

Improving Joint Training of Inference Networks and Structured Prediction Energy Networks

Lifu Tu¹ Richard Yuanzhe Pang^{2*} Kevin Gimpel¹

¹Toyota Technological Institute at Chicago, Chicago, IL 60637, USA

²New York University, New York, NY 10011, USA

lifu@ttic.edu, yzpang@nyu.edu, kgimpel@ttic.edu

Abstract

Deep energy-based models are powerful, but pose challenges for learning and inference (Belanger and McCallum, 2016). Tu and Gimpel (2018) developed an efficient framework for energy-based models by training “inference networks” to approximate structured inference instead of using gradient descent. However, their alternating optimization approach suffers from instabilities during training, requiring additional loss terms and careful hyperparameter tuning. In this paper, we contribute several strategies to stabilize and improve this joint training of energy functions and inference networks for structured prediction. We design a compound objective to jointly train both cost-augmented and test-time inference networks along with the energy function. We propose joint parameterizations for the inference networks that encourage them to capture complementary functionality during learning. We empirically validate our strategies on two sequence labeling tasks, showing easier paths to strong performance than prior work, as well as further improvements with global energy terms.

1 Introduction

Energy-based modeling (LeCun et al., 2006) associates a scalar compatibility measure to each configuration of input and output variables. Belanger and McCallum (2016) formulated deep energy-based models for structured prediction, which they called structured prediction energy networks (SPENs). SPENs use arbitrary neural networks to define the scoring function over input/output pairs. However, this flexibility leads to challenges for learning and inference. The original work on SPENs used gradient descent for structured inference (Belanger and McCallum, 2016; Belanger et al., 2017). Tu

and Gimpel (2018, 2019) found improvements in both speed and accuracy by replacing the use of gradient descent with a method that trains a neural network (called an “inference network”) to do inference directly. Their formulation, which jointly trains the inference network and energy function, is similar to training in generative adversarial networks (Goodfellow et al., 2014), which is known to suffer from practical difficulties in training due to the use of alternating optimization (Salimans et al., 2016). To stabilize training, Tu and Gimpel (2018) experimented with several additional terms in the training objectives, finding performance to be dependent on their inclusion.

Moreover, when using the approach of Tu and Gimpel (2018), there is a mismatch between the training and test-time uses of the trained inference network. During training with hinge loss, the inference network is actually trained to do “cost-augmented” inference. However, at test time, the goal is to simply minimize the energy without any cost term. Tu and Gimpel (2018) fine-tuned the cost-augmented network to match the test-time criterion, but found only minimal change from this fine-tuning. This suggests that the cost-augmented network was mostly acting as a test-time inference network by convergence, which may be hindering the potential contributions of cost-augmented inference in max-margin structured learning (Tsochantzidis et al., 2004; Taskar et al., 2004).

In this paper, we contribute a new training objective for SPENs that addresses the above concern and also contribute several techniques for stabilizing and improving learning. We empirically validate our strategies on two sequence labeling tasks from natural language processing (NLP), namely part-of-speech tagging and named entity recognition. We show easier paths to strong performance than prior work, as well as further improvements with global energy terms. We summarize our list

* Work done at the University of Chicago and Toyota Technological Institute at Chicago.

of contributions as follows.

- We design a compound objective under the SPEN framework to jointly train the “training-time” cost-augmented inference network and test-time inference network (Section 3).
- We propose shared parameterizations for the two inference networks so as to encourage them to capture complementary functionality while reducing the total number of trained parameters (Section 3.1). Quantitative and qualitative analysis shows clear differences in the characteristics of the two networks (Table 3).
- We present three methods to streamline and stabilize training that help with both the old and new objectives (Section 4).
- We propose global energy terms to capture long-distance dependencies and obtain further improvements (Section 5).

While SPENs have been used for multiple NLP tasks, including multi-label classification (Belanger and McCallum, 2016), part-of-speech tagging (Tu and Gimpel, 2018), and semantic role labeling (Belanger et al., 2017), they are not widely used in NLP. Structured prediction is extremely common in NLP, but is typically approached using methods that are more limited than SPENs (such as conditional random fields) or models that suffer from a train/test mismatch (such as most auto-regressive models). SPENs offer a maximally expressive framework for structured prediction while avoiding the train/test mismatch and therefore offer great potential for NLP. However, the training and inference have deterred NLP researchers. While we have found benefit from training inference networks for machine translation in recent work (Tu et al., 2020b), that work assumed a fixed, pretrained energy function. Our hope is that the methods in this paper will enable SPENs to be applied to a larger set of applications, including generation tasks in the future.

2 Background

We denote the input space by \mathcal{X} . For an input $\mathbf{x} \in \mathcal{X}$, we denote the structured output space by $\mathcal{Y}(\mathbf{x})$. The entire space of structured outputs is denoted $\mathcal{Y} = \cup_{\mathbf{x} \in \mathcal{X}} \mathcal{Y}(\mathbf{x})$. A SPEN (Belanger and McCallum, 2016) defines an **energy function** $E_{\Theta} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ parameterized by Θ that computes a scalar energy for an input/output pair. At test time,

for a given input \mathbf{x} , prediction is done by choosing the output with lowest energy:

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

However, solving equation (1) requires combinatorial algorithms because \mathcal{Y} is a structured, discrete space. This becomes intractable when E_{Θ} does not decompose into a sum over small “parts” of \mathbf{y} . Belanger and McCallum (2016) relaxed this problem by allowing the **discrete vector** \mathbf{y} to be continuous; \mathcal{Y}_R denotes the **relaxed output space**. They solved the relaxed problem by using gradient descent to iteratively minimize the energy with respect to \mathbf{y} . The energy function parameters Θ are trained using a structured hinge loss which requires repeated cost-augmented inference during training. Using gradient descent for the repeated cost-augmented inference steps is time-consuming and makes learning unstable (Belanger et al., 2017).

