# Logic Tensor Networks

# 逻辑张量网络

Samy Badreddine , Artur Avila Garce , Luciano Serafini , Michael Spranger

Samy Badreddine , Artur Avila Garce , Luciano Serafini , Michael Spranger

Sony Computer Science Laboratories Inc,3-14-13 Higashigotanda,141-0022, Tokyo, Japan

索尼计算机科学研究所，东京都港区东五反田3-14-13，141-0022，日本

Sony AI Inc,1-7-1 Konan,108-0075, Tokyo, Japan

索尼AI公司，东京都港区海岸1-7-1，108-0075，日本

City, University of London, Northampton Square, EC1V 0HB, London, United Kingdom

伦敦城市大学，Northampton Square，EC1V 0HB，伦敦，英国

Fondazione Bruno Kessler, Via Sommarive 18,38123, Trento, Italy

Bruno Kessler基金会，Via Sommarive 18，38123，特伦托，意大利

# Abstract

# 摘要

Attempts at combining logic and neural networks into neurosymbolic approaches have been on the increase in recent years. In a neurosymbolic system, symbolic knowledge assists deep learning, which typically uses a sub-symbolic distributed representation, to learn and reason at a higher level of abstraction. We present Logic Tensor Networks (LTN), a neurosymbolic framework that supports querying, learning and reasoning with both rich data and abstract knowledge about the world. LTN introduces a fully differentiable logical language, called Real Logic, whereby the elements of a first-order logic signature are grounded onto data using neural computational graphs and first-order fuzzy logic semantics. We show that LTN provides a uniform language to represent and compute efficiently many of the most important AI tasks such as multi-label classification, relational learning, data clustering, semi-supervised learning, regression, embedding learning and query answering. We implement and illustrate each of the above tasks with several simple explanatory examples using TensorFlow 2. The results indicate that LTN can be a general and powerful framework for neurosymbolic AI.

近年来，将逻辑与神经网络结合成神经符号方法的尝试不断增加。在神经符号系统中，符号知识协助深度学习，后者通常使用子符号分布式表示，以在更高层次的抽象上进行学习和推理。我们提出了逻辑张量网络（LTN），这是一个支持使用丰富数据和关于世界的抽象知识进行查询、学习和推理的神经符号框架。LTN引入了一种完全可微分的逻辑语言，称为实数逻辑，通过使用神经网络计算图和一阶模糊逻辑语义，将一阶逻辑签名中的元素映射到数据上。我们展示了LTN提供了一种统一的语言，可以有效地表示和计算许多最重要的AI任务，如多标签分类、关系学习、数据聚类、半监督学习、回归、嵌入学习和查询回答。我们使用TensorFlow 2实现了上述每个任务，并通过几个简单的解释性示例进行了说明。结果表明，LTN可以是神经符号AI的一种通用而强大的框架。

Keywords: Neurosymbolic AI, Deep Learning and Reasoning, Many-valued Logics.

关键词：神经符号AI，深度学习与推理，多值逻辑。

# 1. Introduction

# 1. 引言

Artificial Intelligence (AI) agents are required to learn from their surroundings and reason about what has been learned to make decisions, act in the world, or react to various stimuli. The latest Machine Learning (ML) has adopted mostly a pure sub-symbolic learning approach. Using distributed representations of entities, the latest ML performs quick decision-making without building a comprehensible model of the world. While achieving impressive results in computer vision, natural language, game playing, and multimodal learning, such approaches are known to be data inefficient and to struggle at out-of-distribution generalization. Although the use of appropriate inductive biases can alleviate such shortcomings, in general, sub-symbolic models lack comprehensibility. By contrast, symbolic AI is based on rich, high-level representations of the world that use human-readable symbols. By rich knowledge, we refer to logical representations which are more expressive than propositional logic or propositional probabilistic approaches, and which can express knowledge using full first-order logic, including universal and existential quantification , arbitrary -ary relations over variables, e.g. , and function symbols, e.g. father , etc. Symbolic AI has achieved success at theorem proving, logical inference, and verification. However, it also has shortcomings when dealing with incomplete knowledge. It can be inefficient with large amounts of inaccurate data and lack robustness to outliers. Purely symbolic decision algorithms usually have high computational complexity making them impractical for the real world. It is now clear that the predominant approach to ML, where learning is based on recognizing the latent structures hidden in the data, is insufficient and may benefit from symbolic AI [17]. In this context, neurosymbolic AI, which stems from neural networks and symbolic AI, attempts to combine the strength of both paradigms (see [16, 40, 54] for recent surveys). That is to say, combine reasoning with complex representations of knowledge (knowledge-bases, semantic networks, ontologies, trees, and graphs) with learning from complex data (images, time series, sensorimotor data, natural language). Consequently, a main challenge for neurosymbolic AI is the grounding of symbols, including constants, functional and relational symbols, into real data, which is akin to the longstanding symbol grounding problem [30].

人工智能（AI）代理需要从周围环境中学习并对其所学内容进行推理，以做出决策、在世界上行动或对各种刺激做出反应。最新的机器学习（ML）主要采用纯符号下学习的方法。通过使用实体的分布式表示，最新的ML能够快速做出决策，而无需构建一个可理解的世界模型。尽管在计算机视觉、自然语言、游戏和多模态学习方面取得了令人印象深刻的结果，但这些方法被认为在数据效率和分布外泛化方面存在困难。尽管使用适当的归纳偏置可以缓解这些不足，但一般来说，符号下模型缺乏可理解性。相比之下，符号AI基于丰富的、高级的世界表示，使用人类可读的符号。通过丰富的知识，我们指的是比命题逻辑或命题概率方法更具表现力的逻辑表示，它可以使用完整的一阶逻辑来表达知识，包括全称和存在量化 ，任意的 元关系，例如 ，以及函数符号，例如父亲 等。符号AI在定理证明、逻辑推理和验证方面取得了成功。然而，在处理不完整知识时也存在不足。它可能在处理大量不准确的数据时效率低下，并且对异常值缺乏鲁棒性。纯符号决策算法通常具有很高的计算复杂度，使其在现实世界中不切实际。现在很明显，当前ML的主要方法，即基于识别数据中隐藏的潜在结构的学习，是不够的，并可能从符号AI中受益[17]。在这种背景下，神经符号AI，源自神经网络和符号AI，试图结合这两种范式的优势（参见[16, 40, 54]的最近调查）。也就是说，结合推理与复杂知识表示（知识库、语义网络、本体、树和图）的学习，以及从复杂数据（图像、时间序列、传感器运动数据、自然语言）中学习。因此，神经符号AI的一个主要挑战是将符号，包括常量、函数和关系符号，映射到实际数据中，这类似于长期存在的符号映射问题[30]。

[[1]](#footnote-25)

Logic Tensor Networks (LTN) are a neurosymbolic framework and computational model that supports learning and reasoning about data with rich knowledge. In LTN, one can represent and effectively compute the most important tasks of deep learning with a fully differentiable first-order logic language, called Real Logic, which adopts infinitely many truth-values in the interval . In particular, LTN supports the specification and computation of the following AI tasks uniformly using the same language: data clustering, classification, relational learning, query answering, semi-supervised learning, regression, and embedding learning.

逻辑张量网络（Logic Tensor Networks，LTN）是一种神经符号框架和计算模型，支持使用丰富的知识对数据进行学习和推理。在LTN中，可以使用完全可微的一阶逻辑语言——实数逻辑（Real Logic）来表示和有效计算深度学习的最重要任务，该语言在区间 中采用无限多的真值。特别是，LTN支持使用同一种语言统一规范和计算以下AI任务：数据聚类、分类、关系学习、查询回答、半监督学习、回归和嵌入学习。

LTN and Real Logic were first introduced in [62]. Since then, LTN has been applied to different AI tasks involving perception, learning, and reasoning about relational knowledge. In [18, 19], LTN was applied to semantic image interpretation whereby relational knowledge about objects was injected into deep networks for object relationship detection. In [6], LTN was evaluated on its capacity to perform reasoning about ontological knowledge. Furthermore, [7] shows how LTN can be used to learn an embedding of concepts into a latent real space by taking into consideration ontological knowledge about such concepts. In [3], LTN is used to annotate a reinforcement learning environment with prior knowledge and incorporate latent information into an agent. In [42], authors embed LTN in a state-of-the-art convolutional object detector. Extensions and generalizations of LTN have also been proposed in the past years, such as LYRICS [47] and Differentiable Fuzzy Logic (DFL) [68,69]. LYRICS provides an input language allowing one to define background knowledge using a first-order logic where predicate and function symbols are grounded onto any computational graph. DFL analyzes how a large collection of fuzzy logic operators behave in a differentiable learning setting. DFL also introduces new semantics for fuzzy logic implications called sigmoidal implications, and it shows that such semantics outperform other semantics in several semi-supervised machine learning tasks.

LTN和Real Logic最初在[62]中引入。自那时起，LTN已被应用于涉及感知、学习和关系知识推理的不同AI任务。在[18, 19]中，LTN被应用于语义图像解释，其中关于对象的关系知识被注入深度网络以进行对象关系检测。在[6]中，评估了LTN在执行关于本体知识的推理方面的能力。此外，[7]展示了如何使用LTN通过考虑关于这些概念的本体知识来学习概念嵌入到潜在实数空间中。在[3]中，LTN被用于用先验知识注释强化学习环境并将潜在信息融入代理中。在[42]中，作者将LTN嵌入到最先进的卷积对象检测器中。在过去的几年里，也提出了LTN的扩展和泛化，例如LYRICS [47]和可微分模糊逻辑（DFL）[68,69]。LYRICS提供了一个输入语言，允许使用一阶逻辑定义背景知识，其中谓词和函数符号被映射到任何计算图上。DFL分析了大量模糊逻辑操作符在可微分学习环境中的行为。DFL还引入了模糊逻辑蕴涵的新语义，称为sigmoidal蕴涵，并展示了这种语义在几个半监督机器学习任务中优于其他语义。

This paper provides a thorough description of the full formalism and several extensions of LTN. We show using an extensive set of explanatory examples, how LTN can be applied to solve many ML tasks with the help of logical knowledge. In particular, the earlier versions of LTN have been extended with: (1) Explicit domain declaration: constants, variables, functions and predicates are now domain typed (e.g. the constants John and Paris can be from the domain of person and city, respectively). The definition of structured domains is also possible (e.g. the domain couple can be defined as the Cartesian product of two domains of persons); (2) Guarded quantifiers: guarded universal and existential quantifiers now allow the user to limit the quantification to the elements that satisfy some Boolean condition, e.g. (playsPiano enfantProdige ) restricts the quantification to the cases where age is lower than 10; (3) Diagonal quantification: Diagonal quantification allows the user to write statements about specific tuples extracted in order from variables. For example, if the variables capital and country both have instances such that the -th instance of capital corresponds to the -th instance of country, one can write Diag(capital, country) capitalOf(capital, country).

本文详细描述了LTN的完整形式主义及其几个扩展。我们通过一组广泛的解释性示例展示了如何利用逻辑知识将LTN应用于解决许多机器学习任务。特别是，LTN的早期版本已经扩展了以下内容：（1）显式域声明：常量、变量、函数和谓词现在具有域类型（例如，常量John和Paris可以分别属于人和城市域）。也可以定义结构化域（例如，可以将伴侣域定义为两个人域的笛卡尔积）；（2）守卫量词：守卫的全称和存在量词现在允许用户将量化限制为满足某些布尔条件的元素，例如 (playsPiano enfantProdige ) 将量化限制在年龄小于10的案例中；（3）对角量化：对角量化允许用户编写关于从 变量中按顺序提取的特定元组的陈述。例如，如果变量capital和country都有 实例，使得capital的第 个实例对应于country的第 个实例，则可以编写 Diag(capital, country) capitalOf(capital, country)。

Inspired by the work of [69], this paper also extends the product t-norm configuration of LTN with the generalized mean aggregator, and it introduces solutions to the vanishing or exploding gradient problems. Finally, the paper formally defines a semantic approach to refutation-based reasoning in Real Logic to verify if a statement is a logical consequence of a knowledge base. Example 4.8 proves that this new approach can better capture logical consequences compared to simply querying unknown formulas after learning (as done in [6]).

受[69]的工作启发，本文还扩展了LTN的乘积t-范数配置，加入了广义均值聚合器，并引入了解决梯度消失或爆炸问题的方法。最后，本文正式定义了一种基于反驳的推理的语义方法，用于在实逻辑中验证一个陈述是否是知识库的逻辑后果。示例4.8证明了这种方法与在学习后简单地查询未知公式（如[6]中所做）相比，能更好地捕获逻辑后果。

The new version of LTN has been implemented in TensorFlow 2 [1]. Both the LTN library and the code for the examples used in this paper are available at https://github.com/ logictensornetworks/logictensornetworks

新版本的LTN已在TensorFlow 2中实现[1]。LTN库以及本文中使用的示例代码可在 https://github.com/logictensornetworks/logictensornetworks 获取。

The remainder of the paper is organized as follows: In Section 2, we define and illustrate Real Logic as a fully-differentiable first-order logic. In Section 3, we specify learning and reasoning in Real Logic and its modeling into deep networks with Logic Tensor Networks (LTN). In Section 4 we illustrate the reach of LTN by investigating a range of learning problems from clustering to embedding learning. In Section 5, we place LTN in the context of the latest related work in neurosymbolic AI. In Section 6 we conclude and discuss directions for future work. The Appendix contains information about the implementation of LTN in TensorFlow 2, experimental set-ups, the different options for the differentiable logic operators, and a study of their relationship with gradient computations.

本文的其余部分组织如下：在第2节中，我们定义并说明了作为全微分一阶逻辑的Real Logic。在第3节中，我们指定了在Real Logic中的学习和推理方法，以及将其建模为深度网络中的逻辑张量网络（LTN）。在第4节中，我们通过研究从聚类到嵌入学习的一系列学习问题，展示了LTN的应用范围。在第5节中，我们将LTN置于神经符号AI最新相关工作的背景下。在第6节中，我们总结并讨论未来的研究方向。附录包含关于在TensorFlow 2中实现LTN、实验设置、不同可微分逻辑运算符的选项以及它们与梯度计算之间关系的研究信息。

# 2. Real Logic

# 2. Real Logic

# 2.1. Syntax

# 2.1. 语法

Real Logic forms the basis of Logic Tensor Networks. Real Logic is defined on a first-order language with a signature that contains a set of constant symbols (objects), a set of functional symbols, a set of relational symbols (predicates), and a set of variable symbols. -formulas allow us to specify relational knowledge with variables, e.g. the atomic formula is\_friend may state that the person is a friend of the person , the formula states that the relation is\_friend is symmetric, and the formula states that every person has a friend that is Italian. Since we are interested in learning and reasoning in real-world scenarios where degrees of truth are often fuzzy and exceptions are present, formulas can be partially true, and therefore we adopt fuzzy semantics.

实数逻辑是逻辑张量网络的基础。实数逻辑定义在具有一组常量符号（对象） 、一组函数符号 、一组关系符号（谓词） 和一组变量符号 的第一阶语言 上。 形式的公式允许我们使用变量指定关系知识，例如原子公式 is\_friend 可以表示 是 的朋友，公式 表示关系is\_friend是对称的，公式 表示每个人都有一个意大利朋友。由于我们关注的是在真实世界场景中学习和推理，这些场景中真实度往往模糊且存在例外，因此公式可以是部分真实的，我们因此采用模糊语义。

Objects can be of different types. Similarly, functions and predicates are typed. Therefore, we assume there exists a non-empty set of symbols called domain symbols. To assign types to the elements of we introduce the functions and such that:

对象可以是不同类型的。类似地，函数和谓词也是有类型的。因此，我们假设存在一个非空符号集 ，称为域符号。为了给 的元素分配类型，我们引入了函数 和 ，使得：

* . Intuitively, and returns the domain of a variable or a constant .
* 。直观上， 和 返回变量 或常量 的域。
* , where is the Kleene star of , that is the set of all finite sequences of symbols in . Intuitively, and returns the domains of the arguments of a
* ，其中 是 的克林星，即 中所有有限序列的符号集。直观上， 和 返回函数 或谓词 的参数的域。

function or a predicate . If takes two arguments (for example, ), returns two domains, one per argument.

如果 接受两个参数（例如 ）， 将返回两个域，每个参数一个。

* . Intuitively, returns the range of a function symbol.
* . 直观上， 返回函数符号的范围。

Real Logic may also contain propositional variables, as follows: if is a 0-ary predicate with (the empty sequence of domains) then is a propositional variable (an atom with truth-value in the interval ).

实数逻辑也可能包含命题变量，如下：如果 是一个 0-元谓词，带有 （域的空序列），那么 是一个命题变量（在 区间内具有真值的原子）。

A term is constructed recursively in the usual way from constant symbols, variables, and function symbols. An expression formed by applying a predicate symbol to an appropriate number of terms with appropriate domains is called an atomic formula, which evaluates to true or false in classical logic and a number in in the case of Real Logic. We define the set of terms of the language as follows:

项以通常的方式递归地从常量符号、变量和函数符号构造。由将谓词符号应用于适当数量的项和适当的域形成的表达式称为原子公式，在经典逻辑中它求值为真或假，在实数逻辑的情况下求值为 中的一个数。我们定义语言中的项的集合如下：

* each element of is a term of the domain ;
* 的每个元素 是域 的项；
* if is a term of domain for then (the sequence composed of followed by and so on, up to ) is a term of the domain ;
* 如果 是域 中关于 的项，那么 （由 开始，接着是 等等，直到 ）是域 的项；
* if is a term of the domain then is a term of the domain .
* 如果 是域 的项，那么 是域 的项。

We allow the following set of formula in :

我们允许 中的以下公式集合：

* is an atomic formula for any terms and with ;
* 对于任何项 和 以及 ， 是一个原子公式；
* is an atomic formula if ;
* 如果 ，则 是一个原子公式；
* If and are formula and are distinct variable symbols then and are formula, where is a unary connective, is a binary connective and is a quantifier.

如果 和 是公式， 是 不同的变量符号，那么 和 是公式，其中 是一元连接词， 是二元连接词， 是量词。

We use (negation), (conjunction, disjunction, implication and biconditional, respectively) and (universal and existential, respectively).

我们使用 （否定）、 （分别表示合取、析取、蕴涵和双条件）以及 （分别表示全称量词和存在量词）。

Example 1. Let Town denote the domain of towns in the world and People denote the domain of living people. Suppose that contains the constant symbols Alice, Bob and Charlie of domain People, and Rome and Seoul of domain Town. Let be a variable of domain People and be a variable of domain Town. The term (i.e. the sequence followed by ) has domain People, Town which denotes the Cartesian product between People and Town (People Town). Alice, Rome is interpreted as an element of the domain People, Town. Let lives in be a predicate with input domain (lives\_in) = People, Town. lives\_in(Alice, Rome) is a well-formed expression, whereas lives\_in(Bob, Charlie) is not.

示例 1. 令 Town 表示世界上城镇的域，People 表示活着的人的域。假设 包含域 People 的常量符号 Alice、Bob 和 Charlie，以及域 Town 的 Rome 和 Seoul。令 是域 People 的变量， 是域 Town 的变量。术语 （即 后跟 的序列）具有域 People, Town，它表示 People 和 Town 之间的笛卡尔积（People Town）。Alice, Rome 被解释为域 People, Town 的一个元素。令 lives in 是一个输入域为 （lives\_in）= People, Town 的谓词。lives\_in(Alice, Rome) 是一个合式表达式，而 lives\_in(Bob, Charlie) 不是。

# 2.2. Semantics of Real Logic

# 2.2. 实逻辑的语义

The semantics of Real Logic departs from the standard abstract semantics of First-order Logic (FOL). In Real Logic, domains are interpreted concretely by tensors in the real field T Every object denoted by constants, variables, and terms, is interpreted as a tensor of real values. Functions are interpreted as real functions or tensor operations. Predicates are interpreted as functions or tensor operations projecting onto a value in the interval .

实逻辑的语义与一阶逻辑（FOL）的标准抽象语义有所不同。在实逻辑中，域通过实数域 T 中的张量具体解释。每个由常量、变量和项表示的对象都被解释为实数值的张量。函数被解释为实函数或张量操作。谓词被解释为投影到区间 上的值的功能或张量操作。

[[2]](#footnote-32)

To emphasize the fact that in Real Logic symbols are grounded onto real-valued features, we use the term grounding, denoted by , in place of interpretation . Notice that this is different from the common use of the term grounding in logic, which indicates the operation of replacing the variables of a term or formula with constants or terms containing no variables. To avoid confusion, we use the synonym instantiation for this purpose. associates a tensor of real numbers to any term of , and a real number in the interval to any formula of . Intuitively, are the numeric features of the objects denoted by , and represents the system’s degree of confidence in the truth of ; the higher the value, the higher the confidence.

为了强调在实数逻辑中符号是基于实值特征进行固化的，我们使用术语“固化”，表示为 ，来代替解释 。请注意，这与逻辑中术语固化的通常用法不同，后者指的是将术语或公式中的变量替换为常量或不含变量的项的操作。为了避免混淆，我们为此目的使用同义词“实例化”。 将实数张量与 的任何项相关联，并将区间 中的实数与 的任何公式 相关联。直观上， 是由 表示的对象的数值特征，而 表示系统对 真实的置信度；值越大，置信度越高。

# 2.2.1. Grounding domains and the signature

# 2.2.1. 固化域和签名

A grounding for a logical language on the set of domains provides the interpretation of both the domain symbols in and the non-logical symbols in .

对于逻辑语言 在域集 上的一个固化，提供了 中的域符号和非逻辑符号的解释。

Definition 1. A grounding associates to each domain a set .

定义 1. 一个固化 将每个域 与一个集合 相关联。

For every , that is .

对于每个 ，即 。

Notice that the elements in may be tensors of any rank and any dimensions , as denotes the Kleene star of 3

请注意， 中的元素可以是任意秩 和任意维度的张量，因为 表示 的克林星。

Example 2. Let digit\_images denote a domain of images of handwritten digits. If we use images of RGB pixels, then (digit\_images) . Let us consider the predicate is\_digit . The terms have domains digit\_images, digits. Any input to the predicate is a tuple in (digit\_images, digits) (digit\_images) (digits).

示例 2. 令 digit\_images 表示手写数字图像的领域。如果我们使用 RGB 像素图像，那么 (digit\_images) 。让我们考虑谓词 is\_digit 。术语 的领域为 digit\_images、digits。谓词的任何输入都是 (digit\_images, digits) (digit\_images) (digits) 中的元组。

A grounding assigns to each constant symbol , a tensor in the domain ; It assigns to a variable a finite sequence of tensors , each in . These tensors represent the instances of . Differently from in FOL where a variable is assigned to a single value of the domain of interpretations at a time, in Real Logic a variable is assigned to a sequence of values in its domain, the examples of . A grounding assigns to a function symbol a function taking tensors from as input, and producing a tensor in as output. Finally, a grounding assigns to a predicate symbol a function taking tensors from as input, and producing a truth-value in the interval as output.

一个 grounding 为每个常量符号 分配一个领域 中的张量 ；它为变量 分配一个有限张量序列 ，每个张量都在 中。这些张量代表 的实例。与 FOL 中变量一次被分配为解释域中的单个值不同，在 Real Logic 中，变量被分配为其领域中的一序列值，即 的例子。一个 grounding 为函数符号 分配一个函数，该函数接收 中的张量作为输入，并产生 中的张量作为输出。最后，一个 grounding 为谓词符号 分配一个函数，该函数接收 中的张量作为输入，并产生区间 中的真值作为输出。

Definition 2. A grounding of is a function defined on the signature of that satisfies the following conditions:

定义 2。一个 grounding 的 是定义在 的签名上并满足以下条件的函数：

1. for every variable symbol , with . Notice that is a sequence and not a set, meaning that the same value of can occur multiple times in , as is usual in a Machine Learning data set with "attributes" and "values";

1. 对于每个变量符号 ，带有 。注意 是一个序列而不是集合，这意味着 的相同值可以在 中多次出现，这在具有“属性”和“值”的机器学习数据集中是常见的；

[[3]](#footnote-35)

2. for every function symbol ;

2. 对于每一个函数符号 ；

3. for every predicate symbol .

3. 对于每一个谓词符号 。

If a grounding depends on a set of parameters , we denote it as or interchangeably. Section 4 describes how such parameters can be learned using the concept of satisfiability.

如果一个基项依赖于一组参数 ，我们可以将其表示为 或 ，两者可互换。第4节描述了如何使用可满足性的概念来学习这样的参数。

# 2.2.2. Grounding terms and atomic formulas

# 2.2.2. 基项和原子公式

We now extend the definition of grounding to all first-order terms and atomic formulas. Before formally defining these groundings, we describe on a high level what happens when grounding terms that contain free variables.

我们现在将基项的定义扩展到所有一阶项和原子公式。在正式定义这些基项之前，我们在高层次描述当基项包含自由变量时会发生什么。

Let be a variable that denotes people. As explained in Definition 2, is grounded as an explicit sequence of instances . Consequently, a term height is also grounded in height values, each corresponding to one instance. We can generalize to expressions with multiple free variables, as shown in Example 3.

设 是表示人的变量。如定义2中所述， 被基项化为 实例 的显式序列。因此，一个项高度 也在 高度值中基项化，每个值对应一个实例。我们可以推广到具有多个自由变量的表达式，如示例3所示。

In the formal definition below, instead of considering a single term at a time, it is convenient to consider sequences of terms and define the grounding on (with the definition of the grounding of a single term being derived as a special case). The fact that the sequence of terms contains distinct variables is denoted by . The grounding of , denoted by , is a tensor with corresponding axes, one for each free variable, defined as follows:

在下面的形式定义中，而不是一次考虑一个术语，考虑术语序列 并在 上定义接地（将单个术语的接地定义作为特例导出）更为方便。术语序列 包含 个不同变量 的事实由 表示。 的接地，表示为 ，是一个具有 个对应轴的张量，每个自由变量一个，定义如下：

Definition 3. Let be a sequence of terms containing distinct variables . Let each term in contain variables .

