Combining Conditional Random Fields with Deep Neural Networks for Semantic Segmentation

Rocherno F.M. de Jongh
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
dejon076@morris.umn.edu

ABSTRACT

Semantic segmentation is the task of assigning each individual pixel in an image a predefined class, such as "road", "sidewalk", "person". Semantic segmentation has applications in domains where detailed understanding of an image is required. Traditionally, semantic segmentation has been approached by using structured probabilistic (graphical) models known as Conditional Random Fields (CRFs). Lately, deep neural networks (DNNs) have had great success on semantic segmentation. CRFs and DNNs both have their shortcomings and benefits. There are some areas where CRFs are better than DNNs, and some areas where DNNs are better than CRFs. Combining these two methods allowed for improved performance in semantic segmentation. In this paper, we will be presenting both techniques individually, and then describing how they work together for better results in semantic segmentation.

Keywords

Semantic segmentation, Undirected Graphical Models, Conditional Random Fields, Convolutional Neural Networks

1. INTRODUCTION

Semantic segmentation aims to assign a predefined object class to each pixel within an image. Semantic segmentation has numerous applications where detailed understanding of an image is required, including road segmentation for autonomous driving, cancer cell segmentation for medical diagnosis, and mobile real-time video segmentation. In Figure 1 we can see the difference between semantic segmentation and image classification. The distinguishing factor for semantic segmentation is that every pixel in the image is assigned a predefined class, and also objects that are the same get assigned in the same class as we can see in Figure 1 (b). In contrast, image classification is just describing what objects there are on an image, which we can see in Figure 1 (a).

There are different methods for approaching semantic segmentation, in this paper we will talk about the two most used methods: *Conditional random fields* (CRFs), an early attempt at semantic segmentation based around probabilistic models [1], and *Deep Neural Networks* (DNNs), a more recent attempt at using neural networks [1]. In this paper, we will start by giving a short overview of neural networks in section 2.1, and structured probabilistic models in section 2.2, which will serve as background for the main topic that we will cover in this paper. We will give a review of CRFs in section 3, and DNNs in section 4, in the context of semantic segmentation. Then we will show how we can incorporate CRFs into DNNs to form a joint network, where the advantages of these two models will be combined. Lastly, we will review some DNNs that incorporates CRFs and discuss the results of these frameworks in section 5.

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/.
UMM CSci Senior Seminar Conference, April 2018 Morris, MN.

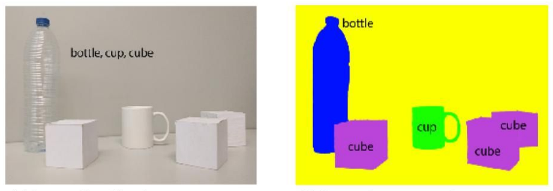

Figure 1: (a) Image Classification and (b) Semantic Segmentation. Modified from [5].

2. BACKGROUND

For the first part of this section, section 2.1, we will give a description of neural networks. For the second part of this section, section 2.2, we will give a description of undirected probabilistic graphical models which we will be using further on for this paper.

2.1 Introduction to Neural Networks

Neural networks were inspired by the biological brain, and and are loosely modeled on the biological brain. [6] As we can see in Figure 2, neural networks are organized into layers of nodes which are often called *neurons*. One layer in a neural network is designated as the *input layer*, as the name suggests the input to the system begin their interaction with the neural network in this layer. The nodes in this layer only have-one-way connections leading out of the layer. In contrast the *output layer* is where information is extracted from the neural network. In Figure 2 the input layer is the leftmost layer, and the output layers is the rightmost. For our purposes, the input of a neural network is an image, where each node in the input layer represents a pixel in the image. The output that we get from the output layer, is either one of the predefined classes, or a vector containing the probabilities for each of the predefined classes that the

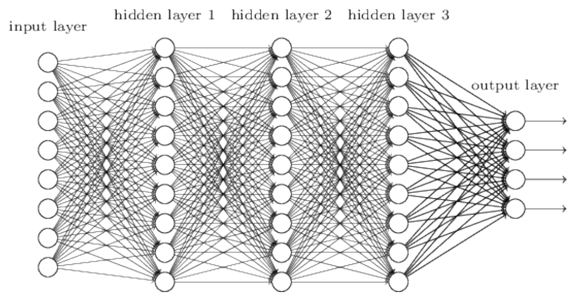

Figure 2: A neural network. This is a neural network containing 3 hidden layers.[2]

image can take. The middle layers of the neural networks are called *hidden layers*. The neural network in Figure 2, contains three hidden layers.