Tu and Gimpel (2018) replaced gradient descent with a neural network trained to do efficient inference. This “inference network” $\mathbf{A}_{\Psi} : \mathcal{X} \rightarrow \mathcal{Y}_R$ is parameterized by Ψ and trained with the goal that

$$\mathbf{A}_{\Psi}(\mathbf{x}) \approx \arg \min_{\mathbf{y} \in \mathcal{Y}_R(\mathbf{x})} E_{\Theta}(\mathbf{x}, \mathbf{y}) \quad (2)$$

When training the energy function parameters Θ , Tu and Gimpel (2018) replaced the cost-augmented inference step in the structured hinge loss from Belanger and McCallum (2016) with a cost-augmented inference network \mathbf{F}_{Φ} :

$$\mathbf{F}_{\Phi}(\mathbf{x}) \approx \arg \min_{\mathbf{y} \in \mathcal{Y}_R(\mathbf{x})} (E_{\Theta}(\mathbf{x}, \mathbf{y}) - \Delta(\mathbf{y}, \mathbf{y}^*)) \quad (3)$$

where Δ is a structured cost function that computes the distance between its two arguments. We use L1 distance for Δ . This inference problem involves finding an output with low energy but high cost relative to the gold standard. Thus, it is not well-aligned with the test-time inference problem.

Here is the specific objective to jointly train Θ (parameters of the energy function) and Φ (parameters of the cost-augmented inference network):

$$\min_{\Theta} \max_{\Phi} \sum_{\langle \mathbf{x}_i, \mathbf{y}_i \rangle \in \mathcal{D}} [\Delta(\mathbf{F}_{\Phi}(\mathbf{x}_i), \mathbf{y}_i) - E_{\Theta}(\mathbf{x}_i, \mathbf{F}_{\Phi}(\mathbf{x}_i)) + E_{\Theta}(\mathbf{x}_i, \mathbf{y}_i)]_+ \quad (4)$$

where \mathcal{D} is the set of training pairs, $[h]_+ = \max(0, h)$, and Δ is a structured cost function that computes the distance between its two arguments.

Tu and Gimpel (2018) alternatively optimized Θ and Φ , which is similar to training in generative adversarial networks (Goodfellow et al., 2014). The inference network is analogous to the generator and the energy function is analogous to the discriminator. As alternating optimization can be difficult in practice (Salimans et al., 2016), Tu & Gimpel experimented with including several additional terms in the above objective to stabilize training.

After the training of the energy function, an inference network \mathbf{A}_Ψ for test-time prediction is fine-tuned with the goal shown in Eq. (2). More specifically, for the fine-tuning step, we first initialize Ψ with Φ ; next, we do gradient descent according to the following objective to learn Ψ :

$$\Psi \leftarrow \arg \min_{\Psi'} \sum_{\mathbf{x} \in \mathcal{X}} E_\Theta(\mathbf{x}, \mathbf{A}_{\Psi'}(\mathbf{x}))$$

where \mathcal{X} is a set of training or validation inputs. It could also be the test inputs in a transductive setting.

3 An Objective for Joint Learning of Inference Networks

One challenge with the above optimization problem is that it requires training a separate inference network \mathbf{A}_Ψ for test-time prediction after the energy function is trained. In this paper, we propose an alternative that trains the energy function and both inference networks jointly. In particular, we use a ‘‘compound’’ objective that combines two widely-used losses in structured prediction. We first present it without inference networks:

$$\min_{\Theta} \sum_{\langle \mathbf{x}_i, \mathbf{y}_i \rangle \in \mathcal{D}} \left[\underbrace{\max_{\mathbf{y}} (\Delta(\mathbf{y}, \mathbf{y}_i) - E_\Theta(\mathbf{x}_i, \mathbf{y}) + E_\Theta(\mathbf{x}_i, \mathbf{y}_i))}_{\text{margin-rescaled hinge loss}} \right]_+ + \lambda \left[\underbrace{\max_{\mathbf{y}} (-E_\Theta(\mathbf{x}_i, \mathbf{y}) + E_\Theta(\mathbf{x}_i, \mathbf{y}_i))}_{\text{perceptron loss}} \right]_+ \quad (5)$$

As indicated, this loss can be viewed as the sum of the margin-rescaled hinge and perceptron losses for SPENs. Two different inference problems are represented. The margin-rescaled hinge loss contains cost-augmented inference, shown as part of Eq. (3). The perceptron loss contains the test-time inference problem, which is shown in Eq. (1). Tu

and Gimpel (2018) used a single inference network for solving both problems, so it was trained as a cost-augmented inference network during training and then fine-tuned as a test-time inference network afterward. We avoid this issue by training two inference networks, \mathbf{A}_Ψ for test-time inference and \mathbf{F}_Φ for cost-augmented inference:

$$\min_{\Theta} \max_{\Phi, \Psi} \sum_{\langle \mathbf{x}_i, \mathbf{y}_i \rangle \in \mathcal{D}} [\Delta(\mathbf{F}_\Phi(\mathbf{x}_i), \mathbf{y}_i) - E_\Theta(\mathbf{x}_i, \mathbf{F}_\Phi(\mathbf{x}_i)) + E_\Theta(\mathbf{x}_i, \mathbf{y}_i)]_+ + \lambda [-E_\Theta(\mathbf{x}_i, \mathbf{A}_\Psi(\mathbf{x}_i)) + E_\Theta(\mathbf{x}_i, \mathbf{y}_i)]_+ \quad (6)$$

