IMPROVING AND STABILIZING DEEP ENERGY-BASED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep energy-based models are powerful, but pose challenges for learning and inference (Belanger & McCallum, 2016). Tu & 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 & 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 & McCallum, 2016; Belanger et al., 2017). Tu & 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 & Gimpel (2018) experimented with several additional terms in the training objectives, finding performance to be dependent on their inclusion.

Also, when using the approach of Tu & 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 & 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 (Tsochantaridis 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 design a compound objective to jointly train both cost-augmented and test-time inference networks along with the energy function. In the context of the new objective, we propose shared parameterizations for the two inference networks that encourage them to capture complementary functionality while reducing the total number of parameters being trained. Quantitative and qualitative analysis shows clear differences in the characteristics of the trained cost-augmented and test-time inference networks. We also present
three methods to streamline and stabilize training that help with both the old and new objectives. 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, and further improvements with global energy terms.

While SPENs have been used for multiple NLP tasks, including multi-label classification [Belanger & McCallum, 2016], part-of-speech tagging [Tu & 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 difficulties have deterred NLP researchers. Our hope is that our methods can enable SPENs to be applied to a larger set of applications, including generation tasks.

2 BACKGROUND

We denote the input space by \( \mathcal{X} \). For an input \( x \in \mathcal{X} \), we denote the structured output space by \( \mathcal{Y}(x) \). The entire space of structured outputs is denoted \( \mathcal{Y} = \bigcup_{x \in \mathcal{X}} \mathcal{Y}(x) \). A SPEN [Belanger & McCallum, 2016] defines an energy function \( E_\Theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \) parameterized by \( \Theta \) that computes a scalar energy for an input/output pair. At test time, for a given input \( x \), prediction is done by choosing the output with lowest energy:

\[
\hat{y} = \arg \min_{y \in \mathcal{Y}(x)} E_\Theta(x, y)
\]

However, solving equation (1) requires combinatorial algorithms because \( \mathcal{Y} \) is a structured, discrete space. This becomes intractable when \( E_\Theta \) does not decompose into a sum over small “parts” of \( y \). Belanger & McCallum [2016] relax this problem by allowing the discrete vector \( y \) to be continuous; \( \mathcal{Y}_R \) denotes the relaxed output space. They solve the relaxed problem by using gradient descent to iteratively minimize the energy with respect to \( y \). The energy function parameters \( \Theta \) 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 & Gimpel [2018] propose an alternative that replaces gradient descent with a neural network trained to do inference, i.e., to mimic the function performed in equation (1). This “inference network” \( A_\Psi : \mathcal{X} \to \mathcal{Y}_R \) is parameterized by \( \Psi \) and trained with the goal that

\[
A_\Psi(x) \approx \arg \min_{y \in \mathcal{Y}_R(x)} E_\Theta(x, y)
\]

When training the energy function parameters \( \Theta \), Tu & Gimpel [2018] replaced the cost-augmented inference step in the structured hinge loss from Belanger & McCallum [2016] with a cost-augmented inference network \( F_\Phi \) and trained the energy function and inference network parameters jointly:

\[
\min_{\Theta} \max_{\Phi} \sum_{(x_i, y_i) \in D} [\Delta(F_\Phi(x_i), y_i) - E_\Theta(x_i, F_\Phi(x_i)) + E_\Theta(x_i, y_i)]_+
\]

where \( D \) is the set of training pairs, \( [h]_+ = \max(0, h) \), and \( \Delta \) is a structured cost function that computes the distance between its two arguments. Tu & Gimpel [2018] alternatively optimized \( \Theta \) and \( \Phi \), which is similar to training in generative adversarial networks [Goodfellow et al., 2014]. 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. We adopt the same learning framework as Tu & Gimpel of jointly learning the energy function and inference network, but we propose a novel objective function that jointly trains a cost-augmented inference network, a test-time inference network, and the energy function.

