Lifted Inference for Faster Training (LIFT) in End-to-End Neural-CRF Models

Yatin Nandwani*, Ankit Anand*, Mausam and Parag Singla
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
{yatin.nandwani,ankit.anand,mausam,parags}@cse.iitd.ac.in

Abstract

Several works have explored the use of CRFs as a post-processing step at the end of a neural model to explicitly impose structure in the output space. This has resulted in improved performance in many domains with inherent structure over the output variables e.g., labels over pixels in an image. As an extension, joint training of neural-CRF models has also been explored albeit with limited success. This is due to the high cost of CRF inference which becomes a bottleneck in each iteration of back-propagation. On the other side of the spectrum, there has been tremendous progress in the graphical models community on the topic of lifted inference. Lifted inference algorithms exploit symmetry of the underlying model to significantly reduce the computational cost of graphical model inference. In this paper, we set out to explore the following question: Can the advances in lifted inference be leveraged to speed up the joint training of neural-CRF models? Answering in affirmative, our analysis shows that while pure lifted inference does not help in reducing the cost of training (without compromising the quality), a hybrid approach does the job where we gradually refine the level of lifting as the learning proceeds. As the main contribution of our paper, we present *LIFT: Lifted Inference for Faster Training*, a generic algorithm for faster joint training of neural-CRF models. Experiments in stereo-vision show that our approach can result in up to 50% reduction in training time without any loss in accuracy.

1 Introduction

Neural models have demonstrated impressive performance gains in a large number of applications including those in NLP, vision, speech and biology. Despite their ability to automatically construct useful features from raw data, e.g., words in a document or pixels in an image, they still suffer from the lack of ability to explicitly model dependency in the output space. Though one could argue that with enough amount of data these could automatically be learned, an alternate approach directly tries to capture these dependencies in the modeling phase itself. Graphical models such as CRFs, naturally do this by explicitly modeling a joint distribution over the complete output space. Naturally, existing literature has explored a combination of the two where either (1) a CRF is used as a post-processing step with the neural model to impose structure on the output, e.g. Chen *et al.* [2018] or (2) a neural-CRF model is jointly trained in an end-to-end fashion where the neural model parameters get fine-tuned based on the feedback from the downstream CRF inference, e.g. Zheng *et al.* [2015].

While both of the above approaches have resulted in improved performance for a large number of tasks, with the latter giving even more benefits, a significant challenge remains. The cost of joint training becomes prohibitively expensive primarily due to the high cost of CRF inference which is required during each step of the learning in the back-propagation algorithm. As a temporary

*First two authors contributed equally to the paper

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

remedy, existing literature has explored the use of specialized inference techniques such as DualMM (Shekhovtsov *et al.* [2016]) or ad-hoc heuristics such as running the inference only for a small number of steps (Zheng *et al.* [2015]), or a combination of the two (Knöbelreiter *et al.* [2017]). But to the best of our knowledge, there has not been a concrete solution to reduce the cost of training by tackling the issue of costly CRF inference. On the other hand, there has been a lot of progress in the lifted inference community, which has independently tried to exploit the underlying symmetry resulting in significant speed ups in the cost of graphical model inference (Kimmig *et al.* [2015]). The key idea is to reduce the size of the underlying CRF graph by merging nodes (variables) that behave similar to each other for the task of inference.

In this paper, we set out to explore whether we can leverage the progress in lifted inference to speed up the end-to-end training of neural-CRF models, by making the CRF inference step more efficient. Our analysis shows that vanilla lifting may not be effective for faster training due to loss in quality of the learned parameters. Inspired by the ideas of Habeeb *et al.* [2017], which refined the lifted partitions in a coarse-to-fine manner during inference, we propose a similar scheme for coarse-to-fine *training* where the refinement takes place over different learning iterations. Specifically, while in each learning iteration we fix the level of lifting, we gradually refine it as the learning proceeds. This has the advantage of roughly moving in the direction of gradient in the beginning at a faster pace, while being more precise as learning converges.

As the main contribution of our paper, we present *LIFT: Lifted Inference for Faster Training*, which is a generic coarse-to-fine lifting algorithm for faster training of neural-CRF models. Our algorithm can achieve almost an order of magnitude speed-up in the initial learning epochs, without any loss in quality of final parameters. We demonstrate the usefulness of our approach on a Stereo-Vision dataset Kitti2015 (Menze *et al.* [2018]) where our approach can reduce the training time up to 50%. In the rest of the paper, we will explain our ideas in the context of computer vision problems where coarse-to-fine lifting seems very natural and that is where we have done most of our experimentation. But we believe these ideas can be extended to other tasks where complex CRF inference is needed.

