A GENERIC CNN-CRF MODEL FOR SEMANTIC SEGMENTATION

Alexander Kirillov[1] **Dmitrij Schlesinger**[1] **Walter Forkel**[1] **Anatoly Zelenin**[1]

Shuai Zheng[2] **Philip Torr**[2] **Carsten Rother**[1]

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

ABSTRACT

Deep Models, such as Convolutional Neural Networks (CNNs), are omnipresent in computer vision, as well as, structured models, such as Conditional Random Fields (CRFs). Combining them brings many advantages, foremost the ability to explicitly model the dependencies between output variables (CRFs) using thereby the incredible power of CNNs. In this work we present a CRF model were factors are dependent on CNNs. Our main contribution is a joint, maximum likelihood-based, learning procedure for all model parameters. Previous work either concentrated on training-in-pieces, or joint learning of restricted model families, such as Gaussian CRFs or CRFs with a few variables only. We empirically observe that our model is superior to prior art for scenarios where repulsive factors are necessary. In particular, we demonstrate this for Kinect-based body part labeling.

1 INTRODUCTION

The success of deep models achieved in a few past years is doubtless []. Especially in Computer Vision, Convolutional Neural Networks were applied for the wide range of applications – starting from low-level vision tasks like segmentation, stereo or optical flow up to the complex problems of scene understanding. Deep models have however their shortcomings as well. The most crucial one in our opinion is that these models are highly data-oriented, i.e. it is quite difficult (if possible at all) to incorporate prior knowledge into a deep learning framework. To this end, graphical models like e.g. Conditional Random Fields give fairly more possibilities. It is well known that they are able to capture for example geometric properties, spatial relations between objects, global properties like e.g. connectivity, shape and many others. Moreover, in contrast to CNNs, Conditional Random Fields allow to design models with desired properties. Obviously, combining CNN and CRF brings many advantages, foremost the ability to explicitly model the dependencies between output variables (CRFs) using thereby the incredible power of CNNs. In this work we present a CRF model were factors are dependent on CNNs. Our main contribution is a joint, maximum likelihood-based, learning procedure for all model parameters.

State of the art. Here, we would like to mention three relevant works, which try to learn CNNs and CRFs on top jointly. The first one is (Lin et al., 2015)). The crucial assumption of the proposed method is that the potentials can be approximated by logarithms of the corresponding marginals. Obviously, this assumption is very restrictive. Moreover, it can be easily seen that for some models it is just wrong. Consider for instance the case, when the underlying graph is a chain. If its pairwise potentials are set to the logarithms of the corresponding marginal pair-probabilities, the unary potentials are *negated* logarithms of the corresponding marginal label probabilities (up to the ends of the chain). For arbitrary graphs the connection between the potentials and the corresponding marginal probabilities is not known and the corresponding computation is NP-hard. We believe however, that this connection is much more complex as the approximation described above. Hence, although (Lin et al., 2015) shows quite impressive results, we do not follow these ideas, i.e. we do not approximate the task we would like to solve. Instead, we consider it as is, and, since it is intractable, use approximate algorithms. To summarize, the main difference of our approach from this one (as well as from the others discussed below) is that there is no model approximation in our method, but only algorithmic one. In other words, we do not want to compute *wrong* quantities *exactly*, our aim is to compute *right* quantities *approximately*.

Another interesting work is (Zheng et al., 2015), where the Mean Field approximation is employed for inference. It is elegantly represented by a Recurrent Neural Network, hence, allowing a joint end-to-end training. Note however, that again a model approximation takes place here, since Mean Field is an approximation of the maximum marginal decision indeed. Another crucial drawback of the method in (Zheng et al., 2015) is that it works with a particular class of pairwise potentials only. In contrast, we are free of these limitations. Using our approach it is possible to learn so-called repulsive pairwise potentials[1] that are especially useful in applications, where the spatial relations between object or object parts are crucial, like body part labeling or semantic segmentation.