The energy functions we use for our sequence labeling tasks are taken from Tu & Gimpel [2018] and are described in detail in the appendix.
When we remove 0-truncation (see Sec. 4.1), the objective for the inference network parameters is:

\[
\min_{\Theta} \sum_{(x, y_i) \in \mathcal{D}} \left[ \max\left(0, -E_\Theta(x_i, y_i) - E_\Theta(x_i, y) + E_\Theta(x_i, y_i)\right) \right]_+ + \lambda \left[ \max\left(0, -E_\Theta(x_i, y) + E_\Theta(x_i, y_i)\right) \right]_+
\]

This objective contains two different inference problems, which are also the two inference problems that must be solved in structured max-margin learning, whether during training or at test time. Eq. (1) shows the test-time inference problem. The other one is cost-augmented inference, defined as follows:

\[
\arg\min_{y \in \mathcal{Y}(x)} (E_\Theta(x, y) - \Delta(y, y^*))
\]

where \(y^*\) is the gold standard output. 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. Tu & Gimpel (2018) used the same inference network for solving both problems, which led them to fine-tune the network at test-time with a different objective. We avoid this issue by training two inference networks, \(A_\Psi\) for test-time inference and \(F_\Phi\) for cost-augmented inference:

\[
\min_{\Theta} \max_{\Phi, \Psi} \sum_{(x, y_i) \in \mathcal{D}} \left[ \Delta(F_\Phi(x), y_i) - E_\Theta(x_i, F_\Phi(x)) + E_\Theta(x_i, y_i) \right]_+ + \lambda \left[ -E_\Theta(x_i, A_\Psi(x_i)) + E_\Theta(x_i, y_i) \right]_+
\]

As indicated, this loss can be viewed as the sum of the margin-rescaled and perceptron losses for SPEN training with inference networks. We treat this optimization problem as a min-max game and find a saddle point for the game similar to Tu & Gimpel (2018) and Goodfellow et al. (2014). We alternatively optimize \(\Theta\), \(\Phi\), and \(\Psi\). The objective for the energy function parameters is:

\[
\hat{\Theta} \leftarrow \arg\min_{\Theta} \left[ \Delta(F_\Phi(x), y_i) - E_\Theta(x_i, F_\Phi(x)) + E_\Theta(x_i, y_i) \right]_+ + \lambda \left[ -E_\Theta(x_i, A_\Psi(x_i)) + E_\Theta(x_i, y_i) \right]_+
\]

When we remove 0-truncation (see Sec. 4.1), the objective for the inference network parameters is:

\[
\hat{\Psi}, \hat{\Phi} \leftarrow \arg\max_{\Psi, \Phi} \Delta(F_\Phi(x), y_i) - E_\Theta(x_i, F_\Phi(x)) - \lambda E_\Theta(x_i, A_\Psi(x_i))
\]

### 3 An Objective for Joint Learning of Inference Networks

We now describe our “compound” objective that combines two widely-used losses in structured prediction. We first present it without inference networks:

\[
\min_{\Theta} \sum_{(x, y) \in \mathcal{D}} \left[ \max\left(0, -E_\Theta(x, y) + E_\Theta(x_i, y_i)\right) \right]_+ + \lambda \left[ \max\left(0, -E_\Theta(x_i, y) + E_\Theta(x_i, y_i)\right) \right]_+
\]

Joint Parameterizations. If we were to train independent inference networks \(A_\Psi\) and \(F_\Phi\), this new objective could be much slower than the original approach of Tu & 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:** \(F_\Phi\) and \(A_\Psi\) are two independent networks with their own architectures and parameters as shown in Figure 1(a).
- **shared:** \(F_\Phi\) and \(A_\Psi\) share a “feature” network as shown in Figure 1(b). We consider this option because both \(F_\Phi\) and \(A_\Psi\) are trained to produce output labels with low energy. However \(F_\Phi\) also needs to produce output labels with high cost \(\Delta\) (i.e., far from the gold standard).
When describing them, we will illustrate their impact by showing training trajectories for the Twitter part-of-speech tagging task described in Section 6 and the appendix.

