Semantic Segmentation

Asaf Bar Zvi
Adi Hayat
Today’s Topics

• Fully Convolutional Networks (FCN) (CVPR 2015)

• Conditional Random Fields as Recurrent Neural Networks (ICCV 2015)

• Gaussian Conditional random field (CVPR 2016)
Some motivation

VR & AR Market
Some motivation
Fully Convolutional Networks for Semantic Segmentation

Evan Shelhamer*  Jonathan Long*  Trevor Darrell
Overview

- Reinterpret standard classification convnets as “Fully convolutional” networks (FCN) for semantic segmentation
- Use transfer learning on AlexNet, VGG, and GoogleNet for experiments
- Novel architecture: combine information from different layers for segmentation (‘deep jet’)
- Inference less than one fifth of a second for a typical image
- State-of-the-art segmentation for PASCAL VOC 2011/2012, NYUDv2, and SIFT Flow at the time
Pixels In, Pixels Out...

Semantic Segmentation

Monocular Depth + Normals (Eigen & Fergus 2015)

Optical Flow (Fischer et al. 2015)

Boundary Prediction (Xie & Tu 2015)

Colorization (Zhang et al. 2016)
Convnets perform classification

< 1 millisecond

1000-dim vector

“tabby cat”

end-to-end learning
R-CNN does detection

Many seconds

R-CNN

“dog”

“cat”
How to make semantic segmentation?

< 1/5 second

end-to-end learning
From classifier to dense FCN

- Convolutionalize proven classification architectures: AlexNet, VGG, and GoogLeNet (reimplementation)
  - Remove classification layer and convert all fully connected layers to convolutions (same as using kernels that cover the entire last convolution layer regions)
  - Adapt architecture to get images at any input size and produce coarse prediction output map downsample by factor of 32
  - Append 1x1 convolution with 21 channels dimensions (20 categories + background for PASCAL) and predict scores at each of the coarse output locations
From classifier to dense FCN

- Convolutionalize proven classification architectures: AlexNet, VGG, and GoogLeNet (reimplementation)
- Remove classification layer and convert all fully connected layers to convolutions (same as using kernels that cover the entire last convolution layer regions)
- Adapt architecture to get images at any input size and produce coarse prediction output map downsample by factor of 32
- Append 1x1 convolution with 21 channels dimensions (20 categories + background for PASCAL) and predict scores at each of the coarse output locations
From classifier to dense FCN

- Convolutionalize proven classification architectures: AlexNet, VGG, and GoogLeNet (reimplementation)
- Remove classification layer and convert all fully connected layers to convolutions (same as using kernels that cover the entire last convolution layer regions)
- Adapt architecture to get images at any input size and produce coarse prediction output map downsample by factor of 32
- Append 1x1 convolution with 21 channels dimensions (20 categories + background for PASCAL) and predict scores at each of the coarse output locations
From classifier to dense FCN

- Convolutionalize proven classification architectures: AlexNet, VGG, and GoogLeNet (reimplementation)
- Remove classification layer and convert all fully connected layers to convolutions (same as using kernels that cover the entire last convolution layer regions)
- Adapt architecture to get images at any input size and produce coarse prediction output map downsample by factor of 32
- Append 1x1 convolution with 21 channels dimensions (20 categories + background for PASCAL) and predict scores at each of the coarse output locations

![Convolution Diagram](image-url)
Dense Predictions

- Shift-and-stitch: trick that yields dense predictions without interpolation. If the output is downscaled by a factor of $F$, shift the input image, $x$ pixels to the right and $y$ pixels down, once for every $(x, y)$ s.t. $0 < x, y < F$. Used in preliminary experiments, but not included in final model.
- Final model uses upsampling via deconvolution filter, initialized with bilinear upsampling weights and learned nonlinear upsampling. Upsampling found to be more effective and efficient.
- Finally, we get end-to-end, pixels-to-pixels network.
Dense Predictions

- **Shift-and-stitch**: trick that yields dense predictions without interpolation. If the output is downsampled by a factor of $F$, shift the input image, $x$ pixels to the right and $y$ pixels down, once for every $(x, y)$ s.t. $0 < x, y < F$. Used in preliminary experiments, but not included in final model.

- Final model use upsampling via deconvolution filter, initialized with bilinear upsampling weights and learned nonlinear upsampling. Upsampling found to be more effective and efficient.

- Finally we get end-to-end, pixels-to-pixels network.
Dense Predictions

• Shift-and-stitch: trick that yields dense predictions without interpolation. If the output is downsampled by a factor of $F$, shift the input image, $x$ pixels to the right and $y$ pixels down, once for every $(x, y)$ s.t. $0 < x, y < F$. Used in preliminary experiments, but not included in final model

• Final model use upsampling via deconvolution filter, initialized with bilinear upsampling weights and learned nonlinear upsampling. Upsampling found to be more effective and efficient

Finally we get end-to-end, pixels-to-pixels network
Choosing the classification net for casting

Cast ILSVRC classifiers into FCNs and compare performance on validation set of PASCAL 2011

<table>
<thead>
<tr>
<th></th>
<th>FCN-AlexNet</th>
<th>FCN-VGG16</th>
<th>FCN-GoogLeNet$^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean IU</td>
<td>39.8</td>
<td>56.0</td>
<td>42.5</td>
</tr>
<tr>
<td>forward time</td>
<td>50 ms</td>
<td>210 ms</td>
<td>59 ms</td>
</tr>
<tr>
<td>conv. layers</td>
<td>8</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>parameters</td>
<td>57M</td>
<td>134M</td>
<td>6M</td>
</tr>
<tr>
<td>rf size</td>
<td>355</td>
<td>404</td>
<td>907</td>
</tr>
<tr>
<td>max stride</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>
Spectrum of deep features

combine *where* (local, shallow) with *what* (global, deep)

(fuse features into deep jet)

