Structured Output Learning with High Order Loss Functions

Daniel Tarlow
Department of Computer Science
University of Toronto

Richard S. Zemel
Department of Computer Science
University of Toronto

Abstract

Often when modeling structured domains, it is desirable to leverage information that is not naturally expressed as simply a label. Examples include knowledge about the evaluation measure that will be used at test time, and partial (weak) label information. When the additional information has structure that factorizes according to small subsets of variables (i.e., is *low order*, or *decomposable*), several approaches can be used to incorporate it into a learning procedure. Our focus in this work is the more challenging case, where the additional information does not factorize according to low order graphical model structure; we call this the *high order* case. We propose to formalize various forms of this additional information as high order loss functions, which may have complex interactions over large subsets of variables. We then address the computational challenges inherent in learning according to such loss functions, particularly focusing on the loss-augmented inference problem that arises in large margin learning; we show that learning with high order loss functions is often practical, giving strong empirical results, with one popular and several novel high-order loss functions, in several settings.

1 Introduction

When formulating a learning objective, there is a range of information available to the modeler. Most commonly, in supervised learning, there are ground truth labels provided by measurement devices or domain experts. In the absence of any other knowledge about a domain, a natural learning objective is simply to minimize the number of errors made on the training data. Consider formulating a learning objective for an image labeling task, where the goal is to label each pixel in an image according to semantic class (e.g., airplane, dog, cow). Given ground truth labels for all pixels in a set of images, it is straightforward to define a learning objective that uses no other knowledge of the domain. In this case, the objective might correspond to penalizing each pixel-wise error, and doing so equally across the training set.

Appearing in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume 22 of JMLR: W&CP 22. Copyright 2012 by the authors.

However, there is typically information available that is not naturally expressed as simply a label. In the above example, we may have access to weakly labeled images (e.g., labeled with bounding boxes), or we may have an evaluation measure that is more suited to the learning task. It is desirable to incorporate this additional information into a learning objective, but doing so often leads to loss functions that do not easily decompose according to the model structure; in other words, the resulting loss functions are *high order.*

High order loss functions arise in many domains. In search engine rankings, for example, it is well known that the accuracy on the top most relevant results is far more important than accuracy on lower ranked results; standard evaluation objectives take this into account, and are high-order. Domain-specific evaluation measures with non-decomposable structure also arise in many other domains, including machine translation, where the BLEU score assesses the precision of n-grams in candidate machine translations against reference human translations [1], and image labeling problems, where the PASCAL VOC Segmentation Challenge scoring function is based on a ratio of counts: # true positives/(# true positives + # false positives + # false negatives) [2]. While incorporating knowledge of the test-time procedure in a learning objective is intuitively appealing, it does not always improve performance; in certain domains such as ranking it has been observed that optimizing the test objective can actually hurt performance [3]. In this work, we will show that it is beneficial to include knowledge of test-time evaluation in an image labeling setting.

The common motivation in each of these examples is that we would like to improve performance of our models using knowledge that is (a) not easily expressed as a simple label, and (b) is non-decomposable. Methodologically, the commonality is that we will express knowledge as non-decomposable or *high order loss functions (HOLs)*, then we will use a common learning procedure to incorporate the loss function into the learning objective. The framework we choose to work with is the *margin-scaled* structural SVM, which aims to maximize a margin between the energy a model assigns to the ground truth and the energy the model assigns to all other assignments. In the margin-scaled SVM, the margin is dependent on the loss — assignments with high loss need to have higher energy. The computational crux that comes up when learning using the margin-scaled SVM is finding the most violating constraint, which is a function of the assignment that maximizes a sum of negative energy and loss. When a decomposable loss function is paired with an otherwise tractable model, finding this most violating constraint is tractable, as the loss can be folded directly into standard inference computations. With high order losses, the problem becomes more difficult. Our approach will work within a high order message passing framework, as described in e.g., [4].

Interestingly, there are certain classes of high order interaction where efficient optimization can be performed either exactly or approximately. Such cases have been the focus of much recent work on *maximum a posteriori (MAP)* inference in high order models (see e.g., [5]), but little work has applied the computational insights at the heart of this work to learning problems; we extend the fundamental insights that are leveraged in the MAP inference community, and apply them to derive efficient learning algorithms for HOLs.

Contributions:

- New algorithms that enable several types of high order loss functions to be incorporated.
- Several new classes of loss function, along with the observation that efficient algorithms from MAP inference, which have not previously been considered in the context of learning objectives, can be used to optimize them.
- An empirical demonstration in image segmentation that optimizing the test objective (based on the PASCAL score) gives an improvement over optimizing other surrogate training objectives.

There are two secondary emphases of our work. The first concerns its modularity and generality. The complex loss functions are expressed as factors that can be used generically in a factor graph formulation, meaning they can be combined to form a range of loss functions and also be combined with any factor graph model, with no need for new formulations, derivations, or code. Second, by putting complexity in the loss function and not the energy, we push complexity to the training procedure and obtain models that can be very efficient at test-time.

2 Related Work

Methods for improving learning in structured output spaces by incorporating the loss function were originally developed for SVMs [7, 8], where the MAP inference problem required for margin maximization was augmented to include the loss function. Recent proposals have also demonstrated improved learning in conditional random field approaches to structured prediction by incorporating loss [9]. An application that drives progress in this area is image segmentation; [10] noted that pairwise segmentation models paired with (decomposable) per-pixel loss functions can be optimized very efficiently by using graph cuts to solve the loss-augmented MAP problem. Our work here can be viewed as an extension that explores high order loss functions, similarly relying on recent advances in MAP inference to achieve efficiency on difficult subproblems.

With the exception of [6] and [11], relatively little work exists on large margin learning with high-order losses; further, the algorithm of [6] is limited to the case of unary-only models, where all variables are assumed to be independent in the model. The recent work of [11] is closely related to one of our applications: the focus is optimizing a labeling model for performance on the aforementioned intersection-over-union loss. [11] utilizes a piecewise approximation to the loss function and solves a separate linear program for each piece. We discuss this method further in Section 4, drawing comparisons once we have given more details of our approach. We note that other works have expressed similar motivations, aiming to optimize specific test-time evaluation procedures, such as the BLEU score [12]. Our approach differs in that we work within a standard well-studied learning framework.

We also note that concurrent with this work, [13] addresses the problem of learning with high order loss functions for a special case where exact loss-augmented inference can be done via reduction to graph cuts. Our message passing approach is more general, but for the problem addressed, our loss-augmented inference is approximate. An empirical comparison between the two approaches is provided in [13].

There is a rapidly growing literature on incorporating high-order potentials into the model likelihood. Notable examples are the pattern potentials of [14, 15]; connectivity potentials of [16, 17]; cardinality potentials of [18, 19, 4]; order-based and composite potentials of [4]; and the near-bounding-box-border constraints of [20]. This research has produced a rich algorithmic toolbox to make MAP inference tractable for these models. We utilize and extend this toolbox in this paper. These methods of expressing desired properties in the model likelihood, such as adding terms to the energy that favor object labelings that are connected or convex, are complementary to our approach of adding structure to the loss function that the model seeks to optimize during learning.

A broader aim of incorporating additional forms of supervision, like we discuss here, has also been a popular theme recently in machine learning. A variety of methods for modifying the learning objective via adding regularization-like terms have been proposed, such as Posterior Regularization [21] and Generalized Expectation [22]. Our approach can be seen as an alternative, which puts this information directly into the loss function.

