Introduction to Data Science Final Project

Topic :
RCNN-Family

111550038 陳易
111550151 徐嘉亨
111550177 吳定霖

R-CNN

1. Region Proposals (selective search)
2. Feature Extractor (AlexNet)
3. Detector (SVM classifier, bbox regressor)

Region Proposals (selective search)

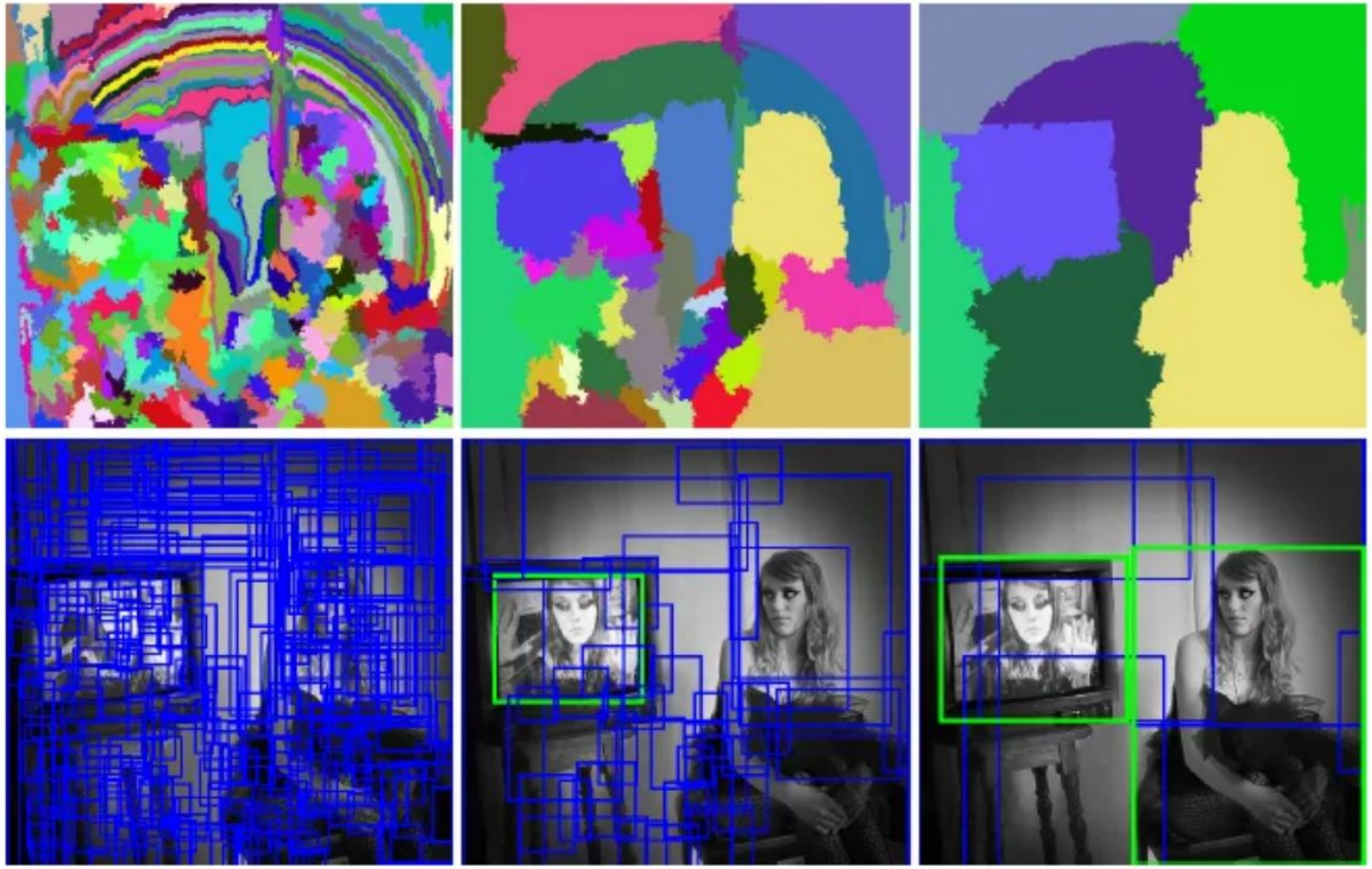
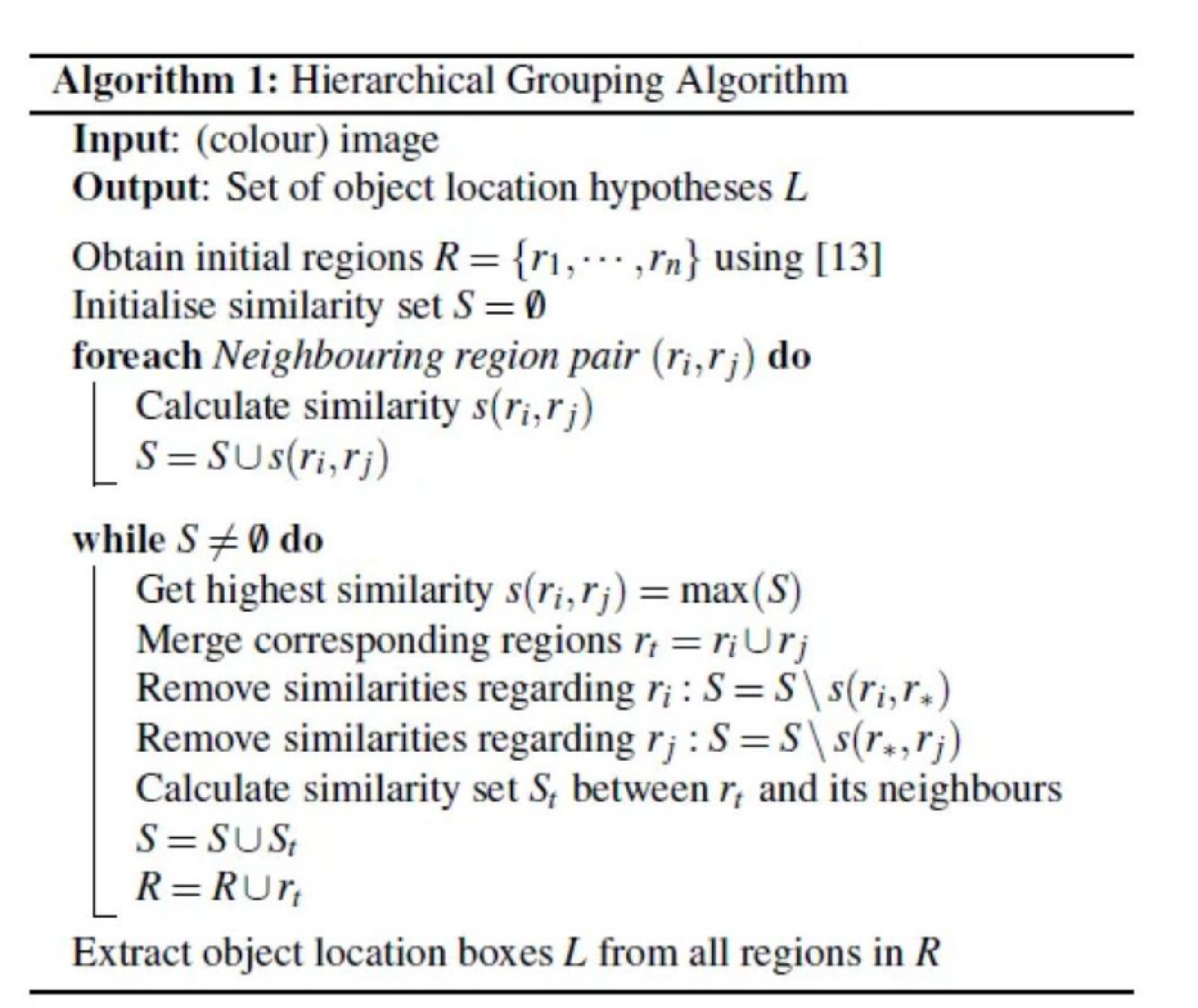


Image from Jasper RR Uijlings [1]

Region Proposals (selective search)



```
Algorithm 1: Hierarchical Grouping Algorithm
Input: (colour) image
Output: Set of object location hypotheses $L$

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach Neighbouring region pair $(r_i, r_j)$ do
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ do
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set $S_t$ between $r_t$ and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes $L$ from all regions in $R$
```

Selective search 演算法流程

Feature Extractor (AlexNet)

