Integrated Inference and Learning of Neural Factors in Structural Support Vector Machines

Rein Houthooft*, Filip De Turck

Ghent University - iMinds, Department of Information Technology
Technologiepark 15, Ghent, B-9052, Belgium

Abstract

Tackling pattern recognition problems in areas such as computer vision, bioinformatics, speech or text recognition is often done best by taking into account task-specific statistical relations between output variables. In structured prediction, this internal structure is used to predict multiple outputs simultaneously, leading to more accurate and coherent predictions. Structural support vector machines (SSVMs) are nonprobabilistic models that optimize a joint input-output function through margin-based learning. Because SSVMs generally disregard the interplay between unary and interaction factors during the training phase, final parameters are suboptimal. Moreover, its factors are often restricted to linear combinations of input features, limiting its generalization power. To improve prediction accuracy, this paper proposes: (i) Joint inference and learning by integration of back-propagation and loss-augmented inference in SSVM subgradient descent; (ii) Extending SSVM factors to neural networks that form highly nonlinear functions of input features. Image segmentation benchmark results demonstrate improvements over conventional SSVM training methods in terms of accuracy, highlighting the feasibility of end-to-end SSVM training with neural factors.

Keywords: structural support vector machine, neural factors, structured prediction, neural networks, image segmentation

1. Introduction

In traditional machine learning, the output consists of a single scalar, whereas in structured prediction, the output can be arbitrarily structured. These models have proven useful in tasks where output interactions play an important role. Examples are image segmentation, part-of-speech tagging, and optical character recognition, where taking into account contextual cues and predicting all output variables at once is beneficial. A widely used framework is the conditional random field (CRF), which models the statistical conditional dependencies between input and output variables, as well as between output variables mutually. However, many tasks only require 'most-likely' predictions, which led to the rise of nonprobabilistic approaches. Rather than optimizing the Bayes' risk, these models minimize a structured loss, allowing the optimization of performance indicators directly [1]. One such model is the structural support vector machine (SSVM) [2] in which a generalization of the hinge loss to multiclass and multilabel prediction is used.

A downside to traditional SSVM training is the bifurcated training approach in which *unary factors* (dependencies of outputs on inputs), and *interaction factors* (mutual output dependencies) are trained sequentially. A unary classification model is optimized, while the interactions are trained post-hoc. However, this two-phase approach is suboptimal, because the errors made during the training of the interaction factors cannot be accounted for during training of the unary classifier. Another limitation is that SSVM factors are linear feature combinations, restricting the SSVM's generalization power. We propose to extend these linearities to highly nonlinear functions by means of multilayer neural networks, to which we refer as *neural factors*. Towards this goal, subgradient descent is extended by combining loss-augmented inference with back-propagation of the SSVM objective error into both unary and interaction neural factors. This leads to better generalization and more synergy between both SSVM factor types, resulting in more accurate and coherent predictions.

Our model is empirically validated by means of the complex structured prediction task of image segmentation on the MSRC-21, KITTI, and SIFT Flow benchmarks. The results demonstrate that integrated inference and learning, and/or using neural factors, improves prediction accuracy over conventional SSVM training methods, such as N-slack cutting plane and subgradient descent optimization [1]. Furthermore, we demonstrate that our model is able to perform on par with current state-of-the-art segmentation models on the MSRC-21 benchmark.

*Corresponding author
Email addresses: rein.houthooft@ugent.be (Rein Houthooft), filip.deturck@ugent.be (Filip De Turck)

2. Related work

Although the combination of neural networks and structured or probabilistic graphical models dates back to the early '90s [3, 4], interest in this topic is resurging. Several recent works introduce nonlinear unary factors/potentials into structured models. For the task of image segmentation, Chen et al. [5] train a convolutional neural network as a unary classifier, followed by the training of a dense random field over the input pixels. Similarly, Farabet et al. [6] combine the output maps of a convolutional network with a CRF for image segmentation, while Li and Zemel [7] propose semisupervised maxmargin learning with nonlinear unary potentials. Contrary to these works, we trade the bifurcated training approach for integrated inference and training of unary and interactions factors. Several works [8, 9, 10, 11] focus on linear-chain graphs, using an independently trained deep learning model whose output serves as unary input features. Contrary to these works, we focus on more general graphs. Other works suggest kernels towards nonlinear SSVMs [12, 13]; we approach nonlinearity by representing SSVM factors by arbitrarily deep neural networks.

Do and Artières [14] propose a CRF in which potentials are represented by multilayer networks. The performance of their linear-chain probabilistic model is demonstrated by optical character and speech recognition using two-hidden-layer neural network outputs as unary potentials. Furthermore, joint inference and learning in linear-chain models is also proposed by Peng et al. [15], however, the application to more general graphs remains an open problem [16]. Contrary to these works, we popose a nonprobabilistic approach for general graphs by also modeling nonlinear interaction factors. More recently, Schwing and Urtasun [17] train a convolutional network as a unary classifier jointly with a fully-connected CRF for the task of image segmentation, similar to [18, 19]. Chen et al. [20] advocate a joint learning and reasoning approach, in which a structured model is probabilistically trained using loopy belief propagation for the task of optical character recognition and image tagging. Other related work includes Domke [21] who uses relaxations for combined message-passing and learning.

Other related work aiming to improve conventional SSVMs are the works of Wang et al. [22] and Lin et al. [23], in which a hierarchical part-based model is proposed for multiclass object recognition and shape detection, focusing on model reconfigurability through compositional alternatives in And-Or graphs. Liang et al. [24] propose the use of convolutional neural networks to model an end-to-end relation between input images and structured outputs in active template regression. Xu et al. [25] propose the learning of a structured model with multilayer deformable parts for action understanding, while Lu et al. [26] propose a hierarchical structured model for action segmentation.

Many of these works use probabilistic models that maximize the negative log-likelihood, such as [14, 15]. In contrast, this paper takes a nonprobabilistic approach, wherein an SSVM is optimized via subgradient descent. The algorithm is altered to back-propagate SSVM loss errors, based on the ground truth and a loss-augmented prediction into the factors. Moreover, all factors are nonlinear functions, allowing the learning of complex patterns that originate from interaction features.

3. Methodology

In this section, essential SSVM background is introduced, after which integrated inference and back-propagation is explained for nonlinear unary factors. Finally, this notion is generalized into an SSVM model using only neural factors which are optimized by an alteration of subgradient descent.

3.1. Background

Traditional classification models are based on a prediction function $f : \mathcal{X} \to \mathbb{R}$ that outputs a scalar. In contrast, structured prediction models define a prediction function $f : \mathcal{X} \to \mathcal{Y}$, whose output can be arbitrarily structured. In this paper, this structure is represented by a vector in $\mathcal{Y} = \mathcal{L}^n$, with $\mathcal{L} \subset \mathbb{N}$ a set of class labels. Structured models employ a compatibility function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, parametrized by $w \in \mathbb{R}^D$. Prediction is done by solving the following maximization problem:

$$f(x) = \arg\max_{y \in \mathcal{Y}} g(x, y; w). \tag{1}$$

This is called inference, i.e., obtaining the most-likely assignment of labels, which is similar to maximum-a-posteriori (MAP) inference in probabilistic models. Because of the combinatorial complexity of the output space $\mathcal{Y}$, the maximization problem in Eq. (1) is NP-hard [20]. Hence, it is important to impose on g some kind of regularity that can be exploited for inference. This can be done by ensuring that g corresponds to a nonprobabilistic factor graph, for which efficient inference techniques exist [1]. In general, g is linearly parametrized as a product of a weight vector w and a joint feature function $\varphi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$.