We treat this optimization problem as a minimax game and find a saddle point for the game similar to Tu and Gimpel (2018) and Goodfellow et al. (2014). We use minibatch stochastic gradient descent and alternately optimize Θ , Φ , and Ψ . The objective for the energy parameters Θ in minibatch \mathcal{M} is:

$$\hat{\Theta} \leftarrow \arg \min_{\Theta} \sum_{\langle \mathbf{x}_i, \mathbf{y}_i \rangle \in \mathcal{M}} [\Delta(\mathbf{F}_\Phi(\mathbf{x}_i), \mathbf{y}_i) - E_\Theta(\mathbf{x}_i, \mathbf{F}_\Phi(\mathbf{x}_i)) + E_\Theta(\mathbf{x}_i, \mathbf{y}_i)]_+ + \lambda [-E_\Theta(\mathbf{x}_i, \mathbf{A}_\Psi(\mathbf{x}_i)) + E_\Theta(\mathbf{x}_i, \mathbf{y}_i)]_+$$

When we remove 0-truncation (see Sec. 4.1 for the motivation), the objective for the inference network parameters in minibatch \mathcal{M} is:

$$\hat{\Psi}, \hat{\Phi} \leftarrow \arg \max_{\Psi, \Phi} \sum_{\langle \mathbf{x}_i, \mathbf{y}_i \rangle \in \mathcal{M}} \Delta(\mathbf{F}_\Phi(\mathbf{x}_i), \mathbf{y}_i) - E_\Theta(\mathbf{x}_i, \mathbf{F}_\Phi(\mathbf{x}_i)) - \lambda E_\Theta(\mathbf{x}_i, \mathbf{A}_\Psi(\mathbf{x}_i))$$

3.1 Joint Parameterizations

If we were to train independent inference networks \mathbf{A}_Ψ and \mathbf{F}_Φ , this new objective could be much slower than the approach of Tu and Gimpel (2018). However, the compound objective offers several natural options for defining joint parameterizations of the two inference networks. We consider three options which are visualized in Figure 1 and described below:

- **separated:** \mathbf{F}_Φ and \mathbf{A}_Ψ are two independent networks with their own architectures and parameters as shown in Figure 1(a).
- **shared:** \mathbf{F}_Φ and \mathbf{A}_Ψ share a ‘‘feature’’ network as shown in Figure 1(b). We consider this option because both \mathbf{F}_Φ and \mathbf{A}_Ψ are trained to produce output labels with low energy. However \mathbf{F}_Φ also needs to produce output labels with high cost Δ (i.e., far from the gold standard).

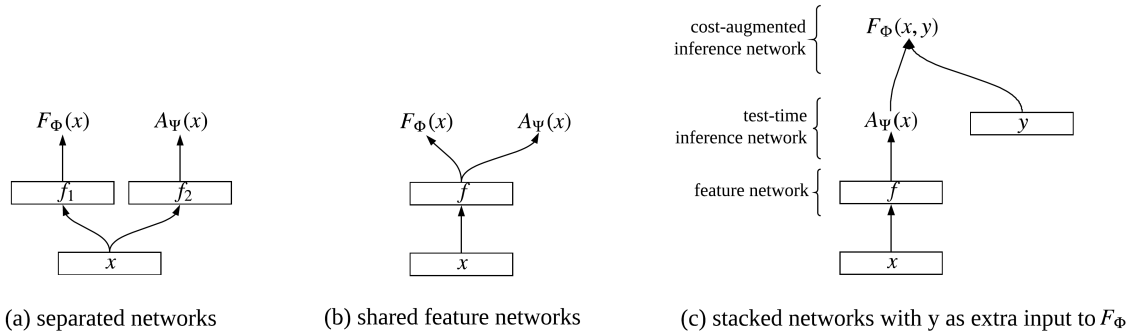


Figure 1: Joint parameterizations for cost-augmented inference network \mathbf{F}_Φ and test-time inference network \mathbf{A}_Ψ .

- **stacked**: the cost-augmented network \mathbf{F}_Φ is a function of the output of the test-time network \mathbf{A}_Ψ and the gold standard output \mathbf{y} . That is, $\mathbf{F}_\Phi(\mathbf{x}, \mathbf{y}) = q(\mathbf{A}_\Psi(\mathbf{x}), \mathbf{y})$ where q is a parameterized function. This is depicted in Figure 1(c). We block the gradient at \mathbf{A}_Ψ when updating Φ .

For the q function in the stacked option, we use an affine transformation on the concatenation of the inference network label distribution and the gold standard one-hot vector. That is, denoting the vector at position t of the cost-augmented network output by $\mathbf{F}_\Phi(\mathbf{x}, \mathbf{y})_t$, we have:

$$\mathbf{F}_\Phi(\mathbf{x}, \mathbf{y})_t = \text{softmax}(W_q[\mathbf{A}_\Psi(\mathbf{x})_t; \mathbf{y}(t)] + b_q)$$

where semicolon (;) is vertical concatenation, $\mathbf{y}(t)$ (position t of \mathbf{y}) is an L -dimensional one-hot vector, $\mathbf{A}_\Psi(\mathbf{x})_t$ is the vector at position t of $\mathbf{A}_\Psi(\mathbf{x})$, W_q is an $L \times 2L$ matrix, and b_q is a bias.

One motivation for these parameterizations is to reduce the total number of parameters in the procedure. Generally, the number of parameters is expected to decrease when moving from separated to shared to stacked. We will compare the three options empirically in our experiments, in terms of both accuracy and number of parameters.

Another motivation, specifically for the third option, is to distinguish the two inference networks in terms of their learned functionality. With all three parameterizations, the cost-augmented network will be trained to produce an output that differs from the gold standard, due to the presence of the $\Delta(\cdot)$ term in the combined objective. However, [Tu and Gimpel \(2018\)](#) found that the trained cost-augmented network was barely affected by fine-tuning for the test-time inference objective. This suggests that the cost-augmented network was

mostly acting as a test-time inference network by the time of convergence. With the stacked parameterization, however, we explicitly provide the gold standard \mathbf{y} to the cost-augmented network, permitting it to learn to change the predictions of the test-time network in appropriate ways to improve the energy function.