(cf. Hariharan et al. CVPR15 “hypercolumn”)
Skip layers architecture

32x upsampled prediction (FCN-32s) → 2x upsampled prediction → 16x upsampled prediction (FCN-16s) → 2x upsampled prediction → 8x upsampled prediction (FCN-8s)

pool3 → pool4 → pool5
Comparison of skip FCNs

- **Input Image**
- **FCN-32s**: no skips
- **FCN-16s**: 1 skip
- **FCN-8s**: 2 skips
- **Ground truth**
Comparison of skip FCNs

Results on subset of validation set of PASCAL VOC 2011

<table>
<thead>
<tr>
<th></th>
<th>pixel acc.</th>
<th>mean IU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN-32s-fixed</td>
<td>83.0</td>
<td>45.4</td>
</tr>
<tr>
<td>FCN-32s</td>
<td>89.1</td>
<td>59.4</td>
</tr>
<tr>
<td>FCN-16s</td>
<td>90.0</td>
<td>62.4</td>
</tr>
<tr>
<td>FCN-8s</td>
<td><strong>90.3</strong></td>
<td><strong>62.7</strong></td>
</tr>
</tbody>
</table>
Training + Testing

- Train full image at a time, with significant filters overlapping, without *patch sampling*, found to be more effective and efficient
- Inference time is ~150ms for 500 x 500 x 21 output
Results – PASCAL VOC 2011/12

VOC 2011: 8498 training images (from additional labeled data)

<table>
<thead>
<tr>
<th>Model</th>
<th>mean IU VOC2011 test</th>
<th>mean IU VOC2012 test</th>
<th>inference time</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN [12]</td>
<td>47.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDS [16]</td>
<td>52.6</td>
<td>51.6</td>
<td>~ 50 s</td>
</tr>
<tr>
<td>FCN-8s</td>
<td>62.7</td>
<td>62.2</td>
<td>~ 175 ms</td>
</tr>
</tbody>
</table>
Results – SIFT Flow

2688 images with pixel labels → 33 semantic categories, 3 geometric categories
Learn both label spaces jointly → learning and inference have similar performance
and computation as independent models

<table>
<thead>
<tr>
<th></th>
<th>pixel acc.</th>
<th>mean acc.</th>
<th>mean IU</th>
<th>f.w. IU</th>
<th>geom. acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liu et al. [23]</td>
<td>76.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tighe et al. [33]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tighe et al. [34]</td>
<td>75.6</td>
<td>41.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tighe et al. [34]</td>
<td>78.6</td>
<td>39.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Farabet et al. [8]</td>
<td>72.3</td>
<td>50.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Farabet et al. [8]</td>
<td>78.5</td>
<td>29.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pinheiro et al. [28]</td>
<td>77.7</td>
<td>29.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FCN-16s</td>
<td>85.2</td>
<td>51.7</td>
<td>39.5</td>
<td>76.1</td>
<td>94.3</td>
</tr>
</tbody>
</table>
Relative to prior state-of-the-art SDS:

- 30% relative improvement for mean IoU
- 286× faster

*Simultaneous Detection and Segmentation Hariharan et al. ECCV14*
Midway conclusions

FCN for semantic segmentation are fast, end-to-end models with quite good results...

Can we do better?
Conditional Random Fields as Recurrent Neural Networks

ICCV2015, won best demo
Pascal 2015 Segmentation benchmark results:

Some quantitative results:

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Mean Intersection-Union</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oxford TVG CRF RNN COCO</td>
<td>74.7</td>
</tr>
<tr>
<td>DeepLab-CRF-COCO-LargeFOV</td>
<td>72.7</td>
</tr>
<tr>
<td>Oxford TVG CRF RNN VOC (trained on VOC 2012)</td>
<td>72.0</td>
</tr>
<tr>
<td>DeepLab-MSc-CRF-LargeFOV</td>
<td>71.6</td>
</tr>
<tr>
<td>MSRA-BoxSup (trained on COCO)</td>
<td>71.0</td>
</tr>
<tr>
<td>Context Deep CNN CRF</td>
<td>70.7</td>
</tr>
<tr>
<td>DeepLab-CRF-COCO-Strong</td>
<td>70.4</td>
</tr>
<tr>
<td>DeepLab-CRF-LargeFOV</td>
<td>70.3</td>
</tr>
<tr>
<td>DeepLab-CRF-MSc</td>
<td>67.1</td>
</tr>
<tr>
<td>DeepLab-CRF</td>
<td>66.4</td>
</tr>
<tr>
<td>CRF RNN (trained on VOC 2011)</td>
<td>65.2</td>
</tr>
<tr>
<td>TTI zoomout 16</td>
<td>64.4</td>
</tr>
<tr>
<td>hypercolumn</td>
<td>62.6</td>
</tr>
<tr>
<td>FCN-8s</td>
<td>62.2</td>
</tr>
<tr>
<td>MSRA CFM</td>
<td>61.8</td>
</tr>
<tr>
<td>TTI zoomout</td>
<td>58.4</td>
</tr>
<tr>
<td>SDS</td>
<td>51.6</td>
</tr>
</tbody>
</table>
Overview

• Combines the strengths of Convolutional Neural Networks (CNNs) and Conditional Random Fields (CRFs), to refine weak and coarse pixel-level label predictions to produce sharp boundaries and fine-grained segmentations

• Formulate CRFs with Gaussian pairwise potentials (filters) and mean-field approximate inference as Recurrent Neural Networks (RNNs)

• end-to-end training network, avoiding offline post-processing methods for object delineation
Conditional random field, Overview

“CRFs are a type of discriminative undirected probabilistic graphical model. It is used to encode known relationships between observations and construct consistent interpretations. It is often used for labeling or parsing of sequential data, such as natural language text or biological sequences[1] and in computer vision.” (Wikipedia)
Conditional random field, for image segmentation

- CRFs typically involve a **local** potential and a **pairwise potential**.
- The local potential is **usually the output of a pixelwise classifier** applied to an image such as FCN. The result is usually not smooth.
- The pairwise potential favors pixel neighbors which don’t have an image gradient between them to have the same label.
- Finally an inference algorithm is ran which finds the best setting of labels to pixels.
- CRFs are used for either foreground estimation or scene parsing.
CRF Model AND Inference Problem

Given a graph $G = (V,E)$, where $V = \{X_1,X_2, \ldots ,X_N\}$, and a global observation (image) $I$, the pair $(I,X)$ can be modelled as a CRF characterized by a Gibbs distribution of the form:

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

Here $E(x)$ is called the energy of the configuration $x \in \mathcal{L}^N$ and $Z(I)$ is the partition function.
CRF Model AND Inference Problem

Given a graph $G = (V,E)$, where $V = \{X_1, X_2, \ldots, X_N\}$, and a global observation (image) $I$, the pair $(I, X)$ can be modelled as a CRF characterized by a Gibbs distribution of the form:

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

Here $E(x)$ is called the energy of the configuration $x \in \mathcal{L}^N$ and $Z(I)$ is the partition function.

