

End-to-End Learning for Structured Prediction Energy Networks

David Belanger¹ Bishan Yang² Andrew McCallum¹

Abstract

Structured Prediction Energy Networks (SPENs) are a simple, yet expressive family of structured prediction models (Belanger & McCallum, 2016). An energy function over candidate structured outputs is given by a deep network, and predictions are formed by gradient-based optimization. This paper presents end-to-end learning for SPENs, where the energy function is discriminatively trained by back-propagating through gradient-based prediction. In our experience, the approach is substantially more accurate than the structured SVM method of Belanger & McCallum (2016), as it allows us to use more sophisticated non-convex energies. We provide a collection of techniques for improving the speed, accuracy, and memory requirements of end-to-end SPENs, and demonstrate the power of our method on 7-Scenes image denoising and CoNLL-2005 semantic role labeling tasks. In both, inexact minimization of non-convex SPEN energies is superior to baseline methods that use simplistic energy functions that can be minimized exactly.

1. Introduction

In a variety of application domains, given an input \mathbf{x} we seek to predict a structured output \mathbf{y} . For example, given a noisy image, we predict a clean version of it, or given a sentence we predict its semantic structure. Often, it is insufficient to employ a feed-forward predictor $\mathbf{y} = F(\mathbf{x})$, since this may have prohibitive sample complexity, fail to model global interactions among outputs, or fail to enforce hard output constraints. Instead, it can be advantageous to define the prediction function implicitly in terms of energy

minimization (LeCun et al., 2006):

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{y}} E_{\mathbf{x}}(\mathbf{y}), \quad (1)$$

where $E_{\mathbf{x}}(\cdot)$ depends on \mathbf{x} and learned parameters.

This approach includes factor graphs (Kschischang et al., 2001), e.g., conditional random fields (CRFs) (Lafferty et al., 2001), and many recurrent neural networks (Sec. 2.1). Output constraints can be enforced using constrained optimization. Compared to feed-forward approaches, energy-based approaches often provide better opportunities to inject prior knowledge about likely outputs and often have more parsimonious models. On the other hand, energy-based prediction requires non-trivial search in the exponentially-large space of outputs, and search techniques often need to be designed on a case-by-case basis.

Structured prediction energy networks (SPENs) (Belanger & McCallum, 2016) help reduce these concerns. They can capture high-arity interactions among components of \mathbf{y} that would lead to intractable factor graphs and provide a mechanism for automatic structure learning. This is accomplished by expressing the energy function in Eq. (1) as a deep architecture and forming predictions by approximately optimizing \mathbf{y} using gradient descent.

While providing the expressivity and generality of deep networks, SPENs also maintain the useful semantics of energy functions: domain experts can design architectures to capture known properties of the data, energy functions can be combined additively, and we can perform constrained optimization over \mathbf{y} . Most importantly, SPENs provide for black-box interaction with the energy, via forward and back-propagation. This allows practitioners to explore a wide variety of models without the need to hand-design corresponding prediction methods.

Belanger & McCallum (2016) train SPENs using a structured SVM (SSVM) loss (Taskar et al., 2004; Tsochantzidis et al., 2004) and achieve competitive performance on simple multi-label classification tasks. Unfortunately, we have found it difficult to extend their method to more complex domains. SSVMs are unreliable when exact energy minimization is intractable, as loss-augmented inference may fail to discover margin violations (Sec. 2.3).

In response, we present end-to-end training of SPENs,

¹University of Massachusetts, Amherst ²Carnegie Mellon University. Correspondence to: David Belanger <belanger@cs.umass.edu>.

where one directly back-propagates through a computation graph that unrolls gradient-based energy minimization. This does not assume that exact minimization is tractable, and instead directly optimizes the practical performance of a particular approximate minimization algorithm. End-to-end training for gradient-based prediction was introduced in Domke (2012) and applied to deep energy models by Brakel et al. (2013). See Sec. 3 for details.

When applying end-to-end training to SPENs for problems with sophisticated output structure, we have encountered a variety of technical challenges. The core contribution of this paper is a set of general-purpose solutions for overcoming these. Sec. 4.1 alleviates the effect of vanishing gradients when training SPENs defined over the convex relaxation of discrete prediction problems. Sec. 4.2 trains energies such that gradient-based minimization is fast. Sec. 4.3 reduces SPENs’ computation and memory overhead. Finally, Sec. 5 provides practical recommendations for specific architectures, parameter tying schemes, and pretraining methods that reduce overfitting and improve efficiency.

We demonstrate the effectiveness of our SPEN training methods on two diverse tasks. We first consider depth image denoising on the 7-Scenes dataset (Newcombe et al., 2011), where we employ deep convolutional networks as priors over images. This provides a significant performance improvement, from 36.3 to 40.4 PSNR, over the recent work of (Wang et al., 2016), which unrolls more sophisticated optimization than us, but uses a simpler image prior. After that, we apply SPENs to semantic role labeling (SRL) on the CoNLL-2005 dataset (Carreras & Màrquez, 2005). The task is challenging for SPENs because the output is discrete, sparse, and subject to rigid non-local constraints. We show how to formulate SRL as a SPEN problem and demonstrate performance improvements over strong baselines that use deep features, but sufficiently simple energy functions that the constraints can be enforced using dynamic programming.

Despite substantial differences between the two applications, learning and prediction for all models is performed using the same gradient-based prediction and end-to-end learning code. This black-box interaction with the model provides many opportunities for further use of SPENs.

2. Structured Prediction Energy Networks

A SPEN is defined as an instance of Eq. (1) where the energy is given by a deep neural network that provides a subroutine for efficiently evaluating $\frac{d}{dy} E_{\mathbf{x}}(\mathbf{y})$ (Belanger & McCallum, 2016). Differentiability necessitates that the energy is defined on continuous inputs. Going forward, \mathbf{y} will always be continuous. Prediction is performed by gradient-based optimization with respect to \mathbf{y} .

This section first motivates the SPENs employed in this paper, by contrasting them with alternative energy-based approaches to structured prediction. Then, we present two families of methods for training energy-based structured prediction models that have been explored in prior work.

2.1. Black-Box vs. Factorized Energy Functions

The definition of SPENs above is extremely general and includes many existing modeling techniques. However, both this paper and Belanger & McCallum (2016) depart from most prior work by employing monolithic energy functions that only provide forward and back-propagation.

