End-to-End Learning for Structured Prediction Energy Networks

David Belanger [1] **Bishan Yang** [2] **Andrew McCallum** [1]

Abstract

Structured Prediction Energy Networks (Belanger & McCallum, 2016) (SPENs) are a simple, yet expressive family of structured prediction models. An energy function over candidate structured outputs is given by a deep network, and predictions are formed by gradient-based optimization. Unfortunately, we have struggled to apply the structured SVM (SSVM) learning method of Belanger & McCallum (2016) to applications with more complex structure than multi-label classification. In general, SSVMs are unreliable whenever exact energy minimization is intractable. In response, we present end-to-end learning for SPENs, where the energy function is discriminatively trained by back-propagating through gradient-based prediction. This paper presents a collection of methods necessary to apply the technique to problems with complex structure. For example, we avoid vanishing gradients when learning SPENs for convex relaxations of discrete prediction problems and explicitly train models such that energy minimization converges quickly in practice. Using end-to-end learning, we demonstrate the power of SPENs on 7-Scenes image denoising and CoNLL-2005 semantic role labeling tasks. In both, we outperform competitive baselines that employ more simplistic energy functions, but perform exact energy minimization. In particular, for denoising we achieve 40 PSNR, outperforming the previous state-of-the-art of 36.

1. Introduction

In a variety of domains, we predict a structured output $\mathbf{y}$, given input $\mathbf{x}$. 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, via energy minimization (LeCun et al., 2006):

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

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

This approach includes factor graphs, eg. conditional random fields (CRFs) (Lafferty et al., 2001) and many recurrent neural networks. Output constraints are enforced using constrained optimization. Compared to feed-forward approaches, energy minimization provides practitioners with better opportunities to inject prior knowledge about likely outputs, and often has more parsimonious models.

On the other hand, energy-based prediction requires nontrivial search in the space of outputs, and search techniques often need to be designed on a case-by-case basis. For factor graphs, the efficiency of prediction often scales poorly with treewidth of the graph.

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 graphical models 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 practioners 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; Tsochantaridis et al., 2004) and achieve competitive performance

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

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, 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).

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. First, we employ specific architectures, parameter tying schemes, and pretraining methods that reduce overfitting and improve efficiency. Second, we alleviate the effect of vanishing gradients when training SPENs defined over the convex relaxation of discrete prediction problems. Third, we train energies such that gradient-based minimization is fast. Fourth, we reduce SPENs' computation and memory overhead.

We demonstrate our SPEN training methods on two substantial tasks. Neither set of experiments presents SSVM results, as we could not achieve reasonable results using the method.

We first consider depth image denoising on the 7-Scenes dataset (Newcombe et al., 2011), where we employ deep convnets as priors over images. This provides substantial performance improvements, from 36 to 40 PSNR, over the state-of-the-art, which unrolls more sophisticated optimization than us, but uses a simpler prior (Wang et al., 2016) .

Next, we show the advantages of SPENs for semantic role labeling (SRL), with evaluation on the CoNLL-2005 dataset (Carreras & Màrquez, 2005). SRL extracts complex semantic structures over verbal predicates and phrasal arguments. The task is challenging for SPENs because the output is discrete, sparse, and subject to rigid non-local constraints. We show how to apply SPENs and demonstrate promising results over strong baselines that use deep features and encode structural constraints using traditional graphical models.

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 energy-based structured prediction Eq. (1) where the energy is given by a deep neural network that provides a subroutine for efficiently evaluating $\frac{d}{d\mathbf{y}} E_{\mathbf{x}}(\mathbf{y})$ (Belanger & McCallum, 2016). Differentiability necessitates that the energy is defined on continuous inputs. For the remainder of the paper, $\mathbf{y}$ 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* assume a graph over components of $\mathbf{y}$, where the energy decomposes over the set of cliques. This structure provides opportunities for (approximate) energy minimizaton using message passing, MCMC, or combinatorial solvers (Koller & Friedman, 2009). Second, *autoregressive models*, such as recurrent neural networks (RNNs) assume a partial ordering on the components of $\mathbf{y}$ such that the energy for component $\mathbf{y}_i$ 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 (Goodfellow et al., 2016).

