Conditional Random Fields as Recurrent Neural Networks

http://www.robots.ox.ac.uk/~szheng/papers/CRFasRNN.pdf

Sima Kiani

Outline

1. Markov Random Fields
2. Conditional Random Field (CRF)
3. Conditional Random Fields as Recurrent Neural Networks
    1. CRF as CNN for One Iteration
    2. CRF as RNN for Multiple Iterations
4. Datasets & Dimension
    1. Pascal VOC Datasets
    2. COCO (Microsoft Common Objects in Context)
5. Results
6. References

1 Markov Random Fields

A Markov random field is an undirected graph where each node captures the (discrete or Gaussian) probability distribution of a variable and the edges represent dependencies between those variables and are weighted to represent the relative strengths of the dependencies. Because of its undirected nature, a Markov random field can contain subgraphs of three or more interconnected nodes that interact in terms of the joint probabilities they represent in a way reminiscent of the Hopfield network. Furthermore, Markov random fields can also be configured to take into account the probabilities of individual variables having certain values (unary factors): for example, the impact of a very improbable value for a given variable on the rest of the network can be reduced on the basis that it is more likely to be a sampling error than a genuine reading.

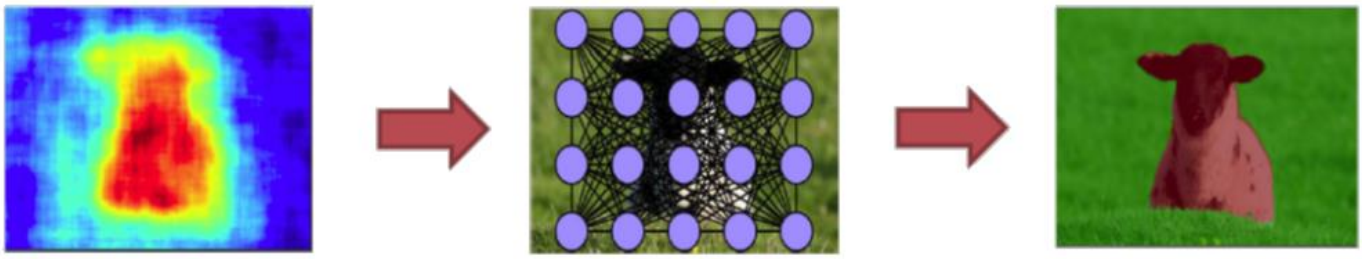
Formal definition

A Markov Random Field is a probability distribution $P(X = x)$ of a particular field configuration x in X. Where X are the set of random variables $X = (Xs)s\in S$, defined by an undirected graph G in which nodes correspond to variables Xs. Because X is a set, the probability of x should be understood to be taken with respect to a joint distribution of the Xs.

---

2 Conditional Random Field (CRF)

A conditional random field is a Markov random field that has specifically been set up to allow the values of a specific variable or variables to be predicted based on the values of the other variables.



Coarse output from the pixel-wise classifier

MRF/CRF modelling

Output after the CRF inference

- **Formal definition and Algorithm**

Definition 1 Assuming an undirected graph $G = (V, E)$ and random variable $Y = \{Y_v | v \in V\}$, if a random variable Y meets Markov property under a given random variable X (that is, $P(Y_v|X, Y_u, u \neq v) = P(Y_v|X, Y_u, u \sim v)$ holds for any node v), the conditional probability distribution $P(Y|X)$ will be referred to as a CRF.