MAP inference:

$$X^* = \arg \max_X P(X \mid I)$$

In the fully connected pairwise CRF model of [29], the energy of a label assignment $x$ is given by:

$$E(x) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j), \quad (1)$$

where the unary energy components $\psi_u(x_i)$ measure the inverse likelihood (and therefore, the cost) of the pixel $i$ taking the label $x_i$, and pairwise energy components $\psi_p(x_i, x_j)$ measure the cost of assigning labels $x_i, x_j$ to pixels $i, j$ simultaneously.
CRF Model AND Inference Problem

Given a graph $G = (V, E)$, where $V = \{X_1, X_2, \ldots, X_N\}$, and a global observation (image) $I$, the pair $(I, X)$ can be modelled as a CRF characterized by a Gibbs distribution of the form:

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

Here $E(x)$ is called the energy of the configuration $x \in \mathcal{L}^N$ and $Z(I)$ is the partition function.

**MAP inference:**

$$X^* = \arg\max_X P(X | I)$$

**Gaussian kernel (m) weight**

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k^{(m)}_G(f_i, f_j), \quad (2)$$

**Label compatibility**

$$\mu(x_i, x_j) = 1 \text{ if } l_{x_i} \neq l_{x_j} \text{ else } 0$$

**Gaussian kernel applied on a feature vector**
Given a graph $G = (V,E)$, where $V = \{X_1, X_2, \ldots, X_N\}$, and a global observation (image) $I$, the pair $(I,X)$ can be modelled as a CRF characterized by a Gibbs distribution of the form:

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

Here $E(x)$ is called the energy of the configuration $x \in \mathcal{L}^N$ and $Z(I)$ is the partition function.

**MAP Inference:**

$$X^* = \arg\max_X P(X | I)$$

**Gaussian Kernel (m) Weight**

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{M} w(m) k_G^{(m)}(f_i, f_j), \quad (2)$$

**Label Compatibility**

$$\mu(x_i, x_j) = 1 \text{ if } [l_i \neq l_j] \text{ else } 0$$

**CRF Conditioning**

**Gaussian kernel applied on a feature vector**
Given a graph $G = (V,E)$, where $V = \{X_1, X_2, \ldots, X_N\}$, and a global observation (image) $I$, the pair $(I,X)$ can be modelled as a CRF characterized by a Gibbs distribution of the form:

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

Here $E(x)$ is called the energy of the configuration $x \in \mathcal{L}^N$ and $Z(I)$ is the partition function.

**MAP inference:**

$$X^* = \arg\max_X P(X | I)$$

*Intractable, Very hard to solve*

**Dense CRF**
Mean field approximation

The effect of all the other pixels on any given pixel is approximated by a single averaged effect.
Mean field approximation - analytic derivation

The Gibbs distribution is given by

\[ P(X) = \frac{1}{Z} \tilde{P}(X) = \frac{1}{Z} \exp \left( \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \right) \]

where the partition function is defined as \( Z = \sum_X \tilde{P}(x) \).

**Mean-Field:**

Find the **target model**

subject to

\[ \{Q(X_i)\} \]

\[ F[\tilde{P}_\Phi, Q] = -D(Q || P) = E_Q \left( \log \frac{Q(X)}{P(X)} \right) \]

\[ Q(X) = \prod_i Q_i(X_i) \]

\[ \sum_{x_i} Q(x_i) = 1 \quad \forall i. \]
Mean field approximation - analytic derivation

The Gibbs distribution is given by

$$P(X) = \frac{1}{Z} \tilde{P}(X) = \frac{1}{Z} \exp \left( \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \right)$$

where the partition function is defined as $Z = \sum_x \tilde{P}(x)$.

$$D(Q||P) = \sum_x Q(x) \log \left( \frac{Q(x)}{P(x)} \right)$$

$$= - \sum_x Q(x) \log P(x) + \sum_x Q(x) \log Q(x)$$

$$= - \mathbb{E}_{U \sim Q}[\log P(U)] + \mathbb{E}_{U \sim Q}[\log Q(U)]$$

$$= - \mathbb{E}_{U \sim Q}[\log \tilde{P}(U)] + \mathbb{E}_{U \sim Q}[\log Z] + \sum_i \mathbb{E}_{U_i \sim Q_i}[\log Q_i(U_i)]$$

$$= \mathbb{E}_{U \sim Q}[E(U)] + \sum_i \mathbb{E}_{U_i \sim Q_i}[\log Q_i(U_i)] + \log Z$$
Mean field approximation - analytic derivation

The Gibbs distribution is given by

\[ P(X) = \frac{1}{Z} \tilde{P}(X) = \frac{1}{Z} \exp \left( \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \right) \]

where the partition function is defined as \( Z = \sum_x \tilde{P}(x) \).

\[ D(Q\|P) = \sum_x Q(x) \log \left( \frac{Q(x)}{P(x)} \right) \]

\[ = -\sum_x Q(x) \log P(x) + \sum_x Q(x) \log Q(x) \]

\[ = -E_{U \sim Q}[\log P(U)] + E_{U \sim Q}[\log Q(U)] \]

\[ = -E_{U \sim Q}[\log \tilde{P}(U)] + E_{U \sim Q}[\log Z] + \sum_i E_{U \sim Q}[\log Q(U_i)] \]

\[ = E_{U \sim Q}[E(U)] + \sum_i E_{U \sim Q_i}[\log Q_i(U_i)] + \log Z \]
Mean field approximation - analytic derivation

