

Convolutional Neural Networks for Object Detection

Giacomo Boracchi

Artificial Neural Networks and Deep Learning

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

Object Detection Networks

Object Detection



Object Detection, the problem

Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- **multiple** labels $\{l_i\}$ from a fixed set of categories $\Lambda = \{"wheel", "cars", \dots, "castle", "baboon"\}$, each corresponding to an instance of that object
- the coordinates $\{(x, y, h, w)_i\}$ of the bounding box enclosing each object

$$I \rightarrow \{(x, y, h, w, l)_1, \dots, (x, y, h, w, l)_N\}$$

Object Detection Task

$$I: \rightarrow \{(1 \times R \times R \times R \times R)\}$$

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)



Instance Segmentation, the problem

Assign to an input image I :

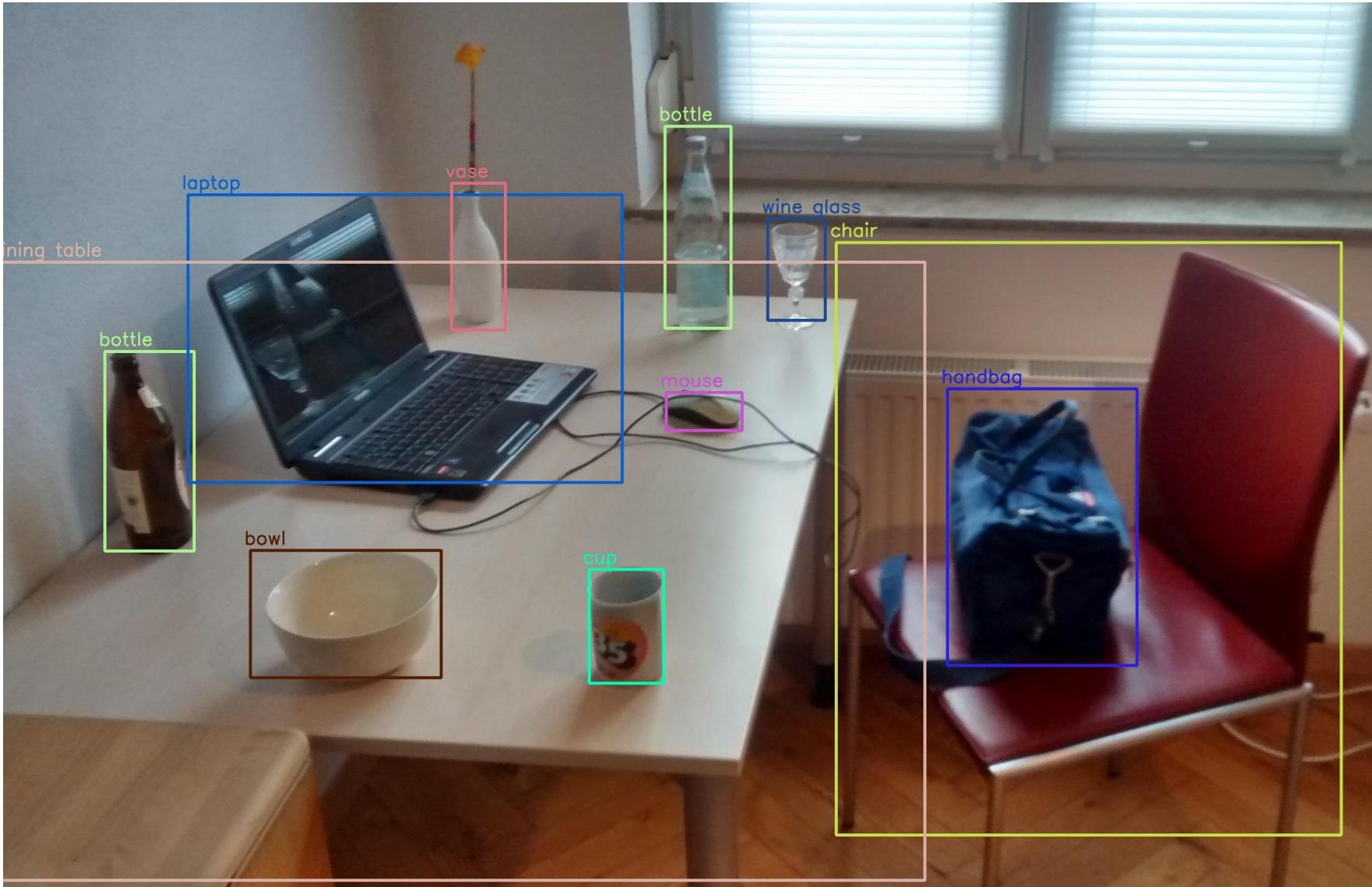
- **multiple labels** $\{l_i\}$ from a fixed set of categories $\Lambda = \{"wheel", "cars", \dots, "castle", "baboon"\}$, each corresponding to **an instance of that object**
- the coordinates $\{(x, y, h, w)_i\}$ of the **bounding box** enclosing each **object**
- the **set of pixels** S in each bounding box corresponding to that label

$$I \rightarrow \{(x, y, h, w, l, S)_1, \dots, (x, y, h, w, l, S)_N\}$$

Instance Segmentation



Annotated Dataset for Object Detection



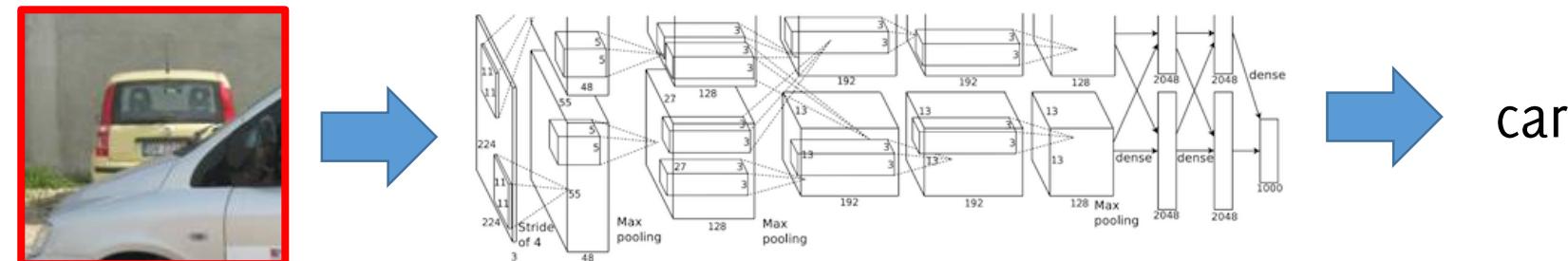
The Straightforward Solution: Sliding Window

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.

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



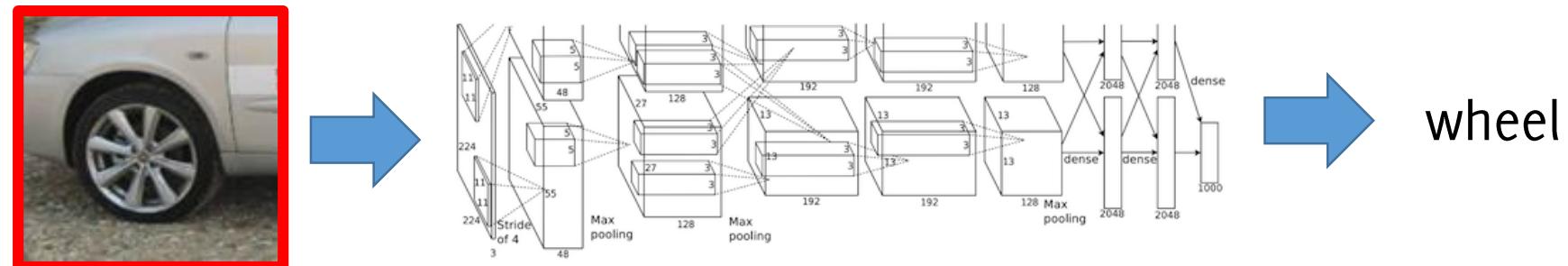
The Straightforward Solution: Sliding Window

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.

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



The Straightforward Solution: Sliding Window

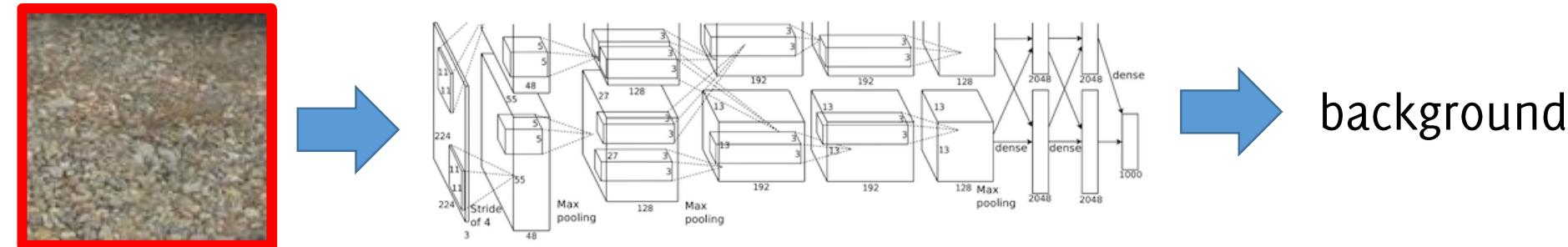
1000 x 2000 pixels



- 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.

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

The background class has to be included!



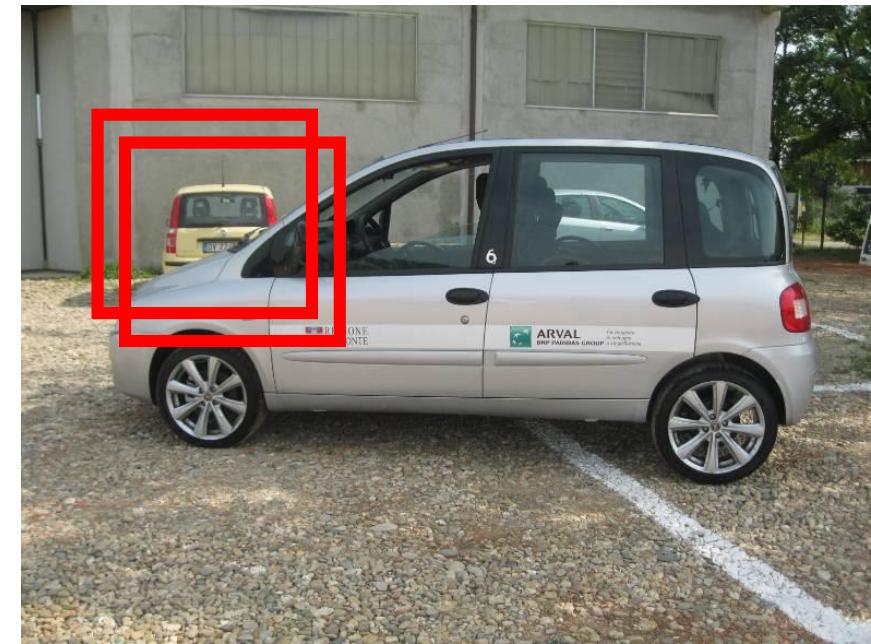
Many drawbacks...

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



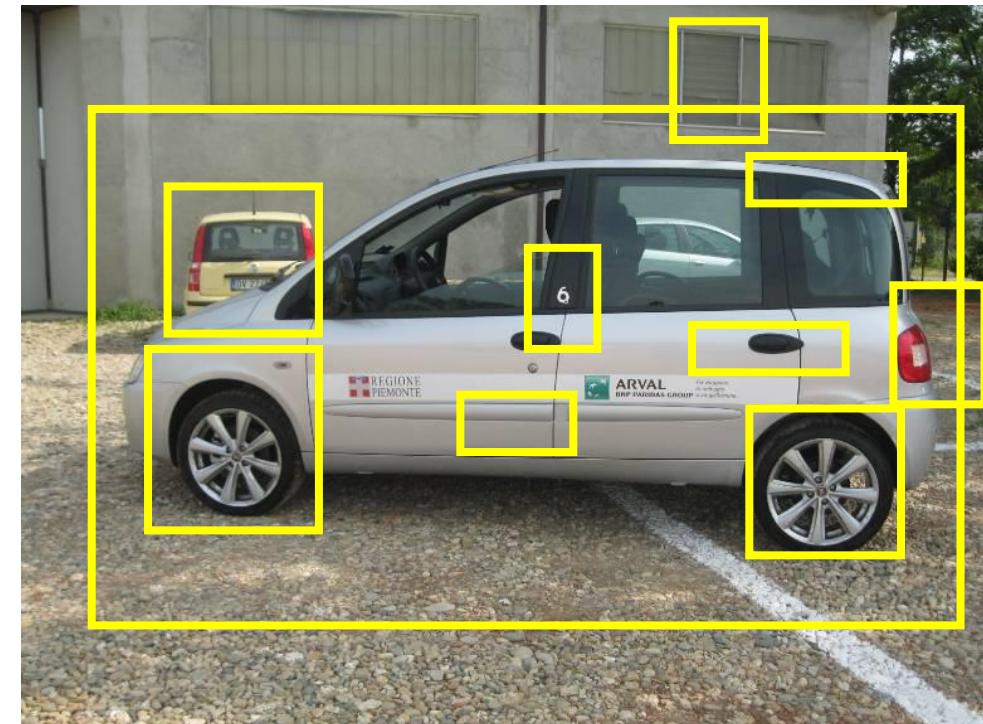
Region Proposal

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





This CVPR2014 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

Rich feature hierarchies for accurate object detection and semantic segmentation

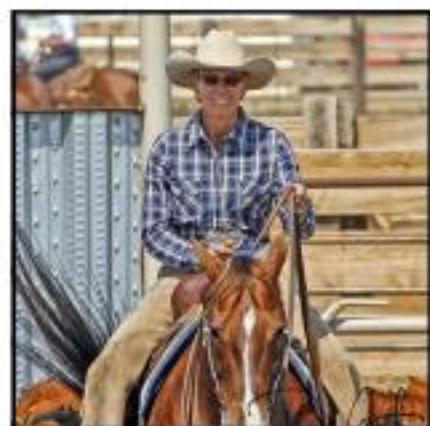
Ross Girshick¹ Jeff Donahue^{1,2} Trevor Darrell^{1,2} Jitendra Malik¹

¹UC Berkeley and ²ICSI

{rbg, jdonahue, trevor, malik}@eecs.berkeley.edu

R-CNN

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



1. Input
image

«external» Region
Proposal Algorithm

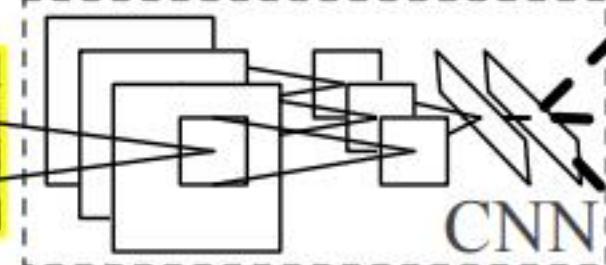


2. Extract region
proposals (~2k)

warped region



Latent features from
AlexNet Pretrained



SVM +
BB regressor

aeroplane? no.

⋮
person? yes.

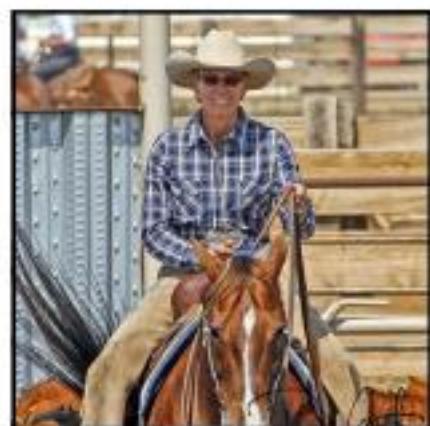
⋮
tvmonitor? no.

3. Compute
CNN features

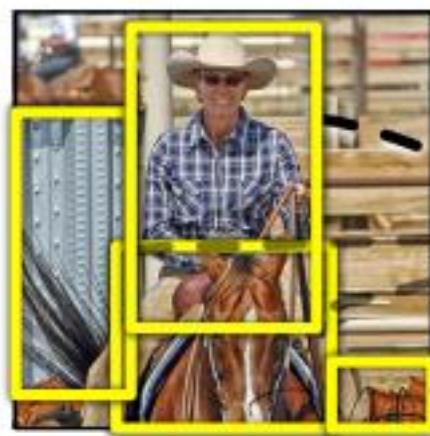
4. Classify
regions

R-CNN

Object detection by means of region proposal



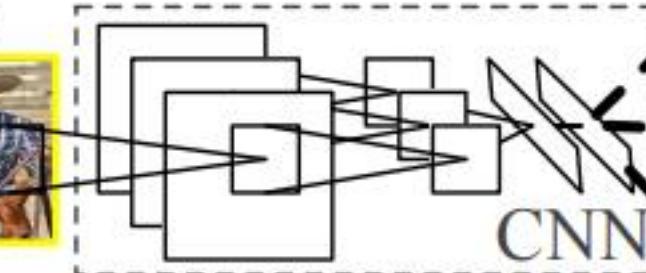
1. Input
image



2. Extract region
proposals (~2k)

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

warped region



SVM +
BB regressor

aeroplane? no.

⋮
person? yes.

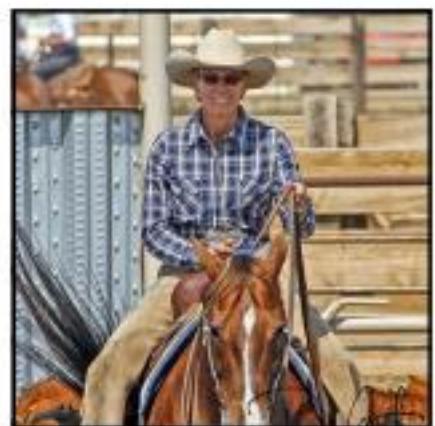
⋮
tvmonitor? no.

3. Compute
CNN features

4. Classify
regions

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

R-CNN



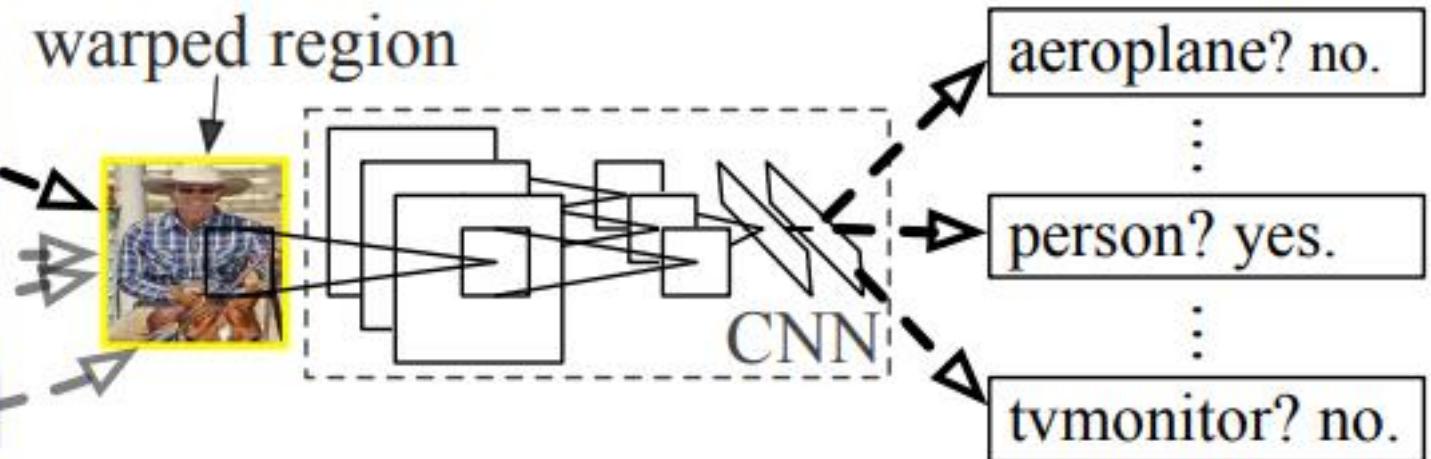
1. Input image



2. Extract region proposals (~2k)

This is AlexNet as a pre-trained network

warped region

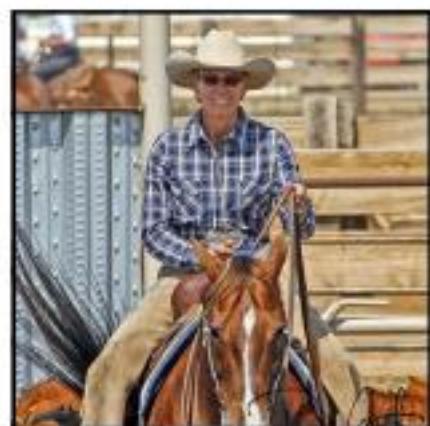


3. Compute CNN features

4. Classify regions

SVM trained to minimize the classification error over the extracted ROI

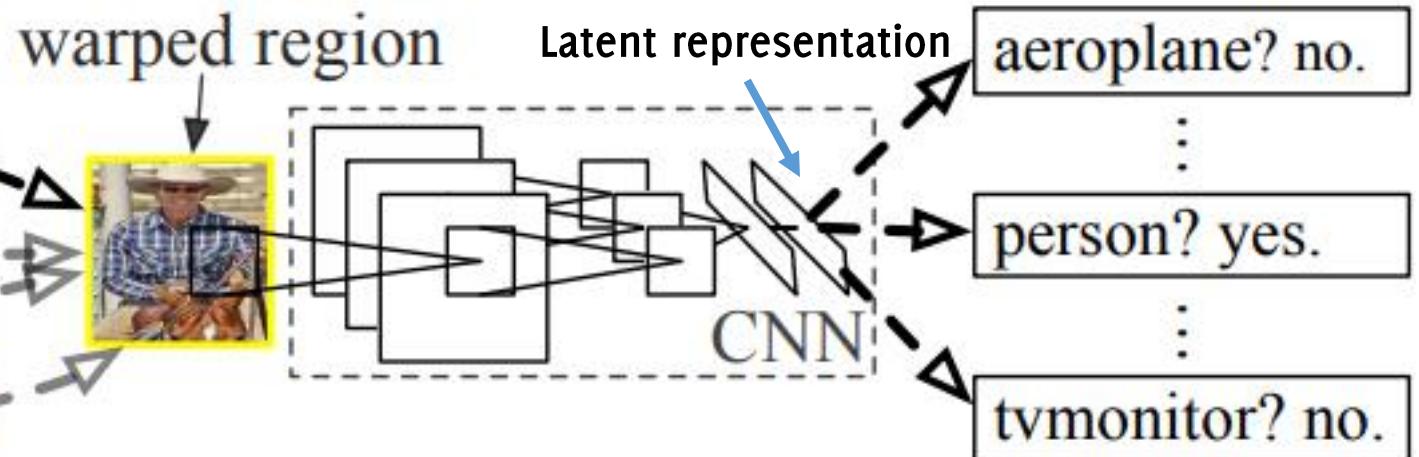
R-CNN



1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features

4. Classify regions

R-CNN

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

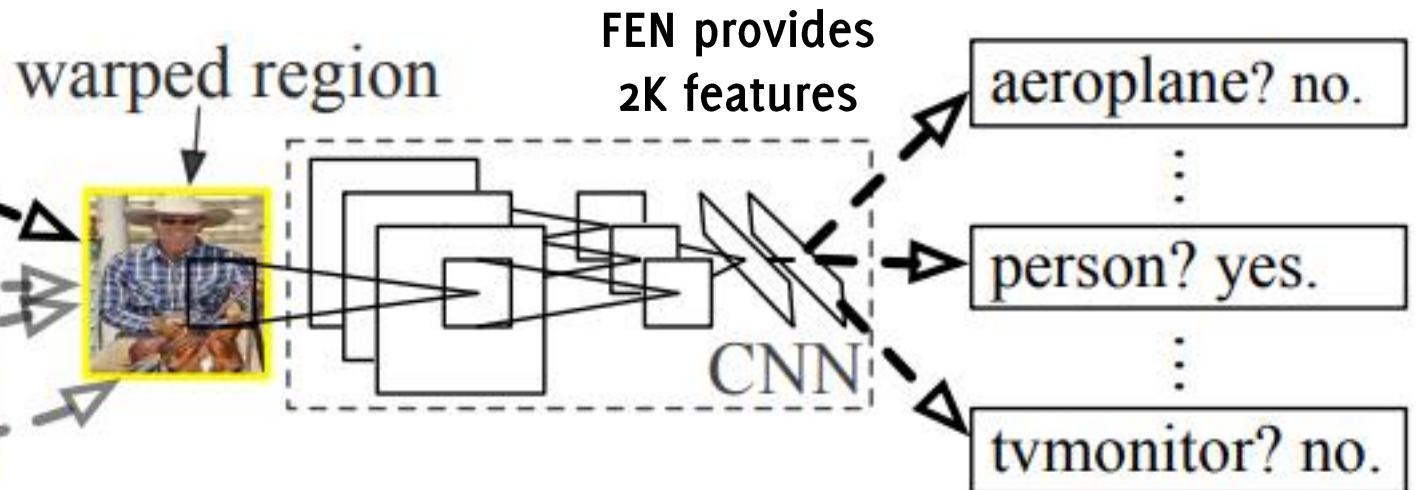
SVM +
BB regressor



1. Input
image



2. Extract region
proposals (~2k)



3. Compute
CNN features

4. Classify
regions

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

Rmk: efficiency in object detection network is key! Otherwise you might want to train a segmentation network instead!