This contrasts with the two principal families of energy-based models in the literature, where the tractability of (approximate) energy minimization depends crucially on the factorization structure of the energy. First, *factor graphs* decompose the energy into a sum of functions defined over small sets of subcomponents of \mathbf{y} (Kschischang et al., 2001). This structure provides opportunities for energy minimization using message passing, MCMC, or combinatorial solvers. Second, *autoregressive models*, such as recurrent neural networks (RNNs) assume an ordering on the components of \mathbf{y} such that the energy for component y_i only depends on its predecessors. Approximate energy minimization can be performed using search in the space of prefixes of \mathbf{y} using beam search or greedy search. See, for example, Sutskever et al. (2014).

By not relying on any such factorization when choosing learning and prediction algorithms for SPENs, we can consider much broader families of deep energy functions. We do not specify the interaction structure in advance, but instead learn it automatically by fitting a deep network. This can capture sophisticated global interactions among components of \mathbf{y} that are difficult to represent using a factorized energy. Of course, the downside of such SPENs is that they provide few guarantees, particularly when employing non-convex energies. Furthermore, for problems with hard constraints on outputs, the ability to do effective constrained optimization may have depended crucially on certain factorization structure.

2.2. Learning as Conditional Density Estimation

One method for estimating the parameters of an energy-based model $E_{\mathbf{x}}(\mathbf{y})$ is to maximize the conditional likelihood of \mathbf{y} :

$$\mathbb{P}(\mathbf{y}|\mathbf{x}) \propto \exp(-E_{\mathbf{x}}(\mathbf{y})). \quad (2)$$

Unfortunately, computing the likelihood requires the distribution’s normalizing constant, which is intractable for black-box energies with no available factorization structure. In *contrastive backprop*, this is circumvented by performing contrastive divergence training, with Hamiltonian

Monte Carlo sampling from the energy surface (Mnih & Hinton, 2005; Hinton et al., 2006; Ngiam et al., 2011). Recently, Zhai et al. (2016) trained energy-based density models for anomaly detection by exploiting the connections between denoising autoencoders, energy-based models, and score matching (Vincent, 2011).

2.3. Learning with Exact Energy Minimization

Let $\Delta(\hat{\mathbf{y}}, \mathbf{y}^*)$ be a non-negative task-specific cost function for comparing $\hat{\mathbf{y}}$ and the ground truth \mathbf{y}^* . Belanger & McCallum (2016) employ a structured SVM (SSVM) loss (Taskar et al., 2004; Tschantz et al., 2004):

$$\sum_{\{\mathbf{x}_i, \mathbf{y}_i\}} \max_{\mathbf{y}} [\Delta(\mathbf{y}, \mathbf{y}_i) - E_{\mathbf{x}_i}(\mathbf{y}) + E_{\mathbf{x}_i}(\mathbf{y}_i)]_+, \quad (3)$$

where $[\cdot]_+ = \max(0, \cdot)$. Each step of minimizing Eq. (3) by subgradient descent requires *loss-augmented inference*:

$$\min_{\mathbf{y}} (-\Delta(\mathbf{y}, \mathbf{y}_i) + E_{\mathbf{x}_i}(\mathbf{y})). \quad (4)$$

For differentiable $\Delta(\mathbf{y}, \mathbf{y}_i)$, a local optimum of Eq. (4) can be obtained using first-order methods.

Solving Eq. (4) probes the model for margin violations. If none exist, the gradient of the loss with respect to the parameters is zero. Therefore, SSVM performance does not degrade gracefully with optimization errors in the inner prediction problem, since inexact energy minimization may fail to discover margin violations that exist. Performance can be recovered if Eq. (4) returns a lower bound, eg. by solving an LP relaxation (Finley & Joachims, 2008). However, this is not possible in general. In Sec. 6.1.3 we compare the image denoising performance of SSVM learning vs. this paper’s end-to-end method. Overall, we have found SSVM learning to be unstable and difficult to tune for non-convex energies in applications more complex than the multi-label classification experiments of Belanger & McCallum (2016).

The *implicit function theorem* offers an alternative framework for training energy-based predictors (Foo et al., 2008; Samuel & Tappen, 2009). See Domke (2012) for an overview. While a naive implementation requires inverting Hessians, one can solve the product of an inverse Hessian and a vector using conjugate gradients, which can leverage the techniques discussed in Sec. 3 as a subroutine. To perform reliably, the method unfortunately requires exact energy minimization and many conjugate gradient iterations.

Overall, both of these learning algorithms only update the energy function in the neighborhoods of the ground truth and the predictions of the current model. On the other hand, it may be advantageous to shape the entire energy surface such that it exhibits certain properties, e.g., gradient descent converges quickly when initialized well (Sec. 4.2).

Therefore, these methods may be undesirable even for problems where exact energy minimization is tractable.

For non-convex $E_{\mathbf{x}}(\mathbf{y})$, gradient-based prediction will only find a local optimum. Amos et al. (2017) present *input-convex neural networks* (ICNNs), which employ an easy-to-implement method for constraining the parameters of a SPEN such that the energy is convex with respect to \mathbf{y} , but perhaps non-convex with respect to the parameters. One simply uses convex, non-decreasing non-linearities and only non-negative parameters in any part of the computation graph downstream from \mathbf{y} . Here, prediction will return the global optimum, but convexity, especially when achieved this way, may impose a strong restriction on the expressivity of the energy. Their construction is a sufficient condition for achieving convexity, but there are convex energies that disobey this property. Our experiments present results for instances of ICNNs. In general, non-convex SPENS perform better.

3. Learning with Unrolled Optimization

The methods of Sec. 2.3 are unreliable with non-convex energies because we cannot simply use the output of inexact energy minimization as a drop-in replacement for the exact minimizer. Instead, a collection of prior work has performed end-to-end learning of gradient-based predictors (Gregor & LeCun, 2010; Domke, 2012; Maclaurin et al., 2015; Andrychowicz et al., 2016; Wang et al., 2016; Metz et al., 2017; Greff et al., 2017). Rather than reasoning about the energy minimum as an abstract quantity, the authors pose a specific gradient-based algorithm for approximate energy minimization and optimize its empirical performance using back-propagation. This is a form of *direct risk minimization* (Tappen et al., 2007; Stoyanov et al., 2011; Domke, 2013).

Consider simple gradient descent:

$$\mathbf{y}_T = \mathbf{y}_0 - \sum_{t=1}^T \eta_t \frac{d}{d\mathbf{y}} E_{\mathbf{x}}(\mathbf{y}_t). \quad (5)$$

To learn the energy function end-to-end, we can back-propagate through the unrolled optimization Eq. (5) for fixed T . With this, it can be rendered API-equivalent to a feed-forward network that takes \mathbf{x} as input and returns a prediction for \mathbf{y} , and can thus be trained using standard methods. Furthermore, certain hyperparameters, such as the learning rates η_t , are trainable (Domke, 2012).