The Gibbs distribution is given by

$$P(X) = \frac{1}{Z} \hat{P}(X) = \frac{1}{Z} \exp \left( \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \right)$$

where the partition function is defined as $Z = \sum_X \hat{P}(x)$.

$$D(Q||P) = \sum_x Q(x) \log \left( \frac{Q(x)}{P(x)} \right)$$

$$= - \sum_x Q(x) \log P(x) + \sum_x Q(x) \log Q(x)$$

$$= -E_{U \sim Q}[\log P(U)] + E_{U \sim Q}[\log Q(U)]$$

$$= -E_{U \sim Q}[\log \hat{P}(U)] + E_{U \sim Q}[\log Z] + \sum_i E_{\hat{U}_i \sim Q}[\log Q(U_i)]$$

$$= E_{U \sim Q}[E(U)] + \sum_i E_{U_i \sim Q_i}[\log Q_i(U_i)] + \log Z + \lambda \left( \sum_i Q_i(X_i) - 1 \right)$$

Lagrange multipliers ensuring the marginals $Q_i(X_i)$ are probability distributions.
Mean field approximation - analytic derivation

The Gibbs distribution is given by

\[
P(X) = \frac{1}{Z} \hat{P}(X) = \frac{1}{Z} \exp \left( \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \right)
\]

where the partition function is defined as \( Z = \sum_x \hat{P}(x) \).

\[
D(Q \| P) = \sum_x Q(x) \log \left( \frac{Q(x)}{P(x)} \right)
\]

\[
= -\sum_x Q(x) \log P(x) + \sum_x Q(x) \log Q(x)
\]

\[
= -E_{U \sim Q}[\log P(U)] + E_{U \sim Q}[\log Q(U)]
\]

\[
= -E_{U \sim Q}[\log \hat{P}(U)] + E_{U \sim Q}[\log Z] + \sum_i E_{U_i \sim Q}[\log Q_i(U_i)]
\]

\[
= E_{U \sim Q}[E(U)] + \sum_i E_{U_i \sim Q_i}[\log Q_i(U_i)] + \log Z + \lambda \left( \sum_i Q_i(X_i) - 1 \right)
\]

Lagrange multipliers ensuring the marginals \( Q_i(X_i) \) are probability distributions.
Mean field approximation - analytic derivation

The Gibbs distribution is given by

\[ P(X) = \frac{1}{Z} \hat{P}(X) = \frac{1}{Z} \exp \left( \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \right) \]

where the partition function is defined as \( Z = \sum_x \hat{P}(x) \).

\[ Q_i(x_i) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{j \neq i} E_{U_j \sim Q_j} [\psi_p(x_i, U_j)] \right\} \tag{5} \]

Substituting the definition of the pairwise potential (Eq. 2) into the mean field update in Equation 5 yields the following formulation of the update equation, which is used in the paper.

\[ Q_i(x_i = l) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{j \neq i} E_{U_j \sim Q_j} \left[ \sum_{m=1}^K w^{(m)} k^{(m)}(f_i, f_j) \right] \right\} \]

\[ = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{m=1}^K w^{(m)} \sum_{j \neq i} E_{U_j \sim Q_j} \left[ \mu(l, U_j) \sum_{m=1}^K k^{(m)}(f_i, f_j) \right] \right\} \]

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

\[ = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^K w^{(m)} \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_{j'}(l') \right\} \tag{6} \]
Mean field approximation - analytic derivation

The Gibbs distribution is given by

$$P(X) = \frac{1}{Z} \hat{P}(X) = \frac{1}{Z} \exp \left( \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \right)$$

where the partition function is defined as $Z = \sum_x \hat{P}(x)$.

$$Q_i(x_i) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{j \neq i} \mathbb{E}_{U_j \sim Q_j} [\psi_p(x_i, U_j)] \right\}$$  \hspace{1cm} (5)

Substituting the definition of the pairwise potential (Eq. 2) into the mean field update in Equation 5 yields the following formulation of the update equation, which is used in the paper.

$$Q_i(x_i = l) = \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{j \neq i} \mathbb{E}_{U_j \sim Q_j} \left[ \mu(l, U_j) \sum_{m=1}^{K} w^{(m)} k^{(m)}(f_i, f_j) \right] \right\}$$

$$= \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{m=1}^{K} w^{(m)} \sum_{j \neq i} \mathbb{E}_{U_j \sim Q_j} \left[ \mu(l, U_j) k^{(m)}(f_i, f_j) \right] \right\}$$

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

$$= \frac{1}{Z_i} \exp \left\{ -\psi_u(x_i) - \sum_{l' \in \mathcal{L}} \sum_{m=1}^{K} w^{(m)} \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \right\}$$  \hspace{1cm} (6)
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, NIPS 2011

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

Algorithm 1 Mean field in fully connected CRFs

1. Initialize \( Q_i \)
2. while not converged do
   3. \( \tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l) \) for all \( m \)
   4. \( Q_i(x_i) \leftarrow \sum_{l \in \mathcal{L}} \mu^{(m)}(x_i, l) \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l) \)
   5. \( Q_i(x_i) \leftarrow \exp \{-\psi_u(x_i) - \tilde{Q}_i(x_i)\} \)
   6. normalize \( Q_i(x_i) \)
   7. end while

\( Q_i(x_i) \leftarrow \frac{1}{Z_i} \exp \{-\phi_u(x_i)\} \)

See Section 6 for convergence analysis

Message passing from all \( X_j \) to all \( X_i \)

Compatibility transform

Local update

Each iteration of Algorithm 1 performs a message passing step, a compatibility transform, and a local update. Both the compatibility transform and the local update run in linear time and are highly efficient. The computational bottleneck is message passing. For each variable, this step requires evaluating a sum over all other variables. A naive implementation thus has quadratic complexity in the number of variables \( N \). Next, we show how approximate high-dimensional filtering can be used to reduce the computational cost of message passing to linear.
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, NIPS 2011

