Semantic Segmentation

Zhongang Qi
qiz@oregonstate.edu
Semantic Segmentation

"Two men riding on a bike in front of a building on the road. And there is a car."

"Two horses are eating grass within the fence. And there are buildings and trees under the sky."

Idea: recognizing, understanding what's in the image in pixel level.

Two sets of label structures (category and object).
Semantic Segmentation

Two sets of label structures (category and object). For the category label, it can be treated as multi-class classification problem.

- Given a 50*50 image, there are 2500 pixel treated as 2500 examples;
- For each example, we aim to find out which class it belongs (one example only belongs to one class);
- Array the 2500 class labels of the 2500 examples as the original image form (50*50), we obtain the semantic label for the given image.
Key point 1: Form the features for each example (pixel), and use them in the multi-class classification.

It is hard to classify the pixel only using the feature of this pixel. Various scales of image regions, even the whole image can be treated as the features of the pixel.
Semantic Segmentation

**Key point 2:** adding the label constraints for different pixels in one image to calculate the relationships between pixels:

- Nearby pixels more likely to have same label
- Pixels with similar color more likely to have same label
- The pixels above the pixels “horse” more likely to be “person” instead of “plane”

Are not nearby, but have similar color
Semantic Segmentation

Most of the existing methods utilize Key point 1 + Key point 2 to solve the semantic segmentation problem.

\[ E(x|I) = \sum_i \phi_u(x_i|I) + \sum_i \sum_{j \in \partial_i} \phi_p(x_i, x_j|I) \]

**Unary term**  
**Pairwise term**

CRF used for semantic segmentation:

- Unary term is used to form the features for each pixel and find the relationship between the features and the classification results;

- Pairwise term is used to add the label constraints for different pixels which incorporate information from various relationships.
Semantic Segmentation

Several existing methods for semantic segmentation:

Before Deep Learning:

- Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, NIPS 2011, Key point 1 + Key point 2 (CRF)

After Deep Learning:

- Fully Convolutional Networks for Semantic Segmentation, CVPR 2015
  mainly focus on Key point 1 (FCN)
- Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs, ICLR 2015
  Key point 1 + Key point 2 (FCN+CRF, separately training)
- Conditional Random Fields as Recurrent Neural Networks, ICCV 2015
  Key point 1 + Key point 2 (FCN+CRF, end to end training)
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

CRF for Semantic Segmentation

\[ E(x|I) = \sum_{i} \phi_u(x_i|I) + \sum_{i} \sum_{j \in \partial i} \phi_p(x_i, x_j|I) \]

Key point 1 + Key point 2

- Unary term
- Pairwise term

\[ X: \text{a random field defined over a set of variables } \{X_1, \ldots, X_N\} \]
- Label of pixels (grass, bench, tree, …)

\[ I: \text{a random field defined over a set of variables } \{I_1, \ldots, I_N\} \]
- Image features
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

CRF for Semantic Segmentation

Key point 1 + Key point 2

\[ E(x | I) = \sum_i \phi_u(x_i | I) + \sum_i \sum_{j \in \partial i} \phi_p(x_i, x_j | I) \]

- **Unary term**
- **Pairwise term**

- **Unary term**
  - Trained on dataset
  - Cannot consider the relationships between labels of different pixels

Result from unary classifiers
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

CRF for Semantic Segmentation

\[ E(x|I) = \sum_i \phi_u(x_i|I) + \sum_i \sum_{j \in \partial i} \phi_p(x_i, x_j|I) \]

 Unary term  
Pairwise term

Key point 1 + Key point 2

• Pairwise term
  • Impose consistency of the labeling
  • Defined over neighboring pixels (grid CRF)
  • Defined over all pixels (fully connected CRF)