Commonly, g decomposes as a sum of unary and interaction factors[1], in which $\varphi = [(\varphi_U)^\top, (\varphi_I)^\top]^\top$. The functions φ_U and φ_I are then sums over all individual joint input-output features of the nodes $\psi_i(y, x)$ and interactions $\psi_{ij}(y, x)$ of the corresponding factor graph [1, 12]. For example in the use case of Section 4, nodes are image regions, while interactions are connections between regions, each with their own joint feature vector. Data samples (x, y) are conform this graphical structure, i.e., x is composed of unary features x^U and interaction features

[1] Maximizing g corresponds to minimizing the state of a nonprobabilistic factor graph, which factorizes into a product of factors. However, by operating in the log-domain, the state decomposes as a sum of factors.

x^I. Moreover, the unary and interaction parameters are generally concatenated as $w = [(w_U)^\top, (w_I)^\top]^\top$.

In this formulation, the unary features are defined as

$$\psi_i(y_i, x_i) = \left(\epsilon_i(x)^\top [y_i = m]\right)^\top_{(m \in \mathcal{L})}, \quad (2)$$

while the interaction features for 2nd-order (edges) interactions are defined as

$$\psi_{ij}(y_i, y_j) = (\xi_{ij}(x)[y_i = m \wedge y_j = n])^\top_{((m,n) \in \mathcal{L}^2)}, \quad (3)$$

with $\epsilon_i(x)$ the unary features corresponding to node i and $\xi_{ij}(x)$ the interaction features corresponding to interaction (edge) (i, j). Similarly, higher-order interaction features can be incorporated by extending this matrix into higher-order combinations of nodes, according to the interactions. In the experiments of this paper, unary features are bag-of-words features corresponding to each superpixel. Interaction features are also bag-of-words, but this time corresponding to all connected superpixels.

In an SSVM the compatibility function is linearly parametrized as $g(x, y; w) = \langle w, \varphi(x, y)\rangle$ and optimized effectively by minimizing an empirical estimate of the regularized structured risk

$$R(w) + \frac{\lambda}{N} \sum_{n=1}^{N} \Delta\left(y^n, f(x^n)\right), \quad (4)$$

with $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a structured loss function for which holds $\forall y, y' \in \mathcal{Y} : \Delta(y, y') \geq 0$, $\Delta(y, y) = 0$, and $\Delta(y', y) = \Delta(y, y')$; R a regularization function; λ the inverse of the regularization strength; for a set of N training samples $\{(x^n, y^n)\}_{n \in \{1,\ldots,N\}} \subset \mathcal{X} \times \mathcal{Y}$ that can be decomposed into V_n nodes and E_n interactions. In this paper, we make use of L_2-regularization, hence $R(w) = \frac{1}{2}\|w\|^2$. Furthermore, in line with our image segmentation use case in Section 4, the loss function is the class-weighted Hamming distance between two label assignments, or

$$\Delta(y^n, y) = \sum_{i=1}^{V_n} \eta(y_i^n)[y_i^n \neq y_i], \quad (5)$$

with $[\cdot]$ the Iverson brackets and V_n the number of nodes (i.e., inputs to the unary factors, which corresponds to the number of nodes in the underlying factor graph) in the n-th training sample. Contrary to maximum likelihood approaches [14, 20, 19], the Hamming distance allows us to directly maximize performance metrics regarding accuracy. By setting $\eta(y_i^n) = 1$ we can focus on node-wise accuracy, while setting $\eta(y_i^n) = (\sum_{i,n}[y_i^n = y_i])^{-1}$ allows us to focus on class-mean accuracy.

Due to the piecewise nature of the loss function Δ, traditional gradient-based optimization techniques are ineffective for solving Eq. (4). However, according to Zhang [27], the equations

$$L(w) = \frac{1}{2}\|w\|^2 + \frac{\lambda}{N} \sum_{n=1}^{N} \max\{\ell(x^n, y^n; w), 0\}, \text{ with} \quad (6)$$

$$\ell(x^n, y^n; w) = \max_{y \in \mathcal{Y}} [\Delta(y^n, y) - g(x^n, y^n; w) + g(x^n, y; w)], \quad (7)$$

define a continuous and convex upper bound for the actual structured risk in Eq. (4) that can be minimized effectively by solving $\arg\min_{w \in \mathbb{R}^D} L(w)$ through numerical optimization [1, 27].

3.2. Integrated back-propagation and inference

Traditional SSVM training methods optimize a joint parameter vector of the unary and interaction factors. However, they restrict these parameters to linear combinations of input features, or allow limited nonlinearity through the addition of kernels. The objective function in case of arbitrary nonlinear factors is often hard to optimize, as many numerical optimization methods require a convex objective function formulation. For example, N-slack cutting plane training requires the conversion of the max-operation in Eq. (7) to a set of $N|\mathcal{Y}|$ linear constraints for its quadratic programming procedure [29]; block-coordinate Frank-Wolfe SSVM optimization [30] assumes linear input dependencies; the structured perceptron similarly assumes linear parametrization [31]; and dual coordinate descent focuses on solving the dual of the linear L_2-loss in SSVMs [32].

Subgradient descent minimization, as described in [1, 33], is a flexible tool for optimizing Eq. (6) as it naturally allows error back-propagation. This algorithm alternates between two steps. First,

$$z^n = \arg\max_{y \in \mathcal{Y}} [\Delta(y^n, y) + \langle w, \varphi(x^n, y)\rangle] \quad (8)$$

is calculated for all N training samples, which is called the loss-augmented inference or prediction step, derived from Eq. (7). In this paper, general inference for determining Eq. (1) is approximated via the α-expansion [34] algorithm, whose effectiveness has been validated through extensive experiments [35]. Loss-augmented prediction as in Eq. (8) is incorporated into this procedure by adding the loss term $\eta(y_i^n)[y_i^n \neq y_i]$ to the unary factors.

Second, these z-values are used to calculate a subgradient[2] of Eq. (6) as $\frac{1}{N}[w + \lambda\left(\varphi\left(x^n, z^n\right) - \varphi(x^n, y^n)\right)]$ for each sample (x^n, y^n), in order to update w. Traditional SSVMs assume that $g(x, y; w) = \langle w, \varphi(x, y)\rangle$ in which φ is a predefined joint input-output feature function. Commonly, this joint function is made up of the outputs of a nonlinear 'unary' classifier $C : \mathcal{X} \to [0, 1]^{|\mathcal{L}|}$, such that $\varphi_U(x, y)$ becomes $\varphi_U(C(x), y)$ [36]. This classifier is trained upfront, based on the different unary inputs corresponding to each node in the underlying factor graph. Due to the linear definition of g, the SSVM model is learning linear combinations of these classifier outputs as its unary factors. In

[2] $v \in \mathbb{R}^D$ is a subgradient of $f : \mathbb{R}^D \to \mathbb{R}$ in a point p_0 if $f(p) - f(p_0) \geq \langle v, p - p_0\rangle$. Due to its piecewise continuous nature, Eq. (6) is nondifferentiable in some points, hence we are forced to rely on subgradients.

