Joint Training of Generic CNN-CRF Models with Stochastic Optimization

A. Kirillov[1], D. Schlesinger[1], S. Zheng[2], B. Savchynskyy[1], P.H.S. Torr[2], C. Rother[1]

[1]Dresden University of Technology, [2]University of Oxford

Abstract. We propose a new CNN-CRF end-to-end learning framework, which is based on joint stochastic optimization with respect to both Convolutional Neural Network (CNN) and Conditional Random Field (CRF) parameters. While stochastic gradient descent is a standard technique for CNN training, it was not used for joint models so far. We show that our learning method is (i) general, i.e. it applies to arbitrary CNN and CRF architectures and potential functions; (ii) scalable, i.e. it has a low memory footprint and straightforwardly parallelizes on GPUs; (iii) easy in implementation. Additionally, the unified CNN-CRF optimization approach simplifies a potential hardware implementation. We empirically evaluate our method on the task of semantic labeling of body parts in depth images and show that it compares favorably to competing techniques.

1 Introduction

Deep learning have tremendous success since a few years in many areas of computational science. In computer vision, Convolutional Neural Networks (CNNs) are successfully used in a wide range of applications – from low-level vision, like segmentation and optical flow, to high-level vision, like scene understanding and semantic segmentation. For instance in the VOC2012 object segmentation challenge[1] the use of CNNs has pushed the quality score by around 28% (from around 50% to currently around 78% [1]). The main contribution of CNNs is their ability to adaptively fine-tune millions of features to achieve best performance for the task at hand. However, CNNs have also their shortcomings. One limitation is that often a large corpus of labeled training images is necessary. Secondly, it is difficult to incorporate prior knowledge into the CNN architecture. In contrast, graphical models like Conditional Random Fields (CRFs) [2] overcome these two limitations. CRFs have been used to model geometric properties, such as object shape, spatial relationship between objects, global properties like object connectivity, and many others. Furthermore, CRFs designed based on e.g. physical properties are able to achieve good results even with few training images. For these reasons, a recent trend has been to explore the combination of these two modeling paradigms by using a CRF, whose factors are dependent

[1] http://host.robots.ox.ac.uk:8080/leaderboard

on a CNN. By doing so, CRFs are able to use the incredible power of CNNs, to fine-tune model features. On the other hand, CNNs can more easily capture global properties such as object shape and contextual information. The study of this fruitful combination (sometimes called "deep structured models" [3]) is the main focus of our work. We propose a generic joint learning framework for the combined CNN-CRF models, based on a sampling technique and a stochastic gradient optimization.

Related work. The idea of making CRF models more powerful by allowing factors to depend on many parameters has been explored extensively over the last decade. One example is the Decision Tree Field approach [4] where factors are dependent on Decision Trees. In this work, we are interested in making the factors dependent on CNNs. Note that one advantage of CNNs over Decision Trees is that CNNs learn the appropriate features for the task at hand, while Decision Trees, as many other classifiers, only combine and select from a pool of simple features, see e.g. [5, 6] for a discussion on the relationship between CNNs and Decision Trees. We now describe the most relevant works that combine CNNs and CRFs in the context of semantic segmentation, as one of the largest application areas of this type of models. The framework we propose in this work is also evaluated in a similar scenario, although its theoretical basis is application-independent.

Since CNNs have been used for semantic segmentation, this field has made a big leap forward, see e.g. [7, 8]. Recently, the advantages of additionally integrating a CRF model have given a further boost in performance, as demonstrated by many works. To the extent that the work [1] is currently leading the VOC2012 object segmentation challenge, as discussed below. In [9] a fully connected Gaussian CRF model [10] was used, where the respective unaries were supplied by a CNN. The CRF inference was done with a Mean Field approximation. This separate training procedure was recently improved in [11] with an end-to-end learning algorithm. To achieve this, they represent the Mean Field iterations as a Recurrent Neural Network. The same idea was published in [12]. In [10], the Mean Field iterations were made efficient by using a so-called permutohedral lattice approximation [13] for Gaussian filters. However, this approach allows for a special class of pairwise potentials only. Besides the approaches [11] and [12], there are many other works that consider the idea of backpropagation with a so-called unrolled CRF-inference scheme, such as [14–20]. These inference steps mostly correspond to message passing operations of e.g. Mean Field updates or Belief Propagation. However the number of inference iterations in such learning schemes remains their critical parameter: too few iterations lead to a quality deterioration, whereas more iterations slow down the whole learning procedure.

Likelihood maximization is NP-hard for CRFs, which implies that it is also NP-hard for joint CNN-CRF models. To avoid this problem, *piece-wise* learning [21] was used in [1]. Instead of likelihood maximization a surrogate loss is considered which can be minimized efficiently. However, there are no guarantees that minimization of the surrogate loss will lead to maximization of the true

likelihood. On the positive side, the method shows good practical results and leads the VOC2012 object segmentation competition at the moment.

Another likelihood approximation, which is based on fractional entropy and a message passing based inference, was proposed in [3]. However, there is no clear evidence that the fractional entropy always leads to tight likelihood approximations. Another point relates to the memory footprint of the method. To avoid the time consuming, full inference, authors of [3] interleave gradient steps w.r.t. the CNN parameters and minimization over the dual variables of the LP-relaxation of the CRF. This allows to solve the issue with a small number of inference iterations comparing to the unrolled inference schemes. However, it requires to store current values of the dual variables for *each* element of a training set. The number of the dual variables is proportional to the number of labels in the used CRF as well as to the number of its pairwise factors. Therefore, the size of such a storage can significantly exceed the size required for the training set itself. We will discuss this point in more details in Section 4.

