How to Tell Deep Neural Networks What We Know

Tirtharaj Dash[1,3], **Sharad Chitlangia**[2,3], **Aditya Ahuja**[1,3], **Ashwin Srinivasan**[1,3]
[1] Department of Computer Science & Information Systems
[2] Department of Electrical and Electronics Engineering
[3] Anuradha and Prashanth Palakurthi Centre for AI Research (APPCAIR)
BITS Pilani, K.K. Birla Goa Campus, Goa 403726, India
{tirtharaj,f20170472,f20170080,ashwin}@goa.bits-pilani.ac.in

Abstract

We present a short survey of ways in which existing scientific knowledge are included when constructing models with neural networks. The inclusion of domain-knowledge is of special interest not just to constructing scientific assistants, but also, many other areas that involve understanding data using human-machine collaboration. In many such instances, machine-based model construction may benefit significantly from being provided with human-knowledge of the domain encoded in a sufficiently precise form. This paper examines the inclusion of domain-knowledge by means of changes to: the input, the loss-function, and the architecture of deep networks. The categorisation is for ease of exposition: in practice we expect a combination of such changes will be employed. In each category, we describe techniques that have been shown to yield significant changes in network performance.

1 Introduction

Science is a cumulative enterprise, with generations of scientists discovering, refining, correcting and ultimately increasing our knowledge of how things are. The accelerating pace of development in software and hardware for machine learning–in particular, the area of deep neural networks–inevitably raises the prospect of Artificial Intelligence for Science [Stevens *et al.*, 2020]. That is, how can we best use AI methods to accelerate our understanding of the natural world? While ambitious plans exist for completely automated AI-based robot scientists [Kitano, 2016], the immediately useful prospect of using AI for Science remains semi-automated. An example of such a collaborative system is in Fig. 1. For such systems to work effectively, we need at least the following: (1) We have to be able to tell the machine what we know, in a suitably precise form; and (2) The machine has to be able to tell us what it is has found, in a suitably understandable form. While the remarkable recent successes of deep neural networks on a wide variety of tasks makes a substantial case for their use in model construction, it is not immediately obvious how either (1) or (2) should be done with deep neural networks. In this paper, we examine ways of achieving (1). Understanding models constructed by deep neural networks is an area of intense research activity, and good summaries exist elsewhere [Lipton, 2016; Arrieta *et al.*, 2019]. To motivate the utility of providing domain-knowledge to a deep network, we reproduce two results from [Dash *et al.*, 2021b] in Fig. 2, which shows that predictive performance can increase significantly, even with a simplified encoding of domain-knowledge (see Fig. 2(a)).

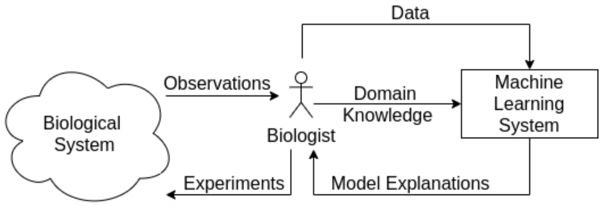
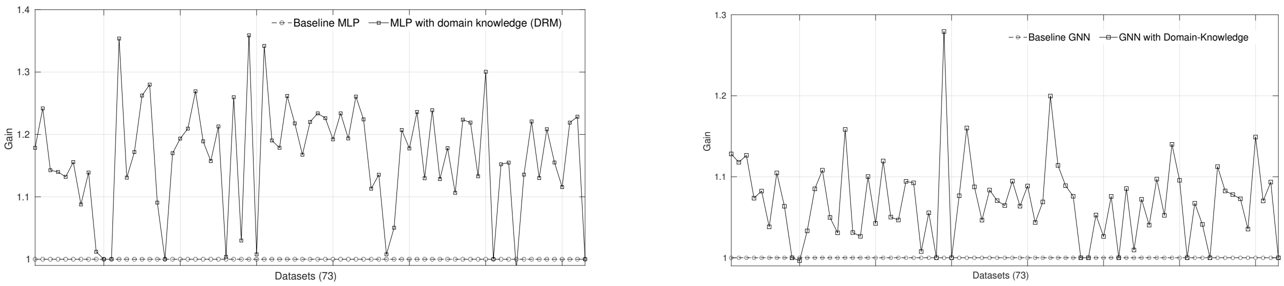


Figure 1: An example of AI for Science. The human-in-the-loop is a biologist. The biologist conducts experiments in a biological system, obtains experimental observations. The biologist then extracts data that can be used to construct machine learning model(s). Additionally, the machine learning system has access to domain knowledge that can be obtained from the biologist. The machine learning system then conveys its explanations to the biologist.

It is unsurprising that a recent report on AI for Science [Stevens *et al.*, 2020] identifies the incorporation of domain-knowledge as one of the 3 Grand Challenges in developing AI systems:

> "Off-the-shelf [ML and AI] practice treats [each of these] datasets in the same way and ignores domain knowledge that extends far beyond the raw data…Improving our ability to systematically incorporate diverse forms of domain knowledge can impact every aspect of AI."

But it is not just the construction of scientific-assistants that can benefit from this form of man-machine collaboration, and "human-in-the-loop" AI systems are likely to play in increasingly important role in engineering, medicine, healthcare, agriculture, environment and so on [Tomašev *et al.*, 2020]. In this survey, we restrict the studies on incorporation of domain knowledge into neural networks, with 1 or more hidden layers (we will sometimes also use the term deep neural network, or DNN). Before we proceed further, we clarify that our focus here is more specific than that



(a)

(b)

Figure 2: The plots from [Dash *et al.*, 2021a] showing gains in predictive accuracy of (a) multilayer perceptron (MLP), and (b) graph neural network (GNN) with the inclusion of domain-knowledge. The domain knowledge inclusion method in (a) is a simple technique known as 'propositionalisation' [Lavrač *et al.*, 1991]; and, the method in (b) is a general technique of incorporating domain-knowledge using bottom-graph construction. The results shown are over 70 datasets. No importance to be given to the line joining two points; this is done for visualisation purpose only.

of research that looks at the development of hybrid neural-symbolic systems (see for example, [Garcez *et al.*, 2012; Raedt *et al.*, 2020]); and different to the use of neural network techniques for either emulating logical reasoning or to represent domain-concepts using a neural architecture. We refer the reader to [Besold *et al.*, 2017] for reviews of some of these other strands of work. These reviews are nevertheless relevant to some of the material in this paper since they identify some key challenges in integrating neural-based learning with symbolic knowledge representation and logical reasoning. More directly related to this paper is the work on "informed machine learning", reviewed in [von Rueden *et al.*, 2019]. We share with this work the interest in prior knowledge as an important source of information that can augment existing data. However, the goals of that paper are more ambitious than here. It aims to identify categories of prior knowledge, using as dimensions: the source of the knowledge, its representation, and its point of use in a machine-learning algorithm. In this survey, we are only concerned with some of these categories. Specifically, in terms of the categories in [von Rueden *et al.*, 2019], we are interested in implicit or explicit sources of domain-knowledge, represented either as logical or numeric constraints, and used at the model-construction stage by DNNs. Therefore, we select the research articles that use any form of domain-knowledge falling under these two categories while constructing deep neural networks.

