Learning Higher Order Potentials For MRFs

Dinesh Khandelwal
IIT Delhi

Parag Singla
IIT Delhi

Chetan Arora
IIIT Delhi

Abstract

Higher order MRF-MAP formulation has been shown to improve solutions in many popular computer vision problems. Most of these approaches have considered hand tuned clique potentials only. Over the last few years, while there has been steady improvement in inference techniques making it possible to perform tractable inference for clique sizes even up to few hundreds, the learning techniques for such clique potentials have been limited to clique size of merely 3 or 4. In this paper, we investigate learning of higher order clique potentials up to clique size of 16. We use structural support vector machine (SSVM), a large-margin learning framework, to learn higher order potential functions from data. It formulates the training problem as a quadratic programming problem (QP) that requires solving MAP inference problems in the inner iteration. We introduce multiple innovations in the formulation by introducing soft submodularity constraints which keep QP constraints manageable and at the same time makes MAP inference tractable. Unlike contemporary approaches to solving the original problem using the cutting plane technique, we propose to solve the problem using subgradient descent. This allows us to scale for problems with clique size even up to 16. We give indicative experiments to show the improvement gained in real applications using learned potentials instead of hand tuned ones.

1. Introduction

In recent years, researchers have increasingly focused their attention on higher order MRF-MAP formulations for their ability to encode interactions among a larger set of pixels [5, 12, 14–16, 23, 31]. However, in most of such works, clique potentials are hand tuned using the expert domain knowledge. This is due to the lack of efficient algorithms for learning the potentials from data.

While most of the earlier work [20, 27] on learning clique potentials in an MRF-MAP formulation has been around pairwise potentials only, a few have also suggested techniques for learning a restricted class of parameterized higher order potential functions [3, 11, 22]. Other notable works have used approximation techniques to learn such potentials [8, 17].

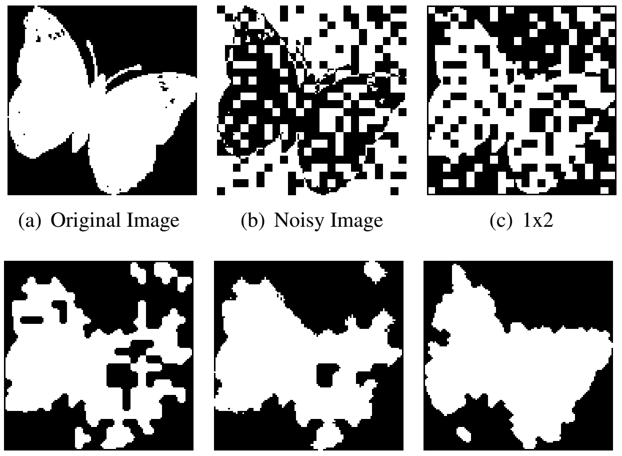
(a) Original Image (b) Noisy Image (c) 1x2 (d) 2x2 (e) 3x3 (f) 4x4

Figure 1: The focus of this paper is on learning higher order clique potentials in an MRF-MAP formulation. While the current state of the art has shown learning of such clique potentials upto size 4, we seek to enhance the tractability of learning upto size 16. The figure shows the improvement in denoising output for input image shown in (a), using learned potentials of various sizes: (c) clique size = 2, (d) clique size = 4,(e) clique size = 9, and (f) clique size = 16.

One of the principled ways to learn the parameters of a probabilistic graphical model is by using the structural support vector machine (SSVM), a margin maximizing framework. SSVM formulates the learning problem as a convex quadratic programming (QP) problem under the regularized risk minimization paradigm. The formulation minimizes an objective function consisting of L_2 norm of the parameters and surrogated hinge loss function. The number of constraints in such a formulation are typically exponential in the number of variables of the MRF.

Two popular approaches to solve the above QP are using the cutting plane algorithm [13] and subgradient descent based algorithms [20, 21]. Cutting plane approach relies on the fact that at the optimal solution of the QP, only a few constraints are active. Therefore, in each iteration, a small QP is solved, and a new most violated constraint is added. Finding such a constraint is done by solving a separation oracle to find a valid cutting plane, which is equivalent to performing the MAP inference.

On the other hand, subgradient based methods [21] convert the problem to an unconstrained optimization problem by incorporating the constraints directly into the objective function. In this scheme, the constraints do not need to be represented explicitly and hence, an exponential number of constraints can be easily dealt with. The transformation comes at the cost of losing differentiability, and therefore subgradients are used instead of gradients. Computing the subgradient amounts to solving a MAP inference problem.

As seen above, efficiently solving MAP inference task is crucial for parameter learning. However, MAP inference in graphical models with arbitrary potential function is an NP-hard problem in general. One of the ways to overcome it is by restricting the clique potentials to be submodular [9, 27, 32]. During parameter learning, submodularity is enforced by adding the corresponding hard linear constraints in the optimization problem. However, adding these constraints to the formulation creates scalability issues, since the number of submodularity constraints increases exponentially with the clique size. Understandably, existing algorithms for learning generic submodular potentials have reported results only upto clique size of 4 and have significant trouble scaling beyond a clique size of 9 [9].

Contributions: We present an algorithm to efficiently learn submodular higher order clique potentials upto size 16. Unlike state of the art [11, 23] we do not restrict the potentials in any other way. Our algorithm is based on the subgradient formulation, which includes submodularity constraints as soft constraints (with a very high penalty) in the objective function. While the steps for computing subgradient for the original max-margin constraints have been known, we show that subgradient for submodularity constraints depends on the constraints that are violated at the current weights. We suggest computing this subgradient by a linear scan through the submodularity constraints. Note that this does not require storing all the constraints in the memory. With number of constraints growing exponentially with clique size in the problem, this is crucial for tractability, and an important novelty, of the proposed algorithm. Our experiments on image denoising and object detection problem indicate that using clique potentials learnt from our algorithm can significantly outperform hand-tuned potentials in the state of the art.

