Advanced School for Computing and Imaging

UNIVERSITY OF AMSTERDAM

Deep Learning Beyond Classification

Cees Snoek, University of Amsterdam

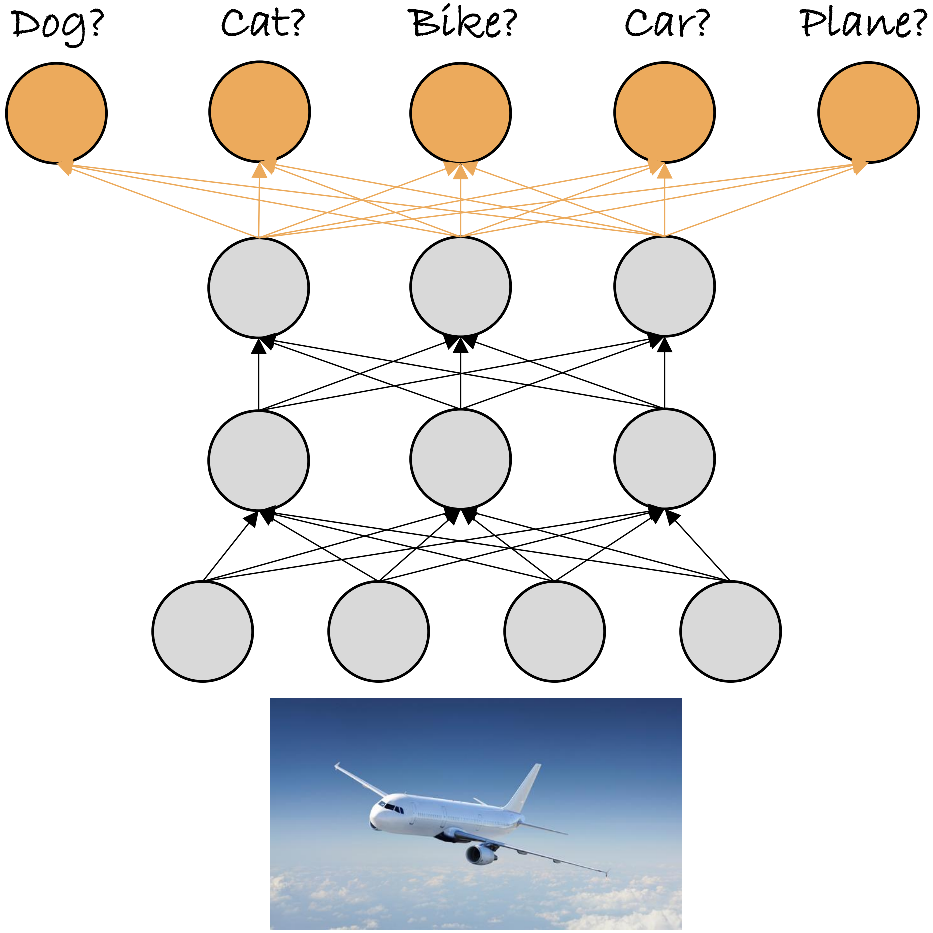
Efstratios Gavves, University of Amsterdam

With an invited tutorial by: Serge Belongie, University of Copenhagen

http://computervisionbylearning.info

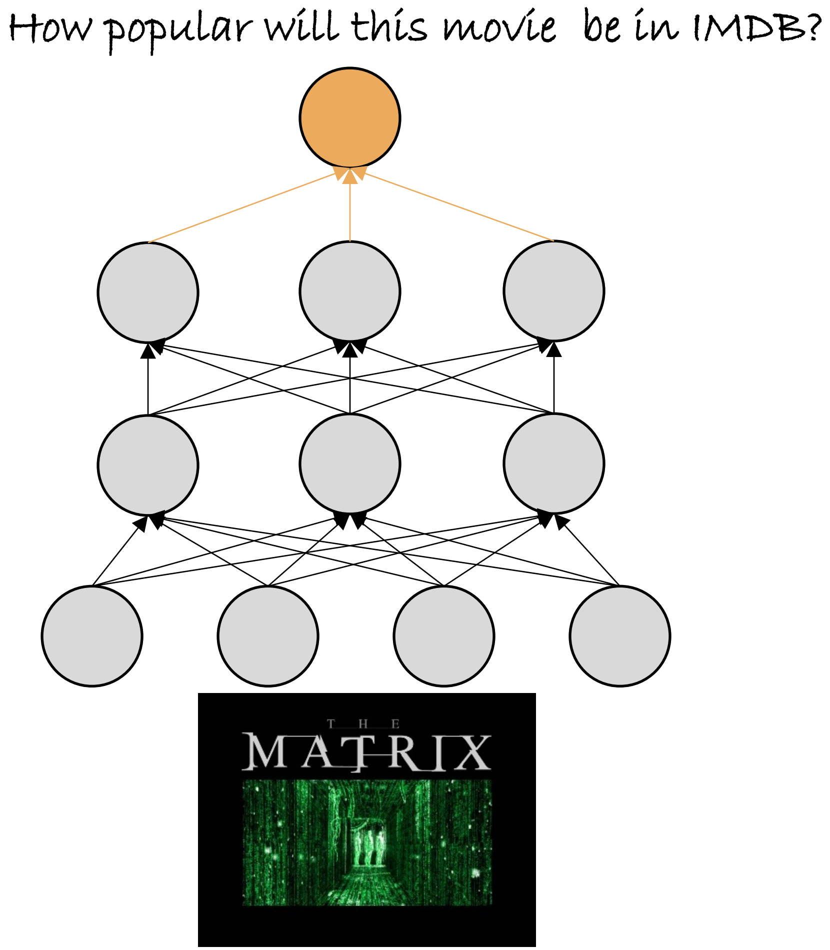
Standard inference

- N-way classification



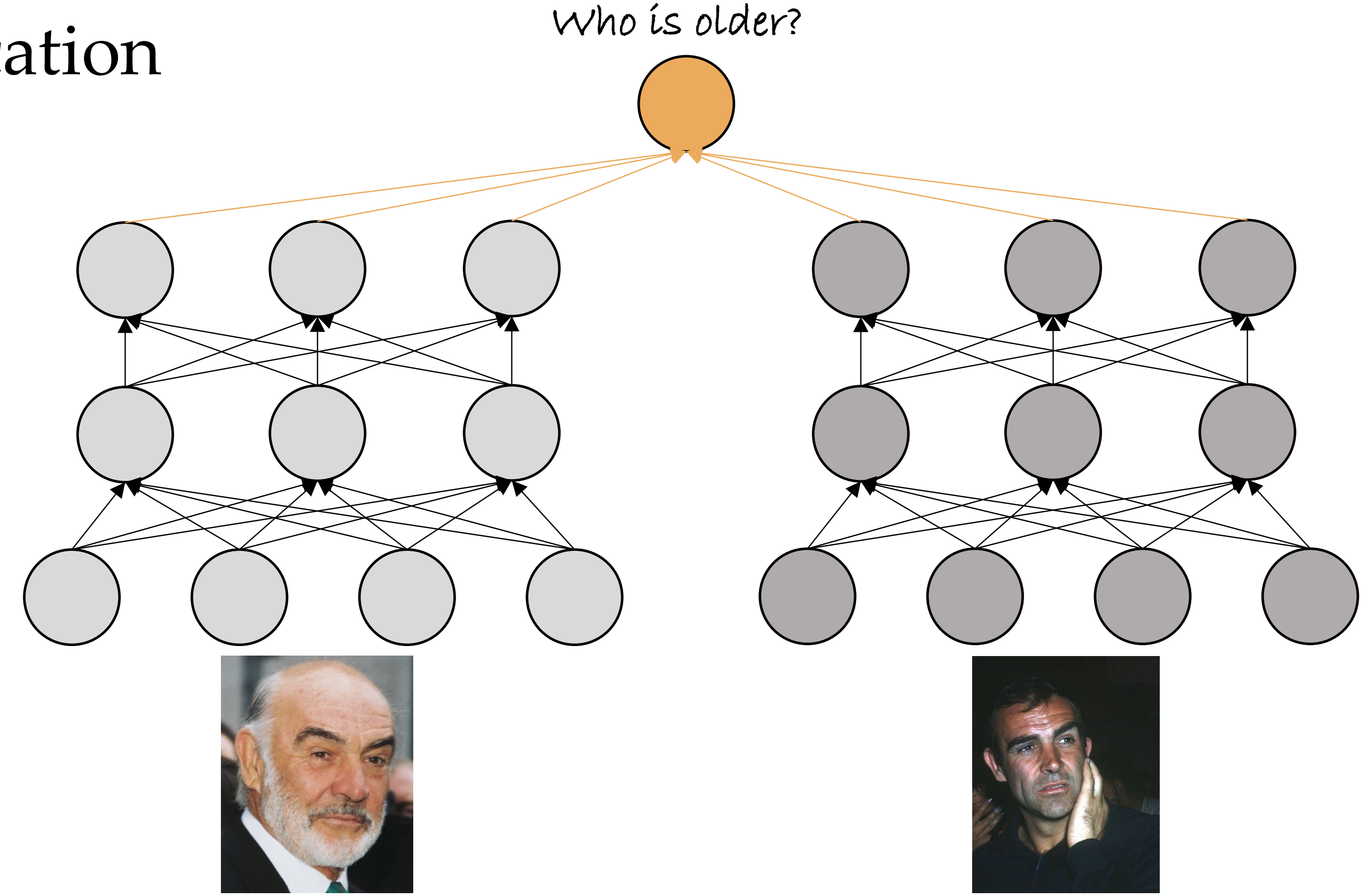
Standard inference

- N-way classification
- Regression



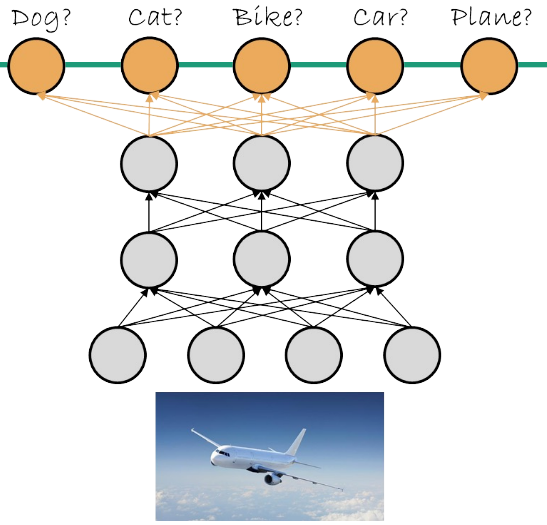
Standard inference

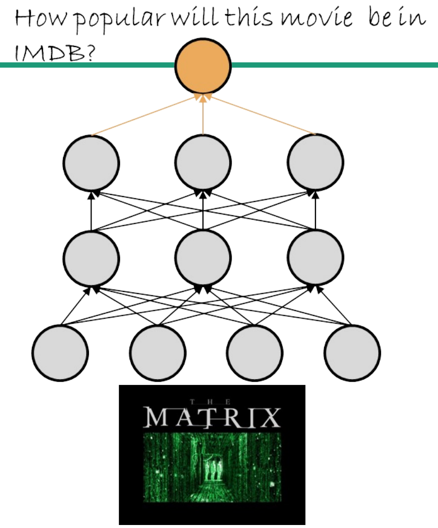
- N-way classification
- Regression
- Ranking
- …

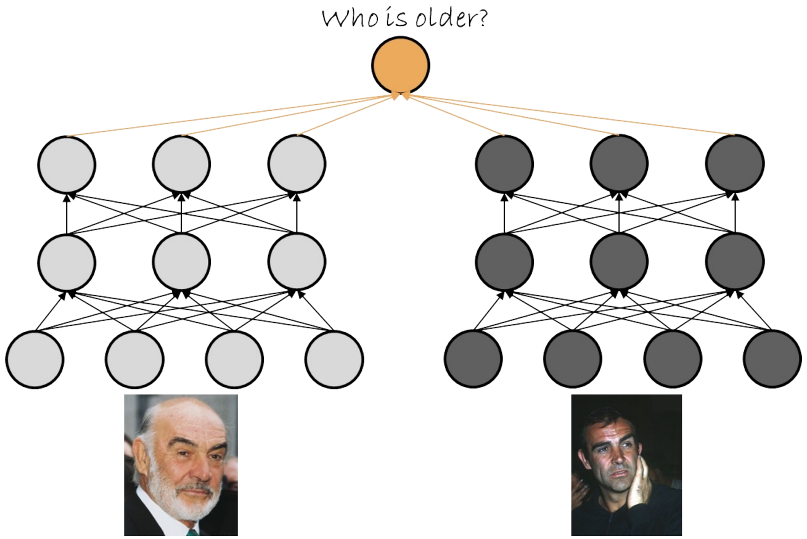


Quiz: What is common?

- N-way classification
- Regression
- Ranking
- …

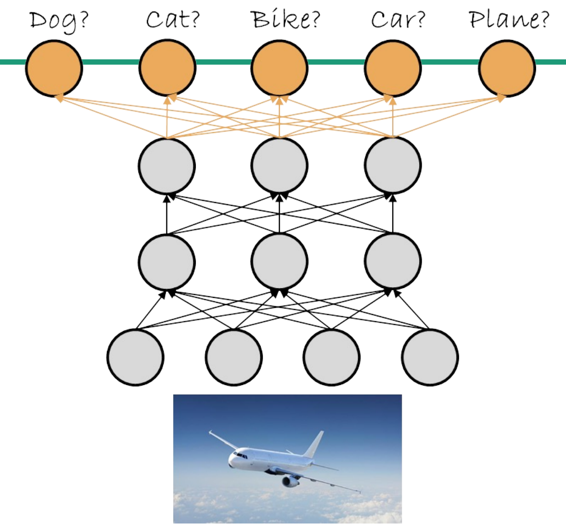


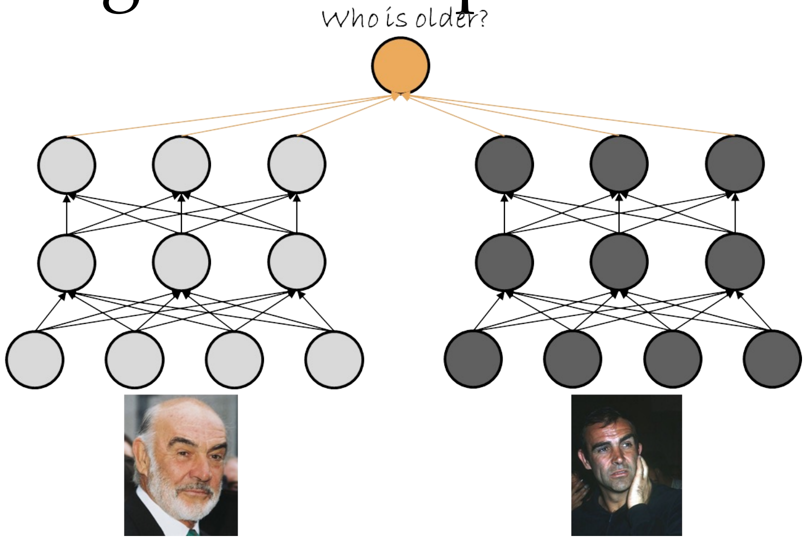


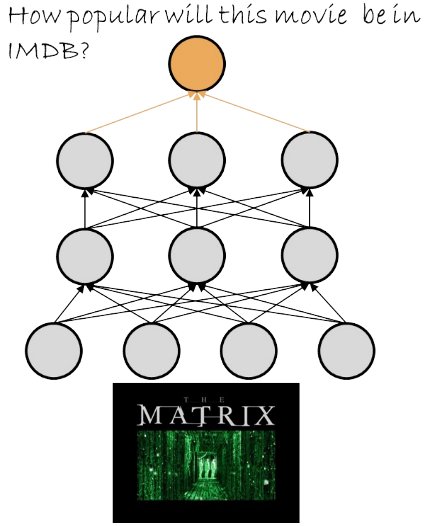


Quiz: What is common?

- They all make "single value" predictions
- Do all our machine learning tasks boil down to "single value" predictions?

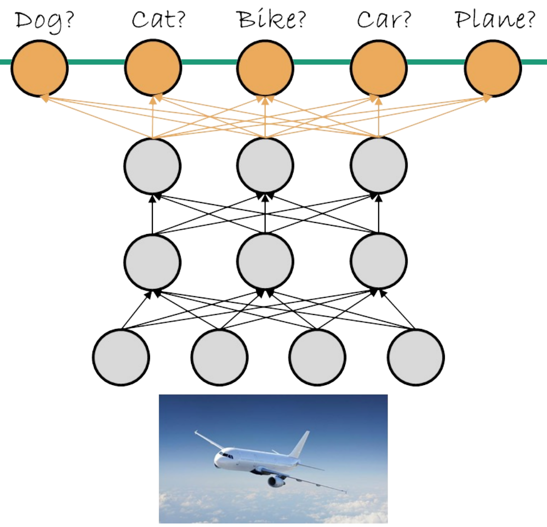


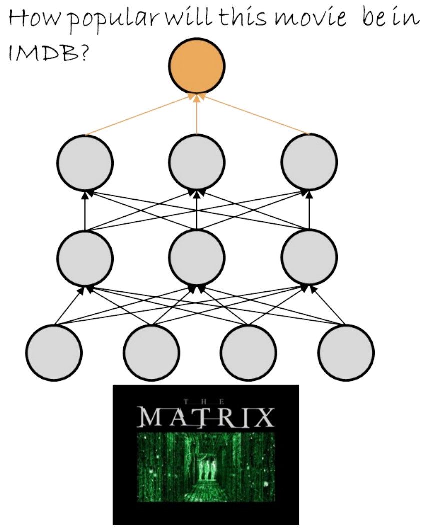




Beyond "single value" predictions?

- Do all our machine learning tasks boil to "single value" predictions?
- Are there tasks where outputs are somehow correlated?
- Is there some structure in this output correlations?
- How can we predict such structures?
  - ❑Structured prediction

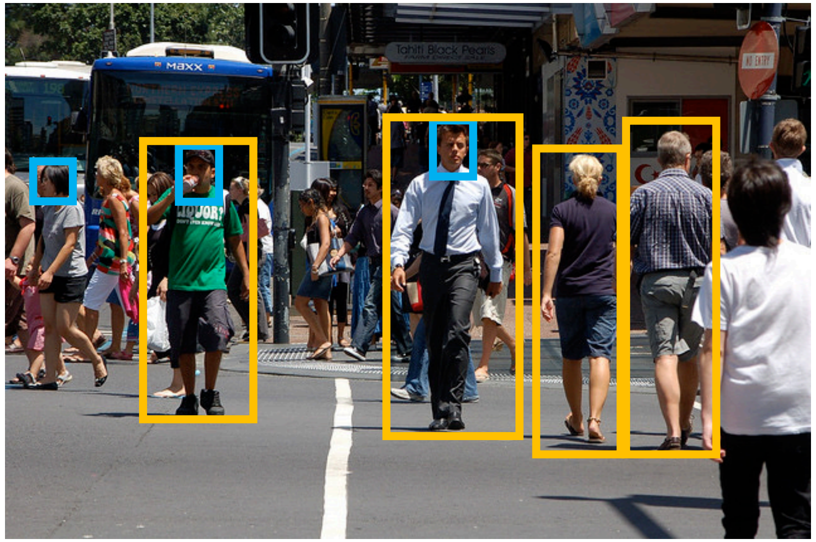




Quiz: Examples?

