Graph Structured Prediction Energy Net Algorithms

Colin Graber cgraber2@illinois.edu
Alexander Schwing aschwing@illinois.edu
University of Illinois at Urbana-Champaign; Champaign, IL; USA

Abstract

For joint inference over multiple variables, a variety of structured prediction techniques have been developed to model correlations among variables and thereby improve predictions. However, many classical approaches suffer from one of two primary drawbacks: they either lack the ability to model high-order correlations among variables while maintaining computationally tractable inference, or they do not allow to explicitly model known correlations. To address this shortcoming, we introduce 'Graph Structured Prediction Energy Networks,' for which we develop inference techniques that allow to both model explicit local and implicit higher-order correlations while maintaining tractability of inference. We demonstrate the general utility via tasks from the computer vision domain.

1 Introduction

Many machine learning tasks involve joint prediction of a set of variables. For instance, semantic image segmentation infers the class label for every pixel in an image. To address joint prediction, it is common to use deep nets which model probability distributions independently over the variables (*e.g.*, the pixels). The downside: correlations between different variables aren't modeled explicitly.

A number of techniques, such as Structured SVMs [1], Max-Margin Markov Nets [2] and Deep Structured Models [3, 4], directly model relations between output variables. But modeling the correlations between a large number of variables is computationally expensive and therefore generally impractical. Hence, SPENS [5, 6] were introduced. SPENs assign a score to an entire prediction, which allows them to both harness global structure and learn complex relations between variables tractably. However, Belanger and McCallum [5] note that it is easy to overfit SPENs to the training data. Additionally, the inference techniques developed for SPENs do not enforce structural constraints among output variables, *i.e.*, they cannot support structured scores and discrete losses. An attempt to combine locally structured scores with joint prediction was introduced very recently by Graber et al. [7]. However, they require the score function to take a specific, restricted form, and inference is formulated as a difficult-to-solve saddle-point optimization problem.

To address these concerns, we develop a new model which we refer to as 'Graph Structured Prediction Energy Network' (GSPEN). GSPENs combine the capabilities of classical structured prediction models and SPENs and have the ability to explicitly model local structure when known or assumed, while providing the ability to learn an unknown or more global structure implicitly. Additionally, the proposed GSPEN formulation generalizes the NLStruct approach by Graber et al. [7]. We show the utility of GSPENs by comparing to related techniques on several tasks: optical character recognition, image tagging, and multilabel classification. In general, we show that GSPENs are able to outperform other models. Our implementation is available at `https://github.com/cgraber/GSPEN`. Please see our main conference paper [8] for a more detailed description.

2 Background

Let $x \in \mathcal{X}$ represent the input provided to a model, such as a sentence or an image. In this work, we consider tasks where the outputs take the form $y = (y_1, \ldots, y_K) \in \mathcal{Y} \coloneqq \prod_{k=1}^{K} \mathcal{Y}_k$, *i.e.*, they are vectors where the k-th variable's domain is the discrete and finite set $\mathcal{Y}_k = \{1, \ldots, |\mathcal{Y}_k|\}$. In general,

NeurIPS 2019 Workshop on Graph Representation Learning, Vancouver, Canada

the number of variables K which are part of the configuration y can depend on the observation x. However, for readability only, we assume all $y \in \mathcal{Y}$ contain K entries.

Prior structured and unstructured models use a function $F(x, y; w)$ which assigns a score to a given configuration y conditioned on input x and is parameterized by weights w. Inference maximizes this function with respect to y. Algorithms vary depending on the form of this function. Unstructured models, such as feed-forward deep nets, assign a score to each label of variable y_k irrespective of the label choice of other variables. This permits to independently find the maximum score for each variable y_k. Classical structured models incorporate dependencies between variables by considering functions that each depend on a subset $r \subseteq \{1, \ldots, K\}$ of the output variables. We refer to the subset of variables via $y_r = (y_k)_{k \in r}$ and use f_r to denote the corresponding function. This formulation allows to explicitly model relations between variables, but it comes at the price of more complex inference which is generally NP-hard [9]. Approximations to this problem have been developed and utilized successfully. But the complexity of these methods scales with the size of the largest region r. For this reason, these models often consider only regions with one or two variables.