Algorithm 1: Integrated SSVM subgradient descent with neural unary and linear interaction factors

Input: # iterations T; learning rate curve $\mu/(t_0+t)$; inverse regularization strength λ; training samples $\{(x^n, y^n)\}_{n\in\{1,\dots,N\}}$

Output: optimized parameters $\theta \in \mathbb{R}^K$ and $w \in \mathbb{R}^L$

Initialize w to $\vec{0}$ and θ according to [28]; the output layer weights are initialized to 0.
for $1 \le t \le T$ **do**
  for $1 \le n \le N$ **do**
    $z^n \leftarrow \arg\max_{y\in\mathcal{Y}}[\Delta(y^n, y) + \langle w, \varphi_I(x^n, y)\rangle + f(x^n, y; \theta)]$ `// loss-augmented prediction in Eq.` (8)
    if $\Delta(y^n, y) - g(x^n, y^n; \theta, w) + g(x^n, z^n; \theta, w) > 0$ **then** `// max-operation in Eq.` (6)
      $\frac{\partial L_n}{\partial w}(\theta, w) \leftarrow w + \lambda\big(\varphi_I(x^n, z^n) - \varphi_I(x^n, y^n)\big)$ `// standard SSVM subgradient computation [1]`
      $\nabla_\theta L_n(\theta, w) \leftarrow \theta + \lambda\big(\nabla_\theta f(x^n, z^n; \theta) - \nabla_\theta f(x^n, y^n; \theta)\big)$ `// gradient computation as in Eq.` (10)
    else
      $\nabla_\theta L_n(\theta, w) \leftarrow \theta$ and $\frac{\partial L_n}{\partial w}(\theta, w) \leftarrow w$
    end
  end
  $w \leftarrow w - \frac{\mu}{t_0+t}\frac{1}{N}\sum_{n=1}^{N}\frac{\partial L_n}{\partial w}(\theta, w)$ `// update linear interaction factors`
  $\theta \leftarrow \texttt{backprop}\left(\frac{1}{N}\sum_{n=1}^{N}\nabla_\theta L_n(\theta, w)\right)$ `// update neural unary factors via back-propagation`
end

general, the interaction factors are not trained through a separate classifier, and are thus linear combinations of the interaction features directly.

We propose to replace the pretraining of a nonlinear unary classifier, and the transformation of its outputs through linear factors, by the direct optimization of nonlinear unary factors. In particular, the unary part of g is represented by a sum f of outputs of an adapted neural network which models factor values. To achieve this, the loss-augmented prediction step defined in Eq. (8) is altered to

$$z^n = \arg\max_{y\in\mathcal{Y}}[\Delta(y^n, y) + \langle w, \varphi_I(x^n, y)\rangle + f(x^n, y; \theta)], \quad (9)$$

in which φ_I represents the joint interaction feature function as described in Section 3.1 and Eq. (3). Eq. (9) is calculated similarly to Eq. (8) through α-expansion by encoding the loss term into the unary factors.

The compatibility function thus becomes $g(x, y; \theta, w) = \langle w, \varphi_I(x, y)\rangle + f(x, y; \theta)$. The calculation of $\frac{\partial L}{\partial w}$, originally defined as the subderivative of the objective function in Eq. (6), remains unaltered. However, we can no longer assume that $\frac{\partial L}{\partial \theta}$ conforms to the definition of a subgradient due to its nonconvexity. However, we can calculate

$$\nabla_\theta L(\theta, w) = \theta + \frac{\lambda}{N}\sum_{n\in\mathcal{N}}\left(\nabla_\theta f\left(x^n, z^n; \theta\right) - \nabla_\theta f\left(x^n, y^n; \theta\right)\right), \quad (10)$$

with $\mathcal{N}$ the set of indices corresponding to training samples for which $\ell(x^n, y^n; w) > 0$ in Eq. (7), for a particular loss-augmented prediction z^n. In case $\ell(x^n, y^n; w) = 0$, we set $\nabla_\theta L = \theta$. This gradient incorporates the loss-augmented prediction of Eq. (9) and is back-propagated through the underlying network to adjust each element of θ. The altered subgradient descent method is shown in Algorithm 1. Herein, L_n represents the objective function for the n-th training sample, i.e., $L_n(\theta, w) = \frac{1}{N}[\frac{1}{2}\|w\|^2 + \lambda(\Delta(y^n, z^n) - g(x^n, y^n; \theta, w) + g(x^n, z^n; \theta, w))]$.

In contrast to gradient descent, subgradient methods [1, 33] do not guarantee the lowering of the objective function value in each step. Therefore, the current best value $L_*^{(t)} = \min\{L_*^{(t-1)}, L(w^{(t)})\}$ is memorized in each iteration t, along with the corresponding parameter values (w_*, θ_*). As such, the objective value L_* decreases at each step as $L_*^{(t)} = \min\{L(w^{(1)}), \dots, L(w^{(t)})\}$. This update rule is omitted from Algorithm 1 to improve readability.

Because the loss terms in Eq. (7) are no longer affine input transformations due to the introduced nonlinearities of the neural network, we can no longer assume Eq. (6) to be convex, as is the case for conventional SSVMs. Although theoretical guarantees can be made for the convergence of (sub)gradient methods for convex functions [37], and particular classes of nonconvex functions [38], no such guarantees can be made for arbitrary nonconvex functions [39]. The problem of optimizing highly nonconvex functions is studied extensively in neural network gradient descent literature. However, it has been demonstrated that nonconvex objectives can be minimized effectively due to the high dimensionality of the neural network parameter space [40]. Dauphin et al. [41] show that saddle points are

Algorithm 2: Integrated SSVM subgradient descent with both unary and interaction neural factors

Input: # iterations T; learning rate; inverse regularization strength λ; training set $\{(x^n, y^n)\}$
Output: optimized parameters $\theta \in \mathbb{R}^K$ and $\gamma \in \mathbb{R}^M$

Initialize θ and γ according to [28]; the weights of the output layers are initialized to 0.
for $1 \leq t \leq T$ **do**
  for $1 \leq n \leq N$ **do**
    $z^n \leftarrow \arg\max_{y \in \mathcal{Y}} [\Delta(y^n, y) + f(x^n, y; \theta) + h(x^n, y; \gamma)]$
    if $\Delta(y^n, y) - g(x^n, y^n; \theta, \gamma) + g(x^n, z^n; \theta, \gamma) > 0$ **then**
      $\nabla_\theta L_n(\theta, \gamma) \leftarrow \theta + \lambda\Big(\nabla_\theta f(x^n, z^n; \theta) - \nabla_\theta(x^n, y^n; \theta)\Big)$
      and $\nabla_\gamma L_n(\theta, \gamma) \leftarrow \gamma + \lambda\Big(\nabla_\gamma h(x^n, z^n; \gamma) - \nabla_\gamma h(x^n, y^n; \gamma)\Big)$
    else
      $\nabla_\theta L_n(\theta, \gamma) \leftarrow \theta$ and $\nabla_\gamma L_n(\theta, \gamma) \leftarrow \theta$
    end
  end
  $\theta \leftarrow \texttt{backprop}\left(\frac{1}{N}\sum_{n=1}^{N} \nabla_\theta L_n(\theta, \gamma)\right)$ and $\gamma \leftarrow \texttt{backprop}\left(\frac{1}{N}\sum_{n=1}^{N} \nabla_\gamma L_n(\theta, \gamma)\right)$
end

much likelier than local minima in multilayer neural network objective landscapes. In particular, the ratio of saddle points to local minima increases exponentially with the parameter dimensionality. Several methods exists to avoid these these saddle points, e.g., momentum [42]. Furthermore, Dauphin et al. [41] show, based on random matrix theory, that the existing local minima are very close to the global minimum of the objective function. This can be understood intuitively as the probability that all directions surrounding a local minimum lead upwards is very small, making local minima not an issue in general. The empirical results presented in Section 4.2 reinforce this believe by demonstrating that the regularized objective function can still be minimized effectively, as we achieve accurate predictions.