Object detection

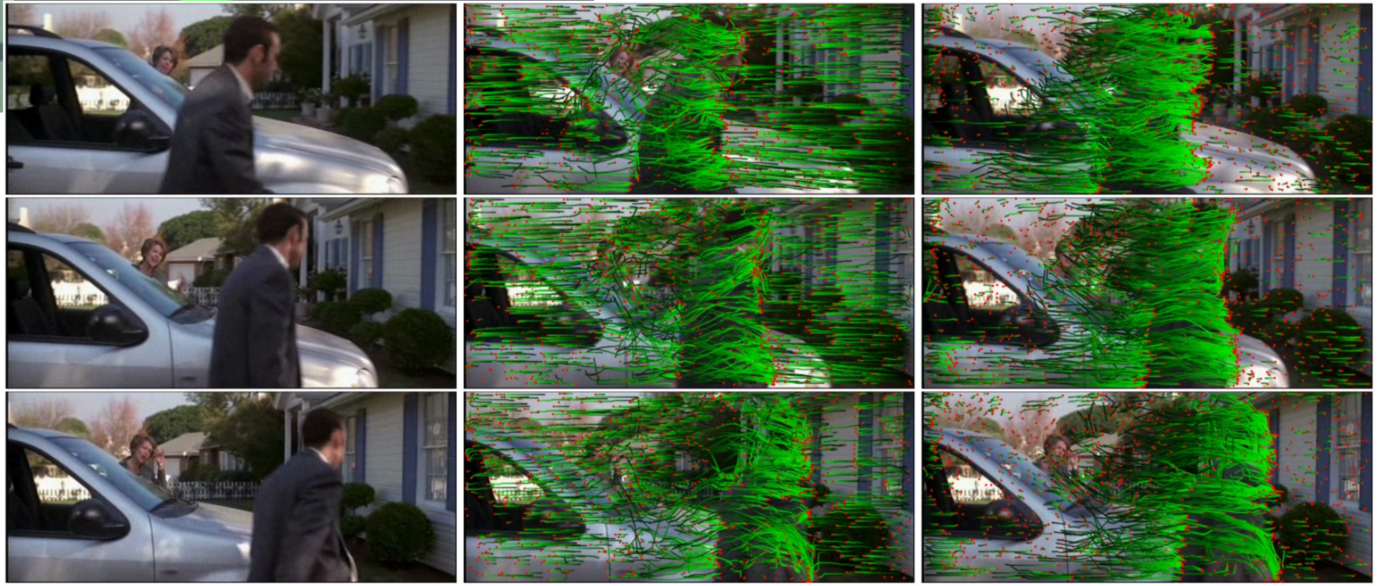
- Predict a box around an object
- Images
  - ❑Spatial location
  - ❑b(ounding) box
- Videos
  - ❑Spatio-temporal location
  - ❑bbox@t, bbox@t+1, …



Object segmentation

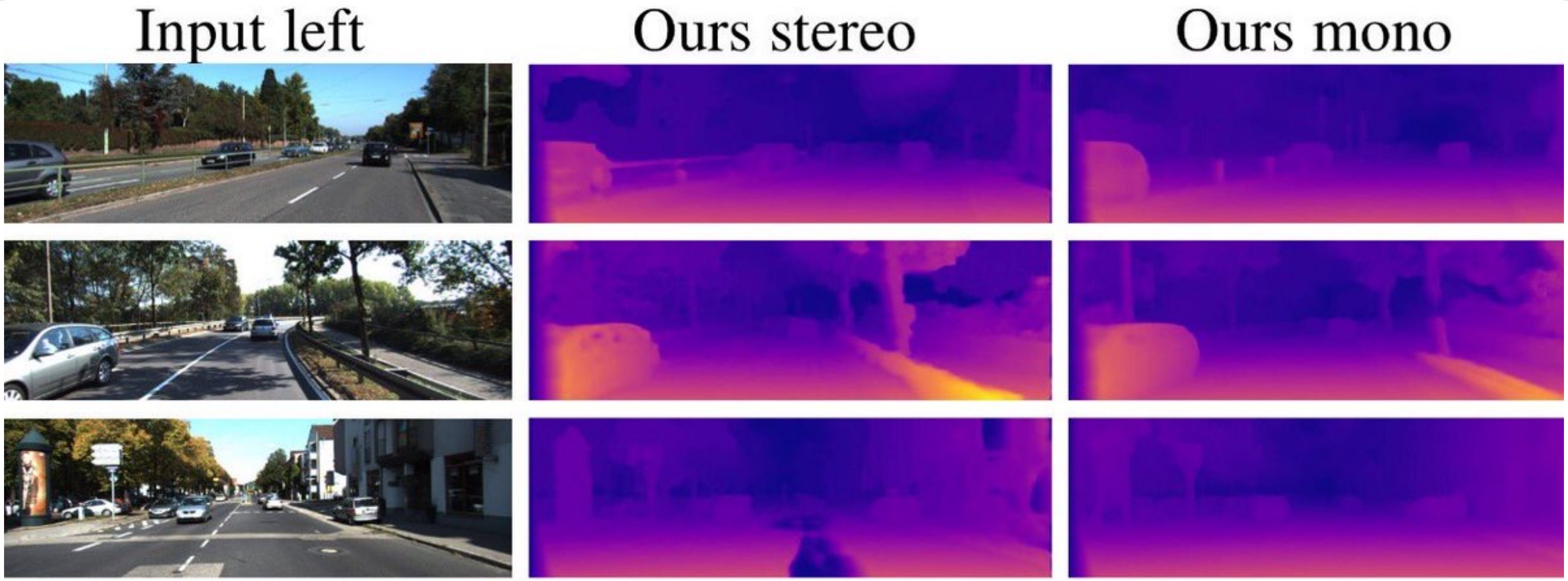
Image | Class map | Instance map | Part map | Part map (high level)

Optical flow & motion estimation



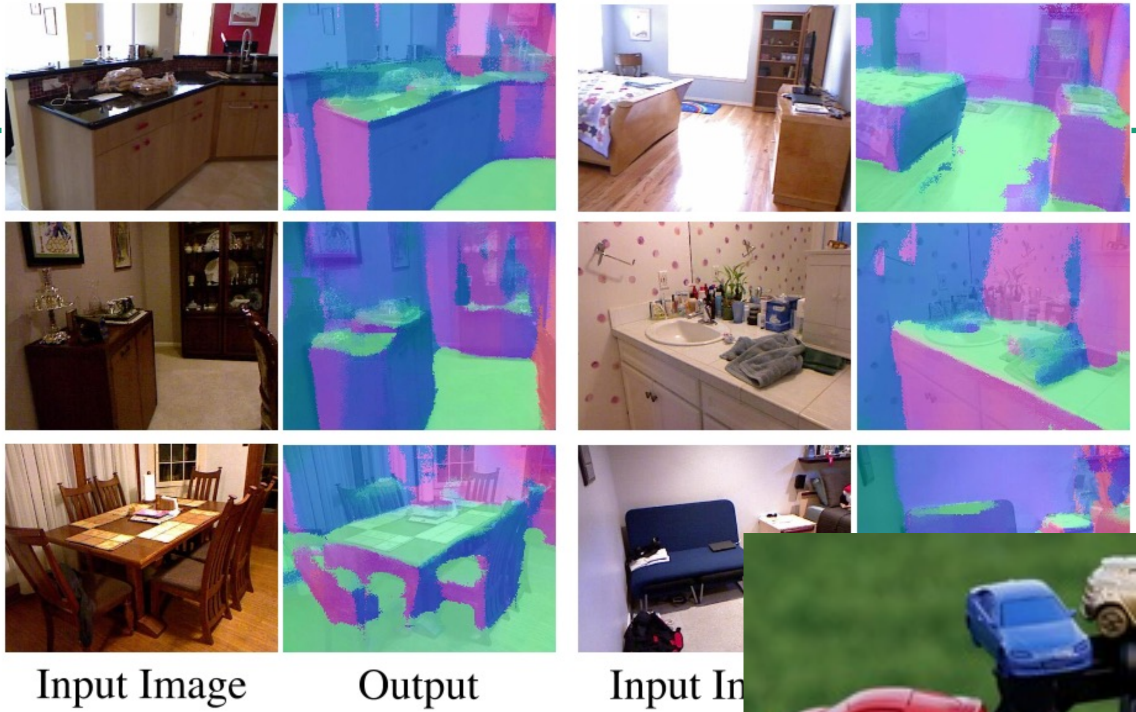
(a) Consecutive frames (b) Trajectories from Optical Flow (c) ω-trajectories

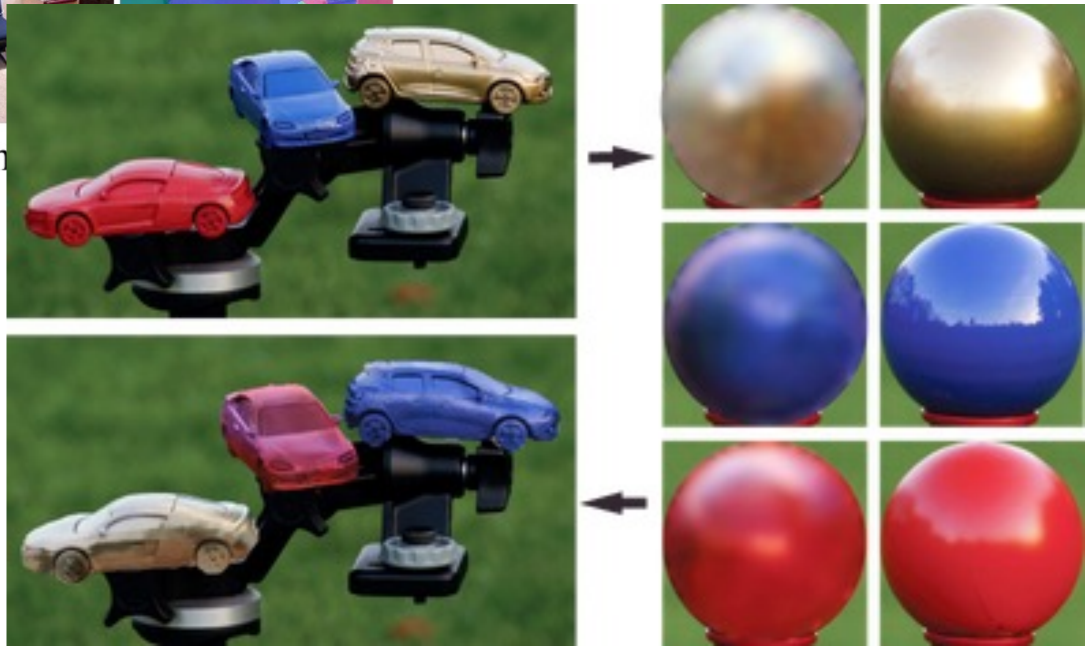
Depth estimation



Godard et al., Unsupervised Monocular Depth Estimation with Left-Right Consistency, 2016

Normals and reflectance estimation





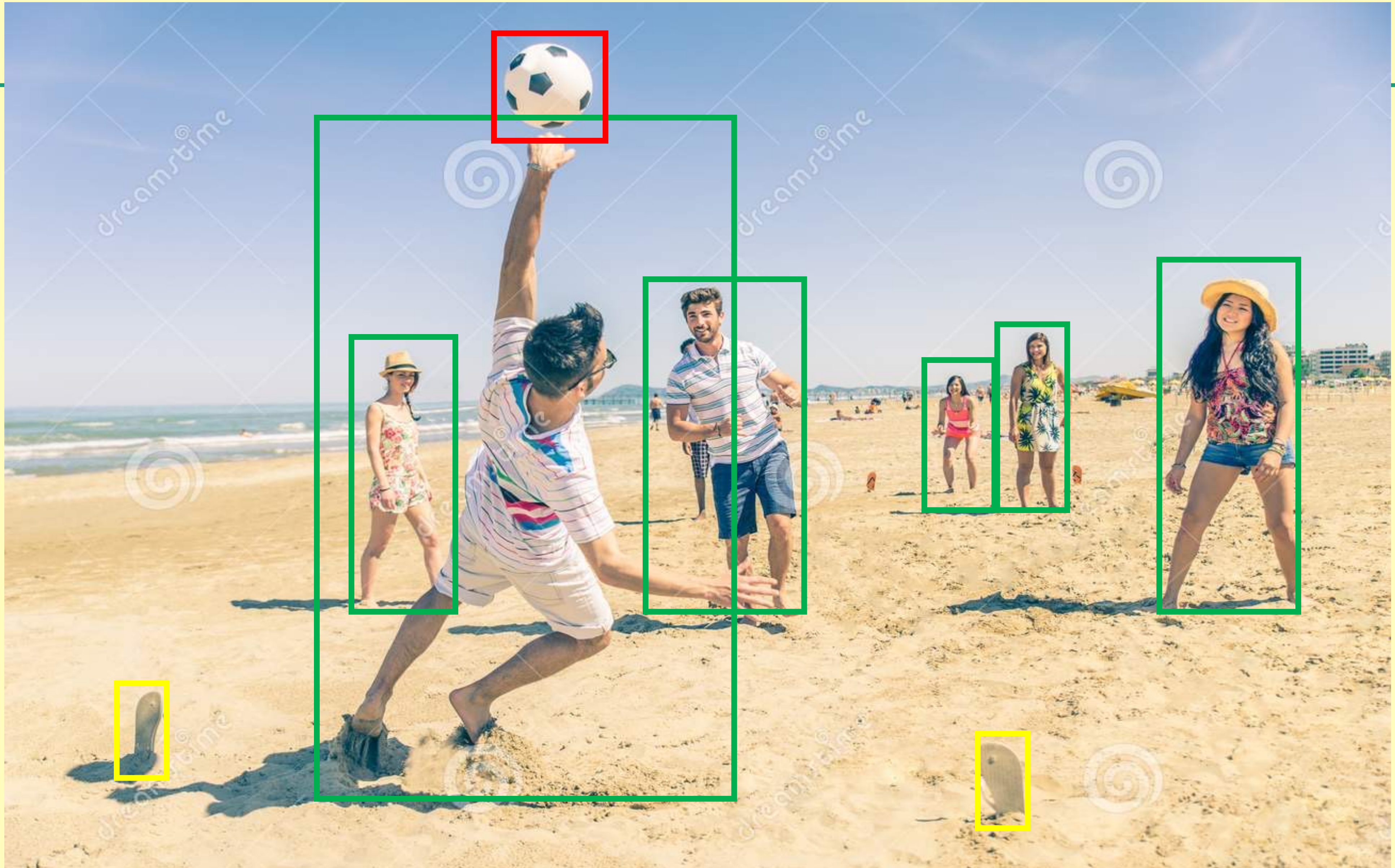
Structured prediction

- Prediction goes beyond asking for “single values”
- Outputs are complex and output dimensions correlated
- Output dimensions have latent structure
- Can we make deep networks to return **structured predictions?**

Structured prediction

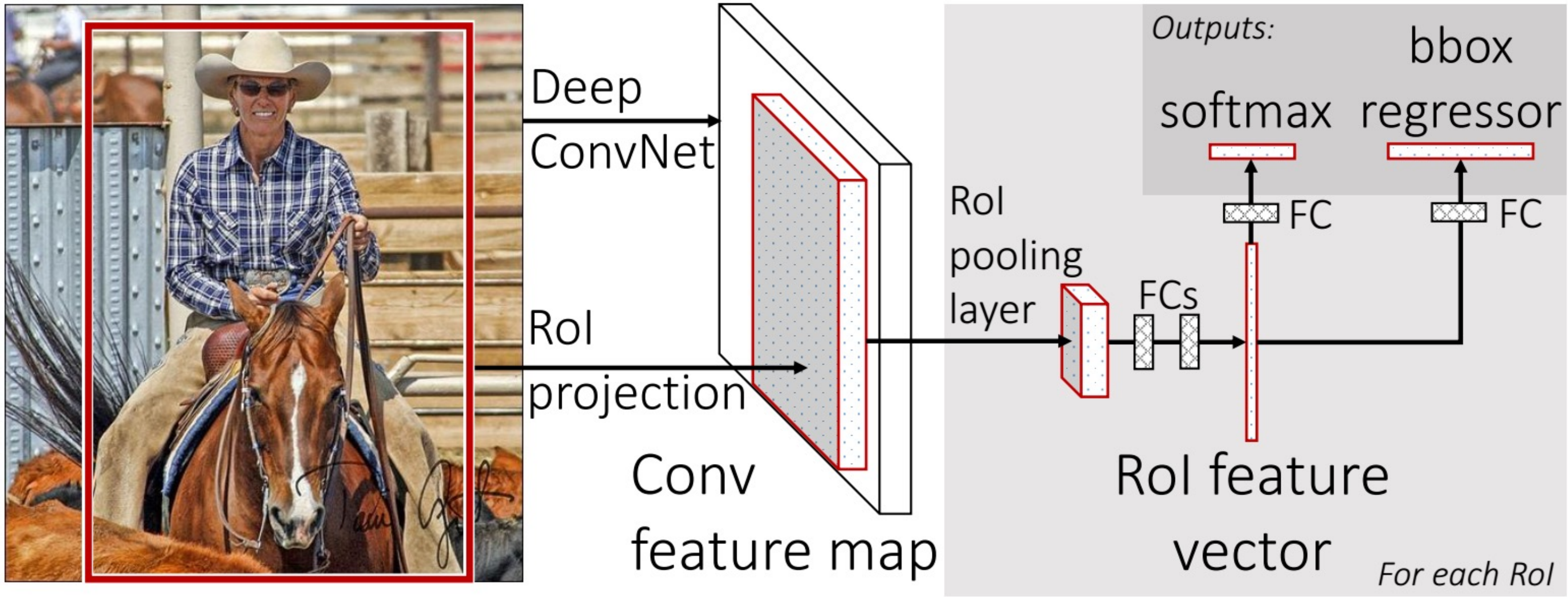
- Prediction goes beyond asking for “single values”
- Outputs are complex and output dimensions correlated
- Output dimensions have latent structure
- Can we make deep networks to return **structured predictions?**

Convnets for structured prediction



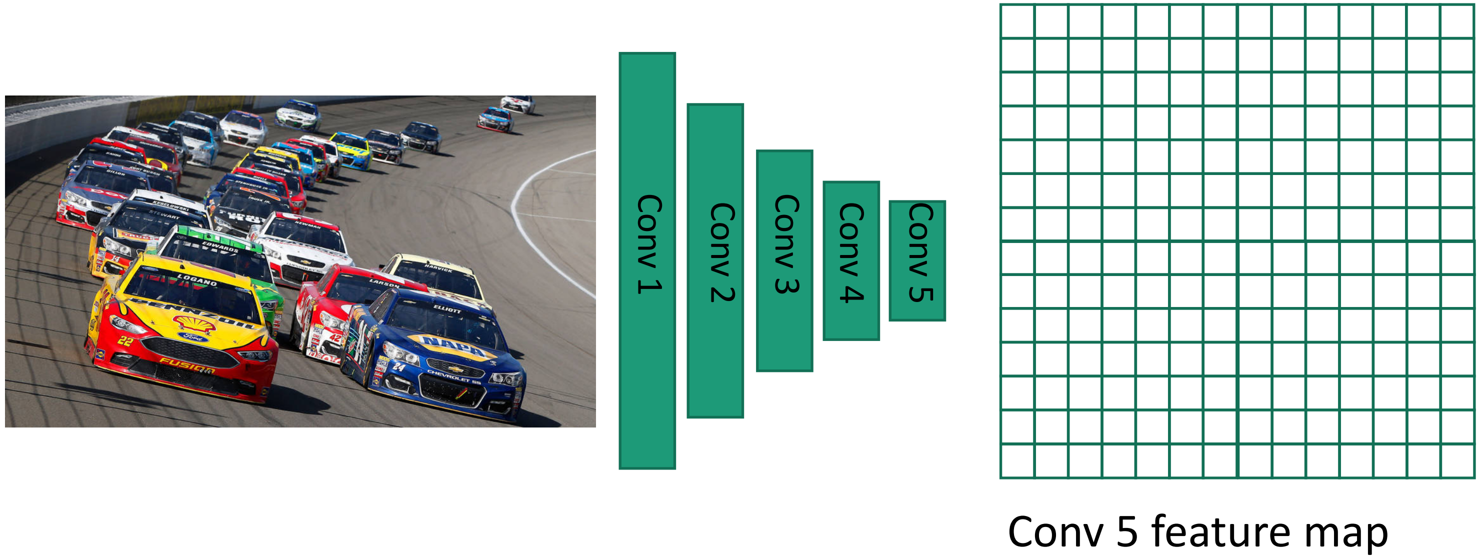
Sliding window on feature maps

- Selective Search Object Proposals [Uijlings2013]
- SPPnet [He2014]
- Fast R-CNN [Girshick2015]



