Fully Connected Deep Structured Networks

Alexander G. Schwing
University of Toronto
aschwing@cs.toronto.edu

Raquel Urtasun
University of Toronto
urtasun@cs.toronto.edu

Abstract

Convolutional neural networks with many layers have recently been shown to achieve excellent results on many high-level tasks such as image classification, object detection and more recently also semantic segmentation. Particularly for semantic segmentation, a two-stage procedure is often employed. Hereby, convolutional networks are trained to provide good local pixel-wise features for the second step being traditionally a more global graphical model. In this work we unify this two-stage process into a single joint training algorithm. We demonstrate our method on the semantic image segmentation task and show encouraging results on the challenging PASCAL VOC 2012 dataset.

1 Introduction

In the past few years, Convolutional Neural Networks (CNNs) have revolutionized computer vision. They have been shown to achieve state-of-the-art performance in a variety of vision problems, including image classification [19, 31], object detection [11], human pose estimation [32], stereo [36], and caption generation [15, 24, 35, 8, 14, 10]. This is mainly due to their high representational power achieved by learning complex, non-linear dependencies.

It is only very recently that convolutional nets have proven also very effective for semantic segmentation [12, 30, 21, 41, 3]. This is perhaps due to the fact that to achieve invariance, pooling operations are performed, often reducing the dimensionality of the prediction. A Markov random field (MRF) is then used as a refinement step in order to obtain segmentations that respect well segment boundaries. The seminal work of [17] showed that inference in fully connected MRFs is possible if the smoothness potentials are Gaussian. Impressive performance was demonstrated in semantic segmentation with hand craft features. Later, [3] extended the unary potentials to incorporate convolutional network features. However, these current approaches train the segmentation models in a piece-wise fashion, fixing the unary weights during learning of the parameters of the pairwise terms which enforce smoothness.

In this paper we present an algorithm that is able to train jointly the parameters of the convolutional network defining the unary potentials as well as the smoothness terms taking into account the dependencies between the random variables. We demonstrate the effectiveness of our approach using the dataset of the PASCAL VOC 2012 challenge [9].

2 Background

We begin by describing how to learn probabilistic deep networks which take into account correlations between multiple output variables $y = (y_1, \dots, y_N)$ that are of interest to us. Moreover, a valid configuration $y \in \mathcal{Y} = \prod_{i=1}^{N} \mathcal{Y}_i$ is assumed to lie in the product space of the discrete variable domains $\mathcal{Y}_i = \{1, \dots |\mathcal{Y}_i|\}$.

For a given data sample $x \in \mathcal{X}$, and a parameter vector $w \in \mathbb{R}^A$, the score F of a configuration $y \in \mathcal{Y}$ is generally modeled by the mapping $F : \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^A \to \mathbb{R}$.

Algorithm: Deep Learning
Repeat until stopping criteria

1. Forward pass to compute $F(x, \hat{y}; w)\ \forall \hat{y} \in \mathcal{Y}$
2. Normalization via soft-max to obtain $p(\hat{y} \mid x, w)$
3. Backward pass through definition of function via chain rule
4. Parameter update

Figure 1: Gradient descent for learning deep models.

The *prediction task* amounts to finding the configuration

$$y^* = \arg\max_{\hat{y} \in \mathcal{Y}} F(x, \hat{y}; w), \tag{1}$$

which maximizes the score $F(x, \hat{y}; w)$. Note that the best scoring configuration y^* is equivalently given as the maximizer of the probability distribution

$$p(\hat{y} \mid x, w) \propto \exp F(x, \hat{y}; w),$$

since the exponential function is a monotone increasing function and the normalization constant is independent of the configuration $\hat{y} \in \mathcal{Y}$, *i.e.*, it is constant indeed.

The *learning task* is concerned with finding a parameter vector

$$w^* = \arg\max_{w \in \mathbb{R}^A} \prod_{(x,y) \in \mathcal{D}} p(y \mid x, w), \tag{2}$$

which maximizes the likelihood of a given training set $\mathcal{D} = \{(x, y)\}$. The training set consists of input-output pairs (x, y) which are assumed to be independent and identically distributed. Note that maximizing the likelihood is equivalent to maximizing the cross entropy between the modeled distribution $p(\hat{y} \mid x, w)$ and a target distribution which places all its mass on the groundtruth configuration y. Throughout this work we make no further assumptions about the dependence of the scoring function $F(x, \hat{y}; w)$ on the parameter vector w, *i.e.*, $F(x, \hat{y}; w)$ is generally neither convex nor smooth.