We now discuss several methods that simplify and stabilize training SPENs with inference networks. Without truncation, the inference network can work well even without any stabilization terms. As shown in Figure 3(a) in the appendix, without any stabilization terms and with truncation, the test-time network will barely move from its starting point and learning fails overall. However, removing the truncation, learning stabilizes and becomes less dependent on these additional terms. This is depicted in Figure 1(c). Note that we block the gradient at $A_\psi$ when updating $\Phi$.

For the $q$ function in the stacked option, we use an affine transform 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 $F_\phi(x, y)_t$, we have:

$$F_\phi(x, y)_t = \text{softmax}(W_q[A_\psi(x)_t; y(t)] + b_q)$$

where semicolon (;) is vertical concatenation, $y(t)$ (position $t$ of $y$) is an $L$-dimensional one-hot vector, $A_\psi(x)_t$ is the vector at position $t$ of $A_\psi(x)$, $W_q$ 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 $\triangle(F_\phi(x), y_t)$ term in the combined objective. However, Tu & 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 $y$ 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 described in Section 6 and the appendix.

### 4.1 Removing Zero Truncation

Tu & Gimpel (2018) used the following objective for the cost-augmented inference network (maximizing it with respect to $\Phi$):

$$l_0 = [\triangle(F_\phi, y) - E_\Theta(x, F_\phi) + E_\Theta(x, y)]_+$$

where $[h]_+ = \max(0, h)$. However, there are two potential reasons why $l$ will equal zero and therefore trigger no gradient update. First, $E_\Theta$ (the energy function, corresponding to the discriminator in a GAN) may already be well-trained, and it does a good job separating the gold standard output and the cost-augmented inference network output. Or, it may be the case that the cost-augmented inference network (corresponding to the generator in a GAN) is so poorly trained that the energy of its output is extremely 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; Socha@ntaridis et al., 2004), the cost-augmented inference step is performed exactly (or approximately with reasonable guarantee of effectiveness), ensuring that when $l_0$ is 0, 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 & 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 $\Theta$.

As shown in Figure 3(a) in the appendix, 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 & Gimpel (2018) proposed adding a local cross entropy 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 3(b) shows faster convergence to high accuracy when adding the local CE term. More comparisons are in Section 7.

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 this, we compute the cost-augmented loss $l_1 = \Delta(F_\Phi, y) - E_\Theta(x, F_\Phi)$ and the margin-rescaled loss $l_0 = \frac{1}{k} \| \Delta(F_\Phi, y) - E_\Theta(x, F_\Phi) + E_\Theta(x, y) \|$ averaged over all training pairs $(x, y)$ after each set of I steps. The I steps seek to maximize these terms and the E steps seek to minimize them. Figs. 2(a) and (b) show $l_1$ and $l_0$ during training for different numbers of I steps for every one E step. Fig. 2(c) shows the norm of $\frac{\partial E_\Theta(x, \Lambda_\Phi)}{\partial \Lambda_\Phi}$ after the E steps, and Fig. 2(d) shows the norm of...
\[ \frac{\partial l}{\partial \Phi} \]

\[ \partial \Phi \]

\[ \partial \text{steps} \]

\[ k = 1 \]

\[ \text{inference network lags behind the energy, making the energy parameter updates very small, as shown by the small norms in Fig. 2(c). The inference network gradient norm (Fig. 2(d)) remains high, indicating underfitting. However, increasing } k \text{ too much also harms learning, as evidenced by the "plateau" effect in the } l_1 \text{ curves for } k = 50; \text{this indicates that the energy function is lagging behind the inference network. Using } k = 5 \text{ leads to more of a balance between } l_1 \text{ and } l_0 \text{ and gradient norms that are mostly decreasing during training. We treat } k \text{ as a hyperparameter that is tuned in our experiments.} \]

\[ 5 \text{ GLOBA\textsc{L} E\textsc{NERGIES FOR S\textsc{EQUENCE LABELING}}) \]