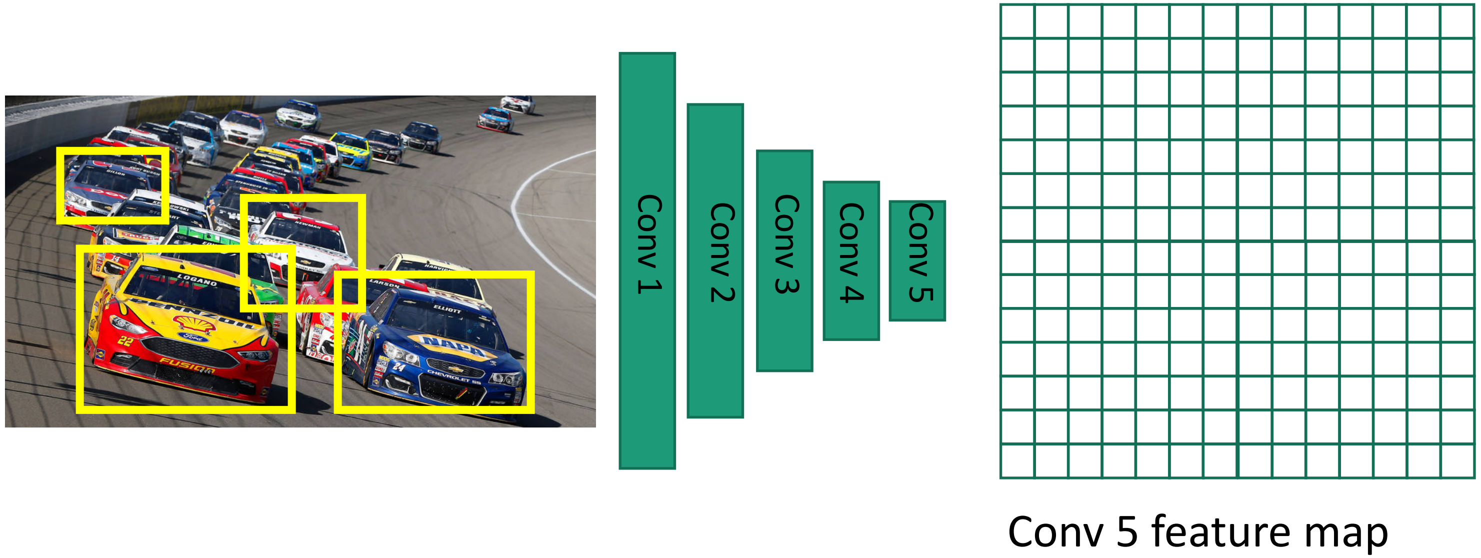
Fast R-CNN: Steps

- Process the whole image up to conv5



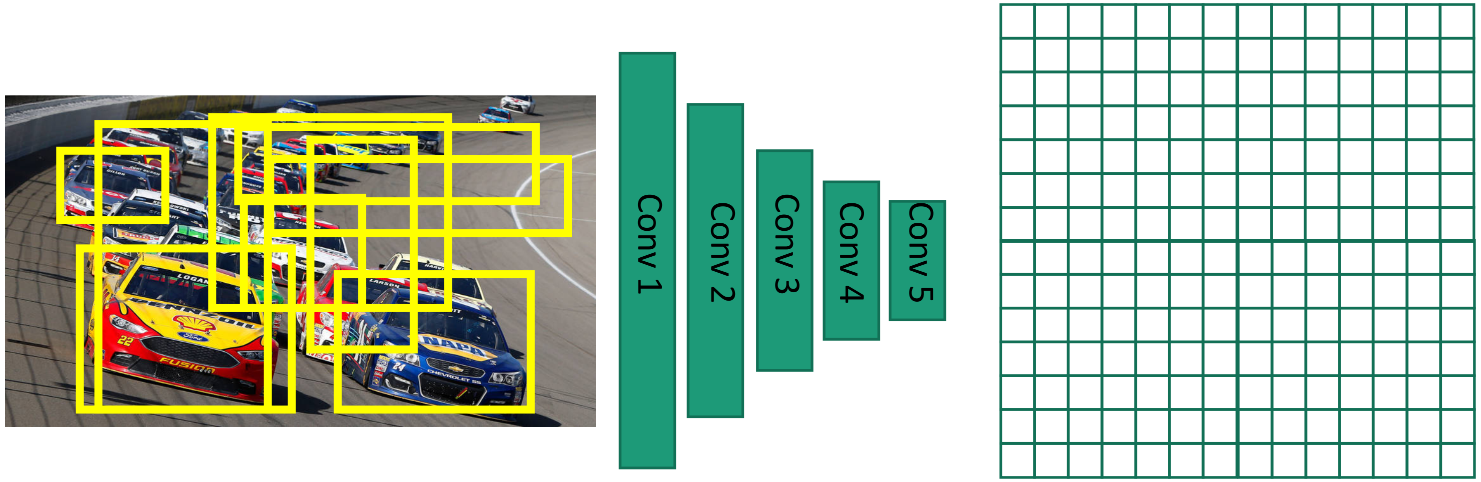
Fast R-CNN: Steps

- Process the whole image up to conv5
- Compute possible locations for objects



Fast R-CNN: Steps

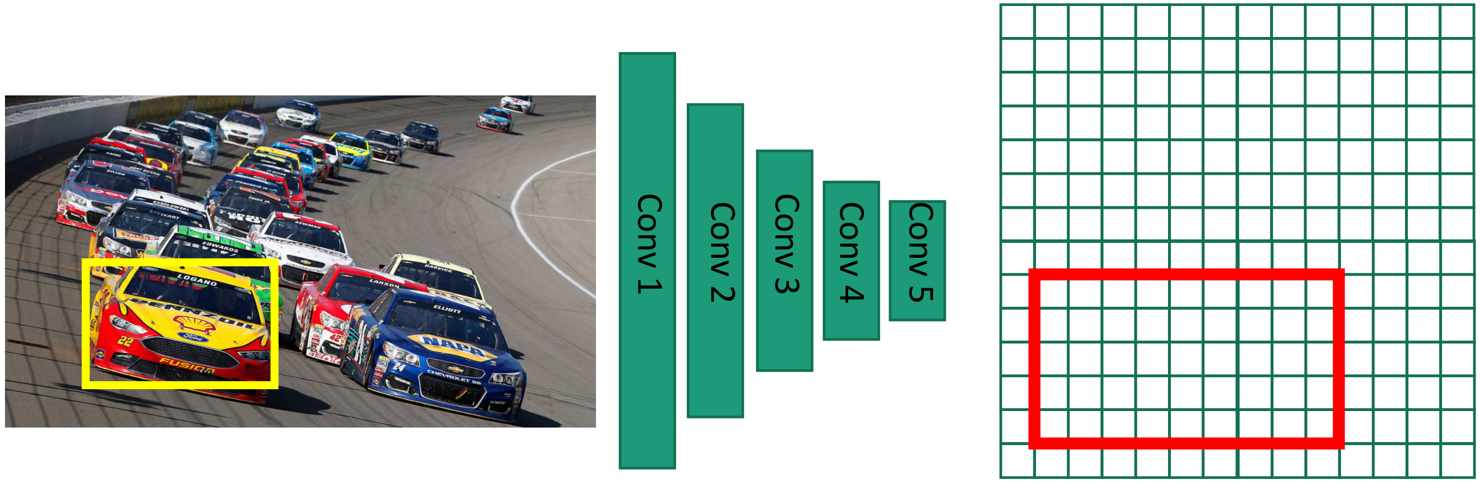
- Process the whole image up to conv5
- Compute possible locations for objects
  - ❑some correct, most wrong



Conv 5 feature map

Fast R-CNN: Steps

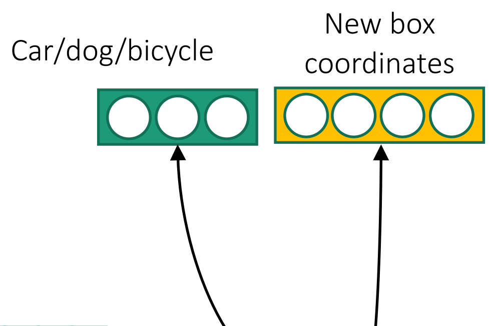
- Process the whole image up to conv5
- Compute possible locations for objects
  - ❑some correct, most wrong
- Given single location

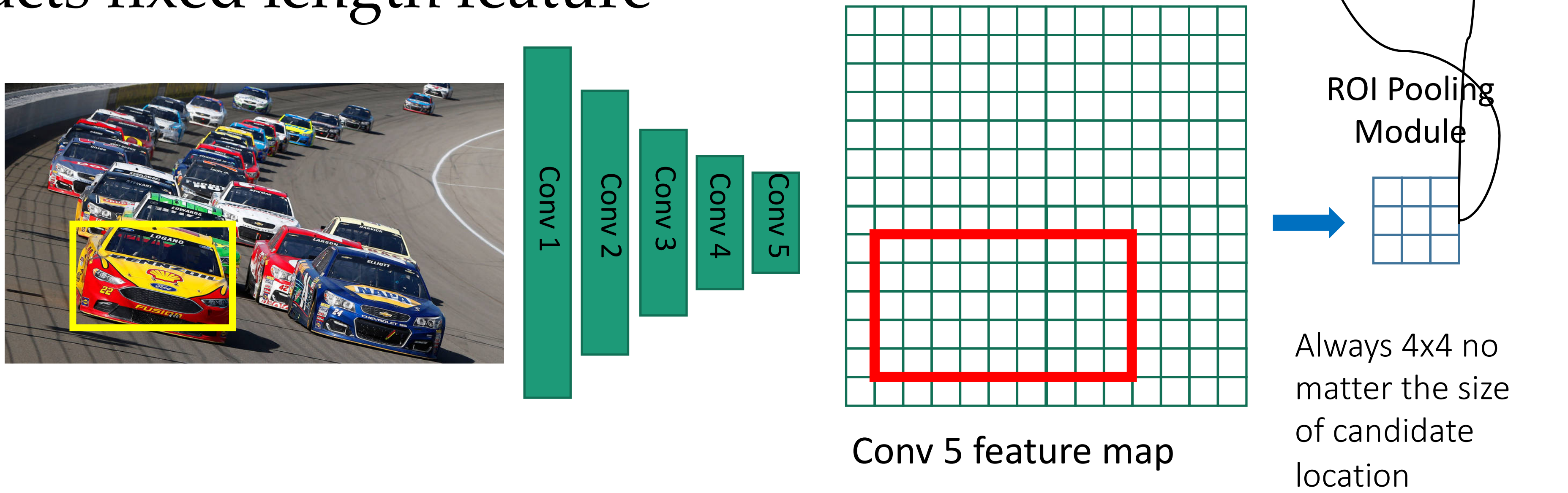


Conv 5 feature map

Fast R-CNN: Steps

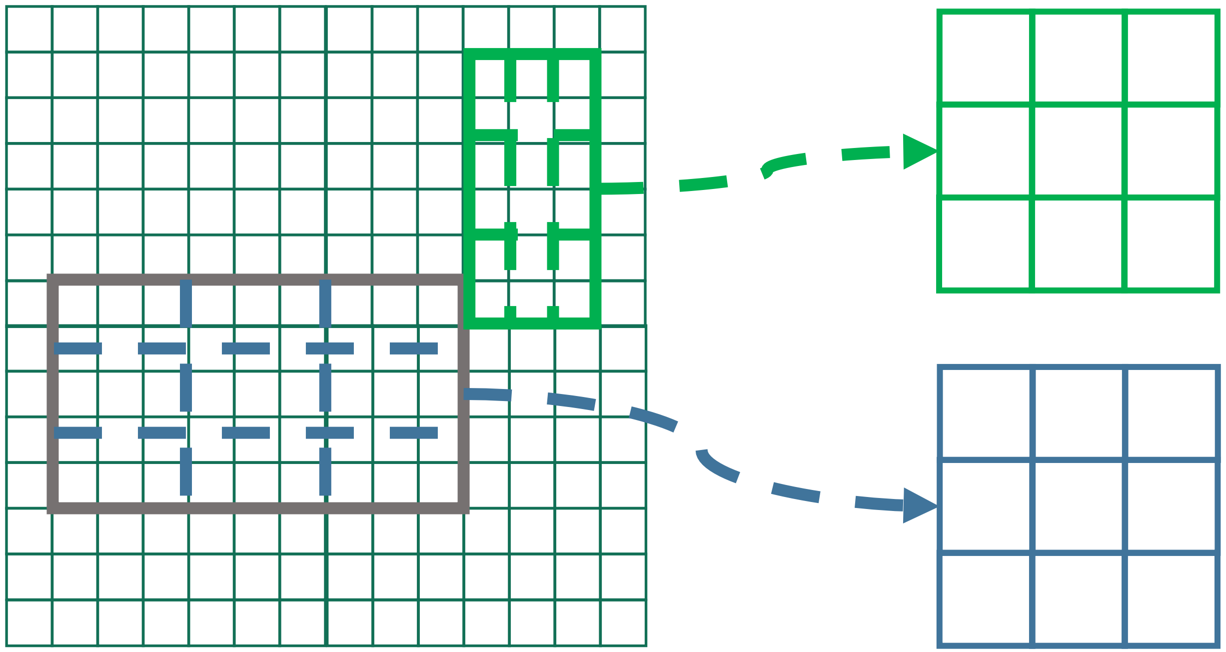
- Process the whole image up to conv5
- Compute possible locations for objects
  - ❑some correct, most wrong
- Given single location → ROI pooling module extracts fixed length feature





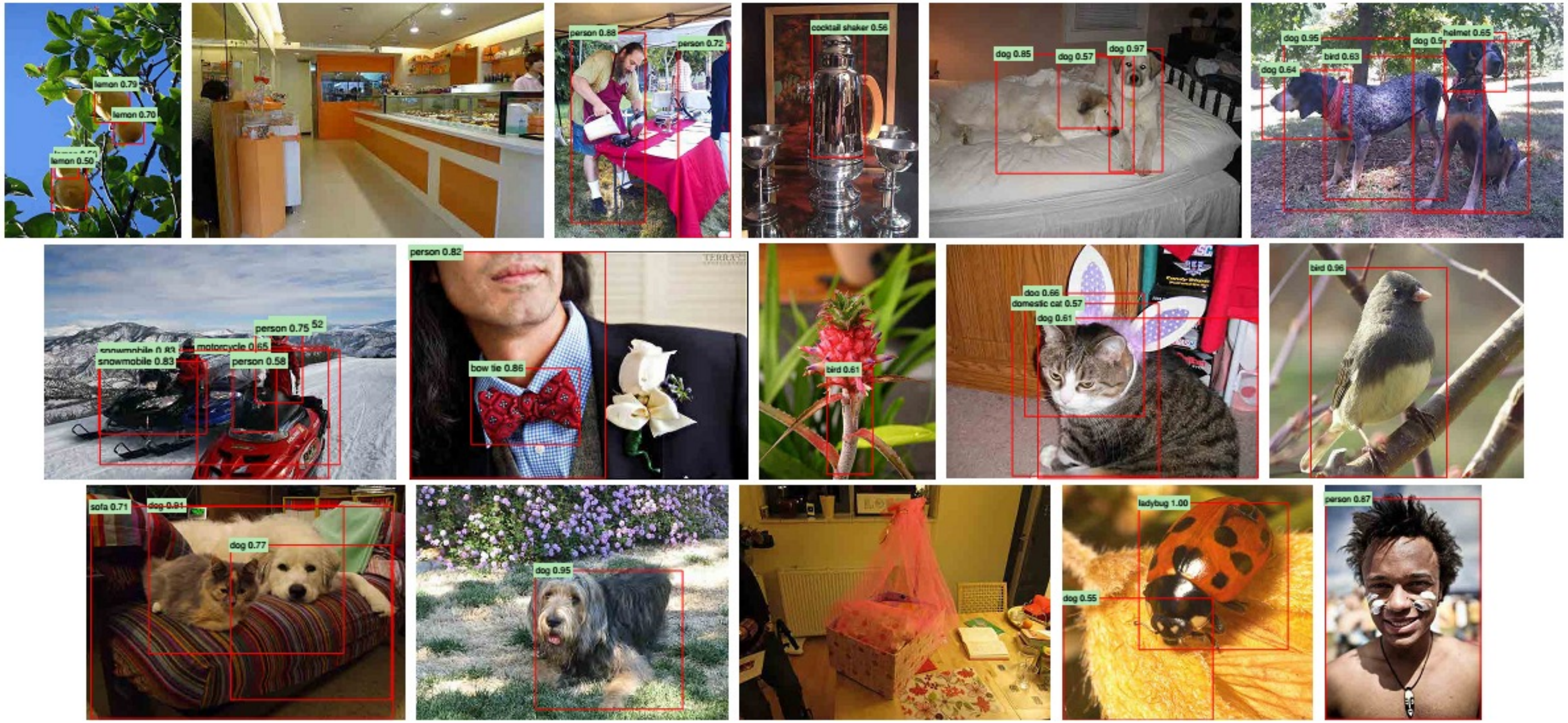
Fast R-CNN: Steps

- Divide feature map in *TxT* cells
- Cell size depends on size of the candidate location



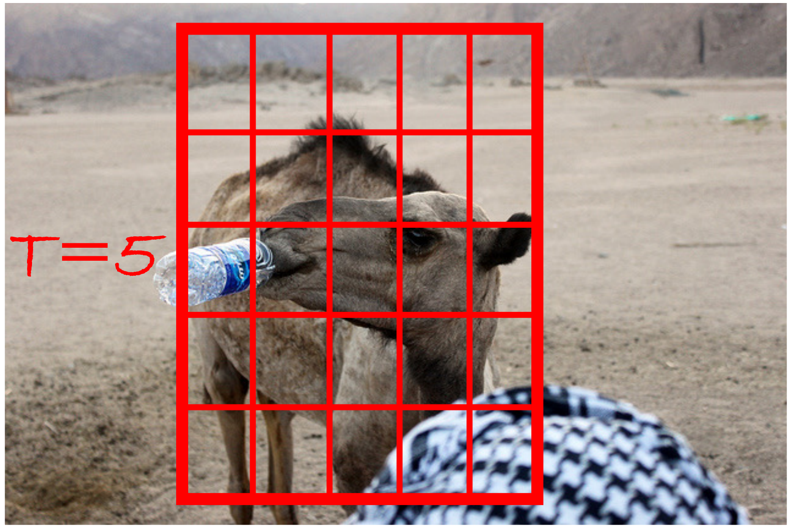
Always 3x3 no matter the size of candidate location

Some results



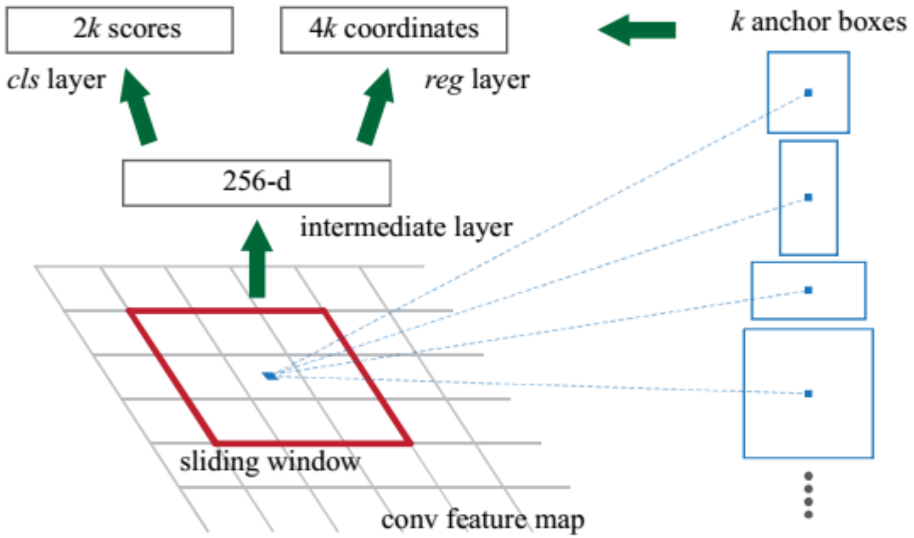
Fast R-CNN

- Reuse convolutions for different candidate boxes
  - ❑Compute feature maps only once
- Region-of-Interest pooling
  - ❑Define stride relatively → box width divided by predefined number of "poolings" T
  - ❑Fixed length vector
- End-to-end training!
- (Very) Accurate object detection
- (Very) Faster
  - ❑Less than a second per image
- External box proposals needed



Faster R-CNN [Girshick2016]

- Fast R-CNN: external candidate locations
- Faster R-CNN: deep network box proposals
- Slide the feature map: *k* anchor boxes per slide



Region Proposal Network

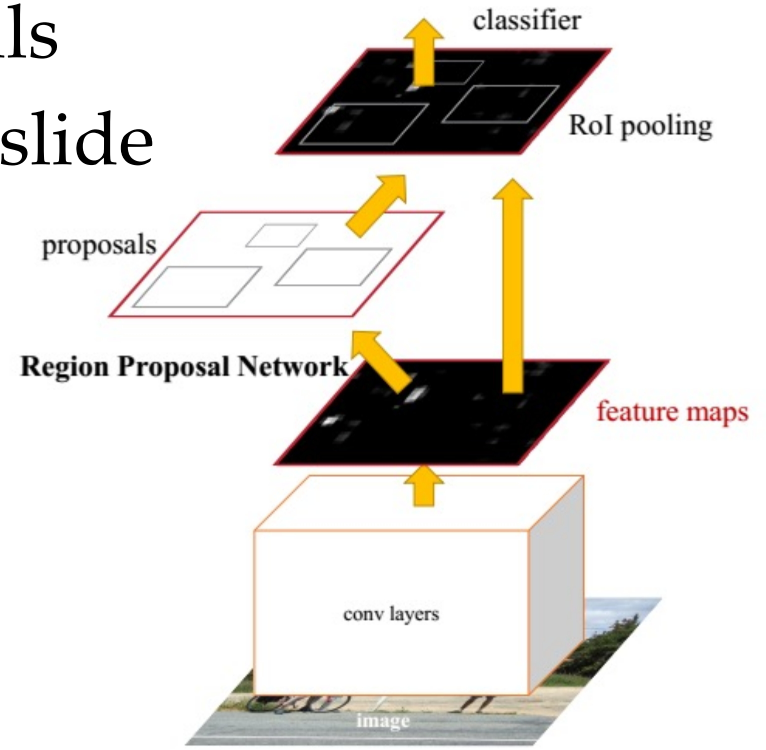
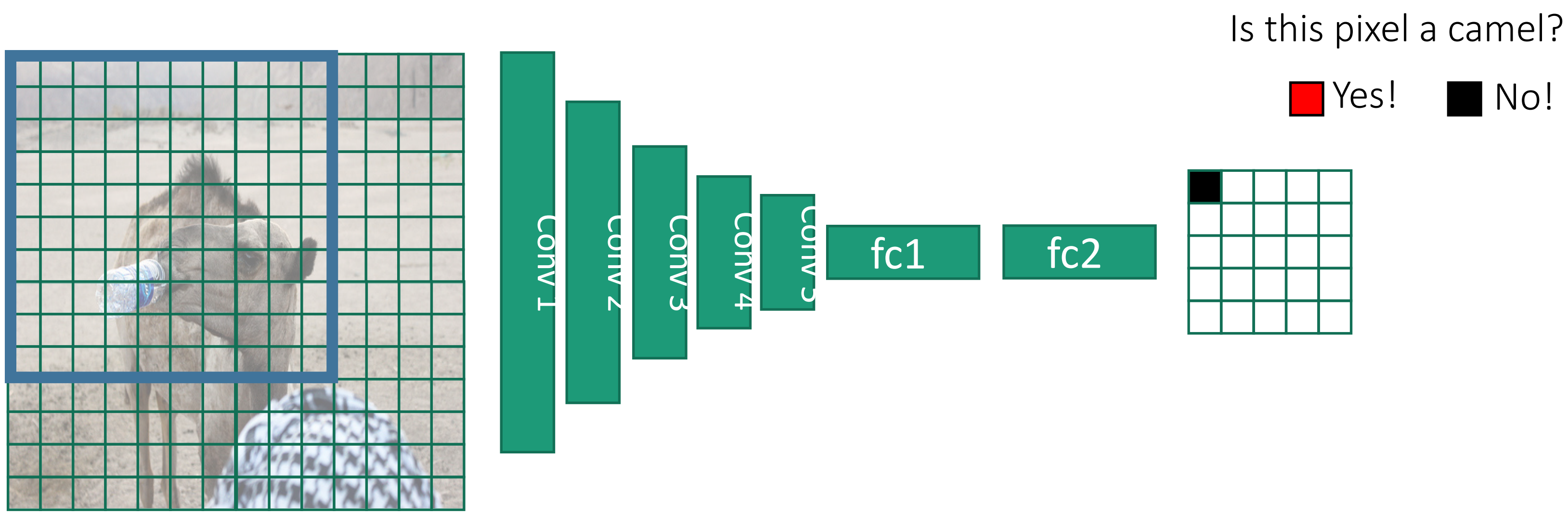


