Aleksander Colovic

Utilizing Conditional Random Fields in Deep Learning Applications to Semantic Segmentation

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme

Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Thomas Pock

Institute for Computer Graphics and Vision

Advisor

Dipl.-Ing. Patrick Knöbelreiter

Institute for Computer Graphics and Vision

Graz, Austria, April 2017

Abstract

In this work we tackle the problem of semantic image segmentation with a combination of convolutional neural networks (CNNs) and conditional random fields (CRFs). The CRF takes contrast sensitive weights in a local neighborhood as input (pairwise interactions) to encourage consistency (smoothness) within the prediction and align our segmentation boundaries with visual edges. We model unary terms with a CNN which outperforms non-data driven models. We approximate the CRF inference with a fixed number of iterations of a linear-programming relaxation based approach. We experiment with training the combined model end-to-end using a discriminative formulation (structured support vector machine) and applying stochastic subgradient descend to train it.

Our proposed model achieves an intersection over union score of 62.4 in the test set of the cityscapes pixel-level semantic labeling task which is comparable to state-of-the-art models.

Kurzfassung

In dieser Arbeit wird die Problemstellung der semantischen Segmentierung mittels einer Kombination aus *convolutional neural networks* (CNNs; zu Deutsch etwa *faltende* neuronale Netzwerke) und *conditional random fields* (CRFs; zu Deutsch bedingte Zufallsfelder) behandelt. Das CRF verwendet kontrastsensitive Gewichte, die aus einer lokalen Nachbarschaft im Bild ermittelt werden, als Eingangsgröße. Dadurch wird die Kontinuität der Vorhersage (Klassenzugehörigkeit) innerhalb eines Objektes erhöht und die Objektgrenzen werden an den Bildkanten ausgerichtet. Weiteres wird von einem CNN ein unärer Eingang für jeden Bildpunkt bereitgestellt. Die Inferenz im CRF wird mit einer fixierten Anzahl an Iterationen eines auf relaxierter linearer Programmierung basierten Ansatzes angenähert. Zu Letzt wird das kombinierte Modell von Anfang bis Ende unter Verwendung eines *structured support vector machine* Ansatzes (zu Deutsch etwa strukturierte Stützstellen Maschine) gemeinsam trainiert. Dazu wird ein stochastischer Subgradienten Abstieg angewandt.

Unser vorgeschlagenes Modell erzielt eine Wertung von 62.4 IoU (*intersection over union*) auf dem Test Set des Cityscapes Datensatzes in der entsprechenden Kategorie (*pixel-level semantic labeling task*). Das Modell ist damit vergleichbar mit dem aktuellen Stand der Forschung.

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Date

Signature

Contents

List of Figures

List of Tables

1 Introduction

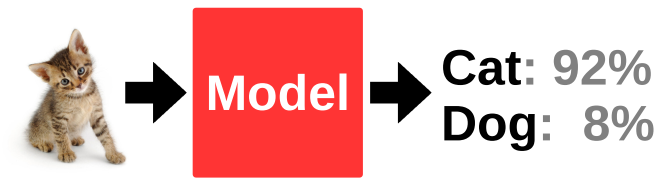
Contents

In this chapter, we motivate the challenging task of semantic segmentation first. We follow this up with sketching our contribution to the field and outlining the content of this thesis.

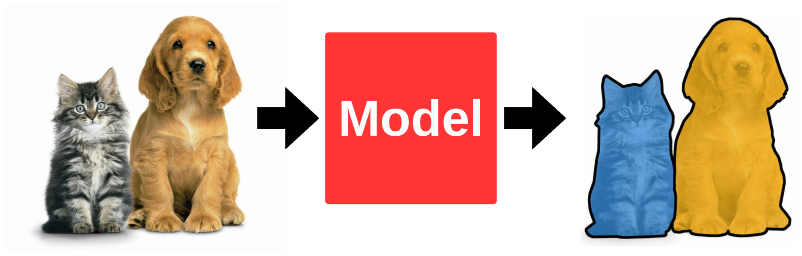
1.1 From Sparse to Dense Predictions

In classical image classification, the aim is to predict the content of images. For instance, given a picture of a cat we want some mathematical model to tell us, that the picture contains a cat. Obviously there exist more useful applications of such a model, *e.g.*, character recognition [3, 6] or classifying street signs [12, 54]. Nowadays, deep learning approaches have become exceptionally good at classifying images [34]. The most severe flaw of such models is that they assume a single label to be sufficient enough to describe the content of an image. In other words, the input is restricted to only contain a single object which is typically not the case.

In a more general setting, we allow multiple objects within a scene and try to detect and classify all of them. A rather straight forward approach would be to split the image into smaller regions and predict the content in each subregion. While this does not solve the problem of multiple objects occurring in a single subregion, it lowers the chances of that scenario happening.



(a) Image classification. One label per image.



(b) Semantic segmentation. One label per pixel (color encoded).

Figure 1.1: Abstract comparison of image classification and semantic segmentation to illustrate the different expected outcomes.

Developing this idea further suggests the question regarding the size of such a subregion. Intuitively, smaller regions are preferable since they have a lower probability to contain more than one object. The smallest regions in discrete images would be just individual pixel. With this, the problem can be reformulated to predict the content of each individual pixel in each image. The result is then called a *dense prediction*.

1.2 Semantic Segmentation

The task of semantic segmentation is to calculate dense predictions, *i.e.*, compute a class label for each pixel in the image. For street scenes such as the ones used in this work, classes can be *e.g.*, *car*, *street*, *pedestrian*. Solving this problem with hand-crafted methods is difficult.

However, recent improvements in computer hardware and the work of Krizhevsky *et al.* [34] made it possible to train deep convolutional neural networks (CNNs) with millions of parameters on graphic processing units (GPUs) for image classification. Since then the knowledge of image classification slowly translated to semantic segmentation making it a feasible task.

1.3 The Flaws of Dense Predictions

State of the art CNN architectures for image classification try to map a given input image to a single output label. To do so, a model needs to abstract the high dimensional input and compress the information into a low dimensional space. Deep layered neural networks like

CNNs use special operations (see section 3.5) which lower the spatial resolution to do so. This is beneficial for image classification, because it allows to learn a robust and abstract representation of the given classes. However, it is detrimental for semantic segmentation due to the loss of spatial information.

1.4 Contribution

Simply upscaling a coarse solution obtained from a CNN is often visually not pleasing due to blurred edges. A common way to compensate for this is to use a conditional random field (CRF) as a post processing method.

CRFs allow to incorporate knowledge about real-world processes (prior) into a model in terms of a regularizer. For semantic segmentation, such knowledge could be that label discontinuities typically only occur across object boundaries. This prior helps to align the predicted segmentation- with actual object boundaries in the image to obtain sharp results.

In this work, we investigate a hybrid CNN+CRF architecture to combine good semantic reasoning (CNN) with sharp object boundaries (CRF). Instead of using the CRF as a post-processing step for the output of the CNN, we are tackling the challenging problem of training this CNN+CRF combination jointly. If the CNN is used for post processing, the CNN is trained to directly predict class labels. The basic idea is, that directly predicting labels might not be the optimal input to a CRF and that there exists an input representation which gives better results *after* the CRF inference. By training the model jointly, we optimize for a sharp prediction which allows the CNN to shift some of its responsibility regarding segmentation boundaries to the CRF and focus on other parts of the problem.

We show how to compute a gradient for training our hybrid model jointly using a structured output support vector machine (SSVM) approach in a non-linear setting.

1.5 Outline

To present the reader an overview of the topics covered in this thesis, we want to discuss the content of the following chapters briefly in this section.

Chapter 2 discusses the current state of the art in semantic segmentation. We cover various approaches which mostly use a CNN or a combination of CNN and CRF as those appear to work best. Additionally, we review works which focus on end-to-end training of such a hybrid model and how those methods relate to our approach.

Chapter 3 gives the reader a wide overview over the field of neural networks. By observing biological networks, we motivate the introduction of a mathematical model which approximates its biological inspiration. We outline how nowadays commonly used artificial concepts are resembled in natural networks throughout the chapter.

Besides the biological roots of certain ideas, we mainly focus on the artificial neural networks and their characteristics. Starting with the definition of single neurons, we continue to gather individual units in layers which are then stacked to form deep networks. Next we introduce the idea of convolution to propagate information through the network as this has many applications in state-of-the-art computer vision models. Additionally, we discuss the concept of abstraction in this context. At last we sketch a special type of model architecture which got a lot of attention in the community in recent years and is used as a subpart of this works proposed model as well.

Chapter 4 focuses in detail on the task of deep learning. We discuss common concepts and derive learning rules to automatically obtain the model parameter values through observing labeled data. We discuss different metrics that can be optimized in this context. At last we cover commonly used strategies to speed up and improve learning in general deep networks.

Chapter 5 covers the fundamentals behind the second building block used in this thesis, namely conditional random fields. Again, we motivate the interpretation of CRFs as an energy minimization problem that consists of two terms. We then elaborate on those terms in the following. At the end of this chapter we discuss gradient estimation of the CRF which is useful in the later chapters where we train a CNN and a CRF together. To this end, we review the structured support vector machine. This allows us to formulate a structured learning problem and derive a gradient approximation. We conclude this chapter with a graphical interpretation of said approximation.

Chapter 6 introduces the proposed combination of the previously defined CNN and CRF. We start with an explicit definition of the used CNN which we call *unary network* in this context. Then we develop model priors that guide the model particularly along object boundaries. Finally, we plug all pieces together to obtain the *CNN+CRF* model which is then used in the following sections.

Chapter 7 gives a detailed overview of the performed joint training. We provide reasons for initializing our model with existing work rather than starting from scratch first. Secondly, we discuss the pre-training of the *unary network* and thirdly we append a CRF for post processing purposes. In a final stage, we then train the whole model in an end-to-end manner.

Chapter 8 contains the main experiments conducted in this work. We utilize the *Cityscapes* dataset which is used for training and evaluation first. Then we review the intersection-over-union score, as this is the most prominent evaluation method in the field. Finally, we report both, qualitative and quantitative results of our work. We compare different modes of our model with each other, as well as discuss its relation to similar state-of-the-art approaches.

Chapter 9 concludes this thesis by discussing the outcome as well as giving insights into future work.

2 Related Work

Contents

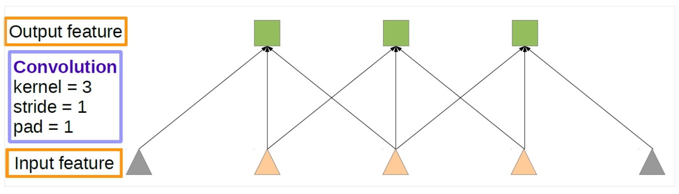
This section briefly outlines the current state-of-the-art in semantic segmentation. See chapters 3 to 5 for a detailed analysis of the fundamental building blocks used in this section.

Our model consists of two such building blocks. First, we use a CNN to extract suitable features for semantic segmentation from data. The output of the CNN is then used as *unary* costs in a CRF formulation. In a final step, the CRF optimizes the joint energy of the data-cost and a consistency-enforcing smoothness prior.

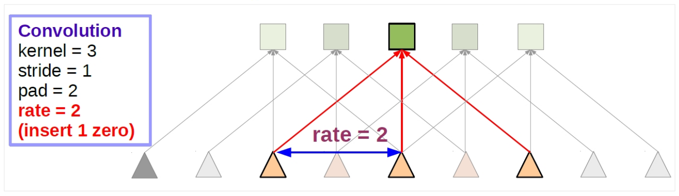
The present work uses a hybrid architecture and training method similar to [30], where the authors applied it to the stereo reconstruction problem. The setting in this thesis is different in the CRF model interactions (classification versus depth), CNN architecture (deep model versus shallow in [30]) and the final application loss (intersection over union versus truncated l_1).

2.1 Convolutional Neural Networks

CNNs are among the top performing models for computer vision tasks like image classification [34, 56], object detection [18, 40, 49] and semantic segmentation [5, 39, 41, 70] for the last couple of years. Backed by steadily increasing computational power learning models with millions of parameters became manageable. In image classification, striding or pooling is used to create abstract, low resolution representations of images to obtain a single label per input image in the end. In contrast to classification, the difficulty in



(a) Sparse feature extraction



(b) Dense feature extraction

Figure 2.1: Illustration of a 1-dimensional feature extraction. (a) Convolution on sparse data (*e.g.*, after pooling). (b) Atrous convolution on dense input. Figure taken from [5].

semantic segmentation is to assign a label to every pixel in an image to obtain a so called *dense prediction*. Loosing spatial information throughout the model makes correct predictions at object boundaries difficult. Nonetheless, abstract representations are important and many approaches to semantic segmentation use a network trained for classification such as [56] as a starting point [5, 17, 39, 41, 45, 63, 68, 70].

2.1.1 Atrous Convolutions

Deep CNNs mostly use pooling or striding at consecutive layers of the network to obtain abstract and spatially invariant representations. For dense predictions, it is a common approach to use upsampling or deconvolution to reconstruct high resolution predictions [41]. In contrast to that, *Atrous convolutions* (or later called dilated convolutions [68]) were originally developed for efficient computation of the undecimated wavelet transform [24]. By filling filters with holes (*trous* in French) this enables large receptive fields without increasing the number of parameters. Figure 2.1 illustrates the convolution. This approach was used by [19, 46, 53] to obtain dense features via a CNN. Using multi-scale context, the works [5, 68] applied spatial pyramid pooling to further increase the performance of such models.

2.1.2 Residual Networks

In the work of He *et al.* [22], the effects of stacking more layers in convolutional networks are investigated. The authors reason that in general, any model is contained in a model with more layers if the added layers just perform an identity mapping. However, current solvers are not capable to find equally good or better solutions in very deep architectures. The introduced residual network accounts for said difficulties by adding skip connections which explicitly perform an identity mapping. Figure 2.2 illustrates the described architecture.

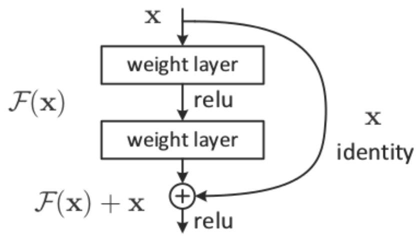

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.2: The skip connections perform an identity mapping which is then added to convolutional layers output. $\mathcal{F}(x)$ is the learned representation which models the residual representation. Figure taken from [22].

Thus, the convolution layers learn to model the residual feature representation. The authors of [22] reason that in an extreme view, it is easier for a solver to drive all weights in a layer towards zero than to find the identity.

While [38] argue that residual models can be interpreted as recurrent ones, the authors of [64] and [66] investigate the relation of residual networks to model ensembles. Latter of which report state-of-the-art results for various image classification and semantic segmentation challenges. They do so by formulating a shallow architecture which outperforms its deeper siblings significantly. Further research trends involve more branching and merging [1, 67, 69] or forming fractal structures that skip over multiple residual sections [57].

2.2 Conditional Random Fields

CRFs are probabilistic models that incorporate relations between nodes in a graph [37]. In the most general case the graph is fully connected. The nodes represent discrete random variables, defined over the set of possible labels, that are conditioned on the input image. The edges in the graph (pairwise terms) model label consistency over node pairs. In the context of semantic segmentation, nodes typically correspond to image pixels or superpixels.

As we discuss in more detail in chapter 5, the CRF defines an energy term for each position which is then seen as a probability distribution. Finding the most probable label corresponds to minimizing the defined energy. Therefore, the CRF computation can be interpreted as an energy minimization problem.

To solve this problem for a fully connected graph, [32] reformulates the computation of the model (mean field approximation [31]) as high dimensional filtering. This formulation is advantageous because it allows the filtering scheme [2] to be applied to the problem. This approach is often used as a post processing method, *e.g.*, for semantic segmentation [5].

Reducing the edges in the graph to a four-connected neighborhood, [55] provides a heavily parallelized GPU implementation. The methods inference is fast enough to be performed during the training stage instead of as post processing afterwards. We use [55] to incorporate the CRF into the learning procedure and train our model in an end-to-end fashion. By doing so, the preceding CNN can adopt to the capabilities of the CRF and hence provide richer feature representations. In this setting the CRF can be seen as a specialized network layer.

2.3 End-to-End Learning

The hybrid training of CNNs and CRFs for structured predictions has been explored by others as well in recent history. Lin *et al.* [39] approximate the gradient of a low resolution CRF using the piecewise training method [60]. In piecewise training, the marginal probability modeled in a CRF is approximated with the product of individual potentials which allows for a simpler gradient computation.

Zheng *et al.* [70] unrolled the mean field iterations of [32] in a recurrent neural network. They did this by expressing a single iteration in the mean field inference of their CRF as a sequence of standard convolutions. This approach requires the pairwise terms to be modeled with high dimensional Gaussian filters but in turn allows for training their model with the exact gradient through back-propagation. This contrasts with our approach where use a structured support vector machine formulation in a nonlinear setting to approximate the gradient of the CRF [50, 62].

3 Neural Networks

Contents

This chapter aims to give a qualitative discussion of general neural networks as they are part of the mathematical model used in this thesis. Therefore, we will develop the model bottom-up, starting with the model of a single neuron. Then we extend that concept to groups of neurons (*layers*) and later flesh that idea out to a full network.

3.1 Biological Roots

As with many ideas, the concept of neural networks was inspired by nature as the human brain consist of roughly 10^{11} neurons which form complex networks [23].

In biology, there are several different kinds of neurons which receive input from many other neurons. Simply put, the higher the accumulated input to a neuron, the more likely the neuron is to *spike*, *i.e.*, produce an output which then stimulates other neurons to be more likely to spike as well. Depending on the thickness and position of the connections between neurons, the importance of an input to the neuron can vary.

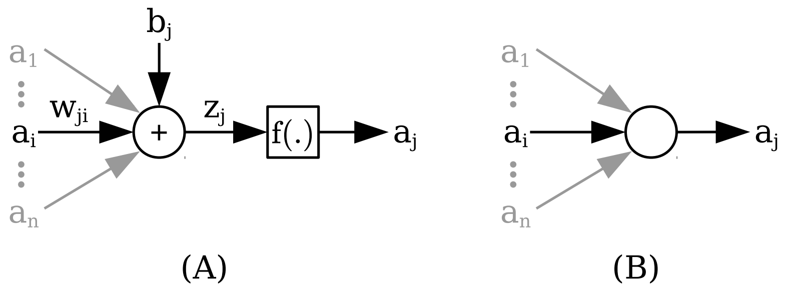


Figure 3.1: (A) contains a detailed model of a single neuron. (B) shows the simplified model which is used in following figures.