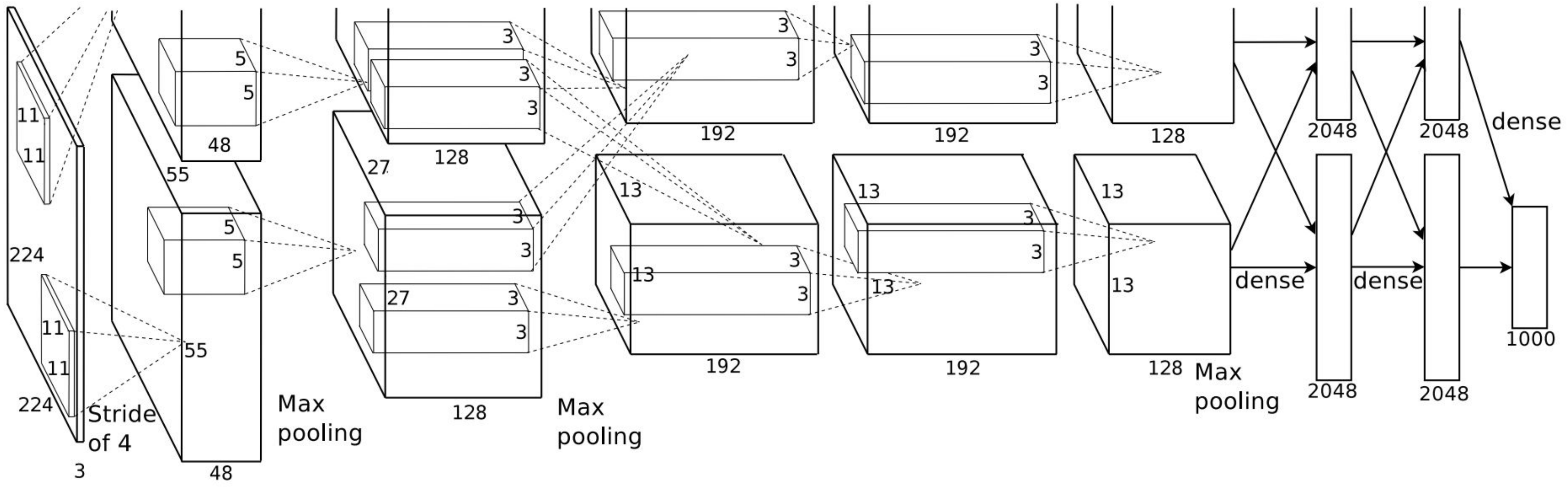
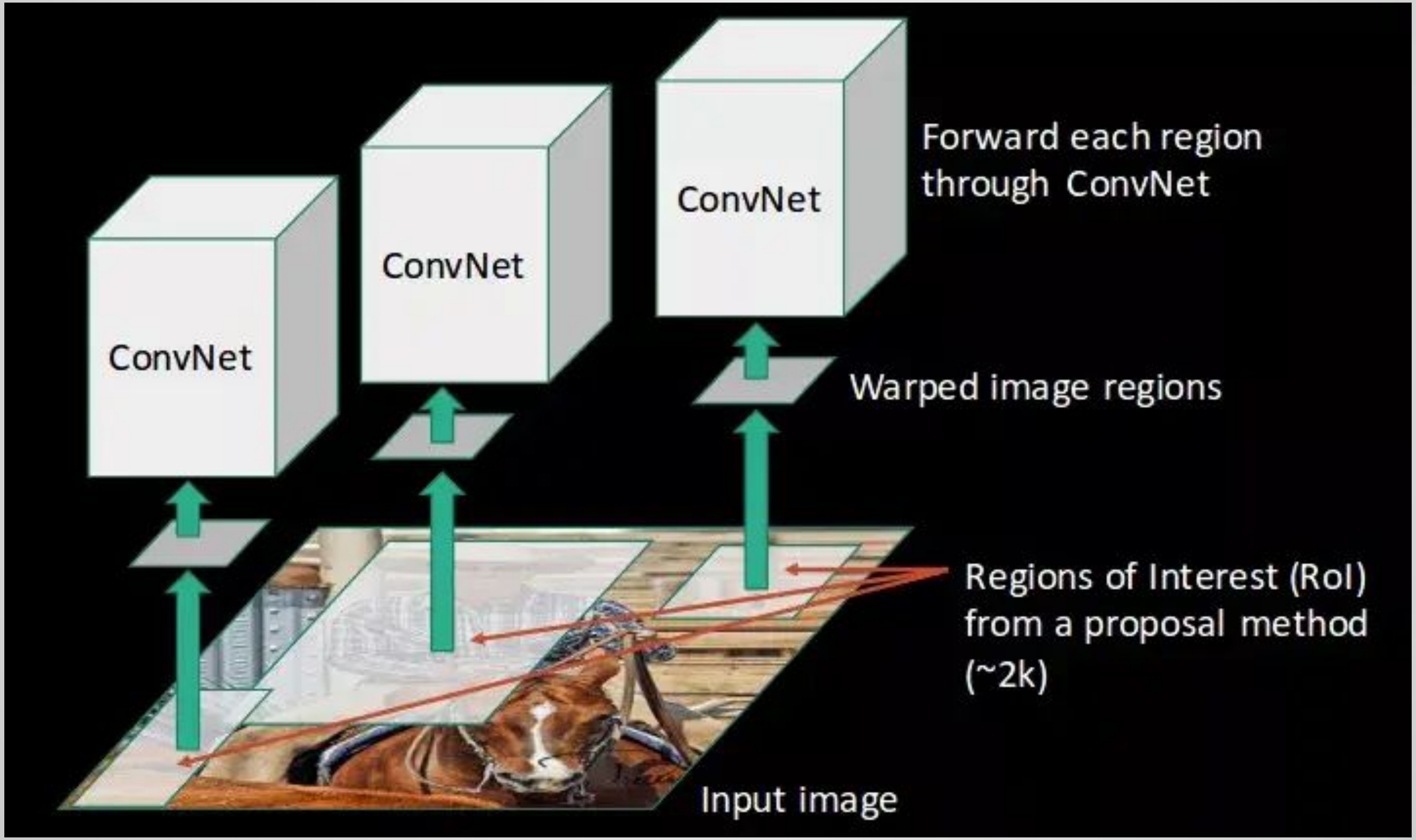


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Feature Extractor (AlexNet)

1. Replacing Sigmoid with ReLU in LeNet
2. Replacing Avg Pooling with Max Pooling
3. Use Dropout in sixth and seventh to replace Fully Connected NN

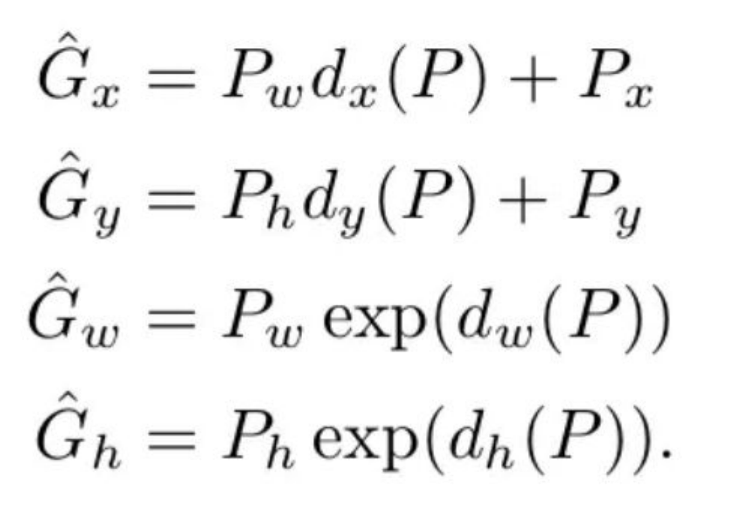
Feature Extractor (AlexNet)


ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

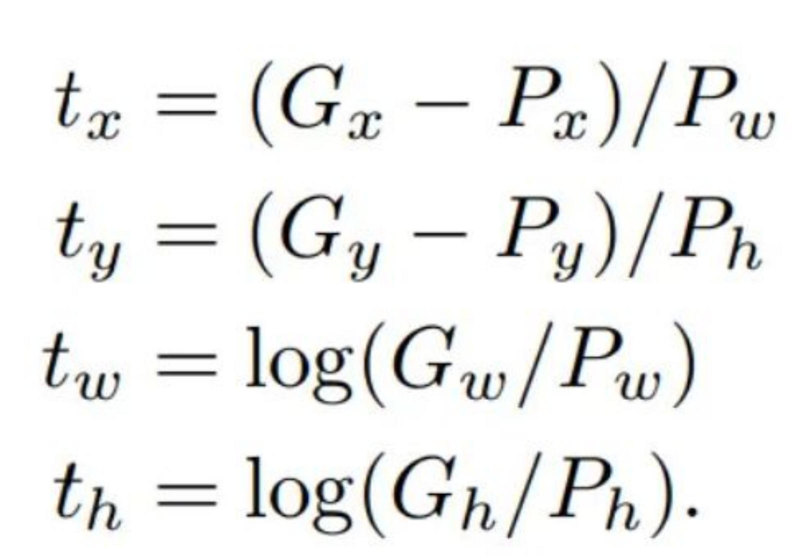
Detector (SVM classifier, Bbox regressor)

1. SVM classifier -> classification
2. Bbox regressor -> regression

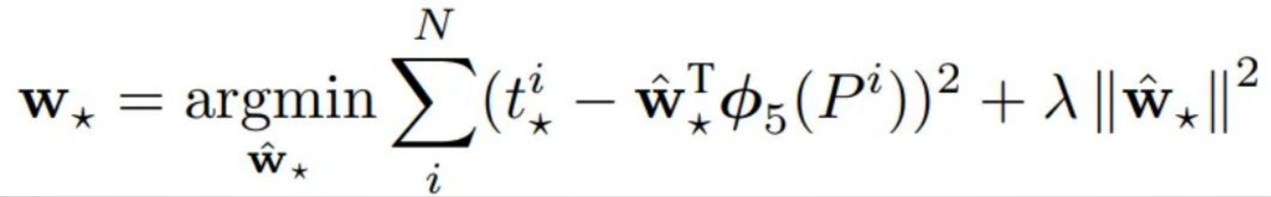
Detector (SVM classifier, Bbox regressor)



$$
\begin{aligned}
\hat{G}_x &= P_w d_x(P) + P_x \\
\hat{G}_y &= P_h d_y(P) + P_y \\
\hat{G}_w &= P_w \exp(d_w(P)) \\
\hat{G}_h &= P_h \exp(d_h(P)).
\end{aligned}
$$

