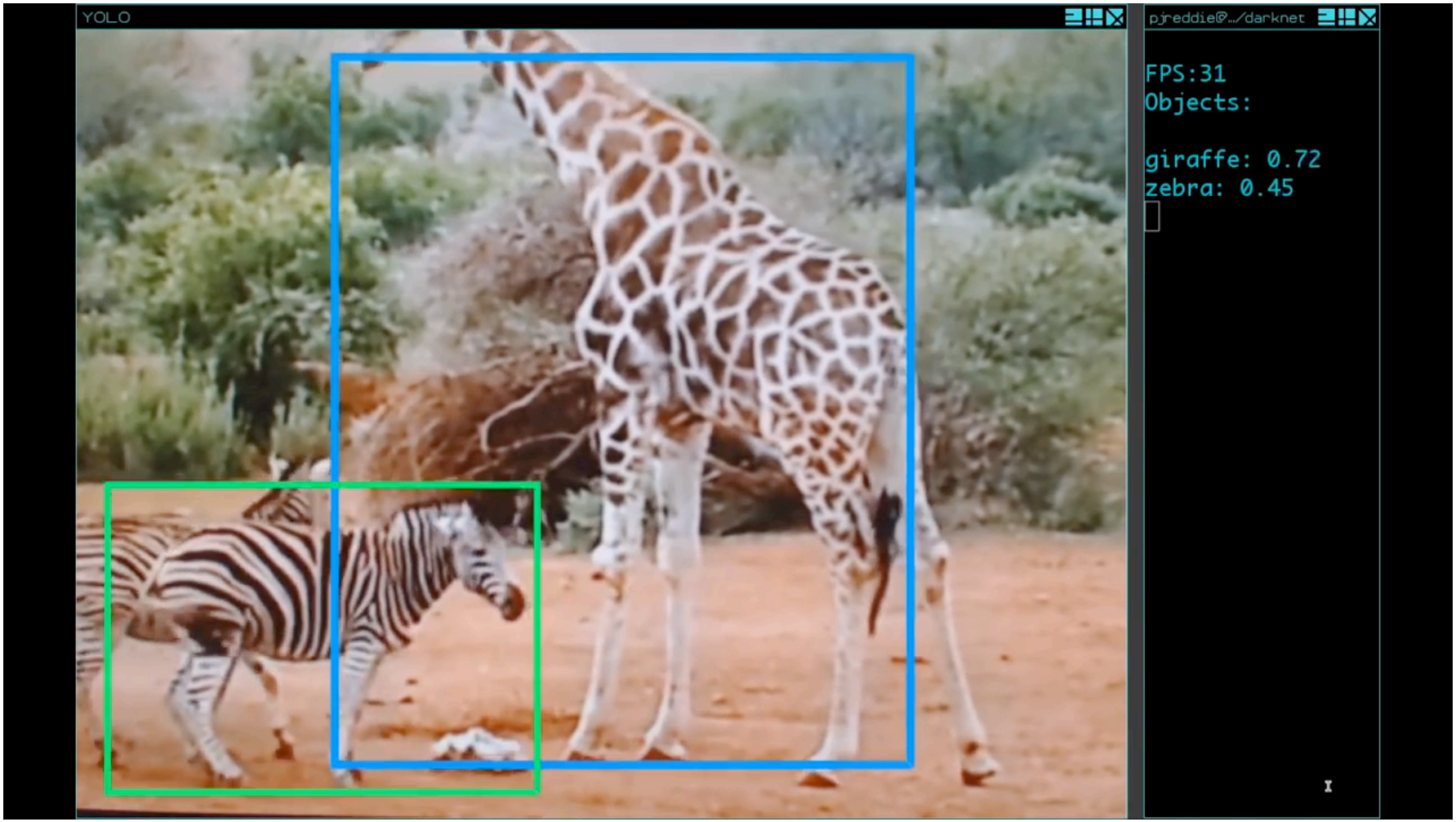
Deep Learning: Bestiary

Vincent Lepetit

Object Recognition and Localization

Classification+Localization Task



VGG [Simonyan & Zisserman, 2014]

At ILSVRC workshop, 2014:

Localisation task: 1st place, 25.3% error
Classification task: 2nd place, 7.3% error

19 weight layers;
Very small convolution kernels (3x3).

Training

Optimization:

- multinomial logistic regression;
- mini-batch gradient descent with momentum;
- dropout and weight decay regularisation;
- fast convergence (74 training epochs).

Initialisation:

1. Shallow net (11 layers) uses Gaussian initialisation;
2. Deeper nets:

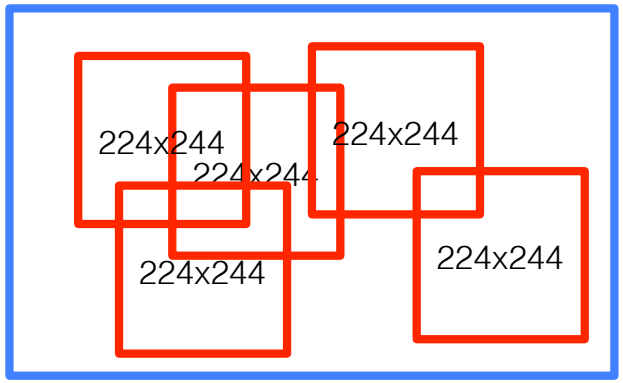
- top 4 convolutional and FC layers initialised with 11 layer net;
- other layers – random Gaussian initialisation.

Training (2)

Input during training: Fixed-size 224x224 crop

But images have various size. Solution:

1. rescale images to a random size preserving aspect ratio;
2. random 224x224 crops:



+ Augmentation: random flip and RGB channel shuffling.

2-3 weeks for training on 4 x NVIDIA Titan GPUs.

Testing

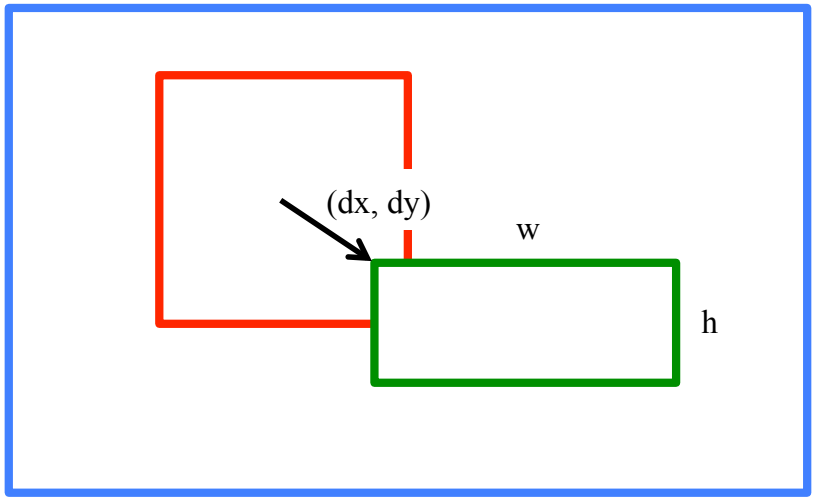
Dense application over the whole image:

Fully Connected layers are converted into convolutional layers;
For each class: sum of the scores over all image locations.

+ Jittering, multiple image sizes, horizontal flips.

Localisation

1. Last layer predicts a bounding box for each class:



Trained with Euclidean loss.

Bounding box parameterisation: **(dx, dy, w, h)**

2. Bounding boxes are merged;
3. Resulting boxes are scored by a classification network.

ResNet [He et al, CVPR 2016] ILSVRC 2015 Winner

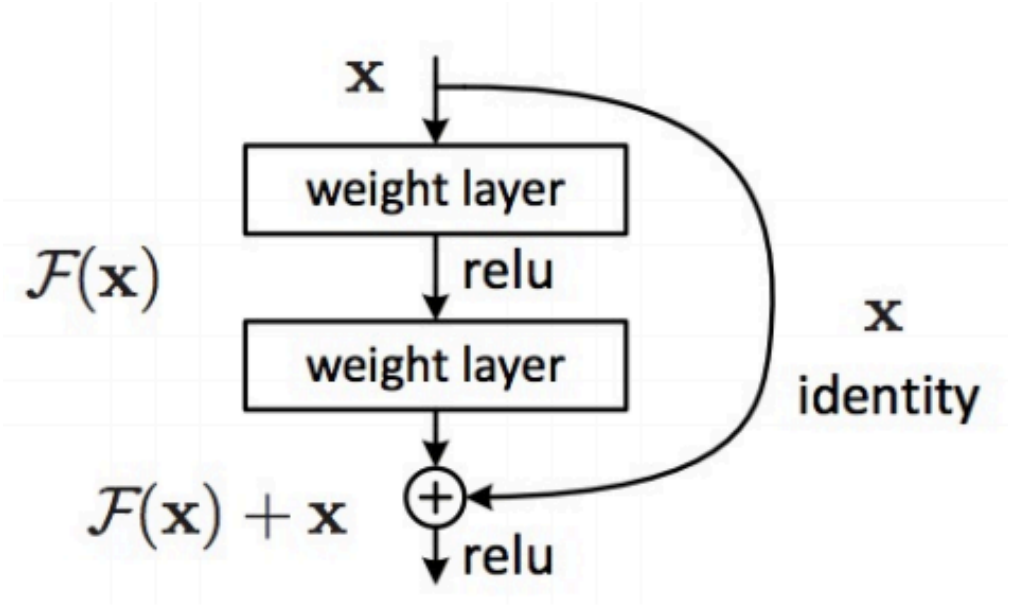
AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

The Residual Module

Uses 'skip' or 'shortcut' connections:



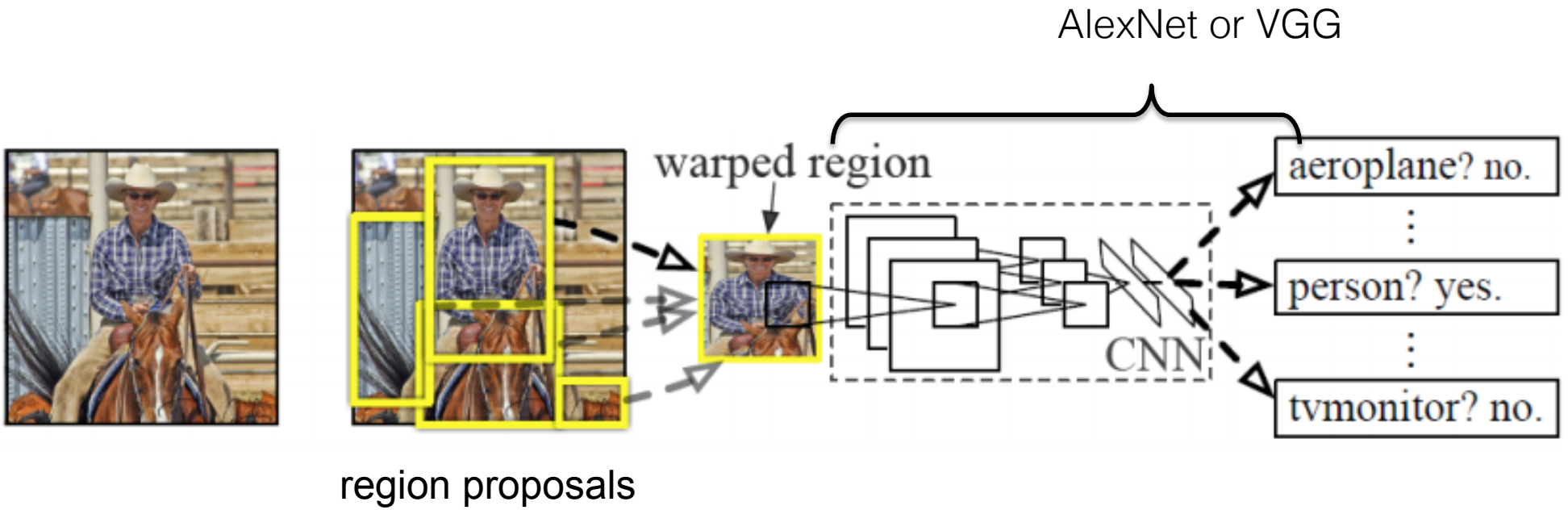
- Make it easy for network layers to represent the identity mapping;
- Need to skip at least two layers (for some reasons).

In Keras

```
model = keras.applications.vgg16.VGG16()

model = keras.applications.resnet50.ResNet50()
```

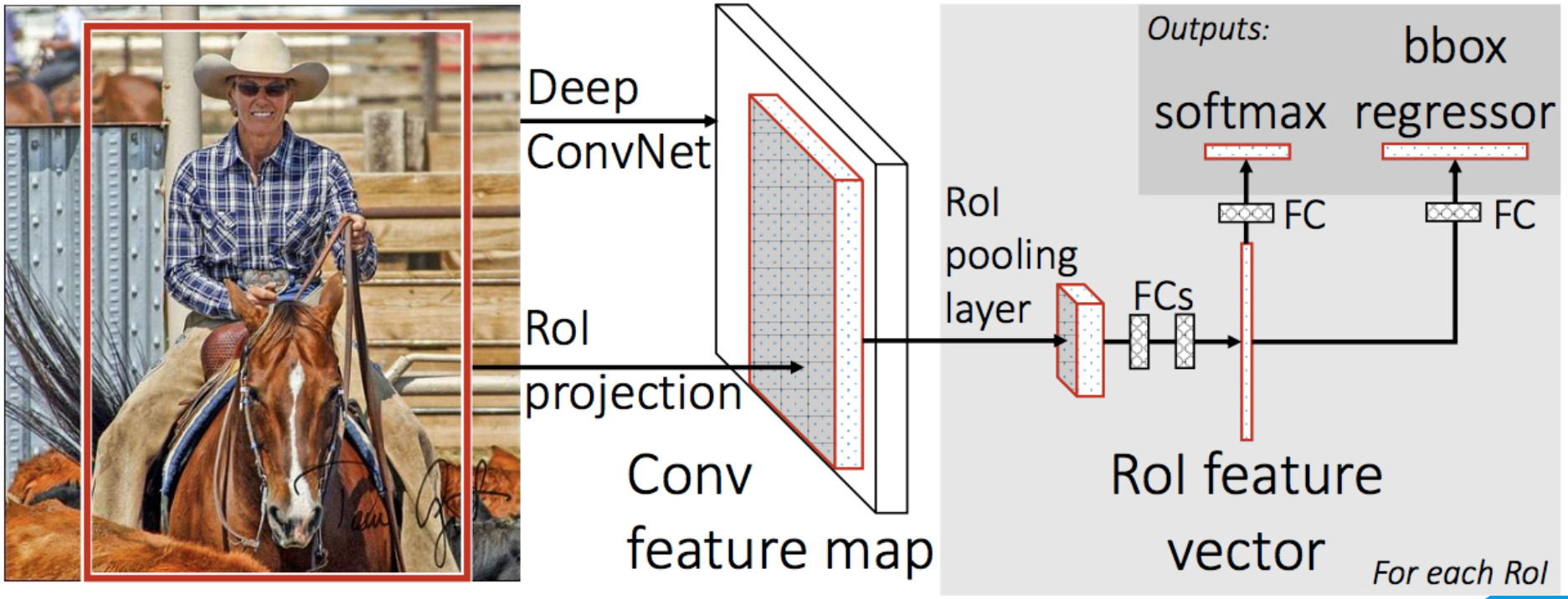
```
# Simple ResNet block:

output = layers.Conv2D(n_input, kernel_size=(3, 3))(input)
output = layers.ReLU(output)
output = layers.Conv2D(n_input, kernel_size=(3, 3))(output)
output = layers.add([input, output])
output = layers.ReLU()(output)
```

