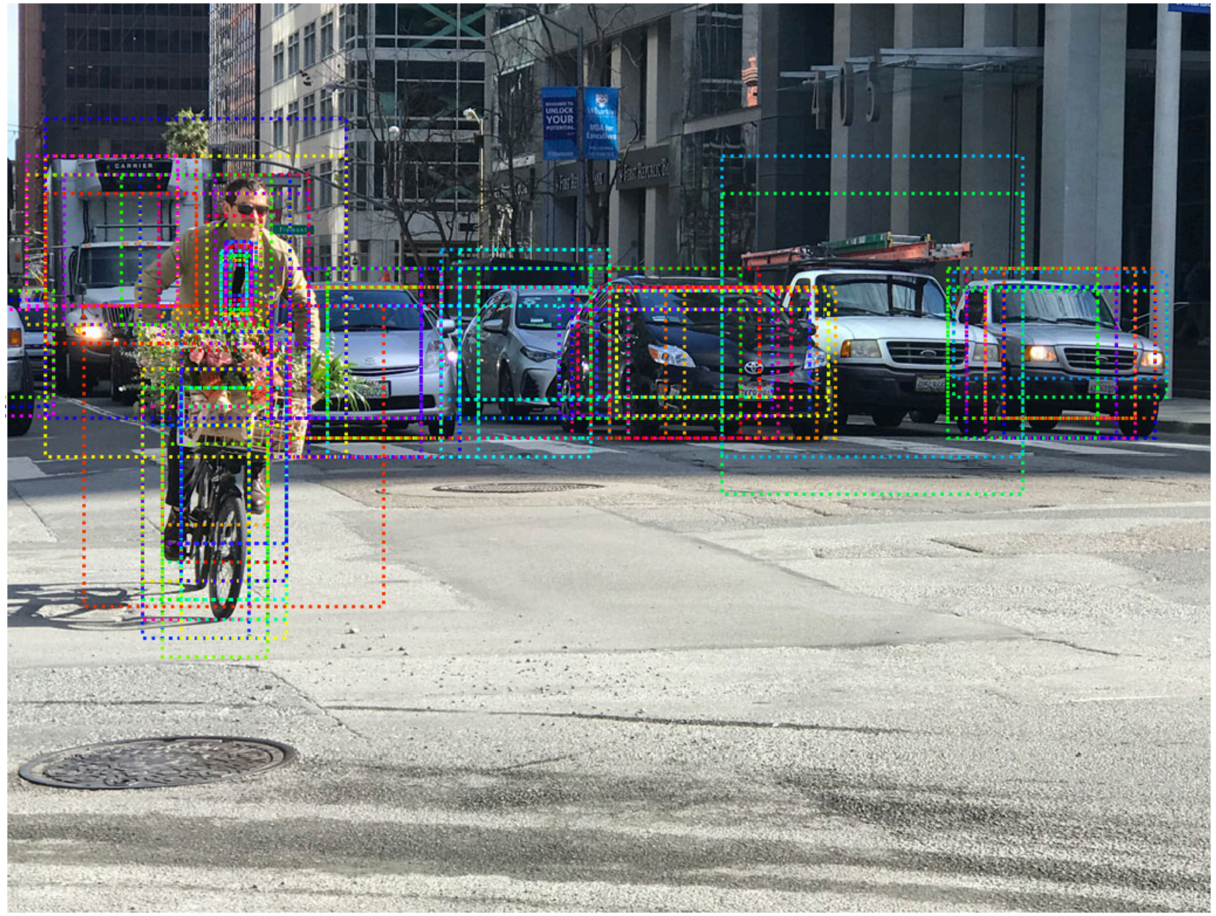
RoIAlign

- Con RoIAlign, ogni punto viene interpolato
  - Recupera precision nella ricostruzione della maschera