Focus of the Paper

We adhere to a generic idea of constructing a deep model, that is, given some data D, a structure and parameters of a deep network (denoted by π and θ respectively), a learner $\mathcal{L}$ constructs a deep model M. Additionally, the learner gets a loss function L that it has to optimise (mostly, minimise) while constructing the model from data. Fig. 3 shows this procedure as a block diagram. Note that: (a) we do not describe how the learner $\mathcal{L}$ constructs a model M given the inputs. But, it is trivial to say that the learner iteratively optimises the loss L while training a model with structure π and parameters θ from the given data D. For this process, it uses an optimisation procedure such as the ones based on gradient descent; and (b) we are not concerned with how the constructed deep model M is going to be used. However, it suffices to say that the (constructed) model M gets an input data instance (described by the same feature space that it was trained with) and produces an output (model prediction).

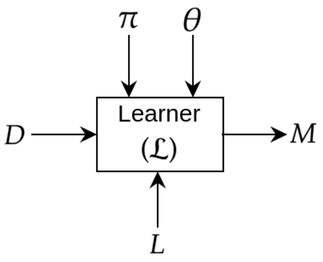


Figure 3: Construction of a deep model M from data (D) using a learner ($\mathcal{L}$). We use π to denote the structure (organisation of various layers, their interconnections etc.) and and θ to denote the parameters (synaptic weights) of the deep network. L denotes the loss function (for example, cross-entropy loss in case of classification).

In this survey, we consider the inclusion of domain knowledge by transforming: (1) The input data to a deep network; (2) The loss-function used; and (3) The model (that is, the structure or parameters) of the deep network. In a sense, this progression reflects a graded increase in the complexity of changes involved. We will restrict ourselves to domain-knowledge that can be represented either as logical or as numerical constraints. Under logical constraints we consider domain knowledge that is represented in propositional logic, predicate logic, including binary and more generally n-ary relations, canonical normal forms, program primitives. The numerical constraints are represented by prior of model structure and parameters. So, we survey on research works that involve these forms of background knowledge. We believe this covers a wide range of potential applications, including those concerned with scientific discovery.

The rest of the paper is organised as follows: Section 2 describes inclusion of domain-knowledge by augmenting or transformation of inputs; Section 3 describes changes that have been employed to loss-functions; and Section 4 describes biases on parameters and changes to the structure of deep networks. Section 5 outlines some major challenges related to the inclusion of domain-knowledge in the ways we describe. In this section, we also present perspectives on the relevance of the use of domain-knowledge to aspects of Responsible AI, including ethics, fairness, and explainability of DNNs.

2 Transforming the Input Data

One of the prominent approaches to incorporate domain-knowledge into deep network is by changing inputs to the network. Here, the domain-knowledge is primarily in symbolic form. The idea is simple: If a data instance could be described using a set of attributes that not only includes the raw feature-values but also includes more details from the domain, then a standard deep network could then be constructed from these new features. A simple block diagram in Fig. 4 shows how domain knowledge is introduced into the network via changes in inputs. In this survey, we discuss broadly two different ways of doing this: (a) using relational features, mostly constructed by a method called propositionalisation [Lavrač *et al.*, 1991] using another machine learning system (for example, Inductive Logic Programming) that deals with data and background knowledge; (b) without propositionalisation.

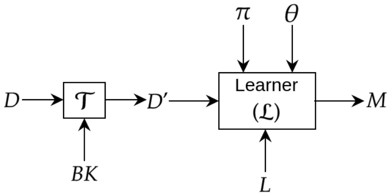


Figure 4: Introducing background knowledge into deep network by transforming data. $\mathcal{T}$ is a transformation block that takes input data D, background knowledge (BK) and outputs transformed data D' that is then used to construct a deep model using a learner $\mathcal{L}$.

2.1 Propositionalisation

The pre-eminent form of symbolic machine learning based on the use of relations in first-order logic is Inductive Logic Programming (ILP) [Muggleton, 1991], which has an explicit role for domain-knowledge being incorporated into learning. The simplest use of ILP [Muggleton, 1991] to incorporate n-ary relations in domain knowledge into a neural network relies on techniques that automatically "flatten" the domain-knowledge into a domain-specific relational features. Although not all DNNs require data to be a set of feature-vectors, this form of data representation is long-standing and still sufficiently prevalent. In logical terms, we categorise feature-based representations as being encodings in a propositional logic. The reader would point out, correctly, that feature-values may not be Boolean. This is correct, but we can represent non-Boolean features by Boolean-valued propositions (for example, a real-valued feature f with value 4.2 would be represented by a corresponding Boolean feature f' that has the value 1 if $f = 4.2$ and 0 otherwise). With the caveat of this rephrasing, it has of course been possible to provide domain-knowledge to neural networks by employing domain-specific features devised by an expert. However, we focus here on ways in which domain-knowledge encoded as rules in propositional logic has been used to constrain the structure or parameters of models constructed by a network. Techniques for automatic construction of Boolean-valued features from relational domain-knowledge have a long history in the field of ILP [Muggleton and de Raedt, 1994; Muggleton *et al.*, 2012; Cropper *et al.*, 2020], originating from the LINUS [Lavrač *et al.*, 1991]. Often called *propositionalisation*, the approach involves the construction of features that identify the conditions under which they take on the value 1 (or 0). For example, given (amongst other things) the definition of benzene rings and of fused rings, an ILP-based propositionalisation may construct the Boolean-valued feature that has the value 1 if a molecule has 3 fused benzene rings, and 0 otherwise. The values of such Boolean-valued features allows us to represent a data instance (like a molecule) as a Boolean-valued feature-vector, which can then provided to a neural network. There is a long history of propositionalisation: see [Kramer *et al.*, 2001] for a review of some of early use of this technique, and [Lavrac *et al.*, 2020; Vig *et al.*, 2017] who examine the links between propositionalisation and modern-day use of embeddings in neural-networks.

A direct application of propositionalisation, demonstrating its utility for deep networks has been its use in Deep Relational Machines [Lodhi, 2013]. A DRM is a deep fully-connected neural network with Boolean-valued inputs obtained from propositionalisation by an ILP engine. In [Dash *et al.*, 2018], Boolean-valued features from an ILP engine are sampled from a large space of possible features. The sampling technique is refined further in [Dash *et al.*, 2019].

The idea of propositionalisation also forms the foundation for a method known as 'Bottom Clause Propositionalisation (BCP)' to propositionalise literals of a most-specific clause, or "bottom-clause" in ILP. Given a data instance, the bottom-clause is the most-specific first-order clause that entails the data instance, given some domain-knowledge. Loosely speaking, the most-specific clause can be thought of "enriching" the data instance with all domain relations that are true, given the data instance. The construction of such most-specific clauses and their subsequent use in ILP was introduced in [Muggleton, 1995]. CILP++ [França *et al.*, 2014] uses bottom-clauses for data instances to construct feature-vectors for neural networks. This is an extension to CIL^{2}P. Here the neural network has recurrent connections.