2 Background

One of the popular approaches for the joint training of a neural-CRF model is based on maximizing the margin loss in a Structured SVM framework. Specifically, we follow the framework used by Knöbelreiter *et al.* [2017]. Given an image I having N pixels, each having possible labels $\{1, 2, \cdots, L\}$, let x^g be the ground truth labeling and $\mathcal{X}$ be the set of all labelings. The maximum-margin loss for structured output spaces is as follows:

$$min_{U,H,\lambda}\, l(x^*, x^g) \;\; s.t \;\; x^* \in \underset{x \in \mathcal{X}}{\arg\min}\; E(x) \tag{1}$$

where $E(x)$ is the energy of a labeling x, given by $E(x) = \sum_{i=1}^{N} U(x_i) + \lambda \sum_{x_c \in \mathcal{C}} H(x_c)$. Here, U is unary energy for each pixel while H represents higher order energies defined over cliques $\mathcal{C}$ capturing smoothness constraints among neighbouring pixels. λ parameter captures the trade-off between two energies. Typically, unary energies are learnt from neural networks, while higher order energies are manually defined. The loss function defined above is non-differentiable and hence requires the use of sub-gradients. The sub-gradient of loss w.r.t. U is given by $\delta(x^g) - \delta(\bar{x})$ where $\delta(x^g)$ is an $N \times L$ matrix, each row being a 1-hot encoding of the ground truth label of that pixel and $\bar{x}$ is the solution to the loss-augmented inference (details in Knöbelreiter *et al.* [2017]). The sub-gradient of loss w.r.t. U is backpropagated through the neural network during end-to-end training. The central issue in this optimization is that each sub-gradient computation includes a computationally expensive CRF inference as a sub-routine. While the backpropagation through the whole network is quick, the costly CRF inference discourages one to use CRF in an end-to-end training framework. A typical neural-CRF training procedure is shown in Algorithm 1.

Lifted Inference: On the other hand, there has been a lot of progress towards improving the efficiency of CRF inference algorithms by exploiting symmetries. Many recent works works, e.g. Habeeb *et al.* [2017]; Jernite *et al.* [2015]; Nath and Domingos [2016], have used approximate lifted inference for speeding up prediction in computer vision and natural language processing applications. Habeeb *et al.* [2017] merge image pixels using a top-K labels heuristic and the color passing algorithm (Kersting *et al.* [2009]) to obtain smaller networks of different granularities. These lifted networks are then used to speedup inference in a coarse-to-fine algorithm. Inspired from these works, we intend to leverage the advances in lifted inference to improve the training efficiency of neural-CRF models.

3 LIFT framework

We set out to answer the question: can the power of lifting be used to make the CRF inference step faster in the end-to-end training of neural-CRF models? It turns out that though vanilla (approximate) lifting may not give much benefit without a loss in the quality of training, a more involved approach where we gradually refine the lifting partitions as we proceed through learning iterations results in desired gains without any loss in quality. The key idea is motivated by the fact that during the first few iterations of learning, we only care about the rough gradient direction, whereas we would like to be more precise as the learning proceeds. We name the framework based on this insight as: *LIFT: Lifted Inference for Faster Training*. It is a generic procedure for faster training in neural-CRF models. While Habeeb *et al.* [2017] use coarse-to-fine for efficient inference in graphical models, *LIFT* uses the coarse-to-fine refinement across different learning iterations and the level of lifting is kept constant for solving inference during any given learning iteration.

Algorithm 2 describes *LIFT* in detail. The lines that differ from Algorithm 1 (standard method for joint training) are highlighted in bold. *LIFT* takes an additional parameter $Lift\_Sched$ to specify the coarse-to-fine lifting schedule. In step 6, lifting algorithm, $\mathbf{Lift\_Unaries}$, returns lifted loss augmented unaries based on the schedule for that epoch as specified by $Lift\_Sched$. Step 7 runs the inference algorithm A as in Algorithm 1, but on a reduced CRF graph LU_L. In step 8, the solution obtained for the lifted network is mapped back to the original ground network by assigning the same label to the pixels which fall in the same lifting cluster. Rest of the learning proceeds as earlier using the modified gradients.