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

**Algorithm 1** Mean field in fully connected CRFs

1. Initialize \( Q \)
2. **while** not converged **do**
   - \( \tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l) \) for all \( m \)
   - \( Q_i(x_i) \leftarrow \sum_{l \in \mathcal{L}} \mu^{(m)}(x_i, l) \sum_{m} w^{(m)} \tilde{Q}_i^{(m)}(l) \)
   - \( Q_i(x_i) \leftarrow \exp\{-\psi_u(x_i) - \tilde{Q}_i(x_i)\} \)
3. **end while**

Each iteration of Algorithm 1 performs a message passing step, a compatibility transform, and a local update. Both the compatibility transform and the local update run in linear time and are highly efficient. The computational bottleneck is message passing. For each variable, this step requires evaluating a sum over all other variables. A naive implementation thus has quadratic complexity in the number of variables \( N \). Next, we show how approximate high-dimensional filtering can be used to reduce the computational cost of message passing to linear.
CRF as RNN, key contribution?

“A key contribution of this paper is to show that the mean-field CRF inference can be reformulated as a Recurrent Neural Network (RNN).”
CRF as RNN, key contribution?

“A key contribution of this paper is to show that the mean-field CRF inference can be reformulated as a Recurrent Neural Network (RNN).”

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

- $Q_i(l) \leftarrow \frac{1}{Z_i} \exp \left( U_i(l) \right)$ for all $i$, $\triangleright$ Initialization

while not converged do

- $\hat{Q}^{(m)}_i(l) \leftarrow \sum_{j \neq i} k^{(m)}(f_i, f_j)Q_j(l)$ for all $m$, $\triangleright$ Message Passing

- $\hat{Q}_i(l) \leftarrow \sum_m w^{(m)}\hat{Q}^{(m)}_i(l)$, $\triangleright$ Weighting Filter Outputs

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

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

- $Q_i \leftarrow \frac{1}{Z_i} \exp \left( \hat{Q}_i(l) \right)$, $\triangleright$ Normalizing
CRF as RNN, key contribution?

“A key contribution of this paper is to show that the mean-field CRF inference can be reformulated as a Recurrent Neural Network (RNN).”

