Learning Deep Structured Models

Liang-Chieh Chen* LCCHEN@CS.UCLA.EDU
University of California Los Angeles, USA
Alexander G. Schwing* ASCHWING@CS.TORONTO.EDU
University of Toronto, 10 King's College Rd., Toronto, Canada
Alan L. Yuille YUILLE@STAT.UCLA.EDU
University of California Los Angeles, USA
Raquel Urtasun URTASUN@CS.TORONTO.EDU
University of Toronto, 10 King's College Rd., Toronto, Canada

* **equal contribution**

Abstract

Many problems in real-world applications involve predicting several random variables that are statistically related. Markov random fields (MRFs) are a great mathematical tool to encode such dependencies. The goal of this paper is to combine MRFs with deep learning to estimate complex representations while taking into account the dependencies between the output random variables. Towards this goal, we propose a training algorithm that is able to learn structured models jointly with deep features that form the MRF potentials. Our approach is efficient as it blends learning and inference and makes use of GPU acceleration. We demonstrate the effectiveness of our algorithm in the tasks of predicting words from noisy images, as well as tagging of Flickr photographs. We show that joint learning of the deep features and the MRF parameters results in significant performance gains.

1. Introduction

Deep learning algorithms attempt to model high-level abstractions of the data using architectures composed of multiple non-linear transformations. A multiplicity of variants have been proposed (Hinton et al., 1984; LeCun et al., 1998; Hinton & Salakhutdinov, 2006; Bengio et al., 2007; Salakhutdinov & Hinton, 2012; Zeiler & Fergus, 2014) and shown to be extremely successful in a wide variety of applications including computer vision, speech recognition as well as natural language processing (Lee et al., 2009; Socher et al., 2012; Jia, 2013; Krizhevsky et al., 2013; Eigen et al., 2014). Recently, state-of-the-art results have been achieved in many computer vision tasks, outperforming competitive methods by a large margin (Krizhevsky et al., 2013; Girshick et al., 2014).

Deep neural networks can, however, be even more powerful when combined with graphical models in order to capture the statistical dependencies between the variables of interest. For example, Deng et al. (2014) exploit mutual exclusion, overlapping and subsumption properties of class labels in order to better predict in large scale classification tasks. In pose estimation, more accurate predictions can be obtained when encoding the spatial relationships between joint locations (Tompson et al., 2014).

It is, however, an open problem how to develop scalable deep learning algorithms that can learn higher-order knowledge taking into account the output variables' dependencies. Existing approaches often rely on a two-step process (Nowozin et al., 2011; Xu et al., 2014) where a non-linear classifier that employs deep features is trained first, and its output is used to generate potentials for the structured predictor. This piece-wise training is, however, suboptimal as the deep features are learned while ignoring the dependencies between the variables of interest. For example, in object recognition, independently learned segmentation and detection features (Hariharan et al., 2014) might be focusing on predicting the same examples correctly, but when learned jointly, they can improve their predictive power by exploiting complementary information to fix additional mistakes.

In this paper we extend deep learning algorithms to learn complex representations taking into account the dependencies between the output random variables. Towards this goal, we propose a learning algorithm that is able to learn structured models with arbitrary graphs jointly with deep features that form potentials in a Markov random field (MRF). Our approach is efficient as it blends learning and inference, resulting in a single loop algorithm which makes use of GPU acceleration. We demonstrate the effectiveness

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

of our method in the tasks of predicting words from noisy images, and tagging of Flickr photographs. We show that joint learning of deep features and MRF parameters results in big performance gains.

2. Learning Deep Structured Models

In this section we investigate how to learn deep features taking into account the dependencies between the output variables. Let $y \in \mathcal{Y}$ be the set of random variables $y = (y_1, \dots, y_N)$ that we are interested in predicting. We assume the space of valid configurations to be a product space, *i.e.*, $\mathcal{Y} = \prod_{i=1}^{N} \mathcal{Y}_i$, and the domain of each individual variable y_i to be discrete, *i.e.*, $\mathcal{Y}_i = \{1, \dots, |\mathcal{Y}_i|\}$. Given input data $x \in \mathcal{X}$ and parameters $w \in \mathbb{R}^A$ of the function $F(x, y; w) : \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^A \to \mathbb{R}$, inference amounts to finding the highest scoring configuration

$$y^* = \arg\max_y F(x, y; w).$$

Note that if F is a deep network, *i.e.*, a composite function, and there are no connections between the output variables to be predicted, inference corresponds to a forward pass to evaluate the function, followed by independently finding the largest response for each variable. This can be interpreted as inference in a graphical model with only unary potentials. However, for arbitrary graphical models it is NP-hard to find the maximizing configuration y^* since the inference program generally requires a search over a space of size $\prod_{i=1}^{N} |\mathcal{Y}_i|$. Note also that log-linear models are a special case of this program, with $F(x, y; w) = w^\top \phi(x, y)$ and $\phi(x, y)$ denoting a feature vector, computed using the input-output pair (x, y).

In this work, we consider the general setting where $F(x, y; w)$ is an arbitrary scalar-valued function of w and (x, y). In our experiments F is a function composition of non-linear base mappings such as convolutions, rectifications and pooling. We let the probability of an arbitrary configuration $\hat{y}$ be given by the annealed soft-max

$$p_{(x,y)}(\hat{y}|w, \epsilon) = \frac{1}{Z_\epsilon(x, w)} \exp(F(x, \hat{y}; w))^{1/\epsilon}.$$

Hereby $Z_\epsilon(x, w)$ refers to the partition function, normalizing the distribution $p_{(x,y)}$ to lie within the probability simplex Δ via $Z(x, w) = \sum_{\hat{y} \in \mathcal{Y}} \exp(F(x, \hat{y}; w))^{1/\epsilon}$. The annealing parameter $\epsilon \geq 0$ is used to adjust the uniformity of the distribution. We consider general graphical models where the computation of $Z_\epsilon(x, w)$ is #P-hard.