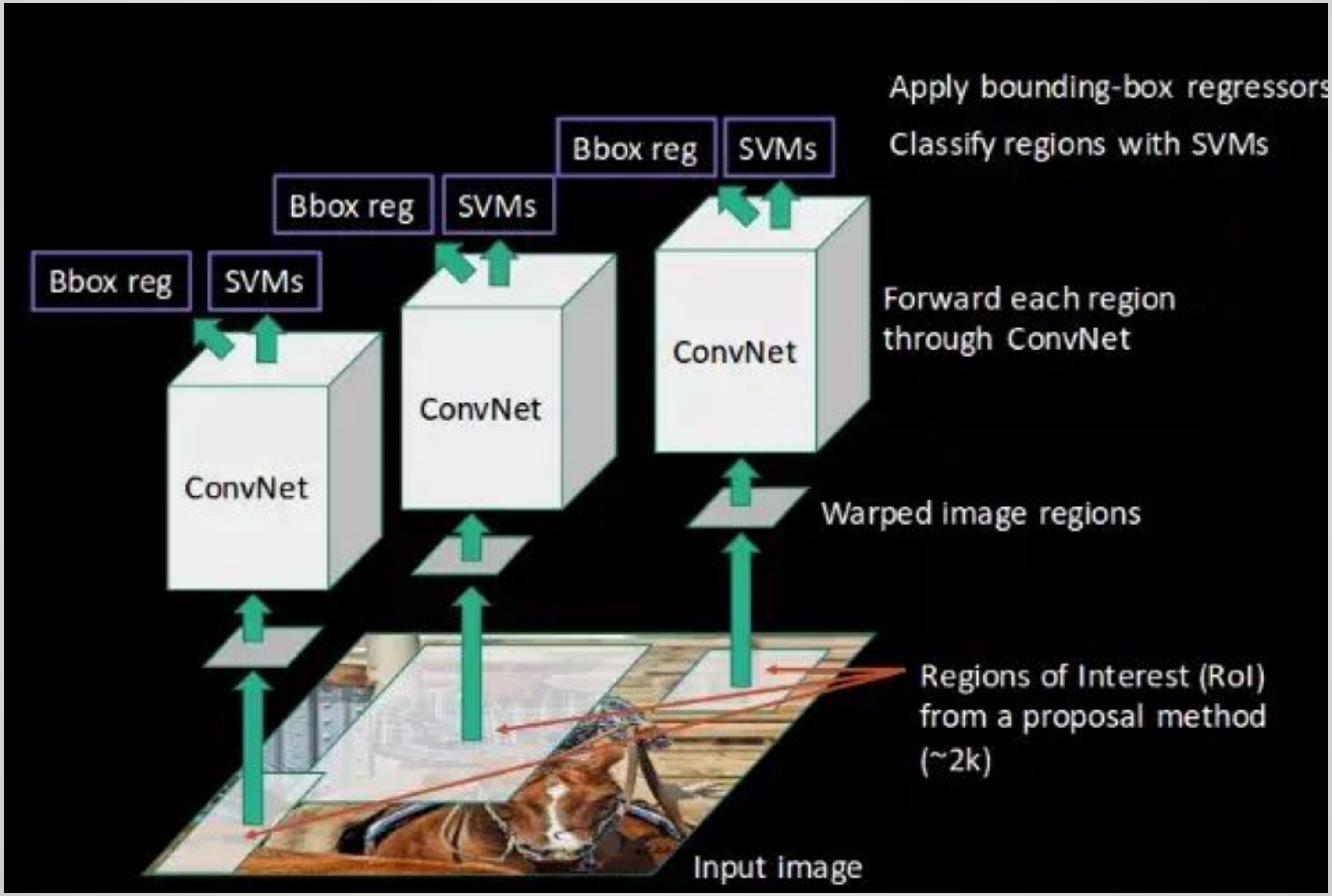


$$
\begin{aligned}
t_x &= (G_x - P_x)/P_w \\
t_y &= (G_y - P_y)/P_h \\
t_w &= \log(G_w/P_w) \\
t_h &= \log(G_h/P_h).
\end{aligned}
$$



$$
\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \left\| \hat{\mathbf{w}}_\star \right\|^2
$$

Detector (SVM classifier, Bbox regressor)


Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

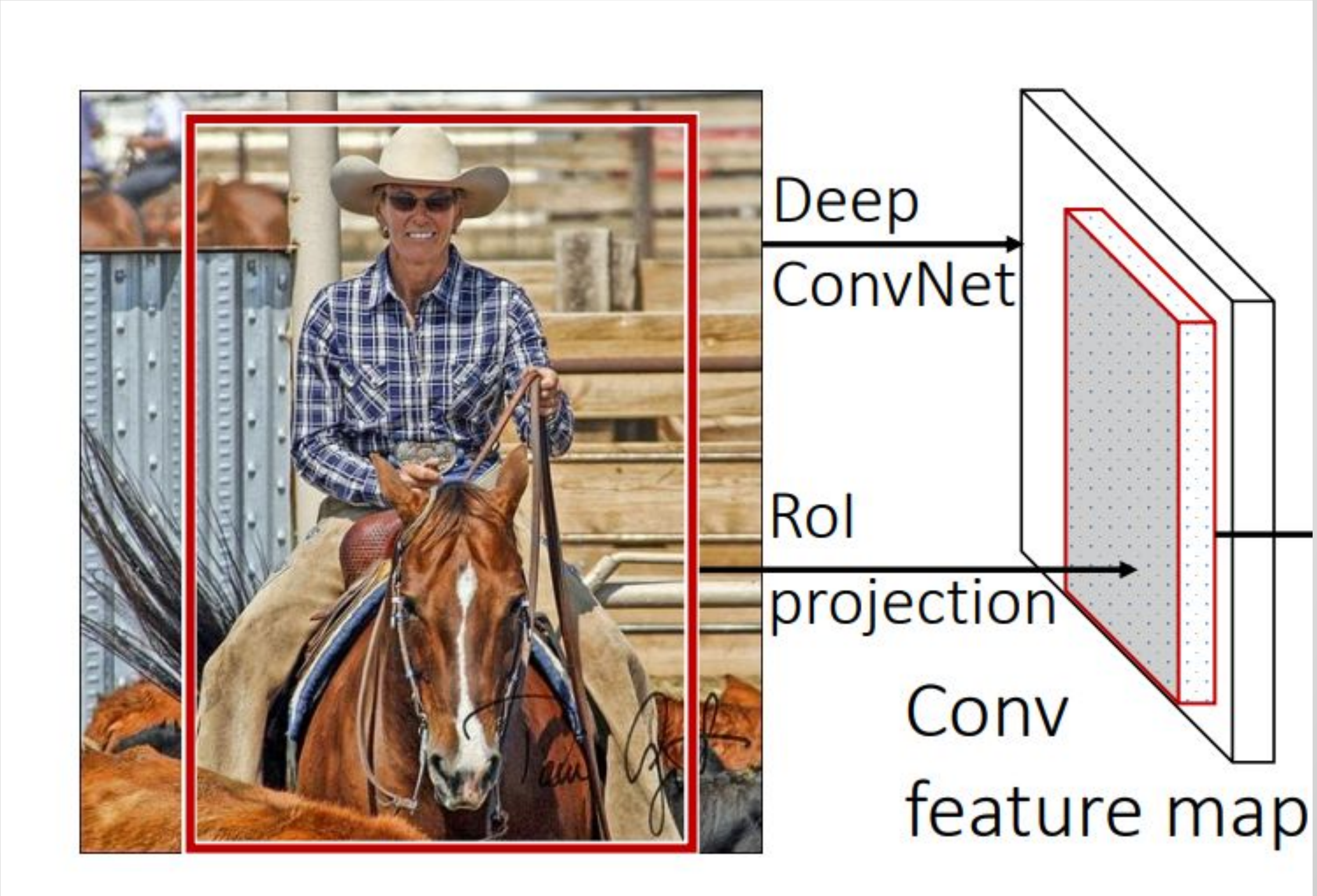
Shortcoming

1. Need large memory to store feature vector of proposal
2. Many times convolution training
3. 47 seconds per detection

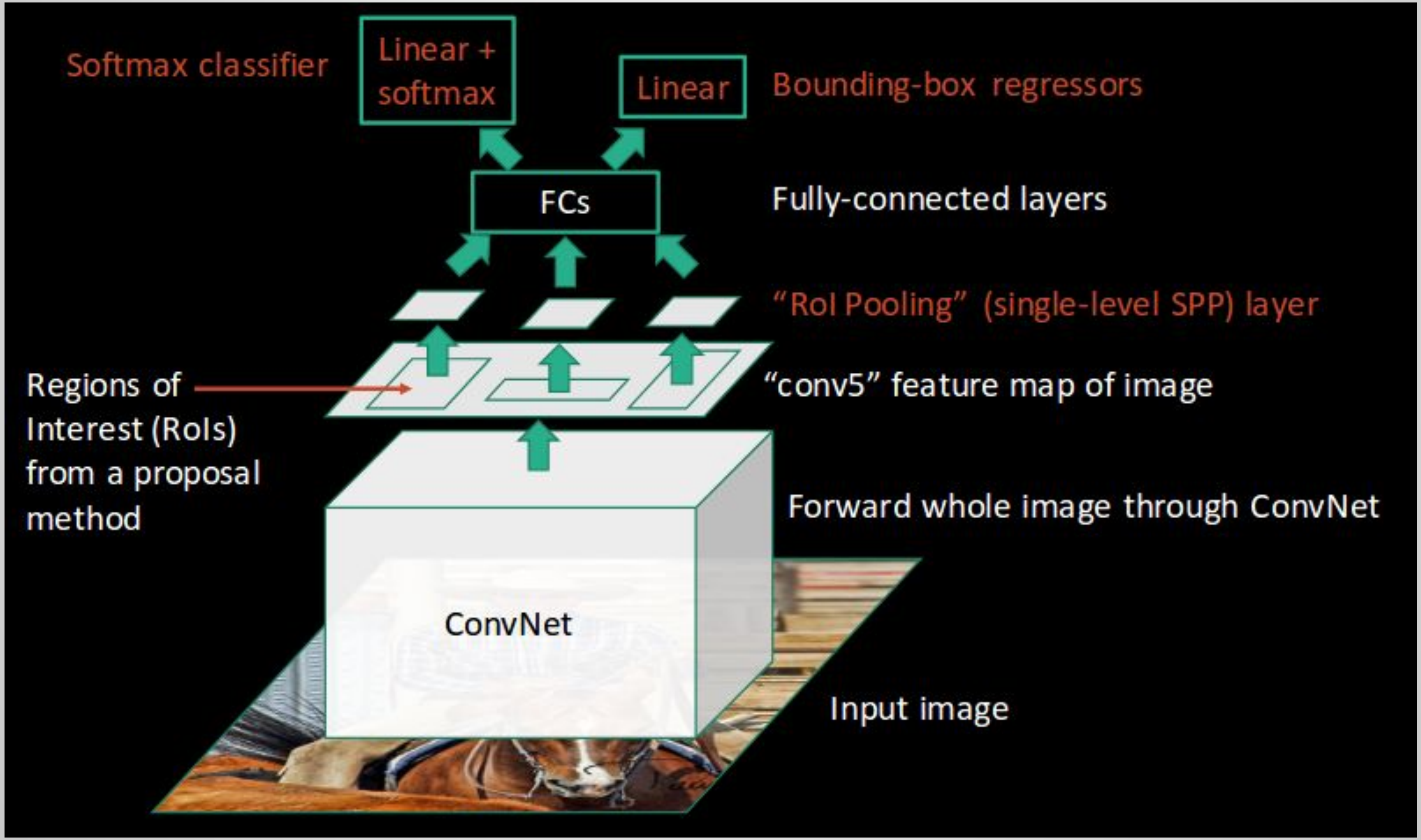
Fast R-CNN

1. Pass the entire image through convolution to generate a feature map.
2. Utilize ROI pooling layer to standardize dimensions.
3. Replace SVM with softmax for classification in the neural network.