定义3。设 为包含 个不同变量 的术语序列 。设 中的每个术语 包含 个变量 。

* is a tensor with dimensions such that the element of this tensor indexed by , written as , is equal to the concatenation of for ;
* 是一个维度为 的张量，该张量的元素由 索引，记作 ，等于 对于 的连接；
* , i.e. the element-wise application of to ;
* ，即对 进行逐元素应用 ；
* , i.e. the element-wise application of to .
* ，即对 进行逐元素应用 。

If term contains variables selected from then can be obtained from with an appropriate mapping of indices to .

如果术语 包含从 中选择的 个变量 ，则 可以通过对 的索引 到 的适当映射获得。

[[4]](#footnote-38)

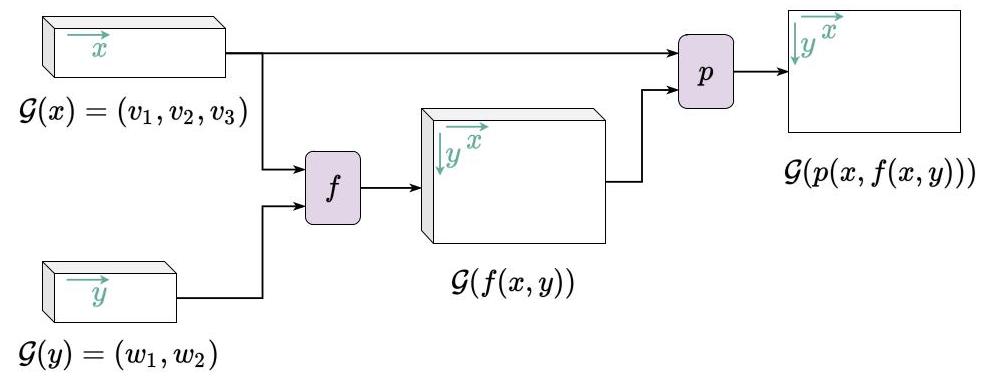


Figure 1: Illustration of Example 3 - and indicate dimensions associated with the free variables and . A tensor representing a term that includes a free variable will have an axis . One can index to obtain results calculated using each of the or values of . In our graphical convention, the depth of the boxes indicates that the tensor can have feature dimensions (refer to the end of Example 3).

图 1：示例 3 的说明 - 和 指的是与自由变量 和 相关的维度。表示包含自由变量 的项的张量将有一个轴 。可以索引 来获取使用 的每个 或 值计算出的结果。在我们图形化的约定中，框的深度表明张量可以有特征维度（参考示例 3 的结尾）。

Example 3. Suppose that contains the variables and , the function , the predicate and the set of domains . Let and . In what follows, an example of the grounding of and is shown on the left, and the grounding of some examples of possible terms and atomic formulas is shown on the right.

示例 3。假设 包含变量 和 ，函数 ，谓词 以及域的集合 。令 和 。以下是一个关于 和 的例子的基设定展示在左侧，右侧展示了某些可能的项和原子公式的基设定示例。

Notice the dimensions of the results. and return values, one for each combination of individuals that occur in the variables. For functions, we can have additional dimensions associated to the output domain. Let us suppose a different grounding such that . Then the dimensions of would have been , where are the dimensions for indexing the free variables and are dimensions associated to the output domain of . Let us call the latter feature dimensions, as captioned in Figure 1. Notice that will always return a tensor with the exact dimensions because, under any grounding, a predicate always returns a value in . Therefore, as the "feature dimensions" of predicates is always 1, we choose to "squeeze it" and not to represent it in our graphical convention (see Figure 1, the box output by the predicate has no depth).

注意结果的大小。 和 返回 值，每个值对应于变量中出现的个体的每种组合。对于函数，我们可以有与输出域相关的额外维度。假设有另一种不同的基设定，使得 。那么 的维度将是 ，其中 是索引自由变量的维度， 是与 输出域相关的维度。让我们称后一种维度为特征维度，如图1所示。注意 总是会返回一个确切维度为 的张量，因为在任何基设定下，谓词总是返回 中的一个值。因此，由于谓词的“特征维度”总是1，我们选择“挤压”它，不在我们的图形约定中表示（见图1，谓词输出的框没有深度）。

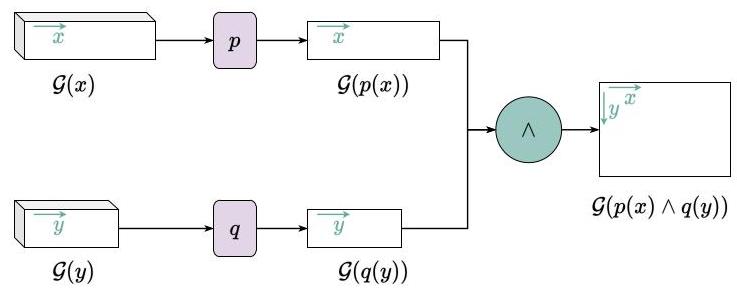


Figure 2: Illustration of an element-wise operator implementing conjunction . We assume that and are two different variables. The result has one number in the interval to every combination of individuals from and .

图2：逐元素运算符实现合取 的示例。我们假设 和 是两个不同的变量。结果是在 区间内的一个数，对应于 和 中个体的每种组合。

# 2.2.3. Connectives and Quantifiers

# 2.2.3. 连接词和量词

The semantics of the connectives is defined according to the semantics of first-order fuzzy logic [28]. Conjunction , disjunction , implication and negation are associated, respectively, with a t-norm , a t-conorm , a fuzzy implication and a fuzzy negation operation FuzzyOp . Definitions of some common fuzzy operators are presented in Appendix B Let and be two formulas with free variables and , respectively. Let us assume that the first variables are common to and . Recall that and denote the set of unary and binary connectives, respectively. Formally:

连接词的语义是根据一阶模糊逻辑的语义来定义的 [28]。合取 、析取 、蕴涵 和否定 分别与 t-范数 、t-共范数 、模糊蕴涵 和模糊否定 操作 FuzzyOp 相关联。附录B中给出了某些常见模糊运算符的定义。设 和 是分别具有自由变量 和 的两个公式。假设前 个变量是 和 共有的。回顾 和 分别表示一元和二元连接词的集合。正式地：

In (2), denote the indices of the common variables, denote the indices of the variables appearing only in , and denote the indices of the variables appearing only in . Intuitively, is a tensor whose elements are obtained by applying element-wise to every combination of individuals from and (see Figure 2).

在 (2) 中， 表示 共同变量的索引， 表示只出现在 中的变量的索引， 表示只出现在 中的变量的索引。直观上， 是一个张量，其元素是通过将 逐元素应用于 和 的每个个体组合得到的（见图2）。

The semantics of the quantifiers is defined with the use of aggregation. Let Agg be a symmetric and continuous aggregation operator, . An analysis of suitable

量词 的语义是通过使用聚合来定义的。设 Agg 为对称且连续的聚合运算符 。对适合的运算符的分析

aggregation operators is presented in Appendix Appendix B. For every formula containing free variables, suppose, without loss of generality, that quantification applies to the first variables. We shall therefore apply Agg to the first axes of , as follows:

附录B中介绍了聚合算子的内容。对于每个包含 自由变量的公式，假设不失一般性，量化应用于前 个变量。因此，我们将对 的前 个轴应用 Agg，如下所示：

where is the aggregation operator associated with the quantifier . Intuitively, we obtain by reducing the dimensions associated with using the operator (see Figure 3).

其中 是与量化词 相关联的聚合算子。直观上，我们通过使用算子 降低与 相关的维度来获得 （见图3）。

Notice that the above grounded semantics can assign different meanings to the three formulas:

注意，上述具体的语义可以为三个公式分配不同的含义：

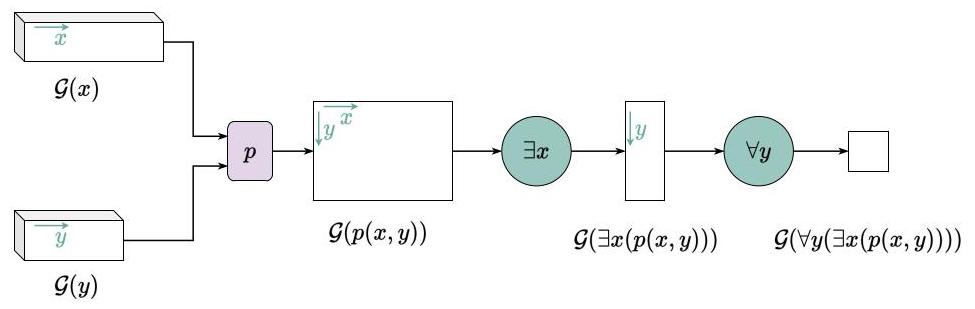


Figure 3: Illustration of an aggregation operation implementing quantification over variables and . We assume that and have different domains. The result is a single number in the interval .

图3：展示了实现量化 对变量 和 的聚合操作的示例。我们假设 和 有不同的定义域。结果是一个在 区间内的单一数值。

The semantics of the three formulas will coincide if the aggregation operator is bi-symmetric. LTN also allows the following form of quantification, here called diagonal quantification (Diag):

如果聚合算子是双对称的，那么三个公式的语义将会一致。LTN 还允许以下形式的量化，这里称为对角量化（Diag）：

quantifies over specific tuples such that the -th tuple contains the -th instance of each of the variables in the argument of Diag, under the assumption that all variables in the argument are grounded onto sequences with the same number of instances. is called diagonal quantification because it quantifies over the diagonal of along the axes associated with , although in practice only the diagonal is built and not the entire , as shown in Figure 4 For example, given a data set with samples and target labels , if looking to write a statement that holds true for each pair of sample and label, one can write given that . As another example, given two variables and whose groundings contain 10 instances of and each, the expression produces 10 results such that the -th result corresponds to the -th instances of each grounding. Without Diag, the expression would be evaluated for all combinations of the elements in and 5 Diag will find much application in the examples and experiments to follow.

对特定的元组进行量化，使得第 个元组包含 Diag 参数中每个变量的第 个实例，假设 Diag 参数中的所有变量都接地到具有相同数量的实例的序列上。 被称为对角量化，因为它量化了沿着与 相关的轴的 的对角线，虽然在实践中只构建对角线而非整个 ，如图 4 所示。例如，给定一个包含样本 和目标标签 的数据集，如果想要编写一个对每个样本和标签对都成立的语句 ，可以编写 ，前提是 。另一个例子是，给定两个变量 和 ，它们的接地包含 和 各 10 个实例，表达式 产生 10 个结果，其中第 个结果对应于每个接地的第 个实例。如果没有 Diag，表达式将为 中 和 元素的所有 组合进行评估。Diag 在接下来的示例和实验中将会得到广泛应用。

# 2.3. Guarded Quantifiers

# 2.3. 有条件量词

In many situations, one may wish to quantify over a set of elements of a domain whose grounding satisfy some condition. In particular, one may wish to express such condition using formulas of the language of the form:

在许多情况下，人们可能希望对满足某些条件的域中的元素集合进行量化。特别是，人们可能希望使用语言形式的公式来表达这样的条件：

The grounding of such a formula is obtained by aggregating the values of parent only for the instances of that satisfy the condition , that is:

这样公式的接地是通过仅聚集满足条件 的 实例的父 的值来获得的，即：

[[5]](#footnote-52)

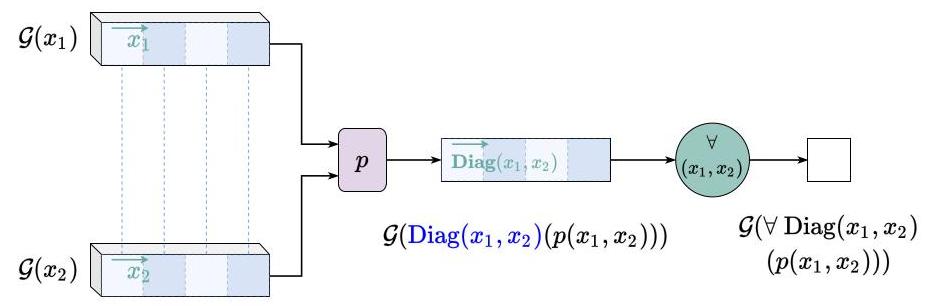


Figure 4: Diagonal Quantification: Diag quantifies over specific tuples only, such that the -th tuple contains the -th instances of the variables and in the groundings and , respectively. Diag assumes, therefore, that and have the same number of instances as in the case of samples and their labels in a typical supervised learning tasks.

图 4：对角量化：Diag 仅量化特定的元组，使得第 个元组包含第 个实例的变量 和 分别在基 和 中。因此，Diag 假设 和 与典型监督学习任务中的样本 及其标签 一样具有相同数量的实例。

The evaluation of which tuple is safe is purely symbolic and non-differentiable. Guarded quantifiers operate over only a subset of the variables, when this symbolic knowledge is crisp and available. More generally, in what follows, is a symbol representing the condition, which we shall call a mask, and associates a function 6 returning a Boolean to .

判断哪个元组是安全的完全基于符号且不可微分。带条件的量词仅在符号知识明确且可用时，操作变量的一个子集。更一般地，在接下来的内容中， 是一个表示条件的符号，我们将其称为掩码，而 将一个返回布尔值的函数 6 与 关联起来。

Notice that the semantics of a guarded sentence is different than the semantics of . In crisp and traditional FOL, the two statements would be equivalent. In Real Logic, they can give different results. Let be a sequence of 3 values, and . Only the second and third instances of are safe, that is, are in the masked subset. Let be defined using the Reichenbach operator and be defined using the mean operator. We have whereas . Also, in the computational graph of the guarded sentence, there are no gradients attached to the instances that do not verify the mask. Similarly, the semantics of is not equivalent to that of .

注意，守护句 的语义与 的语义不同。在清晰且传统的FOL中，这两个陈述将是等价的。在实数逻辑中，它们可能给出不同的结果。设 为一个包含3个值的序列， 和 。只有 的第二和第三次实例是安全的，即它们在掩码子集中。设 使用Reichenbach算子 定义，而 使用均值算子定义。我们有 而不是 。此外，在守护句的计算图中，没有验证掩码的实例上附加梯度。类似地， 的语义也不等同于 的语义。

[[6]](#footnote-56)

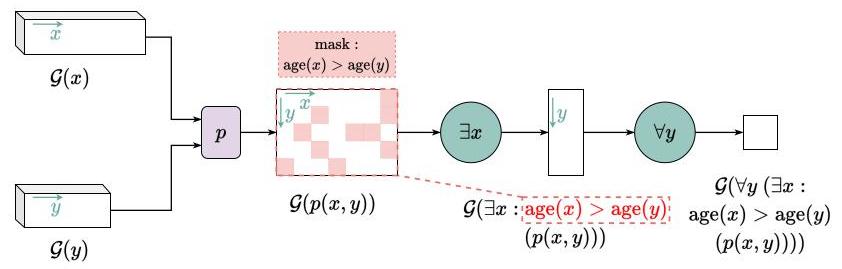


Figure 5: Example of Guarded Quantification: One can filter out elements of the various domains that do not satisfy some condition before the aggregation operators for and are applied.

图5：守护量化的示例：可以在应用 和 的聚合运算符之前，过滤掉不满足某些条件的各种域中的元素。

# 2.4. Stable Product Real Logic

# 2.4. 稳定产品实数逻辑

It has been shown in [69] that not all first-order fuzzy logic semantics are equally suited for gradient-descent optimization. Many fuzzy logic operators can lead to vanishing or exploding gradients. Some operators are also single-passing, in that they propagate gradients to only one input at a time.

已有研究 [69] 表明，并非所有的一阶模糊逻辑语义都同样适用于梯度下降优化。许多模糊逻辑运算符可能导致梯度消失或爆炸。某些运算符也是单次传递的，即它们一次只将梯度传播到一个输入。

In general, the best performing symmetric configuration 7 for the connectives uses the product t-norm for conjunction, its dual t-conorm for disjunction, standard negation , and the Reichenbach implication (the corresponding S-Implication to the above operators). This subset of Real Logic where the grounding of the connectives is restricted to the product configuration is called Product Real Logic in [69]. Given and two truth-values in :

通常，表现最佳的对称配置7对于连接词使用乘积t-范数 作为合取，其对应t-余范数 作为析取，标准否定 以及Reichenbach蕴涵 （上述运算符的对应S-蕴涵）。在文献[69]中，将连接词的基设定限制为乘积配置的实逻辑子集称为乘积实逻辑。给定 和 在 中的两个真值：

Appropriate aggregators for and are the generalized mean with to approximate the existential quantification, and the generalized mean w.r.t. the error with to approximate the universal quantification. They can be understood as a smooth maximum and a smooth minimum, respectively. Given truth-values all in :

对于 和 的适当聚合器是广义均值 以 逼近存在量词，以及关于误差 的广义均值以 逼近全称量词。它们分别可以理解为平滑最大值和平滑最小值。给定 在 中的所有真值 ：

measures the power of the deviation of each value from the ground truth 1 . With , it is equivalent to , where RMSE is the root-mean-square error, is the vector of truth-values and 1 is a vector of .

衡量每个值与真实值1的偏差的力度。使用 ，它等同于 ，其中RMSE是均方根误差， 是真值向量，1是 的向量。

[[7]](#footnote-62)

The intuition behind the choice of is that the higher that is, the more weight that (resp. will give to true (resp. false) truth-values, converging to the max (resp. min) operator. Therefore, the value of can be seen as a hyper-parameter as it offers flexibility to account for outliers in the data depending on the application.

选择的直觉是， 越高， （分别地 ）赋予真（分别地假）的真值的权重就越大，趋近于最大（分别地最小）操作符。因此， 的值可以被视为一个超参数，因为它提供了灵活性，能够根据应用的不同来考虑数据中的异常值。

Nevertheless, Product Real Logic still has the following gradient problems: has vanishing gradients on the edge case has vanishing gradients on the edge case ; has vanishing gradients on the edge case has exploding gradients when tends to has exploding gradients when tends to 0 (see Appendix C for details).

尽管如此，产品实数逻辑仍然存在以下梯度问题： 在边缘情况具有消失梯度； 在边缘情况具有消失梯度； ； 在边缘情况具有消失梯度； 当 趋近于 时具有爆炸梯度；当 趋近于 0 时具有爆炸梯度（详见附录 C）。

To address these problems, we define the projections and below with an arbitrarily small positive real number:

为了解决这些问题，我们定义了以下投影 和 ，其中 是任意小的正实数：

We then derive the following stable operators to produce what we call the Stable Product Real Logic configuration:

然后我们推导出以下稳定的操作符，以产生我们称之为稳定产品实数逻辑配置的东西：

It is important noting that the conjunction operator in stable product semantics is not a T-norm 8 does not satisfy identity in since for any , although can be chosen arbitrarily small. In the experimental evaluations reported in Section 4, we find that the adoption of the stable product semantics is an important practical step to improve the numerical stability of the learning system.

重要的是要注意，稳定产品语义中的合取操作符不是 T-范数 不满足 的恒等性，因为对于任何 ，尽管 可以任意小。在第 4 节报告的实验评估中，我们发现采用稳定产品语义是改善学习系统数值稳定性的一个重要实际步骤。

# 3. Learning, Reasoning, and Querying in Real Logic

# 3. 实数逻辑中的学习、推理和查询

In Real Logic, one can define the tasks of learning, reasoning and query-answering. Given a Real Logic theory that represents the knowledge of an agent at a given time, learning is the task of making generalizations from specific observations obtained from data. This is often called inductive inference. Reasoning is the task of deriving what knowledge follows from the facts which are currently known. Query answering is the task of evaluating the truth value of a certain logical expression (called a query), or finding the set of objects in the data that evaluate a certain expression to true. In what follows, we define and exemplify each of these tasks. To do so, we first need to specify which types of knowledge can be represented in Real Logic.

在实数逻辑中，可以定义学习、推理和查询应答的任务。给定一个在特定时间代表智能体知识的实数逻辑理论，学习是从数据中获取的特定观察中进行概括的任务。这通常被称为归纳推理。推理是从当前已知的事实中推导出知识的任务。查询应答是评估某个逻辑表达式（称为查询）的真值，或找到数据中使某个表达式为真的对象集合的任务。以下，我们定义并示例这些任务。为此，我们首先需要指定哪些类型的知识可以在实数逻辑中表示。

[[8]](#footnote-65)

# 3.1. Representing Knowledge with Real Logic

# 3.1. 使用实数逻辑表示知识

In logic-based knowledge representation systems, knowledge is represented by logical formulas whose intended meanings are propositions about a domain of interest. The connection between the symbols occurring in the formulas and what holds in the domain is not represented in the knowledge base and is left implicit since it does not have any effect on the logic computations. In Real Logic, by contrast, the connection between the symbols and the domain is represented explicitly in the language by the grounding , which plays an important role in both learning and reasoning. is an integral part of the knowledge represented by Real Logic. A Real Logic knowledge base is therefore defined by the formulas of the logical language and knowledge about the domain in the form of groundings obtained from data. The following types of knowledge can be represented in Real Logic.

在基于逻辑的知识表示系统中，知识由逻辑公式表示，这些公式的预期含义是关于某个兴趣领域的命题。公式中出现的符号与领域中的实际情况之间的联系在知识库中没有表示，而是隐含的，因为它对逻辑计算没有影响。相比之下，在实数逻辑中，符号与领域之间的联系通过语言中的 grounding 明确表示，这在学习和推理中都扮演着重要角色。 是实数逻辑表示的知识的一个组成部分。因此，实数逻辑知识库由逻辑语言的公式和数据中获得的接地知识组成。以下类型的知识可以在实数逻辑中表示。

# 3.1.1. Knowledge through symbol groundings

# 3.1.1. 通过符号接地获取知识

Boundaries for domain grounding. These are constraints specifying that the value of a certain logical expression must be within a certain range. For instance, one may specify that the domain must be interpreted in the hyper-cube or in the standard -simplex, i.e. the set such that . Other intuitive examples of range constraints include the elements of the domain "colour" grounded onto points in such that every element is associated with the triplet of values with , or the range of a function as an integer between 0 and 100 .

领域边界的确定。这些是约束条件，指定某个逻辑表达式的值必须在一定的范围内。例如，可以指定领域 必须在 超立方体或在标准的 单纯形中解释，即集合 使得 。范围约束的其他直观例子包括将领域 "颜色" 的元素映射到 上的点，使得每个元素与三值组 相关联，且 ，或者函数 的范围作为介于 0 到 100 之间的整数。

Explicit definition of grounding for symbols. Knowledge can be more strictly incorporated by fixing the grounding of some symbols. If a constant denotes an object with known features , we can fix its grounding . Training data that consists in a set of data items such as images (or tuples known as training examples) can be specified in Real Logic by constants, e.g. , and by their groundings, e.g. . These can be gathered in a variable imgs. A binary predicate sim that measures the similarity of two objects can be grounded as, e.g., a cosine similarity function of two vectors and . The output layer of the neural network associated with a multi-class single-label predicate can be a softmax function normalizing the output such that it guarantees exclusive classification, i.e. ? Grounding of constants and functions allows the computation of the grounding of their results. If, for example, (transp) is the function that transposes a matrix then .

符号确定的显式定义。通过固定某些符号的确定，可以更严格地融入知识。如果一个常量 表示具有已知特征 的对象，我们可以固定它的确定 。训练数据由一组 数据项组成，例如 图像（或称为训练示例的元组），可以在实数逻辑中通过 常量指定，例如 ，以及通过它们的确定，例如 。这些可以收集在一个变量 imgs 中。一个测量两个对象相似性的二元谓词 sim 可以确定为一个余弦相似度函数，例如两个向量 和 的余弦相似度。与多类单标签谓词 相关的神经网络的输出层可以是一个 softmax 函数，以归一化输出，确保排他性分类，即 ？常量和函数的确定允许计算它们结果的确定。例如，如果 (转置) 是一个转置矩阵的函数，那么 。

Parametric definition of grounding for symbols. Here, the exact grounding of a symbol is not known, but it is known that it can be obtained by finding a set of real-valued parameters, that is, via learning. To emphasize this fact, we adopt the notation where is the set of parameter values that determines the value of . The typical example of parametric grounding for constants is the learning of an embedding. Let be a word embedding with parameters which takes as input a word and returns its embedding in . If the words of a vocabulary are constant symbols, their groundings are defined parametrically w.r.t. as . An example of parametric grounding for a function symbol is to assume that is a linear function such that maps each into , with a matrix of real numbers and a vector of real numbers. In this case, , where . Finally, the grounding of a predicate symbol can be given, for example, by a neural network with parameters . As an example, consider a neural network trained for image classification into classes: cat, dog, horse, etc. takes as input a vector of pixel values and produces as output a vector in such that , where is the probability that input image is of class . In case classes are, alternatively, chosen to be represented by unary predicate symbols such as then , etc.

符号的参数化定义接地。在这里，符号 的确切接地是未知的，但已知它可以通过找到一组实值参数来获得，即通过学习。为了强调这一点，我们采用记法 ，其中 是确定 值的参数值的集合。常数参数化接地的典型例子是学习嵌入。设 为一个单词嵌入，其参数为 ，它将单词作为输入并返回其在 中的嵌入。如果词汇表 中的单词是常量符号，它们的接地 是相对于 参数化定义的，即 。函数符号 的参数化接地的一个例子是假设 是一个线性函数，使得 将每个 映射到 ，其中 是一个实数矩阵， 是一个实数向量。在这种情况下， ，其中 。最后，谓词符号的接地可以通过例如具有参数 的神经网络 给出。例如，考虑一个为图像分类到 类别（如猫、狗、马等）而训练的神经网络 ：猫、狗、马等。 接受一个像素值向量 作为输入，并产生一个输出向量 在 中，使得 ，其中 是输入图像 属于类别 的概率。如果类别选择用一元谓词符号如 来表示，那么 等。

[[9]](#footnote-70)

# 3.1.2. Knowledge through formulas

# 3.1.2. 通过公式获取知识

Factual propositions. Knowledge about the properties of specific objects in the domain is represented, as usual, by logical propositions, as exemplified below: Suppose that it is known that is a number eight, is a number nine, and is a number two. This can be represented by adding the following facts to the knowledge-base: nine , eight , . Supervised learning, that is, learning with the use of training examples which include target values (labelled data), is specified in Real Logic by combining grounding definitions and factual propositions. For example, the fact that an image is a positive example for the class nine and a negative example for the class eight is specified by defining alongside the propositions nine and . Notice how semi-supervision can be specified naturally in Real Logic by adding propositions containing disjunctions, e.g. , which state that is either an eight or a nine (or both). Finally, relational learning can be achieved by relating logically multiple objects (defined as constants or variables or even as more complex sequences of terms) such as e.g.: (if is a nine then is not a nine) or nine eight (if an image is a nine then it is not an eight). The use of more complex knowledge including the use of variables such as img above is the topic of generalized propositions, discussed next.

