Harnessing Deep Neural Networks with Logic Rules

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, Eric P. Xing
School of Computer Science
Carnegie Mellon University
{zhitingh,xuezhem,liu,epxing}@cs.cmu.edu, hovy@cmu.edu

Abstract

Combining deep neural networks with structured logic rules is desirable to harness flexibility and reduce uninterpretability of the neural models. We propose a general framework capable of enhancing various types of neural networks (e.g., CNNs and RNNs) with declarative first-order logic rules. Specifically, we develop an iterative distillation method that transfers the structured information of logic rules into the weights of neural networks. We deploy the framework on a CNN for sentiment analysis, and an RNN for named entity recognition. With a few highly intuitive rules, we obtain substantial improvements and achieve state-of-the-art or comparable results to previous best-performing systems.

1 Introduction

Deep neural networks provide a powerful mechanism for learning patterns from massive data, achieving new levels of performance on image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), machine translation (Bahdanau et al., 2014), playing strategic board games (Silver et al., 2016), and so forth.

Despite the impressive advances, the widely-used DNN methods still have limitations. The high predictive accuracy has heavily relied on large amounts of labeled data; and the purely data-driven learning can lead to uninterpretable and sometimes counter-intuitive results (Szegedy et al., 2014; Nguyen et al., 2015). It is also difficult to encode human intention to guide the models to capture desired patterns, without expensive direct supervision or ad-hoc initialization.

On the other hand, the cognitive process of human beings have indicated that people learn not only from concrete examples (as DNNs do) but also from different forms of general knowledge and rich experiences (Minksy, 1980; Lake et al., 2015). *Logic rules* provide a flexible declarative language for communicating high-level cognition and expressing structured knowledge. It is therefore desirable to integrate logic rules into DNNs, to transfer human intention and domain knowledge to neural models, and regulate the learning process.

In this paper, we present a framework capable of enhancing general types of neural networks, such as convolutional networks (CNNs) and recurrent networks (RNNs), on various tasks, with logic rule knowledge. Combining symbolic representations with neural methods have been considered in different contexts. Neural-symbolic systems (Garcez et al., 2012) construct a network from a given rule set to execute reasoning. To exploit *a priori* knowledge in general neural architectures, recent work augments each raw data instance with useful features (Collobert et al., 2011), while network training, however, is still limited to instance-label supervision and suffers from the same issues mentioned above. Besides, a large variety of structural knowledge cannot be naturally encoded in the feature-label form.

Our framework enables a neural network to learn simultaneously from labeled instances as well as logic rules, through an *iterative rule knowledge distillation* procedure that transfers the structured information encoded in the logic rules into the network parameters. Since the general logic rules are complementary to the specific data labels, a natural "side-product" of the integration is the support for semi-supervised learning where unlabeled data is used to better absorb the logical knowledge. Methodologically, our approach can be seen as a combination of the knowledge distillation (Hinton et al., 2015; Bucilu et al., 2006) and the posterior regularization (PR) method (Ganchev et al., 2010).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2410–2420,
Berlin, Germany, August 7-12, 2016.

In particular, at each iteration we adapt the posterior constraint principle from PR to construct a rule-regularized *teacher*, and train the *student* network of interest to imitate the predictions of the teacher network. We leverage soft logic to support flexible rule encoding.

We apply the proposed framework on both CNN and RNN, and deploy on the task of sentiment analysis (SA) and named entity recognition (NER), respectively. With only a few (one or two) very intuitive rules, both the distilled networks and the joint teacher networks strongly improve over their basic forms (without rules), and achieve better or comparable performance to state-of-the-art models which typically have more parameters and complicated architectures.

To the best of our knowledge, this is the first work to integrate logic rules with general workhorse types of deep neural networks in a principled framework. The encouraging results indicate our method can be potentially useful for incorporating richer types of human knowledge, and improving other application domains.

2 Related Work

Combination of logic rules and neural networks has been considered in different contexts. Neural-symbolic systems (Garcez et al., 2012), such as KBANN (Towell et al., 1990) and CILP++ (França et al., 2014), construct network architectures from given rules to perform reasoning and knowledge acquisition. A related line of research, such as Markov logic networks (Richardson and Domingos, 2006), derives probabilistic graphical models (rather than neural networks) from the rule set.

With the recent success of deep neural networks in a vast variety of application domains, it is increasingly desirable to incorporate structured logic knowledge into general types of networks to harness flexibility and reduce uninterpretability. Recent work that trains on extra features from domain knowledge (Collobert et al., 2011), while producing improved results, does not go beyond the data-label paradigm. Kulkarni et al. (2015) uses a specialized training procedure with careful ordering of training instances to obtain an interpretable neural layer of an image network. Karaletsos et al. (2016) develops a generative model jointly over data-labels and similarity knowledge expressed in triplet format to learn improved disentangled representations.

Though there do exist general frameworks that allow encoding various structured constraints on latent variable models (Ganchev et al., 2010; Zhu et al., 2014; Liang et al., 2009), they either are not directly applicable to the NN case, or could yield inferior performance as in our empirical study. Liang et al. (2008) transfers predictive power of pre-trained structured models to unstructured ones in a pipelined fashion.