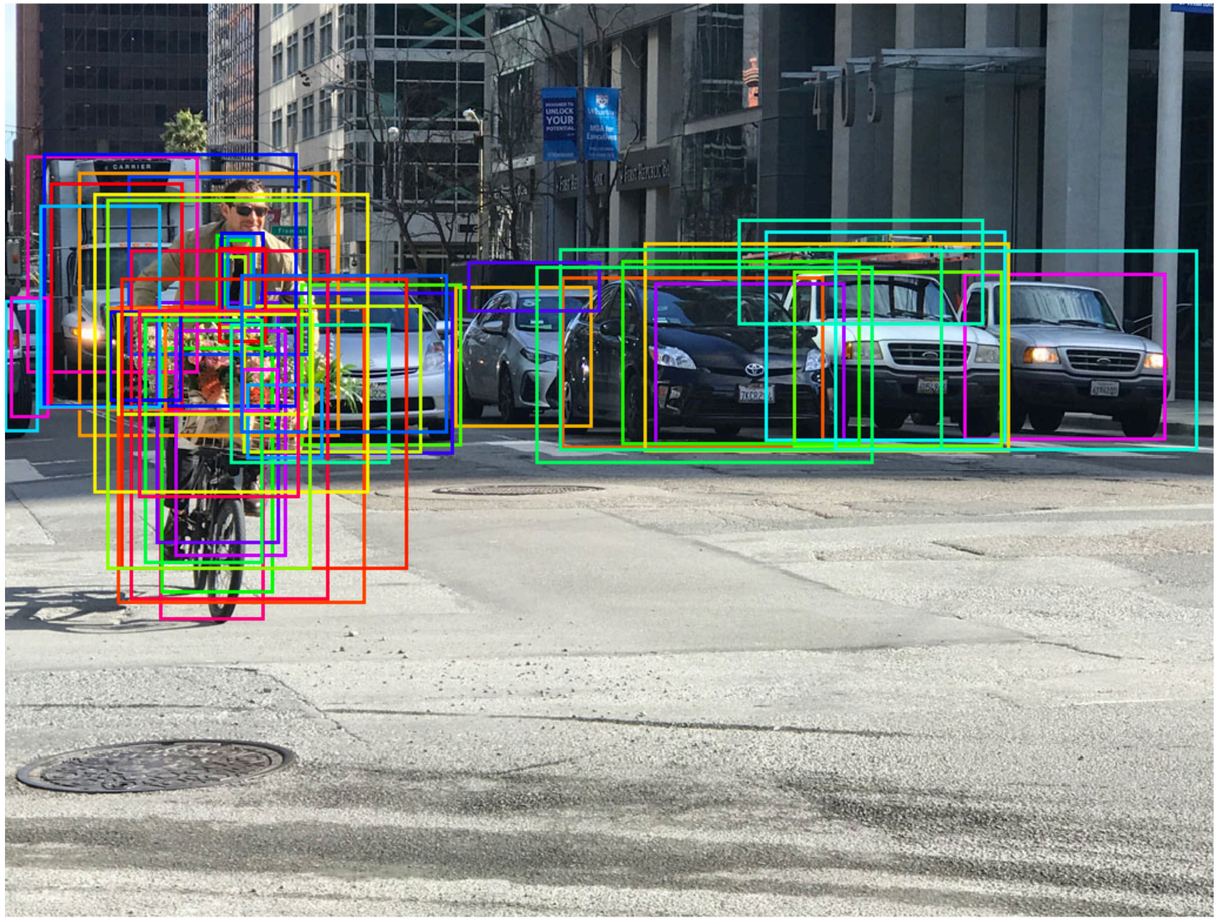
Risultati

- Priors



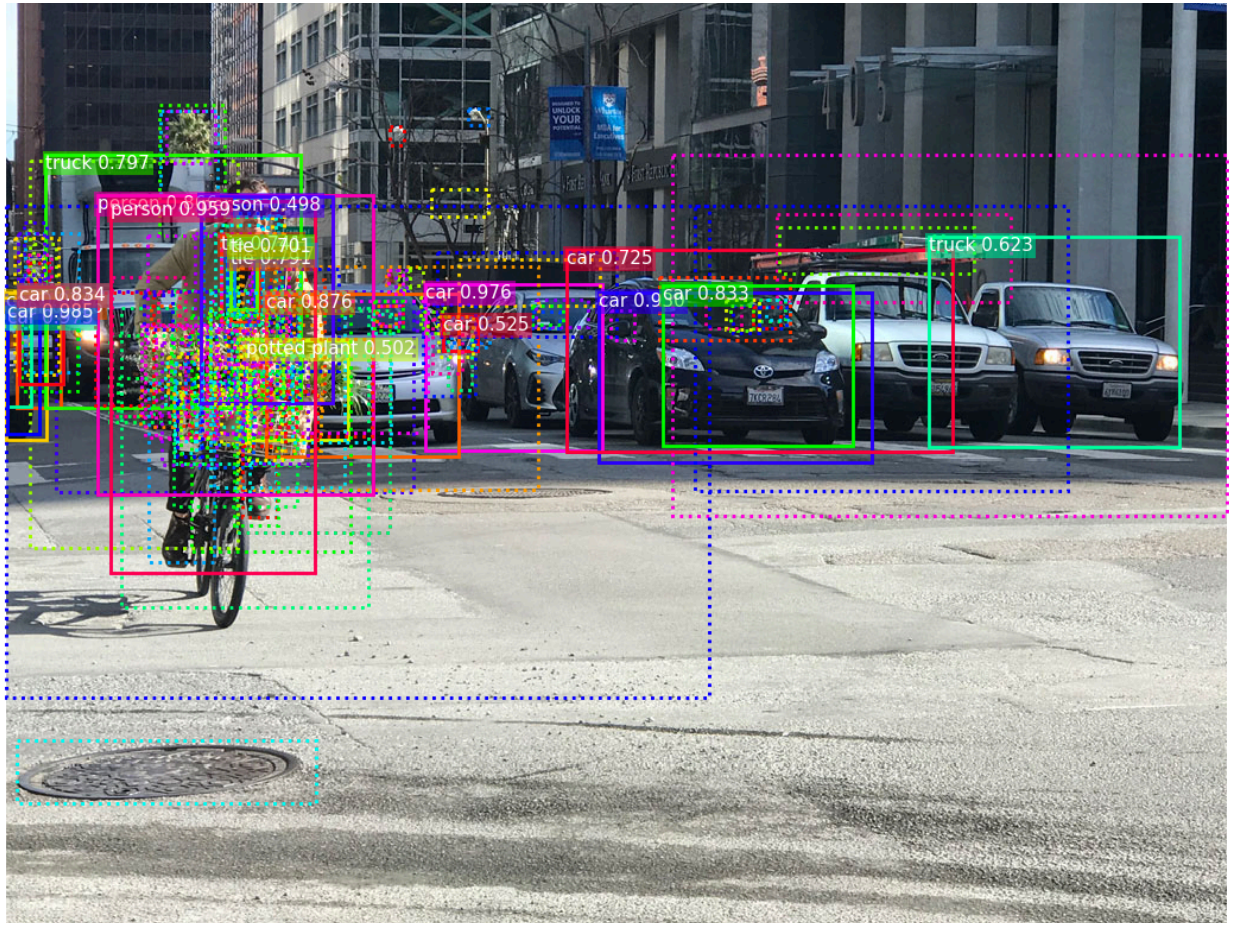
Risultati

- Region Proposals



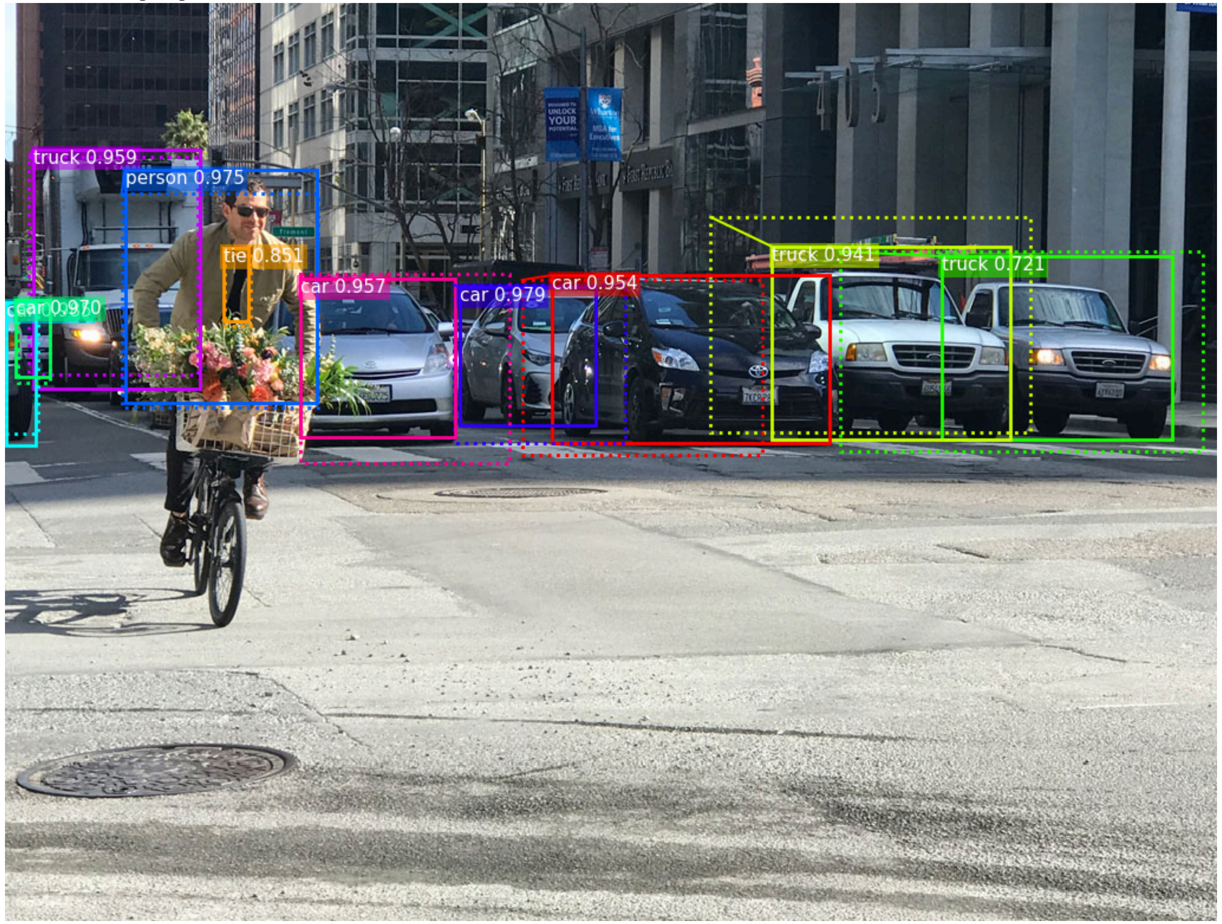