Propositionalisation has conceptual and practical limitations. Conceptually, there is no variable-sharing between two or more first-order logic features [Dash *et al.*, 2018]. That is, all useful compositions have to be pre-specified. Practically, this makes the space of possible features extremely large: this has meant that the feature-selection has usually been done separately from the construction of the neural network. In this context, another work that does not employ either propositionalisation or network augmentation considers a combination of symbolic knowledge represented in first-order logic with matrix factorization techniques [Rocktäschel et al., 2015]. This exploits dependencies between textual patterns to generalise to new relations.

Recent work on neural-guided program synthesis also explicitly includes domain-specific relations. Here programs attempt to construct automatically compositions of functional primitives. The primitives are represented as fragments of functional programs that are expected to be relevant. An example of neural-guided program synthesis that uses such domain-primitives is DreamCoder [Ellis *et al.*, 2018; Ellis *et al.*, 2020]. DreamCoder receives as inputs, the partial specification of a function in the form of some inputs–output pairs, and a set of low-level primitives represented in a declarative language. Higher-level domain-concepts are then abduced as compositions of these primitives via a neurally-guided search procedure based on a version of Bayesian "wake-sleep" algorithm [Hinton *et al.*, 1995]. The deep networks use a (multi-hot) Boolean-vector encoding to represent functional compositions (a binary digit is associated with each primitive function, and takes the value 1 if and only if the primitive is used in the composite function).

2.2 Binary and n-ary Relations

An influential form of representing relational domain-knowledge takes the form *knowledge graph*, which are labelled graphs, with vertices representing entities and edges representing binary relations between entities. We refer the reader to [Hogan *et al.*, 2020] to a comprehensive survey of this form of representation for domain-knowledge.

Incorporation of the information in a knowledge-graph into deep neural models–termed "knowledge-infused learning"–is described in [Kursuncu *et al.*, 2019; Sheth *et al.*, 2019]. This aims to incorporate binary relations contained in application-independent sources (like DBPedia, Yago, WikiData) and application-specific sources (like SNOMED-CT, DataMed). The work examines techniques for incorporating relations at various layers of deep-networks (the authors categorise these as "shallow", "semi-deep" and "deep" infusion). In the case of shallow infusion, both the external knowledge and the method of knowledge infusion is shallow, utilising syntactic and lexical knowledge in the form of word embedding models. In semi-deep infusion, external knowledge is involved through attention mechanisms or learnable knowledge constraints acting as a sentinel to guide model learning, and deep infusion employs a stratified representation of knowledge representing different levels of abstractions in different layers of a deep learning model, to transfer knowledge that aligns with the corresponding layer in the layered learning process.

Knowledge graphs can be encoded directly for use by a graph neural network (GNN). The computational machinery available in GNN then aggregates and combines the information available in the knowledge graph. The final collected information from this computation could be used for further predictions. Some recent works are in [Park *et al.*, 2019; Wang *et al.*, 2019], where a GNN is used for estimation of node importance in a knowledge-graph. The intuition is that the nodes (in a problem involving recommender systems, as in [Wang *et al.*, 2019], a node represents an entity) in the knowledge graph can be represented with an aggregated neighbourhood information with bias while adopting the central idea of aggregate-and-combine in GNNs. The very idea of encoding a knowledge graph directly for a GNN is also used in Knowledge-Based Recommender Dialog (KBRD) framework developed for recommender systems [Chen *et al.*, 2019]. In this work, the authors treat an external knowledge graph, such as DBpedia [Lehmann *et al.*, 2015], as a source of domain-knowledge allowing entities to be enriched with these knowledge. The authors found that the introduction of such knowledge in the form of knowledge graph can strengthen the recommendation performance significantly and can enhance the consistency and diversity of the generated dialogs.

Going beyond binary relations and treating n-ary relations as hyperedges, a technique called *vertex enrichment* is proposed in [Dash *et al.*, 2021b]. Vertex-enrichment is a simplified approach for the inclusion of domain-knowledge into standard graph neural networks (GNNs). The approach incorporates first-order background relations by augmenting the features associated with the nodes of a graph provided to a GNN. The results reported in [Dash *et al.*, 2021b] show significant improvements in the predictive accuracy of GNNs across a large number datasets. The simplification used in vertex-enrichment has been made unnecessary in a recent proposal for transforming the most-specific clause constructed by ILP systems employing mode-directed inverse entailment (MDIE [Muggleton, 1995]). The transformation converts this clause into a graph can be directly used by any standard GNN [Dash *et al.*, 2021a]. Specifically, the transformation results in a labelled bipartite graph consisting of vertices representing predicates (including domain predicates) and ground terms. This approach reports better predictive performance than those reported in [Dash *et al.*, 2021b], and includes knowlede-graphs as a special case. Most recently, this method has been combined successfully with deep generative sequence models for generating target-specific molecules, which demonstrates yet another real-world use-case of incorporating domain knowledge into deep networks [Dash *et al.*, 2021c].

3 Transforming the Loss Function

One standard way of incorporating domain-knowledge into a deep network is by introducing "penalty" terms into the loss (or utility) function that reflect constraints imposed by domain-knowledge. The optimiser used for model-construction then minimises the overall loss that includes the penalty terms. Fig. 5 shows a simple block diagram where a new loss term is introduced based on the background knowledge. We distinguish two kinds of domain constraints–syntactic and semantic–and describe how these have been used to introduce penalty terms into the loss function.

3.1 Syntactic Loss

The usual mechanism for introducing syntactic constraints is to introduce one or more *regularisation* terms into the loss function. The most common form of regularisation introduces penalities based on model complexity (number of hidden layers, or number of parameters and so on: see for example, [Kukačka *et al.*, 2017]).

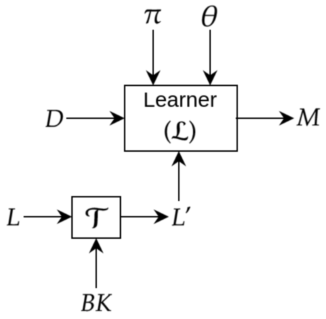


Figure 5: Introducing background knowledge into deep network by transforming the loss function L. $\mathcal{T}$ block takes an input loss L and outputs a new loss function L' by transforming (augmenting or modifying) L based on background knowledge (BK). The learner $\mathcal{L}$ then constructs a deep model using the original data D and the new loss function L'.

A more elaborate form of syntactic constraints involves the concept of *embeddings*. Embeddings refer to the relatively low-dimensional learned continuous vector representations of discrete variables. Penalty terms based on "regularising embeddings" are used to encode syntactic constraints on the complexity of embeddings. [Fu, 1995] was an early work in this direction, in which the authors proposed a strategy to establish constraints by designating each node in a Hopfield Net to represent a concept and edges to represent their relationships and learn these nets by finding the solution which maximises the greatest number of these constraints. [Rocktäschel *et al.*, 2014] was perhaps the first method of regularising embeddings from declarative knowledge encoded in first-order logic. The proposal here is for a mapping between logical statements and their embeddings, and logical inferences and matrix operations. That is, the model behaves as if it is following a complex first-order reasoning process, but operates at the level of simple vectors and matrix representations. [Rocktäschel *et al.*, 2015] extended this to regularisation by addition of differentiable loss terms to the objective-based on propositionalisation of each first-order predicate. Guo *et al.* [Guo *et al.*, 2016] proposed a joint model, called KALE, which embeds facts from knowledge graphs and logical rules simultaneously. Here, the facts are represented as ground atoms with a calculated truth value in $[0, 1]$ suggesting how likely that the fact holds. Logical rules (in grounded form) are then interpreted as complex formulae constructed by combining ground atoms with logical connectives, which are then modelled by fuzzy t-norm operators [Hájek, 2013]. The truth value that results from this operation is nothing but a composition of the constituent ground atoms, allowing the facts from the knowledge graph to be incorporated into the model.

