



THE UNIVERSITY  
*of* ADELAIDE

# Learning Deep Structured Models for Semantic Segmentation

Guosheng Lin

# Semantic Segmentation



**(d)** Testing

**(e)** Truth

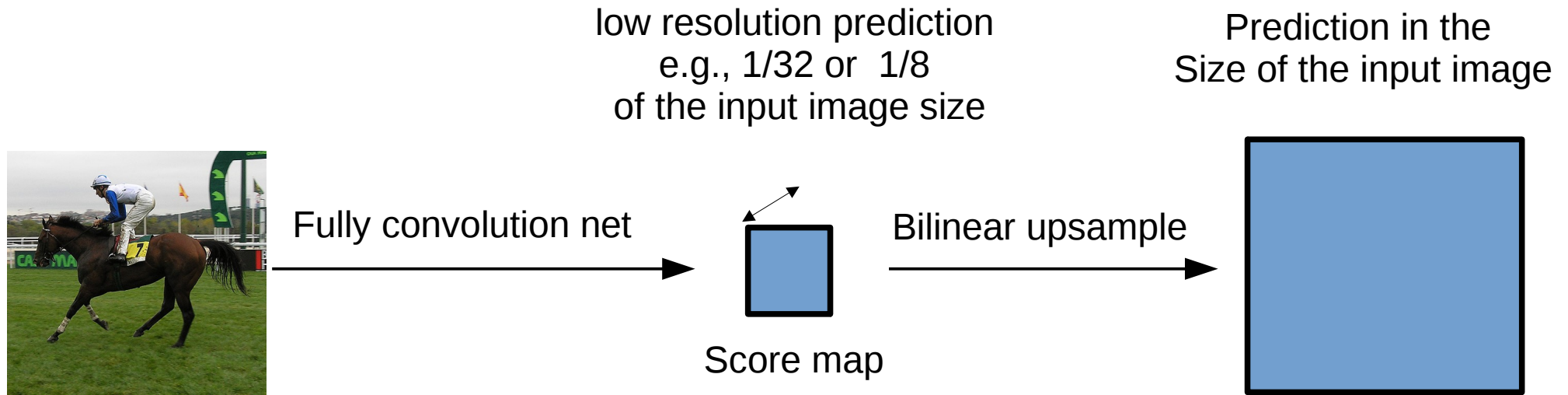
**(f)** Predict

# Outline

- Exploring Context with Deep Structured Models
  - Guosheng Lin, Chunhua Shen, Ian Reid, Anton van den Hengel; Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation; arXiv.
- Learning CNN based Message Estimators
  - Guosheng Lin, Chunhua Shen, Ian Reid, Anton van den Hengel; Deeply Learning the Messages in Message Passing Inference; NIPS 2015.

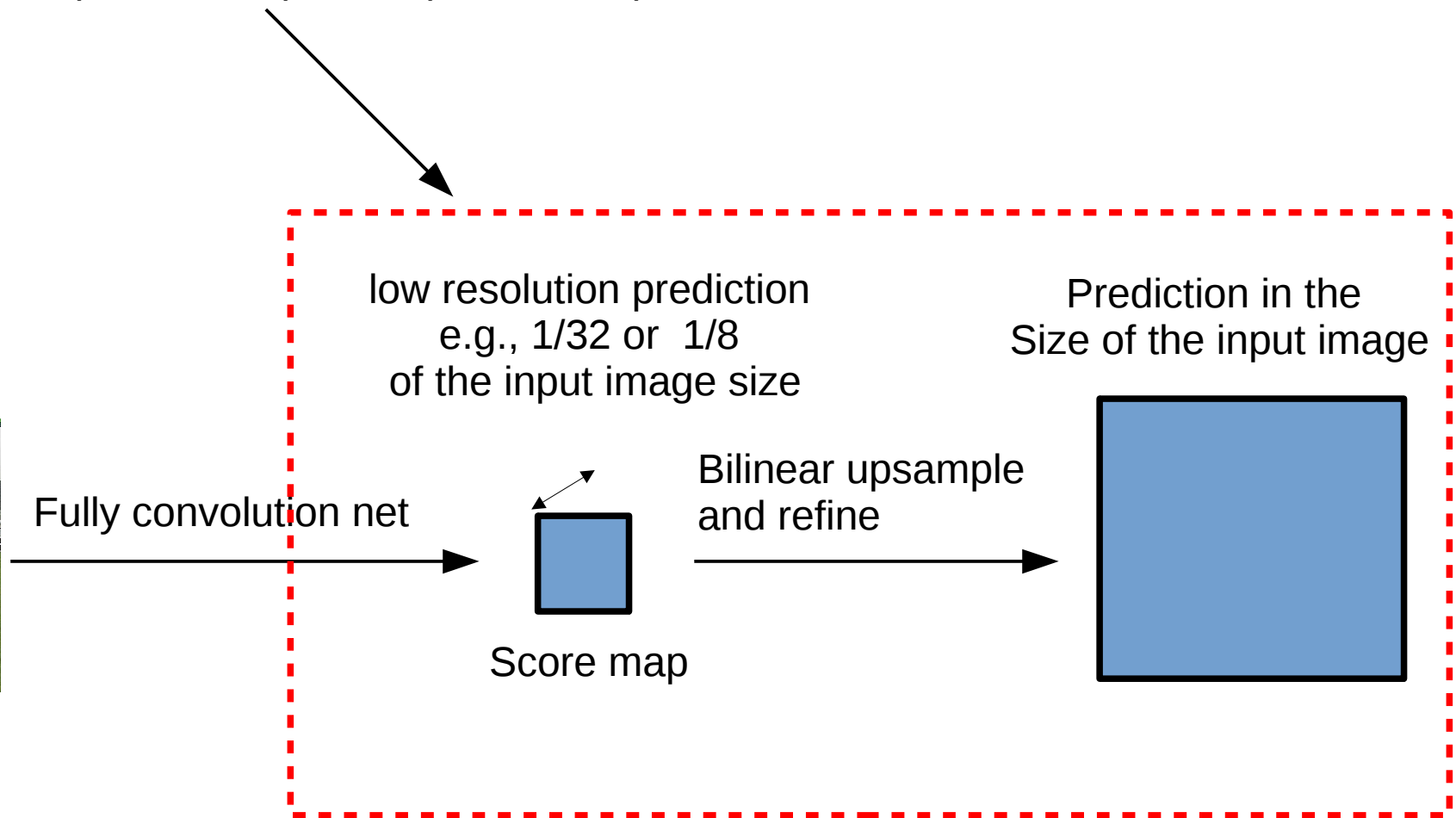
# Background

- Fully convolution network for semantic segmentation
  - Long et al. CVPR2015



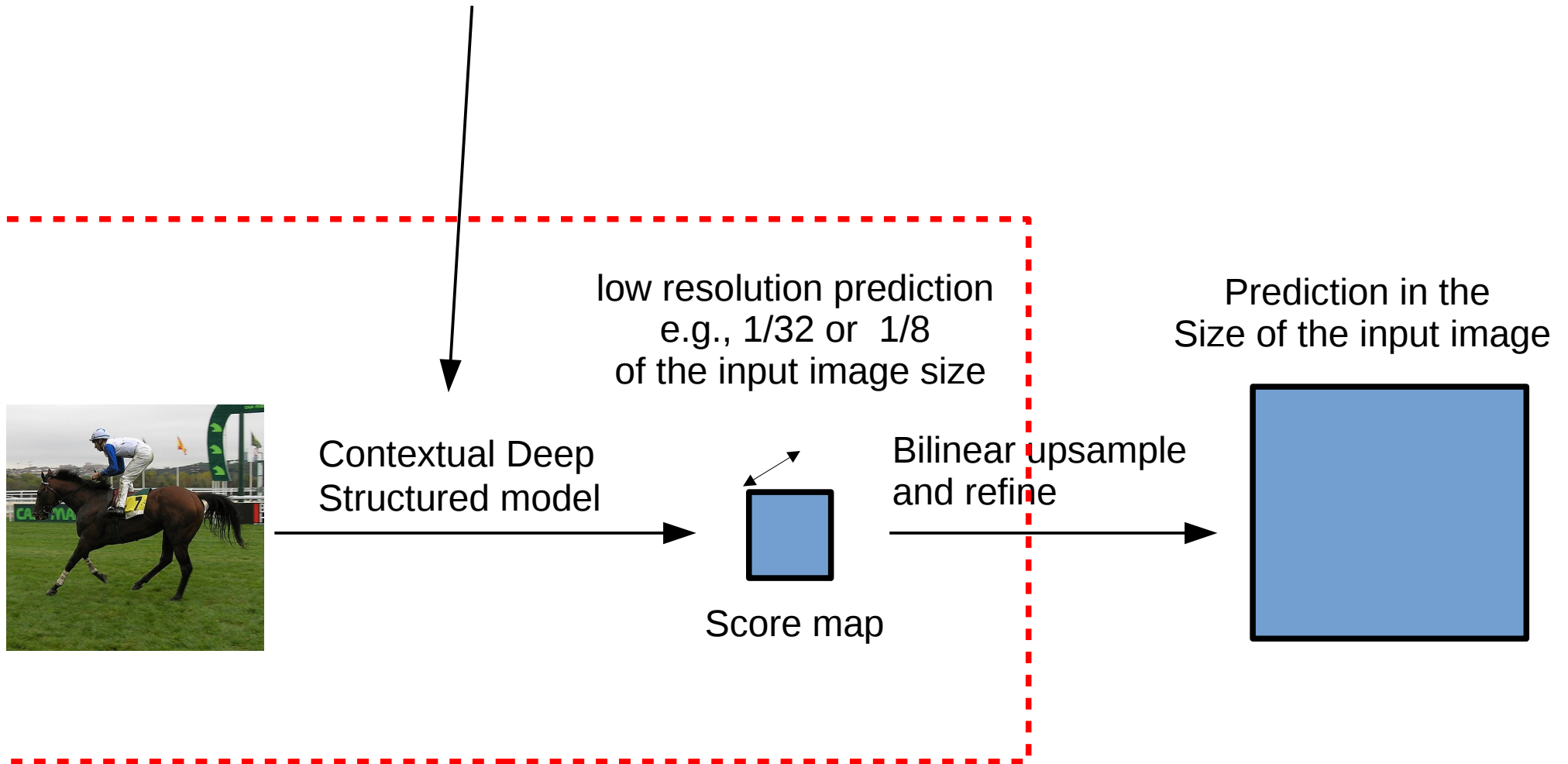
# Background

Recent methods focus on the up-sample and refinement stage.  
e.g., DeepLab (ICLR 2015), CRF-RNN (ICCV 2015),  
DeconvNet (ICCV 2015), DPN (ICCV 2015)