R-CNN [Girshick et al 2013]: Region Proposals + CNN

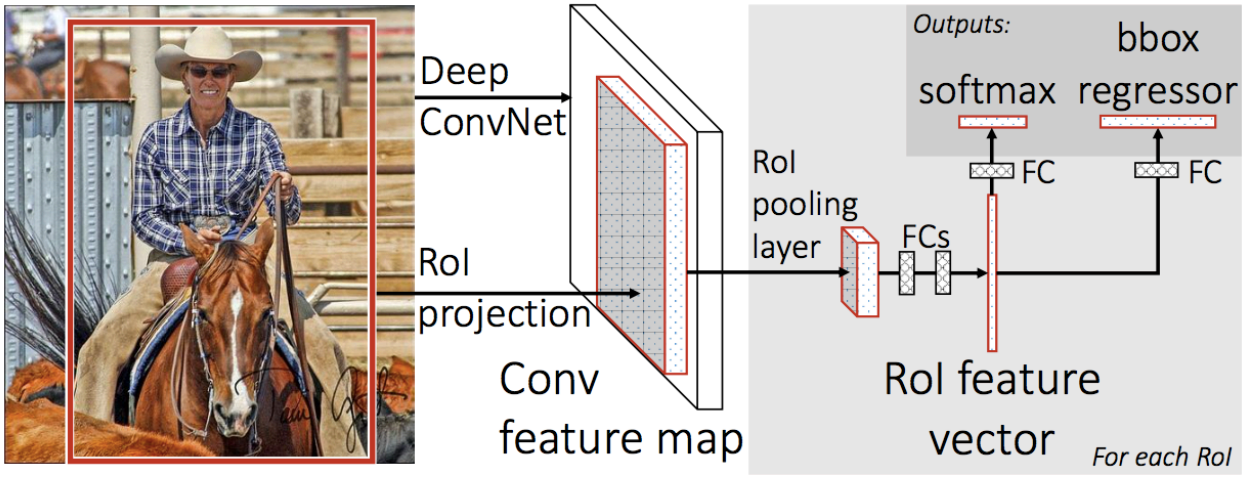


Fast R-CNN

213× faster at test-time than R-CNN.
Also easier to train.
Convolutions are applied only once to the image:



Fast R-CNN



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

$$L_{\text{cls}}(p, u) = -\log p_u$$ u: true class

$u = 0$: Background class:

v: true bounding box, t^u: predicted bounding box

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

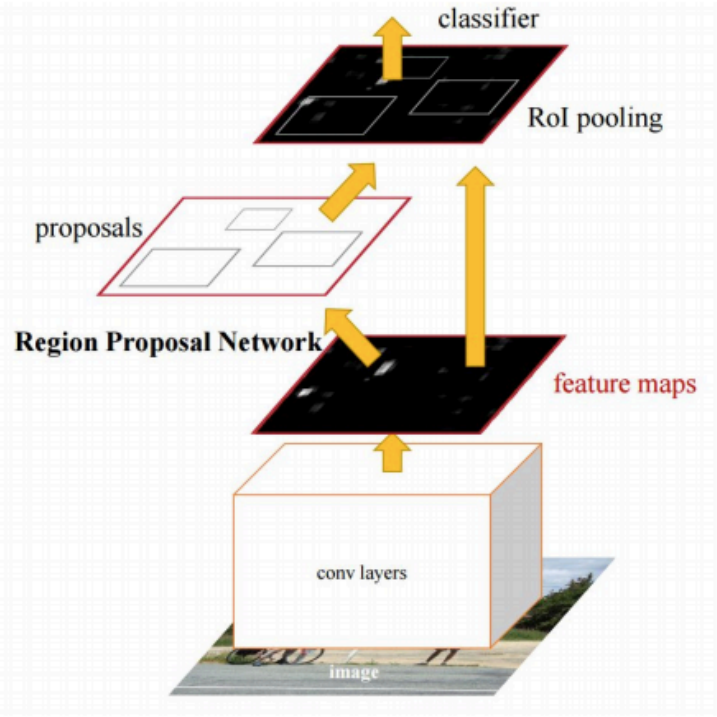
with $$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Faster R-CNN [Ren et al, 2015]

5 frames per second, including all steps.

Learns and integrates the region proposal part (bottleneck in Fast R-CNN).

Introduces Region Proposal Networks that use the same feature maps as the object detector network:

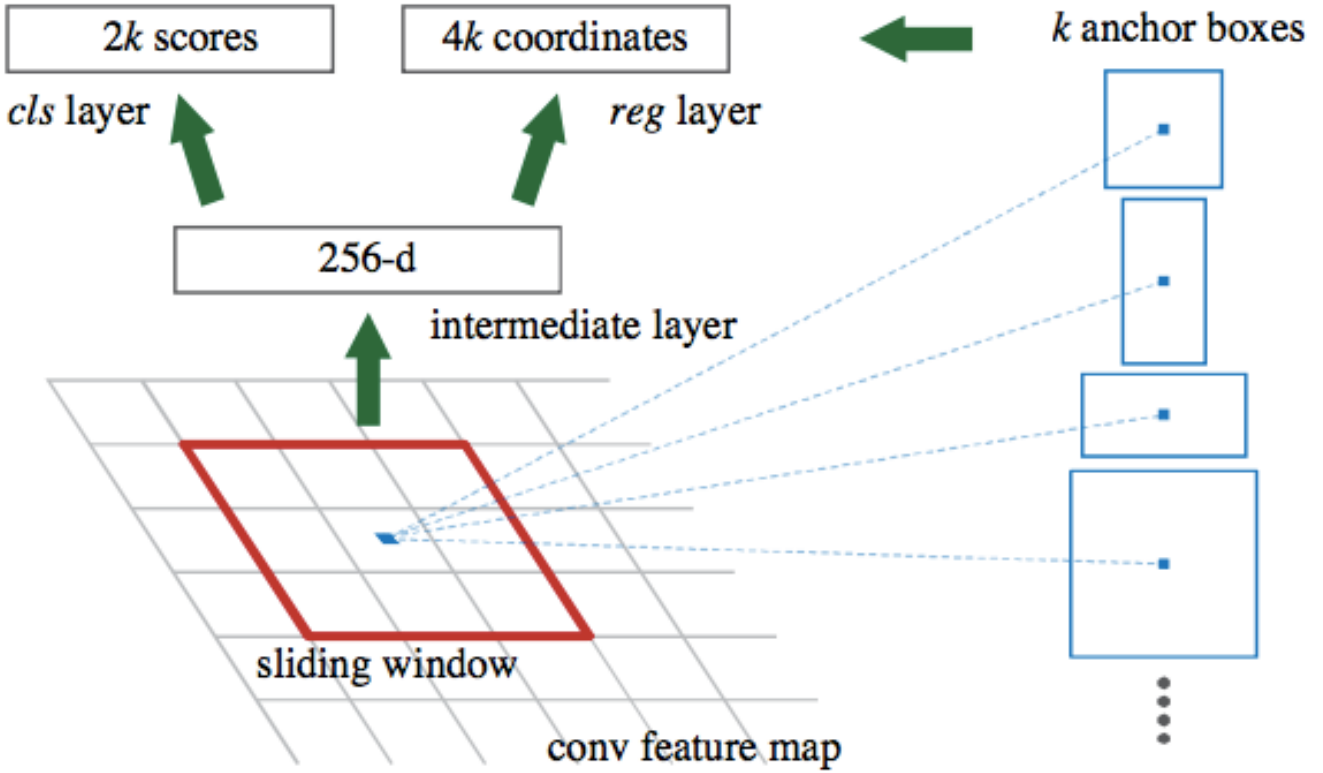


Faster R-CNN: Region Proposal Network

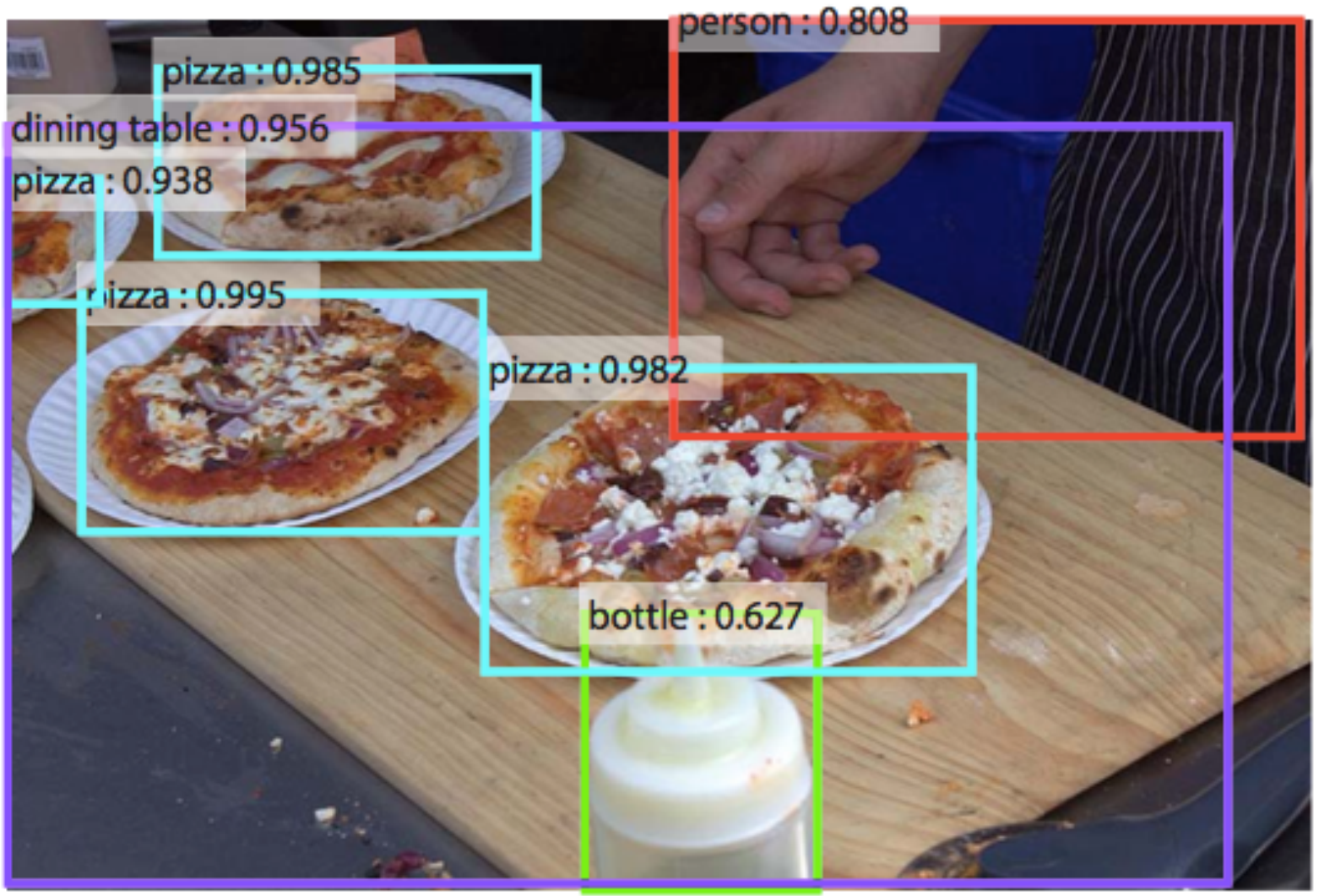
Binary classification for each box: A box is positive if it overlaps with a ground truth box

offset wrt anchor boxes

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

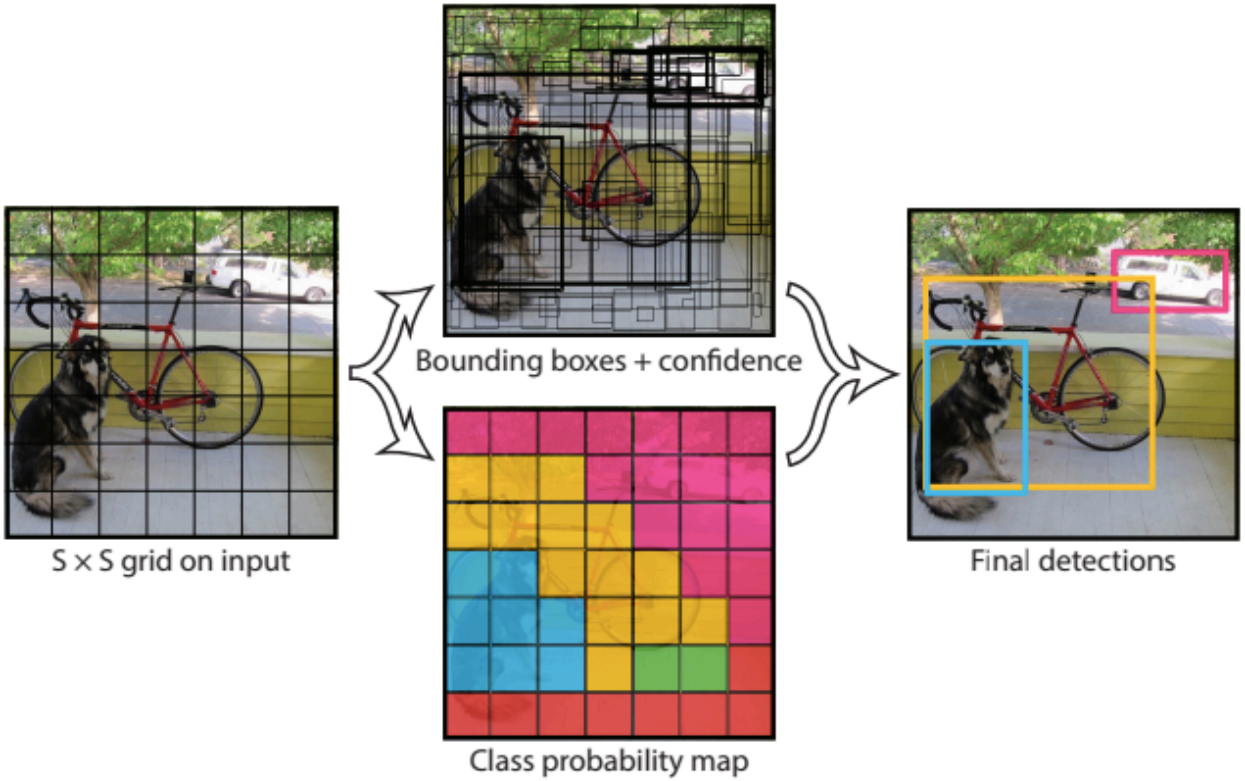


Faster R-CNN: Selected Results



YOLO [Redmon et al, 2015]

"You Only Look Once": Runs once on entire image.