The process from input layer to output layer is as following: Each node contains an *activation function*. A node will assign a number, called a *weight*, to each of its incoming connections. The node receives a different data item, which is a number, over each of its connections, and multiplies each one of these number by the associated weight. The node then adds the resulting products together which yields a single number. The results that each node returns from their activation function are passed to their neighboring nodes. After passing through the the layers of nodes and connections, the results are sent to the output layer, which returns some type of output. [7].

An individual node in a layer might be connected to several other nodes from the layer from which it receives data, and several other nodes from the layer which it sends data to. The most common type of layer in regular neural networks is the fully-connected layer. This is a layer for which each node inside the layer is fully pairwise connected to the nodes in adjacent layers, but the none of the nodes inside this layer share a connection with each other. The neural network in Figure 2 is an example of a neural network where each of its layers are full-connected layer.

Now we will give a brief description on how neural networks are trained. Initially, the weights in a neural network are set to a random values. Then training data is fed through the input layer. The output of the neural network (semantic segmentation of the image) is compared to the known classification. *Training* a neural network consists of modifying the weights in such a way as to improve the accuracy of the output on the training set. (see [6] for details.) Eventually no more improvements occur and the training is complete.

For semantic segmentation, we are interested in a type of neural network called convolutional neural network (CNN), which will we talk about in section 4.1.

2.2 Probabilistic Graphical Models

In this section, we will talk about models where we can represent relationships between pixels in a graphical way. Probabilistic graphical models provide an easy and useful way to visualize the structure of a probabilistic model and can be used in designing and motivation of new models. We will introduce some basic background in probability theory and graph theory, in sections 2.2.1 and 2.2.2, respectively.

[2]http://neuralnetworksanddeeplearning.com/chap6.html

2.2.1 *Introduction to Probability Theory*

In this section we will give introduce some terms in probability theory that we are going to use in this paper.

A *random variable* is variable whose value depends on the outcome of a random phenomenon. Since we are talking about pixels in an image, we will be talking about *discrete random variables*. A discrete random variable has a finite number of states it can take. If we assume that a pixel is a random variable that we want to assign a label to, then its states are all the labels that the pixel can take. A *probability distribution* is a function that gives a description of how likely a random variable or a set of random variables is to take each of its possible states.

We are often concerned about the relationship between two or more random variables, in this case the relationship between two or more pixels in an image. For this, we need to use the *joint distribution* of random variables which allows us to compute the simultaneous behavior of these random variables. Thus, with the joint distribution we can understand the relationship between the random variables. Sometimes we want to know how certain random variables behave, given that we know something about some other variables. For example, say we have a pixel, p_i. And we know the label of another pixel, p_j. Knowing the label for p_j influences how we assess the probabilities of various labels for pixel p_i. Probabilities that are calculated based upon such information are called conditional probabilities. We call the resulting distribution the conditional distribution of p_i given p_j.

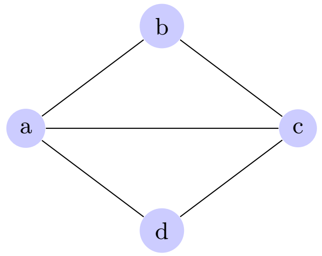

Figure 3: A graph with two maximal cliques.

