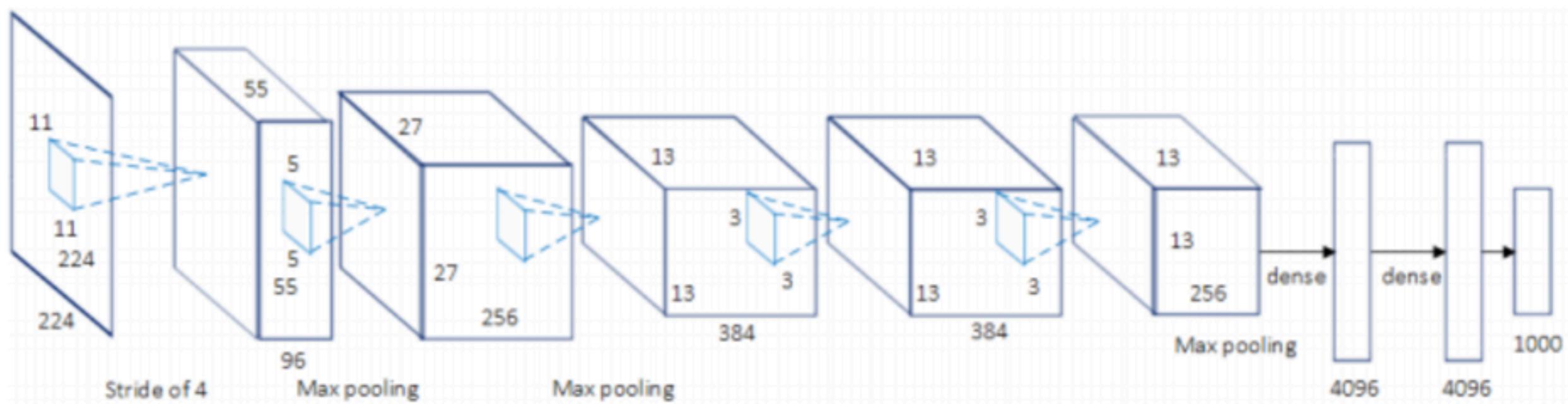




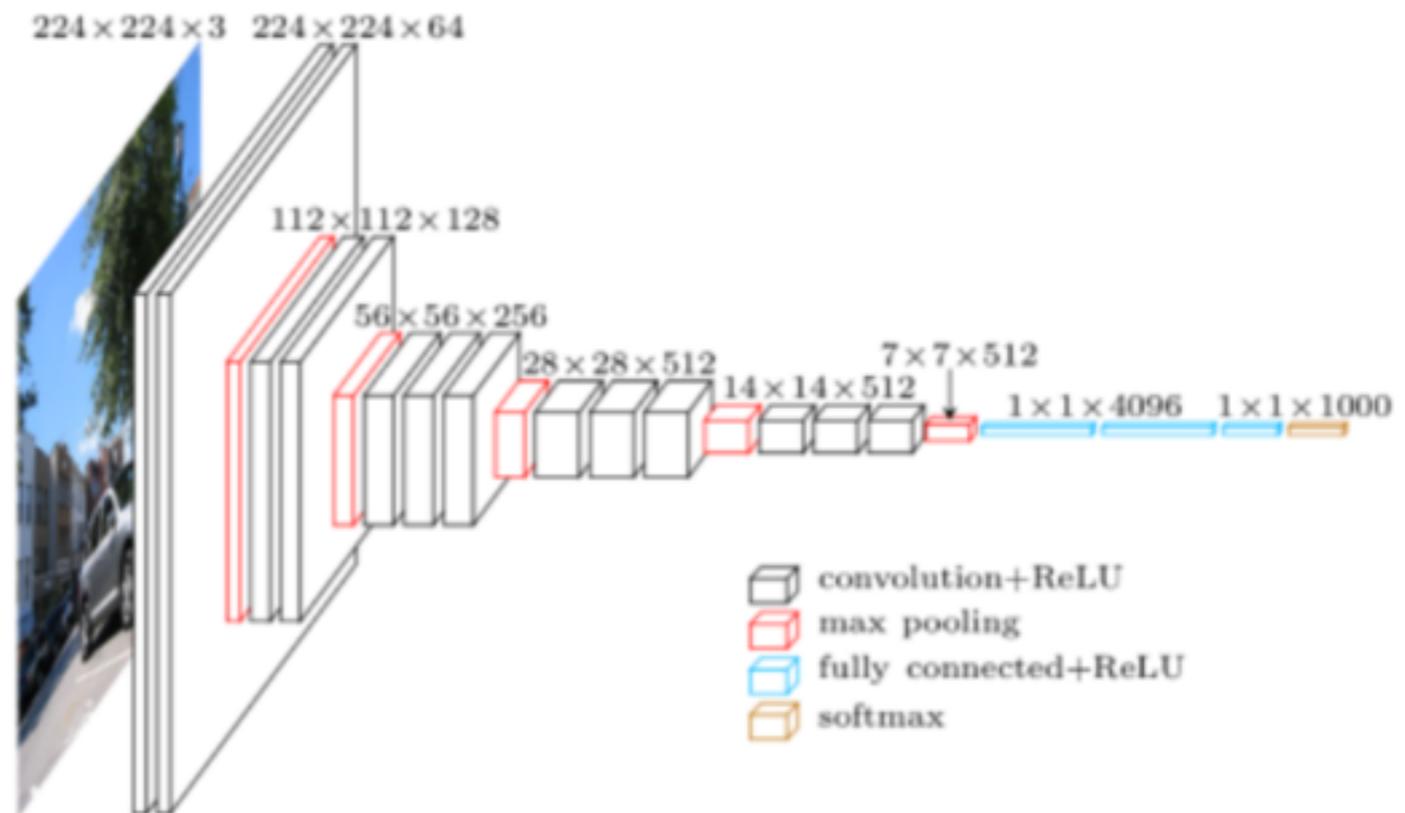
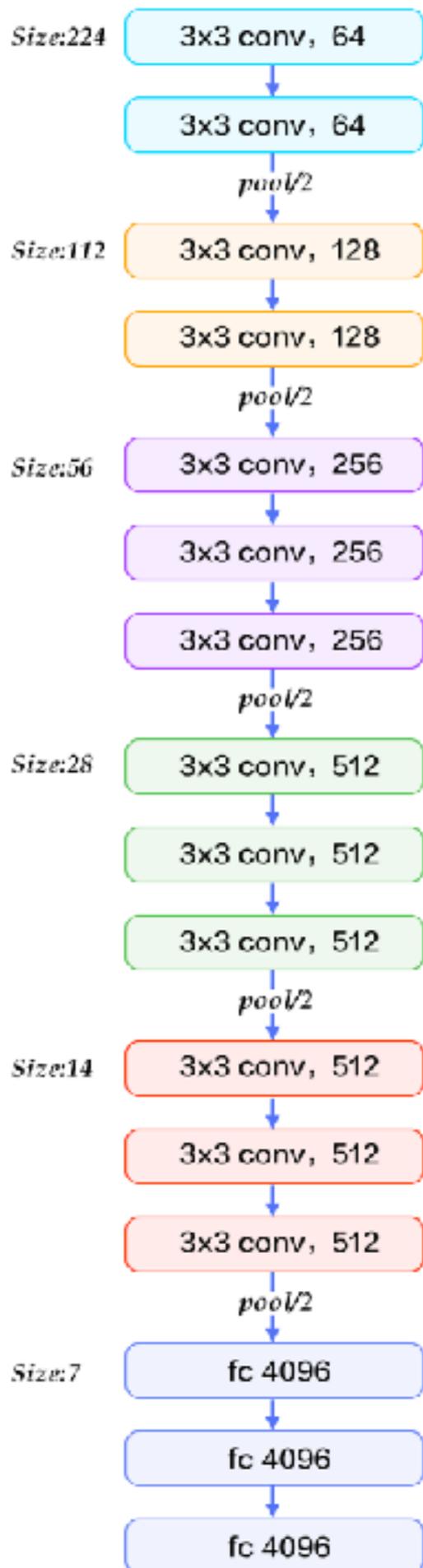
Deep Learning for Retail

# Basic CNN Architecture Review

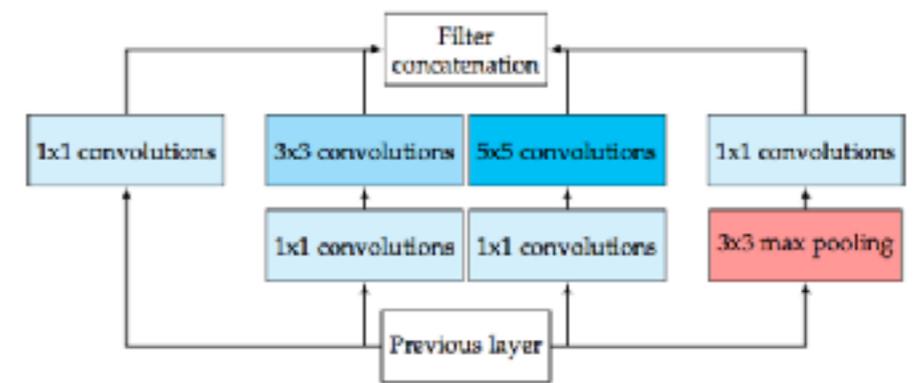
# AlexNet



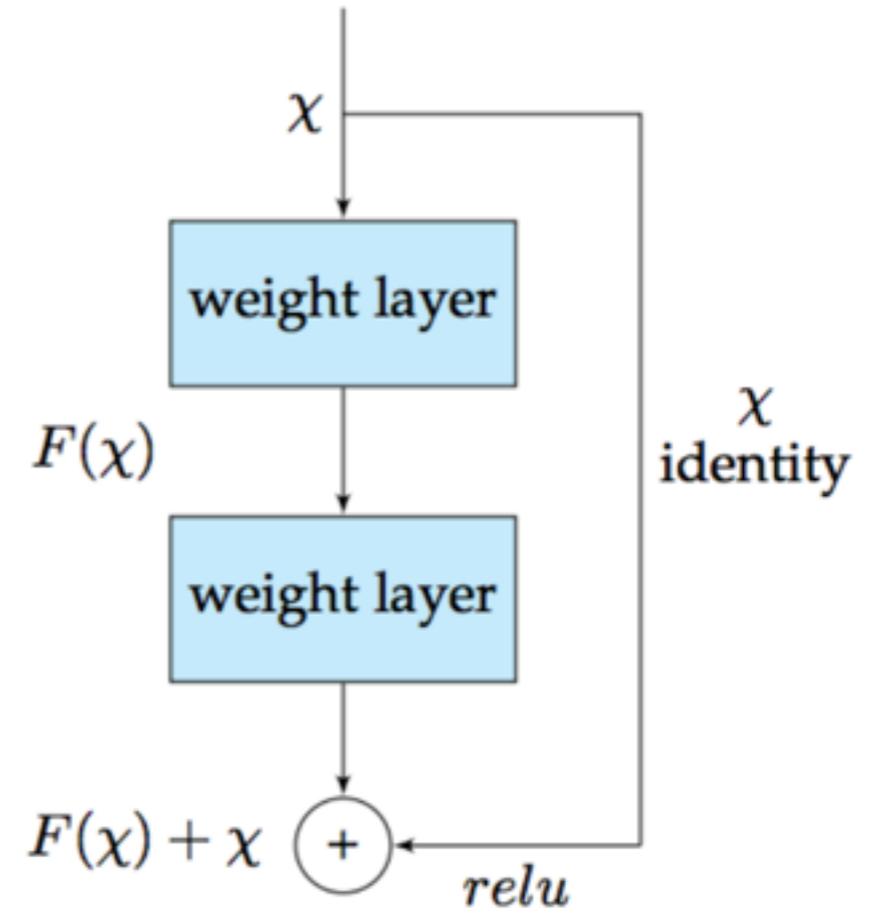
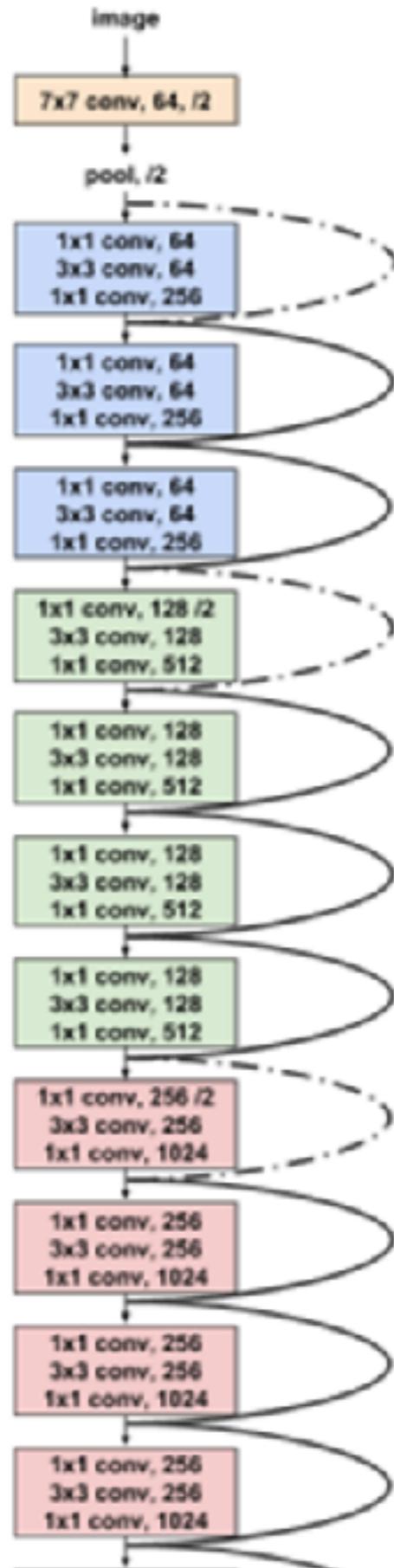
# VGG