2. Related Work

Two prominent methods to learn the clique potentials in an MRF-MAP problem are maximum likelihood and max-margin learning. In maximum likelihood approach, partition function needs to be computed which is intractable for loopy graphical models. Max-margin learning approach does not involve such computation and provides the flexibility to use different performance metrics, which is more suited for the task of MAP inference. We focus on the max-margin learning approach in this paper.

Max-margin [29] based approaches are typically referred to SSVM formulations because of their similarity to the margin-based framework in a SVM problem, albeit in the structured space. Associative Markov Networks [28] was one of the first approaches using max-margin learning. It solved the QP for a particular class of pairwise potentials called associative potential for which the number of margin based constraints could be reduced from exponential to polynomial.

Cutting plane based methods [13] instead solve a small QP by iteratively adding the most violated constraint based on the weights learned in the last iteration. The approach works by exploiting the fact that only a small number of constraints need to be present in the memory to obtain the optimal solution [13]. For finding the most violated constraint, one needs to perform loss-augmented inference. Szummer *et al.* [27] used graph cut inference for the task to learn a restricted form of submodular pairwise potentials.

In subgradient based methods [20, 21], the QP is solved by converting the problem into an unconstrained optimization one. Computing the sub-gradient again requires solving the MAP inference task. Luchi et al. [20] proposed a method which approximates the subgradient by using a working set of most violated constraints as opposed to the last violated constraint. They also avoid expensive loss-augmented inference by random sampling but lose the optimality guarantees of SSVM. None of the above approaches have tried to learn the potentials beyond standard pairwise potentials.

Among the few works focusing on learning higher order potentials, Rother *et al.* [23] learn sparse higher order potentials from data. Sparsity helps in reducing the computation effort during learning and inference but limits the expressive power of the potential function significantly since any configuration not seen during training is given an infinite cost. Gould *et al.* [11] have suggested learning a class of potentials called lower envelope potentials. Since it is a severely restricted subset of general submodular potentials, the applicability of such potentials can be somewhat limited. Fix *et al.* [9] propose a method to learn general higher order submodular potential functions by adding linear submodularity constraints in the QP. Each iteration of cutting plane now solves a QP with all the violated constraints added in the previous iterations and extra submodularity constraints. Fix *et al.* [9] also proposed a higher order inference algorithm to find the most violated constraint. This method does not scale beyond clique size 9 as the number of submodularity constraints increases exponentially in clique size, making the QP solving extremely slow.

In this paper, we use max-margin framework SSVM for learning potentials in a higher order MRF problem. Our method brings the submodularity constraints in the objective function and solves the QP by subgradient descent method. This helps us in scaling the learning algorithm up to clique size 16 and to calculate the subgradient using the recently proposed highly efficient higher order inference algorithm SoS-MinNorm [25].

3. Background

We focus on the problem of learning parameters of a Markov Random Field where $\mathcal{X} = \{1, \cdots, 255\}^N$ denotes an input space and $\mathcal{Y} = \{0, 1\}^N$ denotes a binary structured output space. N denotes the number of variables in the problem. Let $\mathcal{C}$ denote the set of cliques in the problem and $\Psi = \{\psi_\mathbf{c}\}_{\mathbf{c}\in\mathcal{C}}$ the set of potential functions over these cliques. The energy of a configuration, $(\mathbf{x}, \mathbf{y})$, in a MRF is given as:

$$E(\mathbf{x}, \mathbf{y}; \Psi) = \sum_{\mathbf{c}\in\mathcal{C}} \psi_\mathbf{c}(\mathbf{x_c}, \mathbf{y_c}) \tag{1}$$

Here, $\mathbf{x_c}$ and $\mathbf{y_c}$ denote a specific input and output configuration over a clique $\mathbf{c}$. In an alternate formulation, one can use a feature function, $f_\mathbf{c}(\mathbf{x_c}, \mathbf{y_c})$, and an associated weight vector $\mathbf{w_c}$, both of size $2^{|c|}$, and assume the energy expression of the form:

$$E(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{\mathbf{c}\in\mathcal{C}} \mathbf{w_c}^T f_\mathbf{c}(\mathbf{x_c}, \mathbf{y_c}). \tag{2}$$

Representing the $\mathbf{w_c}$'s and $f_\mathbf{c}$'s by single vectors $\mathbf{w}$'s and f's, the energy can be written as $E(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^T f(\mathbf{x}, \mathbf{y})$. The MAP inference problem corresponds to finding the minimum energy state $\mathbf{y}^*$ given the input configuration $\mathbf{x}$ such that

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} E(\mathbf{x}, \mathbf{y}; \mathbf{w}) \tag{3}$$

3.1. Submodularity

MAP inference problem defined in Equation 3 is intractable in general but can be solved in polynomial time if the clique potentials are submodular. Since our output label space is restricted to the set $\{0, 1\}$, the potentials $\psi_\mathbf{c}(\mathbf{x}, \mathbf{y})$ can be treated as functions defined over sets, with labels 0 and 1 indicating the exclusion and inclusion in the set respectively. Submodularity over a set function is defined as:

Definition 3.1. *A set function $\psi : 2^{|S|} \to R$ defined on a set S is submodular if for all $X \subseteq S$, $a, b \in S$ and $a, b \notin X$, ψ satisfies the following property: $\psi(X \cup \{a\}) - \psi(X) \geq \psi(X \cup \{a, b\}) - \psi(X \cup \{b\})$*

Since we deal with binary valued $f_\mathbf{c}$'s, submodularity of $\psi_\mathbf{c}(\mathbf{x}, \mathbf{y})$ can be equivalently expressed in terms of the corresponding weight vector $\mathbf{w_c}$ as: $\mathbf{w}_{X\cup\{a\}} - \mathbf{w}_X \geq \mathbf{w}_{X\cup\{a\}\cup\{b\}} - \mathbf{w}_{X\cup\{b\}}$. Here, X denotes a configuration of clique $\mathbf{c}$ and $\mathbf{w}_X$ denotes the component of $\mathbf{w}$ corresponding to configuration X.