3.2 A Simple Model

We first describe the mathematical model of neurons and then form larger networks that can solve complex problems. In the context of computer vision, such models are usually much simpler than the biological counterpart. We define that each neuron j has a single output a_j. In contrast to biology, where the output is encoded in the frequency in which the neuron produces voltage spikes, in a time discrete setting we use a continuous valued variable to approximate that frequency. This output forms the input to other neurons. The dependence of neuron j on the output of neuron i is described with a weight w_{ji} to model position and thickness of the connection. The weighted sum of inputs is called activation z_j. The models output a_j is then obtained through a non-linear transformation $f(.)$ of the activation, which approximates the non-linear behavior of biological neurons. The model is now given as

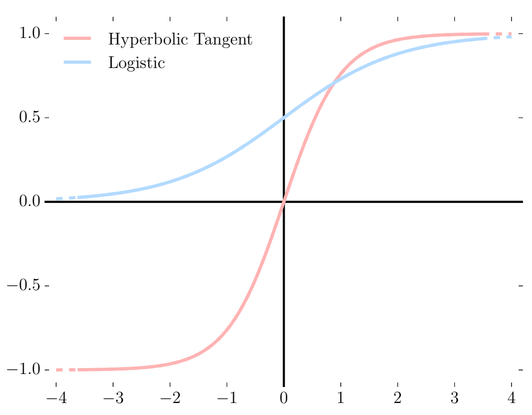
$$z_j = b_j + \sum_{i \in \mathcal{I}} w_{ji} a_i, \tag{3.1}$$

$$a_j = f\left(z_j\right) \tag{3.2}$$

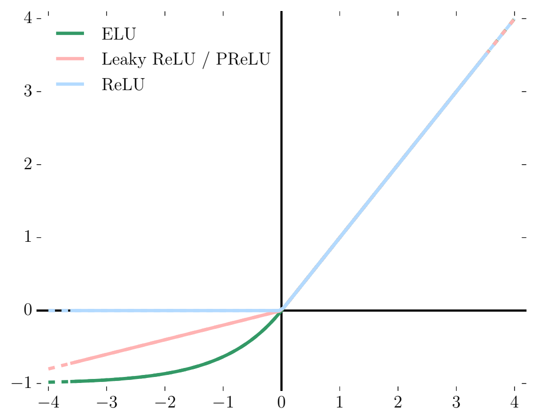
where b_j is called bias and $\mathcal{I}$ denotes the set of input indices connected to neuron j. Figure 3.1 contains a visualization of the described model.

3.2.1 The Source of Non-Linearity

An important part in the previously described model (equation (3.2)) is the activation function f. It fulfills an important role in the model as it is the only source of non-linearity in the neural network. The universal approximation theorem states, that a feed-forward network can approximate any nonlinear function arbitrarily well with a finite number of parameters under mild assumptions. Formally, let f be a non-constant, bounded and monotonically increasing function. We further denote the output of the model as F. Then any continuous function $\hat{F}$ can be approximated universally up to small constant $\epsilon > 0$



(a) Hyperbolic tangent- and logistic function



(b) Rectified linear unit (ReLU) and its extensions leaky/parametrized ReLU and exponential linear unit (ELU)

Figure 3.2: Comparison of common activation functions used in neural networks.

such that

$$\left|F(x) - \hat{F}(x)\right| < \epsilon, \tag{3.3}$$

for all inputs x in a compact subset of $\mathbb{R}^n$. One of the first proves of this theorem was given for sigmoid functions by Cybenko [9]. Later, Hornik [25] showed that not the specific choice of function, but the model structure gives neural networks the universal approximation property.

The theorem only guarantees the existence of a solution. It does not say anything about how to find a suitable solution. Naturally not all activation functions are equally attractive choices, therefor we want to discuss the most relevant characteristics in the following.

- **Non-linear:** If there were only linear functions in the network, *i.e.*, $f(x) = \xi x$, the whole model would merely compute a linear combination of the input. Therefore, in order to model highly complex representations, a non-linear model is necessary.
- **Differentiable:** As we will see in section 4.3, the gradient of the activation function needs to be computed frequently during training. Because of this, we are interested in functions which are (sub)differentiable and efficient in terms of computational power requirements.
- **Bounded Output:** With a bounded output range, a gradient-based training approaches tends to be more stable. However, recent work neglects this property to speed up the overall computation [42, 44].

We cover the most relevant functions for our purpose in the following.

Sigmoid Commonly used sigmoid functions include the logistic- and hyperbolic tangent functions. Both are bounded, monotonically increasing and continuous. The logistic function

$$f(z) = \frac{1}{1+e^{-z}}, \tag{3.4}$$

has the neat property that its derivative can easily be calculated as

$$\begin{aligned} f'(z) &= \frac{\partial}{\partial z}\frac{1}{1+e^{-z}} \\ &= \frac{e^{-z}}{(1+e^{-z})^2} \\ &= \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right) \\ &= f(z)\left(1-f(z)\right). \end{aligned} \tag{3.5}$$

The hyperbolic tangent function

$$f(z) = \tanh(z) = \frac{2}{1+e^{-2z}} - 1, \tag{3.6}$$

has a similarly simple derivative

$$f'(z) = \tanh'(z) = 1 - f(z)^2, \tag{3.7}$$

as it is a scaled and shifted version of the logistic function. Figure 3.2a contains exemplary plots of both sigmoid functions.

Rectified Linear Unit (ReLU) The ReLU [44] neither has a bounded output range ($\lim_{x\to\infty} f(x) = \infty$) nor is its derivative continuous. However still, it is used in many state-of-the-art models in computer vision mainly because its computation

$$f(z) = \begin{cases} z \text{ if } z > 0, \\ 0 \text{ else} \end{cases}, \tag{3.8}$$

$$f'(z) = \begin{cases} 1 \text{ if } z > 0, \\ 0 \text{ else} \end{cases}, \tag{3.9}$$

is very simple. The well-known property that the gradient vanishes for negative inputs is usually referred to as the *dying ReLU* problem. Various extensions such as leaky ReLU [42], parametrized ReLU [21] and exponential linear unit [7] were proposed to tackle this problem by introducing a small non-zero function for negative inputs. Figure 3.2b contains exemplary plots of the ReLU function and some of its extensions.

While it is still not fully understood why ReLUs work, they are heavily used today as their theoretical flaws seem not to be as severe in practical usage. A way to interpret

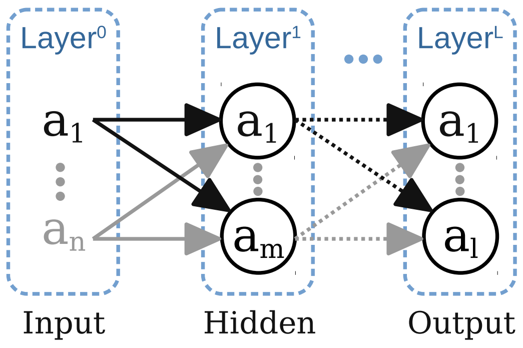


Figure 3.3: Simplified model of a feed forward network. Each layer can contain a different number of neurons. The first layer typically contains the input while the last is considered the networks output. Intermediate layers are denoted as hidden layers and indicated as dashed arrows.

ReLUs is to consider the active units (non-zero output) in the model to form a *linear* subsystem. ReLUs allow to model many such systems within a network which altogether is then highly non-linear.

3.3 Forming Layers

While a single neuron itself can only do rather simple computation, the combination of multiple neurons can, in theory, approximate arbitrary non-linear functions [9, 26]. Here, combination of neurons means that they are connected through weights and form a graph. In the context of *feed forward* neural networks, which are used in this thesis, we assume that the graph is directed and acyclic. This means, that the information flow in the graph is directed from input to output and no loops are allowed. Thus, we group neurons in layers and restrict each group to see the preceding layers output only.

Figure 3.3 illustrates the abstract concept of a feed forward network which is structured into several layers. The first and last layers are considered the input and output of the network respectively. In semantic segmentation, the input is typically the given image and the output a probability distribution which yields the models prediction.

So far, we considered each neuron to be connected to *all* neurons of the previous layer. While this is reasonable in theory, it becomes difficult to handle in practice due to the high number of variables (weights and biases for each neuron). Thus, special layers with less variables are used in this thesis which will be investigated in more detail in the following sections.

3.4 From Fully Connected- to Convolutional Layers

In a fully connected layer, each neuron can *see* all neurons in the preceding layer. This allows them to become specialist not just for patterns in the data, but also for their respective position.

With increasing complexity of a given problem, the number of neurons in these layers must increase as well in order to capture the complexity of the processed data. Having many neurons and their respective parameters (weights to previous neurons and a bias term) increases the dimensionality of the parameter space and therefore also the learning problem becomes more complex. Thus, to maintain a reasonable representation of the parameter space, which is needed to compute and minimize the cost, exponentially more data is needed. In practice the amount of data required cannot be provided and the model will over-fit on the data, *i.e.*, specialize too much on the training data and learn noise patterns that are not present in unseen data.

Convolutional Layer To overcome the problem of high dimensionality in the parameter space, convolutional layers enforce two restrictions on the previously described fully connected layers. Both restrictions were also found to be present in the visual processing pathway of cats by Hubel and Wiesel [27] and reduce the number of parameters in each layer drastically. With less parameters that need to be learned, stacking multiple layers to form deep networks becomes practically feasible.

- The first restriction regards the receptive field of neurons. In the visual processing pathway, the perceptive field (*i.e.*, connections to the preceding layer) of neurons is limited to its local environment. For artificial neural networks, this translates to neurons only seeing a subset of the previous layer. Figure 3.4 illustrates this.

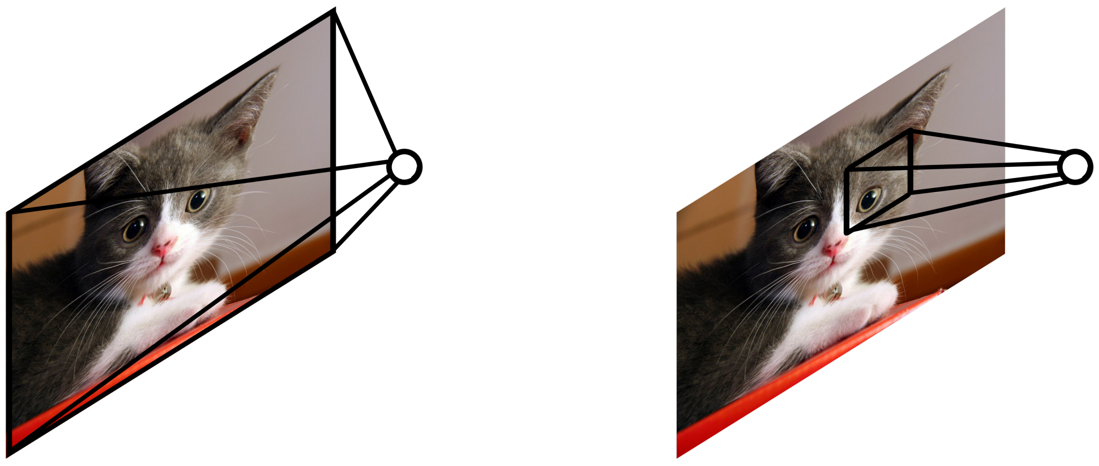
- The second observation is, that the brain contains a lot of redundancy in the sense that *similar* connection patterns occur *scattered* over the visual field. If we treat *similar* as *equal*, this means that multiple neurons (at different locations) can share their weights. Further we approximate *scattered* with a *regular grid* which allows to formally express the computation of the output as a convolution of the input with a group of filters.

The one-dimensional discrete convolution [10] of an image I (just a line for 1D) with a filter f, where f has a finite support set $\{-K, -K+1, \ldots, K-1, K\}$, can be written as

$$(I * f)_x = \sum_{k=-K}^{K} I_{x-k} f_k. \tag{3.10}$$

This can easily be extended to two dimensional images where the kernel has the same support set in both dimensions

$$(I * f)_{x,y} = \sum_{m=-K}^{K} \sum_{n=-K}^{K} I_{x-m,y-n} f_{m,n}. \tag{3.11}$$



(a) Neuron fully connected to the preceding layer.

(b) Neuron connected to a subregion in the preceding layer.

Figure 3.4: Difference in receptive field between a neuron in a fully connected- and a convolutional layer.

Hence the parameter in a convolutional layer are the filter values. By stacking multiple such layers on top of each other, highly nonlinear transformations can be obtained while still having a reasonably small parameter space.

These simplifications only yield good results if the initial assumptions are valid, *i.e.*, (i) having only local features in each layer is sufficient to represent the data and (ii) these local features can occur at any position in the image. If the assumption about locality or position of occurrence is wrong, then a convolutional network will most likely not be suitable. Empirically, they seem to be suitable in the case of visual data which is indicated by the huge success of convolutional neural networks in recent history (see chapter 2).

3.5 Enforcing Abstraction

As motivated during the introduction of this thesis, in a typical image classification model, the high dimensional input image needs to be mapped to a sparse output, *e.g.*, a vector containing a probability for each label. In this process the spatial resolution needs to be reduced. The most prominent way to do so are pooling operations which aggregate information over a wider spatial range.

Pooling We consider the neurons in a layer to have a spatial dimension of $h \times w$ and c being the number of channels. To pool information means to take a subregion in the spatial domain and merge the contained information into a single output. The idea is, that in order to produce good results, the model has to learn abstract features that contain information about their surroundings. If the model succeeds, *merging* a subregion into

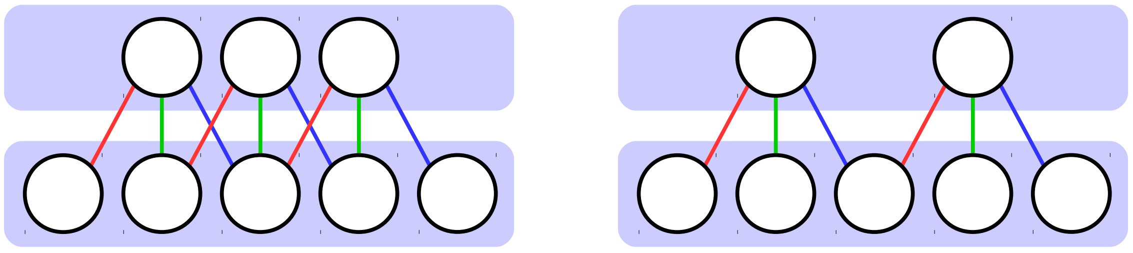

(a) Filtering at every position ($n = 1$). **(b)** Filtering at every 2^{nd} position ($n = 2$).

Figure 3.5: Filtering a one-dimensional input (lower row) with a 3×1 filter (same color indicates the same weight). Different strides n are used to illustrate how increasing strides reduce spatial resolution.

a single neuron preserves the abstract information about the region. The most intuitive merging operation is the arithmetic average of the subregion. However, in practice it turns out that the maximum operation is sufficient and due to the cheaper computational cost, this is usually the used pooling operation.

Application to Semantic Segmentation In the case of semantic segmentation where dense predictions are desired, it is questionable whether using pooling is reasonable. The main argument to do so is, that it allows to use a model trained for image classification as a starting point by keeping the models structure. Considering how simple it is to get a single label for an image as opposed to a dense prediction, it becomes clear that there are simply sparser than densely labeled images available. Therefore, models trained for sparse predictions are used as a starting point and extended to keep spatial information [41] or to directly learn a refinement [17]. For this thesis, we follow [34] and use 2×2 subregions for each of multiple pooling stages and follow [41] to cope with the reduced resolution. Typically, two to three convolutional layers are followed by a pooling layer, slowly decreasing the spatial resolution throughout the model.

Striding Instead of having explicit pooling layers in the model that are responsible for compressing the information, striding layers merge convolution and data compression. They can be seen as a convolutional layer followed by a subsampling operation. After the subsampling, only every n-th element is preserved. In this context n is called *stride*. This means that the *max* or *mean* value computed in pooling layers is simplified to sampling. The recent work of [58] suggests, that neural networks can *learn* a form of pooling if forced to by the model architecture. For obvious reasons, it is useless to compute output which is then dropped at the sampling stage. Therefore, in a practical implementation, those elements would not be computed in the first place. This means, opposed to usual convolutional layers where the input is filtered at every position, striding filters leave out $n - 1$ positions which is illustrated in figure 3.5. For the same reasons as with the

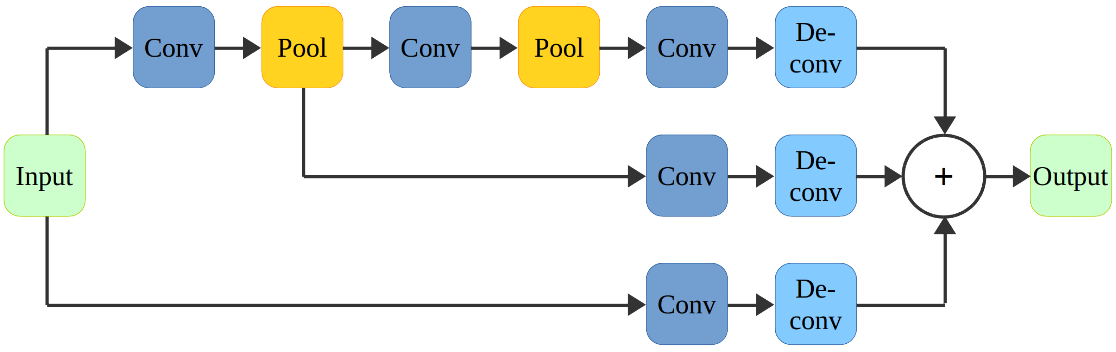


Figure 3.6: Illustration of the core ideas in the fully convolutional network of [41]. *Conv* are convolution-, *Pool* are pooling- and *Deconv* are deconvolution layers. Together they form multiple branches at different spatial resolutions which are combined in the end to obtain the output.

pooling layer, reducing spatial resolution encourages the model to find abstract features that condense information contained in their surroundings. As a result, the model learns its own form of pooling, possibly in a better way than the predefined maximum operation.