事实命题。关于特定领域内对象属性的知识通常由逻辑命题表示，如下所示：假设已知 是数字八， 是数字九， 是数字二。这可以通过将以下事实添加到知识库中来表示：九 ，八 ， 。监督学习，即使用包含目标值（标记数据）的训练示例进行学习，在Real Logic中通过结合基础定义和事实命题来指定。例如，图像 是类别九的正例，类别八的反例这一事实，通过定义 以及命题九 和 来指定。注意半监督如何在Real Logic中通过添加包含析取的命题自然指定，例如 ，它表明 是八或九（或两者都是）。最后，通过逻辑关联多个对象（定义为常量、变量或更复杂的术语序列），可以实现关系学习，例如： （如果 是九，那么 不是九）或者九 八 （如果图像是九，那么它不是八）。使用更复杂的知识，包括使用变量如上述的img，是一般命题的话题，将在下一节讨论。

Generalized propositions. General knowledge about all or some of the objects of some domains can be specified in Real Logic by using first-order logic formulas with quantified variables. This general type of knowledge allows one to specify arbitrary constraints on the groundings independently from the specific data available. It allows one to specify, in a concise way, knowledge that holds true for all the objects of a domain. This is especially useful in Machine Learning in the semi-supervised and unsupervised settings, where there is no specific knowledge about a single individual. For example, as part of a task of multi-label classification with constraints on the labels [12], a positive label constraint may express that if an example is labelled with then it should also be labelled with . This can be specified in Real Logic with a universally quantified formula: Another example of soft constraints used in Statistical Relational Learning associates the labels of related examples. For instance, in Markov Logic Networks [55], as part of the well-known Smokers and Friends example, people who are smokers are associated by the friendship relation. In Real Logic, the formula would be used to encode the soft constraint that friends of smokers are normally smokers.

泛化命题。关于某些领域全部或部分对象的一般知识可以通过使用带有量化变量的第一阶逻辑公式在实逻辑中进行指定。这种一般类型的知识允许人们独立于可用的具体数据，对 grounding 应用任意的约束。它允许人们以简洁的方式指定适用于一个领域所有对象的知识。这在机器学习的半监督和无监督设置中尤其有用，在这些设置中，没有关于单个个体的具体知识。例如，在带有标签约束的多标签分类任务中 [12]，一个正标签约束可能表达的是，如果一个示例被标记为 ，那么它也应该被标记为 。这可以通过实逻辑中的一个全称量化公式来指定： 另一个在统计关系学习中使用的软约束示例是将相关示例的标签关联起来。例如，在马尔可夫逻辑网络 [55] 中，作为著名的吸烟者和朋友示例的一部分，吸烟者之间通过友谊关系关联。在实逻辑中，公式 将用于编码这样一个软约束：吸烟者的朋友通常也是吸烟者。

[[10]](#footnote-73)

# 3.1.3. Knowledge through fuzzy semantics

# 3.1.3. 通过模糊语义获取知识

Definition for operators. The grounding of a formula depends on the operators approximating the connectives and quantifiers that appear in . Different operators give different interpretations of the satisfaction associated with the formula. For instance, the operator that approximates universal quantification can be understood as a smooth minimum. It depends on a hyper-parameter (the exponent used in the generalized mean). If then corresponds to the arithmetic mean. As increases, given the same input, the value of the universally quantified formula will decrease as converges to the min operator. To define how strictly the universal quantification should be interpreted in each proposition, one can use different values of for different propositions of the knowledge base. For instance, a formula where is used with a low value for will in fact denote that holds for some , whereas a formula with a higher may denote that holds for most .

操作符定义。公式 的确定依赖于操作符近似公式中的连接词和量词。不同的操作符给出了与公式相关的满意度的不同解释。例如，近似全称量化的操作符 可以理解为平滑最小值。它依赖于一个超参数 （在广义均值中使用的指数）。如果 那么 对应于算术平均数。随着 的增加，对于相同的输入，全称量化公式的值将会减小，因为 收敛于最小值操作符。为了定义在每一个命题中全称量化应该如何严格解释，可以在知识库的不同命题中使用不同的 值。例如，一个公式 在其中 与较小的 值一起使用，实际上表示 对某些 成立，而一个带有较高 值的公式 可能表示 对大多数 成立。

# 3.1.4. Satisfiability

# 3.1.4. 可满足性

In summary, a Real Logic knowledge-base has three components: the first describes knowledge about the grounding of symbols (domains, constants, variables, functions, and predicate symbols); the second is a set of closed logical formulas describing factual propositions and general knowledge; the third lies in the operators and the hyperparameters used to evaluate each formula. The definition that follows formalizes this notion.

总结来说，一个 Real Logic 知识库有三个组成部分：第一部分描述了关于符号确定（域、常量、变量、函数和谓词符号）的知识；第二部分是一组描述事实命题和一般知识的封闭逻辑公式；第三部分在于用于评估每个公式的操作符和超参数。下面的定义形式化了这个概念。

Definition 4 (Theory/Knowledge-base). A theory of Real Logic is a triple , where is a set of closed first-order logic formulas defined on the set of symbols denoting, respectively, domains, variables, constants, function and predicate symbols; is a parametric grounding for all the symbols and all the logical operators; and is the hypothesis space for each set of parameters associated with symbol .

定义4（理论/知识库）。实数逻辑理论是一个三元组 ，其中 是在符号集合 上定义的闭一等逻辑公式集，这些符号分别表示域、变量、常量、函数和谓词符号； 是所有符号 和所有逻辑运算符的参数化接地； 是与每个参数集 相关的假设空间 。

Learning and reasoning in a Real Logic theory are both associated with searching and applying the set of values of parameters from the hypothesis space that maximize the satisfaction of the formulas in . We use the term grounded theory, denoted by , to refer to a Real Logic theory with a specific set of learned parameter values. This idea shares some similarity with the weighted MAX-SAT problem [43], where the weights for formulas in are given by their fuzzy truth-values obtained by choosing the parameter values of the grounding. To define this optimization problem, we aggregate the truth-values of all the formulas in by selecting a formula aggregating operator SatAgg : .

在实数逻辑理论中的学习和推理都与从假设空间 中搜索和应用参数 的值集相关，以最大化公式集 的满意度。我们使用术语“接地理论”，表示为 ，来指代具有特定学习参数值的实数逻辑理论。这个想法与加权MAX-SAT问题 [43] 有一些相似之处，其中 中公式的权重由通过选择接地参数值得到的模糊真值确定。为了定义这个优化问题，我们通过选择公式聚合运算符 SatAgg : 来聚合 中所有公式的真值。

Definition 5. The satisfiability of a theory with respect to the aggregating operator SatAgg is defined as .

定义5。理论 关于聚合运算符 SatAgg 的可满足性定义为 。

# 3.2. Learning

# 3.2. 学习

Given a Real Logic theory , learning is the process of searching for the set of parameter values that maximize the satisfiability of w.r.t. a given aggregator:

给定一个实数逻辑理论 ，学习是搜索参数值集 的过程，以最大化关于给定聚合器的 的可满足性：

Notice that with this general formulation, one can learn the grounding of constants, functions, and predicates. The learning of the grounding of constants corresponds to the learning of em-beddings. The learning of the grounding of functions corresponds to the learning of generative models or a regression task. Finally, the learning of the grounding of predicates corresponds to a classification task in Machine Learning.

注意，使用这种通用公式，可以学习常量、函数和谓词的接地。常量的接地学习对应于学习嵌入。函数的接地学习对应于生成模型的学习或回归任务。最后，谓词的接地学习对应于机器学习中的分类任务。

In some cases, it is useful to impose some regularization (as done customarily in ML) on the set of parameters , thus encoding a preference on the hypothesis space , such as a preference for smaller parameter values. In this case, learning is defined as follows:

在某些情况下，对参数集 施加一些正则化（如在机器学习中通常所做的）是有用的，从而在假设空间 上编码偏好，例如偏好较小的参数值。在这种情况下，学习定义如下：

where is the regularization parameter and is a regularization function, e.g. or regularization, that is, and .

其中 是正则化参数， 是正则化函数，例如 或 正则化，即 和 。

LTN can generalize and extrapolate when querying formulas grounded with unseen data (for example, new individuals from a domain), using knowledge learned with previous groundings (for example, re-using a trained predicate). This is explained in Section 3.3

当查询用未见过的数据（例如，来自域的新个体）接地的公式时，LTN 能够泛化和外推，使用以前接地学习到的知识（例如，重用一个训练好的谓词）。这在第3.3节中有所解释。

# 3.3. Querying

# 3.3. 查询

Given a grounded theory , query answering allows one to check if a certain fact is true (or, more precisely, by how much it is true since in Real Logic truth-values are real numbers in the interval . There are various types of queries that can be asked of a grounded theory.

给定一个接地的理论 ，查询回答允许检查某个事实是否为真（或者更准确地说，它有多真，因为在实逻辑中真值是区间 中的实数）。可以对接地理论提出各种类型的查询。

A first type of query is called truth queries. Any formula in the language of can be a truth query. The answer to a truth query is the truth value of obtained by computing its grounding, i.e. . Notice that, if is a closed formula, the answer is a scalar in denoting the truth-value of according to . if contains free variables , the answer to the query is a tensor of order such that the component indexed by is the truth-value of evaluated in

第一种查询类型被称为真值查询。语言 中的任何公式都可以作为真值查询。对真值查询 的回答是 的真值，通过计算其 grounding，即 获得。注意，如果 是一个封闭公式，那么答案是 中的一个标量，表示根据 的 的真值。如果 包含 个自由变量 ，那么查询的答案是阶数为 的张量，使得由 索引的分量是 在 中求值的真值。

The second type of query is called value queries. Any term in the language of can be a value query. The answer to a value query is a tensor of real numbers obtained by computing the grounding of the term, i.e. . Analogously to truth queries, the answer to a value query is a "tensor of tensors" if contains variables. Using value queries, one can inspect how a constant or a term, more generally, is embedded in the manifold.

第二种查询类型被称为值查询。语言 中的任何项都可以作为值查询。对值查询 的回答是通过计算项的 grounding，即 获得的实数张量。类似于真值查询，如果 包含变量，那么值查询的答案是“张量的张量”。使用值查询，可以检查一个常数或一个项，更一般地说，是如何嵌入到流形中的。

The third type of query is called generalization truth queries. With generalization truth queries, we are interested in knowing the truth-values of formulas when these are applied to a new (unseen) set of objects of a domain, such as a validation or a test set of examples typically used in the evaluation of machine learning systems. A generalization truth query is a pair , where is a formula with a free variable and is a set of unseen examples whose dimensions are compatible with those of the domain of . The answer to the query is for taking each value , in . The result of this query is therefore a vector of truth-values corresponding to the evaluation of on new data .

第三种查询类型被称为泛化真值查询。在泛化真值查询中，我们希望了解当这些公式应用于一个新（未见过的）领域对象集时的真值，例如在机器学习系统的评估中通常使用的验证集或测试集示例。泛化真值查询是一对 ，其中 是带有自由变量 的公式，而 是一组未见过的示例，其维度与 领域的维度兼容。查询的答案 是 对于 取 中的每个值 。因此，这个查询的结果是一个 真值向量，对应于在新数据 上对 的评估。

The fourth and final type of query is generalization value queries. These are analogous to generalization truth queries with the difference that they evaluate a term , and not a formula, on new data . The result, therefore, is a vector of values corresponding to the evaluation of the trained model on a regression task using test data .

第四种也是最后一种查询类型是泛化值查询。这些与泛化真值查询类似，不同之处在于它们在新数据 上评估一个项，而不是公式。因此，结果是 值的向量，对应于训练模型在回归任务上使用测试数据 的评估。

# 3.4. Reasoning

# 3.4. 推理

# 3.4.1. Logical consequence in Real Logic

# 3.4.1. 实际逻辑中的逻辑后果

From a pure logic perspective, reasoning is the task of verifying if a formula is a logical consequence of a set of formulas. This can be achieved semantically using model theory or syntactically via a proof theory . To characterize reasoning in Real Logic, we adapt the notion of logical consequence for fuzzy logic provided in [9]: A formula is a fuzzy logical consequence of a finite set of formulas , in symbols if for every fuzzy interpretation , if all the formulas in are true (i.e. evaluate to 1) in then is true in . In other words, every model of is a model of . A direct application of this definition to Real Logic is not practical since in most practical cases the level of satisfiability of a grounded theory will not be equal to 1 . We therefore define an interval with and assume that a formula is true if its truth-value is in the interval . This leads to the following definition:

从纯逻辑的角度来看，推理的任务是验证一个公式是否是一组公式的逻辑后果。这可以通过使用模型论 在语义上实现，或者通过证明论 在句法上实现。为了表征实数逻辑中的推理，我们采用了文献 [9] 中为模糊逻辑提供的逻辑后果概念：公式 是有限公式集 的模糊逻辑后果，符号表示为 ，如果对于每一种模糊解释 ，如果 中的所有公式在 中都为真（即评估结果为1），那么 在 中也为真。换句话说， 的每一个模型都是 的模型。将这个定义直接应用于实数逻辑是不切实际的，因为在大多数实际情况下，一个具体理论 的满意度水平不等于1。因此，我们定义了一个区间 ，其中 ，并假设如果公式的真值在这个区间内，那么该公式为真。这导致了以下定义：

Definition 6. A closed formula is a logical consequence of a knowledge-base , in symbols , if, for every grounded theory , if then .

定义6。一个闭公式 是知识库 的逻辑后果，符号表示为 ，如果对于每一个具体理论 ，如果 ，那么 。

# 3.4.2. Reasoning by optimization

# 3.4.2. 通过优化进行推理

Logical consequence by direct application of Definition 6 requires querying the truth value of for a potentially infinite set of groundings. Therefore, we consider in practice the following directions:

通过直接应用定义6得到的逻辑后果需要查询 对于可能无限个具体化的真值。因此，在实际情况中，我们考虑以下方向：

Reasoning Option 1 (Querying after learning). This is approximate logical inference by considering only the grounded theories that maximally satisfy . We therefore define that is a brave logical consequence of a Real Logic knowledge-base if for all the such that:

推理选项1（学习后查询）。这是通过仅考虑最大化满足 的具体化理论来进行的近似逻辑推理。因此，我们定义如果 对于所有满足 的 是 Real Logic 知识库 的勇敢逻辑后果：

The objective is to find all that optimally satisfy the knowledge base and to measure if they also satisfy . One can search for such by running multiple optimizations with the objective function of Section 3.2

目标是找到所有 ，它们最优地满足知识库，并衡量它们是否也满足 。可以通过运行多轮优化，并使用第3.2节中的目标函数来搜索这样的 。

This approach is somewhat naive. Even if we run the optimization multiple times with multiple parameter initializations (to, hopefully, reach different optima in the search space), the obtained groundings may not be representative of other optimal or close-to-optimal groundings. In Section 4.8 we give an example that shows the limitations of this approach and motivates the next one.

这种方法有些天真。即使我们多次运行优化，并使用多个参数初始化（希望达到搜索空间中的不同最优解），得到的实例可能并不代表其他最优或接近最优的实例。在第4.8节中，我们给出了一个示例，说明了这种方法的局限性，并推动了下一种方法的出现。

Reasoning Option 2 (Proof by Refutation). Here, we reason by refutation and search for a counterexample to the logical consequence by introducing an alternative search objective. Normally, according to Definition 6, one tries to verify that 11

推理选项2（反驳证明）。在这里，我们通过反驳进行推理，并通过引入替代的搜索目标来寻找逻辑后果的反例。通常，根据定义6，人们试图验证11

Instead, we solve the dual problem:

相反，我们解决对偶问题：

If Eq.(22) is true then a counterexample to Eq.(21) has been found and the logical consequence does not hold. If Eq. 22 is false then no counterexample to Eq. 21) has been found and the logical consequence is assumed to hold true. A search for such parameters (the counterexample) can be performed by minimizing while imposing a constraint that seeks to invalidate results where . We therefore define:如果等式（22）为真，则找到了等式（21）的反例，逻辑后果不成立。如果等式22为假，则没有找到等式（21）的反例，逻辑后果假定成立。通过最小化 并施加一个寻求使 无效的约束，可以搜索这样的参数 （反例）。因此我们定义：

[[11]](#footnote-88)

Given such that:

给定 使得：

* If : Then for all and therefore .
* 如果 ：那么对于所有 ，因此 。
* If and : Then for all with , we have that and therefore .
* 如果 和 ：那么对于所有满足 的 ，我们有 并且因此 。
* If and Then $\left( {\mathcal{K},\mathcal{G}\left( {\cdot \mid \mathbf{\theta }}\right) ,\mathbf{\Theta }}\right) { \mathrel{\text{\vDash \not{} }} }\_{q}\phi$ .
* 如果 和 那么 $\left( {\mathcal{K},\mathcal{G}\left( {\cdot \mid \mathbf{\theta }}\right) ,\mathbf{\Theta }}\right) { \mathrel{\text{\vDash \not{} }} }\_{q}\phi$ 。

Clearly, Equation (23) cannot be used as an objective function for gradient-descent due to null derivatives. Therefore, we propose to approximate the penalty function with the soft constraint:

显然，方程（23）不能作为梯度下降的目标函数，因为其导数为零。因此，我们提议用软约束来近似惩罚函数：

where and are hyper-parameters (see Figure 6). When , the penalty is linear in with a slope of . Setting high, the gradients for will be high in absolute value if the knowledge-base is not satisfied. When , the penalty is a negative exponential that converges to . Setting low but non-zero seeks to ensure that the gradients do not vanish when the penalty should not apply (when the knowledge-base is satisfied). We obtain the following approximate objective function:

其中 和 是超参数（见图6）。当 时，惩罚在 上是线性的，斜率为 。将 设置得很高，如果知识库不满足，那么 的梯度绝对值将会很高。当 时，惩罚是一个负指数，收敛到 。将 设置为低但非零，旨在确保当惩罚不应应用时（即知识库满足时），梯度不会消失。我们得到以下近似目标函数：

Section 4.8 will illustrate the use of reasoning by refutation with an example in comparison with reasoning as querying after learning. Of course, other forms of reasoning are possible, not least that adopted in [6], but a direct comparison is outside the scope of this paper and left as future work.

第4.8节将通过一个示例，说明使用反驳推理与在学习后进行查询推理的比较。当然，还有其他形式的推理是可能的，例如文献[6]中采用的方法，但直接比较超出了本文的范围，留待以后的工作。

# 4. The Reach of Logic Tensor Networks

# 4. 逻辑张量网络的适用范围

The objective of this section is to show how the language of Real Logic can be used to specify a number of tasks that involve learning from data and reasoning. Examples of such tasks are classification, regression, clustering, and link prediction. The solution of a problem specified in Real Logic is obtained by interpreting such a specification in Logic Tensor Networks. The LTN library implements Real Logic in Tensorflow 2 [1] and is available from GitHub Every logical operator is grounded using Tensorflow primitives such that LTN implements directly a Tensorflow graph. Due to Tensorflow built-in optimization, LTN is relatively efficient while providing the expressive power of first-order logic. Details on the implementation of the examples described in this section are reported in Appendix A. The implementation of the examples presented here is also available from the LTN repository on GitHub. Except when stated otherwise, the results reported are the average result over 10 runs using a confidence interval. Every example uses a stable real product configuration to approximate the Real Logic operators and the Adam optimizer [35] with a learning rate of 0.001 . Table A.3 in the Appendix gives an overview of the network architectures used to obtain the results reported in this section.

本节的目的是展示如何使用实数逻辑的语言来指定涉及从数据学习和推理的多种任务。这类任务的例子包括分类、回归、聚类和链接预测。在实数逻辑中指定的问题的解决方案是通过在逻辑张量网络中解释此类规范来获得的。LTN库在Tensorflow 2 [1] 中实现了实数逻辑，并且可以从GitHub 获取。每个逻辑运算符都使用Tensorflow原语接地，使得LTN直接实现了一个Tensorflow图。由于Tensorflow内置的优化，LTN在提供一阶逻辑表达能力的同时相对高效。本节中描述的示例的实现细节在附录A中报告。此处给出的示例的实现还可在GitHub上的LTN仓库中获得。除非另有说明，报告的结果是使用 置信区间进行10次运行的平均结果。每个示例都使用稳定的实数产品配置来近似实数逻辑运算符，并使用学习率为0.001的Adam优化器 [35]。附录中的表A.3概述了用于获得本节报告结果的网络架构。

[[12]](#footnote-91)

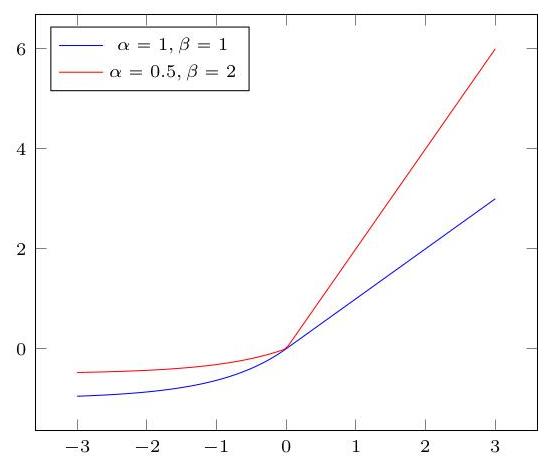


Figure 6: where and are hyper-parameters. The function with low and high is a soft constraint for penalty suitable for learning.

图6： 其中 和 是超参数。函数 在 低和 高的情况下，是对惩罚 的软约束，适合于学习。

# 4.1. Binary Classification

# 4.1. 二元分类

The simplest machine learning task is binary classification. Suppose that one wants to learn a binary classifier for a set of points in . Suppose that a set of positive and negative training examples is given. LTN uses the following language and grounding:

最简单的机器学习任务是二分类。假设某人希望为 中的一组点学习一个二分类器 。假设给定了一组正例和负例训练样本。LTN 使用以下语言和地面表示：

# Domains:

# 领域：

points (denoting the examples).

点（表示示例）。

# Variables:

# 变量：

for the positive examples.

表示正例。

for the negative examples.

表示负例。

for all examples.

表示所有示例。

points.

点。

Predicates:

谓词：

for the trainable classifier.

表示可训练分类器。

points. Axioms:

点。公理：

# Grounding:

# 地面：

(points) .

（点） 。

.

。

, where MLP is a Multilayer Perceptron with a single output neuron, whose parameters are to be learned

，其中 MLP 是具有单个输出神经元的多层感知器，其参数 需要学习 。

# Learning:

# 学习：

Let us define the data set of all examples. The objective function with is given by . In practice, the optimizer uses the following loss function:

让我们定义 为所有示例的数据集。带有 的目标函数由 给出。 在实际中，优化器使用以下损失函数：

where is a mini-batch sampled from The objective and loss functions depend on the following hyper-parameters:

其中 是从 中抽取的迷你批次。目标函数和损失函数依赖于以下超参数：

* the choice of fuzzy logic operator semantics used to approximate each connective and quantifier,
* 选择用于近似每个连接词和量词的模糊逻辑操作符语义，
* the choice of hyper-parameters underlying the operators, such as the value of the exponent in any generalized mean,
* 选择操作符下的超参数，例如任何广义均值中指数 的值。
* the choice of formula aggregator function.
* 公式聚合函数的选择。

Using the stable product configuration to approximate connectives and quantifiers, and for every occurrence of , and using for the formula aggregator also with , yields the following satisfaction equation:

使用稳定的产物配置来近似连接词和量词，并且对于每个 的出现，以及使用 作为公式聚合器并结合 ，得到以下满意度方程：

[[13]](#footnote-105)

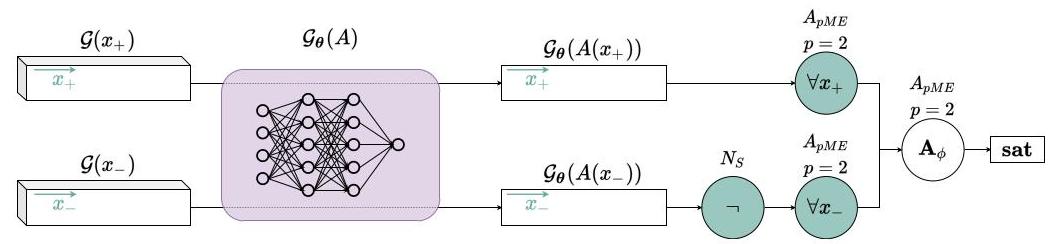


Figure 7: Symbolic Tensor Computational Graph for the Binary Classification Example. In the figure, and are inputs to the network and the dotted lines indicate the propagation of activation from each input through the network, which produces two outputs.

图7：二分类示例的符号张量计算图。在图中， 和 是网络 的输入，虚线表示每个输入通过网络激活的传播，这产生了两个输出。

The computational graph of Figure 7 shows Sat ) as used with the above loss function.

