Convolutional Neural Networks for Detection

Giacomo Boracchi

Advanced Neural Networks and Deep Learning

https://boracchi.faculty.polimi.it/

Localization

The Localization Task

The input image contains a single relevant object to be classified in a fixed set of categories

The task is to:

1) assign the object class to the image

hawk

The Localization Task

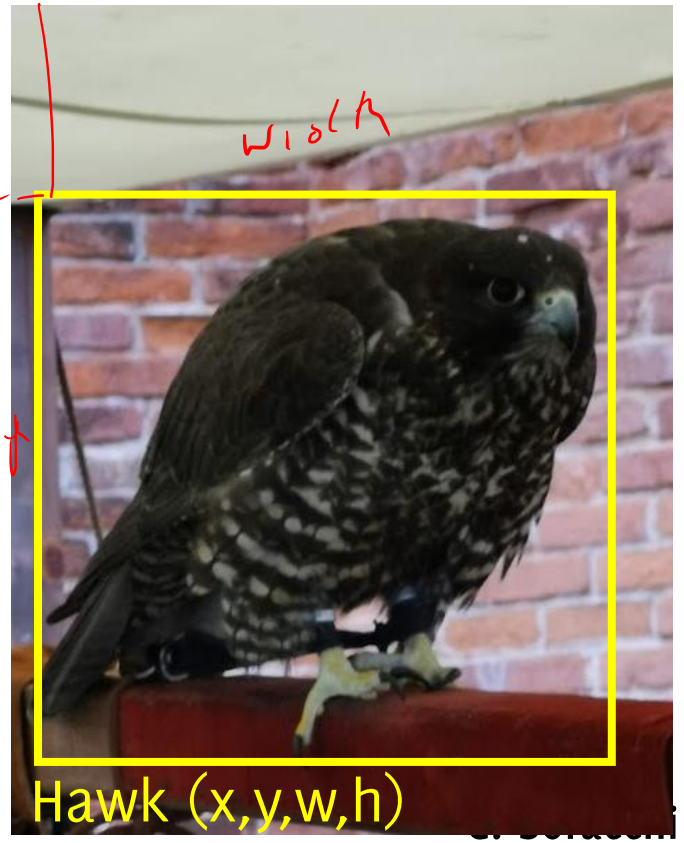
The input image contains a single relevant object to be classified in a fixed set of categories

The task is to:

1) assign the object class to the image
2) locate the object in the image by its bounding box

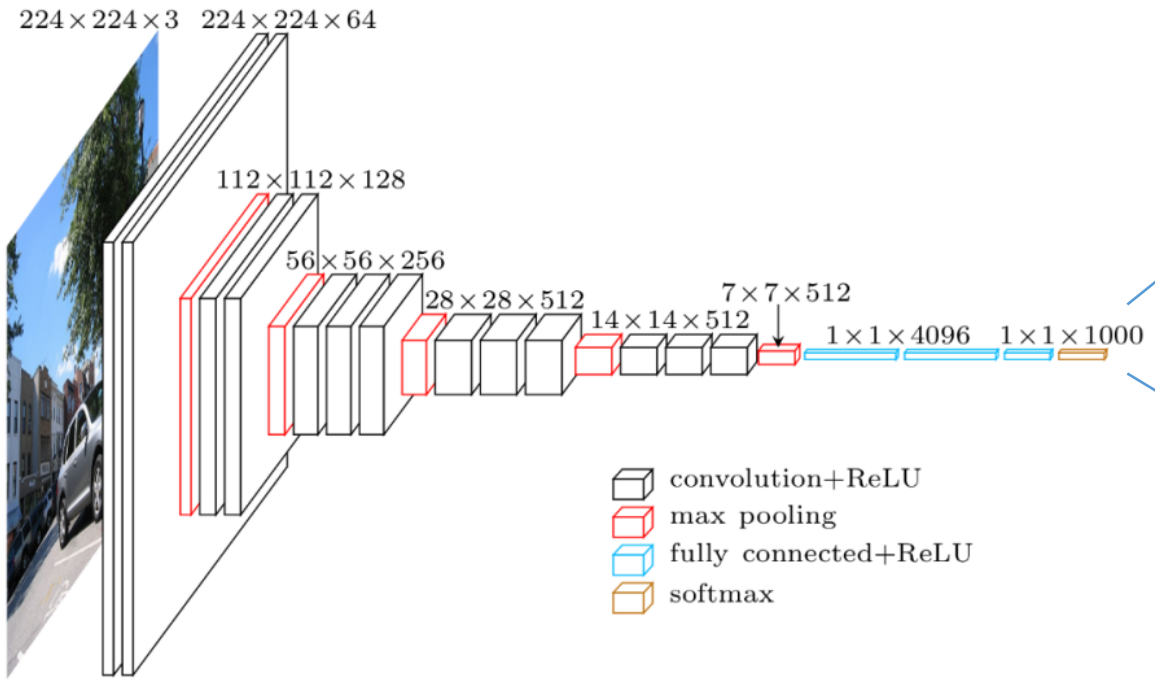
A training set of annotated images with **label and a bounding box** around each object is required

Extended localization problems involve regression over more complicated geometries (e.g. human skeleton)



The Simplest Solution

Train a network to predict both the class label and the bounding box



Target class
Soft-max loss $\mathcal{S}$

$\| [\hat{x}, \hat{y}, \hat{w}, \hat{h}] - [x, y, w, h] \|_2^1$

Bounding Box Coordinates
(x, y, w, h)
Regression loss $\mathcal{R}$, e.g. the $\ell^2, \ell^1, ..$ bounding box.

G. Boracchi

The Simplest Solution

The training loss has to be a single scalar since **we compute gradient of a scalar function** with respect to network parameters.

So one tend to **minimize a multitask loss to merge two losses**:

$$\mathcal{L}(x) = \alpha\, \mathcal{S}(x) + (1 - \alpha)\mathcal{R}(x)$$

and α is an hyper parameter of the network.

Watch out that $\boldsymbol{\alpha}$ **directly influences the loss definition**, so tuning might be difficult, better to do cross-validation looking at some other loss.

It is also possible to **adopt a pre-trained model** and then train the two FC separately… however it is always better to perform at least some fine tuning to train the two jointly.

Extension to Human Pose Estimation

Pose estimation is formulated as a **CNN-regression problem towards body joints.**

This image is licensed under CC-BY 2.0

Represent pose as a set of 14 joint positions:

Left / right foot
Left / right knee
Left / right hip
Left / right shoulder
Left / right elbow
Left / right hand
Neck
Head top

28-dimensional regression problem

Alexander Toshev and Christian Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

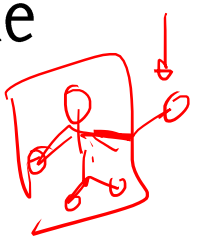
Extension to Human Pose Estimation

Pose estimation is formulated as a **CNN-regression problem towards body joints.**

- **The network receives as input the whole image, capturing the full-context of each body joints.**
- The approach is **very simple to design and train.** Training problems can be **alleviated by transfer learning** of existing classification networks

Pose is defined as a vector of k joints location for the human body, possibly normalized w.r.t. the bounding box enclosing the human

Train a CNN to predict **a $2k$ vector as output** by using an Alexnet-like architecture



Alexander Toshev and Christian Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

Training Human Pose Estimation Networks

Adopt a ℓ^2 regression loss of the estimated pose parameters over the annotations.

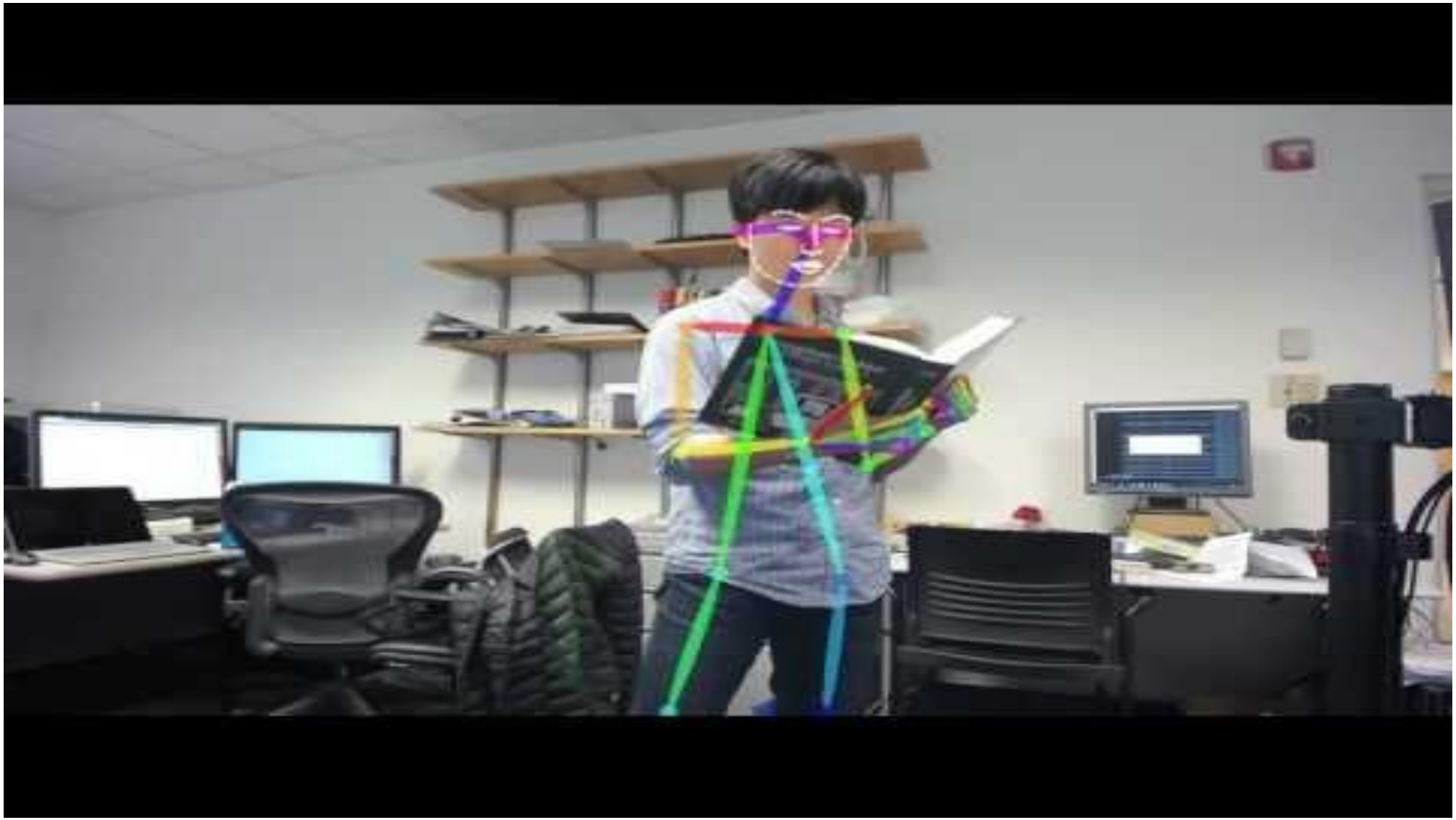
- This can be also defined when a few joints are not visible

Reduce overfitting by augmentation (translation and flips)

Multiple networks have been trained to improve localization by refining joint localtions in a crop around the previous detection

Alexander Toshev and Christian Szegedy, “DeepPose: Human Pose Estimation via Deep Neural Networks”, CVPR 2014

Open Pose



Cao, Z et al.. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. CVPR 2017

Pose Estimation

Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. CVPR2017

Weakly-Supervised Localization

… Global Averaging Pooling Revisited

… visualizing what matters most for CNN predicitons

G. Boracchi

Weakly supervised localization

Perform localization over an image without images with annotated bounding box

- Training set provided as for classification with image-label pairs $\{(I, \ell)\}$ where no localization information is provided

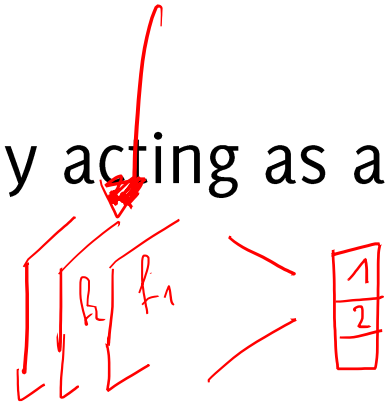
Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
Computer Science and Artificial Intelligence Laboratory, MIT
{bzhou,khosla,agata,oliva,torralba}@csail.mit.edu

G. Boracchi

The GAP revisited

The advantages of GAP layer **extend beyond** simply acting as a structural **regularizer** that prevents overfitting



In fact, the **network can retain a remarkable localization ability** until the final layer. By a simple tweak it is possible to easily identify the discriminative image regions leading to a prediction.

*A **CNN trained on object categorization** is successfully able to **localize the discriminative regions for action classification** as the objects that the humans are interacting with rather than the humans themselves*

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

Class Activation Mapping

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

Class Activation Mapping (CAM)

Identifying exactly **which regions** of an image **are being used for discrimination**.

CAM are very easy to compute. It just requires:

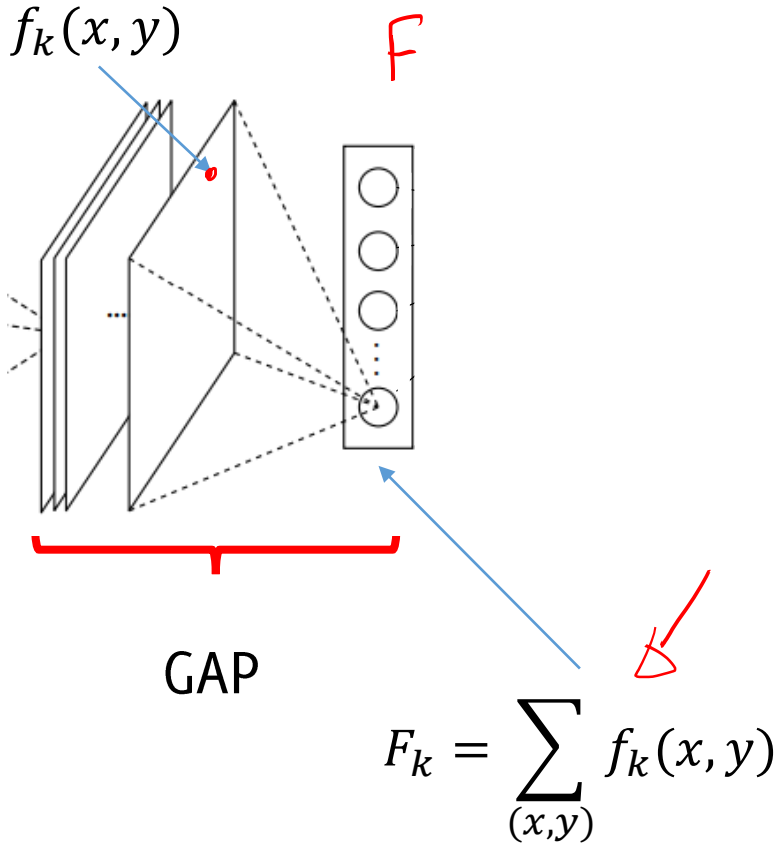
- FC layer after the GAP
- a minor tweak

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

The Global Averaging Pooling (GAP) Layer

A very simple architecture made only of convolutions and activation functions leads to a final layer having:

- n feature maps $f_k(\cdot,\cdot)$ having resolution "similar" to the input image
- a vector after GAP made of n averages F_k



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

The Global Averaging Pooling (GAP) Layer

Add (and train) a **single FC layer** after the GAP.

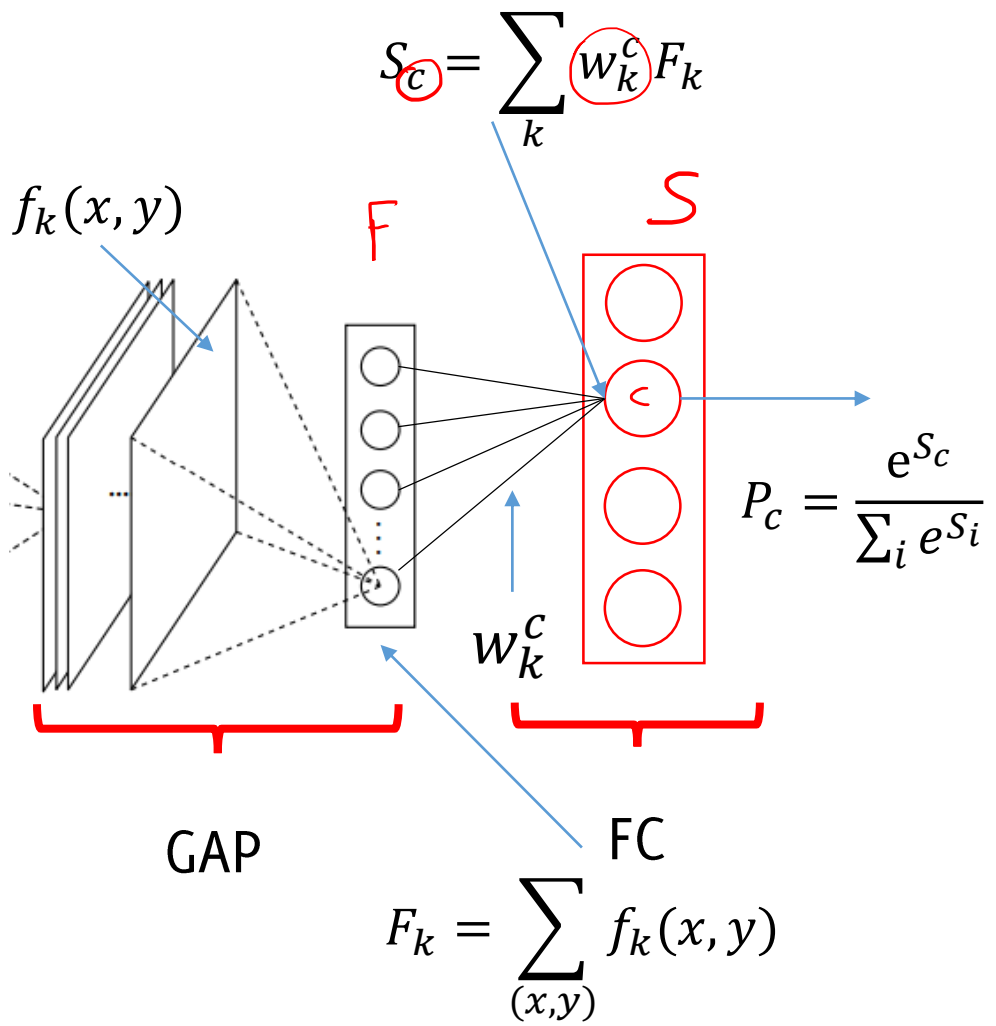
The FC computes S_c for each class c as the weighted sum of $\{F_k\}$, where weights are defined during training

Then, the class probability P_c via soft-max (class c)

Remark: when computing

$$S_c = \sum_k w_k^c \, F_k$$

w_k^c encodes the importance of F_k for the class c.