Structured Prediction Energy Networks (SPENs) [5] were motivated by the desire to represent interactions between larger sets of output variables without incurring a high computational cost. The SPEN score function takes the form $F(x, p_1, \ldots, p_K; w) \coloneqq T\left(\bar{f}(x; w), p_1, \ldots, p_K; w\right)$, where $\bar{f}(x; w)$ is a learned feature representation of the input x, each p_k is a one-hot vector, and T is a function that takes these two terms and assigns a score. This representation of the labels, *i.e.*, p_k, is used to facilitate inference, which is computed via gradient-based optimization techniques.

3 Graph Structured Prediction Energy Nets

Graph Structured Prediction Energy Networks (GSPENs) generalize all aforementioned models. They combine both a classical structured component as well as a SPEN-like component to score an entire set of predictions jointly. Additionally, GSPEN includes NLStruct as a special case. The GSPEN score function is written as follows:

$$F\left(x, p_{\mathcal{R}}; w\right) \coloneqq T\left(\bar{f}(x; w), p_{\mathcal{R}}; w\right),$$

where vector $p_{\mathcal{R}} \coloneqq (p_r(y_r))|_{r \in \mathcal{R}, y_r \in \mathcal{Y}_r}$ contains one marginal per region per assignment of values to that region. This formulation allows for the use of a structured score function while also allowing T to score an entire prediction jointly. Hence, it is a combination of classical structured models and SPENs. For instance, we can construct a GSPEN model by summing a classical structured model and a multilayer perceptron that scores an entire label vector, in which case the score function takes the form $F(x, p_{\mathcal{R}}; w) \coloneqq \sum_{r \in \mathcal{R}} \sum_{y_r \in \mathcal{Y}_r} p_r(y_r) f_r(x, y_r; w) + \text{MLP}\left(p_{\mathcal{R}}; w\right)$. Of course, this is one of many possible score functions that are supported by this formulation. Notably, we recover the NLStruct score function if we use $T(\bar{f}(x; w), p_{\mathcal{R}}; w) = T'(\bar{f}(x; w) \circ p_{\mathcal{R}}; w)$ and let $\bar{f}(x; w) = f_{\mathcal{R}}(x; w)$.

Given this model, the inference problem is

$$\max_{p_{\mathcal{R}} \in \mathcal{M}} T\left(\bar{f}(x; w), p_{\mathcal{R}}; w\right). \tag{1}$$

The probabilities are constrained to lie in the marginal polytope $\mathcal{M}$. In addition, an entropy term over the predictions may optionally be added to the objective to smooth optimization. The results discussed below indicate that adding entropy leads to better-performing models. Also note that it is possible to add a similar entropy term to the SPEN inference objective, which is mentioned by Belanger and McCallum [5] and Belanger et al. [6].

For inference in GSPEN, SPEN procedures cannot be used since they do not maintain the additional constraints imposed by the graphical model, *i.e.*, the marginal polytope $\mathcal{M}$. We also cannot use the inference procedure developed for NLStruct, as the GSPEN score function does not take the same form. Therefore, in the following, we describe two inference algorithms that optimize the program while maintaining structural constraints.

Frank-Wolfe. The Frank-Wolfe algorithm [10] is suitable because the objective in Eq. (1) is non-linear while the constraints are linear. Specifically, using [10], we compute a linear approximation of the objective at the current iterate, maximize this linear approximation subject to the constraints of the original problem, and take a step towards this maximum. This inner maximization problem is equivalent to inference for classical structured prediction. Hence we can use any technique developed for this problem, *e.g.*, [11–44].

$$\min_{w} \sum_{\left(x^{(i)}, p_{\mathcal{R}}^{(i)}\right)} \left[\max_{\hat{p}_{\mathcal{R}} \in \mathcal{M}} \left\{ T\left(\bar{f}(x; w), \hat{p}_{\mathcal{R}}; w\right) + L\left(\hat{p}_{\mathcal{R}}, p_{\mathcal{R}}^{(i)}\right) \right\} - T\left(\bar{f}\left(x^{(i)}; w\right), p_{\mathcal{R}}^{(i)}; w\right) \right]_{+}$$

Figure 1: The GSPEN learning formulation, consisting of a Structured SVM (SSVM) objective with loss-augmented inference. Note that each $p_{\mathcal{R}}^{(i)}$ are one-hot representations of labels y_i.