Algorithm 1 The forward-backward algorithm for the probability calculation of the CRF

Input: The model $P(Y|X)$, the input sequence x, the output sequence y, and the location i

Output: The conditional probabilities $P(Y_i = y_i|x)$, $P(Y_{i-1} = y_{i-1}, Y_i = y_i|x)$

1. Let $M_i(y_{i-1}, y_i|x) = \exp(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, x, i) + \sum_{i,j} \mu_j s_j(y_i, x, i))$, $y_0 = start$, $y_{n+1} = stop$
2. Initialization, $\alpha_0(y_0|x) = 1$, $\beta_{n+1}(y_{n+1}|x) = 1$
3. Recursion, for $k = 1, 2, \ldots, i$

   $\alpha_k^T(y_k|x) = \alpha_{k-1}^T(y_{k-1}|x) M(y_{k-1}, y_k|x)$
4. Recursion, for $j = n, n-1, \ldots, i+1, i, i-1, \ldots, 1$

   $\beta_j(y_j|x) = M_{j+1}(y_j, y_{j+1}|x)\beta_{j+1}(y_{j+1}|x)$
5. Calculation, $Z(x) = 1^T \times \beta_1(x)$
6. Calculation, $P(Y_i = y_i|x) = \alpha_i^T(y_i|x)\beta_i(y_i|x)/Z(x)$

   $P(Y_{i-1} = y_{i-1}, Y_i = y_i|x) = \alpha_{i-1}^T(y_{i-1}|x) M_i(y_{i-1}, y_i|x)\beta_i(y_i|x)/Z(x)$

Algorithm 2 The L-BFGS algorithm for the parameter estimation of the CRF

Input: The eigenfunctions $f_1, f_2, \ldots, f_n$, and the empirical probability distribution $\tilde{P}(X,Y)$

Output: The optimal parameters λ_k and μ_j, the optimal model $\hat{P}(y|x)$

1. When $k = 1,2,\ldots,K_1$, let $\omega_k = \lambda_k$; when $k = K_1 + j$, and $j = 1,2,\ldots,K_2$, let $\omega_k = \mu_j$; $K = K_1 + K_2$; $\omega^{(i)} = (\omega_1^{(i)}, \omega_2^{(i)}, \ldots, \omega_K^{(i)})^T$
2. Initialization, let $i = 0$, choose $\omega^{(0)}$ and B_0
3. Calculation, $g_i = g(\omega^{(i)})$, if $g_i = 0$, stop the algorithm, otherwise, go to step 4
4. Calculation, $p_i = -B_i^{-1} g_i$
5. Calculation, find λ_i, so that $f(\omega^{(i)} + \lambda_i p_i) = \min_{\lambda \geq 0} f(\omega^{(i)} + \lambda_i p_i)$
6. Update, $\omega^{(i+1)} = \omega^{(i)} + \lambda_i p_i$
7. Calculation, $g_{i+1} = g(\omega^{(i+1)})$, if $g_{i+1} = 0$, stop the algorithm, otherwise, updateB_{i+1}
   $B_{i+1} = B_i + \frac{y_i y_i^T}{y_i^T \delta_i} - \frac{B_i \delta_i \delta_i^T B_i}{\delta_i^T B_i \delta_i}$, where $y_i = g_{i+1} - g_i$, $\delta_i = \omega^{(i+1)} - \omega^{(i)}$
8. Let $i = i + 1$, go to step 4

Algorithm 3 The Viterbi algorithm for the prediction of the CRF

Input: The feature vector $F(y,x)$, the weight vector ω, and the observation sequence $x = (x_1, x_2, \ldots, x_n)$

Output: The optimal path $y^* = (y_1^*, y_2^*, \ldots, y_n^*)$

1. Initialization, $\delta_1(j) = \omega \times F_1(y_0 = start, y_1 = j, x)$, $j = 1,2,\ldots m$
2. Recursion, for $i = 2,3,\ldots,n$
   $\delta_i(l) = \max_{1 \leq j \leq m} \{\delta_{i-1}(j) + \omega \times F_i(y_{i-1} = j, y_i = l, x)\}$. $l = 1,2,\ldots,m$
   $\psi_i(l) = \arg\max_{1 \leq j \leq m} \{\delta_{i-1}(j) + \omega \times F_i(y_{i-1} = j, y_i = l, x)\}$. $l = 1,2,\ldots,m$
3. Termination, $\max_y(\omega \times F(y,x)) = \max_{1 \leq j \leq m} \delta_n(j)$
   $y_n^* = \arg\max_{1 \leq j \leq m} \delta_n(j)$
4. Backtraking, $y_i^* = \psi_{i+1}(y_{i+1}^*)$, $i = n-1, n-2, \ldots, l$
   $y^* = (y_1^*, y_2^*, \ldots, y_n^*)$

- The purpose of CRF is to refine the coarse output based on the label at each location itself, and the neighboring positions' labels and locations.