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

The Global Averaging Pooling (GAP) Layer

However

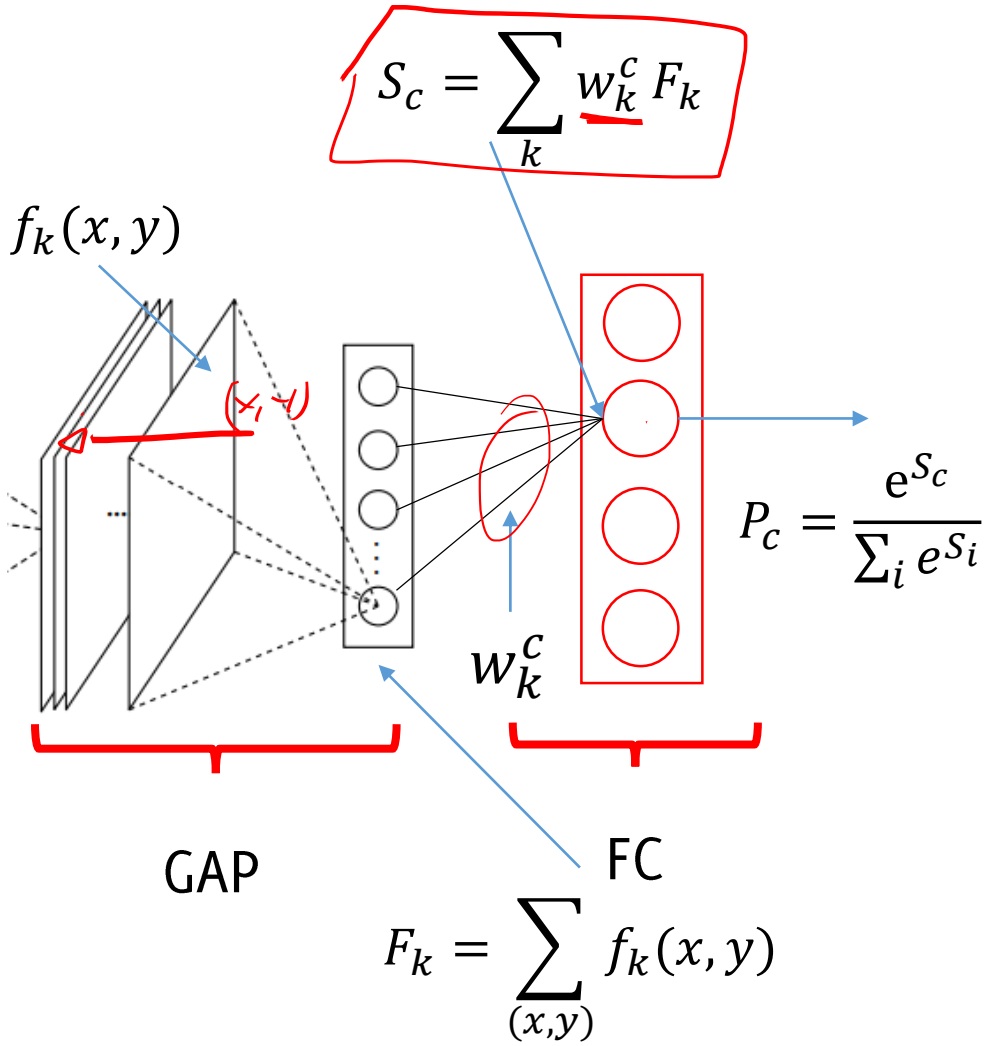
$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c \, f_k(x,y)$$

And CAM is defined as

$$M_c(x,y) = \sum_k w_k^c \, f_k(x,y)$$

where $M_c(x,y)$ directly indicates the importance of the activations at (x,y) for predicting the class c

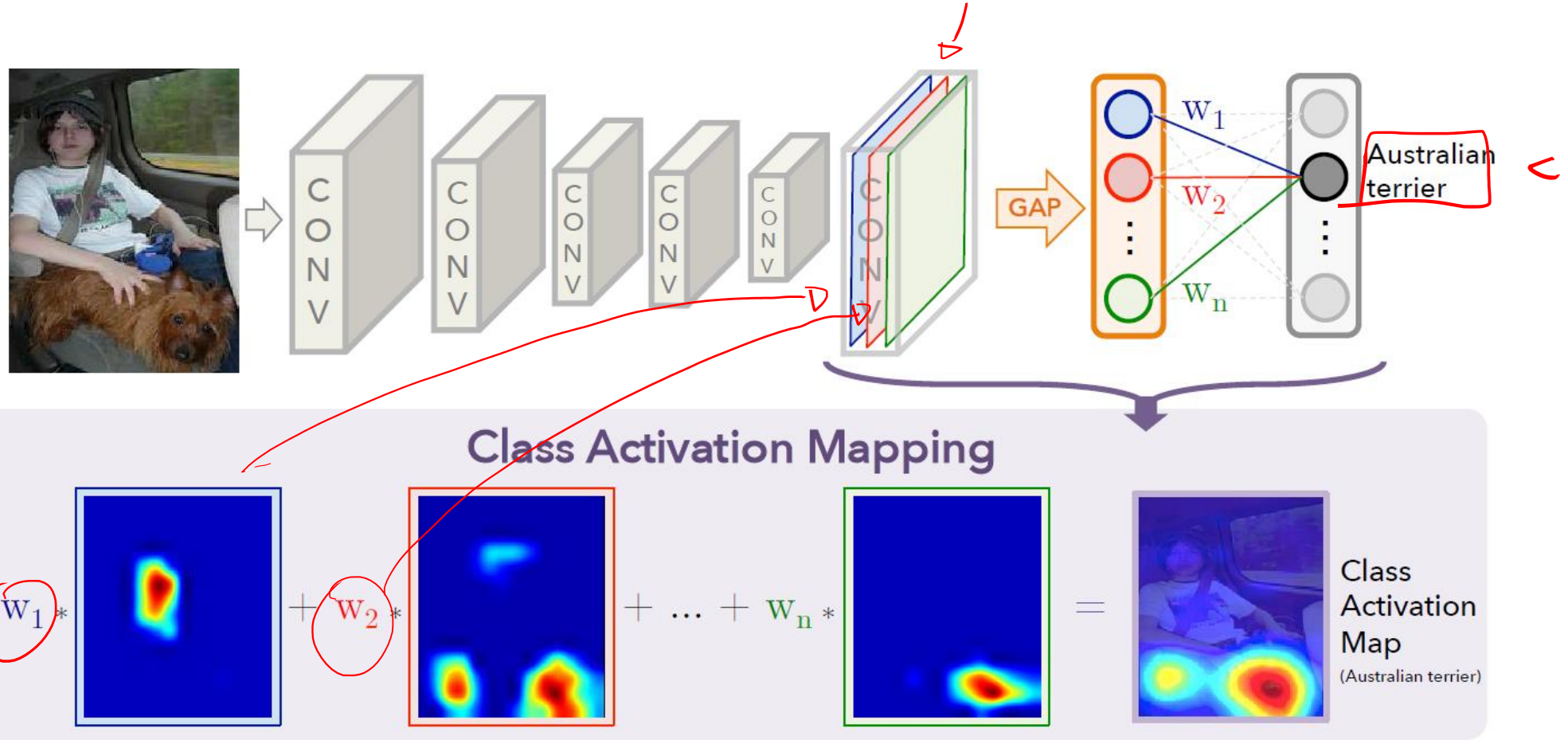
Thanks to the softmax, the depth of the last convolutional volume can be different from the number of classes



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

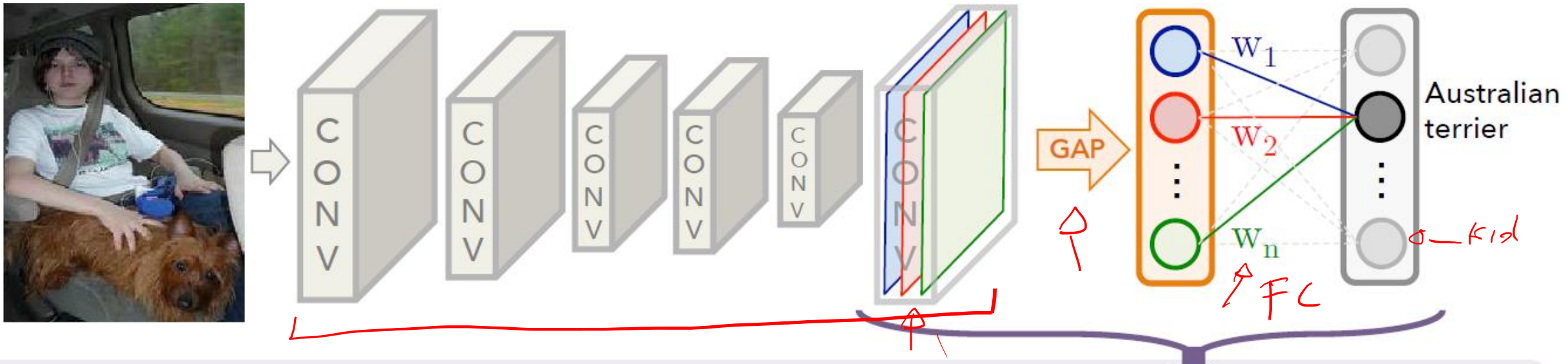
Class Activation Mapping

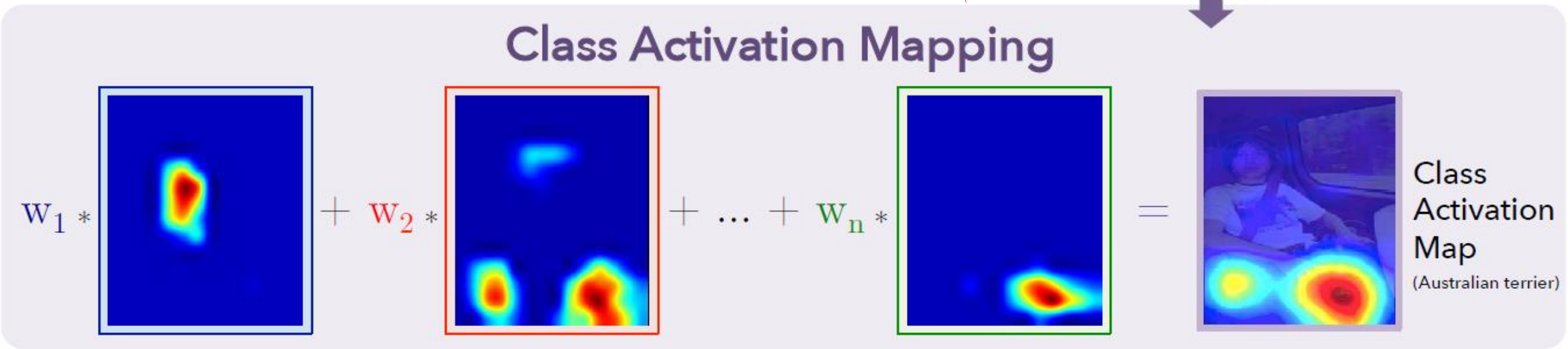
Now, the weights represents the importance of each feature map to yield the final prediction. Upsampling might be necessary to match the input image



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

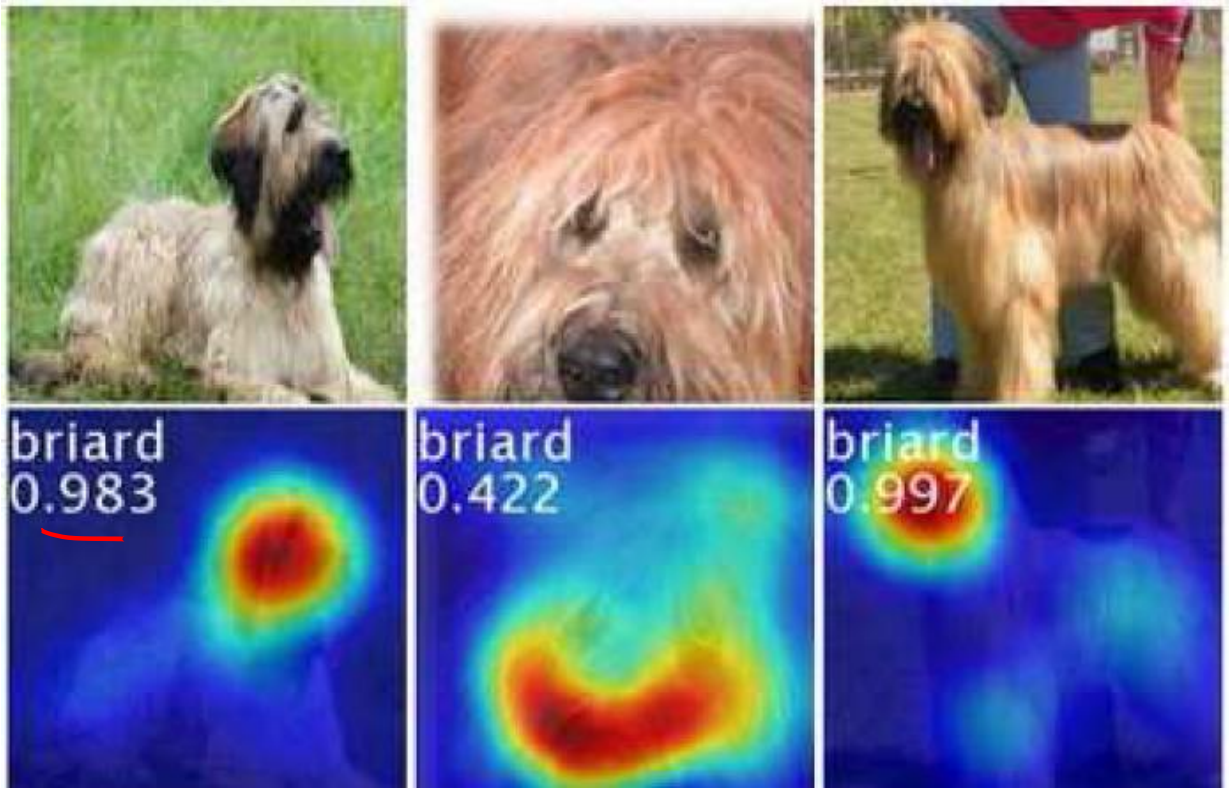
Class Activation Mapping





Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

Class Activation Mapping



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

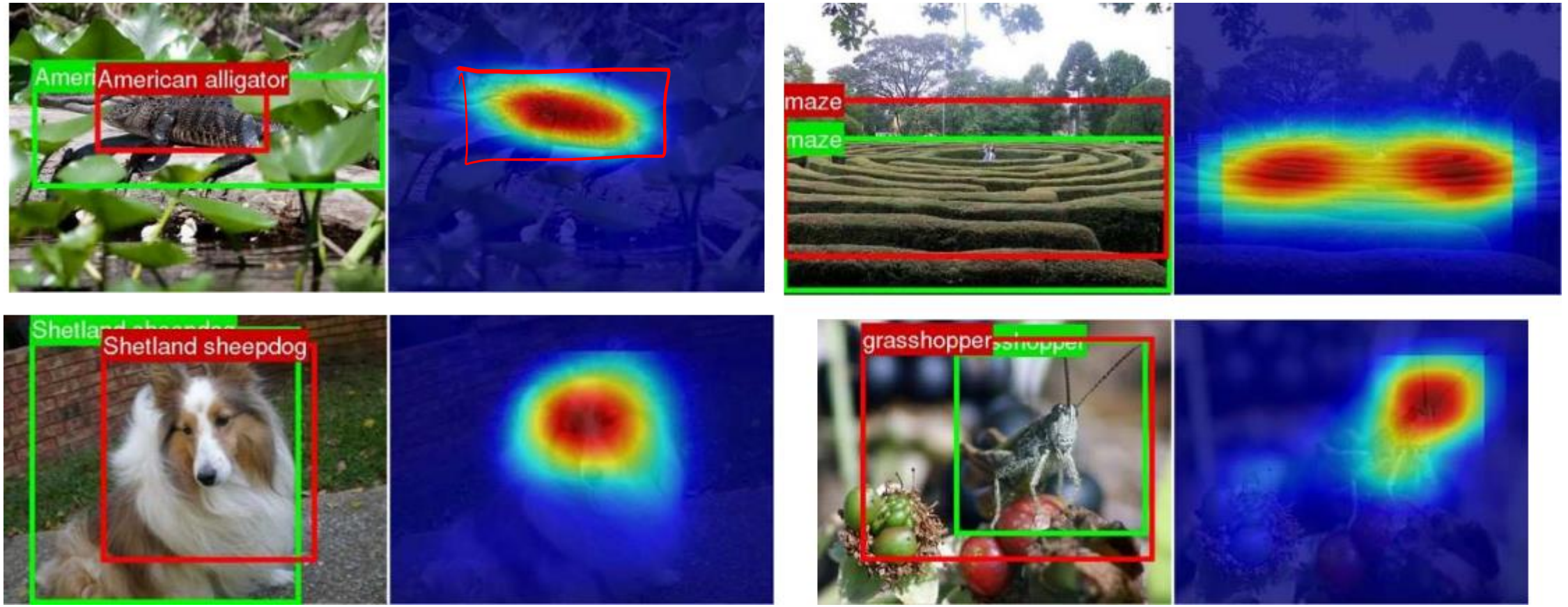
Remarks

- CAM can be **included in any pre-trained network**, as long as all the FC layers at the end are removed
- The **FC used for CAM is simple**, few neurons and no hidden layers
- **Classification performance might drop** (in VGG removing FC means loosing 90% of parameters)
- **CAM resolution** (localization accuracy) can improve by «**anticipating**» GAP to larger convolutional feature maps (but this reduces the semantic information within these layers)
- **GAP**: encourages the **identification of the whole object**, as all the parts of the values in the activation map concurs to the classification
- **GMP (Global Max Pooling)**: it is enough to have a high maximum, thus **promotes specific discriminative features**

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

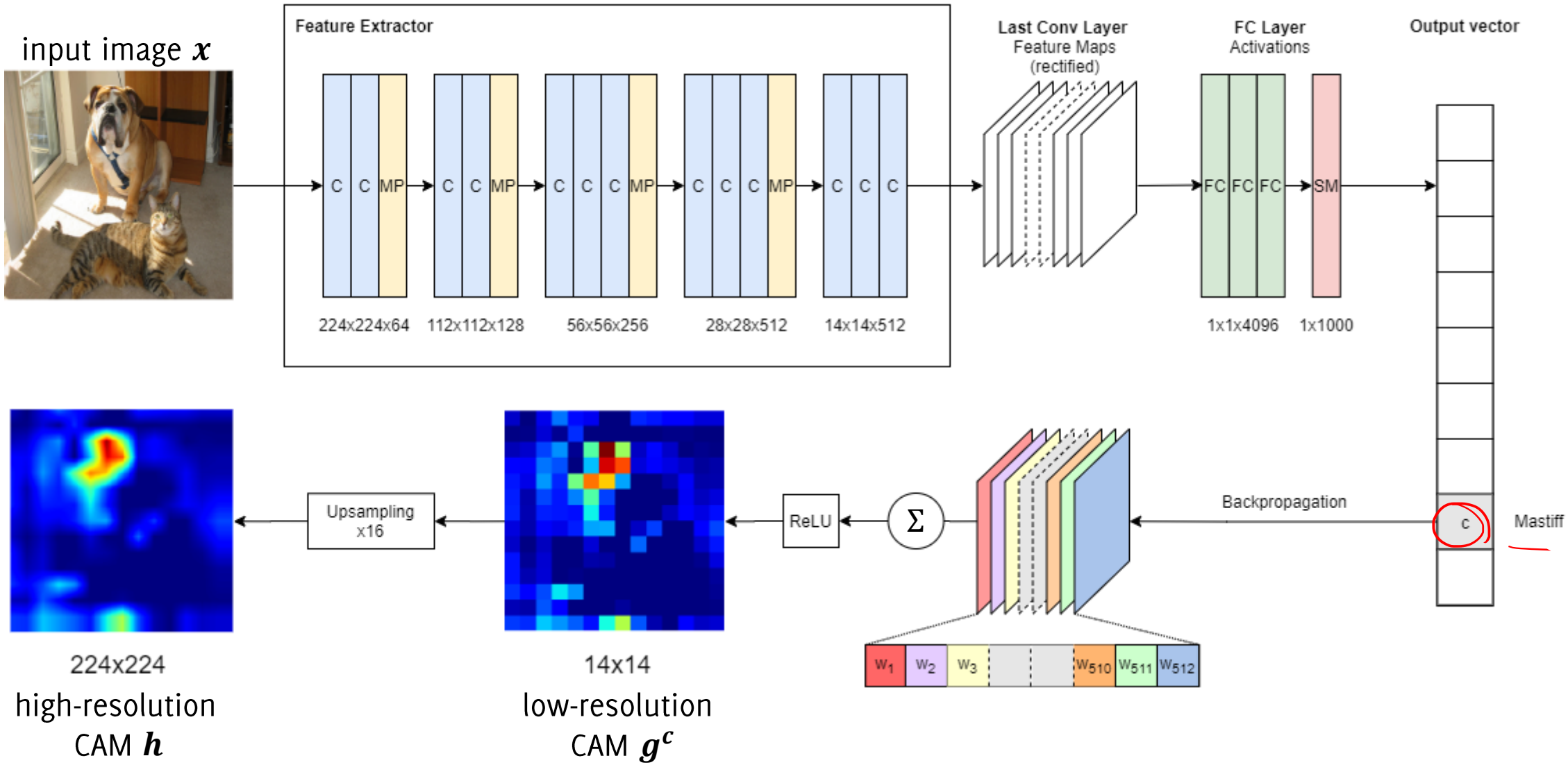
Weakly Supervised Localization

Use thresholding CAM values: > 20% max(CAM), then take the largest componet of the thresholded map (green GT, red estimated location)



