Backpropagating through Markov Logic Networks

Patrick Betz[1], Mathias Niepert[2], Pasquale Minervini[3] and Heiner Stuckenschmidt[1]

[1]*University of Mannheim*
[2]*NEC Laboratories Europe*
[3]*UCL Centre for Artificial Intelligence, University College London*

Abstract

We integrate Markov Logic networks with deep learning architectures operating on high-dimensional and noisy feature inputs. Instead of relaxing the discrete components into smooth functions, we propose an approach that allows us to backpropagate through standard statistical relational learning components using perturbation-based differentiation. The resulting hybrid models are shown to outperform models solely relying on deep learning based function fitting. We find that using noise perturbations is required to allow the proposed hybrid models to robustly learn from the training data.

Keywords

Machine Learning, Reasoning, Markov Logic, Discrete-continuous learning,

1. Introduction

Connectionist models in artificial intelligence (AI) have achieved tremendous success in the last decade. Shortcomings in regard to explainability and the lack of reasoning abilities, however, raise concerns about robustness and generalizability [1, 2]. Neural-symbolic integration [3] seeks to overcome these issues by bridging the gap between deep learning and classical logic-based approaches with the ultimate goal of *combining the best of both worlds*.

Markov Logic Networks (MLNs) [4] have been applied to numerous tasks such as sentiment analysis [5], activity recognition [6], ontologies and the semantic web [7, 8]. However, their impact is still lacking in NLP and related areas due to the success of end-to-end learning and powerful language models such as long short-term memory networks [9] and self-attention based models [10, 11]. In this work, we explore the integration of MLNs in end-to-end machine learning by utilizing recent advances in methods for backpropagating through discrete probability distributions [12, 13]. We focus on settings where inputs are high-dimensional and noisy but logical rules can be utilized to predict dataset specific outcomes. By connecting the discrete inference mechanism of MLNs with continuous learning we hope to enrich their possible application scenarios.

In particular, we demonstrate that the perceptron rule [14, 15], which was developed over a decade ago, may serve as the conceptual entry point for a hybrid model which is solely based on a neural backbone, a maximum a posteriori (MAP) solver, and an implementation of the backward

NeSy'20/21 @ IJCLR: 15th International Workshop on Neural-Symbolic Learning and Reasoning, 25-27 October 2021
patrick@informatik.uni-mannheim.de (P. Betz); mathias.niepert@neclab.eu (M. Niepert); p.minervini@ucl.ac.uk (P. Minervini); heiner@informatik.uni-mannheim.de (H. Stuckenschmidt)

pass for the MLN. We further show that more general model classes can be achieved by utilizing perturbation-based implicit differentiation [13, 16]. We then investigate how different gradient formulations [12, 13, 15] for this simplistic neural-symbolic model behave in the context of datasets based on text inputs which are governed by a set of global regularities. We present experimental results of the hybrid model on a schema dataset constructed for Description Logic reasoning and on the CLUTRR benchmark suite [17]. We find that using noise-perturbations is necessary to achieve proper training behavior within the scope of our experiments which also provides new evidence in regard to recent methods of backpropagating through combinatorial optimization problems [12, 13].

2. Related Work

Multiple directions exist in the literature of integrating classical logic into deep learning. *Semantic based regularization* augments neural training objectives with constraints based on first-order programs [18, 19, 20, 21]. In general, these approaches adapt the learning process of neural models and no explicit reasoning is performed when models are applied to unseen data.

Differentiable interpreters, on the other hand, combine learning and reasoning by using fuzzy-logic, *i.e.*, relaxing the discreteness of the logic side. In Logic Tensor Networks [22], constants are represented as tensors and predicates are parameterized functions which take the constants and output soft truth values. While [23] provides a differentiable formulation of forward chaining, Neural Theorem Provers [24] relax the backward chaining algorithm in the Datalog subset of Prolog. Further works extend them towards better scalability [25, 26] and query adaptive rule learning [27]. In TensorLog [28], a deductive database syntax is proposed which allows for efficient inference performed by using message passing compiled into a differentiable function. Relational neural machines [29] unify the neural and the logic side by defining general exponential family distributions assigned with potentials for the two components. Inference is performed by using fuzzy-logic and gradients are based on average formula satisfaction, hence, requiring to modify and track internals of the logical side. The aforementioned works focus on advanced procedures of making the inference process differentiable while the goal of our work is to explore if the implementation of one backward pass suffices to guarantee learning when reasoning over continuous inputs is required.

