Neurosymbolic Reasoning and Learning with Restricted Boltzmann Machines

Son N. Tran[1], **Artur d'Avila Garcez**[2]

[1]The University of Tasmania, Launceston, Tasmania, 7248, Australia
[2]City, University of London, Northampton Square, London, EC1V 0HB, UK
sn.tran@utas.edu.au, a.garcez@city.ac.uk

Abstract

Knowledge representation and reasoning in neural networks has been a long-standing endeavour which has attracted much attention recently. The principled integration of reasoning and learning in neural networks is a main objective of the area of neurosymbolic Artificial Intelligence. In this paper, a neurosymbolic system is introduced that can represent any propositional logic formula. A proof of equivalence is presented showing that energy minimization in restricted Boltzmann machines corresponds to logical reasoning. We demonstrate the application of our approach empirically on logical reasoning and learning from data and knowledge. Experimental results show that reasoning can be performed effectively for a class of logical formulae. Learning from data and knowledge is also evaluated in comparison with learning of logic programs using neural networks. The results show that our approach can improve on state-of-the-art neurosymbolic systems. The theorems and empirical results presented in this paper are expected to reignite the research on the use of neural networks as massively-parallel models for logical reasoning and promote the principled integration of reasoning and learning in deep networks.

Introduction

Increasing attention has been devoted in recent years to knowledge representation and reasoning in neural networks. The principled integration of reasoning and learning in neural networks is a main objective of the area of neurosymbolic Artificial Intelligence (AI) (d'Avila Garcez and Lamb 2020). In neurosymbolic AI, neural networks and symbolic AI are combined. Typically, an algorithm is provided to translate some form of symbolic knowledge representation into the architecture and initial set of parameters of a neural network. Ideally, a theorem then shows that the neural network can be used as a massively-parallel model of computation capable of reasoning about such knowledge. Finally, when trained with data and knowledge, the network is expected to produce a better performance - either a higher accuracy or faster learning - than when trained from data alone.

Symbolic knowledge may be provided to a neural network in the form of general rules which are known in a given domain, or rules which are expected to be $true$ across domains when performing transfer learning. Over the years, many neurosymbolic approaches have used a form of knowledge representation based on *if-then* rules (Towell and Shavlik 1994; França, Zaverucha, and Garcez 2014; Tran and Garcez 2018; Evans and Grefenstette 2018; Yang, Yang, and Cohen 2017; Manhaeve et al. 2018; Tran 2021). Assuming *Modus-Ponens*[1] as the only rule of inference, given a logical formula of the form $B \leftarrow A$ (read "B is $true$ if A is $true$"), under the convention that 1 represents $true$ and 0 represents $false$, a neurosymbolic network would infer that approximately $B = 1$ given $A = 1$. This has two shortcomings. First, Modus-Ponens alone may not capture the entire reasoning required by the application, e.g. the use of Modus-Tollens[2] may be needed. Second, other forms of knowledge may need to be represented by the neural network, including with the use of negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$) and bi-conditional ($\leftrightarrow$), e.g. $(\neg A \vee B) \leftarrow (C \wedge D)$, read "not A or B holds $true$ if C and D are $true$", or $A \leftrightarrow B$, read "A is $true$ if and only if B is $true$". Although an equivalence between propositional calculus and connectionist networks has been shown in (Pinkas 1991, 1995), the translation and representation of knowledge by such networks can become convoluted, making it difficult to integrate with modern learning techniques.

In this paper, we introduce a method to translate logical formulae into simple 2-layer neural networks. The networks, called Logical Boltzmann Machines (LBM), work as a neurosymbolic system capable of (i) representing any formula in propositional logic, (ii) reasoning efficiently given such knowledge, (iii) learning from knowledge and data. We introduce an algorithm to translate any set of propositional logic formulae into a restricted Boltzmann machine (RBM) and we prove equivalence between the logical formulae and the energy-based connectionist model. In other words, we prove soundness of the translation algorithm. Specifically, the connectionist model (the RBM) is shown to assign minimum energy to the assignments of truth-values that satisfy the formulae. This provides a new way of performing reasoning in symmetrical neural networks by employing the network to search for the models of a logical theory, i.e. search for the assignments of truth-values that map the logi-

[1]If P implies Q and P is $true$ then Q must also be $true$.
[2]If P implies Q and Q is $false$ then P must also be $false$.

cal formulae to $true$[3].

In the experiments, we demonstrate the applications of our neurosymbolic system in reasoning and learning. We show that knowledge-encoded networks (LBMs) can find all satisfying assignments of a class of logical formulae by searching fewer than 0.75% of the possible (approximately 1 billion) assignments. We also show the ability of LBMs at solving satisfiability (SAT) problem without the need for training data as done in (Selsam et al. 2019), although LBMs are not as efficient as long-standing, purpose-built symbolic SAT solvers . When applied to benchmark data sets for neurosymbolic AI, logical Boltzmann machines achieved in five out of seven data sets a higher test set accuracy than a purely-symbolic learning system ALEPH (Srinivasan 2007), a neurosymbolic system based on if-then rules CILP++ (França, Zaverucha, and Garcez 2014), and a standard RBM (Smolensky 1995). Finally, we show that LBM can be deployed as a logical layer on top of convolutional neural networks. We compare effectiveness with Deep Logic Nets (Tran and Garcez 2018), Compositional Neural Logic Programming (Tran 2021), and Logic Tensor Networks (Serafini and d'Avila Garcez 2016; Badreddine et al. 2022) in a typical semantic image interpretation task.

The contribution of this paper is twofold. The paper offers:

- A proof of equivalence between classical propositional logic and restricted Boltzmann machines, and
- A foundation and neurosymbolic system for statistical inference, learning and logical reasoning.

Related Work

One of the earliest work on the integration of neural networks and symbolic knowledge is known as KBANN (Knowledge-based Artificial Neural Network (Towell and Shavlik 1994)), which encodes *if-then* rules into a hierarchical multilayer perceptron. In another early approach (Garcez, Broda, and Gabbay 2001), a single-hidden layer recurrent network is proposed to support logic programming rules. An extension of that approach to work with first-order logic programs, called CILP++ (França, Zaverucha, and Garcez 2014), uses the concept of *propositionalisation* from Inductive Logic Programming (ILP), whereby first-order variables can be treated as propositional atoms in the neural network. Also based on first-order logic programs, (Evans and Grefenstette 2018) propose a differentiable ILP approach that can be implemented by neural networks, and (Cohen, Yang, and Mazaitis 2017) maps stochastic logic programs into a differentiable function also trainable by neural networks. These are all supervised learning approaches.