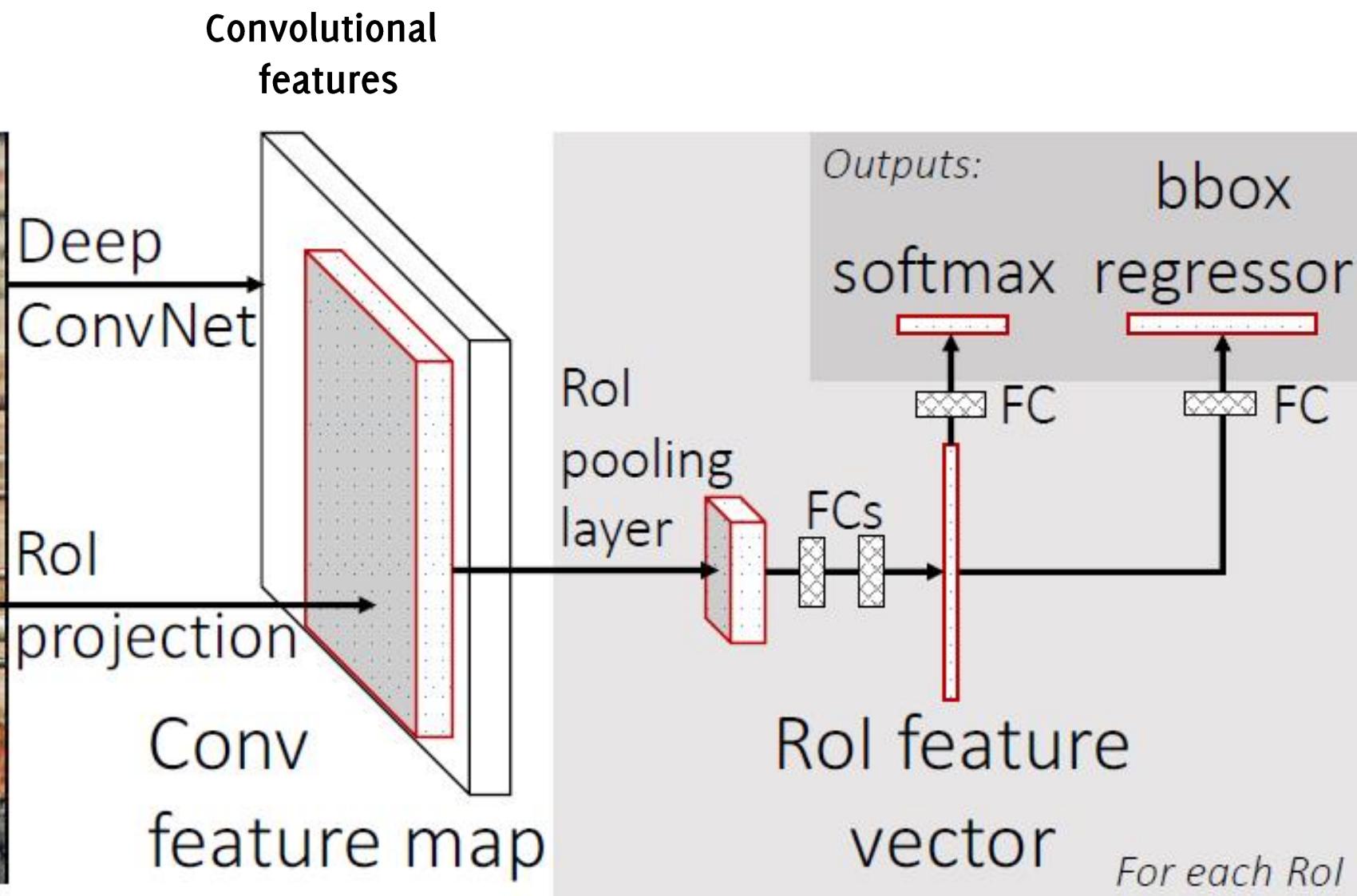
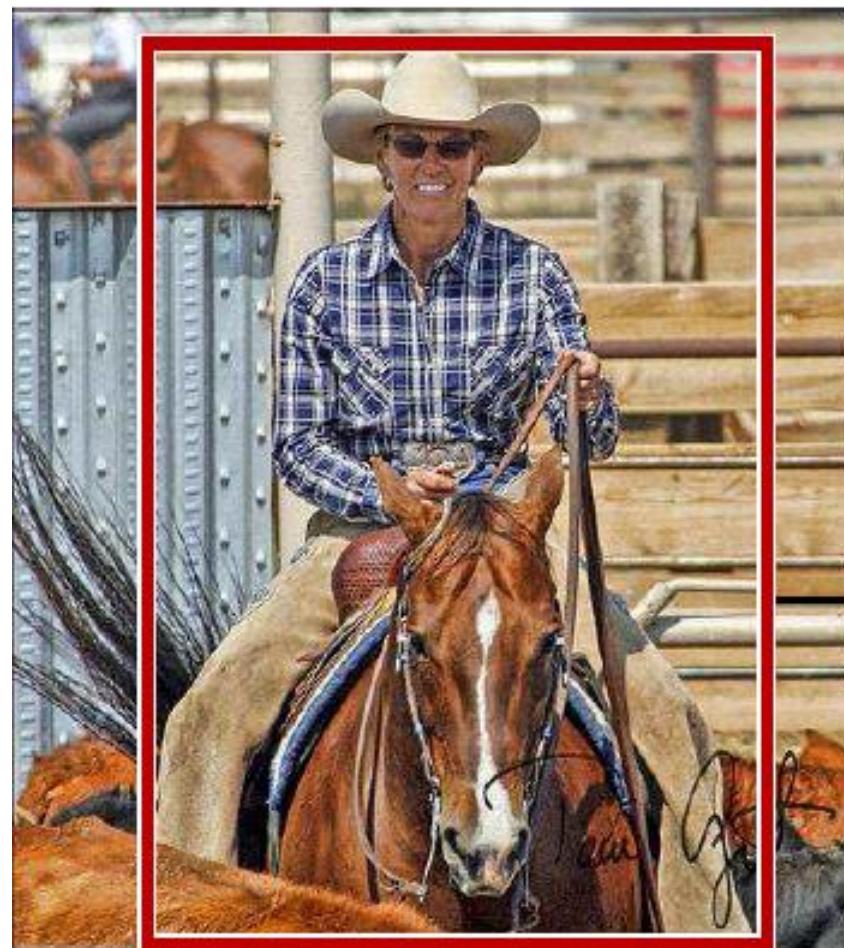


This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

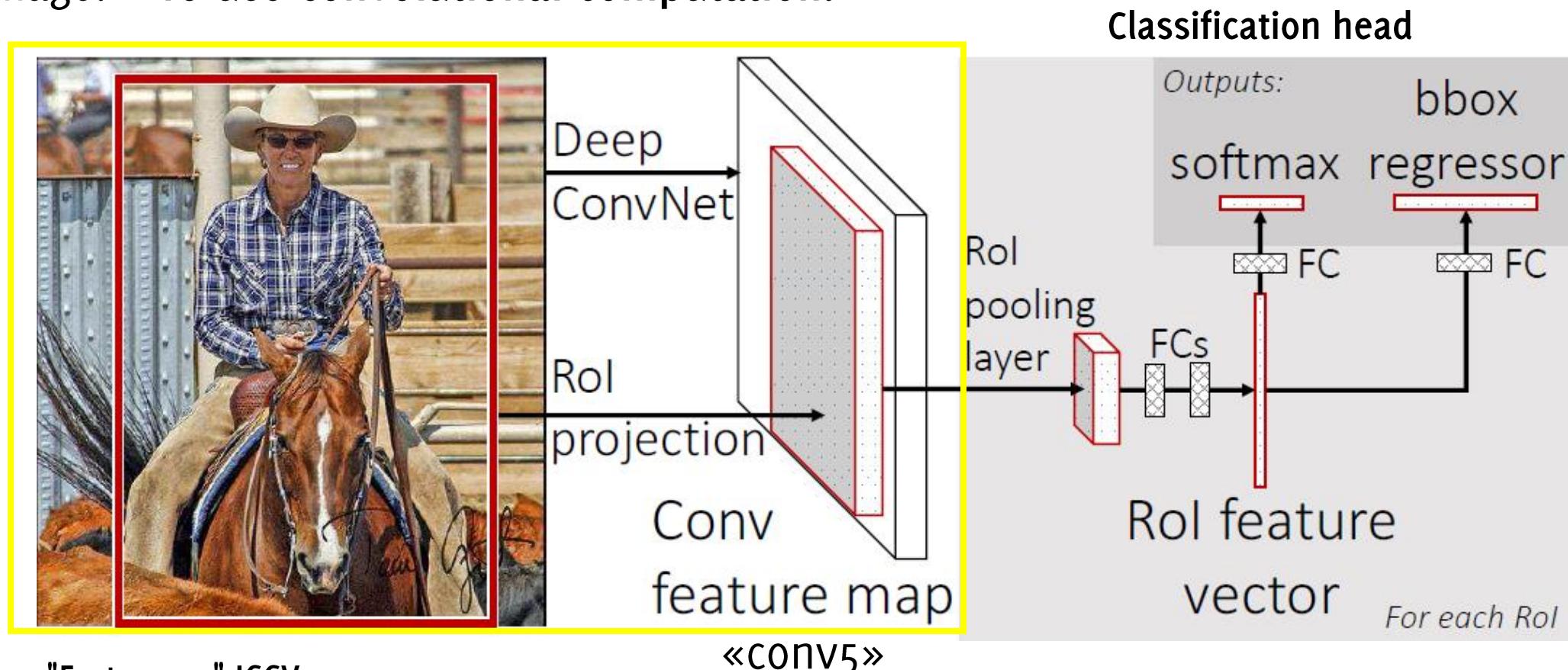
Fast R-CNN



«conv5»

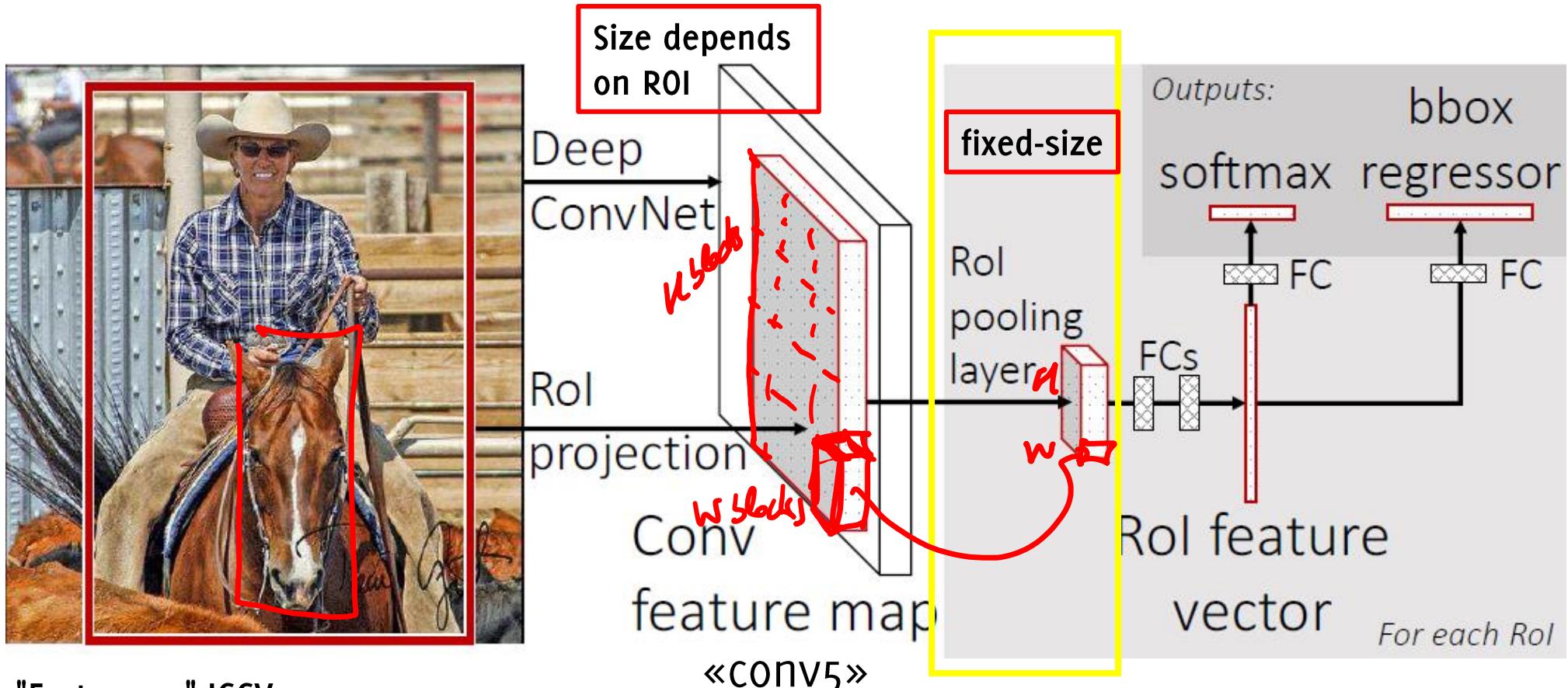
Fast R-CNN

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 from the feature maps, instead from the image: →re-use convolutional computation.



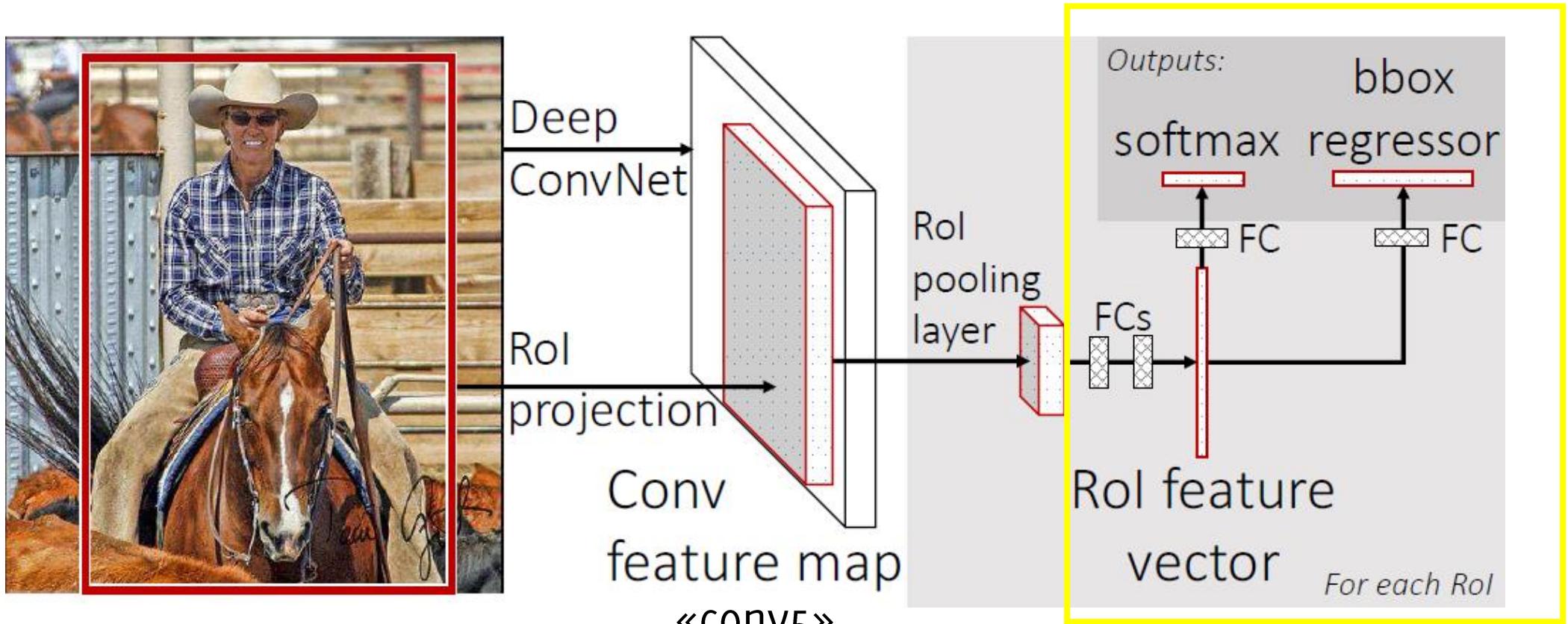
Fast R-CNN

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



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, convolutional part executed only once



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)



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

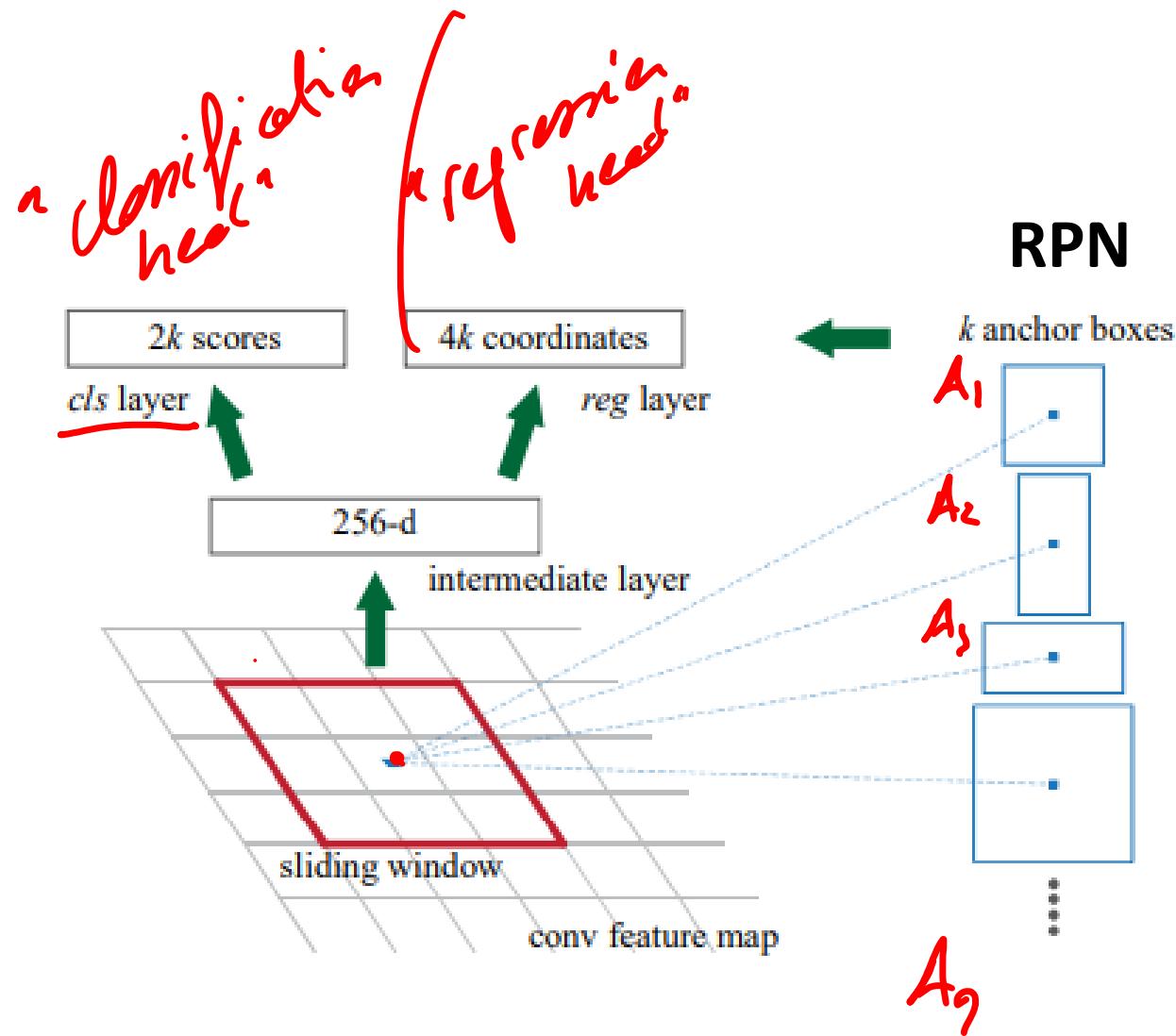
Shaoqing Ren* **Kaiming He** **Ross Girshick** **Jian Sun**

Microsoft Research

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

Faster R-CNN

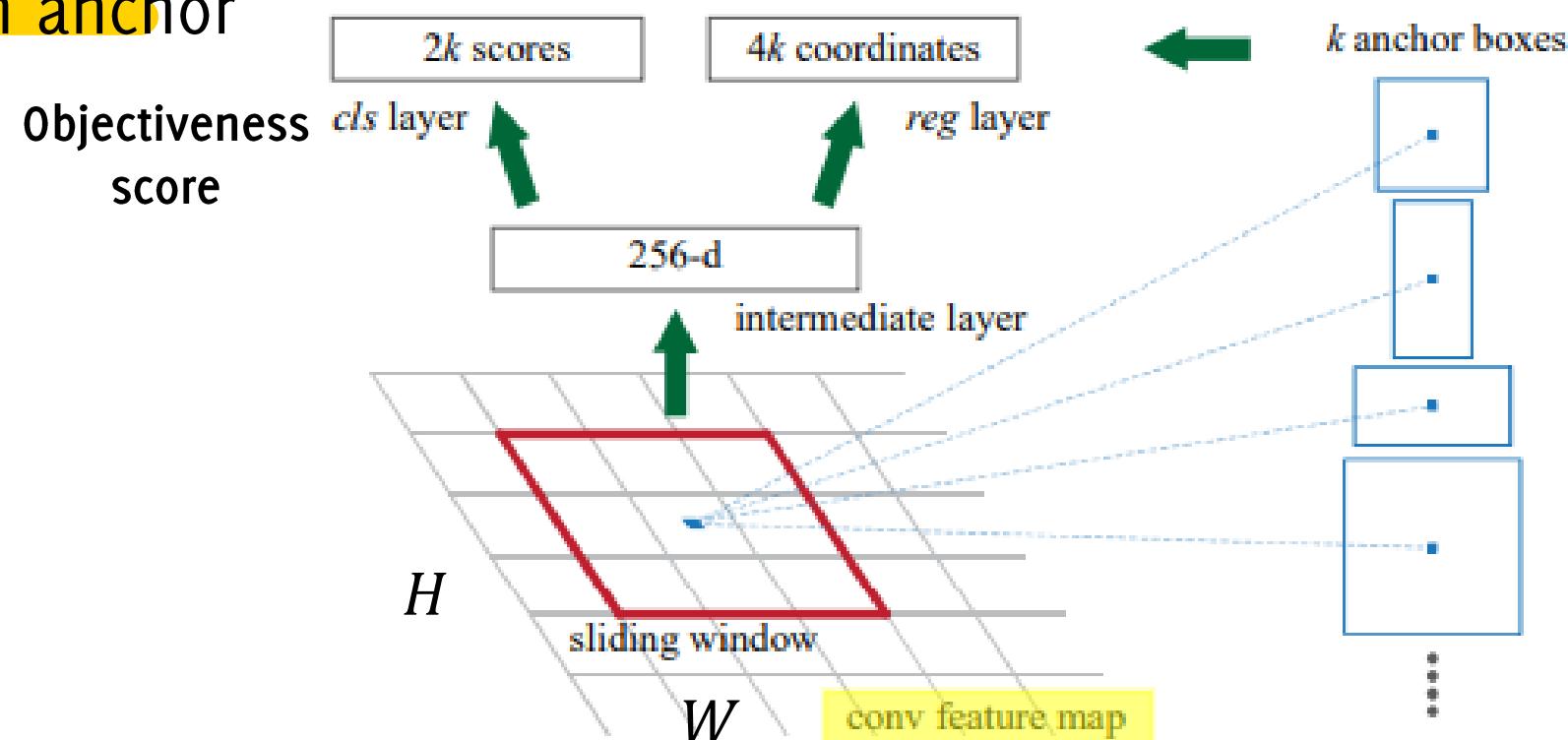
- Instead of the ROI extraction algorithm, train a **region proposal network (RPN)**, which is a F-CNN (3×3 filter size)
- RPN operates on the same feature maps used for classification, thus at the last conv layers
- RPN can be seen as **an additional module that improves efficiency** and focus Fast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



RPN, Region Proposal Network

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

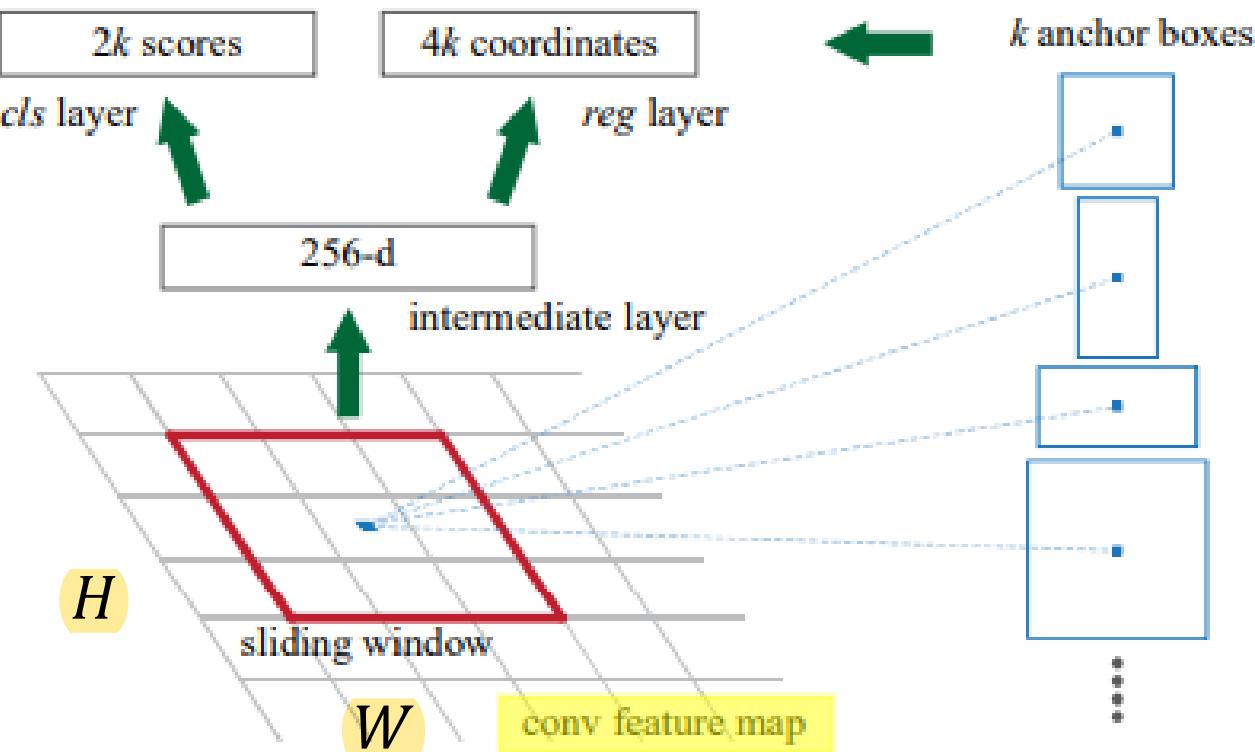
The network outputs $H \times W \times k$ candidates anchor and estimate objectiveness scores for each anchor