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

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

```plaintext

<table>
<thead>
<tr>
<th>Step</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization: $$Q_i(l) \leftarrow \frac{1}{Z_i} \exp (U_i(l))$$ for all $$i$$</td>
<td></td>
</tr>
<tr>
<td>Message Passing: $$\hat{Q}<em>i(m)(l) \leftarrow \sum</em>{j \neq i} k^{(m)}(f_i, f_j) Q_j(l)$$ for all $$m$$</td>
<td></td>
</tr>
<tr>
<td>Message Passing: $$Q_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \hat{Q}_i(l')$$</td>
<td></td>
</tr>
<tr>
<td>Weighting Filter Outputs: $$\hat{Q}_i(l) \leftarrow U_i(l) - \hat{Q}_i(l)$$</td>
<td></td>
</tr>
<tr>
<td>Adding Unary Potentials: $$Q_i \leftarrow \frac{1}{Z_i} \exp (\hat{Q}_i(l))$$</td>
<td></td>
</tr>
<tr>
<td>Normalizing: $$Z_i$$</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 1. A mean-field iteration as a CNN. A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.
CRF as RNN, key contribution?

“A key contribution of this paper is to show that the mean-field CRF inference can be reformulated as a Recurrent Neural Network (RNN).”

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

**Figure 1.** A mean-field iteration as a CNN. A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.
CRF as RNN, key contribution?

“A key contribution of this paper is to show that the mean-field CRF inference can be reformulated as a Recurrent Neural Network (RNN).”

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

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

- Initialization: \( Q_i(l) \leftarrow \frac{1}{Z_i} \exp \left( U_i(l) \right) \) for all \( i \)
- Message Passing: \( \tilde{Q}_i(l) \leftarrow \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l) \) for all \( m \)
- Weighting Filter Outputs: \( \tilde{Q}_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \tilde{Q}_i(l') \)
- Compatibility Transform: \( \tilde{Q}_i(l) \leftarrow U_i(l) - \tilde{Q}_i(l) \)
- Adding Unary Potentials: \( Q_i \leftarrow \frac{1}{Z_i} \exp \left( \tilde{Q}_i(l) \right) \)
- Normalizing

Figure 1. A mean-field iteration as a CNN. A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.
CRF as RNN, key contribution?

“A key contribution of this paper is to show that the mean-field CRF inference can be reformulated as a Recurrent Neural Network (RNN).”

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

Figure 1. A mean-field iteration as a CNN. A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.
CRF as RNN, key contribution?

“A key contribution of this paper is to show that the mean-field CRF inference can be reformulated as a Recurrent Neural Network (RNN).”

Figure 1. A mean-field iteration as a CNN. A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.
CRF as RNN, Architecture:

Two parts:

Figure 3. The End-to-end Trainable Network. Schematic visualization of our full network which consists of a CNN and the CNN-CRF network. Best viewed in colour.
CRF as RNN, Architecture:

Two parts:

Figure 3. The End-to-end Trainable Network. Schematic visualization of our full network which consists of a CNN and the CNN-CRF network. Best viewed in colour.
CRF as RNN, Architecture:

Two parts:

Figure 2. The CRF-RNN Network. We formulate the iterative mean-field algorithm as a Recurrent Neural Network (RNN). Gating functions $G_1$ and $G_2$ are fixed as described in the text.

Figure 3. The End-to-end Trainable Network. Schematic visualization of our full network which consists of a CNN and the CNN-CRF network. Best viewed in colour.
Mean Field iteration Visualization
Mean Field iteration Visualization
CRF as RNN, Architecture: (as observed in code)

\[ Q_1(x_i = l) = \frac{1}{Z_i} \exp \left\{ -\psi_i(x_i) - \sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^{K} w^{(m)} \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \right\} \]
CRF as RNN, Architecture: (as observed in code)

\[
Q_i(x_i = l) = \frac{1}{\mathcal{Z}_i} \exp \left\{ -\psi_i(x_i) - \sum_{i' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^{K} \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \right\}
\]
CRF as RNN, Architecture: (as observed in code)

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

- **Spatial**: The learned weights
- **Bilateral**: The learned weights

**Penalty (spatial)** of pixel \( N \) being assigned to \( c=1 \)

\[ \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \]

**Penalty (spatial)** of pixel \( 1 \) being assigned to \( c=C \)

**Penalty (Bilateral)** of pixel \( N \) being assigned to \( c=1 \)

**Penalty (Bilateral)** of pixel \( 1 \) being assigned to \( c=C \)

**1x1 Conv**
CRF as RNN, Architecture: (as observed in code)
CRF as RNN, Architecture: (as observed in code)

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

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

The learned weights

**Spatial**

\[
\begin{pmatrix}
  w_{1,1} & w_{1,2} & \ldots & w_{1,N} \\
  w_{2,1} & w_{2,2} & \ldots & \vdots \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{N,1} & w_{N,2} & \ldots & w_{N,N}
\end{pmatrix}
\]

Total Message Penalty of pixel \( N \) being assigned to \( c = 1 \)

Total Message Penalty of pixel \( 1 \) being assigned to \( c = C \)

**Bilateral**

\[
\begin{pmatrix}
  w_{b,1,1} & w_{b,1,2} & \ldots & w_{b,1,N} \\
  w_{b,2,1} & w_{b,2,2} & \ldots & \vdots \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{b,N,1} & w_{b,N,2} & \ldots & w_{b,N,N}
\end{pmatrix}
\]

Penalty (spatial) of pixel \( N \) being assigned to \( c = 1 \)

Penalty (spatial) of pixel \( 1 \) being assigned to \( c = C \)

Penalty (Bilateral) of pixel \( N \) being assigned to \( c = C \)

Penalty (Bilateral) of pixel \( 1 \) being assigned to \( c = C \)
CRF as RNN, Architecture: (as observed in code)
CRF as RNN, Architecture: (as observed in code)
If I'm a person pixel, I'm likely to be near: a chair, a sofa.

We are popular in pascal VOC.
CRF as RNN, Architecture: (as observed in code)

If I’m a person pixel, I’m likely to be near:
- a chair