3.6 A Fully Convolutional Network

In the recent work of [41], the model solely consists of convolution-, pooling- and deconvolution layers. The model has not just one forward path but instead has so called skip-connections or -branches. Those branches are used to achieve fine grained predictions.

Figure 3.6 shows the architecture of the unary network. The fundamental concept is, that if the data passes sequentially through pooling layers (indicated as *Pool* boxes in the first row), the spatial resolution is reduced step by step. As discussed earlier, this helps to enforce an abstract representation at the cost of a low spatial resolution. Additionally, the model branches out at earlier stages to skip some pooling operations and therefore preserve a higher spatial resolution. Thus, the semantic confidence will drop due to a less abstract view. Several such branches are then up-sampled or deconvolved at the end (*Deconv* blocks) to match the input resolution. The last operation is to *fuse* the information of all branches through linear combination and obtain the desired output.

3.6.1 Obtaining Predictions

The output of figure 3.6 is a score volume with dimensions $h \times w \times c$ for height, width and channels/labels respectively. To obtain an output which can be interpreted as a probability, a soft-max operation is applied to that volume. Thus, for each pixel, the exponentials of scores of all labels are normalized to sum up to one. This is then treated as the desired probability distribution over the set of possible labels.

We consider $z_{x,y}$ to be the score vector in the volume at the spatial position (x, y) and

$z_{x,y,k}$ to be the score at that position for a particular label k. With this, the probability for a pixel in the image $I_{x,y}$ to be assigned a label k is

$$p(I_{x,y} = k) = \frac{\exp\left(z_{x,y,k}\right)}{\sum_{k'} \exp\left(z_{x,y,k'}\right)}, \tag{3.12}$$

which defines the soft-max operation.

The predicted label $P_{x,y}$ is then obtained by selecting the most likely label

$$P_{x,y} = \arg\max_{k} p(I_{x,y} = k). \tag{3.13}$$

In later sections, we use these predictions as a baseline and extend the model through a CRF. To avoid confusion, we denote the predictions obtained solely from the convolutional network as *CNN only*.

4 Learning Network Parameters

Contents

Traditional hand crafted approaches to computer vision tasks involve manually designing algorithms and rules to solve problems. For instance, to detect a bicycle, one could count the number of wheels and pedals and conclude from that through various manually defined rules that there is a bicycle.

The approach of machine learning is fundamentally different. If we see a bicycle, we immediately *know* what we look at. Neuroscience has not fully unraveled the reasons for this by now. However, a key observation is that we can learn from repetition, *i.e.*, learning how a bicycle looks by seeing hundreds of them throughout our lives. In principle, we can learn to distinguish a given pattern from others through repetition, which slowly causes the neural connections in our brains to change. Note that this is not the only learning process in human brains, but the only one relevant for this thesis.

We want to do something very similar, namely learn to perform semantic segmentation from given data. Instead of manually designing every individual aspect of the model, we use a neural network architecture which models a nonlinear mapping from input to output. The actual modeled function is dependent on millions of parameters (filter weights in the network layers) which are changed during a learning phase to yield reasonable results.

By using pairs of images and known labels (training data), we can define a learning procedure which allows our abstract model to continuously adapt and change given new data input.

4.1 Learning as an Optimization Problem

While defining how a neural network is build we have assumed that the computation performed by the network makes sense. However, at the beginning all the weights and biases in the network are initialized randomly. As a result, the output is just noise as well. Assume a given cost function $J(\theta)$, where θ is a vector holding the model parameters. We further assume the cost function to be high for *bad* models and low for *good* ones. To *train* a network means to figure out how the parameters need to be changed to reduce that cost.

Ideally we would like to compute the global minimum of J in a closed form. Since neural networks are highly non-linear, the shape of the cost function $J(\theta)$ is complex (in fact non-convex) as well. For this reason, no closed form solution exists and an iterative optimization approach needs to be adopted. The gradient of the cost function with respect to a parameter indicates in which direction that parameter needs to be changed to raise the cost. In other words, moving the parameters slightly into the opposite direction will reduce the produced cost. This procedure performs a descent on the gradient and is therefore called gradient descent. The update performed for each network parameter θ at every iteration can be formulated as

$$\theta^{t+1} = \theta^t - \lambda \cdot \frac{\partial J}{\partial \theta}, \tag{4.1}$$

where λ denotes a usually small learning rate.

It must be pointed out that this approach can never guarantee to find the global minimum, but only a local one. Later, in sections 4.4 to 4.7, we will outline possible ways to escape local minima as well as discuss how to improve learning in general.

In the following we want to define a reasonable cost function for classification problems such as semantic segmentation. Then we revisit the error back-propagation scheme of [51] since it is fundamental for this thesis. The algorithm describes how to obtain gradients of the cost function with respect to every model parameter in a computationally efficient manner.

4.2 A Suitable Cost Function

So far our model was designed to predict a label for each pixel. More explicitly we obtain a discrete probability distribution over the finite set of labels for each pixel. Now we use this prediction and compare it with ground truth knowledge (typically humans labeled the data beforehand) to compute a cost value.

Categorical Cross Entropy If a soft-max layer is used to obtain a probability measure at the end of the network, then the categorical cross entropy (CCE) is most commonly used as a cost function. It averages the cross entropy $H(p,q)$ over all pixel and samples in a batch. The cross entropy is a measurement of distance between two probability distributions p and q. In a discrete setting, $H(p,q)$ is defined as

$$H(p,q) = -\sum_k p_k \log q_k, \tag{4.2}$$

where k ranges over the discrete set of possible labels in both distributions. We can consider the ground truth label t for a single pixel as the probability distribution p which is one at the corresponding t^{th} position and zero elsewhere. The distribution q is then the prediction of the network. With this, the cost is

$$J = \frac{1}{N}\sum_n H(p_n, q_n), \tag{4.3}$$

where p_n and q_n denote probability distributions at a given pixel location n.

If p is fixed as it is the case here, then minimizing $H(p,q)$ is equivalent to minimizing the Kullback-Leibler divergence $D_{\text{KL}}(p\|q)$ [36] because they are equal up to an additive constant [20]. Therefore, the D_{KL} is sometimes used in related literature instead of the cross entropy.

Mean Squared Error The mean squared error (MSE) is commonly used for regression where the prediction is continuous valued. While it is not exclusively restricted to regression, using it for classification over the CCE is not beneficial though.

Considering that the last layer in the network is computing the soft-max of equation (3.12), the error is then computed as the MSE between prediction and target

$$J_{\text{MSE}} = \frac{1}{2K}\sum_k (p_k - q_k)^2, \tag{4.4}$$

using the previously introduced naming conventions.

For a more in depth discussion on why this is undesirable in our case see section 4.3.1.

4.3 Error Back-Propagation

What has been left out so far is the question of how to calculate the gradient of the cost function with respect to all the parameters in the network. To address this issue, recall the definition of neural computation of section 3.2

$$a_j^l = f(z_j^l), \tag{4.5}$$

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l, \tag{4.6}$$

where a_j^l denotes the output of the activation function f for the j-th neuron in the l-th layer given that neurons weighted input z_j^l. w_{jk}^l denotes the weight between the k-th neuron in layer $l-1$ and neuron j in layer l and b_j^l is the bias for that neuron.

Given the activations a_j^L of the final layer L, the cost J can be computed as shown in previous sections. For now, an intermediate variable δ_j^l will denote the error for neuron j in layer l

$$\delta_j^l = \frac{\partial J}{\partial z_j^l}. \tag{4.7}$$

Calculating gradients $\partial J / \partial w_{jk}^l$ given that error is easy using 4.6, 4.7 and the chain rule for partial derivatives

$$\begin{aligned}\frac{\partial J}{\partial w_{jk}^l} &= \frac{\partial J}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\ &= \delta_j^l \frac{\partial z_j^l}{\partial w_{jk}^l} \\ &= \delta_j^l \frac{\partial}{\partial w_{jk}^l} \left(\sum_{\tilde{k}} w_{j\tilde{k}}^l a_{\tilde{k}}^{l-1} + b_j^l \right) \\ &= \delta_j^l a_k^{l-1}. \end{aligned} \tag{4.8}$$

We recall that a_k^{l-1} is not dependent on w_{jk}^l, therefore its partial derivative w.r.t. that weight is one. Similarly, $\partial J / \partial b_j^l$ is obtained as

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l \frac{\partial}{\partial b_j^l} \left(\sum_{\tilde{k}} w_{j\tilde{k}}^l a_{\tilde{k}}^{l-1} + b_j^l \right) = \delta_j^l. \tag{4.9}$$

The remaining question is how those errors are computed. Again, we start by applying the chain rule

$$\delta_j^l = \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \underbrace{\frac{\partial J}{\partial z_k^{l+1}}}_{\delta_k^{l+1}}. \tag{4.10}$$

With $\partial z_k^{l+1}/\partial z_j^l$ being

$$\begin{aligned}\frac{\partial z_k^{l+1}}{\partial z_j^l} &= \frac{\partial}{\partial z_j^l}\left(\sum_{\tilde{j}} w_{k\tilde{j}}^{l+1} a_{\tilde{j}}^l + b_k^{l+1}\right)\\ &= w_{kj}^{l+1}\frac{\partial a_j^l}{\partial z_j^l}\\ &= w_{kj}^{l+1} f'(z_j^l),\end{aligned} \tag{4.11}$$

equation 4.10 can be simplified to

$$\delta_j^l = \sum_k w_{kj}^{l+1}\delta_k^{l+1} f'(z_j^l). \tag{4.12}$$

This yields a convenient way of computing a layer's errors given the following ones. By starting at the last layer and propagating the errors backwards through the network, all gradients w.r.t. parameters can be computed given a single forward and backward path.

The only layer that has no following layer is obviously the last one. Therefore, δ_j^L is

$$\begin{aligned}\delta_j^L &= \frac{\partial J}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L}\\ &= \frac{\partial J}{\partial a_j^L} f'(z_j^L),\end{aligned} \tag{4.13}$$

which completes the set of equations needed for the famous back-propagation algorithm of [51] that allows for an efficient parameter update (equation (4.1)).

4.3.1 Vanishing Gradients

It is probably more intuitive to start with the MSE as the used cost function first. To stay in line with the notation introduced in section 4.3 we denote the prediction for label j as a_j^L in the following.

A partial derivative of the MSE in equation (4.4) with respect to an arbitrary model parameter θ is then

$$\frac{\partial J_{\text{MSE}}}{\partial\theta} = \frac{1}{K}\sum_k \left(p_k - a_k^L\right)\frac{\partial a_k^L}{\partial\theta}. \tag{4.14}$$

For label predictions, a_k^L is typically the output of a soft-max layer. In that case, the remaining partial derivative is small if the model is very wrong at predicting a label [20]. This effect is called *saturation.* In turn, this causes the gradient to be small as well in those cases. Thus, the learning is slow at the beginning of the training which is typically unwanted. However, by smartly choosing a cost function in which this is not the case, we can obtain strong gradients also when the model is very wrong which speeds up the training process significantly [20]. We rewrite the soft-max function in equation (3.12) to

$$a_j^L = \frac{\exp a_j^L}{\sum_k \exp a_k^L}, \tag{4.15}$$

whose partial derivative is

$$\frac{\partial a_j^L}{\partial z_m^L} = \begin{cases} a_j^L \left(1 - a_j^L\right) & \text{for } m = j, \\ -a_j^L a_m^L & \text{for } m \neq j. \end{cases} \tag{4.16}$$

With this, the CCE

$$J = -\sum_k p_k \log a_k^L, \tag{4.17}$$

where p again denotes the one-hot encoding of the ground truth, can be differentiated w.r.t. the soft-max input z_m^L at label position m as

$$\begin{aligned} \frac{\partial J}{\partial z_m^L} &= -p_m \frac{1}{a_m^L} \frac{\partial a_m^L}{\partial z_m^L} - \sum_{k \neq m} p_k \frac{1}{a_k^L} \frac{\partial a_k^L}{\partial z_m^L} \\ &= -p_m \left(1 - a_m^L\right) + a_m^L \sum_{k \neq m} p_k \\ &= -p_m + a_m^L \sum_k p_k = a_m^L - p_m. \end{aligned} \tag{4.18}$$

The resulting gradient does not suffer from saturating or vanishing gradients when the model is wrong the most. For this reason, CCE is used over MSE in models which end on a soft-max layer.

4.4 Momentum

As mentioned in the previous section, gradient descent modifies the network parameters in small steps towards the opposite direction of the strongest gradient of the cost function. The closer the cost of the current model is to a stationary point, the smaller becomes the magnitude of its gradient (assuming smoothness in the close vicinity). Thus, the model will not change significantly for a long period of time. This leaves us with two problems: First, since the gradient gets smaller as we approach a local minimum, getting to the bottom takes long. Second, we stop training if the model did not change for a long period and therefore get stuck in such a minimum or saddle points.

Momentum helps with escaping spurious local minima and saddle points [47]. At its core, the idea is to not change parameters directly with gradient updates, but instead model a *velocity* v_θ of the parameters as they move on the cost-surface. This movement is similar to a physical ball with mass that rolls down a hill [47]. It accumulates gradients

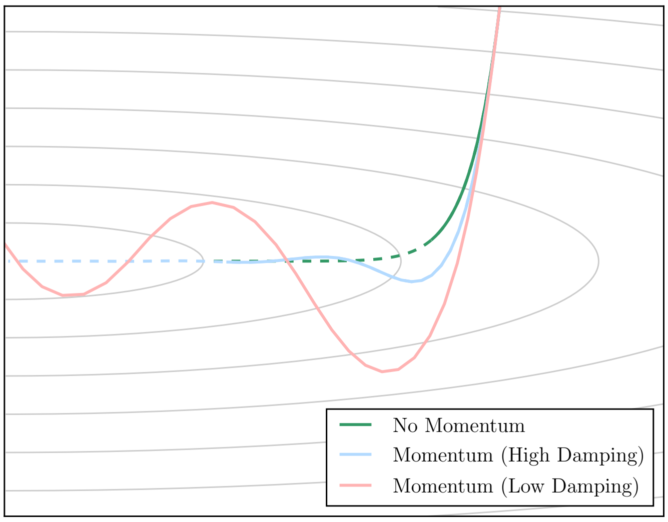


Figure 4.1: Example of trajectories on the cost surface with and without momentum.

and therefore increases its speed. Mathematically, this can be expressed by extending equation (4.1) to

$$v_\theta^{t+1} = \mu \cdot v_\theta^t - \lambda \cdot \frac{\partial J}{\partial \theta}, \tag{4.19}$$

$$\theta^{t+1} = \theta^t + v_\theta^{t+1}. \tag{4.20}$$

The factor μ can be seen as a damping or viscosity constant that is similar to the friction between ball and surface. It ranges from 0 to 1 where low values correspond to a higher damping as the previous velocity is forgotten sooner. Confusingly, this is commonly referred to as *momentum* as it resembles the physical model [47].

Figure 4.1 shows a simple example of a two-dimensional cost surface where thin lines indicate equal cost potentials. The green line is the trajectory of the gradient descent process without momentum and the blue and red line show training with momentum. To show the difference in convergence speed, all lines were drawn solid for the first 30 iterations and dashed afterwards. Note how the movement of the green line almost stops once the vertical center is reached. Using low damping leads to overshooting which causes the movement to oscillate around the local minima, possibly escaping it and discovering lower valleys.

Nowadays not local minima but saddle points are considered as the main problem when training deep neural networks [11]. The reason is that in a very high dimensional space, it is sufficient if a good fraction of dimensions has reached a local minimal to shrink the magnitude of the gradient significantly. Meaning that there are dimensions left in which the optimization could still improve, but the overall gradient is just weak and we do not make enough progress in time. Using a momentum term that accumulates the velocity of

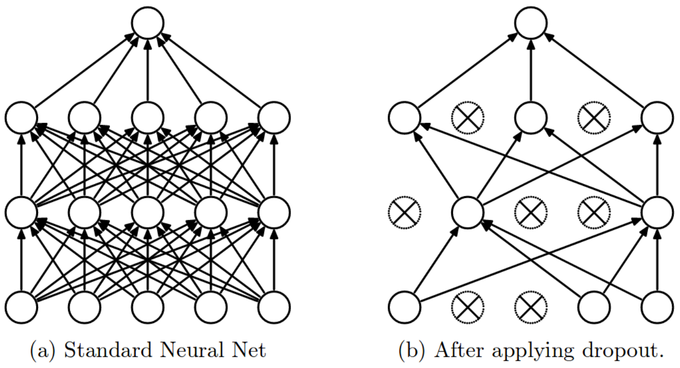


Figure 4.2: Visualization of a neural network before and after applying dropout. (a) An exemplary neural network with two hidden layers. (b) Tinned model after dropping some neurons randomly. Crossed units are dropped. Figure taken from [59].

the trajectory helps to escape from saddle points.

4.5 Regularization

Regularization is referred to as an extension of the cost function by a prior on the parameters, *i.e.*, having a small norm of the weights in the network. The most commonly seen regularization in the context of neural networks is weight decay which assigns a cost to the magnitude of the model parameters. First introduced by [65], the cost J is extended with weight decay term

$$J(\tilde{\theta}) = J(\theta) + \frac{\mu}{2} \sum_i \theta_i^2 \tag{4.21}$$

to explicitly account for too high parameters θ in the network. This new cost reduces model complexity by limiting the growth of weights [35].

One way of interpreting weight decay is that it changes the cost-surface we optimize on. Whereas momentum affects the path we take on the cost-surface towards a (local) optimum.

4.6 Dropout

While momentum and regularization were brought up a long time ago, dropout is a rather new idea. In statistics, it is commonly known that there is a relation between the number of model parameters and data samples. The conception is that one needs more data than parameters in a statistical model [29]. Traditionally the amount of model parameters was restricted to match the amount of training data available by a certain ratio. However nowadays, neural networks which are used in computer vision have a lot more parameters than the number of data samples they are trained on.