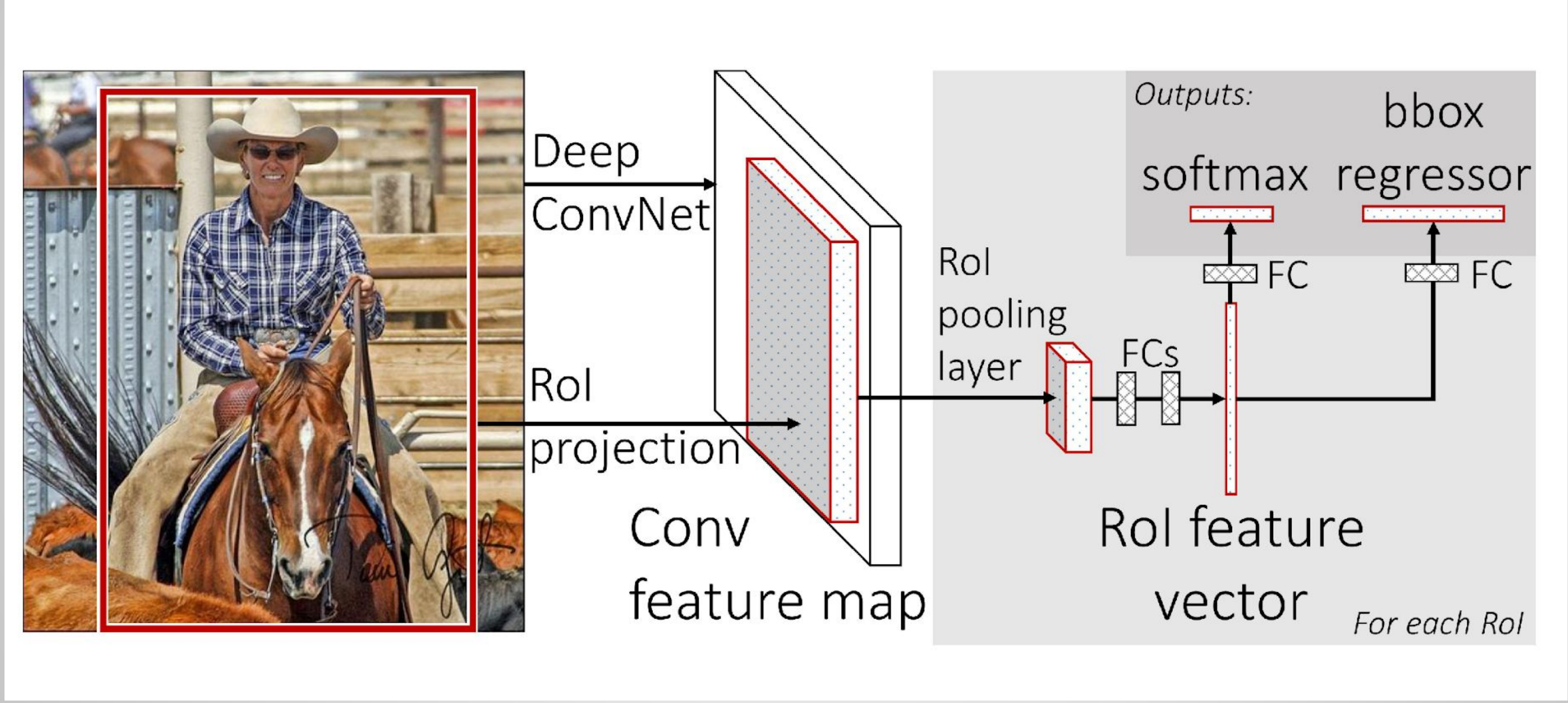
Region of Interest (selective search)



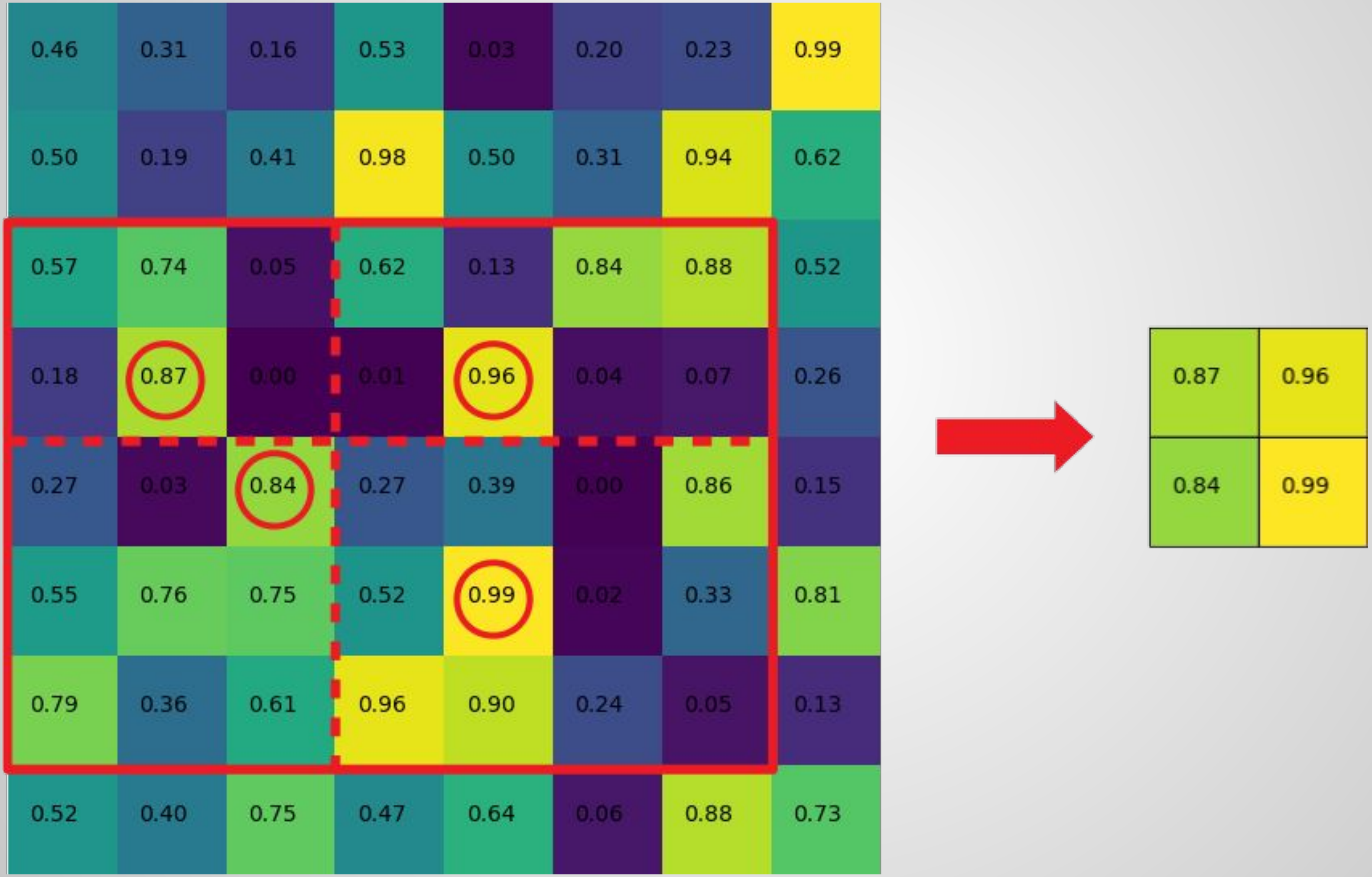
Region Proposals (selective search)


Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Feature Extractor (VGG net)



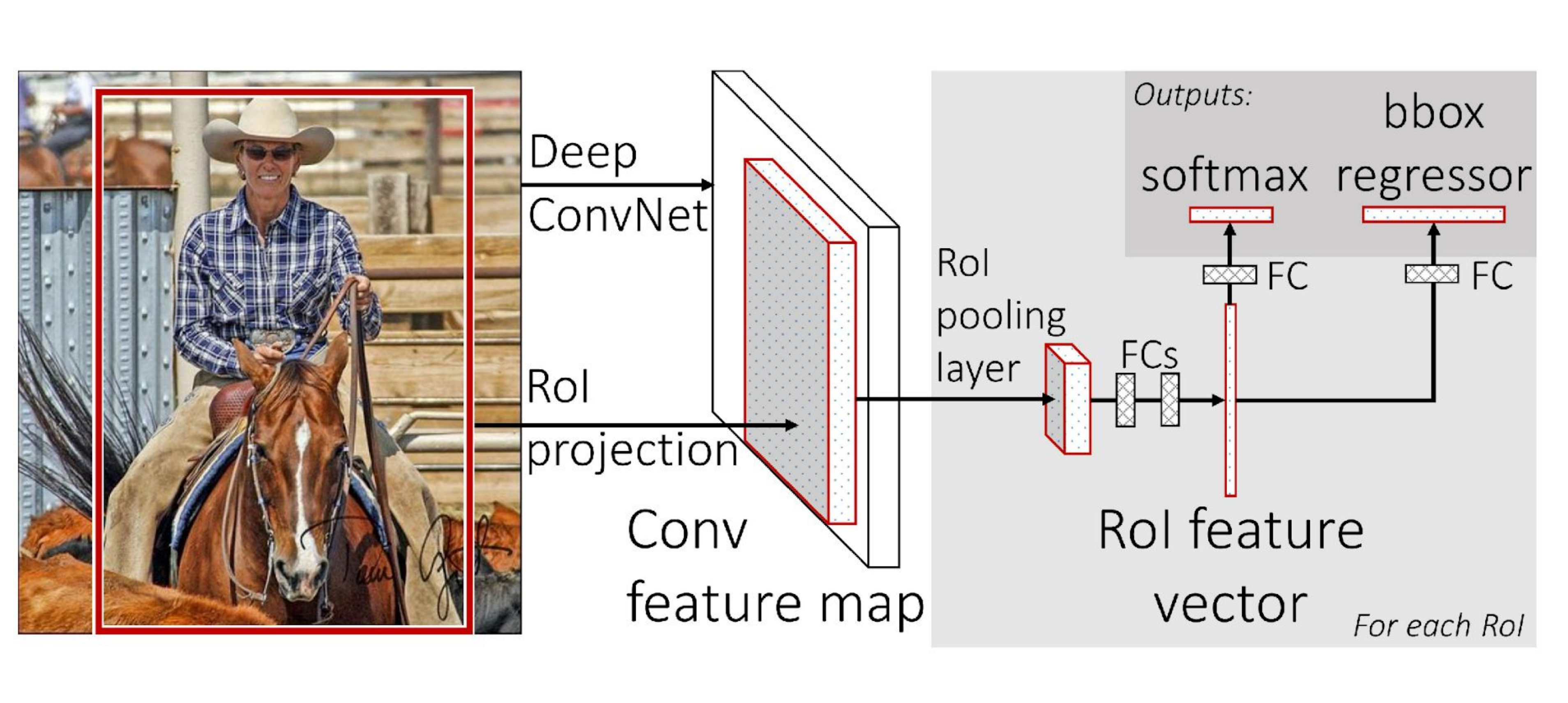
ROI Pooling Layer



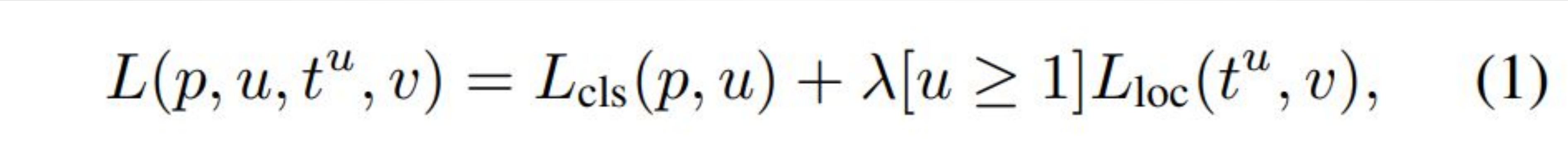
0.46	0.31	0.16	0.53	0.03	0.20	0.23	0.99
0.50	0.19	0.41	0.98	0.50	0.31	0.94	0.62
0.57	0.74	0.05	0.62	0.13	0.84	0.88	0.52
0.18	0.87	0.00	0.01	0.96	0.04	0.07	0.26
0.27	0.03	0.84	0.27	0.39	0.00	0.86	0.15
0.55	0.76	0.75	0.52	0.99	0.02	0.33	0.81
0.79	0.36	0.61	0.96	0.90	0.24	0.05	0.13
0.52	0.40	0.75	0.47	0.64	0.06	0.88	0.73