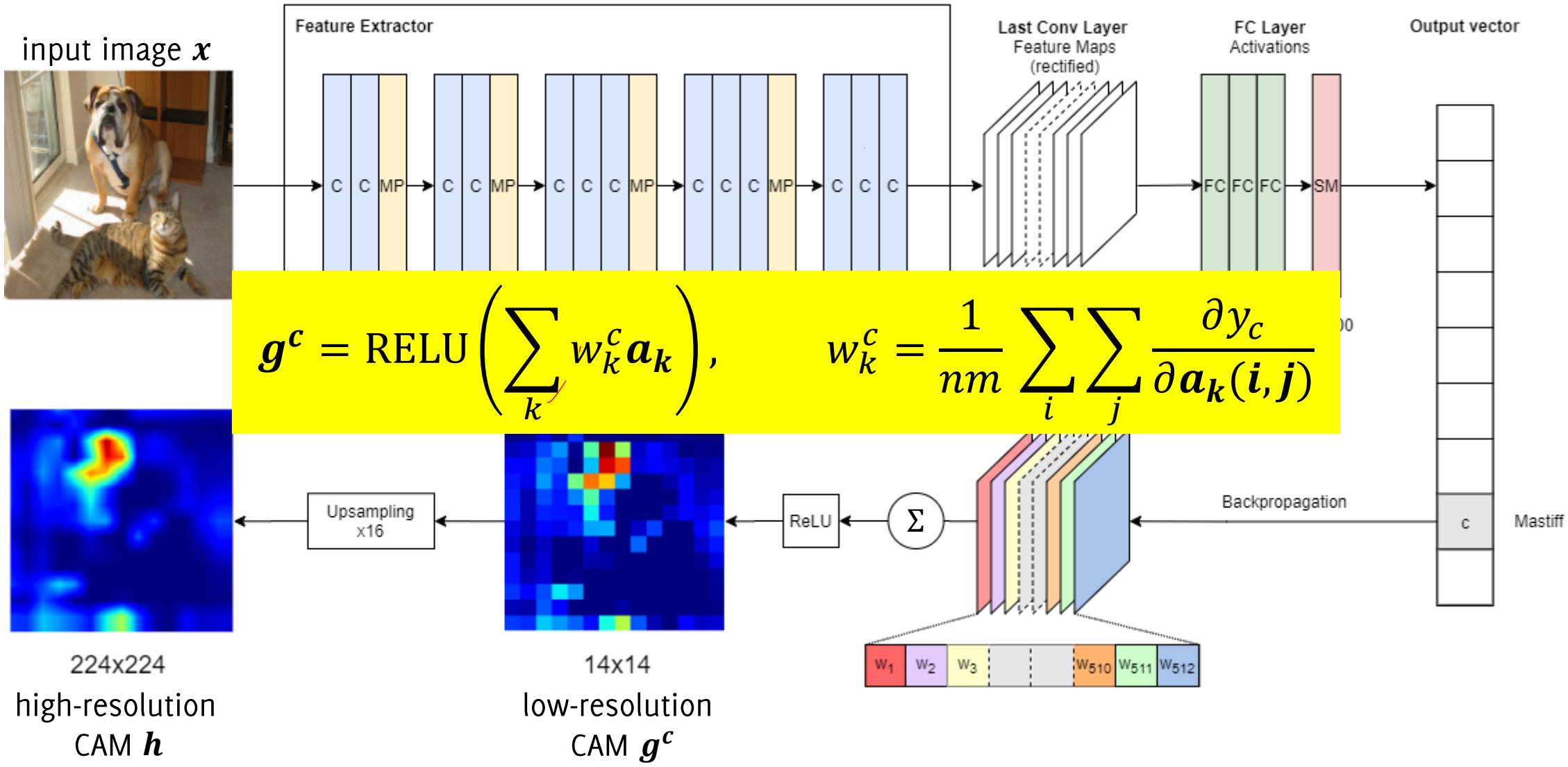
Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016

Grad-CAM and CAM-based techniques



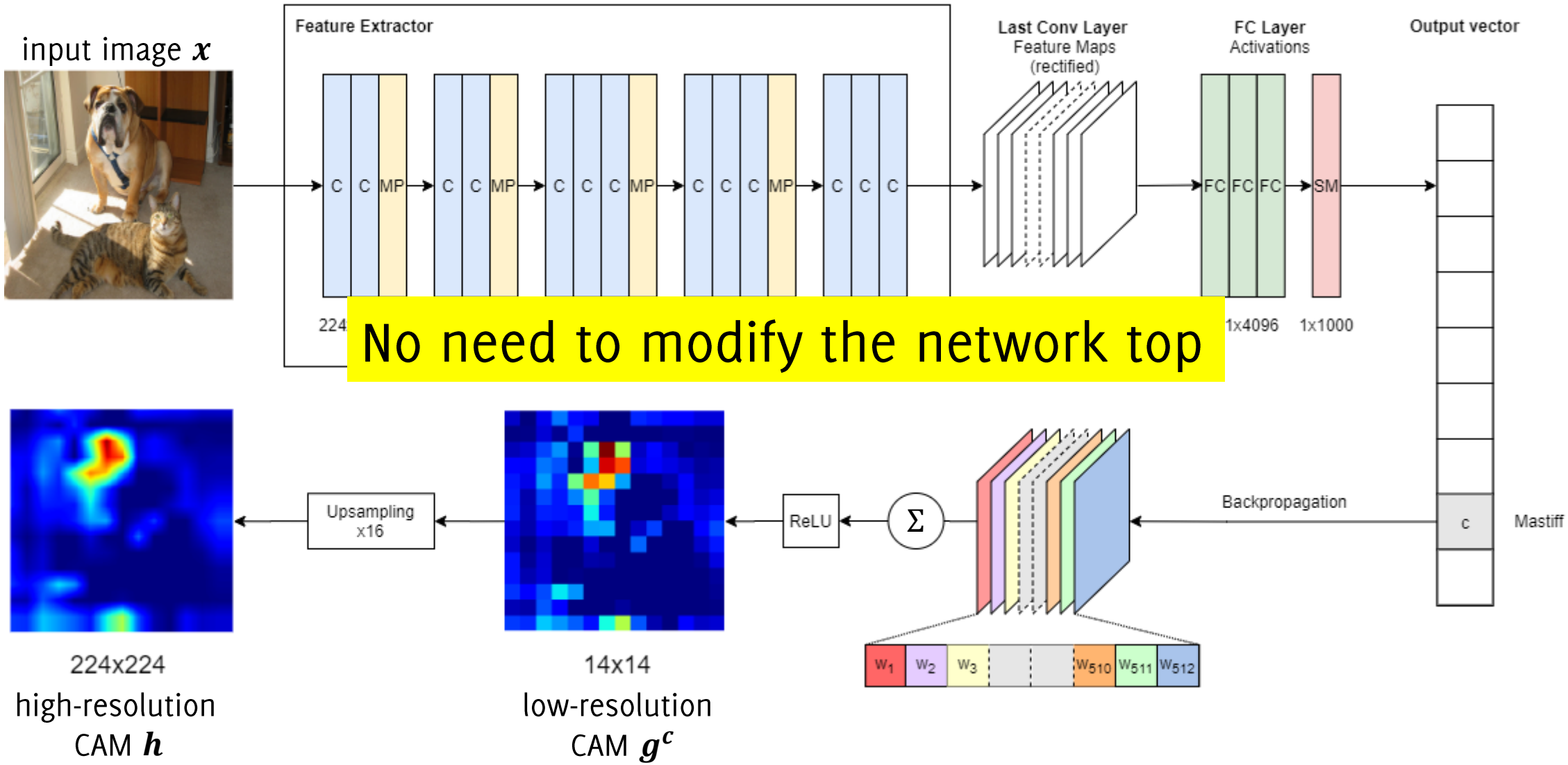
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV 2017*
Type equation here.

Grad-CAM and CAM-based techniques



Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV2017*

Grad-CAM and CAM-based techniques



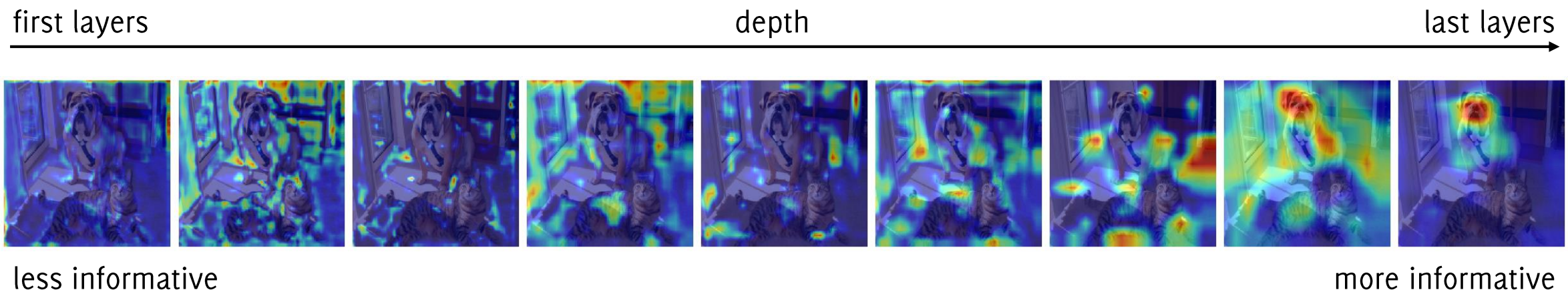
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV2017*

Heatmaps Desiderata

Should be **class discriminative**

Should **capture fine-grained details** (high-resolution)

- This is critical in many applications (*e.g.* medical imaging, industrial processes)

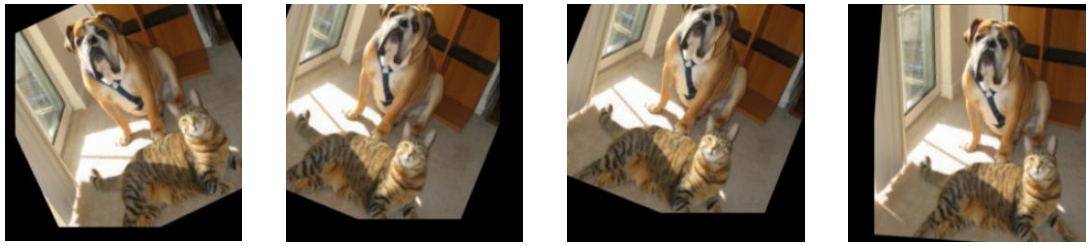


Augmented Grad-CAM

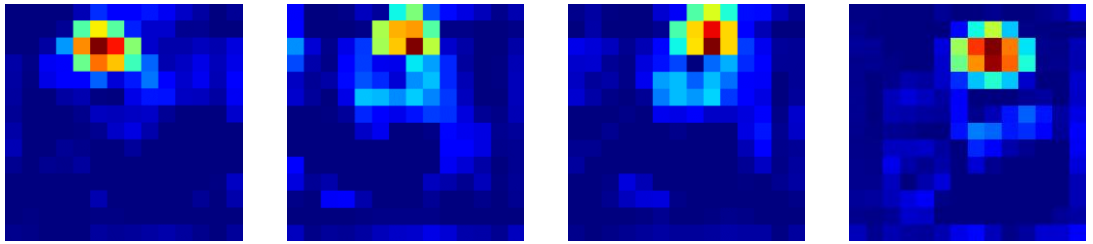
We consider the augmentation operator $\mathcal{A}_l: \mathbb{R}^{N \times M} \rightarrow \mathbb{R}^{N \times M}$, including random rotations and translations of the input image $\boldsymbol{x}$

Augmented Grad-CAM: increase heat-maps resolution through image augmentation

All the responses that the CNN generates to the **multiple augmented versions of the same input image** are very informative for reconstructing the high-resolution heat-map $\boldsymbol{h}$



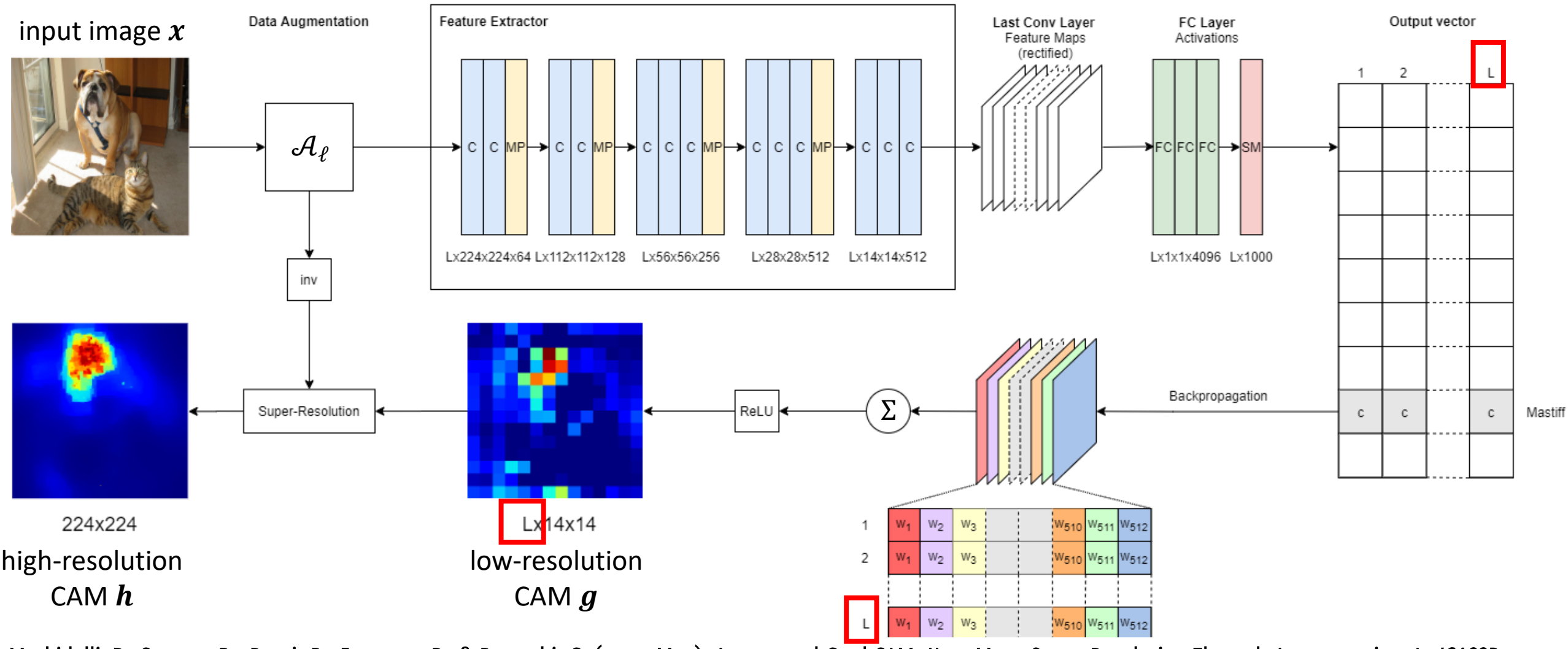
$x_1 = \mathcal{A}_1(x)$ $x_2 = \mathcal{A}_2(x)$ $x_3 = \mathcal{A}_3(x)$ $x_4 = \mathcal{A}_4(x)$



g_1 g_2 g_3 g_4

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

Augmented Grad-CAM

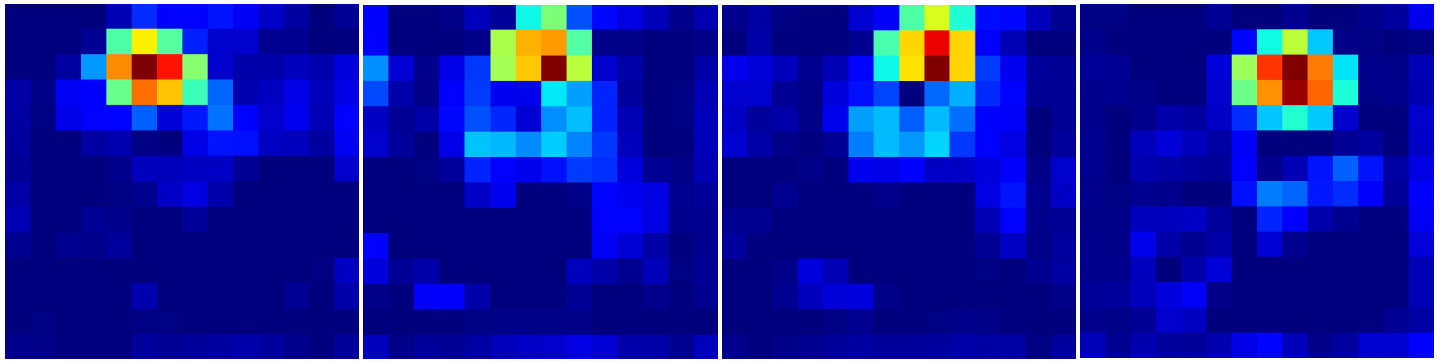


Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

The Super-Resolution Approach

We perform heat-map Super-Resolution (**SR**) by taking advantage of the information shared in multiple low-resolution heat-maps computed from the **same input under different – but known – transformations**

CNNs are in general invariant to roto-translations, in terms of predictions, but each $\boldsymbol{g}_\ell$ actually contains different information



g_1 g_2 g_3 g_4

General approach, our SR framework can be combined with any visualization tool (not only Grad-CAM)

The Super-Resolution Formulation

We model heat-maps computed by Grad-CAM as the result of an unknown downsampling operator $\boldsymbol{\mathcal{D}} : \mathbb{R}^{N \times M} \rightarrow \mathbb{R}^{n \times m}$

The high-resolution heat-map $\boldsymbol{h}$ is recovered by solving an inverse problem

$$\underset{h}{\operatorname{argmin}} \frac{1}{2} \sum_{l=1}^{L} \|\mathcal{D}\mathcal{A}_\ell h - g_\ell\|_2^2 + \lambda TV_{\ell_1}(h) + \frac{\mu}{2}\|h\|_2^2 \quad (1)$$

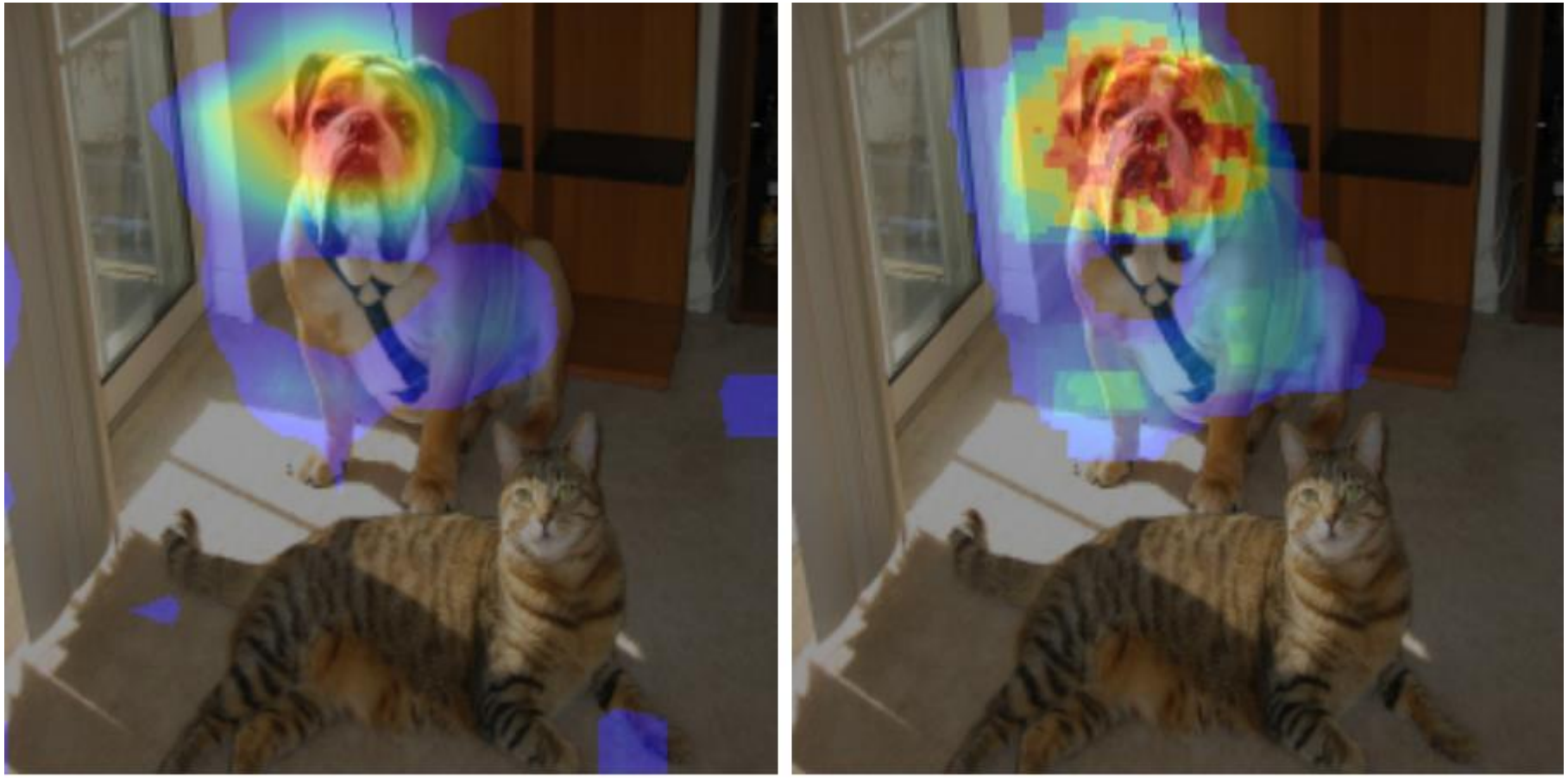
TV_{ℓ_1}: Anistropic Total Variation regularization is used to preserve the edges in the target heat-map (high-resolution)

$$TV_{\ell_1}(\boldsymbol{h}) = \Sigma_{i,j} \|\partial_x \boldsymbol{h}(i,j)\| + \|\partial_y \boldsymbol{h}(i,j)\| \quad (2)$$

This is solved through Subgradient Descent since the function is convex and non-smooth

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G “Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation”. ICASSP 2020

Augmented Grad-CAM



(a) Grad-CAM. (b) Augmented Grad-CAM.