图7的计算图显示了与上述损失函数一起使用的 Sat 。

We are therefore interested in learning the parameters of the MLP used to model the binary classifier. We sample 100 data points uniformly from to populate the data set of positive and negative examples. The data set was split into 50 data points for training and 50 points for testing. The training was carried out for a fixed number of 1000 epochs using backpropagation with the Adam optimizer [35] with a batch size of 64 examples. Figure 8 shows the classification accuracy and satisfaction level of the LTN on both training and test sets averaged over 10 runs using a 95% confidence interval. The accuracy shown is the ratio of examples correctly classified, with an example deemed as being positive if the classifier outputs a value higher than 0.5 .

因此，我们感兴趣于学习用于建模二分类器的多层感知器（MLP）的参数 。我们从 中均匀地抽取100个数据点来构建包含正例和负例的数据集。数据集被分为50个数据点用于训练，另外50个数据点用于测试。训练过程使用带有Adam优化器 [35] 的反向传播算法，在固定数量的1000个训练周期中进行，批量大小为64个示例。图8显示了在训练集和测试集上，经过10次运行的平均分类准确性和LTN的满意度水平，并使用95%的置信区间。所显示的准确度是正确分类的示例比例，如果分类器输出的值高于0.5，则将该示例视为正例。

Notice that a model can reach an accuracy of while satisfaction of the knowledge base is yet not maximized. For example, if the threshold for an example to be deemed as positive is 0.7 , all examples may be classified correctly with a confidence score of 0.7 . In that case, while the accuracy is already maximized, the satisfaction of would still be 0.7, and can still improve until the confidence for every sample reaches 1.0.

注意，一个模型可以达到 的准确度，而知识库的满意度尚未最大化。例如，如果将一个示例视为正例的阈值设为0.7，所有示例都可以以0.7的置信度被正确分类。在这种情况下，虽然准确度已经最大化，但 的满意度仍然是0.7，并且可以一直改进，直到每个样本的置信度达到1.0。

This first example, although straightforward, illustrates step-by-step the process of using LTN in a simple setting. Notice that, according to the nomenclature of Section 3.3 measuring accuracy amounts to querying the truth query (respectively, the generalization truth query) for all the examples of the training set (respectively, test set) and comparing the results with the classification threshold. In Figure 9, we show the results of such queries after optimization. Next, we show how the LTN language can be used to solve progressively more complex problems by combining learning and reasoning.

这个第一个例子虽然简单，但逐步说明了在简单设置中使用LTN的过程。注意，根据3.3节的命名规则，测量准确度相当于对训练集（分别地，测试集）中所有示例进行真实性查询（分别地，泛化真实性查询） ，并将结果与分类阈值进行比较。在图9中，我们展示了优化后的查询 结果。接下来，我们将展示如何通过结合学习和推理，使用LTN语言逐步解决更复杂的问题。

# 4.2. Multi-Class Single-Label Classification

# 4.2. 多类单标签分类

The natural extension of binary classification is a multi-class classification task. We first approach multi-class single-label classification, which assumes that each example is assigned to one and only one label.

二元分类的自然扩展是多类分类任务。我们首先研究多类单标签分类，该分类假设每个示例仅分配给一个标签。

For illustration purposes, we use the Iris flower data set [20], which consists of classification into three mutually exclusive classes; call these , and . While one could train three unary predicates and , it turns out to be more effective if this problem is modeled by a single binary predicate , where is a variable denoting a multi-class label, in this case,

出于说明目的，我们使用Iris花数据集 [20]，该数据集包括分为三个互斥类别的分类；称为 ，和 。虽然可以训练三个一元谓词 和 ，但将这个问题建模为一个单一的二元谓词 往往更有效，其中 是表示多类标签的变量，在这种情况下，

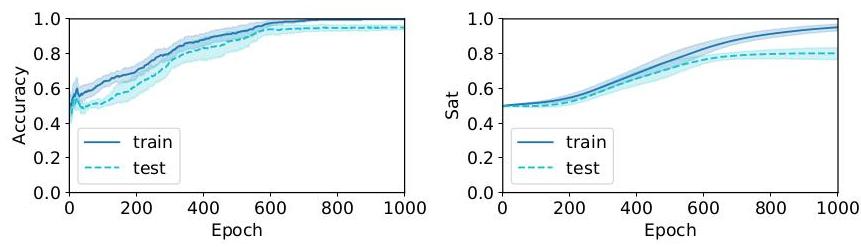


Figure 8: Binary Classification task (training and test set performance): Average accuracy (left) and satisfiability (right). Due to the random initializations, accuracy and satisfiability start on average at 0.5 with performance increasing rapidly after a few epochs.

图8：二元分类任务（训练和测试集性能）：平均准确度（左）和满意度（右）。由于随机初始化，准确度和满意度平均起始值为0.5，在几个周期后性能迅速提高。

classes or . This syntax allows one to write statements quantifying over the classes, e.g. . Since the classes are mutually exclusive, the output layer of the MLP representing will be a softmax layer, instead of a sigmoid function, to ensure the exclusivity constraint on satisfiability scores The problem can be specified as follows:

类别 或 。这种语法允许人们编写量化类别的语句，例如 。由于类别是互斥的，表示 的MLP输出层将是一个softmax层，而不是sigmoid函数，以确保满意度得分 的排他性约束问题可以指定如下：

# Domains:

# 领域：

items, denoting the examples from the Iris flower data set.

项目，指代来自Iris花数据集的示例。

labels, denoting the class labels.

标签，指代类别标签。

# Variables:

# 变量：

for the positive examples of classes .

用于类别 的正例。

for all examples.

用于所有示例。

items.

项目。

# Constants:

# 常量：

, the labels of classes (Iris setosa), (Iris virginica), (Iris versicolor), respectively.

，分别是类别 (Iris setosa)， (Iris virginica)， (Iris versicolor) 的标签。

labels.

标签。

Predicates:

谓词：

denoting the fact that item is classified as .

表示项目 被分类为 的事实。

items, labels.

项目，标签。

Axioms:

公理：

Notice that rules about exclusiveness such as are not included since such constraints are already imposed by the grounding of below, more specifically the softmax function.

注意，关于排他性的规则，如 并未包含，因为此类约束已经由下文的 的基础，更具体地说，是 softmax 函数强加的。

[[14]](#footnote-120)

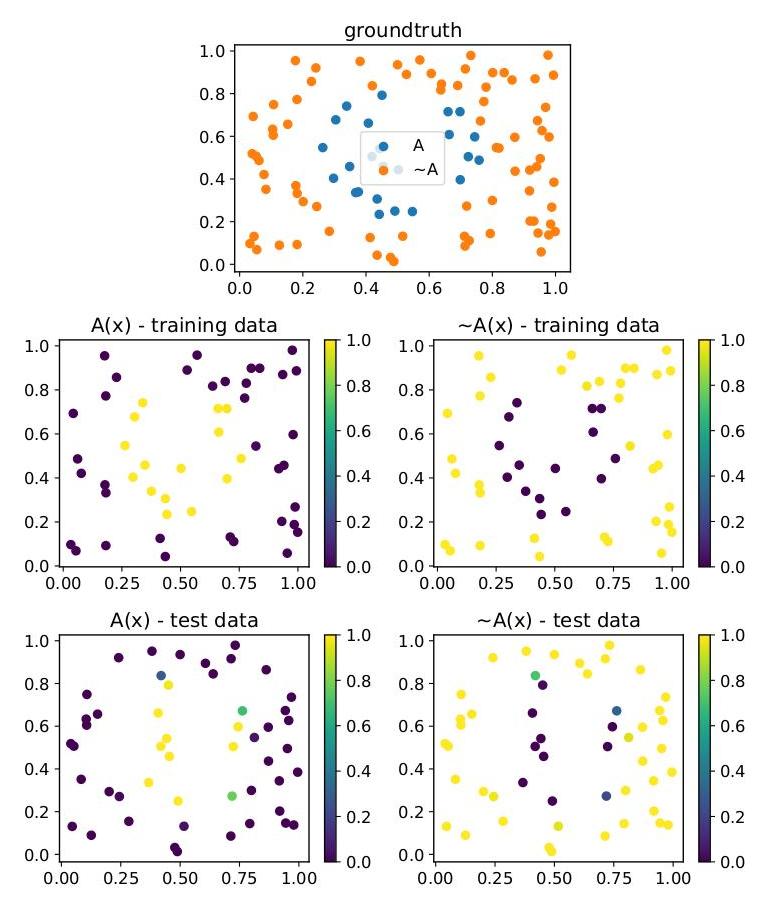


Figure 9: Binary Classification task (querying the trained predicate ): It is interesting to see how could be appropriately named as denoting the inside of the central region shown in the figure, and therefore represents the outside of the region.

图 9：二分类任务（查询训练好的谓词 ）：有趣的是看到 如何恰当地命名为表示图中所示中心区域内部，因此 表示区域外部。

# Grounding:

# 地基：

(items) , items are described by 4 features: the length and the width of the sepals and petals, in centimeters.

（项目） ，项目由四个特征描述：花瓣和花萼的长度和宽度，单位为厘米。

(labels) , we use a one-hot encoding to represent classes.

（标签） ，我们使用独热编码来表示类别。

, that is, is a sequence of examples of class .

，即 是类别 的 个例子的序列。

is a sequence of examples of class .

是类别 的 个例子的序列。

is a sequence of examples of class .

是类别 的 个例子的序列。

is a sequence of all the examples.

是所有例子的序列。

.

。

, where the MLP has three output neurons corresponding to as many classes, and denotes the dot product as a way of selecting an output for ; multiplying the MLP’s output by the one-hot vector gives the truth degree corresponding to the class denoted by .

，其中 MLP 有三个输出神经元，对应于同样数量的类别， 表示点积作为一种为 选择输出的方式；将 MLP 的输出乘以独热向量 给出了与 所表示类别对应的真值度。

# Learning:

# 学习：

The logical operators and connectives are approximated using the stable product configuration with for . For the formula aggregator, is used also with .

逻辑运算符和连接词使用稳定的乘积配置与 来近似 。对于公式聚合器， 也与 一起使用。

The computational graph of Figure 10 illustrates how is obtained. If denotes batches sampled from the data set of all examples, the loss function (to minimize) is:

图10的计算图说明了如何得到 。如果 表示从所有示例的数据集中抽取的批次，那么损失函数（需要最小化）是：

Figure 11 shows the result of training with the Adam optimizer with batches of 64 examples. Accuracy measures the ratio of examples correctly classified, with example labeled as Classification accuracy reaches an average value near 1.0 for both the training and test data after some 100 epochs. Satisfaction levels of the Iris flower predictions continue to increase for the rest of the training (500 epochs) to more than 0.8 .

图11展示了使用Adam优化器并以64个示例为批次进行训练的结果。准确度衡量正确分类的示例比例，示例 被标记为 分类准确度在经过大约100个训练周期后，训练集和测试集的平均值接近1.0。在训练的剩余部分（500个周期）中，对 Iris 花预测的满意度持续增加至超过0.8。

It is worth contrasting the choice of using a binary predicate in this example with the option of using multiple unary predicates , one for each class. Notice how each predicate is normally associated with an output neuron. In the case of the unary predicates, the networks would be disjoint (or modular), whereas weight-sharing takes place with the use of the binary predicate. Since is instantiated into , in practice becomes , which is implemented via three output neurons to which a softmax function applies.

值得对比的是，在本例中使用二元谓词 的选择与使用多个一元谓词 的选项，每个类别使用一个。注意每个谓词通常与一个输出神经元相关联。在 unary predicates 的情况下，网络将是分离的（或模块化的），而使用二元谓词时会发生权重共享。由于 实例化为 ，实际上 变为 ，这是通过三个输出神经元实现的，这些神经元应用了 softmax 函数。

# 4.3. Multi-Class Multi-Label Classification

# 4.3. 多类多标签分类

We now turn to multi-label classification, whereby multiple labels can be assigned to each example. As a first example of the reach of LTNs, we shall see how the previous example can be extended naturally using LTN to account for multiple labels, not always a trivial extension for most ML algorithms. The standard approach to the multi-label problem is to provide explicit negative examples for each class. By contrast, LTN can use background knowledge to relate classes directly to each other, thus becoming a powerful tool in the case of the multi-label problem when typically the labeled data is scarce. We explore the Leptograpsus crabs data set [10] consisting of 200 examples of 5 morphological measurements of 50 crabs. The task is to classify the crabs according to their color and sex. There are four labels: blue, orange, male, and female. The color labels are mutually exclusive, and so are the labels for sex. LTN will be used to specify such information logically.

现在我们转向多标签分类问题，其中每个示例可以分配多个标签。作为LTN（逻辑 tensor 网络）应用范围的一个示例，我们将看到如何使用LTN自然地扩展之前的示例，以处理多个标签，这对于大多数机器学习算法来说并不是一个简单的扩展。处理多标签问题的标准方法是针对每个类别提供显式的负例。相比之下，LTN可以利用背景知识直接将类别相互关联，因此在标签数据稀缺的多标签问题情况下，成为一种强大的工具。我们探讨了Leptograpsus crabs数据集[10]，该数据集包含50只螃蟹的5种形态测量，共200个示例。任务是按照螃蟹的颜色和性别进行分类。有四个标签：蓝色、橙色、雄性和雌性。颜色标签是相互排斥的，性别标签也是如此。LTN将用于逻辑上指定此类信息。

[[15]](#footnote-130)

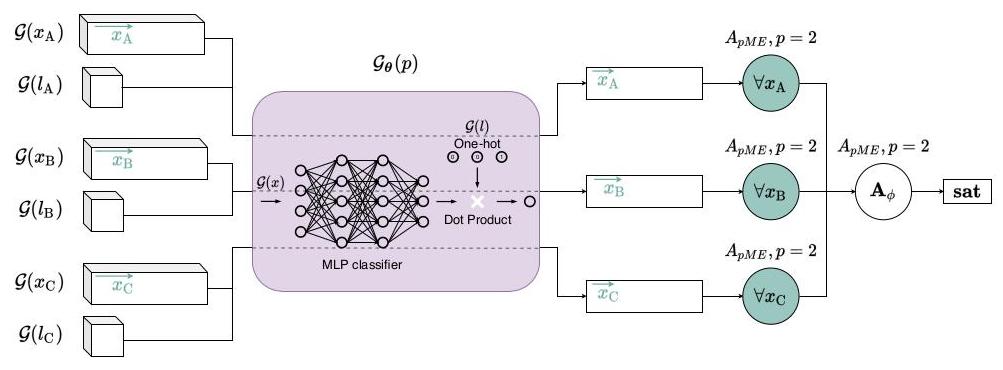


Figure 10: Symbolic Tensor Computational Graph for the Multi-Class Single-Label Problem. As before, the dotted lines in the figure indicate the propagation of activation from each input through the network, in this case producing three outputs.

图10：多类单标签问题的符号张量计算图。与之前一样，图中的虚线表示每个输入通过网络的激活传播，在此情况下产生三个输出。

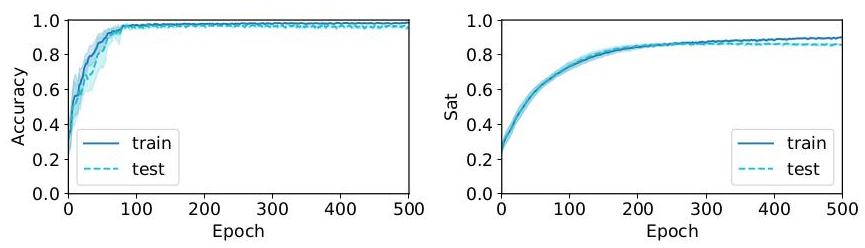


Figure 11: Multi-Class Single-Label Classification: Classification accuracy (left) and satisfaction level (right).

图11：多类单标签分类：分类准确度（左）和满意度（右）。

# Domains:

# 领域：

items denoting the examples from the crabs dataset.

表示来自螃蟹数据集的示例的项目。

labels denoting the class labels.

表示类标签的标签。

# Variables:

# 变量：

for the positive examples of each class.

用于每个类的正例。

, used to denote all the examples.

，用于表示所有示例。

items.

项目。

# Constants:

# 常量：

(the labels for each class).

(the labels for each class).

labels.

labels.

Predicates:

Predicates:

, denotes the fact that item is labelled as .

, denotes the fact that item is labelled as .

items, labels.

items, labels.

# Axioms:

# Axioms:

Notice how logical rules 34 and 35 above represent the mutual exclusion of the labels on

Notice how logical rules 34 and 35 above represent the mutual exclusion of the labels on

colour and sex, respectively. As a result, negative examples are not used explicitly in this specification.

colour and sex, respectively. As a result, negative examples are not used explicitly in this specification.

# Grounding:

# Grounding:

(items) ; the examples from the data set are described using 5 features.

(items) ; the examples from the data set are described using 5 features.

(labels) ; one-hot vectors are used to represent class labels

(labels) ; one-hot vectors are used to represent class labels

. These sequences are not mutually-exclusive, one example can for instance be in both and . . , with the MLP having four output neurons corresponding to as many classes. As before, denotes the dot product which selects a single output. By contrast with the previous example, notice the use of a sigmoid function instead of a softmax function.

. These sequences are not mutually-exclusive, one example can for instance be in both and . . , with the MLP having four output neurons corresponding to as many classes. As before, denotes the dot product which selects a single output. By contrast with the previous example, notice the use of a sigmoid function instead of a softmax function.

[[16]](#footnote-147)

# Learning:

# Learning:

As before, the fuzzy logic operators and connectives are approximated using the stable product configuration with for , and for the formula aggregator, is also used with .

As before, the fuzzy logic operators and connectives are approximated using the stable product configuration with for , and for the formula aggregator, is also used with .

Figure 12 shows the result of the Adam optimizer using backpropagation trained with batches of 64 examples. This time, the accuracy is defined as 1 -HL, where HL is the average Hamming loss, i.e. the fraction of labels predicted incorrectly, with a classification threshold of 0.5 (given an example , if the model outputs a value greater than 0.5 for class then is deemed as belonging to class ). The rightmost graph in Figure 12 illustrates how LTN learns the constraint that a crab cannot have both blue and orange color, which is discussed in more detail in what follows.

图 12 显示了使用反向传播和 64 个样例批训练的 Adam 优化器的结果。这次，准确度定义为 1 -HL，其中 HL 是平均汉明损失，即预测错误的标签比例，分类阈值为 0.5（对于一个示例 ，如果模型为类别 输出一个大于 0.5 的值，那么 被认为是属于类别 ）。图 12 最右边的图表说明了 LTN 如何学习螃蟹不能同时具有蓝色和橙色的约束，这一点将在接下来的内容中详细讨论。

# Querying:

# 查询：

To illustrate the learning of constraints by LTN, we have queried three formulas that were not explicitly part of the knowledge-base, over time during learning:

为了说明 LTN 学习约束的过程，我们在学习过程中查询了三个不是知识库显式部分公式：

For querying, we use when approximating the universal quantifiers with . A higher denotes a stricter universal quantification with a stronger focus on outliers (see Section 2.4). We should expect to hold true (every blue crab cannot be orange and vice-versa , and we should expect (every blue crab is also orange) and (every blue crab is male) to be false. The results are reported in the rightmost plot of Figure 12 Prior to training, the truth-values of to are non-informative. During training one can see, with the maximization of the satisfaction of the knowledge-base, a trend towards the satisfaction of , and an opposite trend of and towards false.

在查询时，我们使用 来近似全称量词 。一个较高的 表示更严格的全称量化，更关注异常值（参见第 2.4 节）。 我们应该期望 成立（每个蓝色的螃蟹不能是橙色，反之亦然 ，并且我们期望 （每个蓝色的螃蟹也是橙色）和 （每个蓝色的螃蟹是雄性）为假。结果报告在图 12 最右边的图表中。在训练之前， 到 的真值是非信息性的。在训练过程中，随着知识库满意度的最大化，可以看到 的满意度趋势，以及 和 朝着假的趋势相反。

# 4.4. Semi-Supervised Pattern recognition

# 4.4. 半监督模式识别

Let us now explore two, more elaborate, classification tasks, which showcase the benefit of using logical reasoning alongside machine learning. With these two examples, we also aim to provide a more direct comparison with a related neurosymbolic system DeepProbLog [41]. The benchmark examples below were introduced in the DeepProbLog paper [41].

现在我们来探讨两个更复杂的分类任务，它们展示了在使用机器学习的同时运用逻辑推理的好处。通过这两个例子，我们还旨在提供一个与相关神经符号系统DeepProbLog [41]的更直接比较。以下基准示例在DeepProbLog论文[41]中有所介绍。

[[17]](#footnote-154)

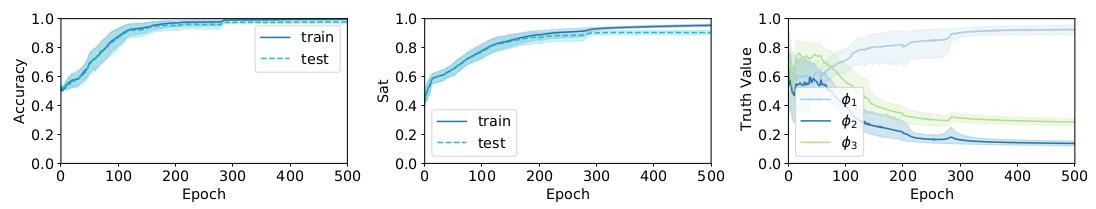


Figure 12: Multi-Class Multi-Label Classification: Classification Accuracy (left), Satisfiability level (middle), and Querying of Constraints (right).

图12：多类多标签分类：分类准确度（左），满意度水平（中），和约束查询（右）。

Single Digits Addition: Consider the predicate addition , where and are images of digits (the MNIST data set will be used), and N is a natural number corresponding to the sum of these digits. This predicate should return an estimate of the validity of the addition. For instance, addition is a valid addition; addition is not.

单位数加法：考虑谓词 addition ，其中 和 是数字的图像（将使用MNIST数据集），而N是这些数字之和的自然数。这个谓词应该返回对加法有效性的估计。例如，addition 是有效的加法；addition 不是。

Multi Digits Addition: The experiment is extended to numbers with more than one digit. Consider the predicate addition and are lists of images of digits, representing two multi-digit numbers; is a natural number corresponding to the sum of the two multi-digit numbers. For instance, addition is a valid addition; addition is not.

多位数加法：实验扩展到超过一个数字的数字。考虑谓词 addition ，其中 是数字图像的列表，代表两个多位数； 是这两个多位数之和的自然数。例如，addition 是有效的加法；addition 不是。

A natural neurosymbolic approach is to seek to learn a single-digit classifier and benefit from knowledge readily available about the properties of addition in this case. For instance, suppose that a predicate gives the likelihood of an image being of digit . A definition for addition in LTN is:

自然神经符号方法之一是尝试学习单个数字分类器并从关于加法属性的知识中受益，例如，假设谓词 给出了图像 是数字 的可能性。在 LTN 中对加法 的定义如下：

In [41], the above task is made more complicated by not providing labels for the single-digit images during training. Instead, training takes place on pairs of images with labels made available for the result only, that is, the sum of the individual labels. The single-digit classifier is not explicitly trained by itself; its output is a piece of latent information that is used by the logic. However, this does not pose a problem for end-to-end neurosymbolic systems such as LTN or DeepProbLog for which the gradients can propagate through the logical structures.

在文献 [41] 中，上述任务通过在训练期间不提供单个数字图像的标签而变得更加复杂。相反，训练是在图像对上进行的，只有结果的标签是可用的，即单个标签的和。单个数字分类器并不是单独显式训练的；其输出是逻辑使用的潜在信息的一部分。然而，这对于像 LTN 或 DeepProbLog 这样的端到端神经符号系统来说并不是问题，因为梯度可以通过逻辑结构传播。

We start by illustrating a LTN theory that can be used to learn the predicate digit. The specification of the theory below is for the single digit addition example, although it can be extended easily to the multiple digits case.

我们首先说明一个 LTN 理论，该理论可以用来学习数字谓词。下面理论的规范是针对单个数字加法示例的，尽管它可以轻松扩展到多位数的情况。

# Domains:

# 领域：

images, denoting the MNIST digit images,

图像，表示 MNIST 数字图像，

results, denoting the integers that label the results of the additions,

结果，表示加法结果标签的整数，

digits, denoting the digits from 0 to 9 .

数字，表示从 0 到 9 的数字。

# Variables:

# 变量：

, ranging over the MNIST images in the data,

，遍历数据集中的 MNIST 图像，

for the labels, i.e. the result of each addition,

表示标签，即每次加法的结果，

ranging over digits.

遍历数字。

images,

图像，

results,

结果，

digits.

数字。

# Predicates:

# 谓词：

for the single digit classifier, where is a term denoting a digit constant or a digit variable. The classifier should return the probability of an image being of digit . images, digits. Axioms:

用于单个数字分类器，其中 是表示数字常量或数字变量的项。分类器应返回图像 是数字 的概率。 图像，数字。公理：

Single Digit Addition:

单位数加法：

Multiple Digit Addition:

多位数加法：

Notice the use of Diag: when grounding with three sequences of values, the -th examples of each variable are matching. That is, is a tuple from our dataset of valid additions. Using the diagonal quantification, LTN aggregates pairs of images and their corresponding result, rather than any combination of images and results.

注意 Diag 的使用：当用三个值序列对 进行定位时，每个变量的第 个示例是匹配的。也就是说， 是我们数据集中有效加法的一个元组。使用对角量化，LTN 聚合图像对及其相应的结果，而不是图像和结果的任意组合。

Notice also the guarded quantification: by quantifying only on the latent "digit labels" (i.e. that can add up to the result label ( , given in the dataset), we incorporate symbolic information into the system. For example, in (39), if , the only valid tuples are . Gradients will only backpropagate to these values.

还要注意保护性量化：通过仅在潜在“数字标签”（即 可以加到结果标签 ，数据集中给出）上进行量化，我们将符号信息纳入系统。例如，在 (39) 中，如果 ，唯一有效的元组 是 。梯度只会回传到这些值。

# Grounding:

# 定位：

(images) . The MNIST data set has images of 28 by 28 pixels. The images are grayscale and have just one channel. The RGB pixel values from 0 to 255 of the MNIST data set are converted to the range .