0.87	0.96
0.84	0.99

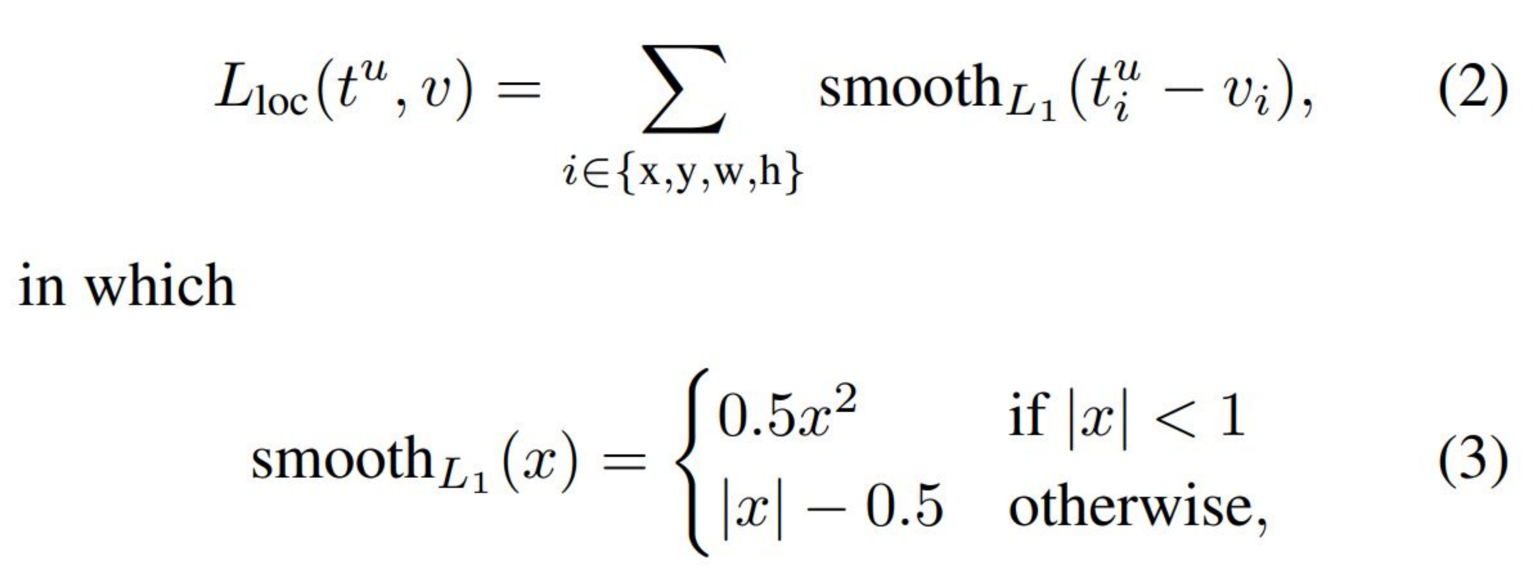
Detector


Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
RoI feature
vector
For each RoI
FC
FC
Outputs:
softmax
bbox
regressor

Loss Function



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



$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i), \qquad (2)$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

Shortcoming

1. 25 times faster, but still too slow
2. Region Proposal bottle neck

Faster R-CNN

1. Use Anchor
2. Replace Selective Search with Region Proposal Network

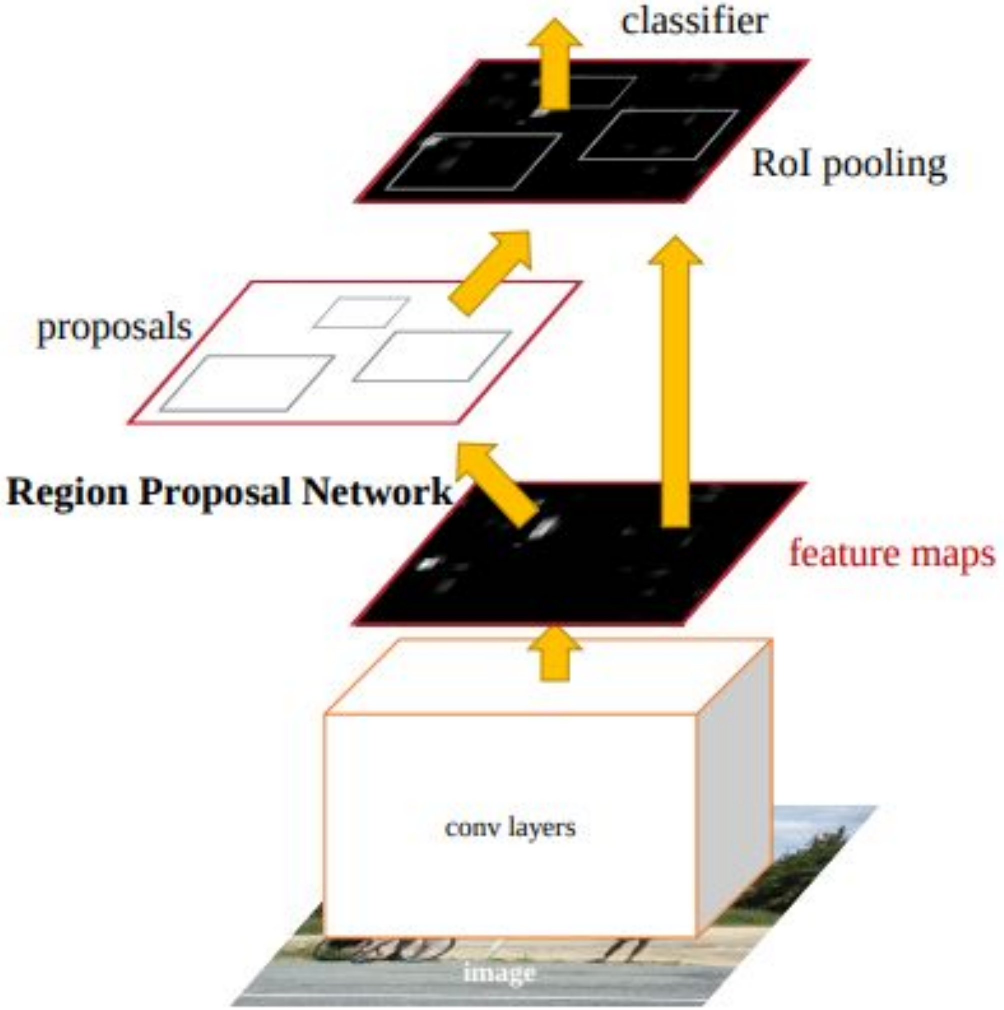
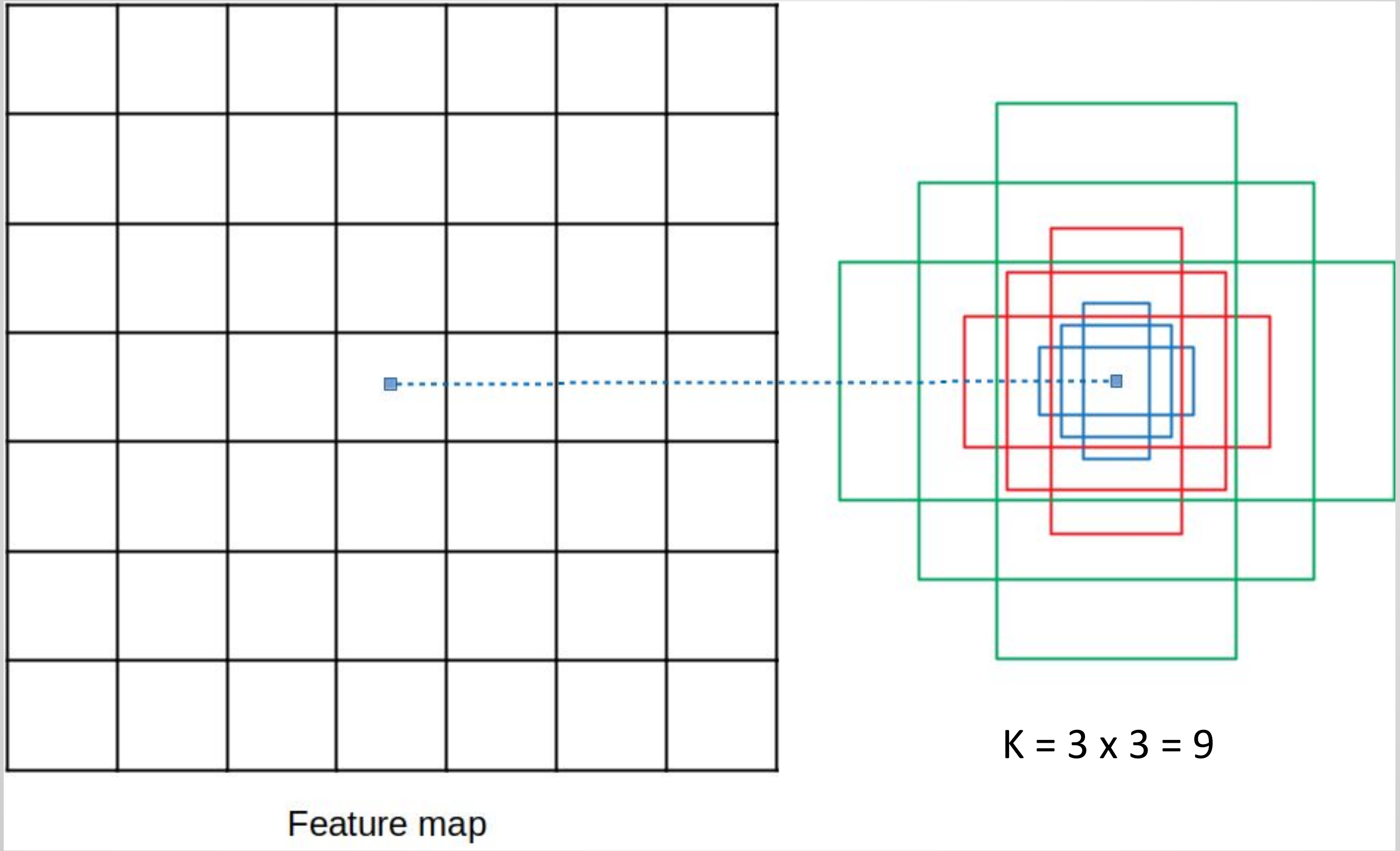