- Fully connected pairwise CRF is considered. Fully connected means all locations are connected as shown in the middle of the figure above. Pairwise means the connections are connected in pairs.

- When we are talking about CRF, we are talking about how to minimize an energy function. Here, we need to minimize the energy of a label assignment. I just treat energy as a kind of cost function. By assigning of the most probable label to each location, we can get lower energy, i.e. lower cost, and thus, higher accuracy.

- The CRF is characterized by Gibbs distribution of a form:

$$P(\mathbf{X} = \mathbf{x}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp(-E(\mathbf{x}|\mathbf{I}))$$

- where *I* is the input. *Xi* is the random variable at location *i* which represents the assigned label. *I* is discarded for simplicity. *E*(*x*) is the energy function and *Z*(*I*) is the partition function which is just the sum of all exp(-*E*(*x*)).

- This CRF distribution *P*(*X*) is approximated by *Q*(*X*), which is a product of independent *Qi*(*Xi*):

$$Q(\mathbf{X}) = \prod_i Q_i(X_i)$$

- In the paper, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials".) The energy function:

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j),$$

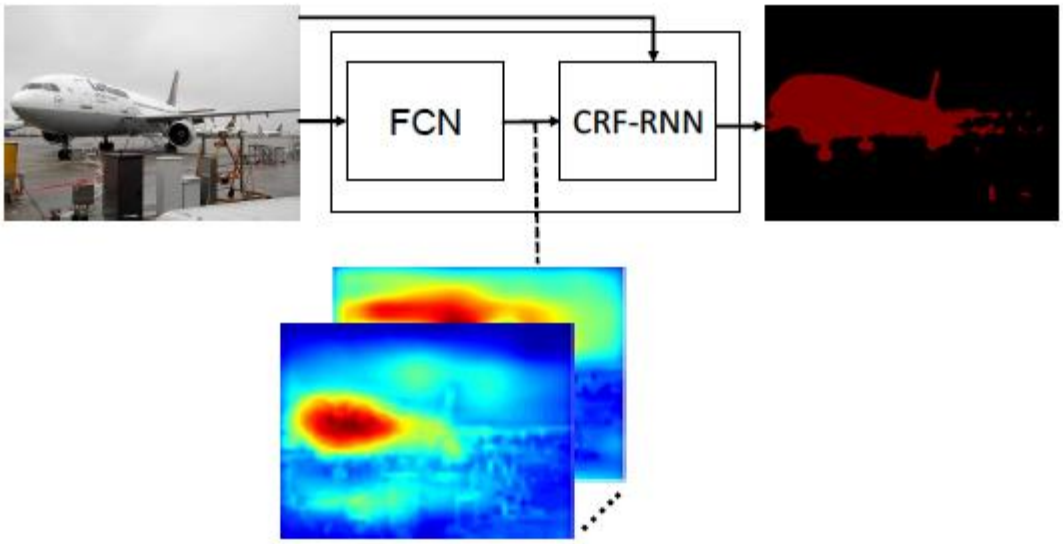
- 1st Term, Unary Energy $\Psi u(xi)$: measures the cost if the label assignment disagrees with the initial classifier. Unary means it just takes the label of the single position into consideration at each time.

- 2nd Term, Pairwise Energy $\Psi p(xi, xj)$: measures the cost if two similar pixels (e.g. neighbor pixels or the pixels have similar color) take different labels:

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k_G^{(m)}(\mathbf{f}_i, \mathbf{f}_j),$$

- where kG is the Gaussian kernel applied on feature vectors. The feature vector can be spatial locations and RGB values, e.g. Gaussian filter and bilateral filter.