Early work in neurosymbolic AI has also shown a correspondence between propositional logic and symmetrical neural networks (Pinkas 1991), in particular Hopfield networks, which nevertheless did not scale well with the number of variables and whose training regimen was inefficient. Variants of this work have been proposed, based on Conjunctive Normal Form, to solve the satisfiability problem using Boltzmann machines and higher-order Boltzmann machines (Hernandez et al. 2001; d'Anjou et al. 1993) . Among unsupervised learning approaches, Penalty Logic (Pinkas 1995) was the first to integrate nonmonotonic logic into symmetrical neural networks. However, these approaches require the use of higher-order connectionist networks, which can be difficult to construct [4] and inefficient to train. More recently, several attempts have been made to extract and encode symbolic knowledge into RBMs trained with the more efficient Contrastive Divergence algorithm (Penning et al. 2011; Tran and Garcez 2018). Such approaches explored the structural similarity between symmetric networks and logical rules using bi-conditional implication, but do not enjoy soundness results. By contrast, and similarly to Penalty Logic, the approach introduced in this paper is based on a proof of equivalence between the logic formulae and the symmetric networks; differently from Penalty Logic, it does not require the use of higher-order networks. Alongside the above approaches, which translate symbolic representations into neural networks, there are hybrid approaches that combine neural networks and symbolic AI systems as communicating modules of a neurosymbolic system. These include DeepProbLog (Manhaeve et al. 2018) and Logic Tensor Networks (LTN) (Serafini and d'Avila Garcez 2016).

Knowledge Representation in RBMs

Our approach is based on classical propositional logic, thus including all five connectives $\{\leftarrow, \neg, \wedge, \vee, \leftrightarrow\}$ and satisfying both Modus-Ponens and Modus-Tollens. Let us return to the simple $B \leftarrow A$ example used earlier. Given $B \leftarrow A$ as knowledge, if neuron A is assigned value 1 in the corresponding neurosymbolic network, we expect the network to converge to a state where neuron B has value approximately 1. If B is assigned value 0, we expect the network to converge to a state where A is approximately 0. If A is assigned 0, B should converge to approximately 0.5, since in the classical interpretation of logical implication, $B \leftarrow A$ is equivalent to $\neg A \vee B$, i.e. $B \leftarrow A$ is $true$ if A is $false$ regardless of the truth-value of B. Finally, if B is assigned 1 then A should converge to approximately 0.5, for the same reason.

An RBM (Smolensky 1995) is a two-layer neural network with bidirectional (symmetric) connections, which is characterised by an energy function $E(\mathbf{x}, \mathbf{h}) = -\sum_{i,j} w_{ij} x_i h_j - \sum_i a_i x_i - \sum_j b_j h_j$, where a_i and b_j are the biases of input unit x_i and hidden unit h_j, respectively, and w_{ij} is the connection weight between x_i and h_j. This RBM represents a joint probability distribution $p(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{x},\mathbf{h})}$, where $Z = \sum_{\mathbf{xh}} e^{-E(\mathbf{x},\mathbf{h})}$ is the partition function, $\mathbf{x} = \{x_i\}$ is the set of visible units, and $\mathbf{h} = \{h_j\}$ is the set of hidden units of the RBM.

In propositional logic, any well-formed formula (WFF) φ can be mapped onto Disjunctive Normal Form (DNF), i.e.

[3] We use the term *model* to refer to both a model of a logical theory and a neural network model. When the intended meaning is not clear from the context, we shall use the term *logic model*.

[4] Building such higher-order networks requires transforming the energy function into a quadratic form by adding hidden variables that are not present in the original logic formulae.

disjunctions ($\vee$) of conjunctions ($\wedge$), as follows:

$$\varphi \equiv \bigvee_j (\bigwedge_{t \in \mathcal{S}_{T_j}} \mathrm{x}_t \wedge \bigwedge_{k \in \mathcal{S}_{K_j}} \neg \mathrm{x}_k)$$

where $(\bigwedge_{t \in \mathcal{S}_{T_j}} \mathrm{x}_t \wedge \bigwedge_{k \in \mathcal{S}_{K_j}} \neg \mathrm{x}_k)$ is called a conjunctive clause with the propositional variables x_i divided into positive literals x_t and negative literals $\neg \mathrm{x}_k$, e.g. $\mathrm{x}_1 \wedge \mathrm{x}_2 \wedge \neg \mathrm{x}_3$ (read "x_1 and x_2 and not x_3").

Definition 1. *Let $s_\varphi(\mathbf{x}) \in \{0, 1\}$ denote the truth-value of a WFF φ given an assignment of truth-values $\mathbf{x}$ to the literals of φ, where truth-value $true$ is mapped to 1 and truth-value $false$ is mapped to 0. Let $E(\mathbf{x}, \mathbf{h})$ denote the energy function of an energy-based neural network $\mathcal{N}$ with visible units $\mathbf{x}$ and hidden units $\mathbf{h}$. φ is said to be* equivalent *to $\mathcal{N}$ if and only if there exists a function ψ, for any assignment $\mathbf{x}$, such that $s_\varphi(\mathbf{x}) = \psi(E(\mathbf{x}, \mathbf{h}))$.*

This definition of equivalence is similar to that of Penalty Logic (Pinkas 1995), whereby all assignments of truth-value satisfying a WFF φ are mapped to global minima of the energy function of network $\mathcal{N}$. In our case, by construction, assignments that do not satisfy the WFF will be mapped to maxima of the energy function.

Definition 2. *A* strict DNF *(SDNF) is a DNF with at most one conjunctive clause that maps to $true$ for any assignment of truth-values $\mathbf{x}$. A* full DNF *is a DNF where each propositional variable must appear at least once in every conjunctive clause.*

Lemma 1. *Any SDNF $\varphi \equiv \bigvee_j (\bigwedge_{t \in \mathcal{S}_{T_j}} \mathrm{x}_t \wedge \bigwedge_{k \in \mathcal{S}_{K_j}} \neg \mathrm{x}_k)$ can be mapped onto an energy function:*

$$E(\mathbf{x}) = -\sum_j (\prod_{t \in \mathcal{S}_{T_j}} x_t \prod_{k \in \mathcal{S}_{K_j}} (1 - x_k))$$

where $\mathcal{S}_{T_j}$ (resp. $\mathcal{S}_{K_j}$) is the set of T_j (resp. K_j) indices of the positive (resp. negative) literals in φ.