3 Background and Notation

Structural SVMs. Given an input image x with N pixels at test time and feature responses $\phi(x)$, our goal is to produce an output image labeling $y \in \{0, \ldots, K-1\}^N$. We will assume $K = 2$ throughout, but the loss formulations apply equally well to problems with larger K. A structural SVM works by mapping x to y via maximizing an input-dependent (log) likelihood function: $y = \arg\max_{\hat{y}} L(\hat{y}; x) = \arg\max_{\hat{y}} w^T \phi(\hat{y}; x)$, which is governed by parameters w. The learning problem is to find a w that maximizes a margin between the ground truth $y^*_{(j)}$ and all other outputs $y_{(j)}$ for all training examples j. This can be written as a quadratic program (QP):

$$\begin{aligned} \textbf{min.}_{w,\xi} \;\; & w^T w + C \sum_n \xi_{(j)} \\ \textbf{s.t.} \;\; & w^T \Big[\phi(y^*_{(j)}, x_{(j)}) - \phi(y_{(j)}, x_{(j)}) \Big] \geq 1 - \xi_{(j)} \end{aligned} \quad (1)$$

where $\xi_{(j)} \geq 0$ and the constraint is replicated for each example j and for all $y_{(j)} \neq y^*_{(j)}$. C is a regularization parameter. To incorporate a loss function Δ, the *margin-scaling* approach enforces a loss-dependent variable margin for different labelings, replacing Eq. (1) with:

$$\textbf{s.t.} \;\; w^T \Big[\phi(y^*_{(j)}, x_{(j)}) - \phi(y_{(j)}, x_{(j)}) \Big] \geq \Delta(y_{(j)}) - \xi_{(j)} \quad (2)$$

Even for a single example, Eq. (2) represents an exponentially large number of constraints, so explicitly instantiating the QP is intractable. A common strategy is to start with an empty set of constraints, successively add the most violated constraint, re-solve the QP, and repeat. To find violated constraints, a *loss-augmented MAP* problem must be solved: $y^- = \max_{\hat{y}} \left[L(\hat{y}; x) + \Delta(\hat{y}) \right]$. This procedure converges in polynomial time [8].

The central interpretation in our approach is that the loss-augmented MAP problem can be phrased as a (possibly approximate) inference problem in a factor graph that includes terms both for the energy and for the loss. In this light, assuming that we know how to compute message for all factors in the model, we can directly apply modern message passing algorithms such as max-product belief propagation (MPBP) or tree-reweighted max-product (TRW). Thus, to admit high order loss functions into a large margin learning formulation, we need only derive efficient message updates for factors representing loss functions of interest.

One Slack Formulation [23]. A more efficient formulation turns out to be equivalent. As usual, at iteration t, on example j, find a negative example $y^-_{(j)}$. Add one constraint in the form of Eq. (2) for a single $y_{(j)}$ to a constraint set $\mathcal{S}_t$. After finding most violated constraints for all instances, add the averaged constraints

$$w^T \sum_{(y^+_{(j)}, y^-_{(j)}) \in \mathcal{S}_t} \Big[\phi(y^+_{(j)}, x_{(j)}) - \phi(y^-_{(j)}, x_{(j)}) \Big] \geq \sum_{(y^+_{(j)}, y^-_{(j)}) \in \mathcal{S}_t} \Delta(y^-_{(j)}) - \xi \quad (3)$$

to the quadratic program for each t. In this formulation, there is a single ξ, and the size of the QP grows with the number of iterations rather than number of iterations times number of training examples. Especially with large data sets, this is a significantly more efficient, yet surprisingly equivalent, formulation. From here forward, we drop subscripts (j) indicating the training example. Later, we will use the notation y_i to refer to the ith variable in the output vector y.

4 Learning with High Order Losses

In this section, we begin with a minor generalization of the standard structural SVM formulation (in Section 4.1). Then in Section 4.2, we return to the interpretation of loss augmented MAP as approximate inference and give our main contributions: defining factor computations that allow several practical high order loss functions to be used within the large margin learning formulation.

4.1 Finding Positive and Negative Examples

Positive Examples: Partially labeled training data do not provide full ground truths. Instead, there is a set of pixels with known labels (e.g., with bounding boxes, pixels outside the bounding box are background) and a set of pixels with unknown labels (e.g., pixels inside the bounding box may be either foreground or background). A loss function defined in terms of partial labels implicitly expresses properties that we desire in a positive example, but we need a single labeling y^+ to compute $\phi(y^+, x)$. To deal with this issue, we run inference to find the labeling that has zero loss and highest likelihood under the model. We call this the *0-Loss Constrained MAP* problem: $y^+ = \max_{\hat{y}:\Delta(\hat{y})=0} L(\hat{y}; x)$. Note that when full labelings are given as training data (such as in our first set of experiments below), there is typically only one labeling with zero loss, so this reduces to using the ground truth as the positive example.

Negative Examples: The loss-augmented MAP problem is equivalent to a MAP inference problem in a factor graph for the model with an additional factor for the loss function. For a factor representing loss function Δ, the needed MPBP messages are,

$$m_{\Delta\to i}(y_i) = \max_{y_{-i}} \left[\Delta(y_i, y_{-i}) + \sum_{i' \neq i} m_{i'\to\Delta}(y_{i'}) \right], \quad (4)$$

where y_{-i} is the set of all variables excluding the ith. Note there is a separate message for each variable and each value it can take on, so naively there are $2N$ optimizations to perform. A theme in developing efficient message passing algorithms is to share the work between these different optimizations. In the following section, we show how to efficiently compute these messages for factors representing several loss functions of interest.

4.2 Factor Computations

Here, we develop three loss functions and algorithms for incorporating additional non-decomposable supervision into the learning objective.

(A) PASCAL Loss. The loss used to evaluate the PASCAL Segmentation Challenge, restricted to a single class, can be written as

$$\Delta^{\text{PASCAL}}_{y^*}(y) = 1 + \frac{\sum_i y_i^*(1-y_i) - \sum_i y_i^*}{\sum_i y_i^* + \sum_i y_i(1-y_i^*)}. \quad (5)$$

Let $N^+ = \sum_i y_i^*$, $N_0 = \sum_{i:y_i^*=1}(1-y_i)$, and $N_1 = \sum_{i:y_i^*=0} y_i$ be the number of ground truth pixels, false negatives, and false positives, respectively. We can rewrite the loss as $\Delta^{\text{PASCAL}}_{y^*}(y) = 1 + \frac{N_0 - N^+}{N^+ + N_1}$ and the objective inside the maximization in Eq. (4) as

$$f(N_0, N_1) = \frac{N_0 - N^+}{N^+ + N_1} + s_0(N_0) + s_1(N_1) + \kappa, \quad (6)$$

where $s_0(N_0)$ is the cumulative sum of the first N_0 sorted negative incoming message values from variables where $y^* = 1$ and $s_1(N_1)$ is the cumulative sum of the first N_1 sorted incoming message values from variables where $y^* = 0$, and κ is a constant.

The full details of the message computations for the PASCAL factor are given in the Appendix. Here, we give a sketch, along with the intuition. The key insight is that the optimal setting of y_{-i}, and thus the full outgoing message optimization Eq. (4) can be easily computed from the choice of N_0 and N_1 that optimizes f. The secondary insight is that after relaxing N_0 and N_1 to be real-valued, f is quasi-concave restricted to the domain of interest: $N_0, N_1 \geq 0$. This leads to the simple (but possibly suboptimal) optimization approach that we found to work well, which is to use local search to find the optimal N_0 or N_1, then to use N_0 and N_1 to compute y_{-i}. Although the procedure could theoretically get stuck at suboptimal points, in experiments to test this, it always reached the optimum, and it did so very quickly. (An alternative choice would use bisection to guarantee convergence to an ϵ-suboptimal solution in $\log_2(\frac{1}{\epsilon})$ iterations.)