2.1. Learning via gradient descent

During learning, given a training set $\mathcal{D}$ of input-output pairs $(x, y) \in \mathcal{D}$, we are interested in finding the parameters w of the model. We do so by maximizing the data likelihood, *i.e.*, minimizing the negative log-likelihood $-\ln \prod_{(x,y) \in \mathcal{D}} p_{(x,y)}(y|w, \epsilon)$ which yields

$$\min_w \sum_{(x,y) \in \mathcal{D}} \left(\epsilon \ln Z_\epsilon(x, w) - F(x, y; w) \right). \tag{1}$$

Algorithm: Deep Structured Learning
Repeat until stopping criteria

1. Forward pass to compute $F(x, \hat{y}; w)$
2. Obtain $p_{(x,y)}(\hat{y}|w, \epsilon)$ via a soft-max
3. Backward pass via chain rule to obtain gradient
4. Update parameters w

Figure 1. Gradient descent for learning deep structured models.

Note that this is equivalent to maximizing the cross-entropy between a target distribution $p_{(x,y),\mathrm{tg}}(\hat{y}) = \delta(\hat{y} = y)$ placing all its mass on the groundtruth label, and the model distribution $p_{(x,y)}(\hat{y}|w, \epsilon)$. Hence Eq. (1) is equivalently obtained by $\max_w \sum_{(x,y),\hat{y} \in \mathcal{Y}} p_{(x,y),\mathrm{tg}}(\hat{y}) \ln p_{(x,y)}(\hat{y}|w, \epsilon)$. It is easily possible to incorporate more general target distributions into Eq. (1). Note also that regularization can be included and $\epsilon \to 0$ recovers the general structured hinge loss objective $\min_w \sum_{(x,y) \in \mathcal{D}} (\max_{\hat{y}} F(x, \hat{y}; w) - F(x, y; w))$, since a margin term is easily incorporated.

Minimizing Eq. (1) w.r.t. w requires computation of the gradient $\frac{\partial}{\partial w} \sum_{(x,y)} - \ln p_{(x,y)}(y|w, \epsilon)$, which is given by a transformed difference between the distributions of the model $p_{(x,y)}(\hat{y}|w, \epsilon)$ and the target $p_{(x,y),\mathrm{tg}}(\hat{y})$:

$$\sum_{(x,y) \in \mathcal{D}} \sum_{\hat{y} \in \mathcal{Y}} \frac{\partial}{\partial w} F(x, \hat{y}; w) \left(p_{(x,y)}(\hat{y}|w, \epsilon) - p_{(x,y),\mathrm{tg}}(\hat{y}) \right). \tag{2}$$

A gradient descent algorithm for minimizing Eq. (1) will iterate between the following steps: (i) For a given w evaluate the function F, (ii) compute the model distribution $p_{(x,y)}(\hat{y}|w, \epsilon)$, (iii) propagate the difference between the model and target distribution using a backward pass (resembling the chain rule for composite functions) and (iv) update the parameters w. This is summarized in Fig. 1.

2.2. Approximate Learning

Note that for general graphical models the exact computation of $p_{(x,y)}(\hat{y}|w, \epsilon)$ is not possible since the state-space size $|\mathcal{Y}| = \prod_{i=1}^{N} |\mathcal{Y}_i|$ is exponential in the number of variables. As a consequence it is intractable to compute the exact gradient of the cost-function given in Eq. (2) and one has to resort to approximate solutions.

Inspired by approximations used for log-linear models, we make use of the following identity (Wainwright & Jordan,

$$\min_{w} \sum_{(x,y)\in\mathcal{D}} \left(\max_{b_{(x,y)}\in\mathcal{C}_{(x,y)}} \left\{ \sum_{r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) f_r(x,\hat{y}_r;w) + \sum_r \epsilon c_r H(b_{(x,y),r}) \right\} - F(x,y;w) \right)$$

Figure 2. The approximated non-linear structured prediction task.

2008; Koller & Friedman, 2009):

$$\epsilon \ln Z_\epsilon(x,w) = \max_{p_{(x,y)}(\hat{y})\in\Delta} \mathbb{E}[F(x,\hat{y};w)] + \epsilon H(p_{(x,y)}), \quad (3)$$

where $\mathbb{E}$ denotes an expectation over the distribution $p_{(x,y)}(\hat{y})$ and H refers to its entropy.

For most applications, $F(x,y;w)$ decomposes into a sum of functions, each depending on a local subset of variables y_r, *i.e.*,

$$F(x,y;w) = \sum_{r\in\mathcal{R}} f_r(x,y_r;w).$$

Hereby r is a restriction of the variable tuple $y = (y_1,\dots,y_N)$ to the subset $r \subseteq \{1,\dots,N\}$, i.e., $y_r = (y_i)_{i\in r}$. All subsets r required to compute the model function F are summarized in the set $\mathcal{R}$. Importantly we note that each local composite function $f_r(x,y_r;w)$ can depend non-linearly on the parameters w.

Plugging this decomposition into Eq. (3), we equivalently get the log-partition function $\epsilon \ln Z_\epsilon(x,w)$ via

$$\max_{p_{(x,y)}(\hat{y})\in\Delta} \sum_{r,\hat{y}_r} p_{(x,y),r}(\hat{y}_r) f_r(x,\hat{y}_r;w) + \epsilon H(p_{(x,y)}),$$

where we use marginals $p_{(x,y),r}(\hat{y}_r) = \sum_{y\setminus y_r} p_{(x,y)}(y)$.