4 Training Stability and Effectiveness

We now discuss several methods that simplify and stabilize training SPENs with inference networks. When describing them, we will illustrate their impact by showing training trajectories for the Twitter part-of-speech tagging task.

4.1 Removing Zero Truncation

[Tu and Gimpel \(2018\)](#) used the following objective for the cost-augmented inference network (maximizing it with respect to Φ): $l_0 =$

$$[\Delta(\mathbf{F}_\Phi(\mathbf{x}), \mathbf{y}) - E_\Theta(\mathbf{x}, \mathbf{F}_\Phi(\mathbf{x})) + E_\Theta(\mathbf{x}, \mathbf{y})]_+$$

where $[h]_+ = \max(0, h)$. However, there are two potential reasons why l_0 will equal zero and trigger no gradient update. First, E_Θ (the energy function, corresponding to the discriminator in a GAN) may already be well-trained, and it can easily separate the gold standard output from the cost-augmented inference network output. Second, the cost-augmented inference network (corresponding to the generator in a GAN) could be so poorly trained that the energy of its output is very large, leading the margin constraints to be satisfied and l_0 to be zero.

In standard margin-rescaled max-margin learning in structured prediction ([Taskar et al., 2004](#); [Tsochantaridis et al., 2004](#)), the cost-augmented inference step is performed exactly (or approximately

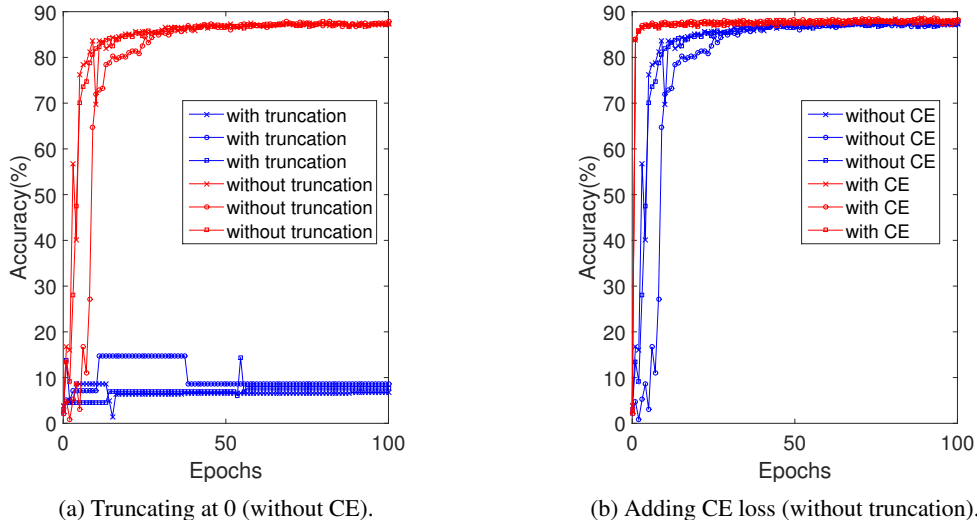


Figure 2: Part-of-speech tagging training trajectories. The three curves in each setting correspond to different random seeds. (a) Without the local CE loss, training fails when using zero truncation. (b) The CE loss reduces the number of epochs for training. Tu and Gimpel (2018) used zero truncation and CE during training.

with reasonable guarantee of effectiveness), ensuring that when l_0 is zero, the energy parameters are well trained. However, in our case, l_0 may be zero simply because the cost-augmented inference network is undertrained, which will be the case early in training. Then, when using zero truncation, the gradient of the inference network parameters will be 0. This is likely why Tu and Gimpel (2018) found it important to add several stabilization terms to the l_0 objective. We find that by instead removing the truncation, learning stabilizes and becomes less dependent on these additional terms. Note that we retain the truncation at zero when updating the energy parameters Θ .

As shown in Figure 2(a), without any stabilization terms and with truncation, the inference network will barely move from its starting point and learning fails overall. However, without truncation, the inference network can work well even without any stabilization terms.

4.2 Local Cross Entropy (CE) Loss

Tu and Gimpel (2018) proposed adding a local cross entropy (CE) loss, which is the sum of the label cross entropy losses over all positions in the sequence, to stabilize inference network training. We similarly find this term to help speed up convergence and improve accuracy. Figure 2(b) shows faster convergence to high accuracy when adding the local CE term. See Section 7 for more details.

4.3 Multiple Inference Network Update Steps

When training SPENs with inference networks, the inference network parameters are nested within the energy function. We found that the gradient components of the inference network parameters consequently have smaller absolute values than those of the energy function parameters. So, we alternate between $k \geq 1$ steps of optimizing the inference network parameters (“I steps”) and one step of optimizing the energy function parameters (“E steps”). We find this strategy especially helpful when using complex inference network architectures.

To analyze, we compute the cost-augmented loss $l_1 = \Delta(\mathbf{F}_\Phi(\mathbf{x}), \mathbf{y}) - E_\Theta(\mathbf{x}, \mathbf{F}_\Phi(\mathbf{x}))$ and the margin-rescaled hinge loss $l_0 = [\Delta(\mathbf{F}_\Phi(\mathbf{x}), \mathbf{y}) - E_\Theta(\mathbf{x}, \mathbf{F}_\Phi(\mathbf{x})) + E_\Theta(\mathbf{x}, \mathbf{y})]_+$ averaged over all training pairs (\mathbf{x}, \mathbf{y}) after each set of I steps. The I steps update Ψ and Φ to maximize these losses. Meanwhile the E steps update Θ to minimize these losses. Figs. 3(a) and (b) show l_1 and l_0 during training for different numbers (k) of I steps for every one E step. Fig. 3(c) shows the norm of the energy parameters after the E steps, and Fig. 3(d) shows the norm of $\frac{\partial E_\Theta(\mathbf{x}, \mathbf{A}_\Psi)}{\partial \Psi}$ after the I steps.