Convergence guarantees for Frank-Wolfe have been proven when the overall objective is concave, continuously differentiable, and has bounded curvature [45], which is the case when T has these properties with respect to the marginals. This is true even when the inner optimization is only solved approximately, which is often the case due to standard approximations used for structured inference. When T is non-concave, convergence can still be guaranteed, but only to a local optimum [46]. Note that entropy has unbounded curvature, therefore its inclusion in the objective precludes convergence guarantees. Other variants of the Frank-Wolfe algorithm exist which improve convergence in certain cases [47]. We do not experiment with these here as we didn't observe any issues.

Structured Entropic Mirror Descent. Mirror descent, another constrained optimization algorithm, is analogous to projected subgradient descent, albeit using a more general distance beyond the Euclidean one [48]. This algorithm has been used in the past to solve inference for SPENs, where entropy was used as the link function ψ and by normalizing over each coordinate independently [5]. We similarly use entropy in our case; given this choice, the "projection" step also reduces to a classical structured prediction problem. When the inference objective is concave and Lipschitz continuous (*i.e.*, when T has these properties), this algorithm has also been proven to converge [48]. Unfortunately, we are not aware of any convergence results if the inner optimization problem is solved approximately and if the objective is not concave. In practice, though, we did not observe any convergence issues during experimentation.

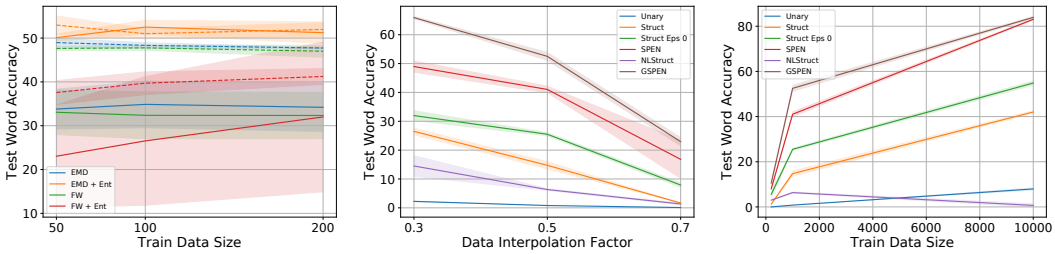
Learning GSPEN Models. GSPENs assign a score to an input x and a prediction p. An SSVM learning objective is applicable, which maximizes the margin between the scores assigned to the correct prediction and the inferred result. The full SSVM learning objective with added loss-augmented inference is summarized in Fig. 1. The learning procedure consists of computing the highest-scoring prediction using one of the previously described inference procedures for each example in a mini-batch and then updating the weights of the model towards making better predictions.

4 Experiments

To demonstrate the utility of our model and to compare inference and learning settings, we report results on the tasks of optical character recognition (OCR), image tagging, and multilabel classification. For each experiment, we also use the following baselines: Unary (an unstructured model), Struct (a classical structured model), SPEN, and NLStruct. For GSPENs, the inner structured inference problems are solved using the same algorithm as for Struct. Additional experimental details, including hyper-parameter settings, are provided in Appendix A.1.

Optical Character Recognition (OCR). For the OCR experiments, we generate data by rendering letters on top of high-variance backgrounds; more details for this process can be found in Appendix A.1. We create three versions of this dataset using different letter interpolation factors of $\alpha \in \{0.3, 0.5, 0.7\}$, where each pixel in the final image is computed as $\alpha x_{\text{background}} + (1 - \alpha) x_{\text{letter}}$. This process was deliberately designed to ensure that information about the structure of the problem (*i.e.*, which words exist in the data) is a strong signal, while the signal provided by each individual letter image can be adjusted. The training, validation, and test set sizes for each dataset are 10,000, 2,000, and 2,000, respectively. During training we vary the training data to be either 200, 1k or 10k.

To study the inference algorithm, we train four different GSPEN models on the dataset containing 1000 training points and using $\alpha = 0.5$. Each model uses either Frank-Wolfe or Mirror Descent and included/excluded the entropy term. To maintain tractability of inference, we fix a maximum iteration count for each model. We additionally investigate the effect of this maximum count on final performance. Additionally, we run this experiment by initializing from two different Struct models, one being trained using entropy during inference and one being trained without entropy. The results for this set of experiments are shown in Fig. 2a. Most configurations perform similarly across the number of iterations, indicating these choices are sufficient for convergence. When initializing from the models trained without entropy, we observe that both Frank-Wolfe without entropy and Mirror Descent with entropy performed comparably. When initializing from a model trained with entropy, the use of mirror descent with entropy led to much better results.