Proof. Each conjunctive clause $\bigwedge_{t \in \mathcal{S}_{T_j}} \mathrm{x}_t \wedge \bigwedge_{k \in \mathcal{S}_{K_j}} \neg \mathrm{x}_k$ in φ can be represented by $\prod_{t \in \mathcal{S}_{T_j}} x_t \prod_{k \in \mathcal{S}_{K_j}} (1 - x_k)$ which maps to 1 if and only if $x_t = 1$ (i.e. $true$) and $x_k = 0$ (i.e. $false$) for all $t \in \mathcal{S}_{T_j}$ and $k \in \mathcal{S}_{K_j}$. Since φ is a SDNF, it is $true$ if and only if one conjunctive clause is $true$. Then, the sum $\sum_j (\prod_{t \in \mathcal{S}_{T_j}} x_t \prod_{k \in \mathcal{S}_{K_j}} (1 - x_k)) = 1$ if and only if the assignment of truth-values to x_t, x_k is a logical model of φ. Hence, the neural network with energy function $E = -\sum_j (\prod_{t \in \mathcal{S}_{T_j}} x_t \prod_{k \in \mathcal{S}_{K_j}} (1 - x_k))$ is such that $s_\varphi(\mathbf{x}) = -E(\mathbf{x})$. □

Theorem 1. *Any SDNF $\varphi \equiv \bigvee_j (\bigwedge_{t \in \mathcal{S}_{T_j}} \mathrm{x}_t \wedge \bigwedge_{k \in \mathcal{S}_{K_j}} \neg \mathrm{x}_k)$ can be mapped onto an equivalent RBM with energy:*

$$E(\mathbf{x}, \mathbf{h}) = -\sum_j h_j (\sum_{t \in \mathcal{S}_{T_j}} x_t - \sum_{k \in \mathcal{S}_{K_j}} x_k - |\mathcal{S}_{T_j}| + \epsilon) \quad (1)$$

where $0 < \epsilon < 1$, $\mathcal{S}_{T_j}$ and $\mathcal{S}_{K_j}$ are, respectively, the sets of indices of the positive and negative literals in each conjunctive clause j of the SDNF, and $|\mathcal{S}_{T_j}|$ is the number of positive literals in conjunctive clause j.

Proof. We have seen in Lemma 1 that any SDNF φ can be mapped onto energy function $E = -\sum_j \prod_{t \in \mathcal{S}_{T_j}} x_t \prod_{k \in \mathcal{S}_{K_j}} (1 - x_k)$. For each expression $\tilde{e}_j(\mathbf{x}) = -\prod_{t \in \mathcal{S}_{T_j}} x_t \prod_{k \in \mathcal{S}_{K_j}} (1 - x_k)$, we define an energy expression associated with hidden unit h_j as $e_j(\mathbf{x}, h_j) = -h_j(\sum_{t \in \mathcal{S}_{T_j}} x_t - \sum_{k \in \mathcal{S}_{K_j}} x_k - |\mathcal{S}_{T_j}| + \epsilon)$. The term $e_j(\mathbf{x}, h_j)$ is minimized with value $-\epsilon$ when $h_j = 1$, written $min_{h_j}(e_j(\mathbf{x}, h_j)) = -\epsilon$. This is because $-(\sum_{t \in \mathcal{S}_{T_j}} x_t - \sum_{k \in \mathcal{S}_{K_j}} x_k - |\mathcal{S}_{T_j}| + \epsilon) = -\epsilon$ if and only if $x_t = 1$ and $x_k = 0$ for all $t \in \mathcal{S}_{T_j}$ and $k \in \mathcal{S}_{K_j}$. Otherwise, $-(\sum_t x_{t \in \mathcal{S}_{T_j}} - \sum_{k \in \mathcal{S}_{K_j}} x_k - |\mathcal{S}_{T_j}| + \epsilon) > 0$ and $min_{h_j}(e_j(\mathbf{x}, h_j)) = 0$ with $h_j = 0$. By repeating this process for each $\tilde{e}_j(\mathbf{x})$ we obtain that any SDNF φ is equivalent to an RBM with the energy function $E(\mathbf{x}, \mathbf{h}) = -\sum_j h_j(\sum_{t \in \mathcal{S}_{T_j}} x_t - \sum_{k \in \mathcal{S}_{K_j}} x_k - |\mathcal{S}_{T_j}| + \epsilon)$ such that $s_\varphi(\mathbf{x}) = -\frac{1}{\epsilon} min_{\mathbf{h}} E(\mathbf{x}, \mathbf{h})$. □

Knowledge Representation Capacity

We now show that any formulae in propositional logic can be encoded in RBMs by translation into SDNF.

Clausal Form. A clause (or disjunctive clause) is one of the most popular forms of knowledge representation used in AI. Horn clauses (or *if-then* rules in implication form) have at most one negated literal. Any statement in propositional logic can be transformed into clausal form by using the rules of equivalence. A set of statements can be transformed into an equivalent set of clauses. Consider a clause:

$$\varphi \equiv \bigvee_{t \in \mathcal{S}_T} \neg \mathrm{x}_t \vee \bigvee_{k \in \mathcal{S}_K} \mathrm{x}_k \quad (2)$$

which can be rearranged as $\varphi \equiv \varphi' \vee \mathrm{x}'$, where φ' is a disjunctive clause obtained by removing x' from φ. x' can be either $\neg \mathrm{x}_t$ or x_k for any $t \in \mathcal{S}_T$ and $k \in \mathcal{S}_K$. We have that:

$$\varphi \equiv (\neg \varphi' \wedge \mathrm{x}') \vee \varphi' \quad (3)$$

because $(\neg \varphi' \wedge \mathrm{x}') \vee \varphi' \equiv (\varphi' \vee \neg \varphi') \wedge (\varphi' \vee \mathrm{x}') \equiv True \wedge (\varphi' \vee \mathrm{x}')$. By De Morgan's law ($\neg(\mathrm{a} \vee \mathrm{b}) \equiv \neg \mathrm{a} \wedge \neg \mathrm{b}$; $\neg(\mathrm{a} \wedge \mathrm{b}) \equiv \neg \mathrm{a} \vee \neg \mathrm{b}$), one can always convert $\neg \varphi'$ (and therefore $\neg \varphi' \wedge \mathrm{x}'$) into a conjunctive clause.