In Neural Markov Logic Networks [30], first-order logic rules are replaced with neural networks and inference is performed using Gibbs sampling on the resulting exponential distribution. The focus of our work is different as for our joint model the MLN is not modified and the model operates on continuous inputs when a set of first-order rules is provided whereas in [30] the input is structured data (knowledge graphs). Similar to our work, logic and neural components operate independently in DeepProbLog [31] where aProbLog [32] is used as the underlying logic formalism. The type of inference in DeepProbLog is query-based whereas in our approach interpretation-based inference is utilized by obtaining the `MAP` state of the MLN which allows to predict many atoms simultaneously as we will describe in Section 5.

3. Markov Logic Networks

MLNs [4] combine undirected probabilistic graphical models, parameterized as log-linear models, with function free first-order logic. We briefly review the concepts required throughout the remaining sections. Let $a := p(t_1, ..., t_{m_a})$ be an atom. In particular, $t_1, ..., t_{m_a}$ is a collection of terms where a term is either a constant or a variable and p is a predicate of arity $m_a \in \mathbb{N}^+$. When terms are variables, they are universally quantified. An atom containing only constants is a ground atom. A clause or rule is the disjunction of atoms or negated atoms, for instance, $F_j := a_1 \vee a_2 \vee ... \vee a_{k_j}$. When every atom in F_j is a ground atom, we term this one possible *grounding* of the clause. Note that a clause can have length one, *i.e.*, it may consist of only one atom.

A MLN is a collection of tuples (F_j, w_j) with $j \in \{1, ..., m\}$ where F_j is a clause and w_j is a real-valued number, the weight or confidence of the clause. Let $\mathscr{C} = \{c_1, ..., c_{|\mathscr{C}|}\}$ be a set of constants. The MLN forms a *ground MLN* by a) assigning a binary node z_i in the graphical model to every possible ground atom that can be formed given $\mathscr{C}$ and all predicates occurring in the set of clauses of the MLN, and b) by adding a binary feature for every possible grounding of every clause. The binary feature is weighted with the underlying weight of the clause and it is one if the associated grounding is true and it is zero otherwise. Moreover, the set of clauses can be extended with *hard* clauses, *i.e.*, clauses with a weight of infinity enforcing that every possible grounding of these clauses must hold true. The ground MLN defines the probability mass over worlds: [1]

$$p(\mathbf{z}; \mathbf{w}) = \frac{1}{Z} \exp\left\{ \sum_j w_j n_j(\mathbf{z}) \right\}. \tag{1}$$

Where $\mathbf{z} \in \{0, 1\}^N$ is a possible world, $n_j(\mathbf{z})$ is the number of true groundings of F_j in $\mathbf{z}$, and Z normalizes the distribution. Note that for clauses of length one, which are termed weighted ground atoms, n_j is either zero or one. The MAP state is defined as the world with the highest probability and we assume access to a MAP solver is provided throughout the following sections.

Learning the weight parameters specified in equation (1) can be performed by gradient-based optimization when defining the loss criterion $L(\mathbf{w})$ to be the negative log likelihood of a given dataset. Taking the derivative of the negative logarithm of (1), we obtain

$$\frac{\partial L}{\partial w_j} = -\frac{\partial \log p(\mathbf{z}; \mathbf{w})}{\partial w_j} = \mathbb{E}_{\mathbf{z} \sim p(z)}[n_j(\mathbf{z})] - n_j(\mathbf{z}). \tag{2}$$

The expectation in equation (2) requires summation over all possible worlds; therefore, the perceptron rule approximates it by using the counts in the MAP state [14, 15] which is much easier to obtain in the most cases (*e.g.*, [33]).

4. Backpropagating through Markov Logic

Instead of promoting a novel neural-symbolic model, the goal of this work is to combine already existing components and investigate how far we can get with a hybrid pipeline that simply relies

[1] Hard clauses define an additional collection of constraints which is omitted for brevity.

on the implementation of one backward pass. We focus on model behavior when intersecting neural and symbolic components and therefore restrict the setting to uncertain clauses of length one, hence, weighted ground atoms, and hard clauses.

Let us assume that we are given a MLN with of a set of hard clauses $\{\tilde{F}_j\}_{j=1}^{k}$ and a collection of clauses $\{(F_i, w_i)\}_{i=1}^{m}$ representing uncertain ground atoms $\{z_i\}_{i=1}^{m}$. We assume continuous input features $\mathbf{x}$ are available and provide information about the atoms of the MLN. As the MLN exclusively deals with symbolic inputs and weights, we use a neural module as an intermediate between the input and the MLN. We do not pose any restriction on the structure of $\mathbf{x}$ but we require the neural module to be queryable in regard to the uncertain atoms, given $\mathbf{x}$. Therefore, the neural module defines a parameterized function, f, with parameters $\boldsymbol{\theta}$, which takes $\mathbf{x}$ as input and outputs a real valued score for some or all of the uncertain ground atoms, *e.g.*, $f(\mathbf{x};\boldsymbol{\theta}) \in \mathbb{R}^m$.

