Conditional Random Fields as Recurrent Neural Networks

S. Zheng, S. Jayasumana, B. Romera-Paredes
V. Vineet, Z. Su, D. Du, C. Huang, P.H.S. Torr
Introduction

• Traditional CNNs have convolutional filters with large receptive fields and hence produce coarse outputs when restructured to produce pixel-level labels.

• They result in non-sharp boundaries and blob-like shapes in semantic segmentation tasks.

• CNNs lack smoothness constraints.
Introduction

• **Probabilistic graphical models** have been developed as effective methods to enhance the accuracy of pixel level labelling tasks.

• **Conditional Random Fields (CRFs)** have observed widespread success in this area and have become one of the most successful graphical models used in computer vision.
Introduction

• One way to utilize CRFs to improve the semantic labelling results produced by a CNN is to apply CRF inference as a **post-processing** step disconnected from the training of the CNN.

• This does not fully harness the strength of CRFs since it is not integrated with the deep network.
Introduction

• This work combines the strengths of both CNNs and CRF based graphical models in one unified framework.

• It formulates mean-field approximate inference for the dense CRF with Gaussian pairwise potentials as an RNN.
Related Work

• Pinheiro and Collobert employed an RNN to model the spatial dependencies during scene parsing.
• Long et al. used FCC, and the notion that top layers obtain meaningful features for object recognition whereas low layers keep information about the structure of the image, such as edges.
Related Work

• Krahenbühl and Koltun used fully connected Conditional Random Fields (CRF) for segmentation.

• Their main contribution is a highly efficient approximate inference algorithm for fully connected CRF models.
Conditional Random Fields

• Consider a random field $X$ defined over a set of variables $\{X_1, \ldots, X_N\}$.

• The domain of each variable is a set of labels $L = \{l_1, l_2, \ldots, l_k\}$

• Consider also a random field $I$ defined over variables $\{I_1, \ldots, I_N\}$.
Conditional Random Fields

• A conditional random field \((\mathbf{I}, \mathbf{X})\) is characterized by a Gibbs distribution:

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

• \(E(\mathbf{x})\) is called the energy of the configuration \(\mathbf{x}\)

\[
E(\mathbf{x}) = \sum_i \psi_u(\mathbf{x}_i) + \sum_{i<j} \psi_p(\mathbf{x}_i, \mathbf{x}_j)
\]

unary energy  pairwise energy
Conditional Random Fields

• In this model, **unary energies** are obtained from a CNN, which predicts labels for pixels without considering the smoothness and the consistency of the label assignments.

• The **pairwise energies** provide an image data-dependent smoothing term that encourages assigning similar labels to pixels with similar properties.
Conditional Random Fields

• Pairwise potentials are modelled as weighted Gaussians:

\[
\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k^{(m)}_G(f_i, f_j)
\]
Conditional Random Fields

- **Minimizing** the above CRF energy $E(x)$ yields the most probable label assignment $x$ for the given image.
- Approximating the CRF distribution $P(X)$ by a simpler distribution $Q(X)$, which can be written as the product of independent marginal distributions, i.e., $Q(X) = \prod_i Q_i(X_i)$
Conditional Random Fields

• **Minimization** yields the following iterative update equation:

\[
Q_i(x_i = l) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^{K} w^{(m)} \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \right\}
\]
Conditional Random Fields

• This update equation leads to the following inference algorithm:

\[
Q_i(l) \leftarrow \frac{1}{Z_i} \exp(U_i(l)) \text{ for all } i \quad \triangleright \text{ Initialization}
\]

\[
\text{while not converged do}
\]

\[
\tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(f_i, f_j)Q_j(l) \text{ for all } m \quad \triangleright \text{ Message Passing}
\]

\[
\tilde{Q}_i(l) \leftarrow \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l) \quad \triangleright \text{ Weighting Filter Outputs}
\]

\[
\hat{Q}_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \tilde{Q}_i(l') \quad \triangleright \text{ Compatibility Transform}
\]

\[
\hat{Q}_i(l) \leftarrow U_i(l) - \hat{Q}_i(l)
\]

\[
Q_i \leftarrow \frac{1}{Z_i} \exp\left(\hat{Q}_i(l)\right) \quad \triangleright \text{ Normalizing}
\]

\[
\text{end while}
\]
Mean-field Iteration

• Initialization:

\[ Q_i(l) \leftarrow \frac{1}{Z_i} \exp(U_i(l)), \text{ where } Z_i = \sum_l \exp(U_i(l)) \]

• This is equivalent to applying a **softmax** function over the unary potentials \( U \) across all the labels at each pixel.
Mean-field Iteration

1. Message Passing

2. Weighting Filter Outputs

3. Compatibility Transform

4. Adding Unary Potentials

5. Normalization

\[ Q_i(x_i = l) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^{K} w^{(m)} \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \right\} \]
CRF as RNN

• In the previous section, it was shown that one iteration of the mean-field algorithm can be formulated as a stack of common CNN layers.
• Multiple mean-field iterations can be implemented by repeating the above stack of layers.
• This is equivalent to treating the iterative mean-field inference as a Recurrent Neural Network (RNN).
CRF as RNN

- Behaviour of the network is given by the following equations where $T$ is the number of mean-field iterations:

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

\[
H_2(t) = f_\theta(U, H_1(t), I), \quad 0 \leq t \leq T,
\]

\[
Y(t) = \begin{cases} 
0, & 0 \leq t < T \\
H_2(t), & t = T.
\end{cases}
\]
CRF as RNN

• Since the calculation of error differentials w.r.t. these parameters in a single iteration was described, they can be learnt in the RNN setting using the standard back-propagation through time algorithm.

• Krahenbühl and Koltun showed that the mean-field iterative algorithm for dense CRF converges in less than 10 iterations (5 in practice).
CRF as RNN

• During the training of the models, standard softmax loss function is used.
• This is the log-likelihood error function.
Experiments

- **Pascal VOC2012** dataset, and the **Pascal Context** dataset are used.
- In addition to the above training set, **Microsoft COCO** dataset is used.
Experiments

<table>
<thead>
<tr>
<th>Input Image</th>
<th>FCN-8s</th>
<th>DeepLab</th>
<th>CRF-RNN</th>
<th>Ground Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
</tr>
</tbody>
</table>
Experiments
# Experiments

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