2.2.2 *Introduction to Graph Theory*

In this section, we will give some basic definitions of graph theory that will use in this paper. For this paper, when we talk about graphs, we mean graphs in a graph theory sense. Let $G = (V, E)$ be a *graph*, where G consists of a set of nodes, V, and a set of edges E. Two distinct vertices are called *adjacent* if there is an edge connecting these two vertices. A graph is called a *complete graph* if there is an edge connecting every pair of vertices in the graph. An *induced subgraph* is a subgraph obtained by deleting a set of nodes and its their corresponding edges. A *clique* is an induced subgraph that is a complete graph. A *maximal clique* is clique where a vertex cannot be added to make it a bigger clique. In Figure 3, we can see that the sets $\{a, b, c\}$ and $\{a, c, d\}$ are the maximal cliques in the graph. We can see that sets such as $\{a, b\}$ and $\{c, d\}$ are cliques in the graph, but are not maximal cliques since they are part of larger cliques in the graph, in fact, they are part of the two maximal cliques in the graphs, $\{a, b, c\}$ and $\{a, c, d\}$. For the rest of the paper, I will be referring to only maximal cliques.

2.2.3 Undirected Graphical Models

Probabilistic graphical models is a way of encoding a probability distribution between the random variables using a graph. In general, each random variable is represented as a node in the graph, and the edges connecting the nodes represents some sort of probability relationship between the nodes. There are different kinds of structured probabilistic models, but for this paper, we will focus on the undirected graphical models. *Undirected graphical models* are graphical models that have a set of nodes and a set of undirected edges which connects a pair of nodes.

A naive way that we can perform semantic segmentation is to independently classify each pixel in an image by only considering the local features of that pixel. With local features we mean things like the colors and hue of that pixel. Because neighboring pixels tend to have the same label, this naive way of classifying each pixels independently will produce unsatisfactory results in semantic segmentation. So, if there was a way to check the joint properties of neighboring pixels, we could get better results when segmenting an image.

This is where using a probabilistic models helps in image segmentation. On of the ideas behind probabilistic models is instead of looking independently at the pixels in an image, we look at the joint distribution of the pixels. In general, pixels that have similar features, such as color, have strong correlation with each other, meaning that they are more likely to have the same label.

In section 3 we will talk about a specific kind of undirected graphical model that is used in semantic segmentation.

3. CONDITIONAL RANDOM FIELDS

In this section we will give a review of conditional random fields that are used for semantic segmentation.

3.1 CRF definition

In the context of semantic segmentation, a CRF models a probability distribution of pixel labels, conditioned on a global observation. In general, the global observation is usually the image.

Let $G = (V, E)$ be a graph. Let X_i be the random variable corresponding to the pixel i, where X_i represents the label assigned to the pixel i, which can be one of the labels from the set of predefined labels, $\mathcal{L} = \{l_1, ..., l_L\}$, where L is the number of predefined labels.. Let $\mathbf{X}$ be set formed by the random variables $X_1, ..., X_P$, where P is the total number of pixels in the image. Then, the set V of the G is defined by $V = \{X_1, ..., X_p\}$.

A formal definition of CRF in context of semantic segmentation is, where it is characterized by a log-linear model is given by:

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

where $\mathbf{I}$ are the input images of size P, C is the set of maximal cliques in graph G. $Z(\mathbf{I})$ is a normalizing function that ensure the probability distribution equals 1, such that it is a probability distribution [10]. $E(\mathbf{x}|\mathbf{I}) = \sum_{c \in C} \phi_c(\mathbf{x}_c|\mathbf{I})$. Each maximal clique produces a potential function [3], ϕ_c. $E(\mathbf{x}|\mathbf{I})$ is called the energy function of $\mathbf{x}$, where $\mathbf{x}$ is all of the possible ways the labels can be assigned to the pixels in the image.

[3] *Potential functions* are arbitrary nonnegative, real-valued functions [8].