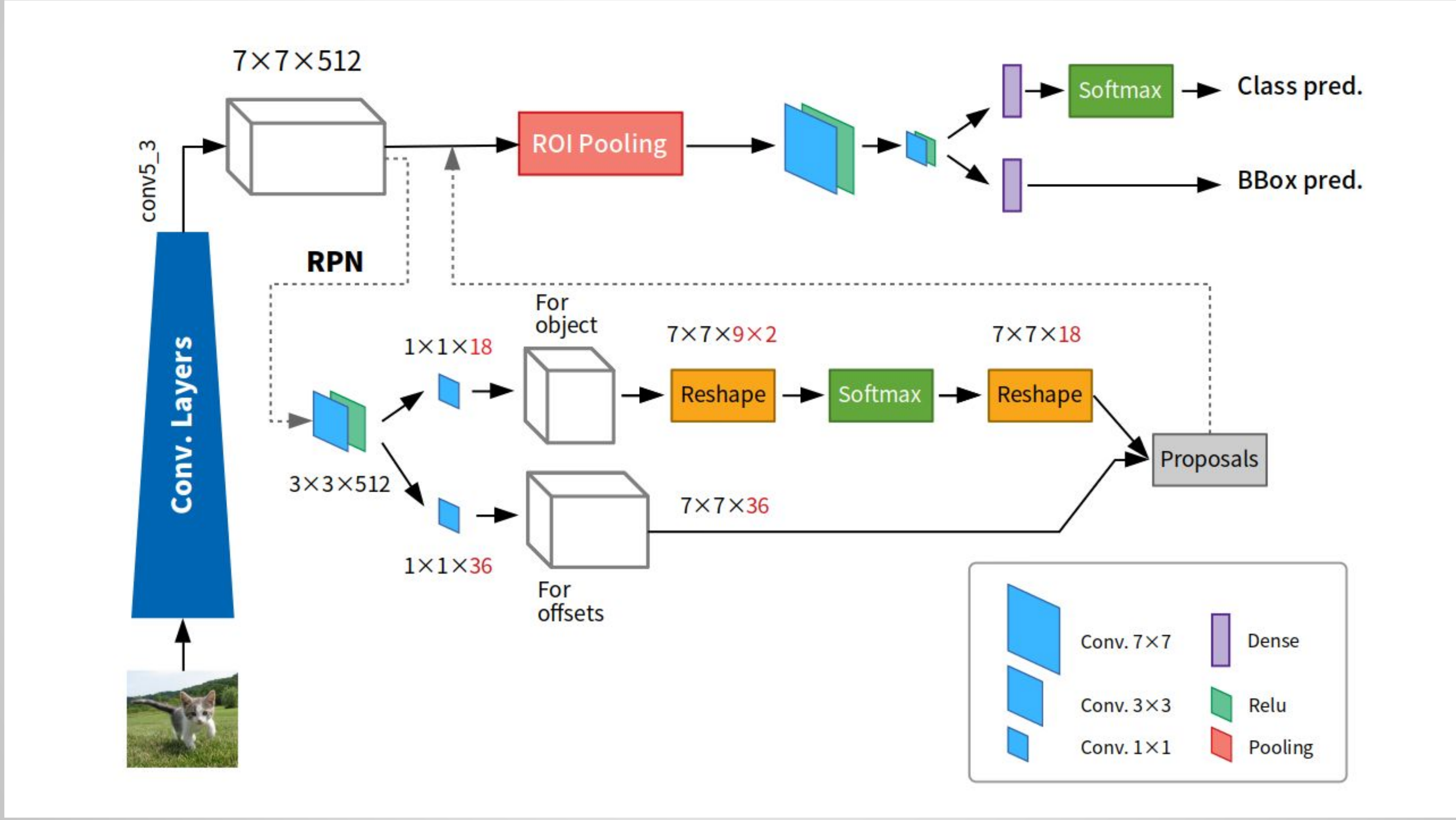
Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Anchor

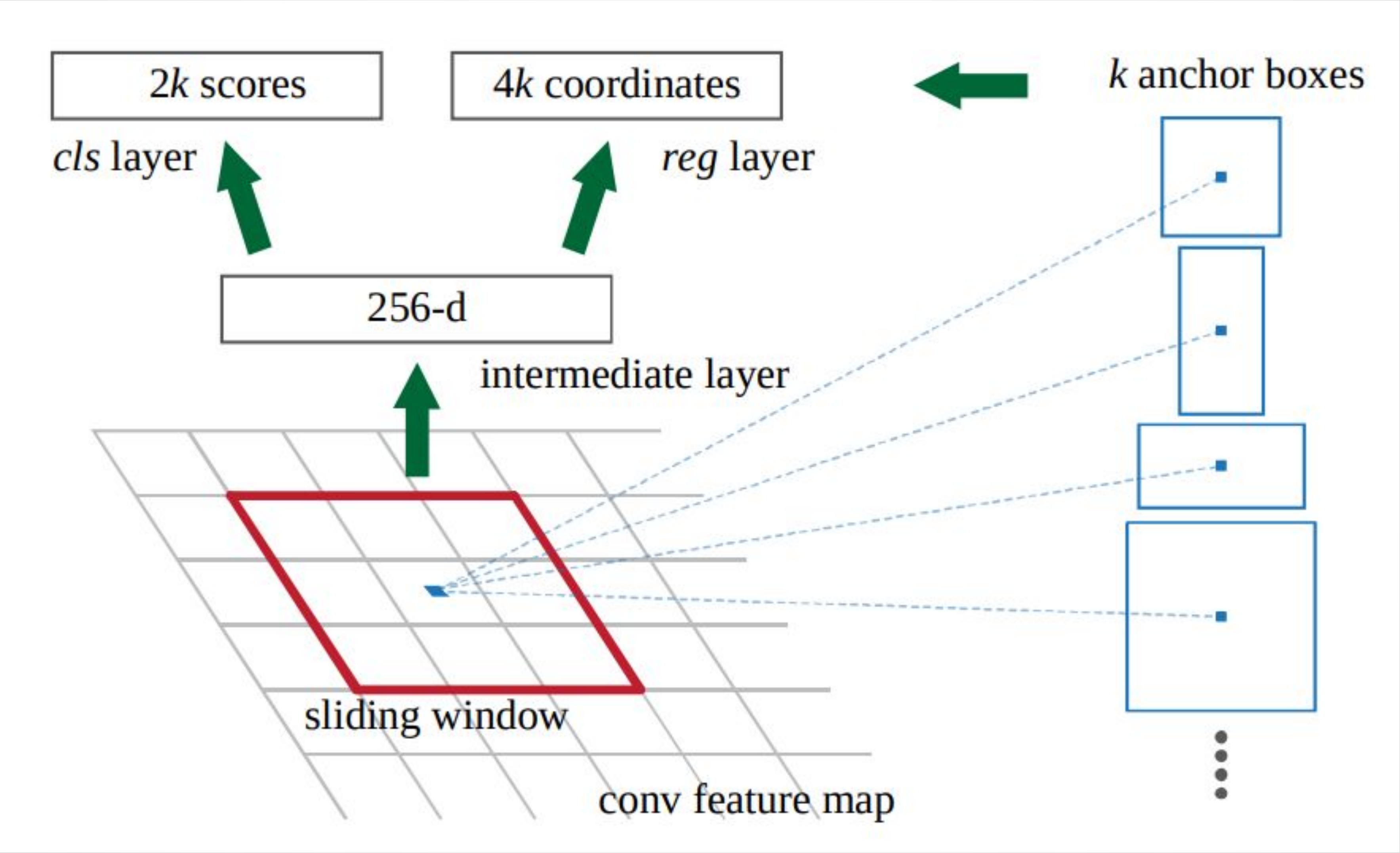


K = 3 x 3 = 9

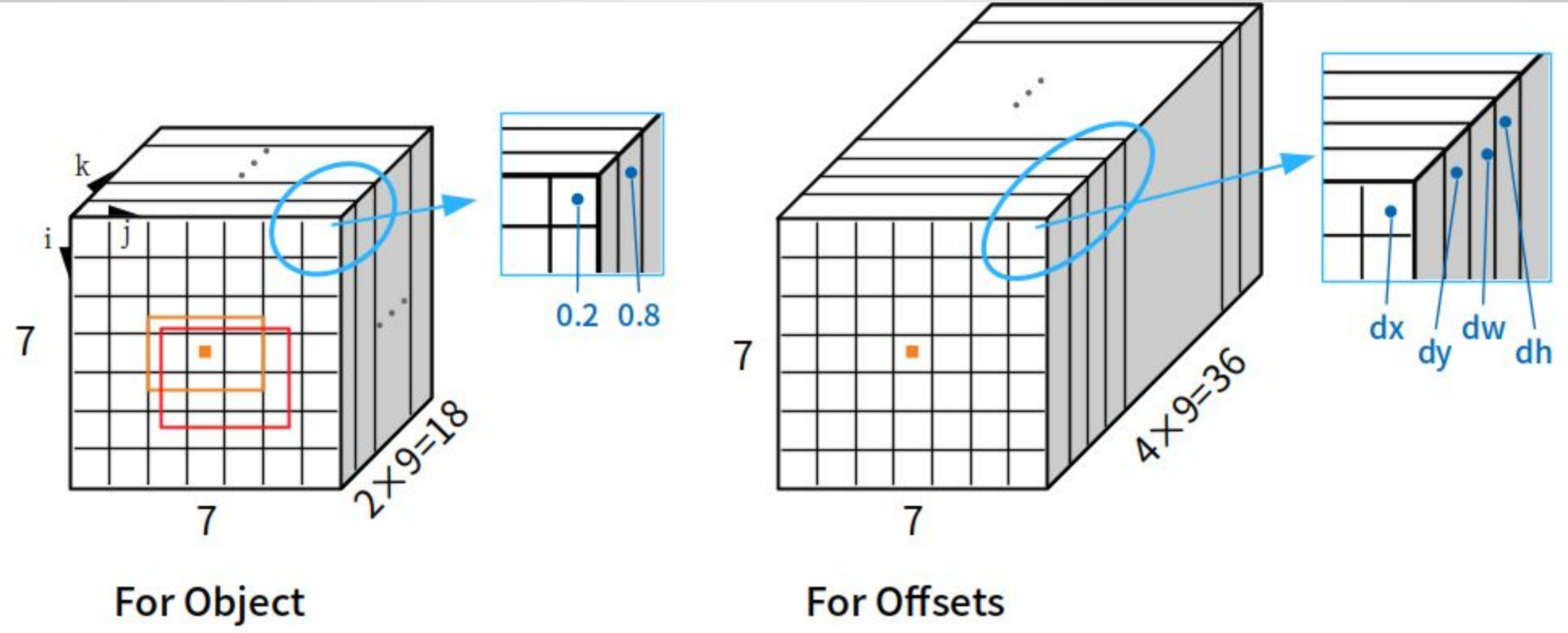
Region Proposal Network (RPN)