Despite the assumed locality of the scoring function, the learning task remains computationally challenging since the entropy $H(p_{(x,y)})$ can only be computed exactly for a very small set of models, *e.g.*, models for which the dependencies of the joint distribution $p_{(x,y)}(y)$ are equivalently characterized by a low tree-width graph. In addition, the marginalization constraints are exponential in size.

To deal with both issues a common solution in log-linear models is to approximate the true marginals $p_{(x,y),r}$ with local beliefs $b_{(x,y),r}$ that are not required to fulfill marginalization constraints globally, but only locally (Wainwright & Jordan, 2008). That is marginals $b_{(x,y),r}$ are not required to arise from a common joint distribution $p_{(x,y)}$. In addition, we approximate the entropy via the fractional entropy (Wiegerinck & Heskes, 2003), *i.e.*, $H(p_{(x,y)}) \approx \sum_r c_r H(b_{(x,y),r})$. Counting numbers c_r are employed to weight the marginal entropies. Putting all this together, we obtain the following approximation for $\epsilon \ln Z_\epsilon(x,w)$:

$$\max_{b_{(x,y)}\in\mathcal{C}_{(x,y)}} \sum_{r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) f_r(x,\hat{y}_r;w) + \sum_r \epsilon c_r H(b_{(x,y),r}). \quad (4)$$

where the beliefs are constrained to the local polytope

$$\mathcal{C}_{(x,y)} = \begin{cases} \forall r & b_{(x,y),r} \in \Delta \\ \forall r,\hat{y}_r, p\in P(r) & \sum_{\hat{y}_p\setminus\hat{y}_r} b_{(x,y),p}(\hat{y}_p) = b_{(x,y),r}(\hat{y}_r), \end{cases}$$

with $P(r)$ the set of parents of region r, *i.e.*, $P(r) \subseteq \{p \in \mathcal{R} : r \subset p\}$, which subsumes those regions for which we want the marginalization constraint to hold. Conversely, we define the set of children as $C(r) = \{c \in \mathcal{R} : r \in P(c)\}$.

We can thus rewrite the learning problem by plugging the approximations derived in Eq. (4) into Eq. (1). This gives rise to the new approximated learning program depicted in Fig. 2.

To iteratively update the parameters for the non-smooth approximated cost function given in Fig. 2 we require a subgradient w.r.t. w, which in turn requires to solve the maximization w.r.t. the beliefs b exactly. This is a non-trivial task in itself as inference in general graphical models is NP-hard. Iterative message passing algorithms (Pearl, 1988; Yedidia et al., 2005; Wainwright et al., 2005; Weiss et al., 2007; Sontag et al., 2008; Meltzer et al., 2009) are typically employed for approximate inference. Importantly, note that combining the procedure outlined in Fig. 1 with iterative message passing to approximate $p_{(x,y)}(\hat{y}|w,\epsilon)$ results in a double-loop algorithm which would be slow for many graphical models of interest.

2.3. Efficient Approximate Learning by Blending Learning and Inference

In this section we propose a more efficient algorithm that is based on the principle of blending learning (*i.e.*, parameter updates) and inference. Thus we are interested in only performing a single message passing iteration before updating the parameters w. Note that simply reducing the number of iterations is generally not an option as the obtained beliefs $b_{(x,y),r}$ are by no means accurate. However, assuming all counting numbers c_r to be positive, we can derive an algorithm that is able to interleave minimization w.r.t. w and maximization of the beliefs b. Such a procedure is more efficient as we are able to update the parameters w much more frequently.

To interleave both programs we first convert maximization of the beliefs into a minimization by employing the dual program as detailed for general scoring functions in the following claim, which was discussed for log-linear models in seminal work by Taskar et al. (2005). This conversion is possible since the maximization problem is concave in $b_{(x,y)}$ if $\forall r, \epsilon c_r \geq 0$.

Algorithm: Efficient Deep Structured Learning
Repeat until stopping criteria

1. Forward pass to compute $f_r(x, \hat{y}_r; w)\ \forall (x, y), r, y_r$

2. Compute approximate beliefs $b_{(x,y),r} \propto \exp \frac{\hat{f}_r(x,\hat{y}_r;w,\lambda)}{\epsilon c_r}$ by iterating for a fixed number of times over r:

$\forall (x, y), p \in P(r), \hat{y}_r$

$$\mu_{(x,y),p\to r}(\hat{y}_r) = \epsilon c_p \ln \sum_{\hat{y}_p \setminus \hat{y}_r} \exp \frac{f_p(x, \hat{y}_p; w) - \sum_{p' \in P(p)} \lambda_{(x,y),p\to p'}(\hat{y}_p) + \sum_{r' \in C(p)\setminus r} \lambda_{(x,y),r'\to p}(\hat{y}_{r'})}{\epsilon c_p}$$

$$\lambda_{(x,y),r\to p}(\hat{y}_r) \propto \frac{c_p}{c_r + \sum_{p\in P(r)} c_p} \left(f_r(x, \hat{y}_r; w) + \sum_{c\in C(r)} \lambda_{(x,y),c\to r}(\hat{y}_c) + \sum_{p\in P(r)} \mu_{(x,y),p\to r}(\hat{y}_r) \right) - \mu_{(x,y),p\to r}(\hat{y}_r)$$

3. Backward pass via chain-rule to obtain gradient $g = \sum_{(x,y),r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) \nabla_w f_r(x, \hat{y}_r; w) - \nabla_w \overline{F}(w)$

4. Update parameters w using stepsize η via $w \leftarrow w - \eta g$

Figure 3. Efficient learning algorithm that blends learning and inference.