The most close work to our is (Chen et al., 2015). In fact, we consider the same class of models as well as we are both using Maximum Likelihood for learning. The main contribution of (Chen et al., 2015) in our opinion is the method for approximating marginals. It is proposed to substitute the true marginals by local beliefs obtained by Loopy Belief Propagation. To our best knowledge, it is an open question, how good these beliefs do approximate the true marginals for general graphs. Hence, in this work again a model approximation is employed. In contrast, we use stochastic gradient ascent, where the necessary samples are drawn from the current posterior probability distribution by Gibbs Sampling. It has following advantages: (i) the marginal probabilities are not needed as such, and (ii) our algorithm is exact (i.e. it computes the true marginals) in the limit of infinity. It is worth to note, that obtaining the local beliefs is a difficult problem by itself, i.e. the necessary inference procedure is quite slow. In order to overcome this a "blending" scheme is proposed in (Chen et al., 2015) that essentially accelerates the method. However, the proposed scheme has very high space complexity, which is a crucial limitation for problems with many variables that is typical for Computer Vision tasks, like e.g. segmentation. Our approach in contrast has both low time- and low space-complexity – they both scale linearly with respect to the model size (number of nodes, number of labels etc.).

Contribution. Our main contribution is a joint, maximum likelihood-based, learning procedure for CRF models, where CNNs are used for potentials. There is no model approximation, like in (Chen et al., 2015; Zheng et al., 2015; Lin et al., 2015). Unlike (Zheng et al., 2015) we are able to learn arbitrary potentials. The proposed algorithm is an exact gradient ascent in the limit of infinity. It is relatively simple, quite fast and easy to parallelize. In contrast to (Chen et al., 2015), it has a low space-complexity that allows efficient GPU-implementation. We demonstrate these benefits on Kinect-based body part labeling, where we outperform state-of-the-art.

2 The model

In short, our model is a Conditional Random Field (CRF) [Lafferty] with unary potentials that depend on the image through a Convolutional Neural Network (CNN) [LeCun?]. Let $G = (R, E)$ be a graph with the node set R and the edge set E. In our applications the nodes correspond to image pixels. Each pixel should be labelled by a label l from a pre-defined finite discrete label set L. For instance, in body-part labelling the labels could have values e.g. "head", "left hand", "torso" etc. The task is to assign a label to each image pixel, i.e. to obtain a mapping $y : R \to L$. We denote by $y_i \in L$ the label chosen in the node $i \in R$, and more general, y_A denotes the restriction of the labelling y to a subset of nodes $A \subseteq R$. To model the posterior probability distribution of labellings y given images x we use pairwise CRF whose energy can be written as

$$E(x, y, \theta) = \sum_{i \in R} \psi_i(y_i, x, \theta) + \sum_c \sum_{ij \in E_c} \psi_c(y_i, y_j, \theta) \tag{1}$$

The unary potentials $\psi_i : L \to \mathbb{R}$ assign values to each label $l \in L$ of the pixel $i \in R$ depending on the image content x. To this end we use CNN (later we consider it in a bit more detail). Concerning the pairwise potentials, the CRF architecture is similar to (Flach & Schlesinger, 2011), i.e. we use densely connected neighbourhood structure. The set of all edges is split into "classes" referred by

[1]There is no common definition of repulsive potentials for more than two labels. Informally, such pairwise potentials suppresses equal labels in contrast to e.g. the Potts model, which promotes spatial smoothness.

the index c in (1). Thereby one class is characterized by the translation vector that connects the corresponding pixels. For example one particular class may consist of all edges connecting pixels that are neighbours in the horizontal direction in the image grid. Another class might be the set of edges so that one node is 2 pixels left and 3 pixels above the other one, etc. A subset of edges for one class is denoted by E_c. The pairwise potentials $\psi_c : L \times L \to \mathbb{R}$ are class specific, i.e. all edges of the same class share parameters. The free model parameters are the network weights w and the pairwise potentials ψ_c for all c. They are summarized by θ in (1). The corresponding probability distribution is defined as[2]

$$p(y|x;\theta) = \frac{1}{Z(x,\theta)} \exp\big[E(x,y,\theta)\big], \tag{2}$$

with the image specific partition function

$$Z(x,\theta) = \sum_y \exp\big[E(x,y,\theta)\big]. \tag{3}$$

3 Learning

Given a training sample $\big((x_1,y_1),(x_2,y_2)\dots(x_T,y_T)\big)$ the task is to maximize its conditional log-likelihood with respect to the unknown parameters θ:

$$(\arg)\max_\theta \sum_{t=1}^{T} \ln p(y_t|x_t;\theta) = (\arg)\max_\theta \sum_{t=1}^{T} \big[E(x_t,y_t,\theta) - \ln Z(x_t,\theta)\big]. \tag{4}$$