（图像） 。MNIST 数据集包含 28x28 像素的图像。图像是灰度的，只有一个通道。MNIST 数据集的 RGB 像素值从 0 到 255 转换为范围 。

(results) .

（结果） 。

.

。

24

24

onehot , where is a Convolutional Neural Network with 10 output neurons for each class. Notice that, in contrast with the previous examples, is an integer label; onehot converts it into a one-hot label.

onehot ，其中 是一个具有每个类别 10 个输出神经元的卷积神经网络。注意，与之前的示例不同， 是一个整数标签；onehot 将其转换为独热标签。

[[18]](#footnote-166)

# Learning:

# 学习：

The computational graph of Figure 13 shows the objective function for the satisfiability of the knowledge base. A stable product configuration is used with hyper-parameter of the operator for universal quantification . Let denote the exponent hyper-parameter used in the generalized mean for existential quantification (3). Three scenarios are investigated and compared in the Multiple Digit experiment (Figure 15):

图13的计算图显示了知识库满意度的目标函数。使用具有超参数 的操作符 的稳定产品配置来实现全称量化 。令 表示在广义均值 中用于存在量化的指数超参数（3）。在多个数字实验（图15）中研究了三种场景并进行了比较：

1. throughout the entire experiment,

1. 在整个实验过程中使用 ，

2. throughout the entire experiment, or

2. 在整个实验过程中使用 ，或者

3. follows a schedule, changing from to gradually with the number of training epochs.

3. 遵循一个计划，随着训练时期的数量逐渐从 变化到 。

In the Single Digit experiment, only the last scenario above (schedule) is investigated (Figure 14).

在单数字实验中，只研究了上述最后一种场景（计划）（图14）。

We train to maximize satisfiability by using batches of 32 examples of image pairs, labeled by the result of their addition. As done in [41], the experimental results vary the number of examples in the training set to emphasize the generalization abilities of a neurosymbolic approach. Accuracy is measured by predicting the digit values using the predicate digit and reporting the ratio of examples for which the addition is correct. A comparison is made with the same baseline method used in [41]: given a pair of MNIST images, a non-pre-trained CNN outputs embeddings for each image (Siamese neural network). The embeddings are provided as input to dense layers that classify the addition into one of the 19 (respectively, 199) possible results of the Single Digit Addition (respectively, Multiple Digit Addition) experiments. The baseline is trained using a cross-entropy loss between the labels and the predictions. As expected, such a standard deep learning approach struggles with the task without the provision of symbolic meaning about intermediate parts of the problem.

我们通过使用32个图像对的批次来训练，这些图像对被它们的加法结果标记。与 [41] 中的做法一样，实验结果通过改变训练集中的示例数量来强调神经符号方法的泛化能力。准确性是通过预测数字值并报告加法正确的示例比例来衡量的。与 [41] 中使用的相同基线方法进行了比较：对于给定的MNIST图像对，未经预训练的CNN为每个图像输出嵌入（Siamese神经网络）。将这些嵌入作为输入提供给密集层，以将加法分类为单数字加法（分别，多数字加法）实验中的19个（分别，199个）可能结果之一。基线是通过标签和预测之间的交叉熵损失来训练的。如预期的那样，这种标准的深度学习方法在没有提供关于问题中间部分的符号意义的情况下，难以应对这项任务。

Experimentally, we find that the optimizer for the neurosymbolic system gets stuck in a local optimum at the initialization in about 1 out of 5 runs. We, therefore, present the results on an average of the 10 best outcomes out of 15 runs of each algorithm (that is, for the baseline as well). The examples of digit pairs selected from the full MNIST data set are randomized at each run.

实验上，我们发现神经符号系统的优化器在大约五分之一的初始化中会陷入局部最优。因此，我们展示了每个算法15次运行中前10个最佳结果的平均值（基线也是如此）。每次运行时，从完整的MNIST数据集中随机选择的数字对都会被随机化。

Figure 15 shows that the use of from the start produces poor results. A higher value for in weighs up the instances with a higher truth-value (see also Appendix C for a discussion). Starting already with a high value for , the classes with a higher initial truth-value for a given example will have higher gradients and be prioritized for training, which does not make practical sense when randomly initializing the predicates. Increasing by following a schedule is the most promising approach. In this particular example, is also shown to be adequate purely from a learning perspective. However, implements a simple average which does not account for the meaning of well; the resulting satisfaction value is not meaningful within a reasoning perspective.

图15显示，从开始就使用 会产生较差的结果。 在 中的较高值会加重具有较高真实值的事例（也参见附录C的讨论）。从较高的 值开始，对于给定示例具有较高初始真实值的类别将有较高的梯度并被优先用于训练，这在随机初始化谓词时不具有实际意义。按照计划增加 是最有希望的方法。在这个特定示例中，还显示 仅仅从学习角度来看也是足够的。然而， 实现了一个简单的平均值，并没有很好地解释 的含义；得到的满意度值在推理视角下没有意义。

Table 1 shows that the training and test times of LTN are of the same order of magnitude as those of the CNN baselines. Table 2 shows that LTN reaches similar accuracy as that reported by DeepProbLog.

表1显示，LTN的训练和测试时间与CNN基线的时间属于同一数量级。表2显示，LTN达到了与DeepProbLog报道的相似精度。

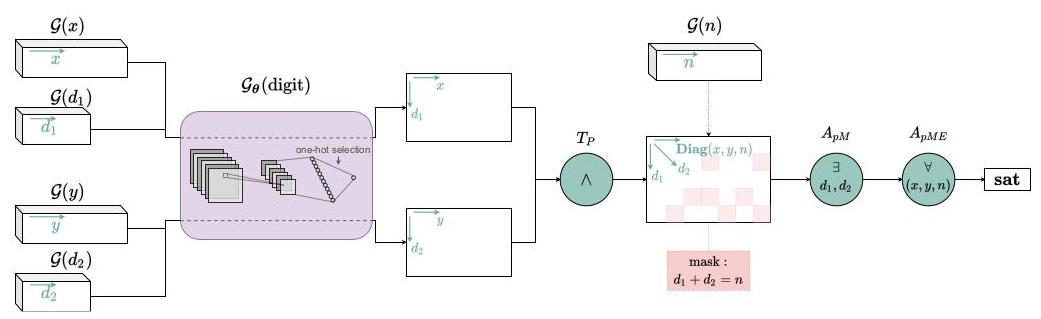


Figure 13: Symbolic Tensor Computational Graph for the Single Digit Addition task. Notice that the figure does not depict accurate dimensions for the tensors; and are in fact 4D tensors of dimensions . Computing results with the variables or corresponds to the addition of a further axes of dimension 10 .

图13：单个数字加法任务的符号张量计算图。注意，该图没有描绘出张量的准确维度； 和 实际上是尺寸为 的4D张量。使用变量 或 计算结果对应于增加一个维度为10的轴。

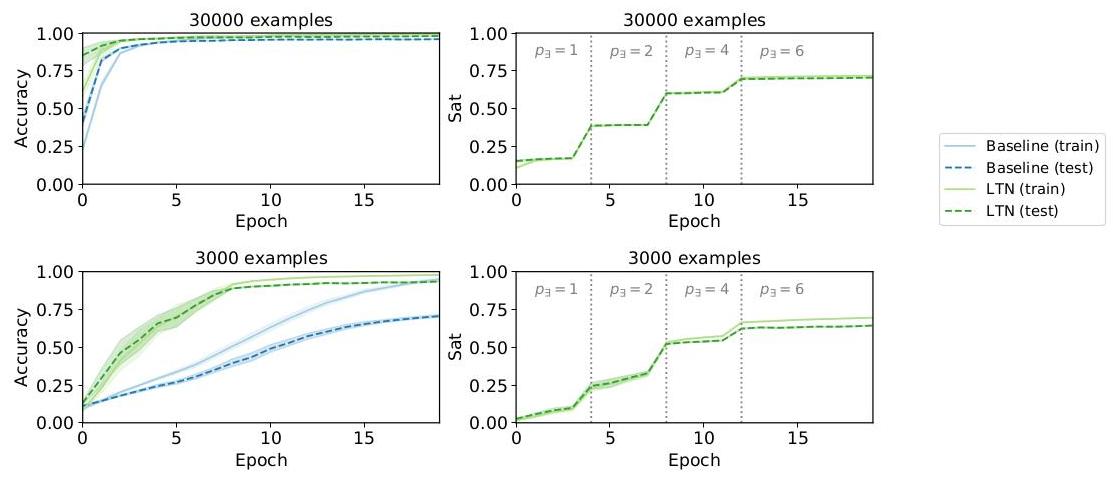


Figure 14: Single Digit Addition Task: Accuracy and satisfiability results (top) and results in the presence of fewer examples (bottom) in comparison with standard Deep Learning using a CNN (blue lines).

图14：单个数字加法任务：准确性和满意度结果（顶部）以及在示例较少的情况下（底部）与使用CNN的标准深度学习（蓝色线条）的比较结果。

| Model | (Single Digits) | | (Multi Digits) | |
| --- | --- | --- | --- | --- |
|  | Train | Test | Train | Test |
| baseline |  |  |  |  |
| LTN |  |  |  |  |

Table 1: The computation time of training and test steps on the single and multiple digit addition tasks, measured on a computer with a single Nvidia Tesla V100 GPU and averaged over 1000 steps. Each step operates on a batch of 32 examples. The computational efficiency of the LTN and the CNN baseline systems are of the same order of magnitude.

表1：在单个和多个数字加法任务上训练和测试步骤的计算时间，在配备单个Nvidia Tesla V100 GPU的计算机上测量，并取1000步的平均值。每一步处理一个包含32个示例的批次。LTN和CNN基线系统的计算效率属于同一数量级。

| Model | Number of training examples | | | |
| --- | --- | --- | --- | --- |
|  | (Single Digits) | | (Multi Digits) | |
|  | 30 000 | 3 000 | 15 000 | 1 500 |
| baseline |  |  |  |  |
| LTN |  |  |  |  |
| DeepProbLog |  |  |  |  |

Table 2: Accuracy (in %) on the test set: comparison of the final results obtained with LTN and those reported with DeepProbLog[41]. Although it is difficult to compare directly the results over time (the frameworks are implemented in different libraries), while achieving similar computational efficiency as the CNN baseline, LTN also reaches similar accuracy as that reported by DeepProbLog.

表2：测试集上的准确率（百分比）：使用LTN获得的最终结果与DeepProbLog[41]报告的结果的比较。尽管直接比较随时间变化的结果是困难的（因为这些框架在不同的库中实现），但LTN在实现与CNN基线相似的计算效率的同时，也达到了与DeepProbLog报告的相似的准确率。

# 4.5. Regression

# 4.5. 回归

Another important problem in Machine Learning is regression where a relationship is estimated between one independent variable and a continuous dependent variable . The essence of regression is, therefore, to approximate a function by a function , given examples such that . In LTN one can model a regression task by defining as a learnable function whose parameter values are constrained by data. Additionally, a regression task requires a notion of equality. We, therefore, define the predicate eq as a smooth version of the symbol to turn the constraint into a smooth optimization problem.

机器学习中的另一个重要问题是回归问题，其中估计一个自变量 与一个连续的因变量 之间的关系。因此，回归的本质是用一个函数 来近似一个函数 ，给出示例 ，使得 。在LTN中，可以通过定义 为一个可学习函数，其参数值受数据约束来模拟回归任务。此外，回归任务需要一个等价的概念。因此，我们定义谓词eq为符号 的平滑版本，将约束 转换为一个平滑优化问题。

In this example, we explore regression using a problem from a real estate data with 414 examples, each described in terms of 6 real-numbered features: the transaction date (converted to a float), the age of the house, the distance to the nearest station, the number of convenience stores in the vicinity, and the latitude and longitude coordinates. The model has to predict the house price per unit area.

在这个示例中，我们使用来自房地产数据的问题来探讨回归，该数据 包含414个示例，每个示例用6个实数特征描述：交易日期（转换为浮点数）、房屋年龄、距离最近车站的距离、周边便利店的数量以及纬度和经度坐标。模型需要预测每单位面积的房价。

Domains:

samples, denoting the houses and their features.

prices, denoting the house prices. Variables:

for the samples.

for the prices.

samples.

prices.

[[19]](#footnote-177)

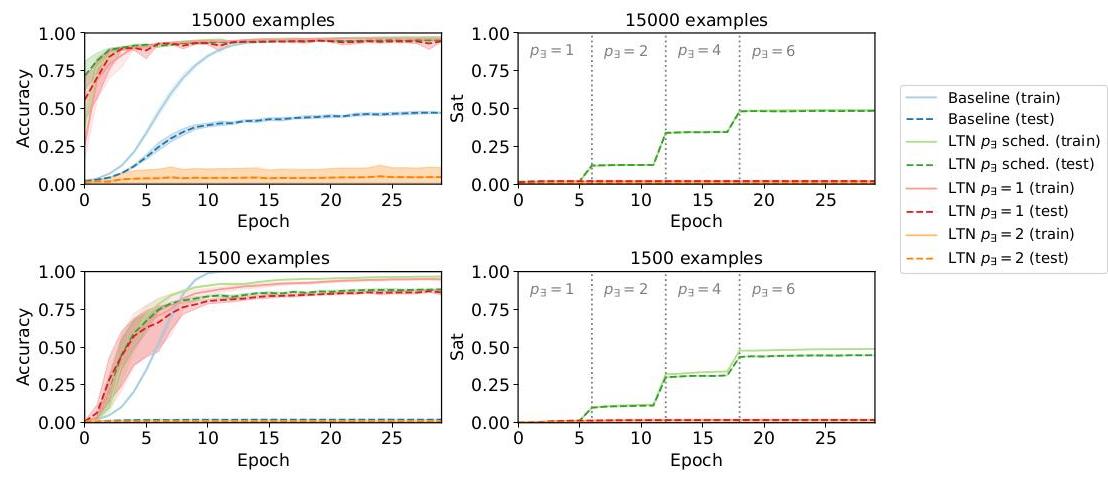


Figure 15: Multiple Digit Addition Task: Accuracy and satisfiability results (top) and results in the presence of fewer examples (bottom) in comparison with standard Deep Learning using a CNN (blue lines).

图15：多位数加法任务：准确性和满意度结果（顶部）以及在示例较少时（底部）与标准深度学习使用CNN（蓝色线条）的比较结果。

# Functions:

# 函数：

, the regression function to be learned.

，需要学习的回归函数。

samples, prices.

示例， 价格。

# Predicates:

# 谓词：

, a smooth equality predicate that measures how similar and are.

，一个平滑等价谓词，用于衡量 和 的相似度。

prices, prices.

价格，价格。

Axioms:

公理：

Notice again the use of Diag: when grounding and onto sequences of values, this is done by obeying a one-to-one correspondence between the sequences. In other words, we aggregate pairs of corresponding samples and prices, instead of any combination thereof.

再次注意Diag的使用：当将 和 映射到值序列时，这是通过保持序列之间的一一对应来完成的。换句话说，我们聚合了对应的样本和价格对，而不是它们的任意组合。

# Grounding:

# 实例化：

(samples) .

(样本) 。

(prices) .

(价格) 。

. Notice that this specification refers to the same number of examples for and due to the above one-to-one correspondence obtained with the use of Diag.

。注意，由于上述使用Diag得到的一一对应，此规范指的是 和 相同数量的示例。

, where the hyper-parameter is a real number that scales how strict the smooth equality is. In our experiments, we use .

，其中超参数 是一个实数，用于调整平滑等价的严格程度。 在我们的实验中，我们使用 。

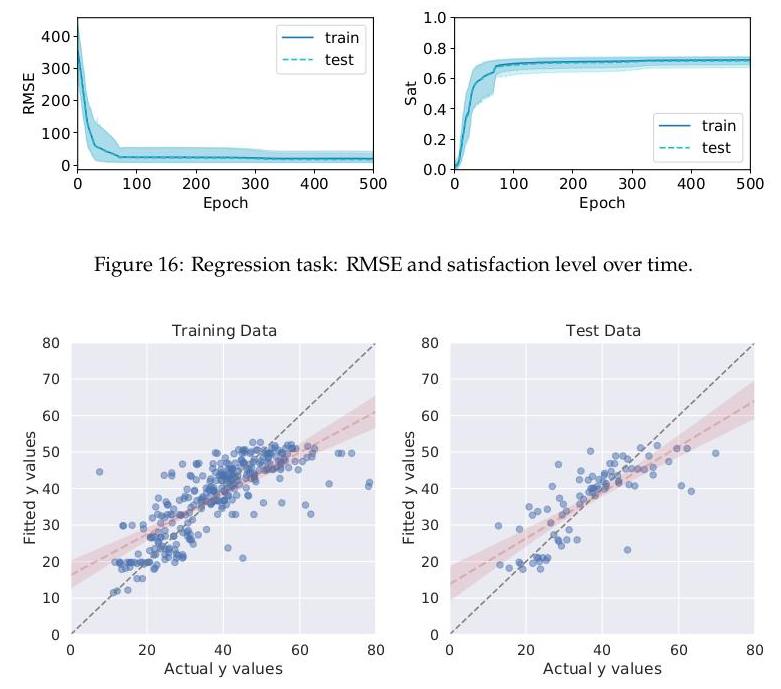


Figure 17: Visualization of LTN solving a regression problem.

图 17：LTN 解决回归问题的可视化。

, where is a multilayer perceptron which ends in one neuron

，其中 是一个以单个神经元结束的多层感知器

corresponding to a price prediction, with a linear output layer (no activation function).

对应于价格预测，具有线性输出层（无激活函数）。

# Learning:

# 学习：

The theory is constrained by the parameters of the model of . LTN is used to estimate such parameters by maximizing the satisfaction of the knowledge-base, in the usual way. Approximating using with , as before, we randomly split the data set into 330 examples for training and 84 examples for testing. Figure 16 shows the satisfaction level over 500 epochs. We also plot the Root Mean Squared Error (RMSE) between the predicted prices and the labels (i.e. actual prices, also known as target values). We visualize in Figure 17 the strong correlation between actual and predicted prices at the end of one of the runs.

该理论受限于 模型的参数。LTN 通过最大化知识库的满意度来估计这些参数，按照通常的方式。使用 和之前的 来近似 ，我们将数据集随机分为 330 个训练示例和 84 个测试示例。图 16 显示了在 500 个时期内的满意度水平。我们还绘制了预测价格和标签（即实际价格，也称为目标值）之间的均方根误差（RMSE）。我们在图 17 中可视化了实际价格和预测价格在一次运行结束时的强烈相关性。

# 4.6. Unsupervised Learning (Clustering)

# 4.6. 无监督学习（聚类）

In unsupervised learning, labels are either not available or are not used for learning. Clustering is a form of unsupervised learning whereby, without labels, the data is characterized by constraints

在无监督学习中，标签要么不可用，要么不用于学习。聚类是一种无监督学习形式，在没有标签的情况下，数据通过约束来表征

[[20]](#footnote-194)

alone. LTN can formulate such constraints, such as:

仅此而已。LTN 可以制定这样的约束，例如：

* clusters should be disjoint,
* 簇应该是不相交的，
* every example should be assigned to a cluster,
* 每个示例都应该被分配到一个簇，
* a cluster should not be empty,
* 一个簇不应该为空，
* if the points are near, they should belong to the same cluster,
* 如果点靠近，它们应该属于同一个簇，
* if the points are far, they should belong to different clusters, etc.

如果点之间的距离较远，它们应该属于不同的簇，等等。

Domains:

领域：

points, denoting the data to cluster.

点，表示要聚类的数据。

points\_pairs, denoting pairs of examples.

点对，表示示例的对。

clusters, denoting the cluster.

簇，表示簇。

Variables:

变量：

for all points.

points.

clusters.

# Predicates:

# 谓词：

, the truth degree of a given point belonging in a given cluster.

，表示一个给定的点属于一个给定簇的真值程度。

points, clusters.

点，簇。

Axioms:

公理：

Notice the use of guarded quantifiers: all the pairs of points with Euclidean distance lower (resp. higher) than a value (resp. ) should belong in the same cluster (resp. should not). and are arbitrary threshold values that define some of the closest and most distant pairs of points. In our example, they are set to, respectively, 0.2 and 1.0.

注意使用守卫量词：所有欧几里得距离小于（resp. 大于）一个值 （resp. ）的点对应该属于同一个簇（resp. 不应该）。 和 是定义一些最近和最远的点对的任意阈值。在我们的例子中，它们分别设置为 0.2 和 1.0。

As done in the example of Section 4.2, the clustering predicate has mutually exclusive satisfiability scores for each cluster using a softmax layer. Therefore, there is no explicit constraint about clusters being disjoint.

如第 4.2 节示例所示，聚类谓词对每个簇使用 softmax 层具有互斥的可满足性得分。因此，没有关于簇互斥的显式约束。

# Grounding:

# 实例化：

(points) .

（点） 。

(clusters) , we use one-hot vectors to represent a choice of 4 clusters.

（簇） ，我们使用独热向量来表示选择 4 个簇的选项。

, that is, is a sequence of points. .

，即 是一个由 点组成的序列。 。

.

。

, where MLP has 4 output neurons corresponding to the 4 clusters.

，其中 MLP 有 4 个输出神经元，对应于 4 个簇。

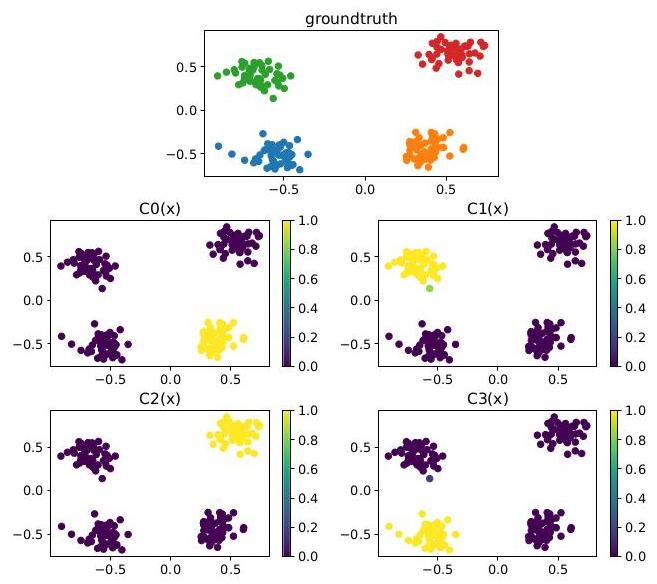


Figure 18: LTN solving a clustering problem by constraint optimization: ground-truth (top) and querying of each cluster and , in turn.

图 18：LTN 通过约束优化解决聚类问题：真实情况（顶部）和轮流查询每个簇 和 。

# Learning:

# 学习：

We use the stable real product configuration to approximate the logical operators. For , we use with . For , we use with during the first 100 epochs, and thereafter, as a simplified version of the schedule used in Section 4.4. The formula aggregator is approximated by with . The model is trained for a total of 1000 epochs using the Adam optimizer, which is sufficient for LTN to solve the clustering problem shown in Figure 18 Ground-truth data for this task was generated artificially by creating 4 centers, and generating 50 random samples from a multivariate Gaussian distribution around each center. The trained LTN achieves a satisfaction level of the clustering constraints of 0.857 .

我们使用稳定的实产品配置来近似逻辑运算符。对于 ，我们使用 与 。对于 ，在最初的100个训练周期内，我们使用 与 ，之后使用 ，作为第4.4节中使用的计划的简化版本。公式聚合器由 与 近似。模型使用Adam优化器训练总共1000个周期，这对于LTN解决图18所示的聚类问题已经足够。此任务的地面真实数据是通过创建4个中心，并从每个中心周围的多元高斯分布中生成50个随机样本人工生成的。训练后的LTN实现了聚类约束的满意度水平为0.857。

# 4.7. Learning Embeddings with LTN

# 4.7. 使用LTN学习嵌入

A classic example of Statistical Relational Learning is the smokers-friends-cancer example introduced in [55]. Below, we show how this example can be formalized in LTN using semi-supervised embedding learning.

统计关系学习的经典例子是引入于[55]的吸烟者-朋友-癌症例子。下面，我们展示如何使用半监督嵌入学习在LTN中形式化这个例子。

There are 14 people divided into two groups and . Within each group, there is complete knowledge about smoking habits. In the first group, there is complete knowledge about who has and who does not have cancer. Knowledge about the friendship relation is complete within each group only if symmetry is assumed, that is, (friends ).

有14个人被分为两个组 和 。在每组内部，对吸烟习惯的了解是完整的。在第一组中，对谁患癌症以及谁没有患癌症的了解是完整的。只有在假设对称性的情况下，即 （朋友 ）时，每组内部对友谊关系的了解才是完整的。

Otherwise, knowledge about friendship is incomplete in that it may be known that e.g. is a friend of , and it may be not known whether is a friend of . Finally, there is general knowledge about smoking, friendship, and cancer, namely that smoking causes cancer, friendship is normally symmetric and anti-reflexive, everyone has a friend, and smoking propagates (actively or passively) among friends. All this knowledge is represented in the axioms further below.

否则，关于友谊的知识是不完整的，因为可能知道例如 是 的朋友，但可能不知道 是否是 的朋友。最后，关于吸烟、友谊和癌症有一般性的知识，即吸烟导致癌症，友谊通常是对称的和反对称的，每个人都有朋友，而且吸烟（主动或被动）在朋友之间传播。所有这些知识都在下面进一步给出的公理中有所体现。

# Domains:

# 领域：

people, to denote the individuals.

人，用来表示个体。

Constants:

常量：

, the 14 individuals. Our goal is to learn an adequate embedding for each

，14个个体。我们的目标是学习每个常量的足够嵌入。

constant.

常量。

Variables:

变量：

ranging over the individuals.

跨个体的范围。

people.

人。

# Predicates:

# 谓词：

for smokes, for friends, for cancer.

代表吸烟， 代表朋友， 代表癌症。

people. people, people.