Contribution. Inspired by the contrastive divergence approach [22], we propose a *generic joint maximum likelihood learning framework* for the combined CNN-CRF models. In this context, "*generic*" means that (i) factors in our CRF are of a non-parametric form, in contrast to e.g. [11], where Gaussian pairwise potentials are considered; and (b) we maximize the likelihood itself instead of its approximations. Our framework is based on a sampling technique and stochastic gradient updates w.r.t. both CNN and CRF parameters. To avoid the time consuming, full inference we interleave sampling-based inference steps with CNN parameters updates. In terms of the memory overhead, our method stores only a single (current) labeling for each element of the training set during learning. This requires less memory than the training set itself. Our method is efficient, scalable and highly parallelizable with a low memory footprint, which makes it an ideal candidate for a GPU-based implementation.

We show the efficiency of our approach on the task of semantic labeling of body parts in depth images.

2 Preliminaries

Conditional Random Fields. Let $\boldsymbol{y} = (y_1, \dots, y_N)$ be a random *state* vector, where each coordinate is a random variable y_i that takes its values from a finite set $\mathcal{Y}_i = \{1, \dots, |\mathcal{Y}_i|\}$. Therefore $\boldsymbol{y} \in \mathcal{Y} := \prod_{i=1}^{N} \mathcal{Y}_i$, where $\prod$ stands for a Cartesian product. Let $\boldsymbol{x}$ be *an observation* vector, taking its values in some set $\mathcal{X}$. The *energy function* $E \colon \mathcal{Y} \times \mathcal{X} \times \mathbb{R}^m \to \mathbb{R}$ assigns a score $E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta})$ to a pair $(\boldsymbol{y}, \boldsymbol{x})$ of a state and an observation vector and is parametrized by a *parameter* vector $\boldsymbol{\theta} \in \mathbb{R}^m$. An exponential posterior distribution related to the energy E reads

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{x}, \boldsymbol{\theta})} \exp(-E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}))\,. \tag{1}$$

Here $Z(\boldsymbol{x}, \boldsymbol{\theta})$ is a partition function, defined as

$$Z(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp(-E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}))\,. \tag{2}$$

Let $I = 1, ..., N$ be a set of *variable indexes* and 2^I denote its powerset. Let also $\mathcal{Y}_A$ stand for the set $\prod_{i \in A} \mathcal{Y}_i$ for any $A \subseteq I$. In CRFs, the energy function E can be represented as a sum of its components depending on the subsets of variables $\boldsymbol{y}_f \in \mathcal{Y}_f,\ f \subset I$:

$$E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \sum_{f \in \mathcal{F} \subset 2^I} \psi_f(\boldsymbol{y}_f, \boldsymbol{x}, \boldsymbol{\theta})\,. \tag{3}$$

The functions $\psi_f \colon \mathcal{Y}_f \times \mathcal{X} \times \mathbb{R}^m \to \mathbb{R}$ are usually called *potentials*. For example, in [9, 11] only CRFs with *unary* and *pairwise* potentials are considered, i.e. $|f| \leq 2$ for any $f \in \mathcal{F}$.

In what follows, we will assume that each ψ_f is potentially a non-linear function of $\boldsymbol{\theta}$ and $\boldsymbol{x}$. It can be defined by e.g. a CNN with the input $\boldsymbol{x}$ and weights $\boldsymbol{\theta}$.

Inference is a process of estimating the state vector $\boldsymbol{y}$ for an observation $\boldsymbol{x}$. There are several inference criteria, see e.g. [23]. In this work we will stick to the so called *maximum posterior marginals*, or shortly *max-marginal* inference

$$y_i^* = \arg\max_{y_i \in \mathcal{Y}_i} p(y_i|\boldsymbol{x}, \boldsymbol{\theta}) := \arg\max_{y_i \in \mathcal{Y}_i} \sum_{(\boldsymbol{y'} \in \mathcal{Y}\colon\, y'_i = y_i)} p(\boldsymbol{y'}|\boldsymbol{x}, \boldsymbol{\theta}) \qquad \text{for all } i\,. \tag{4}$$

Though maximization in (4) can be done directly due to the typically small size of the sets $\mathcal{Y}_i$, computing the marginals $p(y_i|\boldsymbol{x}, \boldsymbol{\theta})$ is NP-hard in general. Summation in (4) can not be performed directly due to the exponential size of the set $\mathcal{Y}$. In our framework we approximate the marginals with Gibbs sampling [24]. The corresponding estimates converge to the true marginals in the limit. We detail this procedure in Section 3.

Learning. Given a training set $\{(\boldsymbol{x}^d, \boldsymbol{y}^d) \in (\mathcal{X} \times \mathcal{Y})\}_{d=1}^D$, we consider the maximum likelihood learning criterion for estimating $\boldsymbol{\theta}$:

$$\arg\max_{\boldsymbol{\theta} \in \mathbb{R}^m} \sum_{d=1}^{D} \log p(\boldsymbol{y}^d|\boldsymbol{x}^d, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta} \in \mathbb{R}^m} \sum_{d=1}^{D} \left[-E(\boldsymbol{y}^d, \boldsymbol{x}^d, \boldsymbol{\theta}) - \log Z(\boldsymbol{x}^d, \boldsymbol{\theta})\right]\,. \tag{5}$$