The following predictions are made for each cell in an S x S grid:

C conditional class probabilities $Pr(Class_i \mid Obj)$

B bounding boxes (4 parameters each)

B confidence scores Pr(Obj)*IoU

YOLO: Loss Function

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

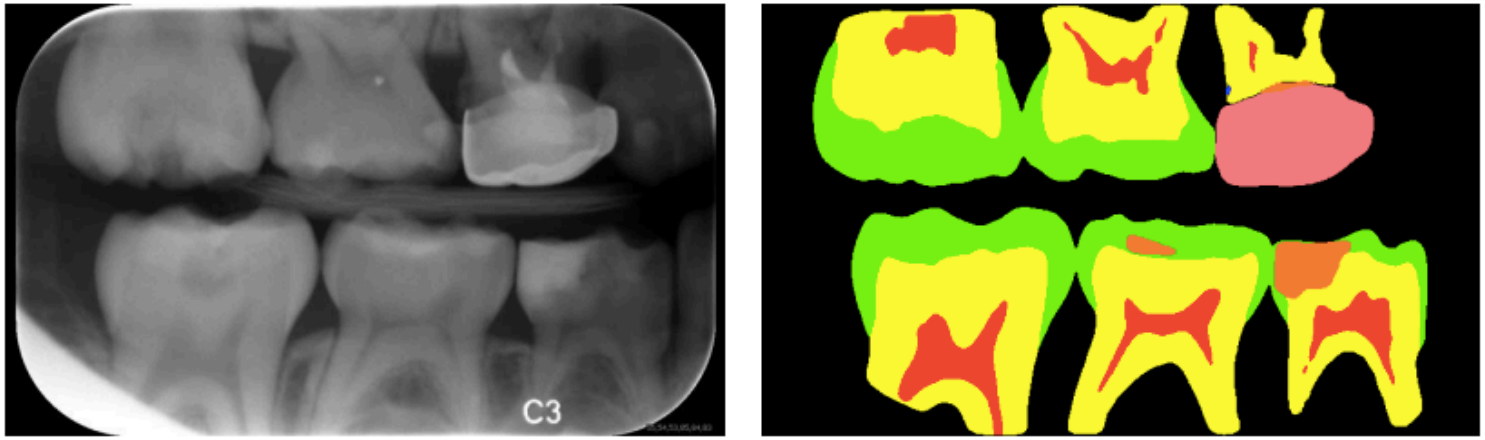
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \quad (3)$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

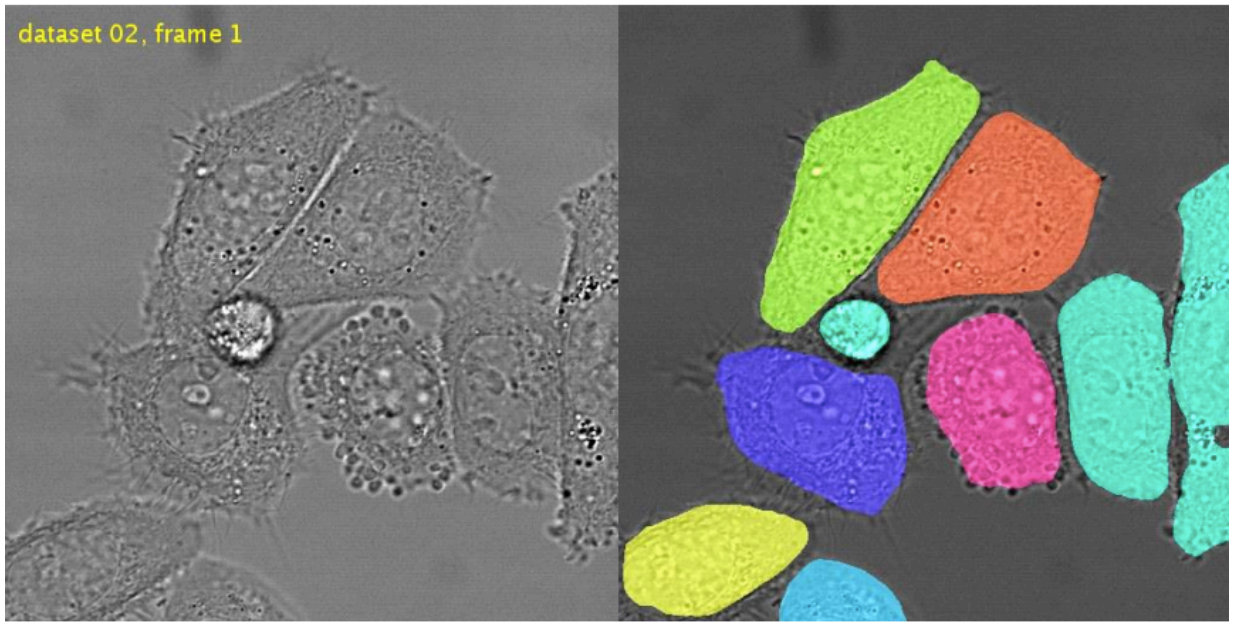
YOLO9000 [Redmon et al, 2015]

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Segmentation: U-Net [Ronneberger, 2015]



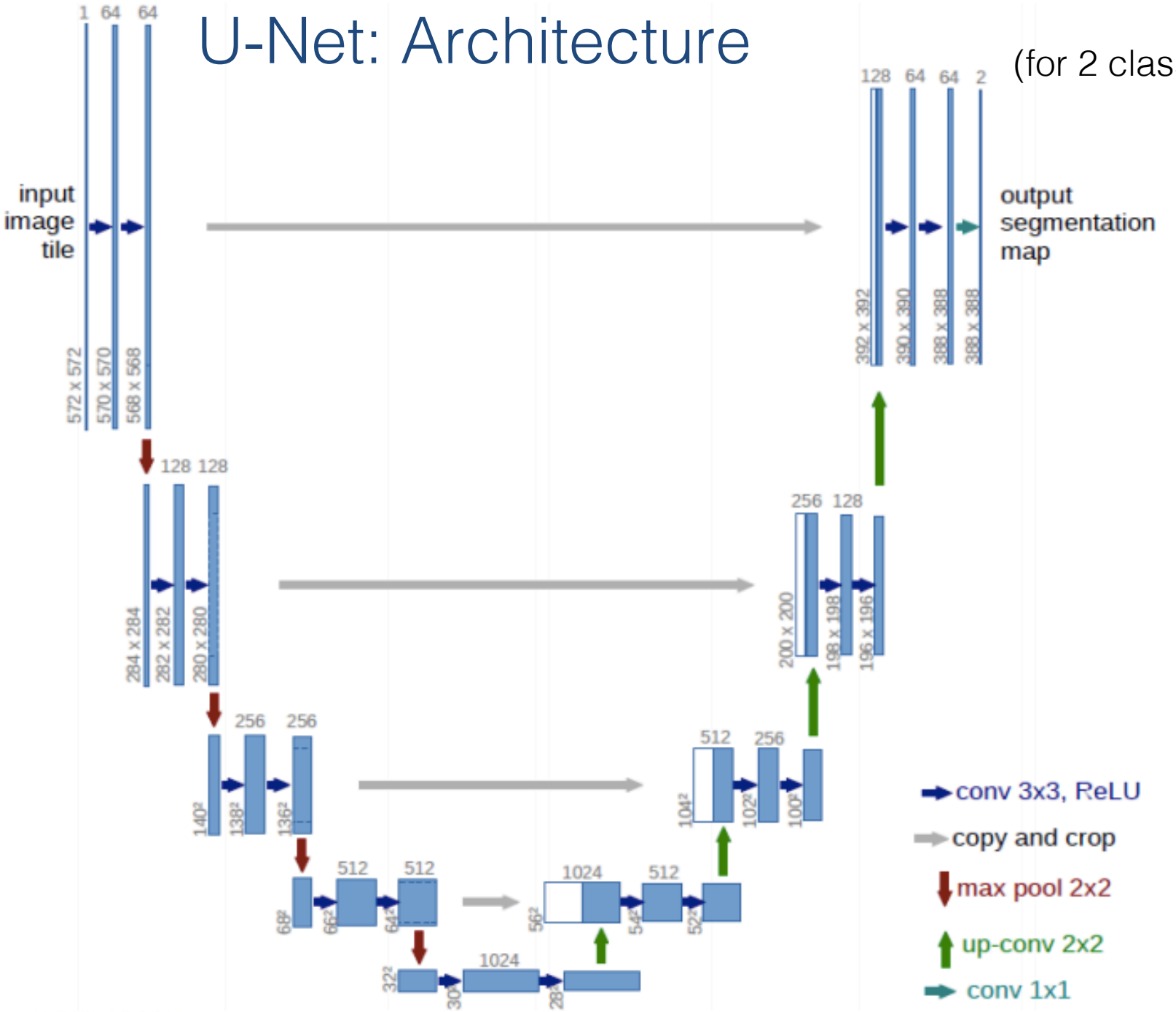
X-ray dental segmentation, ISBI 2015 Challenge, Rank 1



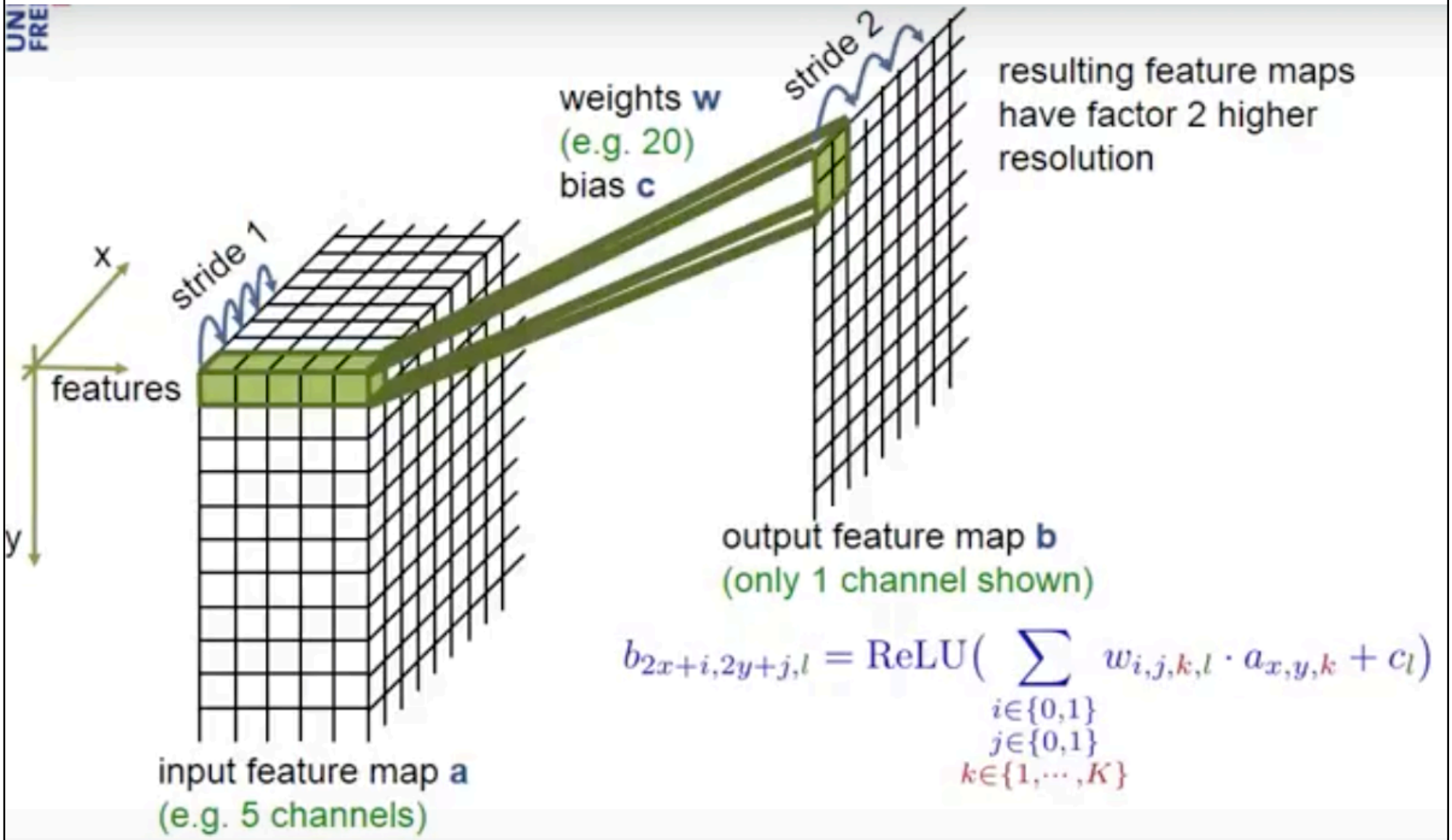
Light microscopy, DIC-HeLa cell tracking
ISBI 2015 Challenge: Rank 1

U-Net: Architecture

(for 2 classes)