For problems where the output-space size $|\mathcal{Y}| = \prod_{i=1}^{N} |\mathcal{Y}_i|$ is in the thousands, we can exactly solve the inference task given in Eq. (1) by searching over all possible output space configurations $\hat{y} \in \mathcal{Y}$. In such a setting, those different configurations are typically referred to as different classes. Similarly, we normalize the distribution $p(\hat{y} \mid x, w)$ by summing up the exponentiated score $\exp F(x, \hat{y}; w)$ over all possibilities $\hat{y} \in \mathcal{Y}$. This is often referred to as a soft-max computation. Non-convexity and non-smoothness of the learning objective w.r.t. the parameters w is answered with stochastic gradient ascent. For efficiency, the gradient is often computed on a small subset of the training data, *i.e.*, a mini-batch.

We summarize the resulting training algorithm in Fig. 1. On a high level it consists of four steps which are iterated until a stopping criterion is met: (i) the forward pass to compute the scoring function $F(x, \hat{y}; w)$ for all output space configurations $\hat{y} \in \mathcal{Y}$. (ii) normalizing the scoring function via a soft-max computation to obtain the probability distribution $p(\hat{y} \mid x, w)$. (iii) computation and back-propagation of the gradient of the loss function, *i.e.*, often the log-likelihood or equivalently the cross-entropy. (iv) an update of the parameters.

However, solving the inference task given in Eq. (1) or the learning problem stated in Eq. (2) is computationally challenging if we consider more complex output spaces $\mathcal{Y}$, *e.g.*, those arising from tasks like image tagging. The situation is even more severe if we target image segmentation where the exponential number of possible output space configurations prevents even storage of $F(x, \hat{y}; w)$ $\forall \hat{y} \in \mathcal{Y}$. Note that this is required in the first line of the algorithm summarized in Fig. 1.

Given an exponential amount of possible configurations $|\mathcal{Y}| = \prod_{i=1}^{N} |\mathcal{Y}_i|$, how do we represent the scoring function $F(x, \hat{y}; w)$ efficiently? Assuming we have an efficient representation, how can we effectively normalize the probability $p(\hat{y} \mid x, w)$? One possible answer to those questions was given by Chen *et al*. [4], who discussed extending log-linear models, *i.e.*, those with a scoring function of

Algorithm: Learning Deep Structured Models
Repeat until stopping criteria

1. Forward pass to compute $f_r(x, \hat{y}_r; w)\ \forall r \in \mathcal{R}, \hat{y}_r \in \mathcal{Y}_r$
2. Computation of marginals $b_{(x,y),r}(\hat{y}_r)$ via loopy belief propagation, convex belief propagation or tree-reweighted message passing
3. Backward pass through definition of function via chain rule
4. Parameter update

Figure 2: Approximated gradient descent for learning deep structured models.

the form $F(x, \hat{y}; w) = w^\top \phi(x, \hat{y})$, to the more general setting, *i.e.*, an arbitrary dependence of the scoring function $F(x, \hat{y}; w)$ on the parameter vector w.

In short, [4] assumed the global scoring function $F(x, \hat{y}; w)$ to decompose into a sum of local scoring functions f_r, each depending on a small subset $r \subseteq \{1, \dots, N\}$ of variables $\hat{y}_r = (\hat{y}_i)_{i \in r}$. All restrictions r required to compute the global function via

$$F(x, \hat{y}; w) = \sum_{r \in \mathcal{R}} f_r(x, \hat{y}_r; w) \tag{3}$$

are subsumed in the set $\mathcal{R}$. If the size of each and every local restriction set $r \in \mathcal{R}$ is small, $F(x, \hat{y}; w)$ is efficiently representable.

To compute the gradient of the log-likelihood cost function, we require a properly normalized distribution $p(\hat{y} \mid x, w)$, or more specifically its marginals $b_{(x,y),r}(\hat{y}_r)$ for each restriction $r \in \mathcal{R}$. To this end, message passing type algorithms were employed by [4]. Such an approach is exact if the distribution $p(\hat{y} \mid x, w)$ is of low tree-width. Otherwise computational complexity is prohibitively large and approximations like loopy belief propagation [26], convex belief propagation [39] or tree-reweighted message passing [37] are alternatives that were successfully applied.

