

Deep Learning with Logical Constraints

Eleonora Giunchiglia¹, Mihaela Catalina Stoian¹ and Thomas Lukasiewicz^{2,1}

¹ Department of Computer Science, University of Oxford, UK

² Institute of Logic and Computation, TU Wien, Austria

eleonora.giunchiglia@cs.ox.ac.uk, mihaela.stoian@cs.ox.ac.uk, thomas.lukasiewicz@cs.ox.ac.uk

Abstract

In recent years, there has been an increasing interest in exploiting logically specified background knowledge in order to obtain neural models (i) with a better performance, (ii) able to learn from less data, and/or (iii) guaranteed to be compliant with the background knowledge itself, e.g., for safety-critical applications. In this survey, we retrace such works and categorize them based on (i) the logical language that they use to express the background knowledge and (ii) the goals that they achieve.

1 Introduction

The recent history of deep learning is a story of successes (see, e.g., [Senior *et al.*, 2020]). Neural networks have been increasingly applied to solve everyday problems, often successfully, but sometimes also showing their potential flaws (see, e.g., [Wexler, 2017]). A major source of such shortcomings is that neural networks are still domain-agnostic in most cases, as they often ignore domain knowledge about the problem at hand [Stevens *et al.*, 2020]. For this reason, in recent years, there has been an increasing interest in incorporating background knowledge into deep learning algorithms. Such background knowledge can be expressed in many different ways (e.g., algebraic equations, logical constraints, and natural language) and incorporated in neural networks (i) to improve their performance (see, e.g., [Li and Srikumar, 2019]), (ii) to make them able to learn from less data (see, e.g., [Xu *et al.*, 2018]), and/or (iii) to guarantee that their behavior is compliant with the background knowledge itself (see, e.g., [Hoernle *et al.*, 2022]). We remark the importance of the last point when considering safety-critical applications in which the background knowledge corresponds to requirements on the models. Indeed, we envision that, in the future, requirement specification will become a standard step in the development of machine learning models, as it is in any software development process.

In this survey, we conduct a comprehensive and fine-grained analysis of the works in which background knowledge is expressed as constraints in a logic-based language and then exploited to obtain better models. We organize the papers

into four macro-categories based on the richness of the language that they use to express the constraints. Furthermore, to maximize the usefulness of this survey, we take a problem-oriented approach, and for each considered paper, we report which shortcomings the authors aim to address with the inclusion of the logical background knowledge. A summary of our categorization is given in Table 2.

The rest of this survey is organized as follows. In Section 2, we first give a formal definition of the problem of learning with logical constraints. Sections 3 to 6 then describe the models that belong to each macro-category, starting from the one with the simplest corresponding language, up to the one with the richest. In Section 7, we end the paper with some concluding remarks and pointers to other related surveys.

2 Learning with Logical Constraints

We formalize the problem of *learning with logical constraints* as a triple $\mathcal{P} = (\mathcal{C}, \mathcal{X}, \Pi)$:

1. \mathcal{C} is a pair $(\mathcal{I}, \mathcal{O})$, where $\mathcal{I} = I_1, I_2, \dots, I_d$ ($d \geq 1$) are the *input features*, and $\mathcal{O} = O_1, O_2, \dots, O_n$ ($n \geq 1$) are the *outputs*. Each input feature I (resp., output O) is associated with a non-empty domain D_I (resp., D_O) of *values*, and I (resp., O) is *Boolean* when $D_I = \{0, 1\}$ (resp., $D_O = \{0, 1\}$). $D_{\mathcal{I}} = D_{I_1} \times \dots \times D_{I_d}$ (resp., $D_{\mathcal{O}} = D_{O_1} \times \dots \times D_{O_n}$) is the set of the possible inputs (resp., outputs). A *data point* is an element of $D_{\mathcal{I}}$.
2. \mathcal{X} is a finite set of pairs (x, y) , where x is a (possibly partially specified) data point, and y is its (possibly partially specified) *ground truth*.
3. Π is a finite non-empty set of first-order constraints, which delimit the set of outputs that can be meaningfully associated with each data point. We assume that the constraints are written using (i) for each Boolean (resp., non-Boolean) input feature I a corresponding d -ary predicate (resp., function) I : intuitively, $I(x)$ (resp., $I(x) = z$) means that the value of the input feature I in the data point x is 1 (resp., z), and analogously for each Boolean (resp., non-Boolean) output O , (ii) variables ranging over data points, (iii) specific values, and (iv) possibly other logical symbols (like “ \neg ”, “ \wedge ”, “ \vee ”, “ \rightarrow ”, “ \forall ”, “ \exists ”, “ $=$ ”, “ \geq ”, and “ $+$ ”). To simplify the

formal treatment and save space, we assume that each constraint is closed and in prenex form (i.e., that each variable is either existentially or universally quantified at the beginning of the constraint).