By applying Eq. (3) repeatedly, each time we can eliminate a variable out of a disjunctive clause by moving it into a new conjunctive clause. The disjunctive clause φ holds $true$ if and only if either the disjunctive clause φ' holds $true$ or the conjunctive clause $(\neg \varphi' \wedge \mathrm{x}')$ holds $true$. The SDNF of the clause in Eq. (2) is:

$$\bigvee_{j \in \mathcal{S}_T \cup \mathcal{S}_K} (\bigwedge_{t \in \mathcal{S}_T \setminus j} \mathrm{x}_t \wedge \bigwedge_{k \in \mathcal{S}_K \setminus j} \neg \mathrm{x}_k \wedge \mathrm{x}'_j) \quad (4)$$

where $\mathcal{S} \setminus j$ denotes a set $\mathcal{S}$ from which j has been removed. $\mathrm{x}'_j \equiv \neg \mathrm{x}_j$ if $j \in \mathcal{S}_T$. Otherwise, $\mathrm{x}'_j \equiv \mathrm{x}_j$. This SDNF only has $|\mathcal{S}_T| + |\mathcal{S}_K|$ clauses, making the translation efficient.

Disjunctive Normal Form (DNF). Any formula φ can be converted into DNF. If φ is not SDNF then by definition there is a group of conjunctive clauses in φ which map to $true$ when φ is satisfied. This group of conjunctive clauses

can always be converted into SDNF by extending Eq. (3) to each conjunctive clause, i.e. replacing x' by a conjunction φ''. Therefore, any WFF can be converted into SDNF. For example:

$$(\mathrm{a} \wedge \mathrm{b}) \vee (\mathrm{a} \wedge \mathrm{c}) \equiv (\underbrace{\neg(\mathrm{a} \wedge \mathrm{b})}_{\neg\varphi'} \wedge \underbrace{(\mathrm{a} \wedge \mathrm{c})}_{\varphi''}) \vee \underbrace{(\mathrm{a} \wedge \mathrm{b})}_{\varphi'}$$

Conjunctive Normal Form (CNF). Every WFF can be converted into CNF. A CNF is a conjunction of disjunctive clauses:

$$\varphi_{\text{CNF}} \equiv \bigwedge_{m=1}^{M} \Big(\bigvee_{t \in \mathcal{S}_T^m} \neg x_t \vee \bigvee_{k \in \mathcal{S}_K^m} x_k \Big) \tag{5}$$

We now discuss the transformation of CNFs into restricted Boltzmann machines. We apply the transformation steps in Eq. (4) to each conjunctive clause in the CNF. The result is a conjunction of M SDNFs.

$$\varphi_{\text{CNF}} \equiv \bigwedge_{m=1}^{M} \Big(\bigvee_{j \in \mathcal{S}_T^m \cup \mathcal{S}_K^m} \Big(\bigwedge_{t \in \mathcal{S}_T^m \setminus j} \mathrm{x}_t \wedge \bigwedge_{k \in \mathcal{S}_K^m \setminus j} \neg \mathrm{x}_k \wedge \mathrm{x}'_j \Big) \Big) \tag{6}$$

This transformation increases the space complexity from $\mathcal{O}(M \times N)$ to $\mathcal{O}(M \times N^2)$), where M is the number of disjunctive clauses and N is the number of variables. This is not much of a problem for current computer systems, especially since inference with RBMs can be performed in parallel.

Although the formula in Eq. (6) is not in SDNF form (it is a conjunction of SDNFs), equivalence between the CNF and the RBM continues to hold. Let:

$$s_\varphi = \begin{cases} 1 & \text{when } -\frac{1}{\epsilon} min_{\mathbf{h}} E(\mathbf{x}, \mathbf{h}) = M \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The CNF is satisfied if and only if all M SDNFs are satisfied. Under such circumstances, $min_{\mathbf{h}} E(\mathbf{x}, \mathbf{h}) = -M\epsilon$, otherwise $min_{\mathbf{h}} E(\mathbf{x}, \mathbf{h}) = -M'\epsilon$ where $M' < M$. In the **Reasoning as Lowering Free Energy** section, we consolidate this result by introducing the concept of confidence value.

Reasoning in LBMs

We have seen how LBMs are constructed by mapping propositional logic formulae onto RBMs. In this section, we discuss inference in LBMs.

Reasoning as Sampling

There is a direct relationship between inference in LBMs and logical satisfiability, as follows.

Proposition 1. *Let $\mathcal{N}$ be an LBM constructed from a formula φ. Let $\mathcal{A}$ be a set of indices of variables that have been assigned to either $true$ or $false$ (we use $\mathbf{x}_\mathcal{A}$ to denote the set $\{x_\alpha | \alpha \in \mathcal{A}\}$). Let $\mathcal{B}$ be a set of indices of variables that have not been assigned a truth-value (we use $\mathbf{x}_\mathcal{B}$ to denote $\{x_\beta | \beta \in \mathcal{B}\}$). Performing Gibbs sampling on $\mathcal{N}$ given $\mathbf{x}_\mathcal{A}$ is equivalent to searching for an assignment of truth-values for $\mathbf{x}_\mathcal{B}$ that satisfies φ.*

Proof. (*sketch*) Theorem 1 showed that the truth-value of φ is inversely proportional to an LBM's rank function, that is: $s_\varphi(\mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{A}) \propto -min_{\mathbf{h}} E(\mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{A}, \mathbf{h})$. Therefore, a value of $\mathbf{x}_\mathcal{B}$ that minimises the energy function also maximises the truth value, because:

$$\mathbf{x}^*_\mathcal{B} = \arg\min_{\mathbf{x}_\mathcal{B}} (\min_{\mathbf{h}} E(\mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{A}, \mathbf{h})) = \arg\max_{\mathbf{x}_\mathcal{B}} (s_\varphi(\mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{A}))$$

We can use an iterative process to search for truth-values $\mathbf{x}^*_\mathcal{B}$ by minimising an LBM's energy function. This can be done by using gradient descent to update the values of $\mathbf{h}$ and then $\mathbf{x}_\mathcal{B}$ one at a time (similarly to the Contrastive Divergence algorithm) to minimise $E(\mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{A}, \mathbf{h})$ while keeping the other variables ($\mathbf{x}_\mathcal{A}$) fixed. The alternating updates are repeated until convergence. In the case of Gibbs sampling, given the assigned variables $\mathbf{x}_\mathcal{A}$, the process starts with a random initialisation of $\mathbf{x}_\mathcal{B}$, and proceeds to infer values for the hidden units h_j and then the unassigned variables x_β in the visible layer of the LBM, using the conditional distributions $h_j \sim p(h_j|\mathbf{x})$ and $x_\beta \sim p(x_\beta|\mathbf{h})$, respectively, where $\mathbf{x} = \{\mathbf{x}_\mathcal{A}, \mathbf{x}_\mathcal{B}\}$. These distributions are monotonic functions of the negative energy's gradient over $\mathbf{h}$ and $\mathbf{x}_\mathcal{B}$. Therefore, performing Gibbs sampling on those functions can be seen as moving towards a local minimum that is equivalent to an assignment of truth-values that satisfies formula φ. □

Since the energy function of the LBM and the satisfiability of the formula are inversely proportional, each step of Gibbs sampling to reduce the energy should intuitively generate a sample that is closer to satisfying the formula.