The resulting iterative method of [4] is summarized in Fig. 2. In a first step the forward pass computes all outputs of every local scoring function. Afterwards (approximate) marginals are obtained in a second step, and utilized to compute the derivative of the (approximated) maximum likelihood cost function w.r.t. the parameters w. The following backward pass computes the gradient of the parameters by repeatedly applying the chain-rule according to the definition of the scoring function $F(x, \hat{y}; w)$. The gradient is then utilized during the final parameter update.

Not only does the approach presented by [4] fail if the decomposition assumed in Eq. (3) is not available. But it is also computationally challenging to obtain the required marginals if too many local functions are required. *I.e.*, computation is slow if the number of restrictions $|\mathcal{R}|$ is large, *e.g.*, when working with densely connected image segmentation models where every pixel is possibly correlated to every other pixel in the image.

3 Approach

Densely connected models were previously considered by [17, 33, 34, 18] and shown to yield impressive results for the image segmentation task. Learning the parameters of densely connected models was considered by Krähenbühl and Koltun [18] in the context of the log-linear setting. Following [4] we aim at extending those fully connected log-linear models to the more general setting of an arbitrary function $F(x, \hat{y}; w)$, *e.g.*, a deep convolutional neural network. Note that a similar approach has been recently discussed by [41] in independent work.

Let us consider within this section how to efficiently combine deep structured prediction [4] with densely connected probabilistic models [17, 33, 34, 18]. Before getting into the details we note that the presented approach trades computational complexity of the general method of [4] with a restriction on the pairwise functions f_{ij} (*i.e.*, $r = \{i, j\}$). Concretely, the local functions f_{ij} are assumed to be mixtures of kernels in a feature space as detailed below. For simplicity we assume that local functions of order higher than two are not required to represent our global scoring function $F(x, \hat{y}; w)$. Generalizations have however been presented, *e.g.*, by Vineet *et al.* [34].

3.1 Inference

We begin our discussion by considering the inference task. To obtain a computationally efficient prediction algorithm we use a mean field approximation of the model distribution $p(\hat{y} \mid x, w)$ for every sample (x, y). More formally, we assume our approximation to factor according to $q_{(x,y)}(\hat{y}) = \prod_{i=1}^{N} q_{(x,y),i}(\hat{y}_i)$. Given some parameters w, we employ a forward pass to obtain our local function representations $f_r(x, \hat{y}_r; w)$. Next we compute the single variable marginals $q_{(x,y),i}(\hat{y}_i)$ by minimizing the Kullback-Leibler (KL) divergence w.r.t. to the assumed factorization of the mean field distribution $q_{(x,y)}(\hat{y})$, *i.e.*,

$$q^*_{(x,y)} = \arg\min_{q \in \Delta} D_{\mathrm{KL}}(q_{(x,y)}(\hat{y}) || p(\hat{y} \mid x, w)). \tag{4}$$

Hereby $q \in \Delta$ requires q to be a valid probability distribution. Due to non-convexity, only convergence to a stationary point of the KL divergence cost function is guaranteed for sequential block-coordinate updates [38, 16]. More precisely, iterating until convergence through the variables $i \in \{1, \ldots, N\}$ using the closed form update

$$q_{(x,y),i}(\hat{y}_i) \propto \exp\left(f_i(\hat{y}_i, x, w) + \sum_{j \in \mathcal{N}(i), \hat{y}_j} f_{ij}(\hat{y}_i, \hat{y}_j, x, w) q_{(x,y),j}(\hat{y}_j) \right), \tag{5}$$

which assumes all marginals but $q_{(x,y),i}$ to be fixed, retrieves a stationary point for the cost function of the program given in Eq. (4). The set of variables neighboring i is denoted $\mathcal{N}(i)$.

In the case of densely connected variables, the computational bottleneck arises from the second summand which involves $\sum_{j \in \mathcal{N}(i)} |\mathcal{Y}_j|$ additions. The sum ranges over $|\mathcal{N}(i)| = N - 1$ terms for densely connected structured models. Hence the complexity of an update for a single marginal is of $O(N)$, and updating all N marginals therefore requires $O(N^2)$ operations as also discussed by Krähenbühl and Koltun [18].