RPN: Intermediate Layer

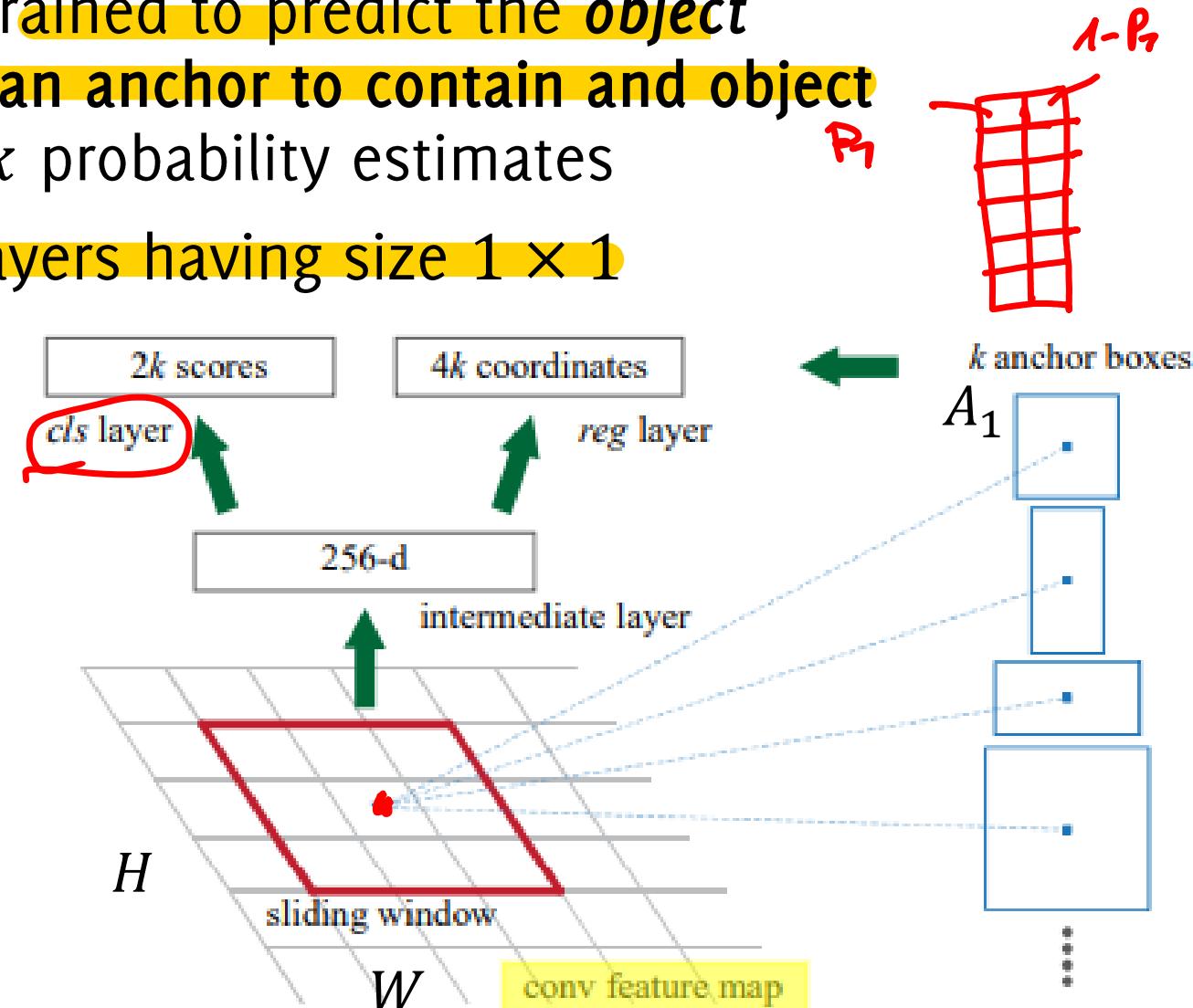
Intermediate layer: is a standard CNN layer that takes as input the last layer of the feature-extraction network and uses 256 filters of size 3×3 .

It **reduces the dimensionality of the feature map** and maps each region to a lower dimensional vector of size (output size $H \times W \times 256$)



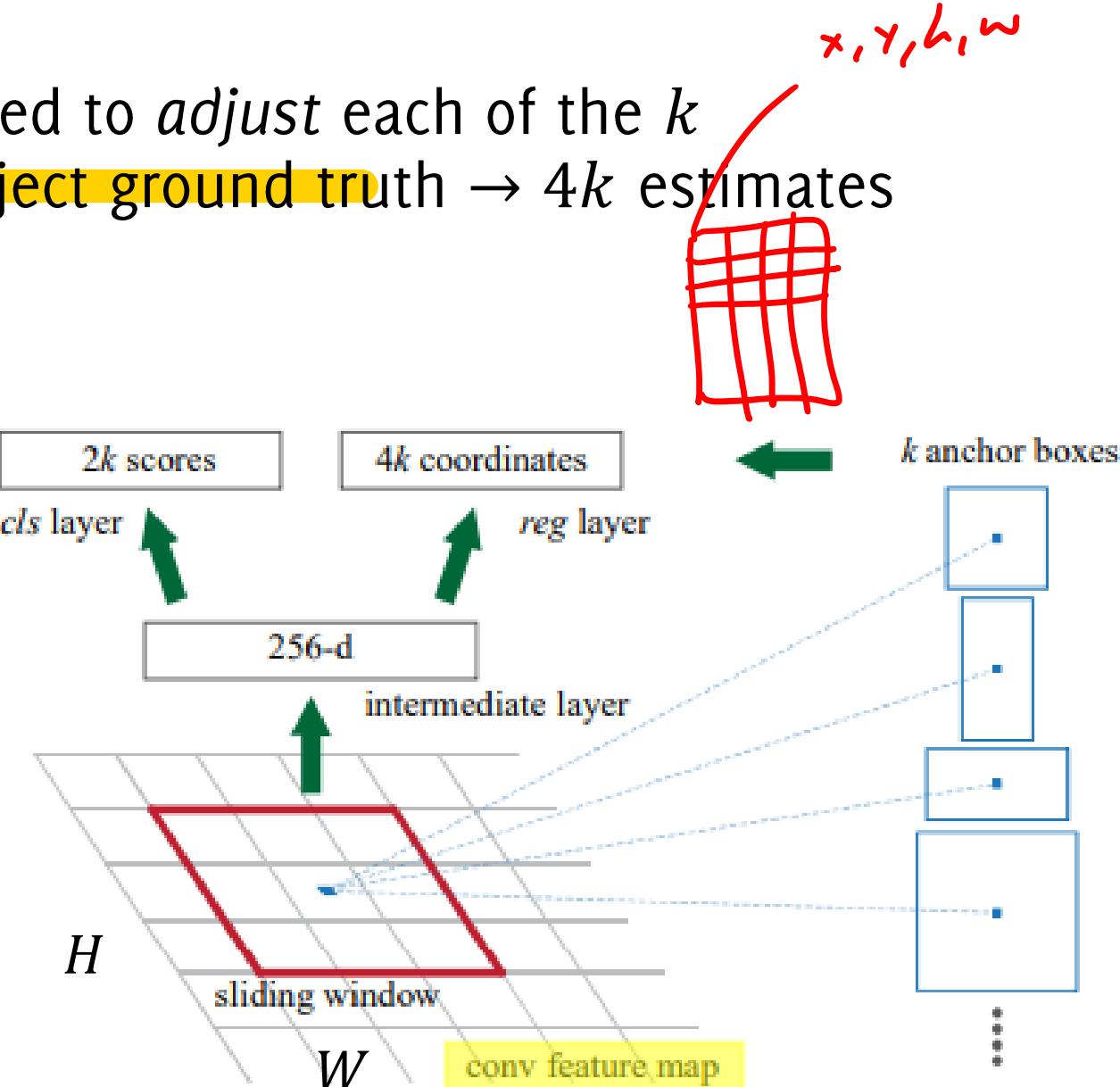
RPN Estimating k Anchors

- 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]** → $2k$ probability estimates
- Made of a **stack of convolutional layers having size 1×1**
- Each of these **k probability 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**



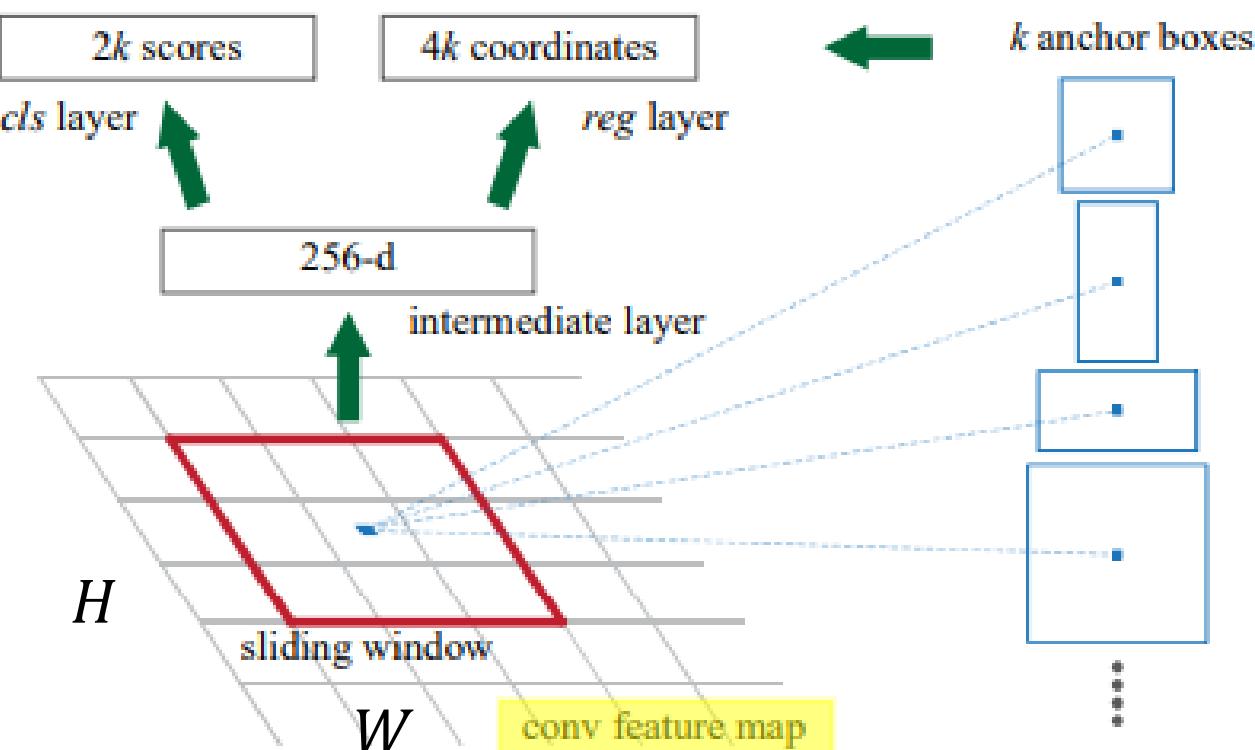
RPN Estimating k anchors

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



RPN estimating k anchors

- If you want to train the network to predict different anchors, there is no need to design different RPN, 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



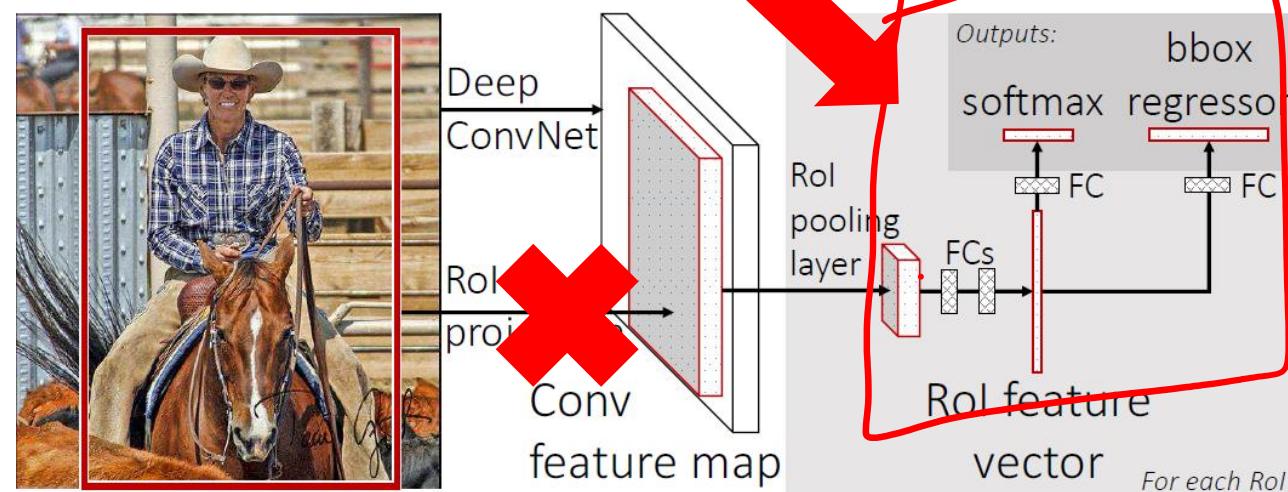
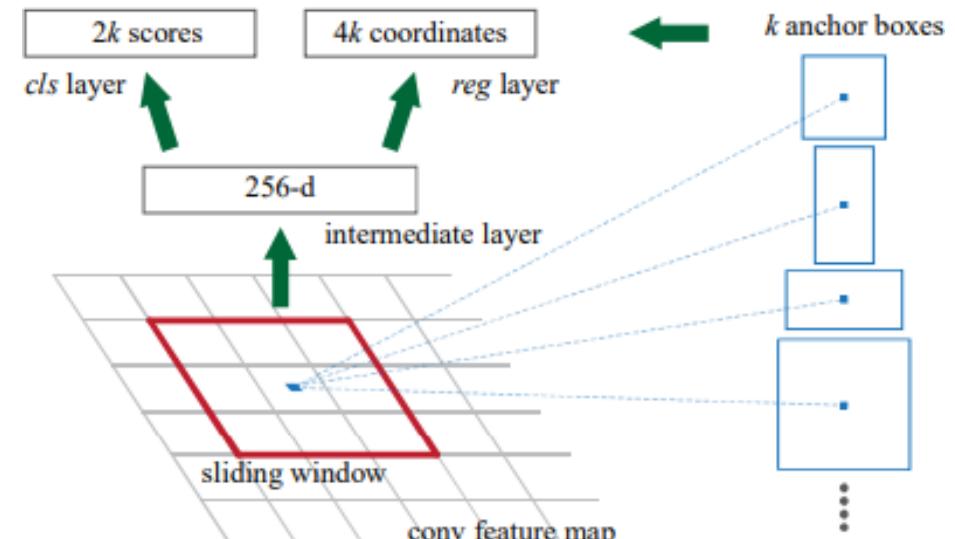
Faster R-CNN

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

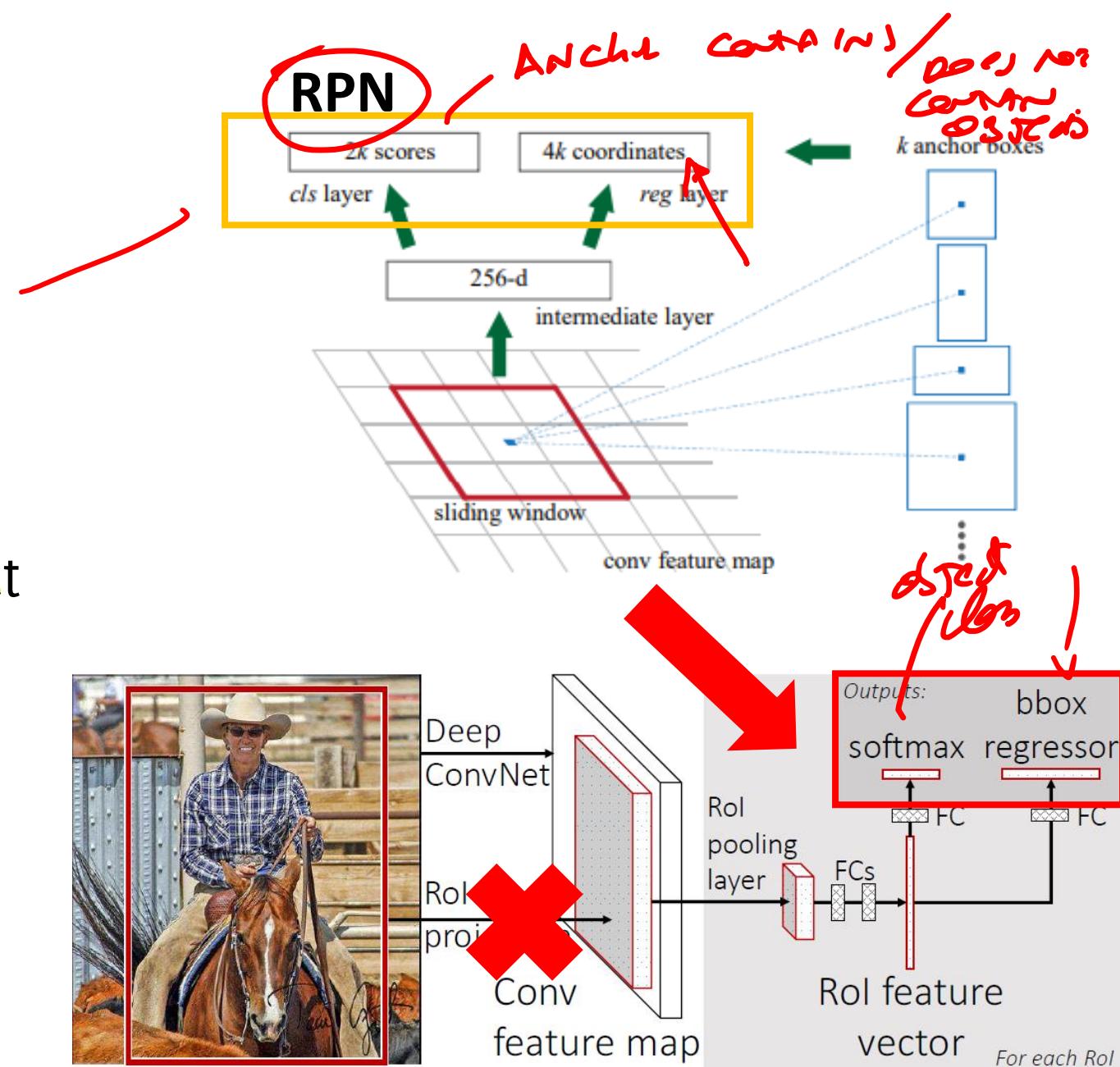
Remaining proposals are then fed to the ROI pooling and then classified by the standard Fast-RCNN architecture

RPN



Faster R-CNN Training

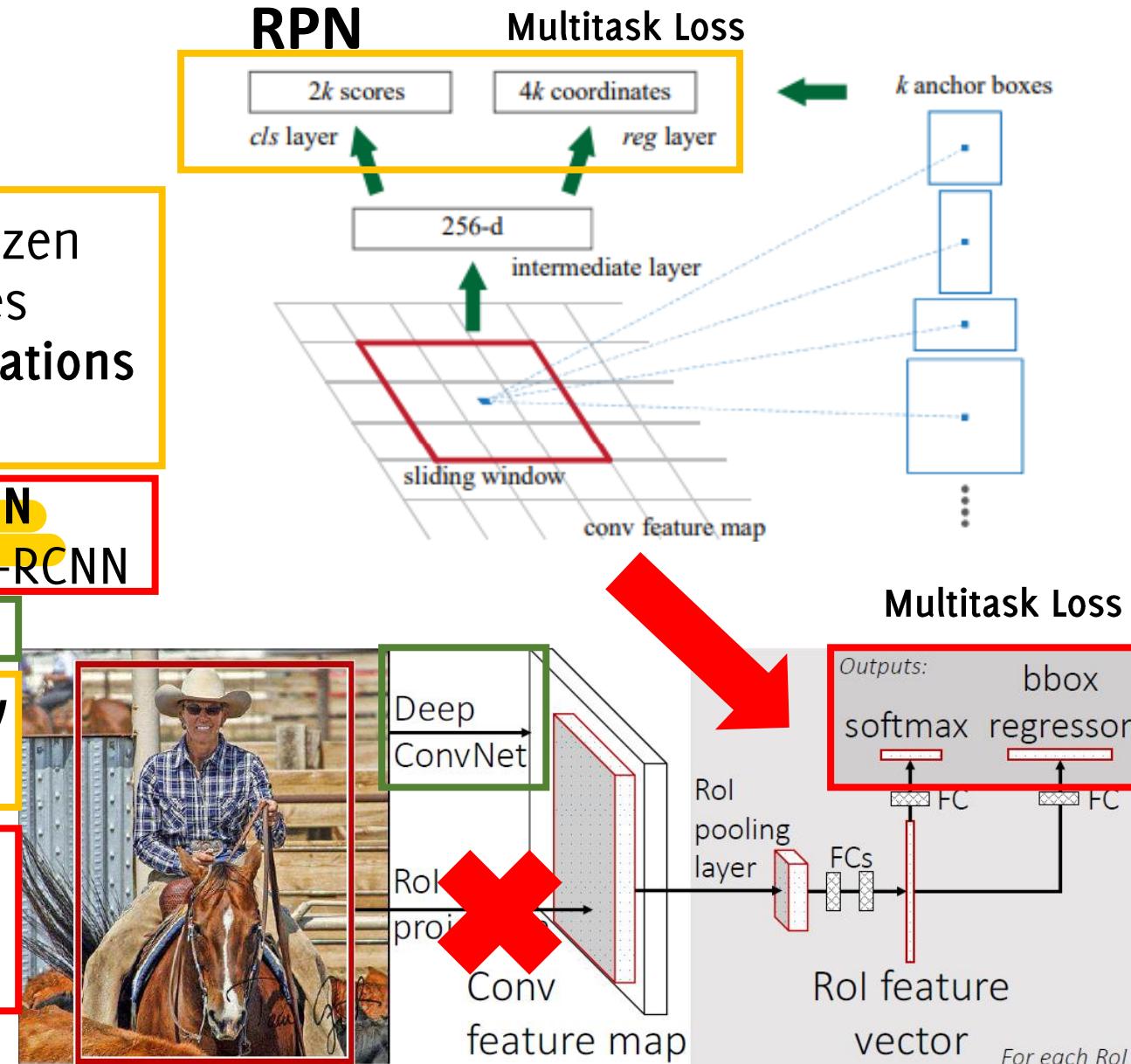
- Training now involves 4 losses:
 - RPN classify object/non object
 - RPN regression coordinates
 - Final classification score
 - Final BB coordinates
- During training, object/non object ground truth is defined by measuring the overlap with annotated BB
- The loss include a term of final BB coordinates, as these are defined over the image, RPN in the latent domain.