# Inception

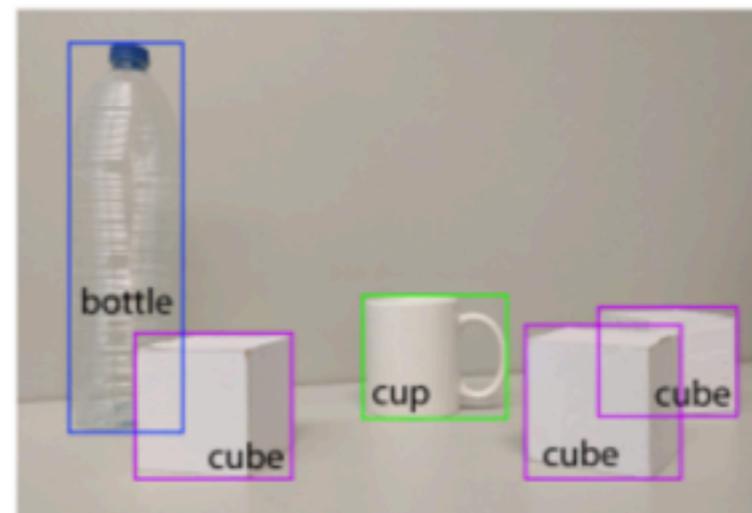


# ResNet

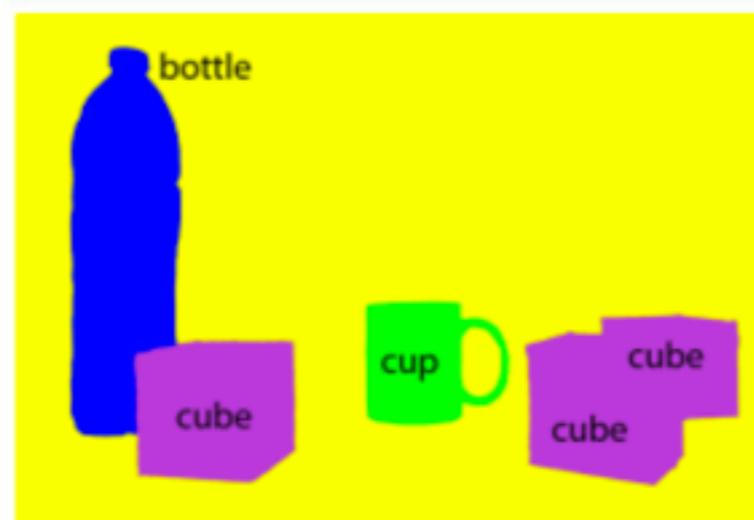




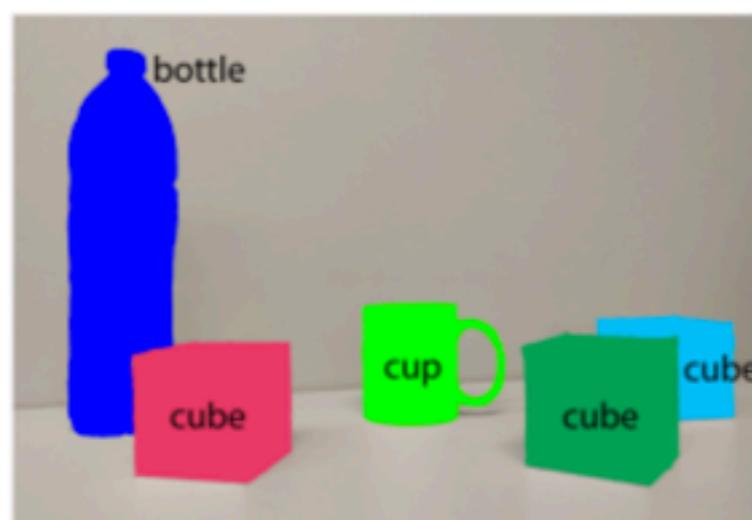
(a) Image classification



(b) Object localization



(c) Semantic segmentation

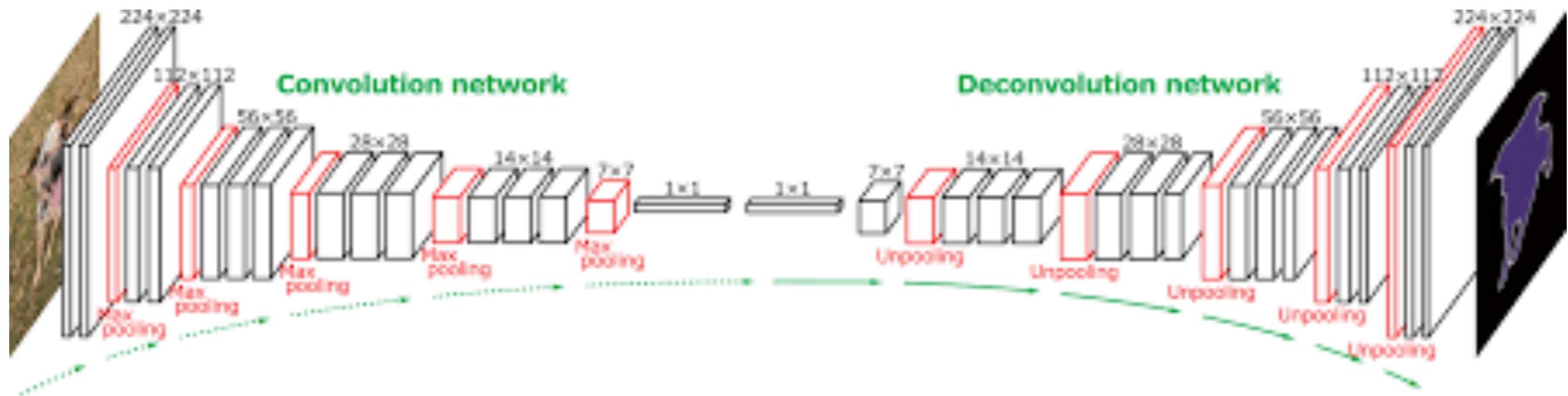


(d) Instance segmentation

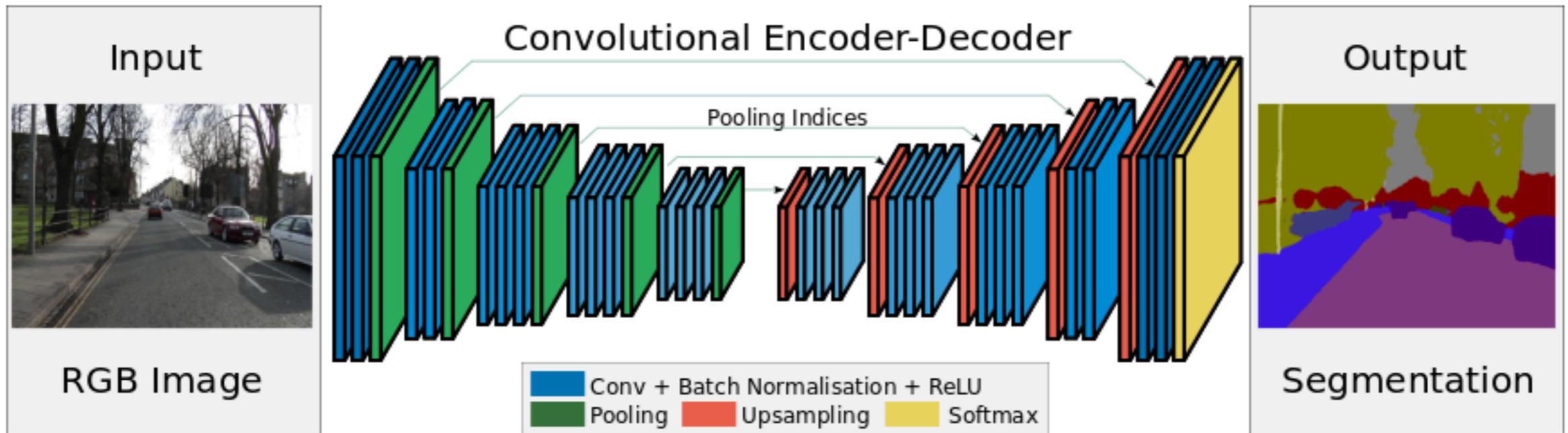
# Segmentation

Name and Reference	Purpose	Year	Classes	Data	Resolution	Sequence	Synthetic/Real	Samples (training)	Samples (validation)	Samples (test)
PASCAL VOC 2012 Segmentation [27]	Generic	2012	21	2D	Variable	✗	R	1464	1449	Private
PASCAL-Context [28]	Generic	2014	540 (59)	2D	Variable	✗	R	10103	N/A	9637
PASCAL-Part [29]	Generic-Part	2014	20	2D	Variable	✗	R	10103	N/A	9637
S3D [30]	Generic	2011	21	2D	Variable	✗	R	8498	2857	N/A
Microsoft COCO [31]	Generic	2014	+80	2D	Variable	✗	R	82783	40504	81434
SYNTIA [32]	Urban (Driving)	2016	11	2D	960 × 720	✗	S	13407	N/A	N/A
Cityscapes (fine) [33]	Urban	2015	30 (8)	2D	2048 × 1024	✓	R	2975	500	1525
Cityscapes (coarse) [33]	Urban	2015	30 (8)	2D	2048 × 1024	✓	R	22973	500	N/A
CamVid [34]	Urban (Driving)	2009	32	2D	960 × 720	✓	R	701	N/A	N/A
CamVid-Sturgess [35]	Urban (Driving)	2009	11	2D	960 × 720	✓	R	367	100	233
KITTI-Layout [36] [37]	Urban/Driving	2012	3	2D	Variable	✗	R	323	N/A	N/A
KITTI-Ros [38]	Urban/Driving	2015	11	2D	Variable	✗	R	170	N/A	46
KITTI-Zhang [39]	Urban/Driving	2015	10	2D/3D	1226 × 370	✗	R	140	N/A	112
Stanford background [40]	Outdoor	2009	8	2D	320 × 240	✗	R	725	N/A	N/A
SiftFlow [41]	Outdoor	2011	33	2D	256 × 256	✗	R	2688	N/A	N/A
Youtube-Objects-Jain [42]	Objects	2014	10	2D	480 × 360	✓	R	10167	N/A	N/A
Adobe's Portrait Segmentation [26]	Portrait	2016	2	2D	600 × 800	✗	R	1500	300	N/A
MINC [43]	Materials	2015	23	2D	Variable	✗	R	7061	2500	5000
DAVIS [44] [45]	Generic	2016	4	2D	480p	✓	R	4219	2023	2180
NYUDv2 [46]	Indoor	2012	40	2.5D	480 × 640	✗	R	795	654	N/A
SUN3D [47]	Indoor	2013	-	2.5D	640 × 480	✓	R	19640	N/A	N/A
SUNRGBD [48]	Indoor	2015	37	2.5D	Variable	✗	R	2666	2619	5050
RGB-D Object Dataset [49]	Household objects	2011	51	2.5D	640 × 480	✓	R	207920	N/A	N/A
ShapeNet Part [50]	Object/Part	2016	16/50	3D	N/A	✗	S	31,963	N/A	N/A
Stanford 2D-3D-S [51]	Indoor	2017	13	2D/2.5D/3D	1080 × 1080	✓	R	70469	N/A	N/A
3D Mesh [52]	Object/Part	2009	19	3D	N/A	✗	S	380	N/A	N/A
Sydney Urban Objects Dataset [53]	Urban (Objects)	2013	26	3D	N/A	✗	R	41	N/A	N/A
Large-Scale Point Cloud Classification Benchmark [54]	Urban/Nature	2016	8	3D	N/A	✗	R	15	N/A	15

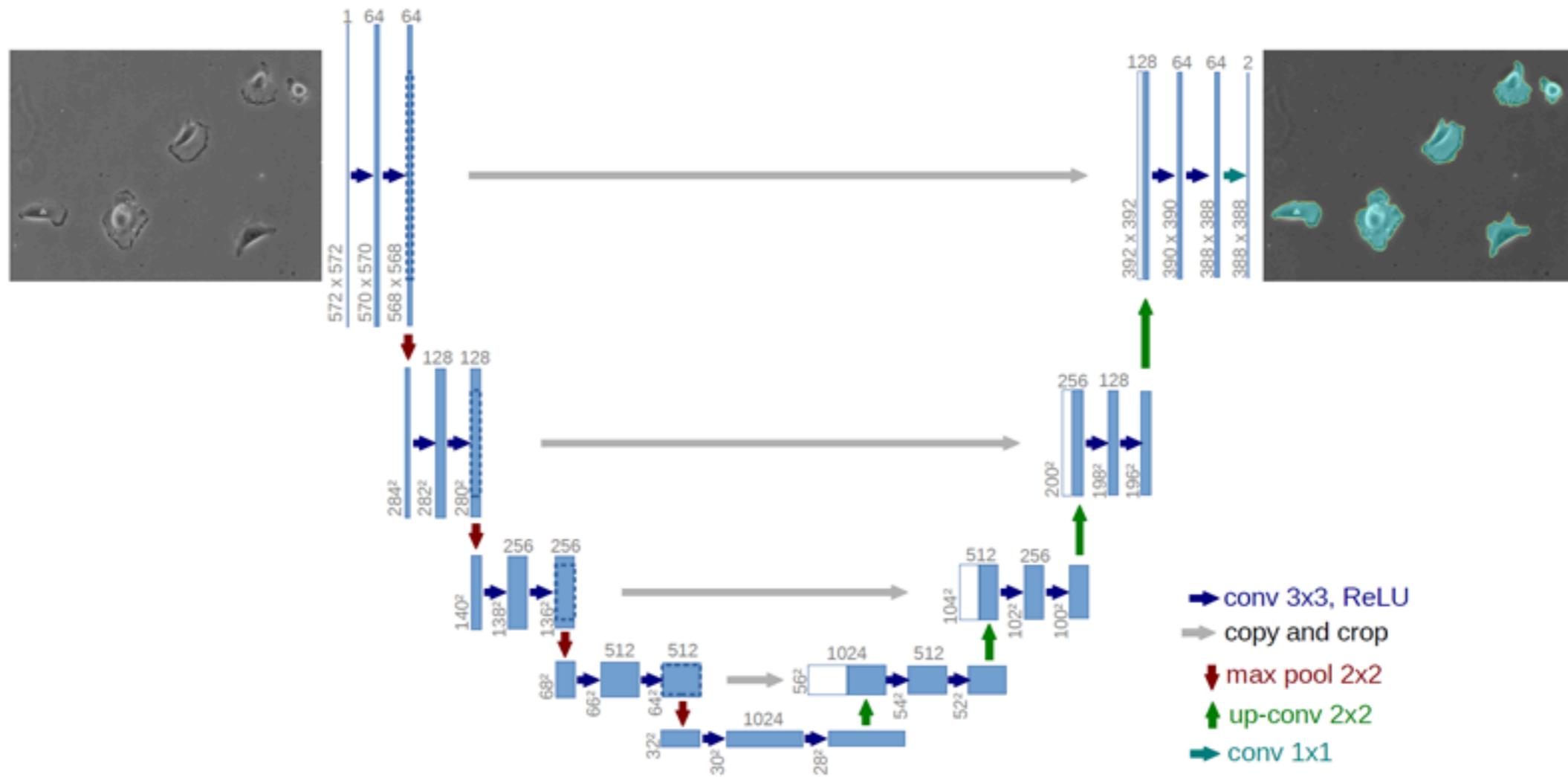
# FCN



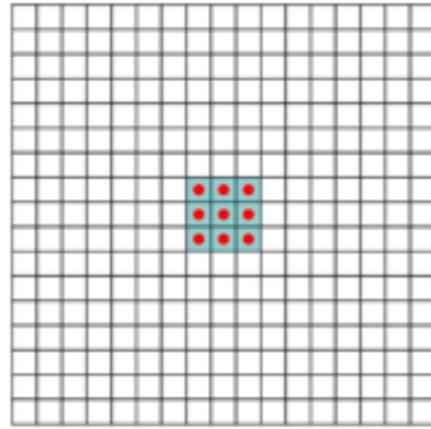
# SegNet



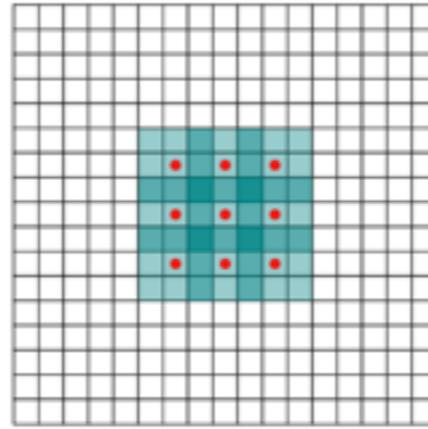
# U-Net



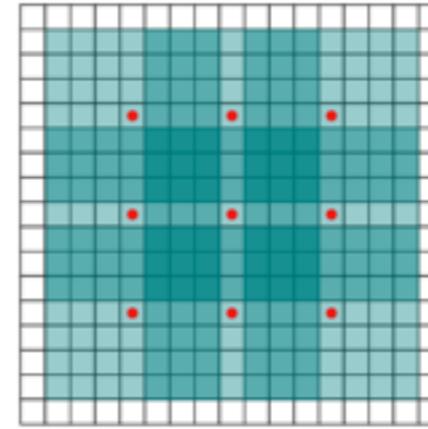
# DILATED CONVOLUTIONS



(a)