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

\[ k^{(m)}(f_i, f_j) \] is a Gaussian Kernel
\[ f_i, f_j \] is the feature vectors for pixel \( i \) and \( j \), e.g., color intensities,
\[ w^{(m)} \] is the weight of the \( m \)-th kernel
\[ \mu(x_i, x_j) \] is the compatibility function
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Grid CRF for Semantic Segmentation

\[ E(x|I) = \sum_i \phi_u(x_i|I) + \sum_i \sum_{j \in \partial i} \phi_p(x_i, x_j|I) \]

- **Unary term**
- **Pairwise term**

- **Key point 1**
- **Key point 2**

- **Grid CRF:**
  - Neighboring pixels
  - Local connections
  - May not capture the sharp boundaries
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Fully connected CRF for Semantic Segmentation

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

Key point 1 + **Key point 2**

- **Fully connected CRF:**
  - Every node is connected to every other node
  - However, MCMC inference, 36 hours!
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

- They propose an efficient approximate algorithm for inference on fully connected CRF

- Inference in 0.2 seconds
  - ~50,000 nodes (apply to pixel level segmentation)

- Based on a mean field approximation to the CRF distribution
The precision of the sampling methods (e.g., MCMC) is higher than that of the deterministic methods (e.g., Mean field inference)

However, the efficiency of the sampling methods is much less than that of the deterministic methods.
Problem:
Given a set of observations $D$, how to calculate the posterior distribution over a set of unobserved variables (including parameters and latent variables) $Z = \{Z_1, \ldots, Z_n\}$: $P(Z|D)$?

In most cases the posterior $P(Z|D)$ is intractable, we need to use a simple and tractable formula $Q(Z)$ to approximate $P(Z|D)$, i.e., $P(Z|D) \approx Q(Z)$, which leads to the following two problems:

(1) **First**: suppose existing such kind of $Q(Z)$, how to measure the dissimilarity between $Q(Z)$ and $P(Z|D)$?

(2) **Second**: how to obtain a simple $Q(Z)$?
For the first Problem of measure the dissimilarity, we use KL-divergence.

For discrete probability distributions $P$ and $Q$, the KL-divergence from $Q$ to $P$ is defined to be:

$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

The continuous version of the KL divergence is:

$$D_{KL}(P||Q) = \int_{-\infty}^{+\infty} P(x) \log \frac{P(x)}{Q(x)}$$

Properties:
(1) $D_{KL}(P||Q) \neq D_{KL}(Q||P)$
(2) $D_{KL}(P||Q) \geq 0$
    $D_{KL}(P||Q) = 0$, if and only if $P = Q$
(3) Do not satisfy triangle inequality
For the first Problem of measure the dissimilarity, we use KL-divergence.

The KL-divergence of distribution $Q$ and distribution $P$ is:

$$D_{KL}(Q||P) = \sum_Z Q(Z) \log \frac{Q(Z)}{P(Z|D)} = \sum_Z Q(Z) \log \frac{Q(Z)}{P(Z,D)} + \log P(D)$$

$$\log P(D) = D_{KL}(Q||P) - \sum_Z Q(Z) \log \frac{Q(Z)}{P(Z,D)} = D_{KL}(Q||P) + L(Q)$$

To minimize $D_{KL}(Q||P)$, we need to maximize $L(Q)$. Select appropriate $Q$ to make $L(Q)$ easy to compute and maximize.

$$L(Q) = \sum_Z Q(Z) \log P(Z,D) - \sum_Z Q(Z) \log Q(Z)$$

$$= E_Q[\log P(Z,D)] + H(Q)$$
For the **second** Problem of make $Q(Z)$ **simple**, we have **Mean Field Theory**.

