A SEMANTIC LOSS FUNCTION FOR DEEP LEARNING WITH SYMBOLIC KNOWLEDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper develops a novel methodology for using symbolic knowledge in deep learning. From first principles, we derive a semantic loss function that bridges between neural output vectors and logical constraints. This loss function captures how close the neural network is to satisfying the constraints on its output. An experimental evaluation shows that our semantic loss function effectively guides the learner to achieve (near-)state-of-the-art results on semi-supervised multi-class classification. Moreover, it significantly increases the ability of the neural network to predict structured objects, such as rankings and shortest paths. These discrete concepts are tremendously difficult to learn, and benefit from a tight integration of deep learning and symbolic reasoning methods.

1 INTRODUCTION

The widespread success of representation learning raises the question of which AI tasks are amenable to deep learning, which require classical model-based symbolic reasoning, and whether we can benefit from an integration of both. In recent years, significant effort has gone towards various ways of using representation learning to solve tasks that were previously tackled by symbolic methods. Such efforts include neural computers, Turing machines, and differentiable programming (e.g., Weston et al. (2014); Reed & De Freitas (2015); Graves et al. (2016); Riedel et al. (2016), relational embeddings, deep learning for graph data, and neural theorem proving (e.g., Bordes et al. (2013); Neelakantan et al. (2015); Duvenaud et al. (2015); Niepert et al. (2016), and many more. Other work has sought to augment deep learning with (symbolic) knowledge (e.g., Hu et al. (2016); Stewart & Ermon (2017); Márquez-Neila et al. (2017); Minervini et al. (2017); Wang et al. (2017).

This paper considers learning tasks where we have symbolic knowledge connecting the different outputs of a neural network. This knowledge takes the form of a constraint (or sentence) in Boolean logic. It can be as simple as an exactly-one constraint for one-hot output encodings, or as complex as a structured output prediction constraint for intricate combinatorial objects such as rankings, subgraphs, and paths. Our goal will be for the neural network to learn how to make predictions subject to these constraints, and use the symbolic knowledge to improve its performance.

Most neuro-symbolic approaches aim to simulate or learn symbolic reasoning in an end-to-end deep neural network, or capture symbolic knowledge in a vector-space embedding. This choice is partly motivated by the need for smooth *differentiable* models; adding symbolic reasoning code (e.g., SAT solvers) to a deep learning pipeline destroys this property. Unfortunately, while making reasoning differentiable, the precise logical meaning of our knowledge is often lost. In this paper, we take a distinctly different approach, and tackle the problem of differentiable but sound logical reasoning from first principles. Starting from a set of intuitive axioms, we derive a differentiable *semantic loss* function that captures how well the outputs of a neural network match a given constraint. This function precisely captures the *meaning* of the constraint, and is independent of its *syntax*.

Next, we show how this semantic loss gives *significant practical improvements* in semi-supervised classification. The semantic loss defined over the exactly-one constraint in this setting permits us to obtain a learning signal from vast amounts of unlabeled data. The key idea is that the semantic loss helps us improve how consistently we are able to classify the unlabeled data. This simple addition to the loss function of standard deep learning architectures yields (near-)state-of-the-art performance in semi-supervised classification.

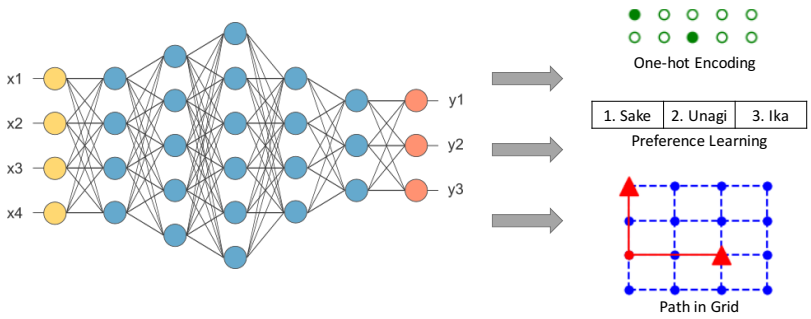

Figure 1: Outputs of a neural net feed into semantic loss functions for constraints representing a one-hot encoding, a total ranking of preferences, and paths in a grid graph

Our final set of experiments study the benefits of the semantic loss function for complex structured output learning tasks, such as preference learning and path prediction in a graph (Chang et al., 2009; Daumé et al., 2009; Choi et al., 2015; Graves et al., 2016). In these scenarios, the task is two-fold: learn both the structure of the output space, and the actual classification function within that space. By capturing the structure of the output space with logical constraints, and minimizing the semantic loss for this constraint during learning, we are able to learn networks that are much more likely to correctly predict structured objects.