(a) (b) (c)

Figure 2: Experimental results on OCR data. The dashed lines in (a) represent models trained from Struct w odels trained from Struct with entropy.

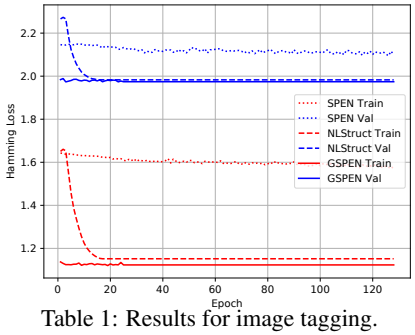


Table 1: Results for image tagging.

Table 2: Multilabel classification results for all models. Entries are macro F1 scores. The top results are taken from the cited works.

	Bibtex		Bookmarks	
	Validation	Test	Validation	Test
SPEN [5]	–	42.2	–	34.4
DVN [51]	–	44.7	–	37.1
Unary	43.3	44.1	38.4	37.4
Struct	45.8	46.1	39.7	38.9
SPEN	46.6	46.5	40.2	39.2
GSPEN	**47.5**	**48.6**	**41.2**	**40.7**

The results for all values of α using a train dataset size of 1000 are presented in Fig. 2b, and results for all train dataset sizes with $\alpha = 0.5$ are presented in Fig. 2c. We observe that, in all cases, GSPEN outperforms all baselines. The degree to which GSPEN outperforms other models depends most on the amount of train data: with a sufficiently large amount of data, SPEN and GSPEN perform comparably. However, when less data is provided, GSPEN performance does not drop as sharply as that of SPEN. It is also worth noting that GSPEN outperformed NLStruct by a large margin. The NLStruct model is less stable due to its saddle-point formulation. Therefore it is much harder to obtain good performance with this model.

Image Tagging. Next, we evaluate on the MIRFLICKR25k dataset [49], which consists of 25,000 images taken from Flickr. Each image is assigned a subset of 24 possible tags. The train/val/test sets for these experiments consist of 10,000/5,000/10,000 images, respectively. The results are shown in Fig. 1. GSPEN obtains similar test performance to the NLStruct model, and both outperform SPEN. However, the NLStruct model was run for 100 iterations during inference without reaching 'convergence' (change of objective smaller than threshold), while the GSPEN model required an average of 69 iterations to converge at training time and 52 iterations to converge at test time. Our approach has the advantage of requiring fewer variables to maintain during inference and requiring fewer iterations of inference to converge. The final test losses for SPEN, NLStruct and GSPEN are 2.158, 2.037, and 2.029, respectively.

Multilabel Classification. We use the Bibtex and Bookmarks multilabel datasets [50]. They consist of binary-valued input feature vectors, each of which is assigned some subset of 159/208 possible labels for Bibtex/Bookmarks. We train unary and SPEN models with architectures similar to [5] and [51]. The results are given in Table 2 alongside those taken from [5] and [51]. We found that the Unary model performs similarly to or better than previous best results. Both SPEN and Struct are able to improve upon Unary results. GSPEN outperforms all configurations, suggesting that the contributions of the SPEN component and the Struct component to the score function are complementary.

5 Conclusions

The GSPEN model combines the strengths of several prior works for structured prediction. It allows machine learning practitioners to include inductive bias in the form of known structure into a model while implicitly capturing higher-order correlations among output variables. The model formulation described here is more general than previous attempts to combine explicit local and implicit global structure modeling while not requiring inference to solve a saddle-point problem.

Acknowledgments: This work is supported in part by NSF under Grant No. 1718221 and MRI #1725729, UIUC, Samsung, 3M, Cisco Systems Inc. (Gift Award CG 1377144) and Adobe. We thank NVIDIA for providing GPUs used for this work and Cisco for access to the Arcetri cluster.

References

[1] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 2005.

[2] B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *Proc. NIPS*, 2003.

[3] L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning Deep Structured Models. In *Proc. ICML*, 2015.

[4] A. G. Schwing and R. Urtasun. Fully Connected Deep Structured Networks. *arXiv preprint arXiv:1503.02351*, 2015.

[5] D. Belanger and A. McCallum. Structured Prediction Energy Networks. In *Proc. ICML*, 2016.

[6] D. Belanger, B. Yang, and A. McCallum. End-to-end learning for structured prediction energy networks. In *Proc. ICML*, 2017.

