UNIVERSITY OF CALIFORNIA
Los Angeles

A Semantic Loss Function for Deep Learning with Symbolic knowledge

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Jingyi Xu

2018

ABSTRACT OF THE THESIS

A Semantic Loss Function for
Deep Learning with Symbolic knowledge

by

Jingyi Xu
Master of Science in Computer Science
University of California, Los Angeles, 2018
Professor Guy Van den Broeck, Chair

This thesis develops a novel methodology incorporating symbolic knowledge in deep learning. We constructed a logical-constraint semantic loss, using a neural network's output vectors as variables. Semantic loss captures the symbolic knowledge and adds previously-lost information to neural networks. An experimental evaluation shows that it leads the neural network to achieve (near-)state-of-the-art performance on semi-supervised multi-class classification. Moreover, it is applicable to more complex constraints. We present a process, at the end of the thesis, of calculating the semantic loss and its derivatives for complex constraints to show its generality. The work presented in this thesis was published at the International Conference on Machine Learning.

The thesis of Jingyi Xu is approved.

Songchun Zhu

Kaiwei Chang

Guy Van den Broeck, Committee Chair

University of California, Los Angeles

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my labmates Yitao Liang, Tal Friedman and Zilu Zhang. I would also want to thank my advisor Professor Guy Van den Broeck. The work presented in this thesis is a joint effort between me, may labmates and my advisor. Without them, this work would not be possible.

CHAPTER 1

Introduction

Neural Networks have attracted a lot of attention due to its empirical success on vision and natural language processing tasks. However, their performance highly depends on a large amount of data being available. How can we reduce its dependence on data? A number of approaches have been proposed [38, 36, 30, 28]. Among them, one particular research interest is to apply constraints to neural network. This resulted from the observation that humans not only learn from concrete examples, but also infer using general knowledge/rules. [35] seeks to use physics and domain knowledge to direct neural network learning. Logic rules, incorporating symbolic knowledge, is naturally a good candidate in formulating constraints. [14] seeks to use probabilistic soft logic (PSL) [17] to construct the constraints. In the form of boolean logic, we propose a novel loss function to capture the semantic knowledge. Neural networks were previously oblivious to semantic knowledge. Semantic loss is calculated using neural networks' last layer's output vectors. By adding the semantic loss function on top of the existing loss function, it directs neural networks to make better predictions subject to constraints. Our goal here is to augment neural networks with the symbolic knowledge to improve their performance.

This paper is written in the following order. In Section 2, we will list the necessary background knowledge and notation to understand the paper. In Section 3, we will present the definition of semantic loss function. In Section 4, we will present the method of using semantic loss function on existing neural network models. Experimental results on semi-supervised learning will be presented, followed by some discussion on the benefits and limits of semantic loss. In this section, we will also compare our method with several competitive models. In Section 5, we will introduce the method of calculating semantic loss and its derivatives for more complex constraints using existing techniques.[7, 1, 3, 13]

CHAPTER 2

Background and Notation

To formally define semantic loss, we make use of concepts in propositional logic. We write uppercase letters (X,Y) for Boolean variables and lowercase letters (x,y) for their instantiation ($X = 0$ or $X = 1$). Sets of variables are written in bold uppercase ($\mathbf{X}$,$\mathbf{Y}$), and their joint instantiation in bold lowercase ($\mathbf{x}$,$\mathbf{y}$). A literal is a variable (x) or its negation ($\neg x$). A logical sentence (α or β) is constructed in the usual way, from variables and logical connectives ($\wedge$, $\vee$, etc.), and is also called a formula or constraint. A state or world $\mathbf{x}$ is an instantiation to all variables $\mathbf{X}$. A state $\mathbf{x}$ satisfies a sentence α, denoted $\mathbf{x} \models \alpha$, if the sentence evaluates to be true in that world, as defined in the usual way. A sentence α entails another sentence β, denoted $\alpha \models \beta$ if all worlds that satisfy α also satisfy β. A sentence α is logically equivalent to sentence β, denoted $\alpha \equiv \beta$, if both $\alpha \models \beta$ and $\beta \models \alpha$.

The output row vector of a neural net is denoted p. Each value in p represents the probability of an output and falls in $[0, 1]$. We use both softmax and sigmoid units for our output activation functions. The notation for states $\mathbf{x}$ is used to refer the an assignment, the logical sentence enforcing the assignment, or the binary vector capturing that same assignment, as these are all equivalent notions.