Risultati

- Predizione



Risultati

- Non-Maximum Suppression

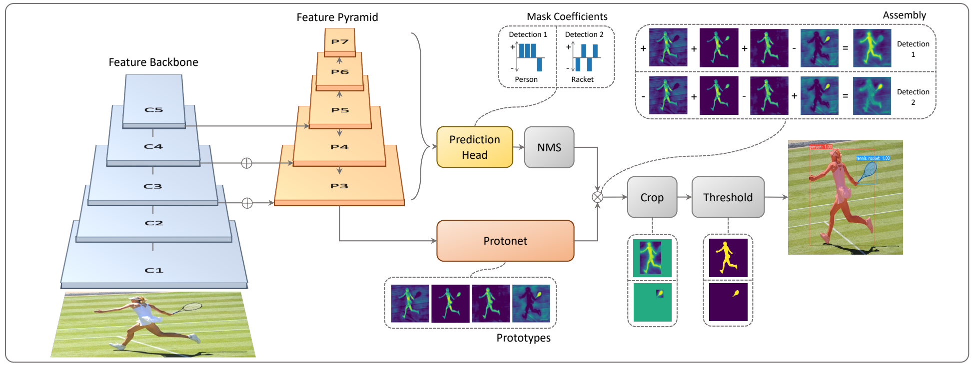


Risultati

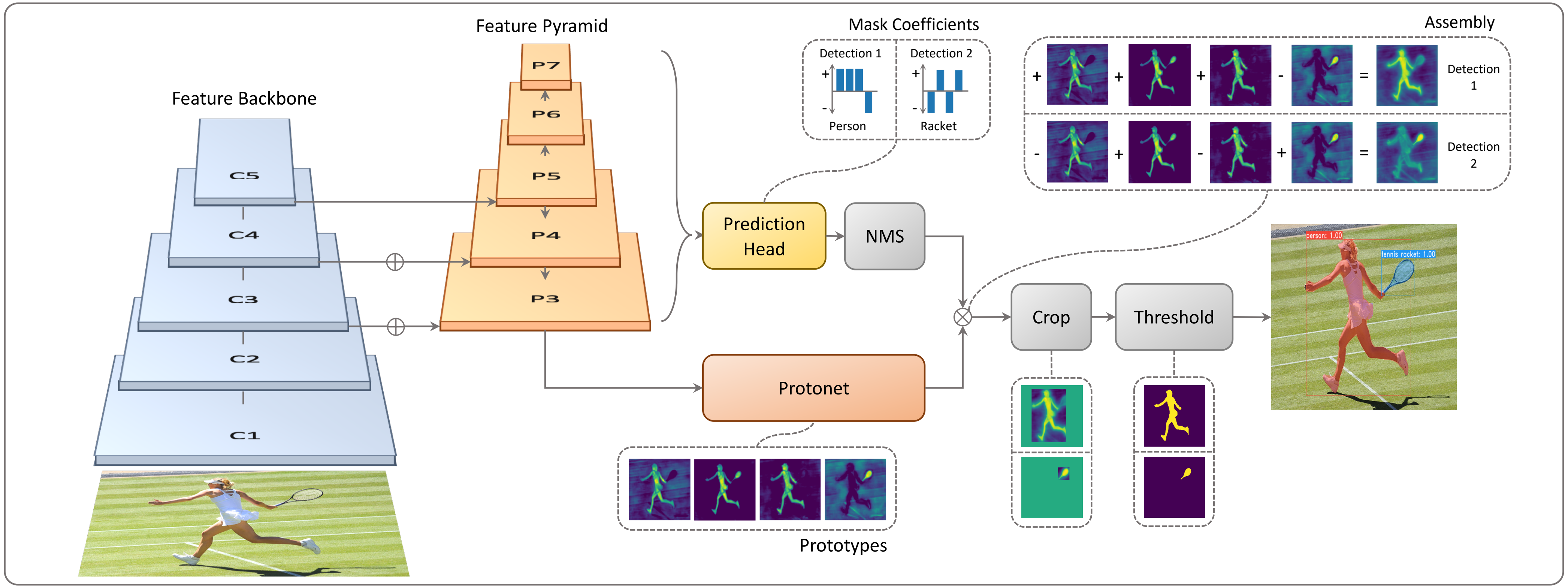