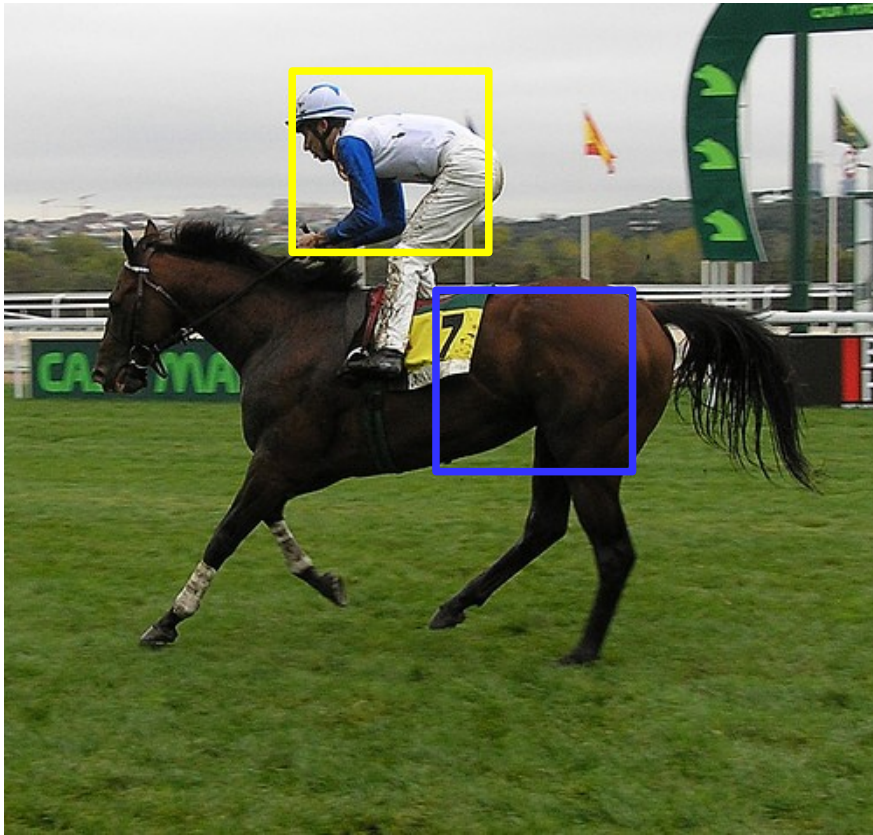
# Background

Our focus: explore contextual information using deep structured model

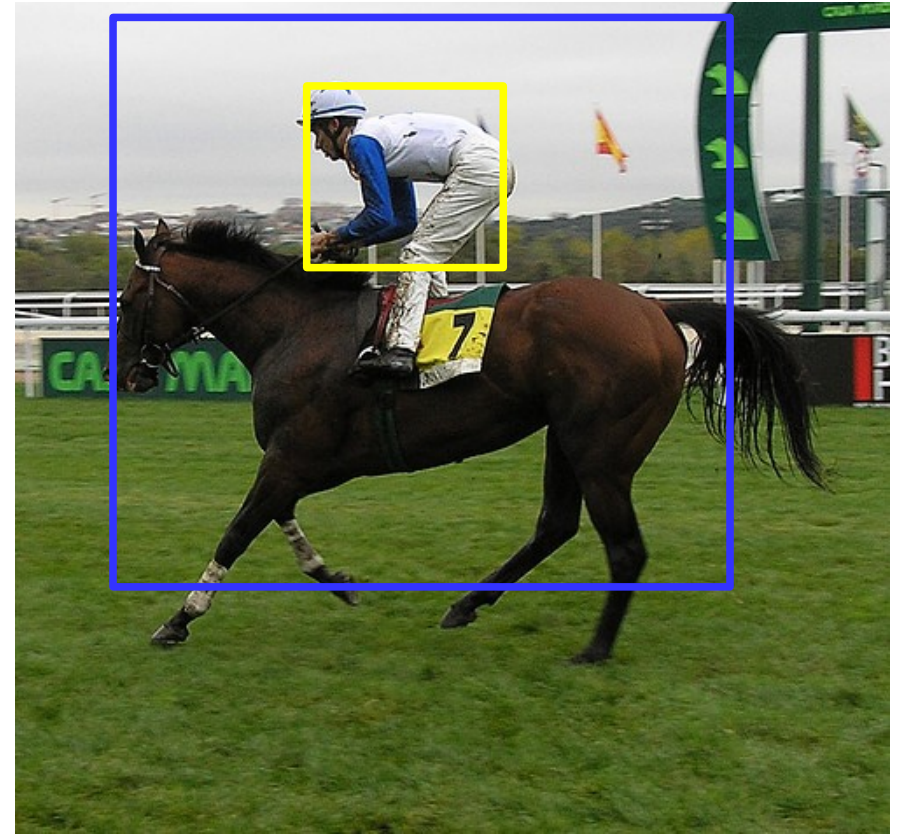


# Explore Context

- Spatial Context:
  - Semantic relations between image regions.
    - e.g., a car is likely to appear over a road
    - A person appears above a horse is more likely than a dog appears above a horse.
  - We focus on two types of context:
    - Patch-Patch context
    - Patch-Background context

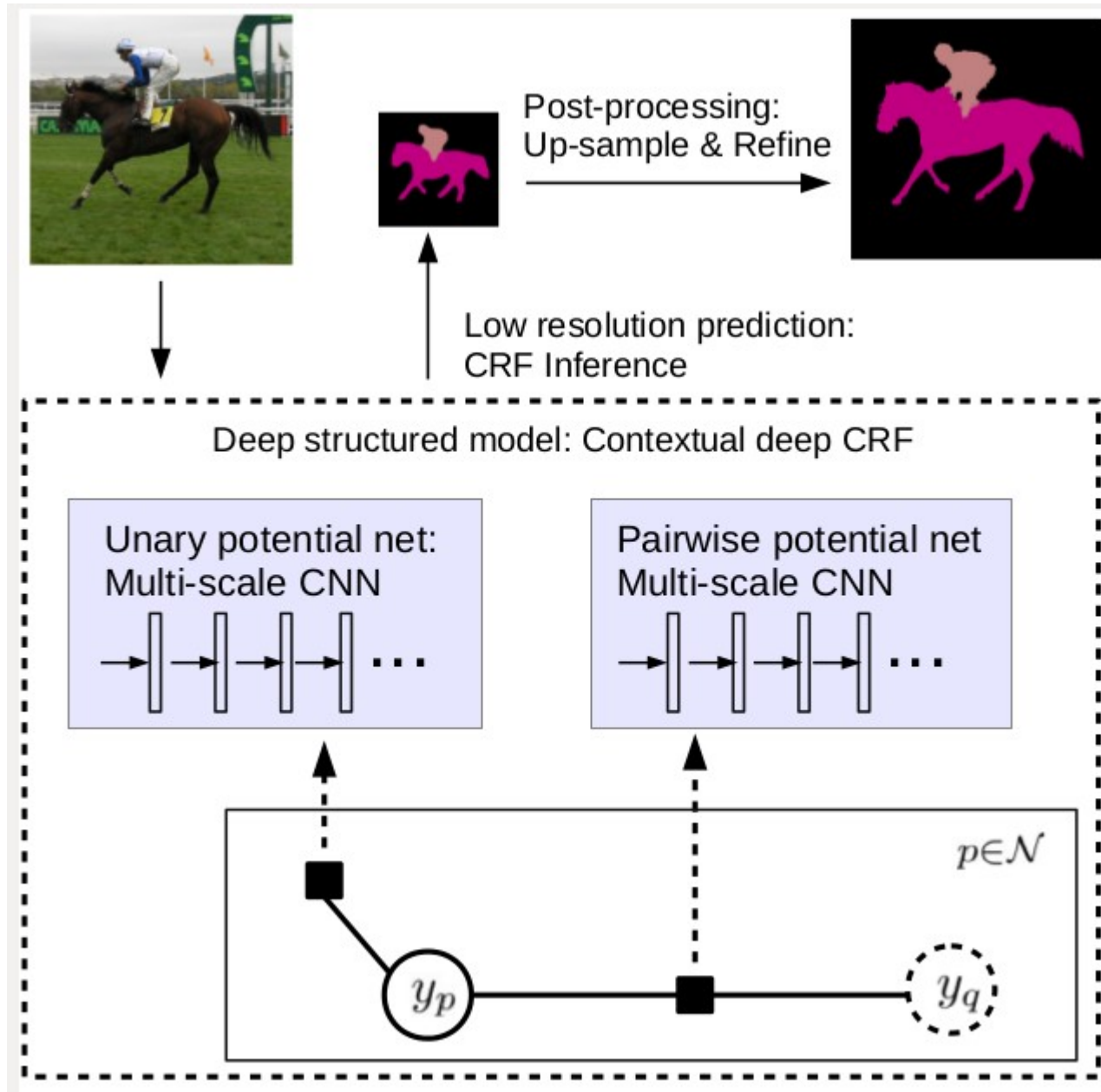


Patch-Patch Context



Patch-Background Context

# Overview

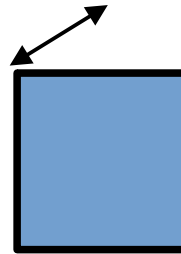


# Patch-Patch Context

- Learning CRFs with CNN based pairwise potential functions.