Claim 1 *Assume* $\epsilon c_r \geq 0\ \forall r$, *and let* $\overline{F}(w) = \sum_{(x,y)\in\mathcal{D}} F(x, y; w)$ *denote the sum of empirical function observations. Let* $\lambda_{(x,y),r\to p}(\hat{y}_r)$ *be the Lagrange multipliers for each marginalization constraint* $\sum_{\hat{y}_p \setminus \hat{y}_r} b_{(x,y),p}(\hat{y}_p) = b_{(x,y),r}(\hat{y}_r)$ *within the polytope* $\mathcal{C}_{(x,y)}$. *Then the approximated general structured prediction task shown in Fig. 2 is equivalent to*

$$\min_{w,\lambda} \sum_{(x,y),r} \epsilon c_r \ln \sum_{\hat{y}_r} \exp \frac{\hat{f}_r(x, \hat{y}_r; w, \lambda)}{\epsilon c_r} - \overline{F}(w), \quad (5)$$

where we employed the re-parameterization score $\hat{f}_r(x, \hat{y}_r; w, \lambda) = f_r(x, \hat{y}_r; w) + \sum_{c\in C(r)} \lambda_{(x,y),c\to r}(\hat{y}_c) - \sum_{p\in P(r)} \lambda_{(x,y),r\to p}(\hat{y}_r)$.

Proof: To obtain the dual of the maximization w.r.t. $b_{(x,y)}$ we utilize its Lagrangian

$$L_{(x,y)} = \sum_{r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) \hat{f}_r(x, \hat{y}_r; w, \lambda) + \sum_r \epsilon c_r H(b_{(x,y),r}).$$

Maximization of the Lagrangian w.r.t. the primal variables b is possible by employing the relationship stated in Eq. (3) locally $\forall r$. We then obtain the dual function being the first term in Eq. (5). For strict convexity, *i.e.*, $\epsilon c_r > 0$, we reconstruct the beliefs to be proportional to the exponentiated, loss-augmented re-parameterization score, *i.e.*, formally $b_{(x,y),r} \propto \exp \frac{\hat{f}_r(x,\hat{y}_r;w,\lambda)}{\epsilon c_r}$. For $\epsilon c_r = 0$ the beliefs correspond to a uniform distribution over the set of maximizers of the loss-augmented re-parameterization score $\hat{f}_r(x, \hat{y}_r; w, \lambda)$. ■

It is important to note that by applying duality we managed to convert the min-max task in Fig. 2 into a single minimization as shown in Eq. (5). Performing block coordinate descent updates to minimize Eq. (5), we are therefore able to interleave both, updating the weights (*i.e.*, learning) and the messages (*i.e.*, inference). This results in a more efficient algorithm, as inference does not have to be run until convergence, even a single update of the messages suffices. We note that this is possible only if $\epsilon c_r \geq 0\ \forall r$. Strictly speaking, we require concavity only within the set of feasible beliefs $\mathcal{C}_{(x,y)}$. However, for simplicity of the derivations and descriptions we neglected such an extension.

Fig. 3 summarizes our efficient deep structured prediction algorithm which iterates between the following steps. Given parameters w we perform a standard forward pass to compute $f_r(x, \hat{y}_r; w)$ for all regions r. Depending on the model, computation of f_r can sometimes be carried out more efficiently via a single convolutional neural network which combines all the data. We then iterate through all regions r and use block-coordinate descent to find the globally optimal value of Eq. (5) w.r.t. $\lambda_{(x,y),r\to p}(\hat{y}_r)$ $\forall (x, y), \hat{y}_r, p \in P(r)$. This can be done in closed form and therefore is computed very efficiently (Globerson & Jaakkola, 2007; Sontag et al., 2008; Hazan & Shashua, 2010; Schwing, 2013). We then compute the gradient using a standard backward pass before we jointly update all the parameters w by performing a step of size η along the negative gradient.

2.4. Implementation Details

We implemented the general algorithm presented in Fig. 3 in C++ as a library for Linux, Windows and Mac platforms. It supports usage of the GPU for the forward and backward pass using both, standard linear algebra packages and manually tuned GPU-kernels. In addition to standard gradient descent, we allow specification of both mini-

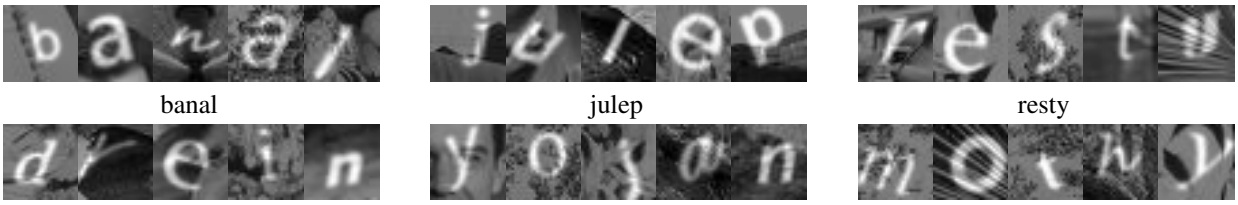


Figure 4. Samples from the Word50 dataset. Note the high degree of rotation, scaling and translation.

