CONDITIONAL RANDOM FIELDS AS RECURRENT NEURAL NETWORKS FOR SEMANTIC IMAGE SEGMENTATION AND OTHER PROBLEMS

Alex Ter-Sarkisov, Postdoctoral Researcher at the School of Computing,
Dublin Institute of Technology

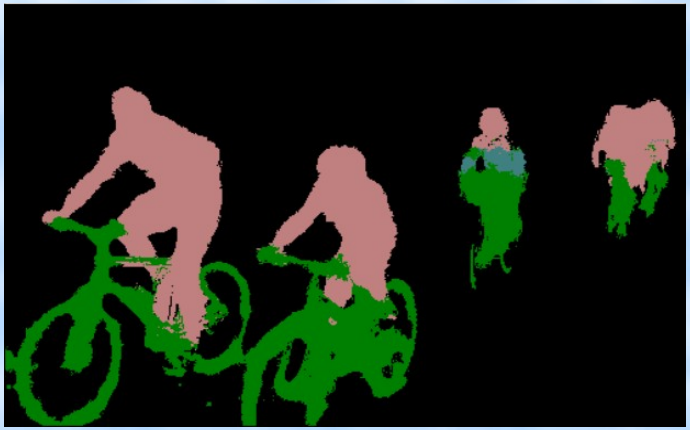
27/06/16

1. Introduction (CRFs and Deep Learning)
2. End-to-End Learning (ConvNN and CRF-RNN)
3. Structure of CRF-RNN
4. Current Results
5. Application to Other Problems

INTRODUCTION

Introduction

Problem formulation: identify and label objects on the image (semantic image segmentation)



Introduction

Existing state-of-the art algorithms:

- Fully Connected ConvNNs (Long et al, 2015)
- Achieved ~62.7% precision on VOC2011 test

Shortcomings:

- Coarse output (ConvNNs) due to the size of the receptive field,
- ConvNNs are tuned to perform classification rather than segmentation problems, i.e. answering the question like 'Which object is present in the image' rather than 'which objects and where are present on the image'
- Fails to capture the structure of the image (interaction between pixels)

Introduction

Solution: Conditional Random Fields (CRFs)

- A form or Markov Random Fields (MRFs), conditional on a global observation (image)
- Takes into consideration the structure of the image (Gaussian filters),
- Error function uses Intersect over Union (IoU) metric to estimate discrepancy between the model and the ground truth,
- Current implementation: Kraehenbuehl, Koltun (2012)
- Good results, but could be improved further!
- Hence, this presentation is based on S.Zheng et al (Conditional Random Fields as Recurrent Neural Networks, 2015)

END-TO-END LEARNING (CRF AS RNN)

End-to-End Learning

CRF as RNN:

- Combines unary (independent) and pairwise (structure) values for each pixel:
- ConvNN returns unary values for each pixel
- CRF is embedded within a Recurrent Neural Network (RNN) framwork – hence the name: CRFasRNN
- CRFasRNN is used to derive label distribution for each pixel based on the image structure (connection between pixels in the image)
- Training using Backpropagation Through Time (BPTT)