Popular CRF graphs are the grid graphs, which only has node interactions between a limited number of neighboring nodes, and fully connected graph, which is where all nodes are connected, thus forming a complete graph. The fully connected CRF, or dense CRF are being favored, because the more connections between the nodes leads to better segmentations. Also, efficient algorithm exists for the fully connected pairwise CRF, which we will explain below.[1]

Because fully connected pairwise CRF are preferred in general, we will talk about this model. For the CRF to be a fully connected pairwise CRF, the graph G needs to be a complete graph. Then, instead of only looking at the maximal clique, which in the case of a complete graph would be the whole set of nodes, we will look at the pairwise cliques, thus we will look at all of the pairs of nodes. We will define the energy function as following:

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

where $1 \leq i, j \leq P$. $\psi_u(x_i|\mathbf{I})$, called the unary potential, is calculated independently for each pixel. The unary potential returns the cost of pixel i taking the label x_i, given the image features. The pairwise potentials, $\psi_p(x_i, x_j)$, returns the cost pixel i taking the label x_i and pixel j taking the label x_j simultaneously. So, the edges connecting the nodes means that the cost$\psi_p(x_i, x_j)$, is defined. The pairwise potentials in this model are defined as:

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)$$

where each $k^{(m)}$ is a Gaussian kernel and $w^{(m)}$ are linear combination weights. $\mathbf{f}_i$ and $\mathbf{f}_j$ are feature vectors associated with the pixels i and j respectively. Feature vectors are features derived from the image features, such as color. In general, the two feature vectors that get used are RGB values and spatial locations, such that the kernel will be defined as a two-kernel potential as following:

$$k(\mathbf{f}_i, \mathbf{f}_j) = w^{(1)} exp\bigg(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2} \bigg) + w^{(2)} exp\bigg(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2} \bigg)$$

Where p_i and p_j are the positions and I_i and I_j the color vectors, of pixels i and j respectively. The first part of $k(\mathbf{f}_i, \mathbf{f}_j)$ is called the *appearance kernel*, and is inspired that that nearby pixels with similar color tends to be in the same label class. The second part of $k(\mathbf{f}_i, \mathbf{f}_j)$ is called the *smoothness kernel*, and it removes small isolated regions [9]. All of the parameters in the model, $w^{(m)}, \theta, \alpha, \beta$ are learned during the inference of the CRF.

$\mu(x_i, x_j)$ is called the compatibility function. Basically, it imposes a penalty when labels that are nearby and similar to each other get assigned different labels.

In summary, with CRFs we want to model the probability that an image has a certain labeling, given the input image. To paraphrase, let's say we assign each pixel a certain label, we want to know then what is the probability that this labeling is true, given the input image.

3.2 CRF Inference

To train the CRF, data sets of already labeled images are used. Different CRFs method uses different ways of training.

A popular inference method for fully connected CRFs *mean field approximation*, which is a heuristic method. Mean field approximation minimizes $E(\mathbf{x}|\mathbf{I})$, which results for the most probably labeling $\mathbf{x}$ for the given image. Since minimizing $E(\mathbf{x}|\mathbf{I})$ is hard, mean field approximation approximates the actual CRF distribution $P(\mathbf{X} = \mathbf{x}|\mathbf{I})$ by a simpler distribution, say $Q(\mathbf{X} = \mathbf{x}|\mathbf{I})$. Then, the parameters are learned by piecewise training, which means the the unary potentials and pairwise potentials are learned separately. The full description of the inference algorithm is out of scope of this paper, but a detailed description can be found in [9].

4. DEEP NEURAL NETWORKS

When a neural network consist of more than 1 hidden layer, we call it a *deep neural network*. As we can see, the neural network in Figure 2 is also an example of a DNN. In the following section, we will talk about one particular kind of DNN called convolutional neural network.

4.1 Convolutional Neural Networks

Convolutional Neural Networks are similar to ordinary neural Networks, they are made up of neurons and can also be trained. As with neural networks, the input for the CNN is image. What is different this time is that the image will be taken as a 3-dimensional array, $h \times w \times d$, where h is height, w is width, and d is the dimension of the array, which is the 3 RGB colors.

Most CNN architectures consists of three main types of layers: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. We will give a brief description of each layer. Then, in section 4.5, we will talk how we can "extend" CNNs to use them for semantic segmentation.