[Li and Srikumar, 2020] develop a method to constraint individual neural layers using soft logic based on massively available declarative rules in ConceptNet. [Hamilton *et al.*, 2018] incorporates first-order logic into low dimensional spaces by embedding graphs nodes and represents logical operators as learned geometric relations in the space. [Demeester *et al.*, 2016] proposed ordering of embedding space based on rules mined from WordNet and found it to better prior knowledge and generalisation capabilities using these relational embeddings. [Li *et al.*, 2018] show that domain-based regularisation in loss function can also help in constructing deep networks with less amount of data in prediction problems concerned with cloud computing. In [Takeishi and Akimoto, 2018], a knowledge-based distant regularisation framework was proposed, in which distance domain information encoded in a knowledge graph was utilised. It defines prior distributions of model parameters using knowledge graph embeddings. They show that this results in an optimisation problem for a regularised factor analysis method.

3.2 Semantic Loss

Penalty terms can also be introduced on the extent to which the model's prediction satisfies semantic domain constraints. For example, the domain may impose specific restrictions on the prediction ("output prediction must be in the range $3 \dots 6$"). One way in which such information is provided is in the form of domain-constraints. Penalty terms are then introduced based on the number and importance of such constraints that are violated.

A recent work that is based on loss function is in [Xu *et al.*, 2018]. Here the authors propose a semantic loss that signifies how well the outputs of the deep network matches some given constraints encoded as propositional rules. The general intuition behind this idea is that the semantic loss is proportional to a negative logarithm of the probability of generating a state that satisfies the constraint when sampling values according to some probability distribution. This kind of loss function is particularly useful for semi-supervised learning as these losses behave like self-information and are not constructed using explicit labels and can thus utilize unlabelled data.

[Hu *et al.*, 2016] proposed a framework to incorporate first-order logic rules with the help of an iterative distillation procedure that transfers the structured information of logic rules into the weights of neural networks. This is done via a modification to the knowledge-distillation loss proposed by Hinton et al. [Hinton *et al.*, 2015]. The authors show that taking this loss-based route of integrating rule-based domain-knowledge allows the flexibility of choosing a deep network architecture suitable for the intended task.

In [Fischer *et al.*, 2019], authors construct a system for training a neural network with domain-knowledge encoded as logical constraints. Here the available constraints are transferred to a loss function. Specifically, each individual logic operation (such as negation, and, or, equality etc.) is translated to a loss term. The final formulation results in an optimisation problem. The authors extract constraints on inputs that capture certain kinds of convex sets and use them as optimisation constraints to make the optimisation tractable. In the developed system, it is also possible to pose queries on the model to find inputs that satisfy a set of constraints. In a similar line, [Muralidhar *et al.*, 2019] proposed domain-adapted neural networks (DANN) that works with a balanced loss function at the the intersection of models based on purely domain-based loss or purely inductive loss. Specifically, they introduce a domain-loss term that requires a functional form of approximation and monotonicity constraints. Without detailing much on the underlying equations, it suffices to say that formulating the domain loss using these constraints enforces the model to learn not only from training data but also in accordance with certain accepted domain rules.

Another popular approach that treats domain knowledge as 'domain constraints' is semantic based regularisation [Diligenti *et al.*, 2017a; Diligenti *et al.*, 2017b]. It builds standard multilayered neural networks (e.g. MLP) having kernel machines at the input layer dealing with continuous-valued features. The output of the kernel machines is input to the higher layers implementing a fuzzy generalisation of the domain constraints that are represented in first-order logic. The regularisation term, consisting of a sum of fuzzy generalisation of constraints using t-norm operations, in the cumulative loss then penalises each violation of the constraints during the training of the deep network. [Silvestri *et al.*, 2020] inject domain knowledge at training time via an approach that combines semantic based regularisation and constraint programming [Rossi *et al.*, 2006]. This approach uses the concept of 'propagators', which is inherent in constraint programming to identify infeasible assignments of variables to values in the domain of the variables. The role of semantic based regularisation is to then penalise these infeasible assignments weighted by a penalty parameter.

4 Transforming the Model

Over the years, many studies have shown that domain knowledge can be incorporated into a deep network by introducing constraints on the model parameters (weights) or by making a design choice of its structure. Fig. 6 shows a simple block diagram showing domain knowledge incorporation at the design stage of the deep network.

4.1 Constraints on Parameters

In a Bayesian formulation, there is an explicit mechanism for the inclusion of domain-knowledge through the use of priors. The regularisation terms in loss-functions, for example, can be seen as an encoding of such prior information, usually on the network's structure. Priors can also be introduced on the parameters (weights) of a network. Explicitly, these would take the form of a prior distribution over the values of the weights in the network. The priors on networks and network weights represent our expectations about networks before receiving any data, and correspond to penalty terms or regularisers. Buntine and Weigend [Buntine and Weigend, 1991] extensively study how Bayesian theory can be highly relevant to the problem of training feedforward neural networks. This work is explicitly concerned with choosing an appropriate network structure and size based on prior domain-knowledge and with selecting a prior the weights.

The work by [Neal, 1995] on Bayesian learning for neural networks also showed how domain-knowledge could help build a prior probability distribution over neural network parameters. In this, the shown priors allow networks to be "self-regularised" to not over-fit even when complexity of the neural network is increased. In a similar spirit, [Krupka and Tishby, 2007] showed how prior domain knowledge could be used to define 'meta-features' that can aid in defining the prior distribution of weights. These meta-features are additional information about each of the features in the available data. For instance, for an image recognition task, the meta-feature could be the relative position of a pixel (x, y) in the image. This meta information can be used to construct a prior over the weights for the original features.

Transfer Learning

Transfer Learning is a mechanism to introduce priors on weights when data is scarce for a problem (usually called the "target" domain). Transfer learning relies on the availability of data for a problem that is similar to the target domain (usually called the "source" domain). Transfer learning from a source domain to a target domain involves a transfer of weights from models constructed for the source domain to the network in the target domain. This has been shown to boost performance significantly. From the Bayesian perspective, transfer learning allows the construction of the prior over the weights of a neural network for the target domain based on the posterior constructed in the source domain. Transfer learning is not limited by the kind of task (such as classification, regression, etc) but rather by the availability of related problems. Language models are a very successful example of the use of transfer learning, where the models are initially learnt on a huge corpus of data and fine-tuned for other more specialised tasks. [Zhuang *et al.*, 2020] provides an in-depth review of some of the mechanisms and the strategies of transfer learning. Transfer learning need not be restricted to deep networks only: in a recent study, [Liu *et al.*, 2018] proposes a model that transfers knowledge from a neural network to a decision tree using knowledge distillation framework. The symbolic knowledge encoded in the decision tree could further be utilised for a variety of tasks.