Importantly, Krähenbühl and Koltun [17] observed that a high dimensional Gaussian filter can be applied to concurrently update all marginals in $O(N)$. This is achievable when constraining ourselves to pairwise functions being mixtures of M kernels in the feature space as mentioned before. Formally, we require

$$f_{ij}(\hat{y}_i, \hat{y}_j, x, w) = \sum_{m=1}^{M} \mu^{(m)}(\hat{y}_i, \hat{y}_j, w) k^{(m)}(\hat{f}_i(x) - \hat{f}_j(x)),$$

where $\mu^{(m)}$ is a label compatibility function, $k^{(m)}$ is a kernel function, and $\hat{f}_i(x)$ are features of variable i depending on the data x.

However, to ensure convergence to a stationary point of the KL divergence cost function for this parallel update, further restrictions on the form of the pairwise functions f_{ij} apply. Formally, if the label compatibility functions $\mu^{(m)}$ are negative semi-definite $\forall m$, and the kernels $k^{(m)}$ are positive definite $\forall m$, the KL divergence is readily given as the difference between a concave and a convex term [18]. Hence the concave-convex procedure (CCCP) [40] is directly applicable. We therefore proceed iteratively by first linearizing the concave term at the current location and second minimizing the resulting linearized but convex program.

As detailed by Krähenbühl and Koltun [18], and as discussed above, finding the linearization is equivalently solved via filtering in time linear in N. Solving the convex program in its original form requires solving a non-linear system of equations independently for each marginal $q_{(x,y),i}(\hat{y}_i)$, *e.g.*, via Newton's method. A further approximation to the cross-entropy term of the KL-divergence relates the efficient filtering based mean field update of the marginals $q_{(x,y),i}(\hat{y}_i)$ to the corresponding cost function for which a stationary point is found.

3.2 Learning

Having observed that mean-field inference can be efficiently addressed with Gaussian filtering, given restrictions on the pairwise functions f_{ij}, we now turn our attention to the learning task. As mentioned before we aim at finding a parameter vector w that maximizes the likelihood objective function. Since the exact likelihood is computationally expensive, we use the log-likelihood based on

Algorithm: Learning Fully Connected Deep Structured Models
Repeat until stopping criteria

1. Forward pass to compute $f_r(x, \hat{y}_r; w)\ \forall r \in \mathcal{R}, y_r \in \mathcal{Y}_r$
2. Computation of marginals $q^t_{(x,y),i}(\hat{y}_i)$ via filtering for $t \in \{1, \dots, T\}$
3. Backtracking through the marginals $q^t_{(x,y),i}(\hat{y}_i)$ from $t = T - 1$ down to $t = 1$
4. Backward pass through definition of function via chain rule
5. Parameter update

Figure 3: Stochastic gradient descent for learning fully connected deep structured models.

the mean-field marginals. Hence our surrogate loss function $L_{(x,y)}$ for a sample (x, y) with corresponding annotated ground truth labeling y is given by

$$L_{(x,y)}(q_{(x,y)}) = -\sum_{i=1}^{N} \log q_{(x,y),i}(y_i). \tag{6}$$

To perform a parameter update step we need the gradient of the surrogate loss function w.r.t. the parameters, *i.e.*,

$$\frac{\partial L_{(x,y)}}{\partial w} = \frac{\partial L_{(x,y)}}{\partial q_{(x,y)}} \cdot \frac{\partial q_{(x,y)}}{\partial w}. \tag{7}$$

The gradient of the surrogate loss function $L_{(x,y)}$ w.r.t. the marginals is easily obtained from Eq. (6). It is given by

$$\frac{\partial L_{(x,y)}}{\partial q_{(x,y),i}(\hat{y}_i)} = -\frac{1}{q_{(x,y),i}(y_i)} [\![\hat{y}_i = y_i]\!], \tag{8}$$

where the Iverson bracket $[\![\hat{y}_i = y_i]\!]$ equals one if $\hat{y}_i = y_i$, and returns zero otherwise.

To perform a gradient step during learning, we additionally require the derivatives of the marginals w.r.t. the parameters, *i.e.*, $\frac{\partial q_{(x,y),i}(\hat{y}_i)}{\partial w}$.