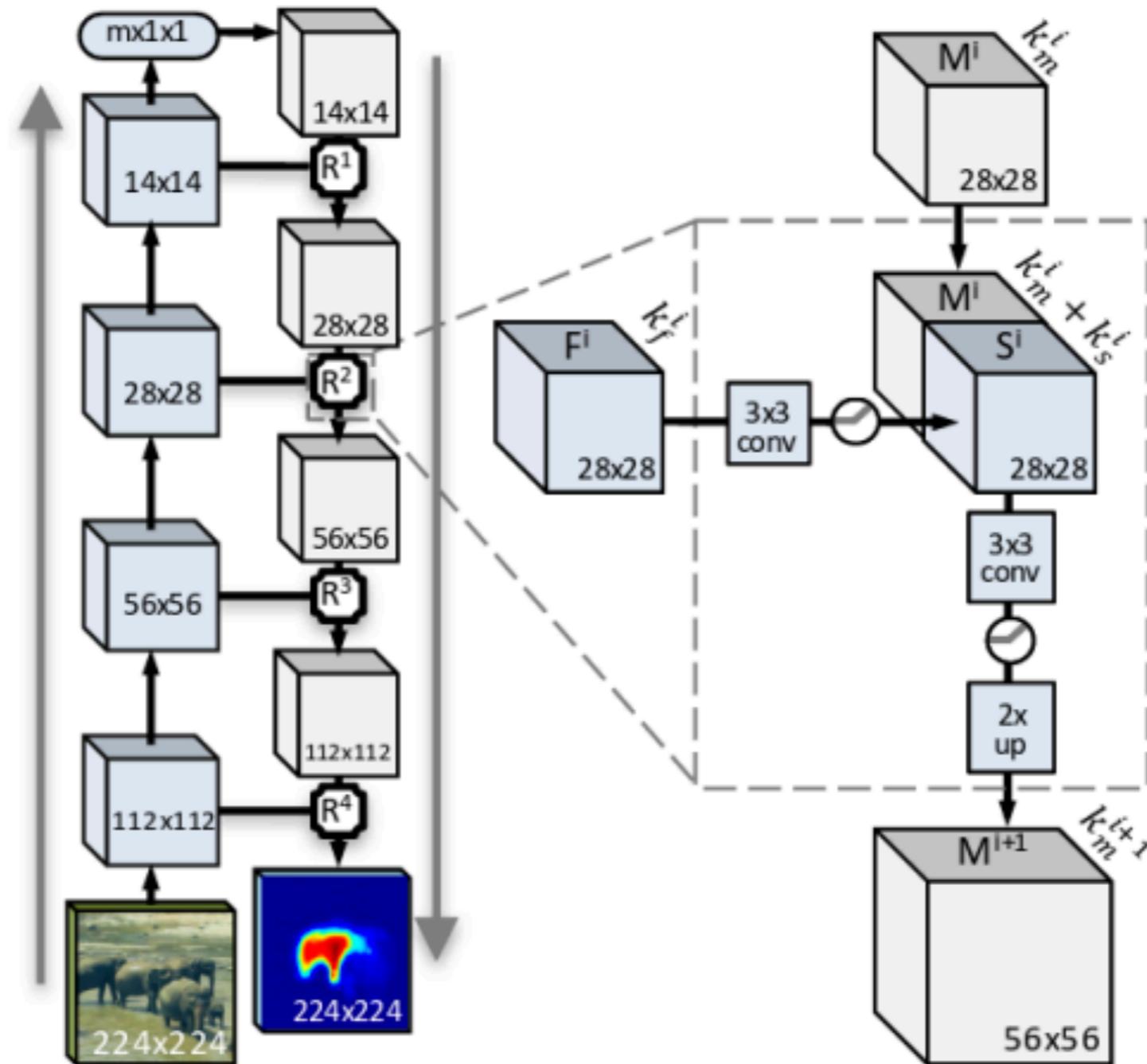


(b)

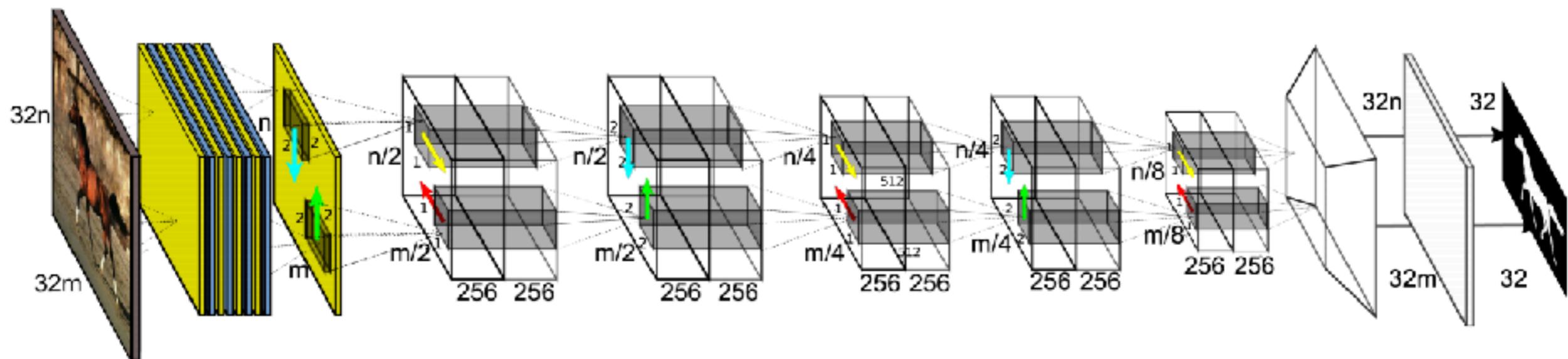


(c)

# Multi-scale CNN

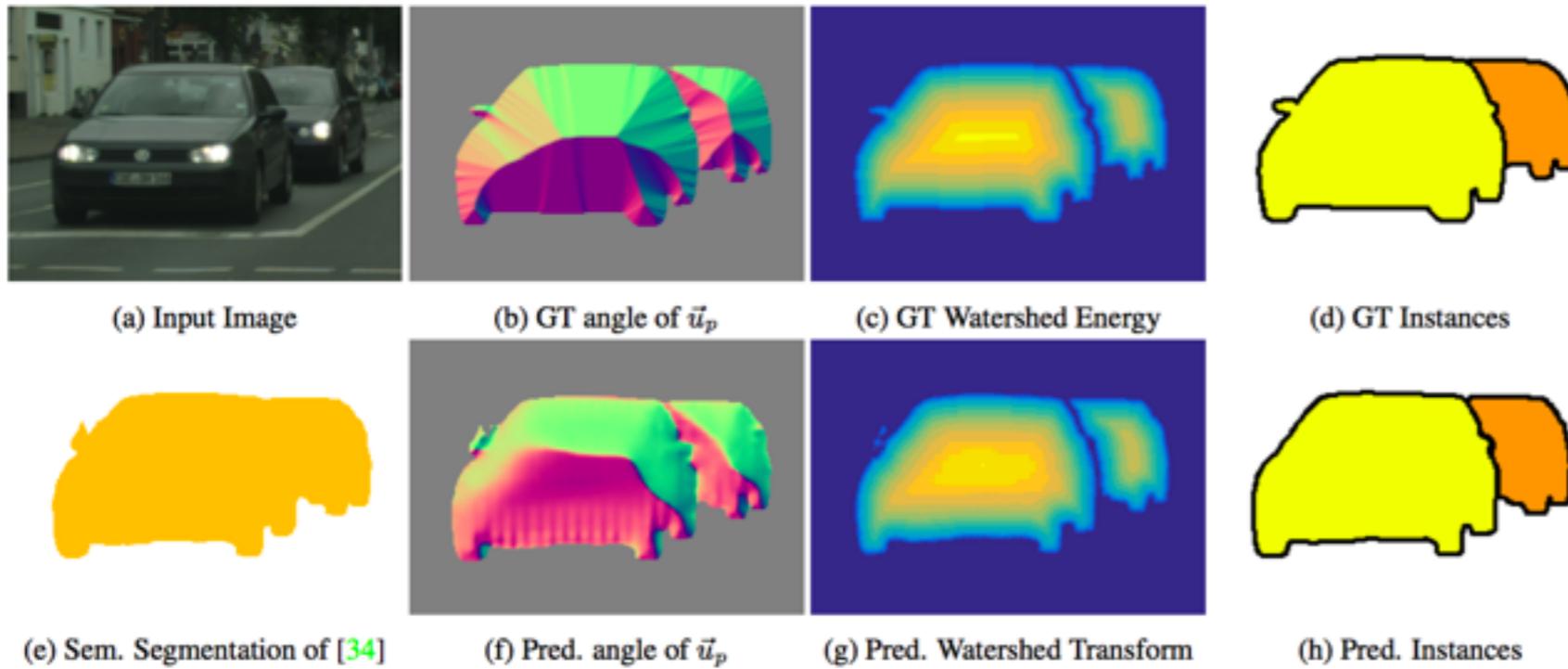


# ReSeg net.



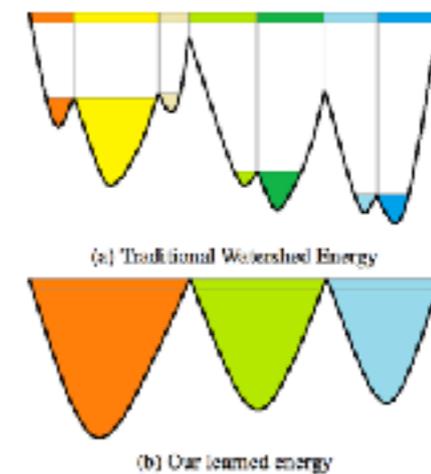
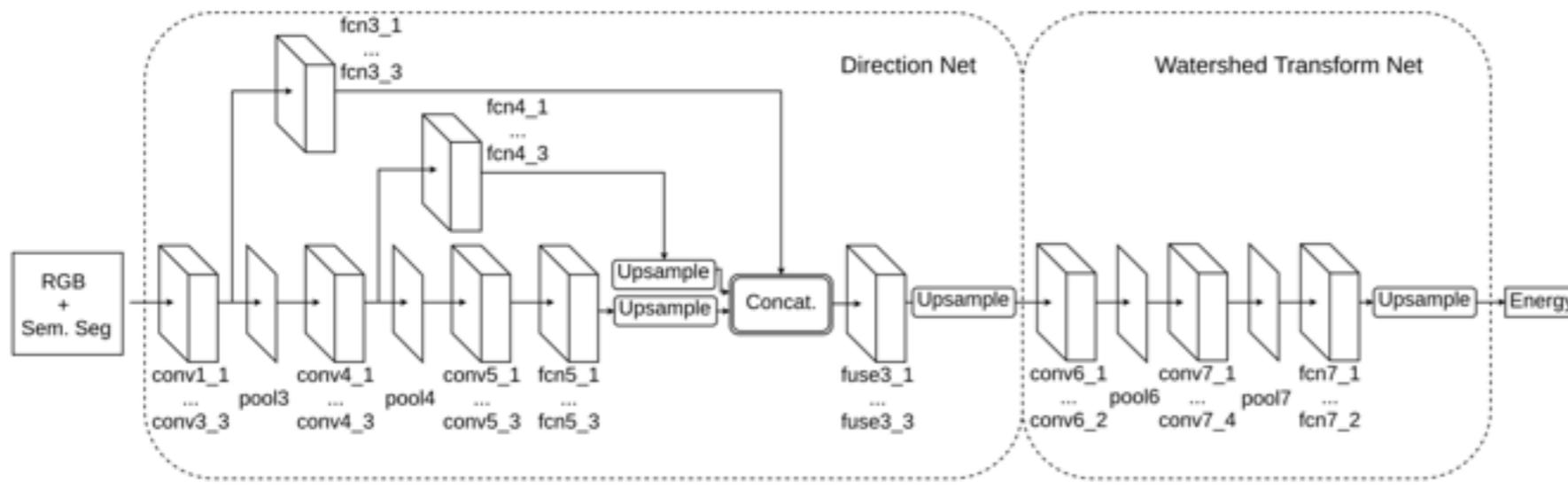
# Instance Segmentation

# Deep Watershed Transform

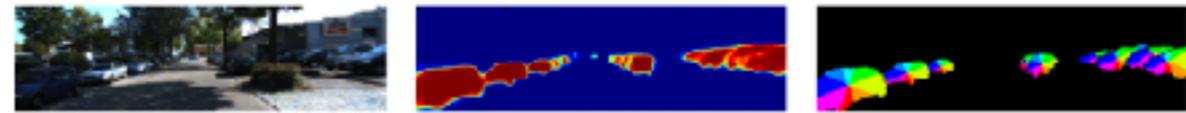
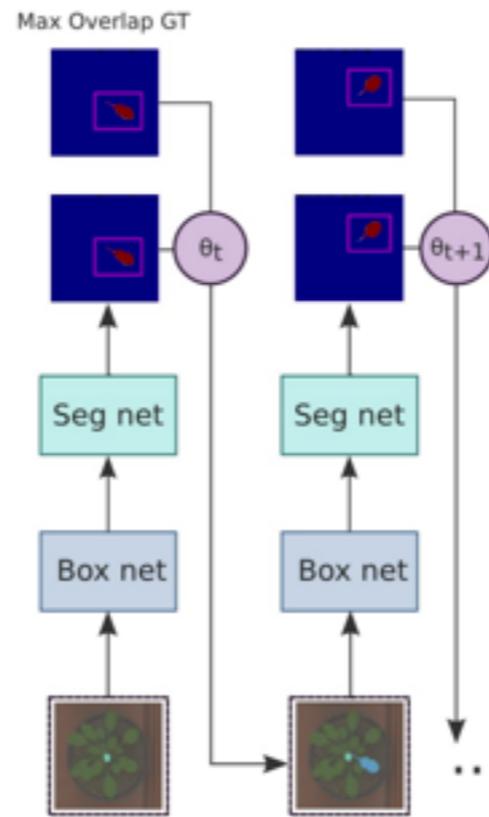
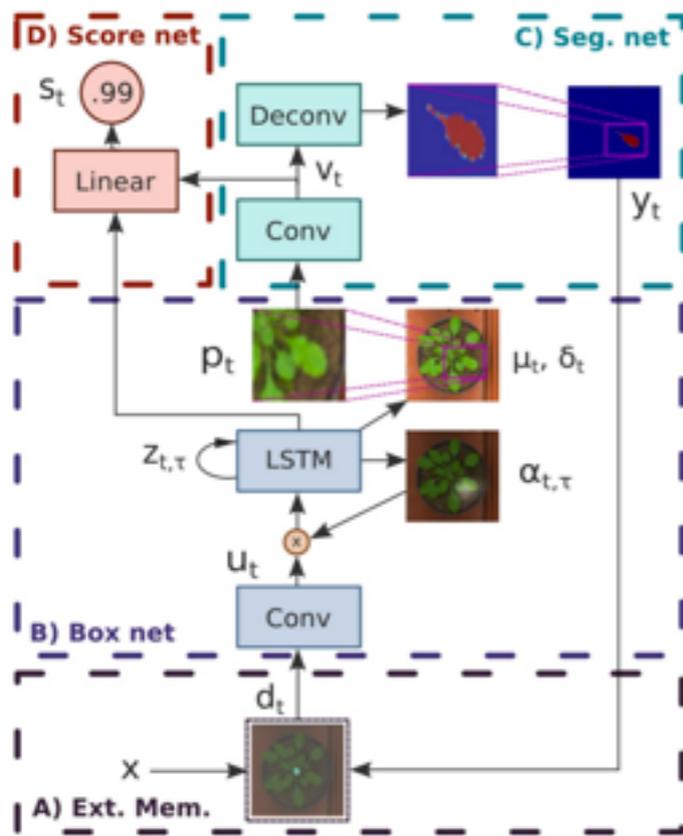
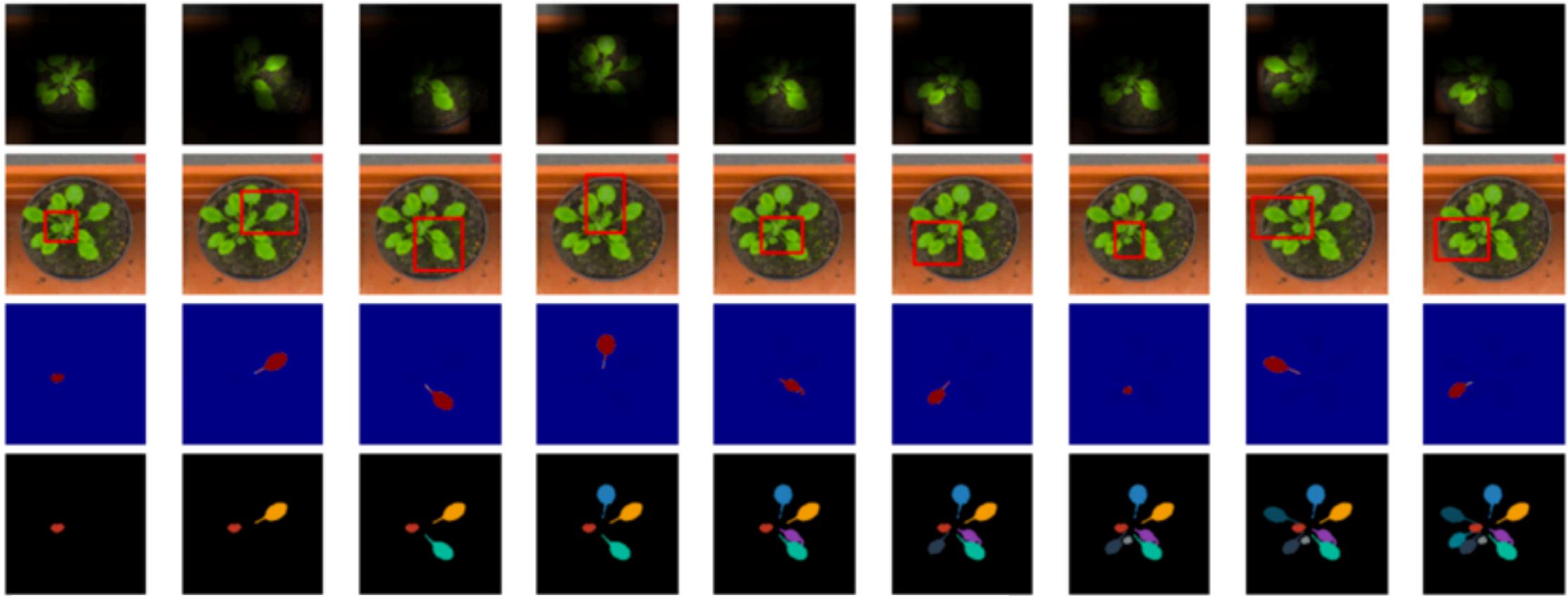