Since a (stochastic) gradient descent is used for CNN training, we stick to it for estimating (5) as well. The gradient of the objective reads:

$$\begin{aligned}
\frac{\partial \sum_{d=1}^{D} \log p(\boldsymbol{y}^d|\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \sum_{d=1}^{D} \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial \log Z(\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right] \\
&= \sum_{d=1}^{D} \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{1}{Z(\boldsymbol{x}^d,\boldsymbol{\theta})} \frac{\partial \sum_{\boldsymbol{y}\in\mathcal{Y}} \exp(-E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} \right] \\
&= \sum_{d=1}^{D} \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{y}\in\mathcal{Y}} \frac{\exp(-E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta}))}{Z(\boldsymbol{x}^d,\boldsymbol{\theta})} \frac{\partial E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right] \\
&= \sum_{d=1}^{D} \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta}) \frac{\partial E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right] \\
&= \sum_{d=1}^{D} \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta})} \frac{\partial E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right] . \qquad (6)
\end{aligned}$$

Direct computation of the gradient (6) is infeasible due to an exponential number of possible variable configurations $\boldsymbol{y}$, which must be considered to compute $\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta})} \frac{\partial E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$. Inspired by [22], in our work we employ sampling based approximation of (6) instead, which we detail in Section 3.

Stochastic Approximation. The stochastic gradient approximation proposed in [25] is a common way to learn parameters of a CNN nowadays. It allows to perform parameter updates for a single randomly selected input observation, or a small subset of observations, instead of computing the update step for the whole training set at once, as the latter can be very costly. Assume that the gradient of some function $f(\theta)$ can be represented as follows:

$$\nabla_\theta f = \mathbb{E}_{p(y|\theta)} \nabla_\theta g(y,\theta) . \qquad (7)$$

Then under mild technical conditions the following procedure

$$\theta_{i+1} = \theta_i - \eta_i \nabla_\theta g(y',\theta_i), \text{ where } y' \sim p(y|\theta_i) \qquad (8)$$

and η_i is a diminishing sequence of step-sizes, converges to a critical point of the function $f(\theta)$. We refer to [25, 26] for details, for the cases of both convex and non-convex functions $f(\theta)$.

3 Stochastic Optimization Based Learning Framework

Stochastic Likelihood Maximization. Since the value $\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ does not depend on y, we can rewrite the gradient (6) as

$$\frac{\partial \sum_{d=1}^{D} \log p(\boldsymbol{y}^d|\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{d=1}^{D} \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta})} \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right] . \qquad (9)$$

The summation over samples from the training set can be seen as an expectation over a uniform distribution and therefore the index d can be seen as drawn from this uniform distribution. According to this observation we can rewrite (9) as

$$\frac{\partial \sum_{d=1}^{D} \log p(\boldsymbol{y}^d|\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = D \cdot \mathbb{E}_{p(\boldsymbol{y},d|\boldsymbol{x}^d,\boldsymbol{\theta})} \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial E(\boldsymbol{y},\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right], \quad (10)$$

where $p(\boldsymbol{y},d|\boldsymbol{x}^d,\boldsymbol{\theta}) = p(d)p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta})$ and $p(d) = \frac{1}{D}$. Assume that we can obtain i.i.d. samples $\boldsymbol{y}'$ from $p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta})$. Then the following iterative procedure converges to a critical point of the likelihood (5) according to (7) and (8)

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta_i \left[-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta_i})}{\partial \boldsymbol{\theta}} + \frac{\partial E(\boldsymbol{y}',\boldsymbol{x}^d,\boldsymbol{\theta_i})}{\partial \boldsymbol{\theta}} \right], \quad (11)$$

where d is uniformly sampled from $\{1,\dots,D\}$ and $\boldsymbol{y}' \sim p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta}_i)$.

Now we turn to the computation of the stochastic gradient $-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial E(\boldsymbol{y}',\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ itself, provided $\boldsymbol{y}^d, \boldsymbol{y}', \boldsymbol{x}^d$ and $\boldsymbol{\theta}$ are given. In the *overcomplete representation* [23] the energy (3) reads

$$E(\boldsymbol{y},\boldsymbol{x},\theta) = \sum_{f\in\mathcal{F}} \sum_{\hat{\boldsymbol{y}}_f \in \mathcal{Y}_f} \psi_f(\hat{\boldsymbol{y}}_f,\boldsymbol{x},\boldsymbol{\theta}) \cdot [\![\boldsymbol{y}_f = \hat{\boldsymbol{y}}_f]\!], \quad (12)$$

where expression $[\![A]\!]$ equals 1 if A is true and 0 otherwise. Therefore $\frac{\partial E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta})}{\partial \psi_f(\hat{\boldsymbol{y}}_f,\boldsymbol{x},\boldsymbol{\theta})} = [\![\boldsymbol{y}_f = \hat{\boldsymbol{y}}_f]\!]$. If the potential $\psi_f(\hat{\boldsymbol{y}}_f,\boldsymbol{x},\boldsymbol{\theta})$ is an output of a CNN, then the value $-\frac{\partial E(\boldsymbol{y}^d,\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \psi_f(\hat{\boldsymbol{y}}_f,\boldsymbol{x}^d,\boldsymbol{\theta})} + \frac{\partial E(\boldsymbol{y}',\boldsymbol{x}^d,\boldsymbol{\theta})}{\partial \psi_f(\hat{\boldsymbol{y}}_f,\boldsymbol{x}^d,\boldsymbol{\theta})} = -[\![\boldsymbol{y}^d_f = \hat{\boldsymbol{y}}_f]\!] + [\![\boldsymbol{y}'_f = \hat{\boldsymbol{y}}_f]\!]$ is the error to propagate to the CNN. During the back-propagation of this error all parameters $\boldsymbol{\theta}$ of the CNN are updated. The overall stochastic maximization procedure for the likelihood (5) is summarized in Algorithm 1. The algorithm is fully defined up to sampling from the distribution $p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta})$ in Step 5. We discuss different approaches in the next subsection.