As described in Algorithm 1, the (sub)gradient is defined over whole data samples, which each consist of multiple nodes. f thus models the unary part of the compatibility function g, which is a sum of the V_n unary factors. Therefore, the function $f(x, y; \theta)$ decomposes as a sum of *neural unary factors*

$$f(x, y; \theta) = \sum_{i=1}^{V_n} f^*(x_i^U; \theta)_{y_i}, \tag{11}$$

with x^U the unary features in x. The nonlinear function $f^* : \mathcal{X} \to \mathbb{R}^{|\mathcal{L}|}$ is a multiclass multilayer neural network parametrized by $\theta \in \mathbb{R}^K$, whose inputs are features corresponding to the V_n different nodes. It forms a template for the neural unary factors. In this network $f^*(x_i^U; \theta)$, the softmax-function is removed from the output layer, such that it matches the unary factor range $\mathbb{R}^{|\mathcal{L}|}$. The argument y of the joint feature function is used as an index y_i to select a particular output unit.

3.3. Neural interaction factors

In this section we extend the notion of nonlinear factors beyond the integration of the training of a unary classifier. We now also replace the linear interaction part $\langle w, \varphi_I(x, y)\rangle$ of the compatibility function g with a function $h(x, y; \gamma)$ that decomposes as a sum of *neural interaction factors*

$$h(x, y; \gamma) = \sum_{i=1}^{E_n} h^*(x_i^I; \gamma)_{\mathcal{N}_i(y)}, \tag{12}$$

with x^I the interaction features in x, $\mathcal{N}_i(y)$ the combination of node labels in the i-th interaction, and E_n the number of interactions in the n-th training sample. The function $h^* : \mathcal{X} \to \mathbb{R}^{|\mathcal{L}|^Q}$ is parametrized by $\gamma \in \mathbb{R}^M$, and forms a template for the interaction factors. Herein, Q depends on the interaction order, e.g., $Q = 2$ in the Section 4 use case as connections between nodes are then edges. Interaction factors are generally not trained upfront. However, neural interaction factors are useful as they can extract complexer interaction patterns, and thus transcend the limited generalization power of linear combinations. In image segmentation for example, interaction features consisting of vertical gradients and a 90°-angle can indicate that the two connected nodes belong to the same class. The loss-augmented inference step in Eq. (9) is now adapted to

$$z^n = \arg\max_{y \in \mathcal{Y}} [\Delta(y^n, y) + f(x^n, y; \theta) + h(x^n, y; \gamma)], \tag{13}$$

while the compatibility function becomes $g(x, y; \theta, \gamma) = f(x, y; \theta) + h(x, y; \gamma)$. The two distinct models f and h are trained in a similar fashion to the method described in Algorithm 1, as depicted in Algorithm 2. Notice that this

method can easily be adjusted for batch or online learning by adapting and moving the weight updates at line 12 into the inner loop.

Like the unary function f^* in Eq. (11), $h^*(x_i^I;\gamma)$ is a multiclass multilayer neural network in which the top softmax-function is removed, shared among all E_n interaction factors. The output layer dimension matches the number of interaction label combinations, $|\mathcal{L}|^Q$ in the most general case. For example in image segmentation, for a problem with symmetric edge features, the number of output units in h^* is $\frac{1}{2}|\mathcal{L}|(|\mathcal{L}|+1)$, which all represent different states for a particular interaction factor (in this case the interactions are undirected edges, thus $\mathcal{N}_i(y)$ consists of the i-th edge's incident nodes).

The resulting structured predictor no longer requires two-phase training in which linear interaction factors are combined with the upfront training of a unary classifier, whose output is transformed linearly into unary factor values. It makes use of highly nonlinear functions for all SSVM factors, by way of multilayer neural networks, using an integration of loss-augmented inference and back-propagation in a subgradient descent framework. This allows the factors to generalize strongly while being able to mutually adapt to each other's parameter updates, leading to more accurate predictions.

4. Experiments

In this section, our model is analyzed on the task of image segmentation. Herein, the goal is to label different image regions with a correct class label. This is cast into a structured prediction problem by predicting all image region class labels simultaneously. There is one unary factor in underlying SSVM graphical structure for every image region, while interactions represent edges between neighboring regions. First, our model is analyzed and its different variants are compared to conventional SSVM training schemes. Second, the best performing variant is compared with state-of-the-art segmentation approaches. Our model is implemented as an extension of PyStruct [43], using Theano [44] for GPU-accelerated neural factor optimization.

4.1. Experimental setup

The model analysis experiments are executed on the widely-used MSRC-21 benchmark [45], which consists of 276 training, 59 validation, and 256 testing images. This benchmark is sufficiently complex with its 21 classes and noisy labels, and focuses on object delineation as well as irregular background recognition. Furthermore, the experiments are executed on the KITTI benchmark [46] consisting of 100 training and 46 testing images, augmented with 49 training images of Kundu et al. [47]. This latter benchmark consists of 11 classes, but we drop the 3 least frequently-occurring ones as they are insufficiently represented in the dataset. Finally, the same experiment is repeated for a larger dataset, namely the SIFT Flow benchmark [48], consisting of 33 classes with 2488 training and 200 testing images.

All image pixels are clustered into ± 300 regions using the SLIC [49] superpixel algorithm. For each region, gradient (DAISY [50]) and color (in HSV-space) features are densely extracted. These features are transformed two times into separate bags-of-words via minibatch k-means clustering (once 60 gradient and 30 color words, once 10 and 5 words). The unary input vectors form $(60+30)$-D concatenations of the first two bags-of-words. The model's connectivity structure links together all neighboring regions via edges. The edge/interaction input vectors are based on concatenations of the second set of bags-of-words. Both $(10+5)$-D input vectors of the edge's incident regions are concatenated into a $(2\times(10+5))$-D vector. Moreover, two edge-specific features are added, namely the distance and angle between adjacent superpixel centers, leading to $(2\times(10+5)+2)$-D interaction feature vectors.

Factors are trained with (regular) momentum, using a learning rate curve $\frac{\mu}{t_0+t}$, with μ and t_0 parameters, and t the current training iteration number as used in Algorithms 1 and 2. The regularization, learning rate, and momentum hyperparameter values are tuned using a validation set by means of a coarse- and fine-grained grid search over the parameter spaces, yielding separate settings for the unary and pairwise factors. The linear parameters w are initialized to 0, while the neural factor parameters θ and γ are initialized according to [28], except for the top layer weights which are set to 0. The class weights $\eta(y_i^n)$ in Eq. (5) are set to correct for class imbalance. The model is trained using CPU-parallelized loss-augmented prediction, while the neural factors are trained using GPU parallelism.