In addition to new training strategies, we also experiment with several global energy terms for sequence labeling. Eq. (7) in the appendix shows the base energy. To capture long-distance dependencies, we include global energy (GE) terms in the form of Eq. (8). Tu & Gimpel (2018) pretrained their tag language model (TLM) on a large, automatically-tagged corpus and fixed its parameters when optimizing \( \Theta \). We instead do not pretrain the TLM and learn its parameters when training \( \Theta \).

We also propose new global energy terms. Define \( \tilde{y}_t = h(y_0, \ldots, y_{t-1}) \) 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 \( \tilde{y}_t \)) analogously to the forward TLM. Second, we include words as additional inputs to forward and backward TLMs. We define \( \tilde{y}_t = g(x_0, \ldots, x_{t-1}, y_0, \ldots, y_{t-1}) \) where \( g \) is a forward LSTM TLM. We define the backward version similarly (denoted \( \tilde{y}_t' \)). The global energy is therefore

\[ E_{GE}(y) = -\sum_{t=1}^{T+1} \log(y_t^\top \tilde{y}_t) + \log(y_t^\top \tilde{y}_t') + \gamma \left( \log(y_t^\top \tilde{y}_t') + \log(y_t^\top \tilde{y}_t) \right) \]  

(6)

Here \( \gamma \) is a hyperparameter that is tuned. We experiment with three settings for the global energy: GE(a): forward TLM as in Tu & Gimpel (2018); GE(b): forward and backward TLMs (\( \gamma = 0 \)); GE(c): all four TLMs in Eq. [6]

\[ 6 \text{ E\textsc{XPERIMENTAL SETUP}}) \]

We consider two sequence labeling tasks: Twitter part-of-speech (POS) tagging (Gimpel et al., 2011) and named entity recognition (NER; Tjong Kim Sang & De Meulder, 2003), described in detail in the appendix. We consider three NER modeling configurations. NER uses only words as input and pretrained, fixed 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]. As a baseline, we use a BiLSTM tagger trained only with the local CE term.

\[ 7 \text{ R\textsc{ESULTS AND A\textsc{NALYSIS}}) \]

Effect of Removing Truncation. Table 1 shows results for the margin-rescaled and perceptron losses when considering the removal of zero truncation and its interaction with the use of the local
Table 2: Test set results for Twitter POS tagging and NER. $|T|$ is the number of trained parameters; $|I|$ is the number of parameters needed during the inference procedure. Training speeds (examples/second) are shown for joint parameterizations to compare them in terms of efficiency. Best setting (highest performance with fewest parameters and fastest training) is in boldface.

<table>
<thead>
<tr>
<th></th>
<th>POS</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>acc (%)</td>
<td>$</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>88.8</td>
<td>166K</td>
</tr>
</tbody>
</table>

SPENs with inference networks \cite{Tu2018}:

<table>
<thead>
<tr>
<th></th>
<th>POS</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>acc (%)</td>
<td>$</td>
</tr>
<tr>
<td>margin-rescaled</td>
<td>89.4</td>
<td>333K</td>
</tr>
<tr>
<td>perceptron</td>
<td>88.6</td>
<td>333K</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>POS</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>acc (%)</td>
<td>$</td>
</tr>
<tr>
<td>separated</td>
<td>89.7</td>
<td>500K</td>
</tr>
<tr>
<td>shared</td>
<td>89.8</td>
<td>339K</td>
</tr>
<tr>
<td>stacked</td>
<td>89.8</td>
<td>335K</td>
</tr>
</tbody>
</table>

Table 3: Left: differences in accuracy/F1 between test-time inference networks $A_{\Psi}$ and cost-augmented networks $F_{\Phi}$ (on development sets). The “margin-rescaled” row uses a SPEN with the local CE term and without zero truncation, where $A_{\Psi}$ is obtained by fine-tuning $F_{\Phi}$ as done by Tu & Gimpel \cite{Tu2018}. Right: most frequent output differences between $A_{\Psi}$ and $F_{\Phi}$ on the development set.