Our proposed approach is distinct in that we use an iterative rule distillation process to effectively transfer rich structured knowledge, expressed in the declarative first-order logic language, into parameters of general neural networks. We show that the proposed approach strongly outperforms an extensive array of other either ad-hoc or general integration methods.

3 Method

In this section we present our framework which encapsulates the logical structured knowledge into a neural network. This is achieved by forcing the network to emulate the predictions of a rule-regularized teacher, and evolving both models iteratively throughout training (section 3.2). The process is agnostic to the network architecture, and thus applicable to general types of neural models including CNNs and RNNs. We construct the teacher network in each iteration by adapting the posterior regularization principle in our logical constraint setting (section 3.3), where our formulation provides a closed-form solution. Figure 1 shows an overview of the proposed framework.

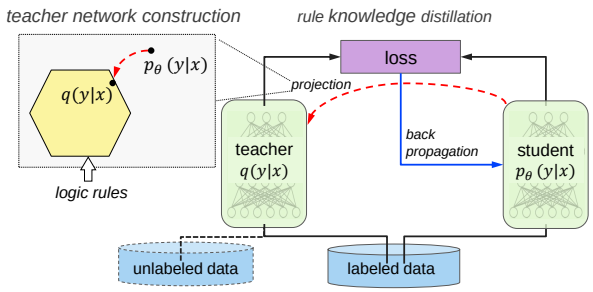

Figure 1: Framework Overview. At each iteration, the teacher network is obtained by projecting the student network to a rule-regularized subspace (red dashed arrow); and the student network is updated to balance between emulating the teacher's output and predicting the true labels (black/blue solid arrows).

3.1 Learning Resources: Instances and Rules

Our approach allows neural networks to learn from both specific examples and general rules. Here we give the settings of these "learning resources".

Assume we have input variable $\boldsymbol{x} \in \mathcal{X}$ and target variable $\boldsymbol{y} \in \mathcal{Y}$. For clarity, we focus on K-way classification, where $\mathcal{Y} = \Delta^K$ is the K-dimensional probability simplex and $\boldsymbol{y} \in \{0,1\}^K \subset \mathcal{Y}$ is a one-hot encoding of the class label. However, our method specification can straightforwardly be applied to other contexts such as regression and sequence learning (e.g., NER tagging, which is a sequence of classification decisions). The training data $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$ is a set of instantiations of $(\boldsymbol{x}, \boldsymbol{y})$.

Further consider a set of first-order logic (FOL) rules with confidences, denoted as $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^L$, where R_l is the lth rule over the input-target space $(\mathcal{X}, \mathcal{Y})$, and $\lambda_l \in [0, \infty]$ is the confidence level with $\lambda_l = \infty$ indicating a hard rule, i.e., all groundings are required to be true (=1). Here a grounding is the logic expression with all variables being instantiated. Given a set of examples $(\boldsymbol{X}, \boldsymbol{Y}) \subset (\mathcal{X}, \mathcal{Y})$ (e.g., a minibatch from $\mathcal{D}$), the set of groundings of R_l are denoted as $\{r_{lg}(\boldsymbol{X}, \boldsymbol{Y})\}_{g=1}^{G_l}$. In practice a rule grounding is typically relevant to only a single or subset of examples, though here we give the most general form on the entire set.

We encode the FOL rules using soft logic (Bach et al., 2015) for flexible encoding and stable optimization. Specifically, soft logic allows continuous truth values from the interval $[0, 1]$ instead of $\{0, 1\}$, and the Boolean logic operators are reformulated as:

$$\begin{aligned} &A \& B = \max\{A + B - 1, 0\} \\ &A \vee B = \min\{A + B, 1\} \\ &A_1 \wedge \cdots \wedge A_N = \textstyle\sum_i A_i / N \\ &\neg A = 1 - A \end{aligned} \tag{1}$$

Here $\&$ and $\wedge$ are two different approximations to logical conjunction (Foulds et al., 2015): $\&$ is useful as a selection operator (e.g., $A \& B = B$ when $A = 1$, and $A \& B = 0$ when $A = 0$), while $\wedge$ is an averaging operator.

3.2 Rule Knowledge Distillation

A neural network defines a conditional probability $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ by using a softmax output layer that produces a K-dimensional soft prediction vector denoted as $\boldsymbol{\sigma}_\theta(\boldsymbol{x})$. The network is parameterized by weights $\boldsymbol{\theta}$. Standard neural network training has been to iteratively update $\boldsymbol{\theta}$ to produce the correct labels of training instances. To integrate the information encoded in the rules, we propose to train the network to also imitate the outputs of a rule-regularized projection of $p_\theta(\boldsymbol{y}|\boldsymbol{x})$, denoted as $q(\boldsymbol{y}|\boldsymbol{x})$, which explicitly includes rule constraints as regularization terms. In each iteration q is constructed by projecting p_θ into a subspace constrained by the rules, and thus has desirable properties. We present the construction in the next section. The prediction behavior of q reveals the information of the regularized subspace and structured rules. Emulating the q outputs serves to transfer this knowledge into p_θ. The new objective is then formulated as a balancing between imitating the soft predictions of q and predicting the true hard labels:

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} &(1 - \pi) \ell(\boldsymbol{y}_n, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)) \\ &+ \pi \ell(\boldsymbol{s}_n^{(t)}, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)), \end{aligned} \tag{2}$$

where ℓ denotes the loss function selected according to specific applications (e.g., the cross entropy loss for classification); $\boldsymbol{s}_n^{(t)}$ is the soft prediction vector of q on $\boldsymbol{x}_n$ at iteration t; and π is the imitation parameter calibrating the relative importance of the two objectives.