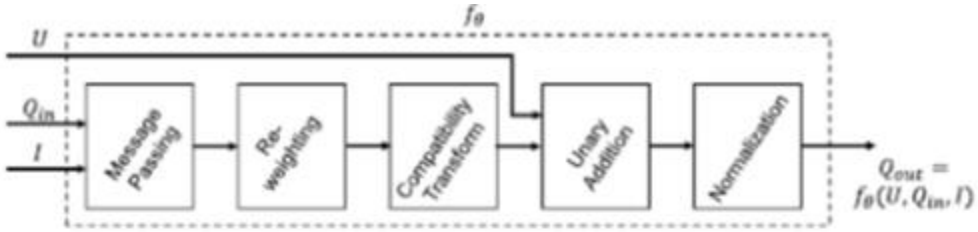
- And μ is the label compatibility function which assigns penalty when the labels are different.



End-to-end Trainable CRF-RNN

3 Conditional Random Fields as Recurrent Neural Networks

3.1 CRF as CNN for One Iteration



Algorithm 1 Mean-field in dense CRFs [29], broken down to common CNN operations.

$Q_i(l) \leftarrow \frac{1}{Z_i} \exp(U_i(l))$ for all i ▷ Initialization

while not converged **do**

$\tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l)$ for all m ▷ Message Passing

$\check{Q}_i(l) \leftarrow \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l)$ ▷ Weighting Filter Outputs

$\hat{Q}_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \check{Q}_i(l')$ ▷ Compatibility Transform

$\breve{Q}_i(l) \leftarrow U_i(l) - \hat{Q}_i(l)$ ▷ Adding Unary Potentials

$Q_i \leftarrow \frac{1}{Z_i} \exp\left(\breve{Q}_i(l)\right)$ ▷ Normalizing

end while

Initialization

- *Ui*(*l*) is the unary potential provided by FCN-8s which based on VGG-16.
- The *Qi*(*l*) is obtained using softmax.

- After initialization, there will be iterations (the while loop) for a sequence of processes.

Message Passing

- *M* Gaussian filters are used.
- Following [29], two Gaussian kernels are used, one spatial and one bilateral.

Weighting Filter Outputs

- A weighted sum of the *M* filter outputs from the previous step for each class label *l*.
- When each label is considered individually, it can be viewed as 1×1 convolution with *M* input channels and one output channel.
- In contrast to [29], individual kernel weights are used for each class label.

Compatibility Transform

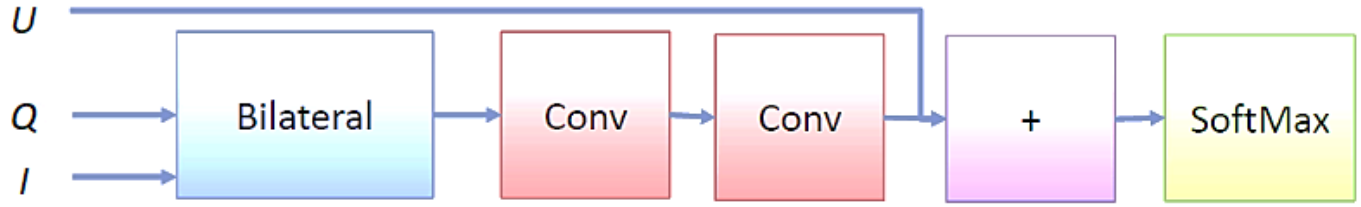
- A penalty is assigned when different labels are assigned.
- e.g.: assigning labels “person” and “bicycle” to nearby pixels should have a lesser penalty than assigning labels “sky” and “bicycle”.
- Thus, $\mu(l, l')$ is learned from the data.

Adding Unary Potentials

- The output from Compatibility Transform step is subtracted element-wise from the unary inputs U.