Based on Mean Field Theory, the variational distribution $Q(Z)$ is usually assumed to factorize over some partition of the unobserved variables, i.e. for some partition of the unobserved variables $Z$ into $Z_1, ..., Z_M$,

$$Q(Z) = \prod_{i=1}^{M} q(Z_i|D)$$

We denote $Q(Z_i) = q(Z_i|D)$. It can be shown using the calculus of variations that the "best" distribution $Q^*(Z_i)$ for each of the factors $Q(Z_i)$ (in terms of the distribution minimizing the KL divergence, or maximizing the lower bound $L(Q)$) can be expressed as:

$$Q^*(Z_i) \propto \frac{1}{C} \exp < \ln P(Z,D) >_{Q(Z_{-i}) \text{ or } Q(mb(Z_i))}$$

$$\ln Q^*(Z_i) = < \ln P(Z,D) >_{Q(Z_{-i}) \text{ or } Q(mb(Z_i))} + \text{const}$$

$<..>_Q(Z_{-i}) \text{ or } Q(mb(Z_i))$ denotes an expectation with respect to $Q(Z_{-i})$ or $Q(mb(Z_i))$; $Z_{-i}$ means the exclusion set of $Z_i$, in Markov blanket, it is $mb(Z_i)$.
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials  

(Variational Inference part)

For the **second** Problem of make $Q(Z)$ **simple**, we have **Mean Field Theory**.

**Iterative Variational Bayes algorithm:**
1. Initialize $Q^{(1)}(Z_i)$ (randomly)
2. In $k$ step, compute the marginal of $Z_{-i}$:
   \[
   Q^{[k]}(Z_{-i}|D) \propto \exp \int_{z_i^*} Q^{[k-1]}(Z_i|D) \log P(Z, D) \, dZ_i
   \]
3. Compute the marginal of $Z_i$:
   \[
   Q^{[k]}(Z_i|D) \propto \exp \int_{z_{-i}^*} Q^{[k]}(Z_{-i}|D) \log P(Z, D) \, dZ_{-i}
   \]
4. Theoretically, $Q^{[\infty]}(Z_i|D)$ will be converged. Repeat (2) and (3) until $Q(Z_i)$ and $Q(Z_{-i})$ are stable.
5. Finally, $Q(Z) = Q(Z_i|D)Q(Z_{-i}|D)$
For the second Problem of make $Q(Z)$ simple, we have Mean Field Theory.

**Variational Message Passing** (VMP, make the marginal probability computation simpler):
Assuming the model has the form of a Bayesian network, let $X = (Z, D)$

$$
P(Z, D) = P(X) = \prod_i P(X_i | pa_i)$$

where $pa_i$ denotes the set of variables corresponding to the parents of node $i$ and $X_i$ denotes the variable or group of variables associated with node $i$.

(1) Apply variational inference to a Bayesian Network;
(2) VMP is applied to a general class of conjugate-exponential models because it uses a factorized variational approximation;
(3) By introducing additional variational parameters, VMP can be applied to models containing non-conjugate distributions.

For more details, see “Variational Message Passing, Foundations and Trends® in Machine Learning, 2008”
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

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 $L = \{l_1, \ldots, l_k\}$. Consider also a random field $I$ defined over variables $\{I_1, \ldots, I_N\}$. $I_j$ is the color vector of pixel $j$ and $X_j$ is the label assigned to pixel $j$.

A conditional random field $(I, X)$ is characterized by a Gibbs distribution

$$P(X|I) = \frac{1}{Z(I)} \exp \left( - \sum_{c \in \mathcal{C}_G} \phi_c(X_c | I) \right)$$

The Gibbs energy of a labeling $x \in L^N$ is $E(x|I) = \sum_{c \in \mathcal{C}_G} \phi_c(X_c | I)$. The maximum a posteriori (MAP) labeling of the random field is

$$x^* = \arg \max_{x \in L^N} P(x|I)$$

In fully connected CRF,

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

Image: 50*50
Unary term: 2500
Pairwise term: 2499+2498+2497+…+2+1
The appearance kernel is inspired by the observation that nearby pixels with similar color are likely to be in the same class. The degrees of nearness and similarity are controlled by parameters $\theta_\alpha$ and $\theta_\beta$.

The smoothness kernel removes small isolated regions. Like Scale filter (introduced last week).