A similar imitation procedure has been used in other settings such as model compression (Bucilu et al., 2006; Hinton et al., 2015) where the process is termed *distillation*. Following them we call $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ the "student" and $q(\boldsymbol{y}|\boldsymbol{x})$ the "teacher", which can be intuitively explained in analogous to human education where a teacher is aware of systematic general rules and she instructs students by providing her solutions to particular questions (i.e., the soft predictions). An important difference from previous distillation work, where the teacher is obtained beforehand and the student is trained thereafter, is that our teacher and student are learned simultaneously during training.

Though it is possible to combine a neural network with rule constraints by projecting the network to the rule-regularized subspace after it is fully trained as before with only data-label instances, or by optimizing projected network directly, we found our iterative teacher-student distillation approach provides a much superior performance, as shown in the experiments. Moreover, since p_θ distills the rule information into the

weights $\boldsymbol{\theta}$ instead of relying on explicit rule representations, we can use p_θ for predicting new examples at test time when the rule assessment is expensive or even unavailable (i.e., the *privileged information* setting (Lopez-Paz et al., 2016)) while still enjoying the benefit of integration. Besides, the second loss term in Eq.(2) can be augmented with rich unlabeled data in addition to the labeled examples, which enables *semi-supervised* learning for better absorbing the rule knowledge.

3.3 Teacher Network Construction

We now proceed to construct the teacher network $q(\boldsymbol{y}|\boldsymbol{x})$ at each iteration from $p_\theta(\boldsymbol{y}|\boldsymbol{x})$. The iteration index t is omitted for clarity. We adapt the posterior regularization principle in our logic constraint setting. Our formulation ensures a closed-form solution for q and thus avoids any significant increases in computational overhead.

Recall the set of FOL rules $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^L$. Our goal is to find the optimal q that fits the rules while at the same time staying close to p_θ. For the first property, we apply a commonly-used strategy that imposes the rule constraints on q through an expectation operator. That is, for each rule (indexed by l) and each of its groundings (indexed by g) on $(\boldsymbol{X}, \boldsymbol{Y})$, we expect $\mathbb{E}_{q(\boldsymbol{Y}|\boldsymbol{X})}[r_{lg}(\boldsymbol{X}, \boldsymbol{Y})] = 1$, with confidence λ_l. The constraints define a rule-regularized space of all valid distributions. For the second property, we measure the closeness between q and p_θ with KL-divergence, and wish to minimize it. Combining the two factors together and further allowing slackness for the constraints, we finally get the following optimization problem:

$$\begin{aligned} \min_{q, \boldsymbol{\xi} \geq 0} \ & \mathrm{KL}(q(\boldsymbol{Y}|\boldsymbol{X}) \| p_\theta(\boldsymbol{Y}|\boldsymbol{X})) + C \sum\nolimits_{l, g_l} \xi_{l, g_l} \\ \text{s.t. } & \lambda_l (1 - \mathbb{E}_q[r_{l, g_l}(\boldsymbol{X}, \boldsymbol{Y})]) \leq \xi_{l, g_l} \\ & g_l = 1, \ldots, G_l, \ \ l = 1, \ldots, L, \end{aligned} \tag{3}$$

where $\xi_{l,g_l} \geq 0$ is the slack variable for respective logic constraint; and C is the regularization parameter. The problem can be seen as projecting p_θ into the constrained subspace. The problem is convex and can be efficiently solved in its dual form with closed-form solutions. We provide the detailed derivation in the supplementary materials and directly give the solution here:

$$q^*(\boldsymbol{Y}|\boldsymbol{X}) \propto p_\theta(\boldsymbol{Y}|\boldsymbol{X}) \exp\left\{ -\sum_{l, g_l} C \lambda_l (1 - r_{l, g_l}(\boldsymbol{X}, \boldsymbol{Y})) \right\} \tag{4}$$

Intuitively, a strong rule with large λ_l will lead to low probabilities of predictions that fail to meet the constraints. We discuss the computation of the normalization factor in section 3.4.

Our framework is related to the posterior regularization (PR) method (Ganchev et al., 2010) which places constraints over model posterior in unsupervised setting. In classification, our optimization procedure is analogous to the modified EM algorithm for PR, by using cross-entropy loss in Eq.(2) and evaluating the second loss term on unlabeled data differing from $\mathcal{D}$, so that Eq.(4) corresponds to the E-step and Eq.(2) is analogous to the M-step. This sheds light from another perspective on why our framework would work. However, we found in our experiments (section 5) that to produce strong performance it is crucial to use the same labeled data $\boldsymbol{x}_n$ in the two losses of Eq.(2) so as to form a direct trade-off between imitating soft predictions and predicting correct hard labels.