Normalization

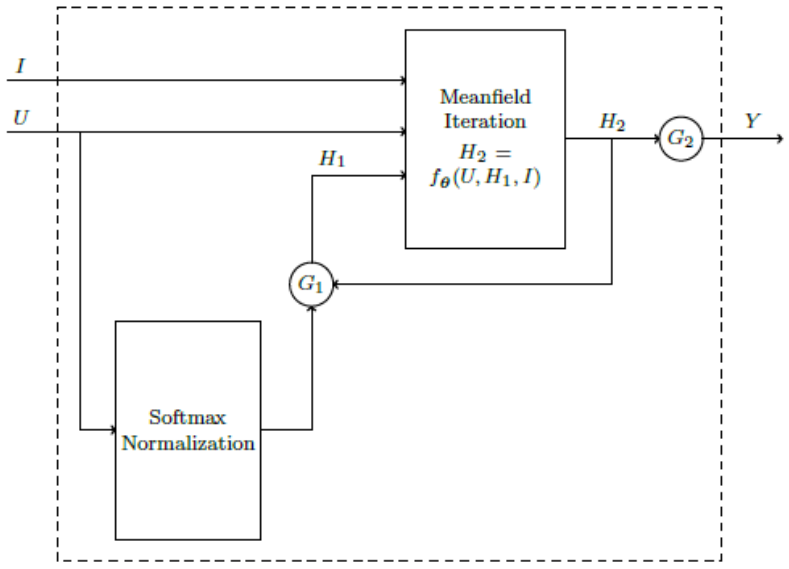
- Another softmax operation.



Fully connected CRFs as a CNN for one mean-field iteration

- Above is the overview of one mean-field iteration.
- By repeating the above module, we can have multiple mean-field iterations.

3.2 CRF as RNN for Multiple Iterations



$$H_1(t) = \begin{cases} \text{softmax}(U), & t = 0 \\ H_2(t-1), & 0 < t \leq T, \end{cases}$$

$$H_2(t) = f_{\boldsymbol{\theta}}(U, H_1(t), I), \quad 0 \leq t \leq T,$$

$$Y(t) = \begin{cases} 0, & 0 \leq t < T \\ H_2(t), & t = T. \end{cases}$$

CRF as RNN for Multiple Iterations

- I is the image. U is the unary potentials from FCN. T is the total number of iterations.
- $f\theta(U,H1(t),I)$ is the mean-field iteration as described in the previous section where θ is the CRF parameters described in the previous section, i.e. w, μ, m, l, l'.
- At t = 0, the first iteration, $H1(t)$ = softmax(U), otherwise $H1(t)$ is the output of the previous mean-field iteration, $H2(t\text{-}1)$.
- $H2(t)$ is the output of the mean-field iteration $f\theta(U,H1(t),I)$.
- The final output, $Y(t)=H2(T)$ when $t=T$, i.e. when the last iterations are finished.
- Recurrent Neural Network (RNN) setting is used, i.e. the parameters here are shared among all iterations.
- During training, T=5 is used to avoid vanishing/exploding gradient problem.
- During testing, T=10.

4. **Datasets & Data Dimension**

4.1. **PASCAL VOC**

http://host.robots.ox.ac.uk/pascal/VOC/

The PASCAL Visual Object Classes (VOC) 2012 dataset contains 20 object categories including vehicles, household, animals, and other: airplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse,

sheep, and person. Each image in this dataset has pixel-level segmentation annotations, bounding box annotations, and object class annotations. This dataset has been widely used as a benchmark for *object detection*, *semantic segmentation*, and *classification* tasks.

The PASCAL VOC dataset is split into three subsets: 1,464 images for training, 1,449 images for validation and a private testing set

4.2 COCO (Microsoft Common Objects in Context)

Introduced by Lin et al. in Microsoft COCO: Common Objects in Context

The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.