This backpropagation requires non-standard interaction with a neural-network library because Eq. (5) computes gradients in the forward pass, and thus it must compute second order terms in the backwards pass. We can save space and computation by avoiding instantiating Hessian terms and instead directly computing Hessian-vector prod-

ucts. These can be achieved three ways. First, the method of Pearlmutter (1994) is exact, but requires non-trivial code modifications. Second, some libraries construct computation graphs for gradients that are themselves differentiable. Third, we can employ finite-differences (Domke, 2012).

It is clear that Eq. (5) can be naturally extended to certain alternative optimization methods, such as gradient descent with momentum, or L-BFGS (Liu & Nocedal, 1989; Domke, 2012). These require an additional state vector \mathbf{h}_t that is evolved along with \mathbf{y}_t across iterations. Andrychowicz et al. (2016) unroll gradient-descent, but employ a learned non-linear RNN to perform per-coordinate updates to \mathbf{y} . End-to-end learning is also applicable to special-case energy minimization algorithms for graphical models, such as mean-field inference and belief propagation (Domke, 2013; Chen et al., 2015; Tompson et al., 2014; Li & Zemel, 2014; Hershey et al., 2014; Zheng et al., 2015).

4. End-to-End Learning for SPENs

We now present details for applying the methods of the previous section to SPENs. We first describe considerations for learning SPENs defined for the convex relaxation of discrete labeling problems. Then, we describe how to encourage our models to optimize quickly in practice. Finally, we present methods for improving the speed and memory overhead of SPEN implementations.

Our experiments unroll either Eq. (5) or an analogous version implementing gradient descent with momentum. We compute Hessian-vector products using the finite-difference method of (Domke, 2012), which allows black-box interaction with the energy.

We avoid the RNN-based approach of Andrychowicz et al. (2016) because it diminishes the semantics of the energy, as the interaction between the optimizer and gradients of the energy is complicated. In recent work, Gygli et al. (2017) propose an alternative learning method that fits the energy function such that $E_{\mathbf{x}}(\cdot) \approx -\Delta(\cdot, \mathbf{y}^*)$, where Δ is defined as in Sec. 2.3. This is an interesting direction for future research, as it allows for non-differentiable Δ . The advantage of end-to-end learning, however, is that it provides a energy function that is precisely tuned for a particular test-time energy minimization procedure.

4.1. End-to-End Learning for Discrete Problems

To apply SPENs to a discrete structured prediction problem, we relax to a constrained continuous problem, apply SPEN prediction, and then round to a discrete output. For example, for tagging each pixel of a $h \times w$ image with a binary label, we would relax from $\{0, 1\}^{w \times h}$ to $[0, 1]^{w \times h}$, and if the pixels can take on one of D values, we would relax from $\mathbf{y} \in \{0, \dots, D\}^{w \times h}$ to $\Delta_D^{w \times h}$, where Δ_D is the

probability simplex on D elements.

While this rounding introduces poorly-understood sources of error, it has worked well for non-convex energy-based prediction in multi-label classification (Belanger & McCallum, 2016), sequence tagging (Vilnis et al., 2015), and translation (Hoang et al., 2017).

Both $[0, 1]^{w \times h}$ and $\Delta_D^{w \times h}$ are Cartesian products of probability simplices, and it is easy to adopt existing methods for projected gradient optimization over the simplex.

First, it is natural to apply Euclidean projected gradient descent. Over $[0, 1]^{w \times h}$, we have:

$$\mathbf{y}_{t+1} = \text{Clip}_{0,1} [\mathbf{y}_t - \eta_t \nabla E_{\mathbf{x}}(\mathbf{y}_t)], \quad (6)$$

This is unusable for end-to-end learning, however, since back-propagation through the projection will yield 0 gradients whenever $\mathbf{y}_t - \eta_t \nabla E_{\mathbf{x}}(\mathbf{y}_t) \notin [0, 1]$. This is similarly problematic for projection onto $\Delta_D^{w \times h}$ (Duchi et al., 2008).

Alternatively, we can apply entropic mirror descent, ie. projected gradient with distance measured by KL divergence (Beck & Teboulle, 2003). For $\mathbf{y} \in \Delta_D^{w \times h}$, we have:

$$\mathbf{y}_{t+1} = \text{SoftMax}(\log(\mathbf{y}_t) - \eta_t \nabla E_{\mathbf{x}}(\mathbf{y}_t)) \quad (7)$$

This is suitable for end-to-end learning, but the updates are similar to an RNN with sigmoid non-linearities, which is vulnerable to vanishing gradients (Bengio et al., 1994).

Instead, we have found it useful to avoid constrained optimization entirely, by optimizing un-normalized logits \mathbf{l}_t , with $\mathbf{y}_t = \text{SoftMax}(\mathbf{l}_t)$:

$$\mathbf{l}_{t+1} = \mathbf{l}_t - \eta_t \nabla E_{\mathbf{x}}(\text{SoftMax}(\mathbf{l}_t)). \quad (8)$$

Here, the updates to \mathbf{l}_t are additive, and thus will be less susceptible to vanishing gradients (Hochreiter & Schmidhuber, 1997; Srivastava et al., 2015; He et al., 2016).

Finally, Amos et al. (2017) present the *bundle entropy method* for convex optimization with simplex constraints, along with a method for differentiating the output of the optimizer. End-to-end learning for Eq. (10) can be performed using generic learning software, since the unrolled optimization obeys the API of a feed-forward predictor, but unfortunately this is not true for their method. Future work should consider their method, however, as it performs very rapid energy minimization.

4.2. Learning to Optimize Quickly

We next enumerate methods for learning a model such that gradient-based energy minimization converges to high-quality \mathbf{y} quickly. When using such methods, we have found it important to maintain the same optimization configuration, such as T , at both train and test time.

First, we can encourage rapid optimization by defining our loss function as a sum of losses on every iterate \mathbf{y}_t , rather than only on the final one. Let $\ell(\mathbf{y}_t, \mathbf{y}^*)$ be a differentiable loss between an iterate and the ground truth. We employ

$$L = \frac{1}{T} \sum_{t=1}^T w_t \ell(\mathbf{y}_t, \mathbf{y}^*), \quad (9)$$

where w_t is a non-negative weight. This encourages the model to achieve high-quality predictions early. It has the additional benefit that it reduces vanishing gradients, since a learning signal is introduced at every timestep. Our experiments use $w_t = \frac{1}{T-t+1}$.

Second, for the simplex-constrained problems of Sec. 4.1, we smooth the energy with an entropy term $\sum_i H(\mathbf{y}_i)$. This introduces extra strong convexity, which helps improve convergence. It also strengthens the parallel between SPEN prediction and marginal inference in a Markov random field, where the inference objective is expected energy plus entropy (Koller & Friedman, 2009, p. 385).