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 depend crucially on leveraging some factorization structure.

2.2. SPEN Learning as Conditional Density Estimation

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

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

Unfortunately, computing the likelihood requires the partition function, 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 denosing autoencoders, energy-based models, and score matching (Vincent, 2011).

Overall, this density estimation may have high sample complexity because it seeks to fit the entire conditional distribution, rather than simply learning an energy function that yields high-quality predictions.

2.3. SPEN 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; Tsochantaridis et al., 2004):

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

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

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

For differentiable $\Delta(\mathbf{y}, \mathbf{y}_i)$, Eq. (4) can be solved by 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 this may fail to discover margin violations that exist. Performance can be recovered if Eq. (4) returns a lower bound (Finley & Joachims, 2008). However, this is not possible in general. Overall, our experiments avoid an empirical comparison between SSVM and end-to-end training, namely because SSVM learning is unstable and difficult to tune.

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 is advantageous to shape the entire energy surface such that is exhibits certain properties, e.g., gradient descent converges quickly when initialized well (Sec. 4.3). Therefore, these methods may be undesirable even for problems where exact energy minimization is tractable.

For non-convex $E_{\mathbf{x}}(\cdot)$, gradient-based prediction will only find a local optimum. Amos et al. (2016) 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 some results for instances of ICNNs. In general, non-convex SPENS perform slightly better.

3. Learning with Unrolled Optimization

The algorithms of Sec. 2.3 are unreliable with non-convex energies because we can not simply use the output of inexact energy minimization as a drop-in replacement for the exact minimizer. Instead, we draw on prior work for 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* (Stoyanov et al., 2011; Domke, 2013).

Consider simple gradient descent:

$$\mathbf{y}_T = y_0 - \sum_{t=1}^{T} \eta_t \frac{d}{d\mathbf{y}} E_{\mathbf{x}}(\mathbf{y}_t). \tag{5}$$

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, and can thus be trained using standard methods. Furthermore, certain hyperparameters, such as the learning rate 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 products. 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 (Domke, 2012). This requires an additional state vector $\mathbf{h}_t$ that is evolved along with $\mathbf{y}_t$ across iterations. Andrychowicz et al. (2016) also unroll gradient-descent, but employ a generic LSTM (Hochreiter & Schmidhuber, 1997), with per-coordinate updates to $\mathbf{y}$.

4. End-to-End Learning for SPENs

We now present details for end-to-end training of SPENs. We first describe architectures suitable for backpropating through gradient descent. We then 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.

4.1. Architectures

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

$$\mathbf{y}_T = \text{Init}(F(\mathbf{x})) - \sum_{t=1}^{T} \eta_t \frac{d}{d\mathbf{y}} E(\mathbf{y}_t \; ; \; F(\mathbf{x})). \quad (6)$$

Here, $\text{Init}(\cdot)$ is a differentiable procedure for predicting an initial iterate y_0. Following Belanger & McCallum (2016), we employ $E_{\mathbf{x}}(\cdot) = E(\cdot \; ; \; F(x))$, where the dependence of $E_{\mathbf{x}}(\cdot)$ on x comes by way of an arbitrary parametrized feature function $F(x)$. This is useful because test-time prediction can avoid back-propagation through $F(x)$.

We assume our energy splits into global and local terms:

$$E(\mathbf{y} \; ; F(x)) = E^g(\mathbf{y} \; ; F(\mathbf{x})) + \sum_i E^l(\mathbf{y}_i \; ; F(\mathbf{x})). \quad (7)$$

Here, i indexes the components of $\mathbf{y}$ and $E^g(\mathbf{y} \; ; F(\mathbf{x}))$ is an arbitrary deep network that provides a global function that couples components together. The local term is analogous to the local potentials in an undirected graphical model. We also use the local term to provide an implementation of the $\text{Init}(\cdot)$ network in Eq. (6).