4.2 Convolutional Layer

The first layer in every CNN is a *convolutional layer*, and is what mainly differentiate CNNs from other types of neural networks. The input that the convolutional layer takes is an array of pixel values, which are the pixels of an image. Then, a smaller (or as big as the input) array of numbers, called a *filter* (or *kernel*) is used that represents a certain feature. To start, the filter is aligned in the top left corner of the input. The area that the filter covers is called the *receptive field*. The filter then multiplies the values in the filter with the pixel values in the image, using element wise multiplication. Then the filter slides one spot, and does the same thing. Every unique position that the filter slides to, produces a number, which are stored in an array called a *feature map*. The feature maps are then used as input for the following layers in the CNN. [4, 11]

4.3 Pooling Layers

A *pooling layer*, or *downsampling layer*, simplify the information that they get from the convolutional layer. A pooling layer takes each output of a feature map, and condenses the feature map. So, a pooling layer will divide the input it gets into regions, and summarizes that region into a smaller regions. Thus, pooling layers are used to reduce the spatial dimension, e.g. array of values, of the input it received. [4, 11]

4.4 Fully Connected Layer

Generally, the last layer of successful CNNs is called the *fully connected layer* This layer takes the input it has gotten from the preceding layer, which can be the the pooling layers, and connects this input to the output neurons. Thus this layer takes an input and returns an output, which is an C-dimensional of probabilities, where C is the number of classes the CNN has to choose from. [4, 11]

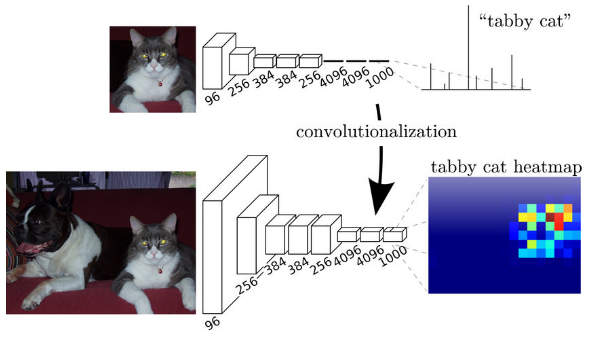

Figure 4: Here we can see how the fully connected layer is transformed into a fully convolutional network, by taking the fully connected layer to be a convolutional layer, where the size of the filter and the input feature map are the same. [12]

4.5 Extending CNNs for Semantic Segmentation

In the previous sections we gave a brief description on how CNNs work. But, we described a CNN that works for image classification, and not semantic segmentation. In this section we will talk how we can extend CNNs, so they can be used for semantic segmentation. There are more than one way that CNNs can be extended to be used for semantic segmentation, but we will describe the most common one. Long et al. [12] showed that the fully connected layer of a CNN can be viewed as a filter that covers that whole region of the input feature map. The authors called them fully convolutional networks (FCNs), that can take an input of any size, and returns a classification map.

In Figure 4, on the top part we can see a CNN that is being used for image classification, we can see that all the way to the top right, there is a vector of probabilities showing to which class the image most likely belongs to. But, we can see on the bottom part of the figure, the last layer, the fully connected layer is being interpreted as a convolutional layer and the filter is taken to be the same as the input feature map, which is 4096 in this case. Then, this will return what the authors call a "heat map", which is just a feature map, but this time at pixel level, as needed for semantic segmentation.

The network is trained really similar to neural network, training data sets with labeled images are used. The network can automatically learn features from the training data set using a method called stochastic gradient descent, or a variant of this method, which minimizes a training objective function. An algorithm called backpropagation, is used to train the network. *Backpropagation* is a method that can compute the gradients of the parameters in the network in an efficient manner, with respect to the objective function. [1] A detailed description of these methods can be found in [6].

5. COMBINING CRFS AND DEEP NEURAL NETWORK

Although FCNs achieved great results for semantic segmentation, this did not always produced smooth outputs. A challenge for CNNs is, since CNNs were initially designed for image classification, is that at the pooling layers, a lot of spatial information get lost, which resulted in feature maps with low spatial resolution for the FCNs. Because of this, for semantic segmentation, the output of these CNNs sometimes resulted in objects with boundaries that are not sharp and blobby shapes. [2]

In this section we will describe two models that combines both CRF and CNN, which is mainly to take advantage of the smoothing constraints in CRFs. In section 5.1, we will describe a method that applies CRF as a post-processing step after the input has been segmented by a CNN. In section 5.2, we will describe a method that incorporates CRFs in CNN to create a unified framework.