3.4 Implementations

The procedure of iterative distilling optimization of our framework is summarized in Algorithm 1.

During training we need to compute the soft predictions of q at each iteration, which is straightforward through direct enumeration if the rule constraints in Eq.(4) are factored in the same way as the base neural model p_θ (e.g., the "but"-rule of sentiment classification in section 4.1). If the constraints introduce additional dependencies, e.g., bigram dependency as the transition rule in the NER task (section 4.2), we can use dynamic programming for efficient computation. For higher-order constraints (e.g., the listing rule in NER), we approximate through Gibbs sampling that iteratively samples from $q(\boldsymbol{y}_i|\boldsymbol{y}_{-i}, \boldsymbol{x})$ for each position i. If the constraints span multiple instances, we group the relevant instances in minibatches for joint inference (and randomly break some dependencies when a group is too large). Note that calculating the soft predictions is efficient since only one NN forward pass is required to compute the base distribution $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ (and few more, if needed, for calculating the truth values of relevant rules).

p v.s. q at Test Time At test time we can use either the distilled student network p, or the teacher network q after a final projection. Our empirical results show that both models substantially improve over the base network that is trained with only data-label instances. In general q performs better than p. Particularly, q is more suitable when the logic rules introduce additional dependencies (e.g., span-

Algorithm 1 Harnessing NN with Rules

Input: The training data $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$,
The rule set $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^{L}$,
Parameters: π – imitation parameter
C – regularization strength

1: Initialize neural network parameter $\boldsymbol{\theta}$
2: **repeat**
3: Sample a minibatch $(\boldsymbol{X}, \boldsymbol{Y}) \subset \mathcal{D}$
4: Construct teacher network q with Eq.(4)
5: Transfer knowledge into p_θ by updating $\boldsymbol{\theta}$ with Eq.(2)
6: **until** convergence

Output: Distill student network p_θ and teacher network q

ning over multiple examples), requiring joint inference. In contrast, as mentioned above, p is more lightweight and efficient, and useful when rule evaluation is expensive or impossible at prediction time. Our experiments compare the performance of p and q extensively.

Imitation Strength π The imitation parameter π in Eq.(2) balances between emulating the teacher soft predictions and predicting the true hard labels. Since the teacher network is constructed from p_θ, which, at the beginning of training, would produce low-quality predictions, we thus favor predicting the true labels more at initial stage. As training goes on, we gradually bias towards emulating the teacher predictions to effectively distill the structured knowledge. Specifically, we define $\pi^{(t)} = \min\{\pi_0, 1 - \alpha^t\}$ at iteration $t \geq 0$, where $\alpha \leq 1$ specifies the speed of decay and $\pi_0 < 1$ is a lower bound.

4 Applications

We have presented our framework that is general enough to improve various types of neural networks with rules, and easy to use in that users are allowed to impose their knowledge and intentions through the declarative first-order logic. In this section we illustrate the versatility of our approach by applying it on two workhorse network architectures, i.e., convolutional network and recurrent network, on two representative applications, i.e., *sentence-level sentiment analysis* which is a classification problem, and *named entity recognition* which is a sequence learning problem.

For each task, we first briefly describe the base neural network. Since we are not focusing on tuning network architectures, we largely use the same or similar networks to previous successful neural models. We then design the linguistically-motivated rules to be integrated.

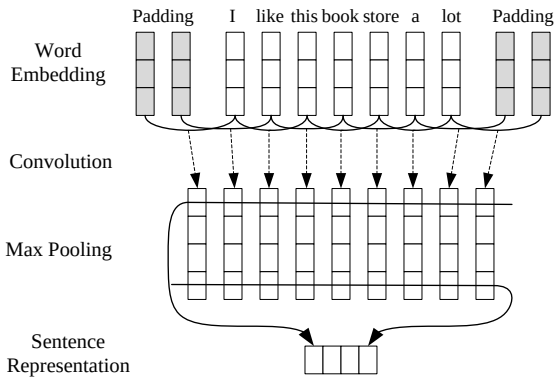

Figure 2: The CNN architecture for sentence-level sentiment analysis. The sentence representation vector is followed by a fully-connected layer with softmax output activation, to output sentiment predictions.

4.1 Sentiment Classification

Sentence-level sentiment analysis is to identify the sentiment (e.g., positive or negative) underlying an individual sentence. The task is crucial for many opinion mining applications. One challenging point of the task is to capture the contrastive sense (e.g., by conjunction "but") within a sentence.

Base Network We use the single-channel convolutional network proposed in (Kim, 2014). The simple model has achieved compelling performance on various sentiment classification benchmarks. The network contains a convolutional layer on top of word vectors of a given sentence, followed by a max-over-time pooling layer and then a fully-connected layer with softmax output activation. A convolution operation is to apply a filter to word windows. Multiple filters with varying window sizes are used to obtain multiple features. Figure 2 shows the network architecture.