Notice that our definitions of both \mathcal{X} and Π are very general. This is necessary (i) for \mathcal{X} , to cover papers dealing with supervised and semisupervised problems, and (ii) for Π , given that each paper uses its own specific language to model the learning problem and exploit the background knowledge. For example, consider the classification problem associated with CIFAR-100 [Krizhevsky, 2009]. If we associate with each class and superclass a separate Boolean output, it is possible to formalize the knowledge that sharks and trouts are fish via

$$\forall x.(Shark(x) \rightarrow Fish(x)), \quad \forall x.(Trout(x) \rightarrow Fish(x)), \quad (1)$$

and that sharks are not trouts via

$$\forall x.(Shark(x) \rightarrow \neg Trout(x)). \quad (2)$$

In the above formulas, *Shark*, *Fish*, and *Trout* are predicates, each corresponding to a Boolean output of the network. Notice that the above constraints can be equivalently written in many different ways, exploiting well-known first-order and propositional logic equivalences. Still, in the literature, constraints are often written and handled as *rules*, i.e., formulas like (1) and (2), having one of the following two forms

$$\forall x.(F(x) \rightarrow A(x)), \quad \forall x.(F(x) \rightarrow \neg A(x)), \quad (3)$$

where $F(x)$ is a conjunction of *literals* (i.e., atomic formulas and negations of atomic formulas, where a formula is *atomic* if it does not contain propositional connectives and/or quantifiers), and A is a predicate associated with one Boolean output. The motivation of introducing rules as a special case of formulas is based on the fact that most of the works express background knowledge as rules of the form (3), and then, for each data point x , use the value computed for $F(x)$ to determine the value to be associated with $A(x)$. Furthermore, in the literature, there has been a special focus on basic rules in which also F (like A) is a predicate associated with a Boolean output. Rules in (1) and (2) are basic.

We first survey papers in which constraints are basic rules (Section 3) and then the ones allowing for general rules (Section 4). All the other surveyed papers are then divided into two categories, depending on whether they allow only for universal quantification (Section 5) or also for existential quantification (Section 6) in the constraints. Thus, the fragment of first-order logic considered in each section allows for a richer language than the one allowed in the preceding sections.

3 Basic Rules

Basic rules are expressions of the form (3) where both F and A are predicates corresponding to Boolean outputs. They have been used in the context of multi-label classification (MC) problems with constraints, being the special case of learning with constraints in which all the outputs are Boolean labels. Thus, each output has a corresponding associated predicate, and the rules specify the existing relations between the output labels. All reviewed papers use rules of the first

form in (3) to express a hierarchical relation between two output labels, and some papers use also rules of the second form in (3) to express a mutual exclusion between two output labels. Thus, we further divide the papers depending on their usage of basic rules of the second form in (3).

Hierarchical rules. In learning problems with hierarchical rules, knowledge is expressed with basic rules of the first form in (3) in which also F (like A) is a Boolean output. Examples of hierarchical rules are (1), which are satisfied whenever a data point x predicted to be a shark or a trout is also predicted to be a fish. Interestingly, an entire field has been developed to deal with such rules, namely, *hierarchical multi-label classification* (HMC) problems, which are MC problems with hierarchical rules satisfying the additional assumption that there are no cycles between the labels, when drawing an arc from the antecedent to the consequent of each rule (see, e.g., [Cerri *et al.*, 2011]). HMC problems naturally arise in many different domains, such as image classification or functional genomics, and HMC models have normally two goals: (i) improve on the SOTA models, and (ii) guarantee the satisfaction of the hierarchical constraints. As expected, many different neural models were developed for HMC problems. To present them, we follow the classic categorization used for general HMC models, which divides them into two groups based on how they exploit the hierarchical knowledge [Silla and Freitas, 2011]:

1. *local approaches* exploit the constraints to decompose the problem into smaller classification ones, and then combine the solutions appropriately, while
2. *global approaches* consist of single models able to associate objects with their corresponding classes in the hierarchy as a whole.

Local approaches can be further divided into three subcategories based on the strategy that they deploy to decompose the main task. The most popular strategy is *local classifier per level*, which is given when a method trains a different classifier for each level of the hierarchy. For example, HMC-LMLP [Cerri *et al.*, 2011; Cerri *et al.*, 2014] is a model consisting of a multi-layer perceptron (MLP) per hierarchical level, and the predictions in one level are used as inputs to the network responsible for the predictions in the next level. This model was later extended in [Wehrmann *et al.*, 2018], where HMCN is proposed. HMCN is considered a hybrid model, because it is trained with both a local and a global loss. Finally, DEEPre [Li *et al.*, 2018], later extended in mlDEEPre [Zou *et al.*, 2019], consists of a neural network for each level of the hierarchy, and it is applied to the challenging problem of enzyme function prediction. On the other hand, if a method trains a classifier for each node of the hierarchy, then we have a *local classifier per node*. An example model for this category is HMC-MLPN [Feng *et al.*, 2018], in which one MLP for each node is deployed. Finally, if a method trains a different classifier per parent node in the hierarchy, then we have a *local classifier per parent node*. For example, Kulmanov *et al.* [2018] propose DeepGO, in which a small neural-based model for each subontology of the Gene Ontology is trained. Global methods, on the other hand, do not have any subclassification, and global neural models are quite recent. The first proposed

global model based on neural networks is AWX [Masera and Blanzieri, 2018], which is just a feed-forward neural network predicting the leaves of the hierarchy, and then inferring the value for the parent nodes. A more complex model is given by C-HMCNN [Giunchiglia and Lukasiewicz, 2020], which builds a constraint layer ensuring the satisfaction of the constraints. Such a layer works synergistically with a constraint loss to exploit the background knowledge of the hierarchy. Finally, MBM [Patel *et al.*, 2022] takes an alternative approach by representing the labels by boxes rather than vectors, and thus it is able to capture taxonomic relations among labels.