First, we examine the exactly-one or *one-hot constraint* capturing the encoding used in multi-class classification. It states that for a set of indicators $\mathbf{X} = \{X_1, \ldots, X_n\}$, one and exactly one of those indicators must be true, with the rest being false. This is enforced through a logical constraint α by conjoining sentences of the form $\neg X_1 \vee \neg X_2$ for all pairs of variables (at most one variable is true), and a single sentence $X_1 \vee \cdots \vee X_n$ (at least one variable is true).

CHAPTER 3

Semantic Loss

In this section, we formally introduce semantic loss by giving the formal definition and our intuition behind it. This definition itself provides all of the necessary mechanics for enforcing constraints, and is sufficient for the understanding of our experiments in Sections 4 and 5.

3.1 Definition

The semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$ is a function of a sentence α in propositional logic, defined over variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, and a vector of probabilities p for the same variables $\mathbf{X}$. Element p_i denotes the predicted probability of variable X_i, and corresponds to a single output of the neural net. For example, the semantic loss between the one-hot constraint from the previous section, and a neural net output vector p, is intended to capture how close the prediction p is to having exactly one output set to true (i.e. 1), and all others set to false (i.e. 0), regardless of which output is correct. The formal definition of this is as follows:

Definition 1 (Semantic Loss)**.** Let p be a vector of probabilities, one for each variable in $\mathbf{X}$, and let α be a sentence over $\mathbf{X}$. The semantic loss between α and p is

$$\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \propto -\log \sum_{\mathbf{x} \models \alpha} \prod_{i : \mathbf{x} \models X_i} \mathsf{p}_i \prod_{i : \mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i).$$

Intuitively, the semantic loss is proportionate to a negative logarithm of the probability of generating a state that satisfies the constraint, when sampling values according to p. Hence, it is the self-information (or "surprise") of obtaining an assignment that satisfies the constraint [16].

CHAPTER 4

Semi-Supervised Classification

The most straightforward constraint that is ubiquitous in multi-class classification is mutual exclusion over one-hot-encoded outputs. That is, for a given example, exactly one class and therefore exactly one binary indicator must be true. How can we formulate this constraint using semantic loss function and use semantic loss to guide the neural network? Our proposed method intends to be generally applicable and compatible with any feedforward neural net. Semantic loss is simply another regularization term that can directly be plugged into an existing loss function. More specifically, with some weight w, the new overall loss becomes

$$\text{existing loss} + w \cdot \text{semantic loss}.$$

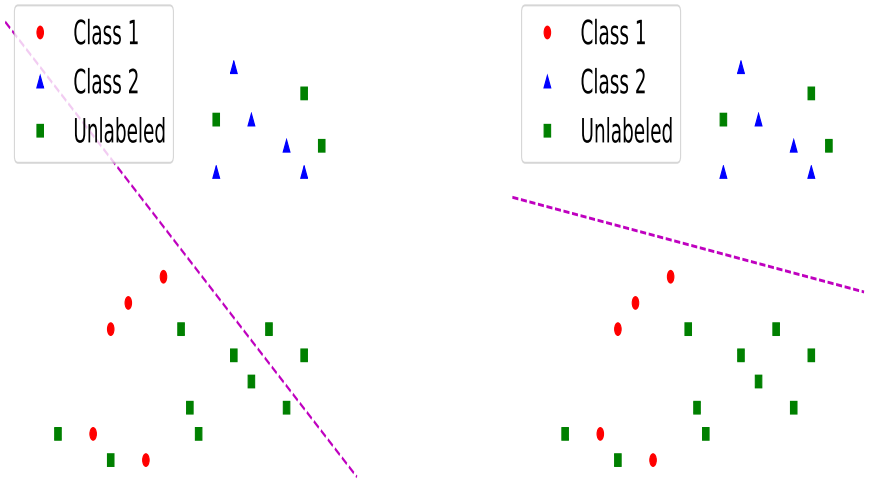
When the constraint over the output space is simple (for example, there is a small number of solutions $\mathbf{x} \models \alpha$), semantic loss can be directly computed using Definition 1. Specifically, for the exactly-one constraint used in n-class classification, semantic loss reduces to

$$\mathrm{L}^{\mathrm{s}}(\text{exactly-one}, \mathsf{p}) \propto -\log \sum_{i=1}^{n} \mathsf{p}_i \prod_{j=1, j\neq i}^{n} (1 - \mathsf{p}_j),$$

where values p_i denote the probability of class i as predicted by the neural net. Semantic loss for the exactly-one constraint is efficient and causes no noticeable computational overhead in our experiments.