The gradient of the likelihood for a particular example t (for many examples their gradients should be summed up) is

$$\nabla_\theta = \frac{\partial E(x_t,y_t,\theta)}{\partial\theta} - \frac{\partial \ln Z(x_t,\theta)}{\partial\theta} = \frac{\partial E(x_t,y_t,\theta)}{\partial\theta} - \mathbb{E}_{p(y|x_t;\theta)}\left[\frac{\partial \ln E(x_t,y,\theta)}{\partial\theta}\right] \tag{5}$$

where $\mathbb{E}_p[\xi]$ denotes the expectation of a random variable ξ in the probability distribution p. Since the computation of this expectation is not tractable, we use stochastic approximation, i.e. we substitute the expectation of a random variable by its realization as follows:

1. Sample a labelling $\hat{y}$ according to the current probability distribution $p(y|x_t;\theta)$,
2. Compute stochastic gradient as

$$\nabla_\theta = \frac{\partial E(x_t,y_t,\theta)}{\partial\theta} - \frac{\partial E(x_t,\hat{y},\theta)}{\partial\theta}. \tag{6}$$

First, we would like to discuss the sampling procedure. We use Gibbs Sampling for this (Geman & Geman, 1984). Starting from an arbitrary labelling it is necessary to sample label in each node i according to the label probability distribution in this node given the image and the fixed rest:

$$p_i(y_i{=}l|x, y_{R\setminus i};\theta) \propto \exp\Big[\psi_i(l) + \sum_c \big(\psi_c(l, y_{j'} + \psi_c(y_{j''}, l)\big)\Big]. \tag{7}$$

Note, that according to our CRF architecture there are exactly two edges (up to pixels close to the borders) in each edge class c, that are incident to a given node i, the corresponding neighbouring nodes are denoted by j' and j'' in (7). We will call one scan over the whole image (i.e. performing (7) for all i) a sampling iteration.

In the theory, one should perform many sampling iterations starting from an arbitrary labelling in order to overcome the so-called burn-in phase of Gibbs Sampling, i.e. to obtain sample that does not depend on the initialization. Moreover, this should be done for each gradient computation (6). In order to accelerate the whole learning procedure, we use the trick, which is similar to the one, known e.g. in the RBF[3]-community under the name "Persistent Contrastive Divergence" [cite]. The intuition behind it is as follows. Assume that we have already a labelling y that is sampled

[2]The parameters are separated from the random variables by semicolon.

[3]Restricted Boltzmann Machines.

according to the current probability distribution (the burn-in phase is over). After applying the gradient, the model parameters (and hence, the probability distribution) is not changing essentially, since the gradient is usually applied with a small step-size. Therefore, it is enough to perform just one sampling iteration in order to obtain sample that matches the changed probability distribution adequately, although the labelling obtained in such a manner obviously depends on the previous one.

Another argument supporting this procedure is the following. At the beginning of the learning we usually start with a conditionally independent model (i.e. all pairwise potentials are zero). Note, that the label sampled according to (7) in a node i depends on the previous labelling only through the labels in the neighbouring nodes. If all pairwise potentials are zero, there is no dependency at all. Hence, for an independent model it is enough to perform just one sampling iteration in order to obtain perfect sample. To summarize, at the beginning of the learning process the samples obtained by such method are really drawn from the target probability distribution. On the other hand, at the end of the learning process gradients are very small (ideally zero). Consequently, the model parameters do not change essentially by the gradient step. Hence, one sampling iteration is enough again.