The following models are compared: unary-only (unary), N-slack cutting plane training (CP) with delayed constraint generation, subgradient descent (SGD)[3], integrated training with neural unary and linear interaction factors (int+lin), bifurcated training with neural interaction factors (bif+nrl), and integrated training with neural unary and neural interaction factors (int+nrl).

Multiclass logistic regression is used as unary classifier, trained with gradient descent by cross-entropy optimization. All unary neural factors contain a single hidden layer with 256 tanh-units, for direct comparison of integrated learning with upfront logistic regression training. The interaction neural factors contain a single hidden layer of 512 tanh-units to elucidate the benefit of nonlinear factors, without overly increasing the model's capacity. The experiment is set up to highlight the benefit of integrated learning by restricting the unary factors to features insufficiently discriminative on their own. This deliberately leads to noisy unary classification, forcing the model to rely on contextual relations for accurate prediction. The interaction factors encode information about their incident region

[3] SGD uses bifurcated training with linear interactions, hence it could be named bif+lin.

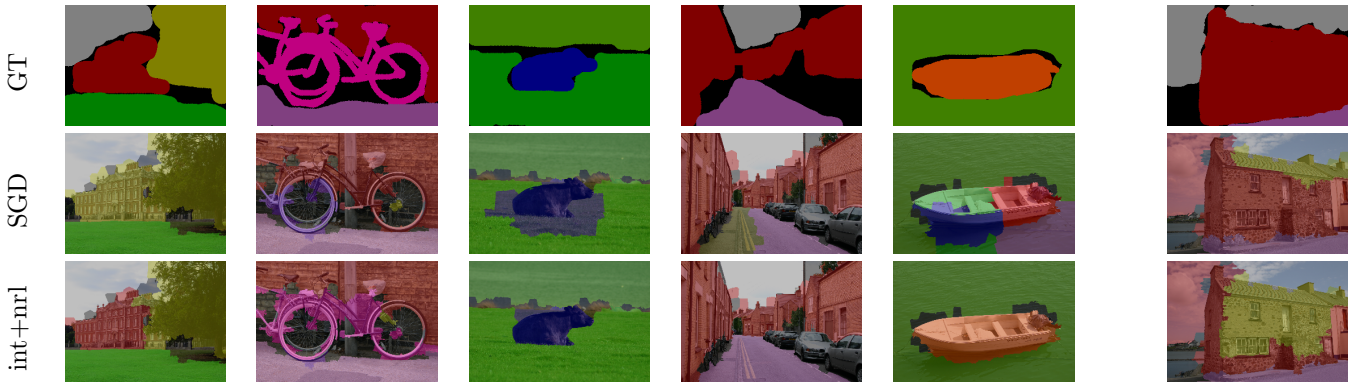

Figure 1: Illustrative examples of the performance of SGD and int+nrl on several MSRC-21 test images. Integrated training with neural factors improves classification accuracy over subgradient descent. The last column presents a case in which our model fails to outperform SGD.

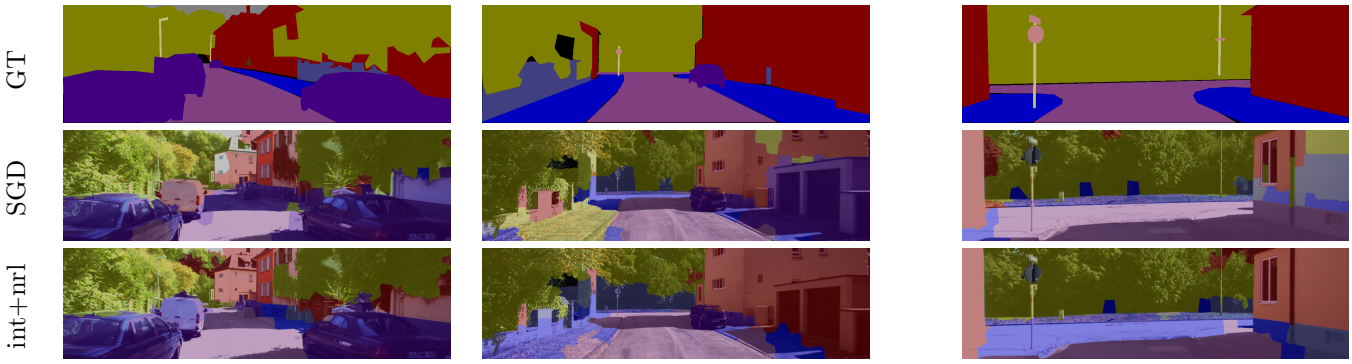

Figure 2: Illustrative examples of the performance of SGD and int+nrl on several KITTI test images. Integrated training with neural factors improves classification accuracy over subgradient descent. The last column presents a case in which our model fails to outperform SGD.

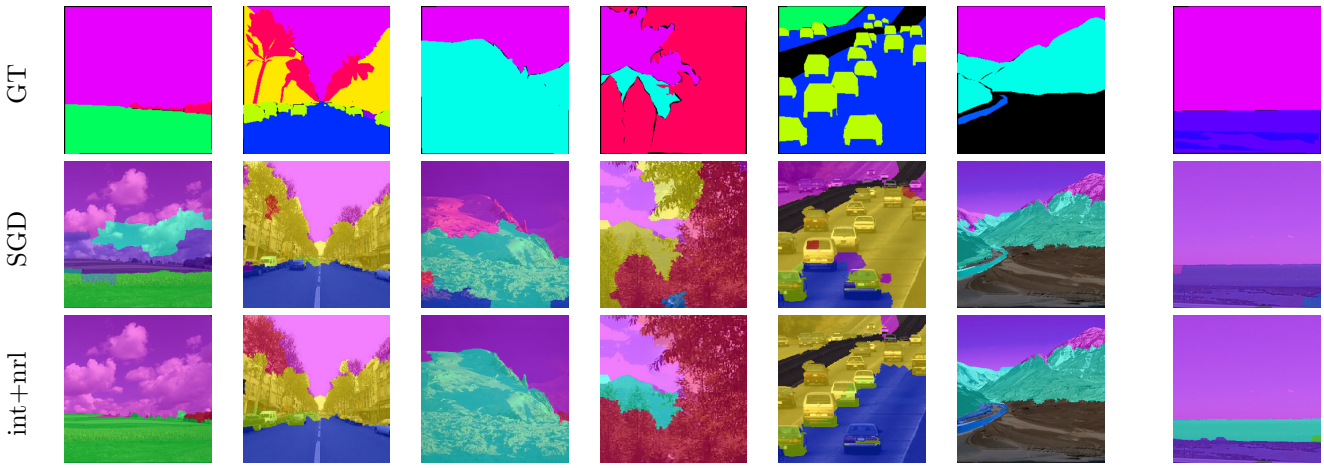

Figure 3: Illustrative examples of the performance of SGD and int+nrl on several SIFT Flow test images. Integrated training with neural factors improves classification accuracy over subgradient descent. The last column presents a case in which our model fails to outperform SGD.