The goal is to infer the correct truth values of the ground atoms $\{z_i\}_{i=1}^{m}$ or a subset thereof. Please note, inference in form of interpretations, *i.e.*, the MAP state of an MLN, can be advantageous already for moderate sizes of m as query-based inference would require to transform every ground atom into a query and infer its truth value separately, given the clauses.

4.1. Hybrid Model

Equation (1) defines a probability distribution in the exponential family with natural parameter $\mathbf{w}$. We model the natural parameter with the neural module, hence, we consider the following probabilistic model

$$\mathbf{z} \sim p(\mathbf{z};\mathbf{w}) \quad \mathbf{w} = f(\mathbf{x};\boldsymbol{\theta}), \tag{3}$$

where $p(\mathbf{z};\mathbf{w})$ is the distribution according to equation (1). In the forward pass, the neural module processes the input and computes atom confidences which serve as input for the MLN. We seek to learn the parameters of the neural module in this hybrid pipeline, *i.e.*, we seek $\frac{dL}{d\boldsymbol{\theta}}$ for a loss criterion $L(\boldsymbol{\theta})$. If we now assume $L(\boldsymbol{\theta})$ is the negative log-likelihood as in Section 3, we can base the backward pass of the hybrid model on the perceptron rule. By using the chain rule and equation (2) with the perceptron approximation, we obtain [2]

$$\frac{dL}{d\boldsymbol{\theta}} = \frac{dL}{d\mathbf{w}}\frac{d\mathbf{w}}{d\boldsymbol{\theta}} \approx \left[\texttt{MAP}(\mathbf{w}) - \mathbf{z}\right]^T \frac{d\mathbf{w}}{d\boldsymbol{\theta}}. \tag{4}$$

In equation (4), $\frac{d\mathbf{w}}{d\boldsymbol{\theta}}$ is simply the gradient of the neural network, f. *Differentiating through* the MLN is achieved by approximating the gradients of the negative log-likelihood $\frac{dL}{d\mathbf{w}} \approx \texttt{MAP}(\mathbf{w}) - \mathbf{z}$, which is the expression of interest throughout the next sections. In the realm of automatic differentiation, it suffices to implement $\frac{dL}{d\mathbf{w}}$without making further conceptual modifications to the logical component.

[2]Note that in the perceptron rule the expectation in equation (2) is substituted with the MAP state and $n(\mathbf{z})$ directly corresponds to $\mathbf{z}$ as the count of a weighted atom in the data is either 0 or 1.

4.2. Perturbation-based Implicit Differentiation

While in the previous section we are restricted to a likelihood criterion directly applied to the MLN output, perturbation-based implicit differentiation [16] allows for more general models and arbitrary loss functions $L(\boldsymbol{\theta})$. In particular, we utilize implicit maximum likelihood estimation (IMLE) [13] which minimizes the KL-divergence between the model distribution of the discrete component, in our case the MLN, and a target distribution whose parameters depend on the weights $\mathbf{w}$ and the downstream gradient of the loss function $\frac{dL}{d\mathbf{z}}$. Our gradient expression of interest is then calculated by using a noise perturbation scheme based on Gumbel perturbations. For any loss criterion L, we compute the gradients as

$$\frac{dL}{d\mathbf{w}} \approx \frac{1}{\lambda}\Big[\frac{1}{S}\sum_{i=1}^{S}\texttt{MAP}\big(\mathbf{w}+\boldsymbol{\epsilon}_i\big) - \frac{1}{S}\sum_{i=1}^{S}\texttt{MAP}\big(\mathbf{w}-\lambda\frac{dL}{d\mathbf{z}}+\boldsymbol{\epsilon}_i\big)\Big] \tag{5}$$

where $S \in \mathbb{N}^+$ and each $\boldsymbol{\epsilon}_i$ is a vector of *i.i.d*. Gumbel samples. We use the sampling scheme proposed in [13] and sample the elements of $\boldsymbol{\epsilon}_i$ from

$$\frac{\tau}{k}\Big[\sum_{c=1}^{C}\{\texttt{Gamma}(1/k, k/c)\} - \log(C)\Big], \tag{6}$$

where τ is a temperature parameter, $C \in \mathbb{N}^+$, and k controls how many entries of $\mathbf{z}$ must be true. As there is no such restriction in the context of our experiments, we treat k as a hyperparameter. By using IMLE, we can extend the model in (3) to cases where the perceptron rule is not applicable or does not provide enough gradient signal. For the extended model, we have