2 Background and Notation

To formally define and introduce our semantic loss, we will make use of certain concepts in propositional logic. We write uppercase letters (X,Y) for Boolean or random variables and lowercase letters (x,y) for their instantiation. 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 is constructed in the usual way, from variables and logical connectives ($\wedge$, $\vee$, etc.). A state or world $\mathbf{x}$ is an instantiation to all variables $\mathbf{X}$. A state or world $\mathbf{x}$ satisfies a sentence α, denoted $\mathbf{x} \models \alpha$, if the sentence evaluates to true in the world, as defined in the usual way. We say that a sentence α entails another sentence β, denoted $\alpha \models \beta$ if all worlds that satisfy α also satisfy β. We say that α is logically equivalent to β, denoted $\alpha \equiv \beta$, if $\alpha \models \beta$ and $\beta \models \alpha$.

With p we denote the output row vector of a neural net. Each value in p represents the probability of a label and falls in $[0, 1]$. The most common output units for this in deep learning are softmax and sigmoid units, both of which are used in this paper. We will use the notation for states $\mathbf{x}$ to sometimes refer to the assignment, to the logical sentence enforcing the assignment, or the binary vector capturing that same assignment, as these are all equivalent notions.

This paper studies three different output constraints (visualized in Figure 1) of varying difficulty, and their effects on learning. First, we will examine the one-hot constraint. This can be thought of as a 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 by conjoining sentences of the form $\neg X_1 \vee \neg X_2$ for all pairs of variables, encoding the requirement that at most one variable is true, and a single sentence $X_1 \vee \cdots \vee X_n$, encoding the requirement that at least one variable is true.

Our experiments will also examine the valid simple path constraint. It states for a given source-destination pair and edge indicators, that the edge indicators which are set to true must form a valid simple path from source to destination. Finally, we explore the ordering constraint, which requires that a set of n^2 indicator variables represent a total ordering over n variables, effectively encoding a permutation matrix. For a full description of the path and ordering constraints, we refer to Section 5.

3 SEMANTIC LOSS

Our goal in this section is to find a semantic loss function that bridges the gap between the continuous world of neural networks, and the symbolic world of propositional logic. We do so by first postulating intuitive high-level properties that we seek in such a function, and that illustrate its desired behavior. A second set of postulates establishes a correspondence between constraints and data. Finally, we uniquely define the semantic loss function used throughout this paper.

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 network. For example, the semantic loss between the one-hot constraint described in 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 (that is, 1).

3.1 HIGH-LEVEL PROPERTIES

The first axiom says that there is no loss when the logical constraint α is always true (it is a logical tautology), regardless of the predicted probabilities p.

Axiom 1 (Truth)**.** The semantic loss of a true sentence is zero: $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p}) = 0$.

Next, when enforcing two constraints on disjoint sets of variables, we want to ability to compute the semantic loss of the two constraints separately, and sum the result to obtain their joint semantic loss.

Axiom 2 (Additive Independence)**.** Let α be a sentence over $\mathbf{X}$ with probabilities p. Let β be a sentence over $\mathbf{Y}$ disjoint from $\mathbf{X}$ with probabilities q. The semantic loss between sentence $\alpha \wedge \beta$ and the joint probability vector $[\mathsf{p}\,\mathsf{q}]$ decomposes additively: $\mathrm{L}^{\mathrm{s}}(\alpha \wedge \beta, [\mathsf{p}\,\mathsf{q}]) = \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) + \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{q})$.

A consequence Axioms 1 and 2 is that the probabilities of variables that are not used on the constraint do not affect the semantic loss. Proposition 6 in Appendix A formalizes this intuition.

To maintain logical meaning, we postulate that semantic loss is monotone in the order of implication.

Axiom 3 (Monotonicity)**.** If $\alpha \models \beta$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{p})$ for any vector p.

Intuitively, as we add stricter requirements to our logical constraint, going from β to α and making it harder to satisfy, the semantic loss cannot decrease. For example, when β enforces the output of our network to encode a subtree of a graph, and we tighten that requirement in α to be a path, then the semantic loss cannot decrease. Every path is also a tree and any solution to α is a solution to β.