Table 1: MSRC-21 class, pixel-wise, and class-mean test accuracy (in %) for different models

	building	grass	tree	cow	sheep	sky	aeropl.	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**pixel**	**class**
unary	15	60	52	8	10	68	35	46	12	21	21	42	9	2	36	0	21	14	5	6	1	36.3	23.1
CP	44	**77**	61	48	21	85	60	69	51	70	63	54	49	16	87	21	41	47	6	16	**33**	59.4	48.5
SGD	49	67	71	39	64	80	81	67	35	74	60	42	19	2	88	**51**	53	38	4	31	26	59.2	49.6
int+lin	48	76	83	67	73	94	78	**67**	59	56	68	65	48	14	**95**	43	61	53	6	45	32	67.4	58.5
bif+nrl	46	74	79	51	51	92	**83**	64	76	64	67	50	53	9	83	34	42	42	0	47	22	62.7	53.7
int+nrl	53	**77**	86	61	73	**95**	**83**	60	**87**	**77**	72	**69**	**77**	27	85	29	**67**	46	0	57	26	70.1	62.3
int†+lin	46	67	80	47	69	83	79	60	35	66	63	53	10	2	89	43	66	**62**	4	45	17	61.2	51.7
3-layer	**62**	76	**87**	**68**	**77**	94	81	66	84	65	**75**	53	69	**33**	81	**51**	**67**	58	**30**	**64**	25	**71.6**	**65.1**

Table 2: KITTI class, pixel-wise, and class-mean test accuracy (in %) for different models

	sky	building	road	sidewalk	fence	vegetation	pole	car	**pixel**	**class**
unary	75	63	59	29	8	71	0	38	53.8	42.8
CP	84	76	75	11	5	75	0	48	61.5	46.7
SGD	77	68	86	19	4	80	0	71	65.5	50.6
int+lin	86	76	82	42	23	81	**6**	67	70.2	57.8
bif+nrl	86	77	81	41	12	80	0	71	70.0	55.9
int+nrl	86	**83**	**88**	50	19	84	4	74	75.6	60.9
int†+lin	81	76	85	22	12	82	0	70	69.2	53.5
3-layer	**90**	82	**88**	**55**	**28**	**87**	1	**78**	**77.6**	**63.6**

Table 3: SIFT Flow pixel-wise and class-mean test accuracy (in %) for different models

	pixel	**class**
unary	44.7	7.5
CP	62.5	13.8
SGD	65.9	15.3
int+lin	70.3	16.2
bif+nrl	68.8	16.1
int+nrl	71.3	17.0
int†+lin	70.2	15.6
3-layer	**71.5**	**17.2**

feature vectors to allow neural factors to extract meaningful patterns from gradient/color combinations. We deliberately encoded less information in the interaction features, such that the model cannot solely rely on interaction factors for accurate and coherent predictions.

4.2. Results and discussion

Accuracy results on the MSRC-21 [45] test images are presented in Table 1, while Figure 1 shows a handful of illustrative examples that compare segmentations attained by SGD with int+nrl. The results of the same experiment for the KITTI benchmark [46], augmented with additional training images Kundu et al. [47], are shown in Table 2 and Figure 2. Qualitative results on the SIFT Flow [48] dataset are shown in Figure 3, while accuracy results are shown in Table 3.

The results show that unary-only prediction is very inaccurate (pixel-wise/class-mean accuracy of 36.3/23.1% for the MSRC-21 dataset, 53.8/42.8% for the KITTI dataset, and 44.7/7.5% for the SIFT Flow dataset). The reason for this is that unary features are not sufficiently distinctive to allow for differentiation between classes due to their low dimensionality. Accurate predictions are only possible by taking into account contextual output relations, demonstrated by the increased accuracy of CP (MSRC-21: 59.4/48.5%; KITTI: 61.5/46.7%; SIFT Flow: 62.5/13.8%) as well as SGD (MSRC-21: 59.2/49.6%; KITTI: 65.5/50.6%; SIFT Flow: 65.9/15.3%). These structured predictors learn linear relations between image regions, which allows them to correct errors originating from the underlying unary classifier. However, the unary factor's linear weights w have only limited capability for error correction in the opposite direction, due to the fact that the SSVM cannot alter the unary classifier parameters post-hoc.

Using an integrated training approach such as int+lin, in which the SSVM is trained end-to-end, improves accuracy (MSRC-21: 67.4/58.5%; KITTI: 70.2/57.8%; SIFT Flow: 70.2/15.6%) over the bifurcated procedures CP and SGD. Although neither the unary or interaction features are very distinctive, the integrated procedure updates parameters in such a way that both factor types have a unique discriminative focus. Their synergistic relationship ultimately results in higher accuracy. To bet-

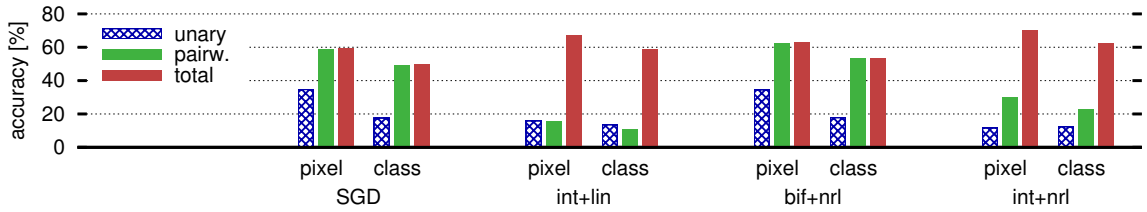

Figure 4: Visualization of the synergy between unary and interaction factors. In bifurcated training the interactions make unary factors redundant as these cannot be adapt to errors made by the interactions. In integrated training, combining both factor types leads to a higher accuracy as they can mutually adapt to each other's weight updates.

Table 4: State-of-the-art comparison: MSRC-21 per-class, class-mean, and global pixel-wise test accuracy (in %) for different models

	building	grass	tree	cow	sheep	sky	aeropl.	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**pixel**	**class**
neural factors	**76**	94	**94**	92	97	92	94	85	93	**88**	94	95	70	**78**	**97**	87	88	91	**78**	**88**	**63**	**88.9**	**87.4**
Liu et al. [51]	71	95	92	87	**98**	**97**	**97**	**89**	**95**	85	**96**	94	75	76	89	84	88	**97**	77	87	52	88.5	86.7
Yao et al. [52]	71	**98**	90	79	86	93	88	86	90	84	94	**98**	**76**	53	**97**	71	**89**	83	55	68	17	86.2	79.3
Lucchi et al. [53]	67	89	85	93	79	93	84	75	79	87	89	92	71	46	96	79	86	76	64	77	50	83.7	78.9
Munoz et al. [54]	63	93	88	84	65	89	69	78	74	81	84	80	51	55	84	80	69	47	59	71	24	78	71
Gonfause et al. [55]	60	78	77	91	68	88	87	76	73	77	93	97	73	57	95	81	76	81	46	56	46	77	75
Shotton et al. [56]	49	88	79	**97**	97	78	82	54	87	74	72	74	36	24	93	51	78	75	35	66	18	72	67
Lucchi et al. [57]	41	77	79	87	91	86	92	65	86	65	89	61	76	48	77	**91**	77	82	32	48	39	73	70

ter compare SGD (which uses 8, 21, and 33 logistic regression outputs as unary input features for the different benchmarks) with int+lin, we also depict the accuracy (MSRC-21: 61.2/51.7%; KITTI: 69.2/53.5%; SIFT Flow: 70.2/15.6%) of a model (int†+lin) with only 8, 21, and 33 unary hidden units for the KITTI, MSRC-21, and SIFT Flow dataset, rather than 256 units. The 2.0/2.1% (MSRC-21), 3.7/2.9% (KITTI), and 4.3/0.3% (SIFT Flow) increases in accuracy over SGD further illustrates the benefit of integrated learning and inference over conventional bifurcated SSVM training.