A simple label compatibility function $\mu$ is given by the Potts model. It introduces a penalty for nearby similar pixels that are assigned different labels:

$$\mu(x_i, x_j) = [x_i \neq x_j]$$
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

**Mean field approximation**

Instead of computing the exact distribution $P(\mathbf{X})$, the mean field approximation computes a distribution $Q(\mathbf{X})$, that minimizes the KL-divergence $D_{KL}(Q||P)$ among all distributions $Q$ that can be expressed as a product of independent marginals, $Q(\mathbf{X}) = \prod_i Q_i(X_i)$

Base on mean field theory, Minimizing the KL-divergence, while constraining $Q(\mathbf{X})$ and $Q_i(X_i)$ to be valid distributions, yields the following iterative update equation:

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

Variational Message Passing

Applying variational message passing, we obtain the following inference algorithm:

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

Algorithm

- Initialize \( Q : Q_i(x_i) = \frac{1}{Z_i} \exp\{-\phi_u(x_i)\} \)
- While not converged
  - Message passing: \( \overline{Q}_i^{(m)} = \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \)
  - Compatibility transform: \( \overline{Q}_i(x_i) = \sum_{l' \in L} \mu(l, l') \sum_{m=1}^{K} w^{(m)} \overline{Q}_i^{(m)}(l) \)
  - Update to calculate \( Q_i(x_i = l) \)
  - Normalization

Each iteration of the Algorithm 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 \( N \).
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Efficient Message Passing

From a signal processing standpoint, the message passing step can be expressed as a convolution with a Gaussian kernel $G^{(m)}$ in feature space:

$$Q_i^{(m)} = \sum_{j \neq i} k^{(m)}(f_i, f_j)Q_j(l') = [G^{(m)} \otimes Q(l)] - Q_i(l)$$

- Gaussian filter $k^{(m)}(f_i, f_j)$
- Apply convolution to $Q_j(l')$
- Smooth, low-pass filter -> can be reconstructed by a set of samples (by sampling theorem)
- We subtract $Q_i(l)$ from the convolved function because the convolution sums over all variables, while message passing does not sum over $Q_i$. 
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Efficient Message Passing

\[
\overline{Q_i^{(m)}} = \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') = [G^{(m)} \otimes Q(l)] - Q_i(l)
\]

- Downsampling \(Q_j(l')\)
- Blur the downsampled signal (apply convolution operator with kernel \(k^{(m)}\))
- Upsampling to reconstruct the filtered signal \(\overline{Q_i^{(m)}}\)
- Reduce the time complexity to \(O(N)\)

They use the permutohedral lattice, a highly efficient convolution data structure that tiles the feature space with simplices arranged along \(d+1\) axes, to perform fast High-Dimensional Filtering.
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Permutohedral Lattice

Naive Gauss Transform

In Naive Gauss Transform, each output (the red dot) gathers data from every input (the blue dots).

Fast Gauss Transform

The fast Gauss transform accelerates this with a uniform grid (the green crosses), where every input contributes to a polynomial approximation about the nearest grid point. The output then gathers from every grid point within some Gaussian-weighted window.

Improved Fast Gauss Transform

The improved fast Gauss transform clusters inputs, computing a polynomial approximation at each cluster. The output gathers from all nearby clusters using Gaussian weights.

Bilateral Grid

The bilateral grid is similar to the fast Gauss transform; it accumulates input values on a grid. However, it trades accuracy for speed by only accumulating constant values rather than polynomial coefficients, and factoring the Gaussian-weighted gather into a separable Gaussian blur followed by multilinear sampling.

Permutohedral Lattice

The permutohedral lattice operates similarly, but uses the lattice instead of a grid. Barycentric weights within each simplex are used to resample into and out of the lattice.
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

**Permutohedral Lattice**

The $d$-dimensional permutohedral lattice is the projection of the scaled regular grid $(d + 1)Z^{d+1}$ along the vector $\vec{1} = [1; ...; 1]$ onto the hyperplane $H_d: \vec{x} \cdot \vec{1} = 0$, which is the subspace of $R^{d+1}$ in which coordinates sum to zero.

Points in the lattice are those points with integer coordinates that sum to zero and have a consistent remainder modulo $d + 1$.