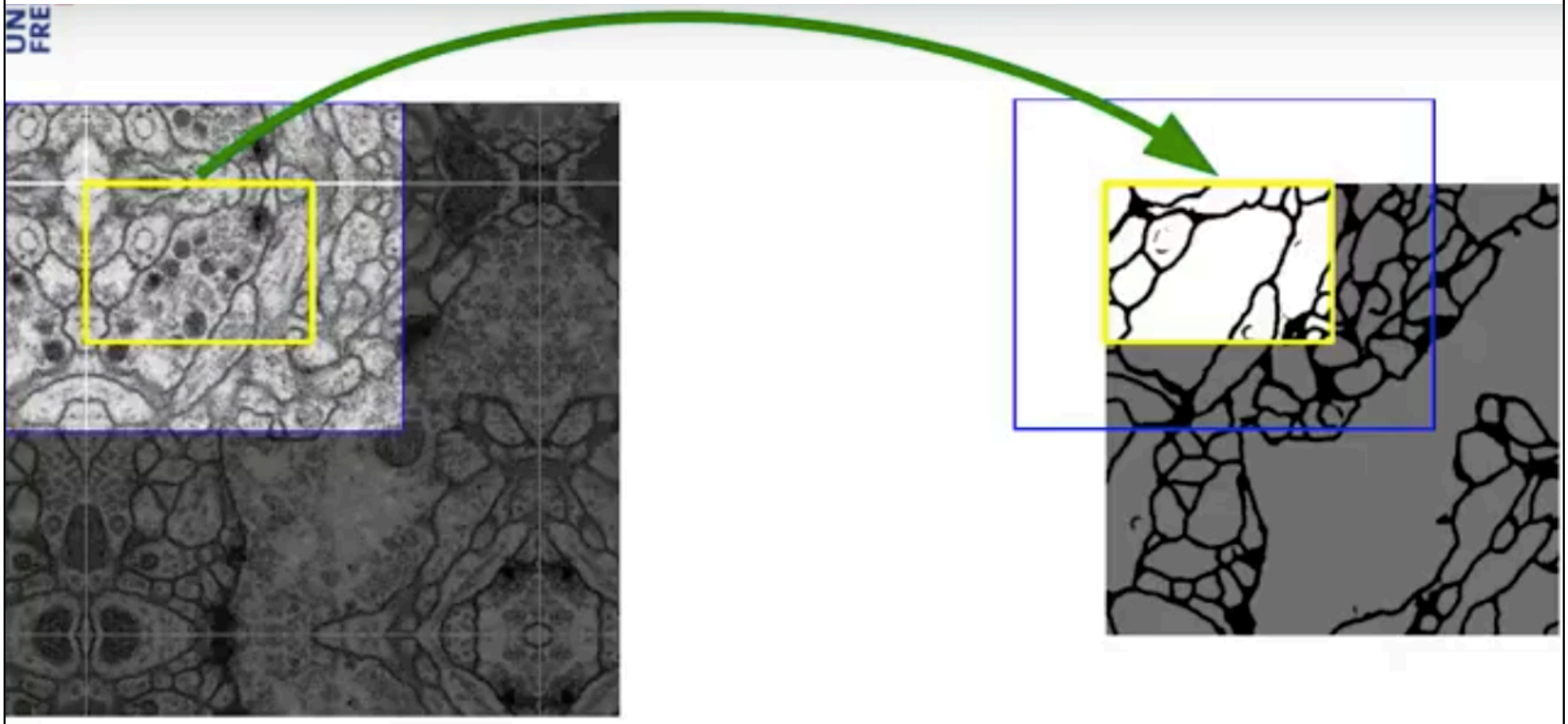


Original Upscaling Layer

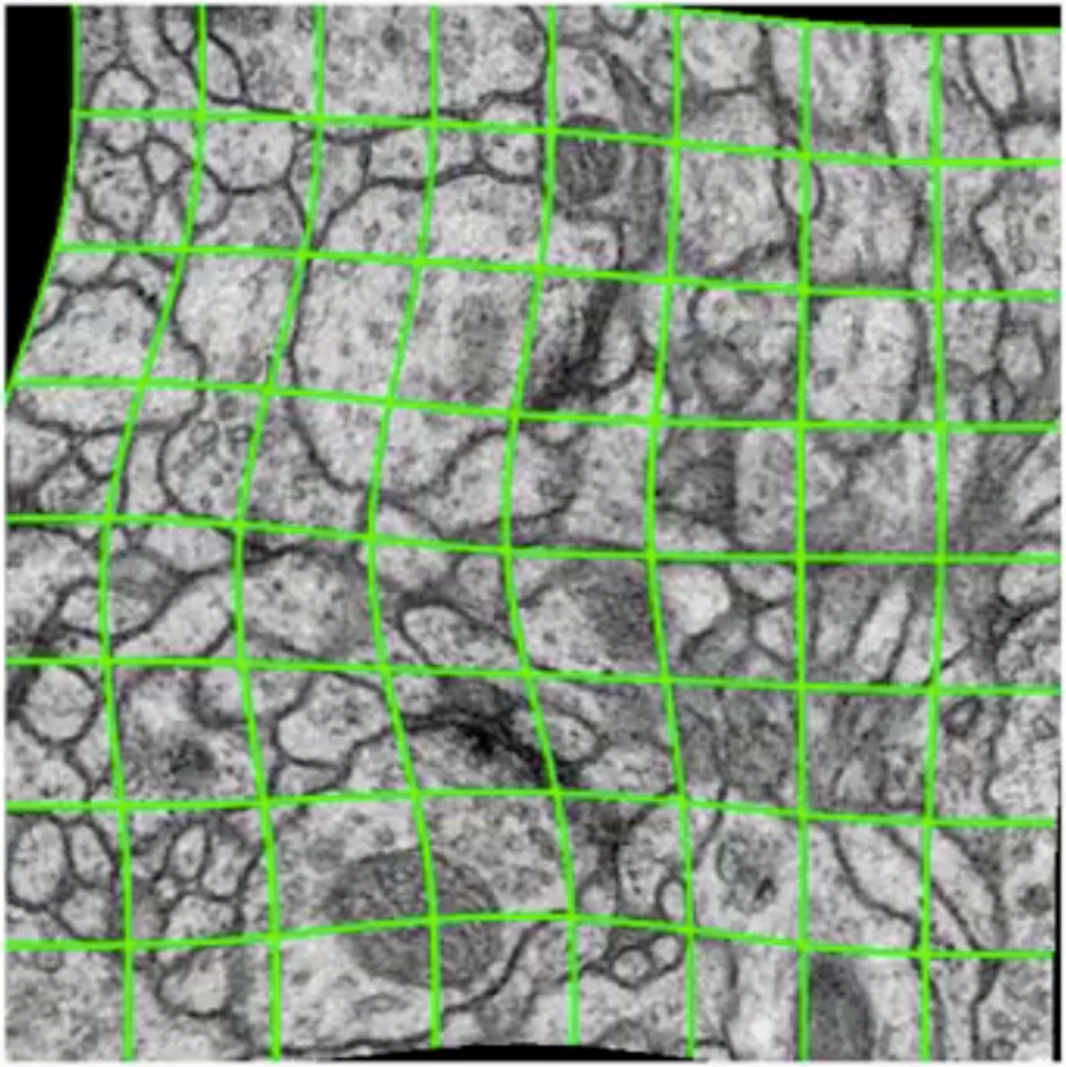


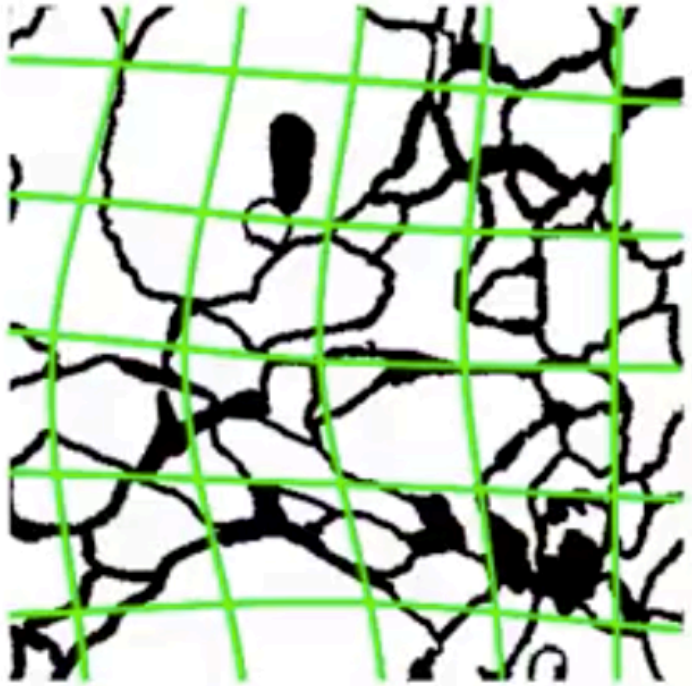
```
concat_axis = 3

conv1 = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)

  …

conv5 = layers.Conv2D(512, (3, 3), activation='relu', padding='same')(pool4)

up_conv5 = layers.UpSampling2D(size=(2, 2))(conv5)
ch, cw = self.get_crop_shape(conv4, up_conv5)
crop_conv4 = layers.Cropping2D(cropping=(ch,cw))(conv4)
up6 = layers.concatenate([up_conv5, crop_conv4], axis=concat_axis)
conv6 = layers.Conv2D(256, (3, 3), activation='relu', padding='same')(up6)

 …

up_conv8 = layers.UpSampling2D(size=(2, 2))(conv8)
ch, cw = self.get_crop_shape(conv1, up_conv8)
crop_conv1 = layers.Cropping2D(cropping=(ch,cw))(conv1)
up9 = layers.concatenate([up_conv8, crop_conv1], axis=concat_axis)
conv9 = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(up9)

ch, cw = self.get_crop_shape(inputs, conv9)
conv9 = layers.ZeroPadding2D(padding=((ch[0], ch[1]), (cw[0], cw[1])))(conv9)
conv10 = layers.Conv2D(num_class, (1, 1))(conv9)
```