With $k = 1$, the setting used by Tu and Gimpel (2018), the inference network lags behind the energy, making the energy parameter updates very small, as shown by the small norms in Fig. 3(c). The inference network gradient norm (Fig. 3(d)) remains high, indicating underfitting. However, increasing k too much also harms learning, as evi-

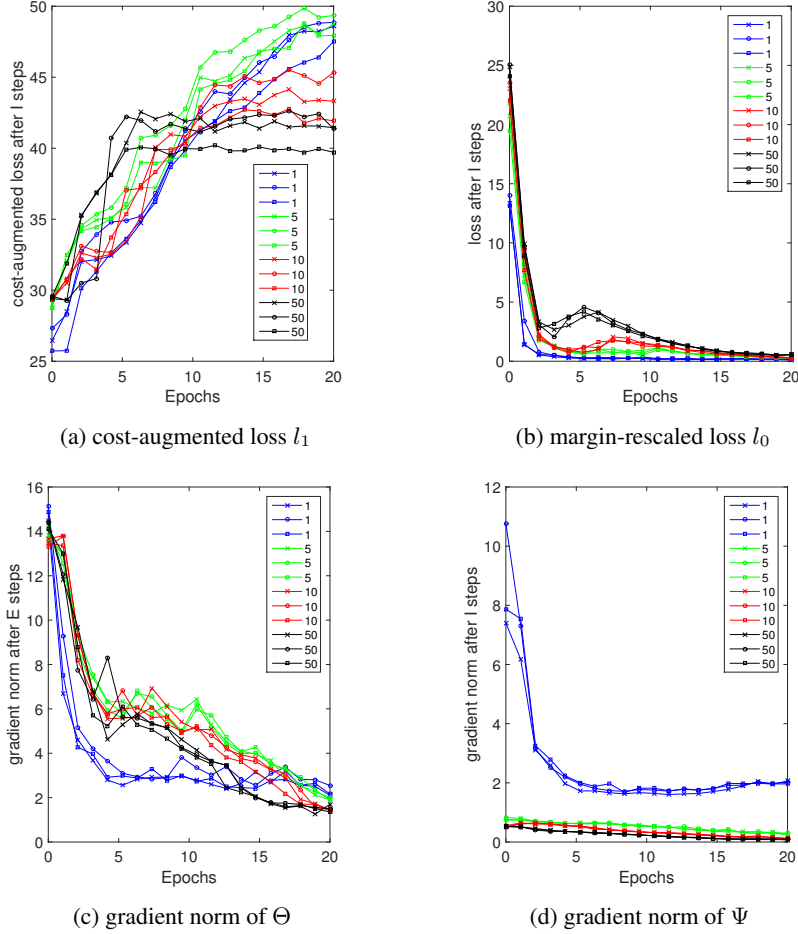


Figure 3: POS training trajectories with different numbers of I steps. The three curves in each setting correspond to different random seeds. (a) cost-augmented loss after I steps; (b) margin-rescaled hinge loss after I steps; (c) gradient norm of energy function parameters after E steps; (d) gradient norm of test-time inference network parameters after I steps.

denced by the “plateau” effect in the l_1 curves for $k = 50$; this indicates that the energy function is lagging behind the inference network. Using $k = 5$ leads to more of a balance between l_1 and l_0 and gradient norms that are mostly decreasing during training. We treat k as a hyperparameter that is tuned in our experiments.

There is a potential connection between our use of multiple I steps and a similar procedure used in GANs (Goodfellow et al., 2014). In the GAN objective, the discriminator D is updated in the inner loop, and they alternate between multiple update steps for D and one update step for G . In this section, we similarly found benefit from multiple steps of inner loop optimization for every step of the outer loop. However, the analogy is limited, since GAN training involves sampling noise vectors and using them to generate data, while there are no noise vectors or explicitly-generated samples in our framework.

5 Energies for Sequence Labeling

For our sequence labeling experiments in this paper, the input \mathbf{x} is a length- T sequence of tokens, and the output \mathbf{y} is a sequence of labels of length T . We use \mathbf{y}_t to denote the output label at position t , where \mathbf{y}_t is a vector of length L (the number of labels in the label set) and where $y_{t,j}$ is the j th entry of the vector \mathbf{y}_t . In the original output space $\mathcal{Y}(\mathbf{x})$, $y_{t,j}$ is 1 for a single j and 0 for all others. In the relaxed output space $\mathcal{Y}_R(\mathbf{x})$, $y_{t,j}$ can be interpreted as the probability of the t th position being labeled with label j . We then use the following energy for sequence labeling (Tu and Gimpel, 2018):

$$E_{\Theta}(\mathbf{x}, \mathbf{y}) = - \left(\sum_{t=1}^T \sum_{j=1}^L y_{t,j} \left(U_j^{\top} b(\mathbf{x}, t) \right) + \sum_{t=1}^T \mathbf{y}_{t-1}^{\top} W \mathbf{y}_t \right) \quad (7)$$

where $U_j \in \mathbb{R}^d$ is a parameter vector for label j and the parameter matrix $W \in \mathbb{R}^{L \times L}$ contains label-pair parameters. Also, $b(\mathbf{x}, t) \in \mathbb{R}^d$ denotes the ‘‘input feature vector’’ for position t . We define b to be the d -dimensional BiLSTM (Hochreiter and Schmidhuber, 1997) hidden vector at t . The full set of energy parameters Θ includes the U_j vectors, W , and the parameters of the BiLSTM.

Global Energies for Sequence Labeling. In addition to new training strategies, we also experiment with several global energy terms for sequence labeling. Eq. (7) shows the base energy, and to capture long-distance dependencies, we include global energy (GE) terms in the form of Eq. (8).