Algorithm 1 Sampling-based maximization of the likelihood (5)

1: Initialize parameters $\boldsymbol{\theta}_0$ of the CNN-CRF model.
2: **for** $i = 1$ to M (*max. number of iterations*) **do**
3: Uniformly sample d from $\{1,\dots,D\}$
4: Perform forward pass of the CNN to get $\psi_f(\hat{\boldsymbol{y}},\boldsymbol{x}^d,\boldsymbol{\theta}_{i-1})$ for each $f \in \mathcal{F}$ and $\hat{\boldsymbol{y}}_f \in \mathcal{Y}_f$
5: Sample $\boldsymbol{y}'$ from the distribution $p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta}_{i-1})$ defined by (1)
6: Compute the error $-[\![\boldsymbol{y}^d_f = \hat{\boldsymbol{y}}_f]\!] + [\![\boldsymbol{y}'_f = \hat{\boldsymbol{y}}_f]\!]$ for each $f \in \mathcal{F}$ and $\hat{\boldsymbol{y}}_f \in \mathcal{Y}_f$
7: Back propagate the error through CNN to obtain a gradient $\nabla_{\boldsymbol{\theta}}$
8: Update the parameters $\boldsymbol{\theta}_i := \boldsymbol{\theta}_{i-1} - \eta_i \nabla_{\boldsymbol{\theta}}$
9: **return** $\boldsymbol{\theta}_M$

Sampling. Obtaining an exact sample from $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$ is a difficult problem for a general CRF due to the exponential size of the set $\mathcal{Y} \ni \boldsymbol{y}$ of all possible configurations. There are, however, ways to mitigate it. The full Markov Chain Monte Carlo (MCMC) sampling method [27] starts from an arbitrary variable configuration $\boldsymbol{y} \in \mathcal{Y}$ and generates the next one $\boldsymbol{y}'$. In our case this generation can be done with e.g. Gibbs sampling [24], as presented in Algorithm 2. Algorithm 2 passes over all variables y_n and updates each of them according to the conditional distribution $p(y_n|\boldsymbol{y}_{\setminus n},\boldsymbol{x},\boldsymbol{\theta})$, where $\setminus n$ denotes all variable indexes except n. Let $\mathrm{nb}(n) = \{k \in I | \exists f \in \mathcal{F}\colon n,k \in f\}$ denote all neighbors of the variable n. Note, that due to the Markov property of CRFs [28], it holds

$$p(y_n|\boldsymbol{y}_{\setminus n},\boldsymbol{x},\boldsymbol{\theta}) = p(y_n|\boldsymbol{y}_{\mathrm{nb}(n)},\boldsymbol{x},\boldsymbol{\theta}) \propto \exp\left(-\sum_{f\in\mathcal{F}:n\in f} \psi_f(\boldsymbol{y}_f,\boldsymbol{x},\boldsymbol{\theta})\right). \quad (13)$$

Therefore, sampling from this distribution can be done efficiently, since it requires evaluating only those potentials $\psi_f(\boldsymbol{y}_f,\boldsymbol{x},\boldsymbol{\theta})$ which are dependent on the variable y_n, i.e. for $f \in \mathcal{F}$ such that $n \in f$. Algorithm 2 summarizes one iteration of the sampling procedure. Note that it is highly parallelizable [29] and allows for efficient GPU implementations. Under mild technical conditions the MCMC sampling process converges to a stationary distribution after a finite number of iterations [27]. This distribution coincides with $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$. However, such a sampling is time-consuming, because convergence to the stationary distribution may require many iterations and must be performed after each update of the parameters $\boldsymbol{\theta}$.

To overcome this difficulty a contrastive-divergence (CD) method was proposed in [30] and theoretically justified in [31]. For a randomly generated index $d \in \{1,\dots,D\}$ of the training sample one performs a single step of the MCMC procedure starting from a ground-truth variable configuration, which in our case boils down to a single run of Algorithm 2 for $\boldsymbol{y} = \boldsymbol{y}^d$. Unfortunately, the sufficient conditions needed to justify this method according to [31] do not hold for CRFs in general. Nevertheless, we provide an experimental evaluation of this method in Section 5 along with a different technique described next.

Persistent contrastive divergence (PCD) [32] is a further development of contrastive divergence, where one step of the MCMC method is performed starting from the sample obtained on a previous learning iteration. It is based on the assumption that the distribution $p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta})$ changes slowly from iteration to iteration and a sample from $p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta}_{i-1})$ is close enough to a sample from $p(\boldsymbol{y}|\boldsymbol{x}^d,\boldsymbol{\theta}_i)$. Moreover, when getting closer to a critical point, the gradient becomes smaller and therefore $p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta}_i)$ deviates less from $p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta}_{i-1})$. Therefore, close to a critical point the generated samples can be seen as samples from the stationary distribution of the full MCMC method, which coincides with the desired one $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$.

With the above description of the possible sampling procedures the whole joint CNN-CRF learning Algorithm 1 is well-defined.

Algorithm 2 Gibbs sampling

Require: A variable configuration $\boldsymbol{y} \in \mathcal{Y}$
1: **for** n = 1, ..., N **do**
2: y'_n is sampled from $p(y_n|\boldsymbol{y}_{\setminus n}, \boldsymbol{x}, \boldsymbol{\theta})$
3: $y_n \leftarrow y'_n$
4: **return** $\boldsymbol{y}$

4 Comparison to Alternative Approaches

Unrolled Inference. In contrast to the learning method with the unrolled inference proposed in [11] and [12], our approach is not limited to Gaussian pairwise potentials. In our training procedure the potentials $\phi_f(\boldsymbol{y}_f, \boldsymbol{x}, \boldsymbol{\theta})$ can have arbitrary form.