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 $\text{cls} + \text{reg}$)
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



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)

Faster R-CNN

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

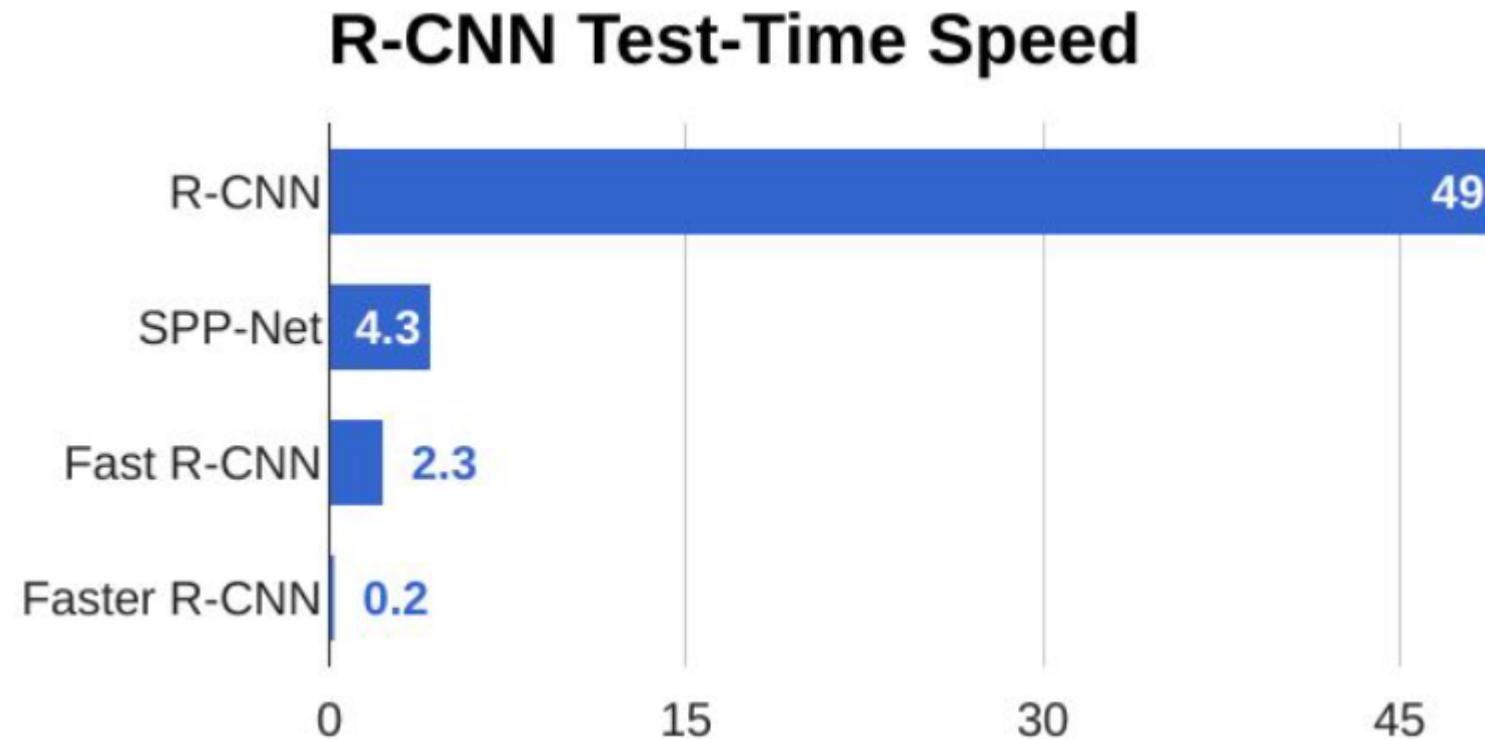
First stage:

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

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

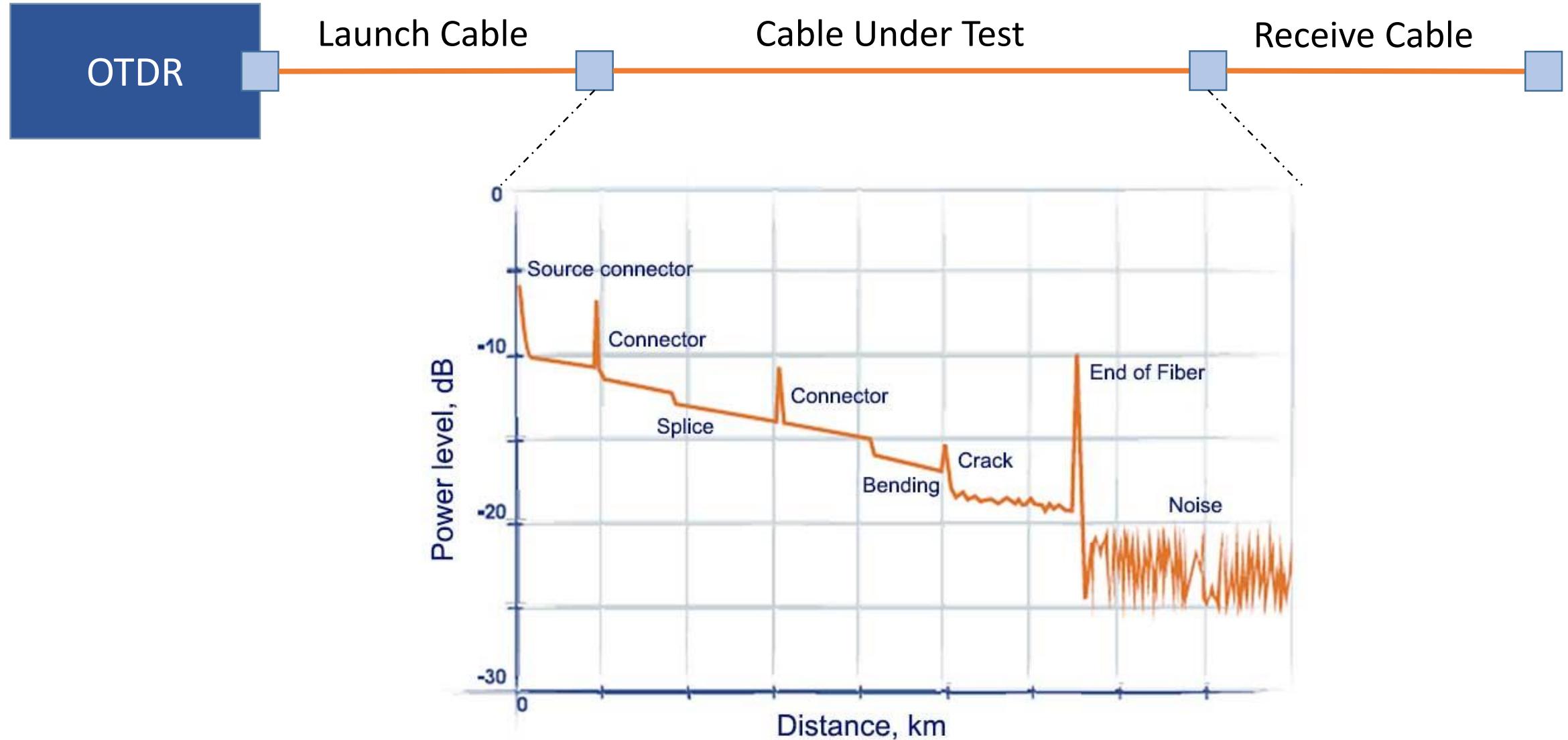
- Crop Features through ROI pooling (with alignment)
- Predict object class using FC + softmax
- Predict bounding box offset to improve localization using FC + softmax

Faster R-CNN

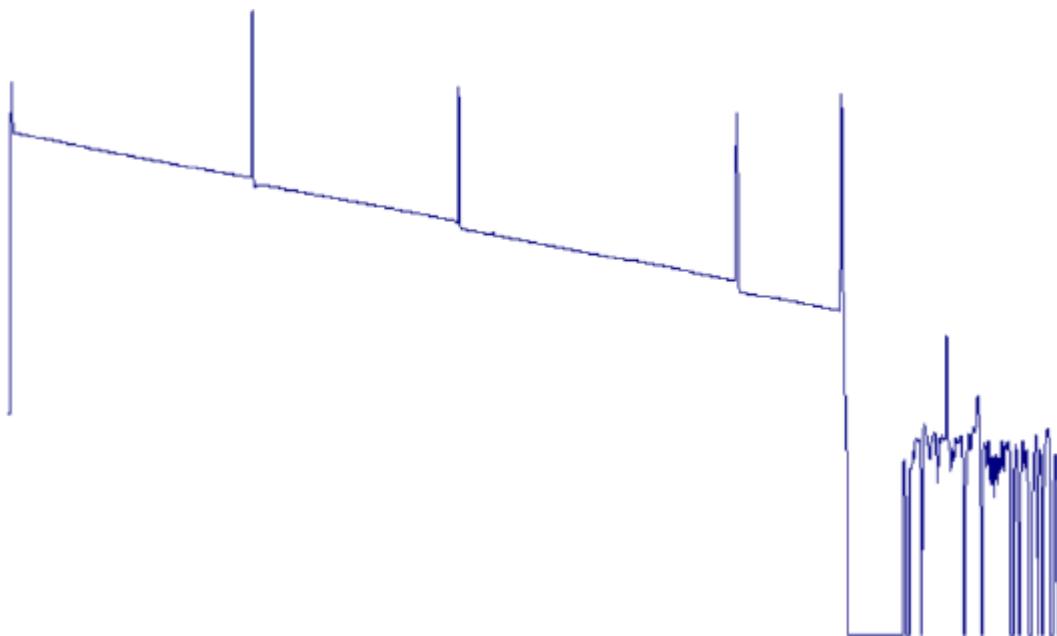


An industrial application

Event Detection in Optical Signals



Event Detection in Optical Signals



OTDR trace: events indicate problems over a span of optical fiber

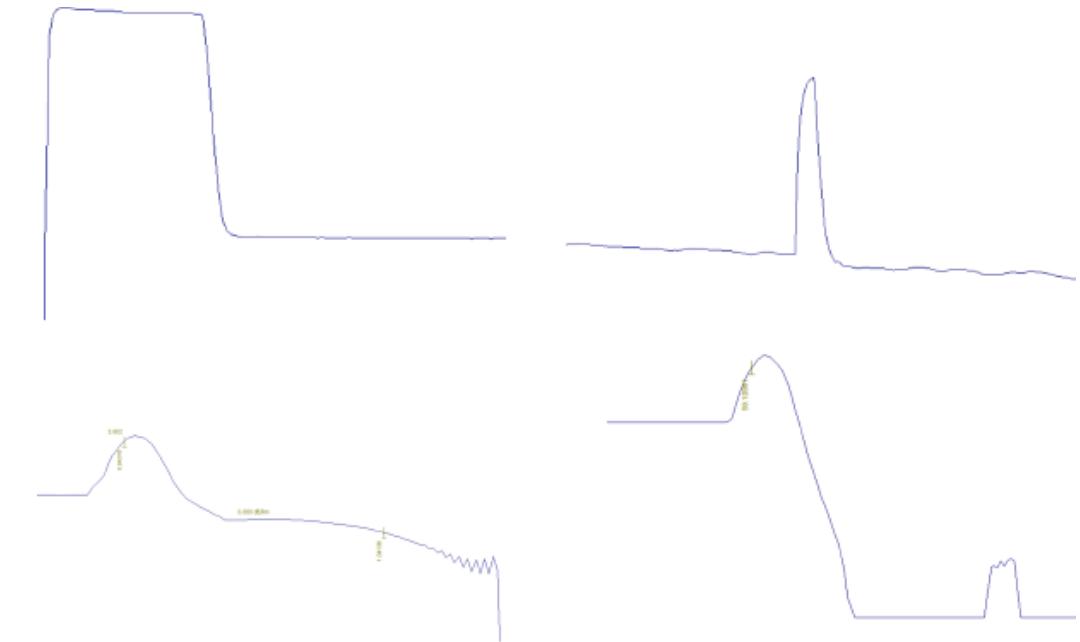
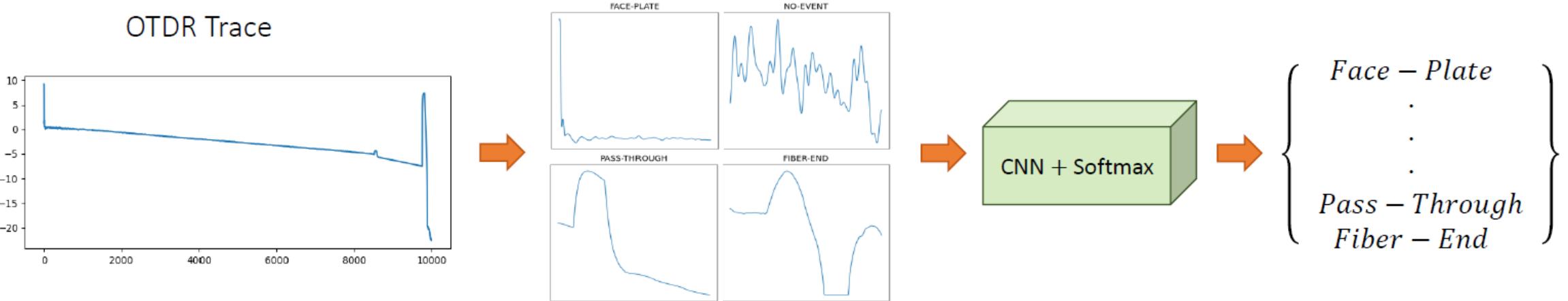


Fig. 1 (b): OTDR events examples: face-plate, pass-through, fiber-cut, fiber-end

Event Detection in Optical Signals



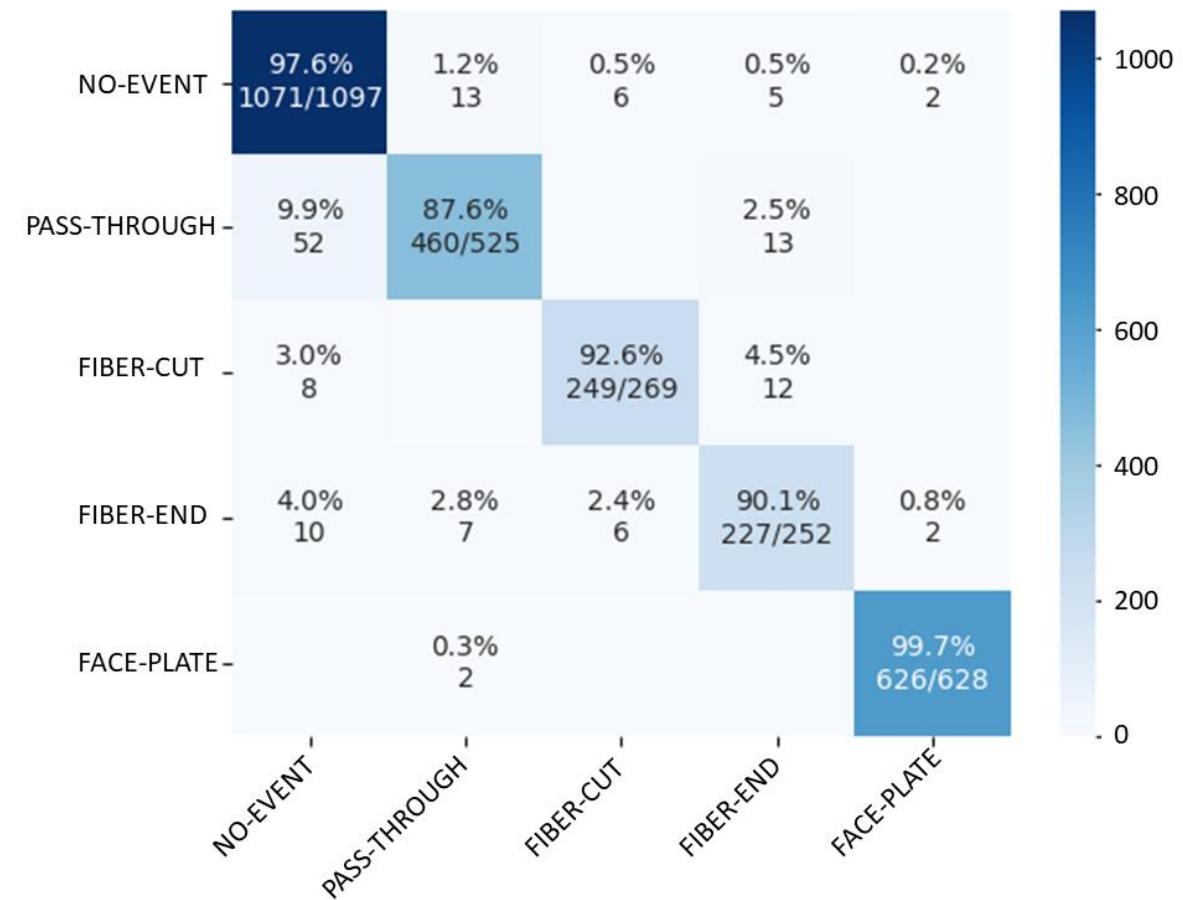
Train a classification network over fixed windows (300 samples)



Event Detection in Optical Signals



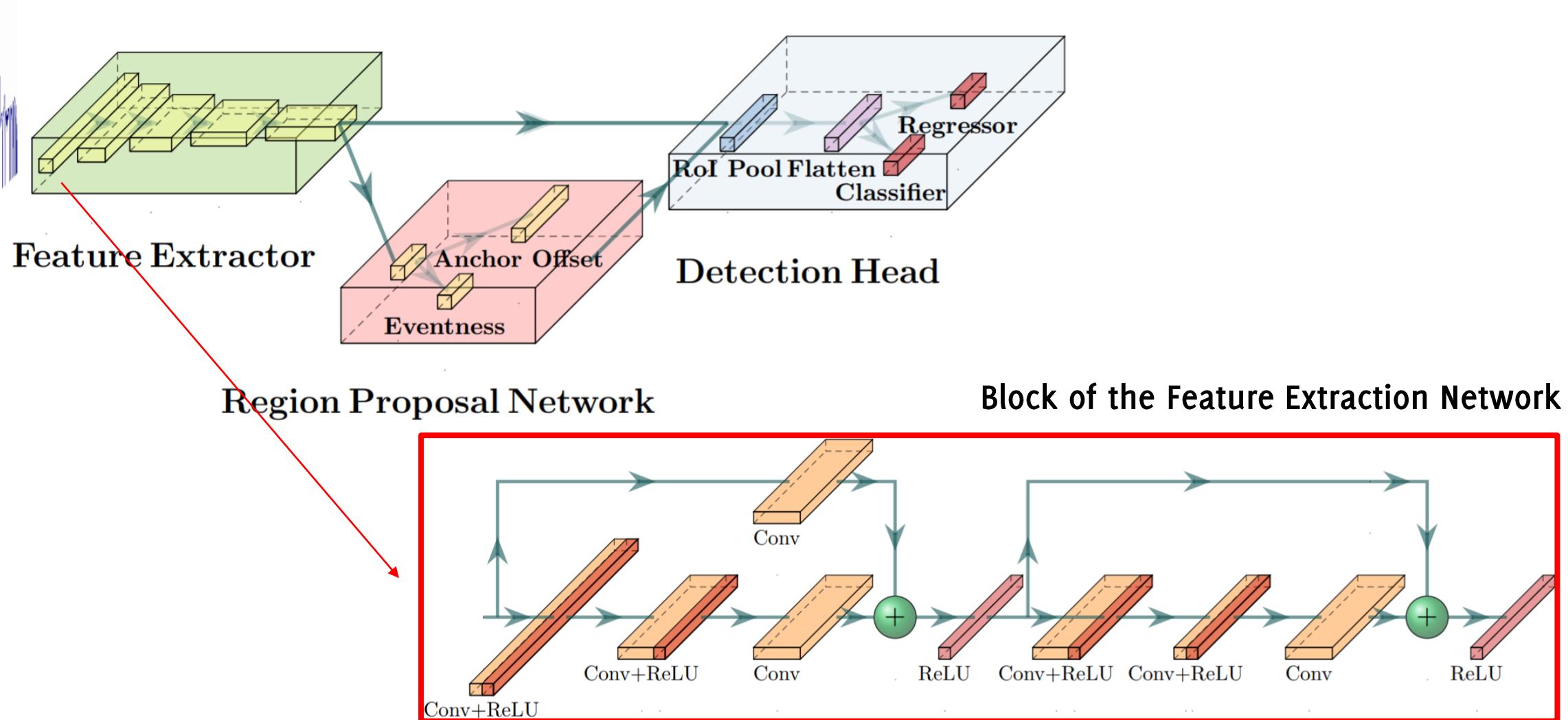
- Requires an additional step to split OTDR traces into windows
- Limited to single-scale events resolution
- Works under the assumption that each window includes at most one event
- Does not consider events position
- Does not share computations