$$l_{\text{direction}} = \sum_{p \in \mathcal{P}_{\text{obj}}} w_p \|\cos^{-1} \langle \vec{u}_{p,\text{GT}}, \vec{u}_{p,\text{pred}} \rangle\|^2$$

$$l_{\text{watershed}} = \sum_{p \in \mathcal{P}_{\text{obj}}} \sum_{k=1}^K w_p c_k (\bar{t}_{p,k} \log \bar{y}_{p,k} + t_{p,k} \log y_{p,k})$$



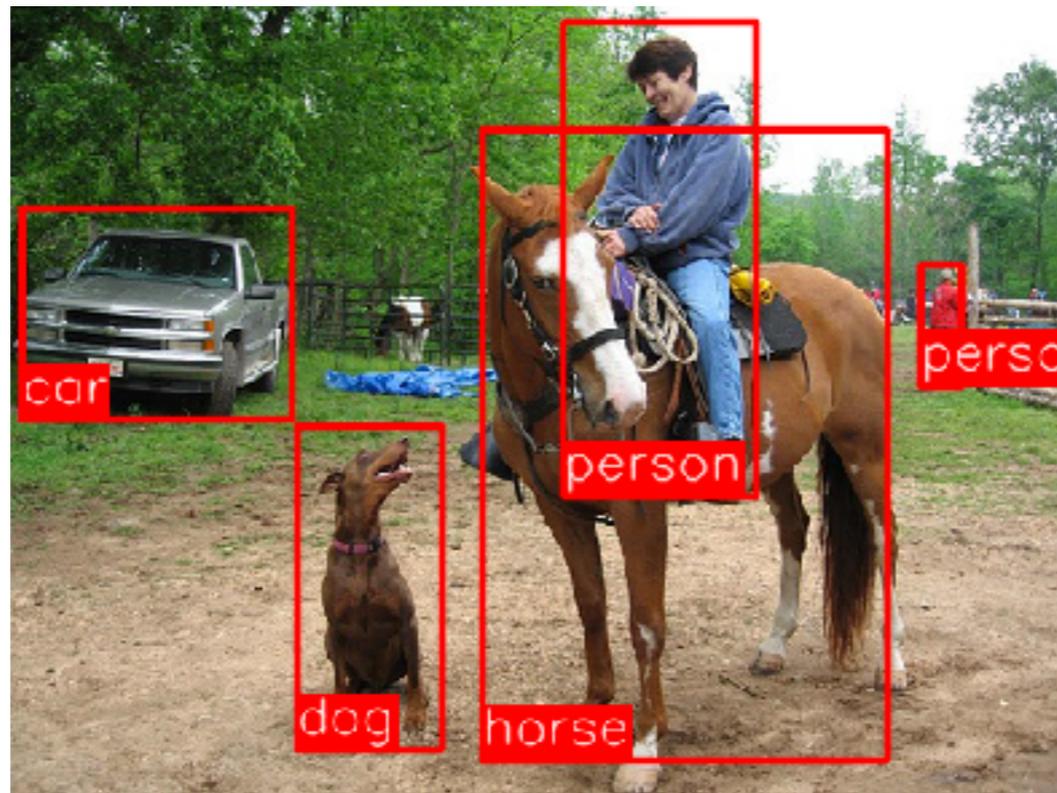
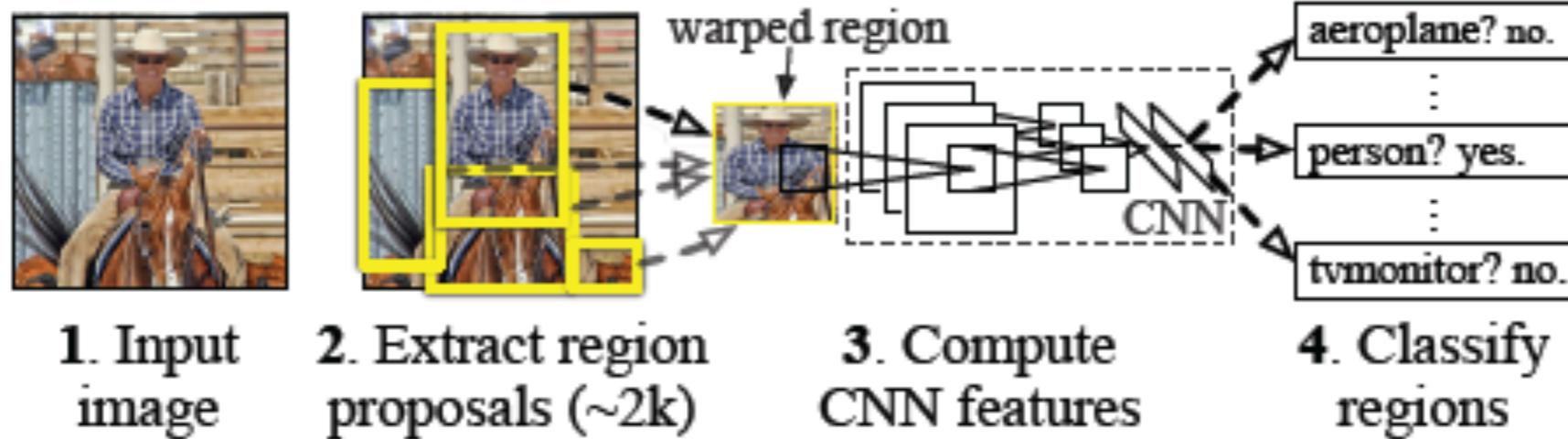
# Recurrent Attention