In general, for arbitrary constraints α, semantic loss is not efficient to compute using Definition 1, and more advanced automated reasoning is required. Section 5 discusses this issue in more detail.

Previously, we tried our method on multi-class supervised learning tasks and observed negligible improvement on performance. This is due to the neural networks being able to overfit the



(a) Trained w/o semantic loss (b) Trained with semantic loss

Figure 4.1: Binary classification toy example: a linear classifier without and with semantic loss.

labeled training data, resulting in the exact binary indicator of the actual class, that the picture belongs to, to be true. Adding the exactly-one constraint gives no additional information to the neural network. However, neural networks cannot ensure the exactly-one constraint on unlabeled data, without semantic loss.

This section shows why semantic loss naturally qualifies for the semi-supervised learning multi-class tasks.

Illustrative Example To illustrate the benefit of semantic loss in the semi-supervised setting, we begin our discussion with a small toy example. Consider a binary classification task as depicted in Figure 4.1. Ignoring the unlabeled examples, a simple linear classifier learns to distinguish the two classes by separating the labeled examples as in Figure 4.1a. However, the unlabeled examples also carry some information by being in the dataset. They only belong to one particular label and hence should also be as far away from the decision boundary as possible. That is how the semantic loss helps as it gives additional information to neural networks. As we we can see in Figure 4.1b,

the decision boundary is more accurate. Next, we further explore how this idea helps improve the performance on real-world classification tasks.

4.1 Experimental Evaluation

In this section, we will begin by presenting the semi-supervised semantic loss experiment settings and results first. Then we will discuss the benefits and limits of semantic loss by comparing it with several competitive models.

Eexperiment Setting As most semi-supervised learners build on a supervised learner, changing the underlying model significantly affects the semi-supervised learner's performance. For comparison, we add semantic loss to the same base models used in ladder nets [29], which previously achieved state-of-the-art results on semi-supervised MNIST and CIFAR-10 [20] at the time we started our experiment. Recently, we became aware that [23] extended their work to CIFAR-10 and achieved state-of-the-art results [24], surpassing our performance by 5%. In the future work, we intend to apply semantic loss on their architecture to see if we yield an even stronger performance.

Specifically, the MNIST base model is a fully-connected multilayer perceptron (MLP), with layers of size 784-1000-500-250-250-250-10. On CIFAR-10, it is a 10-layer convolutional neural network (CNN) with 3-by-3 padded filters. After every 3 layers, features are subject to a 2-by-2 max-pool layer with strides of 2. Furthermore, we use ReLu [25], batch normalization [15], and Adam optimization [18] with a learning rate of 0.002. We refer to Appendix A and B for a specification of the CNN model and additional details about hyper-parameter tuning.

For all semi-supervised experiments, we use the standard 10,000 held-out test examples provided in the original datasets and randomly pick 10,000 from the standard 60,000 training examples (50,000 for CIFAR-10) as validation set. For values of N that depend on the experiment, we retain N randomly chosen labeled examples from the training set, and remove labels for the rest of training instances. We balance classes in the labeled samples to ensure no particular class is over-represented. Images are preprocessed for standardization and Gaussian noise with a standard

deviation 0.3 is added to every pixel.

Table 4.1: MNIST. Previously reported test accuracies followed by semantic loss results ($\pm$ stddev)

Accuracy % with # of used labels	100	1000	ALL
AtlasRBF [28]	91.9 ($\pm$ 0.95)	96.32 ($\pm$ 0.12)	98.69
Deep Generative [19]	96.67($\pm$ 0.14)	97.60($\pm$ 0.02)	99.04
Virtual Adversarial [23]	97.67	98.64	99.36
Ladder Net [29]	**98.94** ($\pm$0.37)	**99.16** ($\pm$0.08)	99.43 ($\pm$ 0.02)
Baseline: MLP, Gaussian Noise	78.46 ($\pm$1.94)	94.26 ($\pm$0.31)	99.34 ($\pm$0.08)
Baseline: Self-Training	72.55 ($\pm$4.21)	87.43 ($\pm$3.07)	
MLP with Semantic Loss	98.38 ($\pm$0.51)	98.78 ($\pm$0.17)	99.36 ($\pm$0.02)

MNIST The permutation invariant MNIST classification task is commonly used as a test-bed for general semi-supervised learning algorithms. This setting does not use any prior information about the spatial arrangement of the input pixels. Therefore, it excludes many data augmentation techniques that involve geometric distortion of images, as well as convolutional neural networks.