Third, we can set T to a small value. Of course, this guarantees that optimization converges quickly on the train data. Here, we lose the contract that Eq. (10) is even performing energy minimization, since it hasn't converged, but this may be acceptable if predictions are accurate. For example, some experiments achieve good performance with $T = 3$.

In future work, it may be fruitful to directly penalize convergence criteria, such as $\|\mathbf{y}_t - \mathbf{y}_{t-1}\|$ and $\|\frac{d}{d\mathbf{y}_t} E_{\mathbf{x}}(\mathbf{y}_t)\|$.

4.3. Efficient Implementation

Since we can explicitly encourage our model to converge quickly, it is important to exploit fast convergence at train time. Eq. (10) is unrolled for a fixed T . However, if optimization converges at $T_0 < T$, it suffices to start back-propagation at T_0 , since the updates to \mathbf{y}_t for $t > T_0$ are the identity. Therefore, we unroll for a fixed number of iterations T , but iterate only until convergence is detected.

To support back-propagation, a naive implementation of Eq. (10) would require T clones of the energy (with tied parameters). We reduce memory overhead by checkpointing the inputs and outputs of the energy, but discarding its internal state. This allows us to use a single copy of the energy, but requires recomputing forward evaluations at specific \mathbf{y}_t during the backwards pass. To save additional memory, we could have reconstructed the \mathbf{y}_t on-the-fly either by reversing the dynamics of the energy minimization method (Domke, 2013; Maclaurin et al., 2015) or by performing a small amount of extra forward-propagation (Geoffrey & Padmanabhan, 2000; Lewis, 2003).

5. Recommended SPEN Architectures for End-to-End Learning

To train SPENs end-to-end, we write Eq. (5) as:

$$\mathbf{y}_T = \text{Init}(F(\mathbf{x})) - \sum_{t=1}^T \eta_t \frac{d}{d\mathbf{y}} E(\mathbf{y}_t; F(\mathbf{x})). \quad (10)$$

Here, $\text{Init}(\cdot)$ is a differentiable procedure for predicting an initial iterate \mathbf{y}_0 . Following Belanger & McCallum (2016), we also employ $E_{\mathbf{x}}(\mathbf{y}) = E(\mathbf{y}; F(\mathbf{x}))$, where the dependence of $E_{\mathbf{x}}(\mathbf{y})$ on \mathbf{x} comes by way of a parametrized feature function $F(\mathbf{x})$. This is useful because test-time prediction can avoid back-propagation in $F(\mathbf{x})$.

We have found it useful in practice to employ an energy that splits into global and local terms:

$$E(\mathbf{y}; F(\mathbf{x})) = E^g(\mathbf{y}; F(\mathbf{x})) + \sum_i E_i^l(\mathbf{y}_i; F(\mathbf{x})). \quad (11)$$

Here, i indexes the components of \mathbf{y} and $E^g(\mathbf{y}; F(\mathbf{x}))$ is an arbitrary global energy function. The modeling benefits of the local terms are similar to the benefits of using local factors in popular factor graph models. We also can use the local terms to provide an implementation of $\text{Init}(\cdot)$.

We pretrain $F(\mathbf{x})$ by training the feed-forward predictor $\text{Init}(F(\mathbf{x}))$. We also stabilize learning by first clamping the local terms for a few epochs while updating $E^g(\mathbf{y}; F(\mathbf{x}))$.

To back-propagate through Eq. (10), the energy function must be at least twice differentiable with respect to \mathbf{y} . Therefore, we can't use non-linearities with discontinuous gradients. Instead of ReLUs, we use a SoftPlus with a reasonably high temperature. Note that $F(\mathbf{x})$ and $\text{Init}(\cdot)$ can be arbitrary networks that are sub-differentiable with respect to their parameters.

6. Experiments

We evaluate SPENs on image denoising and semantic role labeling (SRL) tasks. Image denoising is an important benchmark for SPENs, since the task appears in many prior works employing end-to-end learning. SRL is useful for evaluating SPENs' suitability for challenging combinatorial problems, since the outputs are subject to rigid, non-local constraints. For both, we provide controlled experiments that isolate the impact of various SPEN design decisions, such as the optimization method that is unrolled and the expressivity of the energy function.

In these applications, we employ specific architectures based on our prior knowledge about the problem domain. This capability is crucial for introducing the necessary inductive bias to be able to fit SPENs on limited datasets. Overall, black-box prediction and learning methods for

SPENs are useful because we can select architectures based on their suitability for the data, not whether they support model-specific algorithms.

6.1. Image Denoising

Let $\mathbf{x} \in [0, 1]^{w \times h}$ be an observed grayscale image. We assume that it is a noisy realization of a latent clean image $\mathbf{y} \in [0, 1]^{w \times h}$, which we estimate using MAP inference. Consider a Gaussian noise model with variance σ^2 and a prior $\mathbb{P}(\mathbf{y})$. The associated energy function is:

$$\|\mathbf{y} - \mathbf{x}\|_2^2 - 2\sigma^2 \log \mathbb{P}(\mathbf{y}). \quad (12)$$

Here, the feature network is the identity. The first term is the local energy network and the second, which does not depend on \mathbf{x} , is the global energy network.

There are three general families for the prior. First, it can be hard-coded. Second, it can be learned by approximate density estimation. Third, given a collection of $\{\mathbf{x}, \mathbf{y}\}$ pairs, we can perform supervised learning, where the prior's parameters are discriminatively trained such that the output of a particular algorithm for minimizing Eq. (12) is high-quality. End-to-end learning has proven to be highly successful for the third approach (Tappen et al., 2007; Barbu, 2009; Schmidt et al., 2010; Sun & Tappen, 2011; Domke, 2012; Wang et al., 2016), and thus it is important to evaluate the methods of this paper on the task.

6.1.1. IMAGE PRIORS

Much of the existing work on end-to-end training for denoising considers some form of a field-of-experts (FOE) prior (Roth & Black, 2005). We consider an ℓ_1 version, which assigns high probability to images with sparse activations from K learned filters:

$$\mathbb{P}(\mathbf{y}) \propto \exp \left(- \sum_k \|(\mathbf{f}_k * \mathbf{y})\|_1 \right). \quad (13)$$

Wang et al. (2016) perform end-to-end learning for Eq. (13), by unrolling proximal gradient methods that analytically handle the non-differentiable ℓ_1 term.