Overlapping Tiles



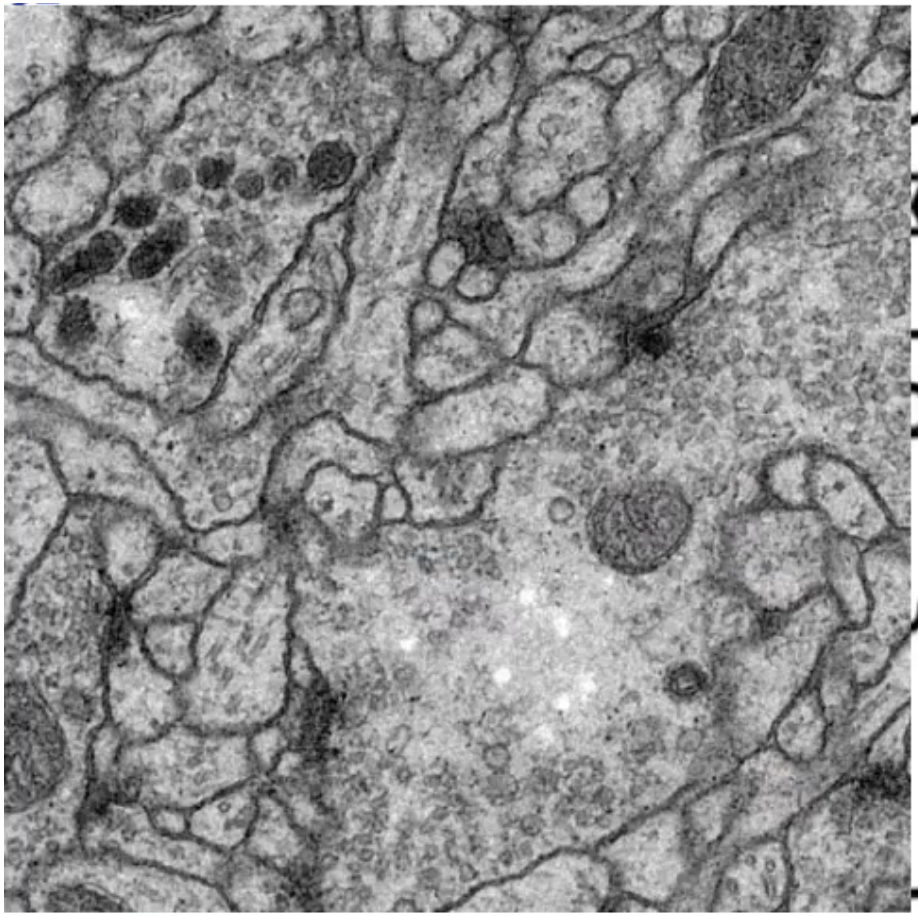
Dataset Augmentation



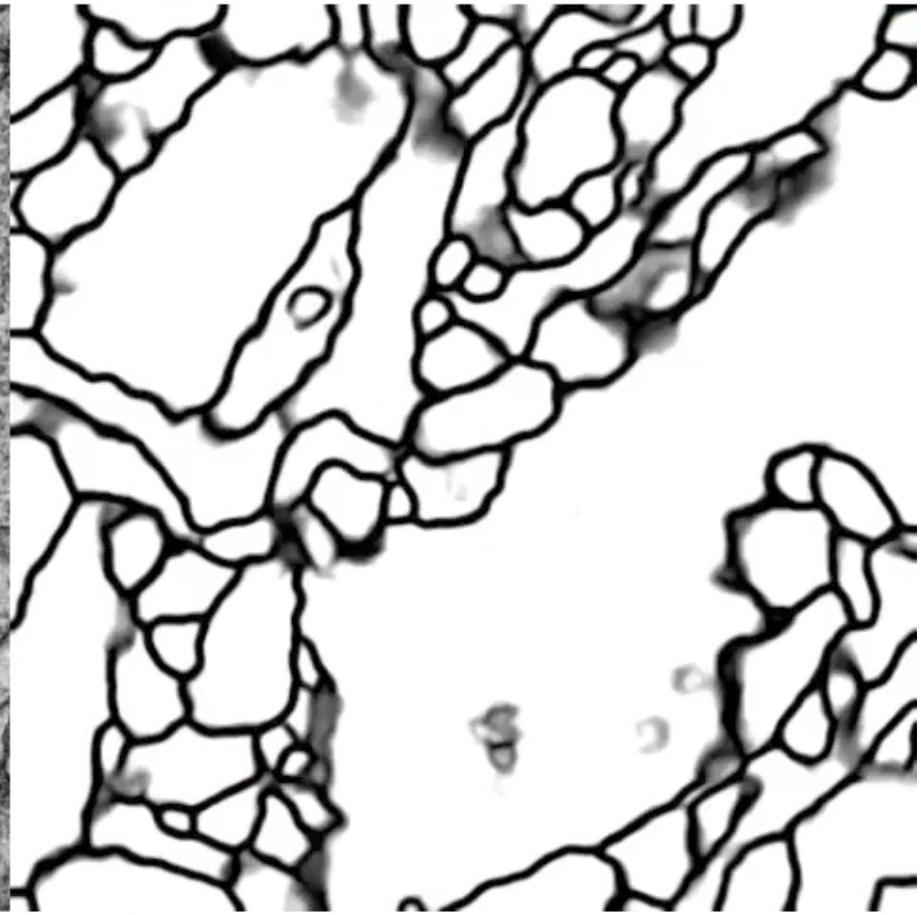


correspondingly deformed
manual labels

U-Net: Results

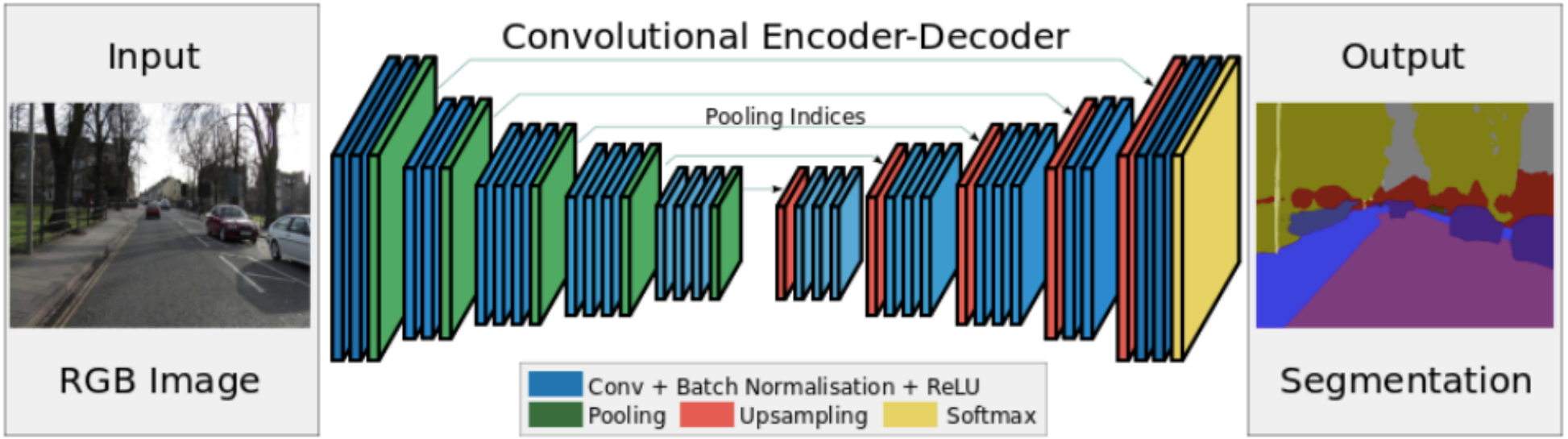


Input image

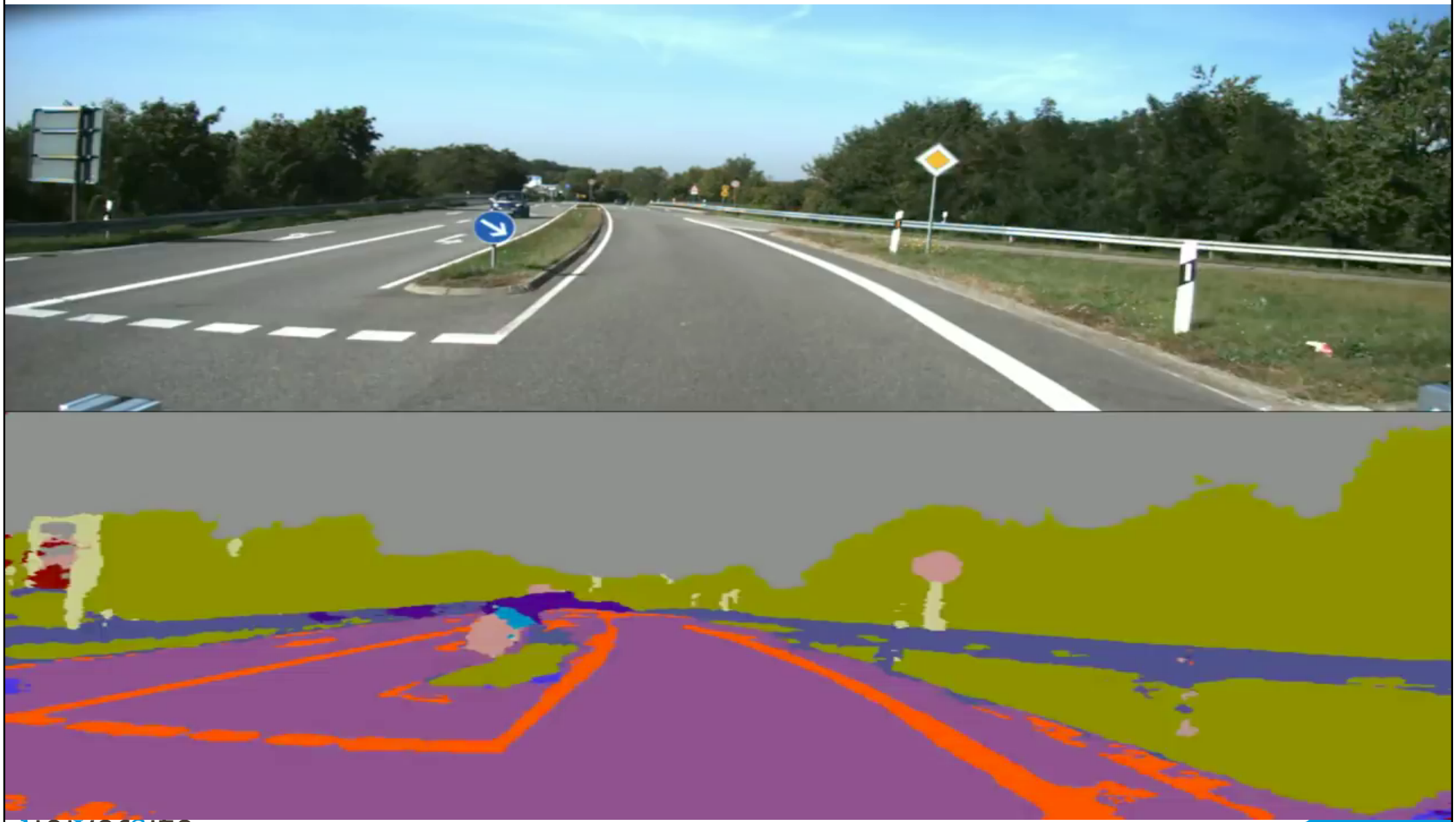


Our result: 0.000353 warping error
(**New best score** at submission march 6th, 2015)
Sliding-window CNN: 0.000420
Training time: 10h, Application: 1s per image

SegNet [Kendall et al, 2015]
Hourglass Network

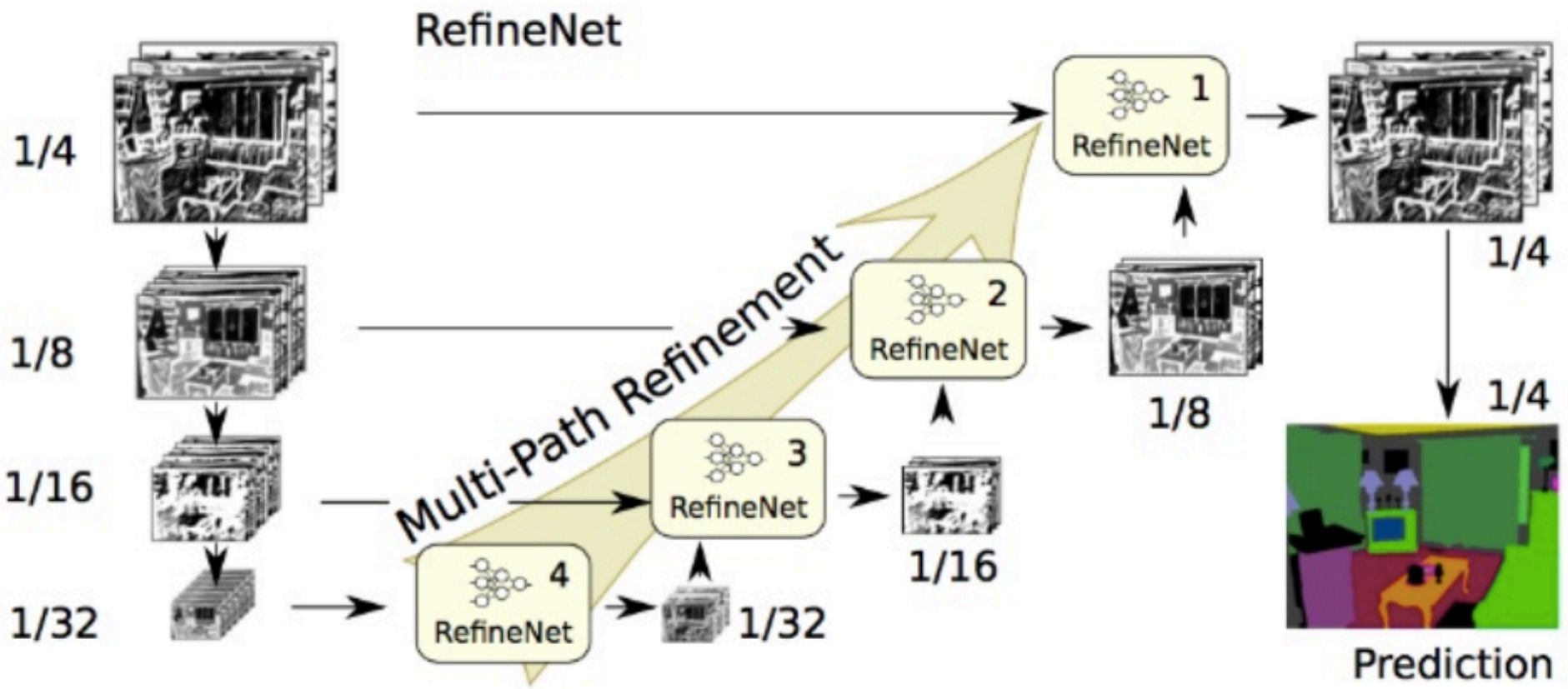


Segnet: Results



RefineNet [Lin et al, 2016] (on Cityscapes)

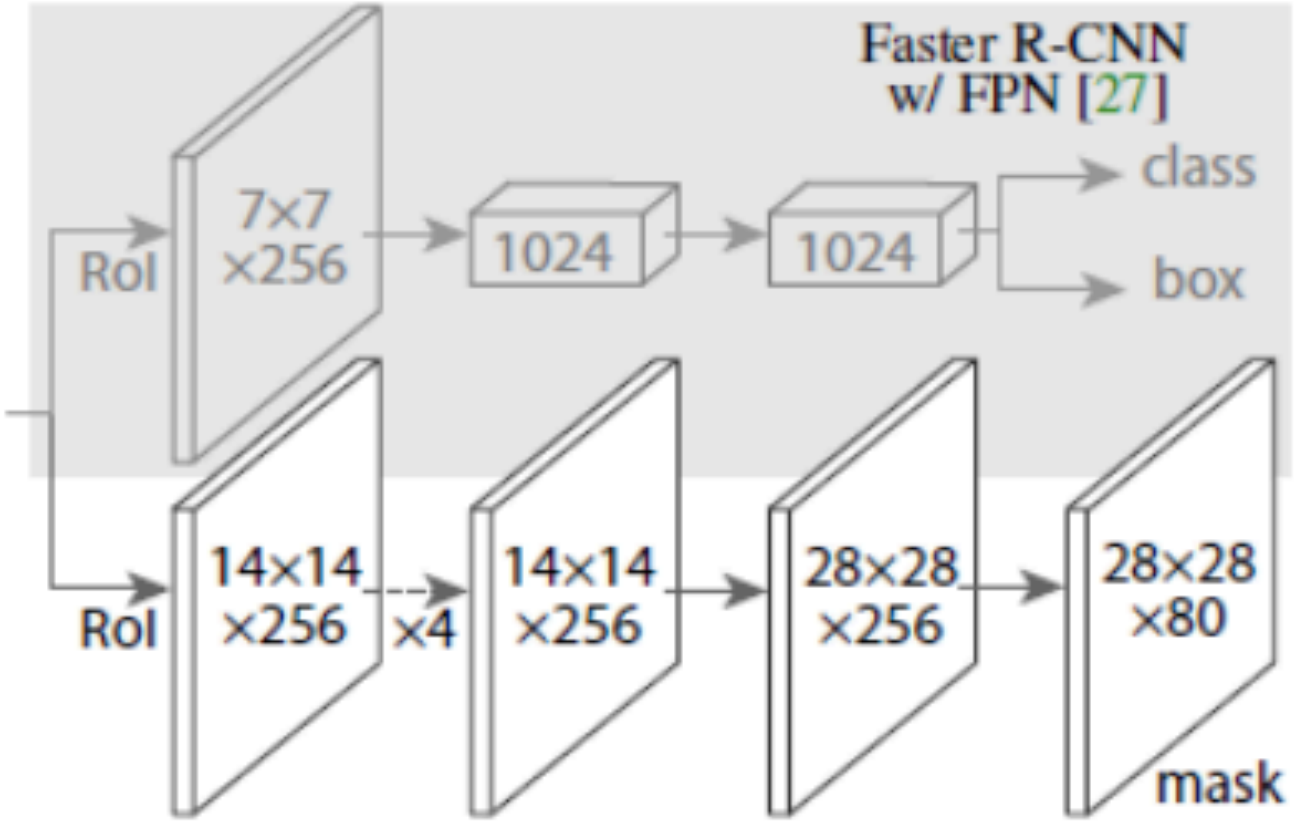
RefineNet [Lin et al, 2016]



RefineNet: More Results

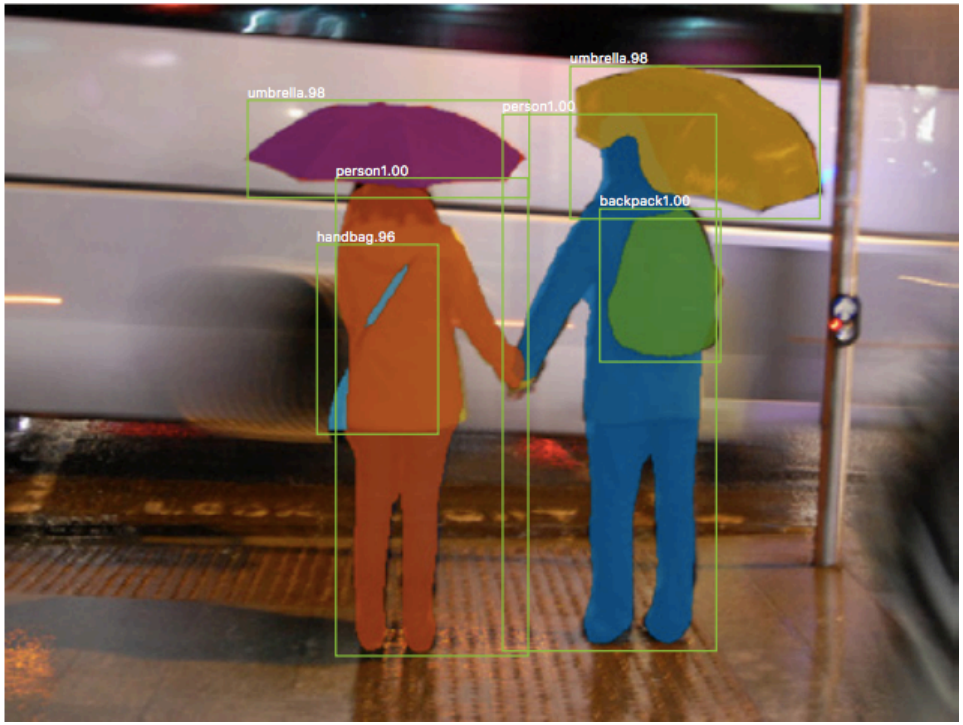
Mask-RCNN [He et al, 2017]