More carefully investigating the mean-field update given in Eq. (5) reveals a recursive definition. More concretely, the derivative $\frac{\partial q^t_{(x,y),i}(\hat{y}_i)}{\partial w}$ of the marginal $q^t_{(x,y),i}(\hat{y}_i)$ after t iterations depends on the results from earlier iterations. Hence, we obtain the desired result by successively back-tracking through the mean-field iterations from the last iteration back to the first. This direct computation is however computationally expensive. Fortunately, back-substitution into the loss gradient yields an algorithm which requires a total of T back-tracking steps, independent of the number of parameters. We refer the interested reader to [18] for additional details regarding the computation of the gradient $\frac{\partial q_{(x,y),i}(\hat{y}_i)}{\partial w}$.

But contrasting [18], we no longer assume the unaries to be given by a logistic regression model. Contrasting [3], we don't assume the unaries to be fixed during CRF parameter updates. Generalizing the gradient of the marginals w.r.t. parameters to arbitrary unaries is straightforward since the gradients are directly given by the marginals. Combined with the gradient of the log-likelihood loss function w.r.t. the marginals, given in Eq. (8), we obtain $\frac{\partial L_{(x,y)}}{\partial w}$ as the difference between the ground-truth and the predicted marginals. This result is then used for back-propagation through any functional structure which provides the unary scoring functions f_i, *e.g.*, convolutional neural networks.

Derivatives w.r.t. to label compatibility and kernel shape parameters are readily given in [18]. The resulting algorithm is summarized in Fig. 3. In short, we first obtain again our functional representation via a forward pass through any functional network. Subsequently we compute our mean-field marginals via filtering. Afterwards we obtain the gradient of the loss function via an efficient back-tracking. In the next step the gradient of the parameters is computed by back-propagating the gradient of the loss-function using the chain-rule dictated by the definition of the scoring function. In a final step we update the parameters.

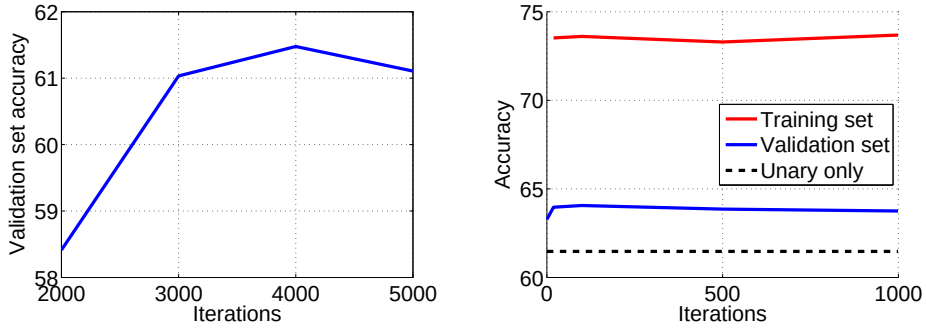

Figure 4: (a) Validation set performance over the number of iterations when fine-tuning the unary parameters only. (b) Validation set performance over the number of iterations when fine-tuning all parameters.

4 Experiments

We evaluate our approach summarized in Fig. 3 on the dataset of the Pascal VOC 2012 challenge [9]. The task is semantic image segmentation of 21 object classes (including background). The original dataset contains 1464 training, 1449 validation and 1456 test images. In addition to this data we make use of the annotations provided by Hariharan *et al*. [13], resulting in a total of 10582 training instances. The reported performance is measured using the intersection-over-union metric. Note that we conduct our tests on the 1449 validation set images which were neither used during training nor for fine-tuning.

4.1 Model

Our model setup follows [3], *i.e*., we employ the 16 layer DeepNet model [31]. Just like [3] we first convert the fully connected layers into convolutions as first discussed in [12, 30]. This is useful since we are not interested in a single variable output prediction, but rather aim at learning probability masks. To obtain a larger probability mask we skip downsampling during the last two max-pooling operations. To take into account the skipped downsampling during subsequent convolutions we employ the 'à trous (with hole) algorithm' [23]. It takes care of the fact that data is stored in an interleaved way, *i.e*., in our case convolutions sub-sample the input data by a factor of two or four respectively. To adapt to the 21 object classes we also replace the top layer of the DeepNet model to yield 21 classes for each pixel.

Similar to [3] we assume the input size of our network to be of dimension 306×306 which results in a 40×40 sized spatial output of the DeepNet which is in our case an *intermediate* result however.