The piece-wise training method [1] is able to handle arbitrary potentials in CRFs. However, maximization of the likelihood (5) in that work is substituted with

$$(\arg)\max_{\boldsymbol{\theta}} \sum_{d=1}^{D} \sum_{f \in \mathcal{F}} \left[-\psi_f(\boldsymbol{y}_f^d, \boldsymbol{x}^d, \boldsymbol{\theta}) - \log \sum_{\boldsymbol{y}_f \in \mathcal{Y}_f} \exp(-\psi_f(\boldsymbol{y}_f, \boldsymbol{x}^d, \boldsymbol{\theta})) \right], \qquad (14)$$

which lacks a sound theoretical justification.

LP-relaxation and fractional entropy based approximation is employed in [3]. As mentioned above, there is no clear evidence that the fractional entropy always leads to tight likelihood approximations. Additionally, the method requires a lot of memory: to avoid the time consuming, full message passing based inference, the gradient steps w.r.t. the CNN parameters $\boldsymbol{\theta}$ and minimization over the dual variables of the LP-relaxation of the CRF are interleaved with each other. This requires to store current values of the dual variables for *each* training sample. The number of dual variables is proportional to the number of labels used in the CRF as well as to the number of its factors. So, for example in our experiments we use a dataset containing 2000 images of the approximate size 320×120. The corresponding CRF has 20 labels and around 10^6 pairwise factors (see Section 5 for details). The dual variables stored by the method [3] would require around 200MB per image and 0.4TB for the whole dataset. Note that our approach requires to store only the current variable configuration $\boldsymbol{y}$ for each of the D training samples, when used with the PCD sampling. Therefore, it requires only 78MB of working storage for the whole dataset. The difference between our method and the method proposed in [3] gets even more pronounced for larger problems and datasets, such as the augmented Pascal VOC dataset [33, 34] containing 10000 images with 500×300 pixels each.

5 Experiments

In the experimental evaluation we consider the problem of semantic body-parts segmentation from a single depth image [35]. We specify a CRF, which has unary potentials dependent on a CNN. We test different sampling options in Algorithm 1 and compare our approach with another CRF-CNN learning framework proposed in [11]. Additionally, we analyze the trained model, in order to understand whether it can capture an object shape and contextual information.

Dataset and evaluation. We apply our approach to the challenging task of predicting human body parts from a depth image. To the best of our knowledge, there is no publicly available dataset for this task that contains real depth images. For this reason, in [35], a set of synthetically rendered depth images along with the corresponding ground truth labelings were introduced (see examples in Fig. 1 (left column)). In total there are 19 different body part labels, and one additional label for the background. The dataset is split into 2000 images for training and 500 images for testing. As a quality measure, the authors use the averaged per-pixel accuracy for body parts labeling, excluding the background. This makes sense since the background can be easily identified from the depth map.

Our model. Following [4], in our experiments, we use a pixel-level CRF that is able to capture geometrical layout and context. The state vector $\boldsymbol{y}$ defines a per pixel labeling. Therefore the number N of coordinates in $\boldsymbol{y}$ is equal to the number of pixels in a depth image, which has dimensions varying around 130×330. For all $n \in \{1,\dots,N\}$ the label space is $\mathcal{Y}_n = \{1,\dots,20\}$. The observation $\boldsymbol{x}$ represents a depth image. Our CRF has the following energy function $E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta})$:

$$E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta}) = \sum_{n=1}^{N} \psi_n(y_n,\boldsymbol{x},\boldsymbol{\theta}) + \sum_{c\in\mathcal{C}} \sum_{(i,j)\in E_c} \psi_c(y_i,y_j,\boldsymbol{\theta})\,, \tag{15}$$

where $\psi_n(y_n,\boldsymbol{x},\boldsymbol{\theta})$ are unary potentials that depend on a CNN. Our CRF has $|\mathcal{C}|$ classes of pairwise potentials. All potentials of one class are represented by a learned value table, which they share. The neighborhood structure of the CRF is visualized in Fig. 2b. All pixels are connected to 64 neighbors, apart from those close to the image border.

The local distribution (13) used by the sampling Algorithm 2 takes the form:

$$p_i(y_i{=}l|x,y_{R\setminus i};\theta) \propto \exp\Big[-\psi_i(l) - \sum_c \big(\psi_c(l,y_{j'}) + \psi_c(y_{j''},l)\big)\Big]. \tag{16}$$

Note that according to our CRF architecture there are exactly two edges (apart from the nodes close to the image border) in each edge class c that are incident to a given node i. The corresponding neighboring nodes are denoted by j' and j'' in (16).

Table 1: CNN archutecture for body parts segmentation.