Each outgoing message requires optimizing f with slightly different incoming message values. Since the problems are so similar, we find that we get significantly faster convergence by warm starting the optimization for one outgoing message with the result from the previous calculation. Empirically, solving the first optimization takes linear time, then solving the latter N problems takes constant time each (usually 0, 1, or 2 steps of local search). Thus, we expect the sort operation to dominate the complexity and runtime for computing *all* N outgoing messages from a factor to scale like $N \log N$ i.e., $\log N$ time amortized per message. Indeed, for an image with ten thousand pixels, like we use in our experiments, exact computation at the loss factor in each iteration takes .03 seconds. Empirical runtimes on larger problems indicate the scaling to be roughly $N \log N$ (10k pixels: .03s, 100k pixels: .32s, 1M pixels: 3.3s, 10M pixels: 34.5s).

Comparison to [11]. The preceding formulation is superior to the piecewise linear approximation (PLA) of [11] in several respects. First, the scale is significantly greater; our algorithms operate on tens of thousands of output variables (one per pixel) as opposed to the approximately 50 output variables used by PLA. The computation benefit comes from the fact that rather than solving many linear programs (one per piece of the approximation to the loss function) to find a violated constraint, the above approach uses highly efficient special-purpose algorithms, which communicate using message passing. Second, the above formulation is exact in cases where the PLA approach is not, because the message passing operates on the exact loss and computes exact outgoing messages. Finally, the above approach is more modular, as different

HOLs can easily be combined with other forms of supervision and/or other modeling components.

(B) Bounding Box % Fullness Loss. Suppose we are given bounding box labels at training time and wish to optimize for performance on a per-pixel segmentation task. This situation may arise if we are training a segmentation model and would like to augment the set of detailed per-pixel ground truths with a set of easier-to-obtain, coarser bounding box ground truths. We assume that bounding boxes contain roughly a fraction R foreground pixels (and $1-R$ background pixels). Let y^* be the coarse labeling where y_i^* indicates whether pixel i is within the bounding box. A reasonable (and novel) loss function is

$$\Delta_{y^*}^{\mathrm{BB}}(y) = \sum_{i:y_i^*=0} y_i + \left| \sum_{i:y_i^*=1} y_i^* y_i - R \sum_{i:y_i^*=1} y_i^* \right|. \quad (7)$$

This loss can be decomposed into low order parts over pixels outside the bounding box and a high order part over pixels inside the bounding box. The high order term can be written as a function of the number of pixels on inside the bounding box i.e., $f(\sum_{i:y_i^*=1} y_i)$, so the cardinality potential from [4] can be used without modification. The 0-loss Constrained MAP can also use an unmodified cardinality potential. A range of other functions could be used for the inside the bounding box term. For example, we might assign zero loss to several different pixel value counts and various penalties to deviations from these preferred values. Any such loss can easily be used in this formulation.

(C) Local Border Convexity Loss. A second form of weak labeling that is easier for humans to produce than a full per-pixel labeling is a rough inner-bound on the true labeling plus a rough outline of the object to serve as an outer bound. For example, a popular form of labeling in interactive image segmentation derives from a user drawing a few strokes to mark the internal skeleton of the object, and a crude circular stroke around the outside. We would like to use such a labeling to define a loss function. We assume that the strokes define the inner core of the object, so if follow any ray extending out from the stroke towards the boundary of the object, the labeling along the ray will be *monotonic*—i.e., of the form $1^m 0^n$.

This motivates our *local border convexity* (LBC) loss, illustrated in Fig. 1 (a) and described intuitively in the caption. Formally, we start with a set of pixels $\mathcal{F}$ that are labeled foreground in the and a set of pixels $\mathcal{B}$ that are labeled background. The remaining unlabeled pixels will be denoted $\mathcal{U}$ (gray in Fig. 1). We construct shortest paths through pixels in $\mathcal{U}$ from an outermost pixel $p \in \mathcal{U}$ (i.e., one that neighbors a pixel in $\mathcal{B}$) to any innermost pixel in $q \in \mathcal{U}$ (i.e., one that neighbors a pixel in $\mathcal{F}$). We then traverse the path backwards, from q to p and add a monotonicity constraint on this ordered set $\mathcal{Q}_0 = (q_0, \ldots, p_0)$ of pixels. We repeat this for each pixel on the $\mathcal{U}$ - $\mathcal{B}$ boundary that has not already been used in a shortest path, then through other pixels in $\mathcal{U}$ that have not been used in any shortest path. From each of these starting pixels, we find a shortest path through $\mathcal{U}$ to $\mathcal{F}$, constructing ordered sets $\mathcal{Q}_k$ for each path. When all pixels in $\mathcal{U}$ have been used in at least one path, we have our set of constraints $\mathcal{Q} = \{\mathcal{Q}_0, \ldots, \mathcal{Q}_m\}$. The loss is then,

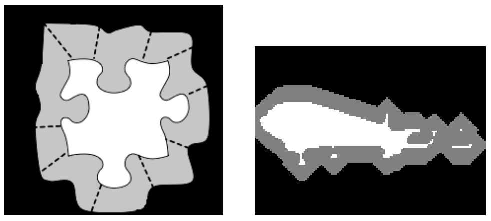


Figure 1: (Left) Illustration of LBC loss. The white region is foreground, and the black is background. The gray can take either label, but a penalty of α is incurred for each outwards path that changes from background to foreground as it moves away from the object. (Right) Example of eroded Aeroplane labeling.

$$\Delta_{y^*}^{LBC}(y) = \sum_{i \in \mathcal{F} \cup \mathcal{B}} 1\{y_i \neq y_i^*\} + \sum_{(q,\ldots,p) \in \mathcal{Q}} g(y_q, \ldots, y_p), \quad (8)$$

where $g(y_0, \ldots, y_m) = 0$ if $y_i \geq y_j$ for all $i < j$ and α otherwise.

To maximize the loss, we need to define a *not-monotonic* factor. This is an order-based potential [4], We describe a dynamic programming algorithm for this factor in the supplementary materials. The algorithm computes all N outgoing messages in $O(N)$ time, i.e., $O(1)$ amortized time per message. The 0-loss constrained MAP uses a slightly modified version of the convexity potential described in [4], which is equally efficient (see supplementary materials).

5 Experimental Evaluation

5.1 Deformable Shape Loss

As a first experiment, we simulated a setting where the relative location of a set of points is important, but the precise location is not. Such a setup is motivated by the task of recognizing deformable shapes given noisy observations of landmarks.