The work of [59] tries to address this issue. They argue that in a Bayesian sense,

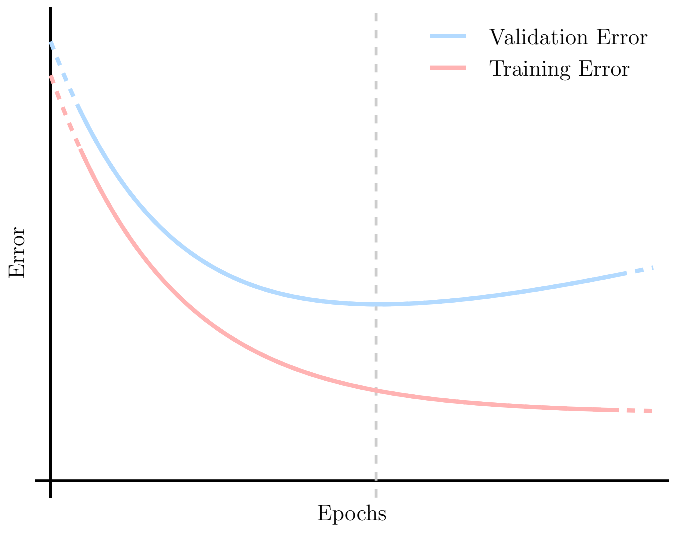


Figure 4.3: Theoretical example of error development throughout the training procedure.

the optimal way of regularizing a fixed-sized model, is to use multiple models and weight each setting with its posterior probability given the training data [59]. As this requires high computational power, they approximate it by sharing weights between the models. Instead of generating multiple networks, they define a probability for each neuron to be excluded (*dropped out*) from the forward computation. The remaining neurons form a sub-model. Since in each training trial, a new sub-model is drawn, an exponential number of such networks is trained. During test time, no neurons are dropped, hence the full network computes an equally weighted geometric mean of all previously trained models [59]. Figure 4.2 gives an example of a sub-model drawn from the full network through applying dropout.

4.7 Recognizing Overfitting

If we train a neural network by gradient descent, we gradually become better in terms of minimizing the cost function. Methods like stochastic gradient descent and momentum might skew or soften that statement a bit, but in principle, we always know how to improve. However, this also implies that we typically do not know when the model is simply learning noise in the data. This is problematic because we want to apply the network to unseen data and let it predict labels we did not know beforehand.

Through gradient descent the model becomes better on the data it trains on, but at a certain point any advance on the training set might not translate to unseen data anymore. To realize when overfitting takes place, we keep a subset of data on holdout and do not use it for training. We can then use that subset and validate our models performance on unseen data regularly during training. Figure 4.3 shows a theoretical example of such error behavior over the training process measured in epochs. The two curves show the progress on the training- and validation set. The vertical line marks the lowest validation error. After this point, the model does not improve on unseen data any longer. In fact, it can even happen that the model concentrates on the noise in the training data so much that it is misled on unseen data. Thus, the model can get worse on the validation set if the training continues. Due to that we should stop the training once the model does not improve on unseen data any more.

Now we can decide when to stop training based on a validation set. However, this introduces a bias towards that set. To be able to access an objective measure of the models capabilities on unseen data, we need a third set that is never used for any decision. That set is then called test set. Common public challenges like [8] keep that sets labels secret to ensure a bias-free evaluation.

5

Conditional Random Fields

Contents

In this chapter, we focus on conditional random fields (CRFs) as they are a main building block of our proposed model.

We will define the CRF first, then we show how to combine it with a convolutional neural network (CNN) and train them together. A major challenge in this context is the gradient computation w.r.t. the model parameters. Because of this, the used gradient estimation is covered at the end of this chapter.

5.1 Optimization Problem

The CRF consists of a set of random variables that are conditioned on the given input image I and the model parameters θ. Each variable is defined over the finite set of possible labels $\mathcal{L} = \{l_1, l_2, \ldots, l_N\}$. The random variables form an energy function $f(x|I,\theta)$ which is interpreted as a Gibbs distribution

$$P(x|I,\theta) = \frac{1}{Z(I,\theta)} \exp\left(-f(x|I,\theta)\right), \tag{5.1}$$

where $Z(I,\theta)$ is the partition function [37]. The distribution describes the likelihood of a particular label assignment given the known image. In this sense, we want to pick the most likely labeling as our prediction.

While the images are given, we can change the model parameter. Since the energy function is also dependent on those parameters, we can interpret learning as the search for the parameter setting which most accurately assigns a high probability to a correct labeling. The energy of a particular assignment $x \in \mathcal{L}^{|\mathcal{V}|}$ is composed of unary terms ψ_i and pairwise interactions $\psi_{i,j}$ and can be written as

$$f(x|I,\theta) = \sum_{i\in\mathcal{V}} \psi_i(x_i) + \sum_{i,j\in\mathcal{E}} \psi_{i,j}(x_i, x_j), \tag{5.2}$$

where $\mathcal{V}$ is the set of variables in the input I and $\mathcal{E}$ is the set of edges between variables. It is noteworthy that $\mathcal{E}$ in this definition is not just a local neighborhood but in general can be any arbitrary set of connections between any two variables.

Having defined our model, we note that finding the most likely labeling $\bar{x}$ is equivalent to maximizing the Gibbs probability and therefor the same as minimizing the CRF energy of equation (5.2)

$$\bar{x} \in \arg\min_x f(x|I,\theta). \tag{5.3}$$

In the following we will investigate the CRF energy in more detail and define the unary- and pairwise terms explicitly.

5.1.1 Unary Terms

In the energy model (5.2), ψ_i are *unary* terms which model each variable individually, *i.e.*, they express our belief in a nodes class belonging, independent of the other nodes it is connected to. The unary terms are derived from the probability distribution over all possible labels. See chapter 6 for more details on how we use a CNN to provide that distribution for each pixel in the image. For clarification, the distribution describing the unary terms is *not* the same as the one in equation (5.1).

With $p_i(x_i)$ being the x_i-th element of that probability distribution at location i, we define

$$\psi_i(x_i) = -p_i(x_i). \tag{5.4}$$

The idea behind this, is that state-of-the-art CNNs are exceptionally good at semantic reasoning such that in regions where they are confident on their prediction, we consider them trustworthy. Therefore, we assign a low energy (negative sign) to likely label. However, the CNN used in this thesis struggles around object boundaries to sharply align its prediction. Thus, the likelihood is lower and we rely more on prior knowledge which is modeled as pairwise interactions.

5.1.2 Pairwise Terms

We defined $\psi_{i,j}$ as pairwise interactions which incorporate prior knowledge about our modeled data explicitly. That is, that real world objects/structures are bound to consistency in terms of labeling across their spatial expansion. By defining the pairwise terms in a way that penalizes label inconsistencies in regions where label jumps are unusual in terms of prior knowledge, while encouraging them where suitable we guide the predicted solution.

We use a contrast-sensitive weighting of pixel neighbors in this thesis since it fits visual data. See section 6.2 for more details on the implementation used in this work.

5.2 Gradient Estimation

To train the full model, we need to propagate gradients of the loss functions though the CRF to the CNN. Despite our final evaluation criterion being the intersection over union score, we start by optimizing the Hamming loss. Given the predicted solution $\bar{x}$ and the ground truth labeling x^*, the Hamming loss is

$$l(\bar{x}, x^*) = \sum_{i \in \mathcal{V}} [\![\bar{x}_i \neq x_i^*]\!]. \tag{5.5}$$

Note that in previous chapters we considered this as the cost function $J(\theta)$. The goal is to optimize this loss in parameters subject to $\bar{x}$ being a minimizer, *i.e.*, satisfies (5.3). In a discrete setting, minimizing w.r.t. x is non-trivial since the solution only changes if the parameters pass certain breakpoints. Therefore, the gradient is zero almost everywhere which makes training impossible. Instead we relax the problem to an upper bound of the loss and minimize that instead. This approach is known as structured support vector machine with margin rescaling [50, 62] and will be reviewed next.

5.3 The Structured Support Vector Machine

Without further restrictions we consider a weighted loss $\gamma l(\bar{x}, x^*)$ hereafter.

The predicted label assignment $\bar{x}$ has the lowest energy $f(\bar{x})$ and is therefore either equal to the energy of the true label $f(x^*)$ (in the case where $\bar{x} = x^*$) or smaller than that energy. Note that in the latter case, the model is making a false prediction. By extending that to all labels that have a lower energy, we can bound from above

$$\gamma l(\bar{x}, x^*) \leq \max_{x \in \{x | f(x) \leq f(x^*)\}} \gamma l(x, x^*). \tag{5.6}$$

Knowing now that for all considered x, $f(x^*) - f(x) \geq 0$ (per definition), we can further estimate

$$\leq \max_{x \in \{x | f(x) \leq f(x^*)\}} \gamma l(x, x^*) + f(x^*) - f(x). \tag{5.7}$$

Now we can extend that statement to all x because even if $f(x^*) - f(x) < 0$, the maximum will remain the same (or get larger)

$$\leq \max_{x} \gamma l(x, x^*) + f(x^*) - f(x) = \bar{l}(x^*), \tag{5.8}$$

which yields the upper bounded loss $\bar{l}(x^*)$.

The value $\bar{l}(x^*)$ can be also equivalently written as finding the minimum ξ such that

$$\forall x : \gamma l(x, x^*) + f(x^*) - f(x) \leq \xi, \tag{5.9}$$

which reveals the margin property, *i.e.*, in the separable case the energy $f(x^*)$ should be smaller than any other energy by at least $\gamma l(x, x^*)$. If this is not possible, then the statement is relaxed by the slack variable ξ. Here, γ weights the margin against the energy barrier.

5.3.1 Gradient Calculation

Since the input to the CRF is provided through a CNN, our goal is to compute a gradient of the upper bounded loss in equation (5.8) with respect to the CNN parameters. The CNNs output forms the unary terms used in the energy volume $f(x)$ of equation (5.2). Therefore, we derive $\frac{\partial}{\partial f(x)}\bar{l}(x^*)$ in the following. To do so, let us first rewrite $\bar{l}(x^*)$ as

$$\begin{aligned}
\bar{l}(x^*) &= f(x^*) + \max_x \gamma l(x, x^*) - f(x), \\
&= f(x^*) - \underbrace{\min_x f(x) - \gamma l(x, x^*)}_{\hat{x} \in \arg\min_x f(x) - \gamma l(x, x^*)}, \\
&= \gamma l(\hat{x}, x^*) + f(x^*) - f(\hat{x}),
\end{aligned} \tag{5.10}$$

where $\hat{x} \in \arg\min_x f(x) - \gamma l(x, x^*)$ is a solution (among possibly many) of the *loss augmented inference* problem. Since we can perform inference only approximately, we take $\hat{x}$ to be an approximate solution resulting after a fixed number of iterations. With this, the gradient at location i is

$$\begin{aligned}
\frac{\partial}{\partial f_i(x_i)}\bar{l}(x^*) &= \underbrace{\frac{\partial}{\partial f_i(x_i)}\gamma l(\hat{x}, x^*)}_{=0 \text{ (const.)}} + \underbrace{\frac{\partial}{\partial f_i(x_i)} f(x^*)}_{=\begin{cases}1 & \text{if } x_i = x_i^* \\ 0 & \text{else}\end{cases}} - \underbrace{\frac{\partial}{\partial f_i(x_i)} f(\hat{x})}_{=\begin{cases}1 & \text{if } x_i = \hat{x}_i \\ 0 & \text{else}\end{cases}}, \\
&= \begin{cases} 1 & \text{if } (x = x^*) \wedge (x^* \neq \hat{x}), \\ -1 & \text{if } (x = \hat{x}) \wedge (x^* \neq \hat{x}), \\ 0 & \text{else.} \end{cases}
\end{aligned} \tag{5.11}$$

This can elegantly be denoted as

$$\frac{\partial}{\partial f_i(x_i)}\bar{l}(x^*) = [\![x_i^* = x_i]\!] - [\![\hat{x}_i = x_i]\!], \tag{5.12}$$

using the previously introduced Iverson bracket. Using the chain rule we obtain gradients for $\psi_i(x_i)$ as

$$\frac{\partial}{\partial \psi_i(x_i)} \bar{l}(x^*) = \frac{\partial \bar{l}(x^*)}{\partial f_i(x_i)} \cdot \underbrace{\frac{\partial f_i(x_i)}{\partial \psi_i(x_i)}}_{=1} = \frac{\partial}{\partial f_i(x_i)} \bar{l}(x^*), \tag{5.13}$$

which are back-propagated to the CNN in the learning phase.

Because of minimizing the upper bound of the loss function, we eventually also lower the actual loss. Another way of interpreting this is that we repeatedly increase the energy of whichever label violates the constraint in equation (5.9) the most while reducing the energy of the correct label at the same time. We will further explain this approach graphically in the next section.

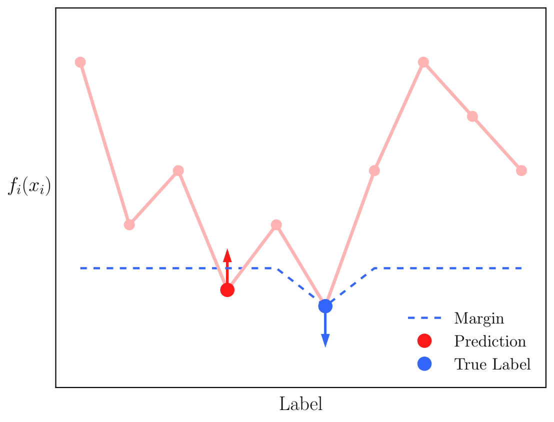
5.3.2 A Graphical Interpretation

To give an intuitive understanding of the above described procedure, we want to give a graphical interpretation of the gradient estimation in the following. Figure 5.1 shows an exemplary plot containing the CRF-energy distribution over a set of possibly assigned labels. Additionally, we plot the margin which in our case is the Hamming loss of equation (5.5). The loss is zero for the true label and a constant γ elsewhere. γ was introduced in section 5.3 to weight the margin against the energy barrier.

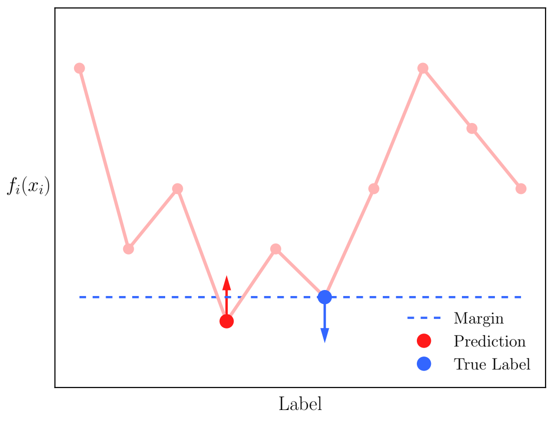
The models prediction is the label which yields the lowest energy evaluated relatively to the margin as seen in figure 5.1a. The arrows indicate the obtained gradients in this example.

The relative energy distribution is shown in figure 5.1b where the margin was subtracted from the energy values to visualize the difference. Finding the relative minimal energy is then considered as the loss-augmented inference problem of equation (5.12).

In the limiting case where γ approaches zero, the margin is flat. As it can be seen in equation (5.12), we only obtain a non-zero gradient if the predicted and true label are distinct (figure 5.2a). This is only the case as long as the predictions corresponding energy value is slightly lower than the true label ones. Once this is not the case the positive and negative peak cancel each other out and the overall gradient vanishes. Thus, the learning stops for that data sample. As visualized in figure 5.2b, this gives poor results - even for training data - as the correct label yields just slightly lower energy than the other possible label candidates. It is intuitive that such a model is weak to input noise. In general, we obtain gradients until the true label is better than any other label by the margin $\gamma l(\bar{x}, x^*)$.

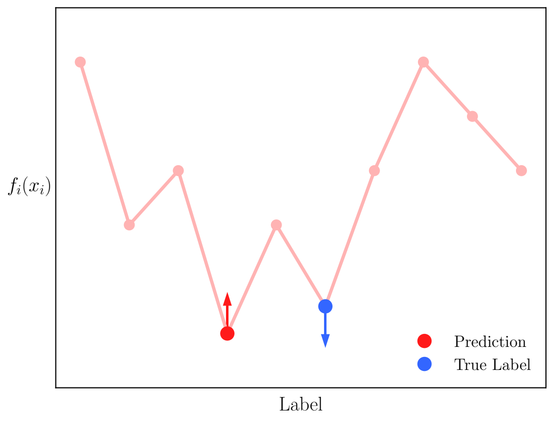


(a) Prediction relative to the margin.

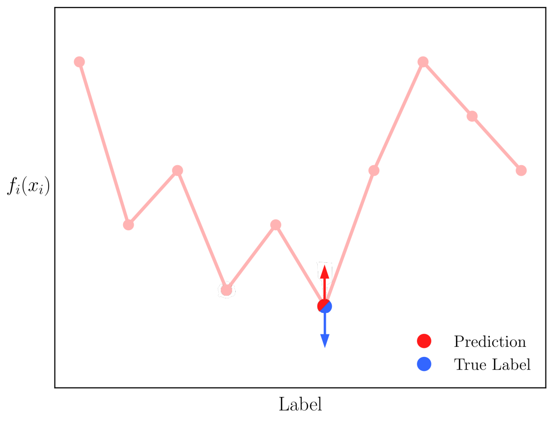


(b) Prediction after loss augmentation.

Figure 5.1: CRF energy over the set of possible labels for a single output variable. The red dot indicates the models prediction while the blue dot denotes the ground truth label. The gradient of the CRF layer (indicated as arrows) is estimated relatively to the loss-proportional margin (dashed blue). Alternatively, this can be interpreted as a loss-augmentation where each energy value is reduced by its corresponding margin.



(a) Prediction without a margin. We still obtain a graident here.



(b) Prediction without a margin. Gradient vanishes once the true lable is slightly better than the second most probable one.

Figure 5.2: Corner case of the gradient estimation in which the loss-augmentation is neglected due to a small weighting γ.

6 Combined Model

Contents

In this chapter, we introduce a combination of convolutional neural network (CNN) and conditional random field (CRF). In the following we will explain the model which was used in later sections to perform the experiments.

6.1 Unary Network

We build on the fully convolutional network architecture of [41]. In this work, we call this model *unary network*, because it computes the unary- or data costs in our conditional random field formulation (defined in (5.2)). The network is based on the prominent *VGG16* model of [56] which was trained for image classification. It is then extended with skip-connections and up-sampling layers in order to achieve fine grained predictions.