Contrasting [3], we jointly optimize for both unary and CRF parameters using the algorithm presented in Fig. 3. To this end, given images downsampled to a size of 306×306, our algorithm first performs a forward pass through the convolutional DeepNet to obtain the $40 \times 40 \times 21$ sized class probability maps in an *intermediate* stage. These intermediate class probability maps are directly up-sampled to the original image dimension using a bi-linear interpolation layer. This yields the actual output of our augmented DeepNet network defining the scoring function $F(x, \hat{y}, w)$. Note that the number N of variables $\hat{y} = (\hat{y}_1, \dots, \hat{y}_N) \in \mathcal{Y}$ is therefore equal to the number of pixels of the original image.

For the second step of our algorithm we perform 5 iterations of mean field updates to compute the marginals $q_{(x,y),i}(\hat{y}_i)$ of the fully connected CRF. Those are then compared to the original groundtruth image segmentations, using as our loss function the sum of cross-entropy terms, *i.e*., the log-likelihood loss, as specified in Eq. (6). In the third step we back-track through the marginals to obtain a gradient of the loss function. Afterwards we back-propagate the derivatives w.r.t. the unary term through both the bi-linear interpolation and the 16-layer convolutional network. The shape and compatibility parameters of the CRF, detailed below, are updated directly.

Data	bkg	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow
Valid.	90.461	77.455	30.355	76.564	60.735	65.075	81.261	74.958	81.505	23.367	66.279
Train	90.159	76.314	64.450	78.677	68.224	68.044	84.491	80.274	86.347	44.567	79.987

Data	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	Our mean	[3]
Valid.	52.219	70.624	66.660	65.725	72.913	42.174	73.452	43.412	71.738	58.322	**64.060**	63.74
Train	62.710	82.987	76.729	76.523	75.399	63.863	79.937	55.146	80.699	70.164	73.604	-

Table 1: Performance of our approach for individual classes. In the last two columns of the lower panel we compare our mean to the recently presented baseline by Chen *et al*. [3].

It was shown independently by many authors [31, 4], that successively increasing the number of parameters during training typically yields better performance due to better initialization of larger models. We therefore train our model in two stages. First, we assume no pairwise connections to be present, *i.e*., we fine-tune the weights obtained from the DeepNet ImageNet model [31, 29] to the Pascal dataset [9]. Standard parameter settings for a momentum of 0.9, a weight decay of 0.0005 and learning rates of 0.01 and 0.001 for the top and all other layers are employed respectively. Due to the 12GB memory restrictions on the Tesla K40 GPU we use a mini-batch size of 20 images.

In a second stage we jointly train the convolutional network parameters as well as the compatibility and shape parameters of the dense CRF arising from the pairwise functions

$$f_{ij}(\hat{y}_i, \hat{y}_j, x, w) = \mu(\hat{y}_i, \hat{y}_j) \sum_{m=1}^{2} w_m k^{(m)}(\hat{f}_i^{(m)}(x) - \hat{f}_j^{(m)}(x)). \tag{9}$$

Hereby, we employ the Potts potential $\mu(y_i, y_j) = [\![y_i = y_j]\!]$ and the Gaussian kernels given by

$$k^{(m)} = \exp\left(-\frac{1}{2}(f_i^{(m)} - f_i^{(m)})^\top \Sigma_m^{-1}(\hat{f}_i^{(m)} - \hat{f}_i^{(m)})\right).$$

As indicated in Eq. (9), we use $M = 2$ kernels, both with diagonal covariance matrix Σ_m. One containing as features $\hat{f}_i(x)$ the two-dimensional pixel positions, the other one containing as features the two dimensional pixel positions as well as the three color channels. Hence we obtain a total of nine parameters, *i.e*., two compatibility parameters w_1 and w_2 and $2+5 = 7$ kernel shape parameters for the diagonal covariance matrices Σ_m.

4.2 Results

As mentioned before, all our results were computed on the validation set of the Pascal VOC dataset. This part of the data was neither used for training nor for fine-tuning.

Unary performance: We first investigate the performance of the first training stage of the proposed approach, *i.e*., fine-tuning of the 16 layer DeepNet parameters on the Pascal VOC data. The validation set accuracy is plotted over the number of iterations in Fig. 4 (a). We observe the performance to peak at around 4000 iterations with a mean intersection over union measure of 61.476%. The result reported by [3] for this experiment is 59.80%, *i.e*., we outperform their unary model by 1.5%.

Joint training: Next we illustrate the performance of the second step, *i.e*., joint training of both convolutional network parameters and CRF compatibility and shape parameters. In Fig. 4 (b) we indicate the best obtained unary performance from the first step and visualize the validation and training set performance over the number of iterations. We observe the results to peak quickly after around 20 iterations and remain largely stable thereafter.