Inputs come in the form of noisy observations of a set of point locations on a 2D grid. Specifically, the task is to jointly predict the location of three special points on a grid, given feature responses at each grid location. We compare two losses: a low order loss that penalizes

Noise	Train \ Evaluate	Pixel Error	HO Error
Low	Pixel-loss	**7.1%**	3.2%
Low	HO-loss	10.1%	**1.4%**
Med	Pixel-loss	**26.9%**	9.7%
Med	HO-loss	28.2%	**5.2%**
High	Pixel-loss	84.3%	44.9%
High	HO-loss	**67.5%**	**15.8%**

Figure 2: Synthetic results. Error on pixel-based error and high order (HO) error for different levels of noise and train time loss functions.

any error in predicting special point locations; and a high order loss that cares only about the relative ordering of the points from top to bottom and from left to right. Due to space, we describe experimental details in the supplementary materials. The important result, however, is that learning according to the high order loss function leads the model to favor the features that are less reliable predictors of the exact location but that are more accurate predictors of the general region where a point lies; learning according to the low order loss leads the model to favor a set of features that gives a stronger indicator of the precise location of the special points but that has a non-local noise model. Quantitative results (Fig. 2) show that except for the high noise case, the high order-trained model performs best on the high order loss, and the low order-trained model performs best according to the low order loss. The low order loss, in a sense, leads to learning a model that cannot see the forest for the trees.

5.2 Three Image Labeling Experiments

Data. We took subsets of images from the PASCAL VOC Segmentation challenge data set containing a given object—Aeroplane, Car, Cow, and Dog. The database provided pixel-wise segmentations and bounding boxes for each image. We then created ground truth labels by assigning a pixel to foreground if it was labeled {Aeroplane, Car, Cow, Dog} in the VOC labels and background otherwise. We scaled the images so that the minimum dimension was 100 pixels. We also created *eroded* data by eroding the per-pixel ground truths with a disk of radius 5.5. We then created an uncertain region by dilating the eroded labels by 10 pixels. See Fig. 1 (b) for an example.

Model. We use 84 per-pixel features that represent color and texture in the patch surrounding a pixel (the **unary** model). We use a standard pairwise 4-connected grid model, with 1 constant pairwise feature and 12 pairwise features that are based on boundary appearance (the **unary + pairwise** model). Weights on pairwise features are constrained to be positive, so that the resulting pairwise potentials are submodular. This produces models with approximately 10,000 to 20,000 variables and 20,000 to 40,000 edges per image.

Inference. We use the COMPOSE framework [24] to compute messages for the entire pairwise model using dynamic graph cuts [25]. With decomposable (per-pixel) losses, this guarantees that inference is exact, even though the grid graph contains loops. With high order losses, inference is not guaranteed to be exact, but we find this framework to work significantly better than standard max-product belief propagation with a static message passing schedule.

We use an asynchronous schedule across subproblems, where the full grid model is treated as one subproblem, and the loss is treated as a second subproblem. We alternate between the loss factor(s) and the submodular grid factor, having each send all outgoing messages at each step. We use damping of .95 for all experiments and set a maximum number of 50 iterations. Solving the QP is very fast, so the bottleneck is the loss-augmented MAP calls. We parallelize them over four CPUs. One loss-augmented MAP call takes between a couple seconds and two minutes, depending on the model and loss function being used. Without pairwise potentials, only one iteration of message passing is typically required for the loss-augmented MAP, so learning is very fast. For learning, we set $C = .01$ for experiments with per-pixel ground truths and $C = .0001$ for experiments with weakly labeled ground truths. All our learned models are submodular, so we run the graph cuts algorithm of [26] at test time to find the optimal MAP labeling.

(A). PASCAL Loss. We first examine how training on different loss functions affects test performance. We look at three loss functions: **0-1 Loss** is the constant-margin structural SVM Eq. (1); **Pixel Loss** and **PASCAL Loss** are loss-augmented structural SVM training with the respective loss functions. We pair all combinations of training objective and test evaluation, between pixel and PASCAL accuracy. We also evaluate the tradeoff associated with using pairwise potentials in the model. On one hand, we know that objects in images are smooth, so introducing pairwise interactions should make the model more realistic. On the other hand, when paired with high order losses, loss-augmented MAP inference is no longer guaranteed to be exact, which might hurt learning performance.

Fig. 3 (a) shows results for a unary only model trained to optimize the three loss functions, where inference is always exact. Fig. 3 (b) shows results for a unary + pairwise model trained with the same loss functions. For the 0-1 and pixel loss, inference is exact. For the

Train \ Evaluate	Pixel Acc.	PASCAL Acc.
0-1 Loss	82.1%	28.6
Pixel Loss	**91.2%**	47.5
PASCAL Loss	88.5%	**51.6**

(a) Unary only model

Train \ Evaluate	Pixel Acc.	PASCAL Acc.
0-1 Loss	79.0%	28.8
Pixel Loss	**92.7%**	54.1
PASCAL Loss	90.0%	**58.4**

(b) Unary + pairwise model

	Train \ Evaluate	Pixel Acc.	PASCAL Acc.
Car	Pixel Loss	**80.4%**	6.7
	PASCAL Loss	72.9%	**37.0**
Cow	Pixel Loss	**80.3%**	23.3
	PASCAL Loss	79.4%	**48.1**
Dog	Pixel Loss	**81.5%**	16.6
	PASCAL Loss	75.9%	**38.3**

(c) Unary + pairwise model; other objects

Figure 3: Test accuracies for training-test loss function combinations. In (c), all models are unary + pairwise. A labeling of all background gives pixel accuracies of 79.8%, 78.9%, and 80.2%, on Car, Cow, and Dog respectively.

PASCAL loss, inference is approximate. However, results show that the combination of richer model, high order loss, and approximate inference produces the best results out of all models when evaluated on the PASCAL objective. In particular, it outperforms both exact PASCAL loss training with unary-only potentials and exact training of a unary + pairwise model but with a pixel loss.

In Fig. 3 (c), we take the strongest baseline (unary + pairwise model, pixel loss) and compare it to a unary + pairwise model trained to optimize the PASCAL loss on a variety of other object classes. As before, we find that training the model for the same loss function as is being evaluated at test time produces results that are clearly superior. In Fig. 4, we show training and test outputs for unary + pairwise models trained on the pixel loss (middle column) and the PASCAL loss (right column). Visually, we can see that the PASCAL-trained model labels more pixels foreground, correcting the overly conservative pixel-trained model.

(B). Bounding Box % Fullness Loss. Here, we consider the problem of learning to label pixels in images when only a bounding box is given as supervision. We use the bounding box loss given in (9) and use a range of settings of the "fullness" parameter R. When $R = 1$, we have a baseline low-order model where the

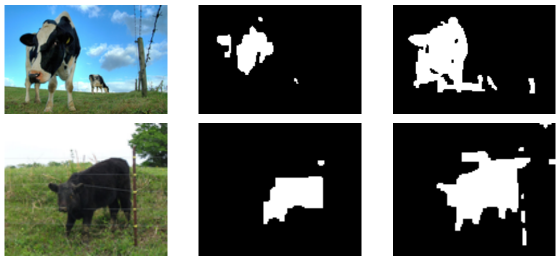
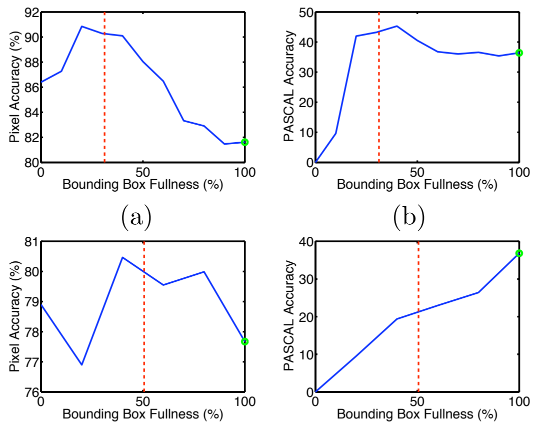


Figure 4: Example test results on Cow dataset. Methods from left to right: (Left) Raw image. (Middle) Pixel Loss. (Right) PASCAL Loss. Additional examples can be found in the supplementary material.

bounding box is treated as the ground truth segmentation. Fig. 5 shows test results for Aeroplane and Cow evaluated on both pixel and PASCAL loss. In Fig. 5 (b,c,d), we see that performance peaks near the true percentage observed in the data, denoted by the vertical red line. Note that the location of this vertical red line can itself be viewed as a type of non-decomposable supervision, which we could reliably predict from domain knowledge or a small number of fully labeled examples. This demonstrates that the model utilizes the weak labels in the bounding box, out-performing the SVM that ignores these pixels.