We use h to denote an LSTM tag language model (TLM) that takes a sequence of labels as input and returns a distribution over next labels. We define $\bar{\mathbf{y}}_t = h(\mathbf{y}_0, \dots, \mathbf{y}_{t-1})$ to be the distribution given the preceding label vectors (under a LSTM language model). Then, the energy term is:

$$E^{\text{TLM}}(\mathbf{y}) = - \sum_{t=1}^{T+1} \log(\mathbf{y}_t^\top \bar{\mathbf{y}}_t) \quad (8)$$

where \mathbf{y}_0 is the start-of-sequence symbol and \mathbf{y}_{T+1} is the end-of-sequence symbol. This energy returns the negative log-likelihood under the TLM of the candidate output \mathbf{y} . Tu and Gimpel (2018) pretrained their h on a large, automatically-tagged corpus and fixed its parameters when optimizing Θ . Our approach has one critical difference. We instead *do not* pretrain h , and its parameters are learned when optimizing Θ . We show that even without pretraining, our global energy terms are still able to capture useful additional information.

We also propose new global energy terms. Define $\bar{\mathbf{y}}_t = h(\mathbf{y}_0, \dots, \mathbf{y}_{t-1})$ where h is an LSTM TLM that takes a sequence of labels as input and returns a distribution over next labels. First, we add a TLM in the backward direction (denoted $\bar{\mathbf{y}}'_t$ analogously to the forward TLM). Second, we include words as additional inputs to forward and backward TLMs. We define $\tilde{\mathbf{y}}_t = g(\mathbf{x}_0, \dots, \mathbf{x}_{t-1}, \mathbf{y}_0, \dots, \mathbf{y}_{t-1})$ where g is a forward LSTM TLM. We define the backward version similarly (denoted $\tilde{\mathbf{y}}'_t$). The global energy is therefore

$$E^{\text{GE}}(\mathbf{y}) = - \sum_{t=1}^{T+1} \log(\mathbf{y}_t^\top \bar{\mathbf{y}}_t) + \log(\mathbf{y}_t^\top \bar{\mathbf{y}}'_t) + \gamma (\log(\mathbf{y}_t^\top \tilde{\mathbf{y}}_t) + \log(\mathbf{y}_t^\top \tilde{\mathbf{y}}'_t)) \quad (9)$$

Here γ is a hyperparameter that is tuned. We experiment with three settings for the global energy: GE(a): forward TLM as in Tu and Gimpel (2018); GE(b): forward and backward TLMs ($\gamma = 0$); GE(c): all four TLMs in Eq. (9).

6 Experimental Setup

We consider two sequence labeling tasks: Twitter part-of-speech (POS) tagging (Gimpel et al., 2011) and named entity recognition (NER; Tjong Kim Sang and De Meulder, 2003).

Twitter Part-of-Speech (POS) Tagging. We use the Twitter POS data from Gimpel et al. (2011) and Owoputi et al. (2013) which contain 25 tags. We use 100-dimensional skip-gram (Mikolov et al., 2013) embeddings from Tu et al. (2017). Like Tu and Gimpel (2018), we use a BiLSTM to compute the input feature vector for each position, using hidden size 100. We also use BiLSTMs for the inference networks. The output of the inference network is a softmax function, so the inference network will produce a distribution over labels at each position. The Δ is L1 distance. We train the inference network using stochastic gradient descent (SGD) with momentum and train the energy parameters using Adam (Kingma and Ba, 2014). We also explore training the inference network using Adam when not using the local CE loss.¹ In experiments with the local CE term, its weight is set to 1.

Named Entity Recognition (NER). We use the CoNLL 2003 English dataset (Tjong Kim Sang and De Meulder, 2003). We use the BIOES tagging scheme, following previous work (Ratinov and Roth, 2009), resulting in 17 NER labels. We use 100-dimensional pretrained GloVe embeddings (Pennington et al., 2014). The task is evaluated using F1 score computed with the `conlleval` script. The architectures for the feature networks in the energy function and inference networks are all BiLSTMs. The architectures for tag language models are LSTMs. We use a dropout `keep-prob` of 0.7 for all LSTM cells. The hidden size for all LSTMs is 128. We use Adam (Kingma and Ba, 2014) and do early stopping on the development set. We use a learning rate of $5 \cdot 10^{-4}$. Similar to above, the weight for the CE term is set to 1.

We consider three NER modeling configurations. **NER** uses only words as input and pretrained, fixed

¹We find that Adam works better than SGD when training the inference network without the local cross entropy term.

loss	zero trunc.	CE	POS	NER	NER+
			acc (%)	F1 (%)	F1 (%)
margin-rescaled	yes	no	13.9	3.91	3.91
	no	no	87.9	85.1	88.6
	yes	yes	89.4*	85.2*	89.5*
	no	yes	89.4	85.2	89.5
perceptron	no	no	88.2	84.0	88.1
	no	yes	88.6	84.7	89.0

Table 1: Test results for POS and NER for several SPEN configurations. Results with * correspond to the setting of Tu and Gimpel (2018). The inference network architecture is a one-layer BiLSTM.

GloVe embeddings. **NER+** uses words, the case of the first letter, POS tags, and chunk labels, as well as pretrained GloVe embeddings with fine-tuning. **NER++** includes everything in **NER+** as well as character-based word representations obtained using a convolutional network over the character sequence in each word. Unless otherwise indicated, our SPENs use the energy in Eq. (7).

7 Results and Analysis

Effect of Zero Truncation and Local CE Loss.

Table 1 shows results for zero truncation and the local CE term. Training fails for both tasks when using zero truncation without CE. Removing truncation makes learning succeed and leads to effective models even without using CE. However, when using the local CE term, truncation has little effect on performance. The importance of CE in prior work (Tu and Gimpel, 2018) is likely due to the fact that truncation was being used.