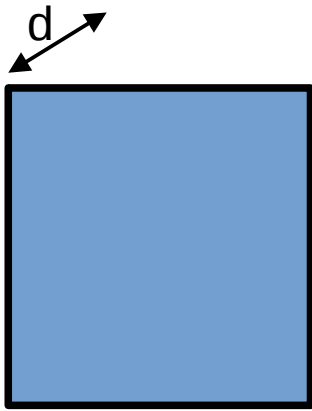
FeatMap-Net



Feature map



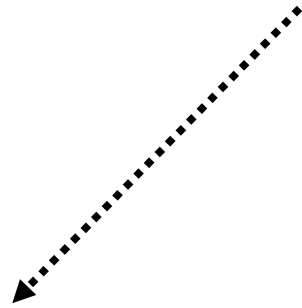
Create the CRF graph  
(create nodes and  
pairwise connections)



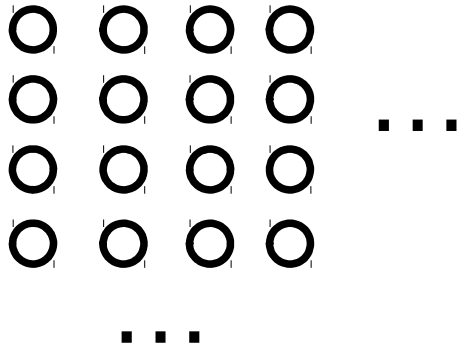
Feature map



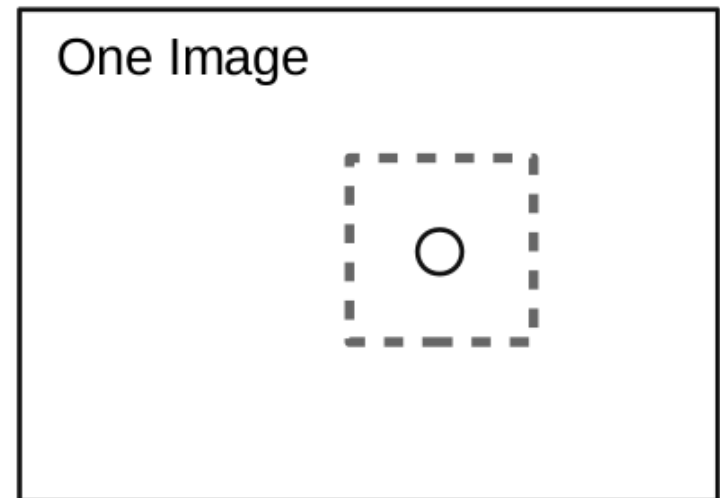
Create the CRF graph  
(create nodes and pairwise connections)



Create nodes in the CRF graph  
One node corresponds to one  
spatial position in the feature map

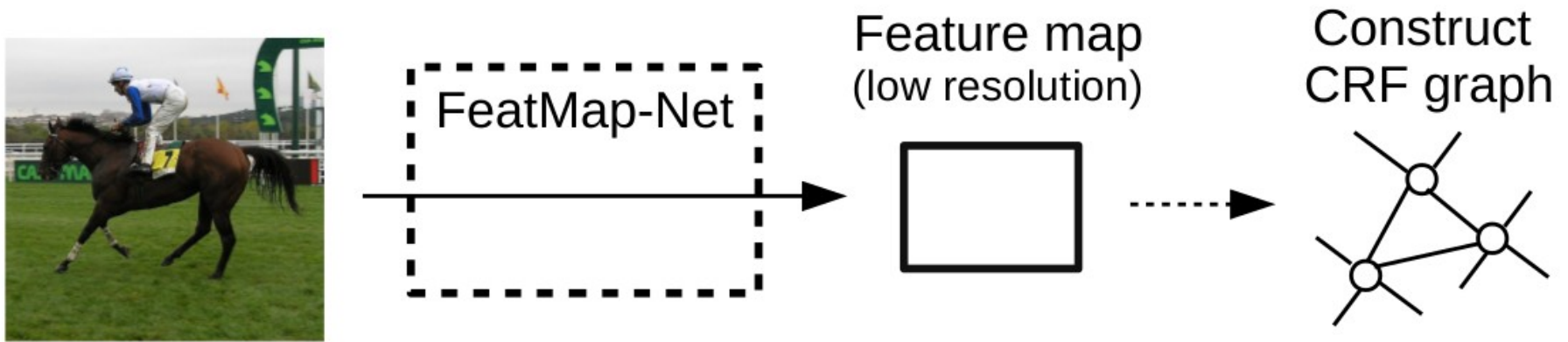


Generate pairwise connection  
One node connects to the nodes  
that lie in a spatial range box  
(box with the dashed lines)

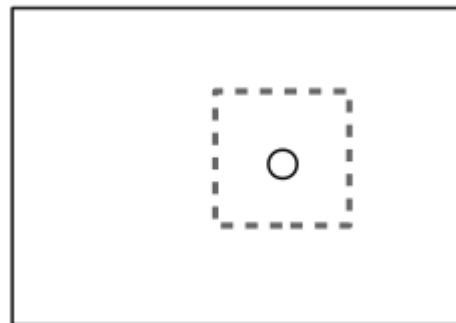


# Patch-Patch Context

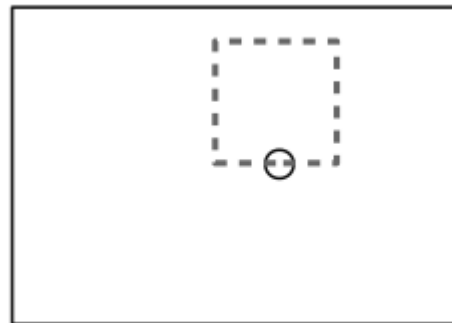
- Construct CRF graph



Constructing pairwise connections in a CRF graph:



Surrounding

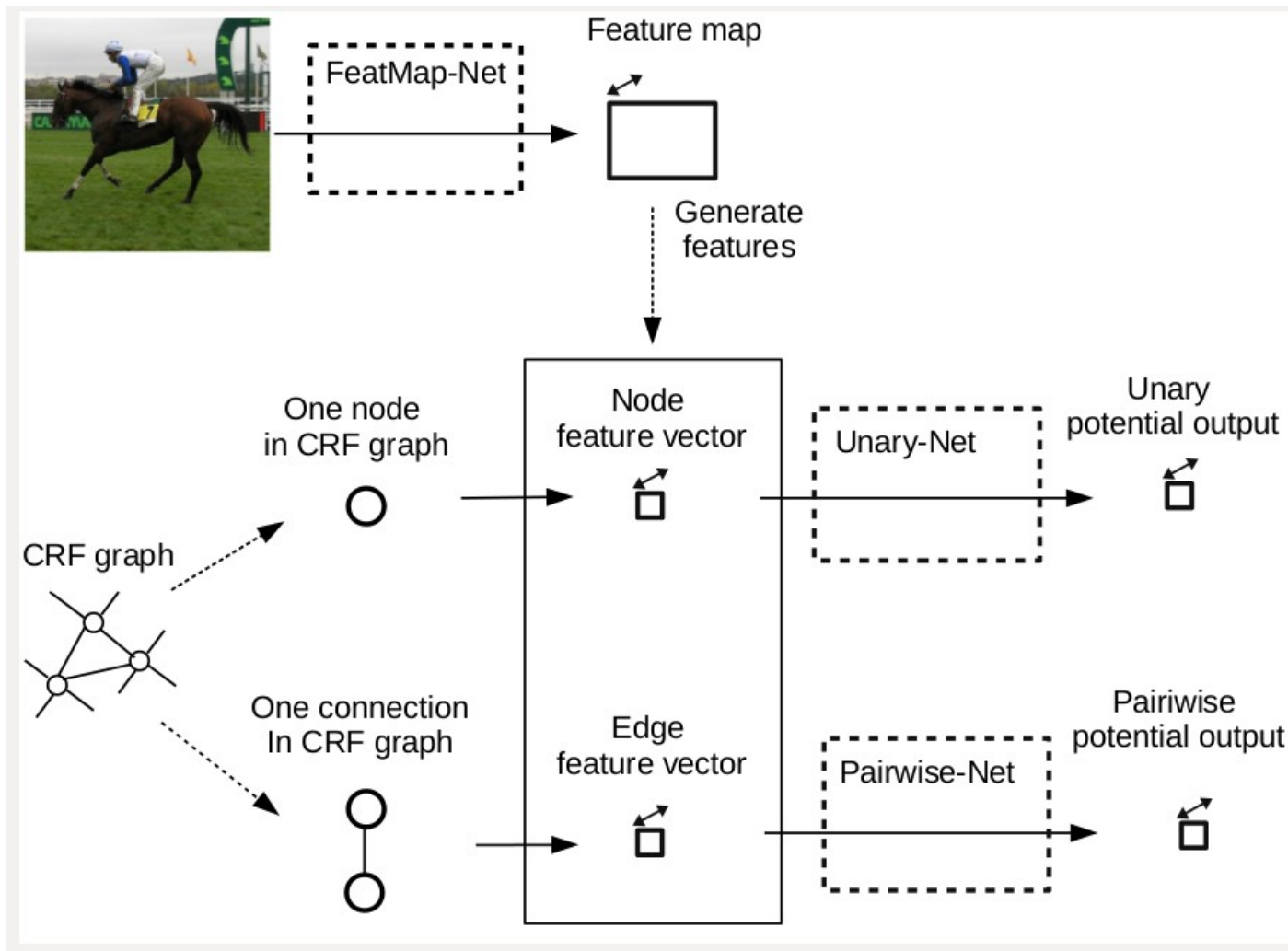


Above/Below

# CRFs with CNN based potentials

The conditional likelihood for one image:  $P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp[-E(\mathbf{y}, \mathbf{x})]$ .

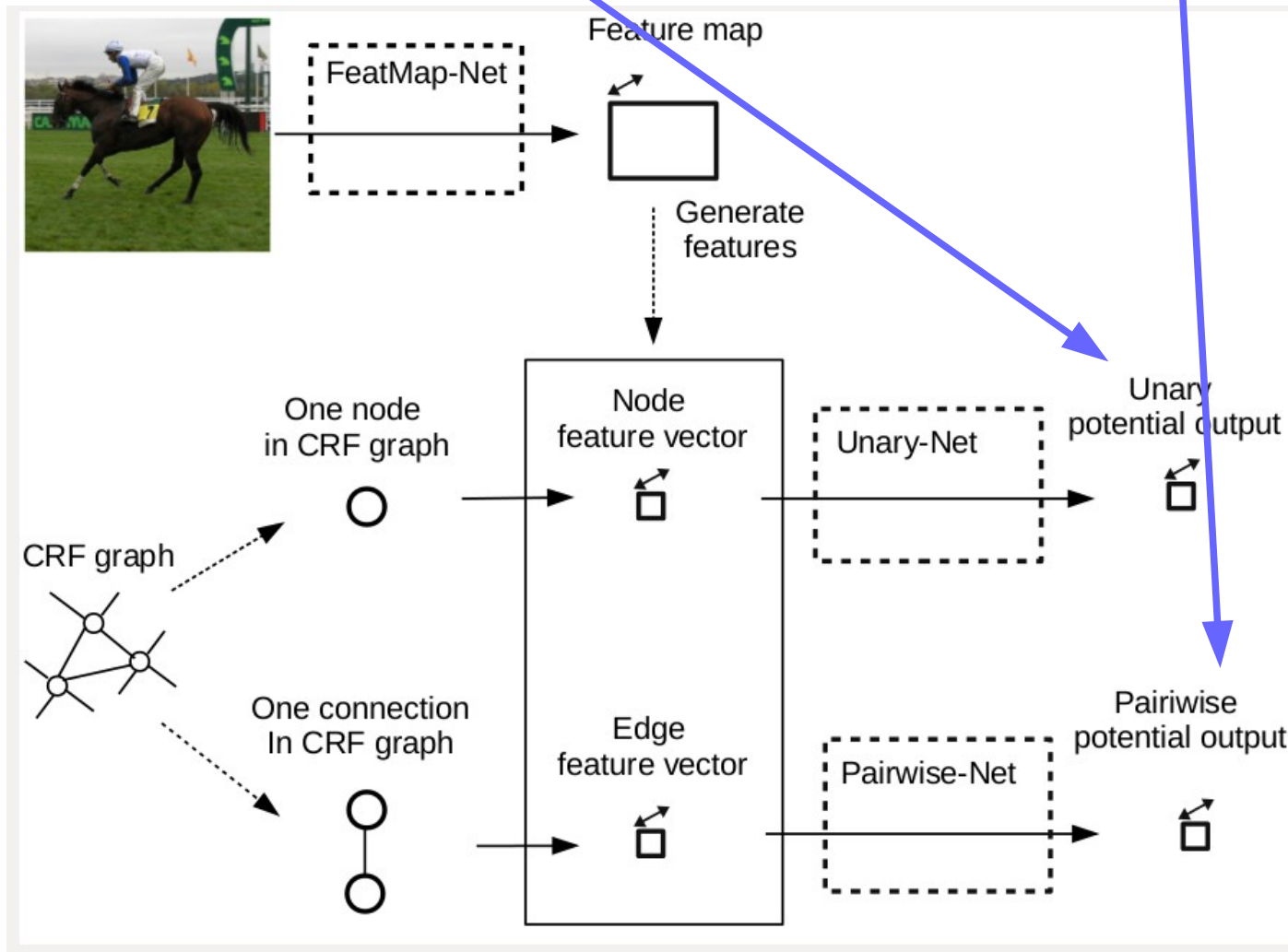
$$E(\mathbf{y}, \mathbf{x}) = \sum_{U \in \mathcal{U}} \sum_{p \in \mathcal{N}_U} U(y_p, \mathbf{x}_p) + \sum_{V \in \mathcal{V}} \sum_{(p,q) \in \mathcal{S}_V} V(y_p, y_q, \mathbf{x}_{pq}).$$