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

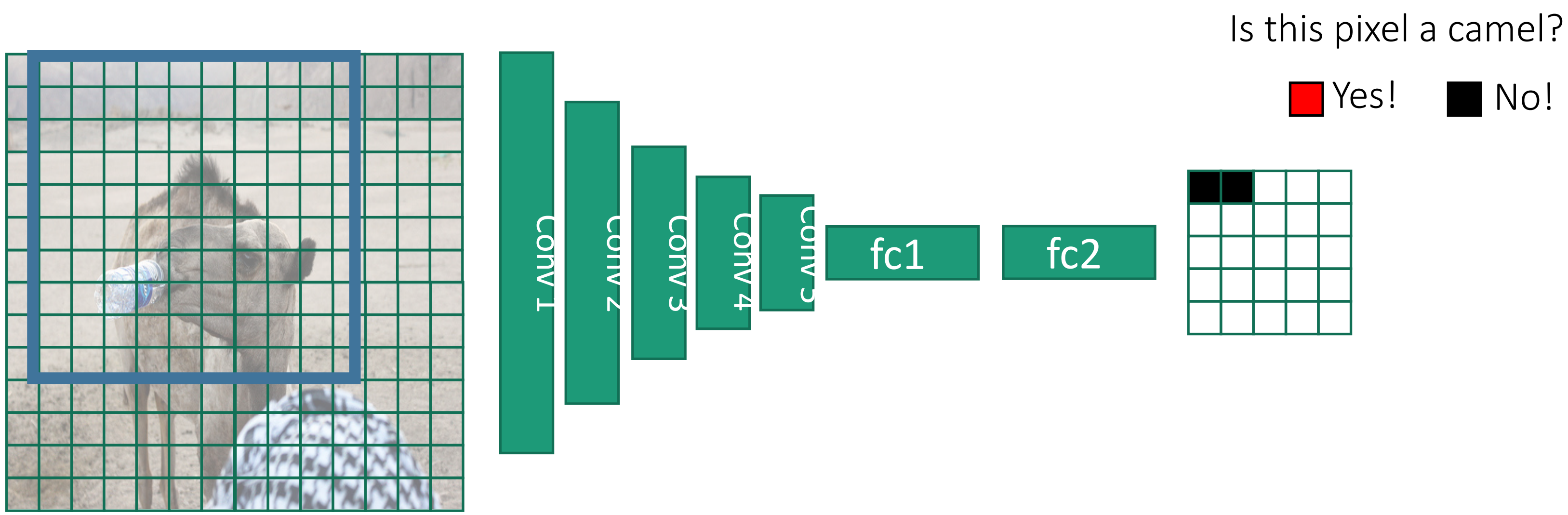
Going Fully Convolutional

- [LongCVPR2014]
- Image larger than network input: slide the network



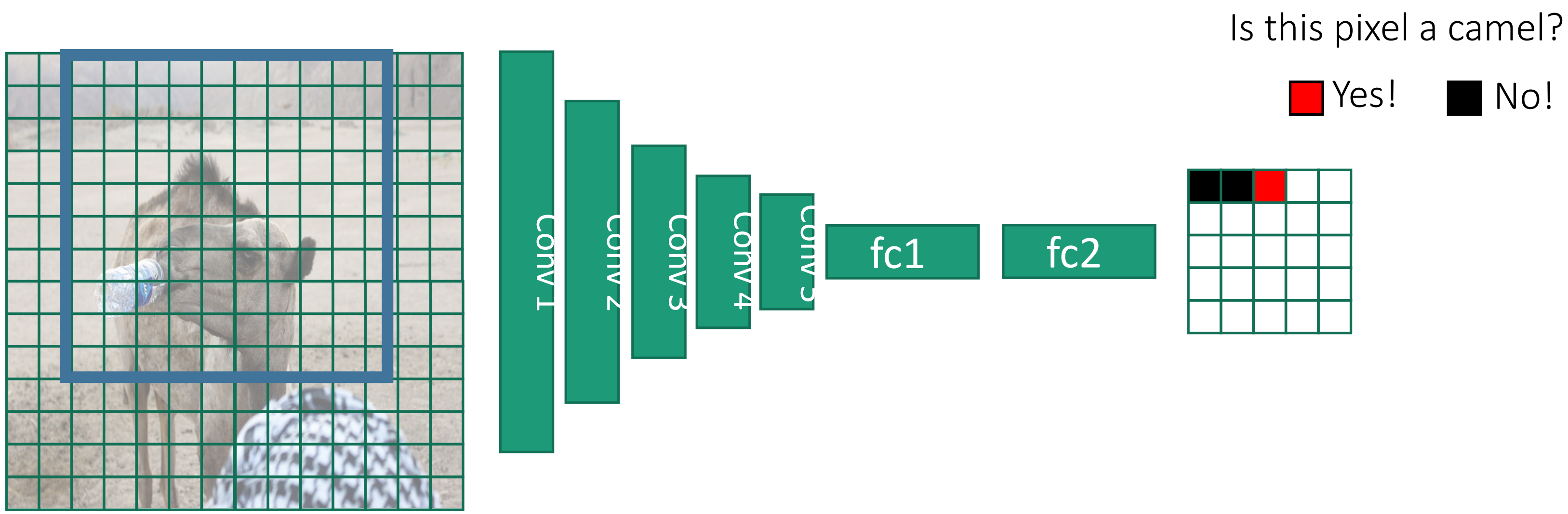
Going Fully Convolutional

- [LongCVPR2014]
- Image larger than network input: slide the network



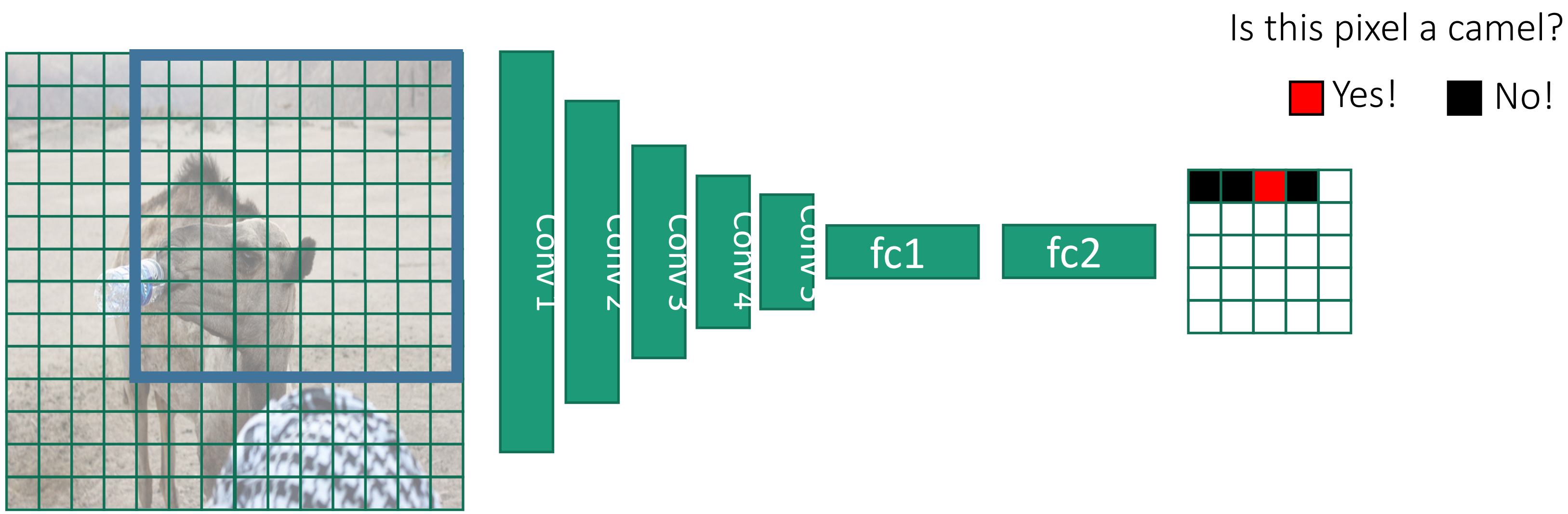
Going Fully Convolutional

- [LongCVPR2014]
- Image larger than network input: slide the network



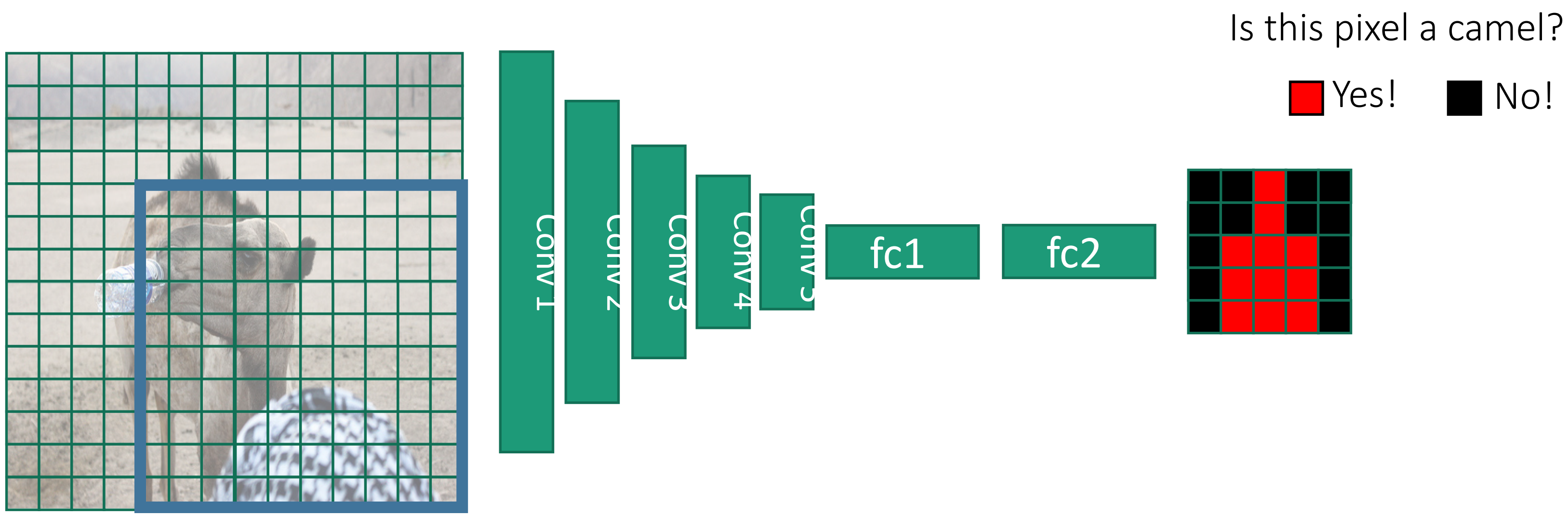
Going Fully Convolutional

- [LongCVPR2014]
- Image larger than network input: slide the network



Going Fully Convolutional

- [LongCVPR2014]
- Image larger than network input: slide the network



Fully Convolutional Networks

- [LongCVPR2014]
- Connect intermediate layers to output

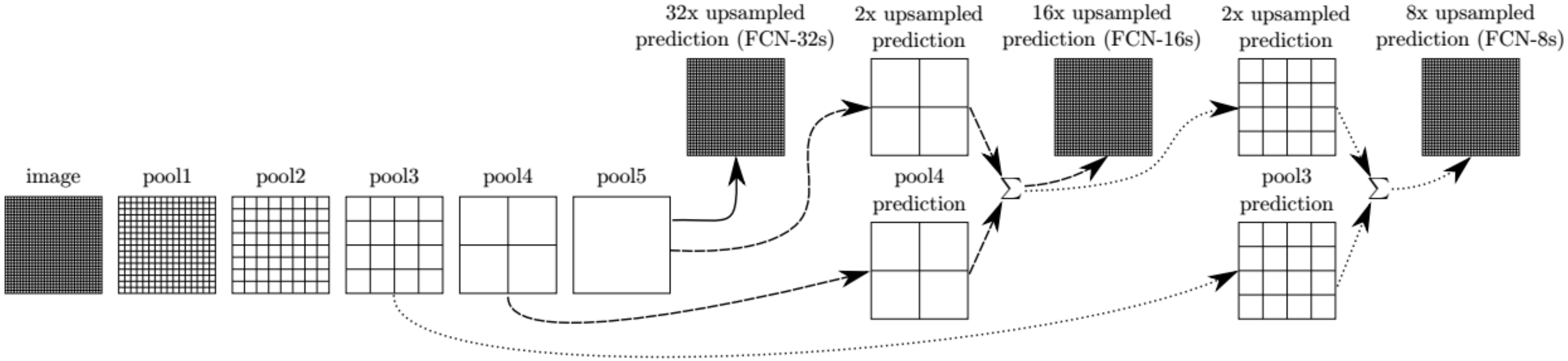


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are omitted. Solid line (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Dashed line (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Dotted line (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

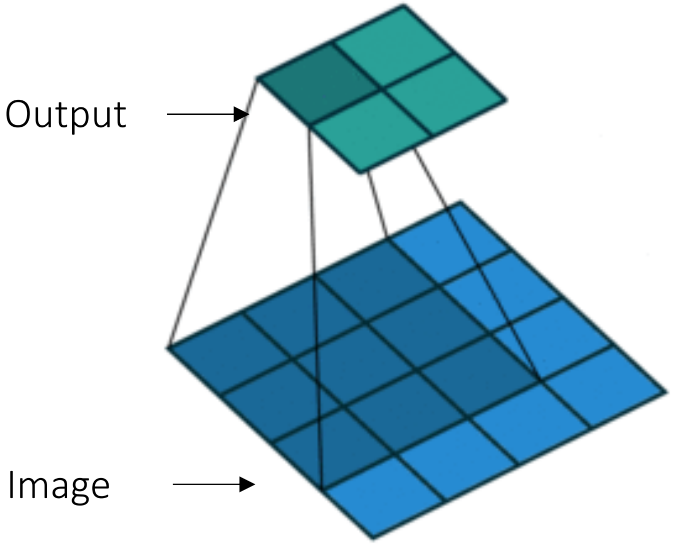
Fully Convolutional Networks

- Output is too coarse
  - Image Size 500x500, Alexnet Input Size: 227x227 → Output: 10x10

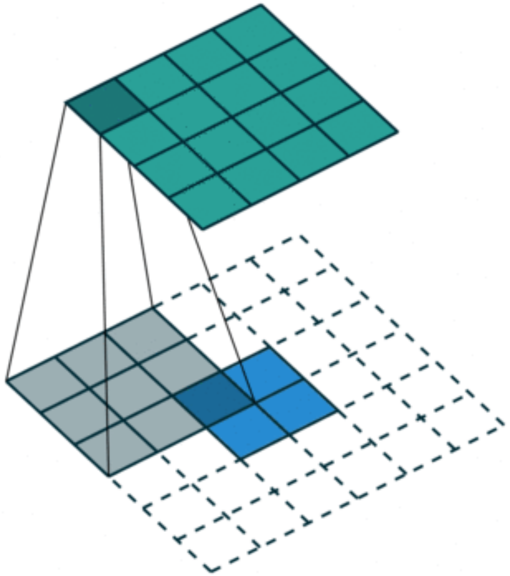
- How to obtain dense predictions?

- Upconvolution
  - Other names: deconvolution, transposed convolution, fractionally-strided convolutions

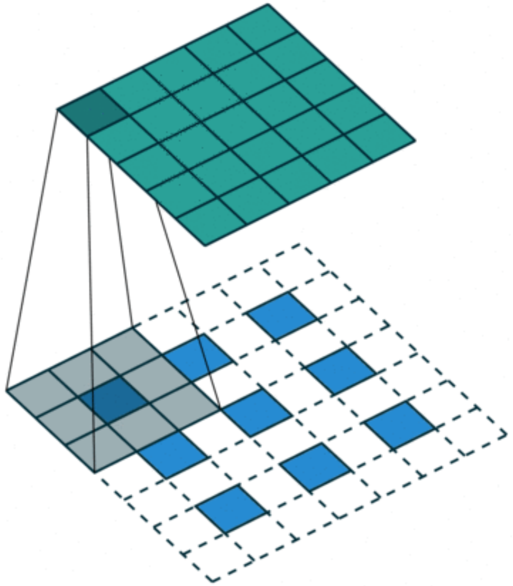
Deconvolutional modules



<u>Convolution</u>
No padding, no strides



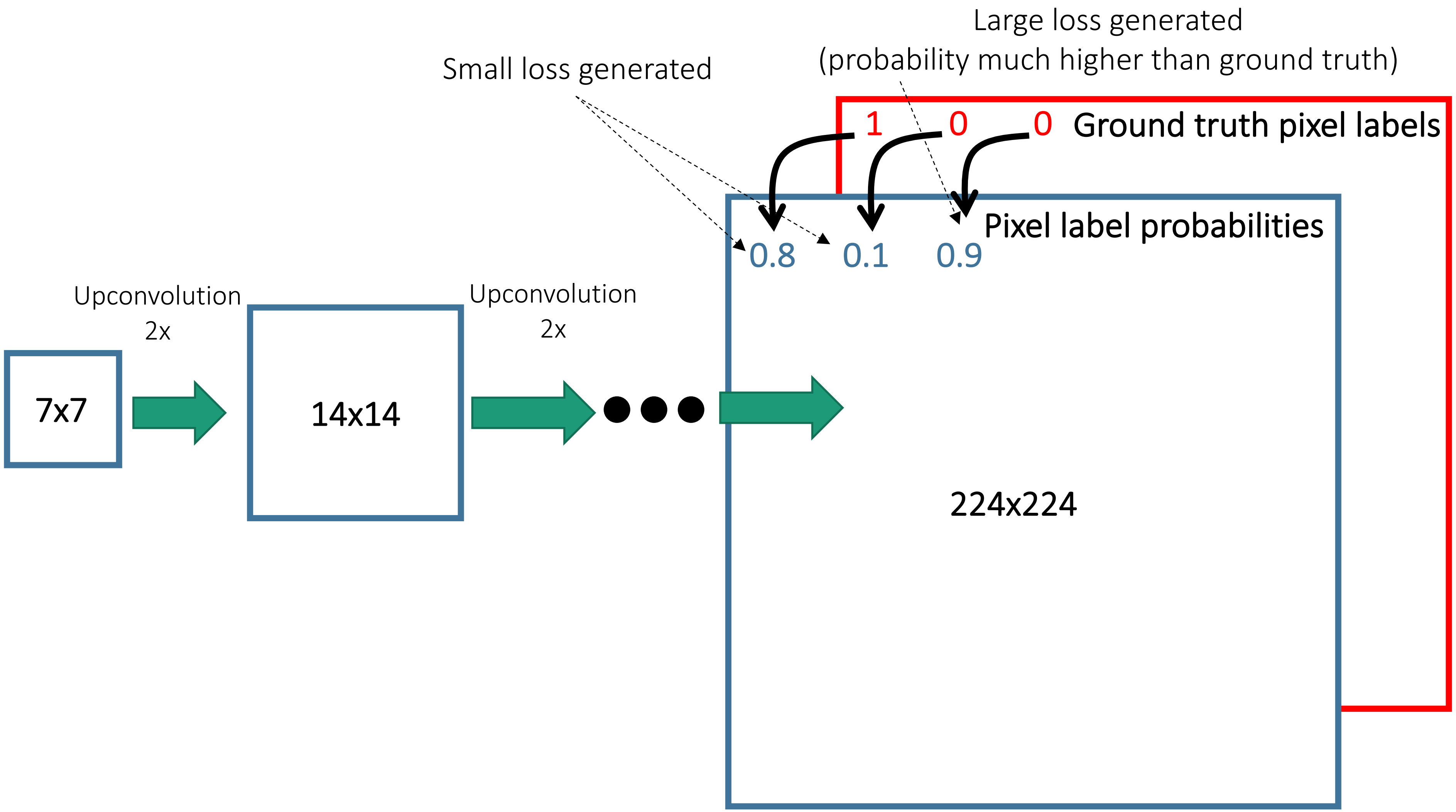
<u>Upconvolution</u>
No padding, no strides