4.2 Specialised Structures

DNN based methods arguably work best if the domain-knowledge is used to inspire their architecture choices [Berner *et al.*, 2021]. There are reports on incorporating first-order logic constructs into the structure of the network. This allows neural-networks to operate directly on the logical sentences comprising domain-knowledge.

Domain-knowledge encoded as a set of propositional rules are used to constrain the structure of the neural network. Parameter-learning (updating of the network weights) then proceeds as normal, using the structure. The result could be thought of as learning weighted forms of the antecedents present in the rules. The most popular and oldest work along this line is Knowledge-Based Artificial Neural Network (KBANN) [Towell *et al.*, 1990] that incorporates knowledge into neural networks. In KBANN, the domain knowledge is represented as a set of hierarchically structured propositional rules that directly determines a fixed topological structure of a neural network [Towell and Shavlik, 1994]. KBANN was successful in many real-world applications; but, its representational power was bounded by pre-existing set of rules which restricted it to refine these existing rules rather than discovering new rules. A similar study is KBCNN [Fu, 1993], which first identifies and links domain attributes and concepts consistent with initial domain knowledge. Further, KBCNN in-

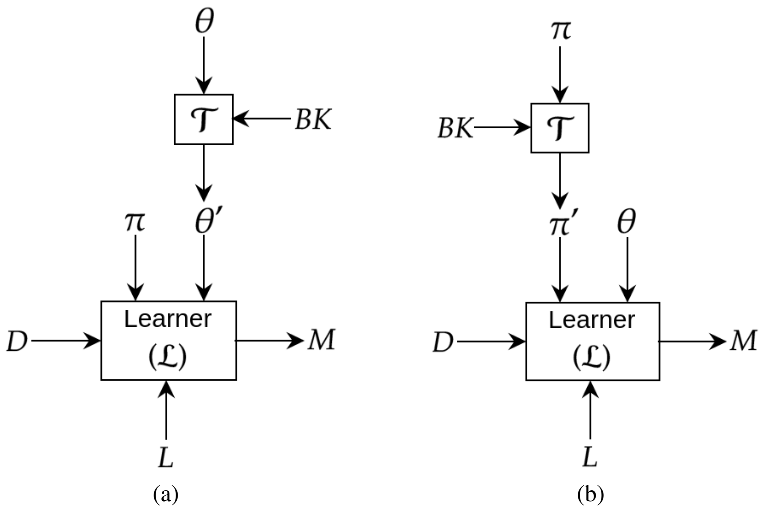


Figure 6: Introducing background knowledge into deep network by transforming the model (structure and parameter). In (a), the transformation block $\mathcal{T}$ takes a input structure of a model π and outputs a transformed structure π' based on background knowledge (BK). In (b), the transformation block $\mathcal{T}$ takes a set of parameters θ of a model and outputs a transformed set of parameters π' based on background knowledge (BK).

troduces additional hidden units into the network and most importantly, it allowed decoding of the learned rules from the network in symbolic form. However, both KBANN and KBCNN were not appropriate for learning new rules because of the way the initial structure was constructed using the initial domain knowledge base.

Some of the limitations described above could be overcome with the proposal of a hybrid system by Fletcher and Obradovic [Fletcher and Obradovic, 1993]. The system was able to learn a neural network structure that could construct new rules from an initial set of rules. Here, the domain knowledge is transformed into an initial network through an extended version of KBANN's symbolic knowledge encoding. It performed incremental hidden unit generation thereby allowing construction or extension of initial rule-base. In a similar manner, there was a proposal for using Cascade ARTMAP [Tan, 1997] which could not only construct a neural network structure from rules but also perform explicit cascading of rules and multistep inferencing. It was found that the rules extracted from Cascade ARTMAP are more accurate and much cleaner than the rules extracted from KBANN [Towell and Shavlik, 1993].

In the late 1990s, Garcez and Zaverucha proposed a massively parallel computational model called CIL^{2}P based on feedforward neural network that integrates inductive learning from examples and domain knowledge, expressed as a propositional logic program [Avila Garcez and Zaverucha, 1999]. A translation algorithm generates a neural network. Unlike KBANN, the approach uses the notion of "bipolar semi-linear" neurons. This allows the proof of a form of correctness, showing the existence of a neural-network structure that can compute the logical consequences of the domain-knowledge. The output of such a network, when combined into subsequent processing naturally incorporates the intended interpretation of the domain predicates. The authors extend this to the use of first-order logic programs: we consider this in a later section.

A recent proposal focuses on embedding symbolic knowledge expressed as logical rules [Xie *et al.*, 2019]. It considers two languages of representations: Conjuctive Normal Form (CNF) and decision-Deterministic Decomposable Negation Normal form (d-DNNF), which can naturally be represented as graph structures. The graph structures can be provided to a graph neural network (GNN) to learn an embedding suitable for further task-specific implementations.

Somewhat in a similar vein to the work by [Avila Garcez and Zaverucha, 1999], the work reported in [Xu *et al.*, 2018] considers as a set of propositional statements representing domain constraints. A deep network is then trained to find satisfying assignments for the constraints. Again, once such a network is constructed, it can clearly be used in subsequent processing, capturing the effect of the domain constraints. The network is trained using a semantic loss that we have described in Sec. 3.2.

In [Li and Srikumar, 2020] it is proposed to augment a language model that uses a deep net architecture with additional statements in first-order logic. Thus, given domain-knowledge encoded as first-order relations, connections are introduced into the network, based on the logical constraints enforced by the domain-relations. The approach is related somewhat to the work in [Sourek *et al.*, 2018a] that does not explicitly consider the incorporation of domain-knowledge but does constrain a deep neural network's structure by first grounding a set of weighted first-order definite clauses and then turning them into propositional programs.

We note that newer areas are emerging that use representations for domain-knowledge that go beyond first-order logic relations. This includes probabilistic first-order logic, as a way of including uncertain domain-knowledge [Manhaeve *et al.*, 2018]. One interesting way this is being used is to constrain the training of "neural predicates", which represent probabilistic relations that are implemented by neural networks, and the framework can be trained in an end-to-end fashion [Manhaeve *et al.*, 2018; Winters *et al.*, 2021]. In DeepProbLog [Manhaeve *et al.*, 2018], for example, high-level logical reasoning can be combined with sub-symbolic discriminative power of deep networks. For instance, a logic program for adding two digits and producing the output sum is straightforward. However, what if the inputs are images of the corresponding digits? Here, a deep network is used to map an image to a digit, while a (weighted) logic program, written in ProbLog [De Raedt *et al.*, 2007], for addition is treated as symbolic domain knowledge. The ProbLog program is extended with a set of ground neural predicates for which the weights correspond to the probability distribution of classes of digits ($0\ldots9$). The parameters (weights of predicates and weights of neural network) are learned in an end-to-end fashion. A recent approach called DeepStochLog [Winters *et al.*, 2021] is a framework that extends the idea of neural predicates in DeepProbLog to definite clause grammars [Pereira and Warren, 1980]. Reader may note that although DeepProbLog and DeepStochLog do not really transform the structure of the deep network, we are still considering these method under the heading of specialised structures. This is because of the fact that the hybrid architecture is a tightly coupled approach combining probabilistic logic and deep neural networks.