# CRFs with CNN based potentials

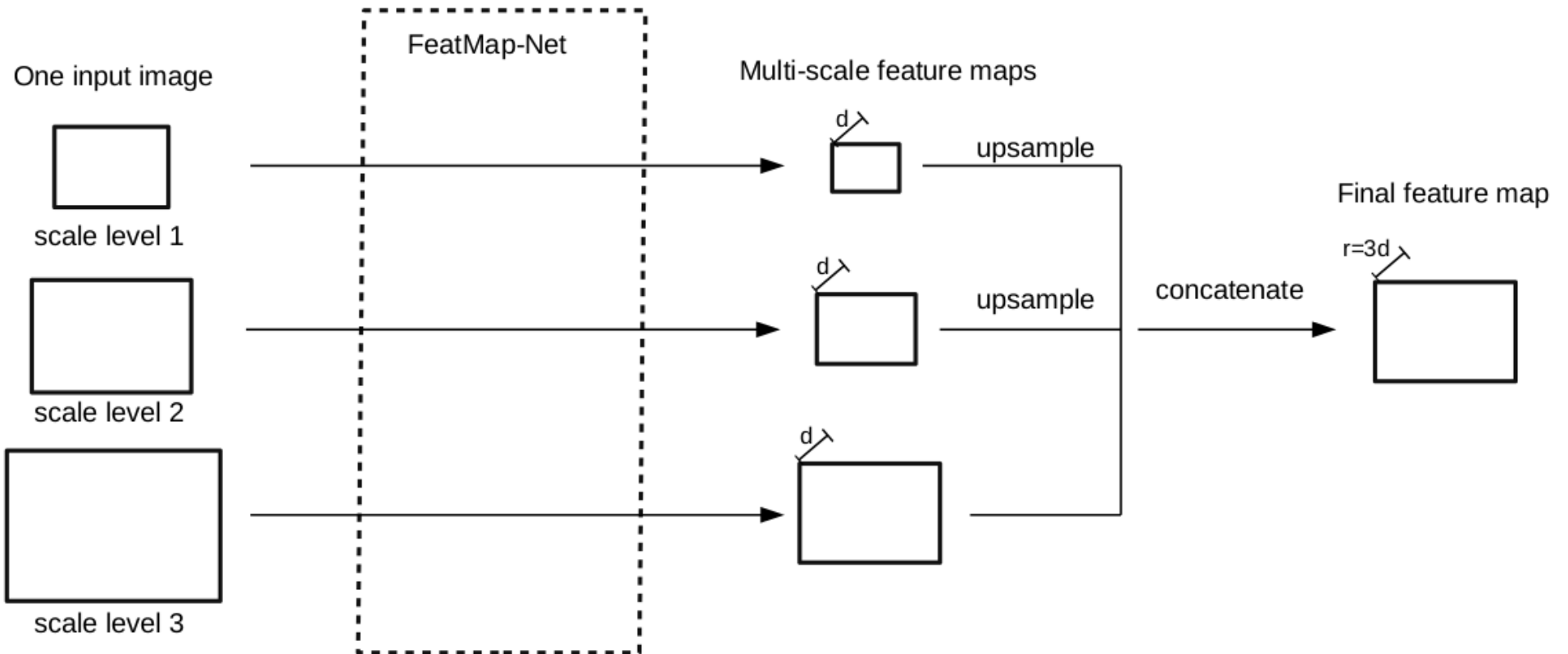
The conditional likelihood for one image:  $P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp[-E(\mathbf{y}, \mathbf{x})]$ .

$$E(\mathbf{y}, \mathbf{x}) = \sum_{U \in \mathcal{U}} \sum_{p \in \mathcal{N}_U} U(y_p, \mathbf{x}_p) + \sum_{V \in \mathcal{V}} \sum_{(p,q) \in \mathcal{S}_V} V(y_p, y_q, \mathbf{x}_{pq}).$$



# Explore background context

FeatMap-Net: multi-scale network for generating feature map



## FeatMap-Net

Conv block 1:

3 x 3 conv 64  
3 x 3 conv 64  
2 x 2 pooling

Conv block 2:

3 x 3 conv 128  
3 x 3 conv 128  
2 x 2 pooling

Conv block 3:

3 x 3 conv 256  
3 x 3 conv 256  
3 x 3 conv 256  
2 x 2 pooling

Conv block 4:

3 x 3 conv 512  
3 x 3 conv 512  
3 x 3 conv 512  
2 x 2 pooling

Conv block 5:

3 x 3 conv 512  
3 x 3 conv 512  
3 x 3 conv 512  
2 x 2 pooling  
7 x 7 conv 4096

Conv block 6:

3 x 3 conv 512  
3 x 3 conv 512

## Unary-Net

2 fully-connected layers:

Fully-con 512  
Fully-con K

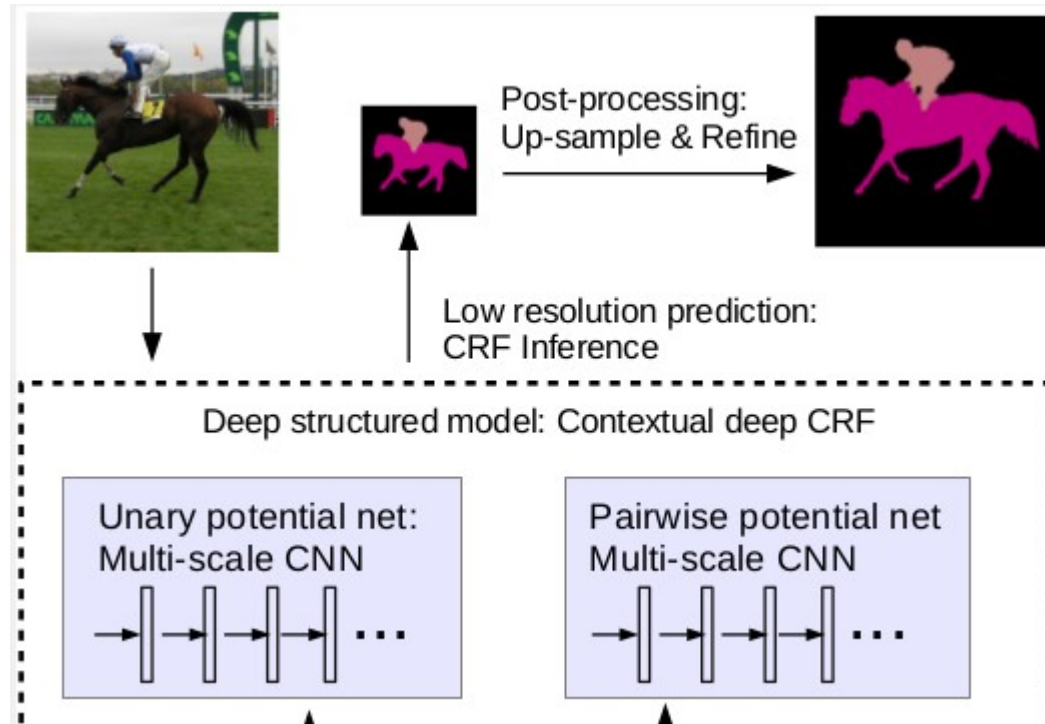
## Pairwise-Net

2 fully-connected layers:

Fully-con 512  
Fully-con  $K^2$

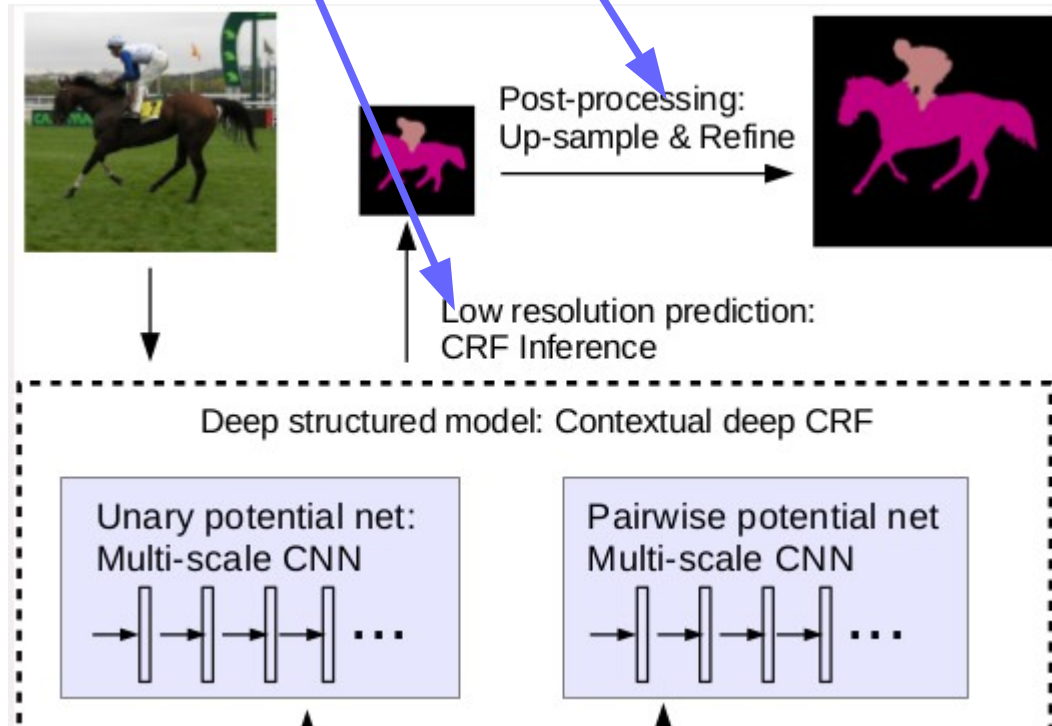
# Prediction

- Coarse-level prediction stage:  $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$ .
  - $P(\mathbf{y}|\mathbf{x})$  is approximated using the mean-field algorithm
- Prediction refinement stage
  - Sharpen the object boundary by leveraging low-level pixel information for smoothness.
  - First up-sample the confidence map of the coarse prediction to the original input image size. Then perform Dense-CRF. (P. Krähenbühl and V. Koltun NIPS2012)



# Prediction

- Coarse-level prediction stage:  $y^* = \operatorname{argmax}_y P(y|x)$ .
  - $P(y|x)$  is approximated using the mean-field algorithm
- Prediction refinement stage
  - Sharpen the object boundary by leveraging low-level pixel information for smoothness.
  - First up-sample the confidence map of the coarse prediction to the original input image size. Then perform Dense-CRF. (P. Krähenbühl and V. Koltun NIPS2012)



# CRF learning

Minimize the negative log-likelihood:

$$\min_{\boldsymbol{\theta}} \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 - \sum_{i=1}^N \log P(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}; \boldsymbol{\theta}).$$
$$\min_{\boldsymbol{\theta}} \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 + \sum_{i=1}^N \left[ E(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}; \boldsymbol{\theta}) + \log Z(\mathbf{x}^{(i)}; \boldsymbol{\theta}) \right].$$

SGD optimization, difficulty in calculating the gradient of the partition function:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log Z(\mathbf{x}; \boldsymbol{\theta}) &= \sum_{\mathbf{y}} \frac{\exp[-E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta})]}{\sum_{\mathbf{y}'} \exp[-E(\mathbf{y}', \mathbf{x}; \boldsymbol{\theta})]} \nabla_{\boldsymbol{\theta}} [-E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta})] \\ &= -\mathbb{E}_{\mathbf{y} \sim P(\mathbf{y} | \mathbf{x}; \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}) \end{aligned}$$

Require marginal inference at each SGD.

Since the huge number of SGD iteration and large number of nodes, this approach is not practical or even intractable.

We apply piecewise training to avoid repeat inference at each SGD iteration.

# Results

Table 1. Segmentation results on the PASCAL VOC 2012 test set. our method achieves the best performance.

method	IoU (VOC trained)	IoU (VOC+COCO trained)
SDS [3]	51.6	-
FCN-8s [5]	62.2	-
Zoom-out [6]	69.6	-
DeepLab [1]	71.6	72.7
DeconvNet [7]	72.5	-
CRF-RNN [8]	72.0	74.7
BoxSup [2]	-	75.2
DPN [4]	74.1	77.5
ours	<b>75.3</b>	<b>77.8</b>

# PASCAL Leaderboard

▶ Adelaide_Context_CNN_CRF_COCO [?]	77.8
▶ CUHK_DPN_COCO [?]	77.5
▶ Adelaide_Context_CNN_CRF_COCO [?]	77.2
▶ Adelaide_Context_CNN_CRF_VOC [?]	75.3
▶ MSRA_BoxSup [?]	75.2
▶ POSTECH_DeconvNet_CRF_VOC [?]	74.8
▶ Oxford_TV_G_CRF_RNN_COCO [?]	74.7
▶ DeepLab-MSc-CRF-LargeFOV-COCO-CrossJoint [?]	73.9
▶ DeepLab-CRF-COCO-LargeFOV [?]	72.7
▶ POSTECH_EDeconvNet_CRF_VOC [?]	72.5
▶ Oxford_TV_G_CRF_RNN_VOC [?]	72.0
▶ DeepLab-MSc-CRF-LargeFOV [?]	71.6
▶ MSRA_BoxSup [?]	71.0
▶ DeepLab-CRF-COCO-Strong [?]	70.4
▶ DeepLab-CRF-LargeFOV [?]	70.3
▶ TTI_zoomout_v2 [?]	69.6

<http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=6>

Table 1. Segmentation results on NYUDv2 dataset (40 classes). We compare to a number of recent methods. Our method significantly outperforms existing methods.

method	training data	pixel accuracy	mean accuracy	IoU
Gupta et al. [15]	RGB-D	60.3	-	28.6
FCN-32s [30]	RGB	60.0	42.2	29.2
FCN-HHA [30]	RGB-D	65.4	46.1	34.0
ours-unary	RGB	65.4	47.2	34.6
ours	RGB	67.6	49.6	37.1

Table 3. Segmentation results on PASCAL-Context dataset (60 classes). Our method performs the best.

method	pixel accuracy	mean accuracy	IoU
O2P [2]	-	-	18.1
CFM [6]	-	-	34.4
FCN-8s [30]	65.9	46.5	35.1
BoxSup [5]	-	-	40.5
ours	<b>71.5</b>	<b>53.9</b>	<b>43.3</b>

Table 4. Segmentation results on SIFT-flow dataset (33 classes). Our method performs the best.

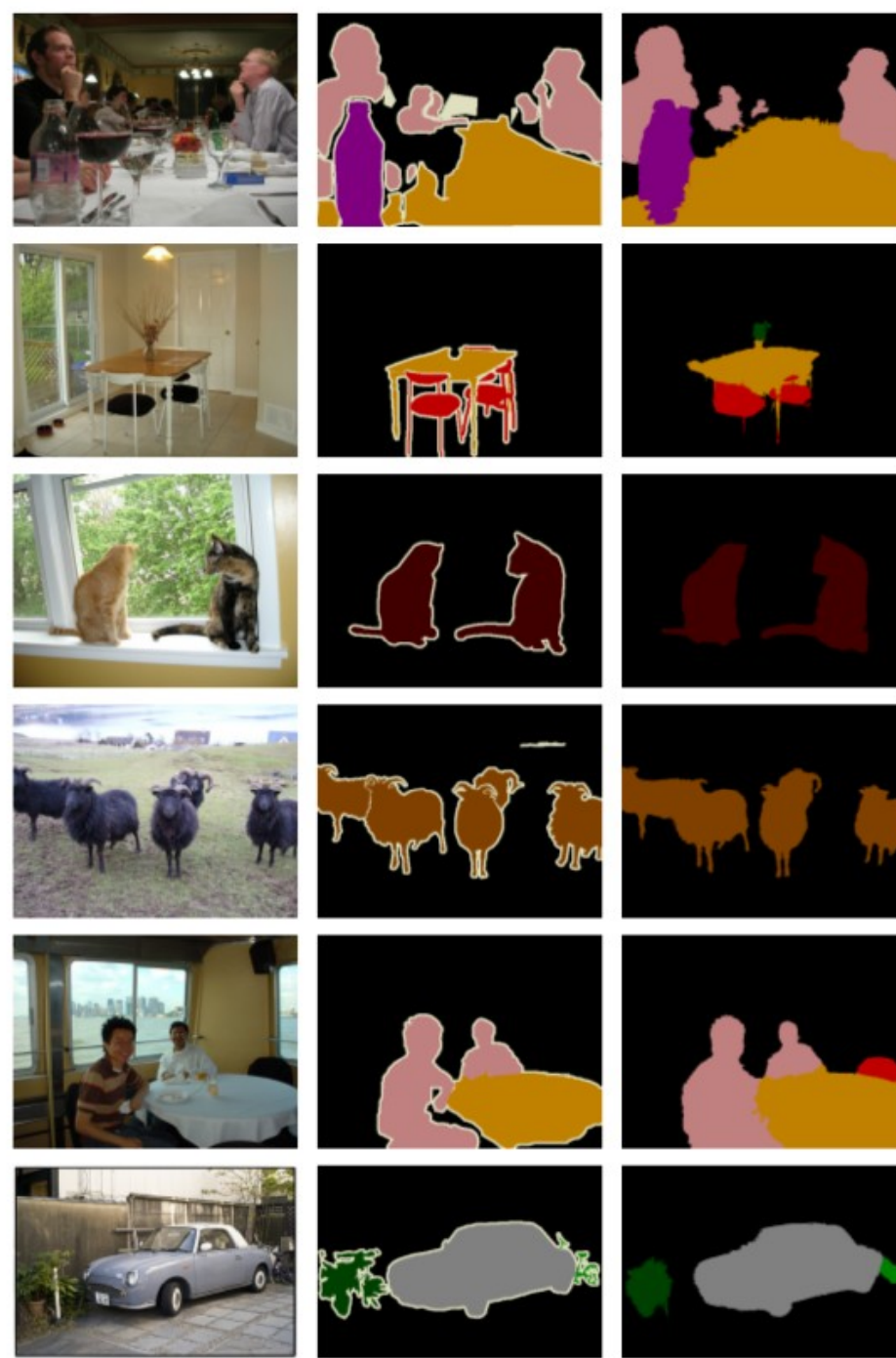
method	pixel accuracy	mean accuracy	IoU
Liu et al. [26]	76.7	-	-
Tighe et al. [41]	75.6	41.1	-
Tighe et al. (MRF) [41]	78.6	39.2	-
Farabet et al. (balance) [12]	72.3	50.8	-
Farabet et al. [12]	78.5	29.6	-
Pinheiro et al. [36]	77.7	29.8	-
FCN-16s [30]	85.2	51.7	39.5
ours	<b>88.1</b>	<b>53.4</b>	<b>44.9</b>