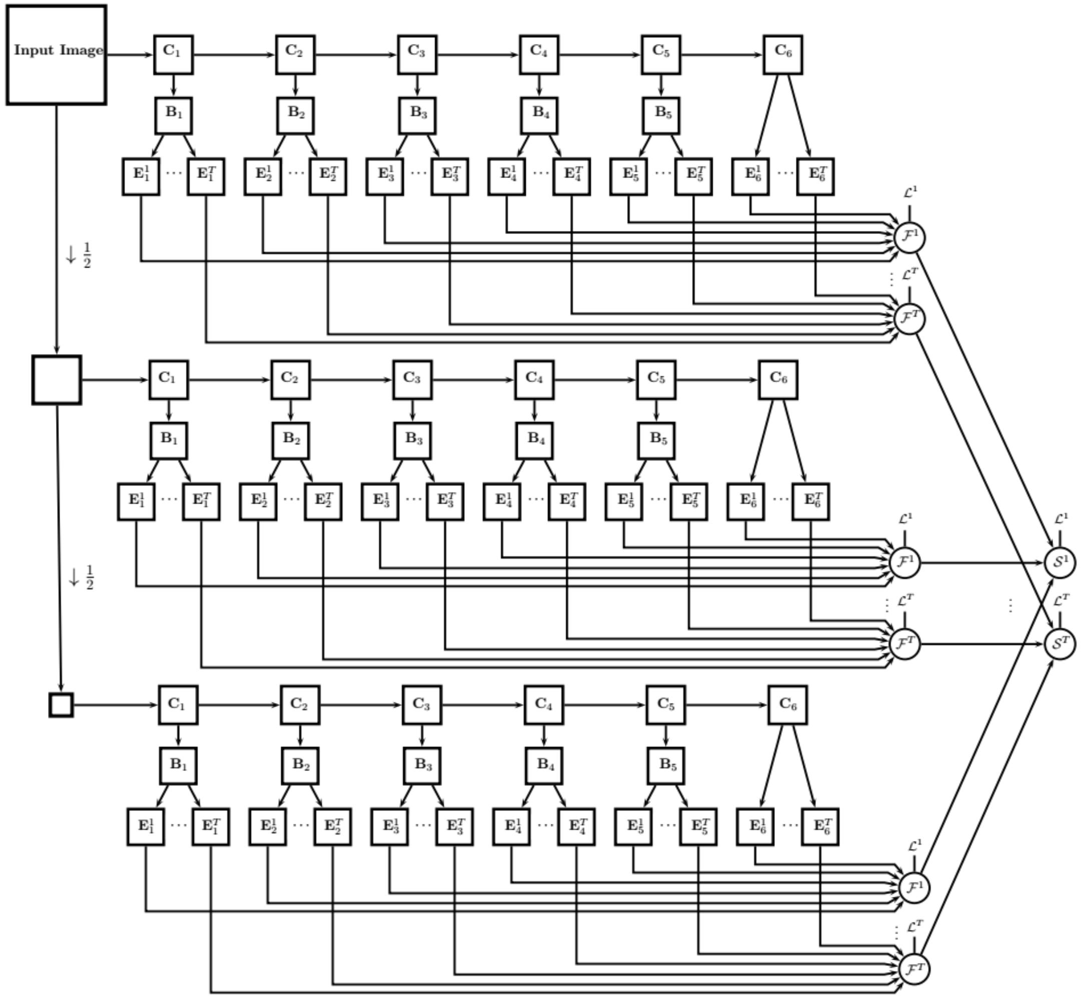
<u>Upconvolution</u>
Padding, strides

https://github.com/vdumoulin/conv_arithmetic

Coarse → Fine Output



Structured losses

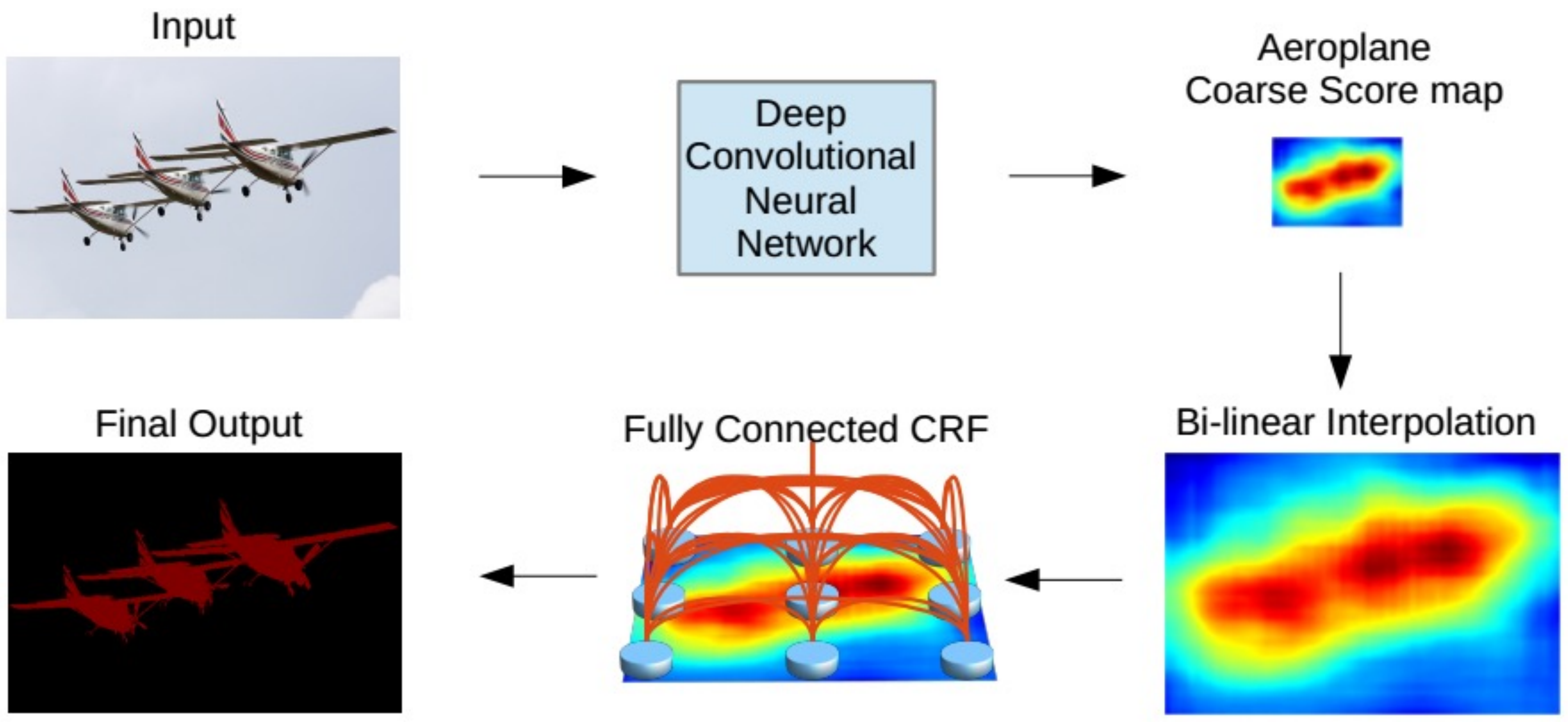


Deep ConvNets with CRF loss

- [Chen, Papandreou 2016]
- Segmentation map is good but not pixel-precise
  - Details around boundaries are lost
- Cast fully convolutional outputs as unary potentials
- Consider pairwise potentials between output dimensions

Deep ConvNets with CRF loss

- [Chen, Papandreou 2016]



Deep ConvNets with CRF loss

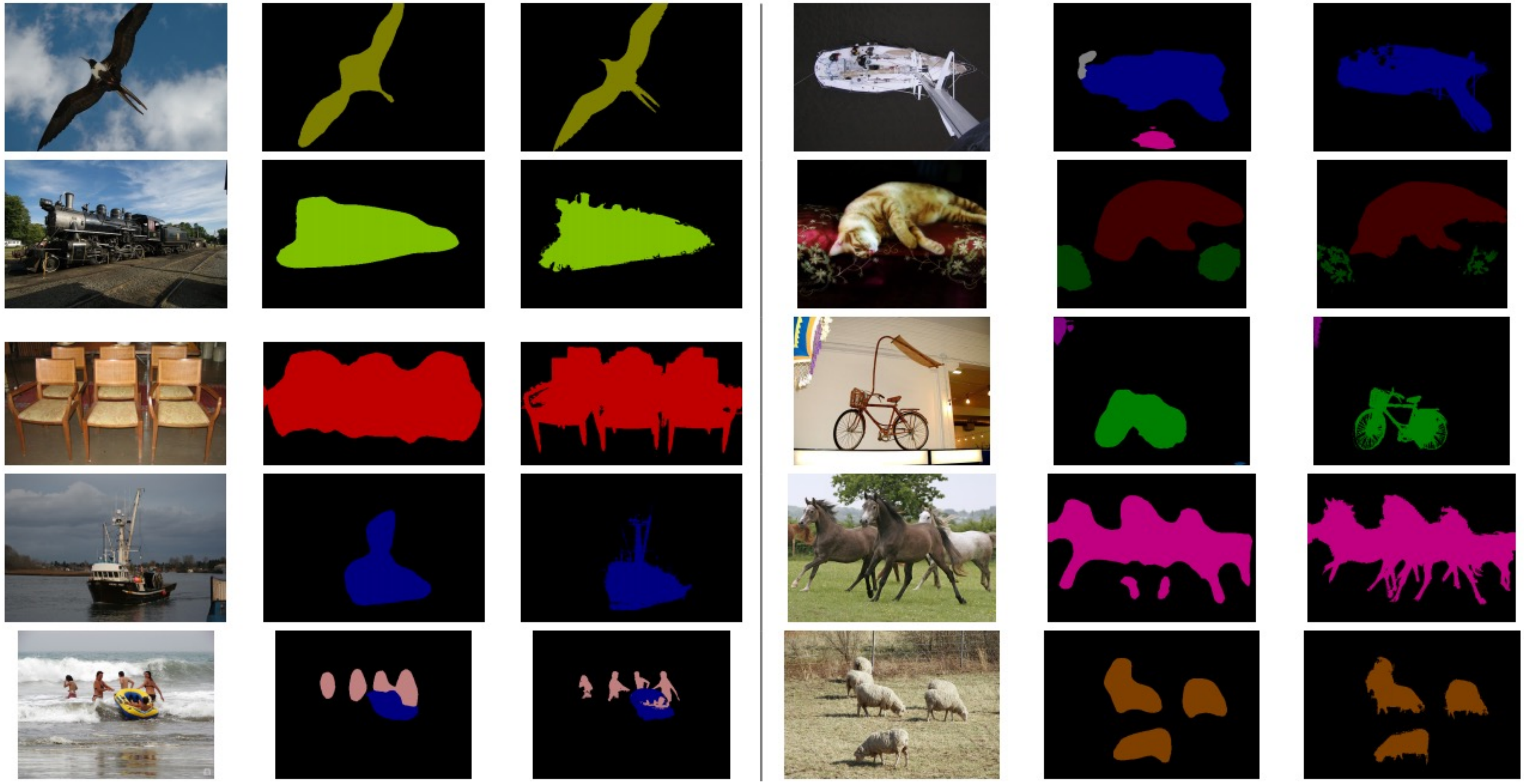
- [Chen, Papandreou 2016]
- Segmentation map is good but not pixel-precise
  - Details around boundaries are lost
- Cast fully convolutional outputs as unary potentials
- Consider pairwise potentials between output dimensions
- Include Fully Connected CRF loss to refine segmentation

Total loss Unary loss Pairwise loss

$$E(x) = \sum \theta_i(x_i) + \sum \theta_{ij}(x_i, x_j)$$

$$\theta_{ij}(x_i, x_j) \sim w_1 \exp\left(-\alpha |p_i - p_j|^2 - \beta |I_i - I_j|^2\right) + w_2 \exp(-\gamma |p_i - p_j|^2)$$

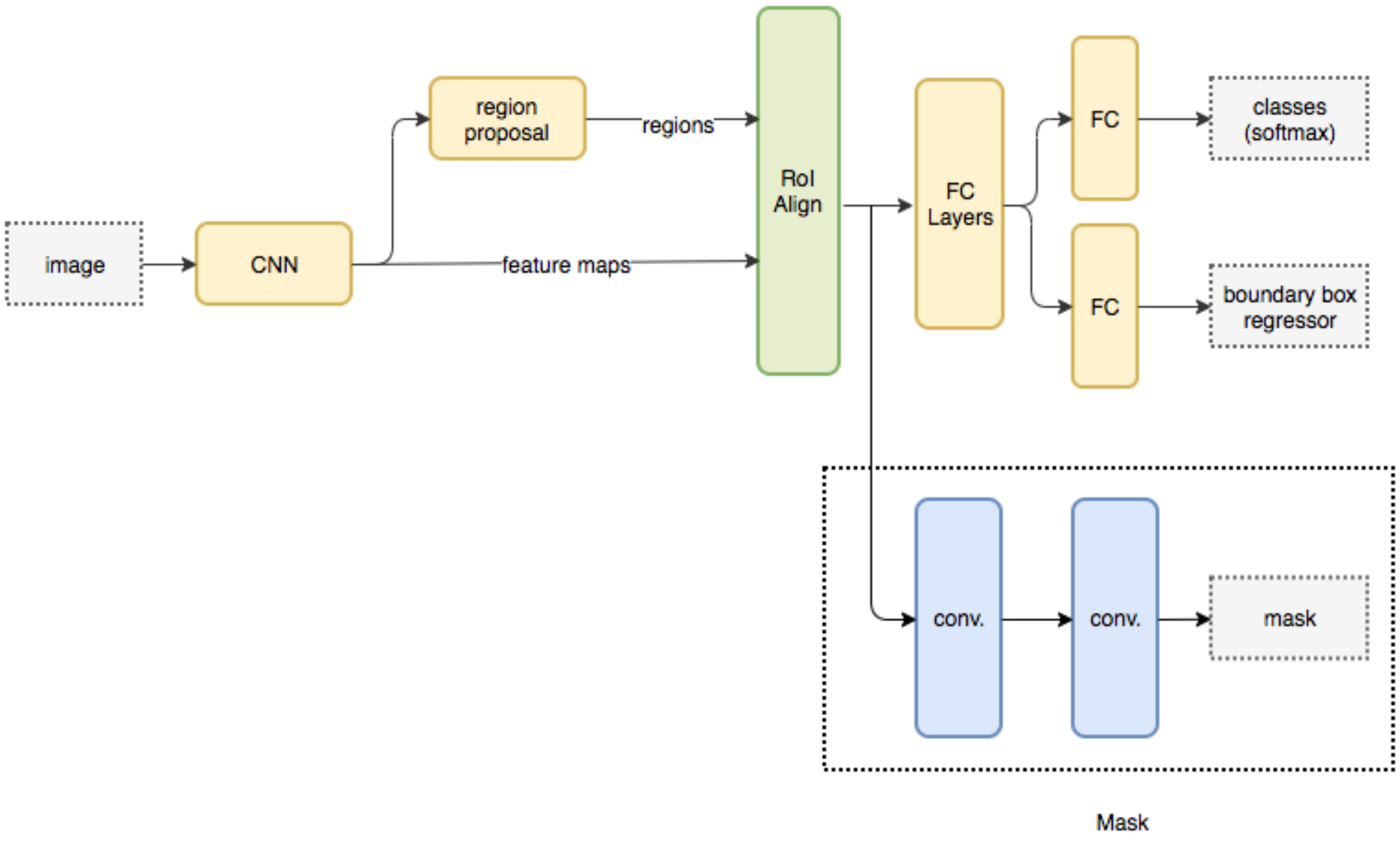
Examples



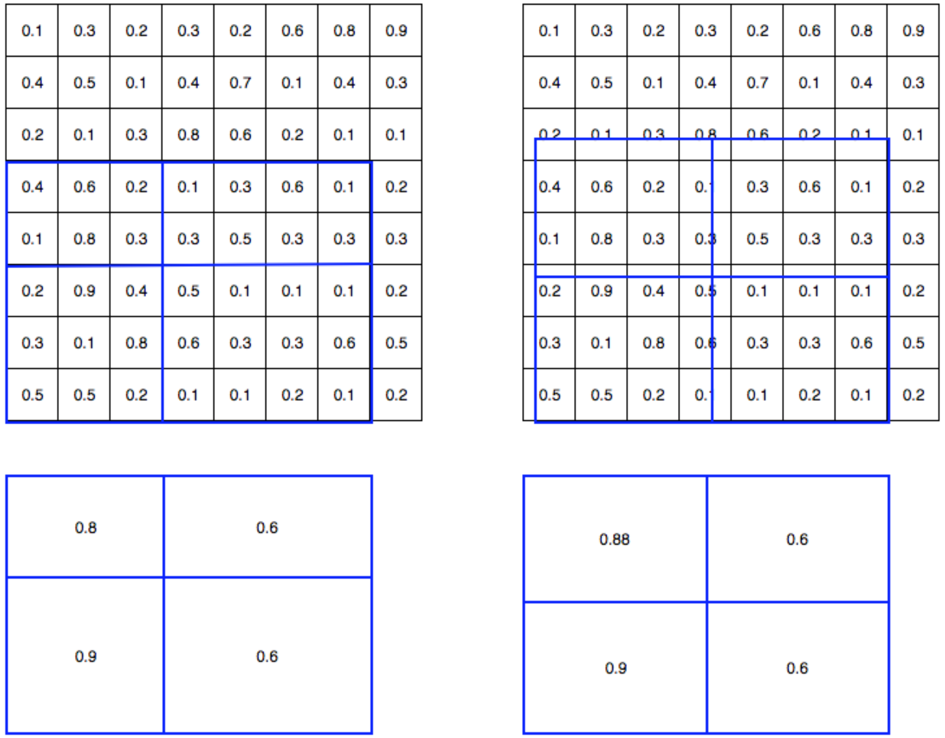
Mask R-CNN

- State-of-the-art in semantic segmentation
- Heavily relies on Fast R-CNN
- Can work with different architectures, also ResNet
- Runs at 195ms per image on an Nvidia Tesla M40 GPU
- Can also be used for Human Pose Estimation

Mask R-CNN: R-CNN + 2 layers

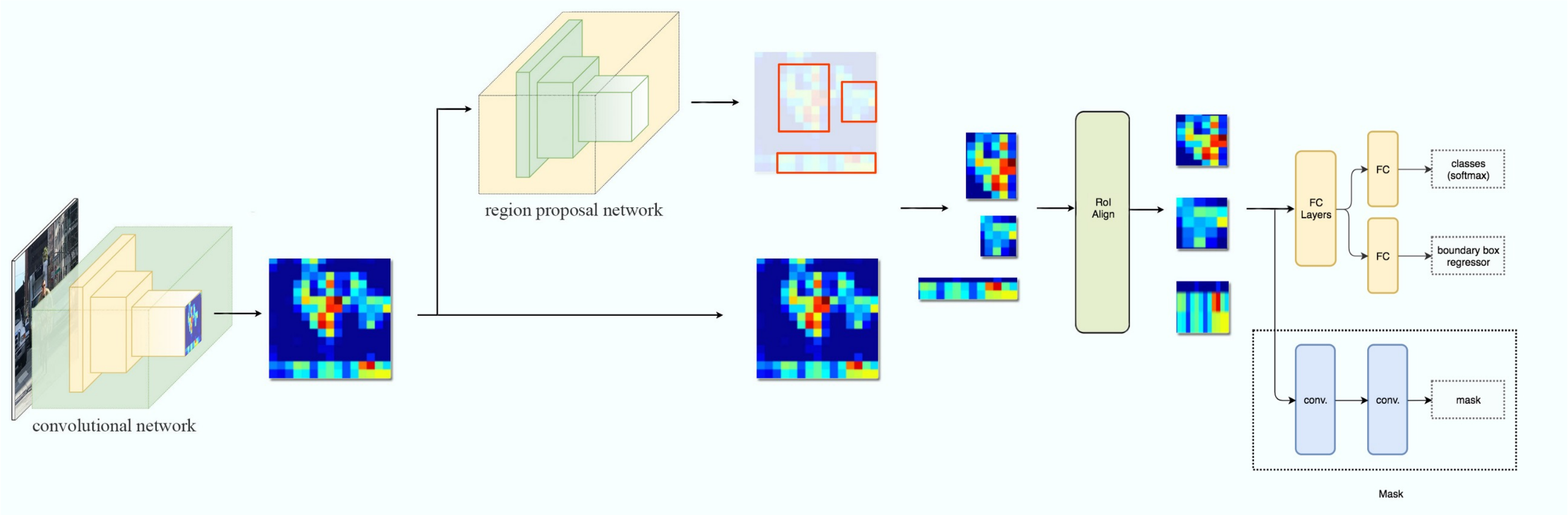


Mask R-CNN: ROI Align



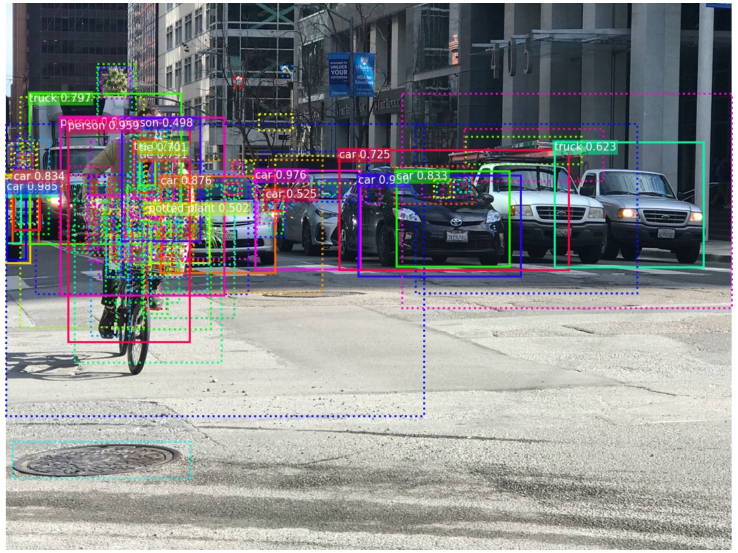
	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

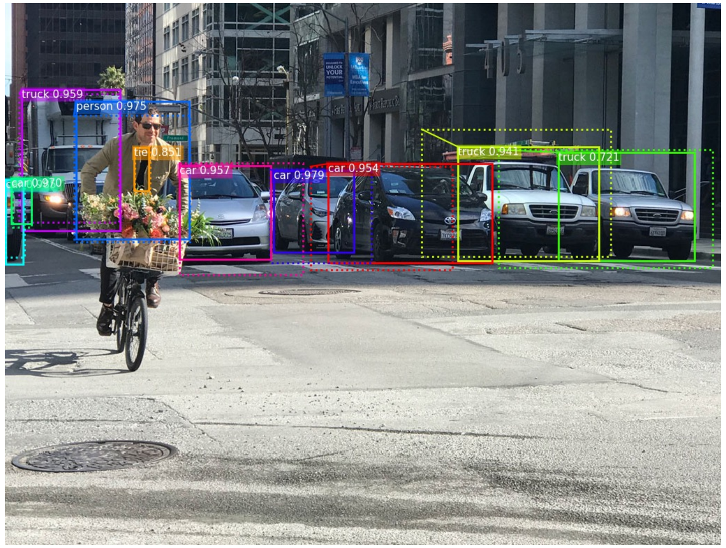
Mask R-CNN

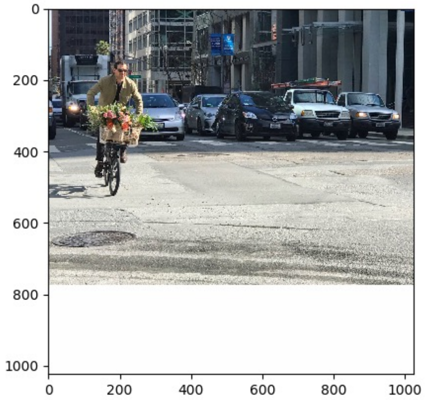


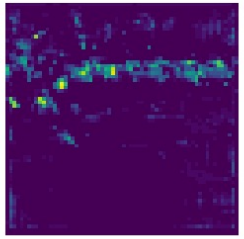
region proposal network
convolutional network
RoI
Align
FC
Layers
FC
FC
classes
(softmax)
boundary box
regressor
conv.
conv.
mask
Mask

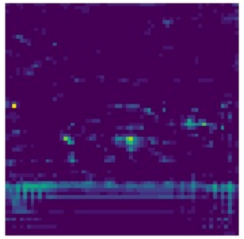
Mask R-CNN

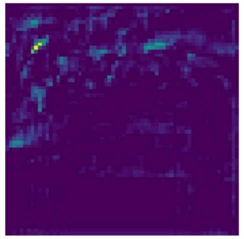

truck 0.797
person 0.959
tie 0.701
car 0.834
car 0.985
car 0.876
car 0.976
car 0.525
car 0.725
car 0.833
truck 0.623
potted plant 0.502

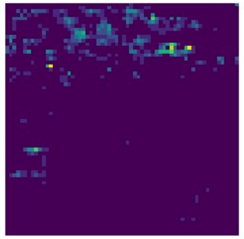

truck 0.959
person 0.975
tie 0.851
car 0.957
car 0.979
car 0.954
truck 0.941
truck 0.721
car 0.970