The dataset has annotations for

- object detection: bounding boxes and per-instance segmentation masks with 80 object categories,
- captioning: natural language descriptions of the images (see MS COCO Captions),
- key points detection: containing more than 200,000 images and 250,000 person instances labeled with key points (17 possible key points, such as left eye, nose, right hip, right ankle),
- stuff image segmentation – per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky (see MS COCO Stuff),
- panoptic: full scene segmentation, with 80 thing categories (such as person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road),
    - dense pose: more than 39,000 images and 56,000 person instances labeled with Dense Pose annotations – each labeled person is annotated with an instance id and a mapping between image pixels that belong to that person body and a template 3D

model. The annotations are publicly available only for training and validation images.

- dataset contains photos of 91 objects types that would be easily recognizable by a 4 year old. With a total of 2.5 million labeled instances in 328k images, the creation of our dataset drew upon extensive crowd worker involvement via novel user interfaces for category detection, instance spotting and instance segmentation. We present a detailed statistical analysis of the dataset in comparison to PASCAL, ImageNet, and SUN. Finally, we provide baseline performance analysis for bounding box and segmentation detection results using a Deformable Parts Model.

5 Results

Method	Without COCO	With COCO
Plain FCN-8s	61.3	68.3
FCN-8s and CRF disconnected	63.7	69.5
End-to-end training of CRF-RNN	69.6	72.9

Mean IU Accuracy on PASCAL VOC 2012 Validation Set

- With/Without COCO: Whether the model is trained by COCO as well.
- Plain FCN-8s: Lowest mean IU accuracy.
- With CRF but disconnected: That means CRF is not trained with FCN in end-to-end manner, higher mean IU accuracy is obtained

- End-to-end CRF-RNN: The highest mean IU accuracy is obtained which means end-to-end FCN+CRF is the best solution.

Method	VOC 2010 test	VOC 2011 test	VOC 2012 test
BerkeleyRC [3]	n/a	39.1	n/a
O2PCPMC [8]	49.6	48.8	47.8
Divmbest [44]	n/a	n/a	48.1
NUS-UDS [16]	n/a	n/a	50.0
SDS [23]	n/a	n/a	51.6
MSRA-CFM [13]	n/a	n/a	61.8
FCN-8s [37]	n/a	62.7	62.2
Hypercolumn [24]	n/a	n/a	62.6
Zoomout [38]	64.4	64.1	64.4
Context-Deep-CNN-CRF [35]	n/a	n/a	70.7
DeepLab-MSc [10]	n/a	n/a	71.6
Our method w/o COCO	73.6	72.4	72.0
BoxSup [12]	n/a	n/a	71.0
DeepLab [10, 41]	n/a	n/a	72.7
Our method with COCO	**75.7**	**75.0**	**74.7**

Mean IU Accuracy on PASCAL VOC 2010, 2011, 2012 Test Set

- CRF-RNN w/o COCO: It outperforms FCN-8s and DeepLab-v1.

- CRF-RNN with COCO: The results are even better.