Figure 6.1 shows the architecture of the unary network. *Conv* blocks indicate multiple convolutional layers as described in section 3.4. They are used to extract trainable feature-maps from the given input. The output of each convolution layer is activated using the *ReLU* function. Stacking multiple layers together allows to learn features of different abstraction. A convolution can be seen as filtering the data with a (typically small) kernel. As a result, each output neuron only sees a small subset of the input called *receptive field*. *Pool* blocks indicate max-pooling layers. They reduce the spatial resolution by aggregating information of nearby pixels. This allows the network to learn hierarchical features at different levels of detail. When information is propagated down through the model, the representation of the data passes more and more pooling layers. Each of

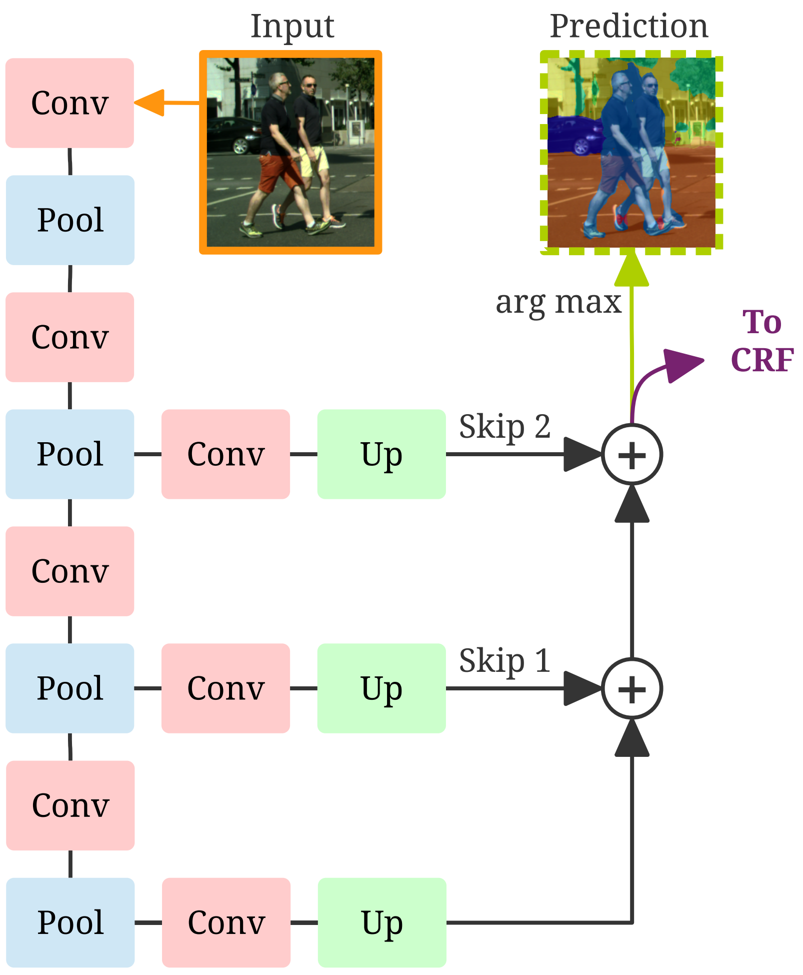


Figure 6.1: Detailed visualization of the unary network. The model produces a predicted label assignment as well as unary input to the subsequent CRF. *Conv* boxes represent multiple convolutions while *Pool* indicates max-pooling. *Up* are upsampling stages that rescale the predictions to the original image resolution.

these layers halves the spatial resolution. The representation at the end of the lowest branch has passed the most pooling layers and therefore has the lowest resolution. By up-sampling that representation bi-linearly, using an *up*-layer, we obtain a full-scale prediction with poor segmentation borders but high confidence within objects. In other words, the network is very certain about the semantic content of the image, *e.g.*, certain to see a car or pedestrian, but is uncertain about the precise position and boundaries of said content. In mathematical terms this means that at the borders of objects, the probability distribution does not clearly suggest a particular label.

Following [41] we add skip branches to the network. As seen in figure 6.1, these branches bypass pooling operations to preserve finer details at the cost of having a less abstract (and therefore less noise resistant) representation of the data. Meaning that those

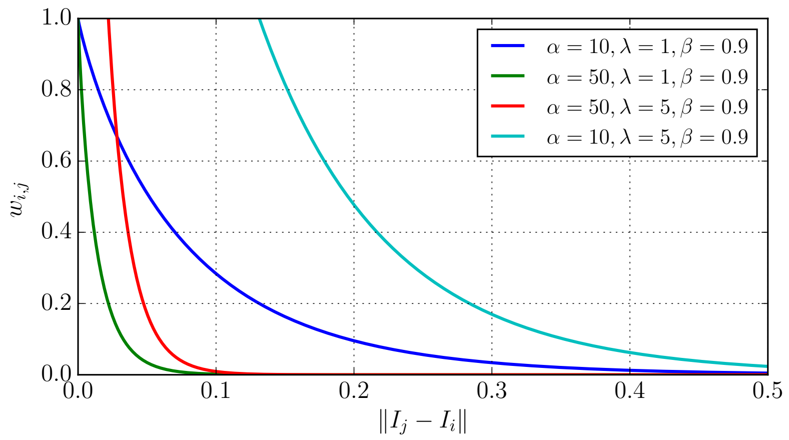


Figure 6.2: Contrast sensitive weighting under various parameter settings. Weights are evaluated over a range of illumination changes. Best viewed in color.

branches are less sure about the semantic content but know the actual object boundaries better. We fuse those branches through a linear combination and let the model learn in a training stage how to combine those branches to obtain good predictions. Thus, the skip-branches guide the low-resolution prediction along object boundaries.

6.2 Pairwise Relations

While in the neural network, the relations between individual output neurons is given by the overlap in their receptive fields, CRFs allow to model certain prior knowledge explicitly.

As outlined before, we can incorporate prior knowledge about natural images into our model in the form of a penalty. We can reasonably assume, that given a neighboring pair of pixel, the assigned labels should only differ if the visible object has changed. In other words, we assume that label discontinuities only occur across object borders. While modeling those borders is difficult in practice we can approximate this by restricting label jumps to visual edges in the image.

We use the Iverson bracket $[\![\cdot]\!]$ notation

$$[\![cond]\!] = \begin{cases} 1 & \text{if } cond \text{ is true,} \\ 0 & \text{else,} \end{cases} \tag{6.1}$$

in the following. With this, we define the locally weighted Potts model [48] as

$$\psi_{i,j}(x_i, x_j) = w_{i,j}[\![x_i \neq x_j]\!], \tag{6.2}$$

where weights $w_{i,j}$ depend on the image I. If x_i and x_j are assigned the same label, $\psi_{i,j}(x_i, x_j)$ is zero and $w_{i,j}$ otherwise. We choose weights that discourage label jumps

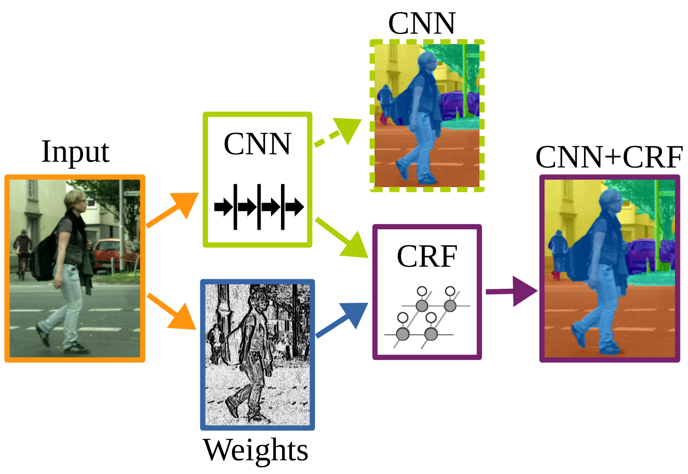


Figure 6.3: The illustration contains an abstract representation of our used model containing its main building blocks. The input is used to compute weights for the pairwise interactions in the CRF. Simultaneously a CNN computes unary potentials which are used in the CRF to produce the final output. No further pre- or postprocessing is performed.

in homogeneous regions (high weight) and encourage them across borders (low weight). Thus, the solution of equation (5.3) is more likely to align its segmentation boundaries with visual edges in the input image I. In that sense, we define *contrast sensitive weights* [4] as

$$w_{i,j} = \lambda \exp\left(-\alpha \|I_j - I_i\|^{\beta}\right), \tag{6.3}$$

where λ weights unary against pairwise terms and α and β are parameters of the model. Furthermore, we restrict the weights to be symmetric *i.e.*, $w_{i,j} = w_{j,i}$. Figure 6.2 shows various such weighting functions for different parameter settings.

6.3 Joint Model

After introducing the *unary model*, we combine it with a CRF to pair good semantic reasoning of the CNN with sharp prediction-borders implied by the prior of the CRF. In a first step, we use a CRF for post-processing which is commonly used in state-of-the-art models such as [5]. Second, we will use the joint training procedure discussed in section 5.2 to train the unary model while the CRF is actively contribution to the model such that the whole model is trained in an end-to-end fashion.

In the formulation of equation (5.2), the CRF takes unary and pairwise terms as its input. The unary terms consist of per-pixel independent data-costs and the pairwise terms guide the smoothing of the final output. We use the linear programming relaxation based method of [55] for CRF inference. This method performs a dual block-coordinate descent algorithm whose main advantage is its massively parallel implementation. We use the publicly available code provided by the authors in our experiments. With a small number of iterations (15 in our case), the inference scheme yields results that are comparable to the best sequential algorithms while consuming less computation time [55].

The main idea of this thesis is that in a final learning phase, the CNN is *not* trained to predict labels directly but receives error gradients from the CRF which does the labeling instead. Thus, the CNN can move some of its responsibilities regarding semantic segmentation (*e.g.*, sharp boundaries) to the CRF. This procedure allows the CNN to deviate from a model that predicts a labeling directly towards providing richer features for the CRF hence increasing the overall performance of the method.

Figure 6.3 shows an abstract representation of our used model. The unary input to the CRF is generated through the unary model. The pairwise interactions are modeled with a nonlinear transformation of the image gradients, the contrast sensitive weighting introduced in equation (6.3). These pairwise interactions encourage label discontinuities across strong object boundaries while discouraging them within objects. To prevent confusion, we call predictions obtained from the joint model *CNN+CRF* where the CRF is used as a post-processing method. Furthermore, we add *joint* in brackets if the predictions were the result of a jointly trained model.

7 Joint Training

Contents

In the following chapter, we outline how we use stochastic gradient descent to train our model throughout different stages. We start by motivating the used initialization, followed by the performed pretraining of the *CNN only* network. Finally, we add the CRF, outline the performed parameter search, and discuss the joint training of our proposed hybrid model.

7.1 Initialization

When observing the groundtruth data in figure 8.1, we can assume that creating these labels involves considerable effort, *i.e.*, drawing the polygons around each class instance or cluster. Therefore it seems intuitive that labeled data for dense predictions, when compared to sparse data, is expensive to obtain. On the other hand, obtaining just a single label is comparably cheap. They can even by obtained (almost) for free by outsourcing the actual labeling task to a crowd, *e.g.*, users on the internet through captchas [13]. As a result we currently have a lot more sparsely- than densely labeled data at our hands.

The *VGG16* model of [56] was originally trained on approximately $1.3M$ images for the ImageNet Challenge [52] which is a popular classification competition. Training on such large datasets induces a high noise robustness to the model, particularly in the early stages of processing. This robustness is usually lacking when learning a model from scratch on significantly less data. Since common datasets for semantic segmentation consist of far less data samples, a common approach in state-of-the-art work (see chapter 8) is to initialize the segmentation model (partially) with the *VGG16* network or similar architectures that are publicly available on the internet. In this thesis, we use this network as well to initialize our model.

7.2 Pretraining the Model

Due to the fixed magnitude of the gradients in equation (5.12) and the consequential slow convergence rate, training our model with the CRF in use right from the start is impracticable. As explained before, we consider the prominent *VGG16* model of [56] as a suitable initialization for our purpose. The model is then extended following [41] to fit the semantic segmentation task. By doing so we transform the last fully connected layers to convolutional ones and add two *skip*-branches as shown in figure 6.1.

We use the training parameters of the *Pascal* model of [41] but adjust the unnormalized learning rate to match the larger images in the dataset. The used parameters are gathered in table 7.1. As we add more branches to the model, we simultaneously decrease the learning rate.

Parameter		Value
	No skip	6×10^{-11}
Learning rate	One skip	6×10^{-13}
	CNN only	6×10^{-15}
Momentum		0.99
Weight decay		5×10^{-4}

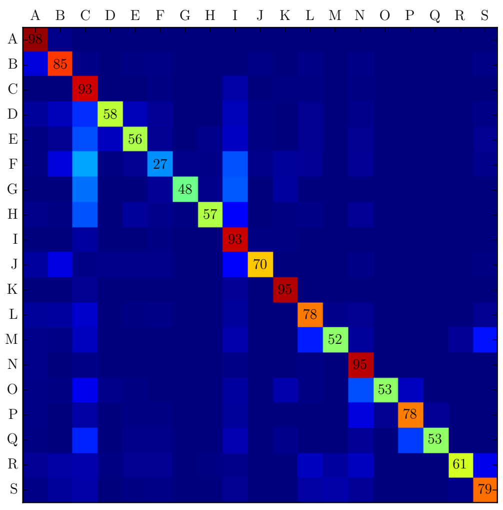
Table 7.1: Parameters used for pretraining the *CNN only* model. The three different learning rates are used in the three training stages respectively.

7.2.1 Staged Training

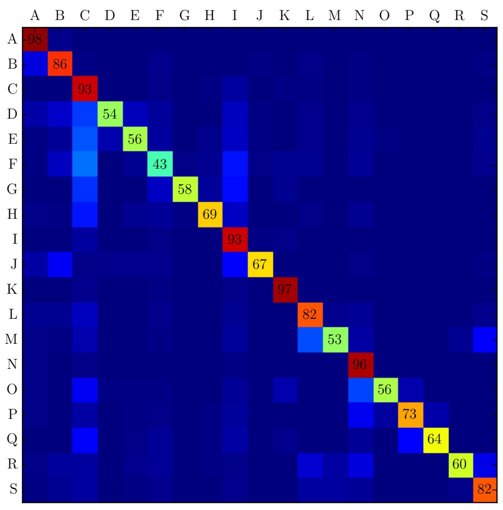
We now use stochastic gradient descend to pretrain our unary model (no pairwise interactions in the CRF) in three stages. Stochastic in this context means that the gradient is not averaged over all training samples but rather over a small subset. Due to hardware limitations we use a single image per training iteration. This procedure stochastically approximates the gradient that is computed on all samples [16].

At the first stage, we do not use any skip branches and train the low-resolution prediction. Then we add one skip branch at a time and repeat the fine tuning with a decreasing learning rate.

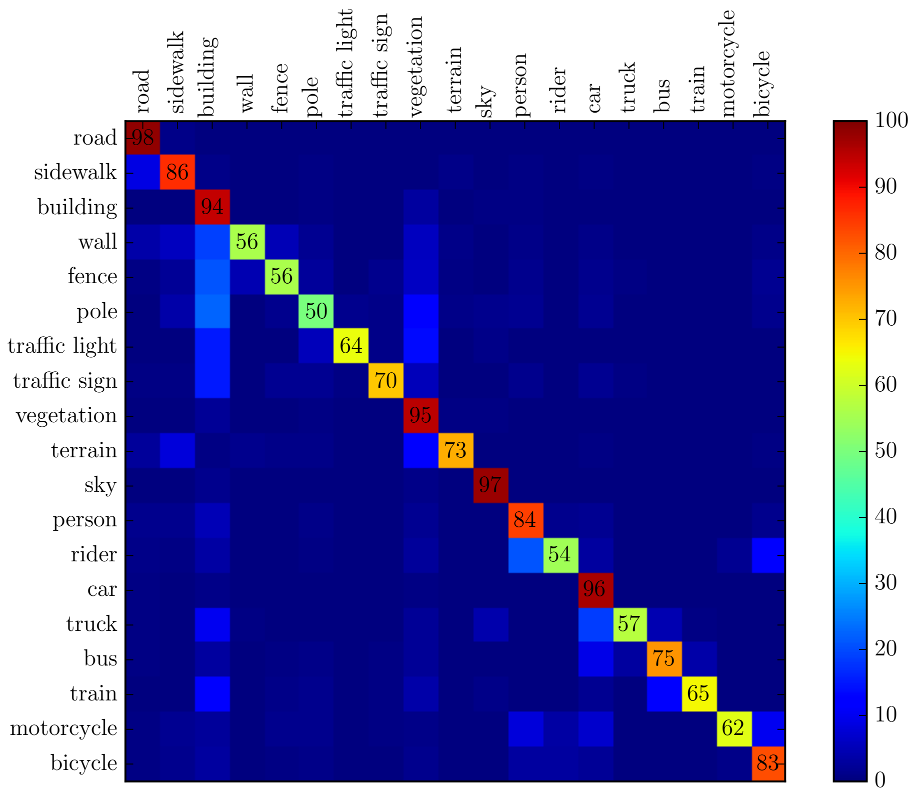
Figure 7.1 contains the confusion matrices obtained after evaluating the model at the end of each training stage on the *Cityscapes*' validation set (see section 8.1.1). We report the normalized intersection over union (IoU; see section 8.1.2) on the 19 classes that are



(a) No skip branches.



(b) One skip branch.



(c) *CNN only* model.

Figure 7.1: Confusion matrices of the *CNN only* model when using no-, one- or all skip branches. This resembles also the networks performance throughout the pretraining stages. This results were estimated on the validation set of the *Cityscapes* dataset [8].