<table>
<thead>
<tr>
<th></th>
<th>POS</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>test-time ($A_{\Psi}$)</td>
<td>cost-augmented ($F_{\Phi}$)</td>
</tr>
<tr>
<td>common noun</td>
<td>proper noun</td>
<td></td>
</tr>
<tr>
<td>proper noun</td>
<td>common noun</td>
<td></td>
</tr>
<tr>
<td>common noun</td>
<td>adjective</td>
<td></td>
</tr>
<tr>
<td>proper noun</td>
<td>proper noun + possessive</td>
<td></td>
</tr>
<tr>
<td>adverb</td>
<td>adjective</td>
<td></td>
</tr>
<tr>
<td>preposition</td>
<td>adverb</td>
<td></td>
</tr>
<tr>
<td>verb</td>
<td>common noun</td>
<td></td>
</tr>
<tr>
<td>adjective</td>
<td>verb</td>
<td></td>
</tr>
</tbody>
</table>

CE term. Training fails for both tasks when using zero truncation without the CE term. 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 \cite{Tu2018} is likely due to the fact that truncation was being used.

Effect of Local CE. The local cross entropy (CE) term is useful for both tasks, though it appears more helpful for tagging. 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 $\sim 100$ to $\sim 25$. On Twitter POS Tagging, using the CE term reduces the number of epochs chosen by early stopping from $\sim 150$ to $\sim 60$.

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

The left part of Table 3 shows how the performance of the test-time inference network $A_{\Psi}$ and the cost-augmented inference network $F_{\Phi}$ vary when using the new compound objective. The differences between $F_{\Phi}$ and $A_{\Psi}$ are larger than in the baseline configuration, showing that the two are learning complementary functionality. With the stacked parameterization, the cost-augmented network $F_{\Phi}$
Table 4: NER test F1 scores with global energy terms. ‡: We took the best configuration from NER/NER+ and evaluated it in the NER++ setting.

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.\(^1\)

The right 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 \(A_{\Psi}\) and \(F_{\Phi}\) when \(A_{\Psi}\) makes the correct prediction.\(^2\) The most frequent combinations show that \(F_{\Phi}\) tends to output tags that are highly confusable with those output by \(A_{\Psi}\). For example, it often outputs proper noun when the gold standard is common noun or vice versa. It also captures the noun-verb ambiguity and 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 only using the forward TLM from [Tu & Gimpel, 2018]. With the global energies, our performance is comparable to several strong results (cf. 90.94 of Lample et al., 2016 and 91.37 of Ma & 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.

8 RELATED WORK

Aside from the relevant work discussed already, there are several efforts aimed at stabilizing and improving learning in adversarial frameworks, for example those developed for generative adversarial networks (GANs) (Goodfellow et al., 2014; Salimans et al., 2016; Zhao et al., 2016; Arjovsky et al., 2017). Progress in training GANs has come largely 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).

9 CONCLUSIONS

We contributed several strategies to stabilize and improve joint training of SPENs and inference networks. Our use of joint parameterizations mitigates the need for fine-tuning of inference networks, leads to complementarity in the learned cost-augmented and test-time networks, and yields improved performance overall. These developments offer promise for SPENs to be more easily trained and deployed for 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, experimenting with constituency parsing using the attention-augmented sequence-to-sequence model of Tran et al. (2017). Preliminary experiments are positive.\(^3\)

\(^1\)We 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 here with the affine architecture.

\(^2\)For this analysis we used the BiLSTM version of the stacked parameterization.

\(^3\)When comparing methods on the Switchboard-NXT (Calhoun et al., 2010) dataset, the seq2seq baseline achieves 82.80 F1 on the development set and the SPEN (stacked parameterization) achieves 83.11.
but significant challenges remain, specifically in terms of defining appropriate inference network architectures to enable efficient learning.