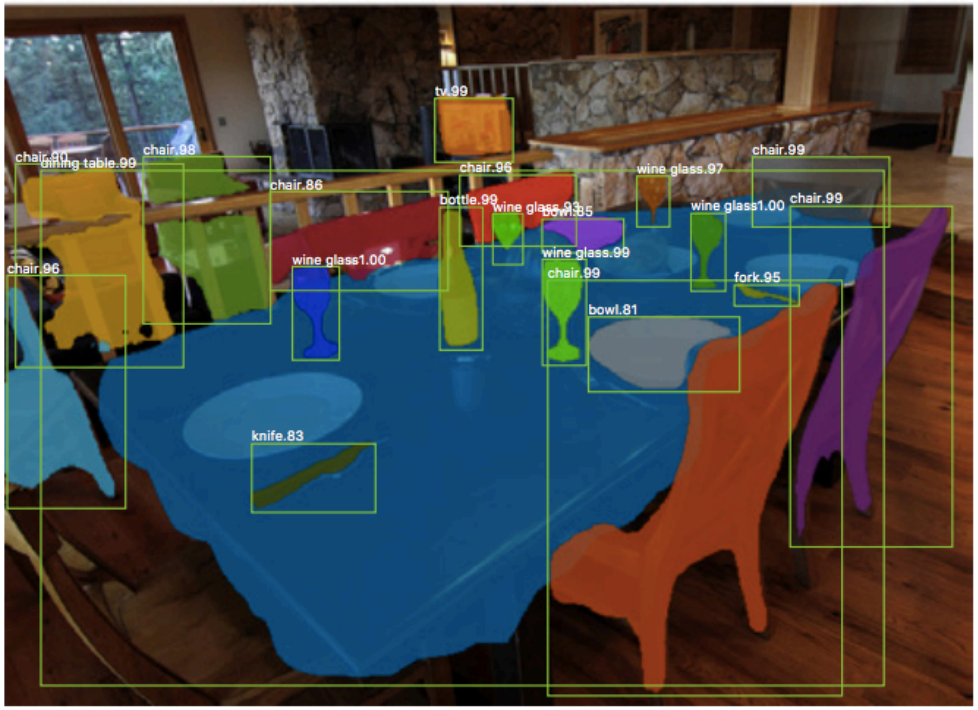
Add segmentation mask prediction for each ROI:



+bilinear interpolation

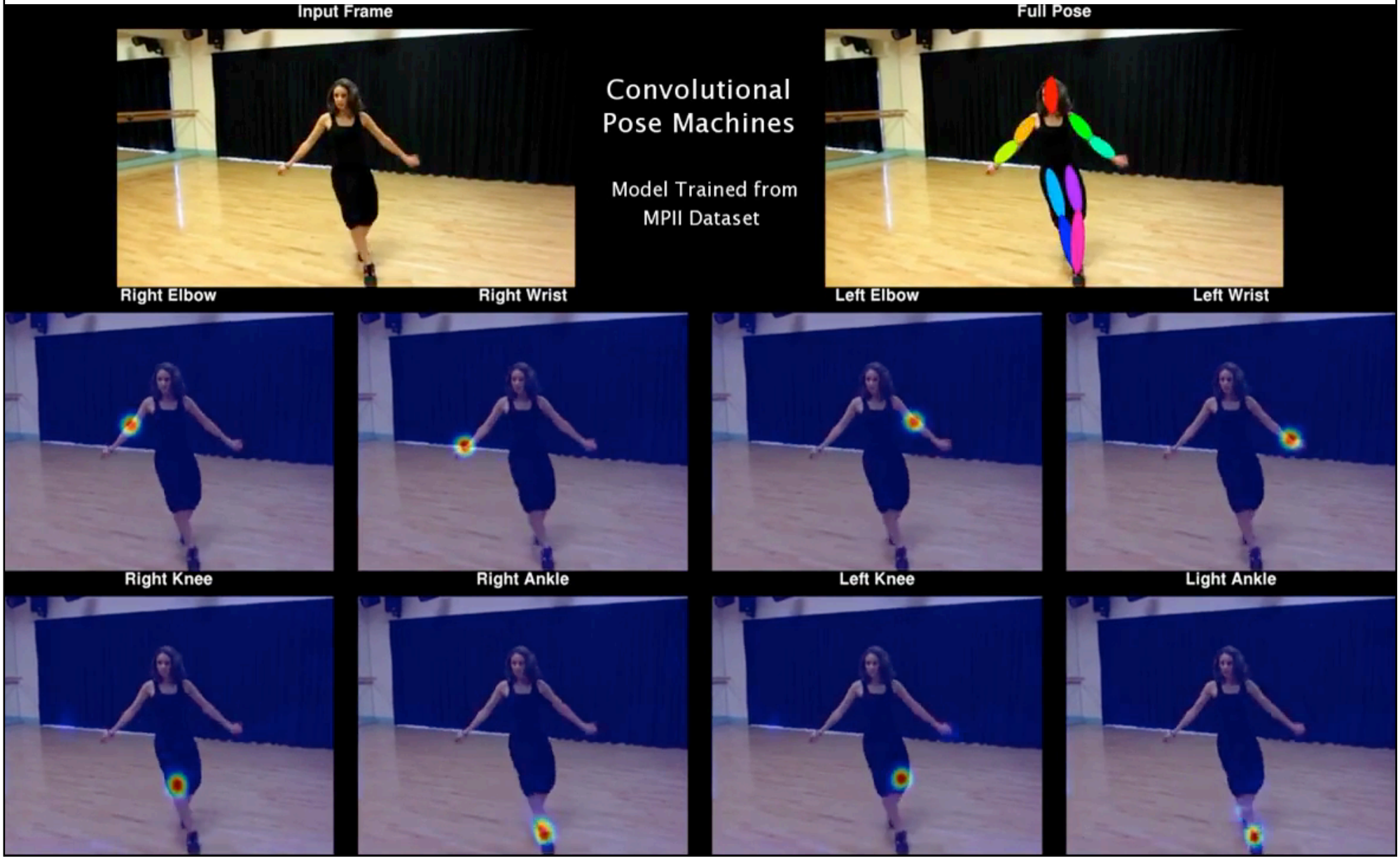
Mask-RCNN (Results on Instance Segmentation)





Articulated Pose Estimation

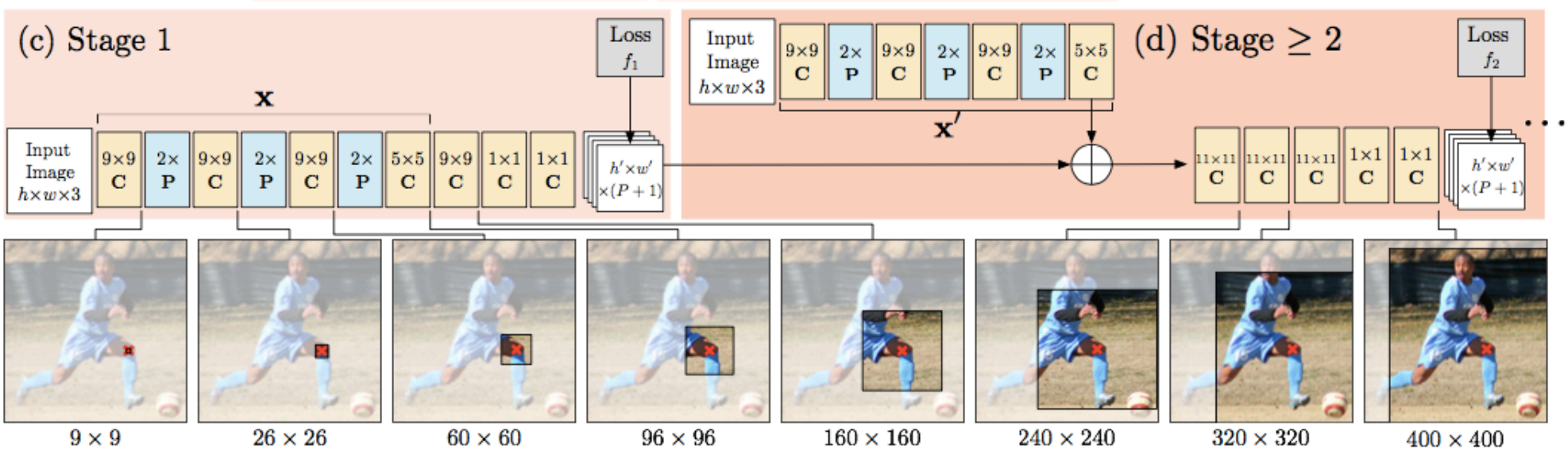
Convolutional Pose Machines



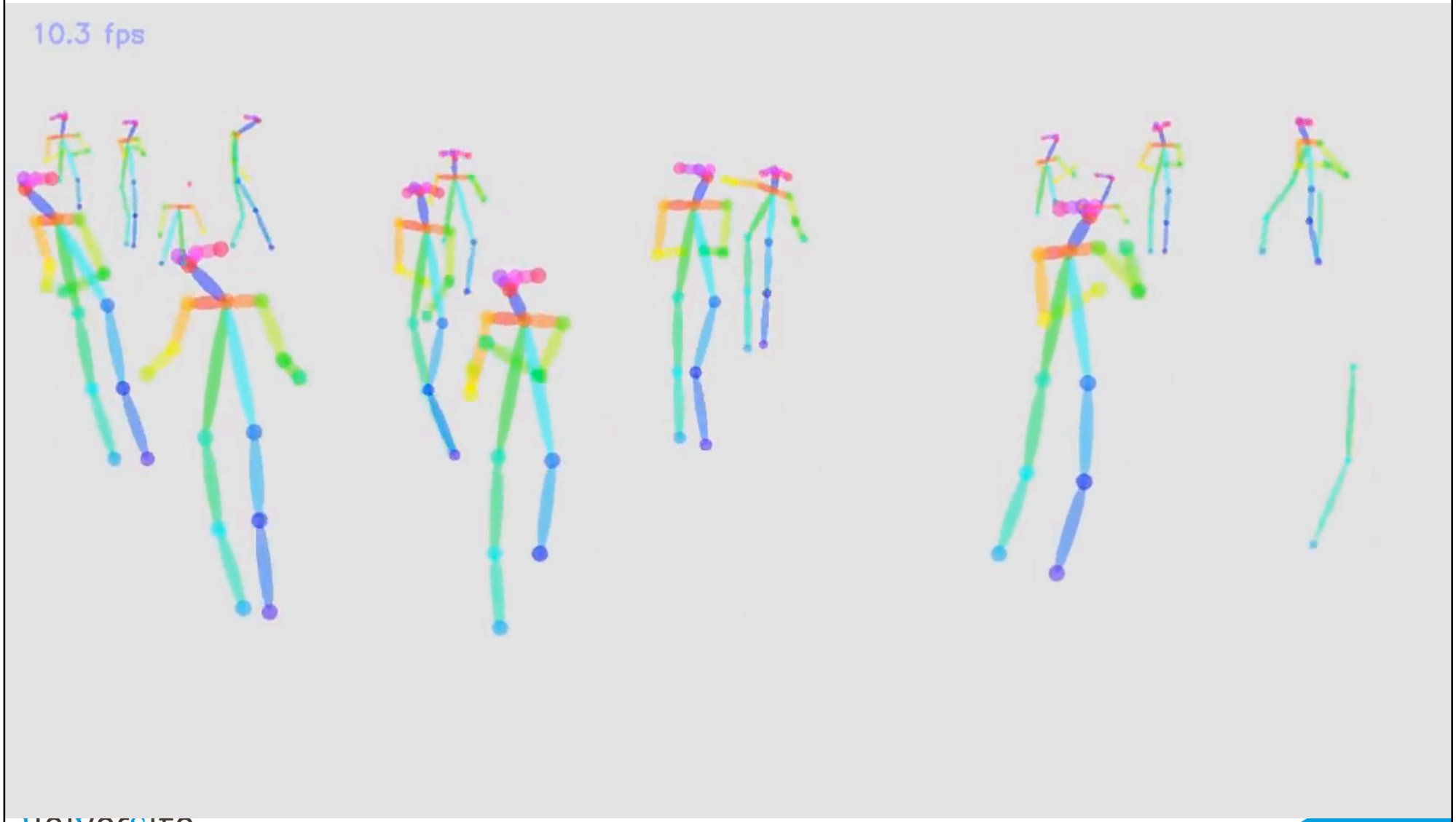
Convolutional Pose Machines

Input Image (a) Stage 1 (b) Stage 2 (c) Stage 3

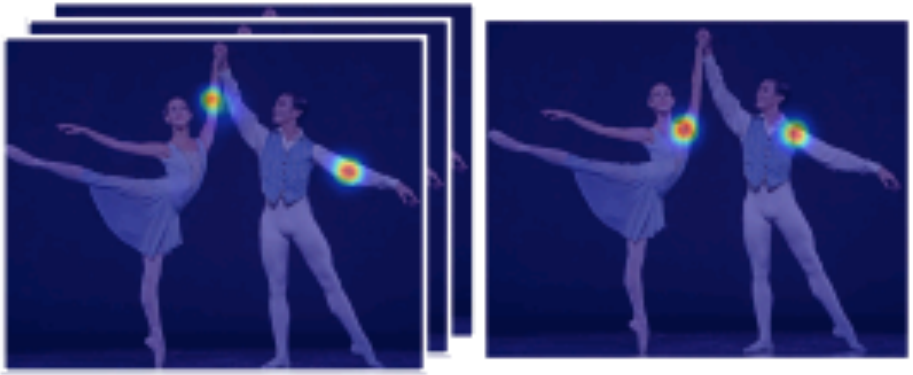
Convolutional Pose Machines



OpenPose



OpenPose



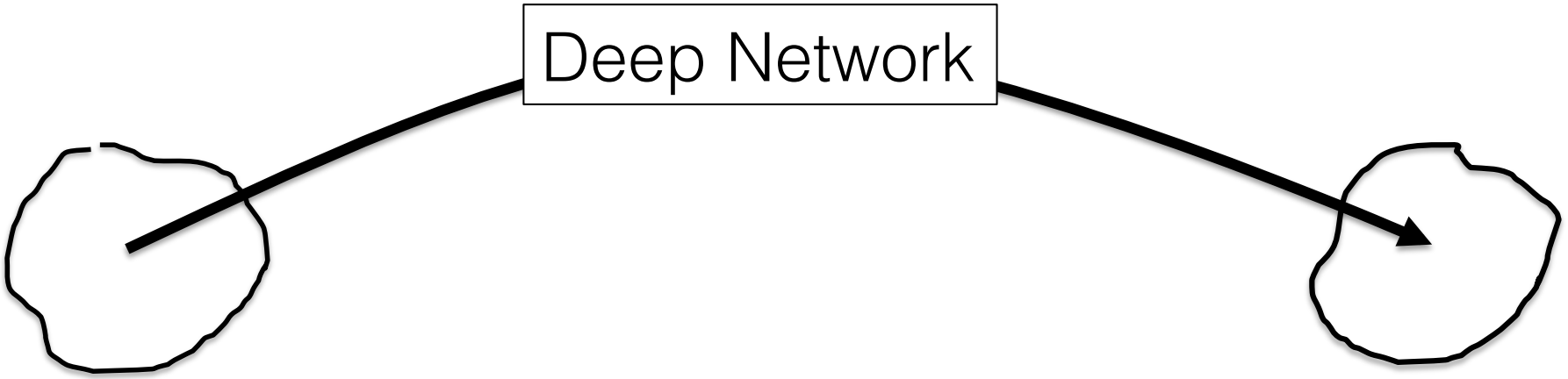
(b) Part Confidence Maps

(d) Bipartite Matching

(c) Part Affinity Fields

(e) Parsing Results

Using Deep Networks for Parameterizing Functions



Unsupervised Learning of Depth and Ego-Motion from Video [Zhou et al, 2017]