3.2. Structured SVMs

Structured Support Vector Machines (SSVMs) [13] generalize the standard Support Vector Machines (SVMs) for the case of structured output spaces. Let $\{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^n$ denote a set of n training examples. The goal of parameter learning in max-margin framework is to learn the parameters $\mathbf{w_c}$'s such that $E(\mathbf{x}^i, \mathbf{y}^i)$ is smaller than $E(\mathbf{x}^i, \mathbf{y})$ $\forall \mathbf{y} \neq \mathbf{y}^i$ (by a margin), for each example $(\mathbf{x}^i, \mathbf{y}^i)$. The optimization problem defined by SSVMs can be written as a Quadratic Program (QP) described as follows:

$$\min_{\mathbf{w},\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^n \xi_i \tag{4}$$

$$\forall i \forall y \in \mathcal{Y} : E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) \geq E(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}) + \Delta(\mathbf{y}^i, \mathbf{y}) - \xi_i \tag{5}$$

$$\forall i, \xi_i \geq 0 \tag{6}$$

Here, $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ acts as a regularizer and ξ_i variables allow the margin constraints (Equation 5) to be violated by paying a penalty in the objective function.

Loss function $\Delta(\mathbf{y}^i, \mathbf{y})$ measures the distance of incorrect labeling $\mathbf{y}$ from the true labeling $\mathbf{y}^i$. Hamming distance is a standard loss metric used in these formulations. We assume that the loss function satisfy conditions $\Delta(\mathbf{y}, \mathbf{y}) = 0$ and $\Delta(\mathbf{y}^i, \mathbf{y}) \geq 0$ for $\mathbf{y} \neq \mathbf{y}^i$.

Consider a constraint in above QP for i^{th} training example and for labeling $\mathbf{y} = \mathbf{y}^i$. Since $E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) = E(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w})$, the constraint becomes $\Delta(\mathbf{y}^i, \mathbf{y}^i) - \xi_i \leq 0$. This implies $\xi_i \geq 0$ because $\Delta(\mathbf{y}^i, \mathbf{y}^i) = 0$. So the constraints $\xi_i \geq 0$ are already enforced by our assumptions on loss function, and hence can be gotten rid of.

There are exponential number of margin based constraints in the QP defined above. We can use a subgradient based approach to solve the above optimization problem.

3.3. Subgradient based Optimization

Equation 4 can be equivalently written as an unconstrained optimization problem by absorbing the constraints into the objective itself:

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^n \max_{y\in\mathcal{Y}}(\Delta(\mathbf{y}^i, \mathbf{y}) - E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) + E(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w})). \tag{7}$$

Note that we have gotten rid of $\xi_i \geq 0$ constraints as explained earlier. The objective function in the above equation is convex. But it is no longer differentiable due to the presence of the max function. A subgradient based optimization [6] generalizes the gradient based optimization for non-differentiable functions.

Definition 3.2. ***subgradient:*** *A vector v is subgradient of a convex function $h : \mathcal{R}^n \to R$ at w if $h(\mathbf{w}') \geq h(\mathbf{w}) + v^T(\mathbf{w}' - \mathbf{w})$, $\forall \mathbf{w}' \in \mathcal{R}^n$.*

Definition 3.3. ***ϵ approximate subgradient:*** *A vector v_ϵ is ϵ approximate subgradient of a convex function $h : \mathcal{R}^n \to R$ at w if $h(\mathbf{w}') \geq h(\mathbf{w}) + v_\epsilon^T(\mathbf{w}' - \mathbf{w}) - \epsilon, \forall \mathbf{w}' \in \mathcal{R}^n$.*

Consider a function, $h(\mathbf{w}) = \max_{j=1,\ldots,r} h_j(\mathbf{w})$, where each h_j is convex. Let k be any index for which $h(\mathbf{w}) = h_k(\mathbf{w})$, i.e., $k = \arg\max_{j=1,\ldots,r} h_j(\mathbf{w})$. Using the property that $\nabla h_k(\mathbf{w})$ is a subgradient for $h(\mathbf{w})$ [6], and $E(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^T f(\mathbf{x}, \mathbf{y})$, a valid subgradient $g(\mathbf{w})$ for the optimization problem in Equation 7 is given as:

$$g(\mathbf{w}) = \mathbf{w} + C \sum_{i=1}^{n} (f(\mathbf{x}^i, \mathbf{y}^i) - f(\mathbf{x}^i, \mathbf{y}^{*i})). \quad (8)$$

$$\text{Here} \quad \mathbf{y}^{*i} = \arg\max_{\mathbf{y} \in \mathcal{Y}} (\Delta(\mathbf{y}, \mathbf{y}^i) - E(\mathbf{x}^i, \mathbf{y}, \mathbf{w})) \quad (9)$$

$$= \arg\min_{\mathbf{y} \in \mathcal{Y}} (E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) - \Delta(\mathbf{y}, \mathbf{y}^i)). \quad (10)$$

We can get rid of the term $E(\mathbf{x}^i, \mathbf{y}^i; \mathbf{w})$ while computing the $\arg\min$ since it does not depend on $\mathbf{y} \in \mathcal{Y}$. Hence, $\mathbf{y}^{i*}$ can be obtained by solving a modified inference problem known as loss augmented inference, where $\Delta(\mathbf{y}^i, \mathbf{y})$ has been subtracted from the energy function. Assuming $\Delta(\mathbf{y}^i, \mathbf{y})$ decomposes over cliques in the graph [1], complexity of this step is same as that of MAP inference on the original problem.