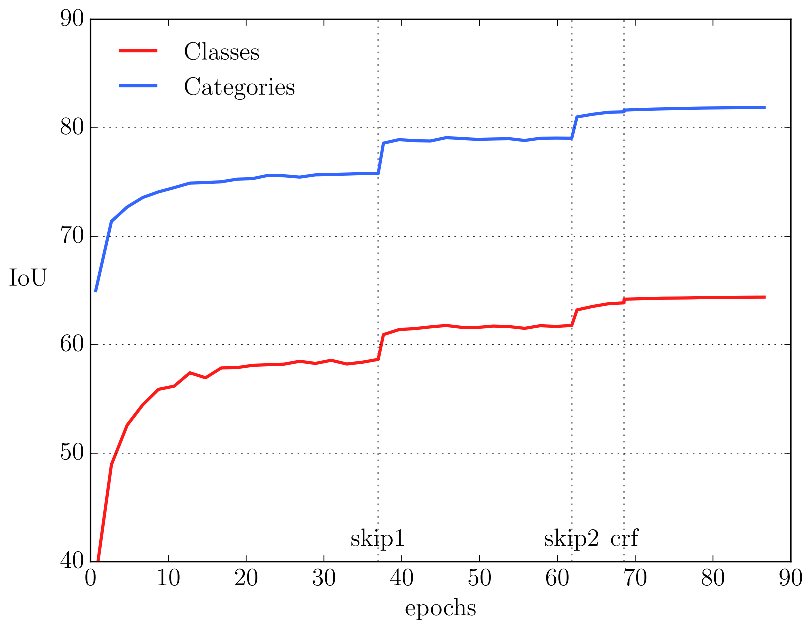


Figure 7.2: Training progress evaluated on the validation set. The IoU is printed for classes and categories (groups of classes). Vertical lines mark the beginning of a new training stage. *Skip* indicates that additional skip branches are added. *CRF* marks the joint training of CNN and CRF.

used for training as well as the public test server. We observe that the model is already quite good in simple categories like *road* or *sky* after the first stage (figure 7.1a). However, classifying fine structures, *e.g.*, the class *pole* is very difficult to detect and likely to be confused with *building* or *vegetation*. After adding all skip branches (figure 7.1c), we see that the model is now more capable of handling finer structures and improves on classes such as *poles*. Generally, we notice two bright vertical regions that indicate that many classes are more likely to be confused with *building* and *vegetation* than with other classes. This is probably due to those two classes being very prominent in the data. These side effects diminish with added skip branches. Note that we subsample the data by a factor of 2 due to hardware limitations which speeds up the process, but limits the overall accuracy of the model.

After adding the CRF and finding useful model parameters, we jointly train CNN and CRF. To that end we use the gradient of section 5.2 and update parameters with a fixed and normalized learning rate of 1×10^{-10} to account for the previously outlined gradient magnitude.

Figure 7.2 shows the training progress by displaying the IoU over the training epochs. The score is evaluated in the validation set (which was not used for learning) of the *Cityscapes* dataset (see section 8.1). Each stage is trained until convergence before altering the architecture. The start of the next stage is marked as a vertical line in the plot (*skip1*, *skip2* and *crf*).

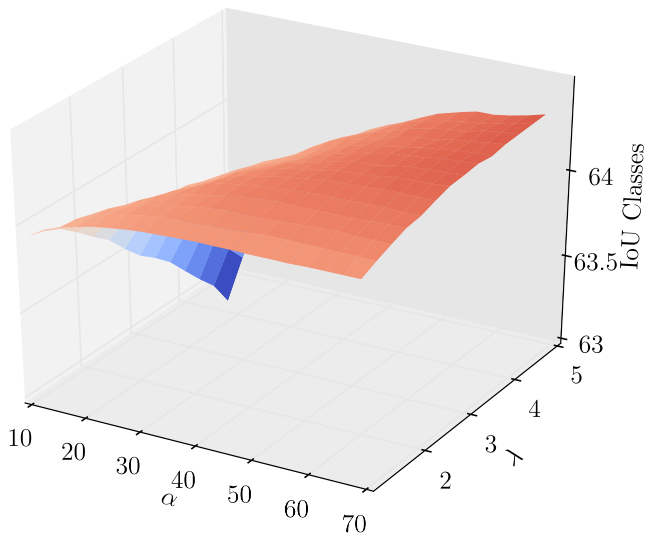


Figure 7.3: Estimated IoU on the Classes as defined in section 8.1.2 over various parameter settings.

7.3 Incorporating Pairwise Interactions

To find an suitable parameter setting for equation (6.3), we perform an exhaustive grid search. Figure 7.3 shows the estimated IoU for various parameter settings. While this estimate seems to favor high values for α and λ, it turns out in practice, that choosing them too high hinders the learning process later on as the non-linearity increases (see figure 6.2). Therefore, we opt for a setting within the plateauing region and let the model learn from there as listed in table 7.2.

Parameter	Value
λ	2.5
α	35
β	0.9

Table 7.2: Parameters used when the CRF is used for post-processing. This setting is also used to start the joint training process.

7.4 End-to-End Training

At last, our network is jointly trained by back-propagating the gradients (equation (5.12)). For this, the loss augmented inference problem must be solved for each training iteration. We can see the slight performance gain at the vertical *crf* line in figure 7.2.

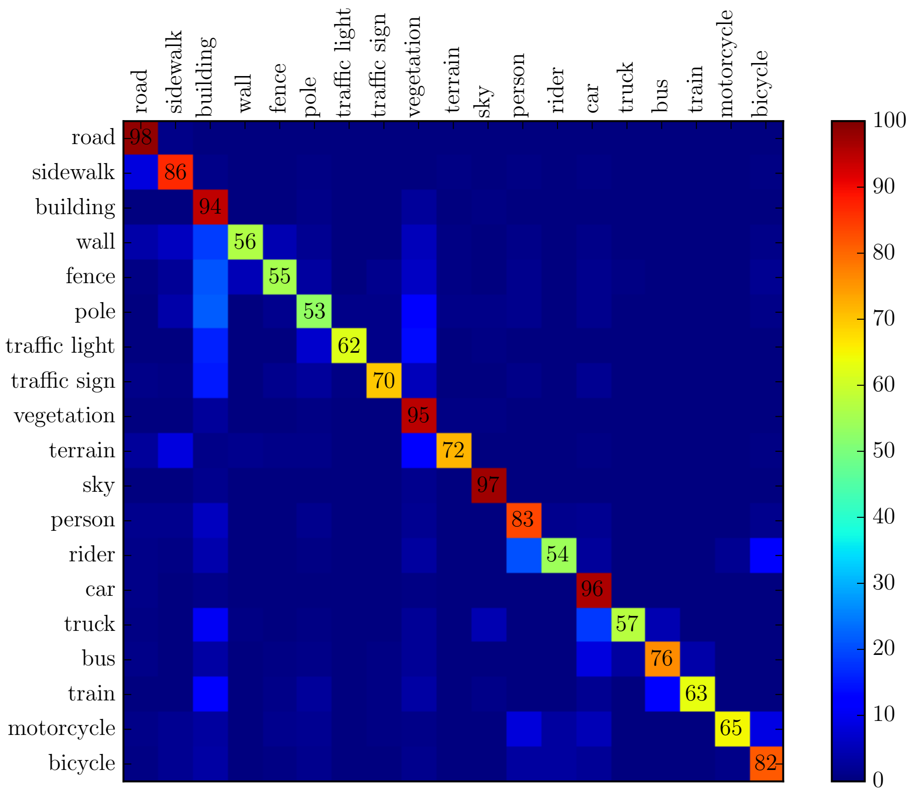


Figure 7.4: Confusion matrices of the *CNN+CRF* model after the joint training. This results were estimated on the validation set of the *Cityscapes dataset [8]*.

The confusion matrix associated to this final model is shown in figure 7.4. When comparing it to figure 7.1c, we see that the model does improve on *e.g.*, the classes *pole* and *motorcycle*. Although the benefits might seem small, chapter 8 shows that this is still a valuable improvement over the CNN alone.

8 Experiments

Contents

Using the *Caffe* [28] and *Theano* framework [61] to implement our models, we investigate the impact of CRFs and joint training on semantic segmentation. We consider two models: our unary- and combined network. We fixate the hyper parameters to the ones determined in table 7.2 throughout the rest of this section and want to give a qualitative comparison of both models. Before that, we introduce the used dataset, as well as the corresponding scoring method which is used to compare different approaches.

8.1 Benchmark

Various research groups across the world pursue different approaches to the task of semantic segmentation. In order to estimate how well a particular approach performs, an common score is beneficial. This is particularly important when it comes to answering the question of where to put research effort.

For visual data, it seems to be a natural choice to agree on a common dataset to train and evaluate on. One of the most used, publicly available datasets is the *Pascal Visual Object Classes Challenge* [15] which provided various challenges from years 2005 to 2012. The set also got various extensions over the years, *e.g.*, *Pascal-Context* [43]. The most recent database for semantic segmentation from that set contains overall approximately 10 thousand images containing objects from 20 different classes. Figure 8.2 contains some examples of the Pascal VOC 2012 challenge that overall contained almost ten thousand images.

Method	Cityscapes Dataset			Intersection over Union	
	Coarse	Stereo	Half Res.	Classes	Categories
CNN only			∗	*63.8*	*81.5*
CNN+CRF			∗	*64.1*	*81.2*
*CNN+CRF (*joint*)*			∗	*64.4*	*81.9*
Ours			∗	62.4	82.3
[45]	∗		∗	64.8	81.3
DeepLab [5]			∗	63.1	81.2
CRF as RNN [70]			∗	62.5	82.7
FCN 8s [41]			∗	61.9	-
LRR-4x [17]	∗			71.8	88.4
Adelaide [39]				71.6	51.7
DeepLab [5]				70.4	86.4
Dilation10 [68]				67.1	86.5
[33]		∗		66.3	85.0
FCN 8s [41]				65.3	85.7
[63]		∗		64.3	85.9

Table 8.1: Comparison of different approaches to the pixel level semantic segmentation task on the cityscapes dataset [8]. *Ours* indicates the score of the jointly trained CNN+CRF model. Italicized methods were evaluated in the validation set, other ones on the test set. The results for other models on this data were taken from their respective paper if available or from [8] otherwise. The score for *FCN 8s* [41] on half sized images was taken from the supplementary material of [8] where no IoU score for categories is listed.

8.1.1 The Cityscapes Dataset

Nowadays, datasets with more challenging content, *i.e.*, images containing many more different objects, are used. One particularly tailored to urban street scenes is the *Cityscapes* dataset. Its advantages over, *e.g.*, the Pascal challenges, are (i) the higher image resolution, (ii) the more complicated and therefore more challenging content. Additionally, the recordings show traffic scenes which are particularly interesting for applications such as autonomous driving.

For those reasons, we use the *Cityscapes Dataset* [8] to train and evaluate our models. This recently released database consists of 5.000 images with high quality fine annotations. The samples were taken in 50 different cities across Germany throughout the year to reduce biases in the data. See figure 8.1 and appendix A.1 for exemplary pictures.

The set is split into subsets of size 2.950, 500, 1.250 for training, validation and test respectively. The 19 different classes in the database represent various things *e.g.*, *cars, humans, streets* or *vegetation.* The dataset provides 20.000 additional images with coarse annotations which are not used in this work.

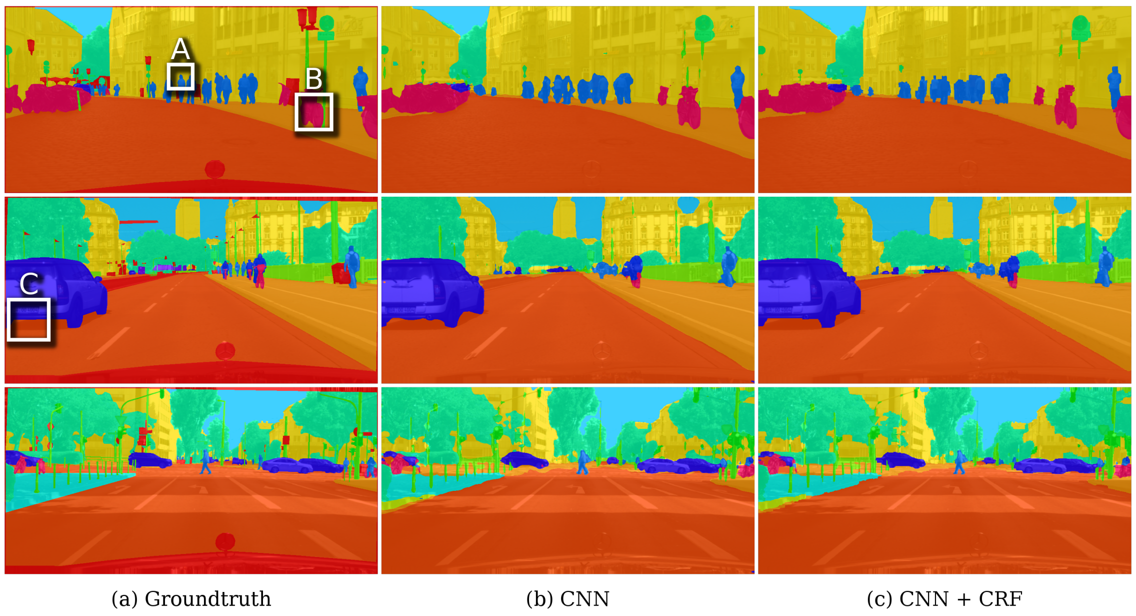


Figure 8.1: Comparison of the CNN only model and the combination of CNN and CRF. All label assignments are blended over the input image to better show the errors around object borders. Column (a) contains the ground-truth image (note that red indicates the ignore-label). Column (b) displays the results obtained without a CRF layer while the last column shows the results of the joint model (CNN followed by a CRF). For better visualization, we marked four regions (A to C) and enlarged them in figure 8.4.

8.1.2 Score

While we discussed the need for an objective comparison previously, the actual used scoring function was left out so far. This section aims to close this gap. It seems intuitive to use the classification rate, *i.e.*, average rate of correctly labeled pixels in the set, for comparison, since it is used for image classification as well. However, if we measure areas with right or wrong label assignment in an image, then usually most regions are classified correctly. The real difficulty in semantic segmentation arises in recovering fine details and structures around object boundaries. The classification rate is usually good if most labels are correct even if the boundaries are not. Imagine a scene in which most content is either *street* or *sky*: missing those few pixels that correspond to *pedestrian* do not lower the classification by a lot since the area is small, but the impact is rather critical. If the classification rate was used, probably most state-of-the art approaches would yield almost equal, close to perfect, scores which makes an objective comparison difficult. What is used instead is a score that penalizes smaller mismatches in object boundaries stronger.

The primary used measurement of prediction quality on the dataset is the PASCAL VOC intersection-over-union (IoU) [14] which is also known as the Jaccard index. The measure is defined with true-positive- (TP), false-positive- (FP) and false-negative (FN)

(c) Original image.

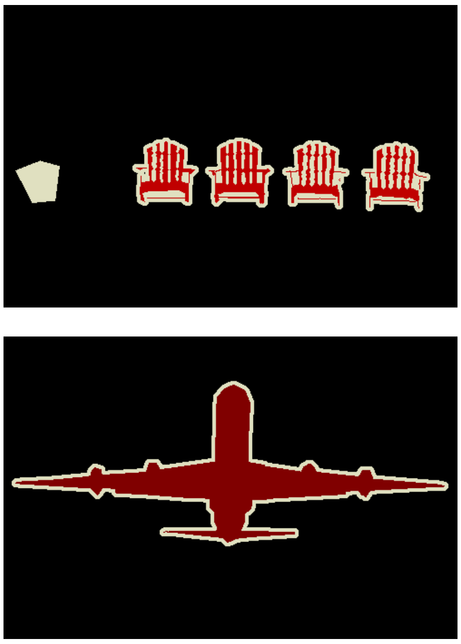

(d) Semantic labeling.

Figure 8.2: Examples of the semantic segmentation task in the Pascal VOC 2012.

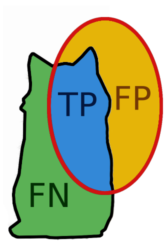


Figure 8.3: Exemplary illustration of the Intersection-Oover-Union score.

pixel predictions for each class as

$$IoU = \frac{TP}{TP + FP + FN}. \tag{8.1}$$

Figure 8.3 shows an example of those three areas for an arbitrary *cat*-labeling. The red contour indicates the predicted area while the black one encircles the ground-truth.

The final score is then computed as the average of the individual class scores. This metric penalizes false-positive predictions, *e.g.*, classifying all pixel as car, as opposed to only considering the average classification rate. Note that *IoU Classes* is computed as the average over individual classes while *IoU Categories* denotes the average over groups of classes *e.g.*, $vehicle = \{car,\ bus,\ bicycle,\ \dots\}$.

8.1.3 Results

Table 8.1 shows the score of our approach on the Cityscapes dataset and compares it to current state-of-the-art models. *CNN only* is our unary network which disregards pairwise interactions. *CNN+CRF* indicates our combined model of CNN and CRF. The lines containing *joint training* in brackets indicates that the model was jointly trained as compared to just using the CRF for post processing. Lines containing *validation* were evaluated locally on the validation set while the *test* line is taken from the public test servers of [8]. Columns *coarse* and *stereo* indicate if a model additionally used weakly labeled (coarse) or stereo data for training respectively. There is an obvious gap between models operating on half sized images and ones that use the full data resolution. We expect that retraining our model on full scale images will give a similar performance increase.

8.2 A Comparison of our Models

To illustrate the benefits of the combined model over the unary network, we provide a comparison of our models. In the first group of models in table 8.1, we compare the three stages of our approach (unary network, CRF for post processing, joint training) on the validation set. We can observe that the use of a CRF as well as the joint training steadily increase the score for the *IoU Classes* task.

In the following, we provide a qualitative comparison of the unary network and the trained combined model based on a few exemplary predictions. Figure 8.1 shows the desired groundtruth (a) as well as predictions of the unary network (b) and the combined model (c). We mark four regions (A to C) which are enlarged in figure 8.4 and further discussed next. In figure 8.4 we additionally print the pairwise interactions in column (e). Hereby, dark regions correspond to low energy which encourages label jumps while bright regions indicate high energy and therefore discourages label discontinuities. For instance, the weights in row A show low cost for a label jump around the head of the person in the center. Thus, the CRF energy is lower in that region and the segmentation boundary is moved towards it. Note that the low-cost region is wide enough to allow multiple boundaries. Due to the four-connected neighborhood in our CRF implementation, the model tends to prefer grid-aligned solutions. We believe that by extending the neighborhood of the CRF to model diagonal or sparse long range connections can lower that tendency. Row B shows similar effects around the front wheel of the bicycle. Additionally, it contains an example of how the CRF enforces consistency (smoothness) of the solution. We can observe that small outliers (visible in the CNN prediction within the pink region) are removed because the corresponding pairwise costs are too high. Finally, row C contains an example of how fattened foreground and curvaceous boundaries become aligned with the true objects in the combined model.

