Conditional Random Fields as Recurrent Neural Network

Roi Eisner
Tamar Weiser
Semantic Segmentation

Achieving close to ground truth segmentation
Intro- The challenge

- CNN has been successful in **high-level** CV tasks (image recognition, object detection etc)
- Semantic segmentation achieves pixel labeling, a **low-level** CV task
- CNN -> large receptive fields -> non-sharp boundaries, blob-like shapes
  
  => There’s a lack *smoothness constraints*

![Input image](image1.jpg)  ![FCN-8](image2.jpg)  ![Ground truth segmentation](image3.jpg)
Our motivation using CRF

• CRF is commonly used in machine learning and pattern recognition
• CRF can refine coarse pixel-label predictions producing sharp-boundaries segmentations
• There are 2 approaches for CRF post processing an FCN output:

1. Adding the CRF after the training process
2. Considering the CRF as a RNN, training it together with the FCN

The network parameters take the CRF into consideration - achieving optimization
Conditional Random Fields- Some math

• X is a random field defined over a set of variables \{X_1, ..., X_N\} each labeled by \{l_1, ..., l_k\}.
• I a random field defined over \{I_1, ..., I_N\}.

• The goal- \(\arg \max_x P(X = x|I)\)

• (I,X) A conditional random field:

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

Gibbs distribution
• Minimizing $E(x)$ the energy of configuration $x$ yields the most probable label assignment $x$ for the given image:

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

**Unary energies:**
Measure the inverse likelihood of pixel $i$ taking the label $x_i$

**Pairwise energies:**
Measure the cost of assigning labels $x_i x_j$ to pixels $i,j$
Pairwise energies

Compatibility function

\[ \psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k_G^{(m)} (f_i, f_j) \]

Gaussian kernels

\[ k(f_i, f_j) = w^{(1)} \exp \left( -\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2} \right) + w^{(2)} \exp \left( -\frac{|p_i - p_j|^2}{2\theta_\gamma^2} \right) \]

appearance kernel

smoothness kernel
Mean field approximation

• Minimizing $E(X)$ is tough, we now assume $P(X) \approx Q(X) = \prod_i Q_i(X_i)$

• The effect of all the other pixels on any given pixel is approximated by a single averaged effect

• Find $\{Q_i(x_i)\}$ subject to minimizing KL-Divergence $D_{KL}(Q \| P)$ - the amount of information lost when Q is used to approximate P.

• Eventually we want to compute in iterations

$$Q_i(x_i = l) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{l' \in \mathcal{L}} \sum_{m=1}^{K} \mu(l, l') w^{(m)} \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \right\}$$
From mean field iteration to CNN

1. Message Passing

2. Weighting Filter Outputs

3. Compatibility Transform

4. Adding Unary Potentials

5. Normalization

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

A softmax function over the unary potentials

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

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

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

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

\[ \tilde{Q}_i(l) \leftarrow U_i(l) - \tilde{Q}_i(l) \quad \text{▷ Adding Unary Potentials} \]

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

end while
From mean field iteration to CNN

Applying M Gaussian filters on Q values
From mean field iteration to CNN

Can be viewed as usual convolution with a $1 \times 1$ filter with M input channels, and one output channel

Different weights for different labels (such as Bike vs TV)
From mean field iteration to CNN

Can be viewed as another convolution layer where the spatial receptive field of the filter is $1 \times 1$, and the number of input and output channels are both $L$.

Using the Potts model, given by

$$\mu(l, l') = [l \neq l']$$

Can be learned from data.
From mean field iteration to CNN

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

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

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

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

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

\[ Q_i(l) \leftarrow U_i(l) - Q_i(l) \quad \triangleright \text{Adding Unary Potentials} \]

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

\[ \text{end while} \]
From mean field iteration to CNN

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

\textbf{while not converged do}

\[ \tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(f_i, f_j)Q_j(l) \text{ for all } m \]  \hspace{1cm} \triangleright \text{Message Passing}

\[ \tilde{Q}_i(l) \leftarrow \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l) \]  \hspace{1cm} \triangleright \text{Weighting Filter Outputs}

\[ \hat{Q}_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \tilde{Q}_i(l') \]  \hspace{1cm} \triangleright \text{Compatibility Transform}

\[ \bar{Q}_i(l) \leftarrow U_i(l) - \hat{Q}_i(l) \]  \hspace{1cm} \triangleright \text{Adding Unary Potentials}

\[ Q_i \leftarrow \frac{1}{Z_i} \exp(\bar{Q}_i(l)) \]  \hspace{1cm} \triangleright \text{Normalizing}

\textbf{end while}
```
CRF as RNN

• We saw that one iteration of the mean-field algorithm can be presented as a stack of common CNN layers.

• Multiple mean-field iterations can be achieved by repeating the above stack of layers-Equivalent to treating the iterative mean-field inference as a Recurrent Neural Network (RNN).
CRF RNN - building the network

Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.
CRF RNN - building the network

• The network is given by the following
  (T is the number of mean-field iterations):

\begin{align*}
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}
\end{align*}

\(f_\theta\) - one iteration of the mean-field algorithm

\(U\) - Pixel-Wise Unary Potential Values (from FCN)

\(\theta\) - the parameter vector (kernels and labels)
CRF RNN- training the network

• The system is trainable using the back-prop algorithm, during the backward pass and SGD.

• One forward pass through the network consists of going once through the FCN, T times through the RNN, and finally to the Softmax Loss Layer.

• It was shown that in practice, after around 5 iterations the mean-field algorithm gave satisfying results. Increasing the number of iterations usually does not improve the result significantly.

• In this case there is no danger suffering from the vanishing/exploding gradient problem- common upon RNN- we can use a simple RNN.
CRF RNN - inside the network

Fully convolutional network stage (FCN-8):
Provides the unary potential for the CRF
The output is blurry and inaccurate
The experiments

Data sets:
• Pascal VOC2012 dataset - training data and validation set
• Pascal Context dataset are used (larger number of classes considered)
• Microsoft COCO dataset is used

Comparison to:
• FCN-8 without CRF
• Other methods
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Without COCO</th>
<th>With COCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain FCN-8s</td>
<td>61.3</td>
<td>68.3</td>
</tr>
<tr>
<td>FCN-8s and CRF disconnected</td>
<td>63.7</td>
<td>69.5</td>
</tr>
<tr>
<td><strong>End-to-end training of CRF-RNN</strong></td>
<td>69.6</td>
<td><strong>72.9</strong></td>
</tr>
</tbody>
</table>

---

**Object appearing in the image:**

- **Helmet**
- **Fence**
- **Shoes**
Conclusions

• the proposed CRF-RNN can be plugged in as a part of a traditional deep neural network: It is capable of passing on error differentials from its outputs to inputs during back-propagation based training of the deep network while learning CRF parameters

• The system achieves a new state-of-the-art on popular dataset benchmarks