When evaluating on MNIST, we run experiments for 10 epochs, with a batch size of 10 labeled and 10 unlabeled examples. Experiments are repeated 10 times with different random seeds. Table 4.1 has shown that given 100 labeled examples ($N = 100$), MLP with semantic loss gains around 20% improvement over the purely supervised baseline. Comparing to the state of the art, ladder nets slightly outperform semantic loss by 0.5% accuracy. This difference may be an artifact of the excessive tuning of architectures, hyper-parameters and learning rates that the MNIST dataset has been subject to. In the coming experiments, we extend our work to more challenging datasets, in order to provide a clearer comparison with ladder nets.

FASHION The FASHION [37] dataset consists of Zalando's article images, aiming to serve as a more challenging drop-in replacement for MNIST. Arguably, it has not been overused and requires more advanced techniques to achieve good performance. As in the previous experiment, we run our method for 10 epochs, whereas ladder nets need 100 epochs to converge. Again, experiments

Table 4.2: FASHION. Test accuracy comparison between MLP with semantic loss and ladder nets.

Accuracy % with # of used labels	100	500	1000	ALL
Ladder Net [29]	81.46 ($\pm$0.64)	85.18 ($\pm$0.27)	86.48 ($\pm$ 0.15)	90.46
Baseline: MLP, Gaussian Noise	69.45 ($\pm$2.03)	78.12 ($\pm$1.41)	80.94 ($\pm$0.84)	89.87
MLP with Semantic Loss	**86.74** ($\pm$0.71)	**89.49** ($\pm$0.24)	89.67 ($\pm$0.09)	89.81

are repeated 10 times and Table 4.2 reports the classification accuracy and its standard deviation (except for $N =$ all where it is close to 0 and omitted for space).

Experiments show that utilizing semantic loss results in a very large 17% improvement over the baseline when only 100 labels are provided. Moreover, our method compares favorably to ladder nets, except when the setting degrades to be fully supervised. Note that our method already nearly reaches its maximum accuracy with 500 labeled examples, which is only 1% of the training dataset.

Table 4.3: CIFAR. Test accuracy comparison between CNN with Semantic Loss and ladder nets.

Accuracy % with # of used labels	4000	ALL
CNN Baseline in Ladder Net	76.67 ($\pm$ 0.61)	90.73
Ladder Net [29]	79.60 ($\pm$0.47)	
Baseline: CNN, Whitening, Cropping	77.13	90.96
CNN with Semantic Loss	**81.79**	90.92

CIFAR-10 To show the general applicability of semantic loss, we evaluate it on CIFAR-10. This dataset consisting of 32-by-32 RGB images in 10 classes. A simple MLP would not have enough representation power to capture the huge variance across objects within the same class. To cope with this spike in difficulty, we switch our underlying model to a 10-layer CNN as described earlier. We use a batch size of 100 samples of which half are unlabeled. Experiments are run for 100 epochs. However, due to our limited computational resources, we report on a single trial. Note that we make slight modifications to the underlying model used in ladder nets to reproduce similar baseline performance. Please refer to Appendix A for the details of this experimental setup.

As shown in Table 4.3, our method compares favorably to ladder nets. However, due to the slight difference in performance between the supervised base models, a direct comparison would be methodologically flawed. Instead, we compare the net improvements over baselines. In terms of this measure, our method scores a gain of 4.66% whereas ladder nets gain 2.93%.

4.2 Benefits and Limits of Semantic Loss

The experiments so far have demonstrated the competitiveness and general applicability of our proposed method on semi-supervised learning tasks. It surpassed the previous state of the art (ladder nets) on FASHION and CIFAR-10, while being close on MNIST.

The first benefit of Semantic Loss is its simplicity. We only add one more term to the existing loss function, with only one extra hyper-parameter to be tuned. Ladder Net has more than 5 hyper parameters to be tuned. This quality of semantic loss requires less human effort when applying it to other semi-supervised tasks, and adds negligible overhead computational cost to the baseline model MLP. Experiments on MNIST, Fashion MNIST and CIFAR has shown that we achieved similar performance with much simplier model.