Hierarchical and exclusion rules. In computer vision, hierarchical rules are often used together with exclusion constraints expressed as basic rules of the second form in (3) in which F (like A) is a Boolean output. An example of a basic rule corresponding to an exclusion constraint is (2), which is satisfied whenever a data point x predicted to be a shark cannot be predicted to also be a trout. Hierarchical and mutual exclusion rules can be represented together in a hierarchy and exclusion (HEX) graph, firstly proposed in [Deng *et al.*, 2014]. An example of a HEX graph is given in Figure 1. In addition to proposing HEX graphs, Deng *et al.* also build a neural model guaranteed to satisfy the rules and able to deal with incomplete labels. HEX graphs are often applied to tackle the problem of fine-grained image classification. Fine-grained image classification refers to the problem of classifying images that possess very subtle discriminatory features, e.g., classifying images of different birds by species, or of flowers by categories. Their usage in the field was firstly proposed in [Xie *et al.*, 2015], where the authors tackle the problem by adding a set of classes representing broader concepts than the initial ones (all the newly added classes are thus parent nodes in the hierarchy), and then acquiring a large number of images from external sources, which are labelled with such newly added classes. The new task is thus to predict both the initial label and the newly annotated labels. The authors show how, due to the added data, they achieve better results on this new task. An even more interesting study is shown in [Chang *et al.*, 2021], where the authors show that it is possible to exploit the background knowledge to build a model that is able to get better results than the SOTA models on the initial set of classes. Chang *et al.*'s results have been recently surpassed by HRN [Chen *et al.*, 2022], a network able to perform hierarchical feature interaction via residual connections.

4 General Rules

In this section, we lift the assumption that $F(x)$ in rules of the type (3) is a single literal, and we review the papers in which $F(x)$ is allowed to be a conjunction of m literals ($m \geq 1$). We classify such papers depending on whether $F(x)$ can (or cannot) contain functions and/or predicates corresponding to input features. In the first case, given a data point x , the rules constrain the output label A also on the basis of the input x , and thus each rule corresponds to an input-output constraint. In the second case, each rule excludes some output configurations independently from the data point x .

Input-output constraints. Interestingly, this type of constraints was the very first to be studied in this field. This

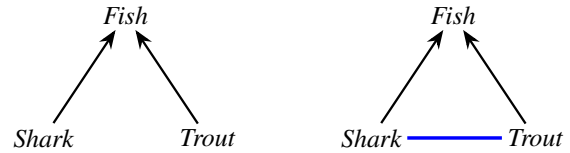


Figure 1: Example of a hierarchy DAG (left) and of a hierarchy and exclusion graph (right). Here, “ \rightarrow ” indicates a hierarchical relation, while “ --- ” indicates mutual exclusion.

is probably due to the fact that before the spread of machine learning systems, many classifiers were still hand-built, and thus researchers had a large availability of rules of the type “if the input has these features, then the data point belongs to this class”. The goal of these models was thus to overcome the flaws of both hand-built classifiers and learned classifiers (which could be seen as almost complementary flaws) by creating hybrid systems. One of the first works able to incorporate such rules in the topology of the neural network was KBANN [Shavlik and Towell, 1989; Towell and Shavlik, 1994]. Given a set of rules, Shavlik and Towell [1989] map the supporting facts to the input neurons, the intermediate conclusions to the hidden neurons, and the final conclusions to the output neurons. Given such a mapping, clearly, KBANN needs the additional assumption that the constraints are acyclic. KBANN was later extended by Fu [1993], who proposed KBCNN. Given a set of acyclic rules, not only KBCNN is able to map the rules into a neural network like KBANN, but also it is able to map back the neural network to a set of learned rules. Such learned rules have the advantage of being completely transparent to the human user. While both KBANN and KBCNN build a neural network directly from the rules, in [Fletcher and Obradovic, 1993], the rules are treated as an expert system that is refined through incremental hidden unit generation, which allows the model to learn new rules without corrupting the initial ones (as often happens in KBANN), and to get a good performance even with severely incomplete rule bases (contrarily to KBANN). Another model proposed to solve such problems is Cascade ARTMAP [Tan, 1997], which is able to learn a set of rules that are more accurate and simpler than the ones extracted from KBANN. While all the methods explored so far can only deal with acyclic rules, one of the first methods able to deal with cyclic rules was CIL²P [d’Avila Garcez and Zaverucha, 1999]. CIL²P and KBANN share the same high-level learning algorithm, however, they map the rules into neural networks in different ways. Due to such a different mapping, not only d’Avila Garcez and Zaverucha can map cyclic rules, but they are also able to prove that CIL²P computes the stable model of the logic program represented by the neural network, thus making it a parallel system for logic programming. Notice that CIL²P was later extended in [França *et al.*, 2014] to represent and learn first-order logic formulas. This new system is called CILP++. Since KBANN and CIL²P are probably the most well-known early models able to combine logic and neural networks, we provide a more detailed comparison between them in Table 1. More details on the approaches from the 90s to combine knowledge engineering and machine learning are given in the surveys [Shavlik, 1994; Mooney and Shavlik, 2021].