It is noteworthy that we can observe a *staircasing* effect in the obtained solution after the CRF. We argue that this is mostly due to definition of the four-connected neighborhood

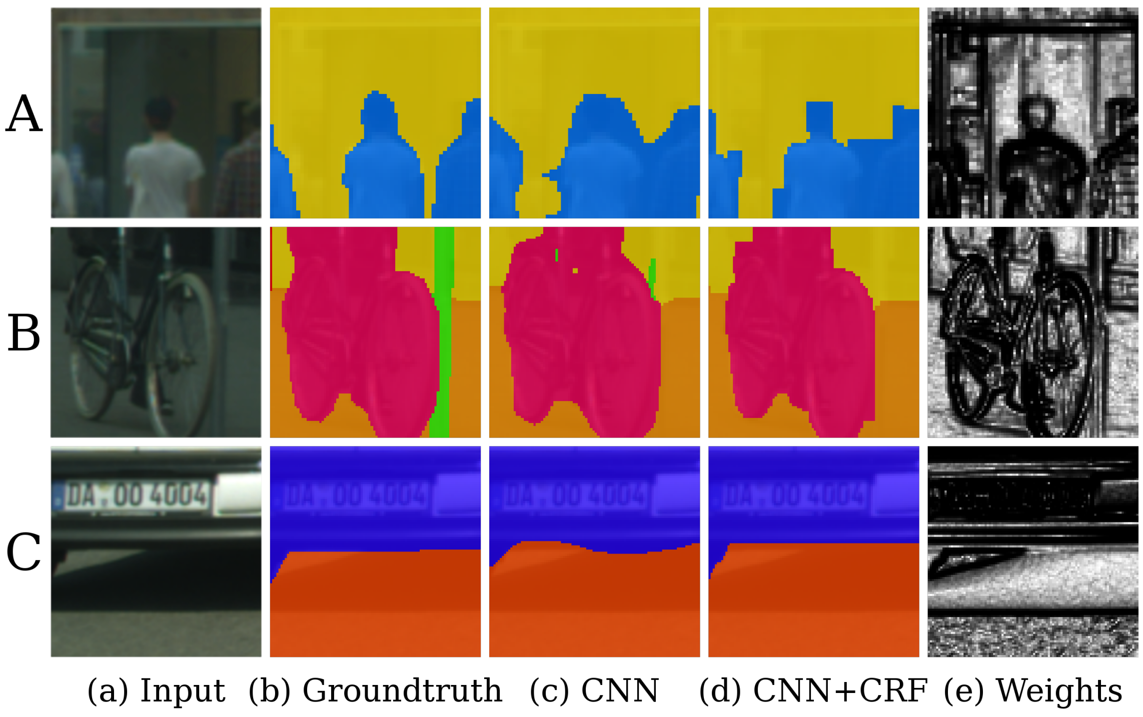


Figure 8.4: Scaled versions of marked regions in figure 8.1. The first three columns contain the groundtruth labeling, the prediction of the CNN model and the results of our CNN+CRF model respectively. The last column shows the pairwise costs of neighboring pixel. Best viewed in color.

in the CRF and could be addressed by extending it.

8.3 Comparison with the State-of-the-Art

To justify our usage of the locally connected CRF implementation of [55] over the commonly used fully connected CRF of [32], we compare our model against the work of [70]. The authors of [70] also use the FCN-8s network of [41] to provide unary terms to the fully connected CRF of [32] which is unrolled and interpreted as layers in a recurrent network. Additionally, they also operate on half sized images. An advantage of that approach is that it allows to directly compute the exact gradient instead of an approximation. Their model is restricted to formulate the pairwise interactions as high dimensional Gaussian filters. In contrast to that, our model approximates the gradient of the CRF using a SSVM formulation which in turn allows for arbitrary definition of pairwise potentials. Table 8.1 shows that both approaches achieve similar improvements over the FCN-8s model and yield almost identical IoU scores, even though our model is simpler.

9 Discussion

9.1 Conclusion

In this thesis, we discussed basic concepts in state-of-the-art neural network architectures. The task of semantic segmentation was tackled using the combination of convolutional neural network and conditional random field. A structured support vector machine formulation allows for joint training of both model building blocks. The combined model allowed the CNN to adapt to the CRFs capabilities and hence improved the quality of the output. We showed how pairwise interactions encouraged label discontinuities across visual edges in the input image. The predicted segmentation boundaries were more likely to align with the true object borders because of that. In the end, the proposed architecture was evaluated on the challenging Cityscapes dataset.

9.2 Future Work

In following work, we want to replace gradient-based pairwise weights with a second CNN. While we already experimented with this, a satisfying training scheme is yet to be found. In general, we believe that contrast sensitive weights have room for improvement and that moving pairwise interactions towards penalizing label jumps within objects is reasonable. Furthermore, we intend to extend the current Potts model to general label compatibilities such that *e.g.*, the label pair $\{car,\ street\}$ is more likely than $\{car,\ bicycle\}$. Other ideas are to tackle the staircasing artifacts discussed in section 8.2 directly by choosing the pairwise-neighborhood in the CRF differently. Possible changes could include to model non-symmetric, diagonal and/or sparse long-range interactions. We expect this to be beneficial for semantic segmentation. Another direction could be to investigate different training methods and loss surrogates, possibly addressing IoU score more directly, within the SSVM approach.

A Appendix

A.1 Additional Results

This section provides additional results on a larger scale than figure 8.1. We provide a qualitative comparison of the CNN only model and the combination of CNN and CRF, jointly trained. All label assignments are blended over the input image to better visualize the boundary alignment around object borders. The images are all contained in the validation set of [8], *i.e.*, were not seen by the model before. In the groundtruth images, red color denotes regions which are ignored during training and evaluation. Thus, the model can predict anything in those regions.

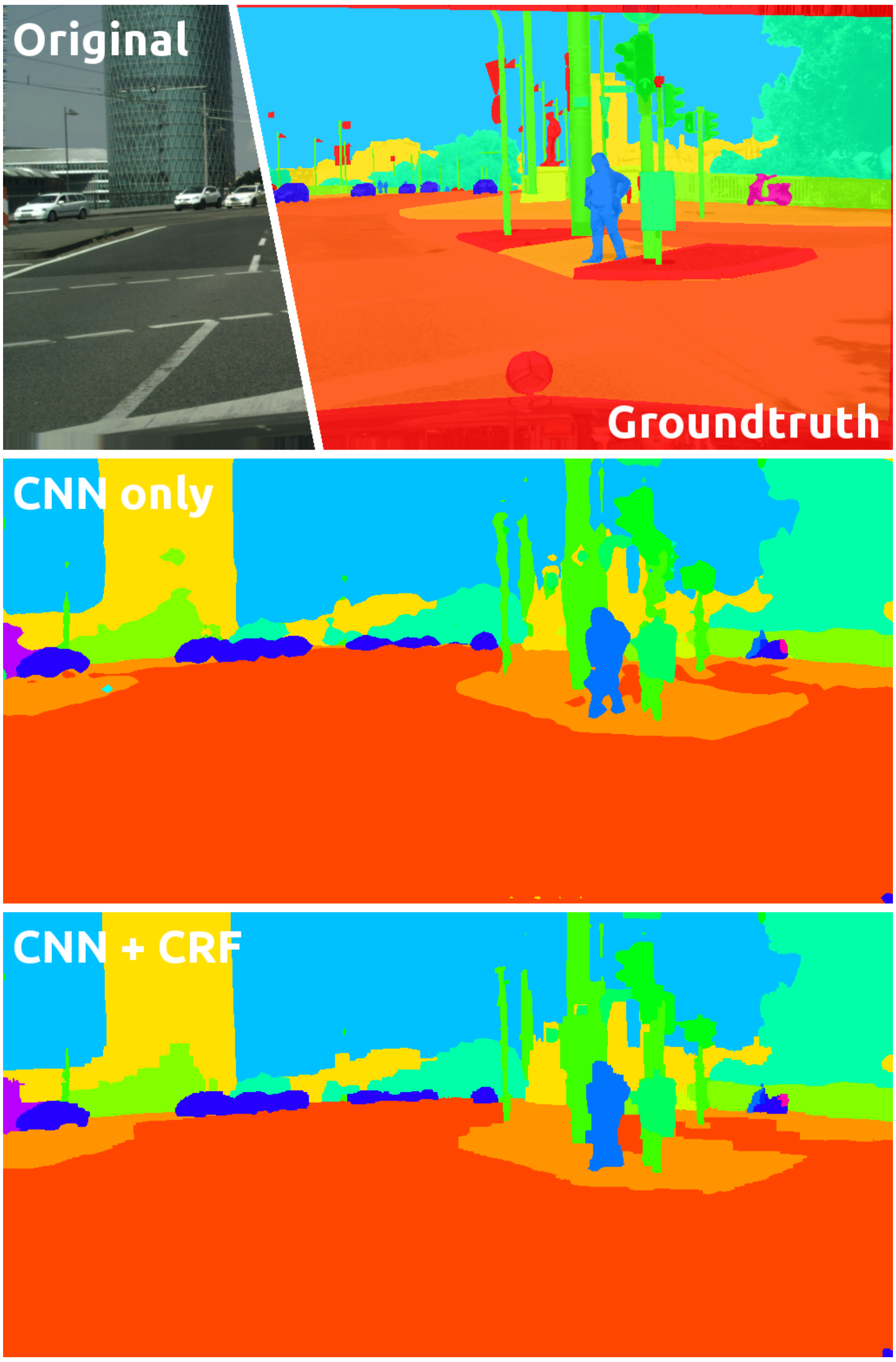


Figure A.1: Comparison of the CNN only model and the combination of CNN and CRF.

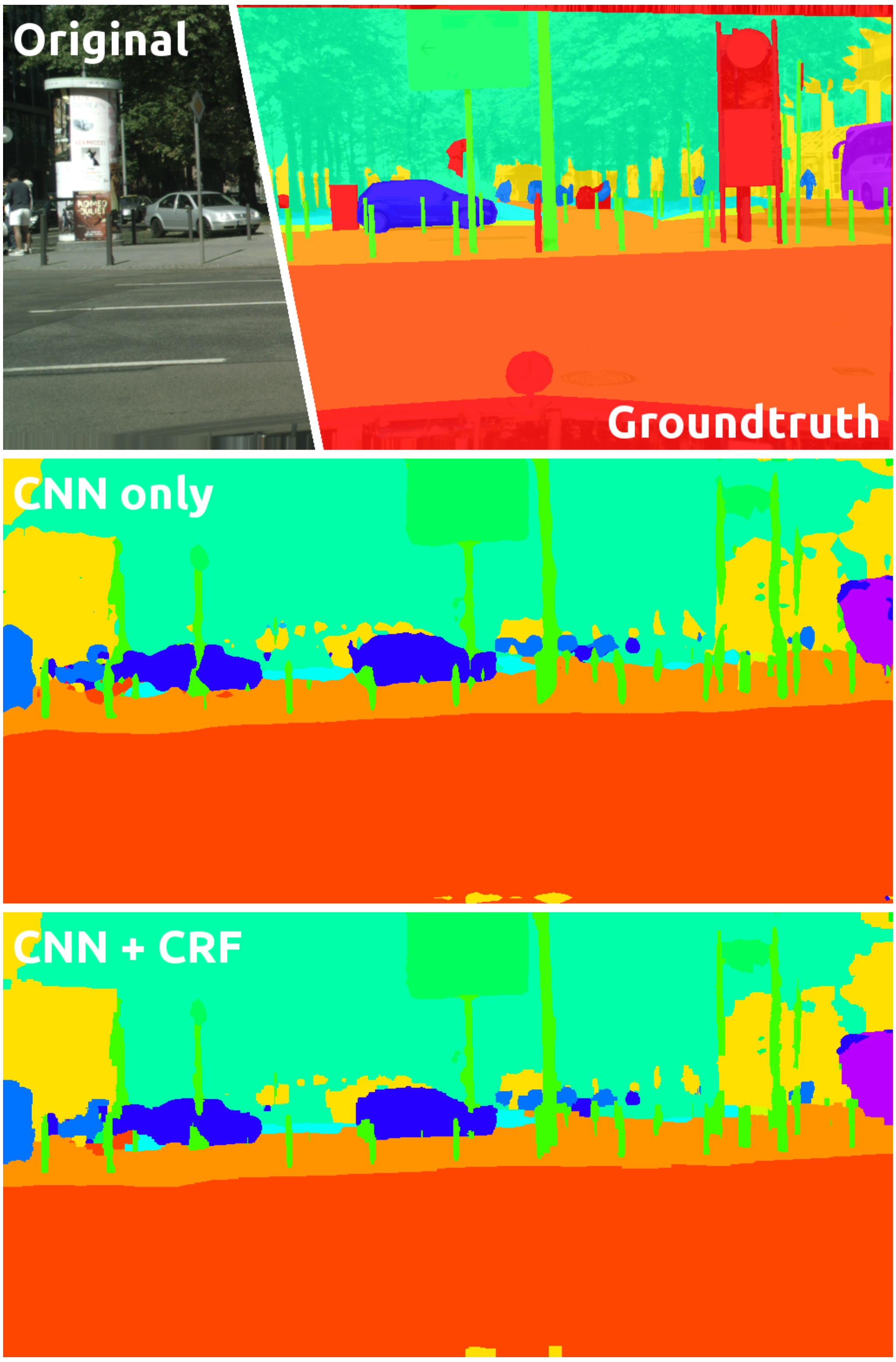


Figure A.2: Comparison of the CNN only model and the combination of CNN and CRF.

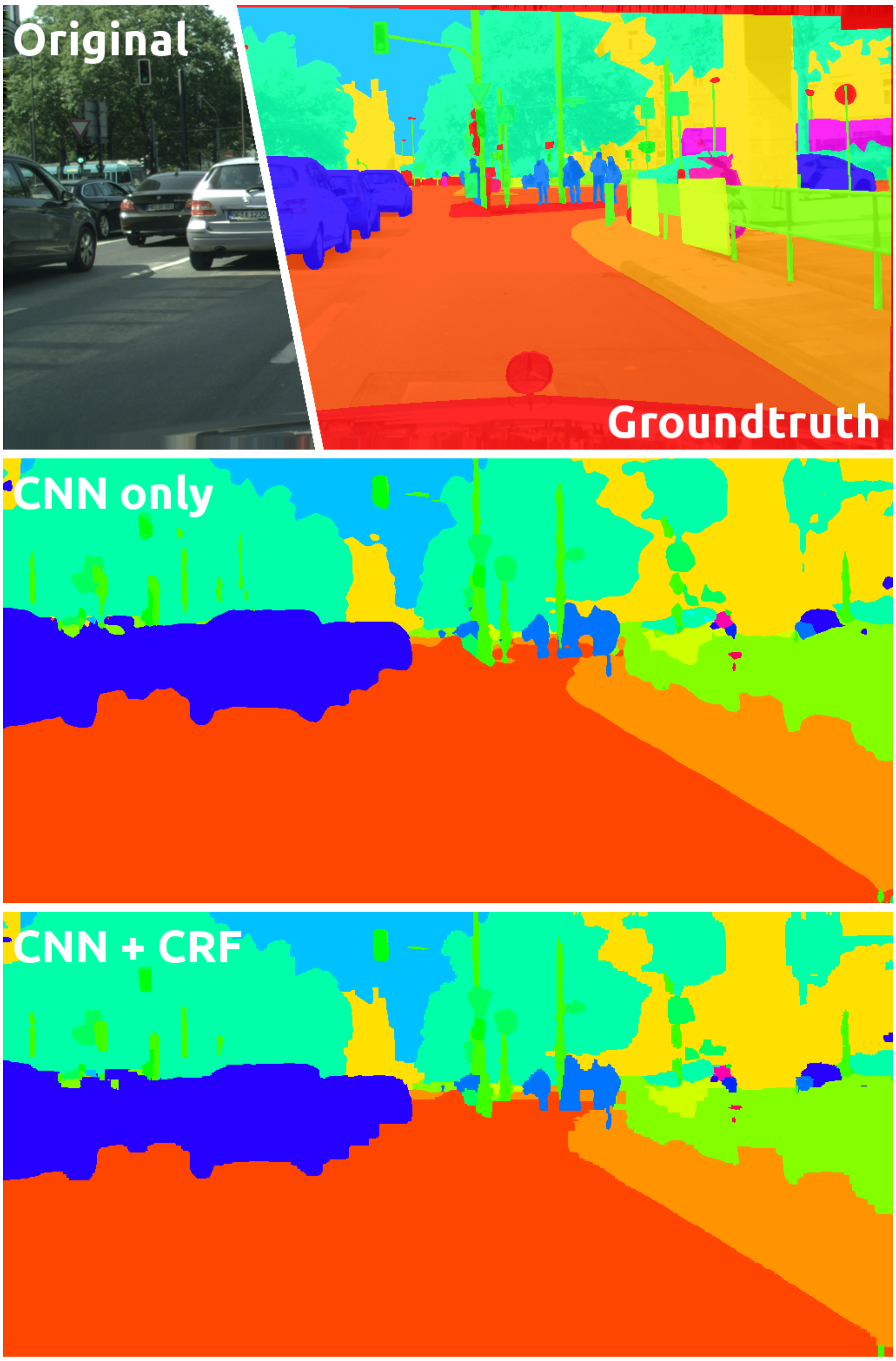


Figure A.3: Comparison of the CNN only model and the combination of CNN and CRF.