0
200
400
600
800
1000
0
200
400
600
800
1000

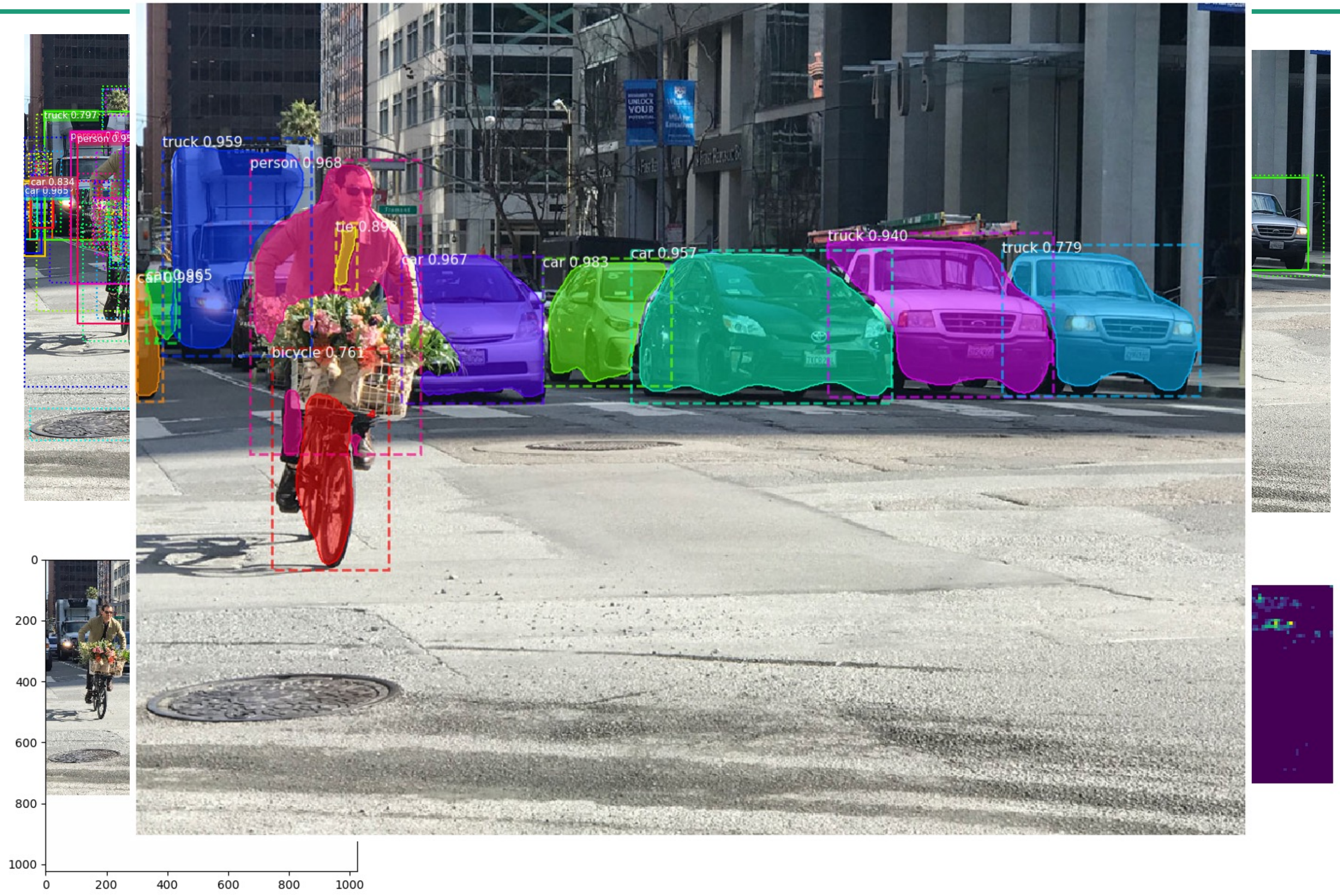




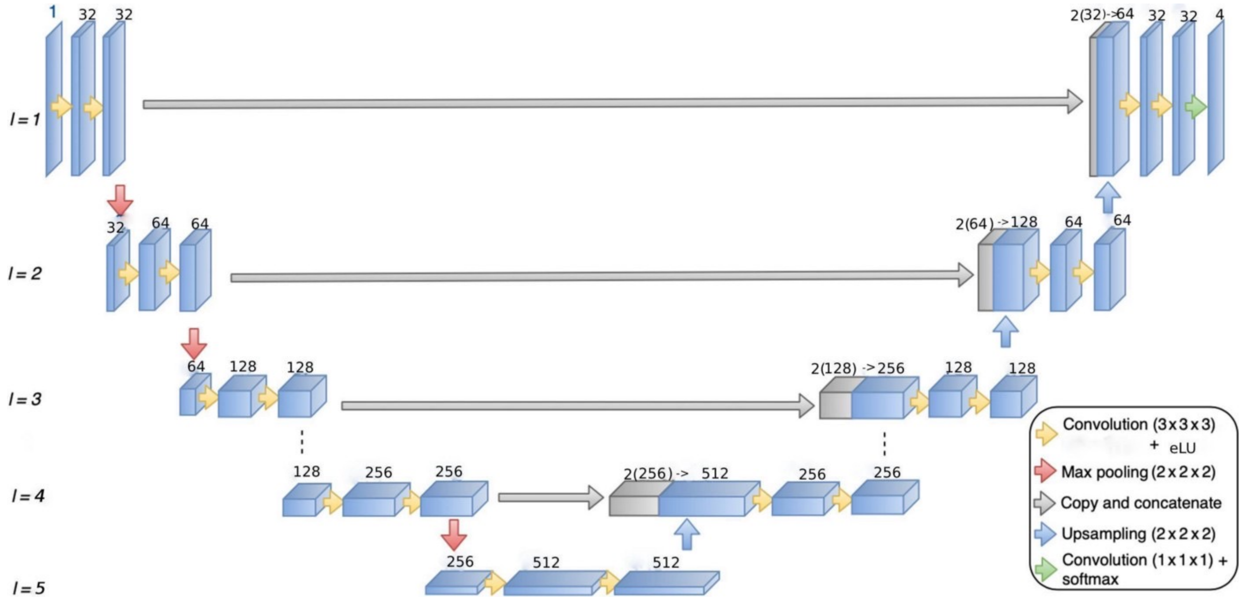




Mask R-CNN


truck 0.959
person 0.968
tie 0.89
car 0.967
car 0.983
car 0.957
truck 0.940
truck 0.779
bicycle 0.761
truck 0.797
car 0.834
0
200
400
600
800
1000
0
200
400
600
800
1000

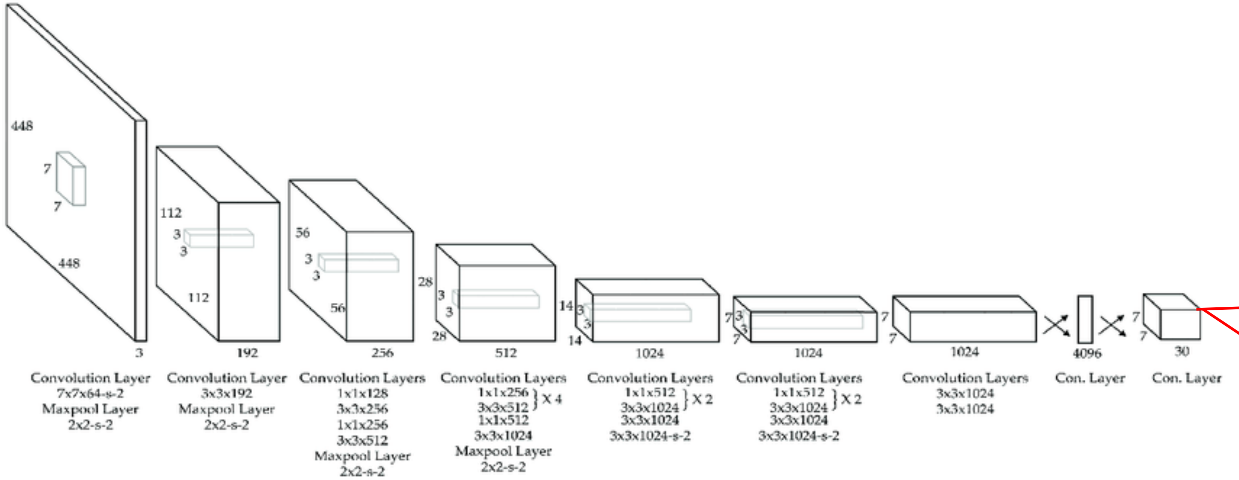
Unet

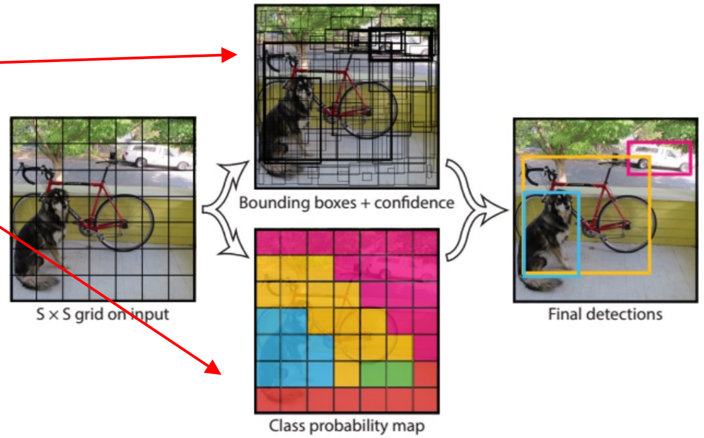


Input Ground truth Prediction

Ronneberger, Fischer, Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

YOLO



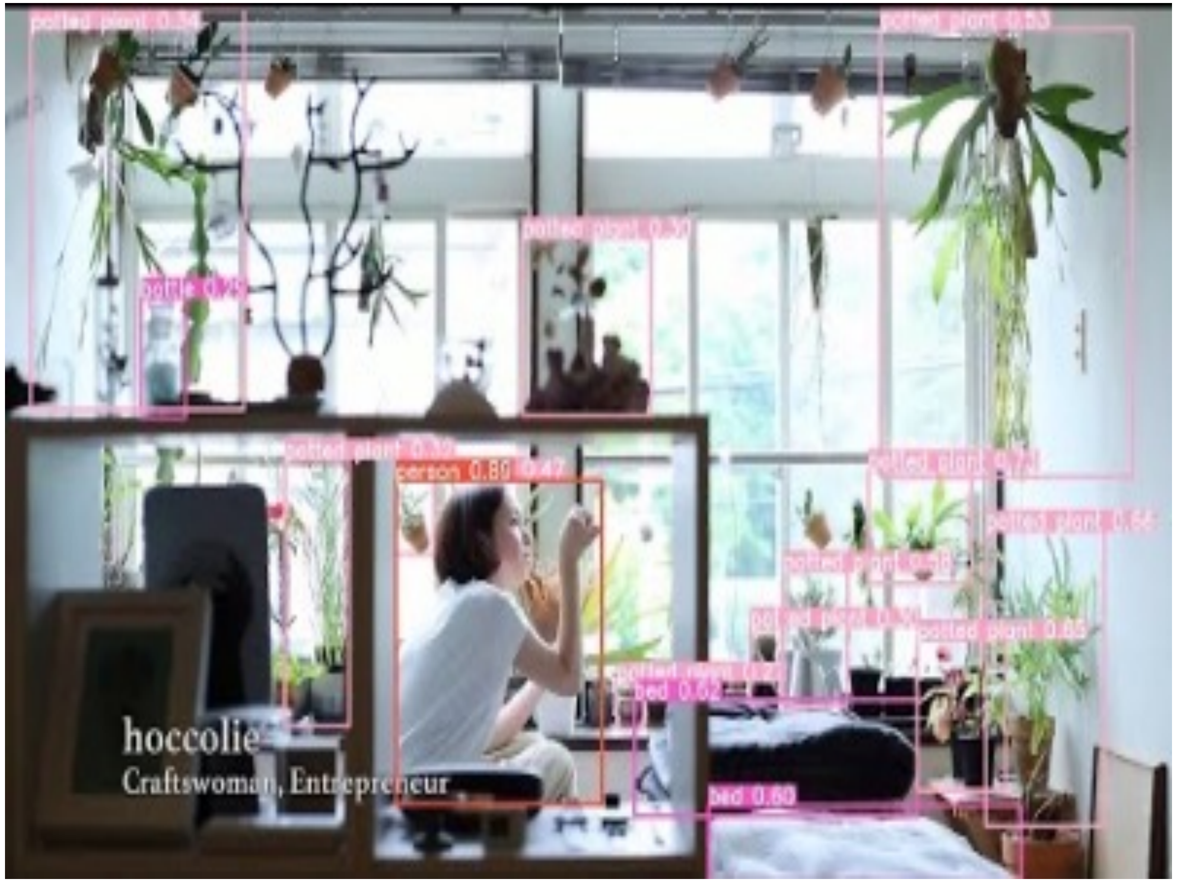


- ‘One-shot’ detection
- No proposals → Much faster

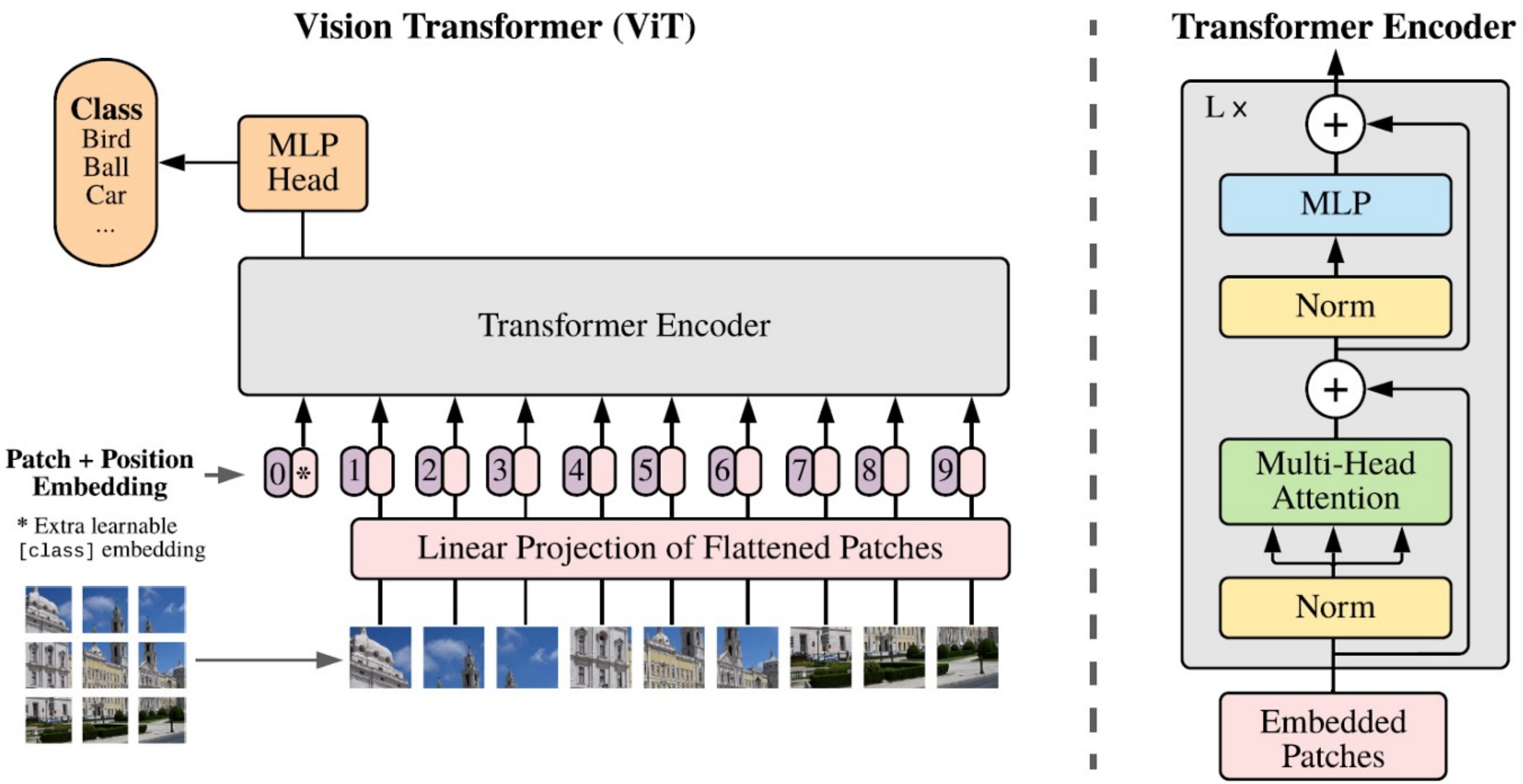
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Redmon, Divvala, Girshick, Farhadi, You Only Look Once: Unified, Real-Time Object Detection, 2015

YOLO v5

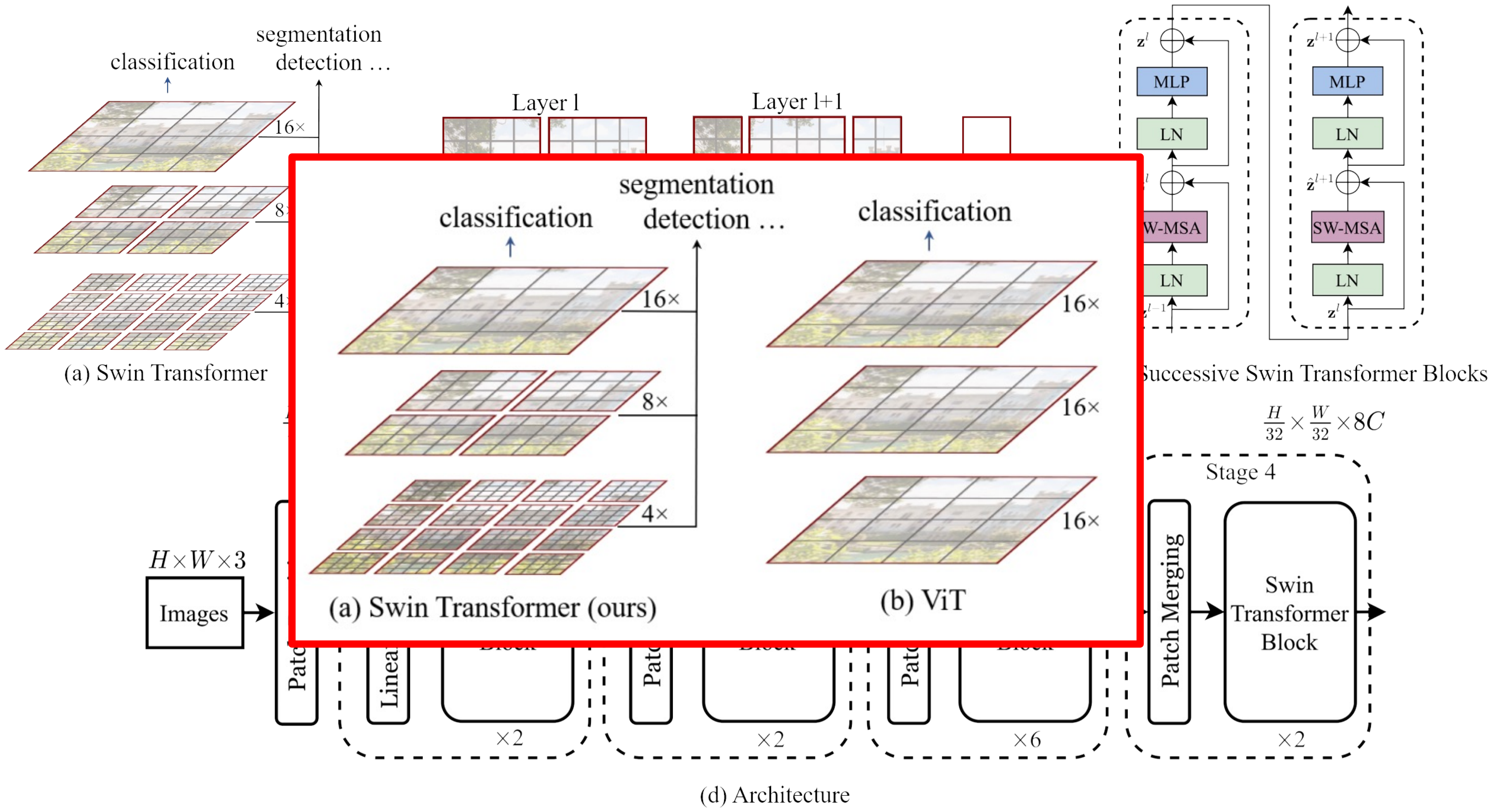

person 0.89
bed 0.62
bed 0.60
hoccolie
Craftswoman, Entrepreneur

ViT



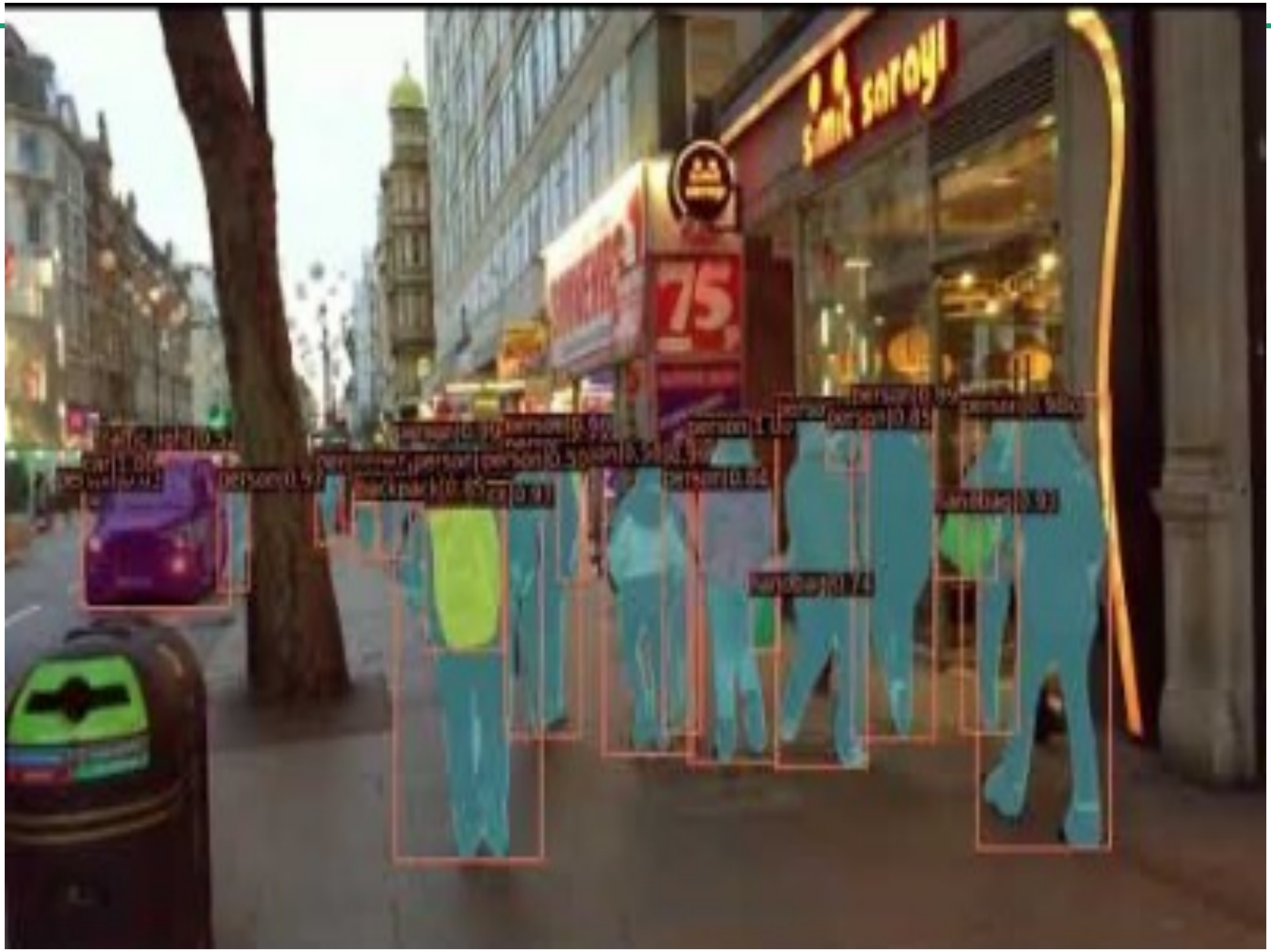
Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2020