- Mask

YOLOACT: You Only Look At CoefficienTs

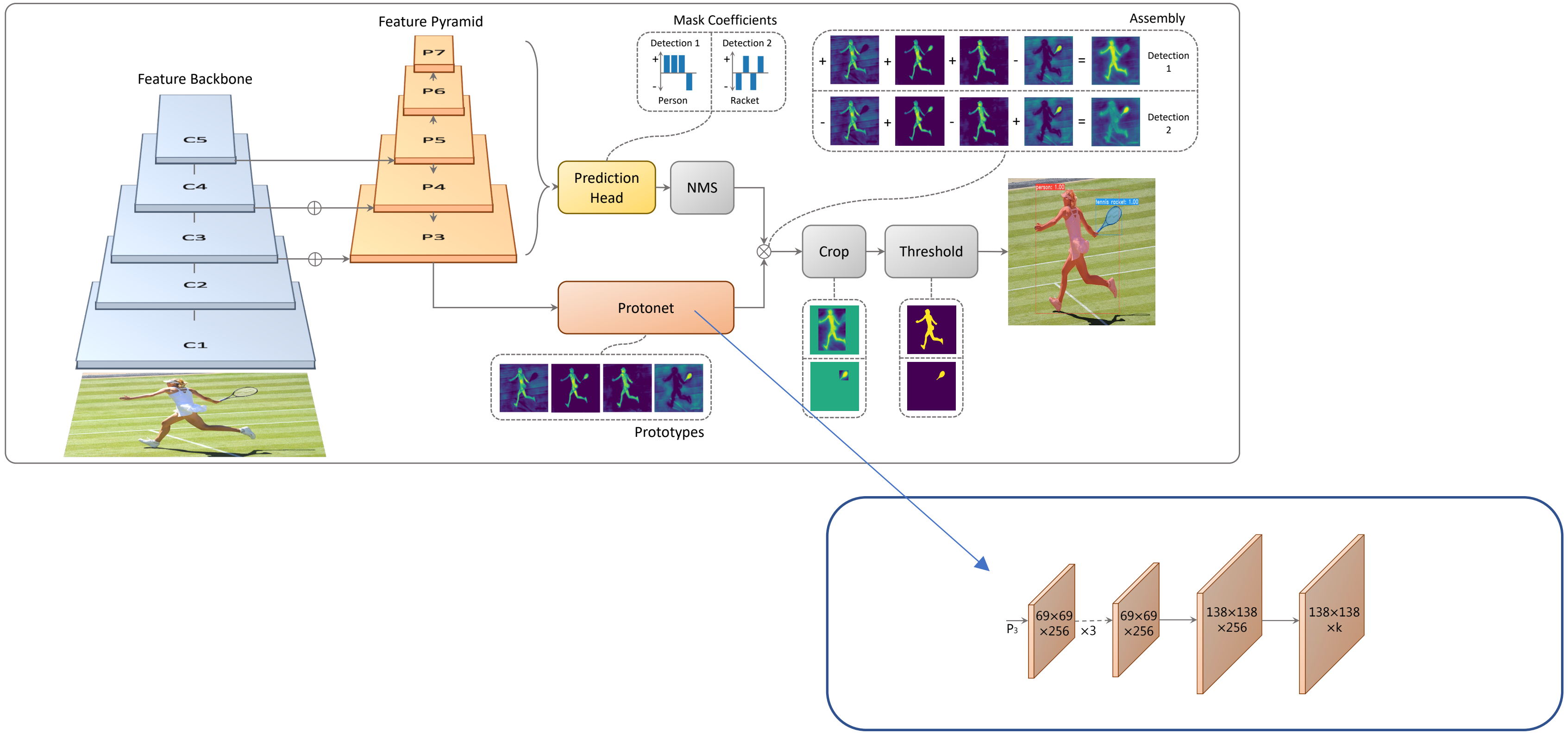
- Due task paralleli:
  - Generazione di un dizionario di non-local prototype masks sull'intera immagine
    - Basato su FCN
  - Predizione di un insieme di coefficienti di combinazione per ogni istanza
    - Aggiunge una componente all'object detection per predire un vettore di "mask coefficients"
  - Per ogni istanza selezionata nel NMS viene costruita una maschera combinando i risultati dei due task.