$$\mathbf{y} = \mathbf{g}(\mathbf{z}; \Omega) \quad \mathbf{z} \sim \mathbf{p}(\mathbf{z}, \mathbf{w}) \quad \mathbf{w} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta}), \tag{7}$$

with $\mathbf{y}$ representing any observable output pattern and g a parameterized function. After defining any loss criterion on $\mathbf{y}$, equation (7) may represent a general computational graph with the MLN embedded as a node, receiving upstream gradients $\frac{dL}{d\mathbf{z}}$ and sending downstream gradients $\frac{dL}{d\mathbf{w}}$ by using equation (5). A special case of equation (5) is also proposed in [12]:

$$\frac{dL}{d\mathbf{w}} \approx \frac{1}{\lambda}\big[\texttt{MAP}\big(\mathbf{w}\big) - \texttt{MAP}\big(\mathbf{w}-\lambda\frac{dL}{d\mathbf{z}}\big)\big] \tag{8}$$

It can be shown that, for large values of λ, equation (8) approximates the perception rule when the Hamming distance is chosen as the loss criterion [13] .

5. Experiments

We evaluate the gradient formulations presented in the previous sections in two scenarios. In particular, we construct a dataset in the context of description logic based schema reasoning and we use the CLUTRR benchmark suite [17]. Both datasets are comprised of textual input instances which describe a symbolic world represented by atoms. The challenging part is that the dataset instances are separated into *explicit* and *implicit* atoms where the explicit atoms can be derived from the textual input but the implicit atoms must be derived by reasoning over the

explicit atoms. Although inference times are an important factor in these settings, backward times roughly double in comparison to neural-only baselines when training the hybrid model, *i.e.,* inducing the MLN does not produce a bottleneck within the scope of our experiments.[3]

One finding of our experiments is that using gradients of equation (4) and (8) does not lead to well behaved training dynamics, *i.e.,* the loss value does not decrease when training under various hyperparemter settings. This is in contrast to using equation (5) on which the experimental results in the following sections are based. We conjecture that noise perturbations are beneficial and we discuss this further in Section 5.4.

5.1. General Architecture

For function f in equation (3), we use the base BERT architecture [10] pre-trained for next sentence prediction and masked language modelling and we make the word embeddings constants throughout the experiments.[4] For obtaining atom confidences, we employ the standard procedures used in the context of question answering with transformers. In particular, the text input is preceded by a [cls] token and the query is appended to the input following a [sep] token. The query may represent an individual target atom (schema) or refers to two target entities (CLUTRR). The embedding of the [cls] token is fed into a fully connected layer for predicting a score (schema) or a vector of scores (CLUTRR). For the neural-only baseline, we directly use these scores for the final prediction.

The confidence score or vector obtained by the fully connected layer defines the atom weights for the MLN as proposed in equation (3). Instead of using the raw confidences, we found applying a leaky-ReLu [37] layer is beneficial in the context of our experiments. The hybrid model is then trained end-to-end using stochastic optimization and the gradient formulations proposed in Section 4 while using absolute error on the MLN output. We use the same random seed for all the experiments and in regard to equation (5) we set S to one and $C = 10$. In regard to the loss criterion as specified in Section 4.2 we use absolute error. Detailed architecture descriptions, training details, and hyperparameter specifications are presented in Appendix B.

5.2. Schema Dataset

Description Logic. We construct a NLP dataset based on schema reasoning in the description logic $\mathbb{EL}++$ [38, 39]. The basic building blocks of the syntax are concepts and roles. A schema, also termed Tbox, consists of axioms expressing knowledge about the concepts and roles. For instance, the axiom $A \sqsubseteq B$ states that every individual belonging to concept A also belongs to concept B. The combination with MLNs is well studied and therefore we only focus on the relevant parts for this work and refer to [7] for a detailed discussion.

In general, schema axioms are associated with a confidence score and they are mapped towards weighted MLN atoms, *i.e.,* clauses with length one. Furthermore, the MLN is equipped with a set of hard clauses, termed the completion rules [39]. Given a collection of weighted atoms and the completion rules, the reasoning task performed by the MLN is to output the most

[3] For all the experiments we use RockIt [34] under the fast cutting plane inference algorithm [35] as the `MAP` solver.

[4] We use the implementation provided by the Huggingface transformer library [36].

probable classified Tbox. A classified Tbox contains all axioms that must hold true. We show the mapping of axioms to atoms in Table 1 in the appendix.