A first consequence of the monotonicity axiom is that logically equivalent sentences must incur an identical semantic loss for the same probability vector p. Hence, the semantic loss is indeed a semantic property of the logical sentence, and *does not depend on the syntax* of the sentence.

Proposition 1. *If $\alpha \equiv \beta$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{p})$ for any vector p.*

A second consequence is that semantic loss must be non-negative (see Proposition 5 in Appendix A).

3.2 DATA-SENTENCE CORRESPONDENCE

A state $\mathbf{x}$ is equivalently represented as a data instance, as well as a logical constraint that enforces a value for every variable in $\mathbf{X}$. When both the constraint and the predicted vector represent the same state (for example, $X_1 \wedge \neg X_2 \wedge X_3$ vs. $[1\,0\,1]$), there should be no semantic loss.

Axiom 4 (Identity)**.** For any state $\mathbf{x}$, there is zero semantic loss between its representation as a sentence, and its representation as a deterministic vector: $\forall \mathbf{x}, \mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) = 0$.

The axioms above together imply that any vector satisfying the constraint must incur zero loss. For example, when our constraint α requires that the output vector encodes an arbitrary total ranking, and the vector $\mathbf{x}$ correctly represents a single specific total ranking, there is no semantic loss.

Proposition 2 (Satisfaction)**.** *If $\mathbf{x} \models \alpha$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x}) = 0$.*

As a special case, logical literals (x or $\neg x$) constrain a single variable to take on a single value, and thus play a role similar to the labels used in supervised learning. Such constraints require an even tighter correspondence: the semantic loss must act like a classical loss function (i.e., cross entropy).

Axiom 5 (Label-Literal Correspondence)**.** The semantic loss of a single literal is proportionate to the cross-entropy loss for the equivalent data label: $\mathrm{L}^{\mathrm{s}}(x, p) \propto -\log(p)$ and $\mathrm{L}^{\mathrm{s}}(\neg x, p) \propto -\log(1-p)$.

Appendix A states two additional axioms, on the symmetry between values and the symmetry between variables, as well as a Lemma 7 that ties together the multiplicative constants mentioned in Axiom 5. Finally, this allows us to prove the following form of the semantic loss for a state $\mathbf{x}$.

Lemma 3. *For state* $\mathbf{x}$ *and vector* p*, we have* $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathsf{p}) \propto -\sum_{i:\mathbf{x}\models X_i} \log \mathsf{p}_i - \sum_{i:\mathbf{x}\models\neg X_i} \log(1-\mathsf{p}_i)$.

3.3 A General Definition

Lemma 3 falls short of a full definition of semantic loss for arbitrary sentences. One can define additional axioms to pin down L^{s}. For example, the following axiom is highly desirable.

Axiom 6 (Differentiability)**.** For any fixed α, the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$ is monotone in each probability in p, continuous and differentiable.

Appendix A makes the notion of semantic loss precise by stating one additional axiom. It is based on the observation that the state loss of Lemma 3 is proportionate to a log-probability. In particular, it corresponds to the probability of obtaining state $\mathbf{x}$ after independently sampling each X_i with probability p_i. We have now derived the semantic loss function from first principles 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

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

Theorem 4 (Uniqueness)**.** *The semantic loss function in Definition 1 satisfies Axioms 1–9 and is the only function that does so, up to a multiplicative constant.*

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 (Jones, 1979).

4 Semi-Supervised Classification

The most straightforward constraint imposed on most real-world machine learning tasks is mutual exclusion over outputs. That is, for a given example exactly one binary label must be true. This problem setting is often referred to as multi-class supervised classification. Over the past few years, the machine learning community has made great strides in this area, mostly due to the invention of assorted deep learning representations and their associated regularization terms (Krizhevsky et al., 2012; He et al., 2016). Many of these models take large amounts of fully labeled data for granted, and indeed big data is becoming an indispensable condition for discovering accurate representation (Hastie et al., 2009). However, due to the curse of dimensionality, further accuracy improvement tends to require exponentially more data, threatening the sustainability of the current learning paradigm. Therefore, there is a growing interest into utilizing unlabeled data to augment a model's predictive power (Stewart & Ermon, 2017; Bilenko et al., 2004). In this section, we show why semantic loss naturally qualifies for this task.