One of the approaches involves transformation of probabilistic logic program to graph-structured representation. For instance, in kLog [Frasconi *et al.*, 2014] the transformed representation is an undirected bipartite graph in the form of 'Proabilistic Entity-Relationship model' [Heckerman *et al.*, 2007] which allows the use of a graph-kernel [Vishwanathan *et al.*, 2010] for data classification purpose, where each data instance is represented as a logic program constructed from data and background-knowledge. Another approach uses weighted logic programs or *templates* with GNNs [Sourek *et al.*, 2020] demonstrating how simple relational logic programs can capture advanced graph convolution operations in a tightly integrated manner. However, it requires use of a language of Lifted Relational Neural Networks (LRNNs) [Sourek *et al.*, 2018b].

An interesting proposal is to transform facts and rules, all represented in (weighted) first-order logic into matrix (or tensor) representations. Learning and inference can then be conducted on these matrices (or tensors) [Serafini and Garcez, 2016; Cohen *et al.*, 2020] allowing faster computation. NeuralLog [Guimarães and Costa, 2021], for example, extends this idea and constructs a multilayered neural network, to some extent, similar to the ones in LRNN consisting of fact layer, rule layer and literal layer etc. The learning here refers to the updates of the weights of the rules. Another work that translates domain-knowledge in first-order logic into a deep neural network architecture consisting of input layer (grounded atoms), propositional layer, quantifier layer and output layer is [Diligenti *et al.*, 2017a]. Similar to LRNN, it uses the fuzzy t-norm operator for translating logical OR and AND operations.

Further emerging areas look forward to providing domain-knowledge as higher-order logic templates (or "meta-rules": see [Cropper *et al.*, 2020] for pointers to this area). To the best of our knowledge, there are, as yet, no reports in the literature on how such higher-order statements can be incorporated into deep networks.

5 Challenges and Concluding Remarks

We summarise our discussion on domain-knowledge as constraints in Table 1. We now outline some challenges in incorporating domain-knowledge encoded as logical or numerical constraints into a deep network. We first outline some immediate practical challenges concerning the logical constraints:

- There is no standard framework for translating logical constraints to neural networks. While there are simplification methods which first construct a representation of the logical constraint that a standard deep network can consume, this process has its limitations as described in the relevant section above.
- Logic is not differentiable. This does not allow using standard training of deep network using gradient based methods in an end-to-end fashion. Propagating gradients via logic has now been looked at in [Evans and Grefenstette, 2018], but the solution is intractable and does not allow day-to-day use.
- Many neural network structures are directed acyclic graphs (DAGs). However, transforming logical formula directly into neural network structures in the manner described in some of the discussed works can introduce cyclic dependencies, which may need a separate form of translations.

There are also practical challenges concerning the numerical constraints:

- We have seen that the numerical constraints are often provided with the help of modification to a loss function. Given some domain-knowledge in a logical representation, constructing a term in loss function is not straight-forward.
- The process of introducing a loss term often results in a difficult optimisation problem (sometimes constrained) to be solved. This may require additional mathematical tools for a solution that can be implemented practically.
- Deep network structures constrained via logical domain-knowledge may not always be scalable large datasets.

The Domain-Knowledge Grand Challenge

Incorporating domain-knowledge into learning is highlighted in [Stevens *et al.*, 2020] as one of the Grand Challenges facing the foundations of AI and ML. The principal difficulties raised in that report are these:

- "Can the constructed deep network model be trusted?" This question involves long-standing discussions on explainability and interpretability of deep models. It also includes the question of whether data used for constructing the deep model contains sufficient information without introducing spurious correlations or bias that would invalidate the model itself.

Principal Approach	**Work (Reference)**	**Type of Learner**
Transforming Data	DRM [Lodhi, 2013; Dash *et al.*, 2018]	MLP
	CILP++ [França *et al.*, 2014]	MLP
	KGCN [Wang *et al.*, 2019]	GNN
	KBRD [Chen *et al.*, 2019]	GNN
	DreamCoder [Ellis *et al.*, 2020]	DNN*
	VEGNN [Dash *et al.*, 2021b]	GNN
	BotGNN [Dash *et al.*, 2021a]	GNN
Transforming Loss	IPKFL [Krupka and Tishby, 2007]	CNN
	ILBKRME [Rocktäschel *et al.*, 2015]	MLP
	HDNNLR [Hu *et al.*, 2016]	CNN, RNN
	SBR [Diligenti *et al.*, 2017a]	MLP
	SBR [Diligenti *et al.*, 2017b]	CNN
	DL2 [Fischer *et al.*, 2019]	CNN
	Semantic Loss [Xu *et al.*, 2018]	CNN
	LENSR [Xie *et al.*, 2019]	GNN
Transforming Model	KBANN [Towell and Shavlik, 1994]	MLP
	Cascade-ARTMAP [Tan, 1997]	ARTMAP
	CIL^{2}P [Avila Garcez and Zaverucha, 1999]	RNN
	DeepProbLog [Manhaeve *et al.*, 2018]	CNN
	LRNN [Sourek *et al.*, 2018b]	MLP
	TensorLog [Cohen *et al.*, 2020]	MLP
	NeuralLog [Guimarães and Costa, 2021]	MLP
	DeepStochLog [Winters *et al.*, 2021]	DNN*

Table 1: Some selected works, in no particular order, showing the principal approach of domain knowledge inclusion into deep neural networks. For each work referred here, we show the type of learner with following acronyms: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Graph Neural Network (GNN), Adaptive Resonance Theory-based Network Map (ARTMAP), DNN* refers to a DNN structure dependent on intended task. We use 'MLP' here to represent any neural network, that conforms to a layered-structure that may or maynot be fully-connected.

- "Why does the AI model work for a problem?" To address this question, there has to be some mapping between the internal representations of the model and the domain-specific concepts. In [Futia and Vetrò, 2020], authors identify that the knowledge mapping of the deep learning components, including input features, hidden unit and layers, and output predictions with domain-knowledge could lead to an understandable model.

Going Beyond Prediction

The issues raised above go beyond just the "how" questions related to the incorporation of domain-knowledge into deep networks. They provide pointers to why the use of domain-knowledge may extend beyond its utility for prediction. Domain-knowledge can also play a role in aspects like explanation and fairness. We mention some of the challenges that result.

One important requirement of machine-constructed models in workflows with humans-in-the-loop is that the models are human-understandable. Domain-knowledge can be used in two different ways to assist this. First, it can constrain the kinds of models that are deemed understandable. Secondly, it can provide concepts that are meaningful for use in a model. Most of the work in this review has been focussed on improving predictive performance. However, the role of domain-knowledge in constructing explanations for deep network models is also being explored (see for example, [Srinivasan *et al.*, 2019]). However, that work only generates *post hoc* explanations that are locally consistent. Explanatory deep network models that identify true causal connections based on concepts provided as domain-knowledge remain elusive.