[7] C. Graber, O. Meshi, and A. Schwing. Deep structured prediction with nonlinear output transformations. In *Proc. NeurIPS*, 2018.

[8] C. Graber and A. Schwing. Graph structured prediction energy networks. In *Proc. NeurIPS*, 2019.

[9] S. E. Shimony. Finding MAPs for Belief Networks is NP-hard. *AI*, 1994.

[10] M. Frank and P. Wolfe. An algorithm for quadratic programming. *NRLQ*, 1956.

[11] M. I. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic Analysis of Two-Dimensional Visual Signals in Noisy Conditions). *Kibernetika*, 1976.

[12] T. Werner. A Linear Programming Approach to Max-Sum Problem: A Review. *PAMI*, 2007.

[13] Y. Boykov, O. Veksler, and R. Zabih. Markov Random Fields with Efficient Approximations. In *Proc. CVPR*, 1998.

[14] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *PAMI*, 2001.

[15] M. J. Wainwright and M. I. Jordan. Variational Inference in Graphical Models: The View from the Marginal Polytope. In *Proc. Conf. on Control, Communication and Computing*, 2003.

[16] A. Globerson and T. Jaakkola. Approximate Inference Using Planar Graph Decomposition. In *Proc. NIPS*, 2006.

[17] M. Welling. On the Choice of Regions for Generalized Belief Propagation. In *Proc. UAI*, 2004.

[18] D. Sontag, D. K. Choe, and Y. Li. Efficiently Searching for Frustrated Cycles in MAP Inference. In *Proc. UAI*, 2012.

[19] D. Batra, S. Nowozin, and P. Kohli. Tighter Relaxations for MAP-MRF Inference: A Local Primal-Dual Gap Based Separation Algorithm. In *Proc. AISTATS*, 2011.

[20] D. Sontag, T. Meltzer, A. Globerson, and T. Jaakkola. Tightening LP Relaxations for MAP Using Message Passing. In *Proc. NIPS*, 2008.

[21] D. Sontag and T. Jaakkola. New Outer Bounds on the Marginal Polytope. In *Proc. NIPS*, 2007.

[22] M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families and Variational Inference. *FTML*, 2008.

[23] D. Sontag and T. Jaakkola. Tree Block Coordinate Descent for MAP in Graphical Models. In *Proc. AISTATS*, 2009.

[24] K. Murphy and Y. Weiss. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proc. UAI*, 1999.

[25] O. Meshi, A. Jaimovich, A. Globerson, and N. Friedman. Convexifying the Bethe Free Energy. In *Proc. UAI*, 2009.

[26] A. Globerson and T. Jaakkola. Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-Relaxations. In *Proc. NIPS*, 2007.

[27] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A New Class of Upper Bounds on the Log Partition Function. *Trans. Information Theory*, 2005.

[28] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. MAP Estimation via Agreement on Trees: Message-Passing and Linear Programming. *Trans. Information Theory*, 2005.

[29] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-Based Reparameterization Framework for Analysis of Sum-Product and Related Algorithms. *Trans. Information Theory*, 2003.

[30] T. Heskes. Convexity Arguments for Efficient Minimization of the Bethe and Kikuchi Free Energies. *AI Research*, 2006.

[31] T. Hazan and A. Shashua. Norm-Product Belief Propagation: Primal-Dual Message-Passing for LP-Relaxation and Approximate Inference. *Trans. Information Theory*, 2010.

[32] T. Hazan and A. Shashua. Convergent Message-Passing Algorithms for Inference Over General Graphs with Convex Free Energies. In *Proc. UAI*, 2008.

[33] C. Yanover, T. Meltzer, and Y. Weiss. Linear Programming Relaxations and Belief Propagation – An Empirical Study. *JMLR*, 2006.

[34] T. Meltzer, A. Globerson, and Y. Weiss. Convergent Message Passing Algorithms: A Unifying View. In *Proc. UAI*, 2009.

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

[36] T. Heskes. Stable Fixed Points of Loopy Belief Propagation are Minima of the Bethe Free Energy. In *Proc. NIPS*, 2002.

[37] T. Heskes, K. Albers, and B. Kappen. Approximate Inference and Constrained Optimization. In *Proc. UAI*, 2003.

[38] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized Belief Propagation. In *Proc. NIPS*, 2001.