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

17.45

Object Detection

Thanks to Davide Rutigliano, former AN2DL student!

Object Detection Task

Given a fixed set of categories and an input image which contains an unknown and varying number of instances

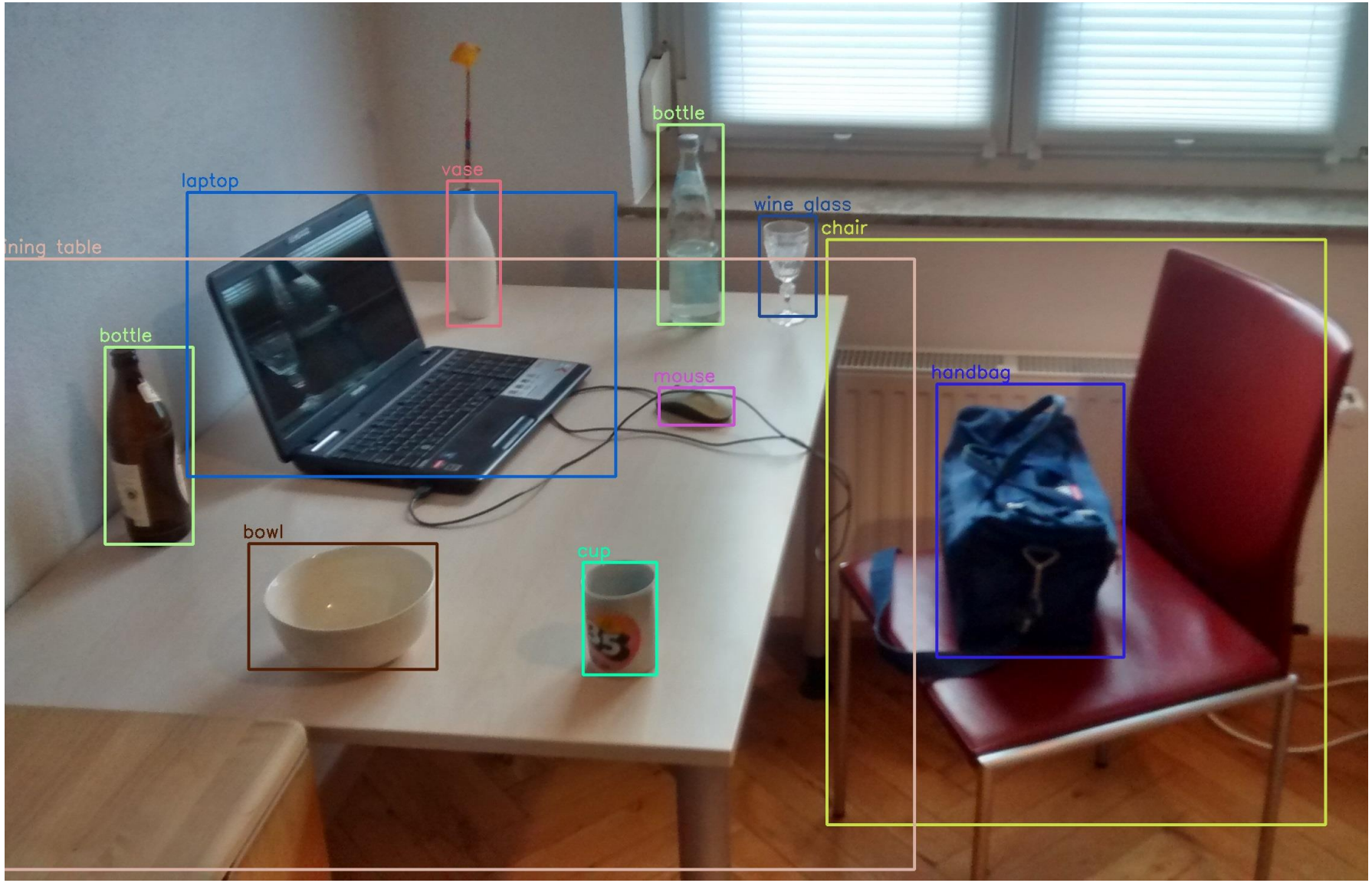
Draw a bounding box on each object instance

A training set of annotated images with labels and bounding boxes for each object is required

Each image requires **a varying number of outputs**

MAN: (x,y,h,w)
KID: (x,y,h,w)
GLOVE: (x,y,h,w)

Annotated Dataset for Object Detection



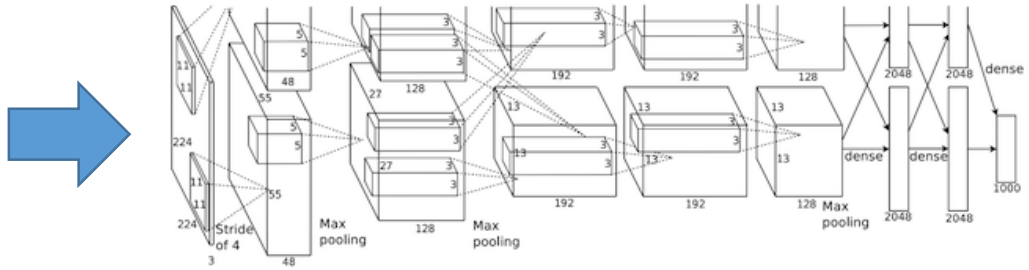
G. Boracchi

The Straightforward Solution: Sliding Window

1000 x 2000 pixels

- Similar to the sliding window for semantic segmentation
- A pretrained model is meant to process a fixed input size (e.g. 224 x 224 x 3)
- Slide on the image a window of that size and classify each region.
- Assign the predicted label to the central pixel

Adopt the whole machinery seen so far to each crop of the image



car

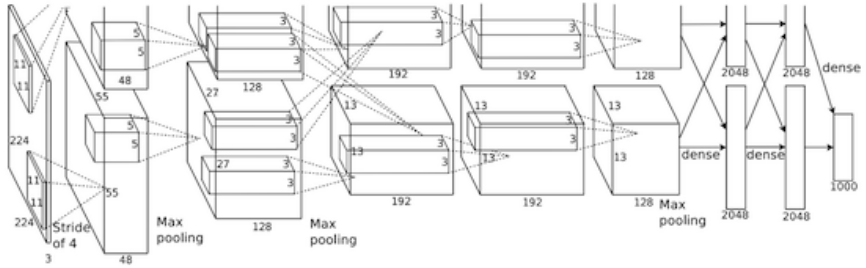
G. Boracchi

The Straightforward Solution: Sliding Window

1000 x 2000 pixels

- Similar to the sliding window for semantic segmentation
- A pretrained model is meant to process a fixed input size (e.g. 224 x 224 x 3)
- Slide on the image a window of that size and classify each region.
- Assign the predicted label to the central pixel

Adopt the whole machinery seen so far to each crop of the image

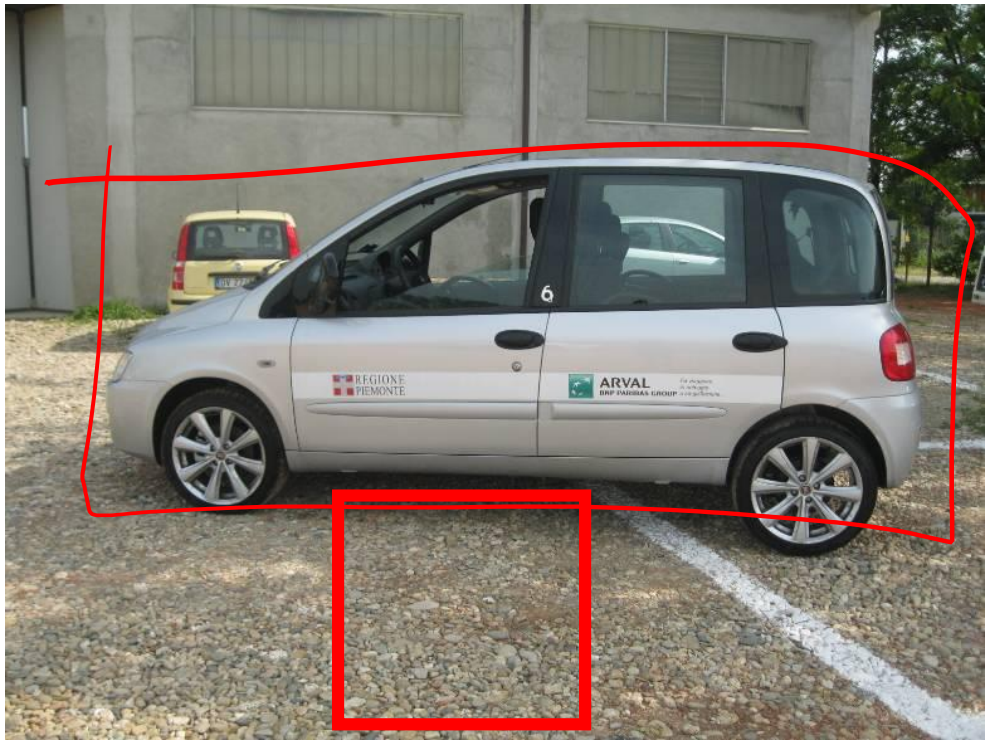


wheel

G. Boracchi

The Straightforward Solution: Sliding Window

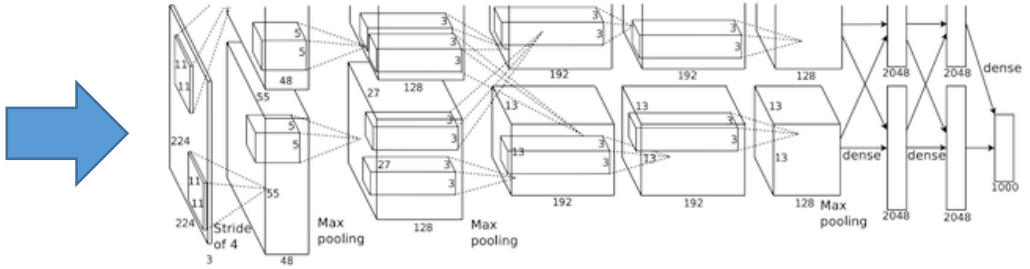
1000 x 2000 pixels



- Similar to the sliding window for semantic segmentation
- A pretrained model is meant to process a fixed input size (e.g. 224 x 224 x 3)
- Slide on the image a window of that size and classify each region.
- Assign the predicted label to the central pixel

Adopt the whole machinery seen so far to each crop of the image

The background class has to be included!



background

G. Boracchi

Many drawbacks…

Cons:

- **Very inefficient!** Does not re-use features that are «shared» among overlapping crops
- How to choose the crop size?
- Difficult to detect objects at different scales!
- A huge number of crops of different sizes should be considered….

Plus:

- No need of retraining the CNN

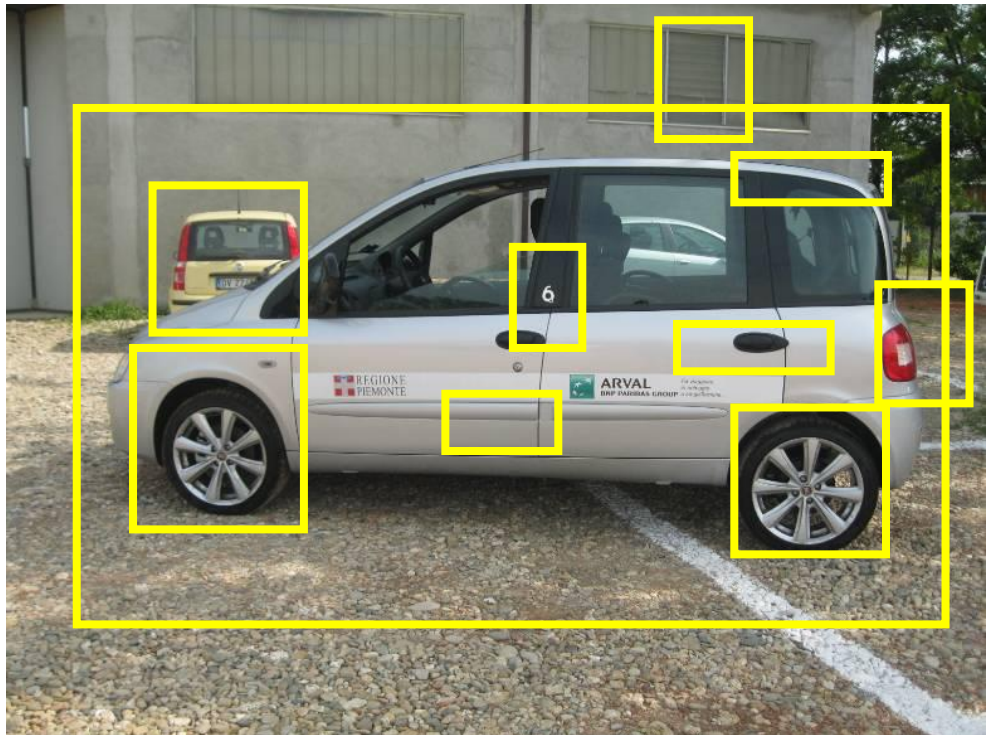
Region Proposal

Region proposal algorithms (and networks) are meant **to identify bounding boxes** that correspond to a **candidate object** in the image.

Algorithms with **very high recall** (but low precision) were there before the deep learning advent

The idea is to:

- Apply a region proposal algorithm
- Classify by a CNN the image inside each proposal regions



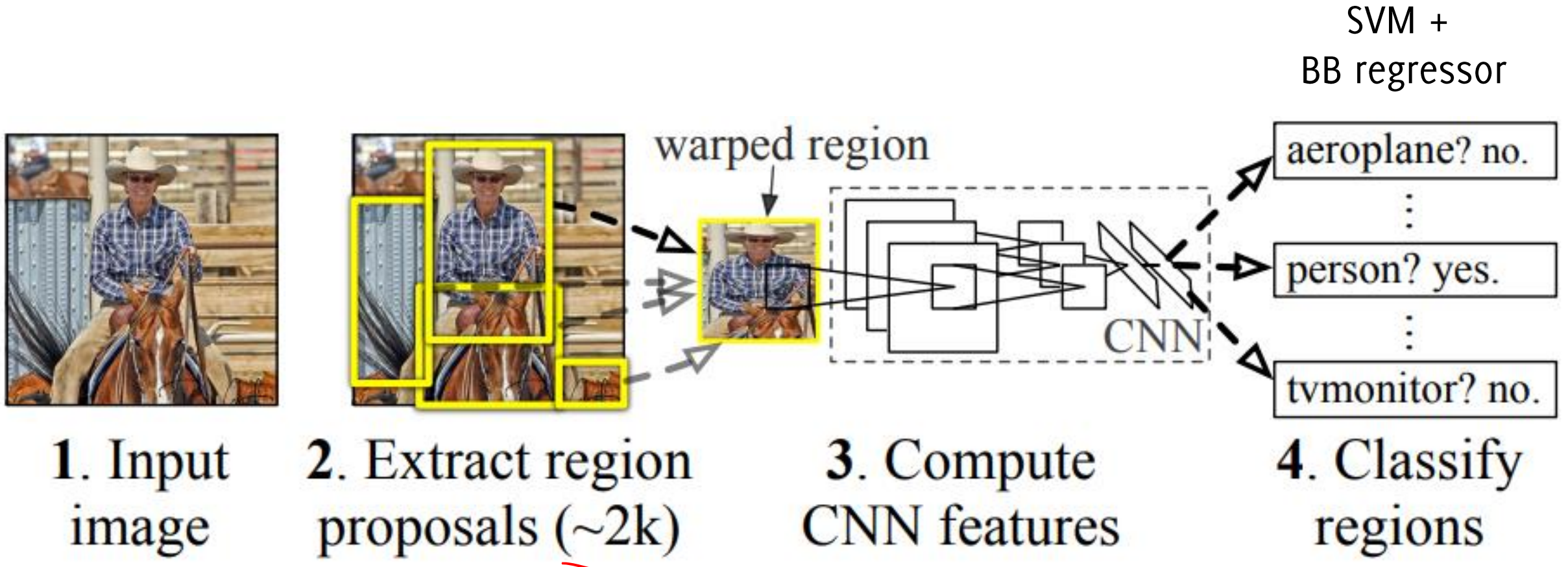
Rich feature hierarchies for accurate object detection and semantic segmentation

Ross Girshick[1] Jeff Donahue[1,2] Trevor Darrell[1,2] Jitendra Malik[1]
[1]UC Berkeley and [2]ICSI
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

G. Boracchi

R-CNN

Object detection by means of region proposal (R stands for **regions**)

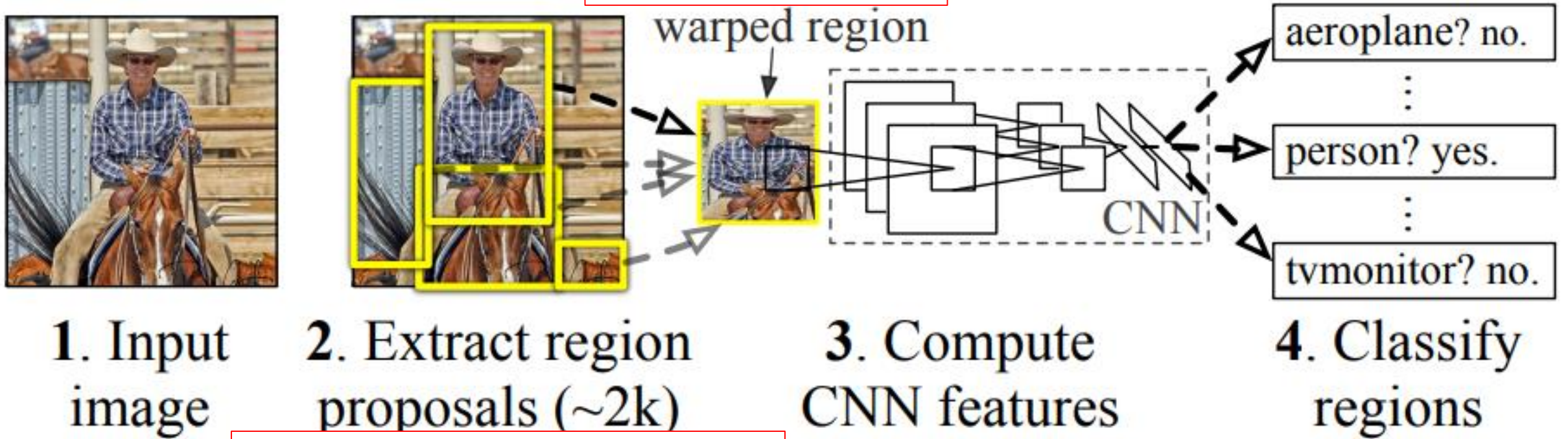


Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *CVPR 2014*

R-CNN

Object detection by means of region proposal

Warping to a predefined size is necessary since the CNN has a FC layer



There is no learning in the region proposal algorithm, **very high recall (e.g. Selective Search)**

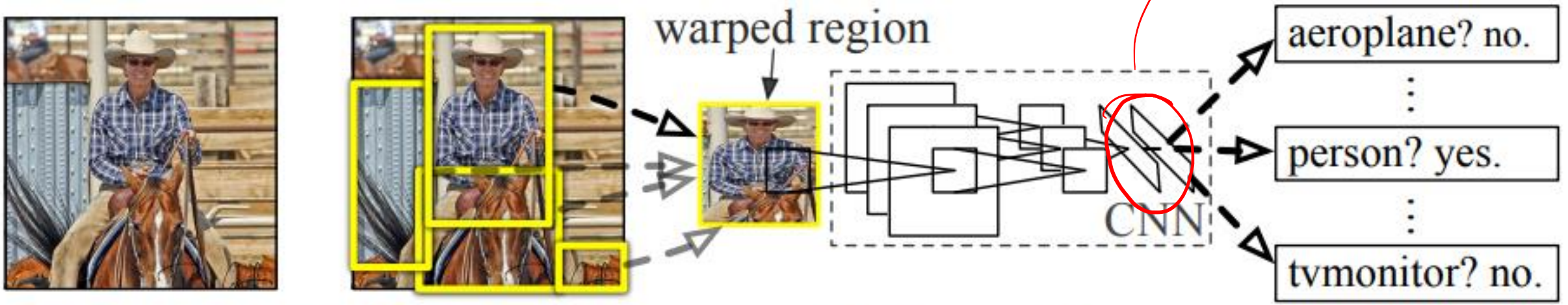
Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *CVPR 2014*