Domain-knowledge can also be used to correct biases [Mehrabi *et al.*, 2021] built into a deep network either declaratively, through the use of constraints, or through the use of loss functions that include "ethical penalty" terms. Demonstrations of the use of domain-knowledge driven, ethics-sensitive machine learning have been available in the literature for some time [Anderson *et al.*, 2005]. Can these carry over to the construction of deep network models? This remains to be investigated.

Finally, the rapid progress in the area of language models, for example, the models based on attention [Vaswani *et al.*, 2017; Brown *et al.*, 2020] raises the possibility of providing domain-knowledge in forms other than logical or numerical. While the precision of these formal representations may continue to be needed for the construction of scientific assistants, their role in representing commonsense knowledge is less evident. Day-to-day machine assistants that can incorporate informal knowledge of the world will be needed. Progress in this is being made (see for example, https://allenai.org/aristo), but there is much more that needs to be done to make the language models required accessible to everyday machinery.

References

[Anderson *et al.*, 2005] Michael Anderson, S. Anderson, and Chris Armen. Medethex: Toward a medical ethics advisor. In *AAAI Fall Symposium: Caring Machines*, 2005.

[Arrieta *et al.*, 2019] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *arXiv preprint arXiv:1910.10045*, 2019.

[Avila Garcez and Zaverucha, 1999] Artur S. Avila Garcez and Gerson Zaverucha. Connectionist inductive learning and logic programming system. *Applied Intelligence*, 1999.

[Berner *et al.*, 2021] Julius Berner, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. The modern mathematics of deep learning. *arXiv preprint arXiv:2105.04026*, 2021.

[Besold *et al.*, 2017] Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*, 2017.

[Brown *et al.*, 2020] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[Buntine and Weigend, 1991] Wray L. Buntine and A. Weigend. Bayesian back-propagation. *Complex Syst.*, 5, 1991.

[Chen *et al.*, 2019] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. Towards knowledge-based recommender dialog system. *arXiv preprint arXiv:1908.05391*, 2019.

[Cohen *et al.*, 2020] William Cohen, Fan Yang, and Kathryn Rivard Mazaitis. Tensorlog: A probabilistic database implemented using deep-learning infrastructure. *Journal of Artificial Intelligence Research*, 67:285–325, 2020.

[Cropper *et al.*, 2020] Andrew Cropper, Sebastijan Dumančić, and Stephen H. Muggleton. Turning 30: New ideas in inductive logic programming. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4833–4839, 2020.

[Dash *et al.*, 2018] Tirtharaj Dash, Ashwin Srinivasan, Lovekesh Vig, Oghenejokpeme I Orhobor, and Ross D King. Large-scale assessment of deep relational machines. In *International Conference on Inductive Logic Programming*, pages 22–37. Springer, 2018.

[Dash *et al.*, 2019] Tirtharaj Dash, Ashwin Srinivasan, Ramprasad S Joshi, and A Baskar. Discrete stochastic search and its application to feature-selection for deep relational machines. In *International Conference on Artificial Neural Networks*, pages 29–45. Springer, 2019.

[Dash *et al.*, 2021a] Tirtharaj Dash, Ashwin Srinivasan, and A Baskar. Inclusion of domain-knowledge into gnns using mode-directed inverse entailment. *arXiv preprint arXiv:2105.10709*, 2021.

[Dash *et al.*, 2021b] Tirtharaj Dash, Ashwin Srinivasan, and Lovekesh Vig. Incorporating symbolic domain knowledge into graph neural networks. *Machine Learning*, pages 1–28, 2021.

[Dash *et al.*, 2021c] Tirtharaj Dash, Ashwin Srinivasan, Lovekesh Vig, and Arijit Roy. Using domain-knowledge to assist lead discovery in early-stage drug design. *bioRxiv*, 2021.

[De Raedt *et al.*, 2007] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467. Hyderabad, 2007.

[Demeester *et al.*, 2016] T. Demeester, Tim Rocktäschel, and S. Riedel. Lifted rule injection for relation embeddings. *ArXiv*, abs/1606.08359, 2016.

[Diligenti *et al.*, 2017a] Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

[Diligenti *et al.*, 2017b] Michelangelo Diligenti, Soumali Roychowdhury, and Marco Gori. Integrating prior knowledge into deep learning. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 920–923. IEEE, 2017.

[Ellis *et al.*, 2018] Kevin Ellis, Lucas Morales, Mathias Sabl Meyer, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping domain-specific languages for neurally-guided bayesian program learning. In *Proceedings of the 2nd Workshop on Neural Abstract Machines and Program Induction*, 2018.

[Ellis *et al.*, 2020] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv preprint arXiv:2006.08381*, 2020.

[Evans and Grefenstette, 2018] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Intell. Res.*, 61:1–64, 2018.

[Fischer *et al.*, 2019] M. Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin T. Vechev. Dl2: Training and querying neural networks with logic. In *ICML*, 2019.

[Fletcher and Obradovic, 1993] Justin Fletcher and Zoran Obradovic. Combining prior symbolic knowledge and constructive neural network learning. *Connection Science*, 5(3-4):365–375, 1993.

[França *et al.*, 2014] Manoel VM França, Gerson Zaverucha, and Artur S d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine learning*, 94(1):81–104, 2014.

[Frasconi *et al.*, 2014] Paolo Frasconi, Fabrizio Costa, Luc De Raedt, and Kurt De Grave. klog: A language for logical and relational learning with kernels. *Artificial Intelligence*, 217:117–143, 2014.

[Fu, 1993] L. M. Fu. Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):173–182, 1993.

[Fu, 1995] Li Min Fu. Introduction to knowledge-based neural networks. *Knowledge-Based Systems*, 1995.

[Futia and Vetrò, 2020] Giuseppe Futia and Antonio Vetrò. On the integration of knowledge graphs into deep learning models for a more comprehensible ai—three challenges for future research. *Information*, 11(2), 2020.

[Garcez *et al.*, 2012] Artur S d'Avila Garcez, Krysia B Broda, and Dov M Gabbay. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media, 2012.

[Guimarães and Costa, 2021] Victor Guimarães and Vítor Santos Costa. Neurallog: a neural logic language. *arXiv preprint arXiv:2105.01442*, 2021.

[Guo *et al.*, 2016] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 192–202, 2016.

[Hájek, 2013] Petr Hájek. *Metamathematics of fuzzy logic*, volume 4. Springer Science & Business Media, 2013.

[Hamilton *et al.*, 2018] William L. Hamilton, P. Bajaj, M. Zitnik, Dan Jurafsky, and J. Leskovec. Embedding logical queries on knowledge graphs. In *NeurIPS*, 2018.

[Heckerman *et al.*, 2007] David Heckerman, Chris Meek, and Daphne Koller. Probabilistic entity-relationship models, prms, and plate models. *Introduction to statistical relational learning*, pages 201–238, 2007.

[Hinton *et al.*, 1995] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[Hogan *et al.*, 2020] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[Hu *et al.*, 2016] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. *ArXiv*, abs/1603.06318, 2016.

[Kitano, 2016] Hiroaki Kitano. Artificial intelligence to win the nobel prize and beyond: Creating the engine for scientific discovery. *AI magazine*, 37(1):39–49, 2016.