Graph	MLP	Method	$H_1 = 128$	$H_1 = 256$	$H_1 = 512$	$H_1 = 768$	$H_1 = 1024$
1st order Markov	One Layer	Unary only	8.60 / 61.32	10.80 / 64.41	12.50 / 65.69	12.95 / 66.66	13.40 / 67.02
		JointTrain	16.80 / 65.28	25.20 / 70.75	**31.80** / 74.90	33.05 / 76.42	34.30 / 77.02
		PwTrain	12.70 / 64.35	18.00 / 68.27	22.80 / 71.29	23.25 / 72.62	26.30 / 73.96
		PreTrainJoint	**20.65 / 67.42**	**25.70 / 71.65**	31.70 / **75.56**	**34.50 / 77.14**	**35.85 / 78.05**
2nd order Markov	One Layer	JointTrain	25.50 / 67.13	34.60 / 73.19	45.55 / 79.60	**51.55 / 82.37**	**54.05 / 83.57**
		PwTrain	10.05 / 58.90	14.10 / 63.44	18.10 / 67.31	20.40 / 70.14	22.20 / 71.25
		PreTrainJoint	**28.15 / 69.07**	**36.85 / 75.21**	**45.75 / 80.09**	50.10 / 82.30	52.25 / 83.39
		$H_1 = 512$	$H_2 = 32$	$H_2 = 64$	$H_2 = 128$	$H_2 = 256$	$H_2 = 512$
1st order Markov	Two Layer	Unary only	15.25 / 69.04	18.15 / 70.66	19.00 / 71.43	19.20 / 72.06	20.40 / 72.51
		JointTrain	35.95 / 76.92	43.80 / 81.64	44.75 / 82.22	46.00 / 82.96	**47.70** / 83.64
		PwTrain	34.85 / 79.11	38.95 / 80.93	42.75 / 82.38	45.10 / 83.67	45.75 / 83.88
		PreTrainJoint	**42.25 / 81.10**	**44.85 / 82.96**	**46.85 / 83.50**	**47.95 / 84.21**	47.05 / **84.08**
2nd order Markov	Two Layer	JointTrain	54.65 / 83.98	61.80 / 87.30	66.15 / 89.09	64.85 / 88.93	68.00 / 89.96
		PwTrain	39.95 / 81.14	48.25 / 84.45	52.65 / 86.24	57.10 / 87.61	62.90 / 89.49
		PreTrainJoint	**62.60 / 88.03**	**65.80 / 89.32**	**68.75 / 90.47**	**68.60 / 90.42**	**69.35 / 90.75**

Table 1. Word / Character accuracy. Performance improves as (1) joint-training is employed, (2) the model is more structured, and (3) deeper unary classifiers are utilized. The number of hidden units for the first and second layer are denoted as H_1 and H_2 respectively.

batches, moments and different regularizers like 2-norm and ∞-norm. Between iterations the step-size can be reduced based on either the negative log-likelihood or validation set performance. Our function F is specified using a directed a-cyclic graph. Hence we support an arbitrarily nested function structure composed of data, parameters and function prototypes (convolution, affine function aka fully connected, dropout, local response normalization, pooling, rectified linear, sigmoid and softmax units). The aforementioned library is accompanied by a program performing learning, inference and gradient checks. To accommodate for large datasets it reads data from HDF5 storage while a second thread simultaneously performs the computation. This is useful since we can prepare the data for the next pass while conducting the computation. Google protocol buffers are employed to effectively specify the function F without the need to modify any source code. The library is released on `http://alexander-schwing.de`.

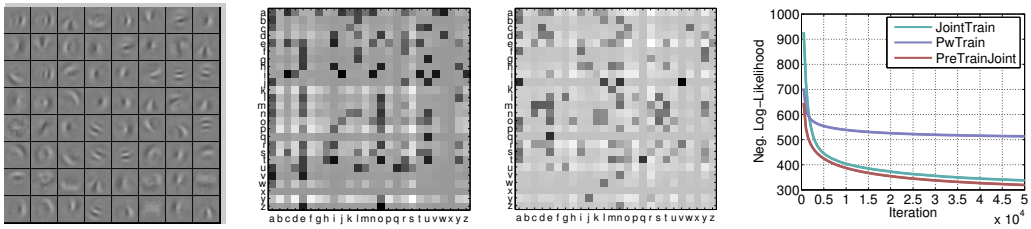
3. Experimental Evaluation

We demonstrate the performance of our model on two tasks: word recognition and image classification. We investigate four strategies to learn the model parameters. '**Unary only**' denotes training only unary classifiers while ignoring the structure of the graphical model, *i.e.*, pairwise weights are equal to 0. '**JointTrain**' initializes all weights at random and trains them jointly. '**PwTrain**' uses piecewise training by first training the unary potentials and then keeping them fixed when learning the pairwise potentials. '**PreTrainJoint**' pre-trains the unaries but jointly optimizes pairwise weights as well as unary weights in a second step.

3.1. Word Recognition: Word50

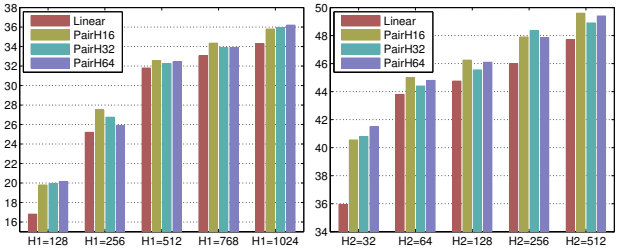
Our first task consists of word recognition from noisy images. Towards this goal, we created a challenging dataset by randomly selecting 50 words, each consisting of five characters. We then generated writing variations of each word as follows: we took the lower case characters from the Chars74K dataset (de Campos et al., 2009), and inserted them in random background image patches (similar to Larochelle et al. (2007)) by alpha matting, *i.e.*, characters have transparency. To increase the difficulty, we perturbed each character image of size 28×28 by scaling, rotation and translation. As shown in Fig. 4 the task is very challenging, some characters are fairly difficult to recognize even for humans. We denote the resulting dataset '**Word50**.' The training, validation and test sets have $10,000$, $2,000$ and $2,000$ variations of words respectively.