**(a)** Testing

**(b)** Truth

**(c)** Predict



**(d)** Testing

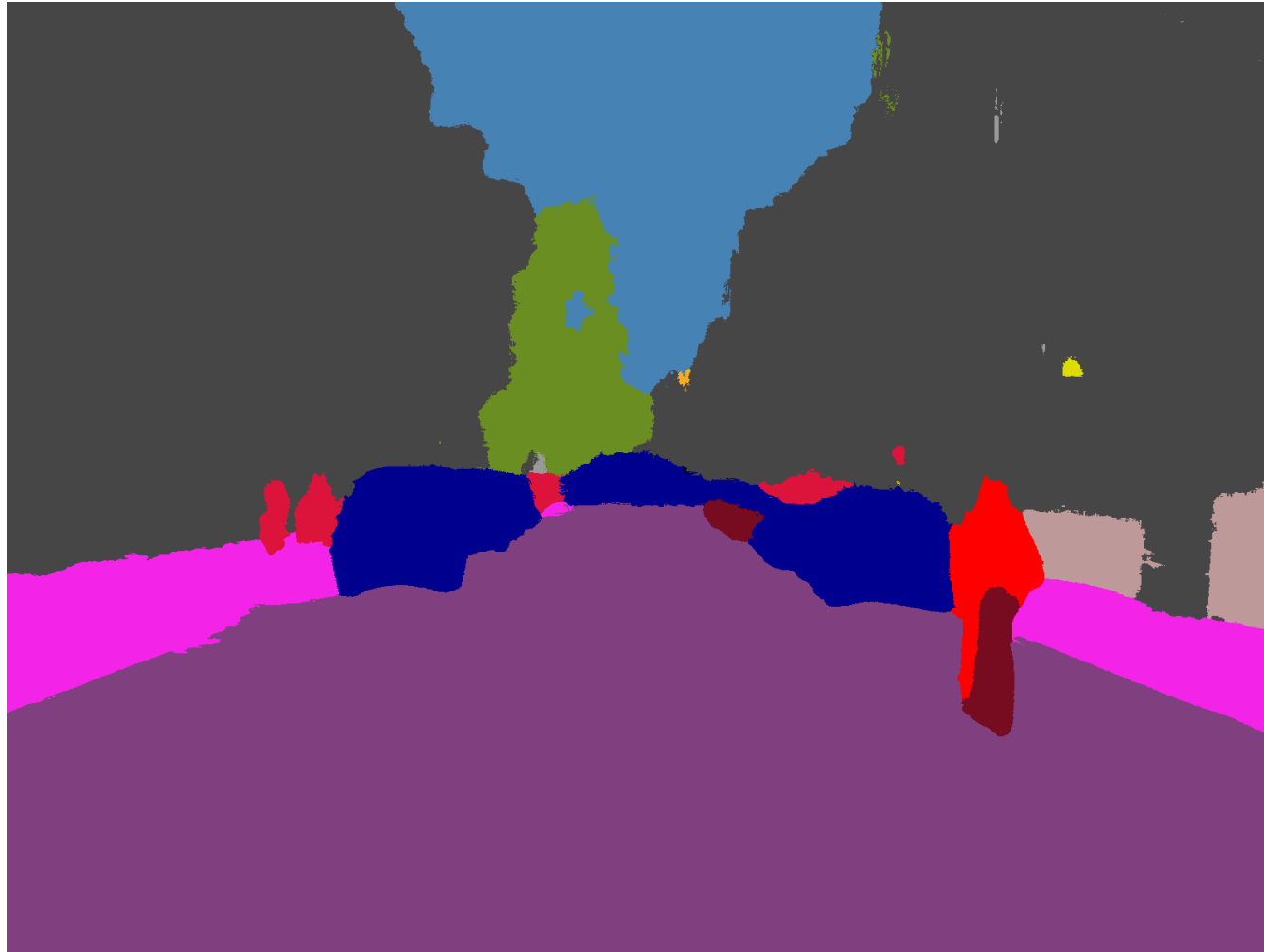
**(e)** Truth

**(f)** Predict

Examples on Internet images



Test image: street scene



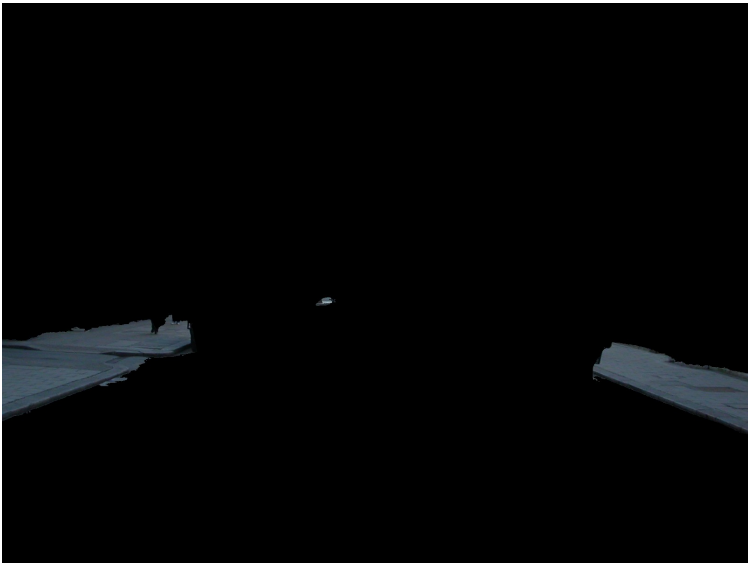
Result from a model trained on street scene images  
(around 1000 training images)