This paper assumes we only have black-box interaction with the energy. In response, we alter Eq. (13) such that it is twice differentiable, so that we can unroll generic first-order optimization methods. We approximate Eq. (13) by leveraging a SoftPlus with temperature 25, replacing $|\cdot|$ by:

$$\text{SoftAbs}(\mathbf{y}) = 0.5 \text{SoftPlus}(\mathbf{y}) + 0.5 \text{SoftPlus}(-\mathbf{y}). \quad (14)$$

The principal advantage of learning algorithms that are not hand-crafted to the problem structure is that they provide the opportunity to employ more expressive energies. In response, we also consider a deeper prior, given by:

$$\mathbb{P}(\mathbf{y}) \propto \exp(-\text{DNN}(\mathbf{y})). \quad (15)$$

Here, $\text{DNN}(\mathbf{y})$ is a general deep convolutional network that takes an image and returns a number. The architecture in our experiments consists of a $7 \times 7 \times 32$ convolution, a SoftPlus, another $7 \times 7 \times 32$ convolution, a SoftPlus, a $1 \times 1 \times 1$ convolution, and finally spatial average pooling. The method of Wang et al. (2016) cannot handle this prior.

6.1.2. EXPERIMENTAL SETUP

We evaluate on the 7-Scenes dataset (Newcombe et al., 2011), where we seek to denoise depth measurements from a Kinect sensor. Our data processing and hyperparameters are designed to replicate the setup of Wang et al. (2016), who demonstrate state-of-the-art results for energy-minimization-based denoising on the dataset. We train using random 96×128 crops from 200 images of the same scene and report PSNR (higher is better) for 5500 images from different scenes. We treat σ^2 as a trainable parameter and minimize the mean-squared-error of \mathbf{y} .

6.1.3. RESULTS AND DISCUSSION

Example outputs are given in Figure 1 and Table 1 compares PSNR. **BM3D** is a widely-used non-parametric method (Dabov et al., 2007). FilterForest (**FF**) adaptively selects denoising filters for each location (Fanello et al., 2014). ProximalNet (**PN**) is the system of Wang et al. (2016). **FOE-20** is an attempt to replicate **PN** using end-to-end SPEN learning. We unroll 20 steps of gradient descent with momentum 0.75 and use the modification in Eq. (14). Note it performs similarly to **PN**, which unrolls 5 iterations of sophisticated optimization. Note that we can obtain 37.0 PSNR using a feed-forward convnet with a similar architecture to our DeepPrior, but without spatial pooling.

The next set of results consider improved instances of the FOE model. First, **FOE-20+** is identical to **FOE-20**, except that it employs the average loss Eq. (9), uses a momentum constant of 0.25, and treats the learning rates η_t as trainable parameters. We find that this results in both better performance and faster convergence. Of course, we could achieve fast convergence by simply setting T to be small. In response, we consider **FOE-3**. This only unrolls for $T = 3$ iterations and obtains superior performance.

The final three results are with the DNN prior Eq. (15). **DP-20** unrolls 20 steps of gradient descent with a momentum constant of 0.25. The gain in performance is substantial, especially considering that a PSNR of 30 can be obtained with elementary signal processing. Similar to **FOE-3** vs. **FOE-20+**, we experience a modest performance gain using **DP-3**, which only unrolls for 3 gradient steps but is otherwise identical.

Finally, the **FOE-SSVM** and **DP-SSVM** configurations use SSVM training. We find that **FOE-SSVM** performs

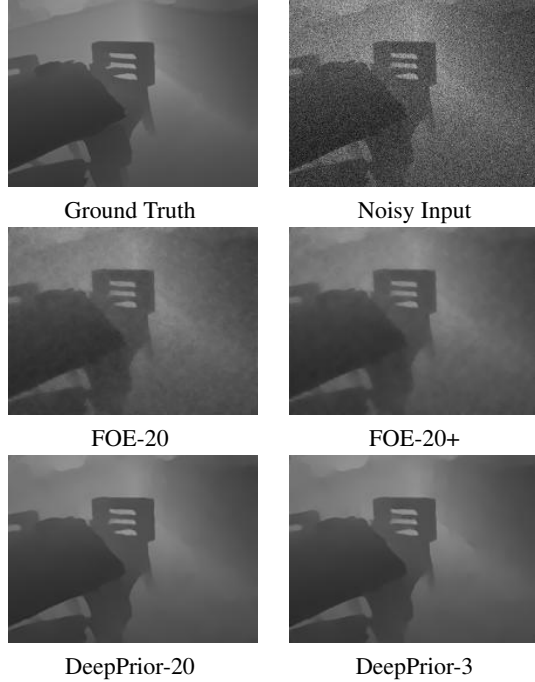


Figure 1. Example Denoising Outputs

BM3D	FF	PN	FOE-20	FOE-SSVM
35.46	35.63	36.31	36.41	37.7
FOE-20+	FOE-3	DP-20	DP-3	DP-SSVM
37.34	37.62	40.3	40.4	38.7

Table 1. Denoising Results (PSNR)

competitively with the other FOE configurations. This is not surprising, since the FOE prior is convex. However, fitting the DeepPrior with an SSVM is inferior to using end-to-end learning. The performance is very sensitive to the energy minimization hyperparameters.

In these experiments, it is superior to only unroll for a few iterations for end-to-end learning. One possible reason is that a shallow unrolled architecture is easier to train. Truncated optimization with respect to y may also provide an interesting prior over outputs (Duvenaud et al., 2016). It is also observed in Wang et al. (2014) that better energy minimization for FOE models may not improve PSNR. Often unrolling for 20 iterations results in over-smoothed outputs.

We are unable to achieve reasonable performance with an ICNN (Amos et al., 2017), which restricts all of the parameters of the convolutions to be positive. Unfortunately, this hinders the ability of the filters in the prior to act as edge detectors or encourage local smoothness. Both of these are important for high-quality denoising. Note that the ℓ_1 FOE is convex, even without the restrictive ICNN constraint.

6.2. Semantic Role Labeling

Semantic role labeling (SRL) predicts the semantic structure of predicates and arguments in sentences (Gildea & Jurafsky, 2002). For example, in the sentence “I want to buy a car,” the verbs “want” and “buy” are two predicates, and “I” is an argument that refers to the wanter and buyer, “to buy a car” is the thing wanted, and “a car” is the thing bought. Given predicates, we seek to identify arguments and their semantic roles in relation to each predicate. Formally, given a set of predicates p in a sentence x and a set of candidate argument spans a , we assign a discrete semantic role r to each pair of predicate and argument, where r can be either a pre-defined role label or an empty label. We evaluate SRL instead of, for example, noun-phrase chunking (Lacoste-Julien et al., 2012), since it is a more challenging task, where the outputs are subject to substantially more complex non-local constraints.