Details: In Tab. 1 we provide the training and test set accuracies for the 21 individual classes. We observe the 'bike' and 'chair' class to be particularly difficult. For both categories the validation set performance is roughly half of the training set accuracy.

Comparison to baseline: As provided in Tab. 1, the peak validation set performance of our approach is 64.060%, which slightly outperforms the separate training result of 63.74% reported by Chen *et al*. [3].

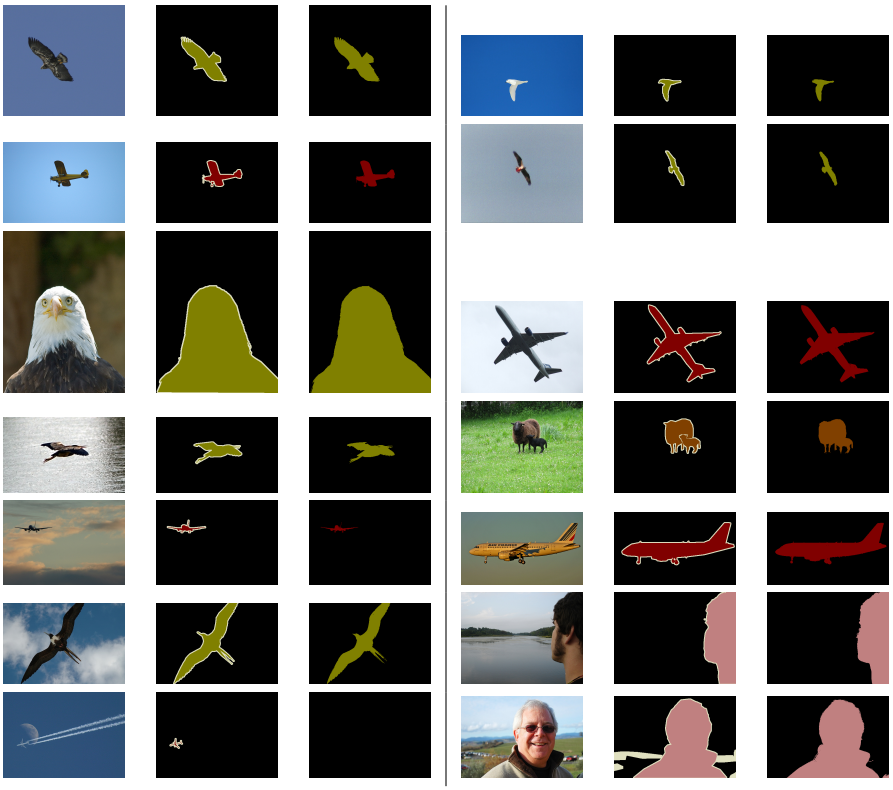

Figure 5: Visual results of good predictions.

Visual results: We illustrate visual results of our approach in Fig. 5. Our method successfully segments the object if the images are clearly apparent. Noisy images and objects with many variations pose challenges to the presented approach as visualized in Fig. 6. Also, we observe our learnt parameters to generally over-smooth results while being noisy on the boundaries.

5 Discussion

We presented a first method that jointly trains convolutional neural networks and fully connected conditional random fields for semantic image segmentation. To this end we generalize [3] to joint training. Note that a method along those lines has also been recently made publicly available in independent work [41]. Whereas the latter combines dense conditional random fields [17] with the fully convolutional networks presented by Long *et al*. [21], we employ and modify the 16 layer DeepNet architecture presented in work by Simonyan and Zisserman [31].

Ideas along the lines of joint training were discussed within machine learning and computer vision as early as the 90's in work done by Bridle [2] and Bottou [1]. More recently [5, 27, 22, 6, 28, 25] incorporate non-linearities into unary potentials but generally assume exact inference to be tractable. Even more recently, Li and Zemel [20] investigate training with hinge-loss objectives using non-linear unaries, but the pairwise potentials remain fixed, *i.e*., no joint training. Domke [7] decomposes the learning objective into logistic regressors which will be computationally expensive in our setting. Tompson *et al*. [32] propose joint training for pose estimation based on a heuristic approximation which ignores the normalization constant of the model distribution. Joint training of conditional random fields and deep networks was also discussed recently by [4] for graphical models in general. Techniques based on convex and non-convex approximations were described for obtaining marginals in the general non-linear setting.