[Kramer *et al.*, 2001] Stefan Kramer, Nada Lavrač, and Peter Flach. *Propositionalization Approaches to Relational Data Mining*, pages 262–291. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[Krupka and Tishby, 2007] Eyal Krupka and Naftali Tishby. Incorporating prior knowledge on features into learning. In *AISTATS*, 2007.

[Kukačka *et al.*, 2017] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017.

[Kursuncu *et al.*, 2019] Ugur Kursuncu, Manas Gaur, and Amit Sheth. Knowledge infused learning (k-il): Towards deep incorporation of knowledge in deep learning. *arXiv preprint arXiv:1912.00512*, 2019.

[Lavrač *et al.*, 1991] Nada Lavrač, Sašo Džeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with linus. In *European Working Session on Learning*, pages 265–281. Springer, 1991.

[Lavrac *et al.*, 2020] Nada Lavrac, Blaz Skrlj, and Marko Robnik-Sikonja. Propositionalization and embeddings: two sides of the same coin. *Mach. Learn.*, 109(7):1465–1507, 2020.

[Lehmann *et al.*, 2015] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

[Li and Srikumar, 2020] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic. In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 2020.

[Li *et al.*, 2018] Lei Li, Min Feng, Lianwen Jin, Shenjin Chen, Lihong Ma, and Jiakai Gao. Domain knowledge embedding regularization neural networks for workload prediction and analysis in cloud computing. *J. Inf. Technol. Res.*, 11(4):137–154, October 2018.

[Lipton, 2016] Zachary C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[Liu *et al.*, 2018] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. *arXiv preprint arXiv:1812.10924*, 2018.

[Lodhi, 2013] Huma Lodhi. Deep relational machines. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.

[Manhaeve *et al.*, 2018] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31:3749–3759, 2018.

[Mehrabi *et al.*, 2021] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[Muggleton and de Raedt, 1994] Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679, 1994. Special Issue: Ten Years of Logic Programming.

[Muggleton *et al.*, 2012] Stephen Muggleton, Luc De Raedt, David Poole, Ivan Bratko, Peter Flach, Katsumi Inoue, and Ashwin Srinivasan. Ilp turns 20. *Machine learning*, 86(1):3–23, 2012.

[Muggleton, 1991] Stephen Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

[Muggleton, 1995] Stephen Muggleton. Inverse entailment and progol. *New generation computing*, 13(3-4):245–286, 1995.

[Muralidhar *et al.*, 2019] Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating Prior Domain Knowledge into Deep Neural Networks. In *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 2019.

[Neal, 1995] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, CAN, 1995. AAINN02676.

[Park *et al.*, 2019] Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. *Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks*, page 596–606. Association for Computing Machinery, New York, NY, USA, 2019.

[Pereira and Warren, 1980] Fernando CN Pereira and David HD Warren. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial intelligence*, 13(3):231–278, 1980.

[Raedt *et al.*, 2020] Luc de Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4943–4950. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Survey track.

[Rocktäschel *et al.*, 2014] Tim Rocktäschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. Low-dimensional embeddings of logic. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 45–49, Baltimore, MD, June 2014. Association for Computational Linguistics.

[Rocktäschel *et al.*, 2015] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[Rossi *et al.*, 2006] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[Serafini and Garcez, 2016] Luciano Serafini and Artur d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.

[Sheth *et al.*, 2019] A. Sheth, M. Gaur, U. Kursuncu, and R. Wickramarachchi. Shades of knowledge-infused learning for enhancing deep learning. *IEEE Internet Computing*, 23(6):54–63, 2019.

[Silvestri *et al.*, 2020] Mattia Silvestri, Michele Lombardi, and Michela Milano. Injecting domain knowledge in neural networks: a controlled experiment on a constrained problem. *arXiv preprint arXiv:2002.10742*, 2020.

[Sourek *et al.*, 2018a] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezný, Steven Schockaert, and Ondrej Kuzelka. Lifted relational neural networks: Efficient learning of latent relational structures. *J. Artif. Intell. Res.*, 62:69–100, 2018.

[Sourek *et al.*, 2018b] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezny, Steven Schockaert, and Ondrej Kuzelka. Lifted relational neural networks: Efficient learning of latent relational structures. *Journal of Artificial Intelligence Research*, 62:69–100, 2018.

[Sourek *et al.*, 2020] Gustav Sourek, Filip Zelezny, and Ondrej Kuzelka. Beyond graph neural networks with lifted relational neural networks. *arXiv preprint arXiv:2007.06286*, 2020.

[Srinivasan *et al.*, 2019] Ashwin Srinivasan, Lovekesh Vig, and Michael Bain. Logical explanations for deep relational machines using relevance information. *Journal of Machine Learning Research*, 20(130):1–47, 2019.

[Stevens *et al.*, 2020] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. Ai for science. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2020.

[Takeishi and Akimoto, 2018] Naoya Takeishi and Kosuke Akimoto. Knowledge-based distant regularization in learning probabilistic models. *arXiv preprint arXiv:1806.11332*, 2018.

[Tan, 1997] Ah Hwee Tan. Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing. *IEEE Transactions on Neural Networks*, 1997.

[Tomašev *et al.*, 2020] Nenad Tomašev, Julien Cornebise, Frank Hutter, Shakir Mohamed, Angela Picciariello, Bec Connelly, Danielle CM Belgrave, Daphne Ezer, Fanny Cachat van der Haert, Frank Mugisha, et al. Ai for social good: unlocking the opportunity for positive impact. *Nature Communications*, 11(1):1–6, 2020.

[Towell and Shavlik, 1993] Geoffrey G Towell and Jude W Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine learning*, 13(1):71–101, 1993.

[Towell and Shavlik, 1994] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[Towell *et al.*, 1990] Geofrey G Towell, Jude W Shavlik, and Michiel O Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the eighth National conference on Artificial intelligence*, volume 861866. Boston, MA, 1990.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[Vig *et al.*, 2017] Lovekesh Vig, Ashwin Srinivasan, Michael Bain, and Ankit Verma. An investigation into the role of domain-knowledge on the use of embeddings. In Nicolas Lachiche and Christel Vrain, editors, *Inductive Logic Programming - 27th International Conference, ILP 2017, Orléans, France, September 4-6, 2017, Revised Selected Papers*, volume 10759 of *Lecture Notes in Computer Science*, pages 169–183. Springer, 2017.

[Vishwanathan *et al.*, 2010] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[von Rueden *et al.*, 2019] Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems. *arXiv preprint arXiv:1903.12394*, 2019.

[Wang *et al.*, 2019] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*, WWW '19, page 3307–3313, New York, NY, USA, 2019. Association for Computing Machinery.

[Winters *et al.*, 2021] Thomas Winters, G. Marra, Robin Manhaeve, and L. D. Raedt. Deepstochlog: Neural stochastic logic programming. *ArXiv*, abs/2106.12574, 2021.

[Xie *et al.*, 2019] Yaqi Xie, Ziwei Xu, Mohan S. Kankanhalli, Kuldeep S. Meel, and Harold Soh. Embedding symbolic knowledge into deep networks. In *Advances in Neural Information Processing Systems*, 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 *35th International Conference on Machine Learning, ICML 2018*, 2018.

[Zhuang *et al.*, 2020] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.