Existing work imposes hard constraints on r , such as excluding overlapping arguments and repeated core roles during prediction. The objective is to minimize the energy:

$$\min_r E(r; x, p, a) \text{ s.t. } r \in \mathcal{Q}(x, p, a), \quad (16)$$

where $\mathcal{Q}(x, p, a)$ is set of feasible joint role assignments. This constrained optimization problem can be solved using integer linear programming (ILP) (Punyakanok et al., 2008) or its relaxations (Das et al., 2012). These methods rely on the output of local classifiers that are unaware of structural constraints during training. More recently, Täckström et al. (2015) account for the constraint structure using dynamic programming at train time. FitzGerald et al. (2015) extend this using neural network features and show improved results.

6.2.1. DATA AND PREPROCESSING AND BASELINES

We consider the CoNLL 2005 shared task data (Carreras & Màrquez, 2005), with standard data splits and official evaluation scripts. We apply similar preprocessing as Täckström et al. (2015). This includes part-of-speech tagging, dependency parsing, and using the parse to generate candidate arguments.

Our baseline is an arc-factored model for the conditional probability of the predicate-argument arc labels:

$$\mathbb{P}(r|x, p, a) = \prod_i \mathbb{P}(r_i|x, p, a). \quad (17)$$

where $\mathbb{P}(r_i|x, p, a) \propto \exp(g(r_i, x, p, a))$. Here, each conditional distribution is given by a multiclass logistic regression model. See Appendix A.2.1 for details of the architecture and training procedure for our baseline.

When using the negative log of Eq. (18) as an energy in Eq. (16), there are variety of methods for finding a near-optimal $r \in \mathcal{Q}(x, p, a)$. First, we can employ simple

heuristics for locally resolving constraint violation. The **Local + H** system uses Eq. (18) and these. We can instead use the AD^3 message passing algorithm (Martins et al., 2011) to solve the LP relaxation of this constrained problem. We use **Local + AD^3** to refer to this system. Since the LP solution may not be integral, we post-process the AD^3 output using the same heuristics as **Local + H**.

6.2.2. SPEN MODEL

The SPEN performs continuous optimization over the relaxed set $\mathbf{y}_i \in \Delta_A$ for each discrete label \mathbf{r}_i , where A is the number of possible roles. The preprocessing generates sparse predicate-argument candidates, but we optimize over the complete bipartite graph between predicates and arguments to support vectorization. We have $\mathbf{y} \in \Delta_A^{n \times m}$, where n and m are the max number of predicates and arguments. Invalid arcs are constrained to the empty label.

We employ a pretrained version of Eq. (18) to provide the local energy term of a SPEN. This is augmented with global terms that couple the outputs together. See Appendix A.2.2 for details of the architecture we use. It has terms, for example, that apply a deep network to the feature representations of all of the arcs selected for a given predicate.

As with Täckström et al. (2015), we seek to account for constraints $Q(\mathbf{x}, \mathbf{p}, \mathbf{a})$ during both inference and learning, rather than only imposing them via post-processing. Therefore, we include additional energy terms that encode membership in $Q(\mathbf{x}, \mathbf{p}, \mathbf{a})$ as twice-differentiable soft constraints that can be applied to \mathbf{y} . All of the constraints in $Q(\mathbf{x}, \mathbf{p}, \mathbf{a})$ express that certain arcs cannot co-occur. For example, two arguments cannot attach to the same predicate if the arguments correspond to spans of tokens that overlap. Consider general binary variables a and b with corresponding relaxations $\bar{a}, \bar{b} \in [0, 1]$. We convert the constraint $\neg(a \wedge b)$ into an energy function $\alpha \text{SoftPlus}(\bar{a} + \bar{b} - 1)$, where α is a learned parameter.

We consider the **SPEN + H** and **SPEN + AD^3** configurations, which employ heuristics or AD^3 to enforce the output constraints. Rather than applying these methods to the probabilities from Eq. (18), we use the soft prediction output by energy minimization.

6.2.3. RESULTS AND DISCUSSION

Table 2 contains results on the CoNLL 2005 WSJ dev and test sets and the Brown test set. We compare the **SPEN** and **Local** systems with the best non-ensemble systems of Täckström et al. (2015) and FitzGerald et al. (2015), which have similar overall setups as us for feature extraction and for the parametrization of the local energy terms. For these, ‘Local’ fits Eq. (18) without regard for the output constraints, whereas ‘Structured’ explicitly considers

Model	Dev (WSJ)	Test (WSJ)	Test (Brown)
Local + H	78.0	79.7	69.7
Local + AD^3	78.2	80.0	69.9
SPEN + H	79.0	80.7	69.3
SPEN + AD^3	79.0	80.7	69.4
Täckström (Local)	77.9	79.3	70.2
Täckström (Structured)	78.6	79.9	71.3
FitzGerald (Local)	78.4	79.4	70.9
FitzGerald (Structured)	78.3	79.4	71.2

Table 2. SRL Results (F1)

them during training. Note that Zhou & Xu (2015) obtain slightly better performance with alternative RNN methods. We were unable to outperform the **Local** systems using a **SPEN** system trained with an SSVM loss.

We select our SPEN configuration by maximizing performance of **SPEN + AD^3** on the dev data. Our best system unrolls for 10 iterations, trains per-iteration learning rates, uses no momentum, and unrolls Eq. (8). Overall, **SPEN + AD^3** performs the best of all systems on the WSJ test data. We expect our diminished performance on the Brown test set is due to overfitting. The Brown set is not from the same source as the train, dev, and test WSJ data. SPENs are more susceptible to overfitting because the expressive global term introduces many parameters.

Note that **SPEN + AD^3** and **SPEN + H** performs identically, whereas **LOCAL + AD^3** and **LOCAL + H** do not. This is because our learned global energy encourages constraint satisfaction during gradient-based optimization of \mathbf{y} . Using the method of Amos et al. (2017) for restricting the energy to be convex wrt \mathbf{y} , we obtain 80.3 on the test set.

7. Conclusion and Future Work

SPENs are a flexible, expressive framework for structured prediction, but training them can be challenging. This paper provides a new end-to-end training method that enables high performance on considerably more complex tasks than those of Belanger & McCallum (2016). We unroll an approximate energy minimization algorithm into a differentiable computation graph that is trainable by gradient descent. The approach is user-friendly in practice because it returns not just an energy function but also a test-time prediction procedure that has been tailored for it.

In the future, it may be useful to employ more sophisticated unrolled optimizers, perhaps where the optimizer’s hyperparameters are a learned function of \mathbf{x} , and to perform iterative optimization in a learned feature space, rather than output space. Finally, we could model gradient-based prediction as a sequential decision making problem and train the energy using value-based reinforcement learning.