(C). Local Border Convexity Loss. Finally, we compare two methods for learning with partially-labeled images, as illustrated in Fig. 1. In the baseline method, we train an SVM with a modified loss function, where unlabeled pixels can be assigned either label without incurring any loss. This amounts to discarding the unlabeled pixels during training. We refer to this as the **Modified Loss SVM**. The second method is our structural SVM with **LBC Loss**. In this loss function, there are many labelings of the unlabeled pixels that are assigned zero loss, but penalties are imposed on non-locally convex labelings.

Fig. 5 shows results on four object classes. In all cases, the LBC loss is an improvement over the baseline when evaluating with respect to the PASCAL score (and in three of four cases with respect to pixel accuracy). Note that the loss being optimized is not exactly either the PASCAL or pixel loss. In the other cases, the improvements are more modest, likely because the Modified Loss SVM is not significantly worse than the methods that train on fully labeled ground truths (Fig. 3), so there is not much room for improvement. The largest gap in performance between strongly labeled and weakly labeled is on Aeroplane, which is likely caused by the high proportion of small planes – our erosion method completely eliminates some small objects, making it impossible for the LBC loss to recover them at training time. Surprisingly, on the Dog data set, the results are even better than training with per-pixel labels.



(a) (b)

(c) (d)

Train \ Evaluate		Pixel Acc.	PASCAL Acc.
Aero	Mod. Loss SVM	90.2%	36.4
	LBC Loss	**90.6%**	**38.1**
Car	Mod. Loss SVM	79.8%	0
	LBC Loss	**80.2%**	**5.3**
Cow	Mod. Loss SVM	**78.4%**	15.6
	LBC Loss	76.8%	**32.3**
Dog	Mod. Loss SVM	80.2%	0
	LBC Loss	**82.4%**	**24.2**

(e)

Figure 5: (a-d) Bounding box results: test accuracy versus bounding box fullness parameter R. Vertical line shows true average bounding box fullness. (a) Aeroplane pixel score. (b) Aeroplane PASCAL score. (c) Cow pixel score. (d) Cow PASCAL score. (e) Local border convexity results for training-test loss function combinations on eroded data. Modified Loss SVM is an independent SVM that discards pixels in the gray region at training time.

6 Discussion and Future Work

Though low order loss functions are convenient for optimization, they can impose a significant bias on the learned model. In this work, we show several examples where model performance can be improved by using more complex loss functions without significantly sacrificing computationally efficiency. A larger class of loss functions provides more flexibility in designing a training criteria, allowing one to tailor the loss to the application, e.g., training a model to optimize the PASCAL loss significantly improves performance when it is evaluated on the PASCAL loss, while training a model to optimize the deformable shape loss is effective at guiding the model to focus its capacity on the features most relevant to the true task. The other two losses give a meaningful way of doing semi-supervised learning, where a loss is defined in terms of how a partial labeling is extended to a full labeling.

Based on these results, and the efficiency of the high order factors (e.g., the PASCAL factor defined over 10 million variables requires only approximately 30 seconds per iteration), we expect there to be many more scenarios where tractable high order loss functions can improve performance, both in computer vision and beyond. Specifically, all of the losses here are applicable to multilabel problems. A second natural extension is to formulate high order losses that do not just apply to many outputs within an image but that apply to many outputs across multiple images. For example, we might define a loss function in terms of the smoothness of a pattern of pixels moving through frames of a video.

Given this extension of the range of loss functions that can be efficiently optimized in structural SVMs, an interesting modeling choice that arises is whether to add structure to the model, to the loss function, or to both. There are two potential benefits of adding structure to the loss rather than the model. First, it may facilitate learning. Some constraints or desired properties of the labeling are easier to express relative to the ground truth, which is available at training time but not at test. For example, if we would like to build some translation-invariance into the model, we can construct a loss function that assigns zero loss to segmentations in which the target object is translated by one or two pixels. This can be expressed as a high-order loss, but would be nearly impossible to put into the likelihood. Second, it allows for flexible modeling while also permitting fast test-time inference. Adding structure to the loss function but not the model creates a variational-like learning scenario, where the model must learn to use its restricted degrees of freedom to best optimize the loss. The simpler likelihood can make test-time inference very fast. Exploring the tradeoffs between these alternative model formulations is a direction of future work. More generally, we believe that the flexibility and efficiency of our loss functions opens up a wide range of exciting applications.

References

[1] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, June 2010.

[3] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: Optimising non-smooth rank metrics. In *WSDM*, 2008.

[4] D. Tarlow, I. Givoni, and R. Zemel. HOP-MAP: Efficient message passing with high order potentials. In *AISTATS*, 2010.

[5] Carsten Rother and Sebastian Nowozin. *Tutorial at CVPR 2010:* Higher-order models in computer vision, 2010.

[6] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005.

[7] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *NIPS*, 2003.

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

[9] T. Hazan and R. Urtasun. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *NIPS*, 2010.

[10] Martin Szummer, Pushmeet Kohli, and Derek Hoiem. Learning CRFs using graph cuts. In *ECCV*, 2008.

[11] Mani Ranjbar, Greg Mori, and Yang Wang. Optimizing complex loss functions in structured prediction. In *ECCV*, 2010.

[12] F. J. Och. Minimum error rate training in statistical machine translation. In *ACL*, 2003.

[13] P. Pletscher and P. Kohli. Learning low-order models for enforcing high-order statistics. In *AISTATS*, 2012.

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

[15] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2009.

[16] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008.

[17] S. Nowozin and C.H. Lampert. Global connectivity potentials for random field models. In *CVPR*, 2009.

[18] R. Gupta, A. Diwan, and S. Sarawagi. Efficient inference with cardinality-based clique potentials. In *ICML*, 2007.

[19] Brian Potetz and Tai Sing Lee. Efficient belief propagation for higher order cliques using linear constraint nodes. *CVIU*, 112(1):39–54, Oct 2008.

[20] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *ICCV*, 2009.

[21] K Ganchev, J. Graa, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models.

[22] G. Mann and A McCallum. Generalized expectation criteria with application to semi-supervised classification and sequence modeling. *Journal of Machine Learning Research (JMLR)*, 2010.

[23] T. Joachims, T. Finley, and Chun-Nam Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[24] John Duchi, Daniel Tarlow, Gal Elidan, and Daphne Koller. Using combinatorial optimization within max-product belief propagation. In *NIPS*, 2007.

[25] Pushmeet Kohli and Philip H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *PAMI*, 29(12):2079–2088, 2007.

[26] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

Appendix — Structured Output Learning with High Order Loss Functions

We include the following:

- Detailed derivation of PASCAL factor messages.
- Detailed derivation of not-monotonic factor messages negative example LBC messages.
- Detailed derivation of modified convexity factor messages for positive example LBC messages.
- Detailed description of the synthetic data experiment.
- Additional results showing test outputs on different classes of object.
- Illustration of typical failure modes of PASCAL-trained model.