The second benefit of Semantic Loss is its recoverability from over-confidence. Table 4.1 compares it with a second baseline—the classic self-training method for semi-supervised learning. After every 1000 iterations, the unlabeled examples that are predicted by the MLP to reach 95% confidence belong to a single class, are assigned a psuedo-label and become labeled data. It is shown that self-training's accuracy is 6% lower than that of the baseline MLP model. The reason we believe is that once an unlabeled example is confidently predicted wrongly and becomes the labeled data, it has a huge irreversible negative effect on the neural network's prediction. However, it is a different case for Semantic Loss Function. Even if semantic loss constraint is fulfilled by confidently assigning a wrong label to an unlabeled instance, semantic loss can later recover from its mistake when the underlying neural network sees more labeled examples.

The above comparison also exposes one limit of semantic loss function as its performance highly depends on the underlying model's performance. Without the underlying predictive power of a strong supervised learning model, we do not expect to see the same benefits we observe here.

The third baseline we attempted to create is a constraint term using Probabilistic Soft Logic (PSL) [17]. PSL uses a designed rule to approximate boolean logic sentence into probabilities. However, different encodings in fuzzy logic will affect the performance. We select 2 reasonable encodings for the exactly-one constraint, which are detailed in Appendix C. Both encodings empirically deviates from 1 by less than 0.02 and has no significant effect on performance. Hence we do not report the results here. People naturally assumes that an approximation method has a lower computational cost than an exact method. In this particular setting, running MLP with either encoding of PSL actually takes longer, as it takes more steps to calculate PSL's derivatives. In the next Chapter, we will see how complex constraints can be constructed easily in semantic loss funciton as the hard boolean logic is a well-studied field. There exists techniques we could use to easily construct semantic loss.

CHAPTER 5

Complex Constraints Construction

The example we used in semi-supervised learning explores a simple constraint. While in the real world, constraints such as a valid path from source to destination in simple path problem and so on, are much more complex. The complexity of encoding more complicated constraints from the definition seems to grow exponentially. A resonable question arised from the previous section is whether semantic loss function is still applicable to complex constraints. In this section, we will present a framework of making semantic loss tractable on highly complex constraints.

We aim to develop a general method for computing both semantic loss and its gradient in a tractable manner. From the definition 1, we can tell that semantic loss is inversely proportional to a well-studied task in propositional logic, called weighted model counting (WMC) [2, 32]. Circuit language can compute WMCs and are amenable to backpropagation [4]. Language and circuit compilation techniques described in [5] can be used to build a Boolean circuit representing semantic loss. We refer to the literature for details of this compilation approach. Both the values and the gradients of semantic loss can be computed in time linear in the size of the circuit [6]. Then semantic loss function can be applied as the additional term in the method described in Section 4.

We present an example here to show the process of calculating semantic loss and its gradients through circuit compilation. Figure 5.1 shows an example Boolean circuit for the exactly-one constraint with 3 variables.

1. We begin with the standard logical encoding for the exactly-one constraint $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \wedge \neg x_3) \wedge (\neg x_2 \wedge \neg x_3)$

2. Then compile above encoding into a circuit that can perform WMC efficiently [2]. Our semantic loss framework can be applied regardless of how the circuit gets compiled. On our

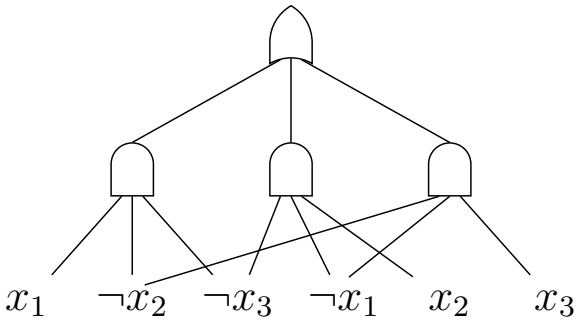


Figure 5.1: A compiled decomposable and deterministic circuit for the exactly-one constraint with 3 variables.