R-CNN

This is AlexNet as a pre-trained network

latent representation

SVM + BB regressor



1. Input image

2. Extract region proposals (~2k)

3. Compute CNN features

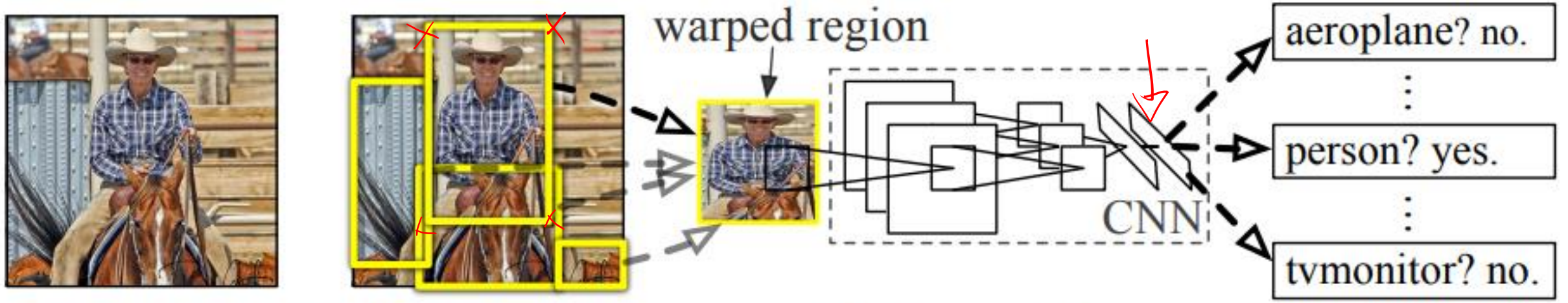
4. Classify regions

SVM trained to minimize the classification error over the extracted ROI

R-CNN

The **regions are refined** by a regression network to correct the bounding box estimate from ROI algorithm

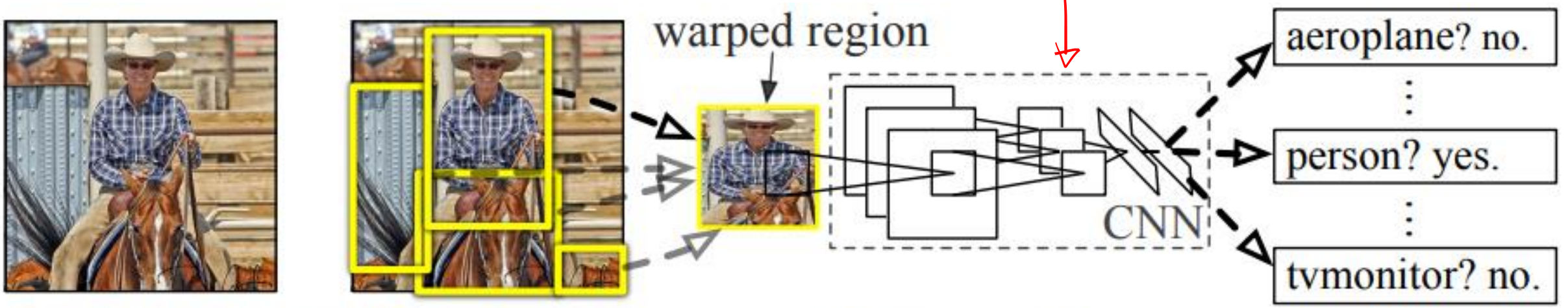
SVM + BB regressor



Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *CVPR 2014*

R-CNN

The pretrained CNN is fine-tuned over the classes to be detected (21 vs 1000 of Alextnet) by placing a FC layer after feature extraction.
No end-to-end training of the SVM



Include **a background class** to get rid of those regions not corresponding to an object

R-CNN limitations

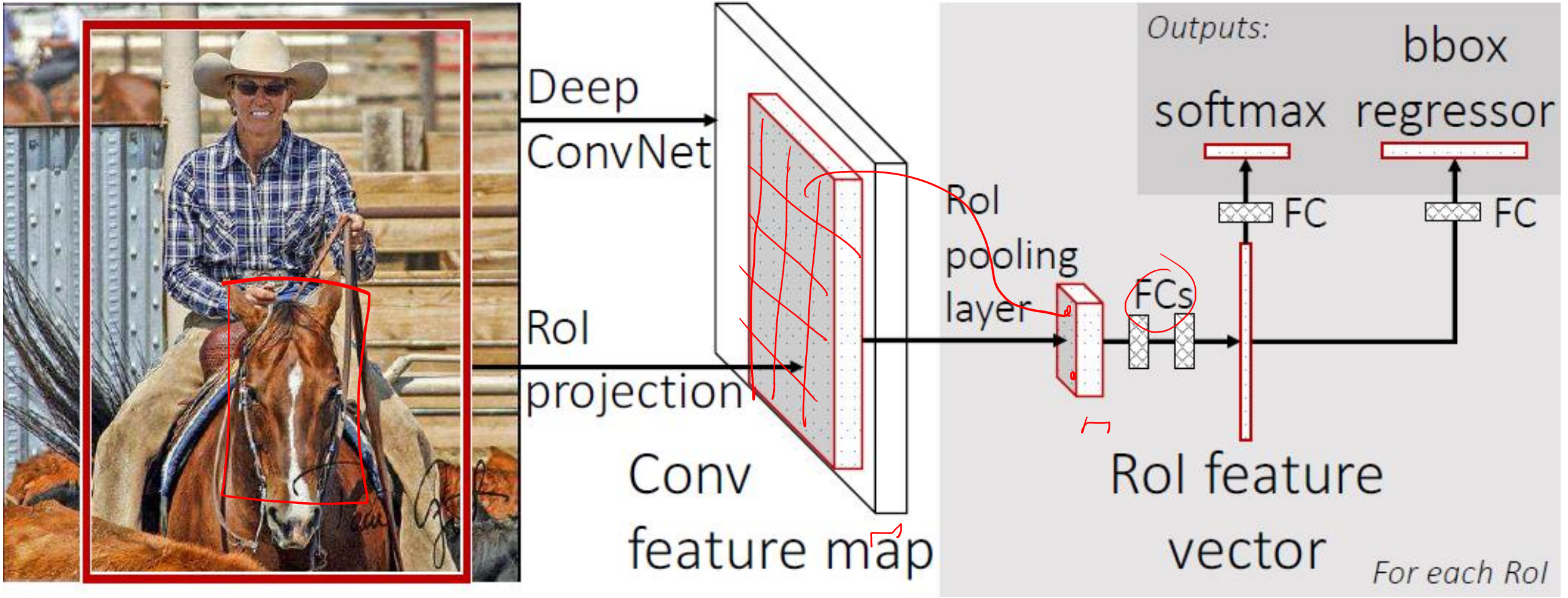
- **Ad-hoc training objectives** and not an end-to-end training
  - Fine-tuning network with softmax classifier (log loss) before training SVM
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- **Region proposals are from a different algorithm** and that part has not been optimized for the detection by CNN
- **Training is slow** (84h), takes **a lot of disk space to store features**
- **Inference** (detection) **is slow** since the CNN has to be executed on each region proposal (**no feature re-use**)
  - 47s / image with VGG16

Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

G. Boracchi

Fast R-CNN

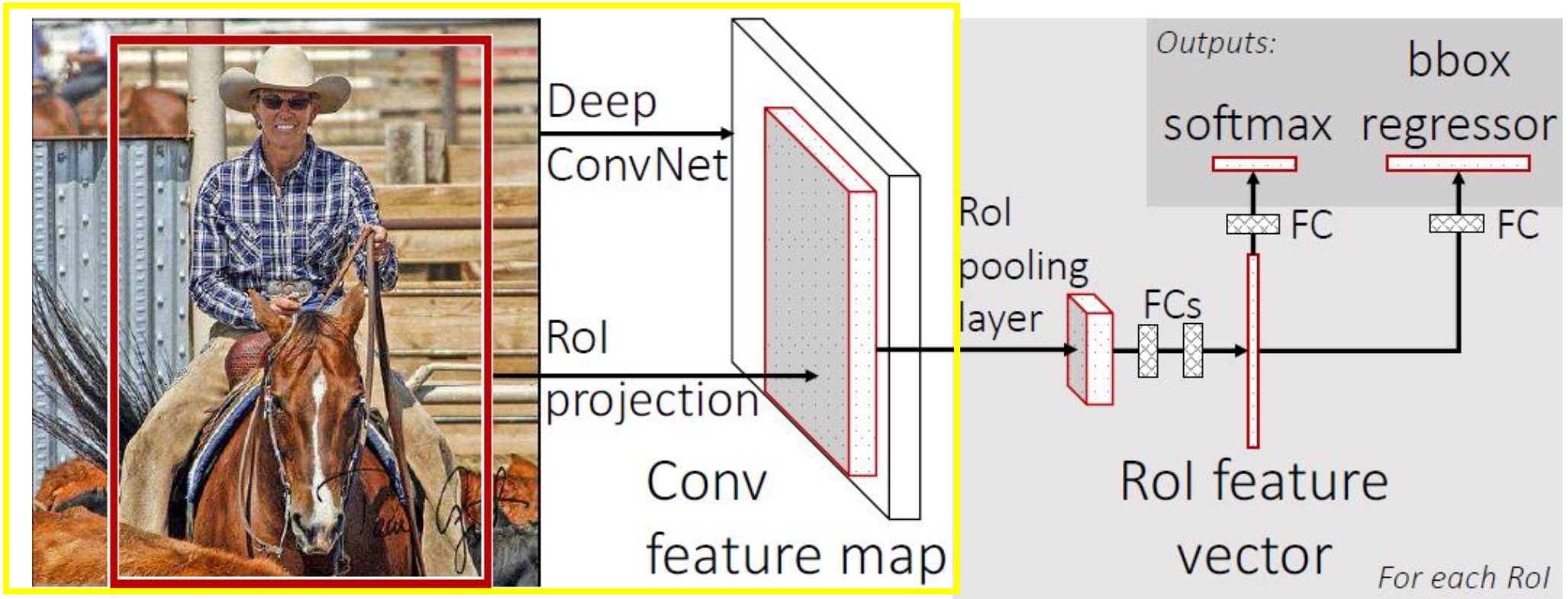


«conv5»

Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

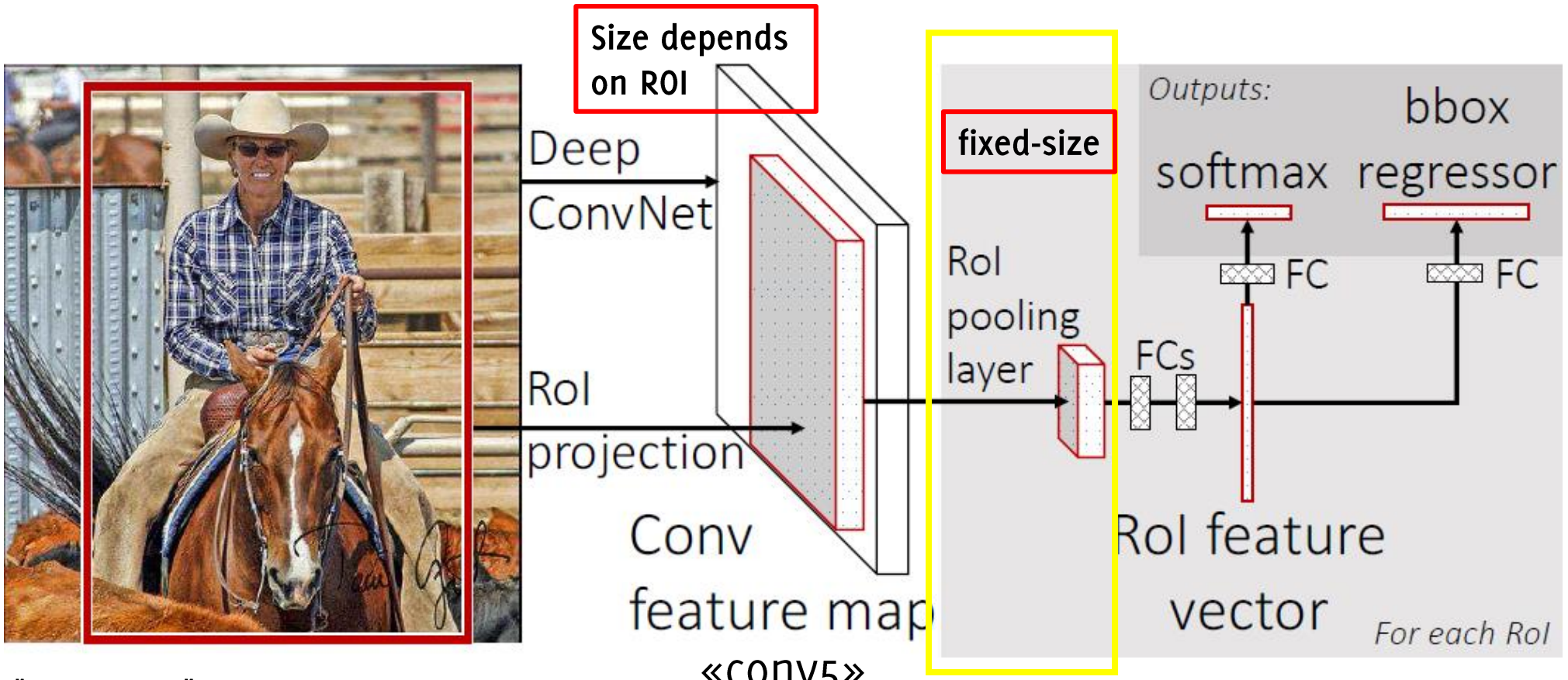
1. The whole image is fed to a CNN that extracts feature maps.
2. **Region proposals** are identified from the image and **projected into the feature maps.** Regions are directly cropped form the feature maps, instead from the image: →**re-use convolutional computation.**



Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

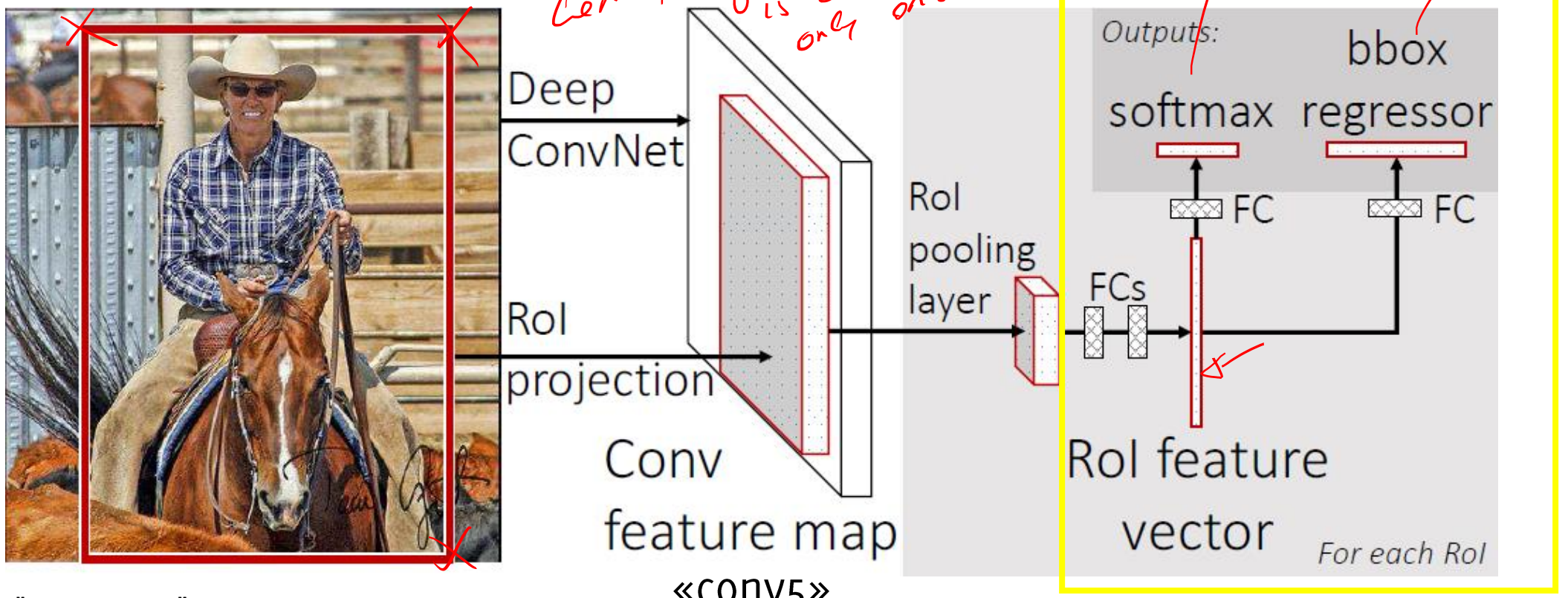
3. Fixed size is still required to feed data to a fully connected layer**. ROI pooling** layers extract a **feature vector of fixed size $H \times W$** from each region proposal. Each **ROI in the feature maps is divided in a $H \times W$ grid** and then **maxpooling** over this provides the feature vector



Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

4. The **FC layers estimate both classes and BB location** (bb regressor)
   A **convex combination of the two** is used as a **multitask loss** to be optimized (as in R-CNN, but no SVM here).
5. Training performed in an **end-to-end manner**



Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

In this new architecture it is possible to **back-propagate through the whole network**, thus train the whole network in an end-to-end manner

It becomes **incredibly faster than R-CNN during testing.**

Now that convolutions are not repeated on overlapping areas, **the vast majority of test time is spent on ROI extraction** (e.g. selective search)

Girshick, Ross. "Fast r-cnn." ICCV 2015

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

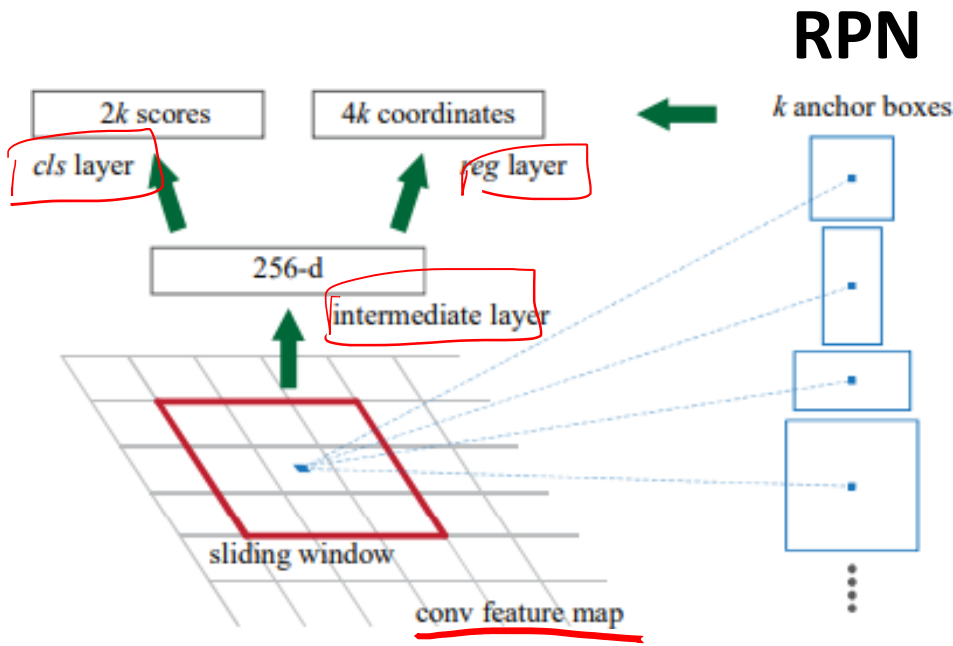
Shaoqing Ren* **Kaiming He** **Ross Girshick** **Jian Sun**

Microsoft Research
{v-shren, kahe, rbg, jiansun}@microsoft.com

G. Boracchi

Faster R-CNN

- Instead of the ROI extraction algorithm, **train a region proposal network** (RPN), which is a F-CNN (3x3 filter size)
- **RPN operates on the same feature maps** used for classification, thus at the last conv layers
- RPN can be seen as an additional module that **improves efficiency** and **focus Fast R-CNN** over the most promising regions for object detection