Example Rules	Common High-Level Algorithm	KBANN Mapping	CIL ² P Mapping
$\forall x.(A_1(x) \wedge \neg A_2(x) \rightarrow A_4(x))$ $\forall x.(A_3(x) \wedge A_4(x) \rightarrow A_5(x))$ $\forall x.(A_5(x) \rightarrow A_4(x))$	<ol style="list-style-type: none"> 1. Map rules to the network. 2. Add n and m neurons to the input and hidden layers, respectively. n, m are user-defined parameters. 3. Fully connect the newly added neurons to the adjacent layers. 4. Perturb all the network’s weights. 		

Table 1: KBANN and CIL²P comparison. Given the sample rules (first column), we provide the high-level algorithm that the two models share to build the networks from the rules (second column), and then we show how the rule mapping (Step 1) is done in KBANN (third column) and CIL²P (fourth column), respectively. We highlighted $\forall x.(A_5(x) \rightarrow A_4(x))$, because while CIL²P can map it in the network, KBANN must exclude it in order to maintain the acyclicity of the rules ($\forall x.(A_5(x) \rightarrow A_4(x))$ is indeed absent in KBANN).

Constraints over the output. While all these early models focus on input-output constraints, the more recent ones focus on constraints over the output domain. From a high-level perspective, we can further divide these models based on how they integrate the neural network with the constraints. In the first set of models, learning and reasoning represent two different steps in a pipeline, which is though trained end-to-end. On the other hand, in the second set of models, the constraints are directly integrated in the network structure and/or loss function, thus we have “single-stage models”.

Pipelines: Probably one of the most famous methods able to constrain neural networks’ outputs is DeepProbLog [Manhaeve *et al.*, 2018]. DeepProbLog extends ProbLog [Raedt *et al.*, 2007] by encapsulating the outputs of the neural network in the form of neural predicates. This can be easily done, because (in ProbLog) atomic expressions are already assigned a probability. Furthermore, since the algebraic extension of ProbLog [Kimmig *et al.*, 2011] already supports automatic differentiation, it is possible to backpropagate the gradient from the loss at the output through the neural predicates into the neural networks, thus allowing the neural networks to learn directly from the logical constraints. DeepProbLog can thus be seen as a two-stage pipeline, where the neural networks handle the low-level perception, and then the reasoning is done at a logical level. Very close to DeepProbLog is NeurASP [Yang *et al.*, 2020], however, differently from DeepProbLog, NeurASP employs reasoning originating from answer set programming, such as defaults, aggregates, and optimization rules. A deeper pipeline, in which each stage models a sub-task of a more complex problem, has been proposed in [Sachan *et al.*, 2018], where the authors present Nuts&Bolts: a framework in which each stage can be a set of different function approximators, and that can be trained end-to-end to minimize a global loss function. Nuts&Bolts is able to exploit the background knowledge by expressing it as rules, then translating them into probabilistic soft logic and incorporating it as one of the function approximators. Notice that, in this case, the rules do not only express constraints over the final outputs of the pipeline, but also over the outputs of the intermediate stages.

Single-stage models: Among these models, we find CNN, a model proposed in [Giunchiglia and Lukasiewicz, 2021]. In this work, the authors present a way to map the constraints (i) into a top-layer (which can be built on top of any neural network), and (ii) into a loss function (which is able to

work synergistically with the layer to exploit the background knowledge and get better results). Thanks to the top layer, Giunchiglia and Lukasiewicz are able to guarantee that the constraints are always satisfied, however, they also make the assumption that the set of constraints must be stratified. Li and Srikumar [2019] also change the structure of neural networks to incorporate the background knowledge expressed by the constraints. To this end, the authors recognize that, in a neural network, some neurons (called *named neurons*) can be endowed with semantics tied to the task, and that logical rules can be written over such named neurons. Thanks to this particular formulation, Li and Srikumar are able to inject constraints over all the outputs of the neural network, even the intermediate ones (e.g., the outputs of the attention layer). The constraints that they are able to capture are thus over (i) the output domain, (ii) the intermediate outputs, and (iii) the relations between the intermediate outputs and the final outputs. Minervini and Riedel [2018], on the other hand, map the constraints to a loss function that is used to generate adversarial examples that cause a model to violate preexisting background knowledge. In particular, they apply their method to the task of recognizing textual entailment, and they add rules like “if sentence s_1 contradicts sentence s_2 , then s_2 contradicts s_1 as well”, ultimately showing that their models obtain a much better performance on adversarial datasets.

5 Universally Quantified Formulas

In this section, we describe methods that allow for constraints expressed as universally quantified first-order formulas. All the methods belonging to this category either inject the constraints in the loss (loss-based methods) or they constrain the value of the outputs (constrained-output methods).