5.1 CRFs For Post-Processing

In this section we will describe the first method that employs both CNN and CRF.

In post-processing, the output of the CNN are post-processed through the CRF to smooth out the results of the CNN. As we said in section 3, CRFs incorporates assumptions in their model, such as, similar pixels and pixels that are close to each other should have the same label. Because of this, CRFs can help produce sharper boundaries and finer grained results from the output received from the CNN [13].

We said that the the CRF takes the output of a CNN. With this we mean that the output of the CNN is taken to be as the unary potential for the CRF. The unary potential will then be defined as, $\psi_u(x_i|\mathbf{I}) = -logP(x_i|\mathbf{I})$, where $P(x_i|\mathbf{I})$ is the probability distribution of labels for the pixel i, that is computed by the CNN. The pairwise term, $\psi_p(x_i, x_j)$, is the same as described in section 3. In the first row in Figure 5 we can see an example of how this method works. Convolutional feature extractor is just the way the convolutional layer works as describes in section 4.2.The linear classifier is the process after the convolutional layer, thus when we get the output from the convolutional layer and pass it through the pooling layers and then the fully connected layer. Then, we see we get a result which is used as the input for the unary potential for the CRF. In the figure they showed that a Dense CRF, which is a fully connected CRF, is used to further smooth out the image. In theory, any kind of CRF can be used, but dense CRF is the most common one, for reasons stated in 3.

In this method, CRF is disconnected from the CNN, and also disconnected from the CNN training. So, the CNN is trained by itself and the inference of the CRF also take place by itself.

5.2 End-to-End CRFs

In this section we will describe the second method that employs both CNN and CRF. In this method, CNNs and CRFs are combined in an end-to-end joint framework. For this section, we will mainly talk about the method used in [13] which interprets CRFs as RNNs in the framework.

The main point in [13] was to show that the mean field inference can be reformulated as a Recurrent Neural Network (RNN), which are a family of neural networks. Each output of an iteration of a RNN is used as the input for the next iteration, and all of the iterations use the same parameters. [1] Because this is similar to how the mean field inference works, the authors proposed to treat the mean field inference as a RNN, and called this structure CRF-RNN. All of the parameters of the CRF-RNN are the same as the parameters for the mean field inference. The structure of this network will start just as a normal CNN for semantic segmentation, as we can see in Figure 5. The input image will go through the convolutional feature extractor and linear classifier as describe in section 5.1, and then the output from the linear classifier will go to the CRF-RNN structure, which performs CRF-based probabilistic modeling. [13]

The way that the CRF is incorporated in this framework, the two methods, CRF and CNN, will not be trained separately, but rather are trained end-to-end using the back-propagation algorithm and Stochastic Gradient Descent. In theory, this method should work outperform the method using post-processing, since everything is trained at the same time. [13]

6. RESULTS

In Table 1 we can see the results of recent algorithms for semantic segmentation. The metric that they are using is *intersection over union* (IoU). This metric assesses the segmentation accuracy by checking each class of labels, and checking the resulting segmentation against the ground truth, which is the correct label assignment. To get the accuracy, the metric divides the correctly assigned pixels over the the total pixels. (So, basically, $\frac{Correctly\ assigned\ pixels}{All\ pixels}$.)

The three methods not using a CRF in the table all employ CNN in a way in there algorithm, but did not use a CRF in any way. Based on the IoU percentage, we can see that these algorithms performed worse than the algorithms that did use CRF in a way in their algorithm.