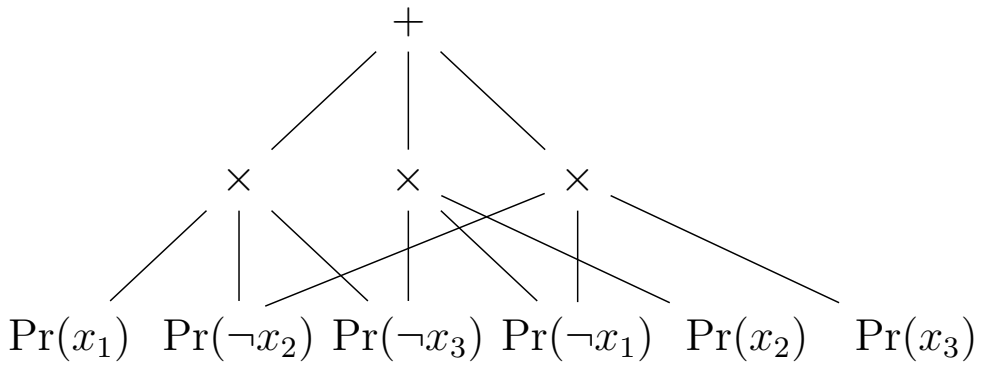


Figure 5.2: The corresponding arithmetic circuit for the exactly-one constraint with 3 variables.

example, following the circuit bottom up, the logical function can be read as $(x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3)$.

3. Once this Boolean circuit is built, we can convert it to an arithmetic circuit, by simply changing `AND` gates into $*$, and `OR` gates into $+$, as shown in Figure 5.2.

4. Now, by pushing the probabilities up through the arithmetic circuit, evaluating the root gives the probability of the logical formula described by the Boolean circuit - this is precisely the exponentiated semantic loss.

5. Finally, we can similarly do another pass down on the circuit to compute partial derivatives [6].

CHAPTER 6

Related Work

Incorporating symbolic meanings into machine learning is always difficult [34]. There has been some work trying to apply logical constraints in deep learning. Particularly, [14] uses probablistic soft logic (PSL) in natural language processing tasks. PSL has shown to be effective in horn clauses, however, as shown in Section 4, PSL is not very efficient when encodings of PSL involve multiple variables.

Various deep learning techniques have been proposed to enforce either arithmetic constraints [27, 21] or logical constraints [31, 14, 8, 35, 22, 9, 10] on the output of a neural network. The common approach is to reduce logical constraints into differentiable arithmetic objectives by replacing logical operators with their fuzzy t-norms and logical implications with simple inequalities. A downside of this fuzzy relaxation is that the logical sentences lose their precise meaning. The learning objective becomes a function of the syntax rather than the semantics (see Section 4). Moreover, these relaxations are often only applied to Horn clauses. One alternative is to encode the logic into a factor graph and perform loopy belief propagation to compute a loss function [26], which is known to have issues in the presence of complex logical constraints [33].

There already exists some methods benefiting from the real-world structure of labels [12, 11], by including entropy penalty terms. In Semi-Supervised experiment setting, both entropy methods and semantic loss make use of subtle information provided by unlabeled data. Semantic loss differs itself from these methods by its general applicability (shown in Section 5). Semantic loss can be extended to construct more complex constraints, while the above entropy methods cannot.

CHAPTER 7

Conclusions

How machine learning can benefit from both data and symbolic meaning has been receiving more attention from research community. The research in this area offers a possible solution in reducing the neural networks' dependence on data. One of the barriers in combining symbolic reasoning with deep learning methods is differentiability of symbolic reasoning. We presented a symbolic meaning method here that is easily differentiable on the outputs of feed-forward neural networks. In addition, the effectiveness of our method has been demonstrated in semi-supervised learning tasks, despite the simplicity of our model. We also showed this method can extend to more complex constraints with a process of calculating semantic loss and its derivatives in linear time of the size of the circuit. In the future, it would be interesting to see how to apply our model when the constraints cannot be directly formulated by the outputs of last-layer of feedforward neural networks.

Appendix A

Specification of the Convolutional Neural Network Model

Table A.1 shows the slight architectural difference between the CNN used in ladder nets and ours. The major difference lies in the choice of ReLu. Note we add standard padded cropping to pre-process images and an additional fully connected layer at the end of the model, neither is used in ladder nets. We only make those slight modification so that the baseline performance reported by [29] can be reproduced.

Table A.1: Specifications of CNNs in Ladder Net and our proposed method.