The local CE term is useful for both tasks, though it appears more helpful for tagging.² This may be because POS tagging is a more local task. Regardless, for both tasks, as shown in Section 4.2, the inclusion of the CE term speeds convergence and improves training stability. For example, on NER, using the CE term reduces the number of epochs chosen by early stopping from ~ 100 to ~ 25 . For POS, using the CE term reduces the number of epochs from ~ 150 to ~ 60 .

Effect of Compound Objective and Joint Parameterizations.

The compound objective is the sum of the margin-rescaled and perceptron losses, and outperforms them both (see Table 2). Across all tasks, the shared and stacked parameterizations are more accurate than the previous objectives. For the separated parameterization, the performance

²We found the local CE term to be useful for both the cost-augmented and test-time inference networks during training.

drops slightly for NER, likely due to the larger number of parameters. The shared and stacked options have fewer parameters to train than the separated option, and the stacked version processes examples at the fastest rate during training.

The top part of Table 3 shows how the performance of the test-time inference network \mathbf{A}_Ψ and the cost-augmented inference network \mathbf{F}_Φ vary when using the new compound objective. The differences between \mathbf{F}_Φ and \mathbf{A}_Ψ are larger than in the baseline configuration, showing that the two are learning complementary functionality. With the stacked parameterization, the cost-augmented network \mathbf{F}_Φ receives as an additional input the gold standard label sequence, which leads to the largest differences as the cost-augmented network can explicitly favor incorrect labels.³

The bottom part of Table 3 shows qualitative differences between the two inference networks. On the POS development set, we count the differences between the predictions of \mathbf{A}_Ψ and \mathbf{F}_Φ when \mathbf{A}_Ψ makes the correct prediction.⁴ \mathbf{F}_Φ tends to output tags that are highly confusable with those output by \mathbf{A}_Ψ . For example, it often outputs proper noun when the gold standard is common noun or vice versa. It also captures the ambiguities among adverbs, adjectives, and prepositions.

Global Energies. The results are shown in Table 4. Adding the backward (b) and word-augmented TLMs (c) improves over using only the forward TLM from Tu and Gimpel (2018). With the global energies, our performance is comparable to several strong results (90.94 of Lample et al., 2016 and 91.37 of Ma and Hovy, 2016). However, it is still lower than the state of the art (Akbik et al., 2018; Devlin et al., 2019), likely due to the lack of contextualized embeddings. In other work, we proposed and evaluated several other high-order energy terms for sequence labeling using our framework (Tu et al., 2020a).

8 Related Work

There are several efforts aimed at stabilizing and improving learning in generative adversarial networks (GANs) (Goodfellow et al., 2014; Salimans et al., 2016; Zhao et al., 2017; Arjovsky et al., 2017). Progress in training GANs has come largely

³We also tried a BiLSTM in the final layer of the stacked parameterization but results were similar to the simpler affine architecture, so we only report results for the latter.

⁴We used the stacked parameterization.

	POS				NER				NER+
	acc. (%)	$ T $	$ I $	speed	F1 (%)	$ T $	$ I $	speed	F1 (%)
BiLSTM	88.8	166K	166K	–	84.9	239K	239K	–	89.3

SPENs with inference networks (Tu and Gimpel, 2018):

margin-rescaled	89.4	333K	166K	–	85.2	479K	239K	–	89.5
perceptron	88.6	333K	166K	–	84.4	479K	239K	–	89.0

SPENs with inference networks, compound objective, CE, no zero truncation (this paper):

separated	89.7	500K	166K	66	85.0	719K	239K	32	89.8
shared	89.8	339K	166K	78	85.6	485K	239K	38	90.1
stacked	89.8	335K	166K	92	85.6	481K	239K	46	90.1

Table 2: Test results for POS and NER. $|T|$ is the number of trained parameters; $|I|$ is the number of parameters needed during inference. Training speeds (examples/second) are shown for joint parameterizations to compare them in terms of efficiency. Best setting (best performance with fewest parameters and fastest training) is in bold.

	POS		NER	
	$A_\Psi - F_\Phi$	$A_\Psi - F_\Phi$	$A_\Psi - F_\Phi$	$A_\Psi - F_\Phi$
margin-rescaled	0.2		0	
compound separated	2.2		0.4	
compound shared	1.9		0.5	
compound stacked	2.6		1.7	

test-time (A_Ψ)	cost-augmented (F_Φ)
common noun	proper noun
proper noun	common noun
common noun	adjective
proper noun	proper noun + possessive
adverb	adjective
preposition	adverb
adverb	preposition
verb	common noun
adjective	verb

Table 3: Top: differences in accuracy/F1 between test-time inference networks A_Ψ and cost-augmented networks F_Φ (on development sets). The “margin-rescaled” row uses a SPEN with the local CE term and without zero truncation, where A_Ψ is obtained by fine-tuning F_Φ as done by Tu and Gimpel (2018). Bottom: most frequent output differences between A_Ψ and F_Φ on the development set.

	NER	NER+	NER++
margin-rescaled	85.2	89.5	90.2
compound, stacked, CE, no truncation	85.6	90.1	90.8
+ global energy GE(a)	85.8	90.2	90.7
+ global energy GE(b)	85.9	90.2	90.8
+ global energy GE(c)	86.3	90.4	91.0

Table 4: NER test F1 scores with global energy terms.

from overcoming learning difficulties by modifying loss functions and optimization, and GANs have become more successful and popular as a result. Notably, Wasserstein GANs (Arjovsky et al., 2017) provided the first convergence measure in GAN training using Wasserstein distance. To compute Wasserstein distance, the discriminator uses weight clipping, which limits network capacity. Weight

clipping was subsequently replaced with a gradient norm constraint (Gulrajani et al., 2017). Miyato et al. (2018) proposed a novel weight normalization technique called spectral normalization. These methods may be applicable to the similar optimization problems solved in learning SPENs. Another direction may be to explore alternative training objectives for SPENs, such as those that use weaker supervision than complete structures (Rooshenas et al., 2018, 2019; Naskar et al., 2020).