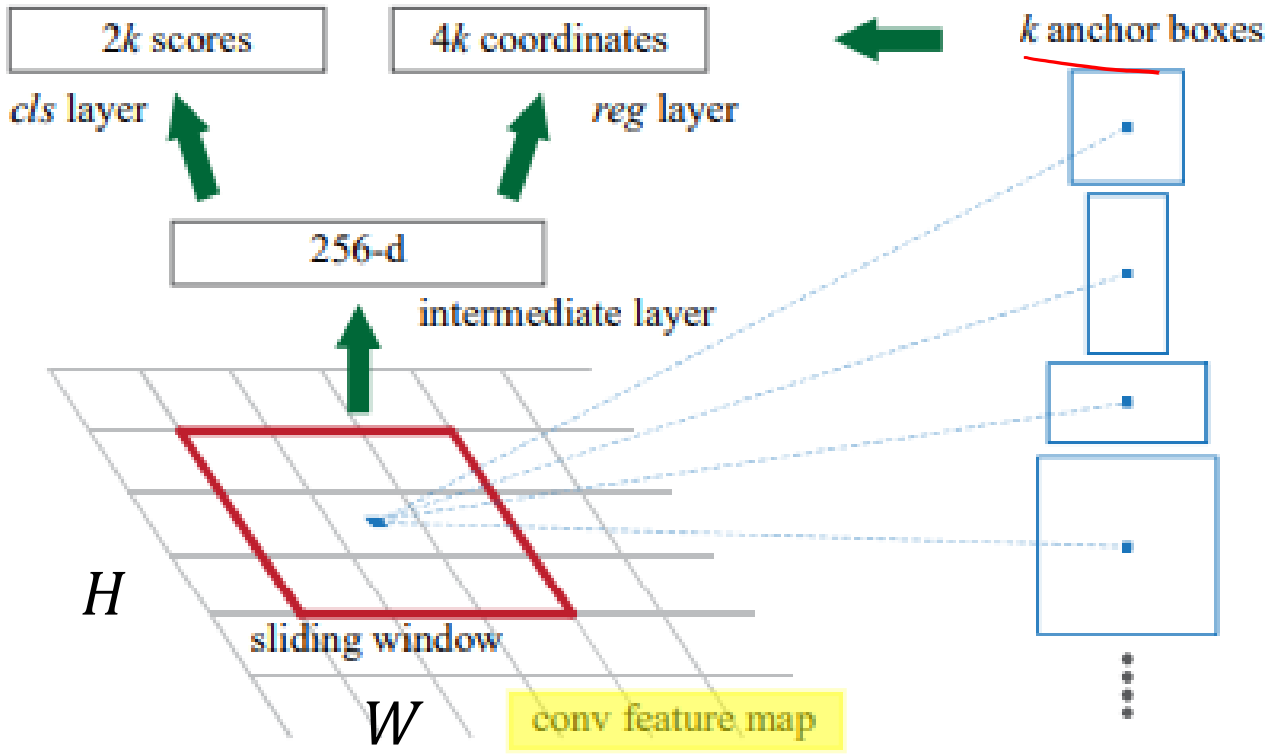


Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

RPN

Goal: Associate to each spatial location k anchor boxes, i.e. ROI having different scales and rations (e.g. $k = 3 \times 3$, 3 sizes of the anchor side, 3 height/width ratios)

The network outputs $H \times W \times k$ candidates anchor and estimate *scores* for each anchor



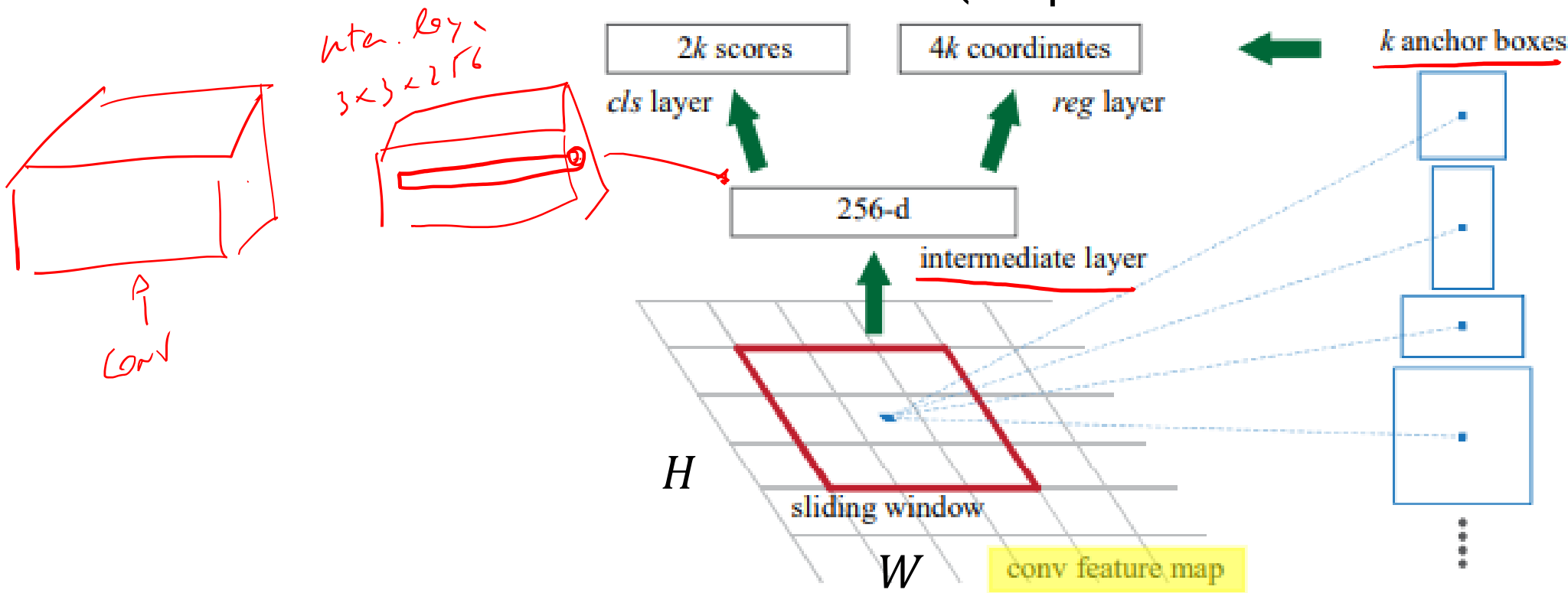
Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

RPN: Intermediate layer

Intermediate layer: is a standard CNN layer that takes as input the last layer of the feature-extraction network and uses 256 filters of size 3×3.

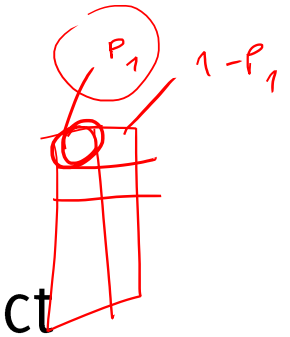
It is used to **reduce the dimensionality of the feature map.**

Maps the region to a lower dimensional vector of size (output size $H \times W \times 256$)

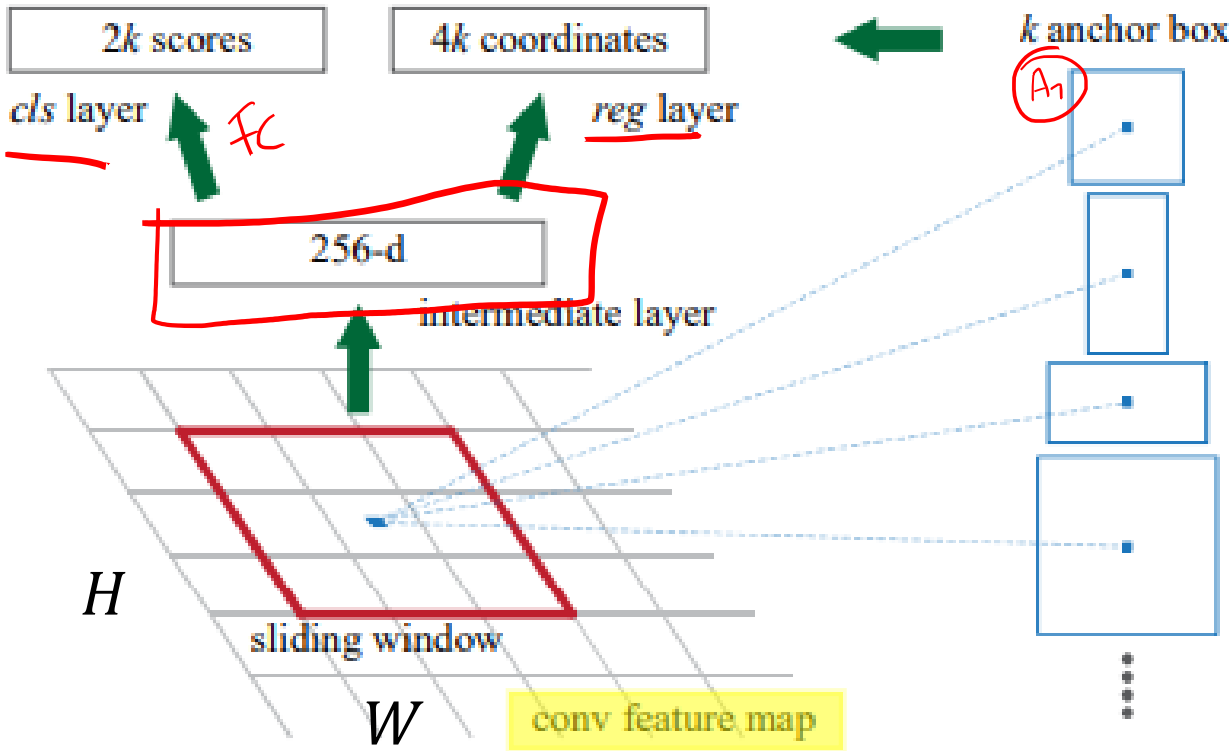


Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

RPN estimating k anchors



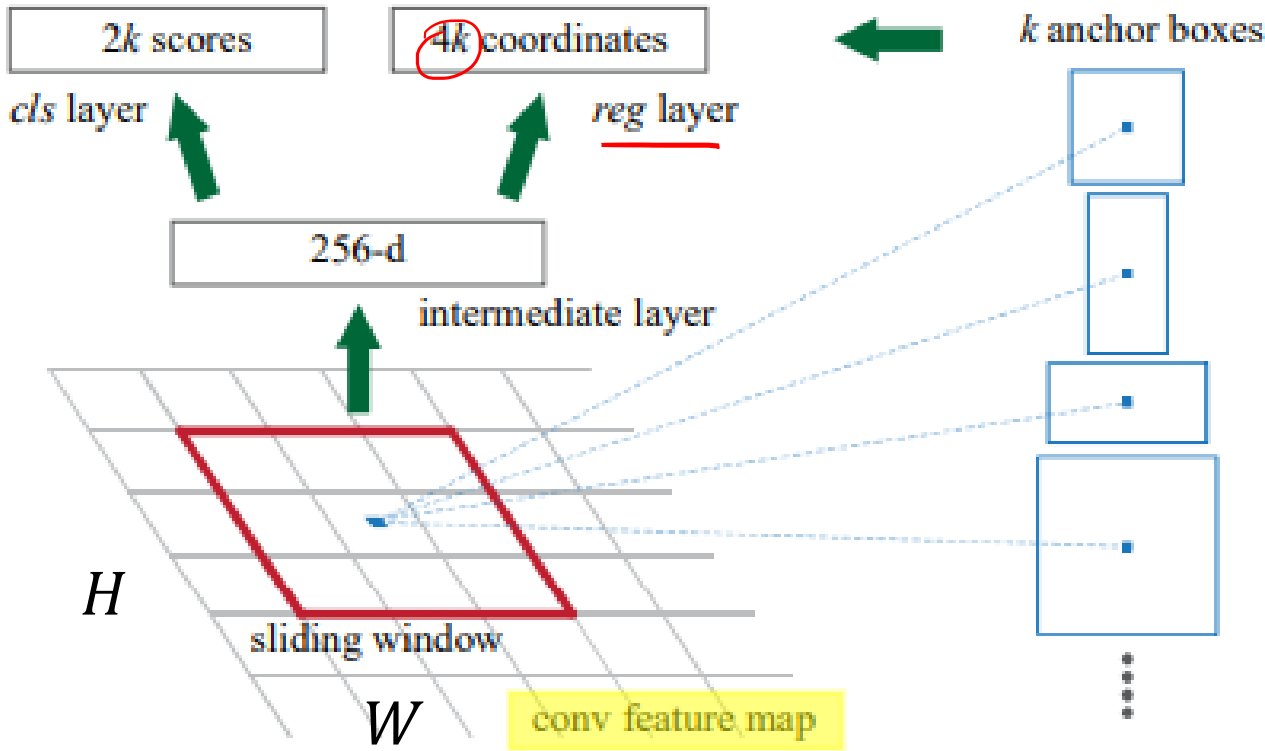
- The **cls** (classification) **network** is trained to predict the ***object probability,*** i.e. the probability for an anchor to contain and object *[contains / does not contain]* $\rightarrow 2k$ probability estimates
- Made of a stack of convolutional layers having size 1×1
- Each of these k proabity pairs corresponds to a specific anchor (having a specific dimension) **and expresses the probability for that anchor in that spatial location to contain *any object***



Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

RPN estimating k anchors

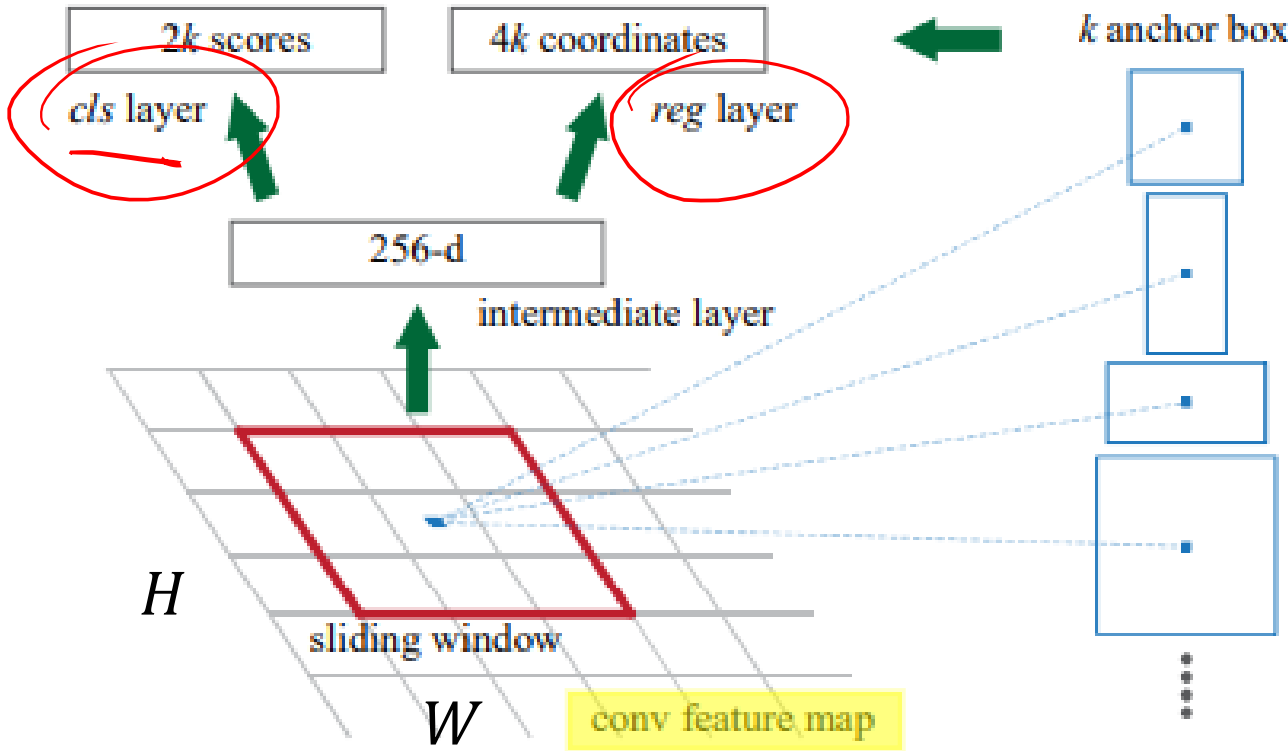
- The **reg** (regression) **network** is trained to *adjust* each of the k predicted anchor to better match object ground truth → $4k$ estimates for the 4 bounding box cooridnates
- Each of these k 4 –tuples **expresses the** refinements for a specific anchor



Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

RPN estimating k anchors

- If you want to train the network to predict different anchors, there is no need to design different RPN, bust just to **define different labels when training the RPN, associated to different anchors**
- Each of these k 4 −tuples **expresses the** refinements for a specific anchor



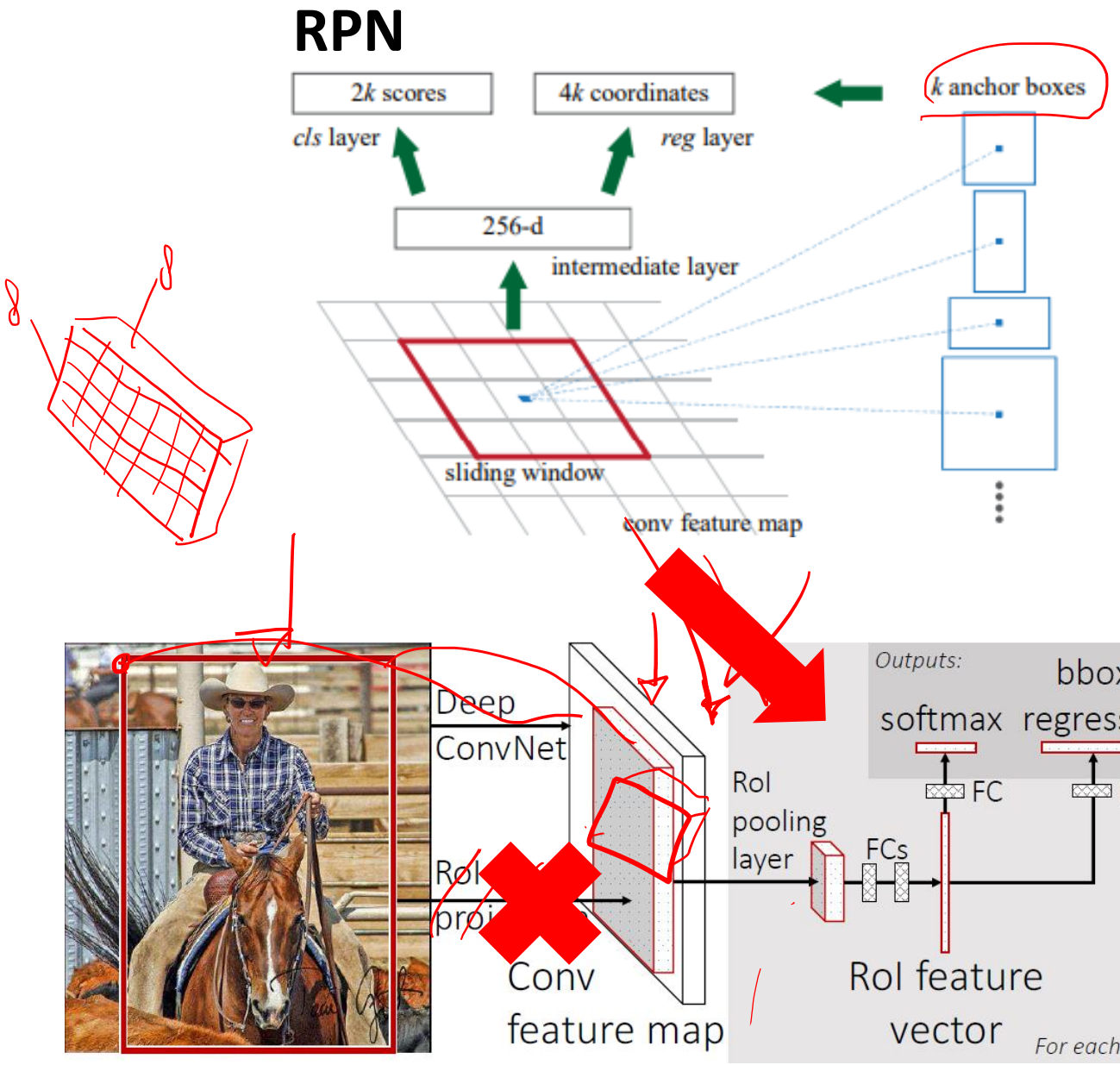
Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN

RPN returns $H \times W \times k$ region proposals,
thus replaces the region proposal algorithms
(selective search)

After RPN there is a **non-maximum suppression**
based on the *objectiveness score*