4.1 Illustration

To illustrate the benefit of semantic loss in the semi-supervised setting, we begin our discussion with a tiny toy example. Suppose we are working on a binary classification task, consisting of around twenty samples with more than half unlabeled (see Figure 2). Ignoring the unlabeled samples, a simple linear classifier learns to distinguish the two classes by maximizing the distance to labeled examples. Though unlabeled, those samples marked by green squares in the plot can be informative, as they must carry some properties which give them a particular label. This is the crux of semantic loss. It asserts that a model must confidently assign a consistent class even to unlabeled data. Indeed, encouraging the model to do so results in a more accurate separation line, as illustrated by the comparison between the left and right side of Figure 2. We further explore this idea and apply it to some real-world classification tasks in the following sections.

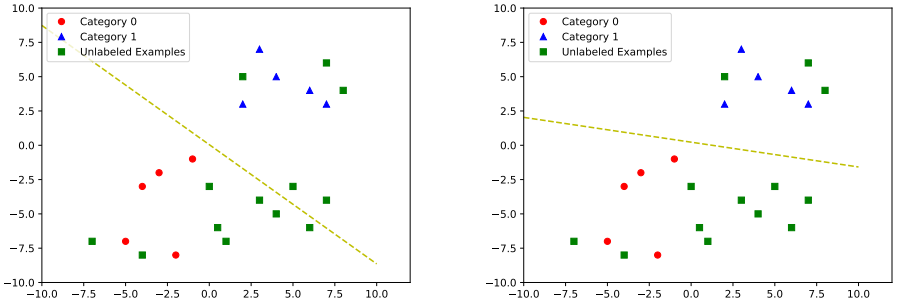

Figure 2: Binary classification example. Left: linear classifier **without** semantic loss. Right: linear classifier **with** semantic loss.

4.2 Algorithm

Our proposed method intends to be generally applicable and compatible with any feedforward neural network. The semantic loss is simply another regularization term that can directly be plugged into an existing loss function. More specifically, the new overall loss becomes

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

with w denoting the associated weight. Note when the constraint over the output space is rather simple (and admits a small number of solutions), the semantic loss can be directly computed using Definition 1. However, once the constraint becomes more complex, a more systematic reasoning approach is required. Section 5 discusses complex constraints in more detail.

4.3 Experiment Details

In this section, we evaluate our work in the semi-supervised setting by comparing it with several competitive models. As most semi-supervised learning methods build on top of some supervised models, it is no surprise that changing the underlying model would significantly affect the final model's representation power. For the sake of comparison, we add semantic loss to the same base models used in ladder nets (Rasmus et al., 2015), which currently achieves state-of-the-art results on MNIST and CIFAR-10 (Krizhevsky & Hinton, 2009).

Specifically, the base model on MNIST is a fully-connected multiple layer 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, ReLu (Nair & Hinton, 2010), normalization (Ioffe & Szegedy, 2015), and Adam optimization (Kingma & Ba, 2015) for weight updates with a learning rate of 0.002 are always used on both. Images are preprocessed with a standardization process and Gaussian noise with a standard deviation of 0.3 is added to every pixel. We refer to Appendix B for more details.

For all experiments in this section, we use the standard 10,000 test samples provided in the original dataset as a held-out test set and randomly split the standard 60,000 training samples (50,000 for CIFAR-10) into a 10,000-sample validation set with the rest as a training set. From the training set, we randomly choose $N = 100, 500, 1000, 4000$, or all, depending on which experiment we run, and remove labels from the rest. We balance classes in the labeled samples to ensure no particular class is over-represented. The validation set is used for tuning the weight associated with semantic loss, the only hyper-parameter that causes noticeable difference in performance for our method. We perform a grid search over $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ to find the optimal. Empirically, 0.005 always gives the best or nearly the best results and we report its results on all experiments.

4.3.1 MNIST

A commonly used test-bed for general semi-supervised learning algorithms is the permutation invariant MNIST classification task. This setting disallows any use of prior information about the

Table 1: MNIST. A collection of previously reported test accuracy followed by MLP with semantic loss. Standard deviation is reported in parenthesis.

Accuracy % with # of used labels	100	1000	ALL
AtlasRBF (Pitelis et al., 2014)	91.9 ($\pm$ 0.95)	96.32 ($\pm$ 0.12)	98.69
Deep Generative (Kingma et al., 2014)	96.67($\pm$ 0.14)	97.60($\pm$ 0.02)	99.04
Virtual Adversarial (Miyato et al., 2015)	97.88	98.68	99.36 ($\pm$0.03)
Ladder Net (Rasmus et al., 2015)	**98.94** ($\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$3.07)	99.34 ($\pm$0.08)
MLP with Semantic Loss	98.38 ($\pm$0.51)	98.78 ($\pm$0.17)	99.36 ($\pm$0.02)

spatial arrangement of the input pixels. In other words, it excludes many data augmentation techniques that involve geometric distortion of images.

