CRF-RNN for Semantic Image Segmentation

Felix Marti Perez
UGent

Contents

1 Introduction

1.1 Motivation and Applications

Since the 90s where Yann LeCun presented the paper "Gradient-Based Learning Applied to Document Recognition" where Convolutional Neural Networks (CNN's) where first introduced for digit recognition. Mayor breakthroughs have revolutionized the field with respect to image tasks. Specially image classification has seen lots of advanced due to this architectures. For example, in 2012 AlexNet revolutionized this field achieving unprecedented results in ImageNet, an image classification data set with 20,000 categories.

However, due to the inherit architecture of Convolutional Neural Networks (CNN's) this same breakthrough has not been achieved in the task of image segmentation. As this type of networks try to down-sample the image to extract the inherit feature of the image, the result image will show blurry shapes with no clear edges. This is not a desire output for image segmentation, as it will result in a low quality output, with poor object delineation and small deceptive regions.

Sideways to the progress in deep learning, other techniques have been developed for the task of image segmentation. In particular Conditional Random Fields (CRF) have become one of the most successful graphical models in computer vision. As a consequence, new research tried to employ CRFs to enhance the semantic labeling outcomes generated by a CNN.

Initially, a CRF was employed as a post-processing technique to refine the output. However, due to the fact that the CRF was not incorporated during the training phase, its parameters were not jointly optimized. Consequently, the full potential of the CRF was not utilized.

Further improvement where made and a newer model was proposed which made it possible to implement the CRF in the training loop. By formulating CRF as RNN, the network could be trained in an end-to-end manner using back-propagation algorithm. This will be the main focus of this report.

1.2 Conditional Random Fields

As mentioned before, CRF are one of the most succesful graphical models in computer vision, this is because of its nature. Compared to a Naive model, where the probability of a class will be given by P(X,Y), so all its features will be independent. This is not desired, as it does not capture the correlation between features, something necessary in segmentation as in an image, adjacent pixels usually have a high cor-

relation.

To capture this, edges between nodes need to be added. CRF accomplishes this by modelling how X comes together to affect the probability of Y (conditional probability of Y given X). Nodes could be colour histograms, texture features, discriminative patches or many more representations. To model this probability we use the following equation:

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

$$E(X,Y) = \left(\sum_s \phi(x_s, y_s) + \sum_{s,t} \psi(x_s, x_t, y_s, y_t)\right)$$

Where the goal is maximize the probability of assigning the correct label to each pixel, this means minimizing the cost function (E(X,Y) also called energy) of assigning the label. The cost function comprises two components: the unary energy and the pairwise energy. The unary energy, represented by the first summation, quantifies the cost associated with label assignments that are inconsistent with the initial classifier. The pairwise energy, represented by the second summation, quantifies the cost incurred when two similar pixels are assigned different labels. The normalizing factor Z is equal to the sum of all exp(-E(X,Y))

2 Mean Field Algorithm

2.1 CRF as CNN for one iteration

When combining CRF into the training loop of CNN a mayor problem appears, now P(XlY) will not be independent, this means the probability cannot be written as the product of the pixels. As a solution, the authors of the paper propose an approximation where:

$$P(X|Y) \approx Q(X|Y) = \prod Qi(Xi|Y)$$

Q is equal to the product of pixel wise distribution.

To calculate this Q , the authors present the mean-field algorithm. The main idea of this algorithm is to encourage similar pixels to have similar labels, this is achieved by looking at the pairwise features. This relationship is calculated as the pair-ways potential:

$$\psi_p(X_i, X_j) = \mu(X_i, X_j)\sum_{m=1}^{M} w^m k_g^m(f_i, f_j)$$

where μ is a parameter that gives the compatibility between labels, and the summation for M Gaussian filters is given as the weighted linear combination of this Gaussian filter and a Gaussian kernel applied on the feature vector. Gaussian distribution is used as if two points are closed to each other, they will have a higher value. **Note: The features are hand designed and could be for example the RGB value of the pixels or their spatial location*

The previous equation is given by the following algorithm:

$Q_i(l) \leftarrow \frac{1}{Z_i}\exp(U_i(l))$ 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

where, as input it receives the normalized unary potentials, this are the individual pixel values after processing the image trough a fully convolutional network (FCN) and applying softmax to this values. Then given a number of iterations (5 in training, 10 in testing), the following steps are followed: 1) Message Passing: For a number of M Gaussian filters, we calculate the Gaussian filter coefficient of all the the other pixels, for every pixel in the image. As this will mean a computational cost of $\emptyset(N^2)$, the authors decided to use Permutohedral lattice implementation to reduce the cost to $\emptyset(N)$. 2) Weighted Filter Outputs: Calculates the weighted sum of the M filters outputs from the previous steps, for each class label. 3) Compatibility Transform: Given the

output from the previous step, the label compatibility is calculated by the label compatibility function $\mu(l, l')$. 4) Adding Unary Potentials: The output is subtracted from the unary inputs U (output of FCN). 5) Normalization: As in the input, we apply softmax to the result to get the probability scores.