Dataset construction. The *schema dataset* is based on textualizing the atoms contained in the schemas, as shown in Table 1 in the appendix. The dataset consists of multiple instances where each defines a textualized schema containing on average 20 atoms and four predicate types. Every instance is generated as follows: We first sample a set of concepts and roles given a vocabulary of words. Subsequently, we sample a set of axioms between the concepts and roles where the corresponding MLN atoms define the set of *explicit* atoms. We then derive the implicit atoms by classifying the schema, *i.e.,* deriving all atoms that follow from the explicit atoms. We additionally sample a set of negative atoms. Finally, we textualize the explicit atoms receiving the textual input for the models. The following example demonstrates one dataset instance with the respective inputs and required outputs if the input schema only contained two atoms.

Simplified dataset example. For the Tbox {(sub(minister, politician)); (sub(politician, person)}, the classified Tbox additionally contains the implicit atom sub(minister, person) derived by the hard clause $sub(A, B) \wedge sub(B, C) \rightarrow sub(A, C)$. The explicit atoms are transformed into the text input *"Minister subset politician. Politician subset person"*. For this simplified example, a model would take the text as input and is required to predict the truth values of the explicit atoms, the implicit atom, and a set of randomly selected false atoms with same cardinality, for instance, {sub(person, politician); sub(politician, minister)}. In the hybrid architectures, the neural module calculates the atom confidences which are passed to the MLN. The `MAP` state of the MLN defines the final output prediction.

Setting. We generate training, validation, and testing splits according to the above procedure. We define the reasoning degree as the proportion of implicit and explicit atoms in the dataset. We generate an additional dataset (*Train middle)* in which we augment the data with instances exclusively containing explicit atoms to investigate if the amount of reasoning in the dataset affects the overall performance. The training datasets contain 20000 atoms and the test and valid splits contain 4000 atoms. During training and at test time, a model receives the textualized explicit atoms as input and has to classify the explicit, implicit and sampled negative atoms.

Results. Table 5.2 depicts results on the test sets for the schema dataset after training for 5 epochs and using early stopping based on the performance on the validation set. *Train middle* and *Train high* denotes training on a dataset with a reasoning degree of 15% and 37%, respectively. The test sets have a reasoning degree of 15% and 38%, respectively. We calculate results for the implicit and explicit atoms separately and due to the unbalance of positives and negatives we report F1 scores. The neural baseline achieves reasonable results, however, the complexity of the dataset is too high in general. The hybrid training strategy leads to strong results for the explicit and implicit atoms, suggesting the combination of neural and reasoning is beneficial for this dataset.

	Train middle				Train high			
	Test middle		Test high		Test middle		Test high	
	implicit	explicit	implicit	explicit	implicit	explicit	implicit	explicit
Hybrid	98.19	99.25	98.98	99.04	98.23	98.45	99.03	98.94
Neural	78.23	91.84	78.15	83.14	78.40	88.69	77.76	82.64
Random	35.47	45.77	44.54	40.23	35.47	45.77	44.54	40.23

Table 1
F1 scores on the schema datasets for the implicit and explicit atoms. For the hybrid model, the gradient formulation of equation (5) is used.

5.3. CLUTRR Dataset

The CLUTRR benchmark suite [17] is designed for testing robustness and reasoning abilities of natural language understanding systems. The datasets consist of small stories describing kinship relationships between entities and the task is to infer the missing relationship between two target entities. The dataset is governed by a set of rules such as $wife(A, B) \wedge hasChild(B, C) \rightarrow hasChild(A, C)$. We cannot use the same dataset as in [17] and compare as we found two erroneous rules in the dataset construction; further details are provided in Appendix B.2. We deleted these rules and recreated two dataset instances with a) using stories with raw facts such as *"A is the sister of B"* and b) facts replaced with paraphrases gathered by Mturk users [17].

Setting. The datasets contain a training split with roughly 10000 stories and 9 test splits with 500 stories respectively. Each test split is characterized by a different number of reasoning hops required to find the target relationships. The training sets contain exclusively stories where two hops are required. Likewise, we use as a validation set the test set containing stories with two hops. A model takes a story as input and has to predict truth values of the atoms defining the relationships between the entities in the story (implicit and explicit). At test time, the model is evaluated in regard to predicting the relationship between the target entities correctly.

Results. Table 2 shows accuracies on the test sets (3-10 hops) after training on stories with two hops for 4 epochs and validating on stories with two hops. The random baseline achieves an accuracy of $\frac{1}{11}\%$ and we also report the majority baseline for every test set. The neural baseline achieves high accuracies for the easier settings sets but the performance is more stable for the hybrid on more difficult test sets suggesting generalizability towards unseen reasoning complexities. The results in the Mturk setting are significantly weaker for both models. Nevertheless, the hybrid model achieves a notable accuracy of 72.37% for the hardest test set. The difference between the Mturk and the Vanilla setting highlights that the model performance strongly depends on the ability of finding the explicit atoms.