Another insight gained by the results is that accuracy increases when replacing linear interaction factors of conventional SSVMs with neural factors, i.e., int+nrl (MSRC-21: 70.1/62.3%; KITTI: 75.6/60.9%; SIFT Flow: 71.3/17.0%) and bif+nrl (MSRC-21: 62.7/53.7%; KITTI: 70.0/55.9%; SIFT Flow:68.8/16.1%) outperform int+lin (MSRC-21: 67.4/58.5%; KITTI: 70.2/57.8%; SIFT Flow: 70.3/16.2%) and SGD (MSRC-21: 59.2/49.6%; KITTI: 65.5/50.6%; SIFT Flow: 65.9/15.3%) respectively. This increase can be attributed to the higher number of parameters, as well as the added nonlinearities in combination with correct regularization. The model has greater generalization power, allowing the factors to extract more complex and meaningful interaction patterns. Neural factors offer great flexibility as they can be stacked to arbitrary depths. This leads to even higher generalization, as indicated by the increased accuracy (MSRC-21: 71.6/65.1%; KITTI: 77.6/63.6%; SIFT Flow: 71.5/17.2%) of the deeper 3-layer (int+nrl) model. Herein both unary and interaction factors are 3-hidden-layer neural networks consisting of 256 and 512 units (rectified linear units for MSRC-21 and KITTI and tanh units for SIFT Flow) in each layer respectively. Our model can thus easily be extended, for example by letting neural factors represent the fully-connected layer in convolutional neural networks. As such, it serves as a foundation for more complex structured models.

All methods converge within 600 epochs, with one epoch taking approximately 12.62 seconds for the MSRC-21 dataset, 4.35 seconds for the KITTI dataset, and 197.27 seconds on the SIFT Flow dataset for the int+nrl algorithm. The algorithm can be further optimized for speed by exploiting CPU parallelism and optimizing the data transfer between CPU and GPU when training the non-linear factors in our model. These optimizations will be part of our future work.

Figure 4 illustrates the synergy between unary and interaction factors achieved through both integrated and bifurcated training, exercised on the MSRC-21 dataset. The bars depict model test accuracy when using only unary or pairwise factors, by setting either the pairwise or unary factors respectively to a zero factor value, thus $\langle w, \varphi_I(x,y)\rangle$ or $\langle w, \varphi_U(x,y)\rangle = 0 \ \forall y \in \mathcal{Y}$. Although the unary factors alone perform well in bifurcated training, nearly all accuracy can be attributed to the interactions. A possible explanation is that both types essentially learn the same information. The interactions correct errors of

the underlying classifier and ultimately make unary factors redundant. In integrated training, neither the unary or interaction factors alone attain a high accuracy, but the combination of both does.

We explain this synergistic relationship with an example: Unary factors assign to a region of class A, a second-to-highest factor value to class A, a highest value to class B, and a low value to class C. The interactions also assign a second-to-highest value to class A, but a highest value to class C, and a low value to class B. Independently both factors incorrectly predict the region of class A as belonging to class B or class C. However, when combined they correctly assign a highest value to class A. In the figure, bifurcated training only shows limited signs of factor synergy, as the optimization procedure is insufficiently able to steer unary and pairwise parameters in different directions, which causes them have a similar discriminative focus. This observation leads us to believe that integrated learning and inference results in higher accuracy by synergistic unary/interaction factor optimization. Both factor types are no longer optimized for independent accuracy, but mutually adapt to each other's parameter updates, which results in enhanced predictive power.

In addition to the previous experiments, the viability of our neural factor model is shown through comparison with the closely related work of Liu et al. [51] on the MSRC-21 dataset. Liu et al. make use of features extracted from square regions of varying size around each superpixel, through means of a pretrained convolutional neural network. We compare our model with theirs by using overfeat features [58] in a similar fashion, trained on individual regions. Furthermore, the model settings have been altered with respect to the previous experiments. More specifically, 1,000 SLIC superpixels are utilized for the over-segmentation preprocessing step, enforcing superpixel connectivity and merging any superpixel with a surface area below a particular threshold. DAISY gradient and HSV color features are extracted according to a regular lattice, and clustered via minibatch k-means clustering. Next, the same type of features are extracted for each individual pixel, leading to unary and pairwise factor feature vectors. Moreover, the (x, y)-position of the superpixel (median-based) center is included in the unary feature vectors, while the distance and angle between the two superpixel centers is encoded into the interaction feature vectors. The neural factors are represented by multilayer neural networks using tanh-units, trained according to our Algorithm 2, using conventional momentum and single image-sized batches per gradient update. Classes are balanced by weighing them with the inverse of the class frequency. The results are presented in Table 4, which indicate that our model is capable of performing on par with the current state-of-practice, when used in conjunction with more advanced methods, e.g., overfeat features. Moreover, similar to Liu et al. [51], we have compared our model with other less closely related methods for completeness, for which the results are shown below the horizontal line in Table 4.

5. Conclusion

A structured prediction model that integrates back-propagation and loss-augmented inference into subgradient descent training of structural support vector machines (SSVMs) is proposed. This model departs from the traditional bifurcated approach in which a unary classifier is trained independently from the structured predictor. Furthermore, the SSVM factors are extended to neural factors, which allows both unary and interaction factors to be highly nonlinear functions of input features. Results on a complex image segmentation task show that end-to-end SSVM training, and/or using neural factors, leads to more accurate predictions than conventional subgradient descent and N-slack cutting plane training. Results show that our model serves as a foundation for more advanced structured models, e.g., by using latent variables, learned feature representations, or complexer connectivity structures.

Acknowledgments

Rein Houthooft is supported by a Ph.D. Fellowship of the Research Foundation - Flanders (FWO). Many thanks to Brecht Hanssens and Cedric De Boom for their insightful comments.

References

[1] S. Nowozin, C. H. Lampert, Structured learning and prediction in computer vision, Foundations and Trends in Computer Graphics and Vision 6 (3–4) (2011) 185–365.

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

[3] L. Bottou, Y. Bengio, Y. LeCun, Global training of document processing systems using graph transformer networks, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997, pp. 489–494.

[4] J. S. Bridle, Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters, in: Advances in Neural Information Processing Systems (NIPS), 1990, pp. 211–217.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, Semantic image segmentation with deep convolutional nets and fully connected CRFs, in: International Conference on Learning Representations, 2015.

[6] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8) (2013) 1915–1929.

[7] Y. Li, R. Zemel, High order regularization for semi-supervised learning of structured output problems, in: 31st Internactional Conference on Machine Learning, 2014, pp. 1368–1376.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of Machine Learning Research 12 (2011) 2493–2537.

[9] J. Morris, E. Fosler-Lussier, Conditional random fields for integrating local discriminative classifiers, IEEE/ACM Transactions on Audio, Speech, Language Processing 16 (3) (2008) 617–628.