The learning involves moving opposite to the subgradient in each step. The weight update equation is given as $\mathbf{w} \leftarrow \mathbf{w} - \eta * g(\mathbf{w})$, where η is the learning rate and $g(\mathbf{w})$ represents the subgradient at $\mathbf{w}$. Given a convex function $h(w)$, Bertsekas *et al.* [7] show that subgradient algorithm is guaranteed to converge to the optimal value $h(w^*)$ if dynamic step-sizes are chosen at each time step such that $\sum_{t=1}^{\infty} \eta_t = \infty$, and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$. Here, η_t denotes the step-size at iteration t. In fact, a stronger result holds. In each learning iteration, if we only have ϵ_t approximate subgradient available at time step t, where $\lim_{t \to \infty} \epsilon_t = \epsilon$, and we move along this direction, the algorithm is guaranteed to have ϵ *convergence* i.e., it will converge to a value $h(w'^*)$ such that $h(w'^*) \leq h(w^*) + \epsilon$. This guarantee will be useful for proving convergence of our learning algorithm when our MAP inference can only return a value which is within ϵ factor of the optimal.

[1] when Δ is the Hamming distance, it decompose over cliques

4. Our Approach

The focus of our work is on learning higher order clique potentials efficiently. We make the parameter tying assumption, i.e., the cliques of the same size in the image share the parameters $\mathbf{w_c}$'s with each other (see Section 3). The learning technique involves performing MAP inference in each iteration. To make the optimal inference possible in polynomial time we restrict our attention to learning submodular clique potentials only.

4.1. Learning with Submodularity Constraints

To enforce submodularity on the learnt potentials, we insert linear inequalities in terms of $\mathbf{w}$, as suggested by Definition (3.1), in the proposed quadratic program. For a clique of size $|c|$, it can be shown that, there are $2^{|\mathbf{c}|-3} * (|\mathbf{c}| * (|\mathbf{c}| - 1))$ such linear inequalities needed to enforce submodularity on the clique potential. This can be compactly written in the matrix form as $A\mathbf{w} \geq 0$, where each row of matrix A corresponds to a submodularity constraint. Each row has two entries as 1, two as -1, and the rest are 0. For example, for a clique of size 3, $\mathbf{w}$ is an 8 dimensional vector with a value for each of the labelings, such as $\{0,0,0\}, \{0,0,1\}, \cdots, \{1,1,1\}$, of the clique. A typical submodularity constraint is of the type: $\mathbf{w}_{\{1,0,0\}} - \mathbf{w}_{\{0,0,0\}} \geq \mathbf{w}_{\{1,1,0\}} - \mathbf{w}_{\{0,1,0\}}$. The row of the A matrix corresponding to this equation is the vector $[-1, 0, 1, 0, 1, 0, -1, 0, 0]$.

With submodularity constraints, the updated quadratic program of Equation 4 can be written as follows:

$$\min_{\mathbf{w}, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \xi_i \quad (11)$$

$$\forall i, \forall y \in \mathcal{Y} : E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) \geq E(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}) + \Delta(\mathbf{y}^i, \mathbf{y}) - \xi_i, \quad (12)$$

$$A\mathbf{w} \geq 0, \quad (13)$$

Fix *et al.* [9] have used the cutting plane technique to solve the above QP. They iteratively solve a smaller set of QPs refining the constraint set in each iteration by adding the most violated margin constraint based on the current set of weights. The form of these constraints are given by Equation 12). This is efficient since there exists a polynomial time separation oracle (by solving an inference problem) for adding such constraints, and only a small subset of constraints needs to be considered to solve the problem optimally [30].

Note that in the approach given by Fix *et al.*, all the submodularity constraints (Equation 13) are added to the constraint set in advance. This limits the applicability of their approach to smaller clique only when the number of submodularity constraints are small. The authors have shown results only up to cliques of size 4, and in our experiments, their approach could not scale beyond a clique size of 9.

In principle, it is possible to apply a separate cutting plane strategy on the submodularity constraints in the same way as done for margin constraints. However, in our experiments, we observe that a large number of such constraints are violated and brought in the QP, rendering the technique ineffective. Exploring this strategy further is the target of our future work.

4.2. Switching to Soft Submodularity Constraints

In this paper we suggest an alternative approach to solve the optimization problem by converting the hard submodularity constraints into soft ones. We introduce slack variables β_j and penalize violating such constraints in the objective function with high cost C_2. The updated quadratic program can be written as follows:

$$\min_{\mathbf{w},\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_1\sum_{i=1}^{n}\xi_i + C_2\sum_{j=1}^{|A|}\beta_j$$
$$\forall i, \forall y \in \mathcal{Y} : E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) \geq E(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}) + \Delta(\mathbf{y}^i, \mathbf{y}) - \xi_i,$$
$$Aw \geq -\beta; \beta \geq 0 \tag{14}$$

Here $|A|$ is the number of rows in matrix A and is equal to the total number of submodularity constraints. Note that, converting to soft constraint implies that the returned potentials may not always be submodular. This may have implications on the overall optimality of the solution. We defer the details to the next section.

With the change to soft constraints, we can now convert the problem to an unconstrained quadratic program, by bringing all the constraints in the objective function.

$$\min_{\mathbf{w}} \zeta(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$+ C_1\sum_{i=1}^{n}\max_{y\in\mathcal{Y}}(\Delta(\mathbf{y}^i, \mathbf{y}) - E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) + E(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}))$$
$$+ C_2\sum_{j=1}^{|A|}\max(0, -a_j\mathbf{w}) \tag{15}$$

Here a_j denotes the j^{th} row vector of the matrix A. It can be shown that a valid subgradient of $\zeta(\mathbf{w})$ is given as:

$$g = \mathbf{w} + C_1\sum_{i=1}^{n}(f(\mathbf{x}^i, \mathbf{y}^i) - f(\mathbf{x}^i, \mathbf{y}^{*i})) + C_2\sum_{j\in V_{\mathbf{w}}} -a_j^T$$

In the equation above, second and third terms on right are the contributions to the subgradient because of margin and submodularity constraints respectively. $\mathbf{y}^{*i}$ is obtained by solving the loss augmented inference problem and V_w is the set of violated submodular constraints at current parameter vector $\mathbf{w}$ and is given as: $V_{\mathbf{w}} = \{j : a_j\mathbf{w} < 0\}$.