Algorithm 1 Base Algorithm

Training(Training Set $\mathbf{I_T}$, Inf-Algo A)

1 Initialize W
2 **for** $i \in \{1, 2 \cdots num\_epochs\}$ **do**
3 **forall** $I \in I_T$ **do**
4 U $\leftarrow$ NN(W, I)
5 $U_L \leftarrow GetLossAugUnaries(U, x^g)$
6 $\bar{x} = Inference(A, U_L, H)$
7 $SubGrad \leftarrow F(\bar{x}, x^g)$
8 Update W
  end
end
9 **return** Model Weights W

Algorithm 2 LIFT Algorithm

Lifted Training(Training Set $\mathbf{I_T}$, Inf-Algo A, Lifting_Schedule $Lift\_Sched$)

1 Initialize W
2 **for** $i \in \{1, 2 \cdots num\_epochs\}$ **do**
3 **forall** $I \in I_T$ **do**
4 U $\leftarrow$ NN(W, I)
5 $U_L \leftarrow GetLossAugUnaries(U, x^g)$
6 $\mathbf{LU_L \leftarrow Lift\_Unaries(U_L, Lift\_Sched[i])}$
7 $\bar{lx} \leftarrow Inference(A, LU_L, H)$
8 $\mathbf{\bar{x} \leftarrow Ground\_Labels(\bar{lx}, Lift\_Sched[i])}$
9 $SubGrad \leftarrow F(\bar{x}, x^g)$
10 Update W
  end
end
11 **return** Model Weights W

4 Experiments on Stereo-Vision

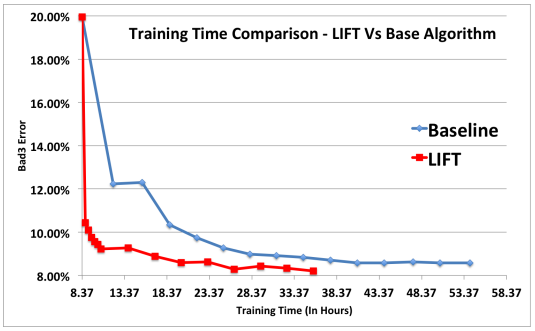

Figure 1: LIFT is anytime better than baseline.

We experimentally answer the question: does LIFT give any performance benefits compared to the baseline (Algorithm 1) for end-to-end learning in the task of stereo-vision. Here, the goal is to

Splitup of Total Training Time			
Algorithm	Type of Inference	Epochs	Time per Epoch (mins)
Baseline	Ground	10	196
LIFT	$K = 1$, $CP_{Iter} = 0$	6	22
	Ground	3	187

Table 1: Detailed train time and lift schedule

Model	bad3 (%)	Total Time (mins)
CNN-7	19.95	494
CNN-7 + CRF (Post-process)	12.34	494
CNN-7 + CRF (End-to-End)	8.58	2454
CNN-7 + CRF (LIFT)	**8.60**	**1187**

Table 2: Train time & accuracy comparison

estimate the depth of each pixel in an image given the left and the right images of a scene. We use the architecture proposed by Knöbelreiter *et al.* [2017], which achieves near state-of-the-art accuracy using a very simple neural model followed by a CRF trained in an end-to-end fashion. Knöbelreiter *et al.* [2017] use a highly specialized inference technique called Dual-MM, which exploits the grid structure of the graph, and, hence, can be parallelized on a GPU. Since the goal of our paper is to work with a more generic inference algorithm applicable across a variety of domains, we replace their Dual-MM with a generic off-the-shelf Alpha-Expansion Fusion algorithm (Ishikawa [2009], implemented in OpenGM library (Andres *et al.* [2012]).

Methodology and Dataset: Due to non-availability of training code by Knöbelreiter *et al.* [2017], we use our own implementation. We use a 7-layered CNN network for learning the unaries. Further, we use the Potts based pairwise potentials in the CRF. We set the parameter λ (see background), to 0.2. As described in Knöbelreiter *et al.* [2017], we learn the parameters using a 2-stage training. In the first phase, a neural network is trained to learn the unaries without the use of a CRF. In the second phase, these unaries are fine-tuned in an end-to-end fashion using a neural-CRF model as described above. Note that first phase of training is identical for LIFT as well the baseline and takes 8.37 hours. We use the benchmark Kitti2015 (Menze *et al.* [2018]) dataset for evaluation. The dataset is split into about 80% training and 20% validation set, obtaining 161 train and 39 validation images. We measure our performance using the standard error metric of *bad3* (for stereo-vision), which computes the percentage of pixels with the difference from true label > 3.