PASCAL Context

Method	O_2P [8]	CFM [13]	FCN-8s [37]	CRF-RNN
Mean IU	18.1	34.4	37.78	**39.28**

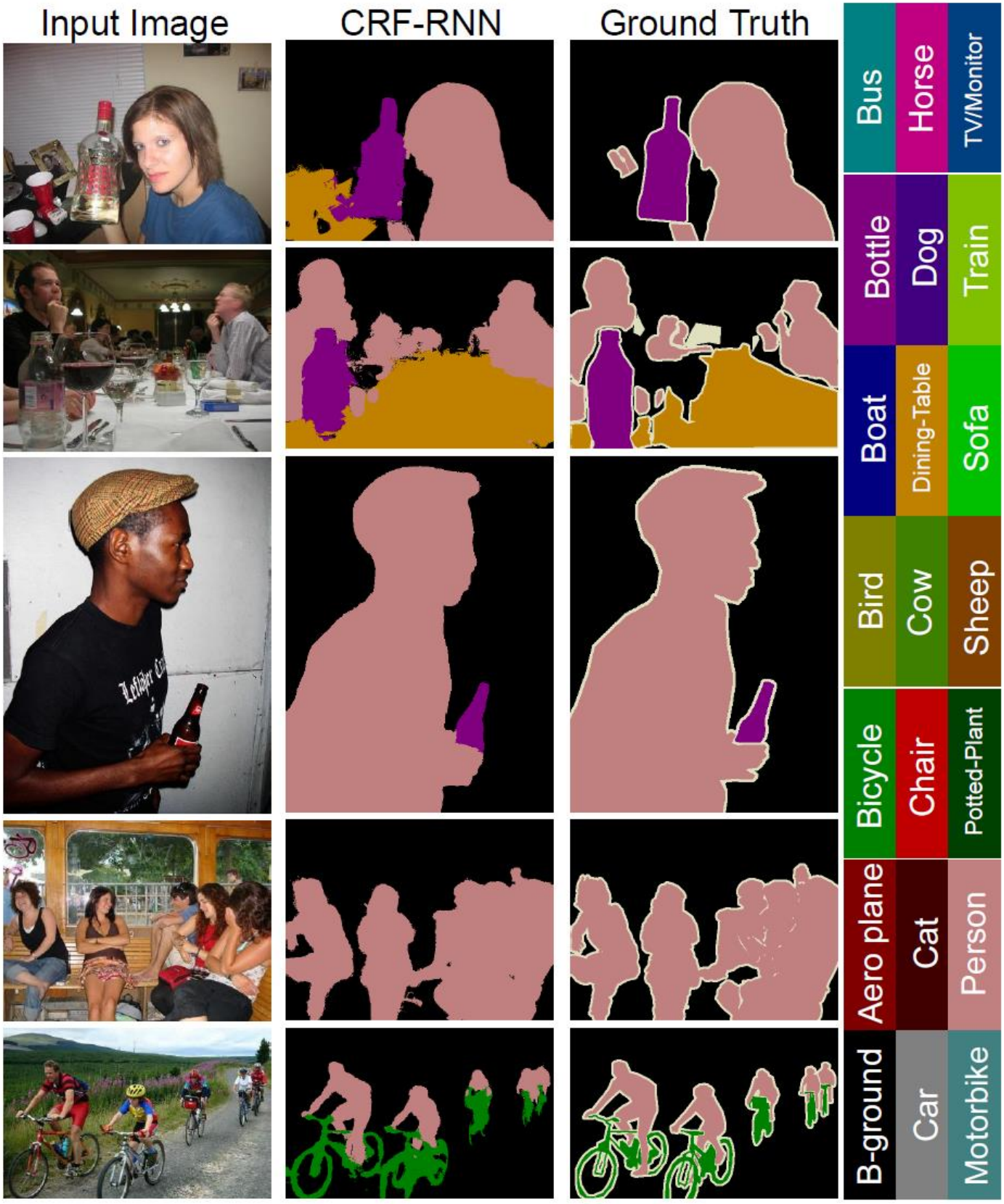
Mean IU Accuracy on PASCAL Context Validation Set

- CRF-RNN: Higher mean IU accuracy than FCN-8s.