Detection

# RCNN

## R-CNN: *Regions with CNN features*

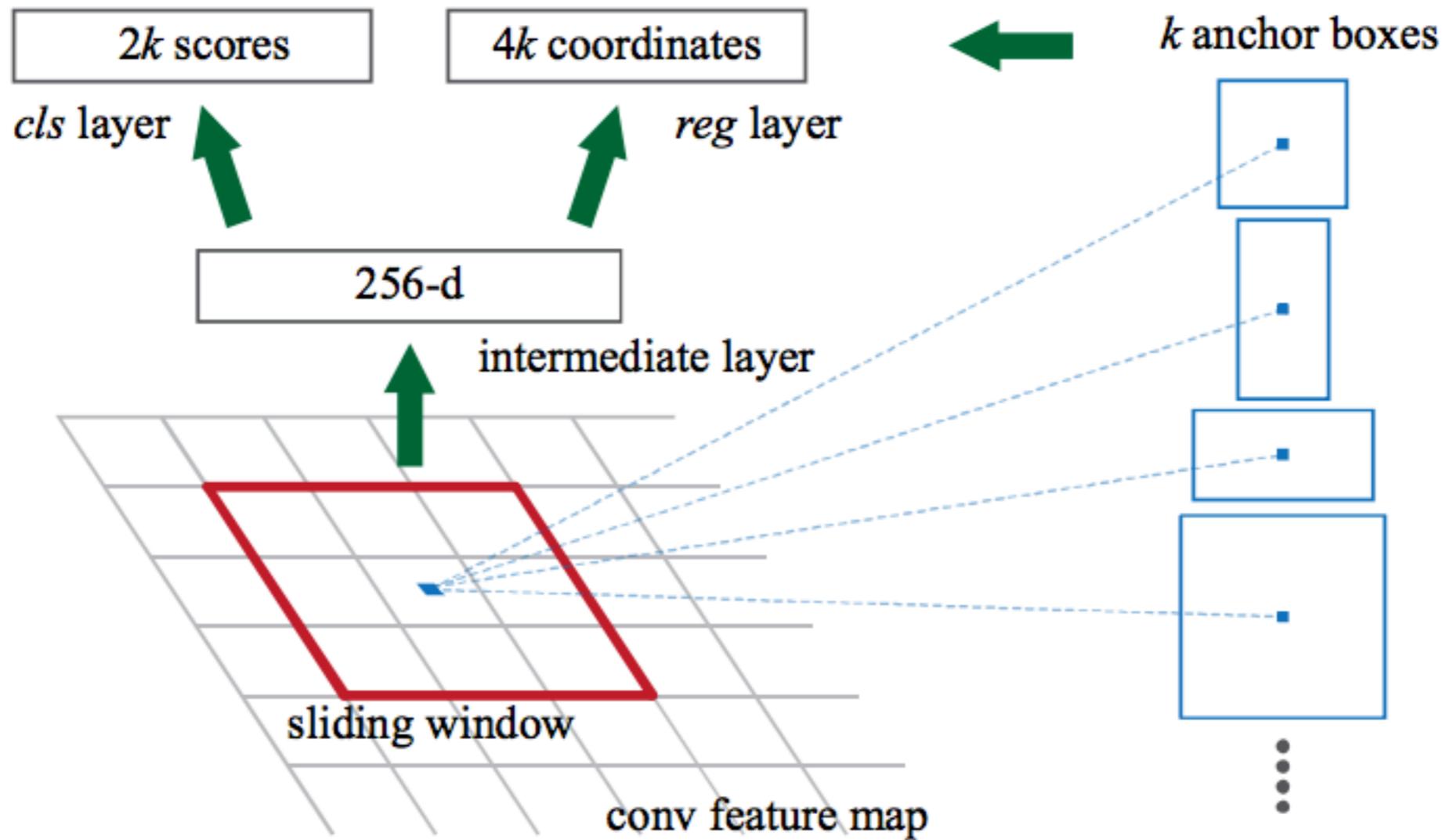


# Yolo



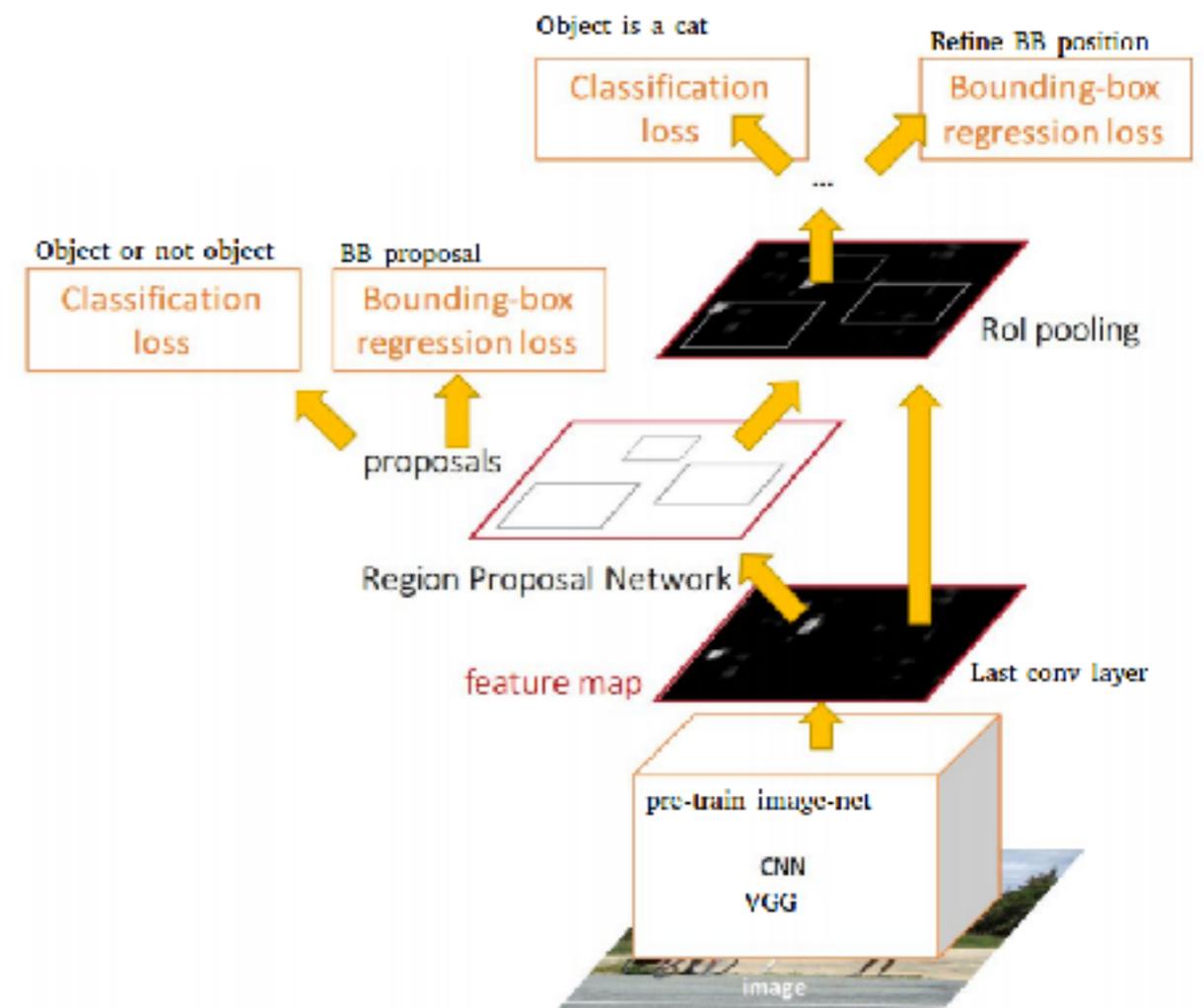
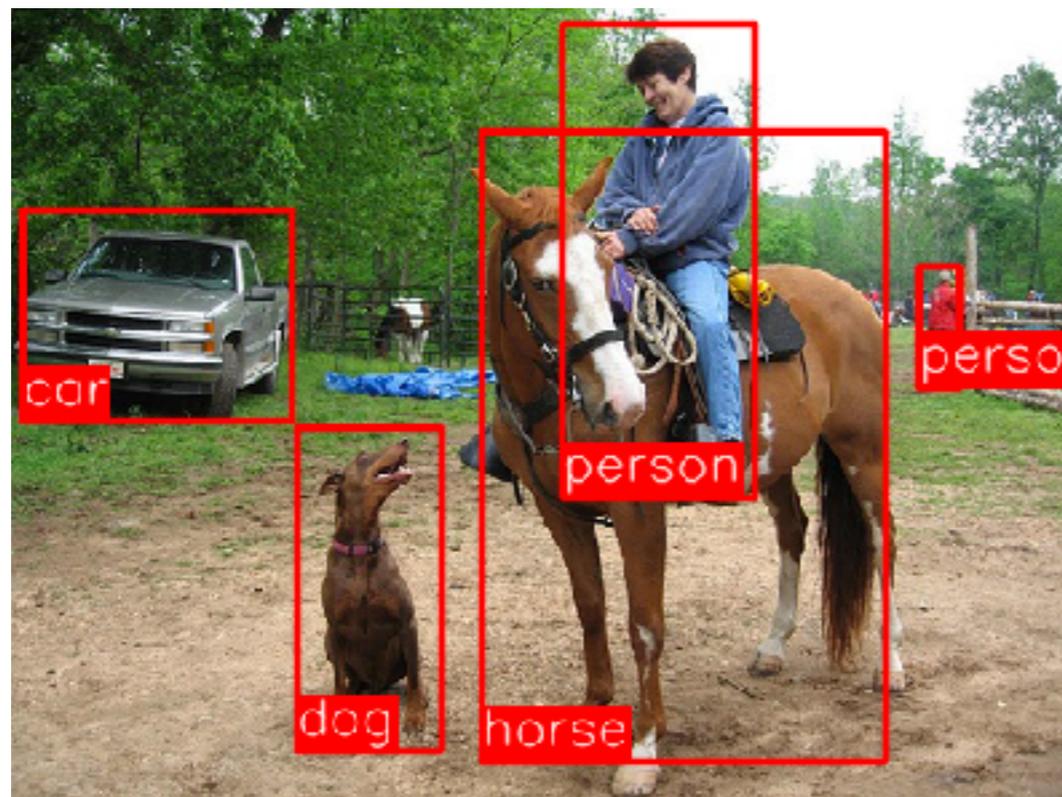
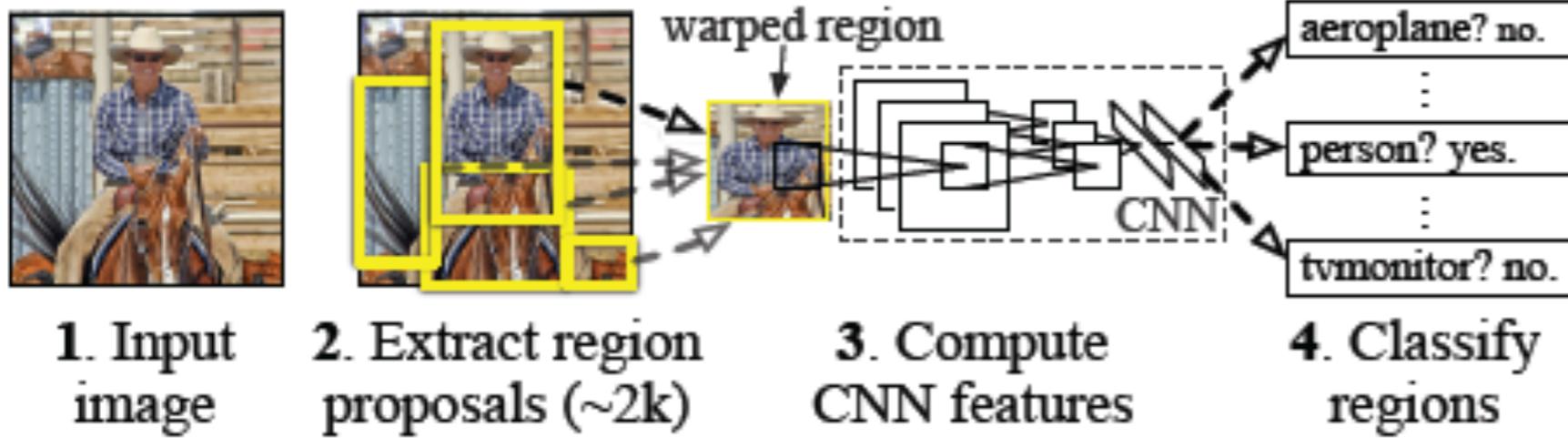
$$\begin{aligned}
 & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \\
 & \quad + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
 & \quad + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)
 \end{aligned}$$