Layer	conv1	relu1	conv2	relu2	conv3	maxpool1	relu3	conv4	Softmax
Kernel size	41×41	-	17×17	-	11×11	3×3	-	5×5	-
Output channels	50	50	50	50	50	50	50	20	20

As mentioned above, the unary terms of our CRF model depend on the image via a CNN. Since most existing pre-trained CNNs [7, 36, 37] use RGB images as input, for the depth input we use our own fully convolutional architecture and train it from scratch. Moreover, since some body parts, such as hands, are relatively small, we use the architecture that does not reduce the resolution in intermediate layers. This allows us to capture fine details. All intermediate layers have 50 output channels and a stride of one. The final layer has 20 output channels that correspond to the output labels. The architecture of our CNN is summarized in Table 1. During training, we optimize the cross-entropy loss. The CNN is trained using stochastic gradient descent with the momentum 0.99 and with the batch size 1.[2]

In our experiments, we consider two learning scenarios: *separate* learning and *joint* (end-to-end) learning. In both cases we start the learning procedure from the same pre-trained CNN. For separate learning only the CRF parameters (pairwise potentials) are updated, whereas the CNN weights (unary potentials of the CRF) are kept fixed. In contrast, for joint (end-to-end) learning all parameters are updated. During the test-time inference we empirically observed that starting Gibbs sampling (Algorithm 2) from a random labeling can lead to extremely long runtimes. To speed-up the burn-in-phase, we use the marginal distribution of the CNN without CRF. This means that the first sample is drawn from the marginal distribution of the pre-trained CNN.

We also experiment with different sampling strategies during the training phase: we considered (i) the contrastive-divergence with K sampling iterations, denoted as `CD-K` for K equal to 1, 2, 5 and (ii) the persistent contrastive-divergence `PCD`.

Baselines. We compare our approach to the method of [35], which introduced this dataset. Their approach is based on a random forest model. Unfortunately, we were not able to compare to the recent work [38], which extends [35], and is also based on random forests. The reason is that in the work [38] its own evaluation measure is used, meaning that the accuracy of only a small subset of pixels is evaluated. This subset is chosen in such a way that each of the 20 classes is represented by the same number of pixels. We are concerned, however, that such small pixel subsets may introduce a bias. Furthermore, we did not have this subset at our disposal. Since our main aim is to evaluate CNN-based CRF models, we compare to the approach [11]. As described above, they incorporate a

[2] We use the commonly adopted terminology from the CNN literature for technical details, to allow reproducibility of our results.

Table 2: Average per-pixel accuracy for all foreground parts. *Separate* learning means that weights of the respective CNN were trained prior to CRF parameters. In contrast, *joint* training means that all weights were learned jointly, starting with a pre-trained CNN. We obverse that joint training is superior to separate training, and furthermore that the model of [11], which is based on a dense Gaussian CRF, is inferior to our generic CRF model.

Method	Learning	Accuracy
Online Random Forest [35]	-	$\approx 79.0\%$
CNN	-	84.47
CNN + CRF [11]	separate	86.55%
CNN + CRF [11]	joint	88.17%
CNN + CRF (ours) `PCD`	separate	87.62%
CNN + CRF (ours) `CD-1`	joint	88.17%
CNN + CRF (ours) `CD-2`	joint	88.15%
CNN + CRF (ours) `CD-5`	joint	88.23%
CNN + CRF (ours) `PCD`	joint	**89.01%**

densely connected Gaussian CRF model into the CNN as a Recurrent Neuronal Network of the corresponding Mean Field inference steps. This approach has recently been the state-of-the-art in the VOC2012 object segmentation challenge.

Results. Qualitative and quantitative results are shown in Fig. 1 and Table 2 respectively. Our method with joint learning is performing best. In particular, the persistent contrastive-divergence version shows the best results, which conforms to the observations made in other works [32]. The CNN-CRF approach of [11] is inferior to ours. Note that the accuracy difference of 1% can mean that e.g. a complete hand is incorrectly labeled. We attribute this to the fact that for this task the spatial layout of body parts is of particular importance. The underlying dense Gaussian CRF model of [11] is rotational invariant and cannot capture contextual information such as "the head has to be above the torso". Our approach is able to capture this, which we explain in detail in Fig. 2 and 3. We expect that even higher levels of accuracy can be achieved by exploring different network designs and learning strategies, which we leave for future work.

6 Discussion and Future Work

We have presented a generic CRF model where a CNN models unary factors. We have introduced an efficient and scalable maximum likelihood learning procedure to train all model parameters jointly. By doing so, we were able to train and test on large-size factor graphs. We have demonstrated a performance gain over competing techniques for semantic labeling of body parts. We have observed that our generic CRF model can capture the shape and context information of relating body parts.