[39] A. T. Ihler, J. W. Fisher, and A. S. Willsky. Message Errors in Belief Propagation. In *Proc. NIPS*, 2004.

[40] W. Wiegerinck and T. Heskes. Fractional Belief Propagation. In *Proc. NIPS*, 2003.

[41] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Globally Convergent Dual MAP LP Relaxation Solvers Using Fenchel-Young Margins. In *Proc. NIPS*, 2012.

[42] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Globally Convergent Parallel MAP LP Relaxation Solver Using the Frank-Wolfe Algorithm. In *Proc. ICML*, 2014.

[43] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Distributed Message Passing for Large Scale Graphical Models. In *Proc. CVPR*, 2011.

[44] N. Komodakis, N. Paragios, and G. Tziritas. MRF Energy Minimization & Beyond via Dual Decomposition. *PAMI*, 2010.

[45] M. Jaggi. Revisiting frank-wolfe: projection-free sparse convex optimization. In *Proc. ICML*, 2013.

[46] S. Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

[47] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Proc. NIPS*, 2015.

[48] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 2003.

[49] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Proc. ICMR*, 2008.

[50] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. *ECML PKDD Discovery Challenge 2008*, 2008.

[51] Michael Gygli, Mohammad Norouzi, and Anelia Angelova. Deep value networks learn to evaluate and iteratively refine structured outputs. In *ICML*, 2017.

[52] T. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proc. VISAPP*, 2009.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proc. NIPS*, 2012.

A Appendix

A.1 Additional Experimental Details

General Details: Unless otherwise specified, all Struct models were trained by using the corresponding pre-trained Unary model, fixing these parameters, and training pairwise potentials. All SPEN models were trained by using the pre-trained Unary model, fixing these parameters, and training the T function. Early stopping based on task performance on validation was used to select the number of epochs for training. For SPEN, GSPEN, and Struct models, loss-augmented inference was used where the loss function equals the sum of the 0-1 losses per output variable, *i.e.*, $L(\hat{y}, y) := \sum_{i=1}^{n} \mathbf{1}[\hat{y}_i \neq y_i]$ where n is the number of output variables.

OCR: For the OCR experiments, we generate data by selecting a list of 50 common 5-letter English words, such as 'close,' 'other,' and 'world.' To create each data point, we choose a word from this list and render each letter as a 28x28 pixel image by selecting a random image of the letter from the Chars74k dataset [52], randomly shifting, scaling, rotating, and interpolating with a random background image patch. A different pool of backgrounds and letter images was used for the training, validation, and test splits of the data. The task is to identify the words given 5 ordered images. We create three versions of this dataset using different interpolation factors of $\alpha \in \{0.3, 0.5, 0.7\}$, where each pixel in the final image is computed as $\alpha x_{\text{background}} + (1 - \alpha) x_{\text{letter}}$.

The Unary model is a single 3-layer multilayer perceptron (MLP) with ReLU activations, hidden layer sizes of 200, and a dropout layer after the first linear layer with keep probability 0.5. Scores for each image were generated by independently passing them into this network. Both Struct and GSPEN use a graph with one pairwise region per pair of adjacent letters, for a total of 4 pairs. Linear potentials are used, containing one entry per per per set of assignments of values to the variable represented by that pair. The score function for both SPEN and GSPEN takes the form $F(x, p; w) = \sum_{r \in \mathcal{R}} \sum_{y_r \in \mathcal{Y}_r} p_r(y_r) b_r(x, y, w) + T(B(x), p)$, where in the SPEN case $\mathcal{R}$ contains only unary regions and in the GSPEN case $\mathcal{R}$ consists of the graph used by Struct. Each b_r represents the outputs of the same model as Unary/Struct for SPEN/GSPEN, respectively, and $B(x)$ represents the vector $(b_r(x, y_i; w)|_{y_r \in \mathcal{Y}_r})$. For every SPEN and GSPEN model trained, T is a 2-layer MLP with softplus activations, an output size of 1, and either 500, 1000, or 2000 hidden units. These hidden sizes as well as the number of epochs of training for each model were determined based on task performance on the validation data. Message-passing inference used in both Struct and GSPEN ran for 10 iterations. GSPEN models were trained by using the pre-trained Struct model, fixing these parameters, and training the T function. The NLStruct model consisted of a 2-layer MLP with 2834 hidden units, an output size of 1, and softplus activations. We use the same initialization described by Graber et al. [7] for their word recognition experiments, where the first linear layer was initialized to the identity matrix and the second linear layer was initialized to a vector of all 1s. NLStruct models were initialized from the Struct models trained without entropy and used fixed potentials. The inference configuration described by Graber et al. [7] was used, where inference was run for 100 iterations with averaging applied over the final 50 iterations.