- a sofa

We are popular in pascal VOC

If I’m near a person I’m likely to be:
- A bus
- A car
- A cat
- A horse
Some results

Experiments

PASCAL VOC

We achieved state-of-the-art comparable performance (mean intersection-over-union 74.7%) in PASCAL VOC 2012 semantic image segmentation benchmark. Further work [1] achieved 77.9%. CRF-RNN GPU version takes less than 0.4 seconds for processing an image with resolution 500 x 500.

Figure 10: Qualitative comparison with other approaches. Sample results with our method on the validation set of the PASCAL VOC 2012 dataset, compared with previous state-of-the-art methods. Segmentation results with DeepLab approach were reproduced from the original publication. Best viewed in color.
Experiments

PASCAL VOC

We achieved state-of-the-art comparable performance (mean intersection-over-union 74.7%) in PASCAL VOC 2012 semantic image segmentation benchmark. Further work [1] achieved 77.9%. CRF-RNN GPU version takes less than 0.4 seconds for processing an image with resolution 500 × 500.
Gaussian Conditional Random Field Network for Semantic Segmentation

Raviteja Vemulapalli, Rama Chellappa
University of Maryland, College Park

Oncel Tuzel, Ming-Yu Liu
Mitsubishi Electric Research Laboratories
Gaussian Conditional random field model

While a discrete CRF is a natural fit for labeling tasks such as semantic segmentation, one needs to use inference techniques that do not have optimality guarantees. **While exact inference is tractable in the case of a Gaussian CRF,** it is not clear if this model is a good fit for discrete labeling tasks.

$$P(y|X) \propto \exp \left\{ -\frac{1}{2} E(y|X) \right\}, \text{ where}$$

$$E(y|X) = \sum_i \|y_i - r_i(X; \theta_u)\|_2^2$$

$$+ \sum_{ij} (y_i - y_j)^\top W_{ij}(X; \theta_p) (y_i - y_j).$$
While a discrete CRF is a natural fit for labeling tasks such as semantic segmentation, one needs to use inference techniques that do not have optimality guarantees. While exact inference is tractable in the case of a Gaussian CRF, it is not clear if this model is a good fit for discrete labeling tasks.

This leads us to the following question: Should we use a better model with approximate inference or an approximate model with better inference?
Gaussian Conditional random field model

While a discrete CRF is a natural fit for labeling tasks such as semantic segmentation, one needs to use inference techniques that do not have optimality guarantees. **While exact inference is tractable in the case of a Gaussian CRF,** it is not clear if this model is a good fit for discrete labeling tasks.

This leads us to the following question: **Should we use a better model with approximate inference or an approximate model with better inference?**
Gaussian Conditional random field model

While a discrete CRF is a natural fit for labeling tasks such as semantic segmentation, one needs to use inference techniques that do not have optimality guarantees. While exact inference is tractable in the case of a Gaussian CRF, it is not clear if this model is a good fit for discrete labeling tasks.

This leads us to the following question: Should we use a better model with approximate inference or an approximate model with better inference?

\[
P(y|X) \propto \exp \left\{ -\frac{1}{2} E(y|X) \right\}, \text{ where}
\]

\[
E(y|X) = \sum_i \|y_i - r_i(X; \theta_u)\|^2_2 
+ \sum_{ij} (y_i - y_j)^\top W_{ij}(X; \theta_p) (y_i - y_j).
\]

Semi positive definite
Should encode both pixels and label compatibility
Gaussian Conditional random field model

GCRF inference can be solved analytically:

\[
\frac{d}{dy} \rightarrow y^T A y + B y + C
\]

\[
P(y|X) \propto \exp \left\{ -\frac{1}{2} E(y|X) \right\}, \text{ where}
\]

\[
E(y|X) = \sum_i \| y_i - r_i(X; \theta_u) \|^2_2 + \sum_{ij} (y_i - y_j)^T W_{ij}(X; \theta_p) (y_i - y_j).
\]

Semi positive definite

Should encode both pixels and label compatibility
Gaussian Conditional random field model

GCRF inference can be solved analytically:

$$\frac{d}{dy} \rightarrow y^T Ay + By + C$$

$$y \in \mathbb{R}^{W \times H \times \text{Labels}}$$

However, this closed form solution involves solving a linear system with number of variables equal to the number of pixels times the number of classes. Since solving such a large linear system could be computationally prohibitive, in this work, we use the iterative mean field inference approach.

In the case of Gaussian, the mean field approximation $Q$ and the original distribution $P$ have the same mean [53]. Hence, finding the MAP solution $y$ is equivalent to finding the mean $\mu$ of the distribution $Q$.

Gaussian Conditional random field model

GCRF inference can be solved analytically:

\[
\frac{d}{dy} y^T Ay + By + C \rightarrow y \\
y \in \mathbb{R}^{W \times H \times \text{Labels}}
\]

However, this closed form solution involves solving a linear system with number of variables equal to the number of pixels times the number of classes. Since solving such a large linear system could be computationally prohibitive, in this work, we use the iterative mean field inference approach.

In the case of Gaussian, the mean field approximation \( Q \) and the original distribution \( P \) have the same mean [53]. Hence, finding the MAP solution \( y \) is equivalent to finding the mean \( \mu \) of the distribution \( Q \).

GCRF Net Arch
“modified version of the popular VGG-16 network”
GCRF Net Arch

Pairwise network

Unary network

DeepLab CNN ($\theta^{\text{CNN}}$) $\rightarrow$ $\mathbf{z}_i$ $\rightarrow$ Similarity layer ($\mathbf{f}_{\text{sim}}$) $\rightarrow$ $\mathbf{s}_{ij}$ $\rightarrow$ Matrix generation layer ($\mathbf{C} > 0$) $\rightarrow$ $\mathbf{W}_{ij}$

Class prediction scores

Bilinear interpolation + Selecting the maximum scoring class

$\mathbf{W}_{ij} = \mathbf{s}_{ij} \mathbf{C}$, $\mathbf{C} \succeq 0$, $\mathbf{C} = \mathbf{R}\mathbf{R}^T$

$\mathbf{s}_{ij} = e^{-\frac{(\mathbf{z}_i - \mathbf{z}_j)^\top \mathbf{F} (\mathbf{z}_i - \mathbf{z}_j)}{2}}$

$\mathbf{F} = \sum_{m=1}^{M} \mathbf{f}_m \mathbf{f}_m^\top$