Finally, we discuss the results for the methods that do employ CRF in a way in their algorithm. The methods that are under "Methods using CRF for post-processing", all employ some kind of CNN, but they also use fully connected CRFs as post-processing, as described in section 5.1. The methods that are under "Methods with end-to-end CRFs", all combine CNNs and CRFs as an unified framework, as described in 5.2. From Table 1 we can see that these two methods that employ CRFs in way with CNNs, outperforms the other methods. The algorithm that employs end-to-end CRFs has the higher IoU percentage. Also, looking at the last two rows of Table 1, we see that methods employing end-to-end training for CNN and pairwise CRF have a higher mean IoU percentage, than method where the CNNs and pairwise CRFs are trained disjointly. Showing evidence that end-to-end training give better results, But, what is clear from the table is that the methods that combines CNNs and CRFs definitely outperforms the methods that do not combine these.

7. CONCLUSION

We gave an overview of CRFs and DNNs in the context of semantic segmentation. Then we discussed and gave examples of some models that combined the powers of CRFs and DNNs to tackle semantic segmentation. We have shown two ways of combining these models, methods using CRFs as postprocessing, and methods using CRFs end-to-end. We have shown and discussed that combining these two models

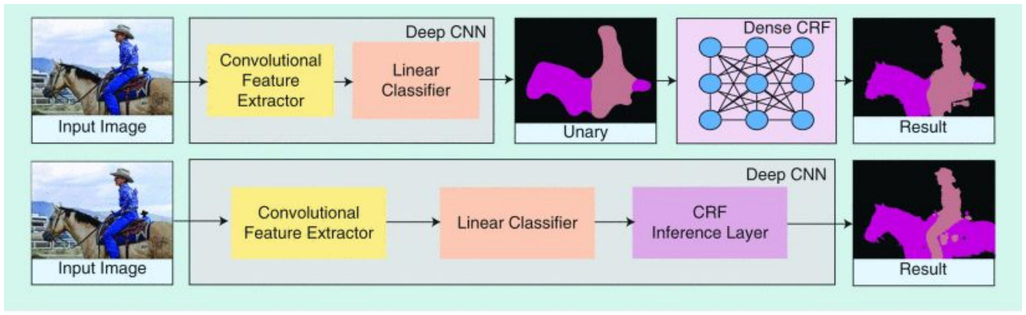

Figure 5: Here we can the structure of the two methods that combines CNN with CRF. The first row uses post-processing CRF and the second row combines CNNs and CRFs as an end-to-end joint framework. Modified from [1]

Method	IoU[%]
Methods not using a CRF	
SDS	51.6
FCN	67.2
Zoom-out	69.6
Methods using CRF for post-processing	
DeepLab	71.6
BoxSup	75.2
Dilated Conv	75.3
DeepLab Attention	76.3
LRR	79.7
Methods with end-to-end CRFs	
CRF as RNNs	74.7
Context	77.8
Deep Gaussian CRF	80.2
Method	Mean IoU[%]
Pairwise CRF trained disjointly	69.5
Pairwise CRF trained end-to-end	72.9

Table 1: Results of algorithms on the Pascal Visual Object Classes (VOC) 2012 test set. Modified from [1].

results in better image segmentation. We also showed that methods employing end-to-end CRFs results in better image segmentation.

Acknowledgements

I want to give a special thanks to my advisor, Professor Peter Dolan, and the senior seminar coordinator, Professor Elena Machkasova, for their guidance and feedback throughout this semester.

8. REFERENCES

[1] A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother, F. Kahl, and P. H. S. Torr. Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction. *IEEE Signal Processing Magazine*, 35(1):37–52, Jan 2018.

[2] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, April 2018.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[4] A. Deshpande. A Beginner's Guide To Understanding Convolutional Neural Networks, 2016. https://adeshpande3.github.io/A-Beginner

[5] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[7] L. Hardesty. Explained: Neural networks. `http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414`, Apr 2017.

[8] M. I. Jordan. An introduction to probabilistic graphical models, 2003.

[9] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[10] V. Kuleshov and S. Ermon. CS 228: Probabilistic Graphical Models, Lecture Notes, 2017. https://cs.stanford.edu/ ermon/cs228/index.html.

[11] M. A. Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

[12] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, April 2017.

[13] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537, Dec 2015.