Algorithm 1 Learning higher order potentials

```
Input: $S = (\mathbf{x}^1, \mathbf{y}^1), \dots, (\mathbf{x}^n, \mathbf{y}^n)$        // Training set
Input: $\eta$ (Learning rate), $T$ (Iterations), $\Delta$ (Loss function)
Output: $\mathbf{w}$                    // The learned parameter vector
 1: $\mathbf{w} = 0$;
 2: for $t \in \{1, \dots, T\}$ do;
 3:     $\mathbf{g_1} = 0$;
 4:     for $i \in \{1, \dots, n\}$ do;
 5:         $\mathbf{y}^{*i} = \arg\max_{\mathbf{y}\in\mathcal{Y}}(\Delta(\mathbf{y}, \mathbf{y}^i) - E(\mathbf{x}^i, \mathbf{y}, \mathbf{w}))$
    // perform loss augmented inference
 6:     end for
 7:     $\mathbf{g_1} = \sum_i f(\mathbf{x}^i, \mathbf{y}^i) - f(\mathbf{x}^i, \mathbf{y}^{*i})$;
    // subgradient component due to margin constraints
 8:     $\mathbf{g_2} = \sum_{j\in V_w} -a_i^T$;
    // subgradient component due to submodularity constraints
 9:     $\mathbf{w} = \mathbf{w} - \frac{\eta}{\sqrt{t}}(\mathbf{w} + C_1\mathbf{g_1} + C_2\mathbf{g_2})$;
    // parameter update
10: end for
11: return $\mathbf{w}$;                    // final weight vector
```

Unlike the earlier approach by Fix *et al.* [9], the proposed formulation does not require storing the whole A matrix in the memory. This is because the component of subgradient due to submodularity violation requires the set of violated submodularity constraints which can be computed on the fly. This is a huge advantage, since, as the size of the clique grows larger, the QP formulation as proposed in Equation 11 becomes increasingly difficult to store in the memory.

The complete details of our algorithm in the form of pseudocode have been given in Algorithm 1. We perform the loss augmented inference using the current set of parameters $\mathbf{w}$ in Line 5 which can be solved in polynomial time if the corresponding potentials are submodular [24]. Subgradient component corresponding to the margin constraints is computed on Line 7 and component due to violated submodularity constraints is computed in Line 8. Parameters are updated in Line 9.

4.3. Convergence Guarantees

Algorithm 1 is guaranteed to converge to the optimal if we can optimally solve the MAP inference task (Line 5). By enforcing submodularity of potentials, this can be done in polynomial time [4, 25]. But there is one caveat. Since in our formulation submodularity is modeled as soft constraints, our potentials may not be exactly submodular, implying that convergence guarantees may not hold. In this case, we analyze our algorithm using the notion of ϵ-convergence (see Section 3). Specifically, we show that if MAP inference returns a value within ϵ_t/n of the optimal (n being the number of examples, and t the iteration step), then we can compute the ϵ_t approximate subgradient in iteration t. This guarantees convergence of Algorithm 1 within ϵ of the optimal subject to $\lim_{t\to\infty} \epsilon_t = \epsilon$, as discussed in Section 3.3. First, we prove the following theorem:

Theorem 4.1. *Define $h(\mathbf{w}) = \max_{j=1,\dots,r} h_j(\mathbf{w})$, here each h_j is convex. Given a parameter vector $\mathbf{w}$, let k denote the index such that $h_k(\mathbf{w})$ is within ϵ of the maximum, i.e., $h_k(\mathbf{w}) \geq h(\mathbf{w}) - \epsilon$. Then, $v_\epsilon = \nabla_{\mathbf{w}} h_k(\mathbf{w})$ is ϵ approximate subgradient of $h(\mathbf{w})$.*

Proof. As v_ϵ is the (sub)gradient of $h_k(\mathbf{w})$ at point $\mathbf{w}$, we have

$$\forall \mathbf{w}', h_k(\mathbf{w}') \geq h_k(\mathbf{w}) + v_\epsilon^T(\mathbf{w}' - \mathbf{w}) \qquad (16)$$

Now $h_k(\mathbf{w}') \leq h(\mathbf{w}')$ by definition of $h(\mathbf{w}')$ and $h_k(\mathbf{w}) \geq h(\mathbf{w}) - \epsilon$ by the given condition. Substituting these in the above equation, we get:

$$\forall \mathbf{w}', h(\mathbf{w}') \geq h(\mathbf{w}) + v_\epsilon^T(\mathbf{w}' - \mathbf{w}) - \epsilon \qquad (17)$$

This is exactly the definition of ϵ approximate subgradient of $h(\mathbf{w})$ at $\mathbf{w}$. Hence, proved. □

At Line 5 in Algorithm 1, we perform the task of inference n times, n being the number of examples in each iterations. Therefore, if MAP inference algorithm returns an answer within ϵ_t/n of the optimal, the overall approximation factor with respect to the optimal is ϵ_t. This is because the objective involves sum of the energy terms from each example (see Equation 15). Note that we do not introduce any approximation for the terms corresponding to submodularity constraints. Finally, an ϵ_t approximation in the energy term of the objective will result in ϵ_t approximate subgradient using Theorem 4.1 in iteration t. If $\lim_{t\to\infty} \epsilon_t = \epsilon$ Algorithm 1 converges to an ϵ optimal solution. In our experiments, we have observed that after a few iterations of learning, there are very few violations in the submodularity constraints (both in terms of the number and the magnitude) indicating that the inference may already be close to the optimal.