When evaluating on MNIST, we run experiments for 10 epochs, with a batch size of 20 where half are labeled. Furthermore, experiments are repeated 10 times with different random seeds to make sure results are consistent. Results are reported in Table 1.

Two baselines are created. The first is purely supervised MLP, which makes no use of unlabelled data. The second is the classic self training method for semi-supervised learning. After every 1000 iterations, the unlabeled examples which have more than 95% likelihood of belonging to some single class, as predicted by the underlying MLP, is assigned with such a psuedo-label. Note when $N = 100$, MLP with semantic loss gains around 20% improvement over the purely supervised learning baseline and the improvement is even larger when comparing with the self-training method. Considering the only change is an additional loss term, such results are very encouraging.

Though indeed falling behind ladder nets, the 0.5% difference may just be an artifact of more careful tuning of hyper-parameters or learning rates. In the coming experiments, we extend our work to other more challenging datasets, so a mor methodological comparison between the two can be conducted. Before that, we want to share a few more thoughts on how semantic loss works. Observe that when $N =$ all, the experiment degrades into the fully supervised setting, in which semantic loss does not provide us with any improvement in classification accuracy. One likely explanation is that by overfitting the training dataset, the MLP inadvertently learns the domain constraint, particularly given that softmax cross entropy is used as the primary loss function. Interpreted as a representation of a categorical distribution, softmax normalizes outputs, implying they are disjoint. In other words, softmax can be viewed as an implicit encoding of mutual exclusion. However, there does not exist a natural way to extend softmax loss to unlabeled samples, whereas semantic loss uses these unlabeled samples to force the underlying classifier to actually make decision across all samples.

4.3.2 FASHION

FASHION (Xiao et al., 2017) is a dataset consisting of Zalando's article images, aiming to serve as a drop-in replacement for MNIST. Arguably, it is a more challenging benchmark, as at least it has not been overused and requires use of more modern machine learning advancements to achieve good performance. As the two 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.

As in the previous experiment, we run our method for 10 epochs, whereas ladder nets need 100 epochs to converge. Again, experiments are repeated 10 times and standard deviation of classification accuracy is reported, except when $N =$all it is close to 0 and omitted to save table space. Results (see Table 2) show utilizing semantic loss results in an enormous 17% improvement over the baseline when only 100 labels are provided. Moreover, our method also compares favorably to ladder nets, except when the setting degrades to be fully supervised. Note our method already nearly reaches its maximum accuracy with 500 provided labels, which is only 1% of the training dataset.

Table 2: FASHION. Test accuracy comparison between MLP with semantic Loss and ladder nets.

Accuracy % with # of used labels	100	500	1000	ALL
Ladder Net (Rasmus et al., 2015)	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

Table 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 (Rasmus et al., 2015)	79.60 ($\pm$0.47)	
Baseline: CNN, Whitening, Cropping	77.13	**90.96**
CNN with Semantic Loss	**81.79**	90.92

4.3.3 CIFAR-10

In order to show the general applicability of our proposed method, we extend our work to CIFAR-10, a dataset consisting of 32-by-32 RGB images in 10 classes, on which a simple MLP would not have enough representation power to capture the huge variance across objects within the same class. Coping with this spike in task difficulty, we switch our underlying model to a 10-layer CNN as mentioned at the beginning of this section.

When evaluating our work on CIFAR-10, we use a batch size of 100 samples of which half are unlabeled. Experiments are run for 100 epochs. However, due to the constraint of computational resources, we can only afford one trial. Note we make slight modifications to the underlying model used in ladder nets to reproduce similar baseline performance. Please refer to Appendix B for details.

As shown in Table 3, our method compares favorably to ladder nets. However, due to the slight difference in terms of performance between two baselines, 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 Net gains 2.93%.