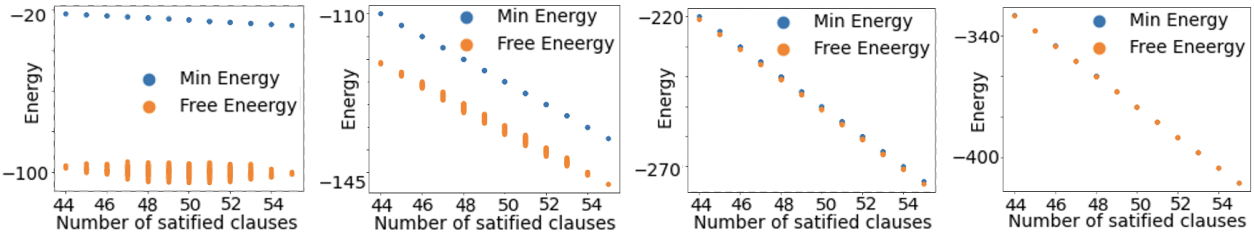
Reasoning as Lowering Free Energy

While the energy function of a LBM is intractable, its free-energy function can be computed analytically as: $\mathcal{F}_= \sum_j(-\log(1+\exp(c\sum_i w_{ij}x_i + b_j)))$. The free energy term $-\log(1+\exp(c\sum_i w_{ij}x_i + b_j))$ is a negative softplus function scaled by a non-negative value c called *confidence value*. It returns a negative output for a positive input and a close-to-zero output for a negative input. The value of c can be adjusted to make the function smooth.

Each free energy term is associated with a conjunctive clause in the SDNF through the weighted sum $\sum_i w_{ij}x_i + b_j$. Therefore, if a truth-value assignment of $\mathbf{x}$ does not satisfy the formula, all energy terms will be close to zero. Otherwise, one free energy term will be $-\log(1+\exp(c\epsilon))$, for a choice of $0 < \epsilon < 1$ obtained from Theorem 1. Thus, the more likely a truth assignment is to satisfy the formula, the lower the free energy. Formally:

$$s_\varphi(\mathbf{x}) = -\frac{1}{c\epsilon} \min_{\mathbf{h}} E(\mathbf{x}, \mathbf{h}) = \lim_{c \to \infty} -\frac{1}{c\epsilon} \mathcal{F}(\mathbf{x}) \tag{8}$$

As an example, Figure 1 shows the values of the energy functions for a CNF with 55 disjunctive clauses. The CNF is satisfied if and only if all 55 clauses are satisfied. As can be seen, the relationship is linear. The strong correlation between the free-energy function and the energy function of LBMs can increase its reasoning capability in practice. Since it is tractable, one can employ the free energy to infer logical variables deterministically.



(a) (c=1) (b) (c=5) (c) (c=10) (d) (c=15)

Figure 1: Linear correlation between satisfiability of a CNF and minimization of the free energy function for various confidence values c. Minimum energy and free energy values converge with an increasing value of c.

Experimental Results

Reasoning

In this experiment we apply LBM to effectively search for satisfying truth assignments of variables in large formulae. Let us define a class of formulae:

$$\varphi \equiv \bigwedge_{i=1}^{M} \mathrm{x}_i \wedge \Big(\bigvee_{j=M+1}^{M+N} \mathrm{x}_j \Big) \tag{9}$$

A formula in this class consists of 2^{M+N} possible truth assignments of the variables, with $2^N - 1$ of them mapping the formula to *true* (call this the *satisfying set*). Converting to SDNF as done before but now for the class of formulae, we obtain:

$$\varphi \equiv \bigvee_{j=M+1}^{M+N} \Big(\bigwedge_{i=1}^{M} \mathrm{x}_i \wedge \bigwedge_{j'=j+1}^{M+N} \neg \mathrm{x}_{j'} \wedge \mathrm{x}_j \Big) \tag{10}$$

Applying Theorem 1 to construct an LBM from φ, we use Gibbs sampling to infer the truth values of all variables. A sample is *accepted* as a satisfying assignment if its free energy is lower than or equal to $-\log(1 + \exp(c\epsilon)$ with $c = 5, \epsilon = 0.5$. We evaluate the *coverage* and *accuracy* of accepted samples. Coverage is measured as the proportion of the satisfying set that is accepted. In this experiment, this is the number of satisfying assignments in the set of accepted samples divided by $2^N - 1$. Accuracy is measured as the percentage of accepted samples that satisfy the formula.

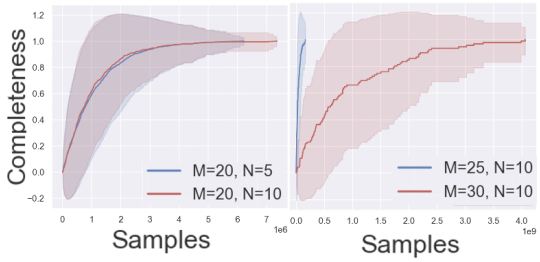


Figure 2: Percentage coverage as sampling progresses with different values for M and N averaged over 100 runs.

We test different values of $M \in \{20, 25, 30\}$ and $N \in \{3, 4, 5, 6, 7, 8, 9, 10\}$. LBM achieves 100% accuracy in all cases, meaning that all accepted samples do satisfy the formula. Figure 2 shows the coverage as Gibbs sampling progresses (after each time that a number of samples is collected). Four cases are considered: M=20 and N=5, M=20 and N=10, M=25 and N=10, M=30 and N=10.

In each case, we run the sampling process 100 times and report the average results with standard deviations. The number of samples needed to achieve 100% coverage is much lower than the number of possible assignments (2^{M+N}). For example, when M=20, N=10, all satisfying assignments are found after $\sim$ 7.5 million samples are collected, whereas the number of possible assignments is $\sim$ 1 billion, producing a ratio of sample size to the search space size of just 0.75%. The ratio for M=30, N=10 is even lower at 0.37% w.r.t. $\sim 10^{12}$ possible assignments. As far as we know, this is the first study of reasoning in neurosymbolic systems to produce results with such low ratios.