7 PASCAL Loss Factor

Our goal is to compute outgoing messages from a factor representing the PASCAL loss,

$$\Delta_{y^*}^{\text{PASCAL}}(y) = \frac{\sum_i y_i^*(1-y_i) - \sum_i y_i^*}{\sum_i y_i^* + \sum_i y_i(1-y_i^*)}. \tag{9}$$

As in the main text, let $N^+ = \sum_i y_i^*$, $N_0 = \sum_{i:y_i^*=1}(1-y_i)$, and $N_1 = \sum_{i:y_i^*=0} y_i$ be the number of ground truth pixels, false negatives, and false positives, respectively. We can rewrite the loss as $\Delta_{y^*}^{\text{PASCAL}}(y) = \frac{N_0-N^+}{N^++N_1}$.

Consider computing a single message,

$$m_{\Delta\to i}(y_i) = \max_{y_{-i}}\left[\Delta(y_i, y_{-i}) + \sum_{i'\neq i} m_{i'\to\Delta}(y_{i'})\right] \tag{10}$$

$$= \max_{y_{-i}}\left[\frac{N_0-N^+}{N^++N_1} + \sum_{i'\neq i} m_{i'\to\Delta}(y_{i'})\right]. \tag{11}$$

We can transform the sum of incoming messages to similarly be functions of N_0 and N_1. First, split the incoming messages into those coming from pixels that are labeled 0 in the ground truth and those that are labeled 1 in the ground truth:

$$\sum_{i'\neq i} m_{i'\to\Delta}(y_{i'}) = \sum_{i'\neq i:y_{i'}^*=0} m_{i'\to\Delta}(y_{i'}) + \sum_{i'\neq i:y_{i'}^*=1} m_{i'\to\Delta}(y_{i'}) \tag{12}$$

$$= s_1^{-i}(N_1) + s_0^{-i}(N_0) + \kappa, \tag{13}$$

where $s_0^{-i}(N_0)$ is constructed by sorting the incoming message differences

$$m_{i'\to\Delta}(0) - m_{i'\to\Delta}(1) \tag{14}$$

for $\{i' | i' \neq i, y_{i'}^* = 0\}$ in descending order, then taking a cumulative sum of the sorted values, and $s_1^{-i}(N_1)$ is constructed by sorting the incoming message differences

$$m_{i'\to\Delta}(1) - m_{i'\to\Delta}(0) \tag{15}$$

for $\{i' | i' \neq i, y_{i'}^* = 1\}$ in descending order, then taking a cumulative sum of the sorted values. By taking message differences, we have added a constant κ, but we have not changed the optimum location or relative values of assignments.

Finally, we must account for the contribution of y_i to N_0 and N_1. There are three possibilities:

- When computing messages for $m_{\Delta\to i}(0)$ and $y_i^* = 1$, optimize

$$f(N_0, N_1) = \frac{N_0 - N^+ + 1}{N^+ + N_1} + s_0^{-i}(N_0) + s_1^{-i}(N_1). \tag{16}$$

- When computing messages for $m_{\Delta\to i}(1)$ and $y_i^* = 0$, optimize

$$f(N_0, N_1) = \frac{N_0 - N^+}{N^+ + N_1 + 1} + s_0^{-i}(N_0) + s_1^{-i}(N_1). \tag{17}$$

- Otherwise, optimize

$$f(N_0, N_1) = \frac{N_0 - N^+}{N^+ + N_1} + s_0^{-i}(N_0) + s_1^{-i}(N_1). \tag{18}$$

Finally, note that the all s_1^{-i} function values needed can be computed using a single sort of all incoming messages from variables where $y_i^* = 0$. That is, a separate sort is not needed for each i. Let the cumulative sum of a sorted array of message differences be s_1. When evaluating $s_1^{-i}(c)$, check whether $m_{\Delta\to i}(1) - m_{\Delta\to i}(0)$ is greater than the cth largest message difference. If it is, compute $s_1^{-i}(c) = s_1(c) - (m_{\Delta\to i}(1) - m_{\Delta\to i}(0)) + (m_{\Delta\to r(c+1)}(1) - m_{\Delta\to r(c+1)}(0))$ where $r(c)$ gives the index of the cth largest message difference. s_0^{-i} can be computed analogously.

As stated in the main body, the empirical runtime for computing all outgoing messages from this factor is $O(N \log N)$ or $O(\log N)$ amortized per message. Runtimes for computing all outgoing messages from the factor are as follows: 10k pixels: .03s, 100k pixels: .32s, 1M pixels: 3.3s, 10M pixels: 34.5s.

8 Local Border Convexity Loss Factor

Recall from the main text that the local border convexity (LBC) loss is defined as,

$$\Delta_{y^*}^{LBC}(y) = \sum_{i \in \mathcal{F} \cup \mathcal{B}} 1\{y_i \neq y_i^*\} + \sum_{(q,\ldots,p) \in \mathcal{Q}} g(y_q, \ldots, y_p), \quad (19)$$

where $g(y_1, \ldots, y_m) = 0$ if $y_i \geq y_j$ for all $i < j$ and α otherwise.

To use this loss within a loss-augmented MAP routine, we just need to show how to compute outgoing messages from factors representing the g functions. The other terms are low order and can be added to singleton potentials as is standard. Thus, we would like to compute

$$m_{g \to i}(y_i) = \max_{y_{-i}} \left[g(y_i, y_{-i}) + \sum_{i' \neq i} m_{i' \to g}(y_{i'}) \right]. \quad (20)$$

We can compute all messages at once with a linear time dynamic programming algorithm. The variables in the scope of g form an ordered set, so we use the convention that the "start" (or "left-most") variable is the one that neighbors a foreground pixel, and the "end" or ("right-most") variable is the one that neighbors a background pixel.

Begin by constructing six arrays to cache the following values. There are three from the "left" and three from the "right":

- $L_1(i)$, which stores the maximum value of an assignment to variables 1 to i where all variables in the range are on.
- $L_2(i)$, which stores the maximum value of an assignment to variables 1 to i where at least one variable in the range has been off.
- $L_3(i)$, which stores the maximum value of an assignment to variables 1 to i where at least one variable in the range has been off, and at least one variable after the off variable(s) has been on.
- $R_1(i)$, which stores the value of an assignment to variables i to m where all variables in the range are off.
- $R_2(i)$, which stores the value of an assignment to variables i to m where at least one variable in the range has been on.
- $R_3(i)$, which stores the value of an assignment to variables i to m where at least one variable in the range has been on, and at least one variable before (i.e., with smaller index) the on variable has been off.

These arrays can be populated in linear passes like is standard in many dynamic programming algorithms. For notational convenience, also define maximums over the arrays in each direction:

$$L^*(i) = \max(L_1(i), L_2(i)) \quad (21)$$
$$R^*(i) = \max(R_1(i), R_2(i)) \quad (22)$$

From these arrays, we can easily compute each outgoing message. For a message i with value 0 $m_{g \to i}(0)$, there are three options:

$$A = L^*(i-1) + R_2(i+1) \quad (23)$$
$$B = L_3(i-1) + R^*(i+1) \quad (24)$$
$$C = L^*(i-1) + R^*(i+1) - \alpha, \quad (25)$$

then the final message is

$$m_{g \to i}(0) = \max(A, B, C). \quad (26)$$

For a message i with value 1 $m_{g \to i}(1)$, there are also three options:

$$D = L^*(i-1) + R_3(i+1) \quad (27)$$
$$E = L_2(i-1) + R^*(i+1) \quad (28)$$
$$F = L^*(i-1) + R^*(i+1) - \alpha, \quad (29)$$

then the final message is

$$m_{g \to i}(1) = \max(D, E, F). \quad (30)$$

The runtime for computing all outgoing messages is $O(N)$, or $O(1)$ amortized time per message.