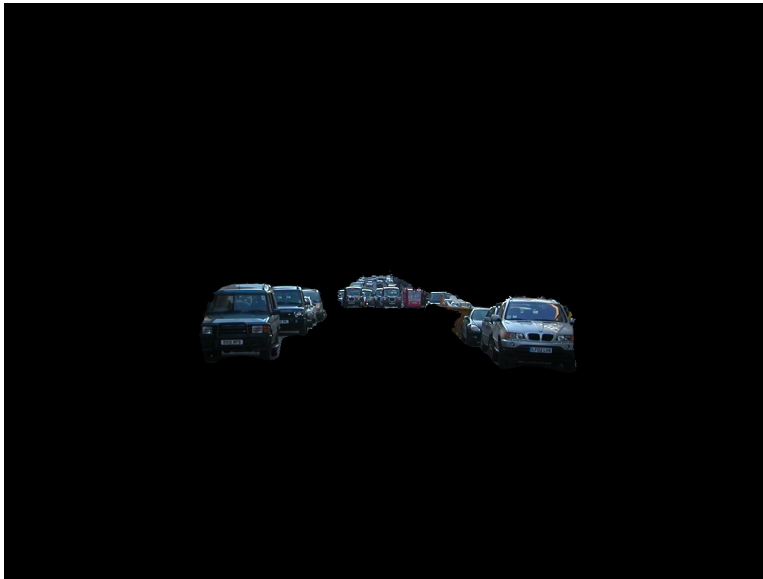
Building



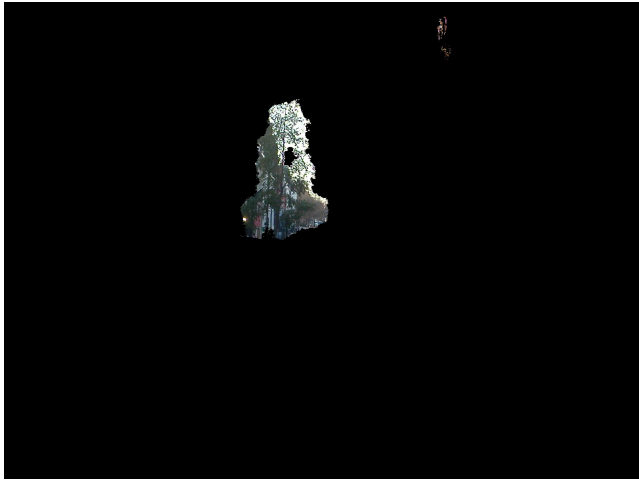
Road



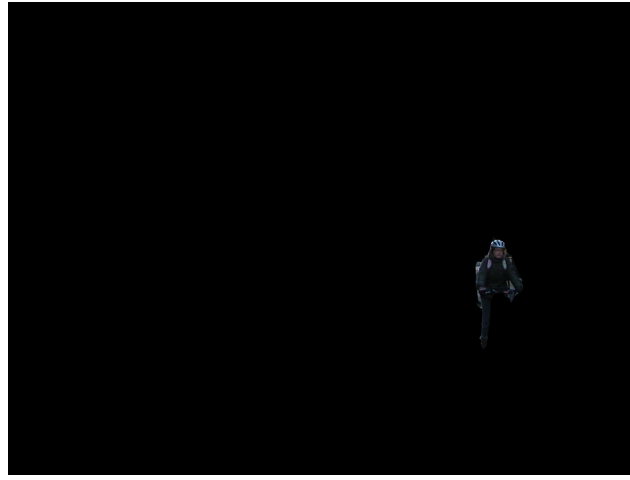
Side-walk



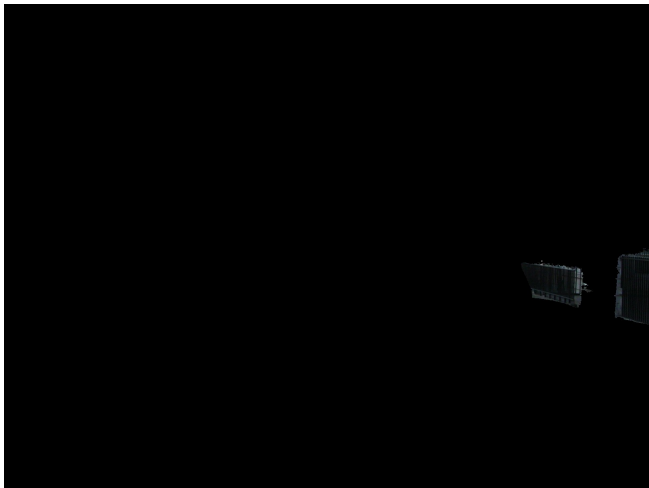
Car



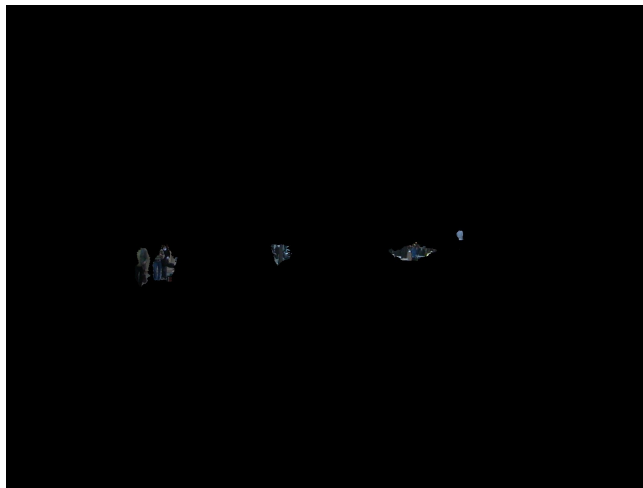
Tree



Rider



Fence



Person

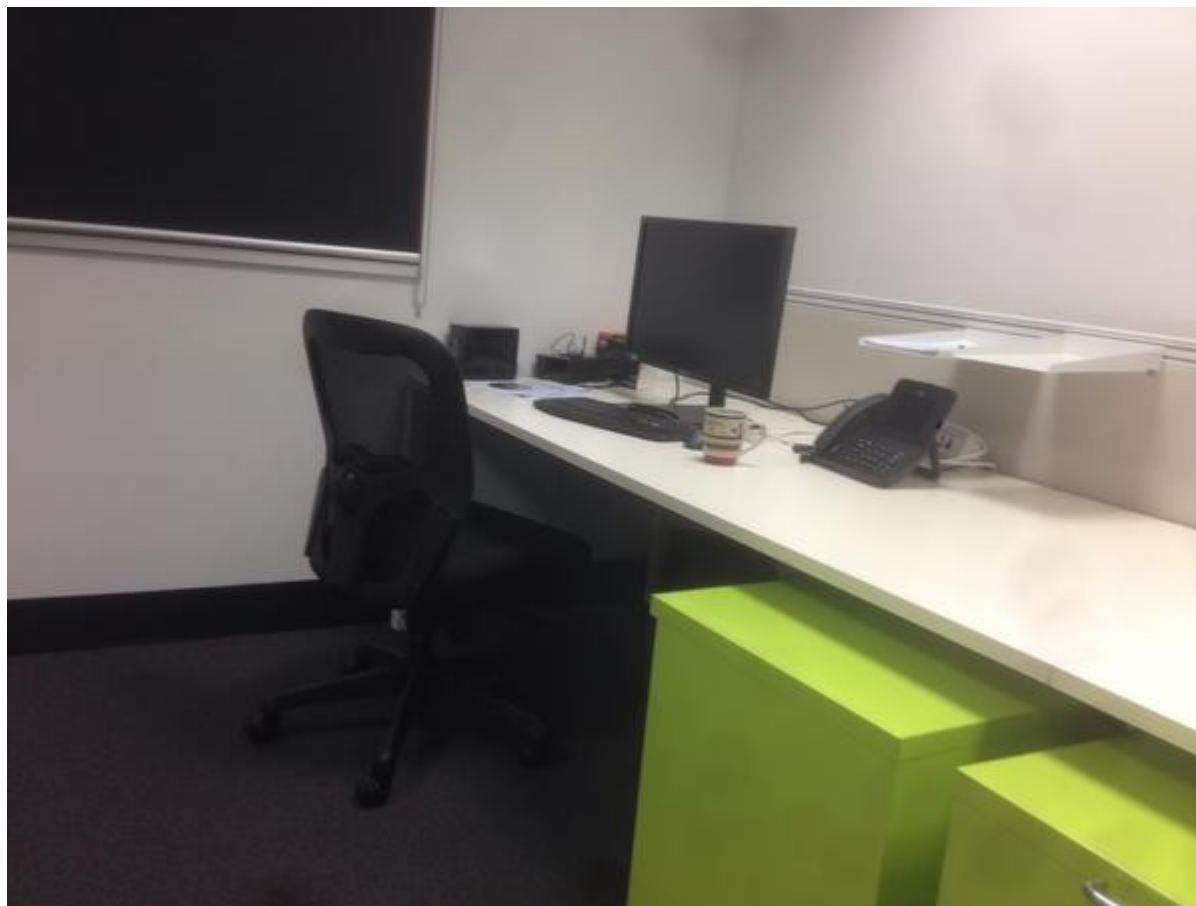




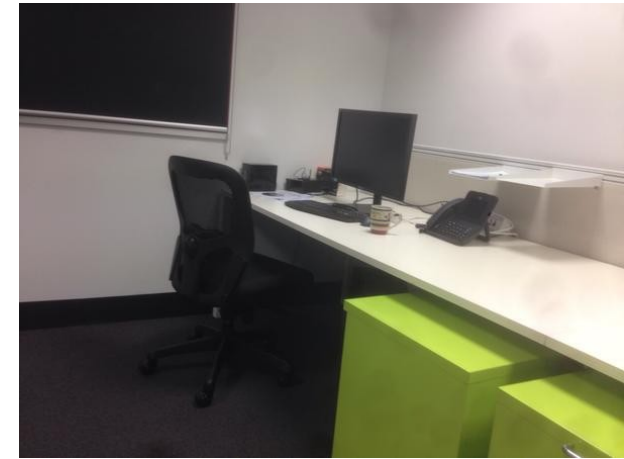
Result from a model trained on street scene images  
(around 1000 training images)



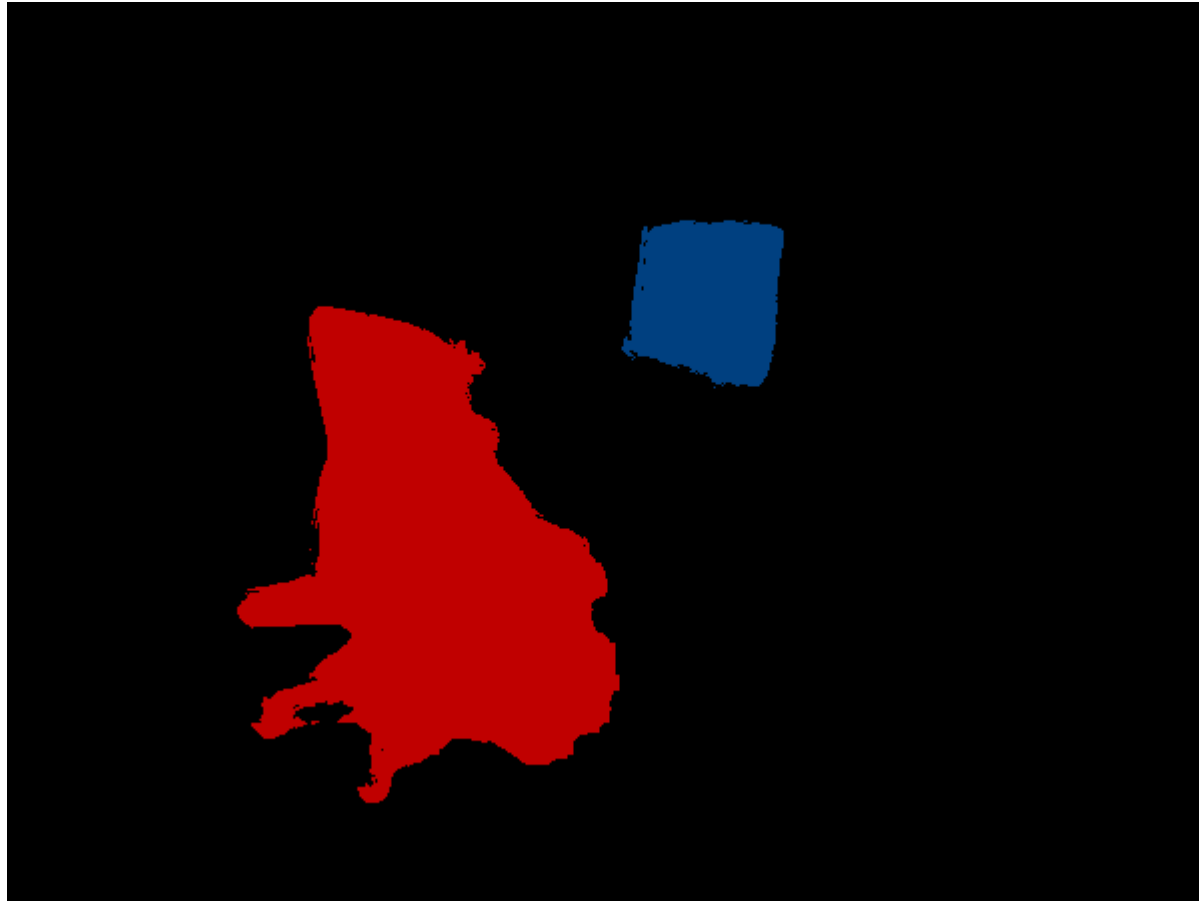
Result from PASCAL VOC model



Test image: indoor scene



Result from NYUD trained model (around 800 training images)



Result from PASCAL VOC trained model





Result from NYUD trained model

# Message Learning

# CRFs+CNNs

Conditional likelihood:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp[-E(\mathbf{y}, \mathbf{x})] = \frac{\exp[-E(\mathbf{y}, \mathbf{x})]}{\sum_{\mathbf{y}'} \exp[-E(\mathbf{y}', \mathbf{x})]}.$$

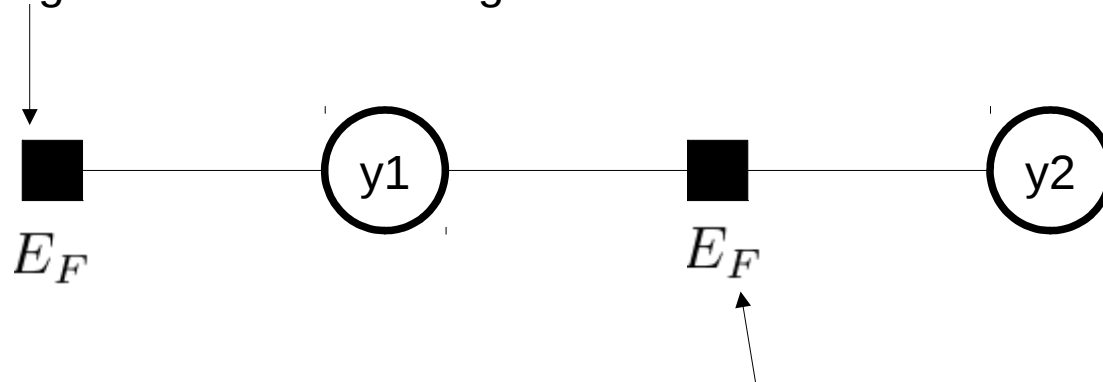
Energy function:  $E(\mathbf{y}, \mathbf{x}) = \sum_{F \in \mathcal{F}} E_F(\mathbf{y}_F, \mathbf{x}_F).$

CNN based (log-) potential function (factor function):  $E_F(\mathbf{y}_F, \mathbf{x}_F)$

The potential function can be a unary, pairwise, or high-order potential function

Factor graph: a factorization of the joint distribution of variables

CNN based unary potential: measure the labelling confidence of a single variable



CNN based pairwise potential, measure the confidence of the pairwise label configuration

# Challenges in Learning CRFs+CNNs

Prediction can be made by marginal inference (e.g. message passing):

$$\forall p \in \mathcal{N} : P(y_p | \mathbf{x}) = \sum_{\mathbf{y} \setminus y_p} P(\mathbf{y} | \mathbf{x}).$$

CRF-CNN joint learning:

learning CNN potential functions by optimizing the CRF objective, typically, minimizing the negative conditional log-likelihood (NLL)

$$\min_{\boldsymbol{\theta}} \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 + \sum_{i=1}^N [E(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}; \boldsymbol{\theta}) + \log Z(\mathbf{x}^{(i)}; \boldsymbol{\theta})].$$

Learning CNN parameters with stochastic gradient descend.