This steps can also be seen as CNN layers, where in the first step we apply a bilateral kernel, then in steps two and three we apply 1x1 convolution, and lastly an addition and softmax.

Given enough iteration, our algorithm will converge to the true probability *P(X, Y)*.

2.2 Advantages and Limitations

The mean-field algorithm, is a very significant algorithm due to its simplicity and its similarity to other Deep Learning models, using back-propagation algorithms makes it easy to implement with very powerful models. However it has been shown that it fails to provide strong theoretical guarantees on the quality of its solutions. Regardless of this, it still allows us to obtain a significant improvement in the accuracy of several computer vision applications compared to sparse CRFs.

2.3 Alternatives

Even though the mean-field is a good approximator and shows good results in the task of image segmentation. Due to not providing theoretical guarantees of the quality of their solution, research has tried to find other alternatives to this algorithm.

Primarily, the research has focused in continuous relaxation-based energy minimization algorithms as they do provide a strong theoretical guarantee. Three main relaxation algorithms can be usually found in literature. Convex quadratic programming (QP) relaxation that can be efficiently minimized using the Frank-Wolfe algorithm. Difference-of-convex (DC) relaxations of the energy minimisation problem can be optimised efficiently using an iterative concave-convex procedure (CCCP). Linear programming (LP) relaxation of the energy minimisation problem can be optimised efficiently via sub-gradient descent.

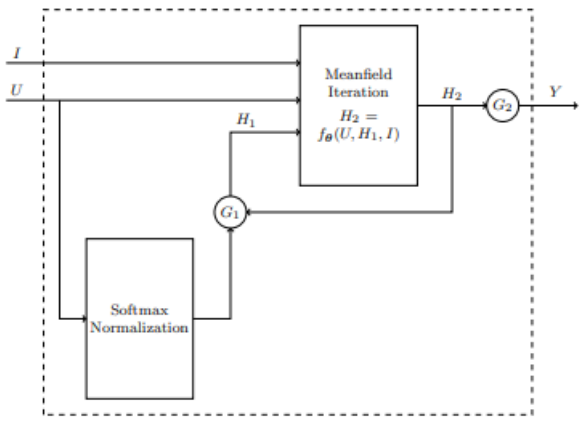
The intention of this section is to give the reader an overview of other approximators that are also used, we will not detail the specifics of each of this implementations as this is not the goal of this report.

3 Deep Dive

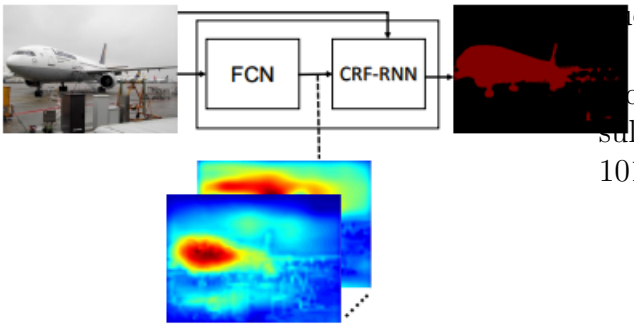
3.1 Implementation Details

Given the explanation in the previous section about the mean-field algorithm. Now we will describe how these fits in our end-to-end trainable model.

As we have seen, one iteration of the mean-field algorithm can be formulated by a stack of CNN layers. If multiple iterations are repeated recursively in this stack of layers, this will be equivalent to a Recurrent Neural Network, as shown in the following figure:



This results in the following final model.



Where the image is first sent through a FCN block which its weights have been initialized to the FCN-8's network. This processed image is then sent through are CRF-RNN block. Where the compatibility transform parameters are initialized using Potts model, and kernel width and weight parameters are obtained from coss-validation. During training the parameters are optimized end-to-end using back-propagation, to be exact Stochastic Gradient Descent (SGD), with a learning rate of $10^{-}13$,momentum set to 0.99 and using log-likelihood as a loss function. Normalization techniques where tried, but did not lead to better results.