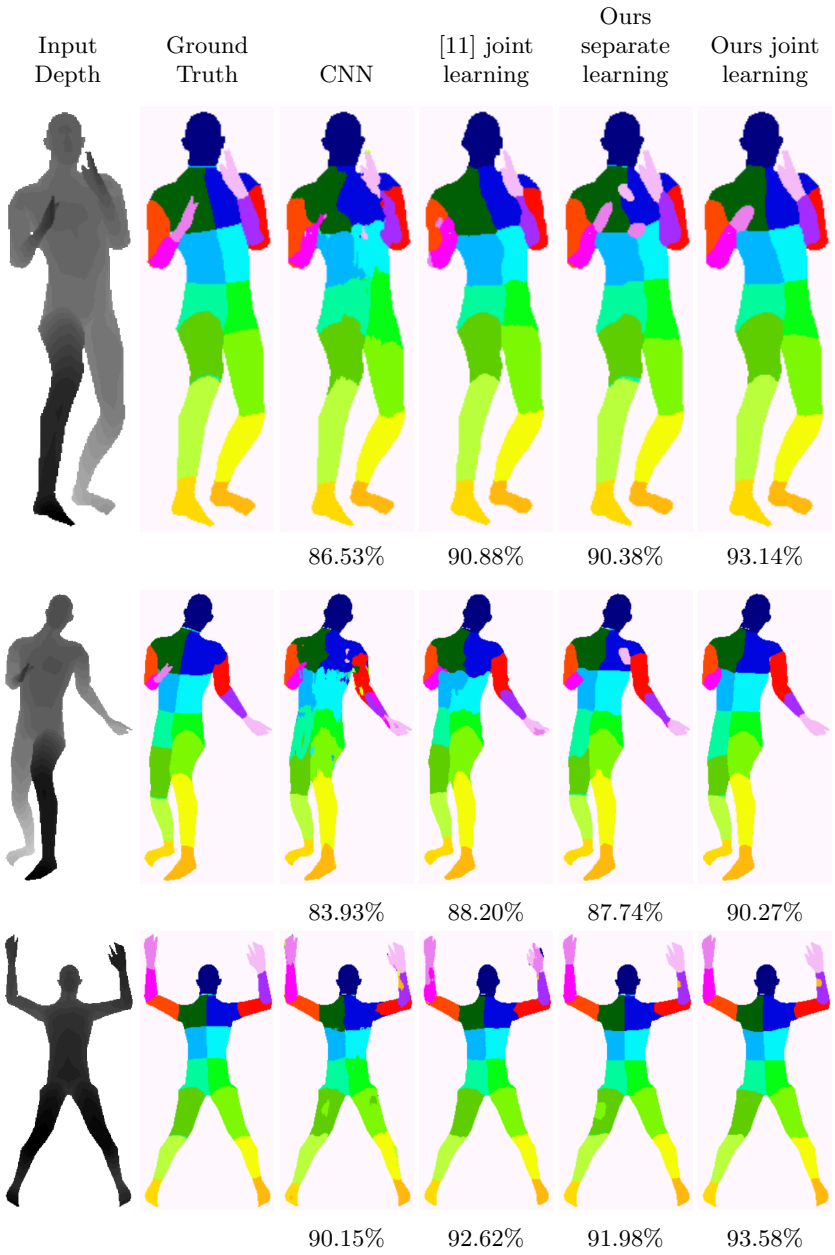


Fig. 1: **Results.** (From left to right). The input depth image. The corresponding ground truth labeling for all body parts. The result of a trained CNN model. The result of [11] using an end-to-end training procedure. Our results with separate learning and joint learning, respectively. Below each result we give the averaged pixel-wise accuracy for all body parts.

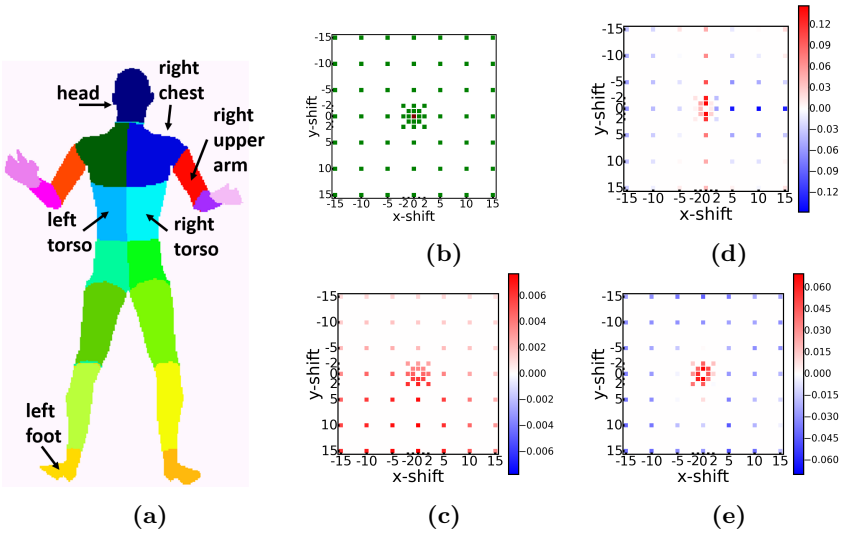
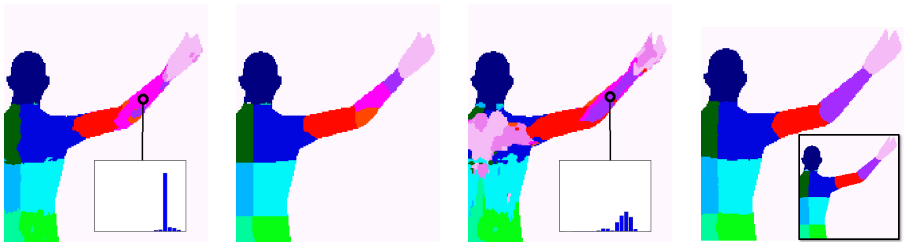


Fig. 2: **Model Insights.** (a) Illustrating the 19 body parts of a human. (c-e) Weights of pairwise factors for different pairs of labels, see details below. (b) Neighborhood structure for pairwise factors. The center pixel (red) is connected via pairwise factors to all green pixels. Note that "opposite" edges share same weights, e.g. the edge with x, y-shift $(5, 10)$ has the same weights as the edge with x, y-shift $(-5, -10)$. (c) Weights for pairwise potentials that connect the label "head" with the label "foot". Red means a high energy value, i.e. a discouraged configuration, while blue means the opposite. Since there is no sample in the training dataset where a foot is close to a head, all edges are positive or close to 0. Note that the zero weights can occur even for very unlikely configurations. The reason is that during training these unlikely configurations did not occur. (d) Weights for pairwise potentials that connect the label "left torso" with the label "right torso". The potentials enforce a straight, vertical border between the two labels, i.e. there is a large penalty for "left torso" on top (or below) of "right torso" (x-shift 0, y-shift arbitrary). Also, it is encouraged that "right torso" is to the right of the "left torso" (Positive x-shift and y-shift 0). (e) Weights for pairwise potentials that connect the label "right chest" with the label "right upper arm". It is discouraged that the "right upper arm" appears close to "right chest", but this configuration can occur at a certain distance. Since the training images have no preferred arm-chest configurations, all directions have similar weights.