Although the proposed sampling schema reminds on Persistent Contrastive Divergence (PCD), we would like to point in that it is not the same, because in fact PCD performs another task – draw samples from a *joint* probability distribution $p(x, y)$, whereas our goal is to draw samples from the *posterior* $p(y|x)$. Thereby, PCD is correct only in the equilibrium state (when the burn-in phase is over), whereas our method draws perfect samples both at the beginning of the learning process and at the end as well (although unfortunately not in the middle). It has a crucial practical consequence, namely, it gives the possibility to start from an arbitrary conditionally independent posterior probability distribution $p(y|x)$, in particular to use Neural Networks that are pre-trained by standard methods without CRF on top.

Similar acceleration tricks as above are very popular also in other setups. For example in (Chen et al., 2015) an iterative procedure is used to obtain local beliefs which approximate marginals. It is proposed to perform only a small number of iterations in each gradient step, recording thereby the current state of the used algorithm. Very often, approaches like this are called “warm start”.

Now it remains only to explain, what is $\partial E(x, y, \theta)/\partial\theta$ in (6). We start from the unary potentials. As mentioned before, we use CNN for this, whose output can be considered as a $d = |R| \cdot |L|$-dimensional vector of “scores”. Let us denote this vector by $F(x, w) \in \mathbb{R}^d$, where w are the network weights. The labeling y can be also represented by the indicator vector $\phi(y)$ of the same dimension – each component of this vector corresponds to a pair $(i \in R, l \in L)$ and the values are 1 if $y_i = l$ and 0 otherwise. Hence, the energy part that corresponds to the unary potentials in (1) can can written as $\langle\phi(y), F(w, x)\rangle$, where $\langle\cdot\rangle$ denotes the inner product[4]. Given this, the derivative of the energy with respect to the network weights can be written according to the chain rule as

$$\frac{\partial E(x, y, w)}{\partial} = \frac{\partial\big\langle\phi(y), F(w, x)\big\rangle}{\partial F(x, w)} \cdot \frac{\partial F(x, w)}{\partial w} = \phi(y) \cdot \frac{\partial F(x, w)}{\partial w}. \tag{8}$$

Consequently, the stochastic gradient (6) is

$$\nabla_w = \phi(y_t) \cdot \frac{\partial F(x_t, w)}{\partial w} - \phi(y) \cdot \frac{\partial F(x_t, w)}{\partial w} = \big[\phi(y_t) - \phi(y)\big] \cdot \frac{\partial F(x_t, w)}{\partial w}, \tag{9}$$

and hence, can be computed via standard Error Back Propagation, where the difference $\big[\phi(y_t) - \phi(y)\big]$ is the error that should be propagated through the network.

The gradient (6) with respect to the pairwise potentials can be represented in a similar manner. Let us denote by $\phi'(y)$ the vector that consists of $C \cdot |L| \cdot |L|$ components (C is the number of edge classes), each one corresponding to a particular pairwise potential, i.e. an edge class c (see (1) and a label pair (l, l'). The value of each component of $\phi'(y)$ is the number of occurrences of the corresponding triple (c, l, l') in the labelling y. Denoting by θ the vector of all unknown pairwise potentials, the second term in (1) is again the inner product $\langle\phi'(y), \theta\rangle$. Hence, the stochastic gradient (6) with respect to the unknown pairwise potentials θ is just $\nabla_\theta = \phi'(y_t) - \phi'(y)$.

[4]Such representation is sometimes called “over-complete representation” and is often used e.g. in LP-relaxation based inference methods, representing a distribution as log-linear model, etc. In fact, the introduced vectors $\phi(y)$ and $\phi'(y)$ (see later) are sufficient statistics, when representing CRF as a member of the exponential family.

Algorithm 1 Computation of the stochastic gradient for one image

```
Input: Training example $x_t$, pre-computed data statistics $\phi(y_t)$ and $\phi'(y_t)$, current
       model parameters: pairwise potentials $\theta$ for CRF and network weights $w$, the
       previously sampled labelling $\hat{y}$ (random at the beginning)
Output: Gradients $\nabla_\theta$ and $\nabla_w$ of the conditional log-likelihood with respect to the pair-
       wise potentials and the network weights respectively
1) Apply the network – compute $F(x_t, w)$, i.e. the unary potentials $\psi_i(x_t, l)$
2) Do one sampling iteration using (7) in each node – obtain the new $\hat{y}$ starting from the old one
3) Compute the model statistics $\phi(y)$ and $\phi'(y)$
4) Compute the network error as $\phi(y_t) - \phi(y)$
5) Obtain the gradient $\nabla_w$ by propagating the error through the network (Error Back Propagation)
6) Compute the gradient $\nabla_\theta = \phi'(y_t) - \phi'(y)$
```