For general continuous output problems, we employ $E^l(\mathbf{y}_i; F(\mathbf{x})) = (\mathbf{y}_i - G_i(F(x))^2$ and $\text{Init}(F(\mathbf{x})) = G_i(F(\mathbf{x}))$. For relaxations of discrete problems, we employ $E^l(\mathbf{y}_i; F(\mathbf{x})) = \mathbf{y}_i^\top G_i(F(\mathbf{x}))$ and $\text{Init}(F(\mathbf{x})) = \text{SoftMax}(G_i(F(\mathbf{x}))$. In both, $\text{Init}(\cdot)$ returns the expected $\mathbf{y}$ under the distribution $\mathbb{P}(\mathbf{y}) \propto \exp(-\sum_i E^l(\mathbf{y}_i \; ; F(\mathbf{x})))$. $G_i(\cdot)$ may be a learned network with extra parameters.

We pretrain our features $F(x)$ by training the feed-forward predictor $\text{Init}(F(x))$. 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. (6), 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. We compute Hessian-vector products using the finite-difference method of (Domke, 2012), as this allows black-box interaction with the energy.

Our experiments unroll either Eq. (6) or an analogous version implementing gradient descent with momentum. We avoid the LSTM-based approach of Andrychowicz et al. (2016) because it diminishes the semantics of the energy, since interaction between the optimizer and the energy is complicated and non-linear.

4.2. 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 probabilty simplex on D elements.

While this rounding introduces poorly-understood sources of error, it has worked well for 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. To optimize over them, we can employ existing methods for projected gradient optimization over the simplex.

For example, it would be natural to apply Euclidean pro-

jected gradient descent. Over $[0, 1]$, we have:

$$\mathbf{y}_{t+1} = \text{Clip}_{0,1}\left[\mathbf{y}_t - \eta_t \nabla E_{\mathbf{x}}(\mathbf{y}_t),\right] \tag{8}$$

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:

$$\mathbf{y}_{t+1} = \text{SoftMax}\left(\log(\mathbf{y}_t) - \eta_t \nabla E_{\mathbf{x}}(\mathbf{y}_t)\right) \tag{9}$$

This is applicable for end-to-end learning. However the updates are similar to a vanilla RNN with sigmoid activations, which is vulnerable to vanishing gradients (Hochreiter et al., 2001).

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

$$l_{t+1} = l_t - \eta_t \nabla E_{\mathbf{x}}\left(\text{SoftMax}(l_t)\right). \tag{10}$$

Here, the updates to l_t are additive, and thus will be less susceptible to vanishing gradients (Srivastava et al., 2015; He et al., 2016).

Finally, Amos et al. (2016) present the *bundle entropy method* for convex optimization with simplex constraints, along with a method for differentiating the output of the optimizer. However, like the implicit function theorem, they assume exact optimization. Also, learning for Eq. (6) can be performed using generic learning software, since the unrolled optimization obeys the API of a feed-forward predictor, but this is not true for their method.

4.3. 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 optimizaton by defining our loss function as a sum of losses on every iterate y_t, rather than only the final one. Let $\ell(y_t, y^*)$ be a differentiable loss between an iterate and the ground truth. We employ

$$L = \frac{1}{T} \sum_{t=1}^{T} w_t \ell(y_t, y^*), \tag{11}$$

where $w_t = \frac{1}{T-t+1}$. 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.

Second, for the simplex-constrained problems of Sec. 4.2, 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).

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. (6) 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 $T = 3$.

4.4. Efficient Implementation

Since we can explicitly encourage our model to converge quickly, it is important to exploit fast convergence at train time. Eq. (6) 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. (6) 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 y_t during the backwards pass. This is necessary for our experiments to be able to run on a 12GB GPU. To save additional memory, we could have avoided storing the intermediate iterates y_t, and instead reconstructed them on the fly (Maclaurin et al., 2015).

5. Image Denoising Experiments

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:

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

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 mimimizing Eq. (12) is high-quality. End-to-end learning has proven to be highly suc-

cesful 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.

5.1. 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:

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

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, replacing $|\cdot|$ with:

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

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:

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

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) can not handle this prior.

5.2. 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. Table 1 summarizes our results for a selection of configurations. Example outputs are given in Figure 1. We train using random 96×128 crops from 200 images of the same scene. We report PSNR for 5500 images from different scenes.