Solving the proposed quadratic program requires performing a MAP inference for each computation of subgradient component due to margin constraints. If all the potential functions are submodular, the energy function in Equation 1 is a sum of submodular function which can be solved efficiently. Recently Shanu et al. [25] have proposed an algorithm SoS-MinNorm, an extension of famous Min Norm Point algorithm [10] to solve the sum of submodular minimization problem. It exploits the sum of submodular structure present in the MRF-MAP inference problems to efficiently optimize the objective for very large clique sizes (going up to 1000). Though there have been other algorithms [4, 15] to efficiently solve the MRF-MAP problem with submodular clique potentials, they become quite slow for clique sizes greater than 12. We have therefore used

Clique Size	Average Pixel Loss
1×2	30.30%
2×2	21.30%
3×3	15.62%
4×4	8.67%

Table 1: Average pixel loss on 10 testing images with increasing clique size. Image size is 120 × 120. We have used structured noise of patch size 4 × 4 and with flipping probability $p = 0.7$.

SoS-MinNorm to compute the subgradient in our algorithm. In future, we would like to analyze the effect of approximate submodularity of the potential functions on the ϵ optimality of our inference algorithm.

5. Experiments

All the experiments have been conducted on a computer with 3.4 GHz core $i7$ processor and 16 GB of RAM, running Ubuntu 16.04 operating system. We show results for the task of image denoising and object detection. In our notation, using a clique size of $H \times W$ implies including potentials over all the overlapping windows of size $H \times W$ with a stride of 1 in the image. We ran the proposed subgradient algorithm for 200 iterations in all the experiments.

We have experimented with different MAP inference algorithm like SoS-IBFS [9], Generic Cuts [5], Lazy Generic Cuts [15] to solve the loss-augmented inference problem and empirically conclude SoS-MinNorm [25] to scale best with increasing clique size both in terms of memory and time. We have used Sos-MinNorm in all the reported results. Our code is publicly available [2].

5.1. Image Denoising

For image denoising, we used binary image dataset [1] containing images of different objects like butterfly, elephant, crown, etc. We scale each image to a size of 120 × 120. For each object, we selected 20 images and randomly divided them into training and testing set having 10 images in each.

Implementation Details: Our MRF $G = (\mathcal{V}, \mathcal{C})$ has the following energy function:

$$\begin{aligned} E(\mathbf{x}, \mathbf{y}; \mathbf{w}) &= \sum_{v\in\mathcal{V}} \psi_v(\mathbf{x}_v, \mathbf{y}_v) + \sum_{\mathbf{c}\in\mathcal{C}} \psi_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}}) \qquad (18) \\ &= \mathbf{w}_u^T f^u(\mathbf{x}, \mathbf{y}) + \mathbf{w}_h^T f^h(\mathbf{y}). \end{aligned}$$

Here $\mathcal{C}$ denotes set of all cliques of having size greater than or equal to 2. $\psi_v(\mathbf{x}_v, \mathbf{y}_v)$ and $\psi_{\mathbf{c}}(\mathbf{y}_{\mathbf{c}})$ are the unary costs at pixel v and higher order clique potentials at clique $\mathbf{c}$ respectively. $f^u(\mathbf{x}, \mathbf{y})$ and $f^h(\mathbf{y})$ denotes the unary and higher

[2] `http://www.cse.iitd.ac.in/~khandelwal/publication/WACV/`

Clique Size	Patch Size	Average Pixel Loss
2×2	2×2	6.27%
2×2	3×3	12.50%
2×2	4×4	21.30%
3×3	2×2	6.01%
3×3	3×3	10.12%
3×3	4×4	15.62%
4×4	2×2	5.98%
4×4	3×3	10.05%
4×4	4×4	8.67%

Table 2: Average pixel loss on 10 testing images with varying clique size of learned potential and patch size of structured noise. Image size is 120×120 and flipping probability $p = 0.7$. We observe that as the patch size of noise increases, we need higher and higher order potentials to achieve a certain accuracy.

order feature vectors corresponding to unary and higher order potentials. The parameters to be learned are $\mathbf{w}_u$ and $\mathbf{w}_h$. Note that our our higher order potentials do not depend on the input values $\mathbf{x}$. Therefore, the functions $\psi_{\mathbf{c}}(\mathbf{y_c}, \mathbf{w_c})$ and $f^h(\mathbf{y})$ do not involve variable $\mathbf{x_c}$ (or $\mathbf{x}$) as an argument. We have selected our feature vectors as, $f^u(\mathbf{x}, \mathbf{y}) = \sum_{v \in \mathcal{V}} f^u_v(\mathbf{x}_v, \mathbf{y}_v)$, and $f^h(\mathbf{y}) = \sum_{\mathbf{c} \in \mathcal{C}} f^h_{\mathbf{c}}(\mathbf{y_c})$. Unary feature vector for a node v is defined as follows:

$$f^u_v(\mathbf{x}_v, \mathbf{y}_v) = \begin{bmatrix} I_v(1 - y_v) & (255 - I_v)y_v \end{bmatrix}^T .$$

Here I_v is the pixel intensity at the pixel v. Higher order clique feature $f^h_{\mathbf{c}}(\mathbf{y_c})$ at clique $\mathbf{c}$ is an indicator vector of size $2^{|c|}$ with 1 at the position corresponding to labeling $\mathbf{y_c}$ and 0 otherwise.

In all our experiments, every pixel shares the same unary parameter vector. $\mathbf{w}_u$ has size 2 for denoising. Every higher order clique has the same size and shares the same parameter vector $\mathbf{w}_h$ of size $2^{|c|}$. So we learn total $2 + 2^{|c|}$ parameters in the denoising experiment. We have used hamming distance as the loss function between the ground truth and the predicted image.