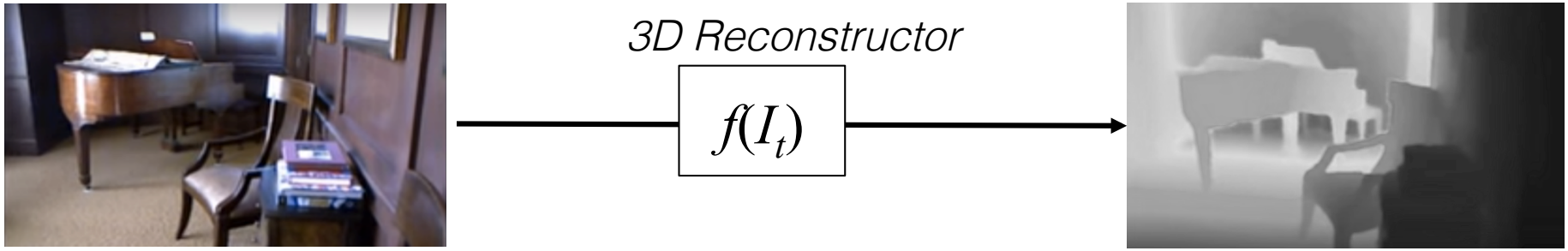


Image I

3D Reconstruction

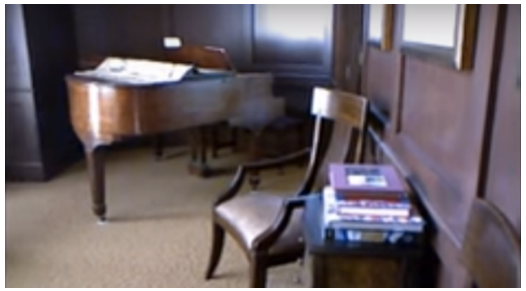


Image I_t

3D Reconstructor

$f(I_t)$

3D Reconstruction

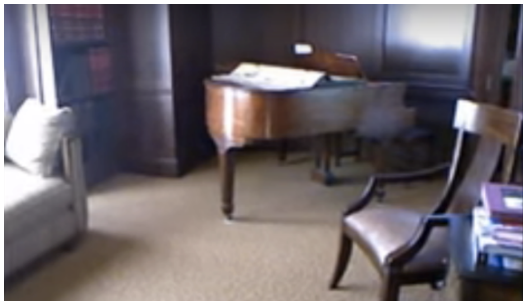


Image I_{t+1}

$f(I_{t+1})$

3D Reconstructor

3D Reconstruction

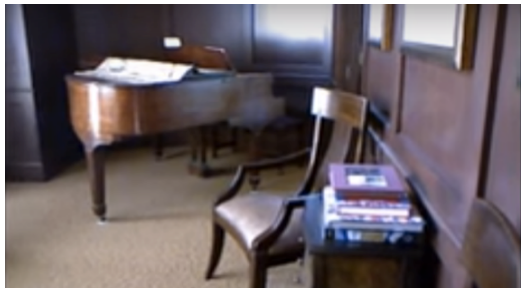


Image I_t

3D Reconstructor

$f(I_t)$

3D Reconstruction

Motion Estimator

$g(I_t, I_{t+1})$

motion between I_t and I_{t+1}

Motion

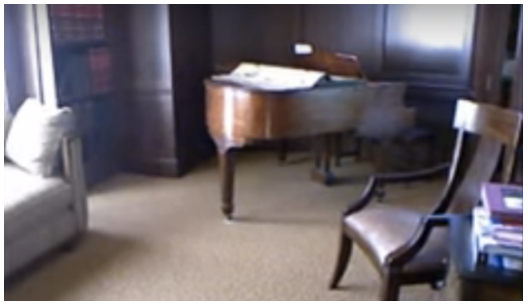


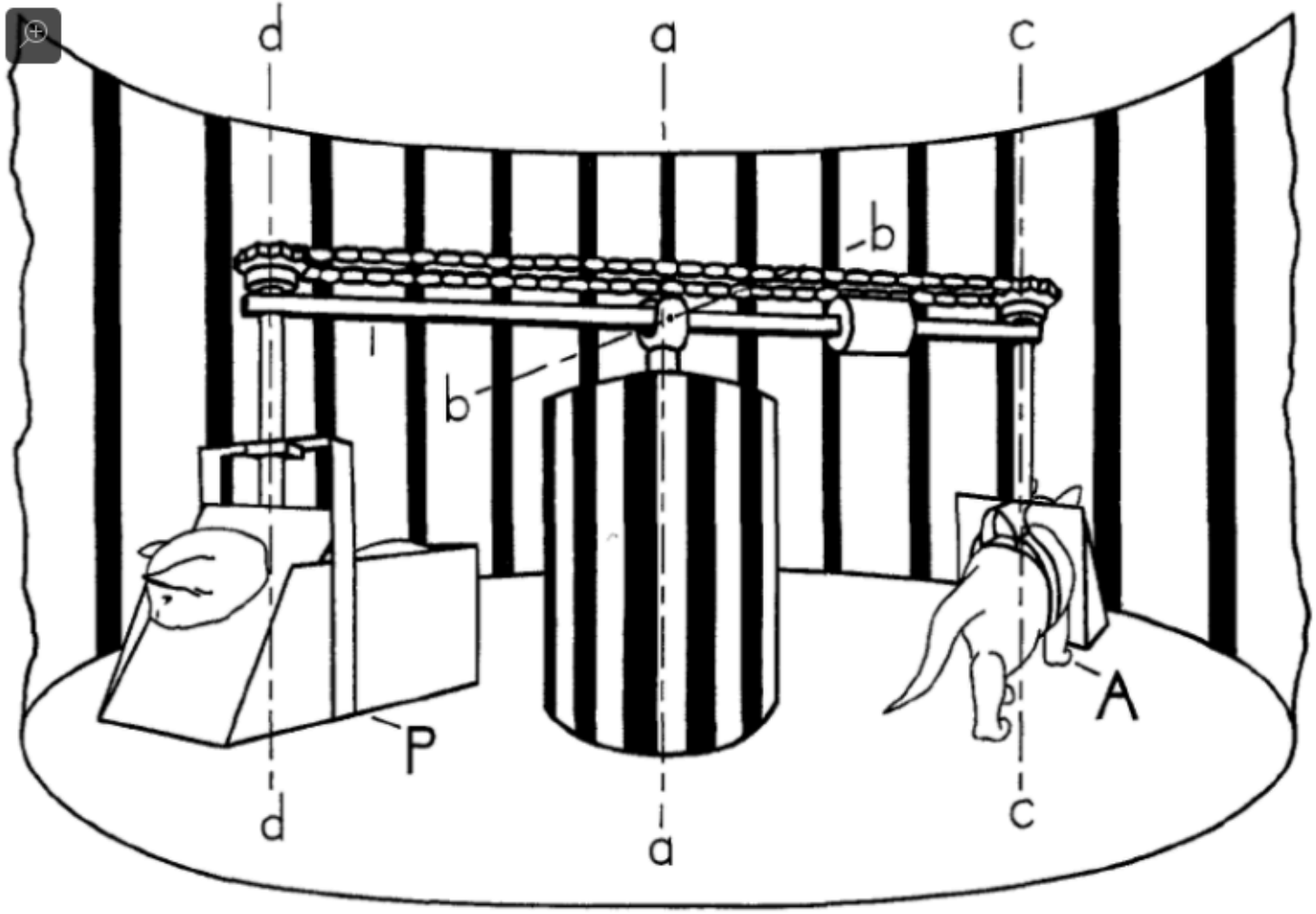
Image I_{t+1}

$f(I_{t+1})$

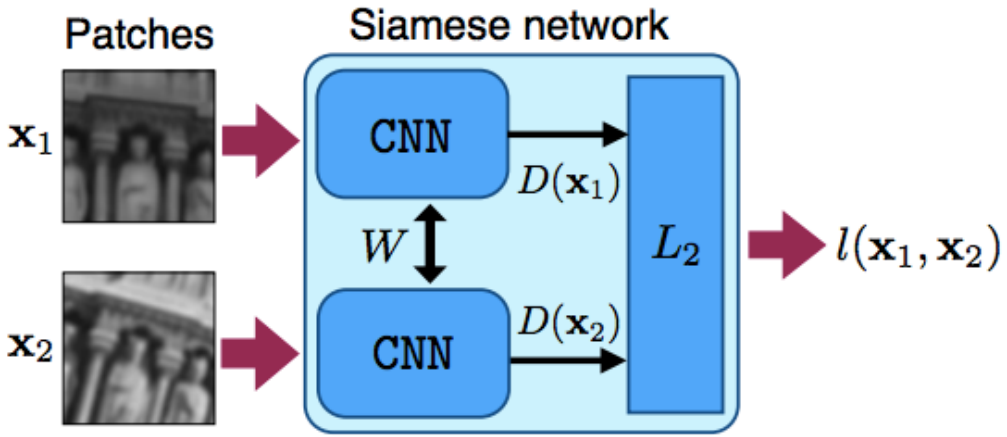
3D Reconstructor

3D Reconstruction

The Two-Kitten Experiment



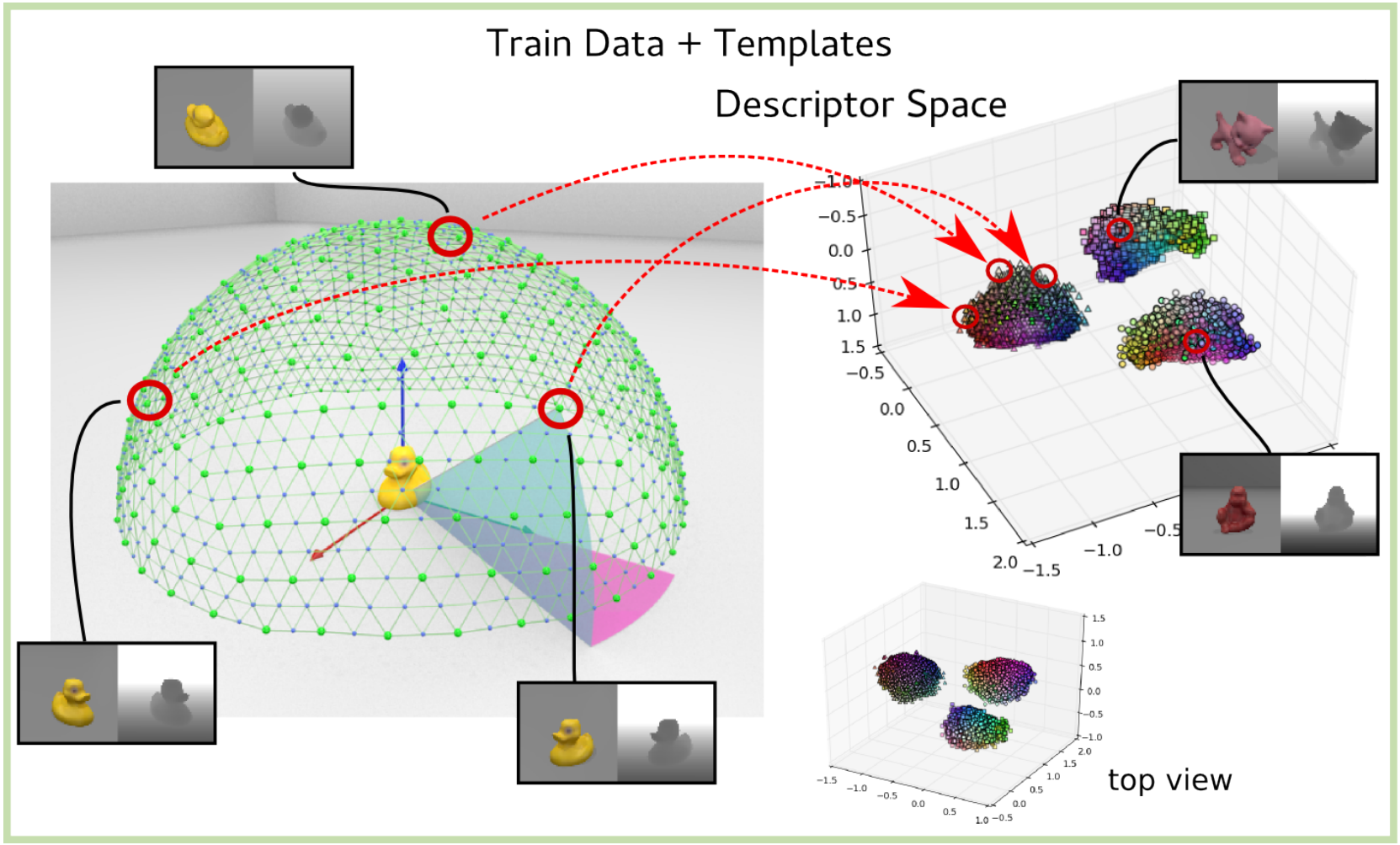
Siamese Networks



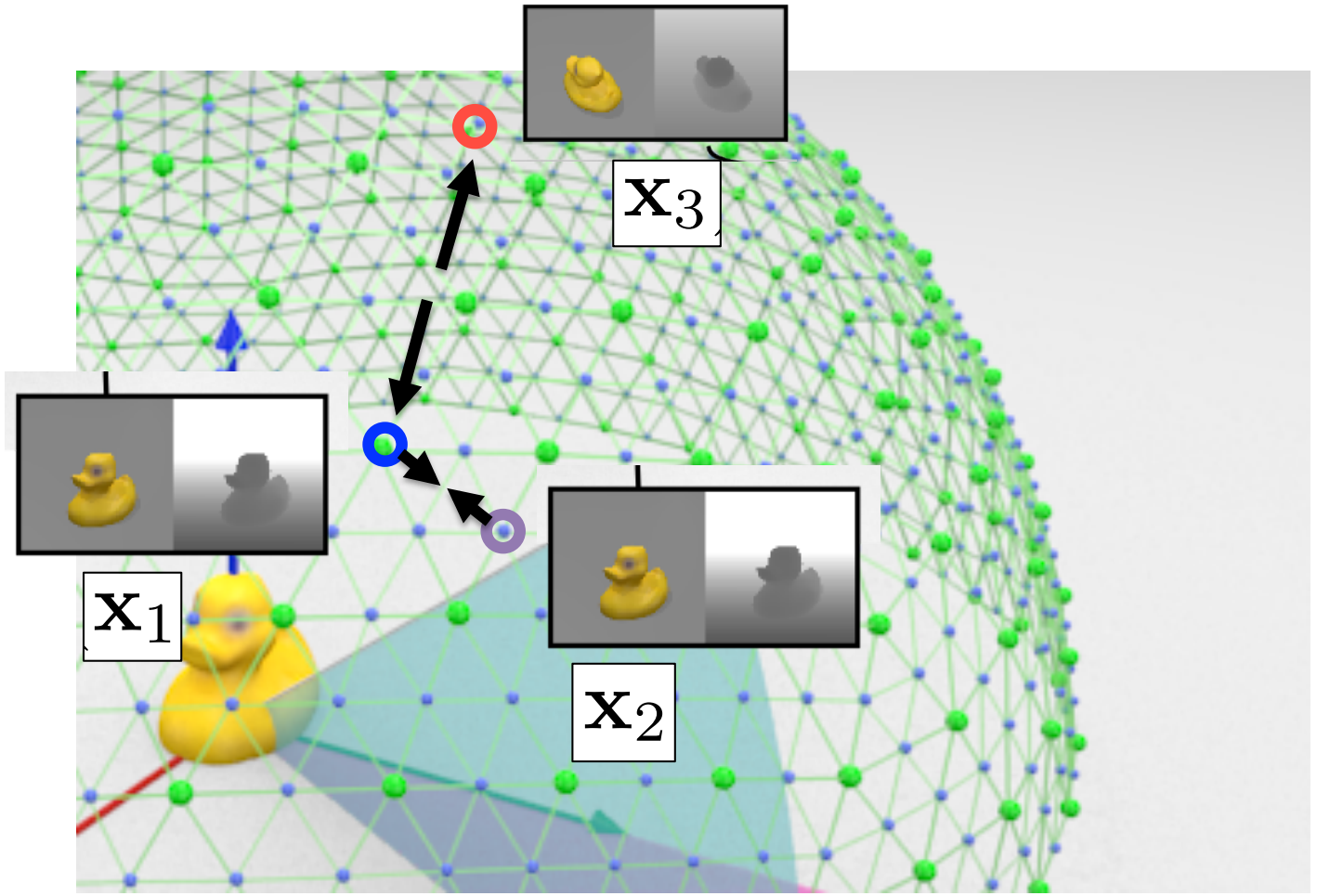
$$l(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \|D(\mathbf{x}_1) - D(\mathbf{x}_2)\|_2, & p_1 = p_2 \\ \max(0, C - \|D(\mathbf{x}_1) - D(\mathbf{x}_2)\|_2), & p_1 \neq p_2 \end{cases}$$