All settings for the OCR experiments used a mini-batch size of 128 and used the Adam optimizer, with Unary, SPEN, and GSPEN using a learning rate of 10^{-4} and Struct using a learning rate of 10^{-3}. Gradients were clipped to a norm of 1 before updates were applied. Inference in both SPEN and GSPEN were run for a maximum of 100 iterations. Inference was terminated early for both models if the inference objective for all datapoints in the minibatch being processed changed by less than 0.0001.

Three different versions of every model, initialized using different random seeds, were trained for these experiments. The plots represent the average of these trials, and the error represented is the standard deviation of these trials.

Tagging: We initialize the structured portion of our GSPEN model for the image tagging experiment using the pre-trained DeepStruct model described by Graber et al. [7], which consists of unary potentials produced from an AlexNet architecture [53] and linear pairwise potentials of the form $f_{i,j}(y_i, y_j, W) = W_{i,j,x_i,x_j}$, *i.e.*, containing one weight per pair in the graph per assignment of values to that pair. A fully-connected pairwise graph was used. Both GSPEN and SPEN use the same score function as in the OCR experiments, with the exception that the T function used for GSPEN is

only a function of the beliefs and does not include the potentials as input. The T function for our GSPEN model consists of a 2-layer MLP with 130 hidden units. It takes as input a concatenation of the unary potentials generated by the AlexNet model and the current prediction. The T function for the SPEN model has the same number of layers and hidden units. We used Frank-Wolfe without entropy for both SPEN and GSPEN inference. Both models were trained using gradient descent with a learning rate of 10^{-2}, a momentum of 0.9, and a mini-batch size of 128. Once again, only the T component was trained for GSPEN, and the pairwise potentials were initialized to a Struct model trained using the settings described in Graber et al. [7].

The message-passing procedure used to solve the inner optimization problem for GSPEN was run for 100 iterations per iteration of Frank-Wolfe. Inference for SPEN and GSPEN was run for 100 iterations and was terminated early if the inference objective for all datapoints in the minibatch being processed changed by less than 0.0001.

Multilabel Classification: For the Bibtex dataset, 25 percent of the training data was set aside to be used as validation data; this was not necessary for Bookmarks, which has a pre-specified validation dataset. For prediction in both datasets and for all models, a threshold determining the boundary between positive/negative label predictions was tuned on the validation dataset.

For the Bibtex dataset, the Unary model consists of a 3-layer MLP taking the binary feature vectors as input and returning a 159-dimensional vector representing the potentials for label assignments $y_i = 1$; the potentials for $y = 0$ are fixed to 0. The Unary model uses ReLU activations, hidden unit sizes of 150, and dropout layers before the first and second linear layers with keep probability of 0.5. The Struct model consists of a 2-layer MLP which also uses the feature vector as input, and it contains 1000 hidden units and ReLU activations. The SPEN model uses the same scoring function form as used in the previous experiments, except the T function is only a function of the prediction vector and does not use the unary potentials as input. The T model consists of a 2-layer MLP which takes the vector $(p_i(y_i = 1))_{i=1}^{59}$ as input. This model has 16 hidden units, an output size of 1, and uses softplus activations. The GSPEN model was trained by starting from the SPEN model, fixing these parameters, and training a pairwise model with the same architecture as the Struct model.

For the bookmarks dataset, the models trained use the same architectures with slightly different configurations. the Unary model consists of a similar 3-layer MLP, except dropout is only applied before the second linear layer. The Struct model uses the same architecture as the one trained on the Bibtex data. The T model for SPEN/GSPEN uses 15 hidden units.

For both datasets and for both SPEN and GSPEN, mirror descent was used for inference with an additional entropy term with a coefficient of 0.1; for Struct, a coefficient of 1 was used. Inference was run for 100 iterations, with early termination as described previously using the same threshold. For Struct and GSPEN, message passing inference was run for 5 iterations. The Unary model was trained using gradient descent with a learning rate of 10^{-2} and a momentum of 0.9, while Struct, SPEN and GSPEN were trained using the Adam optimizer with a learning rate of 10^{-4}.