Logic Rules One difficulty for the plain neural network is to identify contrastive sense in order to capture the dominant sentiment precisely. The conjunction word "but" is one of the strong indicators for such sentiment changes in a sentence, where the sentiment of clauses following "but" generally dominates. We thus consider sentences S with an "A-but-B" structure, and expect the sentiment of the whole sentence to be consistent with the sentiment of clause B. The logic rule is written as:

$$\begin{aligned} &\text{has-'A-but-B'-structure}(S) \Rightarrow \\ &(\mathbf{1}(y = +) \Rightarrow \boldsymbol{\sigma}_\theta(B)_+ \ \wedge \ \boldsymbol{\sigma}_\theta(B)_+ \Rightarrow \mathbf{1}(y = +)), \end{aligned} \tag{5}$$

where $\mathbf{1}(\cdot)$ is an indicator function that takes 1 when its argument is true, and 0 otherwise; class '+' represents 'positive'; and $\boldsymbol{\sigma}_\theta(B)_+$ is the element of $\boldsymbol{\sigma}_\theta(B)$ for class '+'. By Eq.(1), when S has the 'A-but-B' structure, the truth value of the above logic rule equals to $(1 + \boldsymbol{\sigma}_\theta(B)_+)/2$ when $y = +$, and $(2 - \boldsymbol{\sigma}_\theta(B)_+)/2$ otherwise [1]. Note that here we assume two-way classification (i.e., positive and negative), though it is straightforward to design rules for finer grained sentiment classification.

4.2 Named Entity Recognition

NER is to locate and classify elements in text into entity categories such as "persons" and "organizations". It is an essential first step for downstream language understanding applications. The task assigns to each word a named entity tag in an "X-Y" format where X is one of BIEOS (Beginning, Inside, End, Outside, and Singleton) and Y is the entity category. A valid tag sequence has to follow certain constraints by the definition of the tagging scheme. Besides, text with structures (e.g., lists) within or across sentences can usually expose some consistency patterns.

Base Network The base network has a similar architecture with the bi-directional LSTM recurrent network (called BLSTM-CNN) proposed in (Chiu and Nichols, 2015) for NER which has outperformed most of previous neural models. The model uses a CNN and pre-trained word vectors to capture character- and word-level information, respectively. These features are then fed into a bi-directional RNN with LSTM units for sequence tagging. Compared to (Chiu and Nichols, 2015) we omit the character type and capitalization features, as well as the additive transition matrix in the output layer. Figure 3 shows the network architecture.

Logic Rules The base network largely makes independent tagging decisions at each position, ignoring the constraints on successive labels for a valid tag sequence (e.g., I-ORG cannot follow B-PER). In contrast to recent work (Lample et al., 2016) which adds a conditional random field (CRF) to capture bi-gram dependencies between outputs, we instead apply logic rules which does not introduce extra parameters to learn. An example rule is:

$$\text{equal}(y_{i-1}, \text{I-ORG}) \Rightarrow \neg\, \text{equal}(y_i, \text{B-PER}) \qquad (6)$$

[1] Replacing $\wedge$ with & in Eq.(5) leads to a probably more intuitive rule which takes the value $\boldsymbol{\sigma}_\theta(B)_+$ when $y = +$, and $1 - \boldsymbol{\sigma}_\theta(B)_+$ otherwise.

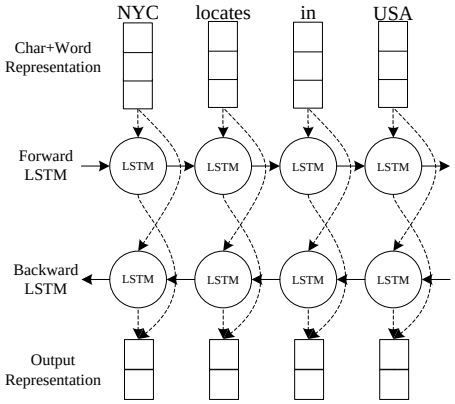

Figure 3: The architecture of the bidirectional LSTM recurrent network for NER. The CNN for extracting character representation is omitted.

The confidence levels are set to ∞ to prevent any violation.

We further leverage the *list* structures within and across sentences of the same documents. Specifically, named entities at corresponding positions in a list are likely to be in the same categories. For instance, in "1. Juventus, 2. Barcelona, 3. ..." we know "Barcelona" must be an organization rather than a location, since its counterpart entity "Juventus" is an organization. We describe our simple procedure for identifying lists and counterparts in the supplementary materials. The logic rule is encoded as:

$$\text{is-counterpart}(X, A) \Rightarrow 1 - \|c(\boldsymbol{e}_y) - c(\boldsymbol{\sigma}_\theta(A))\|_2, \quad (7)$$

where $\boldsymbol{e}_y$ is the one-hot encoding of y (the class prediction of X); $c(\cdot)$ collapses the probability mass on the labels with the same categories into a single probability, yielding a vector with length equaling to the number of categories. We use ℓ_2 distance as a measure for the closeness between predictions of X and its counterpart A. Note that the distance takes value in $[0, 1]$ which is a proper soft truth value. The list rule can span multiple sentences (within the same document). We found the teacher network q that enables explicit joint inference provides much better performance over the distilled student network p (section 5).