9 One-sided Convexity Factor

In the 0-loss constrained MAP inference, we need to enforce the local border convexity constraint—that labelings along paths extending outward from the foreground region are monotonic and decreasing. Here we show how to compute outgoing messages from a factor that enforces this hard constraint. This is straightforward, because there are only $m+1$ legal joint settings of variables of the form $1^k 0^{m-k}$.

Using the same "left" and "right" convention of the previous section, compute six arrays in a linear pass:

- $L_1(i)$, which stores the value of setting the first i variables on.
- $L_2(i)$, which stores the maximum value of an assignment to the first i variables where the joint assignment follows the pattern $1^k 0^{i-k}$.
- $R_1(i)$, which stores the value of an assignment to variables i to m where all variables in the range are off.

- $R_2(i)$, which stores the value of an assignment to variables i to m where the joint assignment follows the pattern $1^k 0^{m-i-k}$.

The messages can then be computed:

$$m_{g\to i}(0) = L_1(i-1) + \max\left(R_1(i+1), R_2(i+1)\right) \tag{31}$$

$$m_{g\to i}(1) = \max\left(L_1(i-1), L_2(i-1)\right) + R_1(i+1). \tag{32}$$

The total computation time to compute all outgoing messages from a factor is $O(N)$ i.e., $O(1)$ amortized per message.

10 Synthetic Experiment

We created data on a 20 x 20 grid of locations. For each of 200 training cases, we assigned three special points, $A = (i_A, j_A)$, $B = (i_B, j_B)$, and $C = (i_C, j_C)$ to a random location on the grid. With each location (i,j), we associated a six dimensional feature vector, $\phi(i,j) = (f_A(i,j), f_B(i,j), f_C(i,j), g_A(i,j), g_B(i,j), g_C(i,j))$. Each feature gives a noisy measurement of the location of the special point it is associated with e.g., f_A gives a noisy detection of A's location. Features take value 0 at all except for two locations–a true location and a distractor location–which are determined as follows: for $X \in \{A,B,C\}$, $f_X(X) \sim \text{Uniform}(0, \frac{1}{\alpha})$ gives a response at the true location, $f_X(Y) \sim \text{Uniform}(0,1)$ gives a response at distractor location Y which is drawn uniformly from the set of all locations. $g_X(X) \sim \text{Uniform}(0, \frac{.5}{\alpha})$ gives an on-average weaker response at the true location, and $g_X(Z) \sim \text{Uniform}(0,1)$ is a *local* distractor: Z is chosen uniformly at random from the 5x5 grid centered at the true location.

Given weights w_A, w_B, w_C, the model predicts the location of special point X as $\arg\max_{ij} w_X^T \phi(i,j)$. We experimented with training according to two loss functions. First, the low order per-pixel loss gives loss of $\frac{1}{3}$ for each incorrect prediction. Second, we trained the model to optimize a high-order order-based loss, which cares only about the relative locations of the three points along the two dimensions. A loss of $\frac{1}{6}$ is incurred for each error in relative orderings. To find violated constraints under the high order loss, we use a modified version of the order-based potentials described in [4]. We used 200 images for training, 200 for validation (for choosing regularization constant C), and 500 for test.

Figure Fig. 6 shows the results. The optimal strategy to optimize the high order loss is to rely more heavily on the weaker but more local feature, while the optimal strategy to optimize the low order loss is to rely on the stronger but non-local noisy feature. The model learns appropriately under both loss functions. Interestingly, when the noise becomes very high ($\alpha = 2$), the pixel-loss-trained model is worse on both measures. In this case, for all settings of C we tried, the low-order loss uses all of its slack and pushes weights to zero, learning nothing. In this case, since the order-based loss is easier, it learns, and is able to outperform the low order loss according to both metrics.

Train \ Evaluate	Pixel Error	Order Error
$\alpha = .5$		
Pixel-loss	**7.1%**	3.2%
Order-loss	10.1%	**1.4%**
$\alpha = 1$		
Pixel-loss	**26.9%**	9.7%
Order-loss	28.2%	**5.2%**
$\alpha = 2$		
Pixel-loss	84.3%	44.9%
Order-loss	**67.5%**	**15.8%**

Figure 6: Synthetic results. Error on Pixel-based error or Order-based error for different values of α and train time loss functions.

11 Additional Qualitative Results

See Figures 2-5 for more results on Cow and for results on Aeroplane, Car, and Dog, respectively. Figure 6 shows failure modes for the PASCAL loss using examples from all of the object classes.

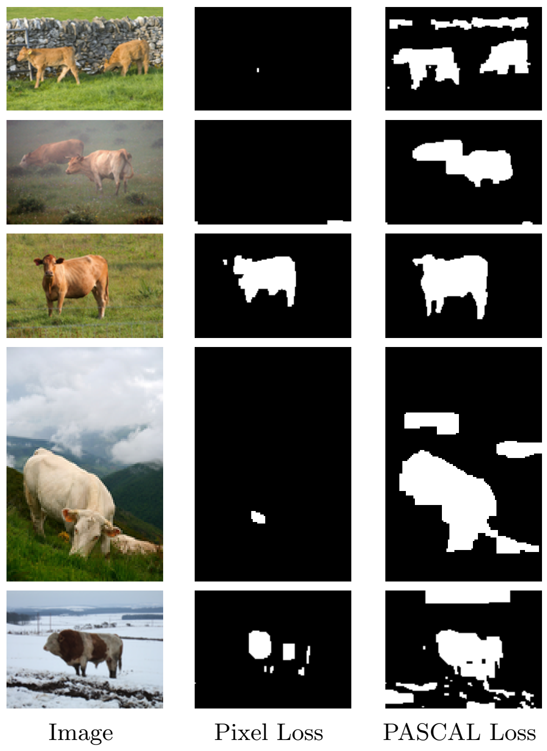


Figure 7: More test results on Cow dataset. Methods from left to right: (Left) Raw image. (Middle) Pixel Loss. (Right) PASCAL Loss.

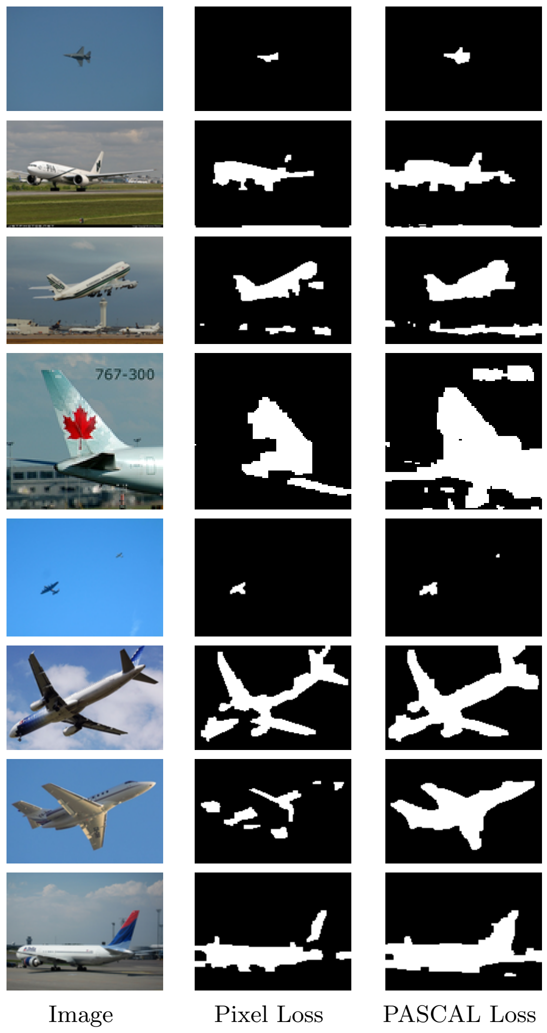


Figure 8: Test results on Aeroplane dataset. Methods from left to right: (Left) Raw image. (Middle) Pixel Loss. (Right) PASCAL Loss.