	Test set (hops)	3	4	5	6	7	8	9	10
Vanilla	Hybrid	100	97.60	95.05	93.81	95.13	94.84	95.22	93.64
	Neural	100	97.46	94.34	84.63	75.63	59.72	54.18	44.67
	Majority	9.09	13.80	11.30	6.30	4.28	1.98	1.39	0.20
Mturk	Hybrid	89.01	89.11	84.95	82.62	80.98	81.08	75.90	72.37
	Neural	92.64	73.66	55.56	45.61	40.49	38.65	32.47	32.20
	Majority	9.09	14.14	10.75	7.99	4.14	1.59	0.39	9.09

Table 2
Accuracy results on the CLUTRR dataset for multiple test sets with varying story lengths. For the hybrid model, the gradient formulation of equation (5) is used.

5.4. Discussion

While gradients calculated by using equations (4) and (8) also may work in different settings [13, 12], in our experiments, the Gumbel perturbations are necessary. One reason is that noise perturbations allow for additional gradient signal even if the corresponding confidence is already correct in regard to an explicit atom. For instance, assume the neural model calculates a confidence of 0.3 for an explicit atom. If this atom is predicted correctly by the MLN, the upstream gradient for the neural model is zero when using equations (4) or (8), hence, no further gradient signal will be provided. When using Gumbel noise, on the other hand, a negative ϵ_i can lead to a negative confidence and a respective prediction of 0 for the label. This will lead to an upstream gradient unequal to zero for the neural module, *e.g.*, allowing for an update towards a higher confidence.

We found a potential limitation of our approach is given by the fact that the atoms for which the neural module calculates confidences need to be specified by a deterministic mechanism. In other words, we need to know for which atoms we require the neural model to pass confidences to the MLN. While this is possible in our experiments, it imposes a limitation in terms of applicability as, for instance, querying the neural module for every possible atom is not feasible in general. Nevertheless, we hypothesize, this expresses a general challenge for neuro-symbolic models where often the outputs of a neural module are directly interpreted as atom or predicate confidences in a predefined logical structure (*e.g.* [31, 40]).

6. Conclusion

In this work, we presented a simple yet effective strategy of integrating MLNs into end-to-end machine learning. We utilize perturbation-based implicit differentiation [16, 13] for achieving a differentiable objective formulation. Our experimental findings suggest that the integration strategy can be beneficial in the context of relational datasets with continuous input when using noise perturbations on the MLN parameters. A more thorough investigation of the different gradient estimators is required in future work.

References

[1] G. Marcus, The next decade in ai: four steps towards robust artificial intelligence, arXiv preprint arXiv:2002.06177 (2020).

[2] G. Marcus, E. Davis, Rebooting AI: Building artificial intelligence we can trust, Vintage, 2019.

[3] A. S. d. Garcez, K. B. Broda, D. M. Gabbay, Neural-symbolic learning systems: foundations and applications, Springer Science & Business Media, 2012.

[4] M. Richardson, P. Domingos, Markov logic networks, in: Machine learning, volume 62, Springer, 2006, pp. 107–136.

[5] C. Zirn, M. Niepert, H. Stuckenschmidt, M. Strube, Fine-grained sentiment analysis with structural features, in: Proceedings of 5th International Joint Conference on Natural Language Processing, 2011, pp. 336–344.

[6] K. Gayathri, K. Easwarakumar, S. Elias, Probabilistic ontology based activity recognition in smart homes using markov logic network, in: Knowledge-Based Systems, volume 121, Elsevier, 2017, pp. 173–184.

[7] M. Niepert, J. Noessner, H. Stuckenschmidt, Log-linear description logics, in: Twenty-Second International Joint Conference on Artificial Intelligence, Citeseer, 2011.

[8] S. Satpal, S. Bhadra, S. Sellamanickam, R. Rastogi, P. Sen, Web information extraction using markov logic networks, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 2011, pp. 1406–1414.

[9] S. Hochreiter, J. Schmidhuber, Long short-term memory, in: Neural computation, volume 9, MIT Press, 1997, pp. 1735–1780.

[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, Advances in Neural Information Pro-cessing Systems (2017) 6000–6010.

[12] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, M. Rolinek, Differentiation of blackbox combinatorial solvers, in: International Conference on Learning Representations, 2020.

[13] M. Niepert, P. Minervini, L. Franceschi, Implicit mle: Backpropagating through discrete exponential family distributions, arXiv preprint arXiv: 2106.01798 (2021).

[14] M. Collins, Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms, in: Proceedings of the 2002 conference on empirical methods in natural language processing., 2002, pp. 1–8.