Figure 6: Failure cases

6 Conclusion

We discussed a method for semantic image segmentation that jointly trains convolutional neural networks and conditional random fields. Our approach combines techniques from deep convolutional neural networks with variational mean-field approximations from the graphical model literature. We obtain good results on the challenging Pascal VOC 2012 dataset.

In the future we plan to train our method on larger datasets. Additionally we want to investigate training with weakly labeled data.

References

[1] L. Bottou, Y. Bengio, and Y. LeCun. Global training of document processing systems using graph transformer networks. In *Proc. CVPR*, 1997.

[2] J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Proc. NIPS*, 1990.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic ImageSegmentation with Deep Convolutional Nets and Fully Connected CRFs. http://arxiv.org/abs/1412.7062, 2015.

[4] L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning Deep Structured Models. http://arxiv.org/abs/1407.2538, 2014.

[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 2011.

[6] T.-M.-T. Do and T. Artieres. Neural conditional random fields. In *Proc. AISTATS*, 2010.

[7] J. Domke. Structured Learning via Logistic Regression. In *Proc. NIPS*, 2013.

[8] J. Donahue, L. A. Hendrikcs, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. http://arxiv.org/abs/1411.4389, 2014.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[10] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig. From captions to visual concepts and back. http://arxiv.org/abs/1411.4952, 2014.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014.

[12] A. Guisti, D. Ciresan, J. Masci, L. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *Proc. ICIP*, 2013.

[13] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic Contours from Inverse Detectors. In *Proc. ICCV*, 2011.

[14] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. http://arxiv.org/abs/1412.2306, 2014.

[15] R. Kiros, R. Salahutdinov, and R. Zemel. Multi- modal neural language models. In *Proc. ICML*, 2014.

[16] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[17] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Proc. NIPS*, 2011.

[18] P. Krähenbühl and V. Koltun. Parameter Learning and Convergent Inference for Dense Random Fields. In *Proc. ICML*, 2013.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proc. NIPS*, 2013.

[20] Y. Li and R. Zemel. High Order Regularization for Semi-Supervised Learning of Structured Output Problems. In *Proc. ICML*, 2014.

[21] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. http://arxiv.org/abs/1411.4038, 2014.

[22] J. Ma, J. Peng, S. Wang, and J. Xu. A conditional neural fields model for protein threading. *Bioinformatics*, 2012.

[23] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[24] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). http://arxiv.org/abs/1412.6632, 2014.

[25] J. Morris and E. Fosler-Lussier. Conditional random fields for integrating local discriminative classifiers. *IEEE Trans. Audio, Speech, and Language Processing*, 2008.

[26] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[27] J. Peng, L. Bo, and J. Xu. Conditional Neural Fields. In *Proc. NIPS*, 2009.

[28] R. Prabhavalkar and E. Fosler-Lussier. Backpropagation training for multilayer conditional random field based phone recognition. In *Proc. ICASSP*, 2010.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[30] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Proc. ICLR*, 2014.

[31] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. http://arxiv.org/abs/1409.1556, 2014.

[32] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. In *Proc. NIPS*, 2014.

[33] V. Vineet, J. Warrell, P. Sturgess, and P. H. S. Torr. Improved initialization and Gaussian mixture pairwise terms for dense random fields with mean-field inference. In *Proc. BMVC*, 2012.

[34] V. Vineet, J. Warrell, and P. H. S. Torr. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. In *Proc. ECCV*, 2012.

[35] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. http://arxiv.org/abs/1411.4555, 2014.

[36] J. Žbontar and Y. LeCun. Computing the Stereo Matching Cost with a Convolutional Neural Network. http://arxiv.org/abs/1409.4326, 2014.

[37] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *Trans. Information Theory*, 2003.

[38] M. J. Wainwright and M. I. Jordan. *Graphical models, exponential families and variational inference*. Foundations and Trends in Machine Learning, 2008.

[39] Y. Weiss, C. Yanover, and T. Meltzer. MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies. In *Proc. UAI*, 2007.

[40] A. L. Yuille and A. Rangarajan. The Concave-Convex Procedure (CCCP). *Neural Computation*, 2003.

[41] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional Random Fields as Recurrent Neural Networks. http://arxiv.org/abs/1502.03240, 2015.