CNN in Ladder Net	CNN in this paper
Input 32×32 RGB image	
	Resizing to 36 × 36 with padding
	Cropping Back
Whitening	
Contrast Normalization	
Gaussian Noise with std. of 0.3	
3×3 conv. 96 BN LeakyReLU	3×3 conv. 96 BN ReLU
3×3 conv. 96 BN LeakyReLU	3×3 conv. 96 BN ReLU
3×3 conv. 96 BN LeakyReLU	3×3 conv. 96 BN ReLU
2×2 max-pooling stride 2 BN	
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
2×2 max-pooling stride 2 BN	
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
1×1 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
1×1 conv. 10 BN LeakyReLU	1×1 conv. 10 BN ReLu
global meanpool BN	
	fully connected BN
10-way softmax	

Appendix B

Hyper-parameter Tuning Details

Validation sets are used for tuning the weight associated with semantic loss, the only hyper-parameter that causes noticeable difference in performance for our method. For our semi-supervised classification experiments, we perform a grid search over $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ to find the optimal value. Empirically, 0.005 always gives the best or nearly the best results and we report its results on all experiments.

For the FASHION dataset specifically, because MNIST and FASHION share the same image size and structure, methods developed in MNIST should be able to directly perform on FASHION without heavy modifications. Because of this, we use the same hyper-parameters when evaluating our method. However, for the sake of fairness, we subject ladder nets to a small-scale parameter tuning in case its performance is more volatile.

The predictive model we use here is a 3 layer MLP with 25 hidden sigmoid units per layer. It is trained using Adam Optimizer with full data batches [18]. Early stopping with respect to validation loss is used as a regularizer.

Appendix C

Probabilistic Soft Logic Encodings

We here give both encodings on the exactly-one constraint over three x_1, x_2, x_3. The first encoding is:

$$(\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3)$$

The second encoding is:

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$$

Following the pattern presented, readers should be able to extend both encodings to cases whether the number of variables is 10 and beyond.

REFERENCES

[1] Kai-Wei Chang, Rajhans Samdani, and Dan Roth. A constrained latent variable model for coreference resolution. In *EMNLP*, 2013.

[2] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *JAIR*, 172(6):772 – 799, 2008.

[3] Arthur Choi, Guy Van den Broeck, and Adnan Darwiche. Tractable learning for structured probability spaces: A case study in learning preference distributions. In *IJCAI*, 2015.

[4] Adnan Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, May 2003.

[5] Adnan Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, volume 22, page 819, 2011.

[6] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 17:229–264, 2002.

[7] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[8] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. In *EMNLP*, pages 1389–1399, 2016.

[9] Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. *JAIR*, 244:143–165, 2017.

[10] Ivan Donadello, Luciano Serafini, and Artur d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI*, pages 1596–1602, 2017.

[11] Ayse Erkan and Yasemin Altun. Semi-supervised learning via generalized maximum entropy. In *AISTATS*, volume PMLR, pages 209–216, 2010.

[12] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2005.

[13] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[14] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. In *ACL*, 2016.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[16] Douglas Samuel Jones. *Elementary information theory*. Clarendon Press, 1979.

[17] Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *NIPS (Workshop Track)*, 2012.

[18] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*, pages 3581–3589. Curran Associates, Inc., 2014.

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[21] Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations. *arXiv preprint arXiv:1706.02025*, 2017.

[22] Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*, 2017.

[23] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *ICLR*, 2016.

[24] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *ArXiv e-prints*, 2017.

[25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814. Omnipress, 2010.

[26] Jason Naradowsky and Sebastian Riedel. Modeling exclusion with a differentiable factor graph constraint. In *ICML (Workshop Track)*, 2017.

[27] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*, pages 1796–1804, 2015.

[28] Nikolaos Pitelis, Chris Russell, and Lourdes Agapito. Semi-supervised learning using an unsupervised atlas. In *ECML-PKDD*, pages 565–580. Springer, 2014.

[29] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

[30] Scott Reed and Nando De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

[31] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *HLT-NAACL*, 2015.

[32] Tian Sang, Paul Beame, and Henry A Kautz. Performing bayesian inference by weighted model counting. In *AAAI*, volume 5, pages 475–481, 2005.

[33] David Smith and Vibhav Gogate. Loopy belief propagation in the presence of determinism. In *AISTATS*, 2014.

[34] Ashwin Srinivasan, S Muggleton, and RD King. Comparing the use of background knowledge by inductive logic programming systems. In *ILP*, pages 199–230, 1995.

[35] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, pages 2576–2582, 2017.

[36] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

[37] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[38] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.