4.4 Discussion

Overall, the experiments outlined so far have demonstrated the competitiveness and general applicability of our proposed method on semi-supervised learning tasks. It achieves state-of-the-art results on two challenging datasets, FASHION and CIFAR-10, while being close on MNIST. Considering the simplicity of our method, such results are encouraging. Another advantage of directly tweaking the loss function is computational efficiency. Without any overhaul of the structure of model itself, we incur almost no additional computational overhead. Conversely, this same property makes our method very sensitive to the underlying model's performance. Without the underlying predictive power of a well-performing supervised learning model, penalties associated with not being confident in decisions would not necessarily result in the same effects we have observed.

5 Learning with Complex Constraints

While much of current machine learning research is focused on problems such as multi class classification, there remain a multitude of difficult problems involving highly constrained output domains. As mentioned in our section on semi-supervised learning, semantic loss shows next to no improvement on the supervised single class classification problem. This led us to seek out more difficult problems to illustrate that semantic loss can also be highly informative in the supervised case, provided the output domain has a sufficiently complex space. As a result of semantic loss being defined by a Boolean formula, it can be used on any output domain which can be fully described in this manner, regardless of complexity. Here, we will develop a framework for tractable semantic loss on highly complex constraints, and evaluate it on some difficult examples.

Table 4: Grid shortest path. Exact and per label accuracy % comparison.

Test accuracy %	Exact	Each
5-layer MLP	5.62	**85.91**
With semantic loss	**28.51**	83.14

Table 5: Peference prediction. Exact and per label accuracy % comparison.

Test accuracy %	Exact	Each
3-layer MLP	0.01	**75.74**
Wit semantic loss	**13.59**	72.43

5.1 A TRACTABLE SEMANTIC LOSS

Our goal here is to develop a method for computing both the semantic loss and its gradients in a tractable manner. Examining Definition 1 of semantic loss, we see that the right-hand side is a well-known automated reasoning task called weighted model counting (WMC) (Chavira & Darwiche, 2008; Sang et al., 2005). A key property of WMC is that its partial derivatives can be computed in terms of other, slightly modified WMCs. Furthermore, we know of circuit languages that compute weighted model counts, and that are amenable to backpropagation (Darwiche, 2003). We use the language and circuit compilation techinques described in Darwiche (2011) to build a Boolean circuit representing our semantic loss. Due to certain properties of this circuit form, we can use it to compute both the values and the gradients of the semantic loss in time linear in the size of the circuit (Darwiche & Marquis, 2002). Once we have constructed this function, we can add it to our standard loss function as described in Section 4.2.

5.2 EXPERIMENTAL EVALUATION

Our ambition when evaluating semantic loss's performance on complex constraints is not to achieve state-of-the-art performance on any particular problem, but rather to highlight its effect. To this end, we evaluate our method on problems with difficult output space, where the model could no longer fit directly from data, and purposefully use simple MLPs for evaluation.

5.2.1 GRIDS

We begin with a classic algorithmic problem, finding the shortest path in a graph. Specifically, we use a 4-by-4 grid $G = (V, E)$ with uniform edge weights. We randomly remove some edges for each example to increase difficulty. Formally, our input is a binary vector of length $|V| + |E|$, with the first $|V|$ variables indicating sources and destinations, and the next $|E|$ which edges are removed. Similarly, the label for a given example is a binary vector of length $|E|$ indicating which edges are in the shortest path. Finally, we require through our constraint that the output form a valid simple path between the desired source and destination.

To compile the valid simple path constraint, we use the method given by Nishino et al. (2017) to generate pairwise simple paths, and logically merge them to enforce the correct source and destination. For more details on the constraint as well as the data generation procedure, see Appendix C.

To evaluate, we use our generated dataset of 1600 examples, with a 60/20/20 training/validation/testing split. The only hyper parameter that needed to be tuned was the weight given to semantic loss, which after trying $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ was selected to be 0.5 based on validation results. Table 4 compares test accuracy between a 5-layer MLP baseline, and the same model augmented with semantic loss. Examining both types of accuracy in this table[1] illustrates the effect of semantic loss. In the case of individual labels, semantic loss has little effect, and in fact slightly reduces the accuracy as it combats the standard sigmoid cross entropy. In regards to getting paths exactly right, however, the semantic loss has a huge effect in guiding the network to jointly learn true paths, rather than optimizing each binary output individually.