The partition function  $Z$  brings difficulties for optimization:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log Z(\mathbf{x}; \boldsymbol{\theta}) &= \sum_{\mathbf{y}} \frac{\exp[-E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta})]}{\sum_{\mathbf{y}'} \exp[-E(\mathbf{y}', \mathbf{x}; \boldsymbol{\theta})]} \nabla_{\boldsymbol{\theta}} [-E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta})] \\ &= -\mathbb{E}_{\mathbf{y} \sim P(\mathbf{y} | \mathbf{x}; \boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}). \end{aligned}$$

For each SGD iteration:

require approximate marginal inference to calculate the factor marginals.

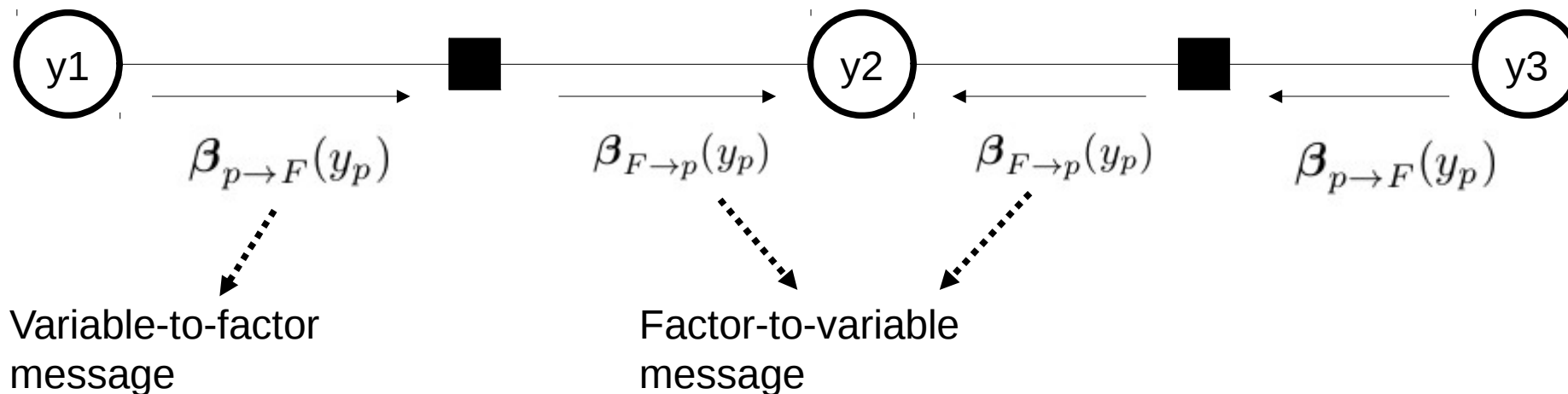
CNN training need a large number of SGD iterations, training become intractable.

# Solutions

- Traditional approach:
  - Applying approximate learning objectives
    - Replace the optimization objectives to avoid inference
    - e.g., piecewise training, pseudo-likelihood
- Our approach
  - Directly target the final prediction
    - Traditional approach aims to learn the potentials function and perform inference for final prediction
  - Not learning the potential function
  - Learning CNN estimators to directly output the required intermediate values in an inference algorithm
    - Focus on message passing based inference for prediction (specifically Loopy BP).
    - Directly learning CNNs to predict the messages.

# belief propagation: message passing based inference

A simple example of the marginal inference on the node  $y_2$ :



Message:  $K$ -dimensional vector,  $K$  is the number of classes (node states)

Variable-to-factor message:

$$\bar{\beta}_{p \rightarrow F}(y_p) = \sum_{F' \in \mathcal{F}_p \setminus F} \beta_{F' \rightarrow p}(y_p).$$

$$\beta_{p \rightarrow F}(y_p) = \log \frac{\exp \bar{\beta}_{p \rightarrow F}(y_p)}{\sum_{y'_p} \exp \bar{\beta}_{p \rightarrow F}(y'_p)}$$

Factor-to-variable message:

$$\beta_{F \rightarrow p}(y_p) = \log \sum_{\mathbf{y}'_F \setminus y'_p, y'_p = y_p} \exp \left[ -E_F(\mathbf{y}'_F) + \sum_{q \in \mathcal{N}_F \setminus p} \beta_{q \rightarrow F}(y'_q) \right].$$

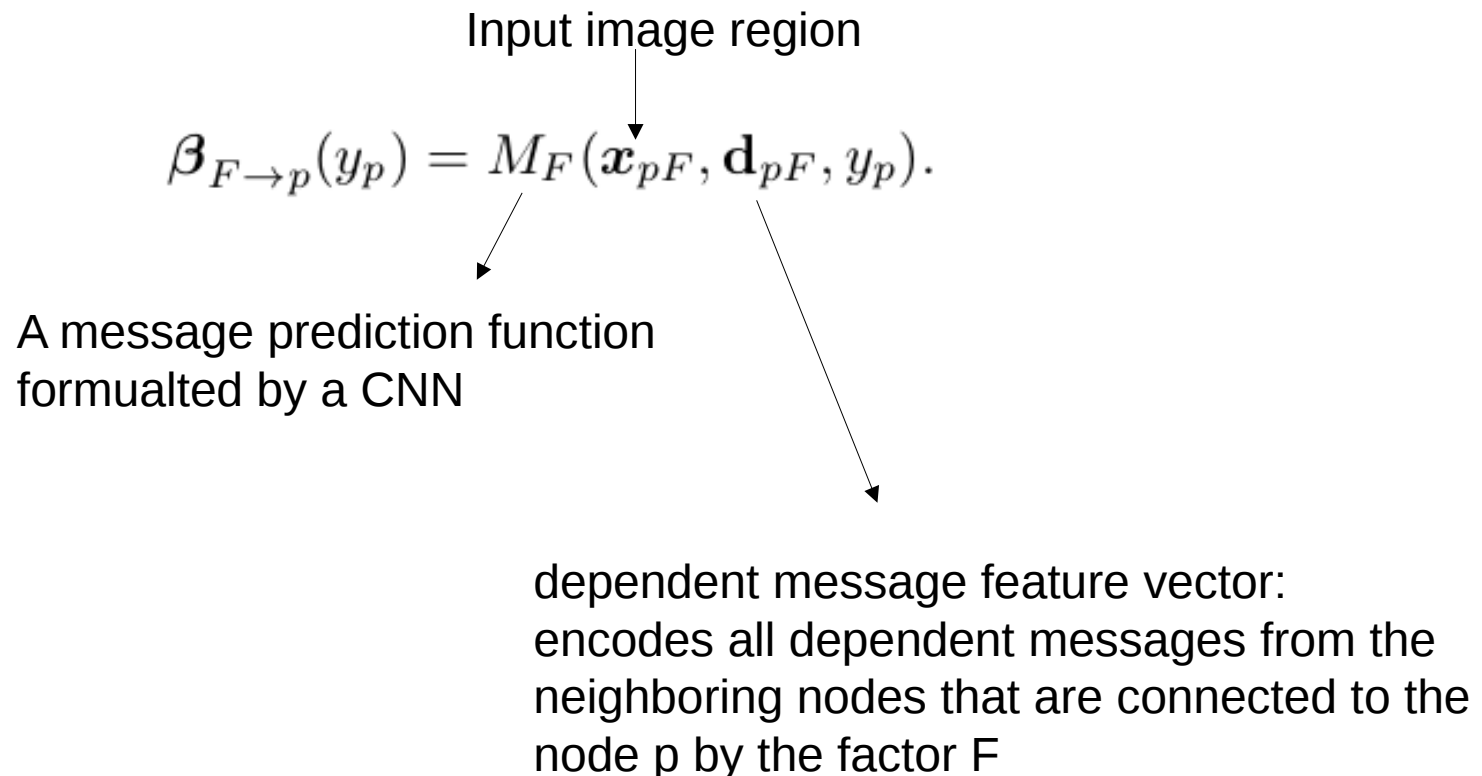
marginal distribution (beliefs) of one variable:

$$P(y_p | \mathbf{x}) = \sum_{\mathbf{y} \setminus y_p} P(\mathbf{y} | \mathbf{x}) = \frac{1}{Z_p} \exp \left[ \sum_{F \in \mathcal{F}_p} \beta_{F \rightarrow p}(y_p) \right]$$

# CNN message estimators

- Directly learn a CNN function to output the message vector
  - Don't need to learn the potential functions

The factor-to-variable message:



# Learning CNN message estimator

The variable marginals estimated by CNN:

$$P(y_p|\mathbf{x}) = \sum_{\mathbf{y} \setminus y_p} P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_p} \exp \left[ \sum_{F \in \mathcal{F}_p} \beta_{F \rightarrow p}(y_p) \right] = \frac{1}{Z_p} \exp \sum_{F \in \mathcal{F}_p} M_F(\mathbf{x}_{pF}, \mathbf{d}_{pF}, y_p; \boldsymbol{\theta}_F).$$

Define the cross entropy loss between the ideal marginal and the estimated marginal:

$$\begin{aligned} J(\mathbf{x}, \hat{\mathbf{y}}; \boldsymbol{\theta}) &= - \sum_{p \in \mathcal{N}} \sum_{y_p=1}^K \delta(y_p = \hat{y}_p) \log P(y_p|\mathbf{x}; \boldsymbol{\theta}) \\ &= - \sum_{p \in \mathcal{N}} \sum_{y_p=1}^K \delta(y_p = \hat{y}_p) \log \frac{\exp \sum_{F \in \mathcal{F}_p} M_F(\mathbf{x}_{pF}, \mathbf{d}_{pF}, y_p; \boldsymbol{\theta}_F)}{\sum_{y'_p} \exp \sum_{F \in \mathcal{F}_p} M_F(\mathbf{x}_{pF}, \mathbf{d}_{pF}, y'_p; \boldsymbol{\theta}_F)}, \end{aligned}$$

The optimization problem for learning:

$$\min_{\boldsymbol{\theta}} \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 + \sum_{i=1}^N J(\mathbf{x}^{(i)}, \hat{\mathbf{y}}^{(i)}; \boldsymbol{\theta}).$$

# Application on semantic segmentation