For example, $d=2$:

- $(0,0,0)$, $0+0+0=0$, remainder modulo 3 is $(0,0,0)$;
- $(1,1,-2)$, $1+1-2=0$, remainder modulo 3 is $(1,1,1)$;
- $(2,-1,-1)$, $2-1-1=0$, remainder modulo 3 is $(2,2,2)$;
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

**Permutohedral Lattice**

For more details, see “Fast High-Dimensional Filtering Using the Permutohedral Lattice, Computer Graphics Forum, 2010”

To perform a Gauss transform using the permutohedral lattice:

- First the position vectors $\vec{p}_i \in \mathbb{R}^d$ are embedded in the hyperplane $H_d$ using an orthogonal basis for $H_d$ (as the previous slide shows).
- Then, each input value **splat**s onto the vertices of its enclosing simplex using barycentric weights.
- Next, lattice points **blur** their values with nearby lattice points using a separable filter.
- Finally, the space is **sliced** at each input position using the same barycentric weights to interpolate output values.
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Efficient Message Passing

\[
\widehat{Q}_i^{(m)} = \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') = [G^{(m)} \otimes Q(l)] - Q_i(l)
\]

Algorithm

- Initialize \( Q : Q_i(x_i) = \frac{1}{Z_i} \exp\{-\phi_u(x_i)\} \)
- While not converged
  - Message passing: \( \widehat{Q}_i^{(m)} = \sum_{j \neq i} k^{(m)}(f_i, f_j) Q_j(l') \)
  - Compatibility transform: \( \widehat{Q}_i(x_i) = \sum_{l' \in L} \mu(l, l') \sum_{m=1}^{K} w^{(m)} \widehat{Q}_i^{(m)}(l) \)
  - Update to calculate \( Q_i(x_i = l) \)
  - Normalization

\( O(N) \)

\( O(N) \)

\( O(N) \)

\( O(N) \)
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Results

Image | Our approach | Ground truth
--- | --- | ---
![Image](image.png) | ![Our approach](our-approach.png) | ![Ground truth](ground-truth.png)
Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials

Results

- A fully connected CRF model for pixel level segmentation
- Efficient inference on the fully connected CRF: linear in number of variables
Fully Convolutional Networks for Semantic Segmentation
Fully Convolutional Networks for Semantic Segmentation

Introduction

◆ A Deep Learning method
◆ Mainly focus on Key point 1: form the features for each pixel and find the relationship between the features and the classification results

◆ Main idea:
◆ Change the fully connected layer to convolution layer
◆ Build “fully convolutional” networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning.
Fully Convolutional Networks for Semantic Segmentation

- Before FCN:
  - Take each pixel as an instance, put the fixed size of image region (around the pixel) as the input, get the classification result for this pixel.

- Disadvantage:
  1. Too many pixels in one image, the cost of memory is high;
  2. The computational efficiency is low, cannot share computational information between different pixels when the input image regions overlap.
  3. The fixed input size limit the size of the perception for CNN
Fully Convolutional Networks for Semantic Segmentation

- **FCN:**
- Change the fully connected layer to convolution layer

![Diagram of FCN network]

- Weight between conv_i and Fc1 becomes Filter \((\text{Size:} 7\times7; \text{Padding:} 0; \text{Stride:} 1; \text{Kernels:} 4096)\) between conv_i and conv1

- Weight between Fc1 and Fc2 becomes Filter \((\text{Size:} 1; \text{Padding:} 0; \text{Stride:} 1; \text{Kernels:} 4096)\) between conv1 and conv2

- Weight between Fc2 and Fc3 becomes Filter \((\text{Size:} 1; \text{Padding:} 0; \text{Stride:} 1; \text{Kernels:} 1000)\) between conv2 and conv3
Fully Convolutional Networks for Semantic Segmentation

- **FCN:**
  - Take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning.