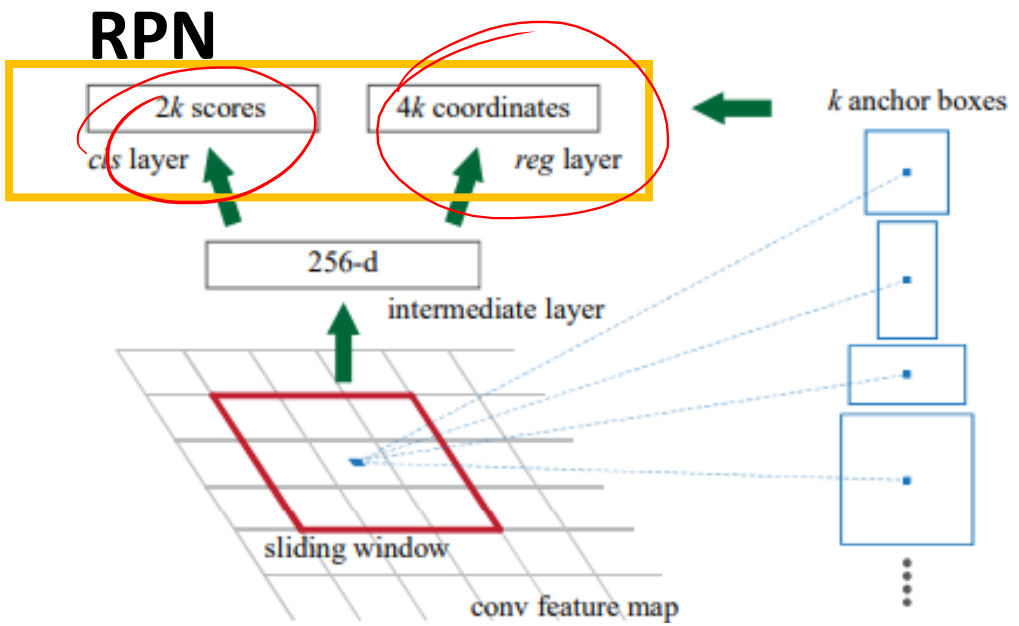
Remaining proposals are then fed to the ROI pooling and then clssifier by the standard Fast-RCNN architecture

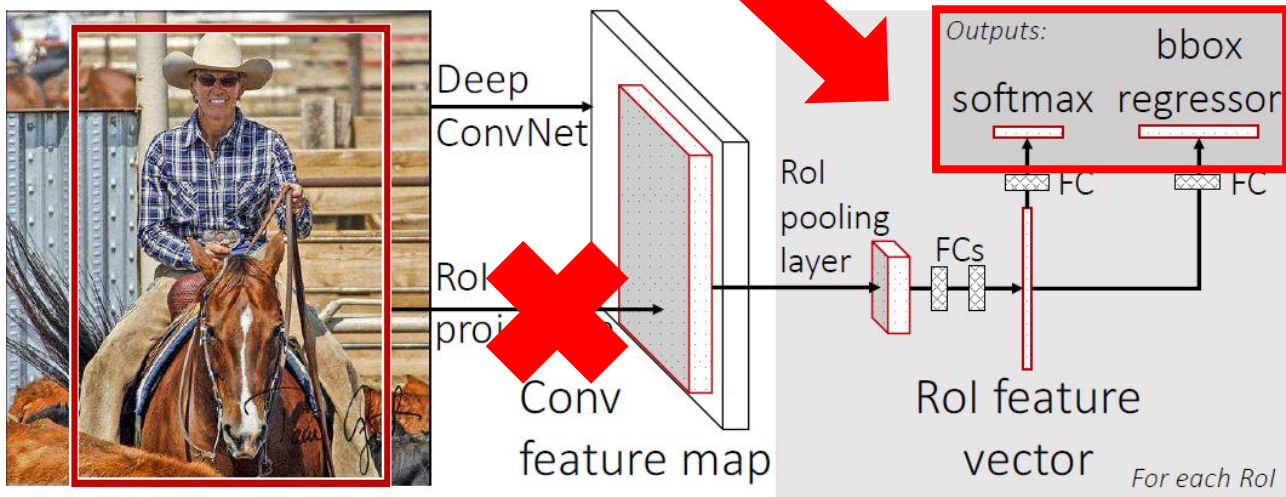


Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN Training

- **Training** now involves 4 losses:
  - RPN classify object/non object
  - RPN regression coordinates
  - Final classification score
  - Final BB coordinates
- During training, object/non object ground truth is defined by measuring the overlap with annotated BB



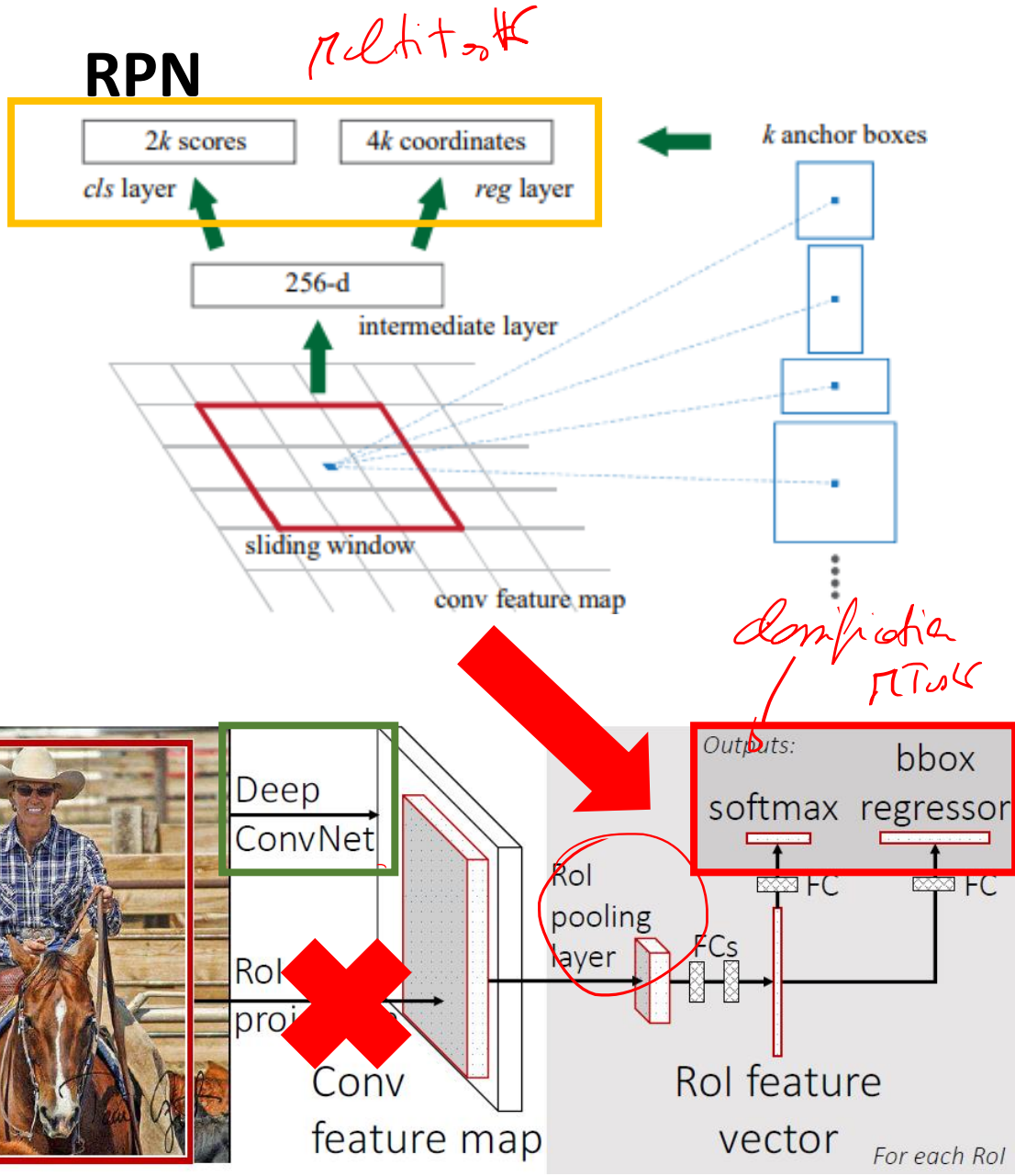


Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN Training

Training procedure

1. **Train RPN** keeping backbone network frozen and training only RPN layers. This ignores object classes but just bounding box locations (Multi-task loss cls + reg)
2. **Train Fast-RCNN** using proposals from **RPN trained before.** Fine tune the whole Fast-RCNN **including the backbone**
3. **Fine tune the RPN in cascade of the new backbone**
4. **Freeze backbone and RPN and fine tune only the last layers of the Faster R-CNN**



Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN

At test time,

- Take the top ~ 300 anchors according to their object scores
- Consider the refined bounding box location of these 300 anchors
- These are the ROI to be fed to a Fast R-CNN
- Classify each ROI and provide the non-background ones as output

Faster R-CNN provides as output to each image a **set of BB** with their **classifier posterior**

The network becomes much faster (0.2s test time per image)

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN

It's still a **two stage detector**

First stage:

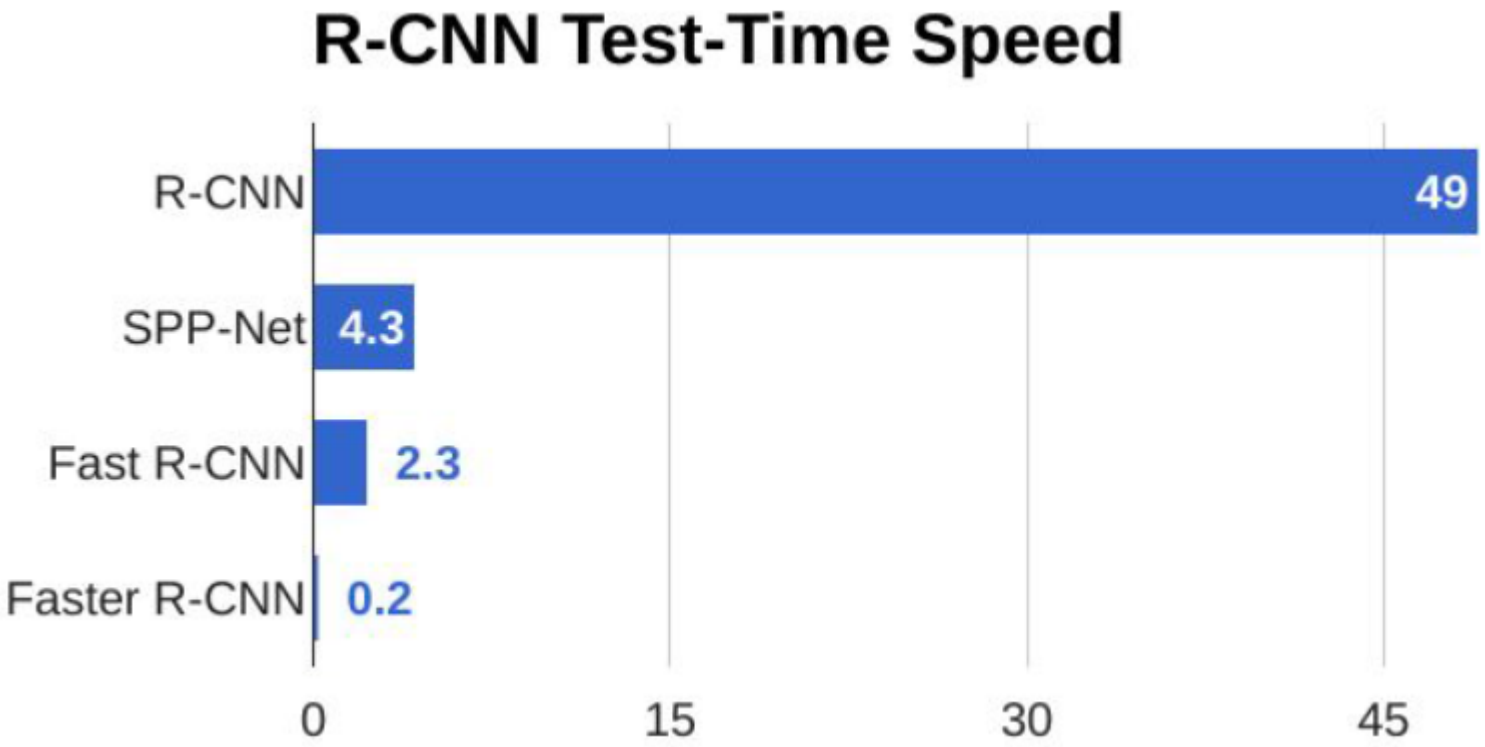
- run a **backbone network** (e.g. VGG16) to **extract featues**
- run the **Region Proposal Network** to **estimate** ~ 300 **ROI**

Second stage (the same as in Fast R-CNN):

- **Crop Features** through ROI pooling (with alignment)
- **Predict object class** using FC + softmax
- **Predict bounding box offset** to improve localization using FC + softmax

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN



Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†

University of Washington*, Allen Institute for AI†, Facebook AI Research¶

http://pjreddie.com/yolo/

G. Boracchi

YOLO/SSD

R-CNN methods are based on region proposals

There are also region-free methods, like:

Yolo: You Only Look Once

SSD: Single Shot Detectors

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." CVPR 2016.

The rationale

Detection networks are indeed a pipeline of multiple steps.

In particular, **region-based methods make it necessary to have two steps** during inference

This can be slow to run and hard to optimize, because each individual component must be trained separately.

In Yolo *"**we reframe the object detection as a single regression problem**, straight from image pixels to bounding box coordinates and class probabilities"*

And solve these regression problems **all at once**, with a large CNN

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.
Liu, Wei, et al. "SSD: Single shot multibox detector." *European conference on computer vision*. Springer, Cham, 2016.

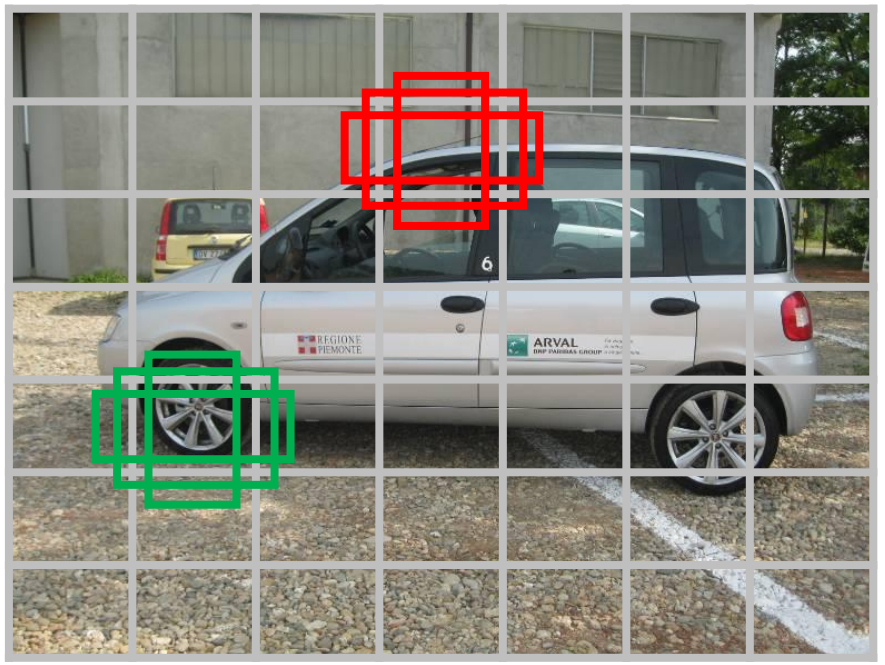
YOLO

1. divide the image in a coarse grid (e.g. 7x7)

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

YOLO

1. divide the image in a coarse grid (e.g. 7x7)
2. each grid cell contains ***B anchors*** *(base bounding box)* associated



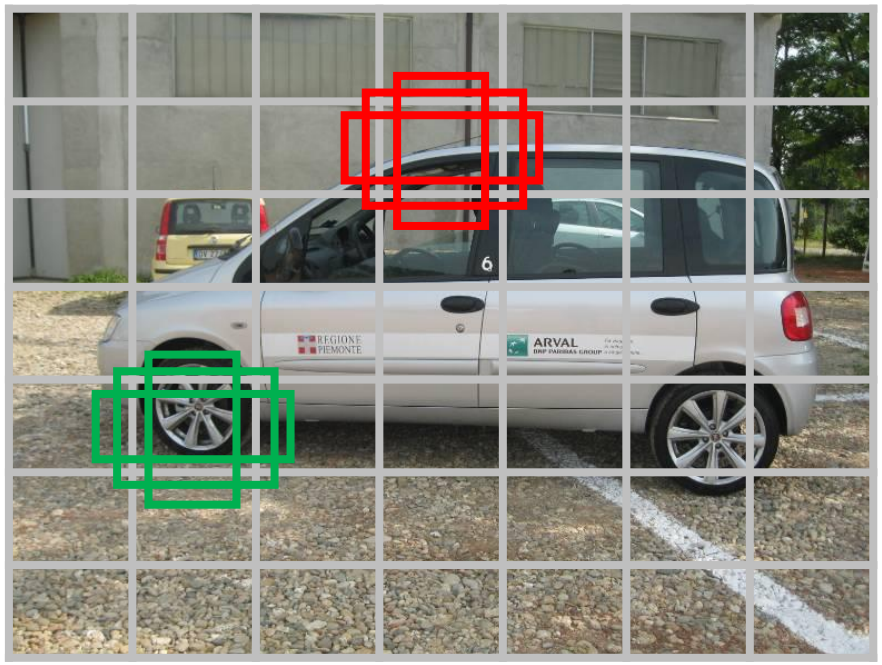
Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

YOLO

3. For each cell and anchor **we predict:**
   - The **offset of the base bounding box**, to better match the object:
     $(dx, dy, dh, dw, objectness\_score)$
   - The **classification score of the base-bounding box** over the C considered categories (including background)

So, the output of the network has dimension

$$7 \times 7 \times B \times (5 + C)$$



Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

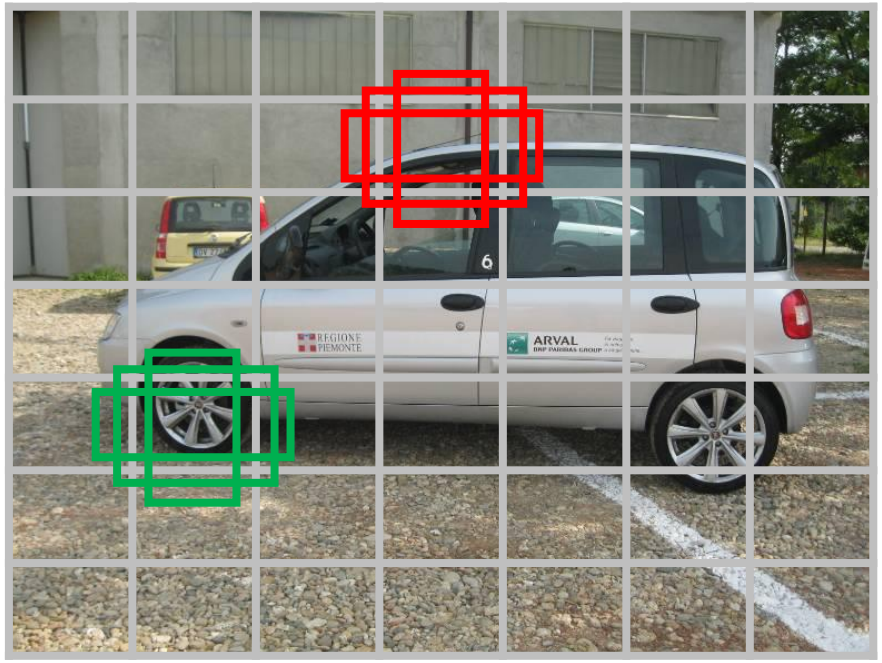
YOLO

The whole **prediction is performed in a single forward pass** over the image, **by a single convolutional network**

Training this network is sort of **tricky to assess the** loss (matched / not matched)