Results: Figure 1 compares the learning curves for *LIFT* and the baseline in the second stage of training where we plot *bad3* error on Y-axis with the training time on X-axis. We observe that LIFT reaches the same *bad3* error as achieved by the baseline algorithm but in much less time. Further, at any given time, LIFT significantly outperforms the baseline algorithm. To our surprise, running *LIFT* further in fact improved the error rate to 8.20% from 8.6% achieved by the baseline. Table 1 shows the detailed training time during different epochs of training. In this case, LIFT has used two lifting levels: (1) $(K = 1, CP_{Iter} = 0)$ for 6 training epochs, followed by (2) completely ground network for last 3 epochs. Observe that time per epoch for the ground network is slightly different for both algorithms - this is due to the fact that we run inference algorithm till the energy of the network plateaus, which may take different number of iterations depending upon unaries. As can be seen, training is extremely fast during the first (coarse) level of lifting where each epoch takes an average of 22 minutes compared to average of 196 per epoch for the baseline. Table 2 compares the total training time for various methods. We observe that adding CRF at the post-processing stage significantly reduces the error, which clearly illustrates the efficacy of using CRF. Further, there is a significant improvement with end-to-end training but it increases the training time by 6x. LIFT maintains the error rate of end-to-end training which is 8.6% and at the same time obtains 50% reduction in training time compared to the baseline.

5 Conclusion and Future Directions

In this work, we have provided a novel application of lifted inference for speeding-up end-to-end training of neural-CRF models. Our idea is based on the fact that during initial learning iterations, it suffices to work with an approximate gradient direction (computed using lifted inference), which can be refined as the learning proceeds. We have demonstrated the applicability of our approach on the task of stereo-vision. Our initial experiments have shown a lot of promise, which paves the way for some interesting future work directions. Some of these include whether we can come up with theoretical guarantees for our learning procedure, how well our results generalize to other applications and finally, whether we can exploit lifting even in the presence of highly specialized inference algorithms, which are customized to specific tasks.

ACKNOWLEDGEMENTS

We thank anonymous reviewers for their comments and suggestions. We also thank Microsoft Azure sponsorships, and the IIT Delhi HPC facility for computational resources. Ankit Anand is supported by the TCS Fellowship. Mausam is supported by grants from 1MG, Google and Bloomberg. Parag Singla is supported by the DARPA Explainable Artificial Intelligence (XAI) Program with number N66001-17-2-4032. Both Mausam and Parag Singla are supported by the Visvesvaraya Young Faculty Fellowships by Govt. of India and IBM SUR awards. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of the funding agencies.

References

Bjoern Andres, Thorsten Beier, and Joerg H. Kappes. OpenGM: A C++ library for discrete graphical models. *CoRR*, abs/1206.0111, 2012.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 40(4):834–848, 2018.

Haroun Habeeb, Ankit Anand, Mausam, and Parag Singla. Coarse-to-fine Lifted MAP Inference in Computer Vision. In *IJCAI*, 2017.

Hiroshi Ishikawa. Higher-order gradient descent by fusion-move graph cut. In *ICCV*, pages 568–574, 2009.

Y. Jernite, A. Rush, and D. Sontag. A Fast Variational Approach for Learning Markov Random Field Language Models. In *ICML*, 2015.

K. Kersting, B. Ahmadi, and S. Natarajan. Counting Belief Propagation. In *UAI*, 2009.

A. Kimmig, L. Mihalkova, and L. Getoor. Lifted Graphical Models: A Survey. *Machine Learning*, 2015.

Patrick Knöbelreiter, Christian Reinbacher, Alexander Shekhovtsov, and Thomas Pock. End-to-End Training of Hybrid CNN-CRF Models for Stereo. In *CVPR*, 2017.

Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *JPRS*, 2018.

A. Nath and P. Domingos. Learning Tractable Probabilistic Models for Fault Localization. In *AAAI*, 2016.

Alexander Shekhovtsov, Christian Reinbacher, Gottfried Graber, and Thomas Pock. Solving dense image matching in real-time using discrete-continuous optimization. *CoRR*, abs/1601.06274, 2016.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional Random Fields As Recurrent Neural Networks. In *ICCV*, pages 1529–1537, 2015.