5.2.2 PREFERENCE LEARNING

The next problem we examine is that of predicting a complete order of preferences. That is, for a given set of user features, we would like to predict how the user would rank their preference over a

[1]We report two different accuracies: "Exact" indicates the percentage of examples for which the classifier gets the entire configuration right, while the "Each " metric measures the percentage of individually correct binary labels, which as a whole may not constitute a valid path at all.

fixed set of items. We encode a preference ordering over n items as a flattened binary matrix $\{X_{ij}\}$, where for each $i, j \in \{1, \ldots, n\}$, it denotes that item i is at position j (Choi et al., 2015). Clearly, not all configurations of outputs correspond to a valid ordering.

For data, we use preference rankings over 10 types of sushi for 5000 individuals, taken from PREFLIB (Mattei & Walsh, 2013). We take the ordering over 6 types of sushi as input features to predict the ordering over the remaining 4 types, with splits identical to those in Shen et al. (2017).

We again split the data 60/20/20 into train/test/split, and employ a 3 layer MLP as our baseline. We initially chose the semantic loss weight from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ to be 0.1 based on validation, and then further tuned the weight to 0.25. Table 5 compares the baseline to the same MLP augmented with semantic loss for valid permutations. Again, we see that semantic loss has a marginal effect on the per label accuracy, but massively improves the network's ability to predict correct orderings.

6 CONCLUSIONS

Both reasoning and semi-supervised learning are often identified as key challenges for deep learning going forward. In this paper, we developed a principled way of combining automated reasoning for propositional logic with existing deep learning architectures. Moreover, we showed that our semantic loss function provides significant benefits during semi-supervised classification, as well as deep structured prediction for highly complex output spaces.

REFERENCES

Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 11. ACM, 2004.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pp. 2787–2795, 2013.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Yuancheng Tu. Unsupervised constraint driven learning for transliteration discovery. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 299–307. Association for Computational Linguistics, 2009.

Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6):772 – 799, 2008. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2007.11.002.

Arthur Choi, Guy Van den Broeck, and Adnan Darwiche. Tractable learning for structured probability spaces: A case study in learning preference distributions. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

Adnan Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, May 2003. ISSN 0004-5411. doi: 10.1145/765568.765570.

Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, pp. 819, 2011.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

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

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

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.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of supervised learning. In *The elements of statistical learning*, pp. 9–41. Springer, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

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

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. 2015.

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 (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3581–3589. Curran Associates, Inc., 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classifcation with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

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

Nicholas Mattei and Toby Walsh. Preflib: A library of preference data HTTP://PREFLIB.ORG. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT 2013)*, Lecture Notes in Artificial Intelligence. Springer, 2013.

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

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing by virtual adversarial examples. *stat*, 1050:2, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 807–814. Omnipress, 2010.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*, 2015.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pp. 2014–2023, 2016.

Masaaki Nishino, Norihito Yasuda, Shin-ichi Minato, and Masaaki Nagata. Compiling graph substructures into sentential decision diagrams. In *Proceedings of the Thirty-First Conference on Artificial Intelligence (AAAI)*, 2017.

Nikolaos Pitelis, Chris Russell, and Lourdes Agapito. Semi-supervised learning using an unsupervised atlas. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 565–580. Springer, 2014.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pp. 3546–3554, 2015.

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

Sebastian Riedel, Matko Bosnjak, and Tim Rocktäschel. Programming with a differentiable forth interpreter. *CoRR, abs/1605.06640*, 2016.

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

Yujia Shen, Arthur Choi, and Adnan Darwiche. A tractable probabilistic model for subset selection. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

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

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. *arXiv preprint arXiv:1709.09994*, 2017.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

A AXIOMATIZATION OF SEMANTIC LOSS: DETAILS

This appendix provides further details on our axiomatization of the semantic loss.

Proposition 5 (Non-Negativity)**.** *Semantic loss is non-negative.*

Proof. Because $\alpha \models \mathit{true}$ for all α, the monotonicity axiom implies that $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p})$. By the truth axiom, $\mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p}) = 0$, and therefore $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq 0$ for all choices of α and p. □

Proposition 6 (Locality)**.** *Let α be a sentence over $\mathbf{X}$ with probabilities p. For any $\mathbf{Y}$ disjoint from $\mathbf{X}$ with probabilities q, the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, [\mathsf{p}\,\mathsf{q}]) = \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$.*

Proof. Follows from the additive independence and truth axioms. Set $\beta = \mathit{true}$ in the additive independence axiom, and observe that this sets $\mathrm{L}^{\mathrm{s}}(\beta, \mathsf{q}) = 1$ because of the truth axiom. □

Proof of Proposition 2. The monotonicity axiom specializes to say that if $\mathbf{x} \models \alpha$, we have that $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$. By choosing p to be $\mathbf{x}$, this implies $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x})$. From the identity axiom, $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) = 0$, and therefore $0 \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x})$. Proposition 5 bounds the loss from below as $\mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x}) \geq 0$. □

Axiom 7 (Value Symmetry)**.** For all p and α, we have that $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\bar{\alpha}, 1 - \mathsf{p})$ where $\bar{\alpha}$ replaces every variable in α by its negation.