**End to end training**
Fully Convolutional Networks for Semantic Segmentation

- **FCN:**
- **Skip layers:** combine the final prediction layer with lower layers with finer strides.
- **Combining fine layers and coarse layers lets the model make local predictions that respect global structure.**

*Diagram showing the structure of a Fully Convolutional Network with skip connections.*

*Red line indicating end-to-end, joint learning of semantics and location.*
Fully Convolutional Networks for Semantic Segmentation

- **FCN:**
- Results for no skips, 1 skip, and 2 skips.
- Combining fine layers and coarse layers lets the model make local predictions that respect global structure.
FCN + CRF:
FCN+CRF

1. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs

2. Conditional Random Fields as Recurrent Neural Networks

Key point 1 + Key point 2

- 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 (lack key point 2).

- 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.
Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs
Semantics Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs

- FCN + CRF
- Key point 1 + Key point 2 (FCN+CRF, separately training)

The coarse score map from Deep Convolutional Neural Network (with fully convolutional layers) is upsampled by bi-linear interpolation. A fully connected CRF is applied to refine the segmentation result.
Conditional Random Fields as Recurrent Neural Networks
Conditional Random Fields as Recurrent Neural Networks

Introduction

- FCN + CRF
- Key point 1 + Key point 2
- Treat CRF as RNN, combine RNN with FCN, and perform end to end training

- The way of paper (Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs) 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.

- 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.
Conditional Random Fields as Recurrent Neural Networks

**Conditional Random Fields**

In fully connected CRF, 

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

- **Unary term**
- **Pairwise term**

- 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.
- Pairwise potentials are modelled as weighted Gaussians:

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

- Label compatibility function
- Gaussian kernel
Conditional Random Fields as Recurrent Neural Networks

**Mean-field Iteration**

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

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

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

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

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

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

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

\[
\text{end while}
\]

- **Initialization:**

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

- This is equivalent to applying a **softmax** function over the unary potentials \(U\) across all the labels at each pixel.
Conditional Random Fields as Recurrent Neural Networks

**Mean-field Iteration**

- The filtering-based approximated mean-field inference can be broken into a series of CNN atomic operations.
- 1. **Message Passing**: convolution layer, use the permutohedral lattice implementation, \( O(N) \)
- 2. **Weighting Filter outputs**: can be viewed as usual convolution with a \( 1 \times 1 \) filter with \( K \) input channels, and one output channel.
- 3. **Compatibility Transform**: 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 \( L \). Learning the weights of this filter is equivalent to learning the label compatibility function \( \mu \).
- 4. **Adding Unary Potentials**: adding bias
- 5. **Normalization**: can be considered as another softmax operation with no parameters.
Conditional Random Fields as Recurrent Neural Networks

**CRF as RNN**

- 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).

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

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

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

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

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.

The authors showed that the mean-field iterative algorithm for dense CRF converges in less than 10 iterations (5 in practice).
Conditional Random Fields as Recurrent Neural Networks

Results

Input Image | FCN-8s | DeepLab | CRF-RNN | Ground Truth
---|---|---|---|---

![Image of input image and its segmentation results](image-url)
Conditional Random Fields as Recurrent Neural Networks

Results
Conclusion

Semantic Segmentation:
For the category label, it can be treated as multi-class classification problem.

- **Key point 1**: form the features for each pixel and find the relationship between the features and the classification results;
- **Key point 2**: add the label constraints for different pixels which incorporate information from various relationships.

Before Deep Learning:

- Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, NIPS 2011,
  Key point 1 + Key point 2 *(CRF)*

After Deep Learning:

- Fully Convolutional Networks for Semantic Segmentation, CVPR 2015
  mainly focus on Key point 1 *(FCN)*
- Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs, ICLR 2015
  Key point 1 + Key point 2 *(FCN+CRF, separately training)*
- Conditional Random Fields as Recurrent Neural Networks, ICCV 2015
  Key point 1 + Key point 2 *(FCN+CRF, end to end training)*

**Key words**: Fully connected CRF, FCN, Variational Inference, Permutohedral Lattice
Thanks!