5.3. Results and Discussion

The first row of Table 1 presents various baselines. **BM3D** is a widely-used non-parametric method (Dabov et al., 2007). **FilterForest** adaptively selects denoising filters for each location (Fanello et al., 2014). **ProximalNet** is the system of Wang et al. (2016). **FOE-20** is an attempt to replicate **ProximalNet** 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 **ProximalNet**, which unrolls 5 iterations of sophisticated optimization. If we train a feed-forward convnet, using the same architecture as our DNN prior, but without spatial pooling, we obtain 36.95.

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. (11), 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 two results are with the DNN prior Eq. (15). **DeepPrior-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 **DeepPrior-3**, which only unrolls for 3 gradient steps but is otherwise identical.

In general, it is superior to only unroll for a few iterations. One possible reason is that the shallow depth of the unrolled architecture is easier to train. Truncated optimization with respect to $\mathbf{y}$ may also provide an interesting prior over outputs (Duvenaud et al., 2016), which can be particularly useful because the Gaussian noise model assumed in Eq. (12) is not characteristic of the data collection process (Wang et al., 2016). This is consistent with the observation of (Wang et al., 2014) that better energy minimization for FOE models may not improve PSNR. Also, note that unrolling for 20 iterations often results in over-smoothed outputs for all of the configurations.

Finally, the best performance we could obtain with an input-convex neural network is 39.77, using the same configuration as the DeepPrior-3. When the model takes so few gradient steps, it is unclear whether convexity matters. Note both Eq. (13) and our modification are convex wrt. $\mathbf{y}$.

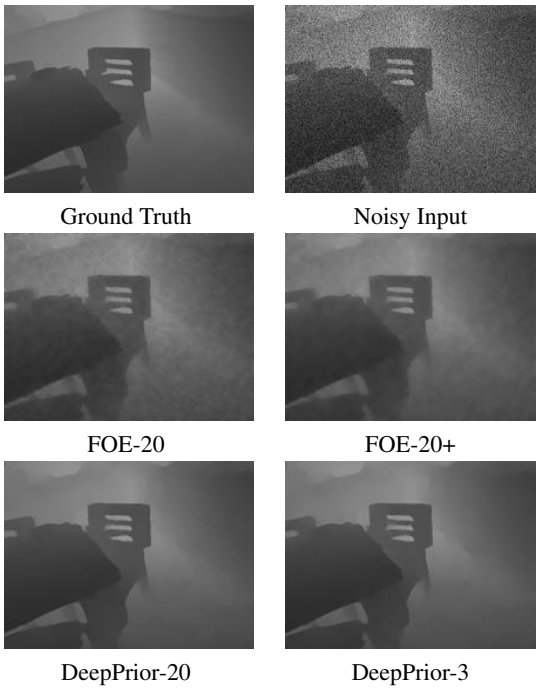

Figure 1. Example Denoising Outputs

BM3D	FilterForest	ProximalNet	FOE-20
35.46	35.63	36.31	36.41
FOE-20+	**FOE-3**	**DeepPrior-20**	**DeepPrior-3**
37.34	37.62	40.3	40.4

Table 1. Denoising Results (PSNR)

6. Semantic Role Labeling Experiments

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 $\mathbf{p}$ in a sentence $\mathbf{x}$ and a set of candidate argument spans $\mathbf{a}$, 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.

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

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

where $\mathcal{Q}(\mathbf{x}, \mathbf{p}, \mathbf{a})$ 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.

6.1. 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:

$$\mathbb{P}(\mathbf{r}|\mathbf{x}, \mathbf{p}, \mathbf{a}) = \Pi_i \mathbb{P}(r_i|\mathbf{x}, \mathbf{p}, \mathbf{a}). \qquad (17)$$

where $\mathbb{P}(r_i|\mathbf{x}, \mathbf{p}, \mathbf{a}) \propto \exp\big(g(r_i, \mathbf{x}, \mathbf{p}, \mathbf{a})\big)$. 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 train using Adam with default settings (Kingma & Ba, 2014) and a standard log loss.

When using the negative log of Eq. (17) as an energy in Eq. (16), there are variety of methods for finding the optimal $\mathbf{r} \in \mathcal{Q}(\mathbf{x}, \mathbf{p}, \mathbf{a})$. First, we can employ simple heuristics for locally resolving constraint violation. The **Local + H** system uses Eq. (17) 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 relaxation does not guarantee feasible outputs, we post-process the AD^3 output using the same heuristics as **Local + H**.