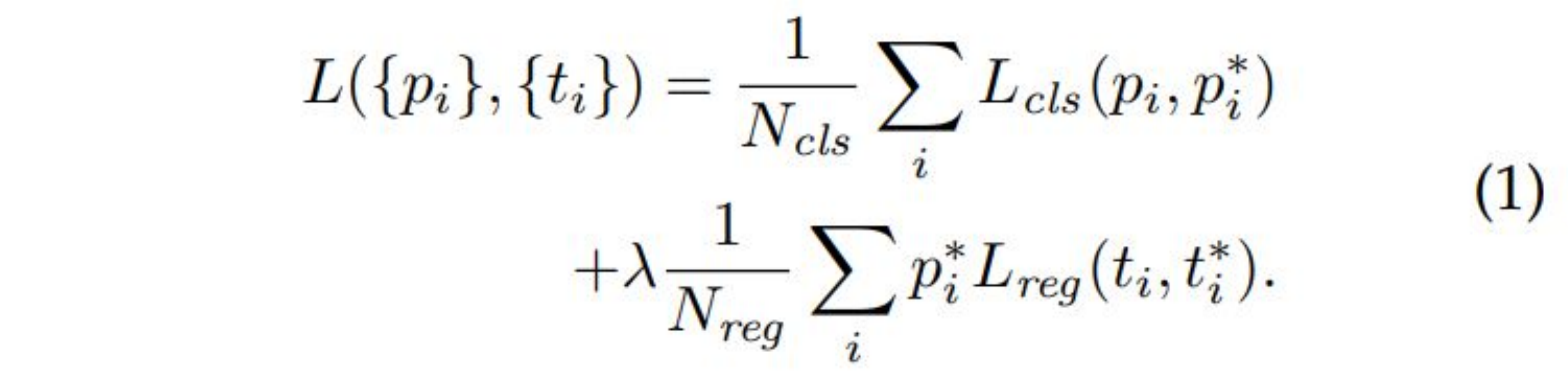
Region Proposal Network (RPN)



Region Proposal Network (RPN)

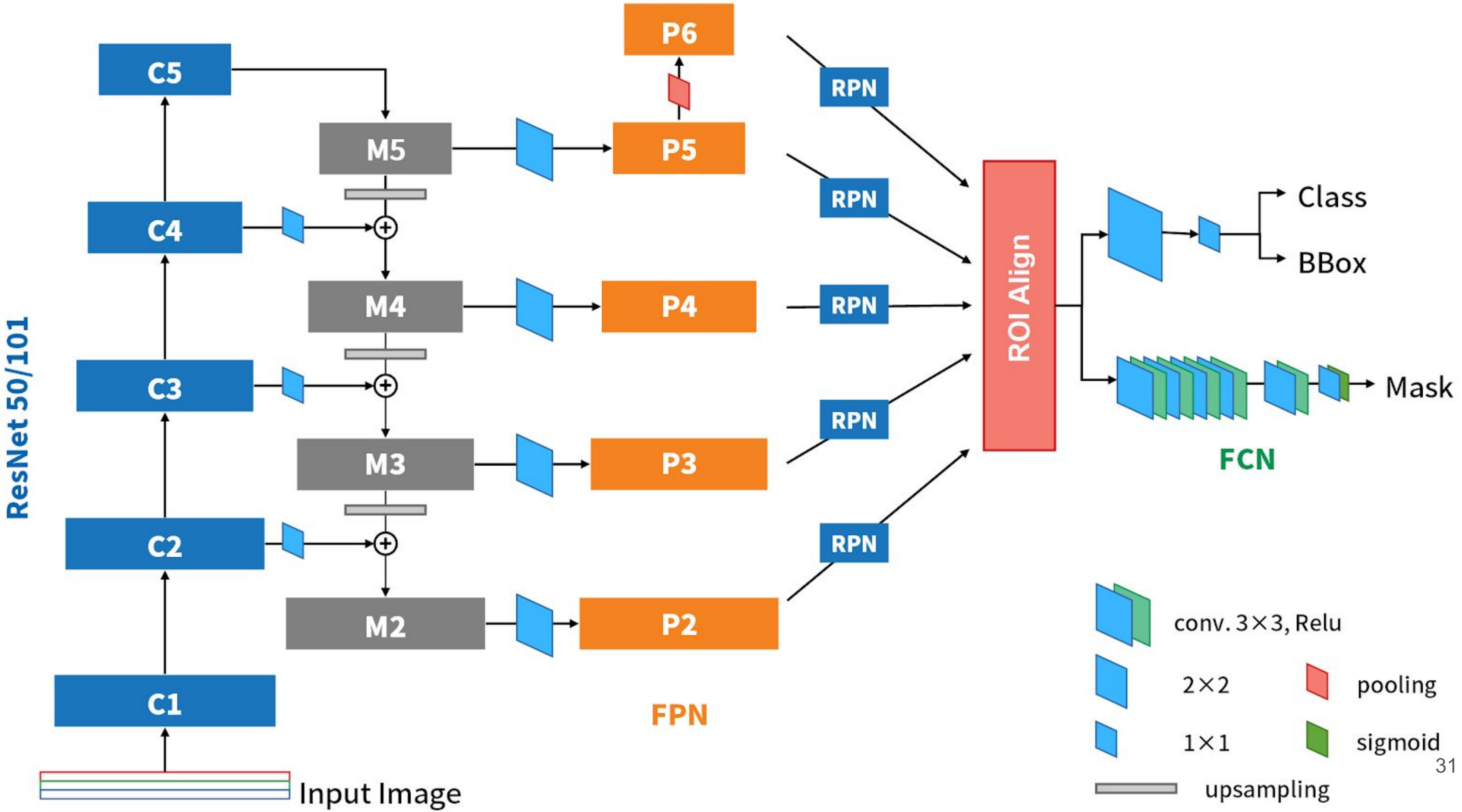


RPN Loss Function

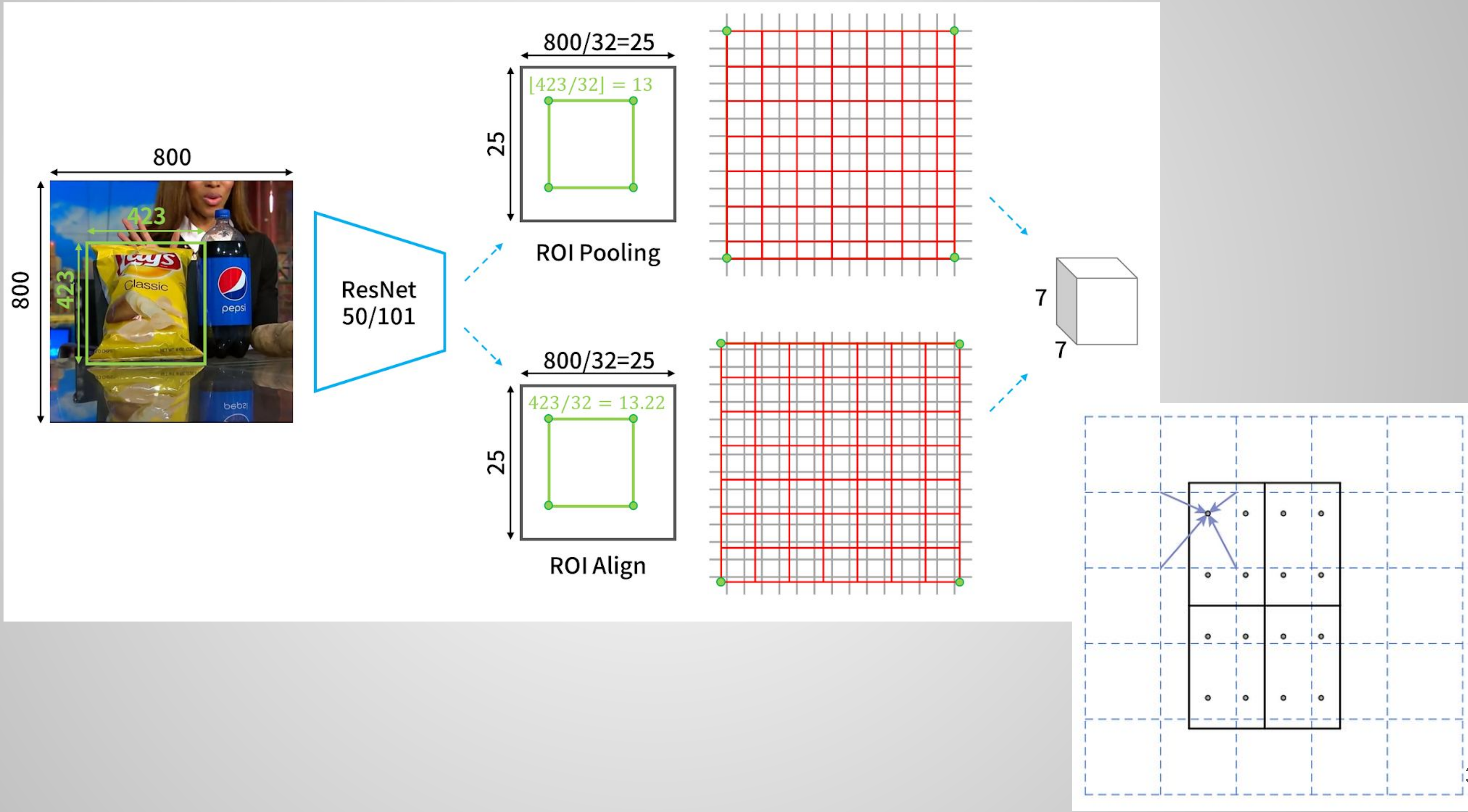


$$\begin{aligned} L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) \\ + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*). \end{aligned} \tag{1}$$