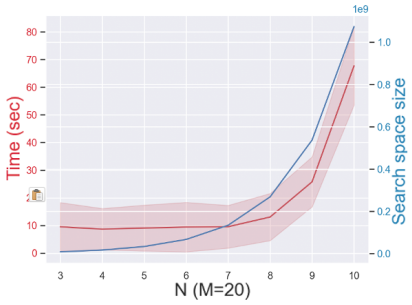
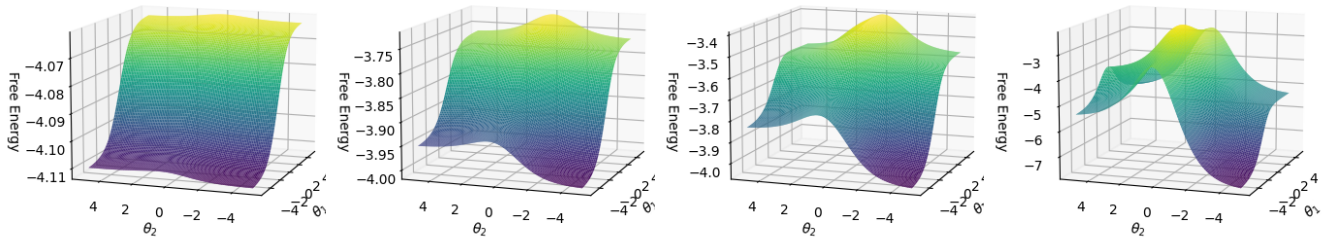


Figure 3: Time taken by LBM to collect all satisfying assignments compared with the size of the search space.

Figure 3 shows the time needed to collect all satisfying assignments for different N in $\{3, 4, 5, 6, 7, 8, 9, 10\}$ with $M = 20$. LBM only needs around 10 seconds for $N <= 8$, $\sim$ 25 seconds for $N = 9$, and $\sim$ 68 seconds for $N = 10$. The curve grows exponentially, similarly to the search space size, but at a much lower scale.

Towards LBM as a SAT Solver

Boolean satisfiability is a long standing challenge in computer science. While symbolic SAT solvers are considerably more effective, solving SAT problems is still a challenge for connectionist networks. Early attempts have modified Boltzmann machines by adding configurations to the energy func-



(a) c=0.1 (b) c=0.5 (c) c=1 (d) c=5

Figure 4: Free energy function with different *confidence values*.

tion. Compared to their dense, higher-order structure, LBM is much simpler. Recent deep learning methods used data generated from SAT-solvers to train classification models (Selsam et al. 2019; Wang et al. 2019). LBM is different in that it only needs to convert the SAT problems into the RBMs without grounding samples for training.

Formulae in SAT problems are represented in CNF. As discussed earlier, CNF can be encoded into LBM where the number of satisfied clauses will be proportional to the minimised energy and also to the free-energy. Therefore, we solve a SAT problem by searching for a minimum of the free-energy function, thus converting the SAT problem into a continuous optimisation problem. Instead of searching in a Boolean space for $x \in \{0, 1\}$ to minimise the free energy function, we search in a continuous space for θ where $x = \sigma(\theta) = 1/(1 + \exp(-\theta))$. In the following example, we show the landscape of the LBM for a SAT problem with 2 variables $((\neg \mathrm{x}_1 \vee \neg \mathrm{x}_2) \wedge (\mathrm{x}_1 \vee \neg \mathrm{x}_2) \wedge (\neg \mathrm{x}_1 \vee \mathrm{x}_2))$. Figure 4 shows the free-energy function for different values of θs ($x_1 = \sigma(\theta_1)$, $x_2 = \sigma(\theta_2)$). When both θ_1 and θ_2 are more negative (x_1, $x_2 \sim 0$), the function is approaching its minimum. This is also a satisfying assignment of the CNF. For all the free-energy functions (Figures 4a, 4b, 4c, 4d), a smaller c would make the landscape smoother but it also narrows the gap between local minima and global minima. Higher values for c would raise the boundaries between the optimal areas, making the global optima search harder.

We use random SAT (Amizadeh, Matusevych, and Weimer 2019) as a case study and employ off-the-shelf methods from the optimisation library in SciPy (https://scipy.org/). Among all the available methods, *dual_annealing* and *differential_evolution* can solve SAT problems with up to ∼200 variables using LBM. Although this is much inferior to symbolic SAT solvers, it is worth noting that LBM does not have any prior knowledge about the particular SAT problem. As future work, inference using LBM can be improved significantly by integrating backtracking tactics from symbolic solvers with the differentiable optimisation in LBM.

Learning from Data and Knowledge

We now evaluate LBM at learning the same Inductive Logic Programming (ILP) benchmark tasks used by neurosymbolic system CILP++ (França, Zaverucha, and Garcez 2014) in comparison with ILP state-of-the-art system Aleph (Srinivasan 2007). As mentioned earlier, the systems Aleph, CILP++ and a fully-connected standard RBM were chosen as the natural symbolic, neurosymbolic and neural system, respectively, for comparison. An initial LBM is constructed from the clauses provided as background knowledge. This process creates one hidden neuron per clause. Further hidden neurons are added using random weights for training and validation from data. Satisfying assignments can be selected from each clause as a training or validation example, for instance given clause $y \leftarrow \mathrm{x}_1 \wedge \neg \mathrm{x}_2$, assignment $y = true, x_1 = true, x_2 = false$ is converted into vector $[x_1, x_2, y] = (1, 0, 1)$ for training. Both the LBM and the standard RBM are trained discriminatively using the conditional distribution $p(y|\mathbf{x})$ for inference (Larochelle et al. 2012). In both cases, all network weights are free parameters for learning, with some weights having been initialized by the background knowledge in the case of the LBM, such that the background knowledge can be revised given data.

We carry out experiments on 7 data sets with available data and background knowledge (BK): Mutagenesis (examples of molecules tested for mutagenicity and BK provided in the form of rules describing relationships between atom bonds) (Srinivasan et al. 1994), KRK (King-Rook versus King chess endgame with examples provided by the coordinates of the pieces on the board and BK in the form of row and column differences) (Bain and Muggleton 1995), UW-CSE (Entity-Relationship diagram with data about students, courses taken, professors, etc. and BK describing the relational structure) (Richardson and Domingos 2006), and the Alzheimer's benchmark: Amine, Acetyl, Memory and Toxic (a set of examples for each of four properties of a drug design for Alzheimer's disease with BK describing bonds between the chemical structures) (King, Sternberg, and Srinivasan 1995). With the clauses converted into their equivalent set of preferred models, i.e. vectors, and combined with the available data, for Mutagenesis and KRK, 2.5% of the data is used to build the initial LBM. For the larger data sets UW-CSE and Alzheimer's, 10% of the data is used as BK. The remaining data are used for training and validation based on 10-fold cross validation for each data set, except for UW-CSE which uses 5 folds (for the sake of comparison). Re-

sults are shown in Table 1. It can be seen that LBM has the best performance in 5 out of 7 data sets. Some of the results of the LBM and RBM are comparable when the BK can be learned from the examples, as in the case of the Alzheimer's amine data set. Aleph is better than all other models in the *alz-acetyl* data set. This task probably relies more heavily on the correctness of the BK than the data.