Summarizing all above, the computation of the stochastic gradient for just one example (x_t, y_t) is presented in Alg. 1. Note that the data statistics $\phi(y_t)$ and $\phi'(y_t)$ are not changed during the learning procedure. Hence, they can be precomputed in advance. The gradients obtained for all images (or for all images in a batch) is obtained by averaging.

At this point we would like to discuss some relevant properties of the proposed algorithm. First, applying the network and Error Back Propagation are standard operations. Hence, we are able to use elaborated third-party methods for this. In particular, we use Caffe-framework [cite] in our experiments. Second, Gibbs Sampling is relatively fast as compared with other techniques like e.g. Belief Propagation. It is also easy to parallelize. Note that our learning scheme does not involves any inference. It is indeed obvious, because the aim is to learn the statistical model, i.e. to fit a probability distribution to the given training data, rather then to optimize some inference results. Consequently, we do not need any inference algorithm, that would be slow. Finally, although we use "warm start" at each iteration, the amount of data to be stored is quite low – from iteration to iteration we only need to store a single labelling for each image. This is in contrast to many other techniques, like e.g. (Chen et al., 2015), where the warm start requires lots of data to be stored. In our case instead, the low space complexity of the proposed algorithm allows among other things an efficient GPU-implementation.

Inference. For such kind of CRFs the Maximum A-posteriori decision (MAP) strategy is usually used for inference. It leads to an Energy Minimization problem, i.e. the energy (1) should be maximized with respect to unknown labelling y. We do not follow this way because of the following reasons. First, from a theoretical viewpoint, we do not think that MAP is generally the best choice for statistical structural models. In fact, according to the Bayesian Decision Theory, MAP follows from the simplistic delta-loss, which does not reflect similarity between labellings. On the other hand, very often the results of the inference are rated by the pixel-accuracy, which corresponds to the Hamming loss. The latter leads to the Maximum-Marginal decision (MMARG), which should be obtained as follows. Given all potentials (i.e. after applying CNN), the posterior marginal node probabilities should be computed, followed by choosing the label having the best marginal in each node.

The next argument supporting MMARG is of a rather generic nature. It is a common opinion that both learning and inference should follow the same computational scheme to some extent. In our case the learning is based on marginals – i.e. the gradients and the network error (that should be propagated back through the network) are obtained from marginals. Hence, it seems to be reasonable to use a marginal-based decision strategy for inference as well.

The last argument against MAP is of technical nature. Note, that our potentials are arbitrary, i.e. we can not use e.g. MinCut based techniques or some search techniques, like e.g. α-expansion, since these methods work with submodular energies only. Hence, we have to resort to approximate algorithms for general case. In our opinion, the most powerful energy minimization framework for general purposes is LP-relaxation. It is however known that these algorithms have problems with "big numbers", for example if some label pair should have close to zero probability in the statistical model, i.e. a huge penalty in the corresponding energy minimization problem. Unfortunately, it is often the case in practice and, it is exactly the case in our experiments (see the next section). In particular, we tried TRWS and it turned out that it is just not able to produce reasonable results in reasonable time in almost all our experiments.

4 EXPERIMENTS

Experimental setup. In our experiments we consider two learning scenarios: separate learning and end-to-end learning. In both cases we use pre-trained CNN of the following architecture (will be described in the full version). For separate learning only CRF parameters are trained whereas the CNN weights are fixed. In contrast for end-to-end learning procedure all parameters are learned jointly.

Dataset and evaluation. We evaluate our approach on the challenging task of predicting human body parts from depth images. We use Kinect-dataset for this [cite]. It consists of rendered depth images along with the corresponding ground truths. There are 19 labels for body parts and one for the background (see Fig. 1 for the meaning of labels and examples). The dataset is split onto 2000 images for training and 500 images for the test. As an accuracy measure we use averaged per-pixel accuracy for body parts labels, i.e. excluding the background.