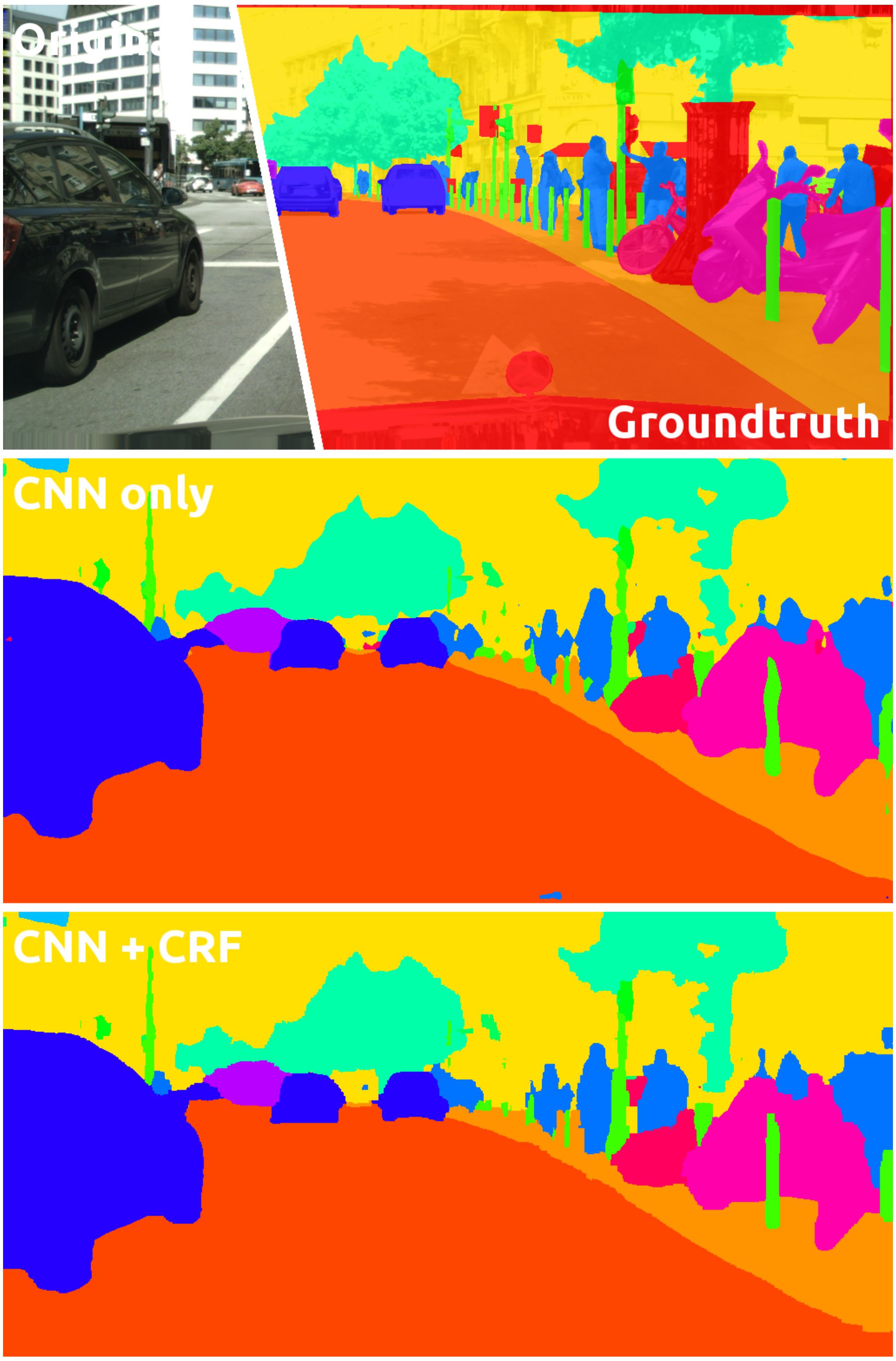


Figure A.4: Comparison of the CNN only model and the combination of CNN and CRF.

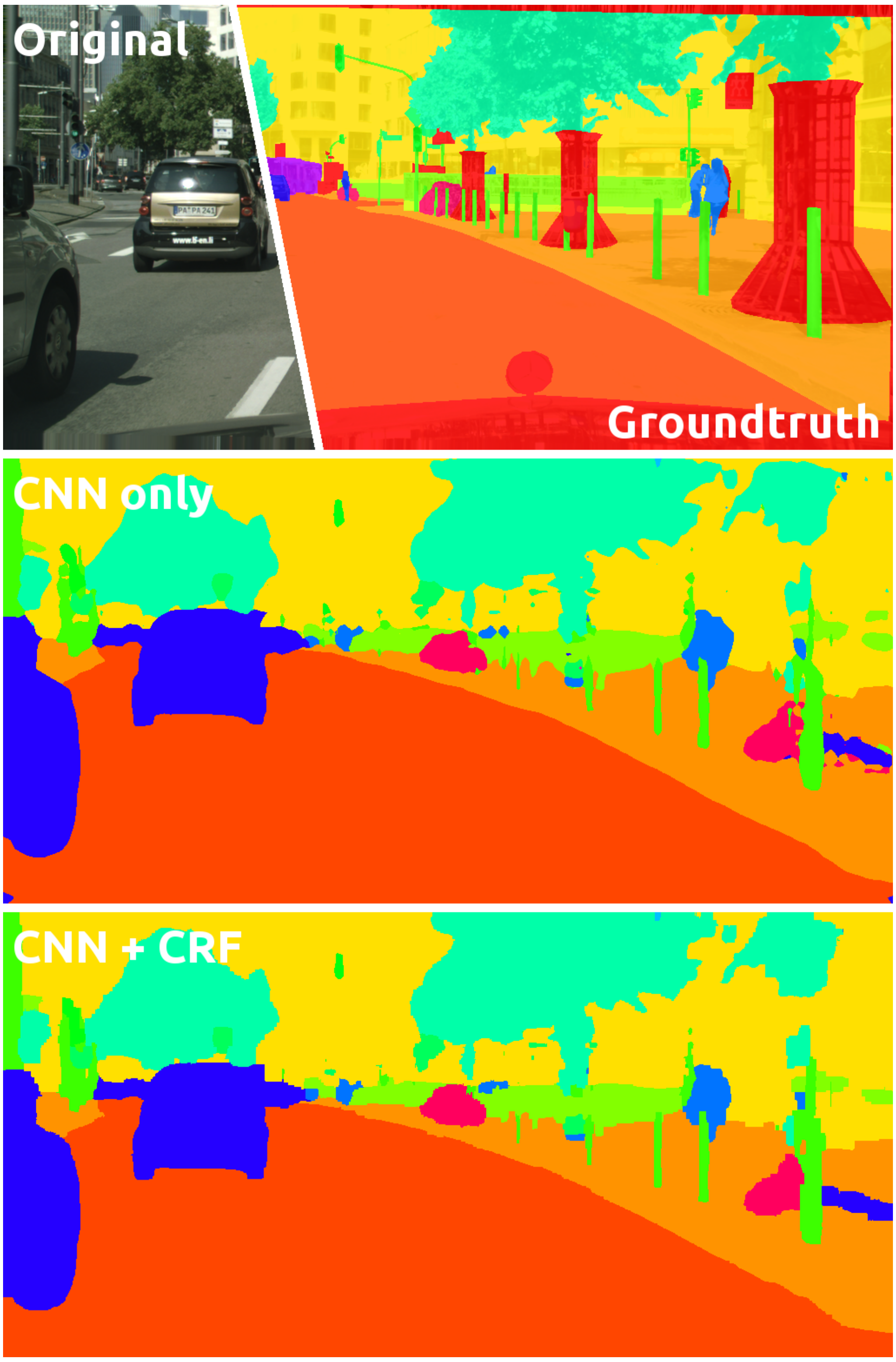


Figure A.5: Comparison of the CNN only model and the combination of CNN and CRF.

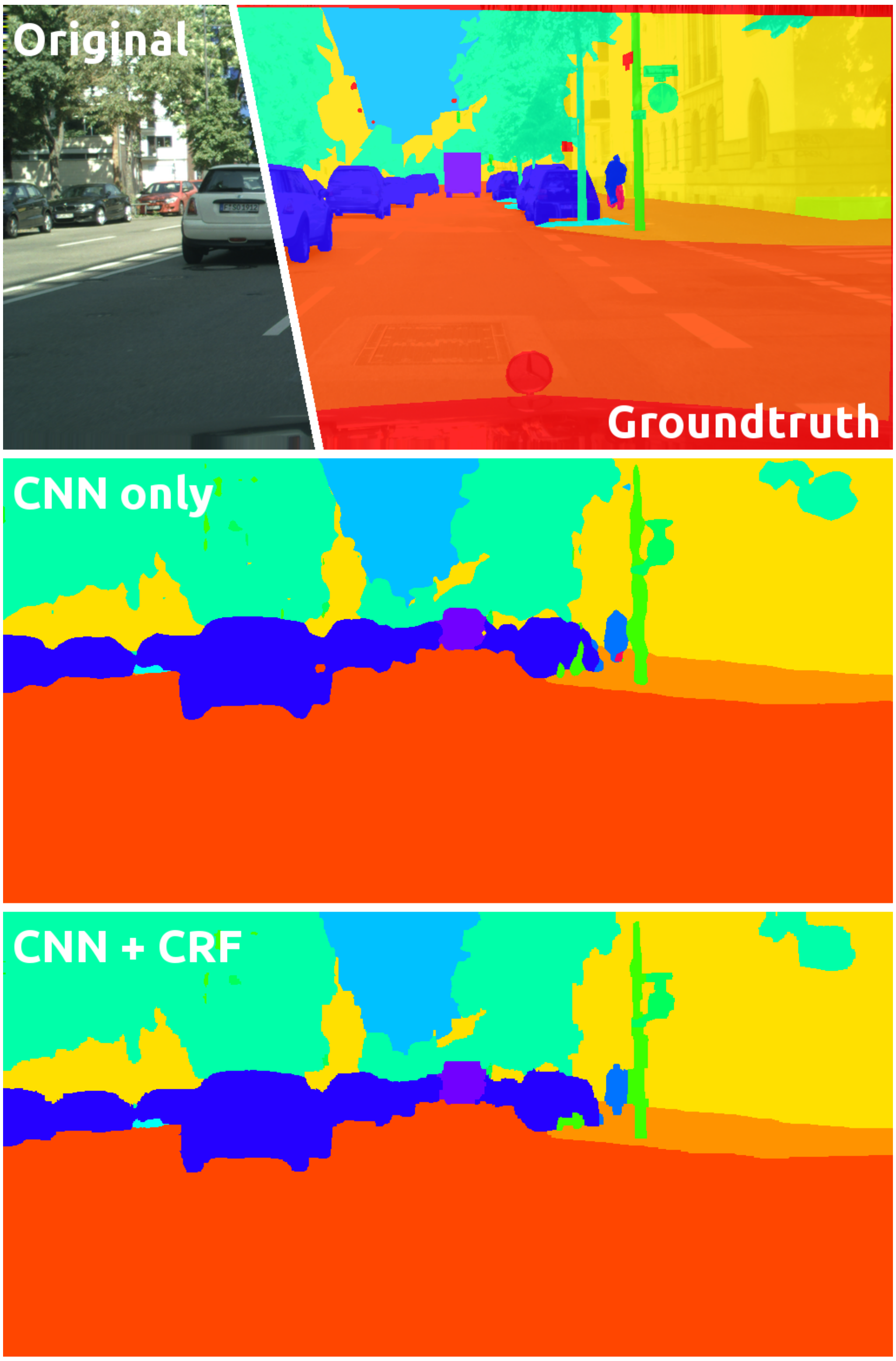


Figure A.6: Comparison of the CNN only model and the combination of CNN and CRF.

B List of Publications

My work at the Institute for Computer Graphics and Vision led to the following peer-reviewed publication. For the sake of completeness of this Thesis, it is noted here along with its abstract.

B.1 End-to-End Training of Hybrid CNN-CRF Models for Semantic Segmentation using Structured Learning

Aleksander Colovic, Patrick Knöbelreiter, Alexander Shekhovtsov and Thomas Pock
In: *Proceedings of Computer Vision Winter Workshop*
February 2017, Retz, Austria
(Accepted for oral presentation)

Abstract. In this work, we tackle the problem of semantic image segmentation with a combination of convolutional neural networks (CNNs) and conditional random fields (CRFs). The CRF takes contrast sensitive weights in a local neighborhood as input (pairwise interactions) to encourage consistency (smoothness) within the prediction and align our segmentation boundaries with visual edges. We model unary terms with a CNN which outperforms non-data driven models. We approximate the CRF inference with a fixed number of iterations of a linear-programming relaxation based approach. We experiment with training the combined model end-to-end using a discriminative formulation (structured support vector machine) and applying stochastic subgradient descend to it.

Our proposed model achieves an intersection over union score of 62.4 in the test set of the cityscapes pixel-level semantic labeling task which is comparable to state-of-the-art models.

Bibliography

[1] Abdi, M. and Nahavandi, S. (2016). Multi-residual networks: Improving the speed and accuracy of residual networks. *arXiv preprint arXiv:1609.05672.* (page 9)

[2] Adams, A., Baek, J., and Davis, M. A. (2010). Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library. (page 9)

[3] Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., et al. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 77–82. IEEE. (page 1)

[4] Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE. (page 40)

[5] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915.* (page 7, 8, 9, 40, 50)

[6] Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220. (page 1)

[7] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289.* (page 14)

[8] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (page 30, 45, 48, 50, 53, 57)

[9] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314. (page 13, 15)

[10] Damelin, S. B. and Miller Jr, W. (2012). *The mathematics of signal processing.* Number 48. Cambridge University Press. (page 16)

[11] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941. (page 27)

[12] De la Escalera, A., Armingol, J. M., and Mata, M. (2003). Traffic sign recognition and analysis for intelligent vehicles. *Image and vision computing*, 21(3):247–258. (page 1)

[13] Doan, A., Ramakrishnan, R., and Halevy, A. Y. (2011). Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96. (page 43)

[14] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015a). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136. (page 51)

[15] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015b). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136. (page 49)

[16] Gardner, W. A. (1984). Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique. *Signal processing*, 6(2):113–133. (page 44)

[17] Ghiasi, G. and Fowlkes, C. (2016). Laplacian reconstruction and refinement for semantic segmentation. *arXiv preprint arXiv:1605.02264*. (page 8, 18, 50)

[18] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158. (page 7)

[19] Giusti, A., Cireşan, D. C., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2013). Fast image scanning with deep max-pooling convolutional neural networks. *arXiv preprint arXiv:1302.1700*. (page 8)

[20] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. (page 23, 25, 26)

[21] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. (page 14)

[22] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. (page 8, 9)

[23] Herculano-Houzel, S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in human neuroscience*, 3:31. (page 11)

[24] Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer. (page 8)

[25] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257. (page 13)

[26] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366. (page 15)

[27] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154. (page 16)

[28] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*. (page 49)

[29] Kelley, K. and Maxwell, S. E. (2003). Sample size for multiple regression: obtaining regression coefficients that are accurate, not simply significant. *Psychological methods*, 8(3):305. (page 28)

[30] Knöbelreiter, P., Reinbacher, C., Shekhovtsov, A., and Pock, T. (2016). End-to-end training of hybrid cnn-crf models for stereo. *arXiv preprint arXiv:1611.10229*. (page 7)

[31] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press. (page 9)

[32] Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst.* (page 9, 10, 54)

[33] Krešo, I., Čaušević, D., Krapac, J., and Šegvić, S. (2016). Convolutional scale invariance for semantic segmentation. In *German Conference on Pattern Recognition*, pages 64–75. Springer. (page 50)

[34] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. (page 1, 2, 7, 18)

[35] Krogh, A. and Hertz, J. A. (1991). A simple weight decay can improve generalization. In *NIPS*, volume 4, pages 950–957. (page 28)

[36] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86. (page 23)

[37] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289. (page 9, 31)

[38] Liao, Q. and Poggio, T. (2016). Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640.* (page 9)

[39] Lin, G., Shen, C., Reid, I., et al. (2015). Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013.* (page 7, 8, 10, 50)

[40] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., and Reed, S. (2015). Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325.* (page 7)

[41] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440. (page 7, 8, 18, 19, 37, 38, 44, 50, 54)

[42] Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30. (page 13, 14)

[43] Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., and Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (page 49)

[44] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814. (page 13, 14)

[45] Papandreou, G., Chen, L.-C., Murphy, K. P., and Yuille, A. L. (2015a). Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1742–1750. (page 8, 50)

[46] Papandreou, G., Kokkinos, I., and Savalle, P.-A. (2015b). Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 390–399. (page 8)

[47] Plaut, D. C. et al. (1986). Experiments on learning by back propagation. (page 26, 27)

[48] Potts, R. B. (1952). Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, pages 106–109. Cambridge Univ Press. (page 39)

[49] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99. (page 7)

[50] Roller, B. T. C. G. D. (2004). Max-margin markov networks. *Advances in neural information processing systems*, 16:25. (page 10, 33)

[51] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1. (page 22, 25)

[52] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252. (page 43)

[53] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*. (page 8)

[54] Sermanet, P. and LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2809–2813. IEEE. (page 1)

[55] Shekhovtsov, A., Reinbacher, C., Graber, G., and Pock, T. (2016). Solving dense image matching in real-time using discrete-continuous optimization. *arXiv preprint arXiv:1601.06274*. (page 9, 40, 41, 54)

[56] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. (page 7, 8, 37, 43, 44)

[57] Smith, L. N. and Topin, N. (2016). Deep convolutional neural network design patterns. *arXiv preprint arXiv:1611.00847*. (page 9)

[58] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*. (page 18)

[59] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958. (page 28, 29)

[60] Sutton, C. and McCallum, A. (2012). Piecewise training for undirected models. *arXiv preprint arXiv:1207.1409*. (page 10)

[61] Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688. (page 49)

[62] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484. (page 10, 33)

[63] Uhrig, J., Cordts, M., Franke, U., and Brox, T. (2016). Pixel-level encoding and depth layering for instance-level semantic labeling. *arXiv preprint arXiv:1604.05096*. (page 8, 50)

[64] Veit, A., Wilber, M. J., and Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pages 550–558. (page 9)

[65] Werbos, P. (1988). Backpropagation: Past and future. Proceeding of International Conference Neural Networks (ICANN). (page 28)

[66] Wu, Z., Shen, C., and Hengel, A. v. d. (2016). Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*. (page 9)

[67] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2016). Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*. (page 9)

[68] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*. (page 8, 50)

[69] Zhang, X., Li, Z., Loy, C. C., and Lin, D. (2016). Polynet: A pursuit of structural diversity in very deep networks. *arXiv preprint arXiv:1611.05725*. (page 9)

[70] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537. (page 7, 8, 10, 50, 54)