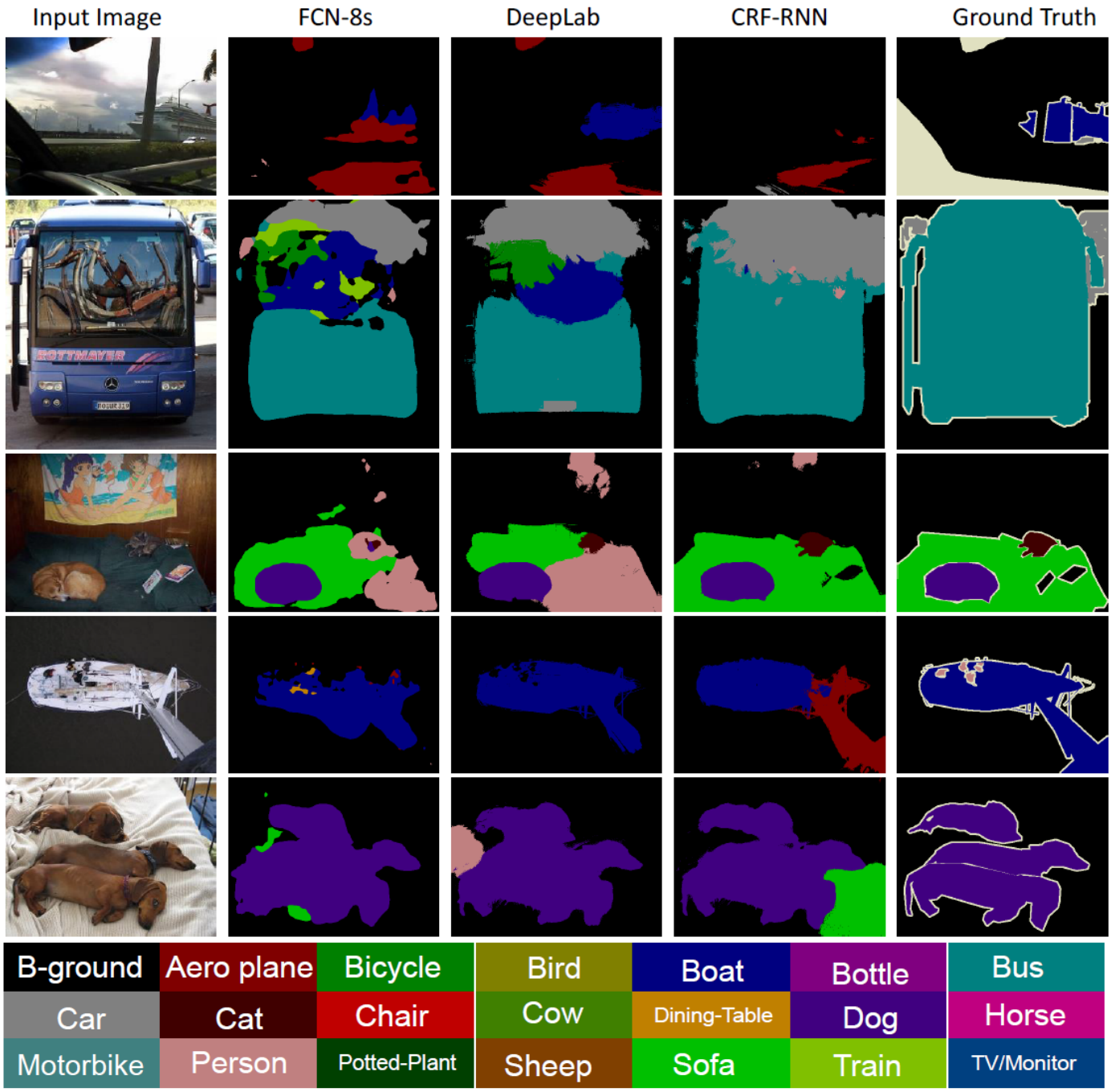
Further Analyses

- Additional experiments are performed on PASCAL VOC 2012 Validation Set.
- Using different weights w for different classes increases 1.8% mean IU.
- T=10 during both training and testing induces 0.7% drops, which argues that there is vanishing gradient effect.
- Independent parameters for each iteration instead of sharing parameters, only 70.9% mean IU accuracy is obtained, which shows that recurrent structure is important.

Qualitative Results



Some Good Results on PASCAL VOC 2012



Comparison with State-of-the-art Approaches

Though CRF-RNN is published in 2015, this paper has introduced an important concept/logic to me, i.e. converting/approximating a

conventional/non-deep-learning approach into deep-learning-based approach and turn it into an end-to-end solution.

6 Reference

Conditional Random Fields as Recurrent Neural Networks

https://towardsdatascience.com/review-crf-rnn-conditional-random-fields-as-recurrent-neural-networks-semantic-segmentation-a11eb6e40c8c

https://github.com/torrvision/crfasrnn/

https://github.com/sadeepj/crfasrnn_keras

http://crfasrnn.torr.vision/