人。 人，人。

# Axioms:

# 公理：

Let and be the two groups of individuals.

设 和 是两个个体群。

Let be the smokers; knowledge is complete in both groups.

设 为吸烟者；两组的知识都是完整的。

Let be the individuals with cancer; knowledge is complete in only.

设 为癌症患者；只有 的知识是完整的。

Let be the

设 为

set of friendship relations; knowledge is complete if assuming symmetry.

友谊关系的集合；如果假设对称性，则知识是完整的。

These facts are illustrated in Figure 20a

这些事实在图20a中有所说明。

We have the following axioms:

我们有以下公理：

for

用于 (49)

for

用于 (50)

for

用于

(51)

Notice that the knowledge base is not satisfiable in the strict logical sense of the word. For

注意到知识库在严格的逻辑意义上是不可满足的。例如， 被说成是吸烟但并没有癌症，这与规则不一致

instance, is said to smoke but not to have cancer, which is inconsistent with the rule

例如， 被认为吸烟但没有癌症，这与规则不一致

. Hence, it is important to adopt a fuzzy approach as done with MLN or a many-valued fuzzy logic interpretation as done with LTN.

因此，采用模糊方法（如 MLN 所做的）或多值模糊逻辑解释（如 LTN 所做的）是很重要的。

# Grounding:

# 地面实况：

(people) . The model is expected to learn embeddings in .

（人） 。模型预计在学习 中的嵌入。

. Every individual is associated with a vector of 5 real

。每个个体都与一个包含5个实数的向量相关联。

numbers. The embedding is initialized randomly uniformly.

嵌入是随机均匀初始化的。

, where MLP\_ has 1 output neuron.

，其中 MLP\_ 有1个输出神经元。

, where MLP\_ has 1 output neuron.

，其中 MLP\_ 有1个输出神经元。

, where MLP has 1 output neuron.

，其中 MLP 有1个输出神经元。

The MLP models for are kept simple, so that most of the learning is focused on the embedding.

的 MLP 模型保持简单，以便大部分学习都集中在嵌入上。

# Learning:

# 学习：

We use the stable real product configuration to approximate the operators. For , we use with for all the rules, except for rules (52) and (53), where we use . The intuition behind this choice of is that no outliers are to be accepted for the friendship relation since it is expected to be symmetric and anti-reflexive, but outliers are accepted for the other rules. For , we use with during the first 200 epochs of training, and thereafter, with the same motivation as that of the schedule used in Section 4.4 The formula aggregator is approximated by with .

我们使用稳定的实数乘积配置来近似操作符。对于 ，我们使用 与 用于所有规则，除了规则（52）和（53），在那里我们使用 。选择 的直觉是，由于友谊关系预计是对称和反对称的，所以不允许有异常值，而对于其他规则则接受异常值。对于 ，我们在训练的前200个周期内使用 与 ，之后使用 ，这与第4.4节中使用的计划的动机相同。公式聚合器通过 与 来近似。

Figure 19 shows the satisfiability over 1000 epochs of training. At the end of one of these runs, we query for each individual; the results are shown in Figure 20b We also plot the principal components of the learned embeddings [51] in Figure 21. The friendship relations are learned as expected: (56) "smoking implies cancer" is inferred for group 2 even though such information was not present in the knowledge base. For group 1, the given facts for smoking and cancer for the individuals and are slightly altered, as these were inconsistent with the rules. (the rule for smoking propagating via friendship (55) is incompatible with many of the given facts). Increasing the satisfaction of this rule would require decreasing the overall satisfaction of the knowledge base, which explains why it is partly ignored by LTN during training. Finally, it is interesting to note that the principal components for the learned embeddings seem to be linearly separable for the smoking and cancer classifiers (c.f. Figure 21, top right and bottom right plots).

图19显示了在1000个训练周期中的满意度。在这些运行之一结束时，我们对每个个体查询 ；结果如图20b所示。我们还在图21中绘制了学习到的嵌入的主要成分 [51]。友谊关系如预期的那样被学习到：(56) “吸烟意味着癌症”被推断为第2组，尽管知识库中没有这样的信息。对于第1组，给定的吸烟和癌症个体信息 和 略有改变，因为这些与规则不一致（通过友谊传播吸烟的规则（55）与许多给定事实不兼容）。增加这个规则的满意度将需要降低知识库的整体满意度，这解释了为什么在训练过程中LTN部分忽略了它。最后，值得注意的是，学习到的嵌入的主要成分对于吸烟和癌症分类器似乎是可以线性分离的（参见图21，右上和右下图表）。

# Querying:

# 查询：

To illustrate querying in LTN, we query over time two formulas that are not present in the knowledge-base:

为了说明在LTN中的查询，我们在知识库中不存在的两个公式上随时间进行查询：

We use when approximating since the impact of an outlier at querying time should be seen as more important than at learning time. It can be seen that as the grounding approaches satisfiability of the knowledge-base, approaches true, whereas approaches false (c.f. Figure 20a).

我们在近似 时使用 ，因为在查询时间异常值的影响应被视为比在学习时间更重要。可以看出，随着接地接近知识库的满意度， 接近于真，而 接近于假（参见图20a）。

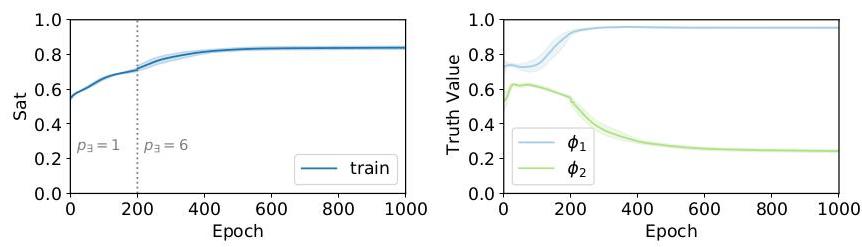
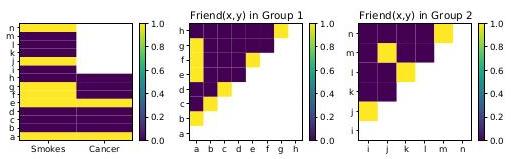


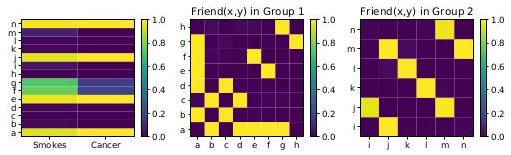
Figure 19: Smoker-Friends-Cancer example: Satisfiability levels during training (left) and truth-values of queries and over time (right).

图19：吸烟者-朋友-癌症示例：训练期间的满意度水平（左）和查询 和 随时间的真值（右）。



(a) Incomplete facts in the knowledge-base: axioms for smokers and cancer for individuals to (left), friendship relations in group 1 (middle), and friendship relations in group 2 (right).

(a) 知识库中的不完整事实：关于吸烟者和癌症的公理 至 （左），团体1中的友谊关系（中），以及团体2中的友谊关系（右）。



(b) Querying all the truth-values using LTN after training: smokers and cancer (left), friendship relations (middle and right).

(b) 训练后使用LTN查询所有真值：吸烟者和癌症（左），友谊关系（中右）。

Figure 20: Smoker-Friends-Cancer example: Illustration of the facts before and after training.

图20：吸烟者-朋友-癌症示例：训练前后的实例说明。

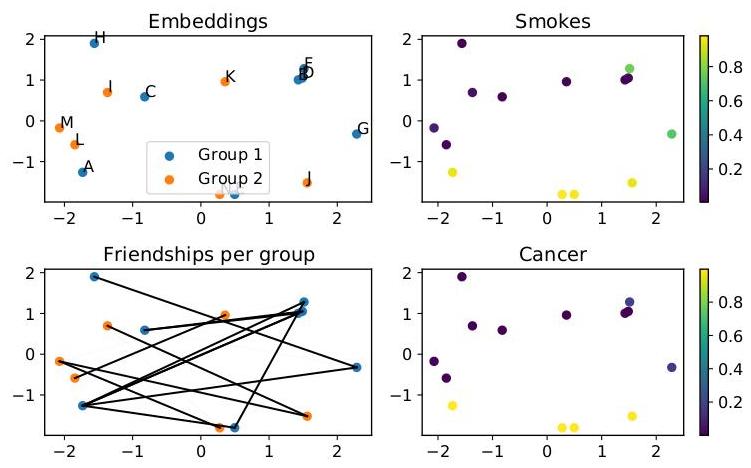


Figure 21: Smoker-Friends-Cancer example: learned embeddings showing the result of applying PCA on the individuals (top left); truth-values of smokes and cancer predicates for each embedding (top and bottom right); illustration of the friendship relations which are satisfied after learning (bottom left).

图21：吸烟者-朋友-癌症示例：学习到的嵌入表示应用PCA后的结果（左上）；每个嵌入的smokes和cancer谓词的真值（右上和右下）；学习后满足的友谊关系的说明（左下）。

# 4.8. Reasoning in LTN

# 4.8 LTN中的推理

The essence of reasoning is to find out if a closed formula is the logical consequence of a knowledge-base . Section 3.4 introduced two approaches to this problem in LTN:

推理的本质是找出一个封闭公式 是否是知识库 的逻辑后果。第3.4节介绍了在LTN中解决此问题的两种方法：

* By simply querying after learning 27 one seeks to verify if for the grounded theories that maximally satisfy , the grounding of gives a truth-value greater than a threshold . This often requires checking an infinite number of groundings. Instead, the user approximates the search for these grounded theories by running the optimization a fixed number of times only.
* 通过在学习后简单地查询，人们试图验证对于最大化满足 的 grounded theories， 的 grounding 是否给出大于阈值 的真值。这通常需要检查无限多个 grounding。相反，用户通过仅运行固定次数的优化来近似这些 grounded theories 的搜索。
* Reasoning by refutation one seeks to find out a counter-example: a grounding that satisfies the knowledge-base but not the formula given the threshold . A search is performed here using a different objective function.
* 通过反驳进行推理，试图找到一个反例：一个满足知识库 但不满足给定阈值 下的公式 的 grounding。这里使用不同的目标函数进行搜索。

We now demonstrate that reasoning by refutation is the preferred option using a simple example where we seek to find out whether .

我们现在通过一个简单的示例来证明反驳推理是首选的选项，在该示例中我们试图找出 是否成立。

# Propositional Variables:

# 命题变量：

The symbols and denote two propositionial variables.

符号 和 表示两个命题变量。

Axioms:

公理：

[[21]](#footnote-234)

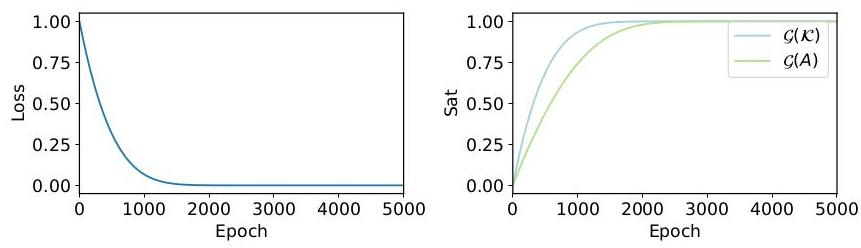


Figure 22: Querying after learning: 10 runs of the optimizer with objective . All runs converge to the optimum ; the grid search misses the counter-example.

图22：学习后的查询：优化器针对目标 运行10次。所有运行都收敛到最优解 ；网格搜索错过了反例。

# Grounding:

# 地面：

, where and are two real-valued parameters. The set of parameters is therefore . At initialization, .

，其中 和 是两个实值参数。因此，参数集为 。在初始化时， 。

We use the probabilistic-sum to approximate , resulting in the following satisfiability measure

我们使用概率和 来近似 ，得到以下满意度度量

There are infinite global optima maximizing the satisfiability of the theory, as any such that (resp. ) gives a satisfiability for any value of (resp. . As expected, the following groundings are examples of global optima:

存在无限多个全局最优解，使理论的可满足性最大化，因为任何满足 的 （分别地 ）为任何 （分别地 ）的值提供了一个可满足性 。如预期的那样，以下地面实例是全球最优解的例子：

# Reasoning:

# 推理：

? That is, given the threshold , does every such that verify . Immediately, one can notice that this is not the case. For instance, the grounding is a counter-example.

吗？也就是说，给定阈值 ，是否每个 使得 都验证 。立刻可以注意到，情况并非如此。例如，地面 是一个反例。

If one simply reasons by querying multiple groundings after learning with the usual objective , the results will all converge to and . Every run of the optimizer will increase and simultaneously until they reach the optimum . Because the grid search always converges to the same point, no counter-example is found and the logical consequence is mistakenly assumed true. This is illustrated in Figure 22

如果在学习后仅仅通过查询多个地面实例，并使用通常的目标 进行推理，结果将都会收敛到 和 。优化器的每次运行都会同时增加 和 直到它们达到最优解 。因为网格搜索总是收敛到同一点，所以找不到反例，逻辑后果被错误地认为是正确的。这在图22中得到了说明。

Reasoning by refutation, however, the objective function has an incentive to find a counterexample with , as illustrated in Figure 23 LTN converges to the optimum , which refutes the logical consequence.

然而，通过反驳进行推理，目标函数有动机找到一个反例 ，如图23所示，LTN收敛到最优解 ，这反驳了逻辑后果。

[[22]](#footnote-242)

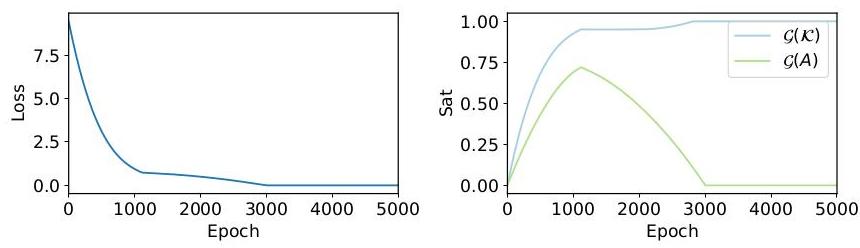


Figure 23: Reasoning by refutation: one run of the optimizer with objective , . In the first training epochs, the directed search prioritizes the satisfaction of the knowledge base. Then, the minimization of starts to weigh in more and the search focuses on finding a counter-example. Eventually, the run converges to the optimum , which refutes the logical consequence.

图23：通过反驳进行推理：优化器在一次运行中使用目标 、 。在最初的训练周期中，有向搜索优先满足知识库的满意度。然后， 的最小化开始占据更大的比重，搜索重点转向寻找反例。最终，运行收敛到最优解 ，这反驳了逻辑后果。

# 5. Related Work

# 5. 相关工作

The past years have seen considerable work aiming to integrate symbolic systems and neural networks. We shall focus on work whose objective is to build computational models that integrate deep learning and logical reasoning into a so-called end-to-end (fully differentiable) architecture. We summarize a categorization in Figure 24 where the class containing LTN is further expanded into three sub-classes. The sub-class highlighted in red is the one that contains LTN. The reason why one may wish to combine symbolic and neural networks into a neurosymbolic system may vary, c.f. [17] for a recent comprehensive overview of approaches and challenges for neurosymbolic AI.

过去几年，有很多工作致力于将符号系统和神经网络集成。我们将关注那些旨在构建计算模型的工作，这些模型将深度学习和逻辑推理集成到所谓的端到端（完全可微分）架构中。我们在图24中总结了分类，其中包含LTN的类别进一步扩展为三个子类别。红色突出显示的子类别是包含LTN的。为什么人们可能希望将符号 和神经网络结合成一个神经符号 系统的原因可能会有所不同，参见[17]了解神经符号AI方法和挑战的最新全面概述。

# 5.1. Neural architectures for logical reasoning

# 5.1. 用于逻辑推理的神经网络架构

These use neural networks to perform (probabilistic) inference on logical theories. Early work in this direction has shown correspondences between various logical-symbolic systems and neural network models [27, 32, 52, 63, 65]. They have also highlighted the limits of current neural networks as models for knowledge representation. In a nutshell, current neural networks (including deep learning) have been shown capable of representing propositional logic, nonmonotonic logic programming, propositional modal logic, and fragments of first-order logic, but not full first-order or higher-order logic. Recently, there has been a resurgence of interest in the topic with many proposals emerging [13, 48, 53]. In [13], each clause of a Stochastic Logic Program is converted into a factor graph with reasoning becoming differentiable so that it can be implemented by deep networks. In [49], a differentiable unification algorithm is introduced with theorem proving sought to be carried out inside the neural network. Furthermore, in [11, 49] neural networks are used to learn reasoning strategies and logical rule induction.

这些方法使用神经网络对逻辑理论进行（概率性）推理。早期在这方面的工作已经展示了各种逻辑符号系统与神经网络模型之间的对应关系 [27, 32, 52, 63, 65]。它们还突显了当前神经网络作为知识表示模型的局限性。简而言之，当前的神经网络（包括深度学习）已经证明能够表示命题逻辑、非单调逻辑编程、命题模态逻辑以及一阶逻辑的片段，但无法表示完整的一阶逻辑或高阶逻辑。最近，这一主题重新引起了人们的兴趣，出现了许多新的提议 [13, 48, 53]。在 [13] 中，随机逻辑程序的每一条子句都被转换为一个因子图，推理变得可微分，因此可以被深度网络实现。在 [49] 中，引入了一种可微的统一算法，定理证明被寻求在神经网络内部进行。此外，在 [11, 49] 中，神经网络被用来学习推理策略和逻辑规则归纳。

Reasoning with LTN (Section 3.4) is reminiscent of this category, given that knowledge is not represented in a traditional logical language but in Real Logic.

使用LTN（第3.4节）进行推理让人联想到这一类别，因为知识不是以传统的逻辑语言表示，而是以实逻辑表示。

# 5.2. Logical specification of neural network architectures

# 5.2. 神经网络架构的逻辑规范

Here the goal is to use a logical language to specify the architecture of a neural network. Examples include [13, 24, 26, 56, 66]. In [26], the languages of extended logic programming (logic programs with negation by failure) and answer set programming are used as background knowledge to set up the initial architecture and set of weights of a recurrent neural network, which is subsequently trained from data using backpropagation. In [24], first-order logic programs in the form of Horn clauses are used to define a neural network that can solve Inductive Logic Programming tasks, starting from the most specific hypotheses covering the set of examples. Lifted relational neural networks [66] is a declarative framework where a Datalog program is used as a compact specification of a diverse range of existing advanced neural architectures, with a particular focus on Graph Neural Networks (GNNs) and their generalizations. In [56] a weighted Real Logic is introduced and used to specify neurons in a highly modular neural network that resembles a tree structure, whereby neurons with different activation functions are used to implement the different logic operators.

在这里，目标是使用逻辑语言来指定神经网络的架构。例子包括 [13, 24, 26, 56, 66]。在 [26] 中，扩展逻辑编程（带有失败否定的逻辑程序）和答案集编程语言被用作背景知识，以建立递归神经网络的初始架构和权重集合，该网络随后通过反向传播从数据中进行训练。在 [24] 中，以 Horn 子句形式的一阶逻辑程序被用来定义一个能够解决归纳逻辑编程任务的神经网络，从覆盖示例集的最具体假设开始。提升关系神经网络 [66] 是一个声明性框架，其中使用 Datalog 程序作为现有各种高级神经架构的紧凑规范，特别关注图神经网络（GNNs）及其泛化。在 [56] 中引入了加权实数逻辑，并用于指定一个高度模块化神经网络的神经元，该网络的结构类似于树，其中使用具有不同激活函数的神经元来实现不同的逻辑运算符。

To some extent, it is also possible to specify neural architectures using logic in LTN. For example, a user can define a classifier as the formula . becomes a computational graph that combines the sub-architectures , and according to the syntax of the logical formula.

在一定程度上，也可以使用 LTN 中的逻辑来指定神经架构。例如，用户可以定义一个分类器 作为公式 。 变成了一个计算图，它根据逻辑公式的语法组合子架构 和 。

# 5.3. Neurosymbolic architectures for the integration of inductive learning and deductive reasoning

# 5.3. 神经符号架构：用于归纳学习和演绎推理的集成

These architectures seek to enable the integration of inductive and deductive reasoning in a unique fully differentiable framework [15, 23, 41, 46, 47]. The systems that belong to this class combine a neural component with a logical component. The former consists of one or more neural networks, the latter provides a set of algorithms for performing logical tasks such as model checking, satisfiability, and logical consequence. These two components are tightly integrated so that learning and inference in the neural component are influenced by reasoning in the logical component and vice versa. Logic Tensor Networks belong to this category. Neurosymbolic architectures for integrating learning and reasoning can be further separated into three sub-classes:

这些架构旨在实现归纳推理和演绎推理在一个独特的全微分框架中的整合 [15, 23, 41, 46, 47]。属于这一类的系统结合了神经组件和逻辑组件。前者由一个或多个神经网络组成，后者提供了一套用于执行逻辑任务的算法，如模型检查、可满足性和逻辑推论。这两个组件紧密集成，使得神经组件中的学习和推理受到逻辑组件推理的影响，反之亦然。逻辑张量网络属于此类。神经符号架构用于整合学习和推理可以进一步分为三个子类别：

1. Approaches that introduce additional layers to the neural network to encode logical constraints which modify the predictions of the network. This sub-class includes Deep Logic Models [46] and Knowledge Enhanced Neural Networks [15].

1. 在神经网络中引入额外的层以编码逻辑约束，这些约束修改网络的预测。这个子类别包括深度逻辑模型 [46] 和知识增强神经网络 [15]。

2. Approaches that integrate logical knowledge as additional constraints in the objective function or loss function used to train the neural network (LTN and [23, 33, 47]).

2. 在用于训练神经网络的损失函数或目标函数中整合逻辑知识作为额外的约束（LTN 和 [23, 33, 47]）。

3. Approaches that apply (differentiable) logical inference to compute the consequences of the predictions made by a set of base neural networks. Examples of this sub-class are DeepProblog [41] and Abductive Learning [14].

3. 应用（可微分的）逻辑推理来计算一组基础神经网络预测的后果。这个子类别的例子包括 DeepProblog [41] 和假设学习 [14]。

In what follows, we revise recent neurosymbolic architectures in the same class as LTN: Integrating learning and reasoning.

接下来，我们回顾了与 LTN 同类的最近神经符号架构：整合学习和推理。

Systems that modify the predictions of a base neural network:. Among the approaches that modify the predictions of the neural network using logical constraints are Deep Logic Models [46] and Knowledge Enhanced Neural Networks [15]. Deep Logic Models (DLM) are a general architecture for learning with constraints. Here, we will consider the special case where constraints are expressed by logical formulas. In this case, a DLM predicts the truth-values of a set of ground atoms of a domain . It consists of two models: a neural network which takes as input the features of the elements of and produces as output an evaluation for all the ground atoms, i.e. , and a probability distribution which is modeled by an undirected graphical model of the exponential family with each logical constraint characterized by a clique that contains the ground atoms, rather similarly to GNNs. The model returns the assignment to the atoms that maximize the weighted truth-value of the constraints and minimize the difference between the prediction of the neural network and a target value . Formally:

系统修改基础神经网络的预测：在修改神经网络预测的逻辑约束方法中，有深度逻辑模型 [46] 和知识增强神经网络 [15]。深度逻辑模型（DLM）是一种用于带约束学习的一般架构。在这里，我们将考虑约束由逻辑公式表达的特殊情况。在这种情况下，DLM 预测一组 领域 的地面原子的真值。它包括两个模型：一个神经网络 ，它以元素 的特征为输入，并为所有地面原子输出一个评估 ，即 ，以及一个由指数族的无向图模型建模的概率分布 ，每个逻辑约束由包含地面原子的团表征，与 GNNs 类似。模型返回最大化约束加权真值并使神经网络预测与目标值 之间的差异最小化的原子分配。

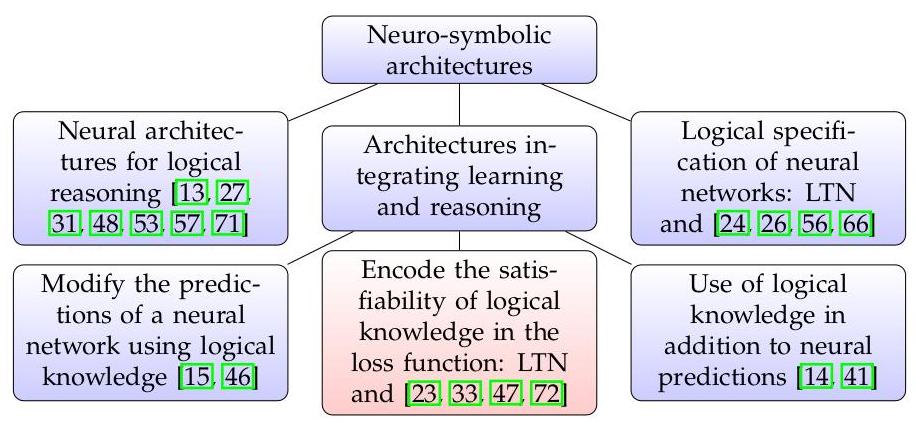


Figure 24: Three classes of neurosymbolic approaches with Architectures Integrating Learning and Reasoning further subdivided into three sub-classes, with LTN belonging to the sub-class highlighted in red.

图 24：三类结合学习和推理的神经符号方法架构进一步细分为三个子类，其中 LTN 属于用红色突出显示的子类。

Each corresponds to a ground propositional formula which is evaluated w.r.t. the target truth assignment , and is the weight associated with formula . Intuitively, the upper model (the undirected graphical model) should modify the prediction of the lower model (the neural network) minimally to satisfy the constraints. and are truth-values of all the ground atoms obtained from the constraints appearing in the upper model in the domain specified by the data input.

每个 对应于一个关于目标真值分配 的地面命题公式的评估， 是与公式 相关联的权重。直观上，上层模型（无向图模型）应尽可能少地修改下层模型（神经网络）的预测以满足约束。 和 是从数据输入指定的领域中出现在上层模型中的约束得到的所有地面原子的真值。

Similar to LTN, DLM evaluates constraints using fuzzy semantics. However, it considers only propositional connectives, whereas universal and existential quantifiers are supported in LTN.