	Aleph	CILP++	RBM	LBM
Mutagenesis	80.85 ±10.5	91.70 ±5.84	95.55 ±1.36	**96.28** ±1.21
KRK	99.60 ±0.51	98.42 ±1.26	99.70 ±0.11	**99.80** ±0.09
UW-CSE	84.91 ±7.32	70.01 ±2.2	89.14 ±0.46	**89.43** ±0.42
alz-amine	78.71 ±5.25	78.99 ±4.46	**79.13** ±1.14	78.25 ±1.07
alz-acetyl	**69.46** ±3.6	65.47 ±2.43	62.93 ±0.31	66.82 ±0.28
alz-memory	68.57 ±5.7	60.44 ±4.11	68.54 ±0.97	**71.84** ±0.88
alz-toxic	80.50 ±3.98	81.73 ±4.68	82.71 ±1.18	**84.95** ±1.04

Table 1: Cross-validation performance of LBM against purely-symbolic system Aleph, neurosymbolic system CILP++ and a standard RBM on 7 benchmark data sets for neurosymbolic AI. We run cross-validation on RBM and LBM 100 times and report the average results with 95% confidence interval.

Integration of Learning and Reasoning

Finally, we integrate LBM as a logical layer on top of deep networks applied to a semantic image interpretation task: to predict the relations between objects and their parts in images. Our knowledge base consists of symbolic facts such as "an object type is part of another object type", e.g. pt("screen", "tvmonitor") (a tv screen is part (pt) of a tv monitor), and a formula for the part-of relation (po):
$(\mathrm{po}(X_1, X_2) \leftrightarrow \mathrm{pt}(T_1, T_2)) \leftarrow \mathrm{type}(X_1, T_1) \wedge \mathrm{type}(X_2, T_2)$
where X_1, X_2 are real-valued variables representing visual features of objects, and T_1, T_2 are symbolic variables representing object types. Predicate type is $true$ when a visual object is of a given type. Predicate pt is $true$ when a type is part of another type. Predicate po states that an object is part of another object. The intended meaning of the formula is that *"if object 1 has type 1 and object 2 has type 2 then the part-of relation between the two objects is the same as the relation between the two object types"*.

We characterise the predicates as functions from a set element to a truth value. In particular, we use faster RCNN to extract features from object images, from which we build two Neural Network Regressors (NNR) $\mathcal{N}^{\mathrm{type}}$ and $\mathcal{N}^{\mathrm{po}}$ as functions for type and po respectively, as done in (Donadello, Serafini, and d'Avila Garcez 2017). We use an autoencoder $\mathcal{N}^{\mathrm{pt}}$ for the symbolic facts as a function for pt, as done in (Tran 2021). We replace the predicates in the above formula by corresponding propositions with values $true/false$ obtained from the functions, and apply Eq. (3) to convert the formula to SDNF:

$$(\mathrm{p}^{\mathrm{po}} \leftrightarrow \mathrm{p}^{\mathrm{pt}}) \leftarrow (\mathrm{p}^{\mathrm{t}_1} \wedge \mathrm{p}^{\mathrm{t}_2}) \equiv (\mathrm{p}^{\mathrm{po}} \wedge \mathrm{p}^{\mathrm{pt}} \wedge \mathrm{p}^{\mathrm{t}_1} \wedge \mathrm{p}^{\mathrm{t}_2}) \vee (\neg \mathrm{p}^{\mathrm{po}} \wedge \neg \mathrm{p}^{\mathrm{pt}} \wedge \mathrm{p}^{\mathrm{t}_1} \wedge \mathrm{p}^{\mathrm{t}_2}) \vee (\neg \mathrm{p}^{\mathrm{t}_1} \wedge \mathrm{p}^{\mathrm{t}_2}) \vee \neg \mathrm{p}^{\mathrm{t}_2}$$

where $\mathrm{p}^{\mathrm{po}} = \mathcal{N}^{\mathrm{po}}(X_1, X_2)$, $\mathrm{p}^{\mathrm{pt}} = \mathcal{N}^{\mathrm{pt}}(T_1, T_2)$, $\mathrm{p}^{\mathrm{t}_1} = \mathcal{N}^{\mathrm{type}}(X_1, T_1)$, $\mathrm{p}^{\mathrm{t}_2} = \mathcal{N}^{\mathrm{type}}(X_2, T_2)$. From this SDNF, we build a LBM as the *logical layer* on top of the neural networks. By using LBM we take advantage of its reasoning capability during learning by backpropagating inferred knowledge to update the functions. In particular, we train the entire system by minimising the following cost function:

$$||\mathcal{N}^{\mathrm{po}}(x_1, x_2) - \mathrm{LBM}(\mathrm{p}^{\mathrm{po}}|\mathcal{K}(x_1, x_2)||_2^2 + ||[\mathcal{N}^{\mathrm{type}}(x_1, t_1), \mathcal{N}^{\mathrm{type}}(x_2, t_2)] - \mathrm{LBM}(\mathrm{p}^{\mathrm{t}_1}, \mathrm{p}^{\mathrm{t}_2}|\mathcal{K}(x_1, x_2))||_2^2$$

where $x_1, x_2, \mathcal{K}(x_1, x_2)$ are drawn from the training data. $\mathcal{K}(x_1, x_2)$ is the knowledge involving x_1, x_2, i.e. the types of x_1, x_2, and whether x_1 is part of x_2. We use $\mathrm{LBM}(\mathrm{p}^{\mathrm{po}}|\mathcal{K}(x_1, x_2))$ and $\mathrm{LBM}(\mathrm{p}^{\mathrm{t}_1}, \mathrm{p}^{\mathrm{t}_2}|\mathcal{K}(x_1, x_2))$ to denote the application of LBM to infer p^{po} and the pair $[\mathrm{p}^{\mathrm{t}_1}, \mathrm{p}^{\mathrm{t}_2}]$, respectively. The first optimisation term above leverages Modus-Ponens reasoning from the LBM to infer p^{po} and update $\mathcal{N}^{\mathrm{po}}$. For example, given $x_1 =$  , $x_2 =$  , and assume we do not know if x_1 is part of x_2, but if we draw from the data and knowledge base that $\mathcal{K}(x_1, x_2) = \{\mathrm{type}(x_1, \text{"screen"}) \equiv true, \mathrm{type}(x_2, \text{"tvmonitor"}) \equiv true\}$ then the LBM can infer that p^{po} is $true$ (because "screen" is part of "tvmonitor") and update $\mathcal{N}^{\mathrm{po}}$. Similarly, the second term leverages Modus-Tollens to use the type of objects for updating $\mathcal{N}^{\mathrm{type}}$.