We experimented with graphical models composed of unary and pairwise regions defined over five random variables, one per character. We encode unary potentials $f_r(x, y_i; w_u)$ using multi-layer perceptrons (MLPs) with rectified linear units (ReLU). Unless otherwise stated, we



Unary weights | distance-1 edges | distance-2 edges | Neg. Log-Likelihood

Figure 5. (left) Subset of the learned unary weights. Pairwise weights (middle two panels), the darker, the larger the weight. (right) Negative log-likelihood for different learning approaches.



One-Layer MLP Chain | Two-Layer MLP Chain

Figure 6. Learning non-linear pairwise functions: Word recognition as a function of the number of hidden units for the unary potential. Colors represent different number of hidden units for the pairwise potentials. The y-axis shows the word accuracy of using Linear function, or 16 (PairH16), 32 (PairH32), and 64 (PairH64) hidden units for the pairwise function.

define all pairwise interactions via

$$f_r(x, y_i, y_j; w_p) = \sum_{mn} W_{mn} \cdot \delta(y_i = m, y_j = n), \quad (6)$$

where $r = \{i, j\}$, $w_p = \{W\}$, W_{mn} is the element of matrix W, and δ refers to the indicator function.

For all experiments, we share all unary weights across the nodes of the graphical model as well as all pairwise weights for all edges. Note that due to the use of ReLU units, the negative log-likelihood is non-smooth, non-linear and non-convex w.r.t. w. Because of the non-smoothness of F, we utilize momentum based sub-gradient descent methods to estimate the weights. In particular, we use a mini-batch size of 100, a step size of 0.01 and a momentum of 0.95. If the unary potential is pre-trained, the initial step size is reduced to 0.001. All the unary classifiers are trained with $100,000$ iterations over mini-batches. For all experiments, the validation set is only used to decrease the step size, *i.e.*, if the accuracy on the validation set decreases, we reduce the step size by 0.5. We use $\epsilon = 1$, set $c_r = 1$ for all regions r, and perform 10 message passing iterations to compute the marginal beliefs $b_{(x,y),r}$ at step 2 in Fig. 3 when dealing with loopy models.

We experiment with two graphical models, Markov models of first (*i.e.*, there are links only between y_i and y_{i+1}) and second order (*i.e.*, there are links between y_i and y_{i+1}, y_{i+2}) as well as two types of unary potentials with varying degree of structure. We report two metrics, the average character and word accuracy, which correspond to Hamming loss and zero-one loss respectively. Table 1 depicts the results for the different models, learning strategies and number of hidden units. We observe the following trends.

Joint training helps: Joint training with pre-trained unary classifiers (PreTrainJoint) outperforms all the other approaches in almost all cases. Piecewise training (PwTrain), unable to adapt the non-linearities while learning pairwise weights, is worst than joint training.

Structure helps: Adding structure to the model is key to capture complex dependencies. As shown in Table 1, more structured models (*i.e.*, second order Markov model) consistently improves performance.

Deep helps: We tested our models using one layer and two-layer perceptrons with both short-range and long-range connections in the MRF. For the two-layer MLP, the number of hidden units in the first layer is fixed to $H_1 = 512$, and we varied the number of hidden units H_2 in the second layer. As shown in Table 1, the deeper and the more structured the model is, the better the performance we achieve. As expected, performance also increases with the number of hidden units.

Efficiency: Using GPUs, it takes on average 0.064s per iteration for the 1st order Markov model and 0.104s for the 2nd order Markov model. The time employed for training one layer *vs*. the multi-layer models is approximately the same. Note that our approach is very efficient, as this is the time per iteration to train 831,166 weights.

Learned parameters: As shown in the left column of Fig. 5, the learned unary weights resemble character strokes. The middle two panels show the learned pairwise weights for distance-1 edges (*i.e.*, edges with only neighboring connections) and distance-2 edges (*i.e.*, edges connecting every other variable). For example, it shows that 'q' is likely to be followed by 'u,' and 'e' is likely to be distance-2 away from 'q' in this dataset. On the right-most panel, we also show the negative log-likelihood as a function of the number of joint training iterations. PreTrainJoint can achieve the lowest cost value, while PwTrain has the highest value.

female/indoor/portrait	sky/plant life/tree	water/animals/sea	animals/dog/indoor	indoor/flower/plant life
female/indoor/portrait	sky/plant life/tree	water/animals/sky	animals/dog	$\emptyset$

Figure 7. Flickr test set images and a subset of the assigned tags as well as our predictions (bottom row).

Method	Mean error
Unary only	9.36
PwTrain	7.70
PreTrainJoint	**7.25**

Table 2. Flickr Hamming loss: Joint training of deep features and the MRF improves performance.

Non-linear pairwise functions: To further demonstrate the generality of our approach, we replaced the linear pairwise function in Eq. (6) by a one-layer MLP, while keeping the other settings identical. For this experiment we utilize a 1st order Markov model. As shown in Fig. 6, our model attains best performance when using a non-linear pairwise function. We found 16 to 64 hidden units for the non-linear pairwise function to be sufficient for modeling the bi-gram combinations in this dataset. In this case the largest model has 974,846 weights and training takes on average 0.068s per iteration.

3.2. Image Tagging: Flickr

We next evaluate the importance of blending learning and inference. Towards this goal, we make use of the Flickr dataset, which consists of $10,000$ training and $10,000$ test images from Flickr. The task is to predict which of 38 possible tags should be assigned to each image. Fig. 7 shows some examples. The graphical model has 38 binary random variables, each denoting the presence/absence of a particular tag. We define the non-linear unaries $f_r(x, y_i; w_u)$ using the 8-layer deep-net architecture from Krizhevsky et al. (2013), followed by a 76-dimensional top layer. Hence the function is composed out of two subsequent stacks of convolution, rectified linear (ReLU), pooling and local response normalization units. Those are followed by three convolution–ReLU function pairs. Afterwards pooling is applied before two fully-connected–ReLU–dropout combinations are employed to yield the input into a fully connected layer which finally computes the unary potentials. We employ pairwise potentials similar to Eq. (6) which now fully model the correlations between any pair of output variables. This amounts to a total of $57,182,408$ parameters arising from the convolutional units, fully connected units and corresponding biases as well as the pairwise weights.