Architettura

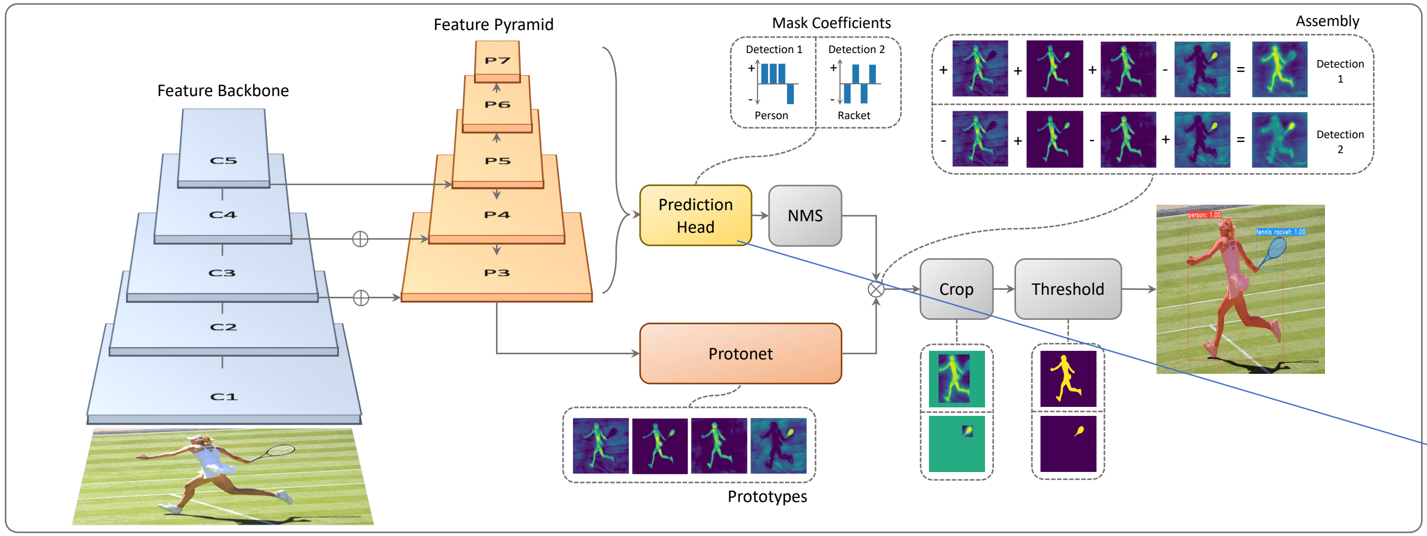


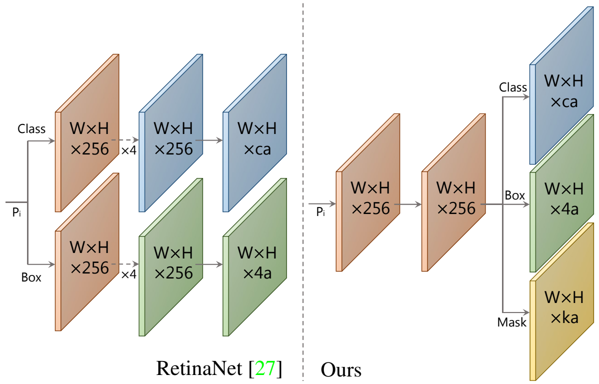
Architettura



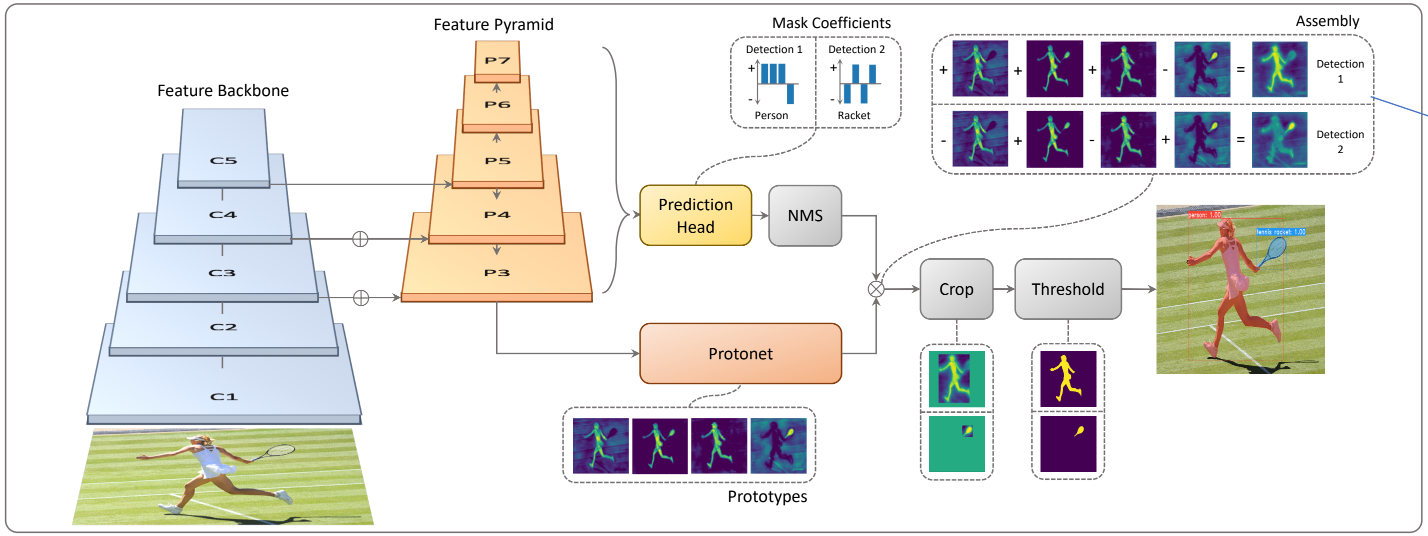
Prototype FCN

Architettura





Architettura

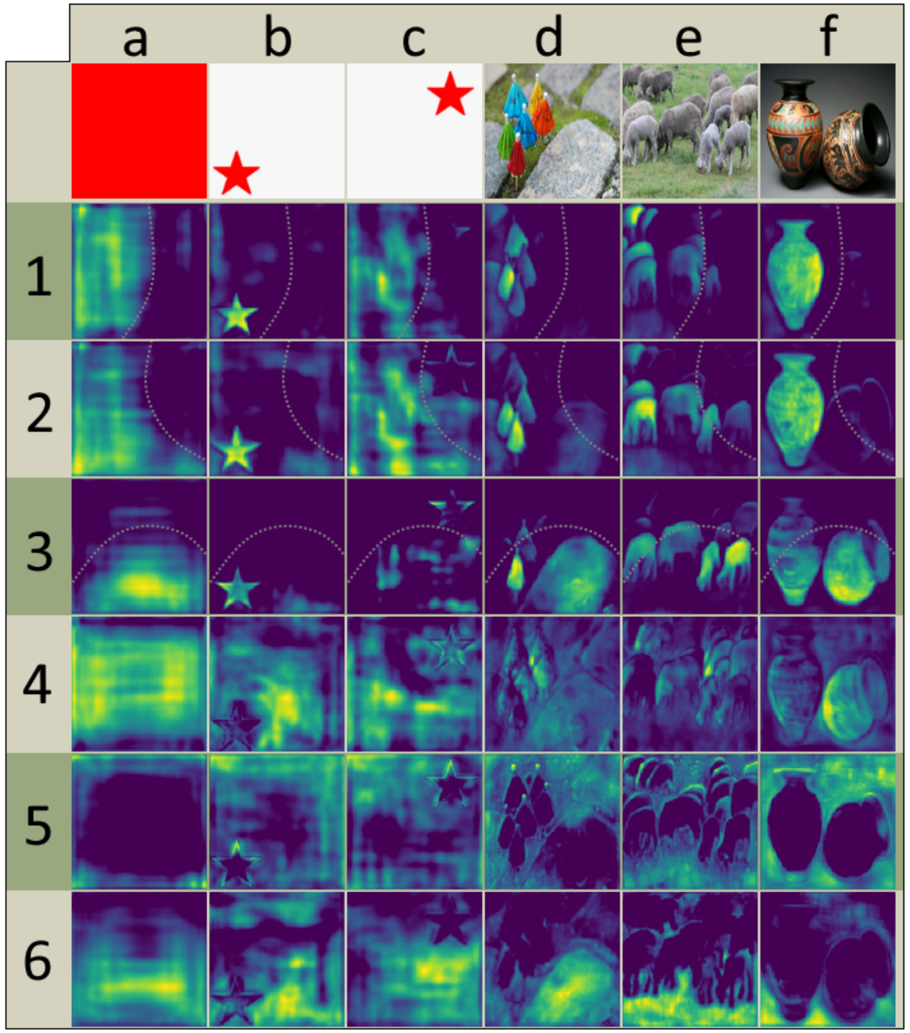


$$M = \sigma(P \cdot C^T)$$

- P matrice di maschere
- C vettore di coefficienti

Caratteristiche

- Loss
  - $L_{cs} + L_{box} + L_{mask}$
  - $L_{mask} = BCE(M, M_{gt})$
- YOLACT impara a localizzare le istanze



Risultati

- 29.8mAP, 33FPS

Riassunto

- Semantic vs. Instance segmentation
- Architetture complesse
- Base per learning task simili
  - Depth estimation
  - Surface normal estimation
  - Colorization