(a) **(b)** **(c)** **(d)**

Fig. 3: **Model Insights.** (a) The most likely labeling for a separately trained CNN. For the circled pixel, the local marginal distribution is shown. (b) Max. marginal labeling of a separately trained CRF, which uses the CNN unaries from (a), i.e. our approach with separate learning. We observe that unaries are spatially smoothed-out. (c) Most likely labeling of a CNN that was jointly trained with the CRF. The labeling looks worse than (a). However, the main observation is that the pixel-wise marginal distributions are more ambiguous than in (a), see the circled pixel. (d) The final, max-marginal labeling of the jointly trained CRF model, which is considerably better than the result in (b). The reason is that due to the ambiguity in the local unary marginals, the CRF has more power to find the correct body part configuration. The inlet shows the ground truth labeling.

There are many exciting avenues for future research. We plan to apply our method to other application scenarios, such as semantic segmentation of RGB images. In this context, it would be interesting to combine the dense CRF model of [11] with our generic CRF model. Note that a dense CRF is implicitly modeling the property that objects have a compact color distribution, see [39], which is a complementary modeling power to our generic CRF model.

Acknowledgements. This work was supported by: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 647769); German Federal Ministry of Education and Research (BMBF, 01IS14014A-D); EPSRC EP/I001107/2; ERC grant ERC- 2012-AdG 321162-HELIOS. The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

The final publication is available at `link.springer.com`.

References

1. Lin, G., Shen, C., Reid, I.D., van den Hengel, A.: Efficient piecewise training of deep structured models for semantic segmentation. preprint arXiv:1504.01013 (2015)

2. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML. (2001)
3. Chen, L., Schwing, A.G., Yuille, A.L., Urtasun, R.: Learning deep structured models. In: ICML. (2015) 1785–1794
4. Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B., Kohli, P.: Decision tree fields. In: ICCV. (2011)
5. Sethi, I.K.: Entropy nets: from decision trees to neural networks. Proceedings of the IEEE **78** (1990) 1605–1613
6. Richmond, D.L., Kainmueller, D., Yang, M.Y., Myers, E.W., Rother, C.: Relating cascaded random forests to deep convolutional neural networks for semantic segmentation. preprint arXiv:1507.07583 (2015)
7. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. preprint arXiv:1411.4038 (2014)
8. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. TPAMI **35** (2013)
9. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. preprint arXiv:1412.7062 (2014)
10. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: NIPS. (2011)
11. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.S.: Conditional random fields as recurrent neural networks. In: ICCV. (2015)
12. Schwing, A.G., Urtasun, R.: Fully connected deep structured networks. preprint arXiv:1503.02351 (2015)
13. Adams, A., Baek, J., Davis, M.A.: Fast high-dimensional filtering using the permutohedral lattice. In: Computer Graphics Forum. Volume 29., Wiley Online Library (2010)
14. Domke, J.: Learning graphical model parameters with approximate marginal inference. TPAMI (2013)
15. Kiefel, M., Gehler, P.V.: Human pose estmation with fields of parts. In: ECCV. (2014)
16. Barbu, A.: Training an active random field for real-time image denoising. Image Processing, IEEE Transactions on **18** (2009) 2451–2462
17. Ross, S., Munoz, D., Hebert, M., Bagnell, J.A.: Learning message-passing inference machines for structured prediction. In: CVPR. (2011)
18. Stoyanov, V., Ropson, A., Eisner, J.: Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In: AISTATS. (2011)
19. Tompson, J.J., J., A., LeCun, Y., Bregler, C.: Joint training of a convolutional network and a graphical model for human pose estimation. In: NIPS. (2014)
20. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: ICCV. (2015)
21. Sutton, C., McCallum, A.: Piecewise training of undirected models. In: Conference on Uncertainty in Artificial Intelligence (UAI). (2005)
22. Richard, X.H., Zemel, R.S., Carreira-perpiñán, M.Á.: Multiscale conditional random fields for image labeling. In: In CVPR, Citeseer (2004)
23. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning **1** (2008) 1–305
24. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. TPAMI **6** (1984)

25. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics (1951) 400–407
26. Spall, J.C.: Introduction to stochastic search and optimization: estimation, simulation, and control. Volume 65. John Wiley & Sons (2005)
27. Geyer, C.J.: Practical markov chain monte carlo. Statistical Science (1992) 473–483
28. Lauritzen, S.L.: Graphical Models. Oxford University Press (1996)
29. Gonzalez, J., Low, Y., Gretton, A., Guestrin, C.: Parallel gibbs sampling: From colored fields to thin junction trees. In: International Conference on Artificial Intelligence and Statistics. (2011) 324–332
30. Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural Computation **14** (2002) 1771–1800
31. Yuille, A.L.: The convergence of contrastive divergences. In: NIPS. (2004)
32. Tieleman, T.: Training restricted boltzmann machines using approximations to the likelihood gradient. In: ICML, ACM (2008)
33. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: (The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results)
34. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: International Conference on Computer Vision (ICCV). (2011)
35. Denil, M., Matheson, D., de Freitas, N.: Consistency of online random forests. In: ICML. (2013)
36. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 25. Curran Associates, Inc. (2012) 1097–1105
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
38. Ren, S., Cao, X., Wei, Y., Sun, J.: Global refinement of random forest. In: CVPR. (2015)
39. Cheng, M.M., Prisacariu, V.A., Zheng, S., Torr, P.H.S., Rother, C.: Densecut: Densely connected crfs for realtime grabcut. Computer Graphics Forum **34** (2015)