5 Experiments

We validate our framework by evaluating its applications of sentiment classification and named entity recognition on a variety of public benchmarks. By integrating the simple yet effective rules with

	Model	SST2	MR	CR
1	CNN (Kim, 2014)	87.2	81.3±0.1	84.3±0.2
2	CNN-Rule-p	88.8	81.6±0.1	85.0±0.3
3	CNN-Rule-q	89.3	**81.7±0.1**	**85.3±0.3**
4	MGNC-CNN (Zhang et al., 2016)	88.4	–	–
5	MVCNN (Yin and Schutze, 2015)	**89.4**	–	–
6	CNN-multichannel (Kim, 2014)	88.1	81.1	85.0
7	Paragraph-Vec (Le and Mikolov, 2014)	87.8	–	–
8	CRF-PR (Yang and Cardie, 2014)	–	–	82.7
9	RNTN (Socher et al., 2013)	85.4	–	–
10	G-Dropout (Wang and Manning, 2013)	–	79.0	82.1

Table 1: Accuracy (%) of Sentiment Classification. Row 1, CNN (Kim, 2014) is the base network corresponding to the "CNN-non-static" model in (Kim, 2014). Rows 2-3 are the networks enhanced by our framework: CNN-Rule-p is the student network and CNN-Rule-q is the teacher network. For MR and CR, we report the average accuracy±one standard deviation using 10-fold cross validation.

the base networks, we obtain substantial improvements on both tasks and achieve state-of-the-art or comparable results to previous best-performing systems. Comparison with a diverse set of other rule integration methods demonstrates the unique effectiveness of our framework. Our approach also shows promising potentials in the semi-supervised learning and sparse data context.

Throughout the experiments we set the regularization parameter to $C = 400$. In sentiment classification we set the imitation parameter to $\pi^{(t)} = 1 - 0.9^t$, while in NER $\pi^{(t)} = \min\{0.9, 1 - 0.9^t\}$ to downplay the noisy listing rule. The confidence levels of rules are set to $\lambda_l = 1$, except for hard constraints whose confidence is ∞. For neural network configuration, we largely followed the reference work, as specified in the following respective sections. All experiments were performed on a Linux machine with eight 4.0GHz CPU cores, one Tesla K40c GPU, and 32GB RAM. We implemented neural networks using Theano [2], a popular deep learning platform.

5.1 Sentiment Classification

5.1.1 Setup

We test our method on a number of commonly used benchmarks, including **1) SST2**, Stanford Sentiment Treebank (Socher et al., 2013) which contains 2 classes (negative and positive), and 6920/872/1821 sentences in the train/dev/test sets respectively. Following (Kim, 2014) we train models on both sentences and phrases since all labels are provided. **2) MR** (Pang and Lee, 2005), a set of 10,662 one-sentence movie reviews with negative or positive sentiment. **3) CR** (Hu and Liu, 2004), customer reviews of various products, containing 2 classes and 3,775 instances. For MR and CR, we use 10-fold cross validation as in previous work. In each of the three datasets, around 15% sentences contains the word "but".

For the base neural network we use the "non-static" version in (Kim, 2014) with the exact same configurations. Specifically, word vectors are initialized using word2vec (Mikolov et al., 2013) and fine-tuned throughout training, and the neural parameters are trained using SGD with the Adadelta update rule (Zeiler, 2012).

5.1.2 Results

Table 1 shows the sentiment classification performance. Rows 1-3 compare the base neural model with the models enhanced by our framework with the "but"-rule (Eq.(5)). We see that our method provides a strong boost on accuracy over all three datasets. The teacher network q further improves over the student network p, though the student network is more widely applicable in certain contexts as discussed in sections 3.2 and 3.4. Rows 4-10 show the accuracy of recent top-performing methods. On the MR and CR datasets, our model outperforms all the baselines. On SST2, MVCNN (Yin and Schutze, 2015) (Row 5) is the only system that shows a slightly better result than ours. Their neural network has combined diverse sets of pre-trained word embeddings (while we use only word2vec) and contained more neural layers and parameters than our model.