与 LTN 相似，DLM 使用模糊语义评估约束。然而，它仅考虑命题连接词，而 LTN 支持全称和存在量词。

Inference in DLM requires maximizing the prediction of the model, which might be prohibitive in the presence of a large number of instances. In LTN, inference involves only a forward pass through the neural component which is rather simple and can be carried out in parallel. However, in DLM the weight associated with constraints can be learned, while in LTN they are specified in the background knowledge.

在 DLM 中进行推理需要最大化模型的预测，这在实例数量众多的情况下可能是禁止的。在 LTN 中，推理仅涉及通过神经组件的前向传递，这相对简单且可以并行执行。然而，在 DLM 中与约束相关的权重可以学习得到，而在 LTN 中它们是在背景知识中指定的。

The approach taken in Knowledge Enhanced Neural Networks (KENN) [15] is similar to that of DLM. Starting from the predictions made by a base neural network , KENN adds a knowledge enhancer, which is a function that modifies based on a set of weighted constraints formulated in terms of clauses. The formal model can be specified as follows:

知识增强神经网络（KENN）[15]采用的方法与 DLM 类似。从基础神经网络 的预测 出发，KENN 添加了一个知识增强器，这是一个基于一组以子句形式表述的加权约束修改 的函数。该形式模型可以指定如下：

where are the pre-activations of , sign is a vector of the same dimension of containing and , such that (resp. ) if the -th atom occurs positively (resp. negatively) in , or otherwise, and is the element-wise product. KENN learns the weights of the clauses in the background knowledge and the base network parameters by minimizing some standard loss,(e.g. cross-entropy) on a set of training data. If the training data is inconsistent with the constraint, the weight of the constraint will be close to zero. This intuitively implies that the latent knowledge present in the data is preferred to the knowledge specified in the constraints. In LTN, instead, training data and logical constraints are represented uniformly with a formula, and we require that they are both satisfied. A second difference between KENN and LTN is the language: while LTN supports constraints written in full first-order logic, constraints in KENN are limited to universally quantified clauses.

其中 是 的预激活，sign 是一个与 维度相同的向量，包含 和 ，使得当 （分别地 ）如果第 个原子在 中以正（分别地负）出现，否则为 ，而 是逐元素乘积。KENN 通过最小化某些标准损失（例如交叉熵）在一组训练数据上学习背景知识中子句的权重 和基础网络参数 。如果训练数据与约束不一致，则约束的权重将接近于零。这直观地表明数据中隐含的知识比约束中指定的知识更受欢迎。在 LTN 中，相反，训练数据和逻辑约束以统一的方式用公式表示，并且我们要求它们都得到满足。KENN 和 LTN 之间的第二个区别是语言：虽然 LTN 支持用完整的一阶逻辑编写的约束，但 KENN 中的约束仅限于全称量化的子句。

Systems that add knowledge to a neural network by adding a term to the loss function:. In [33], a framework is proposed that learns simultaneously from labeled data and logical rules. The proposed architecture is made of a student network and a teacher network, denoted by . The student network is trained to do the actual predictions, while the teacher network encodes the information of the logical rules. The transfer of information from the teacher to the student network is done by defining a joint loss for both networks as a convex combination of the loss of the student and the teacher. If is the prediction of the student network for input , the loss is defined as:

通过向损失函数添加项来向神经网络添加知识的系统：在 [33] 中，提出了一个同时从标记数据和逻辑规则中学习的框架。所提出的架构由一个学生网络 和一个教师网络组成，后者表示为 。学生网络被训练来进行实际预测，而教师网络编码逻辑规则的信息。从教师到学生网络的信息传递是通过为两个网络定义一个联合损失 来实现的，该损失是学生和教师损失的凸组合。如果 是学生网络对输入 的预测，则损失定义为：

where measures how much the predictions satisfy the constraints encoded in the set of clauses . Training is iterative. At every iteration, the parameters of the student network are optimized to minimize the loss that takes into account the feedback of the teacher network on the predictions from the previous step. The main difference between this approach and LTN is how the constraints are encoded in the loss. LTN integrates the constraints in the network and optimizes directly their satisfiability with no need for additional training data. Furthermore, the constraints proposed in [33] are universally quantified formulas only.

其中 衡量预测 满足编码在条款集合 中的约束的程度。训练是迭代的。在每次迭代中，学生网络的参数被优化以最小化损失，该损失考虑了教师网络对上一步预测的反馈。这种方法与 LTN 的主要区别在于约束是如何编码在损失中的。LTN 将约束集成到网络中，并直接优化它们的可满足性，无需额外的训练数据。此外，文献 [33] 提出的约束仅限于全称量化的公式。

The approach adopted by Lxrcs [47] is analogous to the first version of LTN [61]. Logical constraints are translated into a loss function that measures the (negative) satisfiability level of the network. Differently from LTN, formulas in Lyrics can be associated with weights that are hyper-parameters. In [47], a logarithmic loss function is also used when the product t-norm is adopted. Notice that weights can also be added (indirectly) to LTN by introducing a 0-ary predicate to represent a constraint of the form . An advantage of this approach would be that the weights could be learned.

Lxrcs [47] 采取的方法与 LTN [61] 的第一个版本类似。逻辑约束被翻译为一个损失函数，该函数衡量网络的（负）可满足性水平。与 LTN 不同，Lyrics 中的公式可以与权重相关联，这些权重是超参数。在 [47] 中，当采用乘积 t-范数时，也使用了对数损失函数。注意，通过引入一个 0-元谓词 来表示形式为 的约束，也可以（间接地）向 LTN 添加权重。这种方法的一个优点是权重可以被学习。

In [72], a neural network computes the probability of some events being true. The neural network should satisfy a set of propositional logic constraints on its output. These constraints are compiled into arithmetic circuits for weighted model counting, which are then used to compute a loss function. The loss function then captures how close the neural network is to satisfying the propositional logic constraints.

在 [72] 中，一个神经网络计算某些事件为真的概率。神经网络应该满足其输出上的一组命题逻辑约束。这些约束被编译为加权模型计数的算术电路，然后用于计算损失函数。损失函数随后捕捉神经网络满足命题逻辑约束的接近程度。

Systems that apply logical reasoning on the predictions of a base neural network:. The most notable architecture in this category is DeepProblog [41]. DeepProblog extends the ProbLog framework for probabilistic logic programming to allow the computation of probabilistic evidence from neural networks. A ProbLog program is a logic program where facts and rules can be associated with probability values. Such values can be learned. Inference in ProbLog to answer a query is performed by knowledge compilation into a function that computes the probability that is true according to the logic program with relative frequencies . In DeepProbLog, a neural network that outputs a probability distribution over a set of atoms is integrated into ProbLog by extending the logic program with and the respective probabilities . The probability of a query is then given by , where is the input of and is the function corresponding to the logic program extended with . Given a set of queries , input vectors and ground-truths for all the queries, training is performed by minimizing a loss function that measures the distance between the probabilities predicted by the logic program and the ground-truths, as follows:

系统通过对基础神经网络预测应用逻辑推理：这一类别中最著名的架构是DeepProblog [41]。DeepProblog扩展了概率逻辑编程的ProbLog框架，允许从神经网络计算概率证据。ProbLog程序是一个逻辑程序，其中的事实和规则可以与概率值相关联。这些值可以是可学习的。在ProbLog中进行推理以回答一个查询 ，是通过将知识编译成一个函数 ，该函数根据带有相对频率 的逻辑程序计算 为真的概率。在DeepProbLog中，一个输出概率分布 的神经网络 被整合到ProbLog中，通过扩展逻辑程序以包含 及其相应的概率 。查询 的概率随后由 给出，其中 是 的输入， 是扩展了 的逻辑程序对应的函数。给定一组查询 、输入向量 和所有查询的地面真实值 ，训练是通过最小化一个损失函数来进行的，该损失函数衡量逻辑程序预测的概率与地面真实值之间的距离，如下所示：

The most important difference between DeepProbLog and LTN concerns the logic on which they are based. DeepProbLog adopts probabilistic logic programming. The output of the base neural network is interpreted as the probability of certain atoms being true. LTN instead is based on many-valued logic. The predictions of the base neural network are interpreted as fuzzy truth-values (though previous work [67] also formalizes Real Logic as handling probabilities with relaxed constraints). This difference of logic leads to the second main difference between LTN and Deep-Problog: their inference mechanism. DeepProblog performs probabilistic inference (based on model counting) while LTN inference consists of computing the truth-value of a formula starting from the truth-values of its atomic components. The two types of inference are incomparable. However, computing the fuzzy truth-value of a formula is more efficient than model counting, resulting in a more scalable inference task that allows LTN to use full first-order logic with function symbols. In DeepProblog, to perform probabilistic inference, a closed-world assumption is made and a function-free language is used. Typically, DeepProbLog clauses are compiled into Sentential Decision Diagrams (SDDs) to accelerate inference considerably[36], although the compilation step of clauses into the SDD circuit is still costly.

DeepProbLog与LTN之间的最重要区别在于它们所基于的逻辑。DeepProbLog采用概率逻辑编程。基础神经网络的输出被解释为某些原子为真的概率。而LTN基于多值逻辑。基础神经网络的预测被解释为模糊的真值（尽管之前的工作[67]也将实数逻辑形式化为处理具有放松约束的概率）。这种逻辑上的差异导致了LTN和DeepProblog之间的第二个主要区别：它们的推理机制。DeepProblog执行基于模型计数的概率推理，而LTN的推理包括从原子成分的真值计算公式的真值。这两种推理类型是不可比较的。然而，计算公式的模糊真值比模型计数更高效，这使得LTN的推理任务更具可扩展性，允许使用带有函数符号的全阶逻辑。在DeepProblog中，为了执行概率推理，需要做出闭合世界假设并使用无函数语言。通常，DeepProbLog的子句会被编译成句子决策图（SDDs）以显著加速推理[36]，尽管将子句编译成SDD电路的步骤仍然代价高昂。

An approach that extends the predictions of a base neural network using abductive reasoning is [14]. Given a neural network that produces a crisp output for predicates and background knowledge in the form of a logic program , parameters of are learned alongside a set of additional rules that define a new concept w.r.t. such that, for every object with features :

一种使用溯因推理扩展基础神经网络预测的方法是[14]。给定一个产生 crisp 输出 的神经网络 用于 谓词 和以逻辑程序 形式的背景知识， 的参数 与一组额外的规则 一起学习，这些规则定义了一个关于 的新概念 ，对于每个具有特征 的对象 ：

The task is solved by iterating the following three steps:

解决任务的过程是通过迭代以下三个步骤：

1. Given the predictions of the neural network on the set of training objects, search for the best that maximize the number of objects for which (62) holds;

1. 鉴于神经网络 对训练对象集合 的预测，寻找最佳的 以最大化满足 (62) 的对象数量；

2. For each object , compute by abduction on , the explanation ;

2. 对于每个对象 ，通过在 上进行溯因推理计算解释 ；

3. Retrain with the training set .

3. 使用训练集 重新训练 。

Differently from LTN, in [14] the optimization is done separately in an iterative way. The semantics of the logic is crisp, neither fuzzy nor probabilistic, and therefore not fully differentiable. Abductive reasoning is adopted, which is a potentially relevant addition for comparison with symbolic ML and Inductive Logic Programming approaches [50].

与 LTN 不同，在 [14] 中优化是分别以迭代方式进行。该逻辑的语义是明确的，既不是模糊的也不是概率性的，因此不是完全可微分的。采用了溯因推理，这对于与符号机器学习和归纳逻辑编程方法 [50] 进行比较是一个潜在的重要补充。

Various other loosely-coupled approaches have been proposed recently such as [44], where image classification is carried out by a neural network in combination with reasoning from text data for concept learning at a higher level of abstraction than what is normally possible with pixel data alone. The proliferation of such approaches has prompted Henry Kautz to propose a taxonomy for neurosymbolic AI in [34] (also discussed in [17]), including recent work combining neural networks with graphical models and graph neural networks [4, 40, 58], statistical relational learning [21, 55], and even verification of neural multi-agent systems [2, 8].

最近提出了各种松散耦合的方法，例如 [44]，其中图像分类通过神经网络与来自文本数据的推理相结合，进行概念学习，抽象级别高于仅使用像素数据 alone。这类方法的激增促使 Henry Kautz 在 [34]（也在 [17] 中讨论）中提出了一种神经符号 AI 的分类法，包括最近将神经网络与图模型和图神经网络 [4, 40, 58] 结合、统计关系学习 [21, 55] 以及甚至验证神经多代理系统 [2, 8] 的工作。

# 6. Conclusions and Future Work

# 6. 结论与未来工作

In this paper, we have specified the theory and exemplified the reach of Logic Tensor Networks as a model and system for neurosymbolic AI. LTN is capable of combining approximate reasoning and deep learning, knowledge and data.

在本文中，我们详细说明了逻辑张量网络作为神经符号 AI 模型和系统的理论，并示例了其应用范围。LTN 能够结合近似推理和深度学习、知识和数据。

For ML practitioners, learning in LTN (see Section 3.2) can be understood as optimizing under first-order logic constraints relaxed into a loss function. For logic practitioners, learning is similar to inductive inference: given a theory, learning makes generalizations from specific observations obtained from data. Compared to other neuro-symbolic architectures (see Section 5), the LTN framework has useful properties for gradient-based optimization (see Section 2.4) and a syntax that supports many traditional ML tasks and their inductive biases (see Section 4), all while remaining computationally efficient (see Table 1).

对于机器学习从业者来说，LTN（见3.2节）中的学习可以理解为在将一阶逻辑约束松弛为损失函数下的优化。对于逻辑学从业者，学习类似于归纳推理：给定一个理论，学习从数据中获取的特定观察中进行泛化。与其他神经符号架构（见5节）相比，LTN框架在基于梯度的优化（见2.4节）中具有有用的属性，并具有支持许多传统机器学习任务及其归纳偏置（见4节）的语法，同时保持计算效率（见表1）。

Section 3.4 discussed reasoning in LTN. Reasoning is normally under-specified within neural networks. Logical reasoning is the task of proving if some knowledge follows from the facts which are currently known. It is traditionally achieved semantically using model theory or syntactically via a proof system. The current LTN framework approaches reasoning semantically, although it should be possible to use LTN and querying alongside a proof system. When reasoning by refutation in LTN, to find out if a statement is a logical consequence of given data and knowledge-base , a proof by refutation attempts to find a semantic counterexample where and are satisfied. If the search fails then is assumed to hold. This approach is efficient in LTN when we allow for a direct search to find counterexamples via gradient-descent optimization. It is assumed that , the statement to prove or disprove, is known. Future work could explore automatically inducing which statement to consider, possibly using syntactical reasoning in the process.

3.4节讨论了在LTN中的推理。在神经网络中，推理通常是未充分指定的。逻辑推理是证明某些知识是否遵循当前已知事实的任务。传统上，它是通过模型理论或证明系统的句法来实现的。当前的LTN框架从语义上接近推理，尽管应该可以使用LTN和查询与证明系统配合使用。在LTN中进行反驳推理时，为了找出一个陈述 是否是给定数据和相关知识库 的逻辑后果，反驳证明试图找到一个语义反例，其中 和 都得到满足。如果搜索失败，则假定 是成立的。当我们在梯度下降优化的直接搜索中允许寻找反例时，这种方法在LTN中是高效的。假定 ，即要证明或反驳的陈述是已知的。未来的工作可能会探索自动推导要考虑的陈述 ，可能在过程中使用句法推理。

The paper formalizes Real Logic, the language supporting LTN. The semantics of Real Logic are close to the semantics of Fuzzy FOL with the following major differences: 1) Real Logic domains are typed and restricted to real numbers and real-valued tensors, 2) Real Logic variables are sequences of fixed length, whereas FOL variables are a placeholder for any individual in a domain, 3) Real Logic relations relations are interpreted as mathematical functions, whereas Fuzzy Logic relations are interpreted as fuzzy set membership functions. Concerning the semantics of connectives and quantifiers, some LTN implementations correspond to semantics for t-norm fuzzy logic, but not all. For example, the conjunction operator in stable product semantics is not a t-norm, as pointed out at the end of Section 2.4

本文形式化了实数逻辑，这是一种支持LTN的语言。实数逻辑的语义与模糊一阶逻辑的语义相近，但有以下主要区别：1) 实数逻辑的域是类型化的，并且限制为实数和实值张量，2) 实数逻辑中的变量是固定长度的序列，而一阶逻辑中的变量是域中任何个体的占位符，3) 实数逻辑中的关系被解释为数学函数，而模糊逻辑中的关系被解释为模糊集合的成员函数。关于连接词和量词的语义，一些LTN的实现对应于t-范数模糊逻辑的语义，但并非全部。例如，稳定积语义中的合取运算符不是一个t-范数，这一点在2.4节末尾已经指出。

Integrative neural-symbolic approaches are known for either seeking to bring neurons into a symbolic system (neurons into symbols) [41] or to bring symbols into a neural network (symbols into neurons) [60]. LTN adopts the latter approach but maintaining a close link between the symbols and their grounding into the neural network. The discussion around these two options - neurons into symbols vs. symbols into neurons - is likely to take center stage in the debate around neurosymbolic AI in the next decade. LTN and related approaches are well placed to play an important role in this debate by offering a rich logical language tightly coupled with an efficient distributed implementation into TensorFlow computational graphs.

整合的神经符号方法已知要么试图将神经元引入符号系统（神经元到符号）[41]，要么将符号引入神经网络（符号到神经元）[60]。LTN采用后一种方法，但仍然保持符号与它们在神经网络中的具体化之间的紧密联系。关于这两种选择——神经元到符号与符号到神经元——的讨论很可能会在接下来十年的神经符号AI辩论中占据中心舞台。LTN及相关方法通过提供一种丰富的逻辑语言，与高效的分布式实现紧密耦合到TensorFlow计算图中，有望在这一辩论中发挥重要作用。

The close connection between first-order logic and its implementation in LTN makes LTN very suitable as a model for the neural-symbolic cycle [27, 29], which seeks to translate between neural and symbolic representations. Such translations can take place at the level of the structure of a neural network, given a symbolic language [27], or at the level of the loss functions, as done by LTN and related approaches [13,45,46]. LTN opens up a number of promising avenues for further research:

一阶逻辑与其在LTN中的实现之间的紧密联系，使得LTN非常适合作为神经符号循环的模型[27, 29]，该循环旨在在神经和符号表示之间进行转换。这种转换可以在神经网络的结构层面上进行，给定一个符号语言[27]，或者在损失函数的层面上进行，正如LTN和相关方法所做的那样[13,45,46]。LTN为未来的进一步研究开辟了多条充满希望的道路：

Firstly, a continual learning approach might allow one to start with very little knowledge, build up and validate knowledge over time by querying the LTN network. Translations to and from neural and symbolic representations will enable reasoning also to take place at the symbolic level (e.g. alongside a proof system), as proposed recently in [70] with the goal of improving fairness of the network model.

首先，持续学习的方法可能允许我们从非常有限的知识开始，通过询问LTN网络，随着时间的推移建立和验证知识。神经和符号表示之间的转换将使得推理也可以在符号层面上进行（例如，与证明系统并行），正如最近在[70]中提出的那样，目的是提高网络模型的公平性。

Secondly, LTN should be compared in large-scale practical use cases with other recent efforts to add structure to neural networks such as the neuro-symbolic concept learner [44] and high-level capsules which were used recently to learn the part-of relation [38], similarly to how LTN was used for semantic image interpretation in [19].

其次，LTN应该在大规模的实际应用案例中与其他近期尝试向神经网络添加结构的努力进行比较，例如神经符号概念学习者[44]和高层次胶囊网络，后者最近被用于学习部分关系[38]，类似于LTN在[19]中用于语义图像解释的方式。

Finally, LTN should also be compared with Tensor Product Representations, e.g. [59], which show that state-of-the-art recurrent neural networks may fail at simple question-answering tasks, despite achieving very high accuracy. Efforts in the area of transfer learning, mostly in computer vision, which seek to model systematicity could also be considered a benchmark [5]. Experiments using fewer data and therefore lower energy consumption, out-of-distribution extrapolation, and knowledge-based transfer are all potentially suitable areas of application for LTN as a framework for neurosymbolic AI based on learning from data and compositional knowledge.

最后，LTN还应该与张量积表示进行比较，例如[59]，它们表明即使是最先进的循环神经网络也可能在简单的问题回答任务上失败，尽管它们达到了非常高的准确度。在迁移学习领域的努力，主要在计算机视觉领域，旨在建模系统性，也可以被视为一个基准[5]。使用更少数据（因此能耗更低）、分布外推和基于知识的迁移都是LTN作为基于数据学习和组合知识学习的神经符号AI框架的潜在应用领域。

# Acknowledgement

# 致谢

We would like to thank Benedikt Wagner for his comments and a number of productive discussions on continual learning, knowledge extraction and reasoning in LTNs.

我们非常感谢Benedikt Wagner提出的意见以及关于持续学习、知识提取和LTN中推理的一系列富有成效的讨论。

# References

# 参考文献

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Good-fellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Ku-nal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from ten-sorflow.org.

[2] Michael Akintunde, Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. Verifying strategic abilities of neural multi-agent systems. In Proceedings of 17th International Conference on Principles of Knowledge Representation and Reasoning, KR2020, Rhodes, Greece, September 2020.

[3] Samy Badreddine and Michael Spranger. Injecting Prior Knowledge for Transfer Learning into Reinforcement Learning Algorithms using Logic Tensor Networks. arXiv:1906.06576 [cs, stat], June 2019. arXiv: 1906.06576.

[4] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16, pages 4509-4517, USA, 2016. Curran Associates Inc.

[5] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sebastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. In International Conference on Learning Representations, 2020.

[6] Federico Bianchi and Pascal Hitzler. On the capabilities of logic tensor networks for deductive reasoning. In Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019) Stanford University, Palo Alto, California, USA, March 25-27, 2019., Stanford University, Palo Alto, California, USA, March 25-27, 2019., 2019.

[7] Federico Bianchi, Matteo Palmonari, Pascal Hitzler, and Luciano Serafini. Complementing logical reasoning with sub-symbolic commonsense. In International Joint Conference on Rules and Reasoning, pages 161-170. Springer, 2019.

[8] Rafael Borges, Artur d’Avila Garcez, and Luís Lamb. Learning and representing temporal knowledge in recurrent networks. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, 22:2409-21, 122011.

[9] Liber Běhounek, Petr Cintula, and Petr Hájek. Introduction to mathematical fuzzy logic. In Petr Cintula, Petr Hájek, and Carles Noguera, editors, Handbook of Mathematical Fuzzy Logic, Volume 1, volume 37 of Studies in Logic, Mathematical Logic and Foundations, pages 1-102. College Publications, 2011.

[10] N. A. Campbell and R. J. Mahon. A multivariate study of variation in two species of rock crab of the genus Leptograpsus. Australian Journal of Zoology, 22(3):417-425, 1974. Publisher: CSIRO PUBLISHING.

[11] Andres Campero, Aldo Pareja, Tim Klinger, Josh Tenenbaum, and Sebastian Riedel. Logical rule induction and theory learning using neural theorem proving. CoRR, abs/1809.02193, 2018.

[12] Benhui Chen, Xuefen Hong, Lihua Duan, and Jinglu Hu. Improving multi-label classification performance by label constraints. In The 2013 International Joint Conference on Neural Networks (IJCNN), pages 1-5. IEEE, 2013.

[13] William W. Cohen, Fan Yang, and Kathryn Mazaitis. Tensorlog: A probabilistic database implemented using deep-learning infrastructure. J. Artif. Intell. Res., 67:285-325, 2020.

[14] W.-Z. Dai, Q. Xu, Y. Yu, and Z.-H. Zhou. Bridging machine learning and logical reasoning by abductive learning. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, NeurIPS’19, USA, 2019. Curran Associates Inc.

[15] Alessandro Daniele and Luciano Serafini. Knowledge enhanced neural networks. In Pacific Rim International Conference on Artificial Intelligence, pages 542-554. Springer, 2019.

[16] Artur d’Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. FLAP, 6(4):611-632, 2019.

[17] Artur d’Avila Garcez and Luis C. Lamb. Neurosymbolic AI: The 3rd wave, 2020.

[18] Ivan Donadello and Luciano Serafini. Compensating supervision incompleteness with prior knowledge in semantic image interpretation. In 2019 International Joint Conference on Neural Networks (IJCNN), pages 1-8. IEEE, 2019.

[19] Ivan Donadello, Luciano Serafini, and Artur d’Avila Garcez. Logic tensor networks for semantic image interpretation. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pages 1596-1602, 2017.

[20] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

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

[22] Ronald Fagin, Ryan Riegel, and Alexander Gray. Foundations of reasoning with uncertainty via real-valued logics, 2020.

[23] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. D12: Training and querying neural networks with logic. In International Conference on Machine Learning, pages 1931-1941, 2019.

[24] Manoel Franca, Gerson Zaverucha, and Artur d’Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. Machine Learning, 94:81- 104,012014.

[25] Dov M. Gabbay and John Woods, editors. The Many Valued and Nonmonotonic Turn in Logic, volume 8 of Handbook of the History of Logic. Elsevier, 2007.

[26] Artur d’Avila Garcez, Dov M. Gabbay, and Krysia B. Broda. Neural-Symbolic Learning System: Foundations and Applications. Springer-Verlag, Berlin, Heidelberg, 2002.

[27] Artur d’Avila Garcez, Lus C. Lamb, and Dov M. Gabbay. Neural-Symbolic Cognitive Reasoning. Springer Publishing Company, Incorporated, 1 edition, 2008.

[28] Petr Hajek. Metamathematics of Fuzzy Logic. Kluwer Academic Publishers, 1998.

[29] Barbara Hammer and Pascal Hitzler, editors. Perspectives of Neural-Symbolic Integration, volume 77 of Studies in Computational Intelligence. Springer, 2007.