$(\mathbf{z}_i - \mathbf{z}_j)^\top \mathbf{F} (\mathbf{z}_i - \mathbf{z}_j) = \sum_{m=1}^{M} (\mathbf{f}_m^\top \mathbf{z}_i - \mathbf{f}_m^\top \mathbf{z}_j)^2$
GCRF Net Arch

Bipartite graph structure for parallel updates

Figure 2: Each pixel in our CRF is connected to every other pixel along both rows and columns within a spatial neighborhood. Here, all the pixels that are connected to the center black pixel are shown in red. If the black pixel is on odd column, all the pixels connected to it will be on even columns and vice versa.
Training loss function: For training the network, we use the following loss function at each pixel:

\[ L(y^*_i, l_i) = -\min(0, y^*_i, \max_{k \neq l_i} y^*_k - T), \]  

where \( l_i \) is the true class label. This loss function basically encourages the output associated with the true class to be greater than the output associated with all the other classes by a margin \( T \).
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>bkg</th>
<th>areo</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbk</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSRA-CF-M [9]</td>
<td>87.7</td>
<td>75.7</td>
<td>26.7</td>
<td>69.5</td>
<td>48.8</td>
<td>65.6</td>
<td>81.0</td>
<td>69.2</td>
<td>73.3</td>
<td>30.0</td>
<td>68.7</td>
<td>51.5</td>
<td>69.1</td>
<td>68.1</td>
<td>71.7</td>
<td>67.5</td>
<td>50.4</td>
<td>66.5</td>
<td>44.4</td>
<td>58.9</td>
<td>53.5</td>
<td>61.8</td>
</tr>
<tr>
<td>FCN-8s [30]</td>
<td>91.2</td>
<td>76.8</td>
<td>34.2</td>
<td>68.9</td>
<td>49.4</td>
<td>60.3</td>
<td>75.3</td>
<td>74.7</td>
<td>77.6</td>
<td>21.4</td>
<td>62.5</td>
<td>46.8</td>
<td>71.8</td>
<td>63.9</td>
<td>76.5</td>
<td>73.9</td>
<td>45.2</td>
<td>72.4</td>
<td>37.4</td>
<td>70.9</td>
<td>55.1</td>
<td>62.2</td>
</tr>
<tr>
<td>Hypercolumns [18]</td>
<td>89.3</td>
<td>68.7</td>
<td>33.5</td>
<td>69.8</td>
<td>51.3</td>
<td>70.2</td>
<td>81.1</td>
<td>71.9</td>
<td>74.9</td>
<td>23.9</td>
<td>60.6</td>
<td>46.9</td>
<td>72.1</td>
<td>68.3</td>
<td>74.5</td>
<td>72.9</td>
<td>52.6</td>
<td>64.4</td>
<td>45.4</td>
<td>64.9</td>
<td>57.4</td>
<td>62.6</td>
</tr>
<tr>
<td>DeepLab CNN [7]</td>
<td>91.6</td>
<td>78.7</td>
<td>51.5</td>
<td>75.8</td>
<td>59.5</td>
<td>61.9</td>
<td>82.5</td>
<td>76.6</td>
<td>79.4</td>
<td>26.9</td>
<td>67.7</td>
<td>54.7</td>
<td>74.3</td>
<td>70.0</td>
<td>79.8</td>
<td>77.3</td>
<td>52.6</td>
<td>75.2</td>
<td>46.6</td>
<td>66.9</td>
<td>57.3</td>
<td>67.0</td>
</tr>
<tr>
<td>ZoomOut [31]</td>
<td>91.1</td>
<td>85.6</td>
<td>37.3</td>
<td>83.2</td>
<td>62.5</td>
<td>66.0</td>
<td>85.1</td>
<td>80.7</td>
<td>84.9</td>
<td>27.2</td>
<td>73.2</td>
<td>57.5</td>
<td>78.1</td>
<td>79.2</td>
<td>81.1</td>
<td>77.1</td>
<td>53.6</td>
<td>74.0</td>
<td>49.2</td>
<td>71.7</td>
<td>63.3</td>
<td>69.6</td>
</tr>
<tr>
<td>Deep message passing [27]</td>
<td>93.9</td>
<td>90.1</td>
<td>38.6</td>
<td>77.8</td>
<td>61.3</td>
<td>74.3</td>
<td>89.0</td>
<td>83.4</td>
<td>83.3</td>
<td>36.2</td>
<td>80.2</td>
<td>56.4</td>
<td>81.2</td>
<td>81.4</td>
<td>83.1</td>
<td>82.9</td>
<td>59.2</td>
<td>83.4</td>
<td>54.3</td>
<td>80.6</td>
<td>70.8</td>
<td>73.4</td>
</tr>
</tbody>
</table>

**Approaches that use CNNs and discrete CRFs**

<table>
<thead>
<tr>
<th>Method</th>
<th>bkg</th>
<th>areo</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbk</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep structure models [28]</td>
<td>93.6</td>
<td>86.7</td>
<td>36.9</td>
<td>82.3</td>
<td>63.0</td>
<td>74.2</td>
<td>89.8</td>
<td>84.1</td>
<td>84.1</td>
<td>32.8</td>
<td>65.4</td>
<td>52.1</td>
<td>79.7</td>
<td>72.1</td>
<td>77.6</td>
<td>81.7</td>
<td>55.6</td>
<td>77.4</td>
<td>37.4</td>
<td>81.4</td>
<td>68.4</td>
<td>70.3</td>
</tr>
<tr>
<td>DeconvNet + CRF [32]</td>
<td>92.9</td>
<td>87.8</td>
<td>41.9</td>
<td>80.6</td>
<td>63.9</td>
<td>67.3</td>
<td>88.1</td>
<td>78.4</td>
<td>81.3</td>
<td>25.9</td>
<td>73.7</td>
<td>61.2</td>
<td>72.0</td>
<td>77.0</td>
<td>79.9</td>
<td>78.7</td>
<td>59.5</td>
<td>78.3</td>
<td>55.0</td>
<td>75.2</td>
<td>61.5</td>
<td>70.5</td>
</tr>
<tr>
<td>object clique potentials [37]</td>
<td>92.8</td>
<td>80.0</td>
<td>53.8</td>
<td>80.8</td>
<td>62.5</td>
<td>64.7</td>
<td>87.0</td>
<td>78.5</td>
<td>83.0</td>
<td>29.0</td>
<td>82.0</td>
<td>60.3</td>
<td>76.3</td>
<td>78.4</td>
<td>83.0</td>
<td>79.8</td>
<td>57.0</td>
<td>80.0</td>
<td>53.1</td>
<td>70.1</td>
<td>63.1</td>
<td>71.2</td>
</tr>
<tr>
<td>DeepLab CNN-CRF [7]</td>
<td>93.3</td>
<td>84.4</td>
<td>54.5</td>
<td>81.5</td>
<td>63.6</td>
<td>65.9</td>
<td>85.1</td>
<td>79.1</td>
<td>83.4</td>
<td>30.7</td>
<td>74.1</td>
<td>59.8</td>
<td>79.0</td>
<td>76.1</td>
<td>83.2</td>
<td>80.8</td>
<td>59.7</td>
<td>82.2</td>
<td>50.4</td>
<td>73.1</td>
<td>63.7</td>
<td>71.6</td>
</tr>
<tr>
<td>CRF-RNN [55]</td>
<td>94.0</td>
<td>87.5</td>
<td>39.0</td>
<td>79.7</td>
<td>64.2</td>
<td>68.3</td>
<td>87.6</td>
<td>80.8</td>
<td>84.4</td>
<td>30.4</td>
<td>78.2</td>
<td>60.4</td>
<td>80.5</td>
<td>77.8</td>
<td>83.1</td>
<td>80.6</td>
<td>59.5</td>
<td>82.8</td>
<td>47.8</td>
<td>78.3</td>
<td>67.1</td>
<td>72.0</td>
</tr>
<tr>
<td>DeconvNet + FCN + CRF [32]</td>
<td>93.1</td>
<td>89.9</td>
<td>39.3</td>
<td>79.7</td>
<td>63.9</td>
<td>68.2</td>
<td>87.4</td>
<td>81.2</td>
<td>86.1</td>
<td>28.5</td>
<td>77.0</td>
<td>62.0</td>
<td>79.0</td>
<td>80.3</td>
<td>83.6</td>
<td>80.2</td>
<td>58.8</td>
<td>83.4</td>
<td>54.3</td>
<td>80.7</td>
<td>65.0</td>
<td>72.5</td>
</tr>
<tr>
<td>Proposed GCRF network</td>
<td>93.4</td>
<td>85.2</td>
<td>43.9</td>
<td>83.3</td>
<td>65.2</td>
<td>68.3</td>
<td>89.0</td>
<td>82.7</td>
<td>85.3</td>
<td>31.1</td>
<td>79.5</td>
<td>63.3</td>
<td>80.5</td>
<td>79.3</td>
<td>85.5</td>
<td>81.0</td>
<td>60.5</td>
<td>85.5</td>
<td>52.0</td>
<td>77.3</td>
<td>65.1</td>
<td>73.2</td>
</tr>
</tbody>
</table>

**Computation time:** The proposed GCRF network takes around 0.6 seconds to segment a 505 × 505 image on an NVIDIA TITAN GPU.
That's all Folks!