[15] D. Lowd, P. Domingos, Efficient weight learning for markov logic networks, in: European conference on principles of data mining and knowledge discovery, Springer, 2007, pp. 200–211.

[16] J. Domke, Implicit differentiation by perturbation, volume 23, Citeseer, 2010, pp. 523–531.

[17] K. Sinha, S. Sodhani, J. Dong, J. Pineau, W. L. Hamilton, Clutrr: A diagnostic benchmark for inductive reasoning from text, Proceedings of the 2019 Conference on Empirical Methods

in Natural Language Processing. (2019).
[18] M. Diligenti, M. Gori, V. Scoca, Learning efficiently in semantic based regularization, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 33–46.
[19] Z. Hu, X. Ma, Z. Liu, E. Hovy, E. Xing, Harnessing deep neural networks with logic rules, in: 4th Annual Meeting of the Association for Computational Linguistics, 2016, p. 2410–2420.
[20] P. Minervini, T. Demeester, T. Rocktäschel, S. Riedel, Adversarial sets for regularising neural link predictors, Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (2017).
[21] T. Rocktäschel, S. Singh, S. Riedel, Injecting logical background knowledge into embeddings for relation extraction, in: Proceedings of the 2015 conference of the north American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015, pp. 1119–1129.
[22] S. Badreddine, A. d'Avila Garcez, L. Serafini, M. Spranger, Logic tensor networks, arXiv preprint arXiv:2012.13635 (2020).
[23] R. Evans, E. Grefenstette, Learning explanatory rules from noisy data, in: Journal of Artificial Intelligence Research, volume 61, 2018, pp. 1–64.
[24] T. Rocktäschel, S. Riedel, End-to-end differentiable proving, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 2017, pp. 3788–3800.
[25] P. Minervini, M. Bošnjak, T. Rocktäschel, S. Riedel, E. Grefenstette, Differentiable reasoning on large knowledge bases and natural language, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 5182–5190.
[26] L. Weber, P. Minervini, J. Münchmeyer, U. Leser, T. Rocktäschel, Nlprolog: Reasoning with weak unification for question answering in natural language, Association for Computational Linguistics (2019).
[27] P. Minervini, S. Riedel, P. Stenetorp, E. Grefenstette, T. Rocktäschel, Learning reasoning strategies in end-to-end differentiable proving, in: International Conference on Machine Learning, PMLR, 2020, pp. 6938–6949.
[28] W. Cohen, F. Yang, K. R. Mazaitis, Tensorlog: A probabilistic database implemented using deep-learning infrastructure, in: Journal of Artificial Intelligence Research, volume 67, 2020, pp. 285–325.
[29] G. Marra, M. Diligenti, F. Giannini, M. Gori, M. Maggini, Relational neural machines, in: G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), ECAI 2020 - 24th European Conference on Artificial Intelligence, volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 1340–1347.
[30] G. Marra, O. Kuželka, Neural markov logic networks, arXiv preprint arXiv:1905.13462 (2019).
[31] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, Deepproblog: Neural probabilistic logic programming, in: Advances in Neural Information Processing Systems, volume 31, 2018, pp. 3749–3759.
[32] A. Kimmig, G. Van den Broeck, L. De Raedt, An algebraic prolog for reasoning about

possible worlds, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 25, 2011.
[33] G. Papandreou, A. Yuille, S. Nowozin, P. Gehler, J. Jancsary, C. Lampert, Perturb-and-map random fields: Reducing random sampling to optimization, with applications in computer vision, Advanced structured prediction (2014) 159.
[34] J. Noessner, M. Niepert, H. Stuckenschmidt, Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 27, 2013.
[35] S. Riedel, Improving the accuracy and efficiency of map inference for markov logic, in: UAI, 2008.
[36] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, 2020, pp. 38–45.
[37] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, volume 30, Citeseer, 2013, p. 3.
[38] F. Baader, S. Brandt, C. Lutz, Pushing the el envelope, in: IJCAI, volume 5, 2005, pp. 364–369.
[39] F. Baader, C. Lutz, S. Brandt, Pushing the EL envelope further, in: OWLED (Spring), volume 496 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2008.
[40] E. Tsamoura, L. Michael, Neural-symbolic integration: A compositional perspective, Proceedings of the AAAI Conference on Artificial Intelligence (2021).

A. Axioms Mapping

Axiom	MLN Atom	Textualized Representation
$A \sqsubseteq B$	$sub(A, B)$	[A] *subset of* [B]
$A \sqcap B \sqsubseteq C$	$int(A, B, C)$	*The intersection of [A] and [B] is a subset of [C]*
$\exists r.\top \sqsubseteq C$	rsub(T,r,C)	*Everything that is in role [r] with something is also in [C]*
$C \sqsubseteq \exists r.\top$	rsup(C,r,T)	*[C] is a subset of everything that is in role [r] with something*

Table 1
Example axioms and textual representation.