Loss-based methods. While these methods obviously cannot guarantee the satisfaction of the constraints, they are able to generalize from less data and/or exploit unlabelled data. One of the first papers to propose to exploit background knowledge to solve the problem in data-scarce settings was [Stewart and Ermon, 2017]. In this paper, the authors show how it is possible to train neural networks to detect objects without any direct supervision, and by simply injecting in the loss the constraints on the output domain. In their work, they considered both constraints pertaining to dynamics equations and logical constraints. However, in each of the proposed examples, the authors had to manually engineer the loss func-

Model	Expressivity	Guaranteed Satisfaction	Less Data/Supervision	Improve on SOTA	
HMC-LMLP [Cerri <i>et al.</i> , 2011]		X	X	✓	
HEX [Deng <i>et al.</i> , 2014]		X	✓	✓	
HAR-CNN [Xie <i>et al.</i> , 2015]		✓	X	✓	
DEEPre [Li <i>et al.</i> , 2018]		✓	X	✓	
HMCN [Wehrmann <i>et al.</i> , 2018]	basic rules	X	X	✓	
AWX [Masera and Blanzieri, 2018]		✓	X	✓	
DeepGO [Kulmanov <i>et al.</i> , 2018]		✓	X	✓	
mlDEEPre [Zou <i>et al.</i> , 2019]		✓	X	✓	
C-HMCNN [Giunchiglia and Lukasiewicz, 2020]		✓	X	✓	
MBM [Patel <i>et al.</i> , 2022]		X	X	✓	
KBANN [Shavlik and Towell, 1989]			X	X	✓
KBCNN [Fu, 1993]			X	X	✓
Iterative Neurons Addition [Fletcher and Obradovic, 1993]		X	X	✓	
Cascade ARTMAP [Tan, 1997]		X	X	✓	
CIL ² P [d’Avila Garcez and Zaverucha, 1999]	general rules	X	X	✓	
Adversarial Regularisation [Minervini and Riedel, 2018]		X	X	✓	
DeepProbLog [Manhaeve <i>et al.</i> , 2018]		X	✓	X	
NeurASP [Yang <i>et al.</i> , 2020]		X	✓	X	
CCN [Giunchiglia and Lukasiewicz, 2021]		✓	X	✓	
Nuts&Bolts [Sachan <i>et al.</i> , 2018]		X	✓	✓	
Named Neurons [Li and Srikumar, 2019]		X	✓	✓	
Label-free Supervision [Stewart and Ermon, 2017]			X	✓	X
Semantic Loss [Xu <i>et al.</i> , 2018]		X	✓	X	
LENSR [Xie <i>et al.</i> , 2019]	universally quantified formulas	X	X	✓	
DL2 [Fischer <i>et al.</i> , 2019]		X	✓	X	
NESTER [Dragone <i>et al.</i> , 2021]		✓	✓	X	
MultiPlexNet [Hoernle <i>et al.</i> , 2022]		✓	✓	X	
SBR [Diligenti <i>et al.</i> , 2012]		X	✓	X	
CILP++ [França <i>et al.</i> , 2014]		X	X	✓	
LTN [Serafini and d’Avila Garcez, 2016]		X	X	✓	
Iterative Rule Distillation [Hu <i>et al.</i> , 2016a]	universally and existentially quantified formulas	X	✓	✓	
Mutual Iterative Rule Distillation [Hu <i>et al.</i> , 2016b]		X	✓	✓	
LTN-SII [Donadello <i>et al.</i> , 2017]		X	X	✓	
LYRICS [Marra <i>et al.</i> , 2019]		X	✓	X	
ABL [Dai <i>et al.</i> , 2019]		X	✓	X	
DFL [van Krieken <i>et al.</i> , 2020]		X	✓	X	

Table 2: Summary table of the analyzed works. For each work, we report which problem the authors wanted to tackle through the inclusion of the constraints. We identified three common problems tackled in the literature: (i) building models whose outputs are guaranteed to be compliant with a set of constraints, (ii) exploiting the background knowledge to get a better performance in either low-data regimen settings or to exploit unlabelled data, and (iii) beating SOTA models on benchmark datasets (Columns 3 to 5). If a model exploits the background knowledge for one of the three tasks above, then it is marked with “✓”, and with “X”, otherwise.

tion. After this work, a number of papers on how to automatically map universally quantified constraints into a loss function were studied. For example, in [Xu *et al.*, 2018], a *semantic loss* is proposed. Given a data point and a neural network that outputs the vector of probabilities \mathbf{p} , the semantic loss is defined such that it is proportional to the negative logarithm of the probability of generating an interpretation that satisfies the constraints, when sampling values according to \mathbf{p} . This method has the great quality of being completely syntax-independent (i.e., no matter how we write the constraints, the value of the loss does not change). Another model that incorporates the logical constraints in the loss is LENSr [Xie *et al.*, 2019]. However, in this model, the formulas are first rewritten in either conjunctive normal form or decision deterministic decomposable negation normal form, and then projected onto a manifold where entailment is related to distance. Once learned, the logic embeddings can then be used to form a logic loss that guides the neural network training by encouraging formula embeddings to be close to satisfying assignments, and far from unsatisfying assignments. Yet another method that translates the constraints in continuous terms to be added to the loss is DL2 [Fischer *et al.*, 2019]. What is particularly interesting about this method is that it allows for

queries. For example, a user can inquire the models on which neurons took part in a decision, or to generate adversarial examples violating a given constraint. Furthermore, notice that the language includes Boolean combinations of comparisons between terms, where a term is any real-valued function that may appear as a subexpression of a loss function.