# Region Proposal Networks

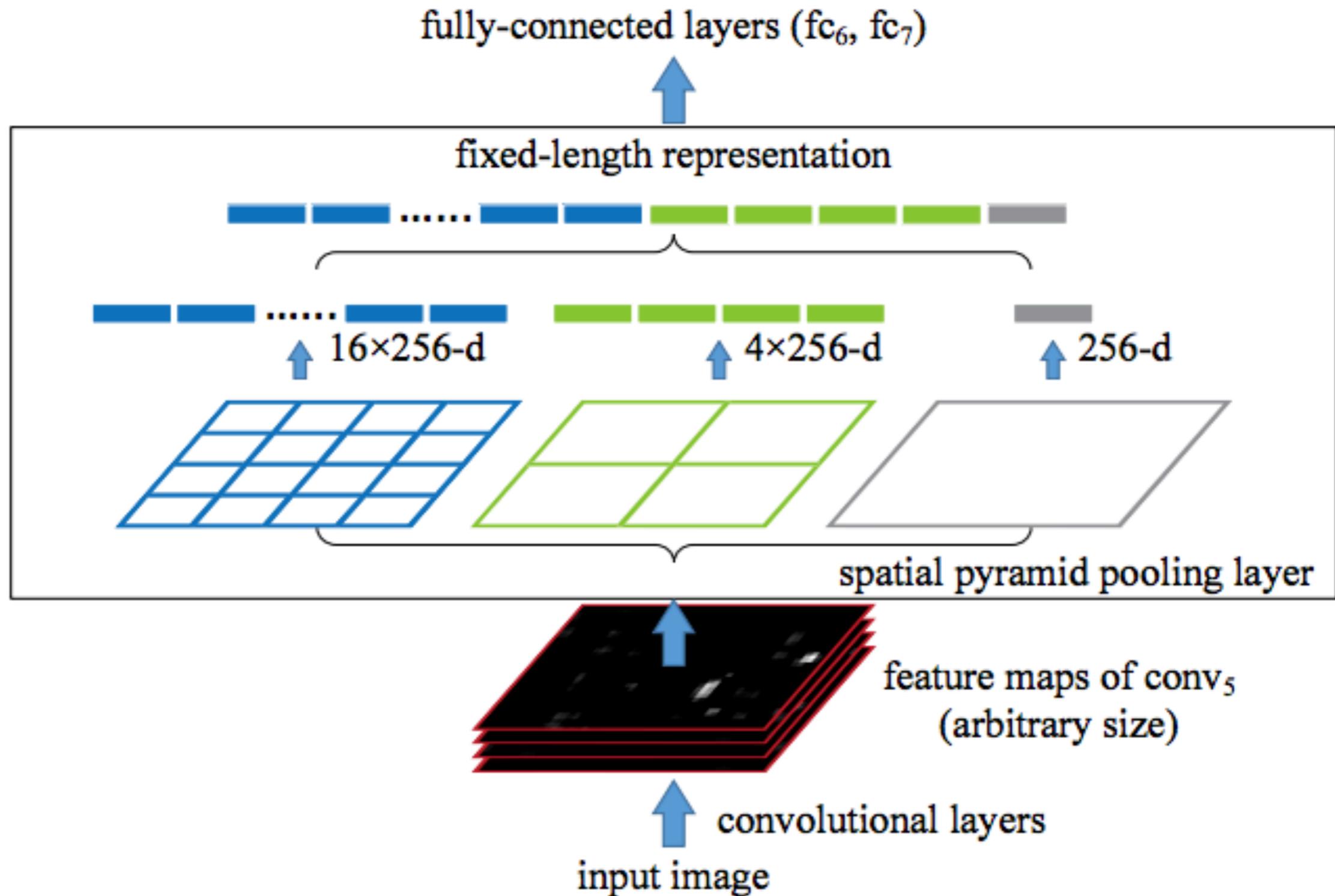


# Faster RCNN

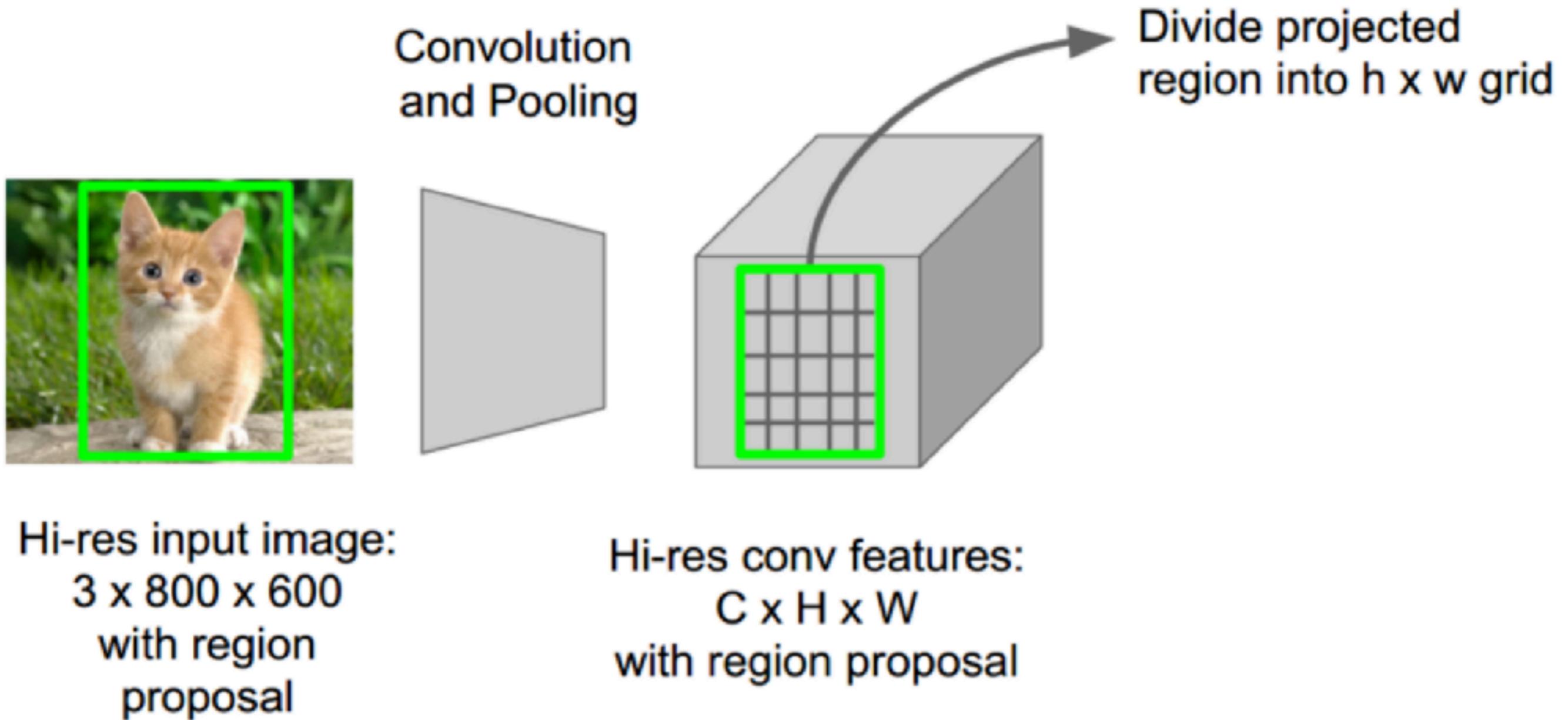
## R-CNN: *Regions with CNN features*



# Spatial Pyramid Pooling



# ROI Pooling



# FPNs

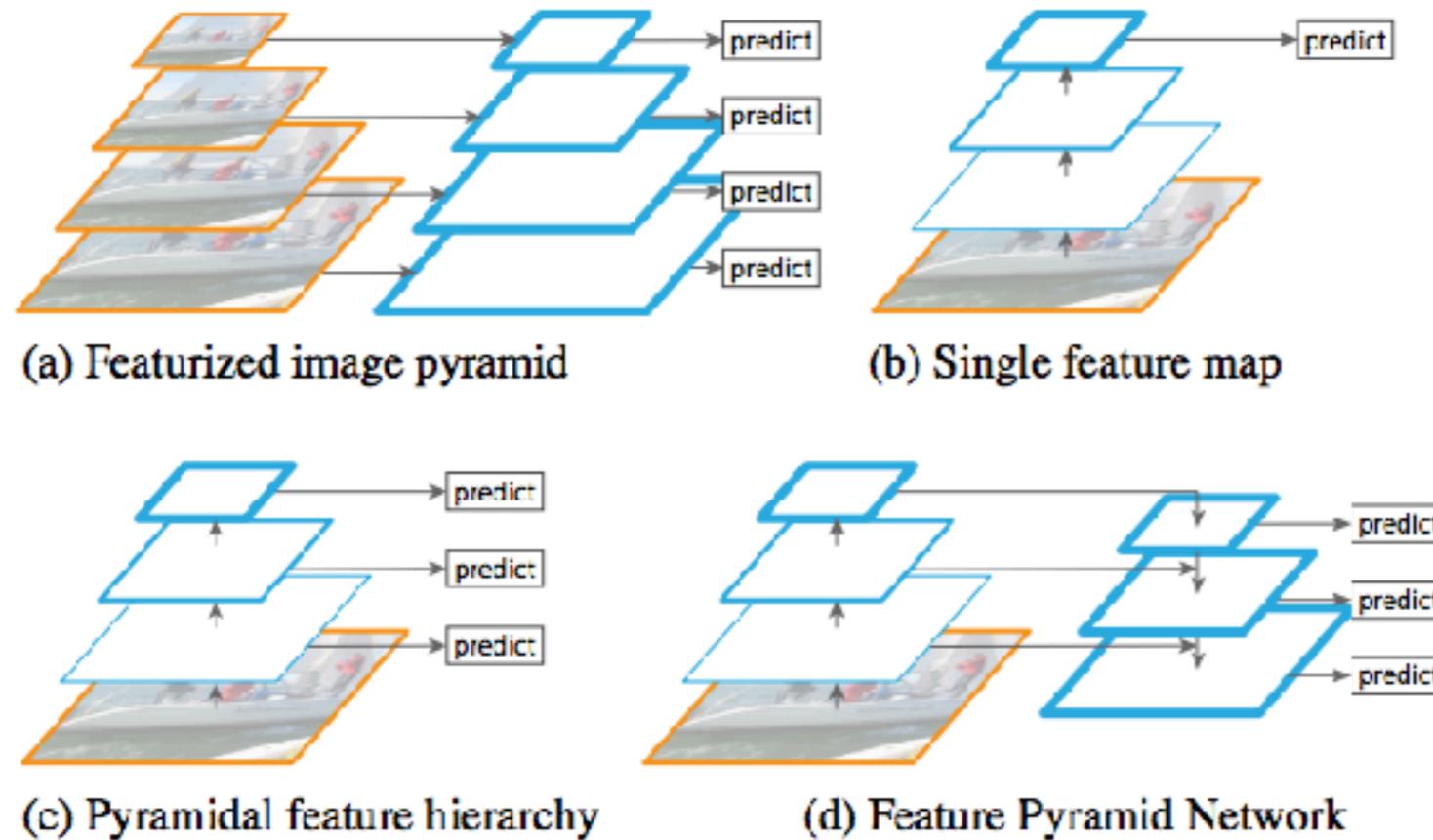
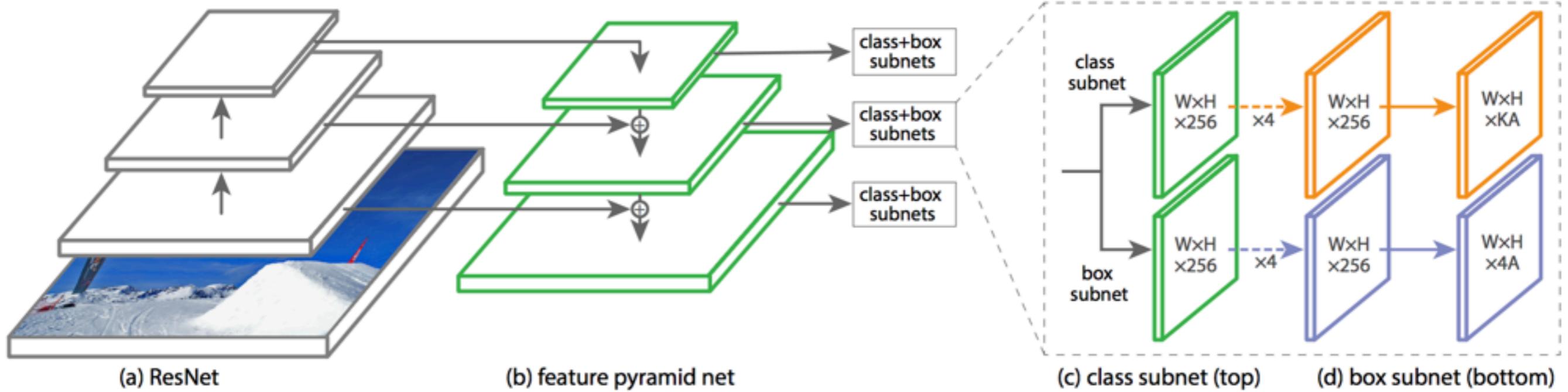


Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicated by blue outlines and thicker outlines denote semantically stronger features.

# RetinaNet



# Mask RCNN

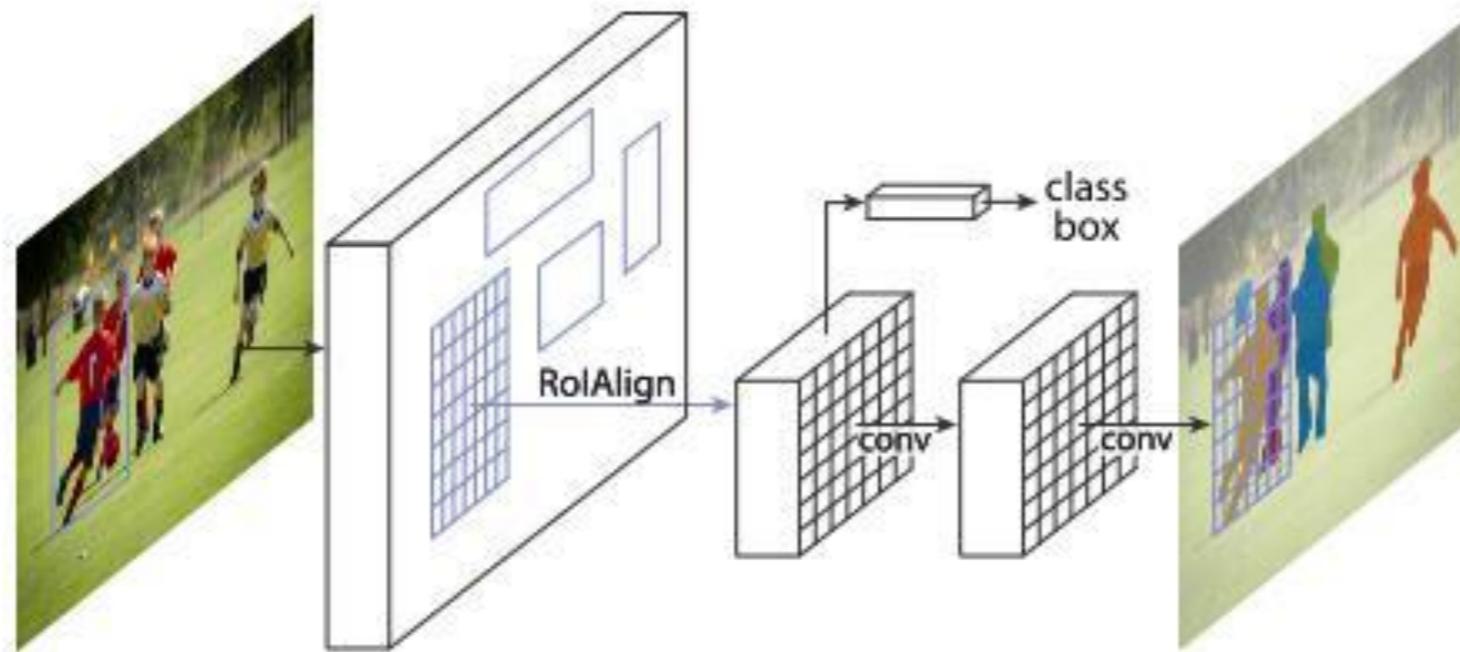
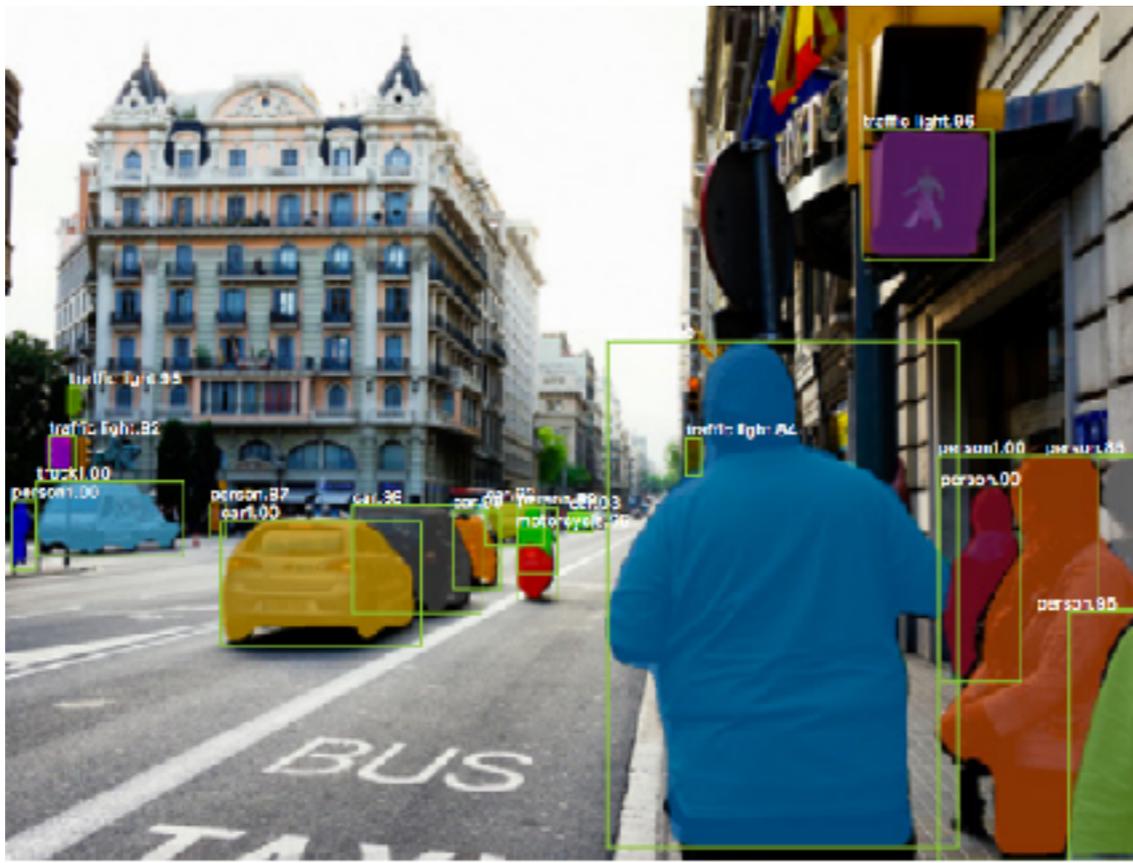


Figure 1. The **Mask R-CNN** framework for instance segmentation.



- R-CNN: <https://arxiv.org/abs/1311.2524>
- Fast R-CNN: <https://arxiv.org/abs/1504.08083>
- Faster R-CNN: <https://arxiv.org/abs/1506.01497>
- Mask R-CNN: <https://arxiv.org/abs/1703.06870>

## **Faster R-CNN**

Caffe: <https://github.com/rbgirshick/py-faster-rcnn>

PyTorch: [https://github.com/longcw/faster\\_rcnn\\_pytorch](https://github.com/longcw/faster_rcnn_pytorch)

MatLab: [https://github.com/ShaoqingRen/faster\\_rcnn](https://github.com/ShaoqingRen/faster_rcnn)

## **Mask R-CNN**

PyTorch: [https://github.com/felixgwu/mask\\_rcnn\\_pytorch](https://github.com/felixgwu/mask_rcnn_pytorch)

TensorFlow: <https://github.com/CharlesShang/FastMaskRCNN>

IF WE HAVE TIME:

Discussion on evaluation

# Intersection over Union

- **Pixel Accuracy (PA):** it is the simplest metric, simply computing a ratio between the amount of properly classified pixels and the total number of them.

$$PA = \frac{\sum_{i=0}^k p_{ii}}{\sum_{i=0}^k \sum_{j=0}^k p_{ij}}$$

- **Mean Pixel Accuracy (MPA):** a slightly improved PA in which the ratio of correct pixels is computed in a per-class basis and then averaged over the total number of classes.

$$MPA = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij}}$$

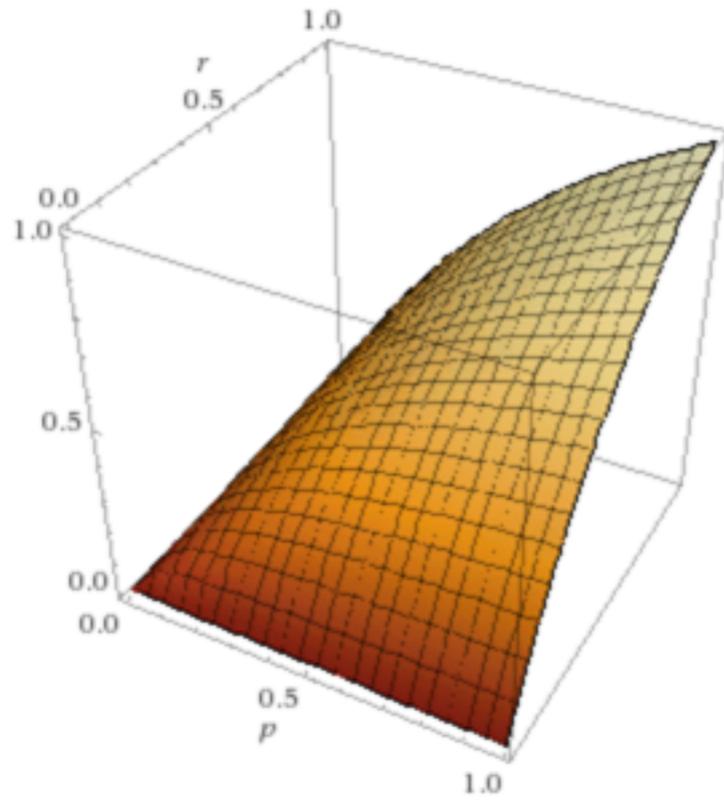


$$Accuracy = \frac{TP}{TP+FP+FN}$$

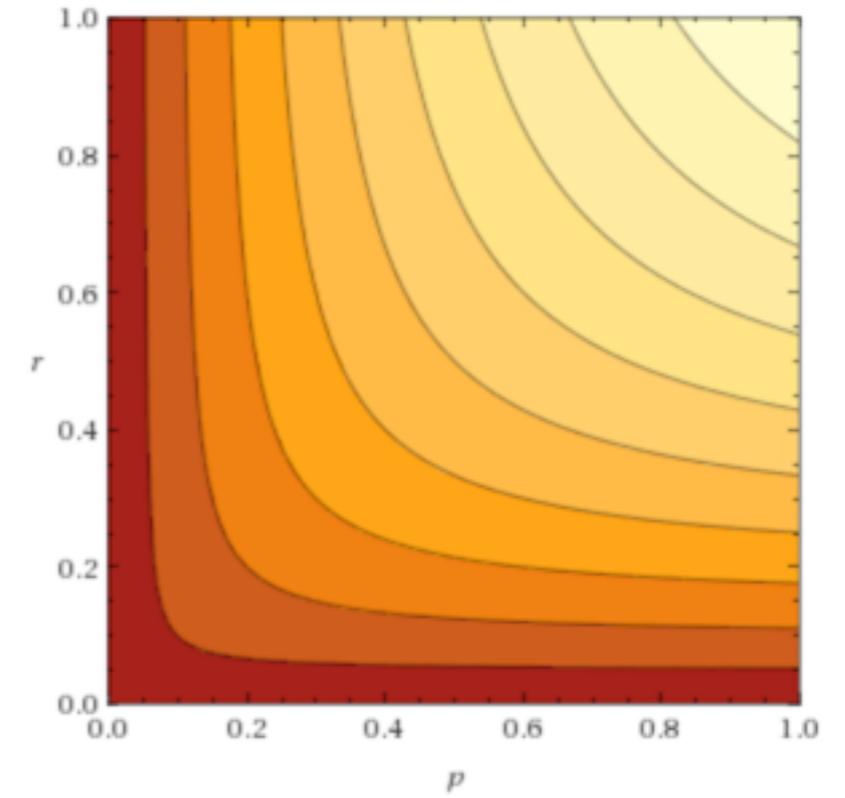
TP: True Positive  
FP: False Positive  
FN: False Negative

$$MIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{p_{ii}}{\sum_{j=0}^k p_{ij} + \sum_{j=0}^k p_{ji} - p_{ii}}$$

3D plot:



# F1



	Y_hat = True	Y_hat = False
Y = True	50	50
Y = False	450	450

$$\text{Precision} = \frac{TP}{TP + FP}$$

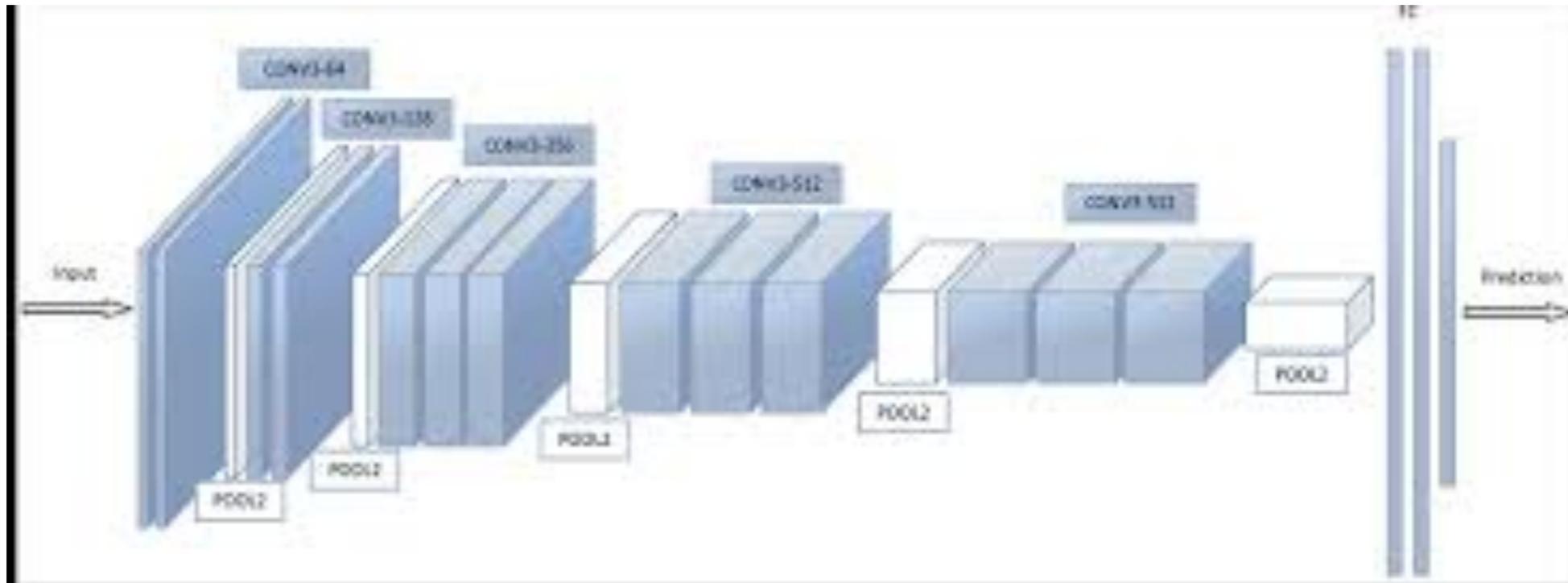
$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

# Atrous Models in Semantic Segmentation

Why Atrous?

# VGG 16



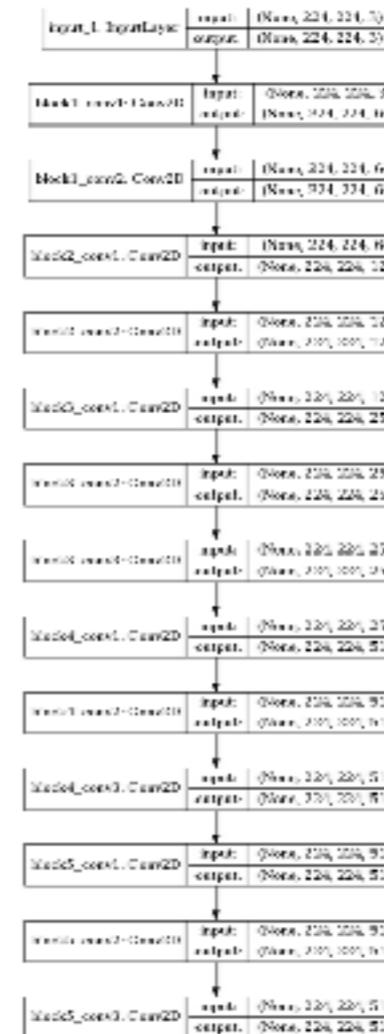
Let's remove the fully connected layers  
and do segmentation.

Problem?

Output Size is too small!

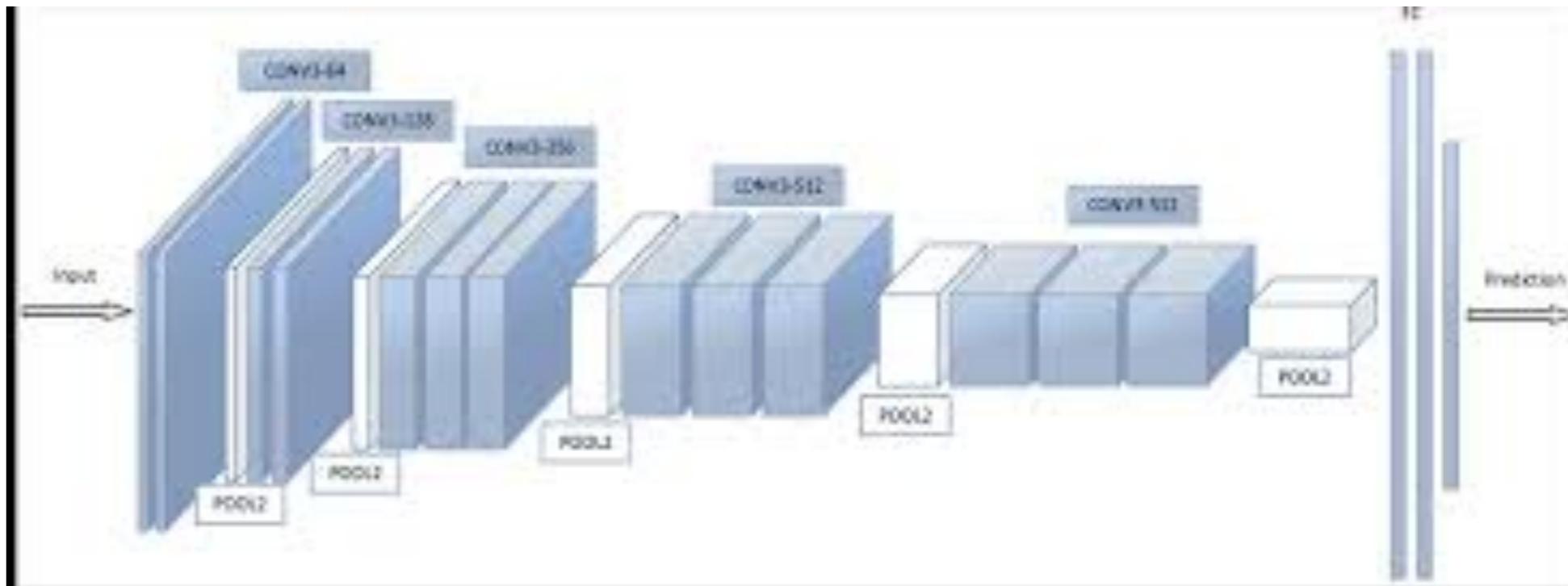
So let's remove all max pools!

What's the problem?



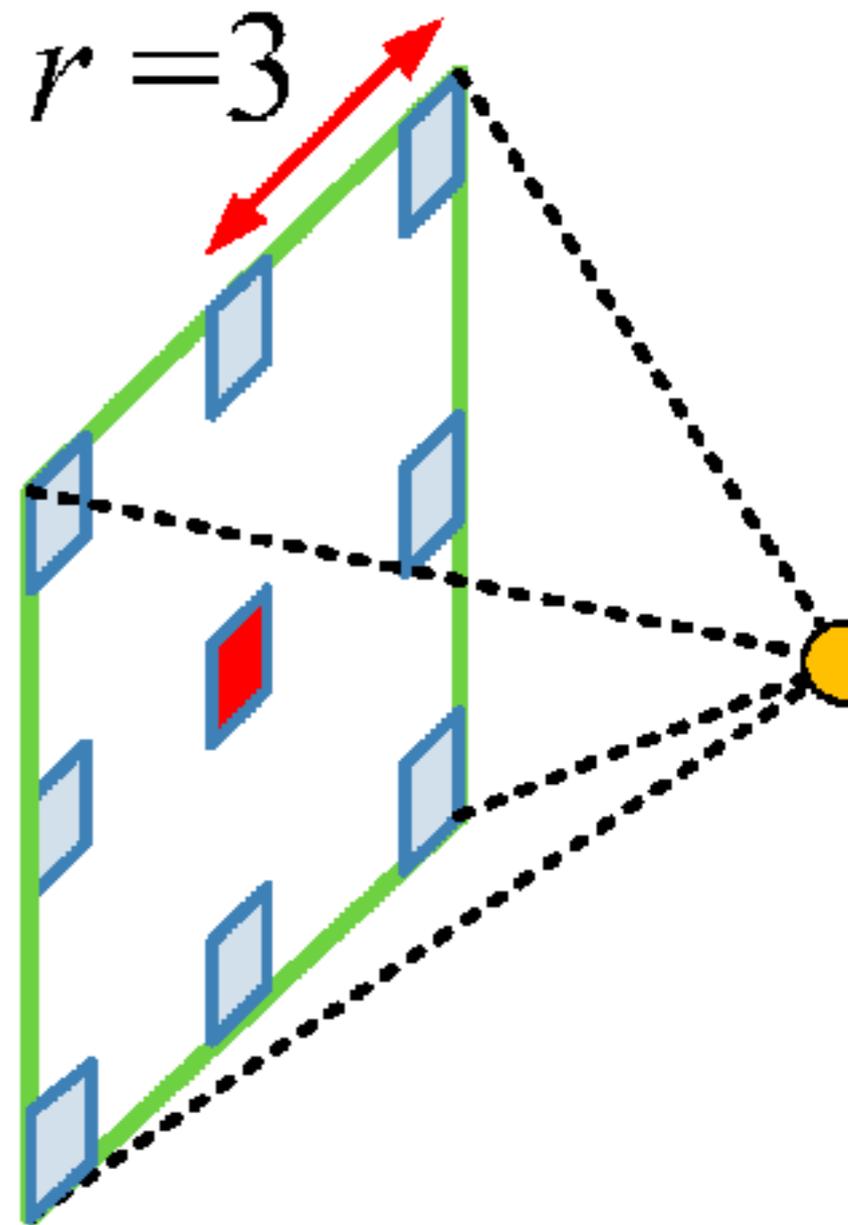
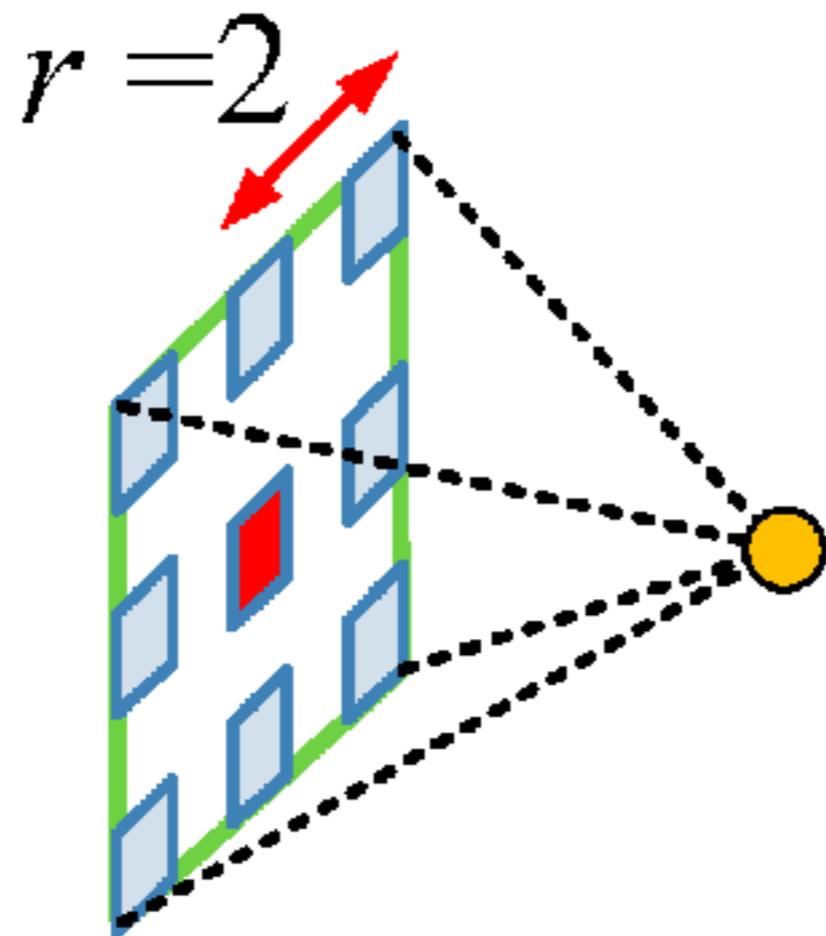
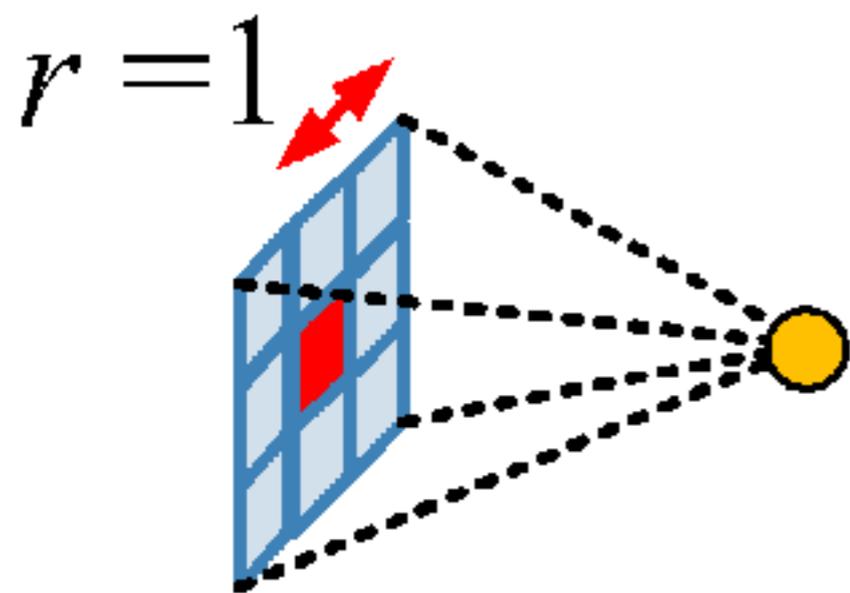
Receptive Field is too small!