Noise Model: The benefit of using higher order models is more evident when there is a structured noise. A simple mathematics would suggest that in case of salt and pepper noise both pairwise and higher order models are expected to perform similarly. We synthetically generate images corrupted with structured noise. We select overlapping patches of size $k \times k$ with stride 1 in the ground truth images and flip all the pixel with probability p. We have experimented with different values of p and patch size.

Quantitative Analysis: We analyze our algorithm empirically in terms of how the objective function value decreases over iterations. Figure 2 (left plot) shows the result. To show the practical utility of the training objective formulation, we also analyze if the improvement in objective function corresponds to decrease in pixel loss. Figure 2 (right plot) shows average training pixel loss variations during the iterations of the algorithm. The plots indicate that, although the objective value keeps decreasing over the iterations, the pixel loss stabilizes very soon to a low value.

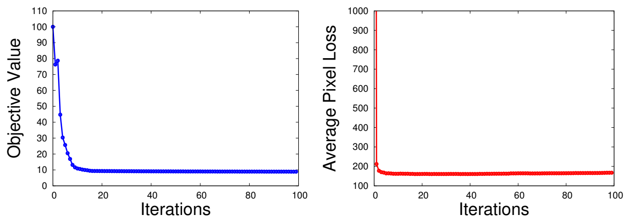

Figure 2: Change of objective value (left) and average pixel loss on the training data (right), during different iterations.

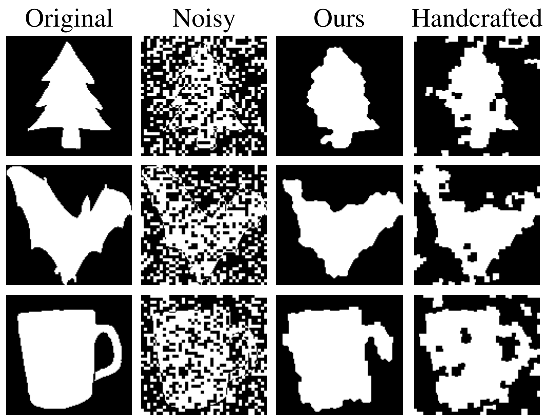

Figure 3: Comparison of the denoising output using clique potentials learnt from the proposed approach vs using handcrafted count based potential of size 3×3 clique potentials. We have used structured noise by choosing random patch of size 3×3 and flip pixels of the patch with probability $p = 0.3$. Image size is 120×120.

Table 1 compares the average pixel loss on 10 noisy butterfly images with increasing clique size using clique potentials learnt from the proposed method. We see that by increasing clique size from 1×2 to 2×2 reduces the pixel loss from 30.3% to 8.67%. For these experiments, we have created noisy input images by adding structured noise of patch size 3×3 with a probability of flipping $p = 0.7$. The table indicates the scalability of our algorithm as well as serves to justify the use of larger cliques to improve the quality of the solution.

We also do a quantitative analysis of learnt potentials of different clique sizes. Table 2 shows the average pixel loss as we vary both clique size and patch. We observe that as the patch size of noise increases, we need higher and higher order potentials to achieve a certain accuracy.

Visual Comparison: We also compared learned potential with hand tuned potentials such as the count based potential used in previous works [25, 26] for the task of binary object segmentation. The cost of a labeling under count based po-

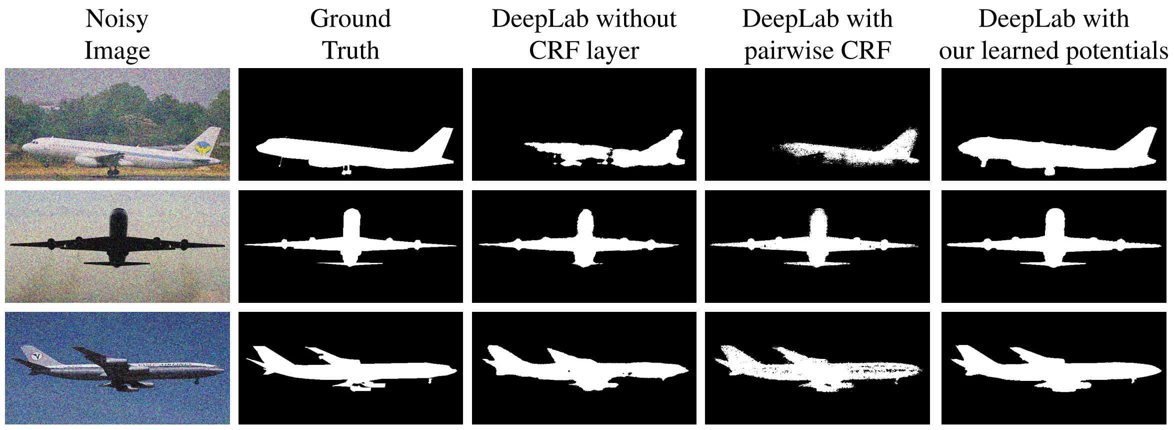
Noisy Image
Ground Truth
DeepLab without CRF layer
DeepLab with pairwise CRF
DeepLab with our learned potentials

Figure 4: Comparing object detection output using DeepLab Vs our approach using learnt clique potentials of size 3×3.

tential is $k * (n - k)$ where n is the clique size and k is the number of 1's in the labeling. In our experiments, the count based potential performed significantly poor compared to the learnt potentials. Figure 3 shows the visual comparison.

5.2. Object Detection

In this section, we show results of using higher order learnt potentials on the task of pixel level object detection. For comparison, we have used state of the art deep neural network DeepLab [19]. Since the proposed algorithm is for binary label problems only, we have selected images containing a single object from the PASCAL VOC-2012 dataset. We observe that DeepLab works excellent on these images. However, in a more realistic scenario, even if there is a little noise in the image, the solution quickly degrades. To get noisy images, we have added Gaussian noise with zero mean and intensity dependent variance. We report our comparison on such noisy images.