We use a momentum based sub-gradient method for training with a mini-batch size of 300, a step size of 0.0001, a momentum of 0.95 and set $\epsilon = 1$ and $c_r = 1\ \forall r$. We initialized the deep-net parameters using a model pre-trained on ImageNet (Deng et al., 2009). Our error metric is the classification error, *i.e.*, Hamming loss.

Joint training helps: Results on the test set are summarized in Table 2. Similar to the Word50 dataset we observe that joint training is beneficial. We provide examples for perfect (two left-most images), roughly accurate and failing predictions (right image) in Fig. 7.

Learned pairwise weights: In Fig. 8 we illustrate the learned correlations for a subset of the 38 classes. We observe that the class 'people' correlates highly with 'female,' 'male,' and 'portrait.' The 'indoor' tag does not co-occur with 'sky,' 'structures,' 'plant life' and 'tree.' 'Sea' appears typically with 'water,' 'clouds,' 'lake' and 'sky.'

Efficiency of Blending: To illustrate that blending is indeed beneficial we compare the negative log-likelihood and the training error as a function of run-time in Fig. 9. The standard approach is limited to 20 iterations of message passing to avoid time-consuming, repeated computation of a stopping criterion involving both the approximated log-partition function and its dual. As show in Fig. 9 blending learning and inference speeds up parameter estimation significantly. For larger graphical models, we expect the differences to be even more significant.

4. Discussion

Joint training of neural networks and graphical models: Neural Networks have been incorporated as unary potentials in graphical models. One of the earliest works by Bridle (1990) jointly optimizes a system consisting of multilayer perceptrons and hidden Markov models for speech recognition. For document processing systems, Bottou et al. (1997) propose Graph Transformer Networks to jointly optimize sub-tasks, such as word segmentation and character recognition. Several works (Collobert et al., 2011; Peng et al., 2009; Ma et al., 2012; Do & Artieres, 2010; Prabhavalkar & Fosler-Lussier, 2010; Morris & Fosler-Lussier, 2008) have extended the linear unary

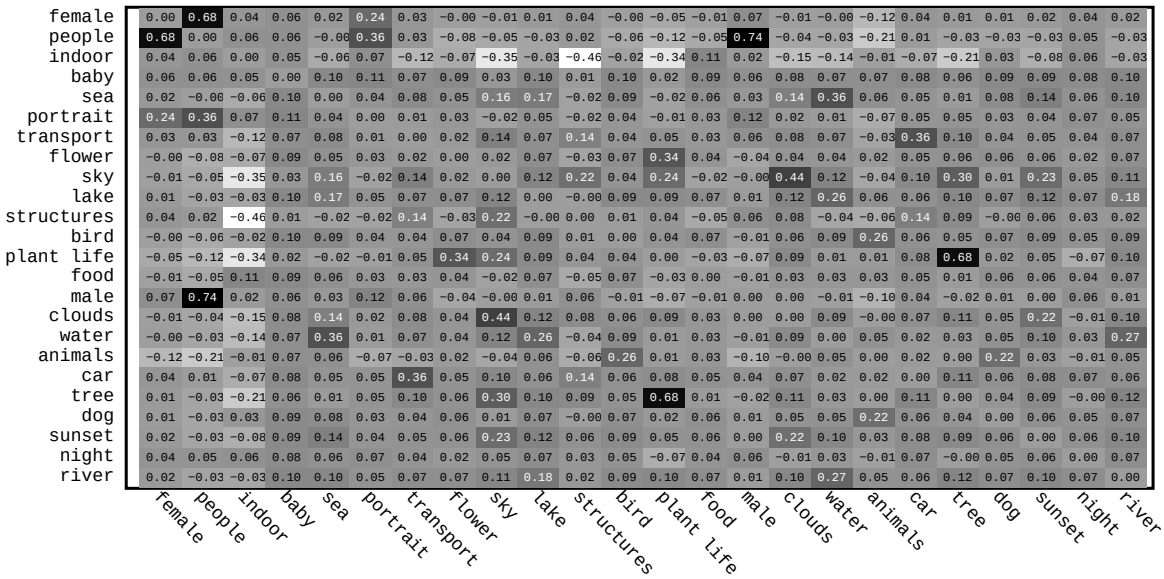


Figure 8. Correlation matrix (*i.e.*, pairwise potentials) learned on the Flickr dataset.

potential in MRFs to incorporate non-linearities. However, they assume that exact inference can be performed either via a forward-backward pass within the graphical model or dynamic programming. In contrast, in this paper we present learning algorithms for general graphical models, where inference is NP-hard. Moreover, all the previous works (except Do & Artieres (2010)) do not consider max-margin loss during training which is incorporated into our framework by choosing $\epsilon = 0$. More recently, Li & Zemel (2014) use a hinge loss to learn the unary term defined as a neural net, but keep the pairwise potentials fixed (*i.e.*, no joint training). Domke (2013) considers non-linear structured prediction and decomposes the learning problem into a subset of logistic regressors, which require the parameter updates to be run till convergence before updating the messages. Tompson et al. (2014) also jointly train convolutional neural networks and a graphical model for pose estimation. However, the MRF inference procedure is approximated by their Spatial-Model which ignores the partition function. Jancsary et al. (2012; 2013) showed the benefits of a combination of structured models with classification trees. Since the submission of our work, Schwing & Urtasun (2015); Zheng et al. (2015) proposed the use of joint training using a double loop algorithm when efficient mean field updates are possible. State-of-the-art was achieved when using the semantic segmentation graphical model of (Chen* et al., 2015).