3.2 Metrics and Evaluation

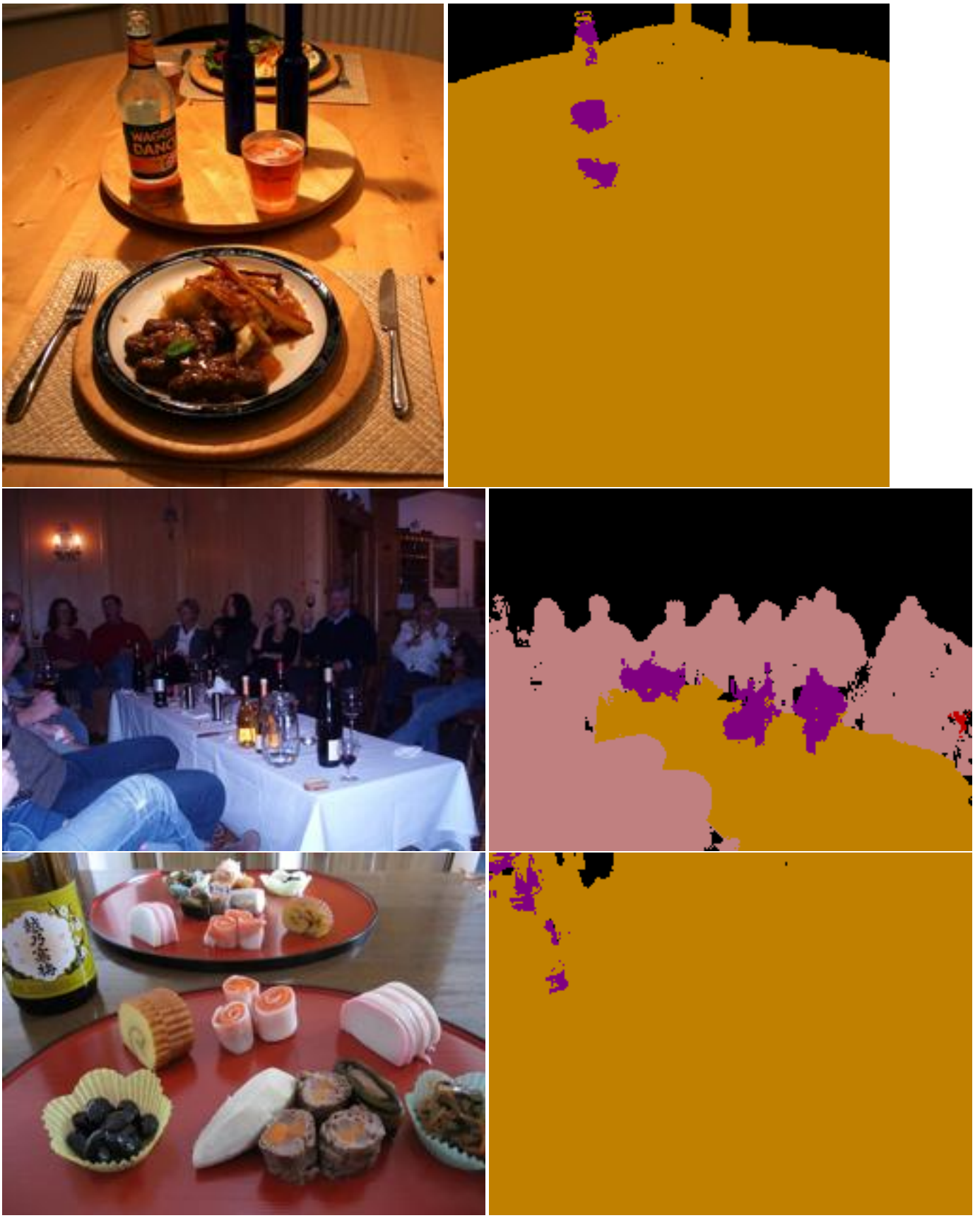
For image segmentation, various evaluation metrics can be used to compare the results of the model to a ground truth. The most popular are pixel accuracy, mean accuracy, mean intersection over union (IU) and frequency weighted intersection over union (IU). In the paper, the authors evaluate their model using mean IU. They compare their results to other models using the Pascal VOC 1012 dataset.

In image segmentation, evaluation metrics are not always a reliable way of evaluating models. If we look closely in the previous figure, CRF-RNN outperforms the ground truth on the first image, however if only comparing with metrics, DeepLab will be a more accurate representation of the ground truth.

4 Demo

Given the following resource provided by the authors, we have implemented a simple Jupyter Notebook which can be accessed to try the implementation by yourself. (*https : //colab.research.google.com/github/sadeepj/crfasrnn_demo/blob/master/crfasrnn_demo.ipynb#scrollTo=kAbdmRmvq0Je*)

Upon evaluating the model in various use cases, its performance can be better assessed. Generally, the model exhibits satisfactory segmentation capabilities, even in challenging environments with abundant information. However, the paper forgets to address instances in which the model struggles or fails to perform its task. For instance, the model is unable to segment objects of diminutive size. This may be attributed to the nature of CNNs; small details may be lost when the output of the CNN is fed into the CRF block. Additionally, the model experiences difficulty with counter tops and tables.



References

[1] Alban Desmaison, Rudy Bunel, Pushmeet Kohli, Philip H. S. Torr, and M. Pawan Kumar. Efficient continuous relaxations for dense CRF. *CoRR*, abs/1608.06192, 2016.

[2] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *CoRR*, abs/1210.5644, 2012.

[3] Bengong Yu and Zhaodi Fan. A comprehensive review of conditional random fields: Variants, hybrids and applications. *Artif. Intell. Rev.*, 53(6):4289–4333, aug 2020.

[4] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240, 2015.