Constrained-output methods. The methods presented here have the goal of guaranteeing the satisfaction of the constraints, and thus they apply the constraints directly on the outputs. MultiPlexNet [Hoernle *et al.*, 2022] further extends the language, as its constraints can consist of any quantifier-free linear arithmetic formula over the rationals (thus, involving “+”, “≥”, “−”, “^”, and “v”). In MultiPlexNet, the formulas are expressed in disjunctive normal form (DNF), and then the output layer of an existing neural network is augmented to include a separate transformation for each term in the DNF formula. Thus, the network’s output layer can be viewed as a multiplexer in a logical circuit that permits for a branching of logic, and exactly from this property, the model gets the name MultiPlexNet. Another work that limits the output space of the neural network is NESTER [Dragone *et al.*, 2021]. However, in this case, the constraints are not mapped into the last layer of the network (like MultiPlexNet

or CNN), but they are enforced by passing the outputs of the neural network to a constraint program, which enforces the constraints. NESTER can enforce hard and soft constraints over both categorical and numerical variables, and the entire architecture can be trained end-to-end by backpropagation.

6 Universally and Existentially Quantified Formulas

Our final category contains the models that allow for both universal and existential quantification. We divide the works between those that inject the constraints into the loss (loss-based methods) and those that create specialized neural structures to incorporate the constraints into the topology of the neural network (specialized structure-based methods).

Loss-based methods. One of the first models developed to this end was *semantic-based regularization* (SBR) [Diligenti *et al.*, 2012; Diligenti *et al.*, 2017]. Like the above loss-based methods, SBR was developed with the goal of exploiting unlabelled data to train machine learning models. In this work, the authors map the constraints to a differentiable regularization term that can be added to any loss. To this end, they perform the following steps: (i) the first-order logic (FOL) expressions are grounded, (ii) the quantifier-free formulas are mapped to real-valued functions using the t-norms [Klement *et al.*, 2000], and (iii) each FOL formula containing a universally (resp., existentially) quantified variable x is mapped to a function that returns the average (resp., maximum) of the t-norm generalization when grounding x over its domain. Other methods take inspiration from the distillation method proposed in [Hinton *et al.*, 2015], and have the goal of creating models able to both beat the SOTA models and exploit unlabelled data. In [Hu *et al.*, 2016a], the authors propose an *iterative rule knowledge distillation* procedure that transfers the structured information encoded in the logical rules into the network. In particular, at each iteration, a rule-regularized neural teacher is built, and then a student network is trained to imitate the predictions of the teacher network, while also trying to predict the right label for as many data points as possible. This work was later extended in [Hu *et al.*, 2016b]. This framework iteratively transfers information between the neural network and the structured knowledge, thus resulting in an effective integration of the representation learning capacity of neural networks and the generalization power of the structured knowledge. Different from distillation-based methods, but following a similar reasoning about how unifying two complementary paradigms can yield mutual benefits and surpass SOTA models, an abductive learning (ABL) approach is introduced in [Dai *et al.*, 2019]. ABL combines a neural network, responsible for interpreting subsymbolic data into primitive logical facts, with a logical component able to reason on these facts. If the generated facts are inconsistent with the background knowledge, ABL generates new pseudo-labels that more likely satisfy the background knowledge and retrains the neural network accordingly.

Specialized structure-based methods. Instead of mapping the constraints into the loss function, Serafini and d’Avila Garcez [2016] and Badreddine *et al.* [2022] map functions

and predicates to matrices, and constants to vectors. They propose real logic: a logical formalism on a first-order language whereby formulas have a truth value in the interval $[0,1]$ and a semantics defined concretely on the domain of real numbers. They then show how real logic can be encoded in logic tensor networks (LTNs). Donadello *et al.* [2017] later show how to use LTNs for semantic image interpretation, and Marra *et al.* [2019] present LYRICS, an extension of LTNs that provides an input language allowing for background knowledge in FOL, where predicate and function symbols are grounded onto any computational graph. All the logics in the works above belong to the family of differentiable fuzzy logics (DFL) of [van Krieken *et al.*, 2020; van Krieken *et al.*, 2022]. DFL is a family of differentiable logics, where the term *differentiable logics* refers to a logic along with a translation scheme from logical expressions to differentiable loss functions. So, differentiable fuzzy logics stand for the case where the logic is a fuzzy logic and the translation scheme applies to logical expressions that include fuzzy operators. Van Krieken *et al.* analyze DFL’s behavior over different choices of fuzzy logic operators and give some recommendations on the choice of the operators.

7 Summary and Outlook

We conducted a comprehensive and fine-grained analysis of deep learning approaches in which background knowledge is expressed and then exploited as logical constraints in first-order logic. We categorized the papers based on the richness of the logical language used to express the constraints. Furthermore, for each approach, we reported which shortcomings the authors wanted to address via the background knowledge. For coarser surveys on how to include background knowledge expressed in different ways, see [von Rueden *et al.*, 2021; Dash *et al.*, 2022]. All the presented models fall into the broader field of neural-symbolic computing, whose aim is to integrate the abilities of learning and of reasoning about what has been learned. For a broad survey on neural-symbolic models see [d’Avila Garcez *et al.*, 2019].

We envision that the specification and exploitation of logical constraints in deep learning models will become more and more widespread in the future, especially in safety-critical applications, where requirements are often formally specified as logical formulas, thus combining the advantages of manually engineering safety-critical features and automatically learning all other features. One open challenge for future work thus includes an AI system engineering approach that gives strict guarantees in the form of logical constraints on the system behavior based on such a combined system design.