REFERENCES


Sasha Calhoun, Jean Carletta, Jason M Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 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, 2010.


A  APPENDIX

A.1  ENERGY FUNCTIONS AND INFERENCE NETWORKS FOR SEQUENCE LABELING

Our experiments in this paper consider sequence labeling tasks, so the input $x$ is a length-$T$ sequence of tokens where $x_t$ denotes the token at position $t$. The output $y$ is a sequence of labels also of length $T$. We use $y_t$ to denote the output label at position $t$, where $y_t$ 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 $y_t$. In the original output space $\hat{y}(x)$, $y_{t,j}$ is 1 for a single $j$ and 0 for all others. In the relaxed output space $\tilde{y}(x)$, $y_{t,j}$ 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 & Gimpel, 2018):

$$E_{\Theta}(x, y) = - \left( \sum_{t=1}^{T} \sum_{j=1}^{L} y_{t,j} (U_j^\top b(x, t)) + \sum_{t=1}^{T} y_{t-1}^\top W y_t \right)$$  \hspace{1cm} (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, $b(x, t) \in \mathbb{R}^d$ denotes the “input feature vector” for position $t$. We define it to be the $d$-dimensional BiLSTM (Hochreiter & Schmidhuber, 1997) hidden vector at $t$. The full set of energy parameters $\Theta$ includes the $U_j$ vectors, $W$, and the parameters of the BiLSTM.

Tu & Gimpel (2018) also added a global energy term that they referred to as a “tag language model” (TLM). We use $h$ to denote an LSTM TLM that takes a sequence of labels as input and returns a distribution over next labels. We define $\bar{y}_t = h(y_0, \ldots, y_{t-1})$. Then, the energy term is:

$$E_{TLM}(y) = - \sum_{t=1}^{T+1} \log \left( y_t^\top \bar{y}_t \right)$$ \hspace{1cm} (8)

where $y_0$ is the start-of-sequence symbol and $y_{T+1}$ is the end-of-sequence symbol. This energy returns the negative log-likelihood under the TLM of the candidate output $y$.

For inference networks, we use architectures similar to those used by Tu & Gimpel (2018). In particular, we choose BiLSTMs as the inference network architectures in our experiments. We also use BiLSTMs for the baselines and both the inference networks and baseline models use the same hidden sizes.

A.2  EXPERIMENTAL SETUP DETAILS

Twitter Part-of-Speech (POS) Tagging. We use the Twitter POS data from Gimpel et al. (2011) and Owoputi et al. (2013) which contains 25 tags. We use 100-dimensional skip-gram (Mikolov et al., 2013) embeddings from Tu et al. (2017). Like Tu & Gimpel (2018), we use a BiLSTM to compute the input feature vector for each position, using hidden dimension of 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 $\Delta$ is L1 distance. We train the inference network using stochastic gradient descent (SGD) with momentum and train the energy parameters using Adam (Kingma & Ba, 2014). We also explore training the inference network using Adam when we do not use the local CE loss.\footnote{We find that Adam works better than SGD when training the inference network without the local cross entropy term.}

Named Entity Recognition (NER). We use the CoNLL 2003 English dataset (Tjong Kim Sang & De Meulder, 2003) and Ma & Hovy (2016) which contains 17 NER labels. We use the BIOES tagging scheme, following previous work (Ratinov & 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 & Ba, 2014) and do early stopping on the development set.

The hyperparameter $k$ (the number of I steps) is tuned over the set $\{1, 2, 5, 10, 50\}$. $\gamma$ is tuned over the set $\{0, 0.5, 1\}$.\footnote{We find that Adam works better than SGD when training the inference network without the local cross entropy term.}
Figure 3: Training trajectories with different settings. The three curves for each setting correspond to different random seeds. \cite{Tu2018} use truncation and CE during training.