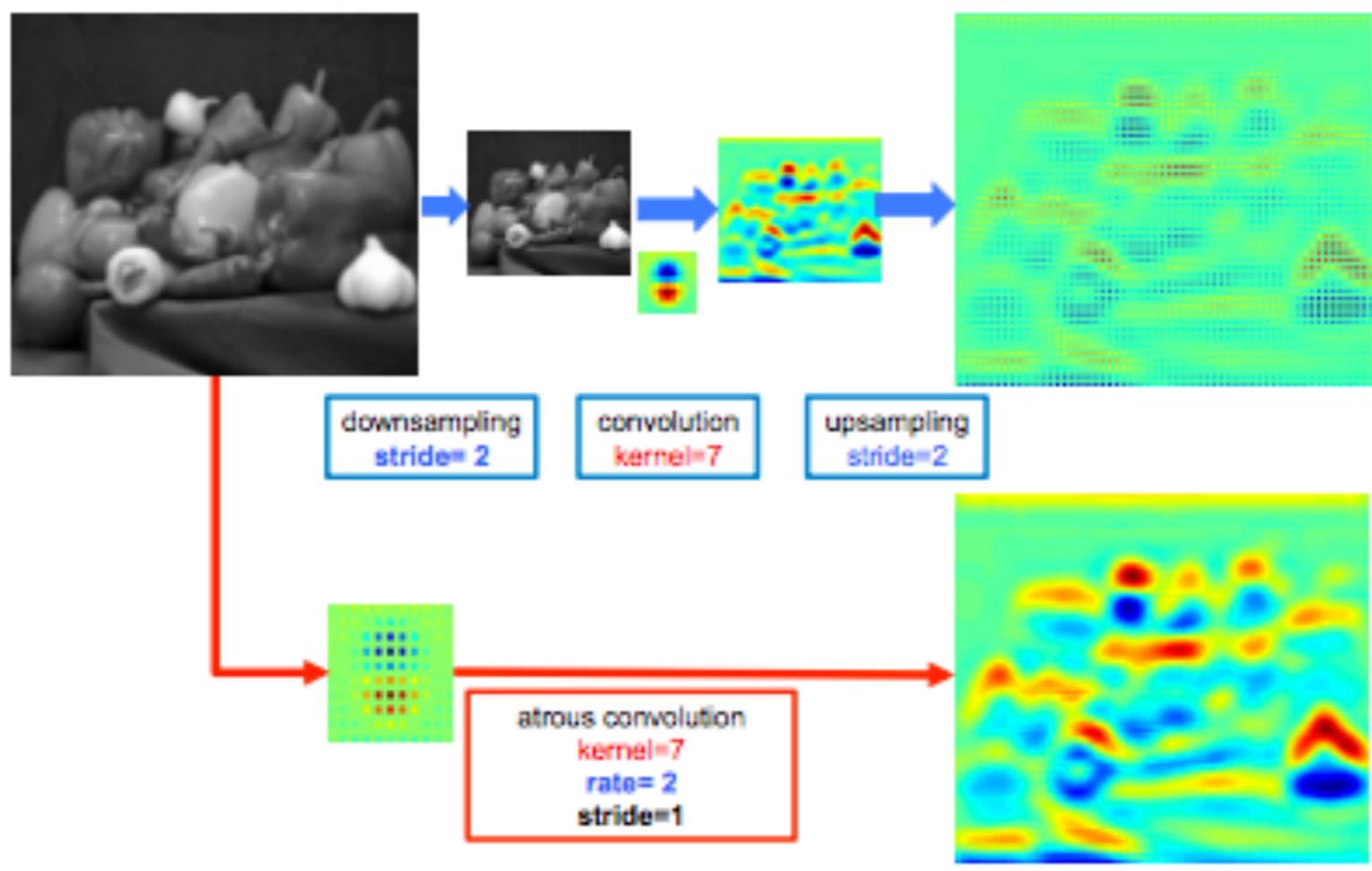
So let's stride the convolutions by 2!



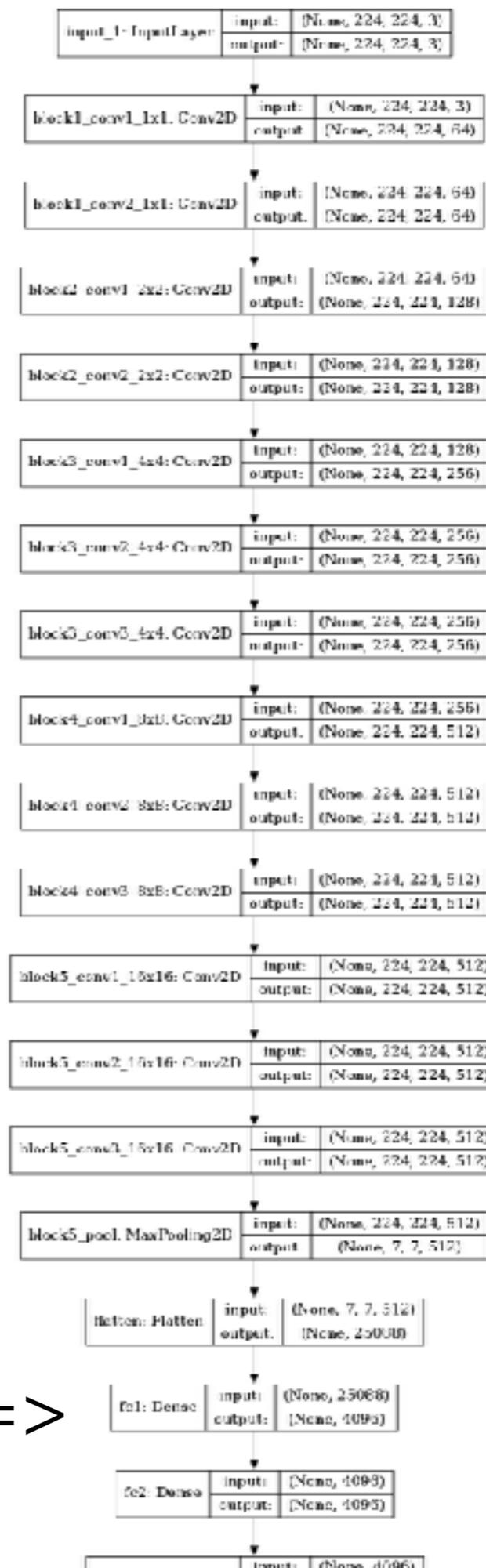
Size Problem again!

Solution: Atrous



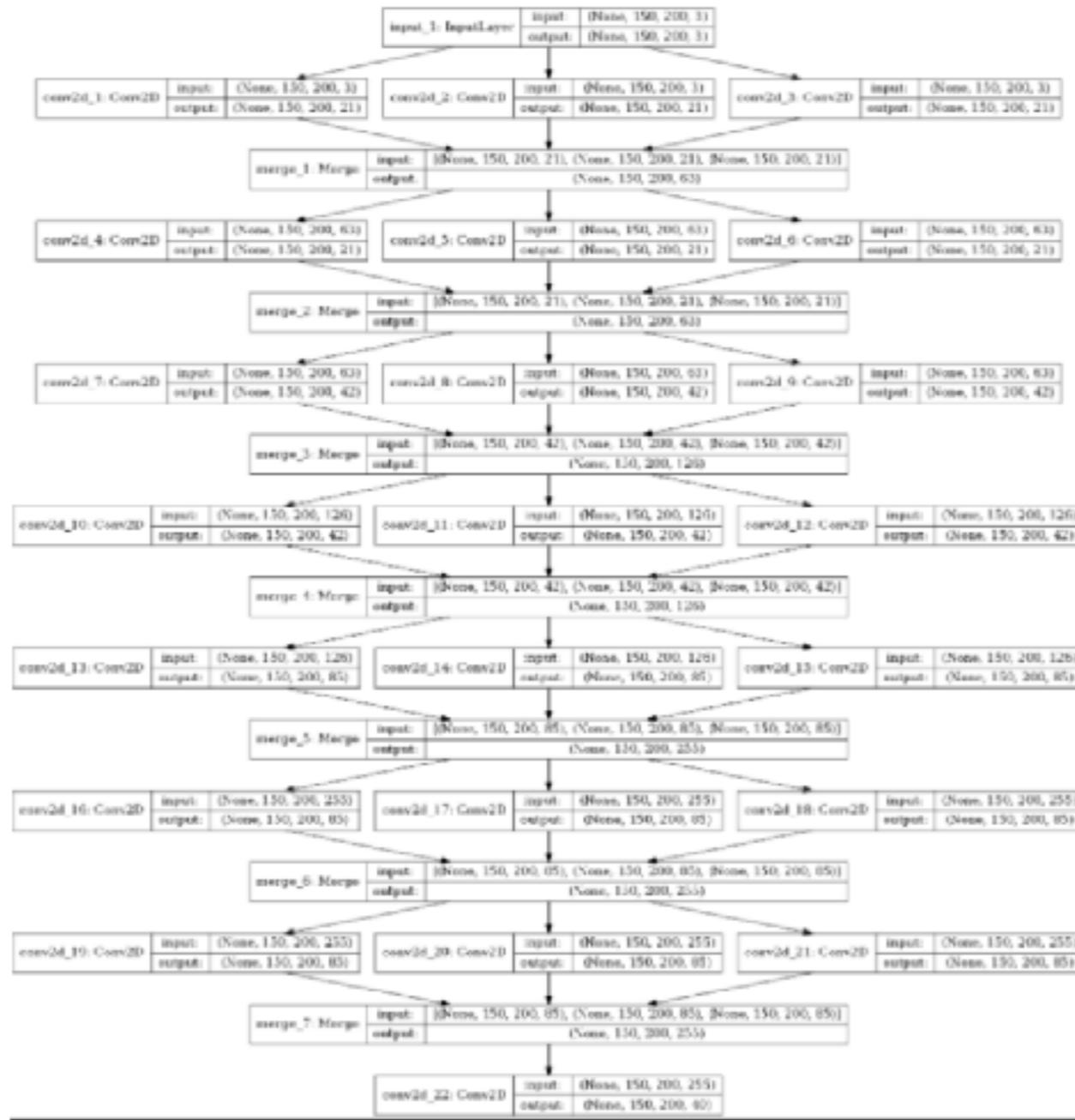


# Atrous VGG



Can still classify =>

# Size invariance w/ Atrous: Atrous-Ception



F1 LOSS

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

F1 Score:

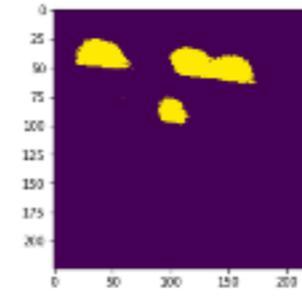
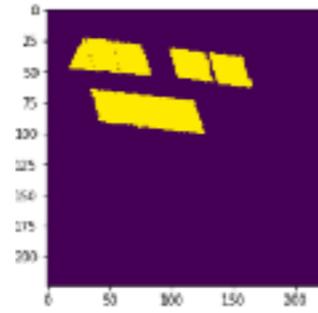
$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

## Binary Crossentropy Loss

$$L(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N H(p_n, q_n) = -\frac{1}{N} \sum_{n=1}^N \left[ y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right],$$

Let's Optimize for the right thing!



$y = \text{labels}, y' = \text{predictions}$

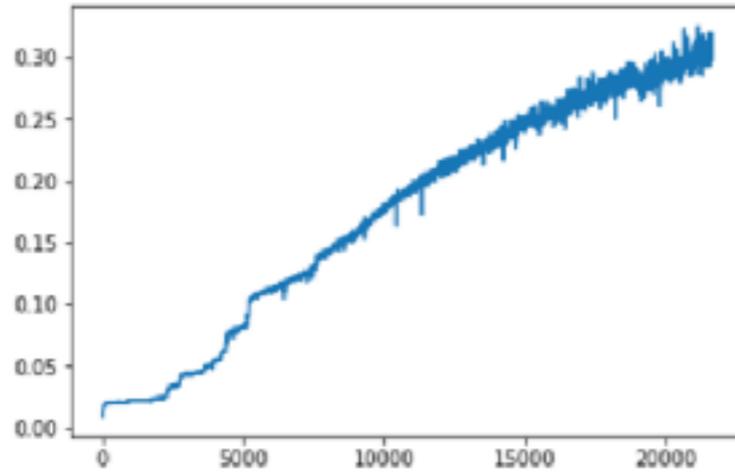
$$TP = \text{sum}(y \cdot y')$$

$$TP + FP = \text{sum}(y')$$

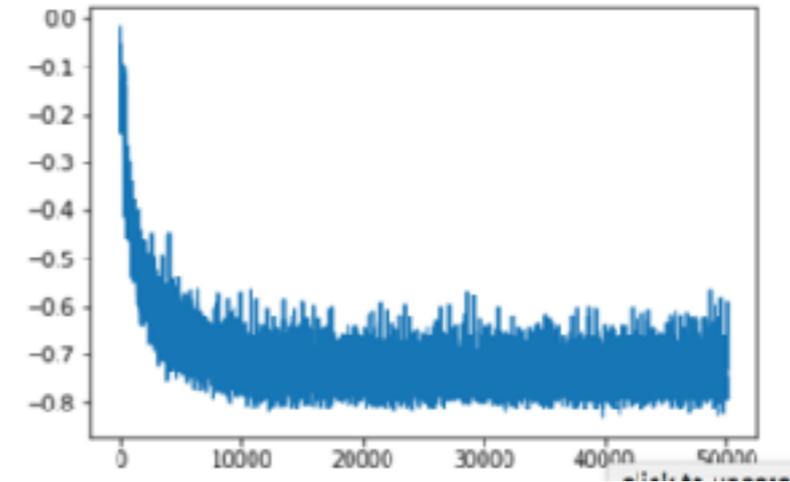
$$TP + FN = \text{sum}(y)$$

$$\text{Loss} = -F1$$

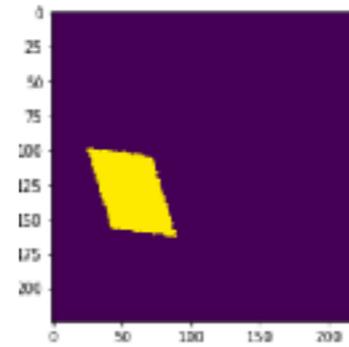
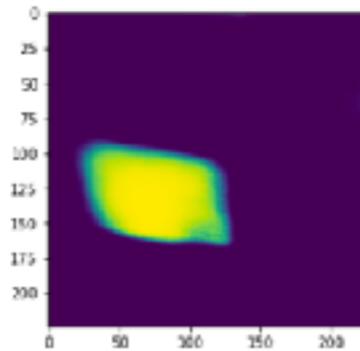
# Results:



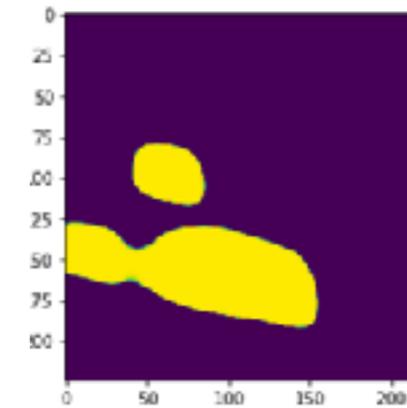
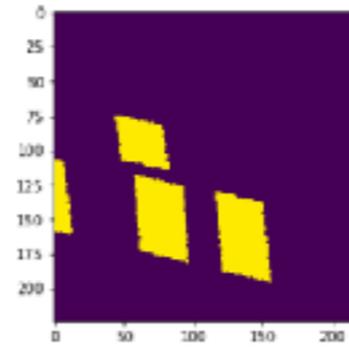
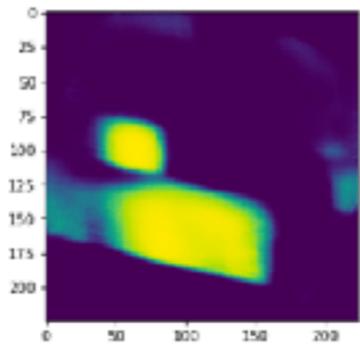
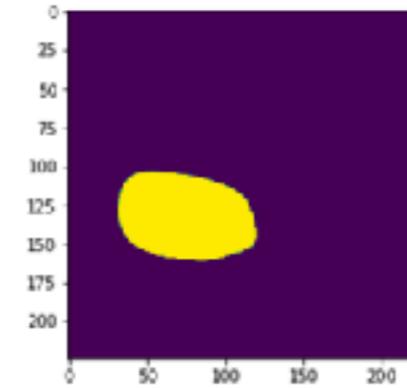
l\_loss -0.743874248784



## Cross entropy



## - F1



## GT