Another open challenge for future research concerns the exploitation of logical constraints in explainable AI. More precisely, logical constraints and reasoning about the functionality of neural networks and their output data can actually be exploited in order to model the abstract reasoning that is underlying the predictions of these networks, and it can thus be used in order to naturally create explanations for these predictions. Only few existing papers, such as [Shi *et al.*, 2020; Majumder *et al.*, 2021], have started to explore this large potential of logical constraints in deep learning to date in different ways.

Acknowledgments

Eleonora Giunchiglia is supported by the EPSRC under the grant EP/N509711/1 and by an Oxford-DeepMind Graduate Scholarship. Mihaela Catalina Stoian is supported by the EPSRC under the grant EP/T517811/1. This work was also supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1, by the AXA Research Fund, and by the EPSRC grant EP/R013667/1.

References

- [Badreddine *et al.*, 2022] Samy Badreddine, Artur d’Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artif. Intell.*, 303, 2022.
- [Cerri *et al.*, 2011] Ricardo Cerri, Rodrigo Barros, and André Carlos Ponce de Leon Ferreira de Carvalho. Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks. In *Proc. of ISDA*, 2011.
- [Cerri *et al.*, 2014] Ricardo Cerri, Rodrigo Barros, and André Carlos Ponce de Leon Ferreira de Carvalho. Hierarchical multi-label classification using local neural networks. *J. Comput. Syst. Sci.*, 80, 2014.
- [Chang *et al.*, 2021] Dongliang Chang, Kaiyue Pang, Yixiao Zheng, Zhanyu Ma, Yi-Zhe Song, and Jun Guo. Your “flamingo” is my “bird”: Fine-grained, or not. In *Proc. of CVPR*, 2021.
- [Chen *et al.*, 2022] Jingzhou Chen, Peng Wang, Jian Liu, and Yuntao Qian. Label relation graphs enhanced hierarchical residual network for hierarchical multi-granularity classification. 2022. *arXiv:2201.03194*.
- [Dai *et al.*, 2019] Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. Bridging machine learning and logical reasoning by abductive learning. In *Proc. of NeurIPS*, 2019.
- [Dash *et al.*, 2022] Tirtharaj Dash, Sharad Chitlangia, Aditya Ahuja, and Ashwin Srinivasan. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Sci. Rep.*, 12, 2022.
- [d’Avila Garcez and Zaverucha, 1999] Artur d’Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intell.*, 1999.
- [d’Avila Garcez *et al.*, 2019] Artur d’Avila Garcez, Marco Gori, Luís Lamb, Luciano Serafini, Michael Spranger, and Son Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6, 2019.
- [Deng *et al.*, 2014] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *Proc. ECCV*, 2014.
- [Diligenti *et al.*, 2012] Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Mach. Learn.*, 86, 2012.
- [Diligenti *et al.*, 2017] Michelangelo Diligenti, Soumali Roychowdhury, and Marco Gori. Integrating prior knowledge into deep learning. In *Proc. of ICMLA*, 2017.
- [Donadello *et al.*, 2017] Ivan Donadello, Luciano Serafini, and Artur D’Avila Garcez. Logic tensor networks for semantic image interpretation. In *Proc. of IJCAI*, 2017.
- [Dragone *et al.*, 2021] Paolo Dragone, Stefano Teso, and Andrea Passerini. Neuro-symbolic constraint programming for structured prediction. In *Proc. of IJCLR-NeSy*, 2021.
- [Feng *et al.*, 2018] Shou Feng, Ping Fu, and Wenbin Zheng. A hierarchical multi-label classification method based on neural networks for gene function prediction. *Biotechnol. Biotechnol. Equip.*, 32, 2018.
- [Fischer *et al.*, 2019] Marc Fischer, Mislav Balunovic, Dana Drachler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. DL2: Training and querying neural networks with logic. In *Proc. of ICML*, 2019.
- [Fletcher and Obradovic, 1993] Justin Fletcher and Zoran Obradovic. Combining prior symbolic knowledge and constructive neural network learning. *Conn. Sci.*, 5, 1993.
- [França *et al.*, 2014] Manoel França, Gerson Zaverucha, and Artur d’Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Mach. Learn.*, 94, 2014.
- [Fu, 1993] Li Min Fu. Knowledge-based connectionism for revising domain theories. *IEEE TSMC*, 23, 1993.
- [Giunchiglia and Lukasiewicz, 2020] Eleonora Giunchiglia and Thomas Lukasiewicz. Coherent hierarchical multi-label classification networks. In *Proc. of NeurIPS*, 2020.
- [Giunchiglia and Lukasiewicz, 2021] Eleonora Giunchiglia and Thomas Lukasiewicz. Multi-label classification neural networks with hard logical constraints. *JAIR*, 72, 2021.
- [Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.
- [Hoernle *et al.*, 2022] Nicholas Hoernle, Rafael-Michael Karampatsis, Vaishak Belle, and Kobi Gal. MultiplexNet: Towards fully satisfied logical constraints in neural networks. In *Proc. of AAAI*, 2022.
- [Hu *et al.*, 2016a] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. In *Proc. of ACL*, 2016.
- [Hu *et al.*, 2016b] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric Xing. Deep neural networks with massive learned knowledge. In *Proc. of EMNLP*, 2016.
- [Kimmig *et al.*, 2011] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. An algebraic Prolog for reasoning about possible worlds. In *Proc. of AAAI*, 2011.
- [Klement *et al.*, 2000] Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular Norms*. Springer, 2000.
- [Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images Ch. 3. *Tech. Report*, 2009.
- [Kulmanov *et al.*, 2018] Maxat Kulmanov, Mohammad Asif Khan, and Robert Hoehndorf. DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinform.*, 34, 2018.