Images from Simo-Serra et al, ICCV'15

Learning Descriptors for Object Recognition and 3D Pose Estimation

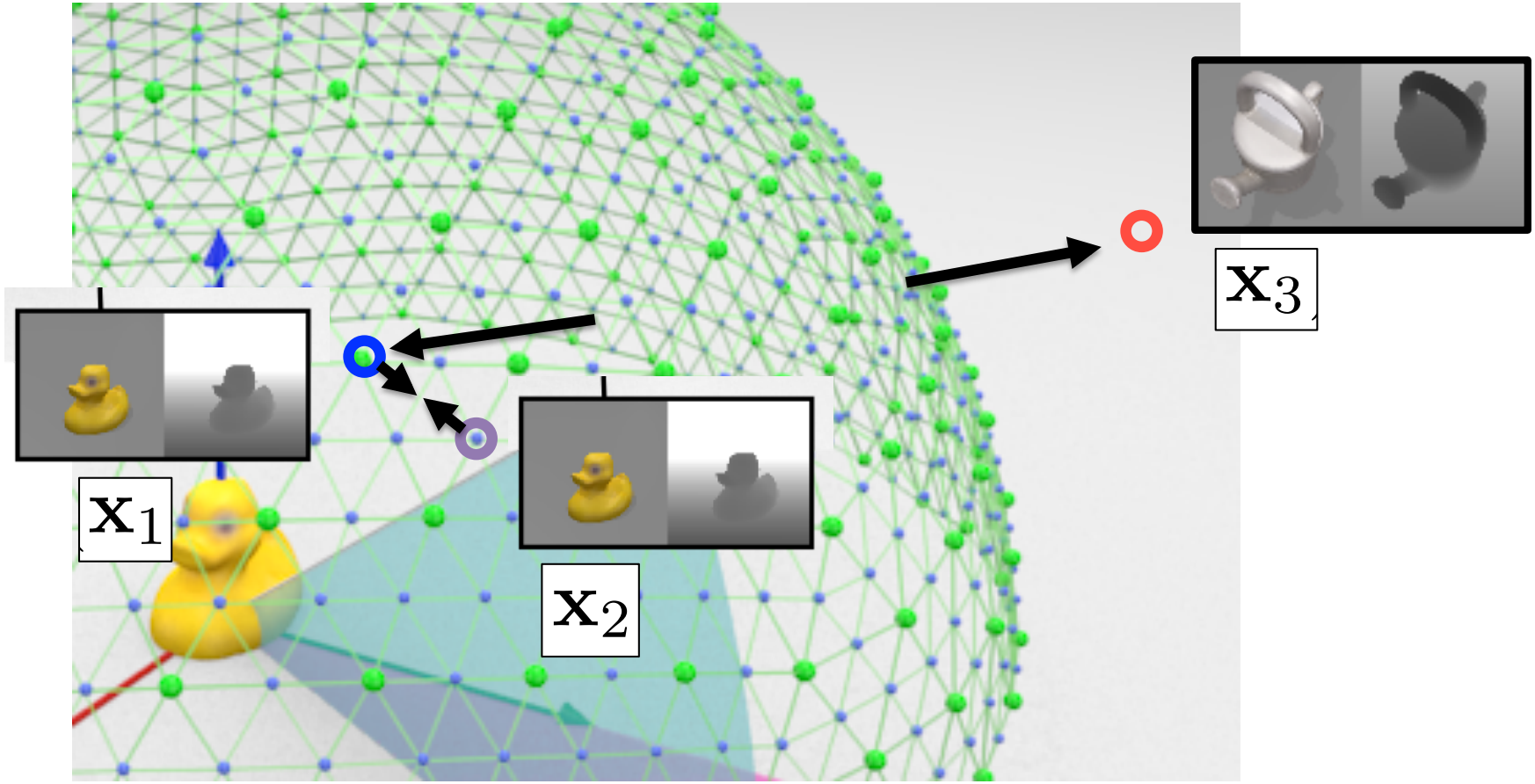


Triplet Constraints



$$l(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \max\left(0, 1 - \frac{\|D(\mathbf{x}_1) - D(\mathbf{x}_3)\|}{\|D(\mathbf{x}_1) - D(\mathbf{x}_2)\| + m}\right)$$

Triplet Constraints



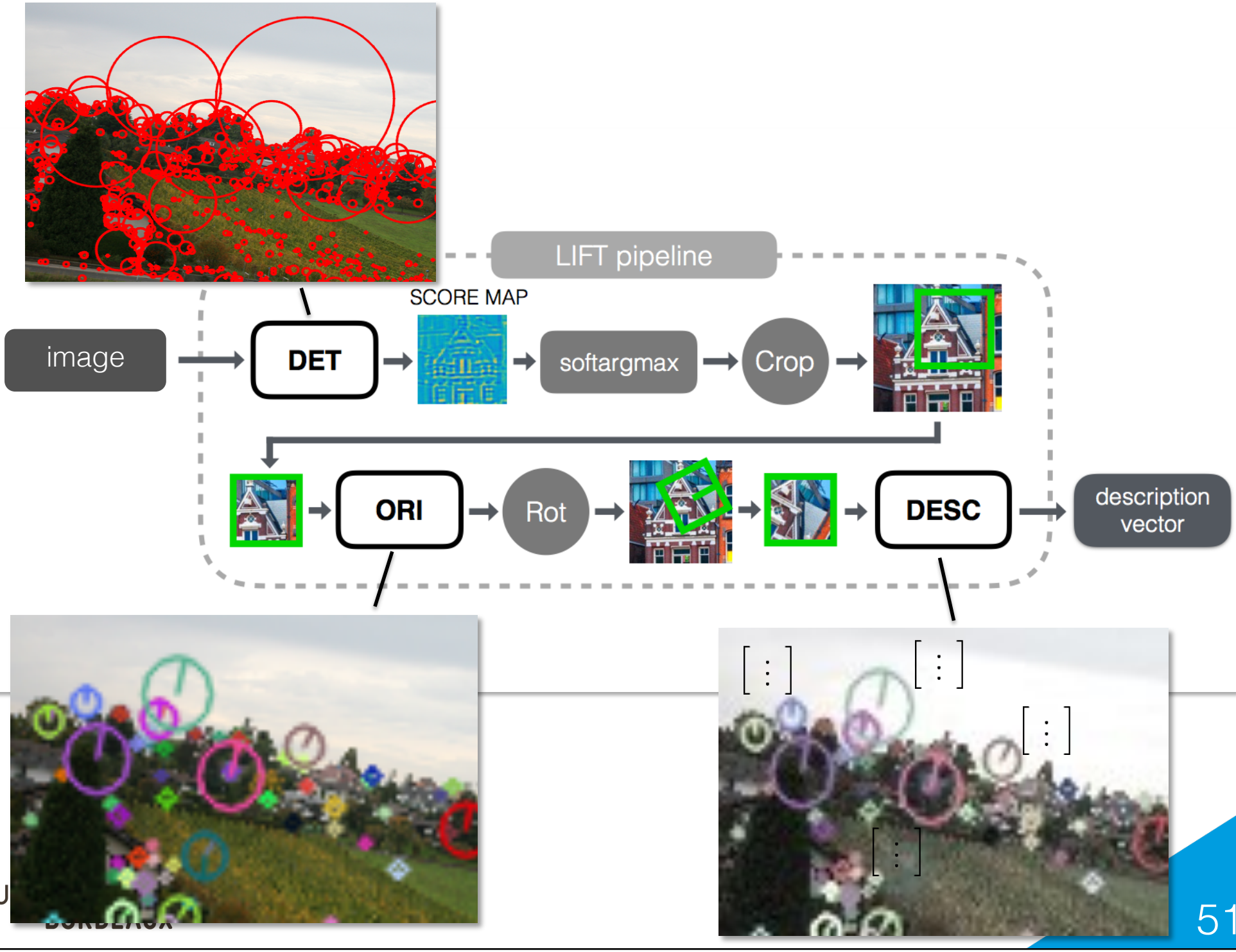
$$c(s_i, s_j, s_k) = \max\left(0, 1 - \frac{||f_w(x_i) - f_w(x_k)||_2}{||f_w(x_i) - f_w(x_j)||_2 + m}\right)$$

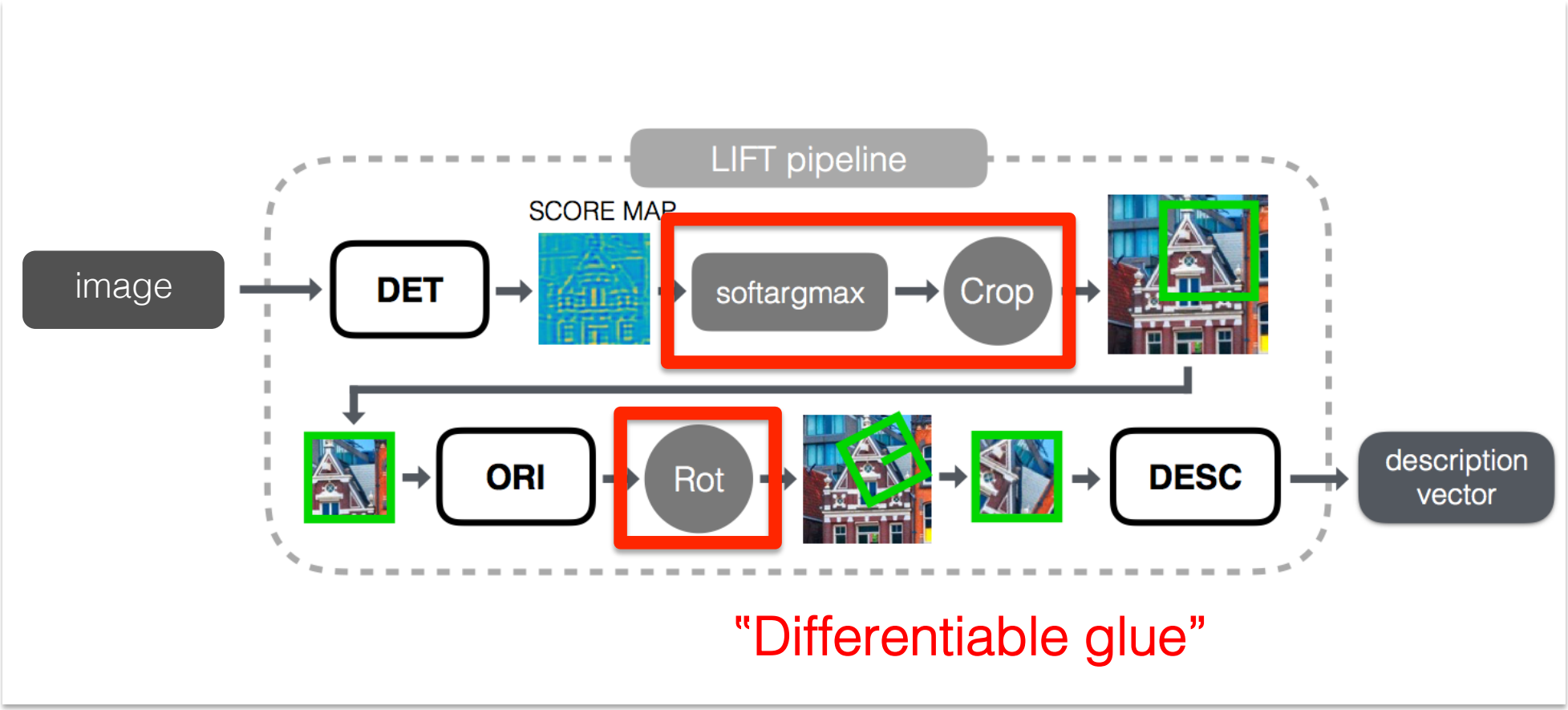
3-dim Descriptors for 3 Objects

Learning Descriptors for Object Recognition and 3D Pose Estimation

We train a CNN to map input images to low dimensional descriptors

Recognition and pose estimation can then be performed by efficient nearest neighbor retrieval


LIFT pipeline
image
DET
SCORE MAP
softargmax
Crop
ORI
Rot
DESC
description vector


LIFT pipeline
image
DET
SCORE MAP
softargmax
Crop
ORI
Rot
DESC
description vector
"Differentiable glue"

A Single, Global Cost Function

$$\min_{\{\mathsf{DET},\mathsf{ORI},\mathsf{DESC}\}} \sum_{\{(\mathbf{P},y)\}} \max(0, 1 - y \, \mathrm{softmax}(\mathsf{DET}(\mathbf{P})))^2 +$$

$$\sum_{(\mathbf{P}_1,\mathbf{P}_2)} \|\mathsf{DESC}\left(G(\mathbf{P}^1, \mathrm{softargmax}(\mathsf{DET}(\mathbf{P}^1))\right) - \mathsf{DESC}\left(G(\mathbf{P}^2, \mathrm{softargmax}(\mathsf{DET}(\mathbf{P}^2))\right)\|^2 +$$

$$\sum_{(\mathbf{P}_1,\mathbf{P}_3)} \max(0, 1 - \|\mathsf{DESC}\left(G(\mathbf{P}^1, \mathrm{softargmax}(\mathsf{DET}(\mathbf{P}^1))\right) - \mathsf{DESC}\left(G(\mathbf{P}^3, \mathrm{softargmax}(\mathsf{DET}(\mathbf{P}^3)))\right)\|^2)$$

$$G(\mathbf{P}, \mathbf{x}) = \mathrm{Rot}\left(\mathbf{P}, \mathbf{x}, \mathrm{angle}_{\mathsf{ORI}}(\mathrm{Crop}(\mathbf{P}, \mathbf{x}))\right)$$