We have used publicly available implementation of DeepLab [2] for comparison. The architecture of DeepLab consists of a Deep Convolutional Neural Networks (DCNN) followed by a densely connected pairwise CRF [18]. For experimentation, we have used DeepLab model which uses ResNet-101 architecture. Unary potential of densely connected CRF are computed from the label assignment probability of the deep network and pairwise potential are a weighted linear combination of two Gaussian kernel and parameters of these kernels are computed from cross-validation as described in [19]. We use following energy function in our formulation:

$$E(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}_u^T f^u(\mathbf{x}, \mathbf{y}) + \mathbf{w}_p^T f^p(\mathbf{y}) + \mathbf{w}_h^T f^h(\mathbf{y}).$$

$f^u(\mathbf{x}, \mathbf{y})$, $f^p(\mathbf{y})$ and $f^h(\mathbf{y})$ denotes the unary, pairwise and higher order feature vectors corresponding to unary, pairwise and higher order potentials. We use unary potentials as described for image denoising problem, using probabilities of DCNN instead of image intensities as the input. Pairwise and higher order potential are learned using our proposed algorithm. Figure 4 shows the visual comparison. Learned higher order potential using our method produces much better output in comparison to DeepLab both with and without dense CRF.

6. Conclusion

In this paper, we have suggested a novel algorithm for learning higher order clique potentials in an MRF-MAP formulation. We incorporate the submodularity constraints in the objective function and then solve resulting optimization problem using the subgradient descent algorithm. Our method is particularly appealing for learning large clique potentials. To the best of our knowledge, ours is the first work that can learn potentials up to clique size 16. Our experiments indicate the improvement in image denoising and object detection outputs using the learned potentials compared to the hand tuned ones. The experiments also indicate the benefit of using larger cliques over smaller ones. Directions for future work include further scaling to larger clique sizes by enforcing additional parameter tying within each clique, extending our ideas to multi-label setting and experimenting with other computer vision tasks.

7. Acknowledgements

We would like to thank Ankit Anand and Ishant Shanu for helpful discussions. Parag Singla is being supported by the DARPA Explainable Artificial Intelligence (XAI) Program with number N66001-17-2-4032, IBM Shared University Research Award and the Visvesvaraya Young Faculty Research Fellowship by the Govt. of India. Chetan Arora is being supported by the Infosys Center for Artificial Intelligence and the Visvesvaraya Young Faculty Research Fellowship by the Govt. of India.

References

[1] A database containing 1400 binary shape images. http://www.imageprocessingplace.com/root_files_V3/image_databases.htm. Accessed: 2014-10-15.

[2] DeepLab implementaion. https://bitbucket.org/aquariusjay/deeplab-public-ver2. Accessed: 2017-11-02.

[3] K. Alahari, C. Russell, and P. H. Torr. Efficient piecewise learning for conditional random fields. In *Proc. of CVPR*, pages 895–901, 2010.

[4] C. Arora, S. Banerjee, P. Kalra, and S. Maheshwari. Generic cuts: an efficient algorithm for optimal inference in higher order MRF-MAP. In *Proc. of ECCV*, pages 17–30, 2012.

[5] C. Arora, S. Banerjee, P. K. Kalra, and S. Maheshwari. Generalized flows for optimal inference in higher order MRF-MAP. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1323–1335, 2015.

[6] D. P. Bertsekas, A. Nedi, and A. E. Ozdaglar. *Convex analysis and optimization*. Athena Scientific, 2003.

[7] D. P. Bertsekas and A. Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[8] J. Domke. Learning graphical model parameters with approximate marginal inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2454–2467, 2013.

[9] A. Fix, T. Joachims, S. M. Park, and R. Zabih. Structured learning of sum-of-submodular higher order energy functions. In *Proc. of ICCV*, pages 3104–3111, 2013.

[10] S. Fujishige and S. Isotani. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7(1):3–17, 2011.

[11] S. Gould. Learning weighted lower linear envelope potentials in binary markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1336–1346, 2015.

[12] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *Proc. of CVPR*, pages 2993–3000, 2009.

[13] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Journal of Machine Learning Research*, 77(1):27–59, 2009.

[14] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother. A comparative study of modern inference techniques for structured discrete energy minimization problems. In *Proc. of CVPR*, pages 1328–1335, 2013.

[15] D. Khandelwal, K. Bhatia, C. Arora, and P. Singla. Lazy generic cuts. *Computer Vision and Image Understanding*, 143:80–91, 2016.

[16] P. Kohli, L. Ladicky, and P. H. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[17] N. Komodakis. Efficient training for pairwise or higher order crfs via dual decomposition. In *Proc. of CVPR*, pages 1841–1848, 2011.

[18] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Proc. of NIPS*, pages 109–117, 2011.

[19] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *Proc. of ICLR*, 2015.

[20] A. Lucchi, Y. Li, and P. Fua. Learning for structured prediction using approximate subgradient descent with working sets. In *Proc. of CVPR*, pages 1987–1994, 2013.

[21] N. D. Ratliff, J. A. Bagnell, and M. Zinkevich. (approximate) subgradient methods for structured prediction. In *Proc. of AISTATS*, pages 380–387, 2007.

[22] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proc. of CVPR*, pages 860–867, 2005.

[23] C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *Proc. of CVPR*, pages 1382–1389, 2009.

[24] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[25] I. Shanu, C. Arora, and P. Singla. Min norm point algorithm for higher order MRF-MAP inference. In *Proc. of CVPR*, pages 5365–5374, 2016.

[26] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Proc. of NIPS*, pages 2208–2216, 2010.

[27] M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. *Proc. of ECCV*, pages 582–595, 2008.

[28] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proc. of ICML*, page 102, 2004.

[29] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Proc. of NIPS*, pages 25–32, 2004.

[30] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(9):1453–1484, 2005.

[31] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2115–2128, 2009.

[32] W. Zaremba and M. B. Blaschko. Discriminative training of crf models with probably submodular constraints. In *Proc. of WACV*, pages 1–7, 2016.