9 Conclusions and Future Work

We contributed several strategies to stabilize and improve joint training of SPENs and inference networks. Our use of joint parameterizations mitigates the need for inference network fine-tuning, leads to complementarity in the learned inference networks, and yields improved performance overall. These developments offer promise for SPENs to be more easily applied to a broad range of NLP tasks.

Future work will explore other structured prediction tasks, such as parsing and generation. We have taken initial steps in this direction, considering constituency parsing with the sequence-to-sequence model of Tran et al. (2018). Preliminary experiments are positive,⁵ but significant challenges remain, specifically in defining appropriate inference network architectures to enable efficient learning.

Acknowledgments

We would like to thank the reviewers for insightful comments. This research was supported in part by an Amazon Research Award to K. Gimpel.

⁵On NXT Switchboard (Calhoun et al., 2010), the baseline achieves 82.80 F1 on the development set and the SPEN (stacked parameterization) achieves 83.22. More details are in the appendix.

References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. [Contextual string embeddings for sequence labeling](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*.
- David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on Machine Learning*.
- David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning*.
- Sasha Calhoun, Jean Carletta, Jason M Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The NXT-format Switchboard Corpus: a rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language resources and evaluation*, 44(4):387–419.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Kevin Gimpel, Nathan Schneider, Brendan O’Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. [Part-of-speech tagging for twitter: Annotation, features, and experiments](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47, Portland, Oregon, USA. Association for Computational Linguistics.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. [Generative adversarial nets](#). In *Advances in Neural Information Processing Systems*, pages 2672–2680.
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. [Improved training of Wasserstein GANs](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. [Neural architectures for named entity recognition](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.
- Yann LeCun, Sumit Chopra, Raia Hadsell, Marc’Aurelio Ranzato, and Fu-Jie Huang. 2006. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press.
- Xuezhe Ma and Eduard Hovy. 2016. [End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. [Distributed representations of words and phrases and their compositionality](#). In *Advances in Neural Information Processing Systems*, pages 3111–3119.
- Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.
- Subhajt Naskar, Amirmohammad Rooshenas, Simeng Sun, Mohit Iyyer, and Andrew McCallum. 2020. Energy-based reranking: Improving neural machine translation using energy-based models. *arXiv preprint arXiv:2009.13267*.
- Olutobi Owoputi, Brendan O’Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. [Improved part-of-speech tagging for online conversational text with word clusters](#). In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia. Association for Computational Linguistics.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [Glove: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

- Lev Ratinov and Dan Roth. 2009. **Design challenges and misconceptions in named entity recognition**. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155.
- Amirmohammad Rooshenas, Aishwarya Kamath, and Andrew McCallum. 2018. **Training structured prediction energy networks with indirect supervision**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 130–135.
- Amirmohammad Rooshenas, Dongxu Zhang, Gopal Sharma, and Andrew McCallum. 2019. **Search-guided, lightly-supervised training of structured prediction energy networks**. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13522–13532.
- Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. **Improved techniques for training GANs**. In *Advances in Neural Information Processing Systems*, pages 2234–2242.
- Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. **Max-margin Markov networks**. In *Advances in Neural Information Processing Systems*, pages 25–32.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. **Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition**. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. 2018. **Parsing speech: a neural approach to integrating lexical and acoustic-prosodic information**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 69–81, New Orleans, Louisiana. Association for Computational Linguistics.
- Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*.
- Lifu Tu and Kevin Gimpel. 2018. **Learning approximate inference networks for structured prediction**. In *Proceedings of International Conference on Learning Representations (ICLR)*.
- Lifu Tu and Kevin Gimpel. 2019. **Benchmarking approximate inference methods for neural structured prediction**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3313–3324, Minneapolis, Minnesota. Association for Computational Linguistics.
- Lifu Tu, Kevin Gimpel, and Karen Livescu. 2017. **Learning to embed words in context for syntactic tasks**. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 265–275.
- Lifu Tu, Tianyu Liu, and Kevin Gimpel. 2020a. **An exploration of arbitrary-order sequence labeling via energy-based inference networks**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.
- Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. 2020b. **ENGINE: Energy-based inference networks for non-autoregressive machine translation**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2826, Online. Association for Computational Linguistics.
- Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. 2017. **Energy-based generative adversarial network**. In *Proceedings of International Conference on Learning Representations (ICLR)*.

A Appendices

A.1 Constituency Parsing Experiments

We linearize the constituency parsing outputs, similar to [Tran et al. \(2018\)](#). We use the following equation plus global energy in the form of Eq. (8) as the energy function:

$$E_{\Theta}(\mathbf{x}, \mathbf{y}) = - \left(\sum_{t=1}^T \sum_{j=1}^L y_{t,j} \left(U_j^{\top} b(\mathbf{x}, t) \right) + \sum_{t=1}^T \mathbf{y}_{t-1}^{\top} W \mathbf{y}_t \right)$$

Here, b has a seq2seq-with-attention architecture identical to [Tran et al. \(2018\)](#). In particular, here is the list of implementation decisions.

- We can write $b = g \circ f$ where f (which we call the “feature network”) takes in an input sentence, passes it through the encoder, and passes the encoder output to the decoder feature layer to obtain hidden states; g takes in the hidden states and passes them into the rest of the layers in the decoder. In our experiments, the cost-augmented inference network \mathbf{F}_{Φ} , test-time inference network \mathbf{A}_{Ψ} , and b of the energy function above share the same feature network (defined as f above).

- The feature network (f) component of b is pretrained using the feed-forward local cross-entropy objective. The cost-augmented inference network F_{Φ} and the test-time inference network A_{Ψ} are both pretrained using the feed-forward local cross-entropy objective.

The seq2seq baseline achieves 82.80 F1 on the development set in our replication of [Tran et al. \(2018\)](#). Using a SPEN with our stacked parameterization, we obtain 83.22 F1.