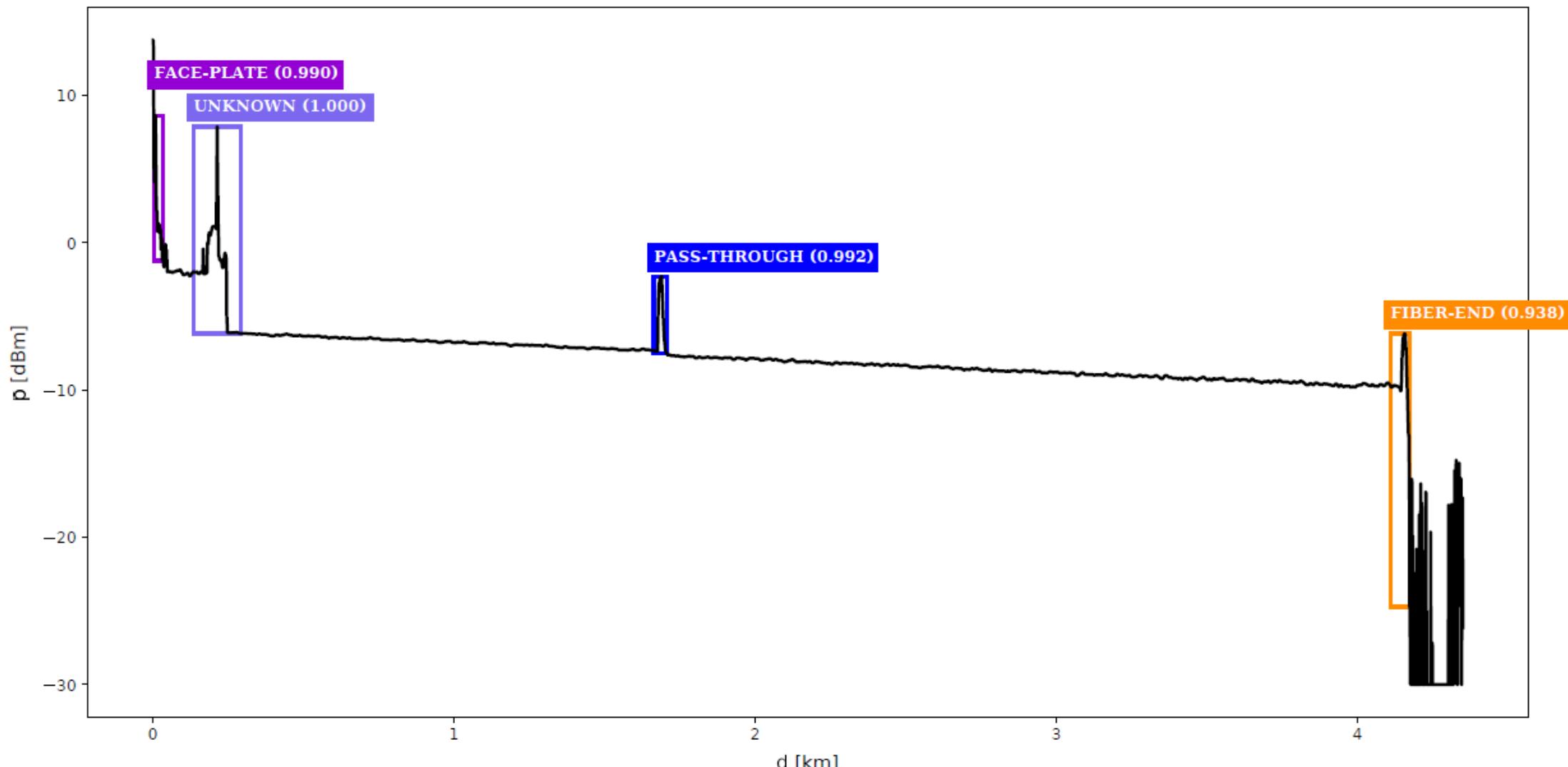
Event Detection in Optical Signals



Faster-Like Object Detection Network



Able to detect both known and unknown events





This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

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/>

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

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

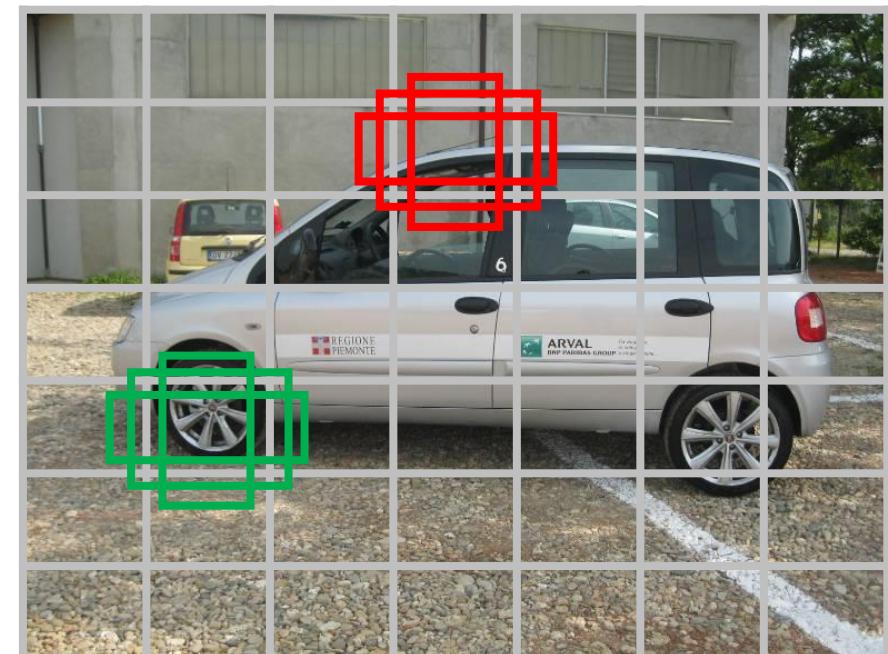
YOLO

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



YOLO

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



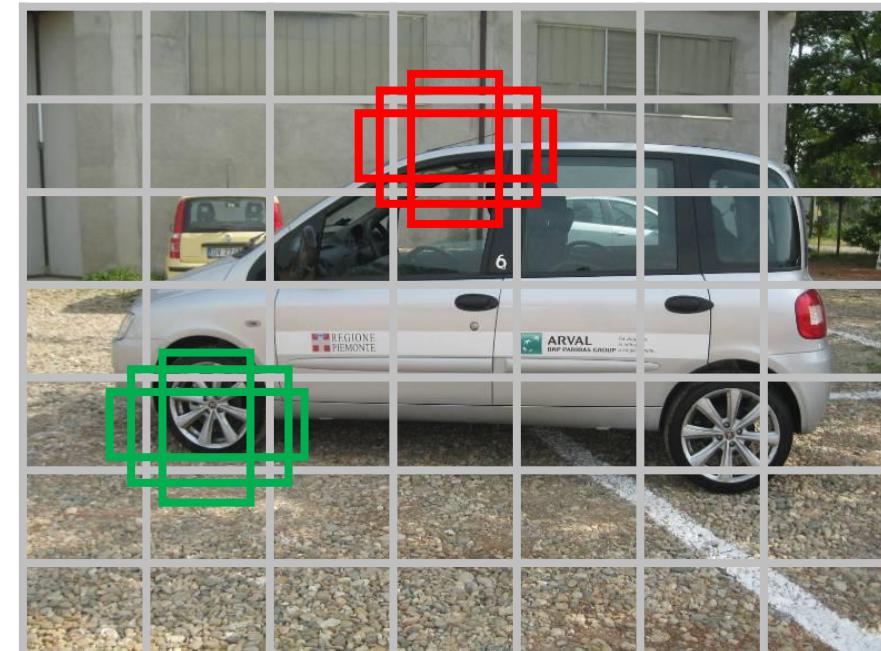
YOLO

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

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



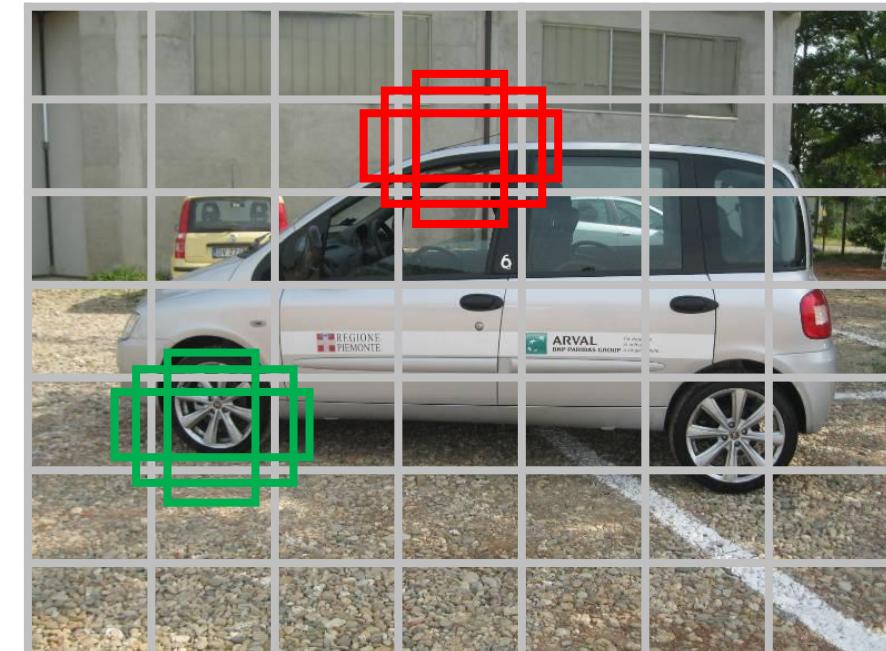
YOLO

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

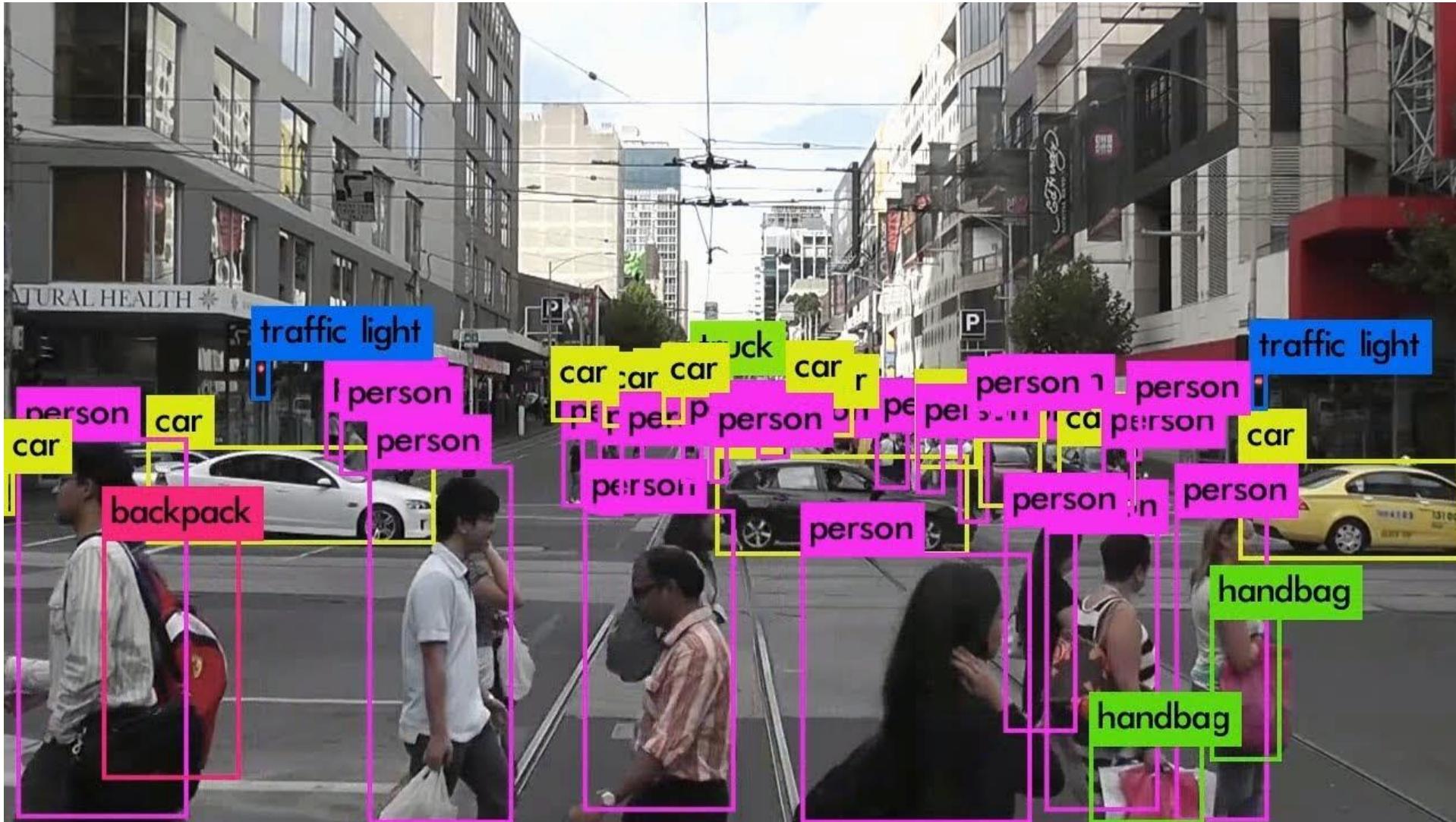
Training this network is sort of tricky to assess the loss (matched / not matched)

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

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



Object Detection

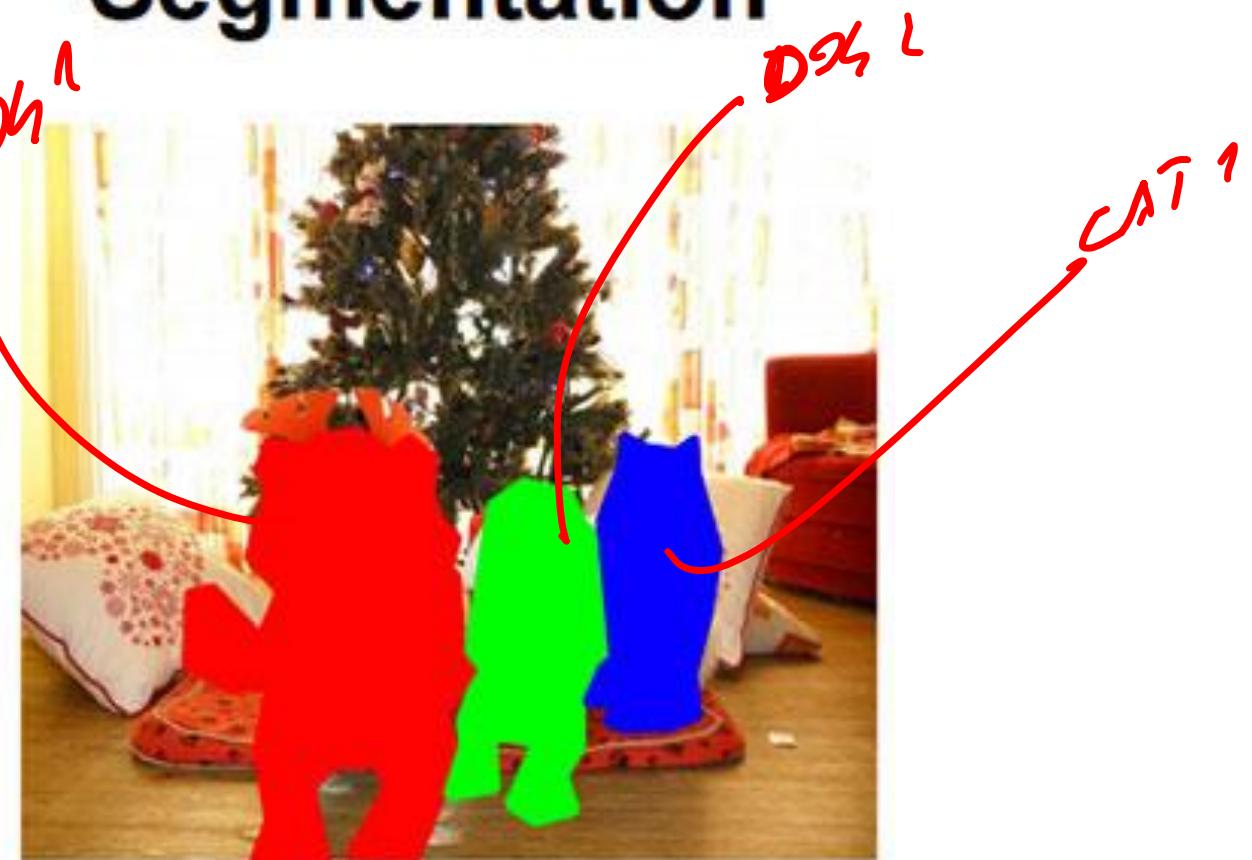


Object Detection vs Instance Segmentation

**Object
Detection**



**Instance
Segmentation**





This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick
Facebook AI Research (FAIR)

Instance Segmentation

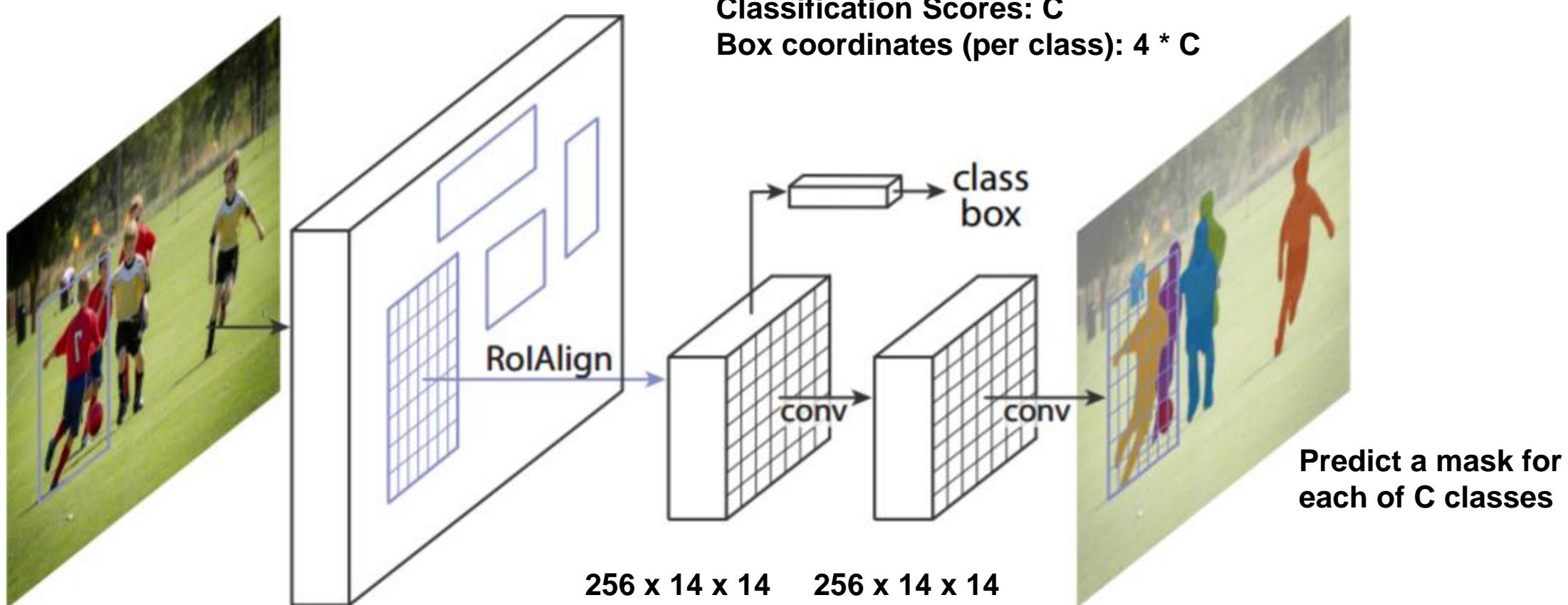
It combines the challenges of:

- **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 Faster R-CNN, each ROI is classified and
the bounding boxes are estimated

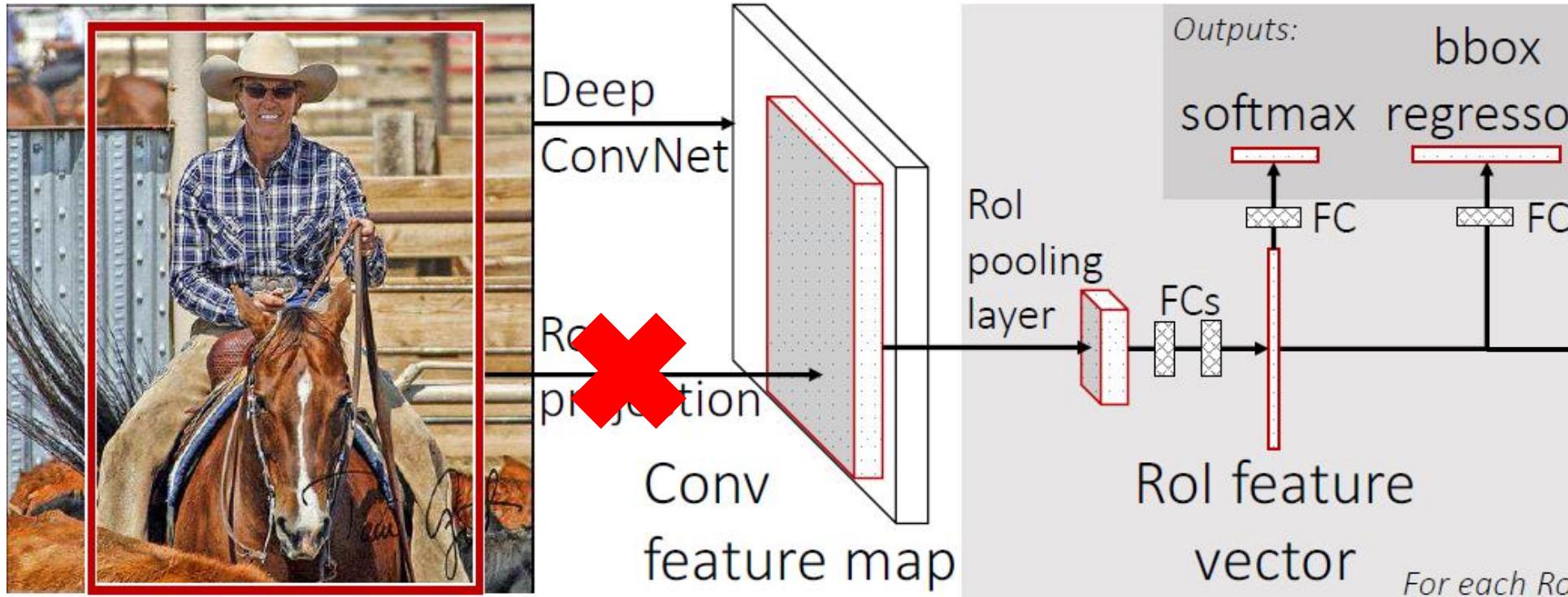
Semantic segmentation
inside each ROI



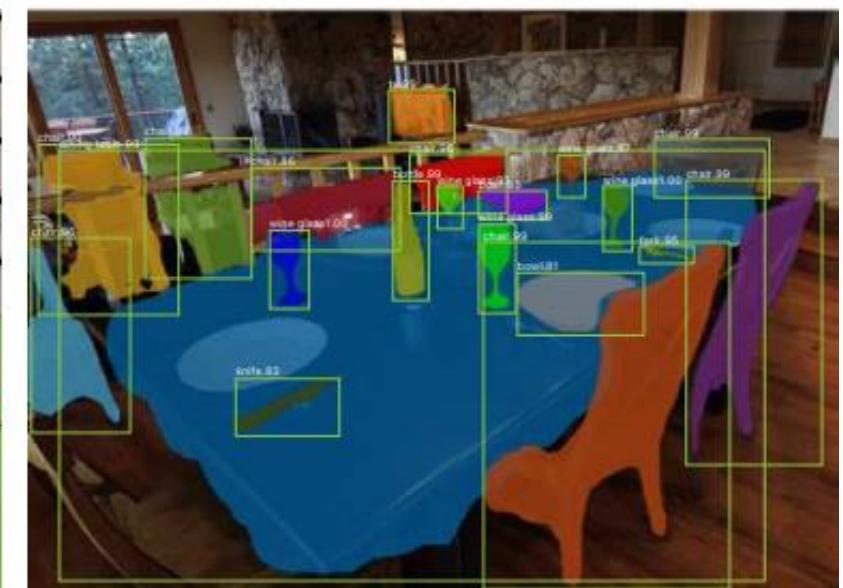
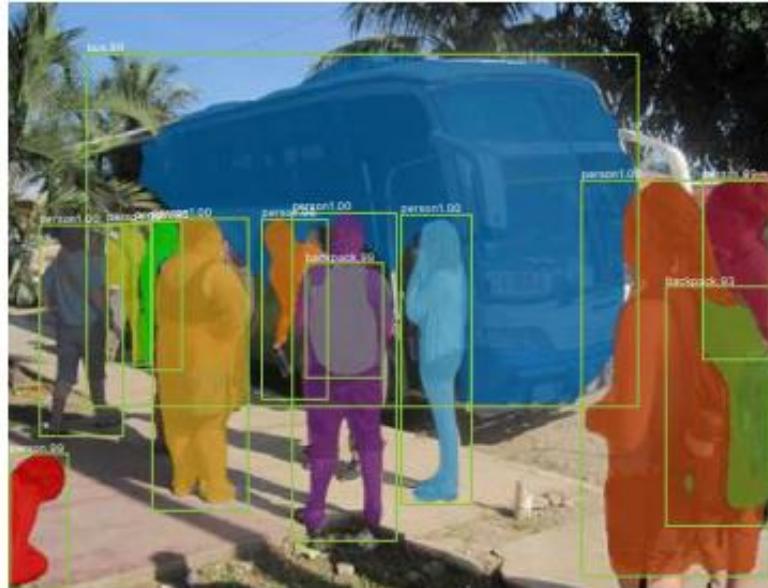
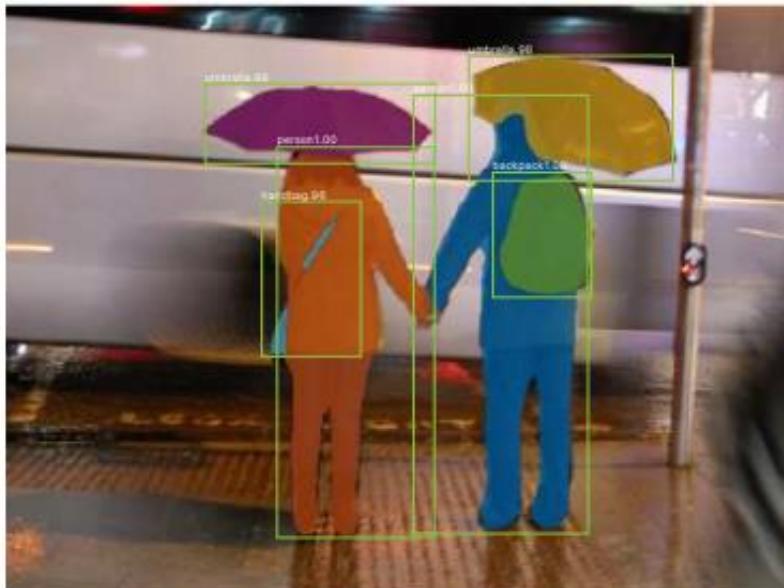
Mask R-CNN

Mask is estimated for each ROI and each class

Extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition



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



Mask R-CNN (including Pose estimation)



Mask R-CNN (including Pose estimation)



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

Instance Segmentation

Research Project with Ikonisys

Credits Roberto Basla

In collaboration with

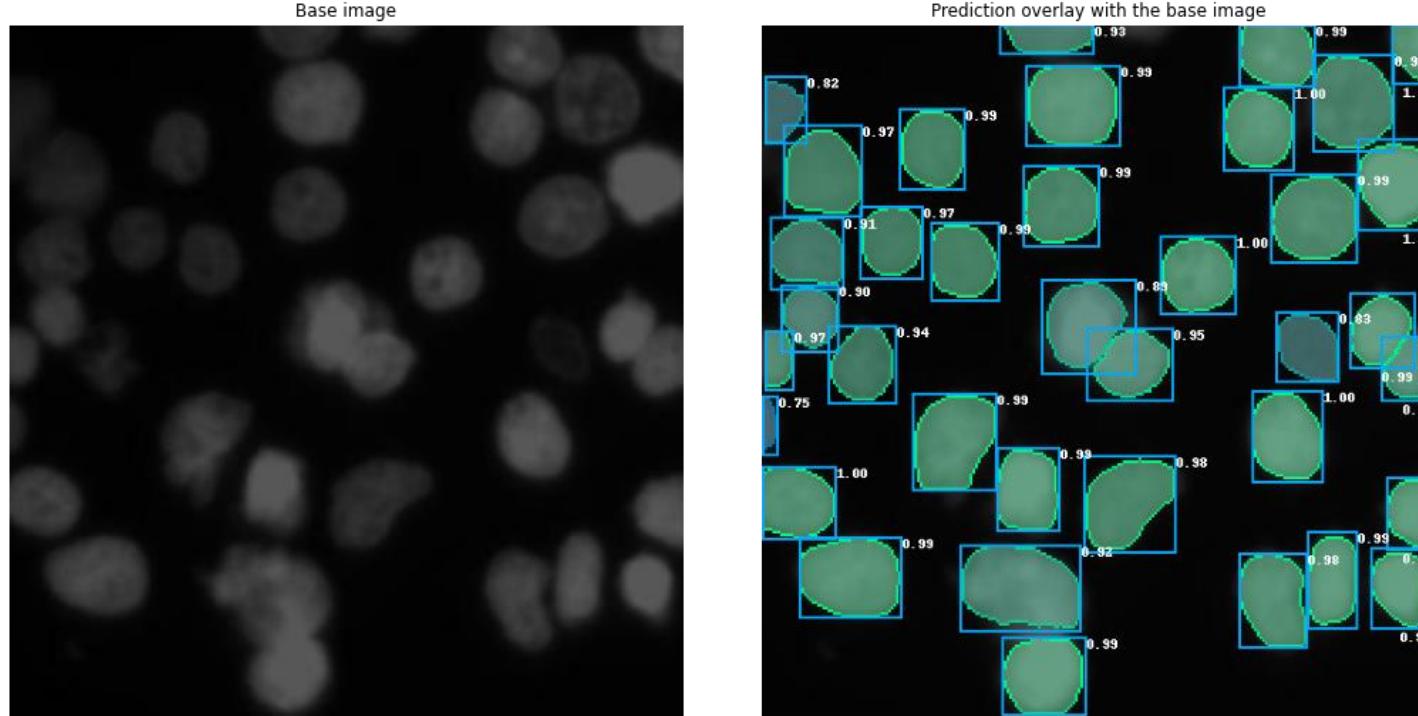


FINDING THE CELLS THAT MATTER

G. Boracchi

Instance Segmentation Network

The instance segmentation network is able to identify nuclei in fluoroscopy images.