Axiom 8 (Variable Symmetry)**.** Let α be a sentence over $\mathbf{X}$ with probabilities p. Let π be a permutation of the variables $\mathbf{X}$, let $\pi(\alpha)$ be the sentence obtained by replacing variables x by $\pi(x)$, and let $\pi(\mathsf{p})$ be the corresponding permuted vector of probabilities. Then, $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\pi(\alpha), \pi(\mathsf{p}))$.

The value and variable symmetry axioms together imply the equality of the multiplicative constants in the label-literal duality axiom for all literals.

Lemma 7. *There exists a single constant K such that $\mathrm{L}^{\mathrm{s}}(X, p) = -K \log(p)$ and $\mathrm{L}^{\mathrm{s}}(\neg X, p) = -K \log(1 - p)$ for any literal x.*

Proof. Value symmetry implies that $\mathrm{L}^{\mathrm{s}}(X_i, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\neg X_i, 1 - \mathsf{p})$. Using label-literal correspondence, this implies $K_1 \log(p_i) = K_2 \log(1-(1-p_i))$ for the multiplicative constants K_1 and K_2 that are left unspecified by that axiom. This implies that the constants are identical. A similar argument based on variable symmetry proves equality between the multiplicative constants for different i. □

Proof of Lemma 3. A state $\mathbf{x}$ is a conjunction of independent literals, and therefore subject to the additive independence axiom. Each literal's loss in this sum is defined by Lemma 7. □

The following and final axiom requires that the semantic loss is proportionate to the logarithm of a function that is additive for mutually exclusive sentences.

Axiom 9 (Exponential Additivity)**.** Let α and β be mutually exclusive sentences (i.e., $\alpha \wedge \beta$ is unsatisfiable), and let $f^s(K, \alpha, \mathsf{p}) = K^{-\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})}$. Then, there exists a positive constant K such that $f^s(K, \alpha \vee \beta, \mathsf{p}) = f^s(K, \alpha, \mathsf{p}) + f^s(K, \beta, \mathsf{p})$.

Proof of Theorem 4. The truth axiom states that $\forall \mathsf{p}, f^s(K, \mathit{true}, \mathsf{p}) = 1$ for all positive constants K. This is the first Kolmogorov axiom of probability. The second Kolmogorov axiom for $f^s(K, ., \mathsf{p})$ follows from the additive independence axiom of semantic loss. The third Kolmogorov axiom (for the finite discrete case) is given by the exponential additivity axiom of semantic loss. Hence, $f^s(K, ., \mathsf{p})$ is a probability distribution for some choice of K, which implies the definition up to a multiplicative constant. □

Table 6: Specifications of CNNs in Ladder Net and our proposed method.

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

B Specification of the Convolutional Neural Network Model

Table 6 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 preprocess 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 (Rasmus et al., 2015) can be reproduced.

C Specification of Complex Constraint Models

C.1 Grids

To compile our grid constraint, we first use Nishino et al. (2017) to generate a constraint for each source destination pair. Then, we conjoin each of these with indicators specifying which source and destination pair must be used, and finally we disjoin all of these together to form our constraint.

To generate the data, we begin by randomly removing one third of edges. We then filter out connected components with fewer than 5 nodes to reduce degenerate cases, and proceed with randomly selecting pairs of point to create data points.

The predictive model we employ as our baseline is a 5 layer MLP with 50 hidden sigmoid units per layer. It is trained using Adam Optimizer, with full data batches (Kingma & Ba, 2015). Early stopping with respect to validation loss is used as a regularizer.

C.2 Preference Learning

We split each user's ordering into their ordering over sushis 1,2,3,5,7,8, which we use as the features, and their ordering over 4,6,9,10 which are the labels we predict. The constraint is compiled directly from logic, as this can be done in a straightforward manner for an n-item ordering.

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 (Kingma & Ba, 2015). Early stopping with respect to validation loss is used as a regularizer.