Swin Transformer

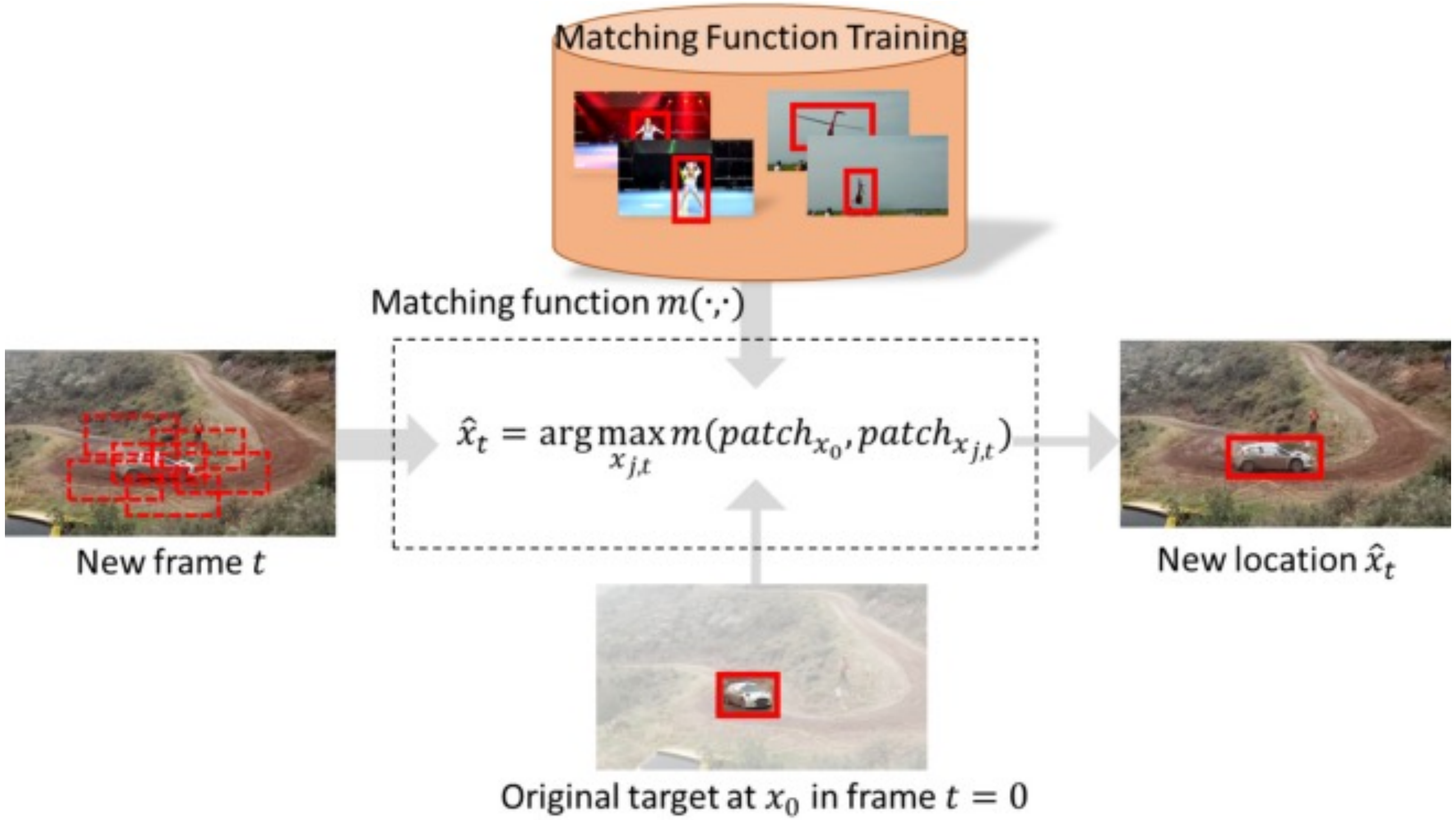


Liu et al., Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, 2021

Swin Transformer



SINT: Siamese Networks for Tracking



Tao, Gavves, Smeulders, Siamese Instance Search for Tracking, 2016

SINT: Siamese Networks for Tracking

- While tracking, the only definitely correct training example is the first frame
  - All others are inferred by the algorithm

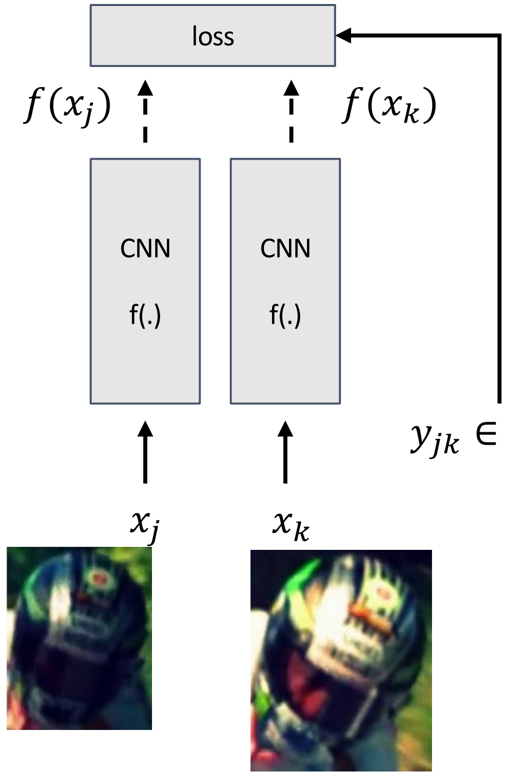
- If the “inferred positives” are correct, then the model is already good enough and no update is needed

- If the “inferred positives” are incorrect, updating the model using wrong positive examples will eventually destroy the model

Basic Idea

- No model updates through time to avoid model contamination
- Instead, learn invariance model $f(\boldsymbol{d}x)$
  - invariances shared between objects
  - reliable, external, rich, category-independent, data
- Assumption
  - The appearance variances are shared amongst object and categories
  - Learning can accurate enough to identify common appearance variances
- Solution: Use a Siamese Network to compare patches between images
  - Then “tracking” equals finding the most similar patch at each frame (no temporal modelling)

Training



Marginal Contrastive Loss:

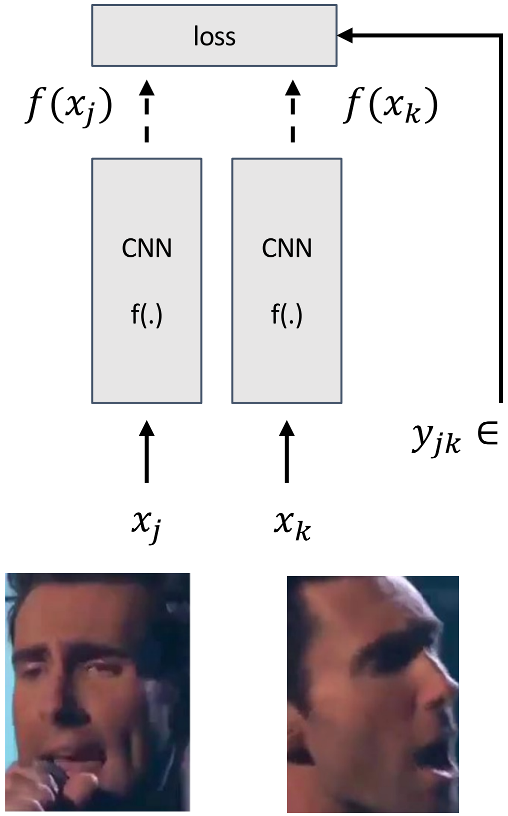
$$L(x_j, x_k, y_{jk}) = \frac{1}{2} y_{jk} D^2 + \frac{1}{2}(1 - y_{jk}) \max(0, \sigma - D^2)$$

$y_{jk} \in \{0,1\}$ $\quad D = \left\| f(x_j) - f(x_k) \right\|_2$

Matching function (after learning):

$$m(x_j, x_k) = f(x_j) \cdot f(x_k)$$

Training



$y_{jk} \in \{0,1\}$

Marginal Contrastive Loss:

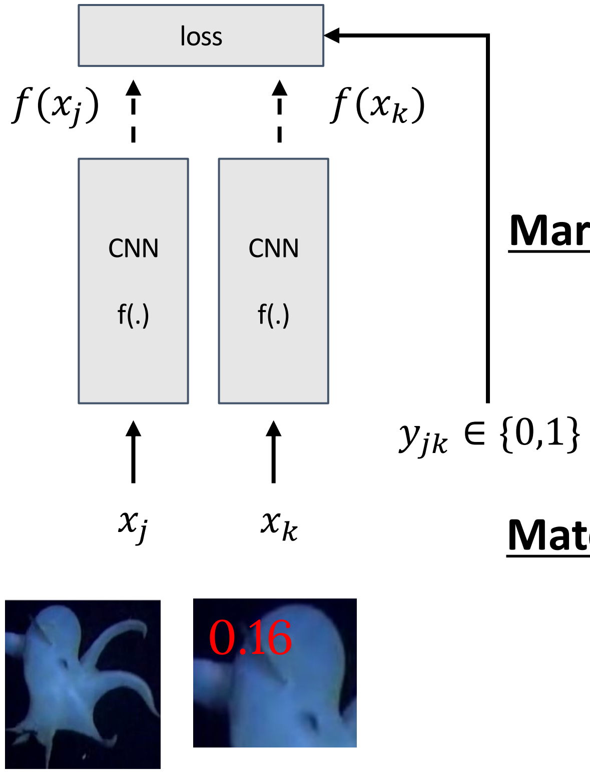
$$L(x_j, x_k, y_{jk}) = \frac{1}{2} y_{jk} D^2 + \frac{1}{2}(1 - y_{jk}) \max(0, \sigma - D^2)$$

$$D = \left\| f(x_j) - f(x_k) \right\|_2$$

Matching function (after learning):

$$m(x_j, x_k) = f(x_j) \cdot f(x_k)$$

Training



Marginal Contrastive Loss:

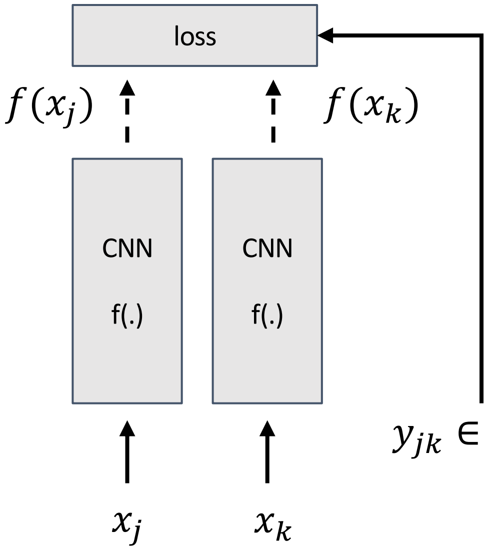
$$L(x_j, x_k, y_{jk}) = \frac{1}{2} y_{jk} D^2 + \frac{1}{2}(1 - y_{jk}) \max(0, \sigma - D^2)$$

$$D = \left\| f(x_j) - f(x_k) \right\|_2$$

Matching function (after learning):

$$m(x_j, x_k) = f(x_j) \cdot f(x_k)$$

Training



Marginal Contrastive Loss:

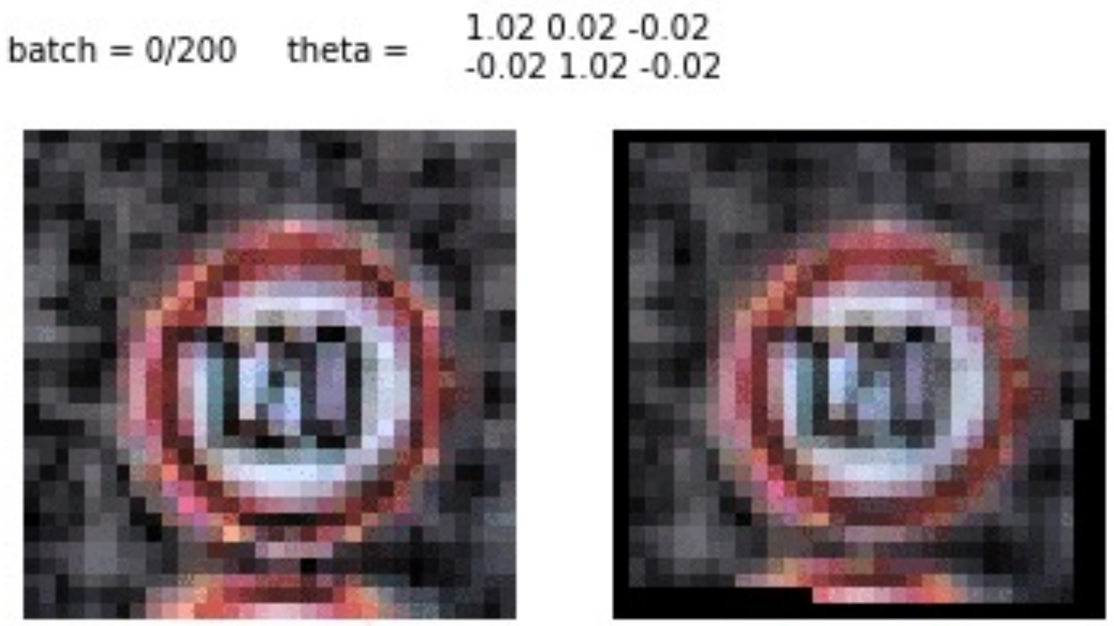
$$L(x_j, x_k, y_{jk}) = \frac{1}{2} y_{jk} D^2 + \frac{1}{2}(1 - y_{jk})\max(0, \sigma - D^2)$$

$y_{jk} \in \{0,1\}$ $\quad D = \left\| f(x_j) - f(x_k) \right\|_2$

Matching function (after learning):

$$m(x_j, x_k) = f(x_j) \cdot f(x_k)$$

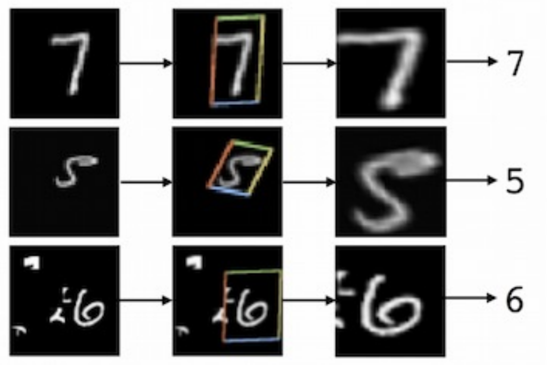
Spatial Transform Networks

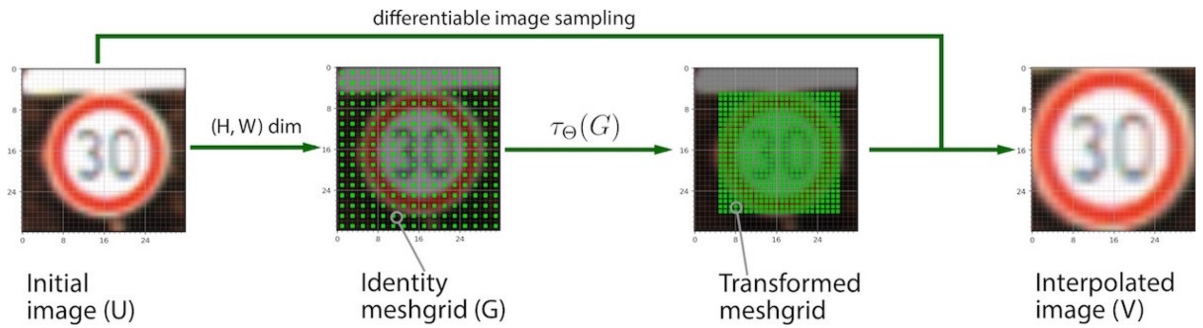


Problem

- ConvNets sometimes are robust enough to input changes
  - While pooling gives some invariance, only in deeper layers the pooling receptive field is large enough for this invariance to be noteworthy
  - One way to improve robustness: Data augmentation

- Smarter way: Spatial Transformer Networks



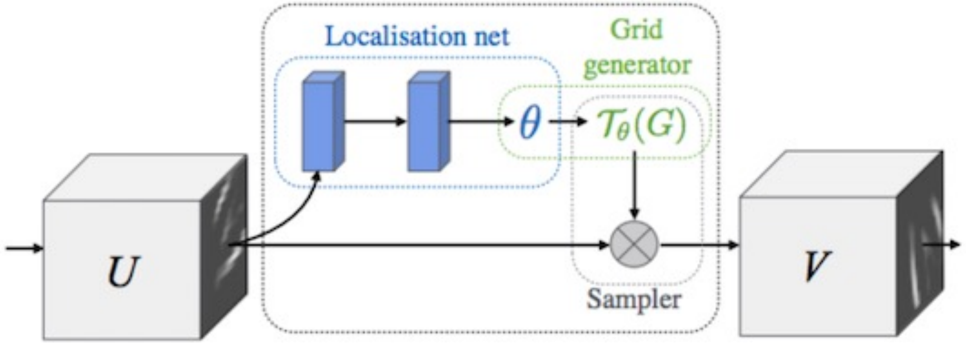


Basic Idea

- Define a geometric transformation matrix
- $\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix}$
- Four interesting transformations
  - Identity, i.e. $\Theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
  - Rotation, e.g., $\Theta \approx \begin{bmatrix} 0.7 & -0.7 & 0 \\ 0.7 & 0.7 & 0 \end{bmatrix}$ for 45^o, as $\cos(\frac{\pi}{4}) \approx 0.7$
  - Zooming in, e.g. $\Theta \approx \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}$ for 2X zooming in
  - Zooming in, e.g. $\Theta \approx \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$ for 2X zooming out

Basic Idea

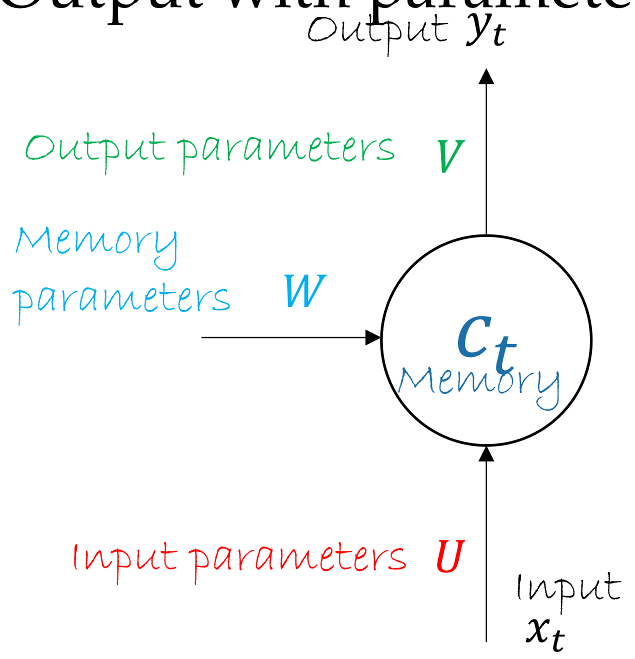
- Then, define a mesh grid (x_i^t, y_i^t) on the original image and apply the geometric transformations
- $\begin{bmatrix} x_i^s \\ y_i^s \end{bmatrix} = \Theta \cdot \begin{bmatrix} x_i^t \\ y_i^t \\ 1 \end{bmatrix}$
- Produce the new image using the transformation above and an interpolation method
- Learn the para … om the data
- A localization … ven a new image