To further investigate the effectiveness of our framework in integrating structured rule knowledge, we compare with an extensive array of other

[2] http://deeplearning.net/software/theano

	Model	Accuracy (%)
1	CNN (Kim, 2014)	87.2
2	-but-clause	87.3
3	-ℓ_2-reg	87.5
4	-project	87.9
5	-opt-project	88.3
6	-pipeline	87.9
7	-Rule-p	88.8
8	-Rule-q	**89.3**

Table 2: Performance of different rule integration methods on SST2. 1) CNN is the base network; 2) "-but-clause" takes the clause after "but" as input; 3) "-ℓ_2-reg" imposes a regularization term $\gamma\|\boldsymbol{\sigma}_\theta(S) - \boldsymbol{\sigma}_\theta(Y)\|_2$ to the CNN objective, with the strength γ selected on dev set; 4) "-project" projects the trained base CNN to the rule-regularized subspace with Eq.(3); 5) "-opt-project" directly optimizes the projected CNN; 6) "-pipeline" distills the pre-trained "-opt-project" to a plain CNN; 7-8) "-Rule-p" and "-Rule-q" are our models with p being the distilled student network and q the teacher network. Note that "-but-clause" and "-ℓ_2-reg" are ad-hoc methods applicable specifically to the "but"-rule.

possible integration approaches. Table 2 lists these methods and their performance on the SST2 task. We see that: 1) Although all methods lead to different degrees of improvement, our framework outperforms all other competitors with a large margin. 2) In particular, compared to the pipelined method in Row 6 which is in analogous to the structure compilation work (Liang et al., 2008), our iterative distillation (section 3.2) provides better performance. Another advantage of our method is that we only train one set of neural parameters, as opposed to two separate sets as in the pipelined approach. 3) The distilled student network "-Rule-p" achieves much superior accuracy compared to the base CNN, as well as "-project" and "-opt-project" which explicitly project CNN to the rule-constrained subspace. This validates that our distillation procedure transfers the structured knowledge into the neural parameters effectively. The inferior accuracy of "-opt-project" can be partially attributed to the poor performance of its neural network part which achieves only 85.1% accuracy and leads to inaccurate evaluation of the "but"-rule in Eq.(5).

We next explore the performance of our framework with varying numbers of labeled instances as well as the effect of exploiting unlabeled data. Intuitively, with less labeled examples we expect the general rules would contribute more to the performance, and unlabeled data should help better learn from the rules. This can be a useful property especially when data are sparse and labels are expensive to obtain. Table 3 shows the results. The subsampling is conducted on the sentence level. That is, for instance, in "5%" we first selected 5% training sentences uniformly at random, then trained the models on these sentences as well as their phrases. The results verify our expectations. 1) Rows 1-3 give the accuracy of using only data-label subsets for training. In every setting our methods consistently outperform the base CNN. 2) "-Rule-q" provides higher improvement on 5% data (with margin 2.6%) than on larger data (e.g., 2.3% on 10% data, and 2.0% on 30% data), showing promising potential in the sparse data context. 3) By adding unlabeled instances for semi-supervised learning as in Rows 5-6, we get further improved accuracy. 4) Row 4, "-semi-PR" is the posterior regularization (Ganchev et al., 2010) which imposes the rule constraint through only unlabeled data during training. Our distillation framework consistently provides substantially better results.

	Data size	5%	10%	30%	100%
1	CNN	79.9	81.6	83.6	87.2
2	-Rule-p	81.5	83.2	84.5	88.8
3	-Rule-q	82.5	83.9	85.6	**89.3**
4	-semi-PR	81.5	83.1	84.6	–
5	-semi-Rule-p	81.7	83.3	84.7	–
6	-semi-Rule-q	**82.7**	**84.2**	**85.7**	–

Table 3: Accuracy (%) on SST2 with varying sizes of labeled data and semi-supervised learning. The header row is the percentage of labeled examples for training. Rows 1-3 use only the supervised data. Rows 4-6 use semi-supervised learning where the remaining training data are used as unlabeled examples. For "-semi-PR" we only report its projected solution (in analogous to q) which performs better than the non-projected one (in analogous to p).

5.2 Named Entity Recognition

5.2.1 Setup

We evaluate on the well-established CoNLL-2003 NER benchmark (Tjong Kim Sang and De Meulder, 2003), which contains 14,987/3,466/3,684 sentences and 204,567/51,578/46,666 tokens in train/dev/test sets, respectively. The dataset includes 4 categories, i.e., *person, location, organization*, and *misc*. BIOES tagging scheme is used.

	Model	F1
1	BLSTM	89.55
2	BLSTM-Rule-trans	p: 89.80, q: 91.11
3	BLSTM-Rules	p: 89.93, q: **91.18**
4	NN-lex (Collobert et al., 2011)	89.59
5	S-LSTM (Lample et al., 2016)	90.33
6	BLSTM-lex (Chiu and Nichols, 2015)	90.77
7	BLSTM-CRF$_1$ (Lample et al., 2016)	90.94
8	Joint-NER-EL (Luo et al., 2015)	91.20
9	BLSTM-CRF$_2$ (Ma and Hovy, 2016)	**91.21**

Table 4: Performance of NER on CoNLL-2003. Row 2, BLSTM-Rule-trans imposes the transition rules (Eq.(6)) on the base BLSTM. Row 3, BLSTM-Rules further incorporates the list rule (Eq.(7)). We report the performance of both the student model p and the teacher model q.

Around 1.7% named entities occur in lists.

We use the mostly same configurations for the base BLSTM network as in (Chiu and Nichols, 2015), except that, besides the slight architecture difference (section 4.2), we apply Adadelta for parameter updating. GloVe (Pennington et al., 2014) word vectors are used to initialize word features.