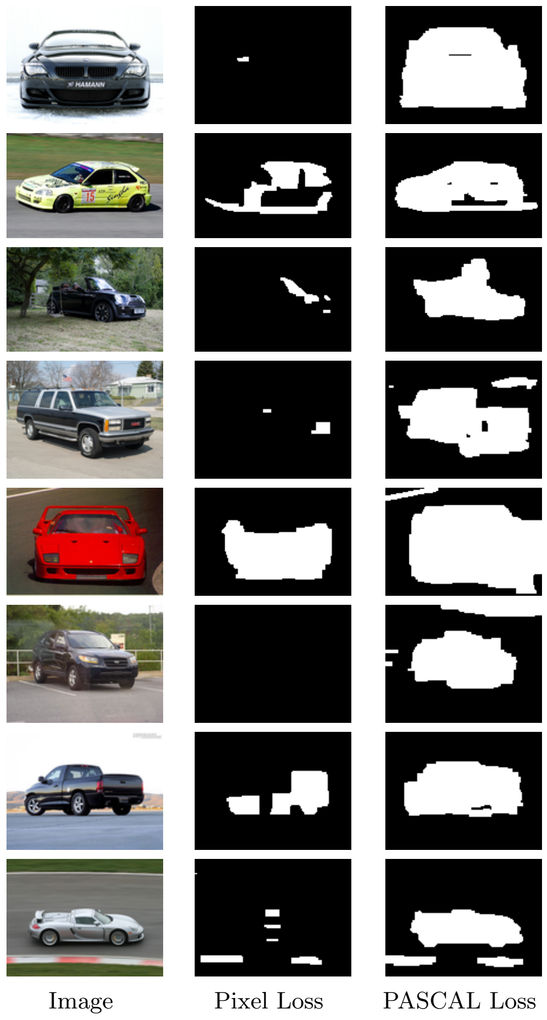


Figure 9: Test results on Car dataset. Methods from left to right: (Left) Raw image. (Middle) Pixel Loss. (Right) PASCAL Loss.

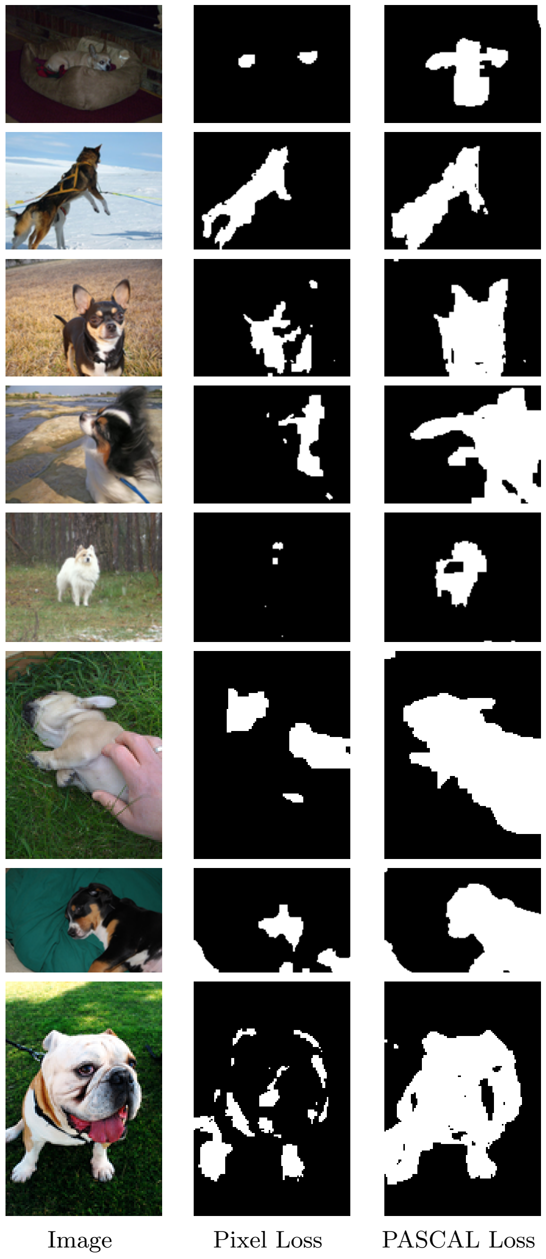


Figure 10: Test results on Dog dataset. Methods from left to right: (Left) Raw image. (Middle) Pixel Loss. (Right) PASCAL Loss.

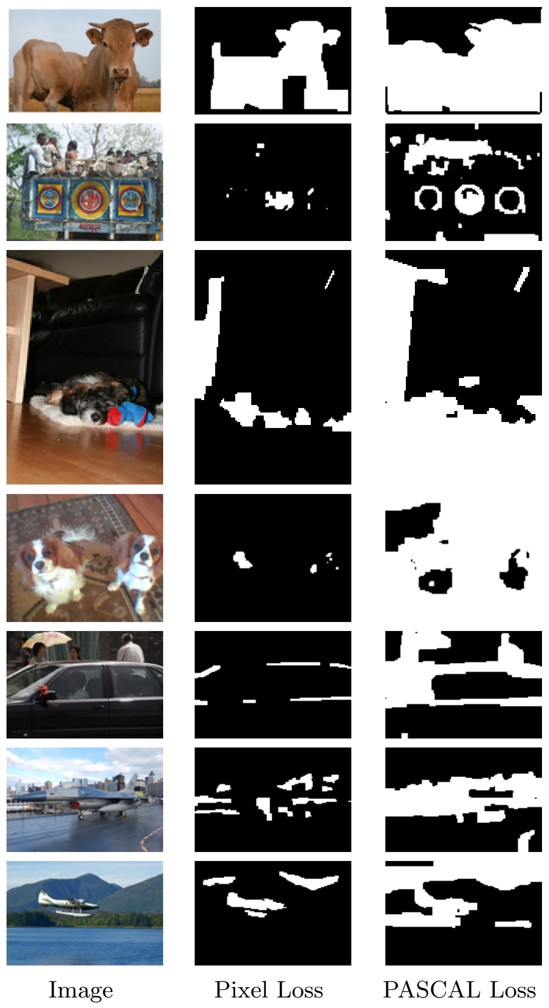


Figure 11: Typical failures of PASCAL loss-trained models. On difficult images, the PASCAL loss-trained model tends to label too many pixels as foreground. Errors are sometimes exaggerated when edge information in the image is weak, as in the second dog example. Methods from left to right: (Left) Raw image. (Middle) Pixel Loss. (Right) PASCAL Loss.