Results. Qualitative results can be seen on Fig. 1.

REFERENCES

Chen, Liang-Chieh, Schwing, Alexander G., Yuille, Alan L., and Urtasun, Raquel. Learning deep structured models. In *ICML 2015*, pp. 1785–1794, 2015.

Flach, Boris and Schlesinger, Dmitrij. Modelling composite shapes by gibbs random fields. In *CVPR 2011*, pp. 2177–2182, 2011.

Geman, Stuart and Geman, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721–741, 1984.

Lin, Guosheng, Shen, Chunhua, Reid, Ian D., and van den Hengel, Anton. Efficient piecewise training of deep structured models for semantic segmentation. *CoRR*, 2015. URL `http://arxiv.org/abs/1504.01013`.

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

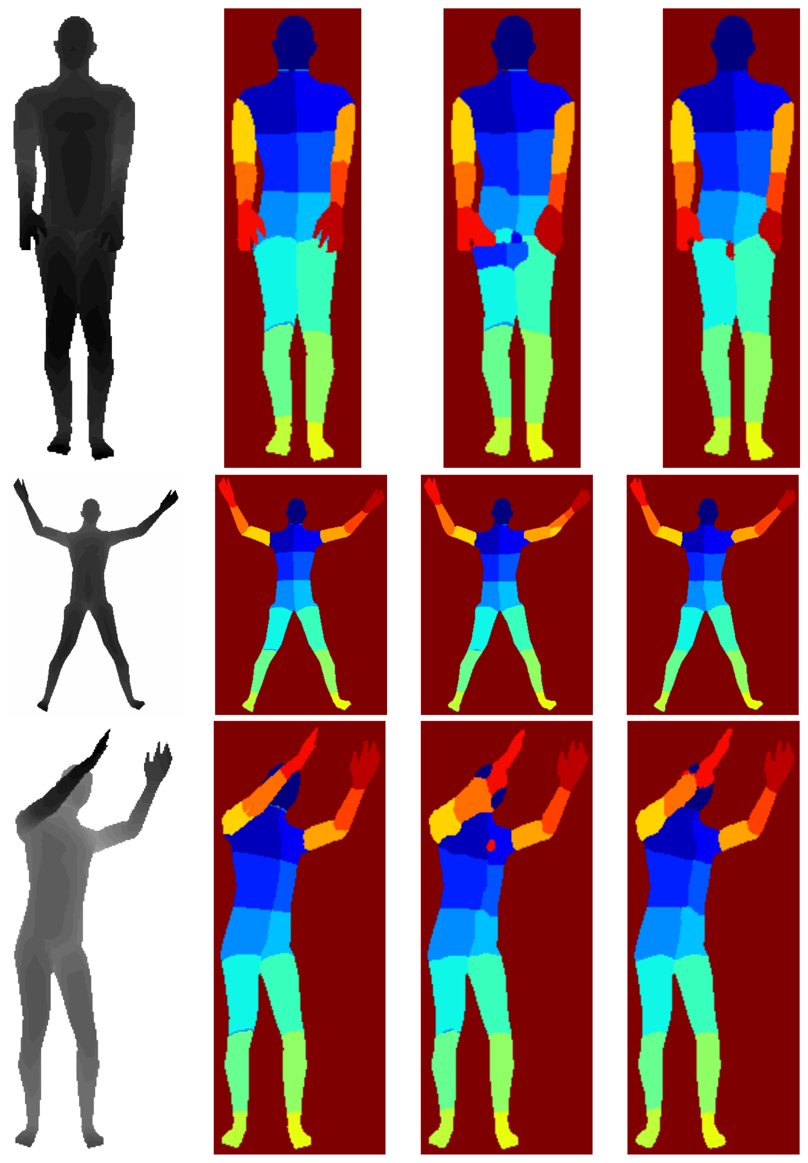

Figure 1: Kinect body part dataset. In the first column depth images are shown. The second column contains ground truths. Third and forth columns present the results for separate and end-to-end learned models respectively.