Blending learning and inference: In this paper we defined learning to be a min-max task. The blending strategy, which was previously employed for learning log-linear models by (Meshi et al., 2010; Hazan & Urtasun, 2010), amounts to converting the maximization task into a minimization problem using its dual. Subsequently we make use of block-coordinate descent strategies to obtain a more efficient algorithm. Importantly any order of block-updates is possible. It remains an open problem to find the optimal tradeoff.

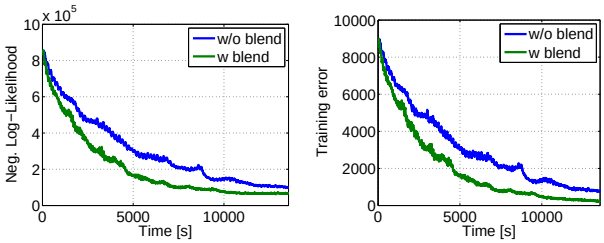


Figure 9. Blending learning and inference speeds-up training significantly.

5. Conclusions

We have proposed an efficient algorithm to learn deep models enriched to capture the dependencies between the output variables. Our experiments on word prediction from noisy images and image tagging showed that the deeper and the more structured the model, the better the performance we achieve. Furthermore, joint learning of all weights outperforms all other strategies. In the future we plan to learn deeper models in applications such as holistic scene understanding. We will also extend our approach to deal with hidden variables as discussed for log-linear models, *e.g.*, in work by Schwing et al. (2012).

Acknowledgments: We thank NVIDIA Corporation for the donation of GPUs used in this research. This work was partially funded by ONR-N00014-14-1-0232, ONR-N00014-12-1-0883, ONR-N000014-10-1-0933 and a Google Faculty Research Award.

References

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy Layer-Wise Training of Deep Networks. In *Proc. NIPS*, 2007.

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

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

Chen*, L.-C., Papandreou*, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *Proc. ICLR*, 2015. * equal contribution.

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

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

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., and Adam, H. Large-Scale Object Classification using Label Relation Graphs. In *Proc. ECCV*, 2014.

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

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

Eigen, D., Rolfe, J., Fergus, R., and LeCun, Y. Understanding Deep Architectures using a Recursive Convolutional Network. In *Proc. ICLR*, 2014.

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

Globerson, A. and Jaakkola, T. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Proc. NIPS*, 2007.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. Simultaneous detection and segmentation. In *Proc. ECCV*, 2014.

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

Hazan, T. and Urtasun, R. A Primal-Dual Message-Passing Algorithm for Approximated Large Scale Structured Prediction. In *Proc. NIPS*, 2010.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 2006.

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. Boltzmann Machines: Constraint Satisfaction Networks that Learn. Technical report, University of Toronto, 1984.

Jancsary, J., Nowozin, S., Sharp, T., and Rother, C. Regression Tree Fields – An Efficient, Non-parametric Approach to Image Labeling Problems. In *Proc. CVPR*, 2012.

Jancsary, J., Nowozin, S., and Rother, C. Learning Convex QP Relaxations for Structured Prediction. In *Proc. ICML*, 2013.

Jia, Y. Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding. http://caffe.berkeleyvision.org/, 2013.

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

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

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proc. ICML*, 2007.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. ICML*, 2009.

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

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

Meltzer, T., Globerson, A., and Weiss, Y. Convergent Message Passing Algorithms: a unifying view. In *Proc. UAI*, 2009.

Meshi, O., Sontag, D., Jaakkola, T., and Globerson, A. Learning Efficiently with Approximate inference via Dual Losses. In *Proc. ICML*, 2010.

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

Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B., and Kohli, P. Decision tree fields. In *Proc. ICCV*, 2011.

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

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

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

Salakhutdinov, R. R. and Hinton, G. E. An Efficient Learning Procedure for Deep Boltzmann Machines. *Neural Computation*, 2012.

Schwing, A. G. *Inference and Learning Algorithms with Applications to 3D Indoor Scene Understanding*. PhD thesis, ETH Zurich, 2013.

Schwing, A. G. and Urtasun, R. Fully Connected Deep Structured Networks. http://arxiv.org/abs/1503.02351, 2015.

Schwing, A. G., Hazan, T., Pollefeys, M., and Urtasun, R. Efficient Structured Prediction with Latent Variables for General Graphical Models. In *Proc. ICML*, 2012.

Socher, R., Huval, B., Bhat, B., Manning, C. D., and Ng, A. Y. Convolutional-Recursive Deep Learning for 3D Object Classification. In *Proc. NIPS*, 2012.

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

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. Learning Structured Prediction Models: A Large Margin Approach. In *Proc. ICML*, 2005.

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

Wainwright, M. J. and Jordan, M. I. *Graphical Models, Exponential Families and Variational Inference*. Foundations and Trends in Machine Learning, 2008.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. A new class of upper bounds on the log partition function. *Trans. Information Theory*, 2005.

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

Wiegerinck, W. and Heskes, T. Fractional belief propagation. In *Proc. NIPS*, 2003.

Xu, J., Schwing, A. G., and Urtasun, R. Tell me what you see and I will show you where it is. In *Proc. CVPR*, 2014.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. Constructing free-energy approximations and generalized belief propagation algorithms. *Trans. Information Theory*, 2005.

Zeiler, M. D. and Fergus, R. Visualizing and Understanding Convolutional Networks. In *Proc. ECCV*, 2014.

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