We compare this LBM-based model with three neurosymbolic systems, including DLN (Tran and Garcez 2018), LTN (Donadello, Serafini, and d'Avila Garcez 2017; Badreddine et al. 2022) and CNLP (Tran 2021). The data set for this experiment is the same as in (Donadello, Serafini, and d'Avila Garcez 2017) with the exception of the above formula for po. The Area Under the Curve (AUC) results in Table 2 show the effectiveness of the LBM-based model with higher performance in the prediction of the part-of relation as a result of the end-to-end learning and reasoning. For the object type prediction, the RBM-based model is comparable to CNLP and better than DLN and LTN.

	Object type (AUC)	Part-of (AUC)
DLN	0.791 ± 0.032	0.605 ± 0.024
CNLP	0.816 ± 0.004	$\mathbf{0.644 \pm 0.015}$
LTN	0.800	0.598
LBM-based model	$\mathbf{0.828 \pm 0.002}$	$\mathbf{0.645 \pm 0.027}$

Table 2: AUC for semantic image interpretation.

Conclusion and Future Work

We introduced an approach and neurosymbolic system for reasoning with knowledge in energy-based neural networks. We showed equivalence between propositional logic and RBMs. The findings led to a system, named Logical Boltzmann Machines, integrating learning and reasoning in neural networks with improved performance. Future work will focus on scaling up applications to SAT and end-to-end learning and reasoning. Extensions include the use of probabilistic programming with weighted clauses in comparison with our confidence values in probabilistic learning tasks.

References

Amizadeh, S.; Matusevych, S.; and Weimer, M. 2019. Learning To Solve Circuit-SAT: An Unsupervised Differentiable Approach. In *ICLR*.

Badreddine, S.; d'Avila Garcez, A.; Serafini, L.; and Spranger, M. 2022. Logic Tensor Networks. *Artificial Intelligence*, 303: 103649.

Bain, M.; and Muggleton, S. 1995. Machine Intelligence 13. chapter Learning Optimal Chess Strategies, 291–309. New York, NY, USA: Oxford University Press, Inc. ISBN 0-19-853850-2.

Cohen, W. W.; Yang, F.; and Mazaitis, K. 2017. TensorLog: Deep Learning Meets Probabilistic DBs. *CoRR*, abs/1707.05390.

d'Anjou, A.; Graña, M.; Torrealdea, F. J.; and Hernandez, M. C. 1993. Solving Satisfiability Via Boltzmann Machines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5): 514–521.

d'Avila Garcez, A.; and Lamb, L. C. 2020. Neurosymbolic AI: The 3rd Wave. arXiv:2012.05876.

Donadello, I.; Serafini, L.; and d'Avila Garcez, A. S. 2017. Logic Tensor Networks for Semantic Image Interpretation. In *IJCAI-17*, 1596–1602.

Evans, R.; and Grefenstette, E. 2018. Learning Explanatory Rules from Noisy Data. *JAIR*, 61: 1–64.

França, M.; Zaverucha, G.; and Garcez, A. 2014. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Mach. Learning*, 94(1): 81–104.

Garcez, A.; Broda, K.; and Gabbay, D. 2001. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artif. Intel.*, 125(1–2): 155–207.

Hernandez, C.; Albizuri, F.; DAnjou, A.; Graña, M.; and Torrealdea, F. 2001. EFFICIENT SOLUTION OF MAX-SAT AND SAT VIA HIGHER ORDER BOLTZMANN. *Revista Investigación Operacional*, 22.

King, R. D.; Sternberg, M. J. E.; and Srinivasan, A. 1995. Relating chemical activity to structure: An examination of ILP successes. *New Generation Computing*, 13(3).

Larochelle, H.; Mandel, M.; Pascanu, R.; and Bengio, Y. 2012. Learning Algorithms for the Classification Restricted Boltzmann Machine. *J. Mach. Learn. Res.*, 13(1): 643–669.

Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2018. DeepProbLog: Neural Probabilistic Logic Programming. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 3749–3759. Curran Associates, Inc.

Penning, L. d.; Garcez, A. d.; Lamb, L.; and Meyer, J.-J. 2011. A Neural-Symbolic Cognitive Agent for Online Learning and Reasoning. In *IJCAI*, 1653–1658.

Pinkas, G. 1991. Symmetric Neural Networks and Propositional Logic Satisfiability. *Neural Comput.*, 3(2): 282–291.

Pinkas, G. 1995. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artif. Intell.*, 77(2): 203–247.

Richardson, M.; and Domingos, P. 2006. Markov logic networks. *Mach. Learn.*, 62(1-2): 107–136.

Selsam, D.; Lamm, M.; Bünz, B.; Liang, P.; de Moura, L.; and Dill, D. L. 2019. Learning a SAT Solver from Single-Bit Supervision. In *International Conference on Learning Representations*.

Serafini, L.; and d'Avila Garcez, A. S. 2016. Learning and Reasoning with Logic Tensor Networks. In *AI*IA*, 334–348. ISBN 978-3-319-49129-5.

Smolensky, P. 1995. Constituent Structure and Explanation in an Integrated Connectionist/Symbolic Cognitive Architecture. In *Connectionism: Debates on Psychological Explanation*.

Srinivasan, A. 2007. The Aleph Manual. http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html. Accessed: 2021-01-23.

Srinivasan, A.; Muggleton, S. H.; King, R.; and Sternberg, M. 1994. Mutagenesis: ILP experiments in a nondeterminate biological domain. In *Proceedings of the 4th International Workshop on Inductive Logic Programming, volume 237 of GMD-Studien*, 217–232.

Towell, G.; and Shavlik, J. 1994. Knowledge-Based Artificial Neural Networks. *Artif. Intel.*, 70: 119–165.

Tran, S.; and Garcez, A. 2018. Deep Logic Networks: Inserting and Extracting Knowledge From Deep Belief Networks. *IEEE T. Neur. Net. Learning Syst.*, (29): 246–258.

Tran, S. N. 2021. Compositional Neural Logic Programming. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 3059–3066. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Wang, P.; Donti, P. L.; Wilder, B.; and Kolter, J. Z. 2019. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. *CoRR*, abs/1905.12149.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 2319–2328. Curran Associates, Inc.