Mask R-CNN

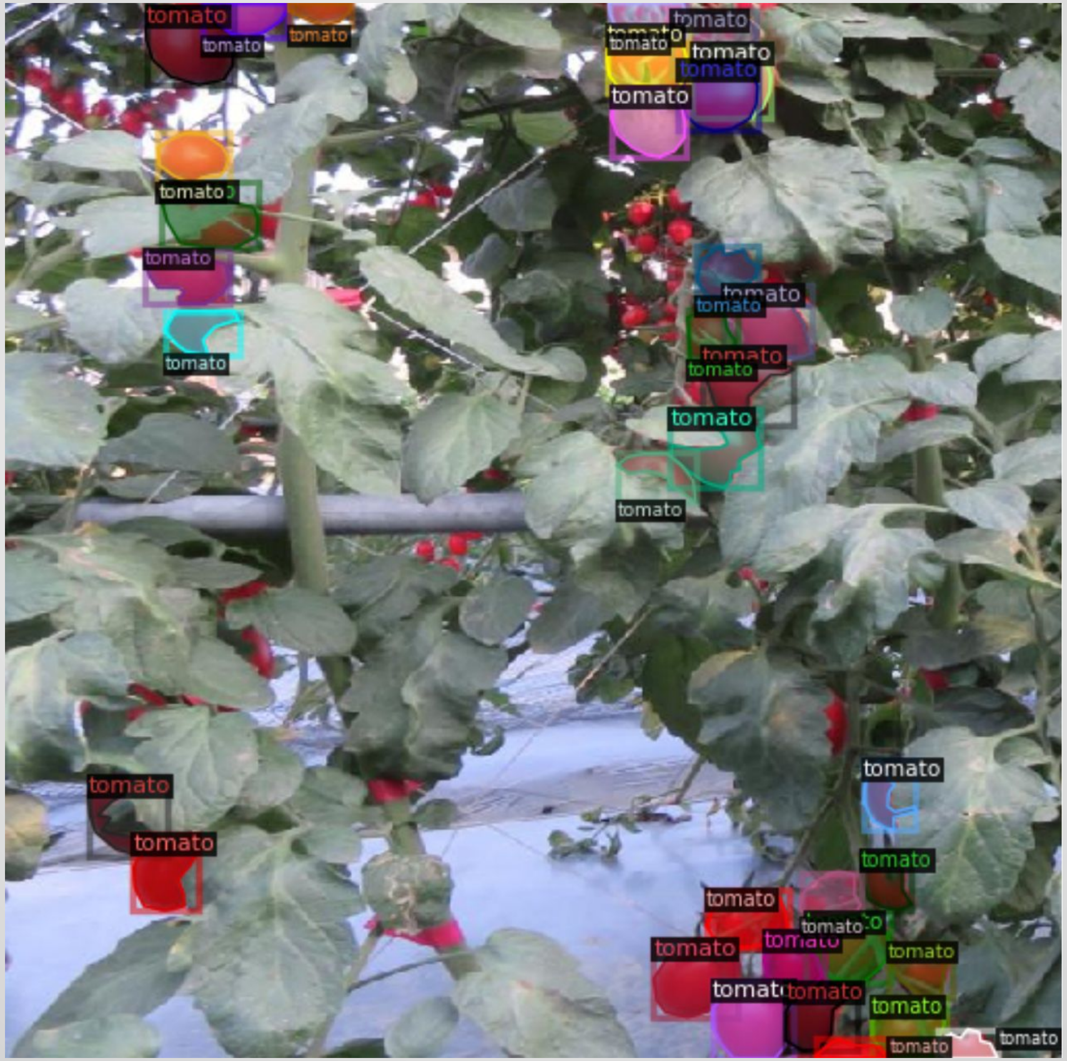

ResNet 50/101
C1
C2
C3
C4
C5
Input Image
M2
M3
M4
M5
P2
P3
P4
P5
P6
FPN
RPN
RPN
RPN
RPN
RPN
ROI Align
Class
BBox
Mask
FCN
conv. 3×3, Relu
2×2
pooling
1×1
sigmoid
upsampling

ROI Align


800
800
423
423
ResNet 50/101
800/32=25
25
⌊423/32⌋ = 13
ROI Pooling
800/32=25
25
423/32 = 13.22
ROI Align
7
7

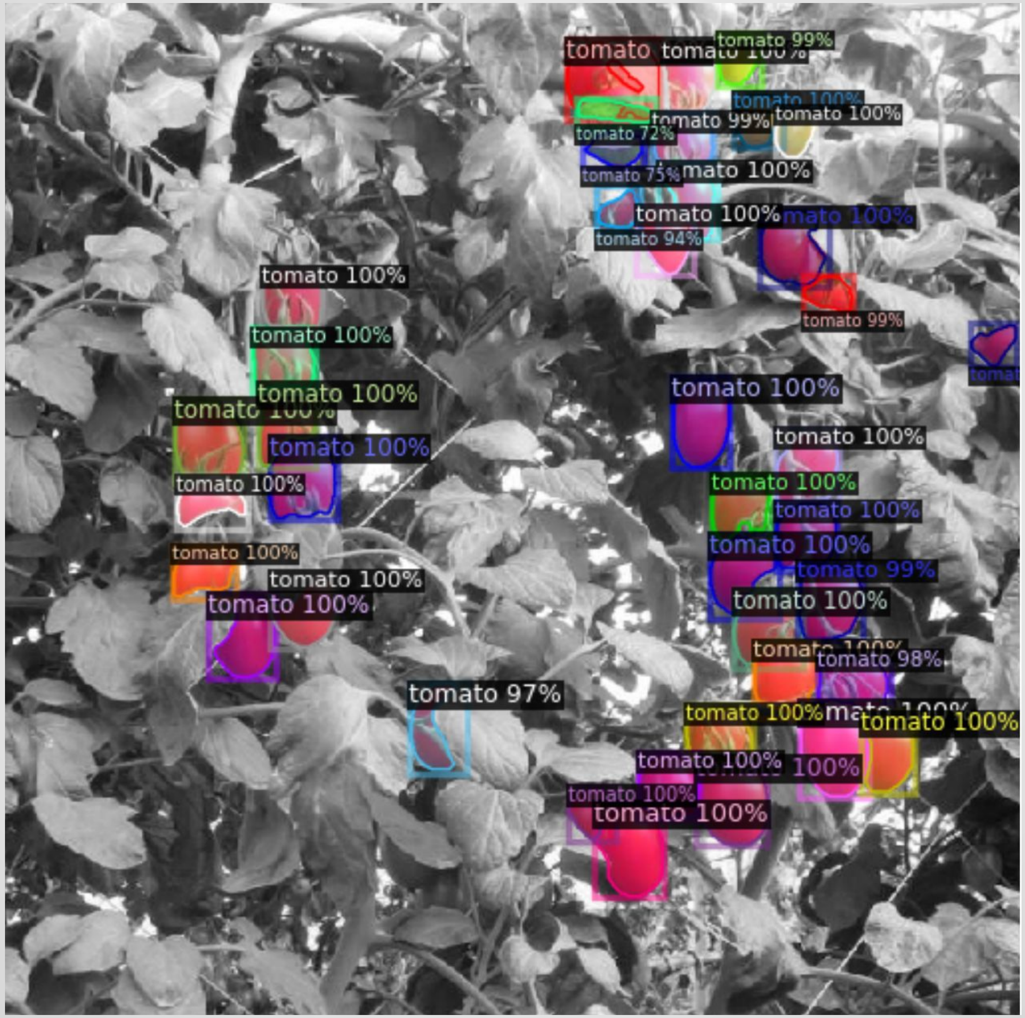
Detection
Tomato

Label

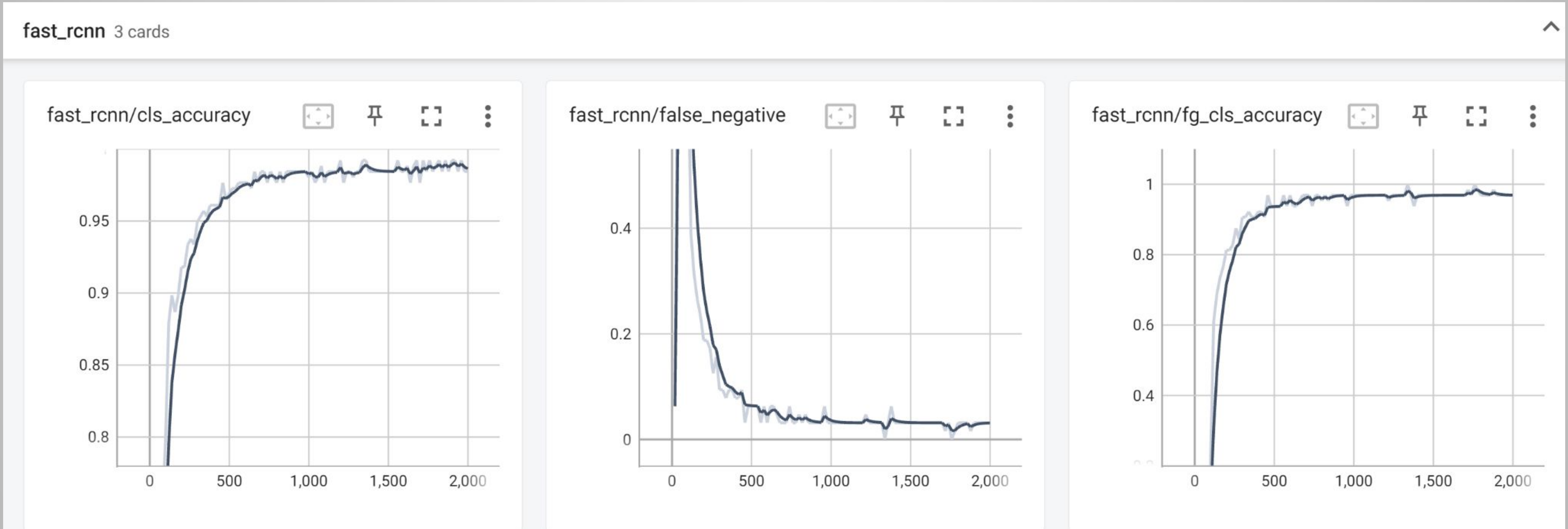


Detection Tomato

Predict



Detection
Tomato



Code :

https://colab.research.google.com/drive/1uy_55Ziq3U-FBIOlv-v8A2Hy3od4rWyr?usp=sharing

Reference :

https://github.com/facebookresearch/detectron2
https://arxiv.org/abs/1703.06870

For more information about Tomato Detection Demo, ask us in QA time :

https://hackmd.io/@2peEy1j8QSygBgdKIEZpJQ/HkIsfQih2