6.2. SPEN Model

We employ a pretrained version of Eq. (17) to provide the local energy term of a SPEN. This is augmented with global terms that couple the outputs together.

The SPEN performs continuous optimization over the re-

laxed 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.

6.2.1. Global Energy Terms

From the pre-trained model Eq. (17), 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 https://www.youtube.com/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.

6.2.2. Constraint Enforcement

As with Täckström et al. (2015), we seek to account for constraints $\mathcal{Q}(\mathbf{x}, \mathbf{p}, \mathbf{a})$ during both inference and learning, rather than only imposing them via post-processing. Therefore, we include additional energy terms that encode membership in $\mathcal{Q}(\mathbf{x}, \mathbf{p}, \mathbf{a})$ as twice-differentiable soft constraints that can be applied to $\mathbf{y}$. All of the constraints in $\mathcal{Q}(\mathbf{x}, \mathbf{p}, \mathbf{a})$ express that certain arcs can not co-occur. For example, two arguments can not 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 **SPEN + H** and **SPEN + AD^3**, which employ heuristics or AD3 to enforce the output constraints. Rather than applying these methods to the probabilities from Eq. (17), we use the output of energy minimization.

6.3. 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** systems and **Local** systems and the best non-ensemble systems of Täckström et al. (2015) and FitzGerald et al. (2015), which have similar overall setups as us. For these, 'Local' refers to fitting Eq. (17) without regard for the output constraints, whereas 'Structured' explicitly considers them during training. We select our SPEN configuration by maximizing performance of **SPEN + AD^3** on the dev data. Our best system unrolls for 5 iterations, trains per-iteration learning rates, and uses no momentum.

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 global term introduces many parameters.

Note that the gap between the **SPEN + AD^3** and **SPEN + H** systems is negligible. This is because our global energy encourages constraint satisfaction during gradient-based optimization of $\mathbf{y}$. Using the method of Amos et al. (2016) for restricting the energy to be convex wrt $\mathbf{y}$, we obtain 80.2 on the test set. When taking so few gradient steps, it is hard to understand the impact of convex energies.

7. Conclusion and Future Work

SPENs are a flexible, expressive framework for structured prediction. This paper provides a new training method that allows them to be applied to considerably more complex tasks than those of Belanger & McCallum (2016). We unroll an approximate energy minimization algorithm into a fixed-size computation graph that is trainable by gradient descent. In our experiments, good performance can often be achieved using just a few gradient steps on the energy surface. We encourage future work exploring properties of

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	78.5	80.3	69.5
SPEN + AD^3	78.5	**80.4**	69.6
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)

this truncated optimization. We also would like to perform iterative optimization in a learned feature space, rather than output space, as in Nguyen et al. (2016).

References

Amos, Brandon, Xu, Lei, and Kolter, J Zico. Input convex neural networks. *arXiv preprint: 1609.07152*, 2016.

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 nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3), 2003.

Belanger, David and McCallum, Andrew. Structured prediction energy networks. In *ICML*, 2016.

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.

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*, volume 22, pp. 318–326, 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*, pp. 377–384, 2008.

Gildea, Daniel and Jurafsky, Daniel. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016.

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.

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.

Hochreiter, Sepp, Bengio, Yoshua, Frasconi, Paolo, and Schmidhuber, Jürgen. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Lafferty, John D, McCallum, Andrew, and Pereira, Fernando CN. 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.

Li, Yujia and Zemel, Richard S. Mean-field networks. *ICML Workshop on Learning Tractable Probabilistic Models*, 2014.

Maclaurin, Dougal, Duvenaud, David, and Adams, Ryan P. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, July 2015.

Martins, André FT, Figeuiredo, Mario AT, Aguiar, Pedro MQ, Smith, Noah A, and Xing, Eric P. An augmented lagrangian approach to constrained map inference. 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.

Nguyen, Anh, Yosinski, Jason, Bengio, Yoshua, Dosovitskiy, Alexey, and Clune, Jeff. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv pre-print 1612.00005*, 2016.

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.

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.

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*, pp. 19–24, 2016.