Acknowledgments

Many thanks to Justin Domke, Tim Vieira, Luke Vilnis, and Shenlong Wang for helpful discussions. The first and third authors were supported in part by the Center for Intelligent Information Retrieval and in part by DARPA under agreement number FA8750-13-2-0020. The second author was supported in part by DARPA under contract number FA8750-13-2-0005. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

- Amos, Brandon, Xu, Lei, and Kolter, J Zico. Input-convex deep networks. *ICML*, 2017.
- Andrychowicz, Marcin, Denil, Misha, Gomez, Sergio, Hoffman, Matthew W, Pfau, David, Schaul, Tom, and de Freitas, Nando. Learning to learn by gradient descent by gradient descent. *NIPS*, 2016.
- Barbu, Adrian. Training an active random field for real-time image denoising. *IEEE Transactions on Image Processing*, 18(11):2451–2462, 2009.
- Beck, Amir and Teboulle, Marc. Mirror descent and non-linear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3), 2003.
- Belanger, David and McCallum, Andrew. Structured prediction energy networks. In *ICML*, 2016.
- Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994.
- Brakel, Philémon, Stroobandt, Dirk, and Schrauwen, Benjamin. Training energy-based models for time-series imputation. *JMLR*, 14, 2013.
- Carreras, Xavier and Màrquez, Lluís. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL*, 2005.
- Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, and Yuille, Alan L. Semantic image segmentation with deep convolutional nets and fully connected crfs. *ICLR*, 2015.
- Dabov, Kostadin, Foi, Alessandro, Katkovnik, Vladimir, and Egiazarian, Karen. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.
- Das, Dipanjan, Martins, André FT, and Smith, Noah A. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Conference on Lexical and Computational Semantics*, 2012.
- Domke, Justin. Generic methods for optimization-based modeling. In *AISTATS*, 2012.
- Domke, Justin. Learning graphical model parameters with approximate marginal inference. *Pattern Analysis and Machine Intelligence*, 2013.
- Duchi, John, Shalev-Shwartz, Shai, Singer, Yoram, and Chandra, Tushar. Efficient projections onto the l_1 -ball for learning in high dimensions. In *ICML*, 2008.
- Duvenaud, David, Maclaurin, Dougal, and Adams, Ryan P. Early stopping as nonparametric variational inference. In *AISTATS*, 2016.
- Fanello, Sean Ryan, Keskin, Cem, Kohli, Pushmeet, Izadi, Shahram, Shotton, Jamie, Criminisi, Antonio, Pattacini, Ugo, and Paek, Tim. Filter forests for learning data-dependent convolutional kernels. In *CVPR*, 2014.
- Finley, Thomas and Joachims, Thorsten. Training structural svms when exact inference is intractable. In *ICML*, 2008.
- FitzGerald, Nicholas, Täckström, Oscar, Ganchev, Kuzman, and Das, Dipanjan. Semantic role labeling with neural network factors. In *EMNLP*, pp. 960–970, 2015.
- Foo, Chuan-sheng, Do, Chuong B, and Ng, Andrew Y. Efficient multiple hyperparameter learning for log-linear models. In *NIPS*, 2008.
- Geoffrey, Zweig and Padmanabhan, Mukund. Exact alpha-beta computation in logarithmic space with application to map word graph construction. 2000.
- Gildea, Daniel and Jurafsky, Daniel. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.
- Greff, Klaus, Srivastava, Rupesh K, and Schmidhuber, Jürgen. Highway and residual networks learn unrolled iterative estimation. *ICLR*, 2017.
- Gregor, Karol and LeCun, Yann. Learning fast approximations of sparse coding. In *ICML*, 2010.
- Gygli, M., Norouzi, M., and Angelova, A. Deep Value Networks Learn to Evaluate and Iteratively Refine Structured Outputs. In *ICML*, 2017.
- He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, 2016.

- Hershey, John R, Roux, Jonathan Le, and Weninger, Felix. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*, 2014.
- Hinton, Geoffrey, Osindero, Simon, Welling, Max, and Teh, Yee-Whye. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, 30(4):725–731, 2006.
- Hoang, Cong Duy Vu, Haffari, Gholamreza, and Cohn, Trevor. Decoding as continuous optimization in neural machine translation. *arXiv preprint:1701.02854*, 2017.
- Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 1997.
- Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *ICLR*, 2015.
- Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- Kschischang, Frank R, Frey, Brendan J, and Loeliger, H-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.
- Lacoste-Julien, Simon, Jaggi, Martin, Schmidt, Mark, and Pletscher, Patrick. Block-coordinate frank-wolfe optimization for structural svms. *arXiv preprint arXiv:1207.4747*, 2012.
- Lafferty, John, McCallum, Andrew, and Pereira, Fernando. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.
- LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, M, and Huang, F. A tutorial on energy-based learning. *Predicting Structured Data*, 1, 2006.
- Lewis, Bil. Debugging backwards in time. *arXiv preprint cs/0310016*, 2003.
- Li, Yujia and Zemel, Richard S. Mean-field networks. *ICML Workshop on Learning Tractable Probabilistic Models*, 2014.
- Liu, Dong C and Nocedal, Jorge. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.
- Maclaurin, Dougal, Duvenaud, David, and Adams, Ryan P. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.
- Martins, André FT, Figueiredo, Mario AT, Aguiar, Pedro MQ, Smith, Noah A, and Xing, Eric P. An augmented lagrangian approach to constrained map inference. In *ICML*, 2011.
- Metz, Luke, Poole, Ben, Pfau, David, and Sohl-Dickstein, Jascha. Unrolled generative adversarial networks. *ICLR*, 2017.
- Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
- Mnih, Andriy and Hinton, Geoffrey. Learning nonlinear constraints with contrastive backpropagation. In *IJCNN*, 2005.
- Newcombe, Richard A, Izadi, Shahram, Hilliges, Otmar, Molyneaux, David, Kim, David, Davison, Andrew J, Kohi, Pushmeet, Shotton, Jamie, Hodges, Steve, and Fitzgibbon, Andrew. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE international symposium on Mixed and augmented reality*, 2011.
- Ngiam, Jiquan, Chen, Zhenghao, Koh, Pang W, and Ng, Andrew Y. Learning deep energy models. In *ICML*, 2011.
- Pearlmutter, Barak A. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.
- Punyakanok, Vasin, Roth, Dan, and Yih, Wen-tau. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34, 2008.
- Roth, Stefan and Black, Michael J. Fields of experts: A framework for learning image priors. In *CVPR*, 2005.
- Samuel, Kegan GG and Tappen, Marshall F. Learning optimized map estimates in continuously-valued mrf models. In *CVPR*, 2009.
- Schmidt, Uwe, Gao, Qi, and Roth, Stefan. A generative perspective on mrfs in low-level vision. In *CVPR*, 2010.
- Srivastava, Rupesh K, Greff, Klaus, and Schmidhuber, Jürgen. Training very deep networks. In *NIPS*, 2015.
- Stoyanov, Veselin, Ropson, Alexander, and Eisner, Jason. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *AISTATS*, 2011.
- Sun, Jian and Tappen, Marshall F. Learning non-local range markov random field for image restoration. In *CVPR*, 2011.
- Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *NIPS*, 2014.
- Täckström, Oscar, Ganchev, Kuzman, and Das, Dipanjan. Efficient inference and structured learning for semantic role labeling. *TACL*, 2015.