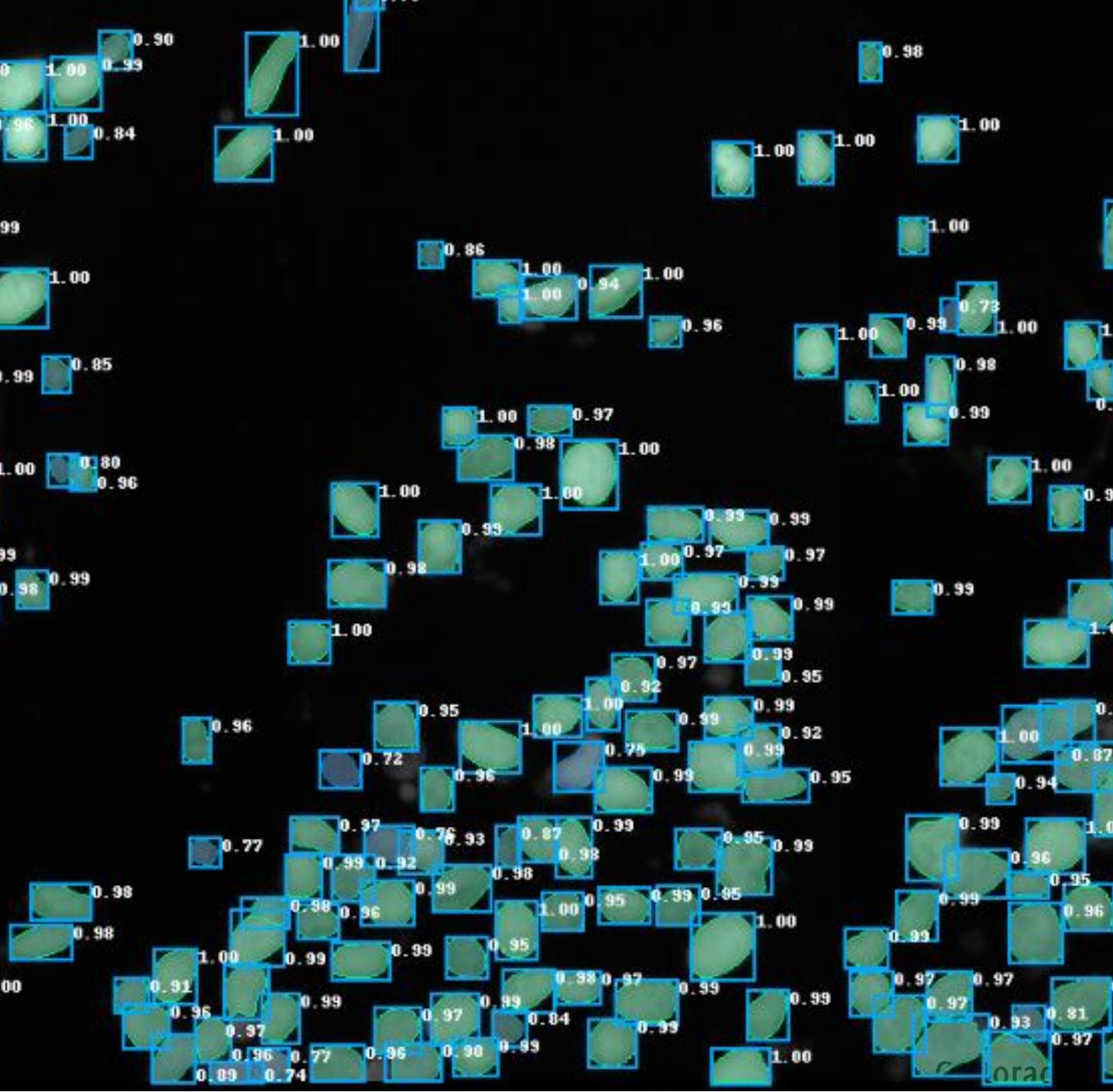
The network provides, for each identified nucleus, its ^{Image}segmentation mask, ^{Prediction}bounding box, and prediction confidence.

In collaboration with
ikonisys
FINDING THE CELLS THAT MATTER i

Model Performance

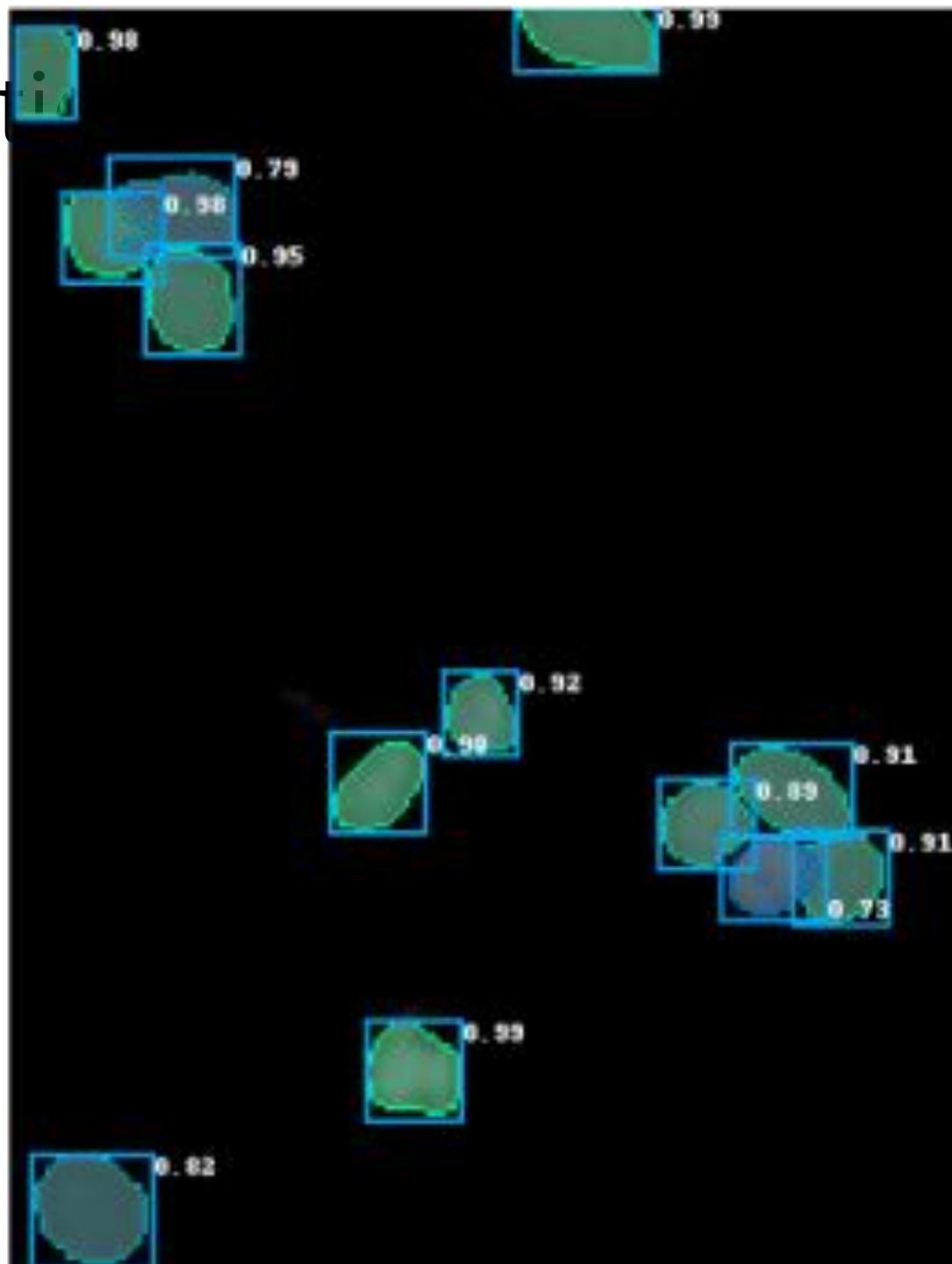
The model achieves great results even when trained with **very few images from the target domain**, by making use of transfer learning from these datasets:

- DSB2018
- DSB2019
- BBBCo39
- NSDE



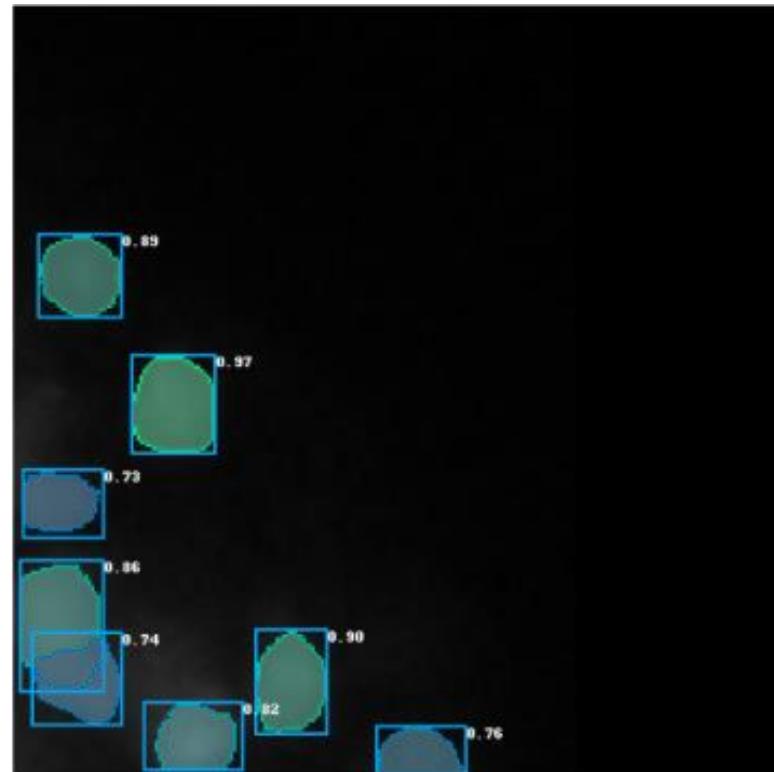
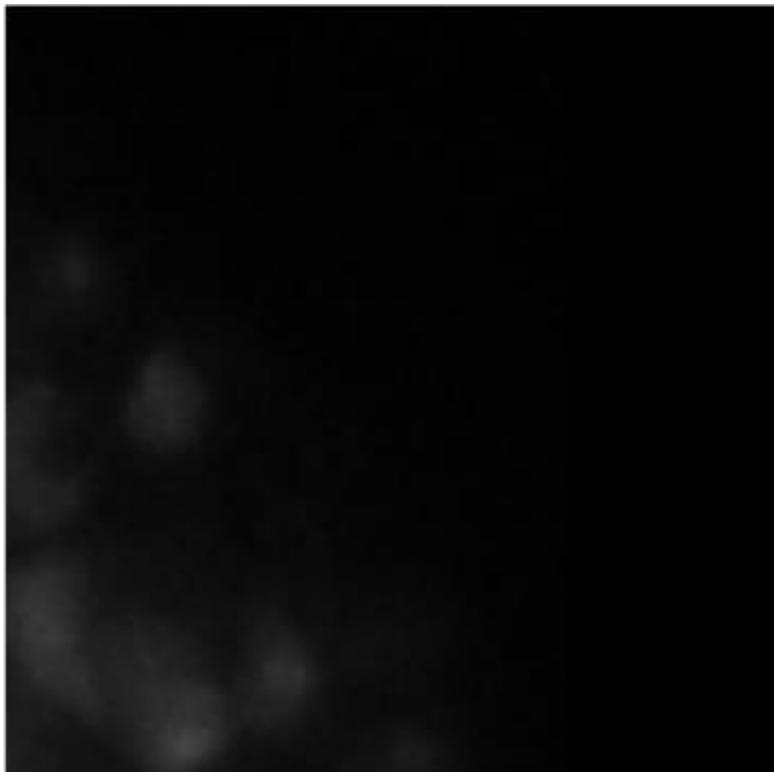
Overlapping segmentation

Overlapping cells can be identified and masked independently.



Robustness

The model can identify even the faintest of nuclei in the image, further proving the performance of the system.



In collaboration with
ikonisys
FINDING THE CELLS THAT MATTER

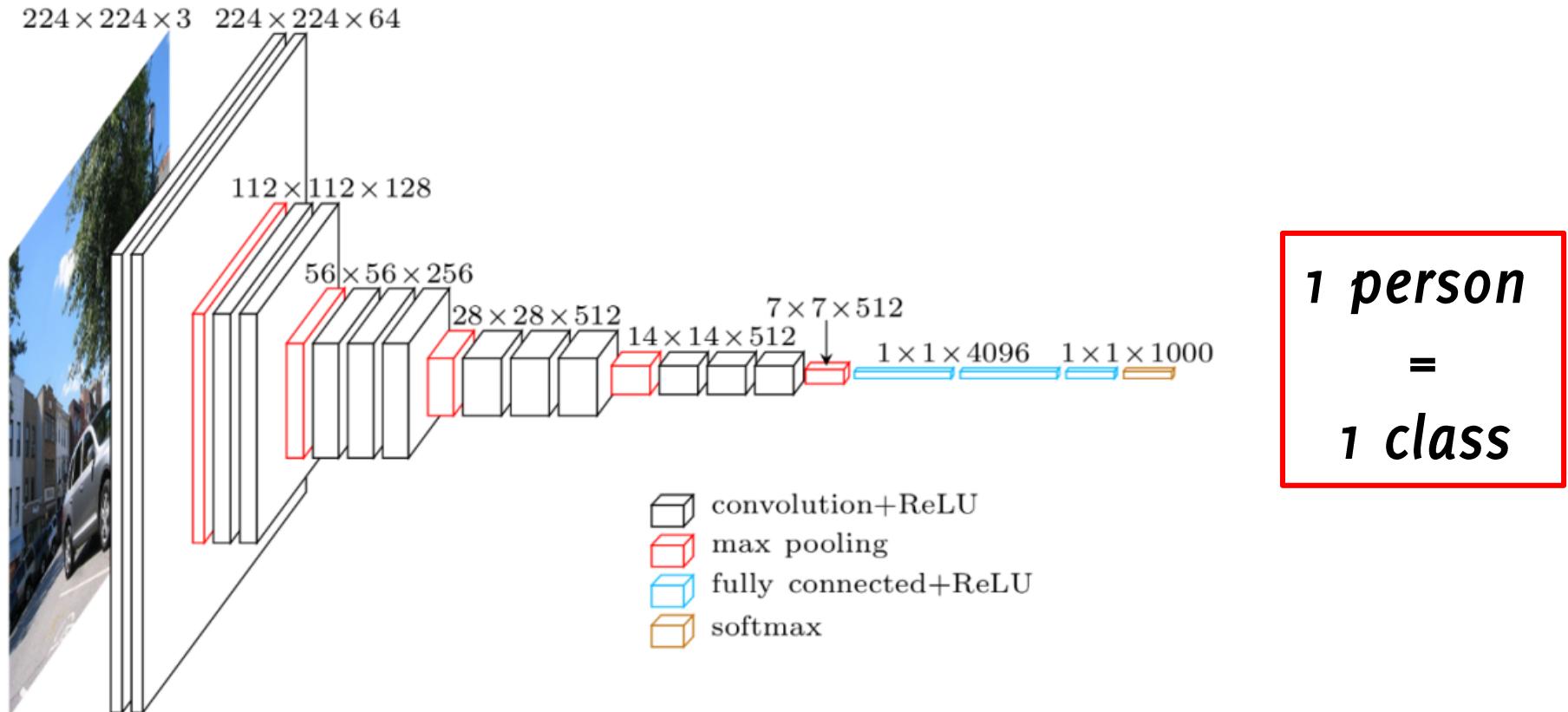
Metric Learning

Say you are asked to implement a face
identification system
to open Politecnico di Milano door!

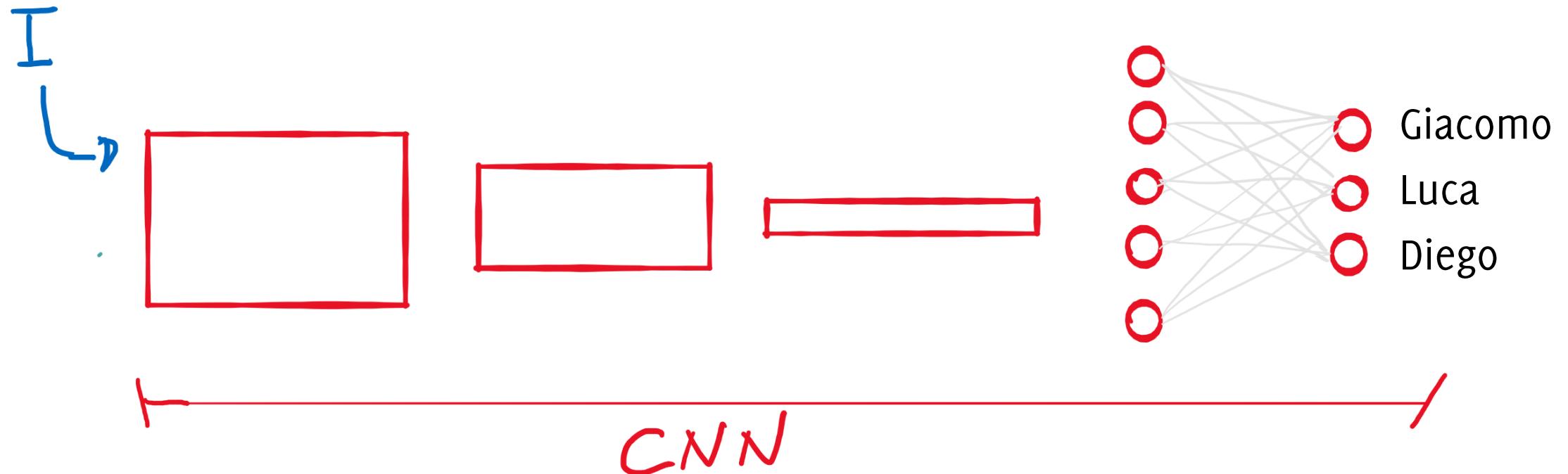


Classification? Detection? Segmentation?

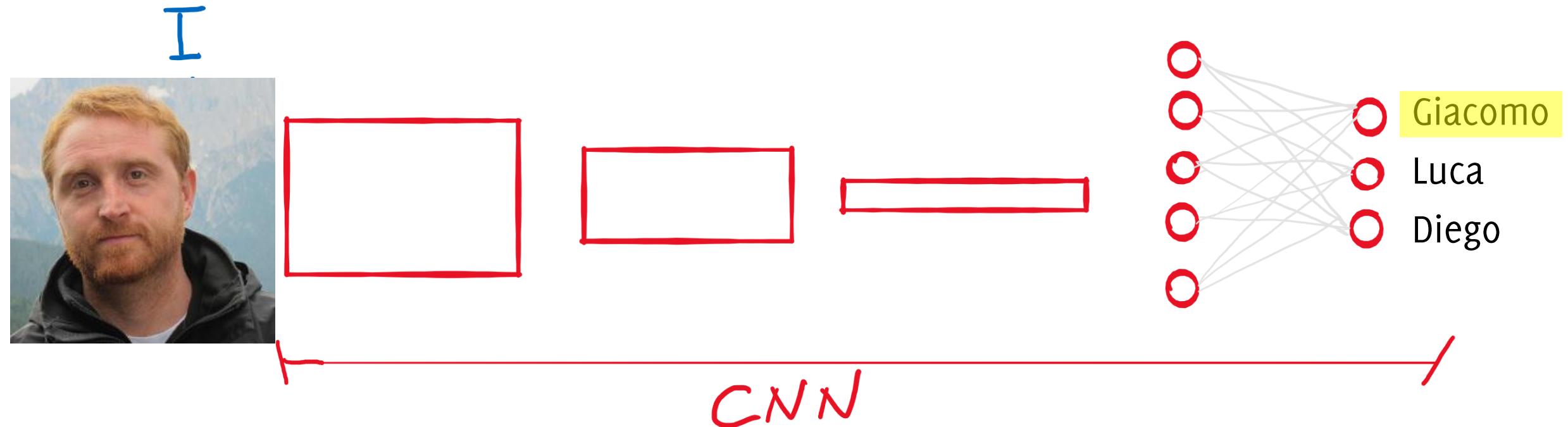
Let's start fresh from a Convolutional Neural Network for classification



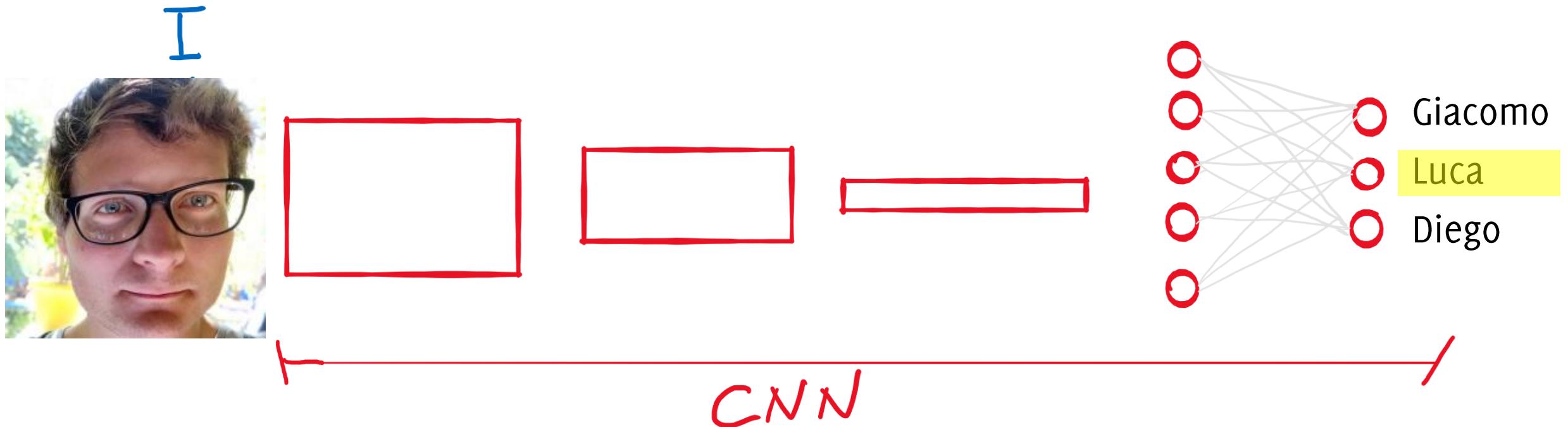
Face identification by Classification



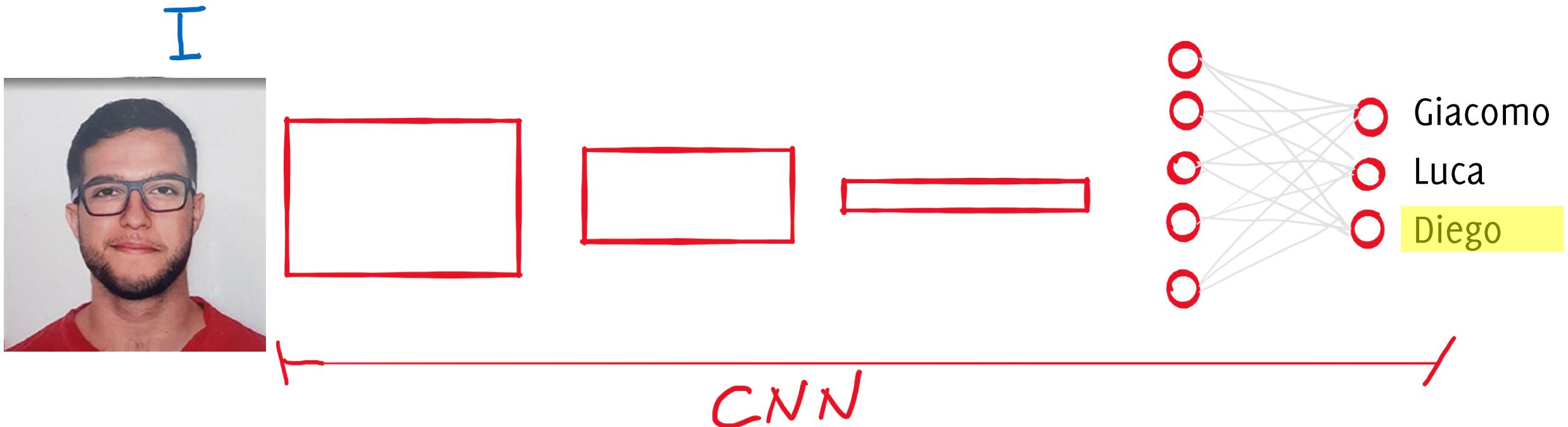
Face identification by Classification



Face identification by Classification



Face identification by Classification



What do we need?

A training set

- A few images per class / person
- Images in different conditions (position, light, expression, clothes ...)
- A few Py snippets and a GPU...

That's easy...

What do we need?

A training set

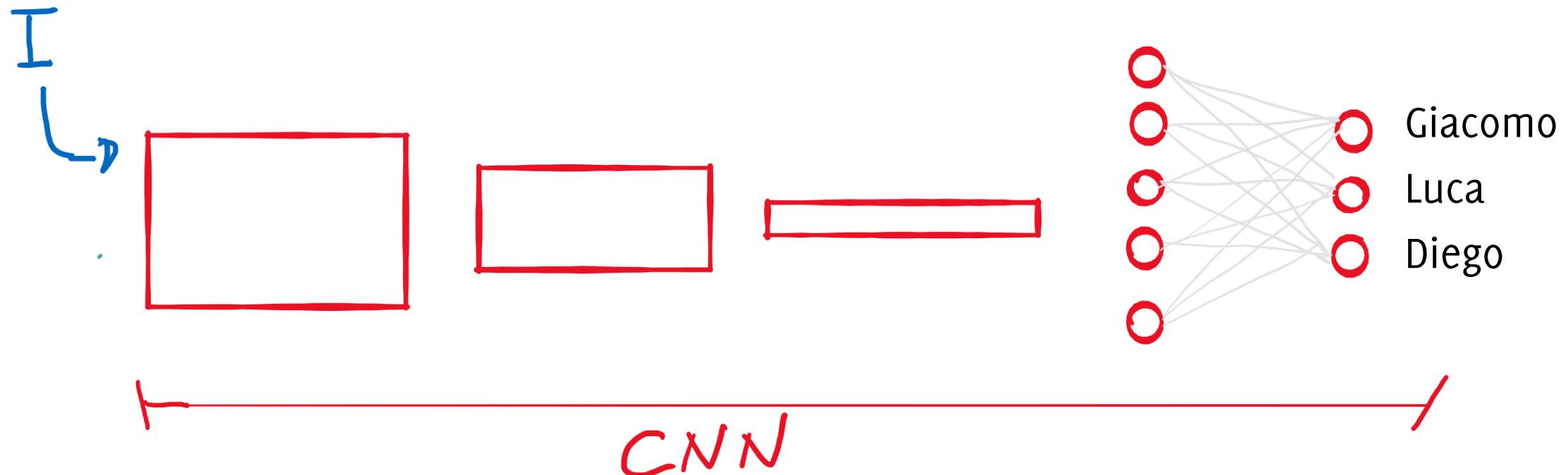
- A few images per class / person
- Images in different conditions (position, light, expression, clothes ...)
- A few Py snippets and a GPU...

That's easy...

Are we happy with this solution?