Sequential data

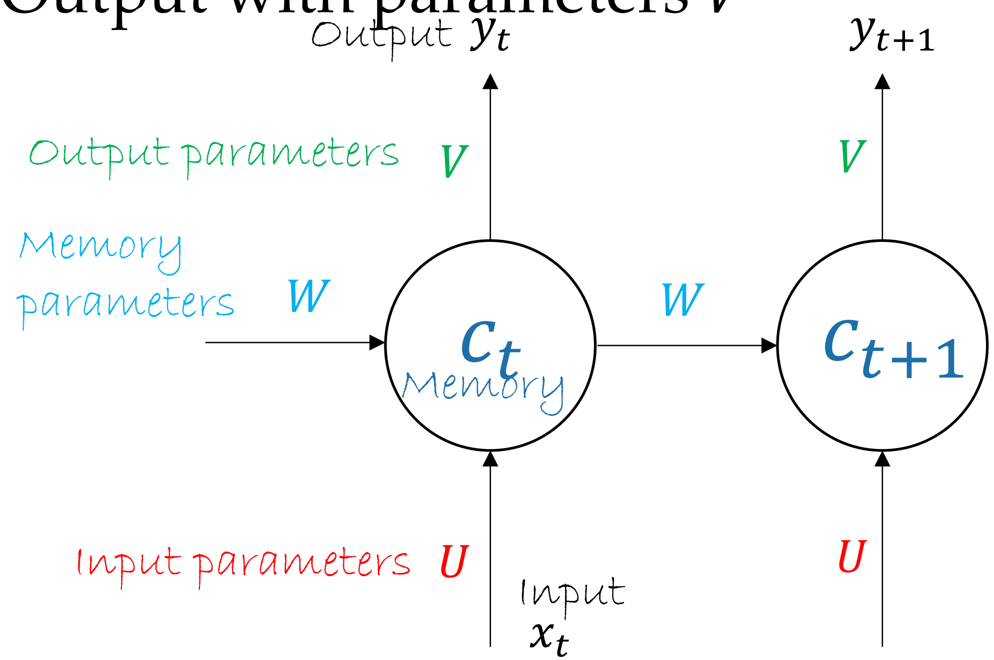
Recurrent Networks

- Simplest model
  - Input with parameters U
  - Memory embedding with parameters W
  - Output with parameters V



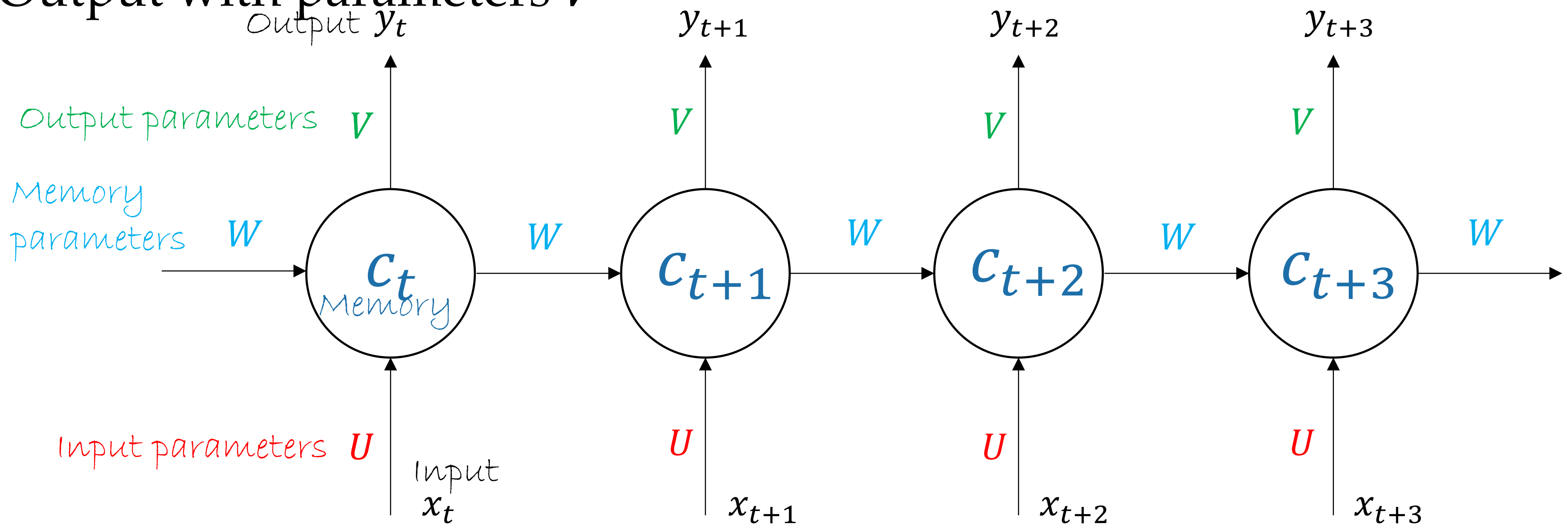
Recurrent Networks

- Simplest model
  - Input with parameters U
  - Memory embedding with parameters W
  - Output with parameters V

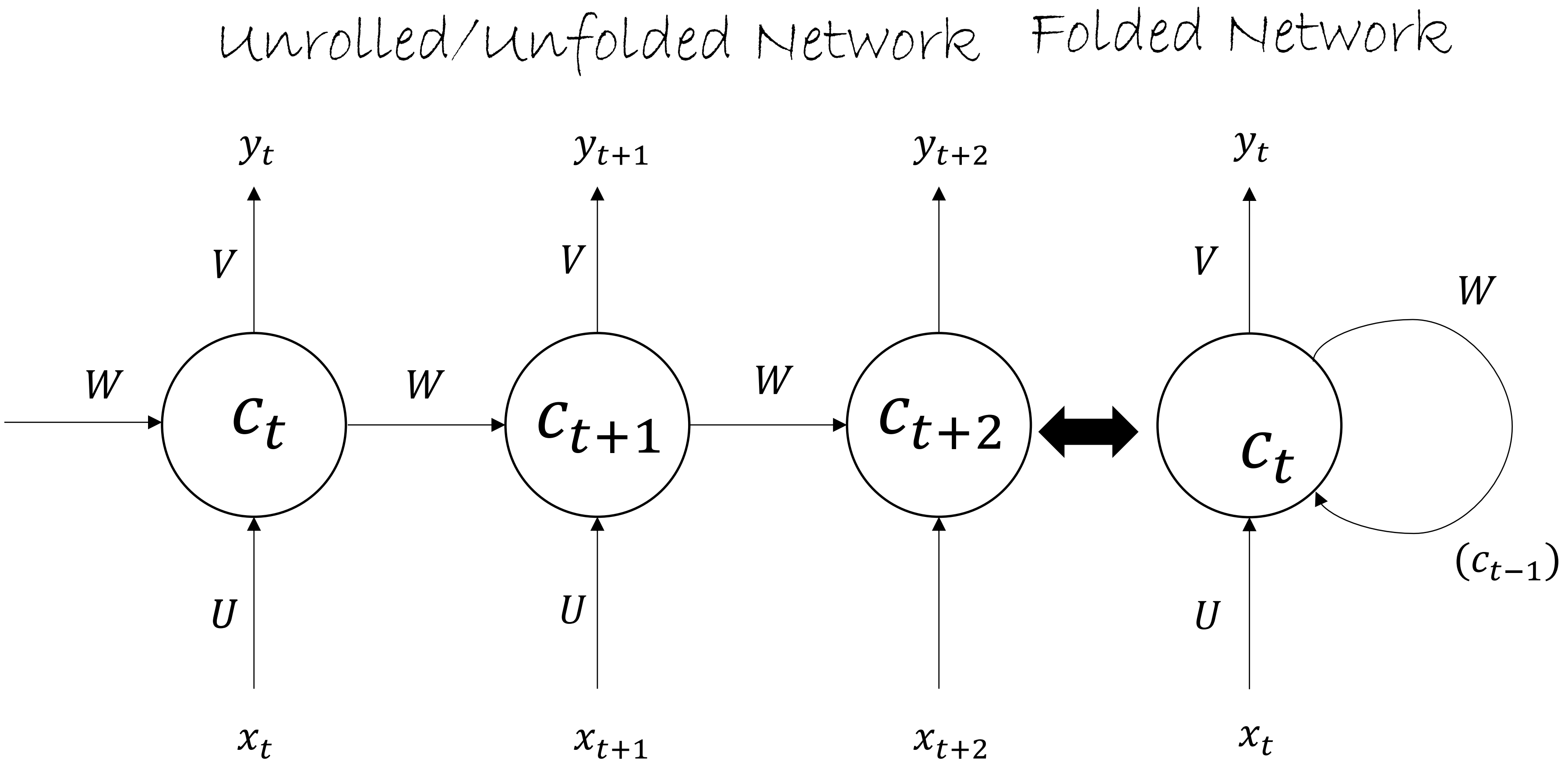


Recurrent Networks

- Simplest model
  - Input with parameters U
  - Memory embedding with parameters W
  - Output with parameters V

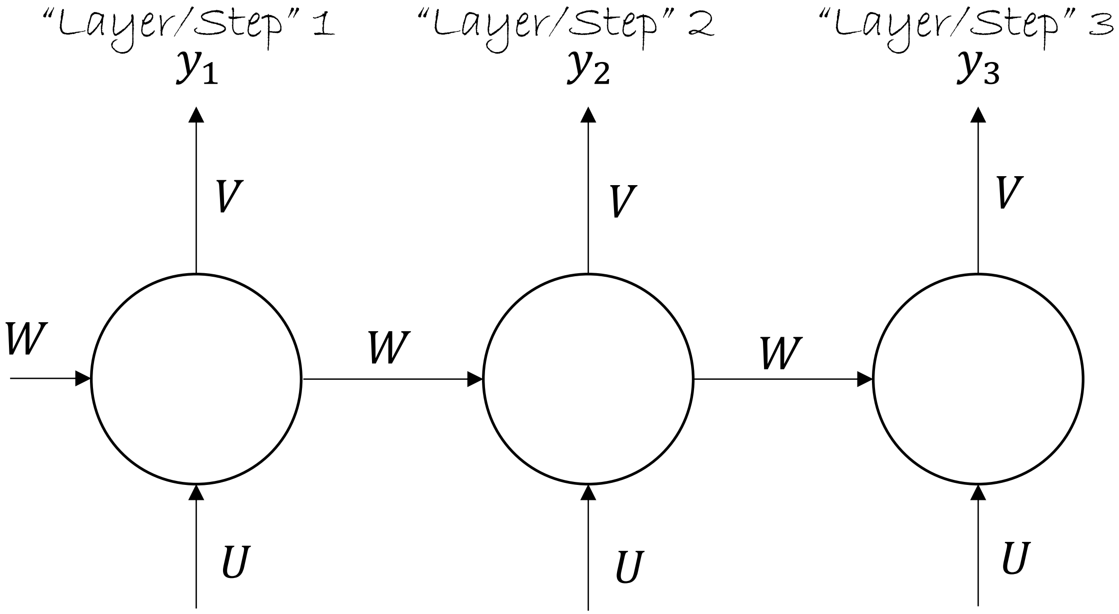


Folding the memory

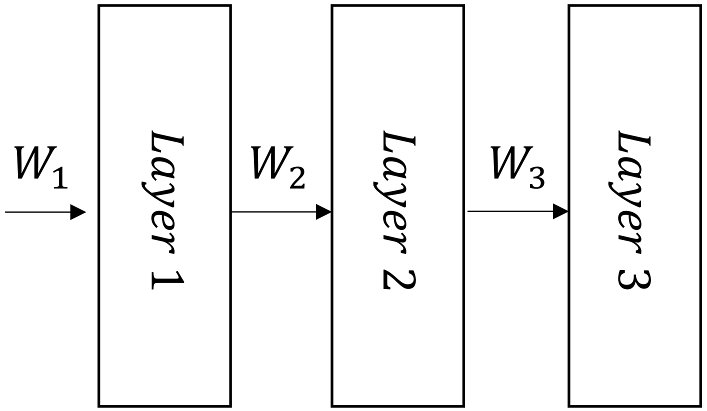


RNN vs NN

- What is really different?
  - Steps instead of layers
  - Step parameters shared whereas in a Multi-Layer Network different



3-gram Unrolled Recurrent Network



3-layer Neural Network

Training an RNN

- Cross-entropy loss

$$P = \prod_{t,k} y_{tk}^{l_{tk}} \quad \Rightarrow \quad \mathcal{L} = -\log P = \sum_t \mathcal{L}_t = -\frac{1}{T}\sum_t l_t \log y_t$$

- Backpropagation Through Time (BPTT)
- Be careful of the recursion. The non-linearity is influencing itself. The gradients at one time step depends on gradients on previous time steps
  - Like in NN → Chain Rule
  - Only difference: Gradients survive over time steps

RNN Gradients

$$\mathcal{L} = L(c_T(c_{T-1}(\dots(c_1(x_1, c_0; W); W); W); W)$$

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{\mathrm{t}} \frac{\partial \mathcal{L}_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$$

$$\frac{\partial \mathcal{L}}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} = \frac{\partial \mathcal{L}}{\partial \mathrm{c_t}} \cdot \frac{\partial c_t}{\partial \mathrm{c_{t-1}}} \cdot \frac{\partial c_{t-1}}{\partial \mathrm{c_{t-2}}} \cdot \dots \cdot \frac{\partial c_{\tau+1}}{\partial \mathrm{c_\tau}} \leq \eta^{t-\tau} \frac{\partial \mathcal{L}_t}{\partial c_t}$$

- The RNN gradient is a recursive product of $\frac{\partial c_t}{\partial \mathrm{c_{t-1}}}$

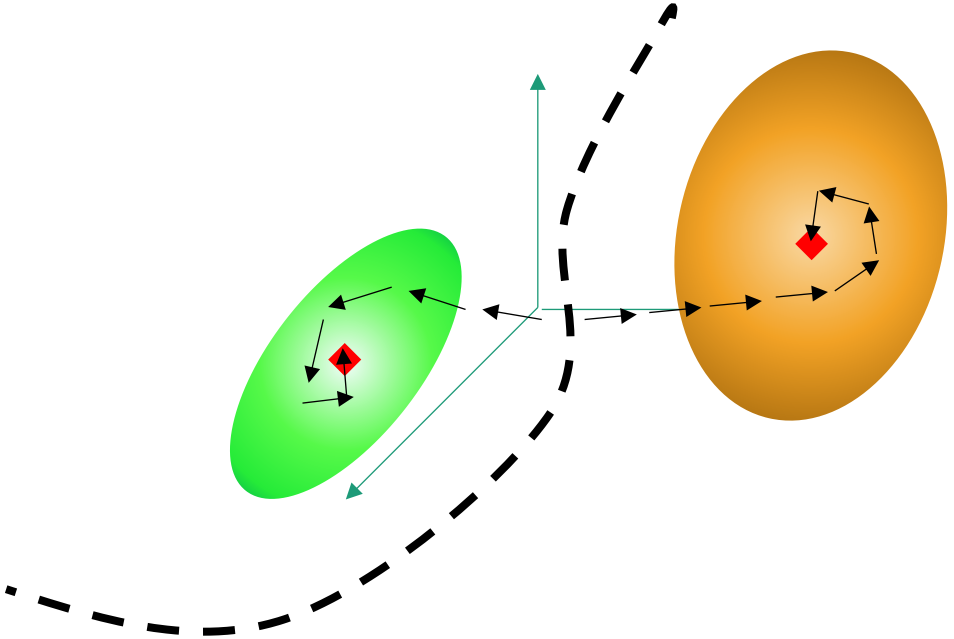
Vanishing/Exploding gradients

- $$\underbrace{\frac{\partial \mathcal{L}}{\partial c_t} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathrm{c_T}}}_{<1} \cdot \underbrace{\frac{\partial c_T}{\partial \mathrm{c_{T-1}}}}_{<1} \cdot \frac{\partial c_{T-1}}{\partial \mathrm{c_{T-2}}} \cdot \ldots \cdot \underbrace{\frac{\partial c_{t+1}}{\partial \mathrm{c}_{c_t}}}_{<1}} \quad \frac{\partial \mathcal{L}}{\partial W} \ll 1 \Longrightarrow \text{Vanishing gradient}$$

- $$\frac{\partial \mathcal{L}}{\partial c_t} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathrm{c_T}}}_{>1} \cdot \underbrace{\frac{\partial c_T}{\partial \mathrm{c_{T-1}}}}_{>1} \cdot \frac{\partial c_{T-1}}{\partial \mathrm{c_{T-2}}} \cdot \ldots \cdot \underbrace{\frac{\partial c_1}{\partial \mathrm{c}_{c_t}}}_{>1} \quad \frac{\partial \mathcal{L}}{\partial W} \gg 1 \Longrightarrow \text{Exploding gradient}$$

RNN & Chaotic Systems

- The latent memory space is composed of multiple dimensions
- A subspace of the memory state space can store information if multiple basins ◆ of attraction in some dimensions exist
- Gradients must be strong near the basin boundaries



RNN & Chaotic Systems

- In the figures $\mathrm{x_t} \propto c_t$ and $x_t \propto \mathrm{F}(Wx_{t-1} + Uu_t + b)$

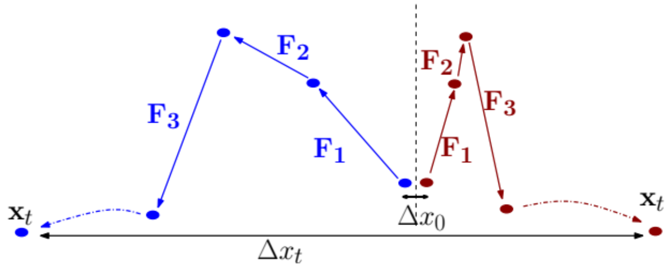


Figure 4. This diagram illustrates how the change in $\mathbf{x}_t$, $\Delta\mathbf{x}_t$, can be large for a small $\Delta\mathbf{x}_0$. The blue vs red (left vs right) trajectories are generated by the same maps $F_1, F_2, \ldots$ for two different initial states.

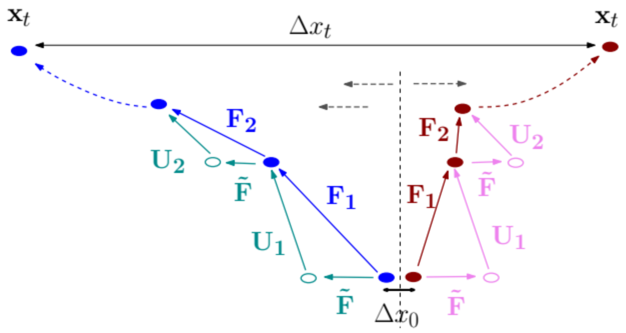


Figure 5. Illustrates how one can break apart the maps $F_1, ..F_t$ into a constant map $\tilde{F}$ and the maps $U_1, .., U_t$. The dotted vertical line represents the boundary between basins of attraction, and the straight dashed arrow the direction of the map $\tilde{F}$ on each side of the boundary. This diagram is an extension of Fig. 4.

Figures from: Glorot and Bengio, Understanding the difficulty of training deep feedforward neural networks, 2010

Advanced RNN: LSTM

- $\sigma \in (0, 1)$: control gate – something like a switch
- $\tanh \in (-1, 1)$: recurrent nonlinearity

$$i = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

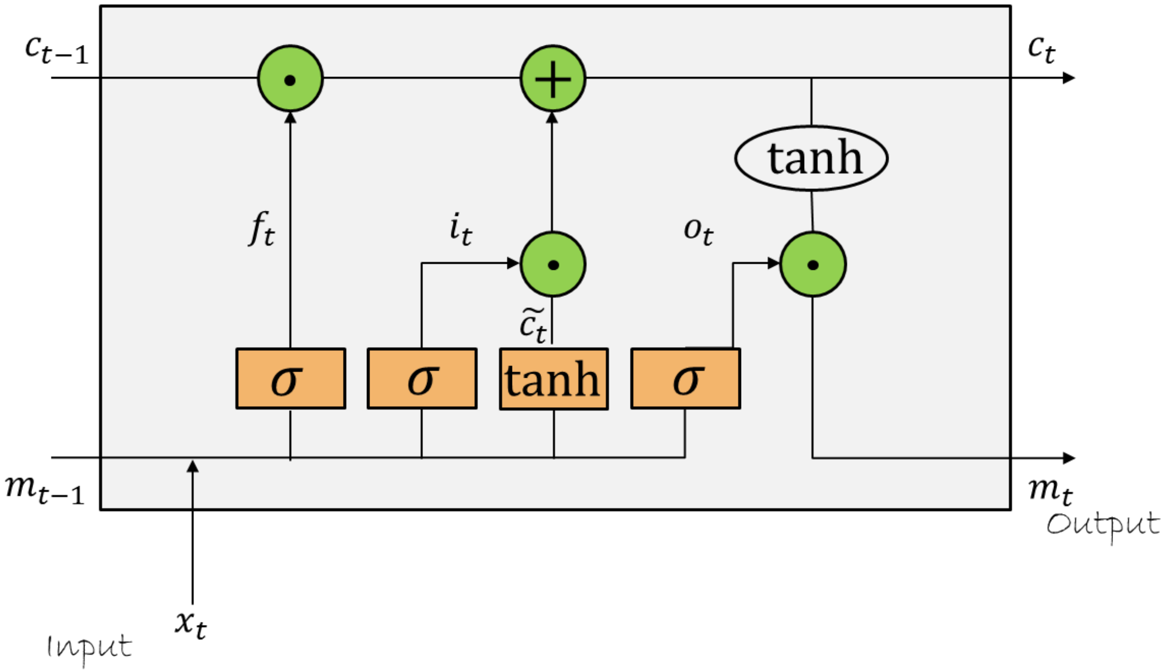
$$f = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c}_t = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \widetilde{c}_t \odot i$$

$$m_t = \tanh(c_t) \odot o$$



Take away message

- Deep Learning is good not only for classifying things
- Structured prediction is also possible
- Multi-task structure prediction allows for unified networks
- Discovering structure in data is also possible
- Training neural networks with sequences with recurrent nets