- Tappen, Marshall F, Liu, Ce, Adelson, Edward H, and Freeman, William T. Learning gaussian conditional random fields for low-level vision. In *CVPR*, 2007.
- Taskar, B., Guestrin, C., and Koller, D. Max-margin Markov networks. *NIPS*, 2004.
- Tompson, Jonathan J, Jain, Arjun, LeCun, Yann, and Bregler, Christoph. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pp. 1799–1807, 2014.
- Tsochantaridis, Ioannis, Hofmann, Thomas, Joachims, Thorsten, and Altun, Yasemin. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.
- Vilnis, Luke, Belanger, David, Sheldon, Daniel, and McCallum, Andrew. Bethe projections for non-local inference. *UAI*, 2015.
- Vincent, Pascal. A connection between score matching and denoising autoencoders. *Neural Computation*, 2011.
- Wang, Shenlong, Schwing, Alex, and Urtasun, Raquel. Efficient inference of continuous markov random fields with polynomial potentials. In *NIPS*, 2014.
- Wang, Shenlong, Fidler, Sanja, and Urtasun, Raquel. Proximal deep structured models. In *NIPS*, 2016.
- Zhai, Shuangfei, Cheng, Yu, Lu, Weining, and Zhang, Zhongfei. Deep structured energy based models for anomaly detection. In *ICML*, 2016.
- Zheng, Shuai, Jayasumana, Sadeep, Romera-Paredes, Bernardino, Vineet, Vibhav, Su, Zhizhong, Du, Dalong, Huang, Chang, and Torr, Philip HS. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.
- Zhou, Jie and Xu, Wei. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL*, 2015.

A. Appendix

A.1. General Learning Setup

The method described in Sections 3 and 4 provides a gradient of the loss with respect to the parameters of the model. To update the parameters, one can use any standard optimization method for neural networks. Our experiments use Adam (Kingma & Ba, 2015) with default settings. SPENs are vulnerable to overfitting, as the energy network is often very expressive. We reduce overfitting by performing early stopping, by taking the model that performs best on development data. Often, we have found that early stopping with a model that has a higher capacity energy (e.g., higher-dimensional hidden layers in the energy network) is superior to using a low-capacity energy.

A.2. Architectures for SRL Experiments

A.2.1. BASELINE ARC-FACTORED ARCHITECTURE

Our baseline is an arc-factored model for the conditional probability of the predicate-argument arc labels:

$$\mathbb{P}(\mathbf{r}|\mathbf{x}, \mathbf{p}, \mathbf{a}) = \prod_i \mathbb{P}(r_i|\mathbf{x}, \mathbf{p}, \mathbf{a}). \quad (18)$$

where $\mathbb{P}(r_i|\mathbf{x}, \mathbf{p}, \mathbf{a}) \propto \exp(g(r_i, \mathbf{x}, \mathbf{p}, \mathbf{a}))$. Here, each conditional distribution is given by a logistic regression model. We compute $g(r_i, \mathbf{x}, \mathbf{p}, \mathbf{a})$ using a multi-layer perceptron (MLP) similar to FitzGerald et al. (2015). Its inputs are discrete features extracted from the argument span and the predicate (including words, pos tags, and syntactic dependents), and the dependency path and distance between the argument and the predicate. These features are transformed to a 300-dimensional representation linearly, where the embeddings of word types are initialized using newswire embeddings from (Mikolov et al., 2013). We map from 300 dimensions to 250 to 47 (the number of semantic roles in CoNLL) using linear transformations separated by tanh layers. We apply dropout to the embedding layer with rate 0.5 and a standard log loss.

A.2.2. GLOBAL ENERGY TERM FOR SPEN

From the pre-trained model Eq. (18), we define \mathbf{f}_r as the predicate-argument arc features. We also have predicate features \mathbf{f}_p and argument feature \mathbf{f}_a , given by the average word embedding of the token spans. The hidden layers of any MLP below are 50-dimensional. Each MLP is two layers, with a SoftPlus in the middle. All parameters are trained discriminatively using end-to-end training.

Let $\mathbf{y}_p \in \Delta_A^m$ be the sub-tensor of \mathbf{y} for a given predicate p and let $\mathbf{z}_p = \sum_k \mathbf{y}_p[:, k] \in [0, 1]^m$, where $\mathbf{z}_p[a]$ is the total amount of mass assigned to the arc between predicate p and argument a , obtained by summing over possible labels. We also define $\mathbf{w}_p = \sum_k \mathbf{y}_p[k, :] \in \mathbb{R}_+^A$. This is a length- A

vector containing how much total mass of each arc label is assigned to predicate p . Finally, define $\mathbf{s}_r = \sum_k \mathbf{y}[:, :, k]$. This is the total mass assigned to arc r , obtained by summing over the possible labels that the arc can take on.

The global energy is defined by the sum of the following terms. The first energy term scores the set of arguments attached to each predicate. It computes a weighted average of the features \mathbf{f}_a for the arguments assigned to predicate p , with weights given by \mathbf{z}_p . It then concatenates this with \mathbf{f}_p , and passes the result through a two-layer multi-layer perceptron (MLP) that returns a single number. The total energy is the sum of the MLP output for every predicate. The second energy term scores the labels of the arcs attached to each predicate. We concatenate \mathbf{f}_p with \mathbf{w}_p and pass this through an MLP as above. The third energy term models how many arguments a predicate should take on. For each predicate, we predict how many arguments should attach to it, using a linear function applied to \mathbf{f}_p . The energy is set to the squared difference between this and the total mass attached to the predicate under \mathbf{y} , which is given by $\sum_k \mathbf{w}_p[k]$. The fourth energy term averages \mathbf{w}_p over all p and applies an MLP to the result. The fifth term computes a weighted average of the arc features \mathbf{f}_r , with weights given by \mathbf{s}_r and also applies an MLP to the result. The last two terms capture general topical coherence of the prediction.