- [Li and Srikumar, 2019] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic. In *Proc. of ACL*, 2019.
- [Li *et al.*, 2018] Yu Li, Sheng Wang, Ramzan Umarov, Bingqing Xie, Ming Fan, Lihua Li, and Xin Gao. DEEPRe: Sequence-based enzyme EC number prediction by deep learning. *Bioinform.*, 34, 2018.
- [Majumder *et al.*, 2021] Bodhisattwa Prasad Majumder, Oana-Maria Camburu, Thomas Lukasiewicz, and Julian J. McAuley. Rationale-inspired natural language explanations with commonsense. 2021. *arXiv:2106.13876*.
- [Manhaeve *et al.*, 2018] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Proc. of NeurIPS*, 2018.
- [Marra *et al.*, 2019] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. LYRICS: A general interface layer to integrate logic inference and deep learning. In *Proc. of ECML-PKDD*, 2019.
- [Masera and Blanzieri, 2018] Luca Masera and Enrico Blanzieri. AWX: An integrated approach to hierarchical-multi-label classification. In *Proc. of ECML-PKDD*, 2018.
- [Minervini and Riedel, 2018] Pasquale Minervini and Sebastian Riedel. Adversarially regularising neural NLI models to integrate logical background knowledge. In *Proc. of CoNLL*, 2018.
- [Mooney and Shavlik, 2021] Raymond Mooney and Jude Shavlik. A recap of early work on theory and knowledge refinement. In *Proc. of AAAI-MAKE*, 2021.
- [Patel *et al.*, 2022] Dhruvesh Patel, Jay-Yoon Lee Pavitra Dangati, Michael Boratko, and Andrew McCallum. Modeling label space interactions in multi-label classification using box embeddings. In *Proc. of ICLR*, 2022.
- [Raedt *et al.*, 2007] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *Proc. of IJCAI*, 2007.
- [Sachan *et al.*, 2018] Mrinmaya Sachan, Kumar Avinava Dubey, Tom Mitchell, Dan Roth, and Eric Xing. Learning pipelines with limited data and domain knowledge: A study in parsing physics problems. In *Proc. of NeurIPS*, 2018.
- [Senior *et al.*, 2020] Andrew Senior *et al.* Improved protein structure prediction using potentials from deep learning. *Nature*, 577, 2020.
- [Serafini and d’Avila Garcez, 2016] Luciano Serafini and Artur d’Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In *Proc. of NeSy-HLAI*, 2016.
- [Shavlik and Towell, 1989] Jude Shavlik and Geoffrey Towell. Combining explanation-based learning and artificial neural networks. In *Proc. of ML*, 1989.
- [Shavlik, 1994] Jude Shavlik. Combining symbolic and neural learning. *Mach. Learn.*, 14, 1994.
- [Shi *et al.*, 2020] Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On tractable representations of binary neural networks. In *Proc. of KR*, 2020.
- [Silla and Freitas, 2011] Carlos Nascimento Jr. Silla and Alex Alves Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22, 2011.
- [Stevens *et al.*, 2020] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. AI for science: Report on the department of energy (DOE) town halls on artificial intelligence (AI) for science. *Tech. Report*, 2020.
- [Stewart and Ermon, 2017] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Proc. of AAAI*, 2017.
- [Tan, 1997] Ah-Hwee Tan. Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing. *IEEE TNN*, 8, 1997.
- [Towell and Shavlik, 1994] Geoffrey Towell and Jude Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70, 1994.
- [van Krieken *et al.*, 2020] Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy implications. In *Proc. of KR*, 2020.
- [van Krieken *et al.*, 2022] Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *Artif. Intell.*, 302, 2022.
- [von Rueden *et al.*, 2021] Laura von Rueden *et al.* Informed machine learning—A taxonomy and survey of integrating prior knowledge into learning systems. *IEEE TKDE*, 2021.
- [Wehrmann *et al.*, 2018] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. Hierarchical multi-label classification networks. In *Proc. of ICML*, 2018.
- [Wexler, 2017] Rebecca Wexler. When a computer program keeps you in jail: How computers are harming criminal justice. *New York Times*, 2017.
- [Xie *et al.*, 2015] Saining Xie, Tianbao Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *Proc. of CVPR*, 2015.
- [Xie *et al.*, 2019] Yaqi Xie, Ziwei Xu, Kuldeep Meel, Mohan Kankanhalli, and Harold Soh. Embedding symbolic knowledge into deep networks. In *Proc. of NeurIPS*, 2019.
- [Xu *et al.*, 2018] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proc. of ICML*, 2018.
- [Yang *et al.*, 2020] Zhun Yang, Adam Ishay, and Joohyung Lee. NeurASP: Embracing neural networks into answer set programming. In *Proc. of IJCAI*, 2020.
- [Zou *et al.*, 2019] Zhenzhen Zou, Shuye Tian, Xin Gao, and Yu Li. mlDEEPRe: Multi-functional enzyme function prediction with hierarchical multi-label deep learning. *Front. Genet.*, 9, 2019.