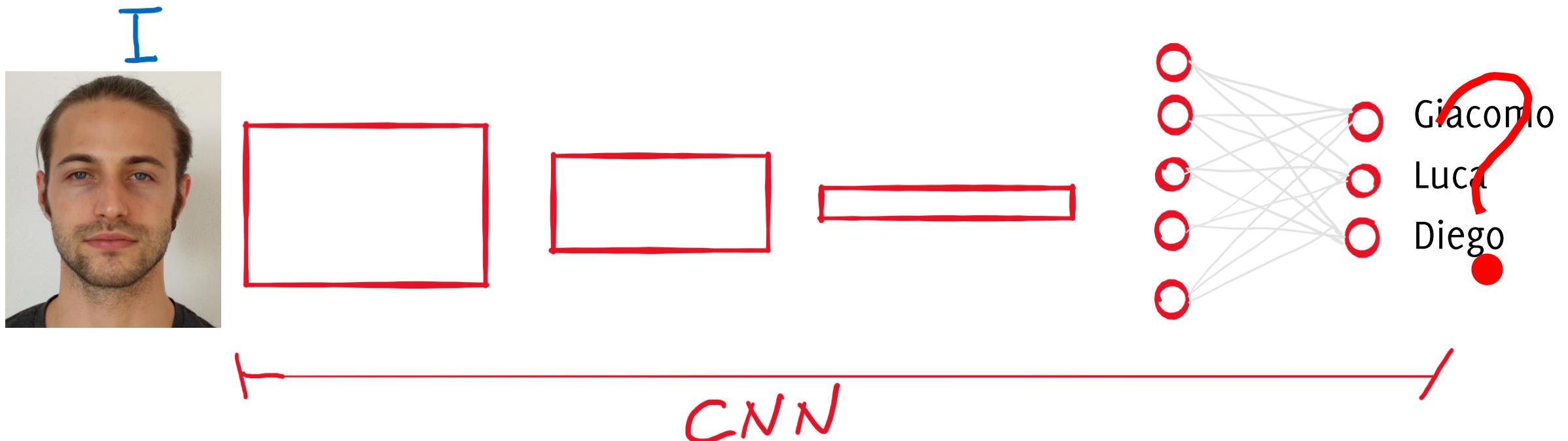
... Not quite

What happens when you need to enroll a new employee?



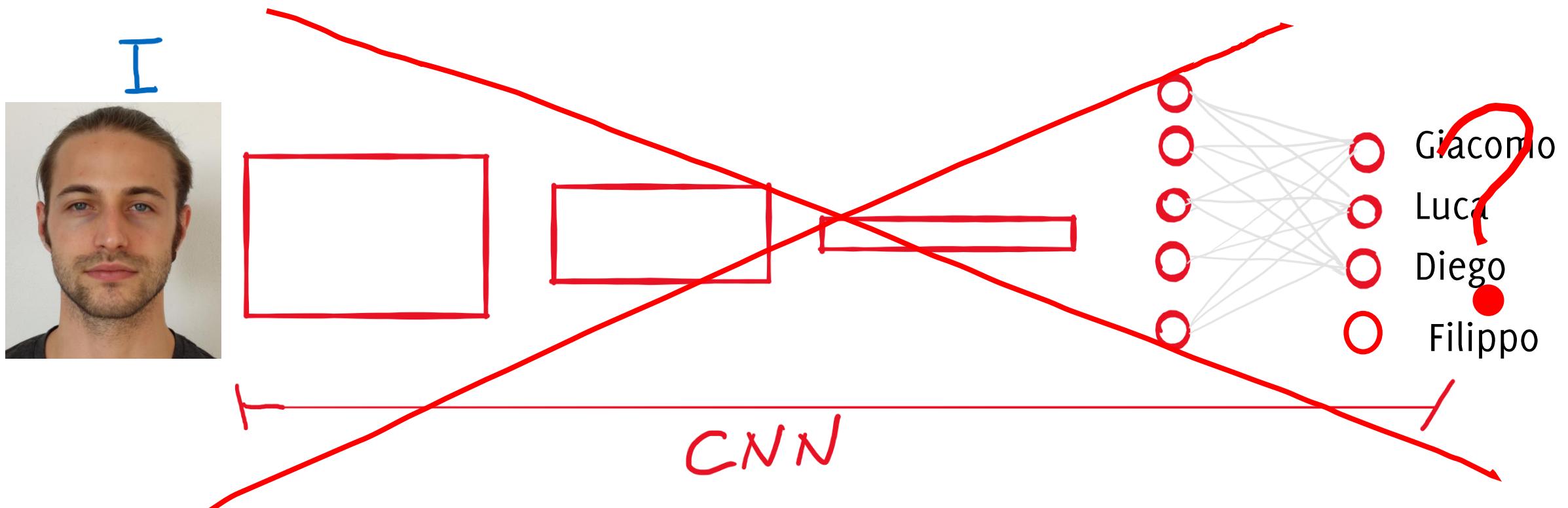
... Not quite

What happens when you need to enroll a new employee?



... Not quite

What happens when you need to enroll a new employee?



The whole network has to be retrained for each new person to be identified

A Different Approach is Needed!

What about image comparison?

Why don't we store a picture for each employee (**template**), and then perform identification by **pairing the input to its closest template**?

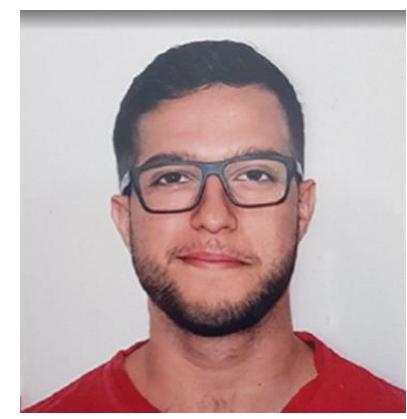
T_1



T_2



T_3



What about image comparison?

Why don't we store a picture for each employee (**template**), and then perform identification by **pairing the input to its closest template**?

Enrolling a new employee
would be straightforward



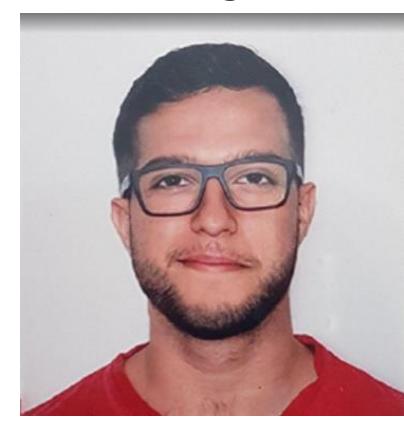
T_1



T_2



T_3



What about image comparison?

Why don't we store a picture for each employee (**template**), and then perform identification by **pairing the input to its closest template**?

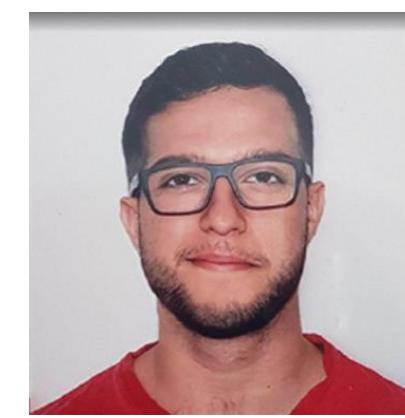
T_1



T_2



T_3



T_4



What about image comparison?

... but how to perform identification?

$$\hat{i} = \operatorname{argmin}_{j=1,\dots,4} \|I - T_j\|_2$$

I



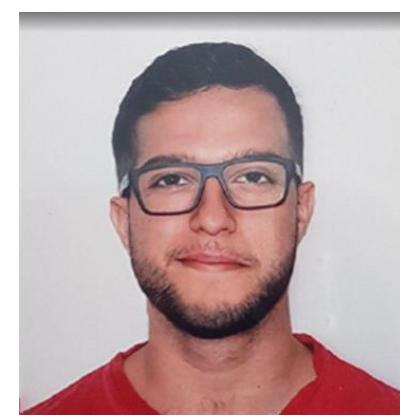
T_1



T_2



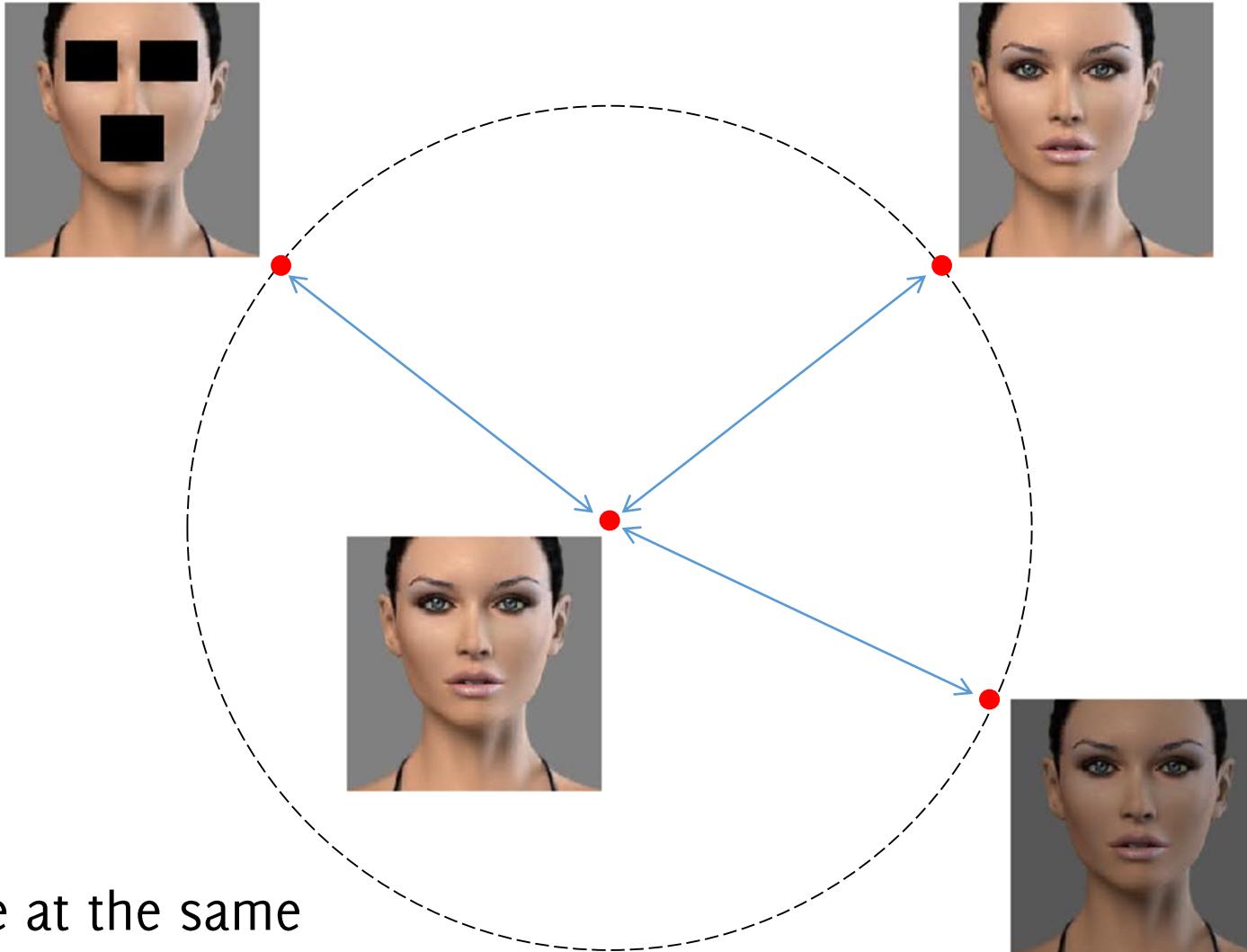
T_3



T_4



Bad News...



The three images are at the same distance from the reference at the center

What about image comparison?

... but how to perform identification?

$$\hat{i} = \operatorname{argmin}_{j=1,\dots,4} \|I - T_j\|_2$$

I



T_1



T_2



T_3



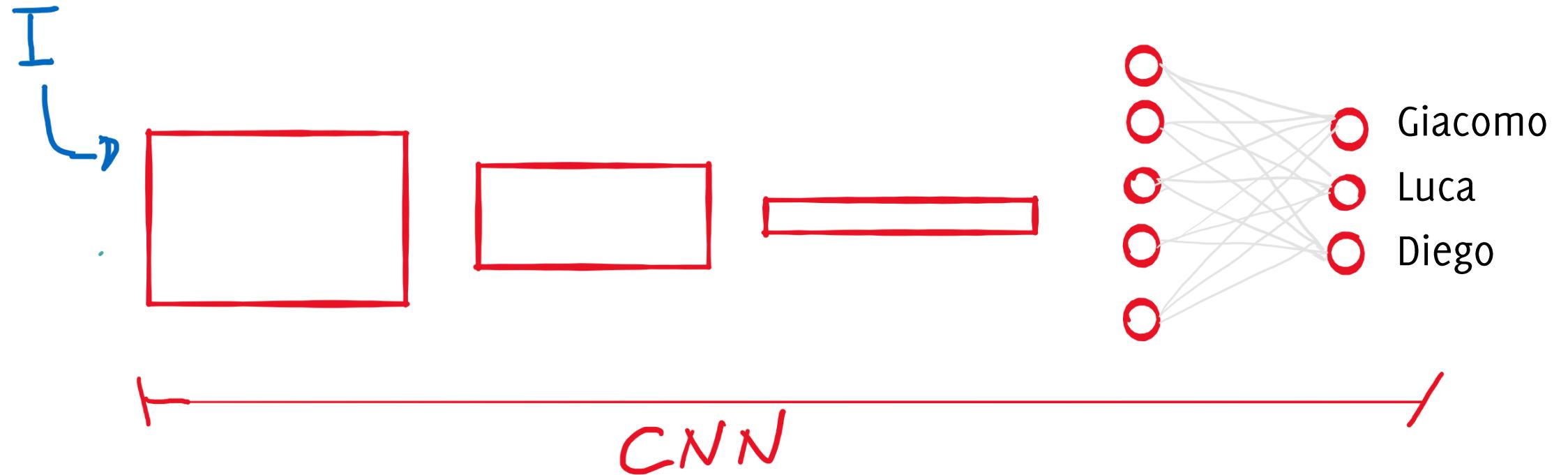
T_4



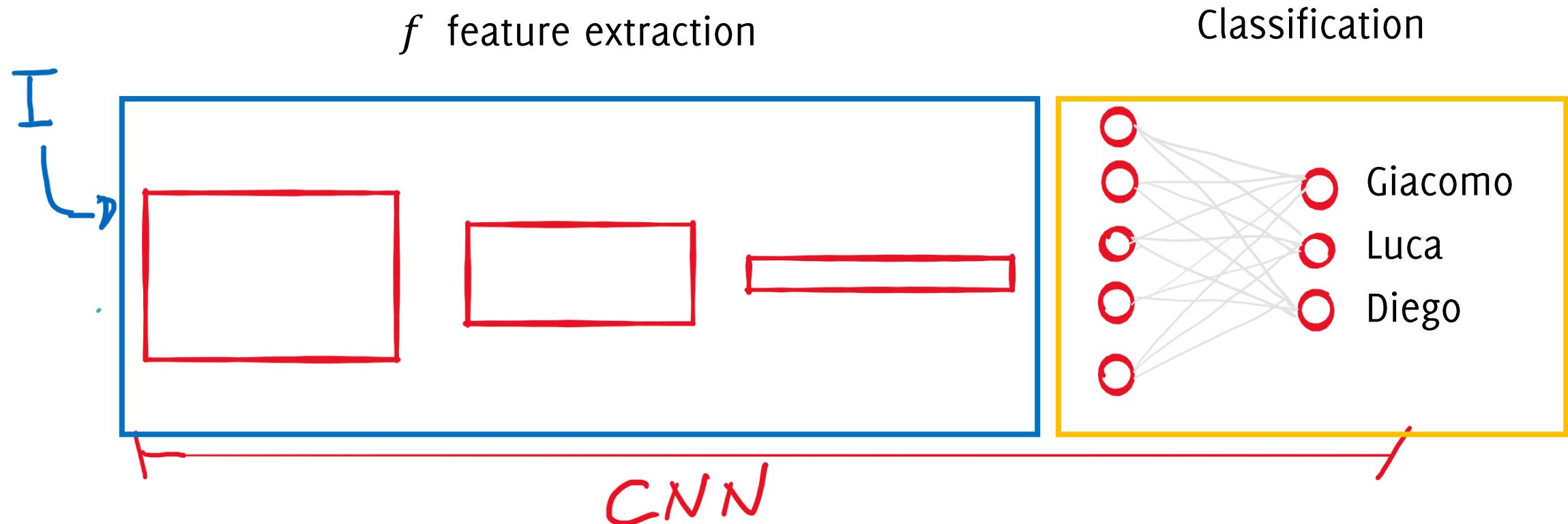
How to train I relying only on these examples?

...we need a better distance measure for
face identification!

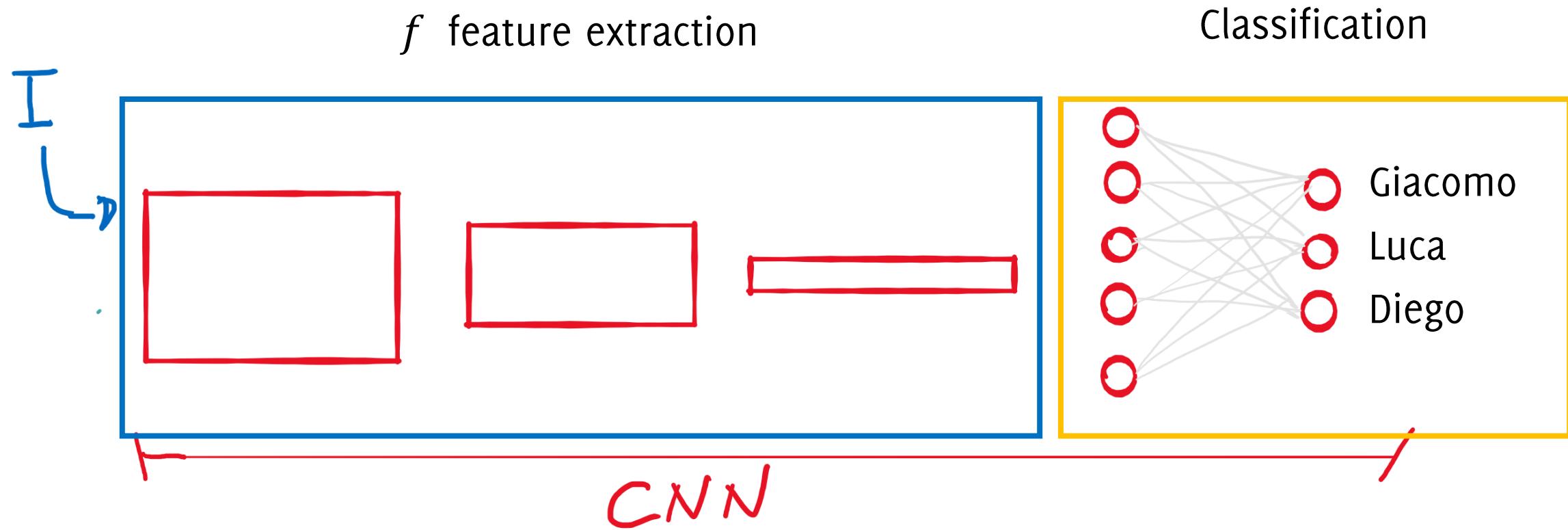
Dissecting CNN



Dissecting CNN

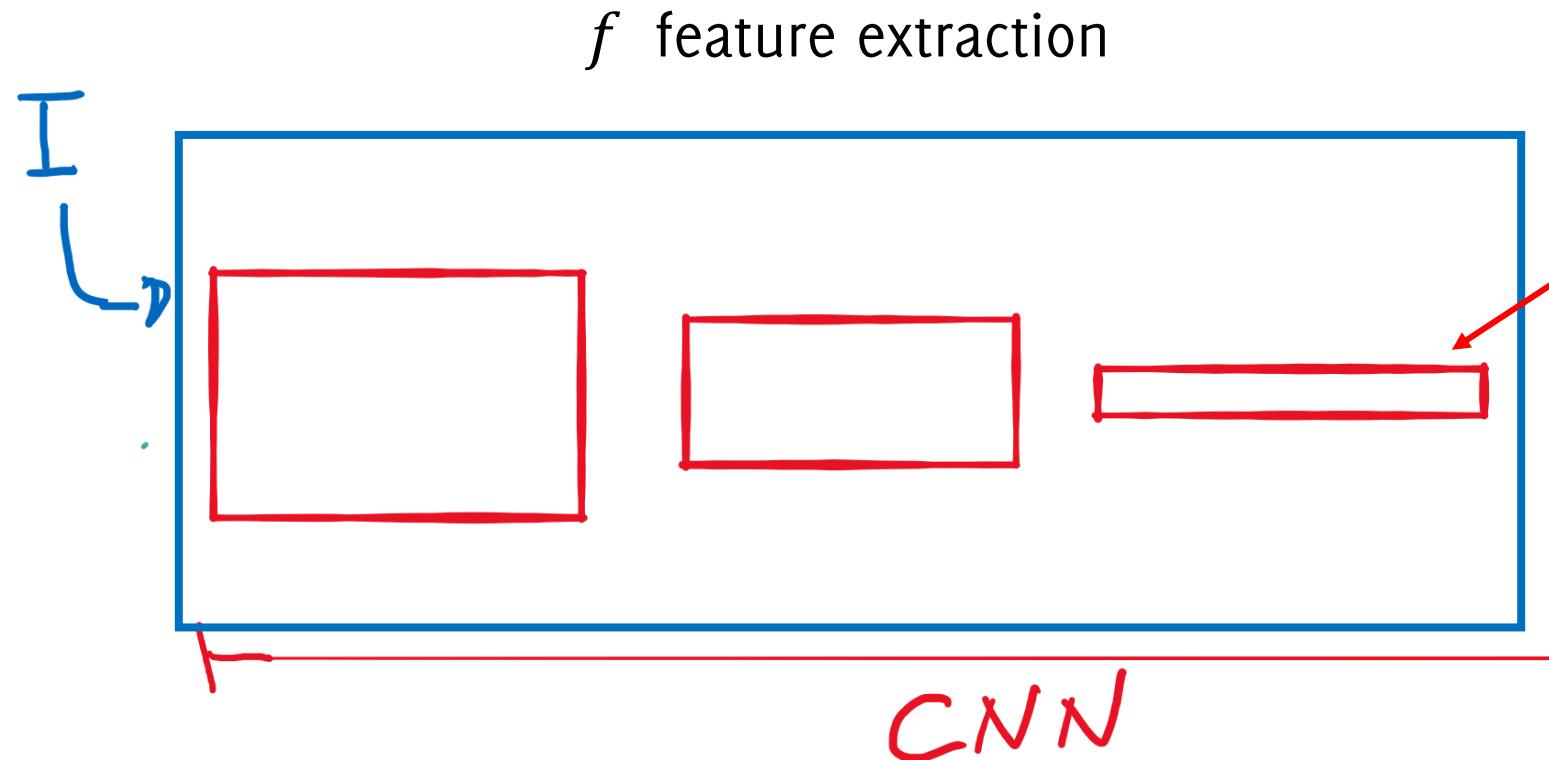


Dissecting CNN



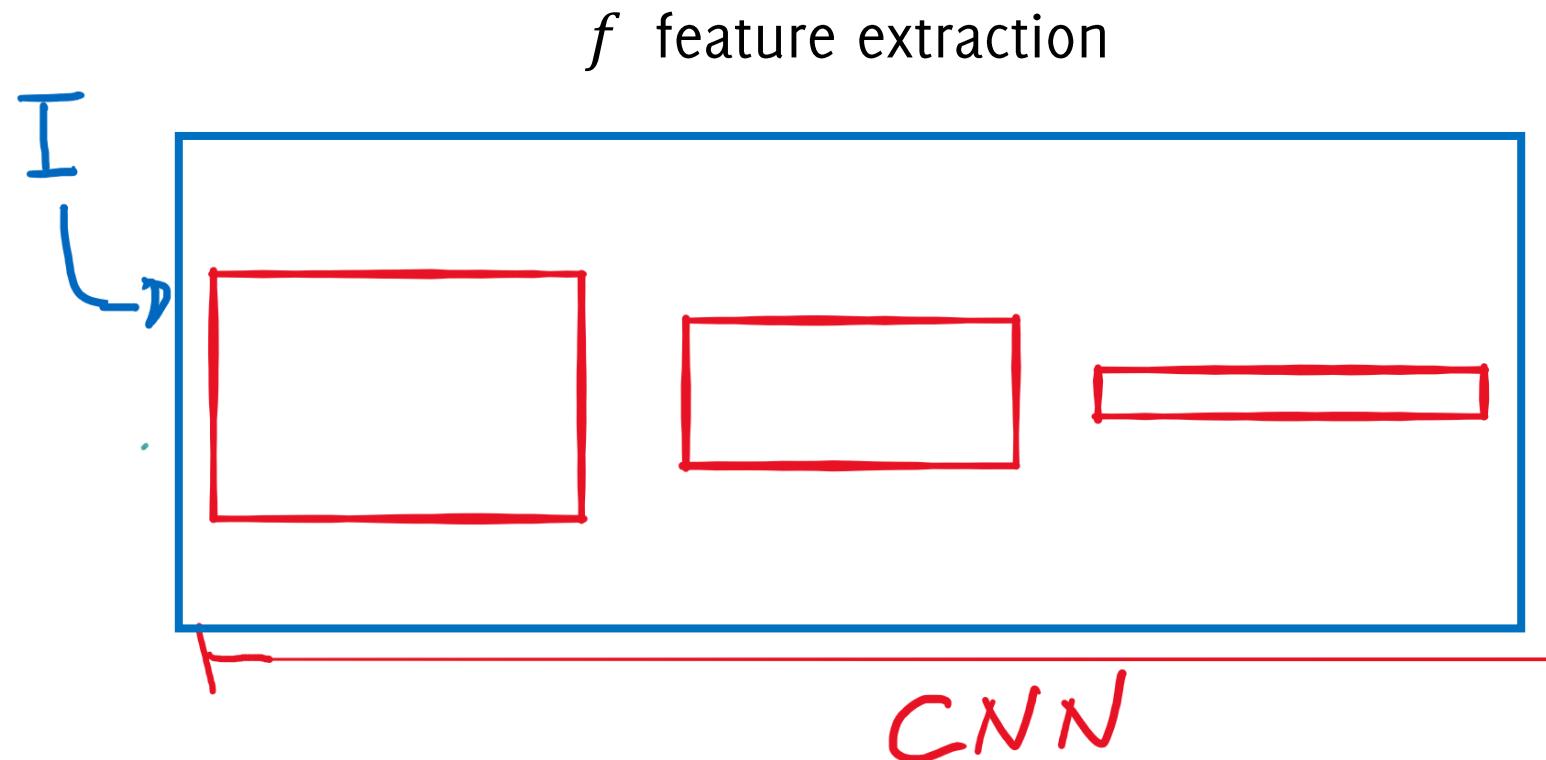
The feature extraction part is typically more general-purpose than the classification part (see Transfer Learning / Domain Adaptation)