End-to-End Learning

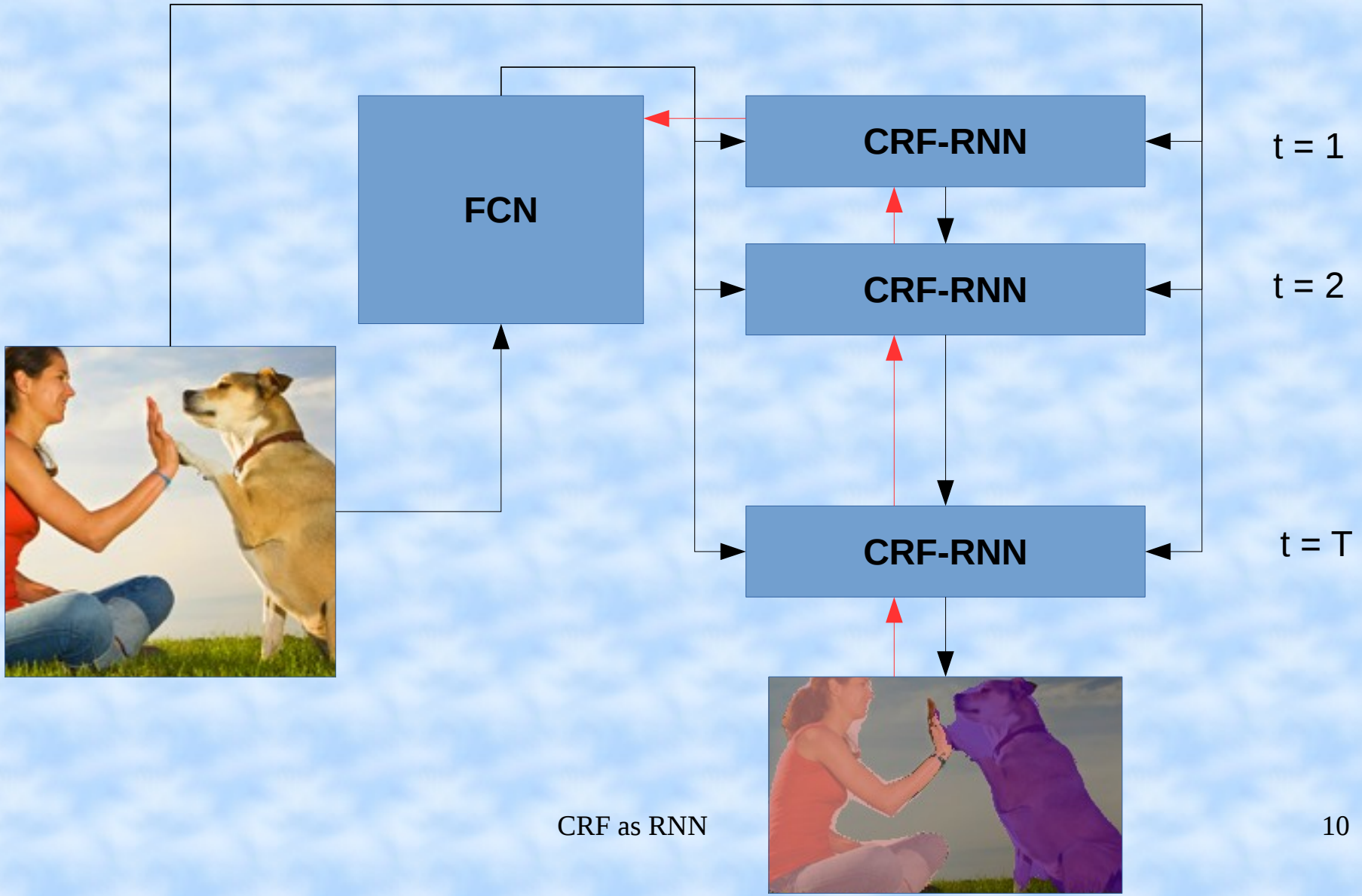
CRF as RNN:

- Feed-forward stage: one unary input from the ConvNN, T iterations within CRF-RNN
- Output of the network is taken only after T iterations: probability distribution over every pixel in the image (L labels per N pixels)
- Objective function: Energy over image

$$E(x) = \sum_{i=1}^{n} \psi_u(x_i) + \sum_{j=i+1}^{n} \psi_p(x_i, x_j)$$

- Loss function: Intersect over Union

End-to-End Learning



End-to-End Learning

CRF as RNN:

- Backpropagation stage: error differentials are computed from the IoU loss function
- One of the key contributions of the article: end-to-end training. Partial derivatives wrt to weights are backprop'd through the CRFasRNN (T times, as usual for unfolded RNN) and through ConvNN
- Parameters of CRFasRNN and ConvNN are learnt jointly during backprop stage

STRUCTURE OF CRF-RNN

Structure of CRF-RNN

CRF as RNN takes ConvNN's output (unary values) as the initial input and proceeds with the mean-field approximation of the underlying label distribution.

The main job of CRFasRNN is to understand the interaction between pixels on the image by extracting features such as color and position.

The output of CRF is probability distribution over labels in each pixel that considers label compatibility, i.e. the structure of the image, something that other approaches like ConvNN are not capable of.

Structure of CRF-RNN

CRF as RNN:

- For T iterations:
  - For each pixel:
    - For each label:
      - Message Passing (loop through pixels)
      - Filter weighing (loop through filters)
      - Compatibility (loop through labels)
      - Adding Unary Potentials
      - Normalization (loop through labels)

Structure of CRF-RNN

Message Passing:

- Taking variables like RGB and pixel position we obtain Gaussian kernels of the pairs of pixels,
- Kernels are then convolved with inputs (unary potentials)
- In total, for every label and every pair of pixels there are a total of M (here M=2: bilateral and spatial) filters/kernels stored

$$\tilde{Q} = \sum_{j \neq k} k^m(\boldsymbol{f}_i, \boldsymbol{f}_j) Q_j(l)$$

Structure of CRF-RNN

Filter Weighing:

- All M filters are weighed and convolved:

$$k(\boldsymbol{f}_i, \boldsymbol{f}_j) = w_1 \exp\left(\frac{-\left|p_i - p_j\right|^2}{2\theta_1^2} - \frac{\left|I_i - I_j\right|^2}{2\theta_2^2}\right) + w_2 \exp\left(\frac{-\left|p_i - p_j\right|^2}{2\theta_3^2}\right)$$

$$\bar{Q}(l) = \sum_{m=1}^{M} w_m \tilde{Q}_m$$

Structure of CRF-RNN

Compatibility Transform:

- For each label in every pixel weighed filter outputs are convolved with a parameter μ(l,l')
- This parameter penalizes assignment of different labels to pixels with similar properties
- In CRFasRNN setting this parameter is learned rather than fixed

$$\hat{Q}(l) = \sum_{l \neq \acute{l}} \mu(l, \acute{l}) \overline{Q_i}(\acute{l})$$

Structure of CRF-RNN

Unary Potentials, Normalization and BPTT:

- Unary values are taken from the ConvNN (FCN-8s, stride length = 8)
- Softmax normalization over all labels in every pixel
- Error differentials are taken wrt compatibility and kernel weights; they are backprop'd for T iterations in the unfolded CRFasRNN and ConvNN

APPLICATION TO OTHER PROBLEMS

Application to other problems

Instance segmentation and tracking

- CRFasRNN distinguishes between 20 different classes, often finding sharp contours, but not instances of the same class,
- This is often a problem if your objective is automation of object tracking in a challenging environment:
  - bad lighting, shadows,
  - cluttered objects, occlusion,
  - color confusion (object looks a lot like a background)

Application to other problems

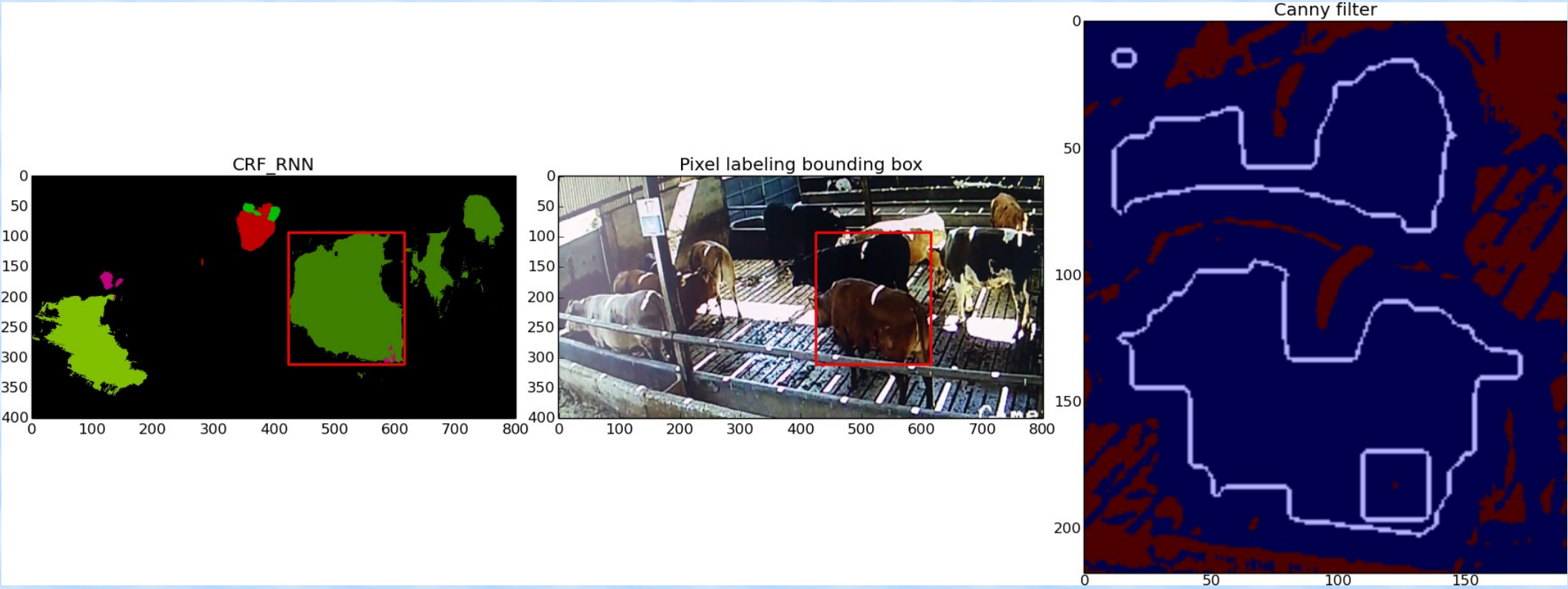
Instance segmentation and tracking

- Most existing out-of-the box algorithms like TLD, MIL, MeanShift, CamShift fail to do this
- Solution: CRFasRNN in combination with other algorithms:
  - Detect 'cow blobs' on the image
  - Zoom to each blob, improve the contrast using threshold,
  - Extract and refine edges/contours of objects

Application to other problems

Instance segmentation and tracking

- Work in progress:



Website:

http://www.robots.ox.ac.uk/~szheng/crfasrnndemo

THANK YOU