B. Experimental Details

B.1. Schema Dataset

Architecture/Training details. For each instance of the dataset, the neural backbone predicts confidences for the explicit, implicit and sampled false atoms. Each of these query atoms is textualized and appended to the input after a [sep] token as explained in Section 5.1. This procedure is applied for every query atom separately to not hard-code specific world configurations. The fully connected layer, which is trained from scratch, outputs one real-valued confidence score on which the leaky-ReLu is applied and the so obtained atom confidences are forwarded to the MLN. During training the hybrid model, we use absolute error as a loss criterion based on the MLN output (`MAP` state). During test time, the output of the MLN as used as the final atom predictions. The neural baseline uses the exact same architecture as the neural backbone and is trained using binary cross entropy.

Hyperparameter for the schema dataset. We use the Adam optimizer with betas equal to 0.5. The weight decay is set to 5×10^{-4}. We search over learning rates, τ and the negative slope of the leaky-ReLu activations on the validation set and use absolute error as a loss criterion. We set k=10, the average number of true atoms in a schema instance. Finally, we use a learning rate of 10^{-4}, $\tau = 0.5$, $\lambda = 10$, and a negative slope of 5×10^{-2}.

B.2. CLUTRR dataset

Dataset recreation. The dataset is governed by a set of rules such as *"When A is the wife of B and B has child C then A has child C"*. We found two erroneous rules in the dataset construction which rendered almost every entity pair as siblings on some instances. We noticed this when

observing an initial bad performances of the hybrid model on the CLUTRR dataset. Despite the confidences for the explicit relationships in the input story were reasonable, the MLN was not able to infer the relationship between the two target entities correctly. Therefore, for debugging purposes, we enforced the the target relationship by adding it as an evidence atom. In the new `MAP` state (with the target relationship treated as hard evidence) all entities in the story were rendered as siblings despite the neural confidences for these relationships being negative. Therefore, we further investigated the dataset internals and we found the two erroneous rules used during dataset construction:

When A has child B and B has uncle C then A and C are siblings.

This is not always the case, however, as C could be the sibling of A's spouse.

When A has child B and B has grandparent C then A is child of C.

Here, A is not in general child of C as C could be the parent of A's spouse. We cloned the project repository and deleted these rules from the dataset construction. We recreated the two dataset types with a) using stories with raw facts such as *"A is the sister of B"* and b) facts replaced with paraphrases gathered by Mturk users [17]. For a), we used the following command:

```
python main.py --train_tasks 1.2,1.3 --test_tasks 1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,1.10
--train_rows 10000 --test_rows 500 --holdout
```

For b), we used the command:

```
python main.py --train_tasks 1.2,1.3 --test_tasks 1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,1.10
--train_rows 10000 --test_rows 500 --holdout --use_mturk_template
```

Architecture/Training details. For the hybrid model, the neural backbone takes the story inputs and is queried for the relationship between the entities whose relationships are described in the story (explicit and implicit) and passes the confidences to the MLN. We only consider gender neutral relations, for instance, treating *mother* and *father* as *parent*. Therefore, the fully connected layer takes the embedding of the [cls] token and outputs a vector with 11 scores for the relationships in regard to one query atom. We follow [17] and substitute the entity names randomly with digit placeholders and likewise for the Mturk setting we augment the stories with two irrelevant facts. The hybrid model is then trained end-to-end using absolute error on the output of the MLN. During test time, the output of the MLN defines the final predictions where we only regard the predicted atom between the target entities in a respective story. When training the neural baseline, the same neural architecture as the neural backbone is used and trained using binary cross entropy, however, in favor of the neural model it is exclusively trained on the target relationship.

Hyperparameter for CLUTRR. We use the same optimizer parameters as for the schema experiments except that we use a learning rate of 10^{-5}. Furthermore, we use k=5, the same value of τ as above, and a value of one for λ. Hyperparameters are found by training and validating on a different dataset instance created under the same configurations as described above.

C. Ablation results

	Perceptron	*M-M*	τ=0.5	τ=1.0	τ=2.0	τ=3.0	τ=4.0
implicit	$\approx$ 35.47	$\approx$ 35.47	99.03	98.05	98.08	98.30	77.53
explicit	$\approx$ 45.77	$\approx$ 45.77	98.94	98.41	98.19	98.39	78.74

Table 2
F1 scores on the validation set of the schema dataset under different model specifications.