Latent representation



$f(I)$: latent representation of I
Provides a meaningful description of the image in terms of patterns that are useful for solving the classification problem

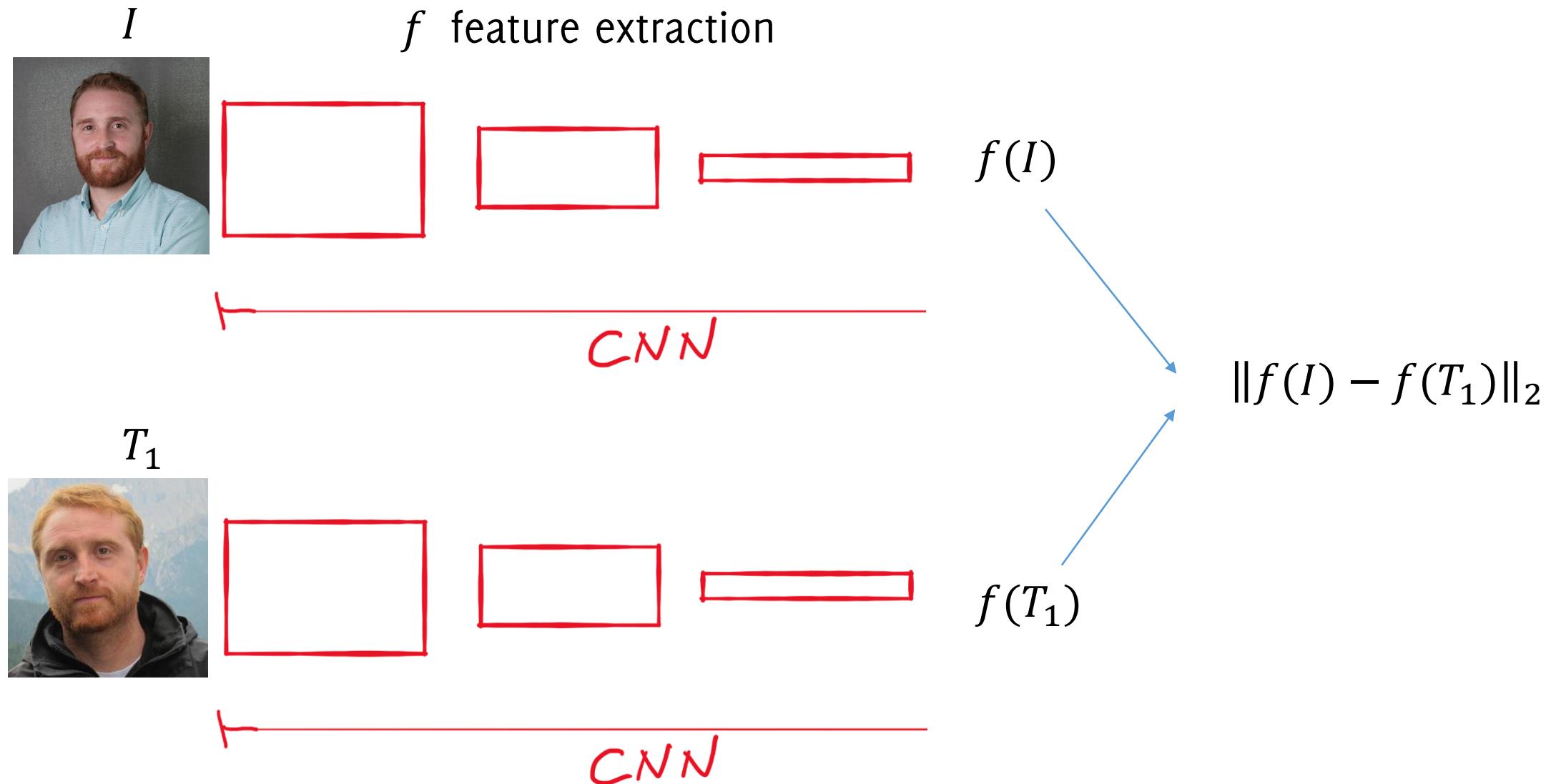
Distance among latent representations



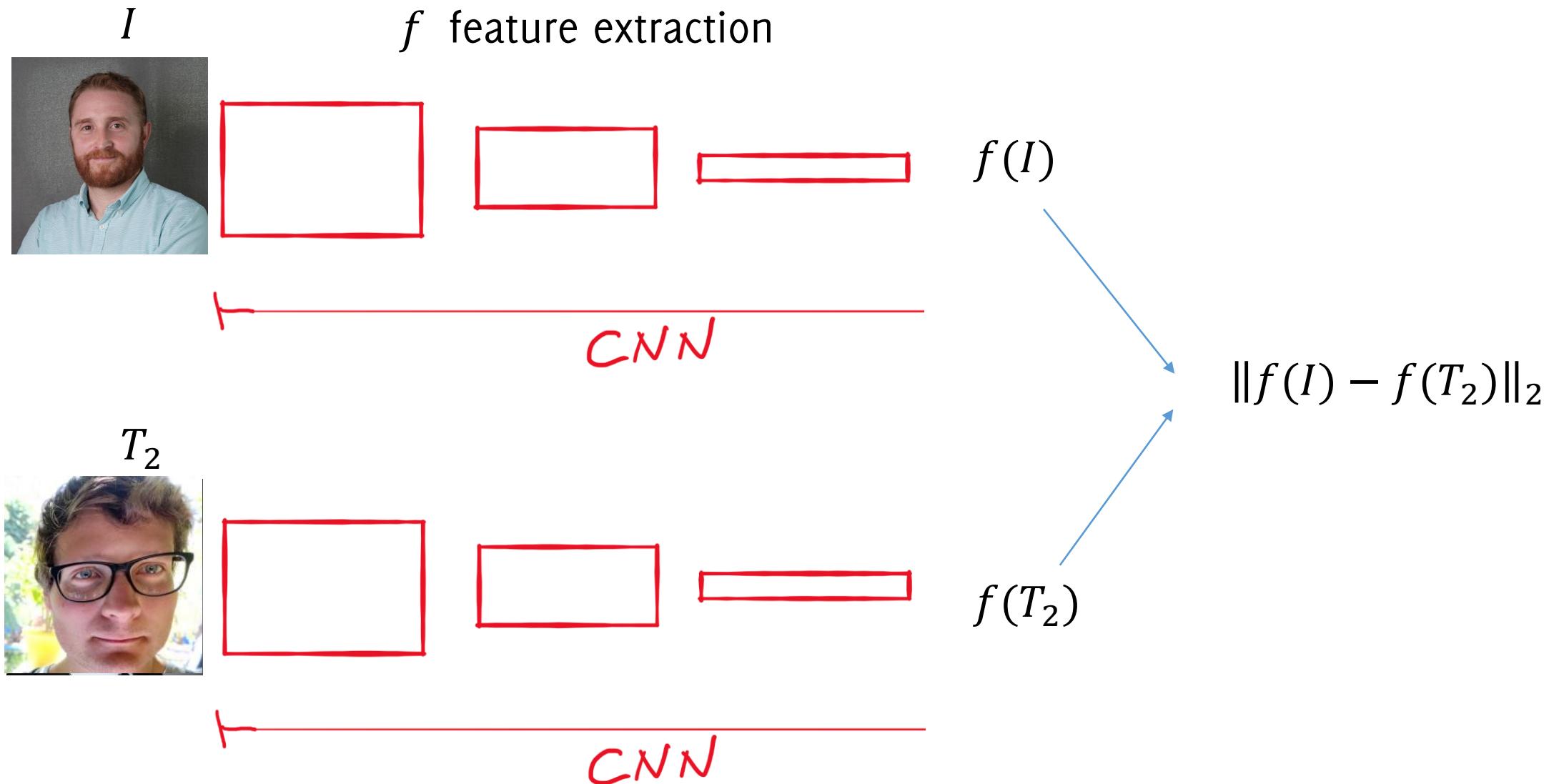
What about using?

$$\hat{i} = \operatorname{argmin}_{j=1,\dots,4} \|f(I) - f(T_j)\|_2$$

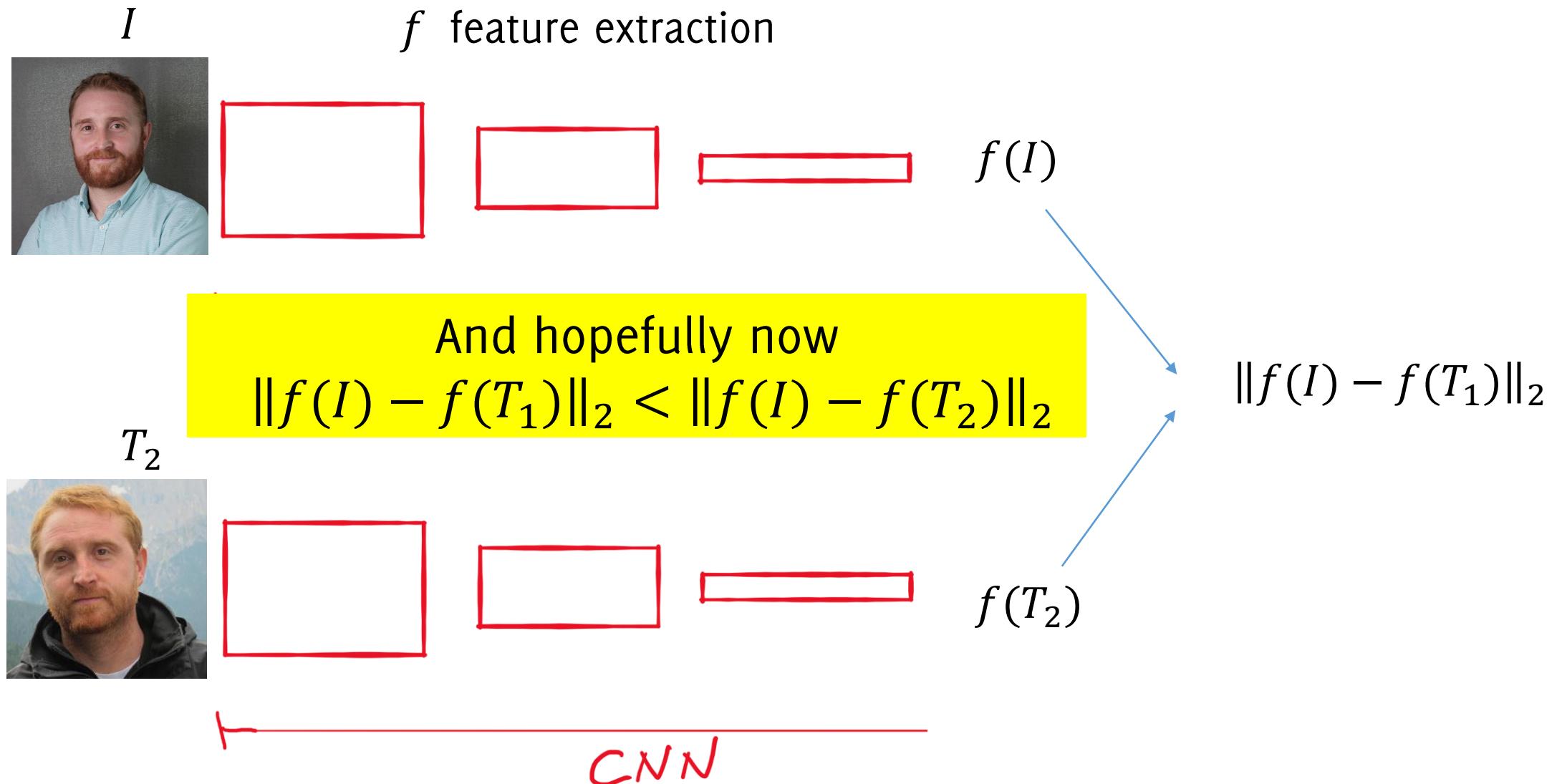
Distance among latent representations



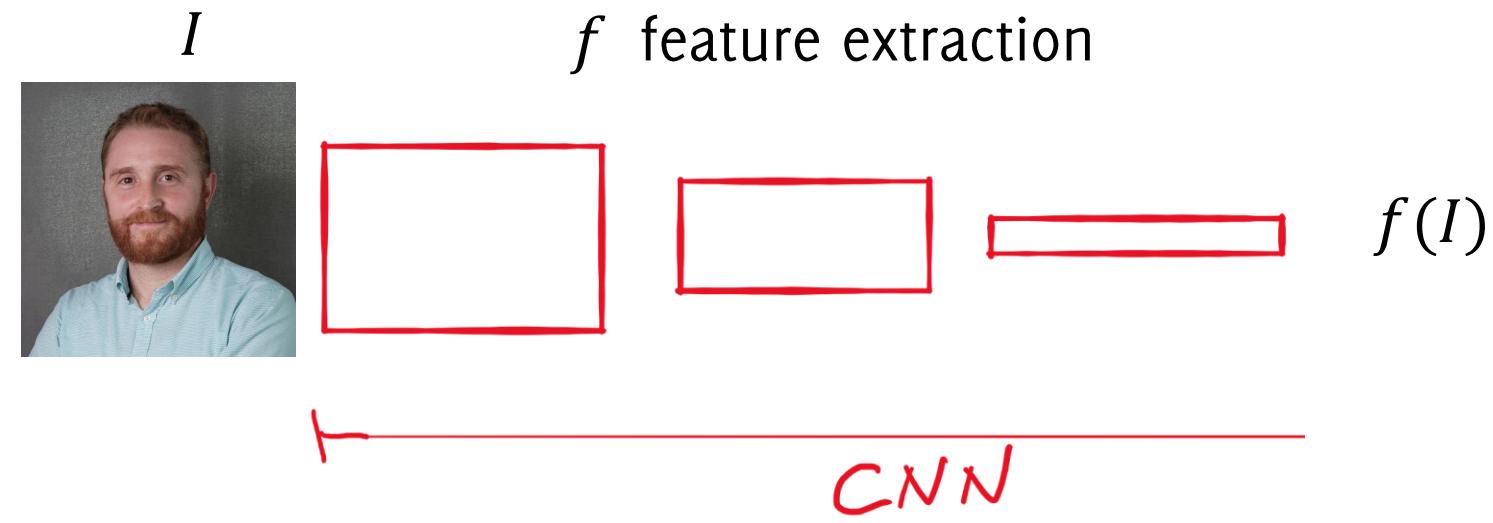
Distance among latent representations



Distance among latent representations



In practice



1. Extract features from the first image $f(I)$
2. Perform identification as
 $\hat{i} = \operatorname{argmin}_{j=1,\dots,4} \|f(I) - f(T_j)\|_2$

No need to compute $f(T_j) \forall j$,
these can be directly stored

Can we do any better?
.... After all, the network was not trained
for this purpose...

Metric Learning

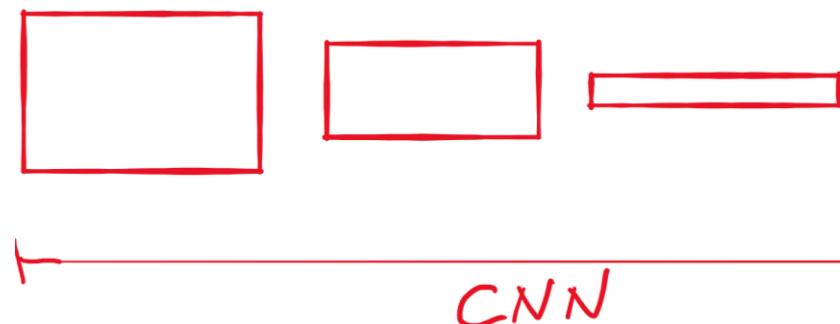
We are still comparing latent representations trained for classification (over a few persons?), while not for comparing images

A more appealing perspective would be to train the network to measure distances between images

We would like to train the weights W of our CNN such that

$$\|f_W(I) - f_W(T_i)\|_2 < \|f_W(I) - f_W(T_j)\|_2 \quad \forall j \neq i$$

When I belongs to class i



Siamese Networks

I



T_2

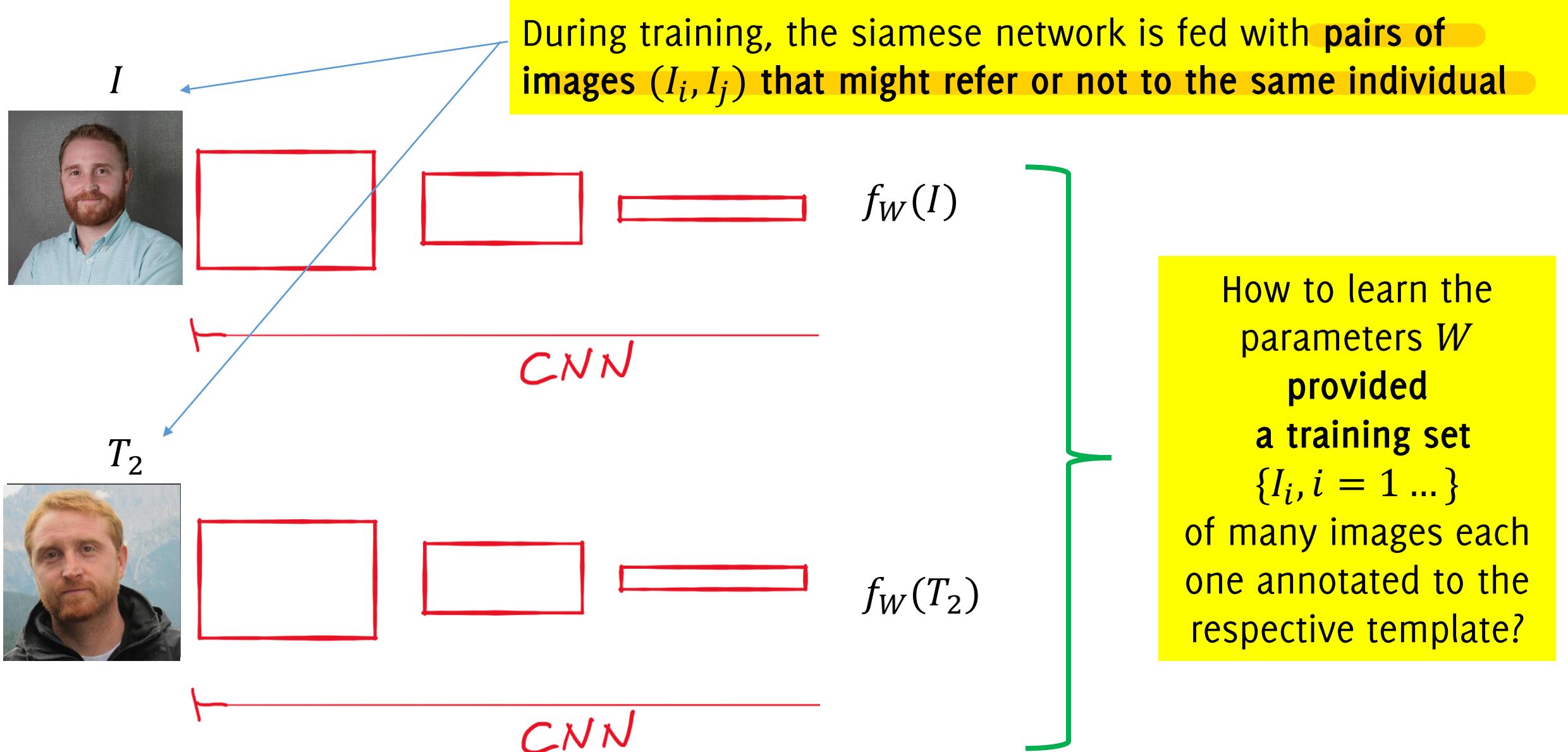


$$f_W(I)$$

$$f_W(T_2)$$

The «two» networks have to **perform the same operations**, using the same weights W . Hence the term **Siamese**

How to Train the Siamese Network?



Contrastive Loss

The *contrastive loss function* is defined as follow:

$$W = \operatorname{argmin}_{\omega} \sum_{i,j} \mathcal{L}_{\omega}(I_i, I_j, y_{i,j})$$

Where:

$$\mathcal{L}_{\omega}(I_i, I_j, y_{i,j}) = \frac{(1 - y_{i,j})}{2} \|f_{\omega}(I_i) - f_{\omega}(I_j)\|_2 + \frac{y_{i,j}}{2} \max(0, m - \|f_{\omega}(I_i) - f_{\omega}(I_j)\|_2)$$

- $y_{i,j} \in \{0,1\}$ is the label associate with the input pair (I_i, I_j) :
 - 0 when (I_i, I_j) refers to the same person
 - 1 otherwise
- m is a hyperparameter indicating the margin we want (like in Hinge Loss)
- $\|f_{\omega}(I_i) - f_{\omega}(I_j)\|_2$ is the distance in the latent space.

Triplet Loss

A loss function such that a training sample I is compared against

- P a positive input, referring to the same person
- N a negative input, referring to a different person

We train the network to minimize the distance from the positive samples and maximize the distance from the negative ones

$$\mathcal{L}_\omega(I, P, N) = \max(0, m + (\|f_\omega(I) - f_\omega(N)\|_2 - \|f_\omega(I) - f_\omega(P)\|_2))$$

Triplet loss forces that a pair of samples from the same individual are smaller in distance than those with different ones.

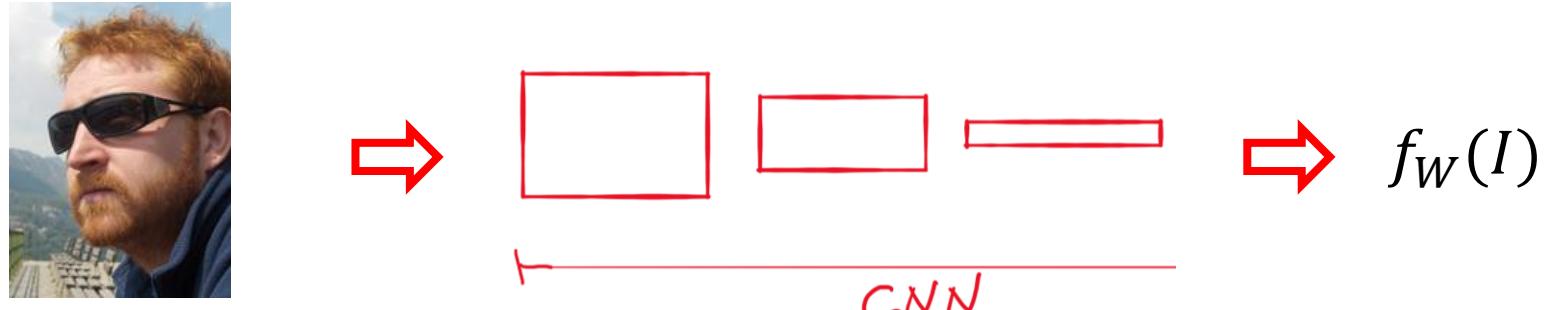
m always play the role of the margin.

The selection of triplets for training is a matter of study

Just to Recap

When a new image has to be verified

1. We feed the image I to the trained network, thus compute $f_W(I)$



2. Identify the person having average minimum distance from templates (in case there are many associated to the same individual)

$$\hat{i} = \operatorname{argmin}_u \frac{\sum_{T_{u,j}} \|f_W(I) - f_W(T_{u,j})\|_2}{\#\{T_u\}}$$

3. Assess whether

$$\frac{\sum_{T_{i,j}} \|f_W(I) - f_W(T_{i,j})\|_2}{\#\{T_{i,j}\}} < \gamma$$

is sufficiently small, otherwise no identification

Other decision rules can be adopted (e.g. searching for the person giving the a template with a minimum distance)

... and if you want more
A few opportunities in DL research / study

Advanced Deep Learning Models and Methods

A PhD course, but you are welcome to attend (with or without registering)

- 6 half-a-day lectures in Sala Conferenze streaming on my webex room,
 - exam by a project.
-
- **Deep Unsupervised Learning in Images, Giacomo Boracchi (Polimi) 14 – 18 February 7th**
 - **Learning with Limited Supervision, Alessandro Giusti (USI-SUPSI) 9 – 13 February 9th**
 - **Deep Reinforcement Learning Alessandro Lazaric (Facebook Paris) 14 – 18 February 10th**
 - **Variational AutoEncoders with Applications to Anomaly Detection, Luigi Malagò (TINS, RM) 14 – 18 February 14th**
 - **Deep Learning on Graphs and Structured Computation Models, Jonathan Masci (NNAISENSE) 14 – 18 February 16th**
 - **Privacy preserving learning, Matteo Matteucci (Polimi) 14 – 18 February 17th**