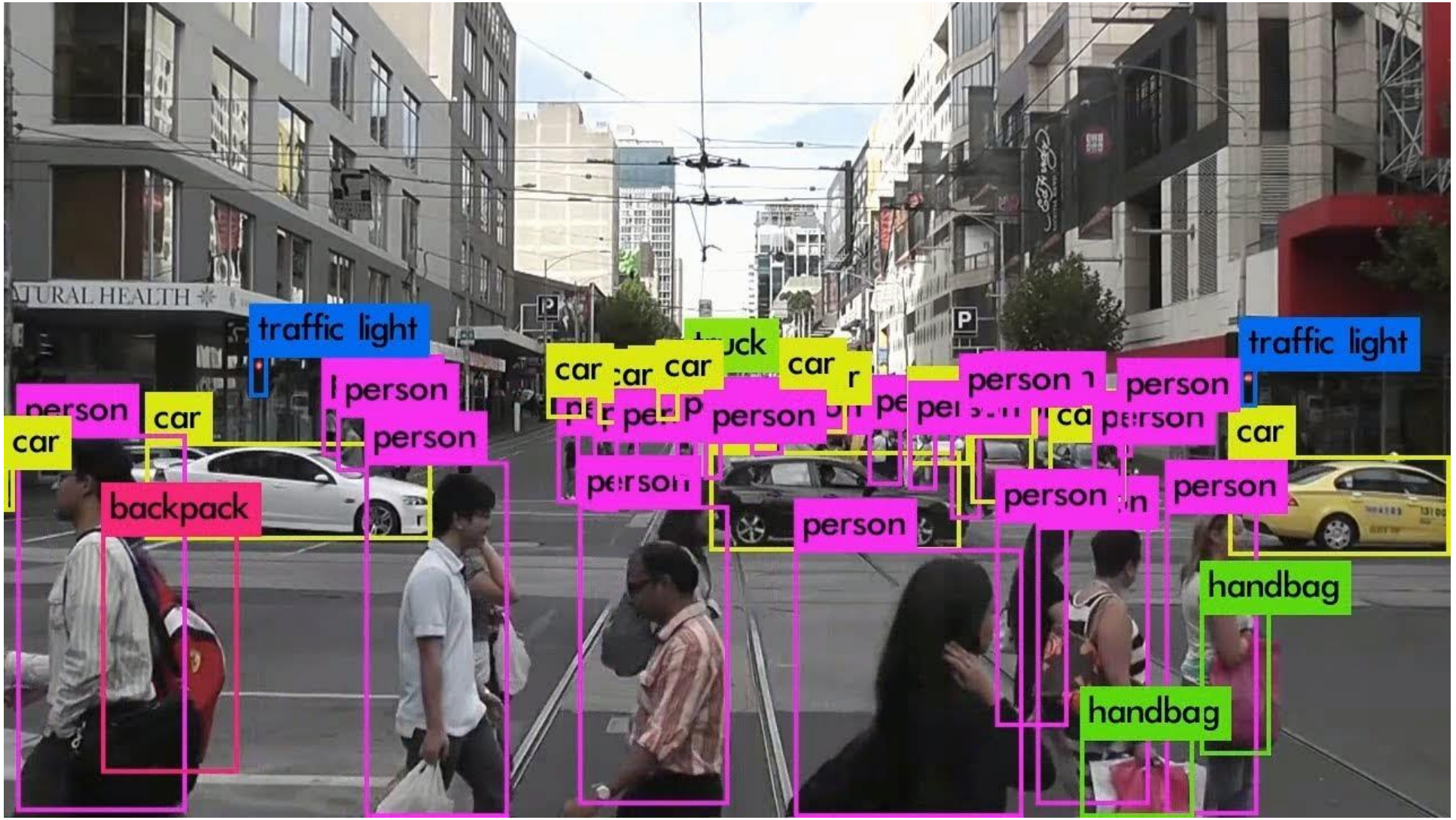
YOLO/SSD shares a similar ground of the RPN used in Faster R-CCN

Typically networks based on region-proposal are more accurate, single shot detectors are faster but less accurate



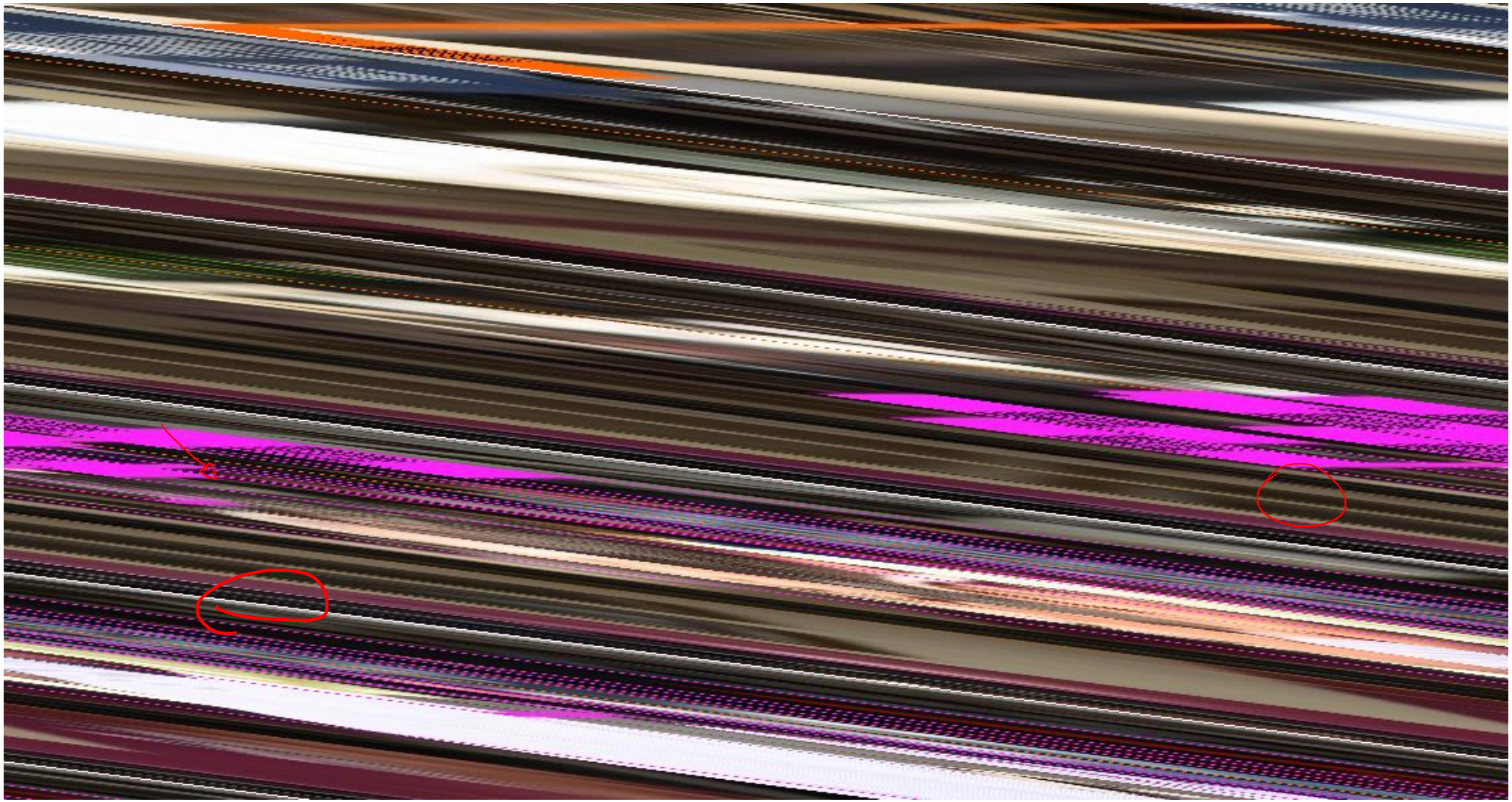
Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

Object Detection



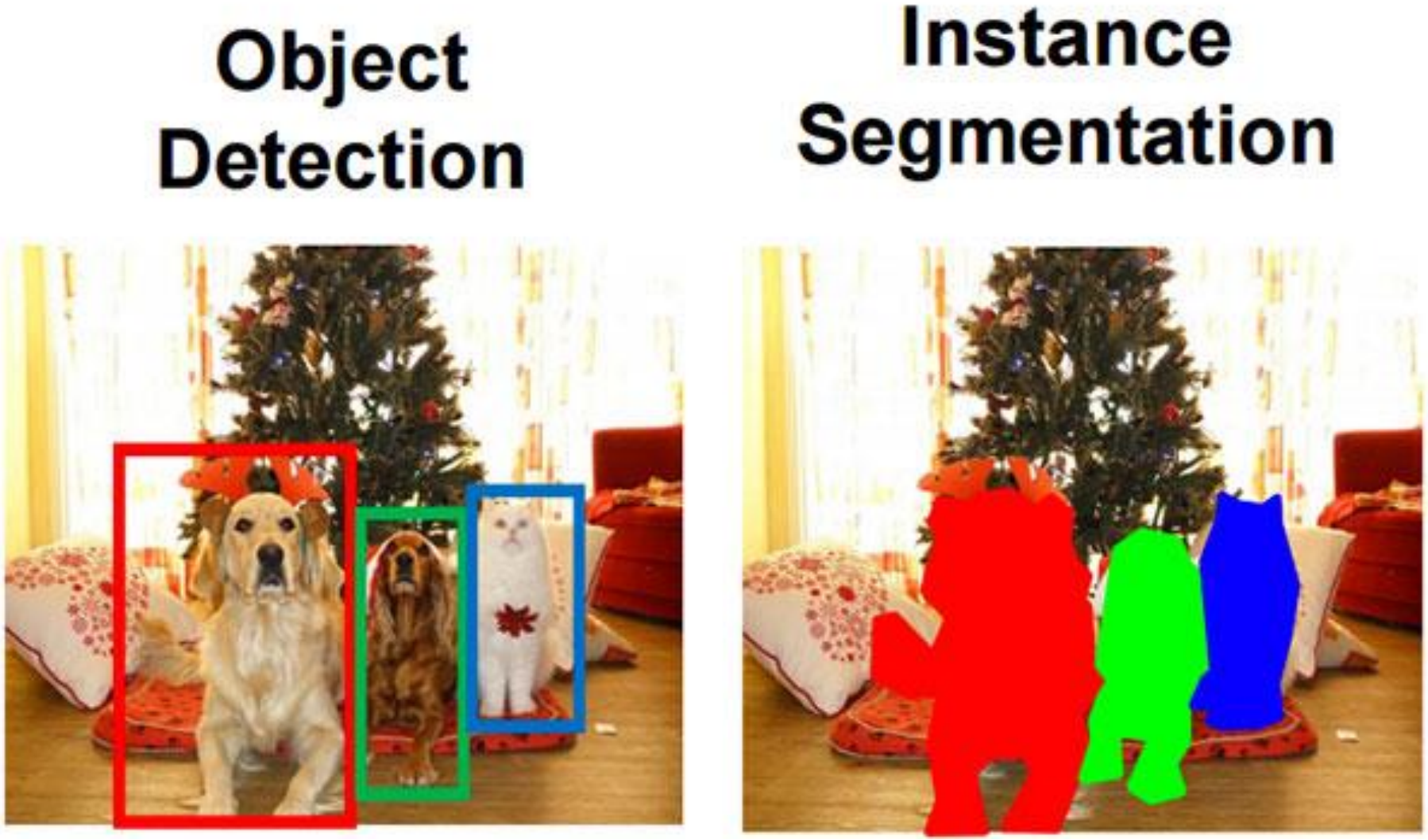
Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Object Detection



Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Object Detection vs Instance Segmentation



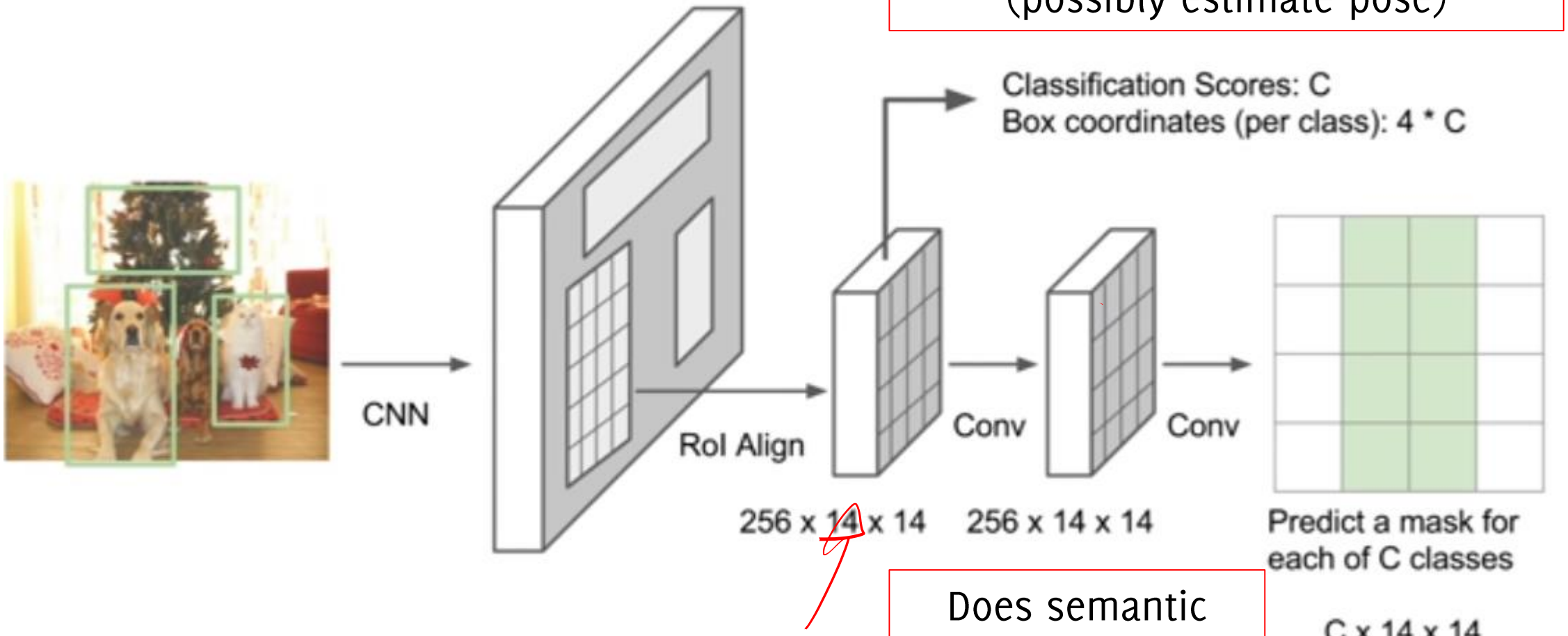
Instance Segmentation

It combines the challenges of:

- **Object detection** (multiple instances present in the image)
- **Semantic segmentation** (associate a label to each pixel) separating each object instance

Mask R-CNN

As in Fast R-CNN classify the whole ROI and regress the bounding box (possibly estimate pose)



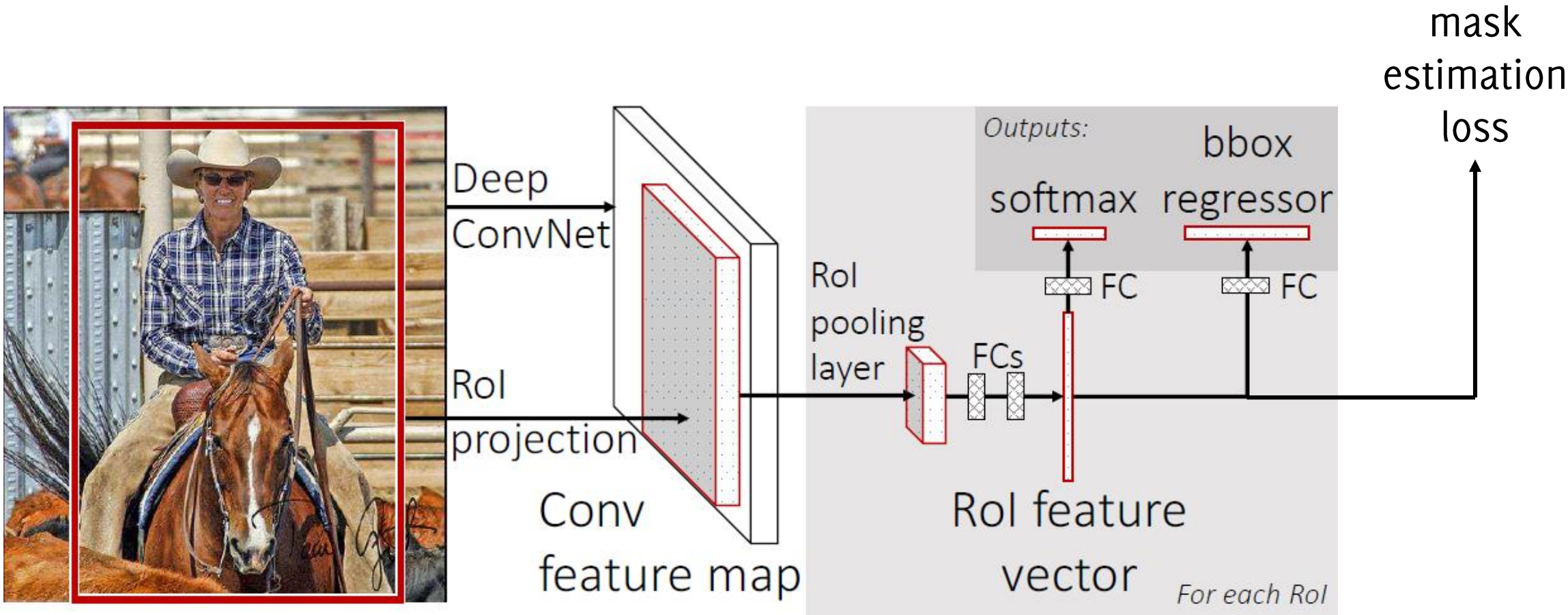
Does semantic segmentation inside each ROI

He et al, "Mask R-CNN", arXiv 2017

He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*

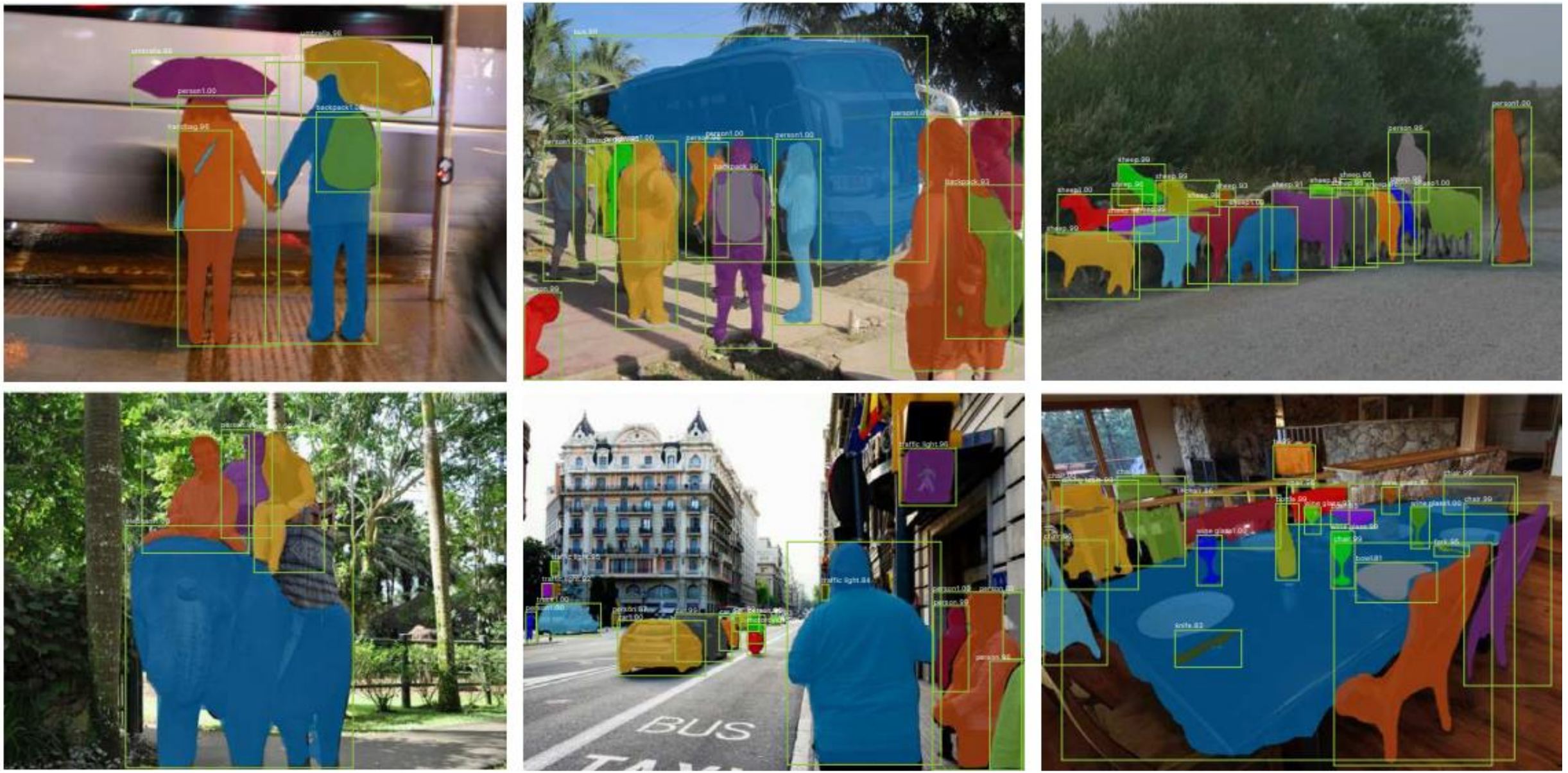
Mask R-CNN

Mask is estimated for each ROI and each class



He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*

Mask R-CNN (end-to-end training)



He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*

Mask R-CNN (including Pose estimation)

He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*

Mask R-CNN (including Pose estimation)

He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*

Mask R-CNN (including Pose estimation)

This can be also trained from segmentation dataset (like COCO dataset), from where you can infer bounding boxes
Microsoft COCO dataset contains 200.000 images segmented over 80 categories. Persons are provided with joints annotated
Since there are many instances per image, this provides a lot of training data

He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*