5.2.2 Results

Table 4 presents the performance on the NER task. By incorporating the bi-gram transition rules (Row 2), the joint teacher model q achieves 1.56 improvement in F1 score that outperforms most previous neural based methods (Rows 4-7), including the BLSTM-CRF model (Lample et al., 2016) which applies a conditional random field (CRF) on top of a BLSTM in order to capture the transition patterns and encourage valid sequences. In contrast, our method implements the desired constraints in a more straightforward way by using the declarative logic rule language, and at the same time does not introduce extra model parameters to learn. Further integration of the list rule (Row 3) provides a second boost in performance, achieving an F1 score very close to the best-performing systems including Joint-NER-EL (Luo et al., 2015) (Row 8), a probabilistic graphical model optimizing NER and entity linking jointly with massive external resources, and BLSTM-CRF (Ma and Hovy, 2016), a combination of BLSTM and CRF with more parameters than our rule-enhanced neural networks.

From the table we see that the accuracy gap between the joint teacher model q and the distilled student p is relatively larger than in the sentiment classification task (Table 1). This is because in the NER task we have used logic rules that introduce extra dependencies between adjacent tag positions as well as multiple instances, making the explicit joint inference of q useful for fulfilling these structured constraints.

6 Discussion and Future Work

We have developed a framework which combines deep neural networks with first-order logic rules to allow integrating human knowledge and intentions into the neural models. In particular, we proposed an iterative distillation procedure that transfers the structured information of logic rules into the weights of neural networks. The transferring is done via a teacher network constructed using the posterior regularization principle. Our framework is general and applicable to various types of neural architectures. With a few intuitive rules, our framework significantly improves base networks on sentiment analysis and named entity recognition, demonstrating the practical significance of our approach.

Though we have focused on first-order logic rules, we leveraged soft logic formulation which can be easily extended to general probabilistic models for expressing structured distributions and performing inference and reasoning (Lake et al., 2015). We plan to explore these diverse knowledge representations to guide the DNN learning. The proposed iterative distillation procedure also reveals connections to recent neural autoencoders (Kingma and Welling, 2014; Rezende et al., 2014) where generative models encode probabilistic structures and neural recognition models distill the information through iterative optimization (Rezende et al., 2016; Johnson et al., 2016; Karaletsos et al., 2016).

The encouraging empirical results indicate a strong potential of our approach for improving other application domains such as vision tasks, which we plan to explore in the future. Finally, we also would like to generalize our framework to automatically learn the confidence of different rules, and derive new rules from data.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported by NSF IIS1218282, NSF IIS1447676, Air Force FA8721-05-C-0003, and FA8750-12-2-0342.

References

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss Markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Proc. of ICLR*.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proc. of KDD*, pages 535–541. ACM.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNs. *arXiv preprint arXiv:1511.08308*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

James Foulds, Shachi Kumar, and Lise Getoor. 2015. Latent topic networks: A versatile probabilistic programming framework for topic models. In *Proc. of ICML*, pages 777–786.

Manoel VM França, Gerson Zaverucha, and Artur S dAvila Garcez. 2014. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine learning*, 94(1):81–104.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *JMLR*, 11:2001–2049.

Artur S d'Avila Garcez, Krysia Broda, and Dov M Gabbay. 2012. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of KDD*, pages 168–177. ACM.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. 2016. Structured VAEs: Composing probabilistic graphical models and variational autoencoders. *arXiv preprint arXiv:1603.06277*.

Theofanis Karaletsos, Serge Belongie, Cornell Tech, and Gunnar Rätsch. 2016. Bayesian representation learning with oracle constraints. In *Proc. of ICLR*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proc. of EMNLP*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proc. of ICLR*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*, pages 1097–1105.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. 2015. Deep convolutional inverse graphics network. In *Proc. of NIPS*, pages 2530–2538.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Proc. of ICML*.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proc. of ICML*, pages 592–599. ACM.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proc. of ICML*, pages 641–648. ACM.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2016. Unifying distillation and privileged information. *Prof. of ICLR*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119.

Marvin Minksy. 1980. Learning meaning. *Technical Report AI Lab Memo. Project MAC. MIT*.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. of CVPR*, pages 427–436. IEEE.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of ACL*, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, volume 14, pages 1532–1543.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *Proc. of ICML*.

Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. 2016. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, volume 1631, page 1642. Citeseer.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *Proc. of ICLR*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*, pages 142–147. Association for Computational Linguistics.

Geoffrey G Towell, Jude W Shavlik, and Michiel O Noordewier. 1990. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the eighth National conference on Artificial intelligence*, pages 861–866. Boston, MA.

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *Proc. of ICML*, pages 118–126.

Bishan Yang and Claire Cardie. 2014. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In *Proc. of ACL*, pages 325–335.

Wenpeng Yin and Hinrich Schutze. 2015. Multichannel variable-size convolution for sentence classification. *Proc. of CONLL*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Ye Zhang, Stephen Roller, and Byron Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. *Proc. of NAACL*.

Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent SVMs. *JMLR*, 15(1):1799–1847.