[10] R. Prabhavalkar, E. Fosler-Lussier, Backpropagation training for multilayer conditional random field based phone recognition, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2010, pp. 5534–5537.
[11] D. Yu, L. Deng, S. Wang, Learning in the deep-structured conditional random fields, in: NIPS Workshop Deep Learning for Speech Recognition and Related Applications, 2009.
[12] A. Lucchi, Y. Li, K. Smith, P. Fua, Structured image segmentation using kernelized features, in: European Conference on Computer Vision, 2012, pp. 400–413.
[13] L. Bertelli, T. Yu, D. Vu, B. Gokturk, Kernelized structural SVM learning for supervised object segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 2153–2160.
[14] T.-M.-T. Do, T. Artières, Neural conditional random fields, in: International Conference on Artificial Intelligence and Statistics, 2010, pp. 177–184.
[15] J. Peng, L. Bo, J. Xu, Conditional neural fields, in: Adv. Neural Inf. Process. Syst., 2009, pp. 1419–1427.
[16] A. C. Müller, Methods for learning structured prediction in semantic segmentation of natural images, Ph.D. thesis, University of Bonn (2014).
[17] A. G. Schwing, R. Urtasun, Fully connected deep structured networks, arXiv preprint arXiv:1503.02351.
[18] J. J. Tompson, A. Jain, Y. LeCun, C. Bregler, Joint training of a convolutional network and a graphical model for human pose estimation, in: Advances in Neural Information Processing Systems, 2014, pp. 1799–1807.
[19] P. Krähenbühl, V. Koltun, Parameter learning and convergent inference for dense random fields, in: 30th International Conference on Machine Learning, 2013, pp. 513–521.
[20] L.-C. Chen, A. G. Schwing, A. L. Yuille, R. Urtasun, Learning deep structured models, in: 32nd International Conference on Machine Learning, 2015.
[21] J. Domke, Structured learning via logistic regression, in: Advances in Neural Information Processing Systems, 2013, pp. 647–655.
[22] X. Wang, L. Lin, L. Huang, S. Yan, Incorporating structural alternatives and sharing into hierarchy for multiclass object recognition and detection, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 3334–3341.
[23] L. Lin, X. Wang, W. Yang, J.-H. Lai, Discriminatively trained and-or graph models for object shape detection, Pattern Analysis and Machine Intelligence, IEEE Transactions on 37 (5) (2015) 959–972.
[24] X. Liang, S. Liu, X. Shen, J. Yang, L. Liu, L. Lin, S. Yan, Deep human parsing with active template regression, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (12).
[25] R. Xu, G. Chen, C. Xiong, W. Chen, J. J. Corso, Compositional structure learning for action understanding, arXiv preprint arXiv:1410.5861.
[26] J. Lu, R. Xu, J. J. Corso, Human action segmentation with hierarchical supervoxel consistency, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3762–3771.
[27] T. Zhang, Statistical behavior and consistency of classification methods based on convex risk minimization, Annals of Statistics 32 (1) (2004) 56–85.
[28] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: International Conference Artificial Intelligence and Statistics, 2010, pp. 249–256.
[29] T. Joachims, T. Finley, C.-N. J. Yu, Cutting-plane training of structural svms, Machine Learning 77 (1) (2009) 27–59.
[30] S. Lacoste-julien, M. Jaggi, M. Schmidt, P. Pletscher, Block-coordinate Frank-Wolfe optimization for structural SVMs, in: 30th International Conference on Machine Learning, 2013, pp. 53–61.
[31] M. Collins, Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms, in: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, Association for Computational Linguistics, 2002, pp. 1–8.
[32] M.-W. Chang, W.-t. Yih, Dual coordinate descent algorithms for efficient large margin structured prediction, Transactions of the Association for Computational Linguistics 1 (2013) 207–218.
[33] N. Z. Shor, K. C. Kiwiel, A. Ruszcaynski, Minimization methods for non-differentiable functions, Springer-Verlag New York, Inc., New York, NY, USA, 1985.
[34] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1222–1239.
[35] B. Peng, L. Zhang, D. Zhang, A survey of graph theoretical approaches to image segmentation, Pattern Recognition 46 (3) (2013) 1020 – 1038.
[36] R. Houthooft, C. De Boom, S. Verstichel, F. Ongenae, F. De Turck, Structured output prediction for semantic perception in autonomous vehicles, in: Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA, 2016.
[37] A. Nedić, D. Bertsekas, Convergence rate of incremental subgradient algorithms, in: Stochastic optimization: algorithms and applications, Springer, 2001, pp. 223–264.
[38] A. M. Bagirov, L. Jin, N. Karmitsa, A. Al Nuaimat, N. Sultanova, Subgradient method for nonconvex nonsmooth optimization, Journal of Optimization Theory and Applications 157 (2) (2013) 416–435.
[39] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, A. Y. Ng, On optimization methods for deep learning, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 265–272.
[40] R. Pascanu, Y. N. Dauphin, S. Ganguli, Y. Bengio, On the saddle point problem for non-convex optimization, arXiv preprint arXiv:1405.4604.
[41] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, in: Advances in Neural Information Processing Systems, 2014, pp. 2933–2941.
[42] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proceedings of the 30th international conference on machine learning (ICML-13), 2013, pp. 1139–1147.
[43] A. C. Müller, S. Behnke, PyStruct - Learning structured prediction in Python, Journal of Machine Learning Research 15 (2014) 2055–2060.
[44] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, Y. Bengio, Theano: new features and speed improvements, in: NIPS Workshop Deep Learning and Unsupervised Feature Learning, 2012.
[45] J. Shotton, J. Winn, C. Rother, A. Criminisi, Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context, International Journal of Computer Vision 81 (1) (2009) 2–23.
[46] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez, A. M. Lopez, Vision-based offline-online perception paradigm for autonomous driving, in: IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2015, pp. 231–238.
[47] A. Kundu, Y. Li, F. Dellaert, F. Li, J. Rehg, Joint semantic segmentation and 3d reconstruction from monocular video, in: European Conference on Computer Vision (ECCV), Vol. 8694, 2014, pp. 703–718.
[48] C. Liu, J. Yuen, A. Torralba, Sift flow: Dense correspondence across scenes and its applications, Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (5) (2011) 978–994.
[49] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Susstrunk, SLIC superpixels compared to state-of-the-art superpixel methods, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (11) (2012) 2274–2282.
[50] E. Tola, V. Lepetit, P. Fua, DAISY: An efficient dense descriptor applied to wide baseline stereo, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (5) (2010) 815–830.
[51] F. Liu, G. Lin, C. Shen, CRF learning with cnn features for

image segmentation, Pattern Recogn. 48 (10) (2015) 2983–2992.

[52] J. Yao, S. Fidler, R. Urtasun, Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 702–709.

[53] A. Lucchi, Y. Li, P. Fua, Learning for structured prediction using approximate subgradient descent with working sets, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 1987–1994.

[54] D. Munoz, J. A. Bagnell, M. Hebert, Stacked hierarchical labeling, in: Computer Vision–ECCV 2010, Springer, 2010, pp. 57–70.

[55] J. M. Gonfaus, X. Boix, J. Van de Weijer, A. D. Bagdanov, J. Serrat, J. Gonzalez, Harmony potentials for joint classification and segmentation, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 3280–3287.

[56] J. Shotton, M. Johnson, R. Cipolla, Semantic texton forests for image categorization and segmentation, in: Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[57] A. Lucchi, Y. Li, K. Smith, P. Fua, Structured image segmentation using kernelized features, in: Computer Vision–ECCV 2012, Springer, 2012, pp. 400–413.

[58] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: Integrated recognition, localization and detection using convolutional networks, arXiv preprint arXiv:1312.6229.