[30] Stevan Harnad. The symbol grounding problem. Physica D: Nonlinear Phenomena, 42(1-3):335- 346, 1990.

[31] Patrick Hohenecker and Thomas Lukasiewicz. Ontology reasoning with deep neural networks. Journal of Artificial Intelligence Research, 68:503-540, 2020.

[32] Steffen Hölldobler and Franz J. Kurfess. CHCL - A connectionist infernce system. In Bertram Fronhöfer and Graham Wrightson, editors, Parallelization in Inference Systems, International Workshop, Dagstuhl Castle, Germany, December 17-18, 1990, Proceedings, volume 590 of Lecture Notes in Computer Science, pages 318-342. Springer, 1990.

[33] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2410-2420, Berlin, Germany, August 2016. Association for Computational Linguistics.

[34] Henry Kautz. The Third AI Summer, AAAI Robert S. Engelmore Memorial Lecture, Thirty-fourth AAAI Conference on Artificial Intelligence, New York, NY, February 10, 2020.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs], January 2017. arXiv: 1412.6980.

[36] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR), July 2014.

[37] Erich Peter Klement, Radko Mesiar, and Endre Pap. Triangular Norms, volume 8 of Trends in Logic. Springer Netherlands, Dordrecht, 2000.

[38] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoen-coders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 15512-15522. Curran Associates, Inc., 2019.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS’12, page 1097-1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[40] Luís C. Lamb, Artur d’Avila Garcez, Marco Gori, Marcelo O. R. Prates, Pedro H. C. Avelar, and Moshe Y. Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. In Christian Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020 [scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic], pages 4877-4884. ijcai.org, 2020.

[41] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, NeurIPS’18, pages 3753-3763, USA, 2018. Curran Associates Inc.

[42] Francesco Manigrasso, Filomeno Davide Miro, Lia Morra, and Fabrizio Lamberti. Faster-LTN: a neuro-symbolic, end-to-end object detection architecture. arXiv:2107.01877 [cs], July 2021.

[43] Vasco Manquinho, Joao Marques-Silva, and Jordi Planes. Algorithms for weighted boolean optimization. In International conference on theory and applications of satisfiability testing, pages 495-508. Springer, 2009.

[44] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. CoRR, abs/1904.12584, 2019.

[45] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Constraint-based visual generation. In Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis, editors, Artificial Neural Networks and Machine Learning - ICANN 2019: Image Processing - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part III, volume 11729 of Lecture Notes in Computer Science, pages 565-577. Springer, 2019.

[46] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Integrating learning and reasoning with deep logic models. In Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Wiirzburg, Germany, September 16-20, 2019, Proceedings, Part II, volume 11907 of Lecture Notes in Computer Science, pages 517-532. Springer, 2019.

[47] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Lyrics: A general interface layer to integrate logic inference and deep learning. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 283-298. Springer, 2019.

[48] Giuseppe Marra and Ondřej Kuželka. Neural markov logic networks. arXiv preprint arXiv:1905.13462, 2019.

[49] Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rock-täschel. Learning reasoning strategies in end-to-end differentiable proving, 2020.

[50] Stephen H. Muggleton, Dianhuan Lin, Niels Pahlavi, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning: Application to grammatical inference. Mach. Learn., 94(1):25-49, January 2014.

[51] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11):559-572, 1901.

[52] Gadi Pinkas. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. Artif. Intell., 77(2):203-247, 1995.

[53] Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In Advances in Neural Information Processing Systems, pages 7712-7722, 2019.

[54] Luc De Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence, 2020.

[55] Matthew Richardson and Pedro Domingos. Markov logic networks. Mach. Learn., 62(1-2):107- 136, February 2006.

[56] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Is-mail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Sha-jith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. Logical Neural Networks. arXiv:2006.13155 [cs], June 2020. arXiv: 2006.13155.

[57] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In Advances in Neural Information Processing Systems, pages 3788-3800, 2017.

[58] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfar-dini. The graph neural network model. Trans. Neur. Netw., 20(1):61-80, January 2009.

[59] Imanol Schlag and Jürgen Schmidhuber. Learning to reason with third-order tensor products. CoRR, abs/1811.12143, 2018.

[60] Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. Enhancing the transformer with explicit relational encoding for math problem solving. CoRR, abs/1910.06611, 2019.

[61] 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.

[62] Luciano Serafini and Artur d’Avila Garcez. Learning and reasoning with logic tensor networks. In Conference of the Italian Association for Artificial Intelligence, pages 334–348. Springer, 2016.

[63] Lokendra Shastri. Advances in SHRUTI-A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. Appl. Intell., 11(1):79-108, 1999.

[64] Yun Shi. A deep study of fuzzy implications. PhD thesis, Ghent University, 2009.

[65] Paul Smolensky and Géraldine Legendre. The Harmonic Mind: From Neural Computation to Optimality-Theoretic GrammarVolume I: Cognitive Architecture (Bradford Books). The MIT Press, 2006.

[66] 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.

[67] Emile van Krieken, Erman Acar, and Frank van Harmelen. Semi-Supervised Learning using Differentiable Reasoning. arXiv:1908.04700 [cs], August 2019. arXiv: 1908.04700.

[68] Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing Differentiable Fuzzy Implications. In Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, pages 893-903, 92020.

[69] Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing Differentiable Fuzzy Logic Operators. arXiv:2002.06100 [cs], February 2020. arXiv: 2002.06100.

[70] Benedikt Wagner and Artur d’Avila Garcez. Neural-Symbolic Integration for Fairness in AI. Proceedings of the AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering 2021, page 14, 2021.

[71] Po-Wei Wang, Priya L Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. arXiv preprint arXiv:1905.12149, 2019.

[72] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In International Conference on Machine Learning, pages 5502-5511. PMLR, July 2018. ISSN: 2640-3498.

# Appendix A. Implementation Details

# 附录A. 实现细节

The LTN library is implemented in Tensorflow 2 [1] and is available from GitHub . Every logical operator is grounded using Tensorflow primitives. The LTN code implements directly a Tensorflow graph. Due to Tensorflow built-in optimization, LTN is relatively efficient while providing the expressive power of FOL.

LTN库在Tensorflow 2 [1] 中实现，并且可以从GitHub 获取。每个逻辑运算符都使用Tensorflow原语进行接地。LTN代码直接实现了一个Tensorflow图。由于Tensorflow内置的优化，LTN在提供FOL表现力的同时，相对高效。

Table A.3 shows an overview of the network architectures used to obtain the results of the examples in Section 4 The LTN repository includes the code for these examples. Except if explicitly mentioned otherwise, the reported results are averaged over 10 runs using a confidence interval. Every example uses a stable real product configuration to approximate Real Logic operators, and the Adam optimizer [35] with a learning rate of 0.001 to train the parameters. are usually used with some additional layer(s) to ground symbols. For instance, in experiment 4.2 in softmax , the softmax layer normalizes the raw predictions of MLP to probabilities in , and the multiplication with the one-hot label selects the probability for one given class.

表A.3展示了用于获得第4节中示例结果的网络架构概览。LTN仓库包含了这些示例的代码。除非明确提及，报告的结果是使用 置信区间，在10次运行中平均得到的。每个示例都使用稳定的实时产品配置来近似实数逻辑运算符，并使用学习率为0.001的Adam优化器 [35] 来训练参数。通常与一些额外的层一起使用来接地符号。例如，在 softmax 的实验4.2中，softmax层将MLP的原始预测归一化为 中的概率，乘以独热标签 选择给定类的概率。

| Task | Network | Architecture |
| --- | --- | --- |
| 4.1 4.2 | MLP | Dense(16)\*, Dense(16)\*, Dense(1) |
| MLP | Dense \*, \*, , |
|  | Dense(8)\*, Dropout(0.2), Dense(1) |
| 4.3 4.4 | MLP |  |
| CNN | MNISTConv, Dense(84)\*, Dense(10) |
| baseline – SD |  |
| baseline – MD | MNISTConv , Dense , Dense , Softmax |
| 4.5 4.6 4.7 | MLP | Dense(8)\*, Dense(8)\*, Dense(1) |
| MLP | Dense(16)\*, Dense(16)\*, Dense(16)\*, Dense(1) |
| MLP\_S | Dense(8)\*, Dense(8)\*, Dense(1) |
| MLP\_F | Dense(8)\*, Dense(8)\*, Dense(1) |
| MLP\_C | Dense(8)\*, Dense(8)\*, Dense(1) |

\* : layer ends with an elu activation

\*：层以elu激活结束

Dense : regular fully-connected layer of units

Dense ：具有 单元的常规全连接层

Dropout : dropout layer with rate

Dropout ：dropout层，比率为

: 2D convolution layer with filters and a kernel of size

：具有 过滤器和大小为 的内核的2D卷积层

: max pooling operation with a pooling window

：具有 池化窗口的最大池化操作

MNISTConv : Conv

MNISTConv：卷积

Table A.3: Overview of the neural network architectures used in each example. Notice that in the examples, the networks

表A.3：每个示例中使用的神经网络架构概览。注意，在示例中，网络架构

[[23]](#footnote-265)

# Appendix B. Fuzzy Operators and Properties

# 附录 B. 模糊运算符和性质

This appendix presents the most common operators used in fuzzy logic literature and some noteworthy properties [28, 37, 64, 69].

本附录介绍了在模糊逻辑文献中最常见的运算符以及一些值得注意的性质 [28, 37, 64, 69]。

Appendix B.1. Negation

附录 B.1. 否定

Definition 7. A negation is a function that at least satisfies:

定义 7. 否定是一种函数 ，它至少满足：

N1. Boundary conditions: and ,

N1. 边界条件： 和 ，

N2. Monotonically decreasing: .

N2. 单调递减： 。

Moreover, a negation is said to be strict if is continuous and strictly decreasing. A negation is said to be strong if .

此外，如果 是连续且严格递减的，那么该否定被认为是严格的。如果 ，那么该否定被认为是强的。

We commonly use the standard strict and strong negation .

我们通常使用标准的严格和强否定 。

Appendix B.2. Conjunction

附录 B.2. 合取

Definition 8. A conjuction is a function that at least satisfies:

定义 8. 合取是一种函数 ，它至少满足：

C1. boundary conditions: and ,

C1. 边界条件： 和 ，

C2. monotonically increasing: , if , then and .

C2. 单调递增：如果 ，那么 时， 和 。

In fuzzy logic, t-norms are widely used to model conjunction operators.

在模糊逻辑中，t-范数被广泛用于建模合取运算符。

Definition 9. A t-norm (triangular norm) is a function that at least satisifies:

定义 9. t-范数（三角范数）是一种函数 ，它至少满足：

T1. boundary conditions: ,

T1. 边界条件： ，

T2. monotonically increasing,

T2. 单调递增，

T3. commutative,

T3. 交换律，

T4. associative.

T4. 结合律。

Example 4. Three commonly used t-norms are:

示例 4. 常用的三种 t-范数是：

Appendix B.3. Disjunction附录B.3. 析取

| Name |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goedel |  |  | if otherwise |  |
| Goguen/Product |  |  |  |  |
| Lukasiewicz |  |  |  |  |

Table B.4: Common Symmetric Configurations

表B.4: 常见对称配置

Definition 10. A disjunction is a function that at least satisfies:

定义10. 析取是一个函数 ，至少满足：

D1. boundary conditions: and ,

D1. 边界条件： 和 ，

D2. monotonically increasing: , if , then and .

D2. 单调递增： ，如果 ，那么 和 。

Disjunctions in fuzzy logic are often modeled with t-conorms.

模糊逻辑中的析取通常用t-范数建模。

Definition 11. A t-conorm (triangular conorm) is a function that at least satisfies:

定义11. t-范数（三角范数）是一个函数 ，至少满足：

S1. boundary conditions: ,

S1. 边界条件： ，

S2. monotonically increasing,

S2. 单调递增，

S3. commutative,

S3. 交换律，

S4. associative.

S4. 结合律。

Example 5. Three commonly used t-conorms are:

示例5. 常用的三种t-范数是：

Note that the only distributive pair of t-norm and t-conorm is and - that is, distributivity of the t-norm over the t-conorm, and inversely.

注意，唯一的分配律对是 和 - 即，t-范数对t-范数的分配性，反之亦然。

Definition 12. The -dual t-conorm of a t-norm w.r.t. a strict fuzzy negation is defined as:

定义12. 对于严格模糊否定 ，t-范数 的 -对偶t-范数 定义为：

If is a strong negation, we also get:

如果 是强否定，我们还得到：

Appendix B.4. Implication

附录B.4. 蕴含

Definition 13. An implication is a function that at least satisfies:

定义13. 蕴含是一个函数 ，至少满足：

I1. boundary Conditions: and

I1. 边界条件： 和

Definition 14. There are two main classes of implications generated from the fuzzy logic operators for negation, conjunction and disjunction.

定义14. 有两类主要蕴含由模糊逻辑的否定、合取和析取运算符生成。

S-Implications Strong implications are defined using (material implication).

S-蕴含 强蕴含使用 （材料蕴含）定义。

R-Implications Residuated implications are defined using . One way of understanding this approach is a generalization of modus ponens: the consequent is at least as true as the (fuzzy) conjunction of the antecedent and the implication.

R-蕴涵 蕴涵残差是通过 定义的。理解这种方法的一种方式是模态推理的泛化：结论至少与前提和蕴涵的（模糊）合取一样真实。

Example 6. Popular fuzzy implications and their classes are presented in Table B. 5

示例 6. 表 B.5 中展示了流行的模糊蕴涵及其类别。

| Name |  | S-Implication | R-Implication |
| --- | --- | --- | --- |
| Kleene-Dienes |  |  | - |
| Goedel |  |  |  |
| Reichenbach |  |  | - |
| Goguen |  | - |  |
| Lukasiewicz |  |  |  |

Table B.5: Popular fuzzy implications and their classes. Strong implications (S-Implications) are defined using a fuzzy negation and fuzzy disjunction. Residuated implications (R-Implications) are defined using a fuzzy conjunction.

表 B.5：流行的模糊蕴涵及其类别。强蕴涵（S-蕴涵）是通过模糊否定和模糊析取定义的。蕴涵残差（R-蕴涵）是通过模糊合取定义的。

# Appendix B.5. Aggregation

# 附录 B.5. 聚合

Definition 15. An aggregation operator is a function that at least satisfies:

定义 15. 聚合算子是一个至少满足以下条件的函数 ：

A1. whenever for all ,

A1. 当 对所有 成立时，

A2. forall ,

A2. 对所有 成立，

A3. and .

A3. 和 。

Example 7. Candidates for universal quantification can be obtained using t-norms with and :

示例 7. 使用 t-范数与 和 ，可以获得全称量词 的候选者：

Similarly, candidates for existential quantification can be obtained using s-norms with and :

类似地，使用 s-范数与 和 ，可以获得存在量词 的候选者：

|  |  | |  | |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Commutativity of , |  |  |  |  |  |
| Associativity of , |  |  |  |  |  |
| Distributivity of over |  |  |  |  |  |
| Distributivity of over |  |  |  |  |  |
| Distrib. of over |  |  |  |  |  |
| Double negation |  |  |  |  |  |
| Law of excluded middle |  |  |  |  |  |
| Law of non contradiction |  |  |  |  |  |
| De Morgan’s laws |  |  |  |  |  |
| Material Implication |  |  |  |  |  |
| Contraposition |  |  |  |  |  |

Table B.6: Common properties for different configurations

表 B.6：不同配置的常见属性

Following are other common aggregators:

下面是其他常见的聚合器：

Where is the generalized mean, and can be understood as the generalized mean measured w.r.t. the errors. That is, measures the power of the deviation of each value from the ground truth 1. A few particular values of yield special cases of aggregators. Notably:

其中 是广义均值， 可以理解为相对于错误的广义均值。也就是说， 衡量每个值与真实值 1 的偏差的力度。一些特定的 值会产生聚合器的特殊情况。特别地：

These "smooth" min (resp. max) approximators are good candidates for (resp. ) in a fuzzy context. The value of leaves more or less room for outliers depending on the use case and its needs. Note that and are related in the same way that and are related using the definition , where would be approximated by the standard negation.

这些“平滑”的最小（相应地，最大）逼近器是模糊背景下 （相应地， ）的好候选者。 的值根据用例及其需求为异常值留出了或多或少的余地。请注意， 和 与 和 使用定义 相关联的方式相关，其中 将通过标准否定来逼近。

We propose to use with to approximate and with to approximate . When , these operators resemble the norm of a vector , where . In our case, many properties of the norm can apply to (positive homogeneity, triangular inequality,...).

我们提议使用 与 来逼近 和 ，以及使用 来逼近 。当 时，这些运算符类似于向量 的 范数，其中 。在我们的情况下， 范数的许多属性可以应用于 （正齐次性，三角不等式……）。

# Appendix C. Analyzing Gradients of Generalized Mean Aggregators

# 附录 C. 分析广义均值聚合器的梯度

[69] show that some operators used in Fuzzy Logics are unsuitable for use in a differentiable learning setting. Three types of gradient problems commonly arise in fuzzy logic operators.

[69] 显示，在模糊逻辑中使用的某些运算符不适合在可微学习环境中使用。在模糊逻辑运算符中通常会出现三种类型的梯度问题。

Single-Passing The derivatives of some operators are non-null for only one argument. The gradients propagate to only one input at a time.

单向传递 某些运算符的导数仅对一个参数非零。梯度一次仅传播到一个输入。

Vanishing Gradients The gradients vanish on some part of the domain. The learning does not update inputs that are in the vanishing domain.

消失梯度 梯度在域的某部分消失。学习不会更新消失域中的输入。

Exploding Gradients Large error gradients accumulate and result in unstable updates.

爆炸梯度 大的误差梯度累积并导致更新不稳定。

Tables C. 7 and C. 8 summarize their conclusions for the most common operators. Also, we underline here exploding gradients issues that arise experimentally in and , which are not in the original report. Given the truth values of propositions in :

表 C.7 和 C.8 总结了他们针对最常见运算符的结论。此外，我们在此强调在 和 中实验上出现的爆炸梯度问题，这些问题未在原始报告中提及。给定 中 命题 的真值：

The partial derivatives are .

部分导数是 。

When , the operator weights more for inputs with a higher true value -i.e. their partial derivative is also higher - and suits for existential quantification. When , the operator weights more for inputs with a lower true value and suits for universal quantification.

当 时，运算符更倾向于为具有较高真实值的输入分配更高的权重——即它们的偏导数也更高——并且适用于存在量化。当 时，运算符更倾向于为具有较低真实值的输入分配更高的权重，并适用于全称量化。

Exploding Gradients When , if , then and the gradients explode. When , if , then .

爆炸梯度 当 时，如果 ，那么 并且梯度爆炸。当 时，如果 ，那么 。

The partial derivatives are . When , the operator weights more for inputs with a lower true value -i.e. their partial derivative is also higher - and suits for universal quantification. When , the operator weights more for inputs with a higher true value and suits for existential quantification.

部分导数是 。当 时，运算符更倾向于为具有较低真实值的输入分配更高的权重——即它们的偏导数也更高——并适用于全称量化。当 时，运算符更倾向于为具有较高真实值的输入分配更高的权重，并适用于存在量化。

Exploding Gradients

爆炸梯度

When , if , then and the gradients

当 时，如果 ，那么 并且梯度

explode. When , if , then .

爆炸。当 时，如果 ，那么 。

We propose the following stable product configuration that does not have any of the aforemen-

我们提出了以下稳定的产物配置，该配置不具有前面提到的任何梯度问题。

|  | Single-Passing | Vanishing | Exploding |
| --- | --- | --- | --- |
| Goedel (mininum) | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Goguen (product) | | | |
| , |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Lukasiewicz | | | |
|  |  |  |  |
|  |  |  |  |

Table C.7: Gradient problems for some binary connectives. means that the problem only appears on an edge case.

表C.7：某些二元连接词的梯度问题。 表示问题仅出现在边界情况。

|  | Single-Passing | Vanishing | Exploding |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table C.8: Gradient problems for some aggregators. means that the problem only appears on an edge case.

表C.8：某些聚合器的梯度问题。 表示问题仅出现在边界情况。

tioned gradient problems:

梯度问题：

is the operator for negation, for conjunction, for disjunction, for implication, for existential aggregation, for universal aggregation.

是否定运算符， 是合取运算符， 是析取运算符， 是蕴涵运算符， 是存在聚合运算符， 是全称聚合运算符。

1. \*Corresponding author

   \*通讯作者

   Email addresses: badreddine. samy@gmail.com (Samy Badreddine), a. garcez@city.ac.uk (Artur d’Avila Garcez),

   电子邮件地址：badreddine.samy@gmail.com (Samy Badreddine), a.garcez@city.ac.uk (Artur d’Avila Garcez),

   serafini@fbk.eu (Luciano Serafini), michael . spranger@sony . com (Michael Spranger)

   serafini@fbk.eu (Luciano Serafini), michael.spranger@sony.com (Michael Spranger) [↑](#footnote-ref-25)
2. In the rest of the paper, we commonly use "tensor" to designate "tensor in the real field".

   在本文的其余部分，我们通常使用“张量”来指代“实数域中的张量”。 [↑](#footnote-ref-32)
3. An interpretation is an assignment of truth-values or , or in the case of Real Logic a value in [0,1], to a formula. A model is an interpretation that maps a formula to true

   解释是将真值 或 ，或者在实数逻辑中 [0,1] 区间内的一个值，赋给一个公式。模型是一种解释，它将公式映射为真。

   A tensor of rank 0 corresponds to a scalar, a tensor of rank 1 to a vector, a tensor of rank 2 to a matrix and so forth, in the usual way.

   零阶张量对应于标量，一阶张量对应于向量，二阶张量对应于矩阵，以此类推，按照通常的方式。 [↑](#footnote-ref-35)
4. We assume the usual syntactic definition of free and bound variables in FOL. A variable is free if it is not bound by a quantifier .

   我们假设FOL中自由变量和约束变量的通常句法定义。如果一个变量没有被量词 绑定，那么它是自由的。 [↑](#footnote-ref-38)
5. Notice how Diag is not simply "syntactic sugar" for creating a new variable pairs\_xy by stacking pairs of examples from and . If the groundings of and have incompatible ranks (for instance, if denotes images and denotes their labels), stacking them in a tensor (pairs\_xy) is non-trivial, requiring several reshaping operations.

   注意 Diag 不仅仅是创建新变量 pairs\_xy 的 "语法糖"，通过堆叠来自 和 的示例对。如果 和 的基有不相容的秩（例如，如果 表示图像而 表示它们的标签），在张量 （pairs\_xy）中堆叠它们并非易事，需要执行多个重塑操作。 [↑](#footnote-ref-52)
6. In some edge cases, a masking may produce an empty sequence, e.g. if for some value of , there is no value in that satisfies age , we resort to the concept of an empty semantics: returns 1 and returns 0 .

   在某些边缘情况下，掩码可能产生一个空序列，例如，如果对于某个 的值， 中没有满足年龄 的值，我们求助于空语义的概念： 返回1，而 返回0。 [↑](#footnote-ref-56)
7. We define a symmetric configuration as a set of fuzzy operators such that conjunction and disjunction are defined by a t-norm and its dual t-conorm, respectively, and the implication operator is derived from such conjunction or disjunction operators and standard negation (c.f. Appendix B for details). In [69], van Krieken et al. also analyze non-symmetric configurations and even operators that do not strictly verify fuzzy logic semantics.

   我们将对称配置定义为这样一个模糊运算符集合，其中合取和析取分别由t-范数及其对应t-余范数定义，蕴涵运算符则由这样的合取或析取运算符和标准否定派生而来（详情参见附录B）。在文献[69]中，van Krieken等人还分析了非对称配置，甚至是那些不严格验证模糊逻辑语义的运算符。 [↑](#footnote-ref-62)
8. Recall that a T-norm is a function satisfying commutativity, monotonicity, associativity and identity, that is, .

   回想一下，T-范数是一个满足交换律、单调性、结合律和恒等性的函数 ，即 。 [↑](#footnote-ref-65)
9. Notice that softmax is often used as the last layer in neural networks to turn logits into a probability distribution. However, we do not use the softmax function as such here. Instead, we use it here to enforce an exclusivity constraint on satisfiability scores.

   注意，softmax函数通常在神经网络的最后一层中使用，将logits转换为概率分布。然而，在这里我们并不以这种方式使用softmax函数。相反，我们在这里使用它来对满意度得分施加排他性约束。 [↑](#footnote-ref-70)
10. This can also be specified using a guarded quantifier where is a threshold value in .

    这也可以通过使用一个守卫量化器 来指定，其中 是 中的一个阈值。 [↑](#footnote-ref-73)
11. For simplicity, we temporarily define the notation .

    为了简单起见，我们暂时定义记法 。 [↑](#footnote-ref-88)
12. In the objective function, should satisfy before reducing because the penalty which is greater than 1 is higher than any potential reduction in which is smaller or equal to 1 .

    在目标函数中， 应该在减少 之前满足 ，因为惩罚 （大于1）高于任何可能的 （小于或等于1）的减少。

    https://github.com/logictensornetworks/logictensornetworks

    https://github.com/logictensornetworks/logictensornetworks [↑](#footnote-ref-91)
13. are, by definition in this example, the training examples with Euclidean distance to the center smaller than the threshold of 0.09 .

    在此例中，按照定义，是到中心 的欧几里得距离小于阈值0.09的训练示例。

    are, by definition, the training examples with Euclidean distance to the centre larger or equal to the threshold of 0.09 .

    按照定义，是到中心 的欧几里得距离大于或等于阈值0.09的训练示例。

    The notation means that the variable is grounded with the data (that is, ) when grounding

    该符号 表示变量 在数据 被接地时（即 ）。

    .

    。

    As usual in ML, while it is possible to compute the loss function and gradients over the entire data set, it is preferred to use mini-batches of the examples.

    如机器学习中的通常做法，虽然可以在整个数据集上计算损失函数和梯度，但更倾向于使用示例的小批量。 [↑](#footnote-ref-105)
14. [↑](#footnote-ref-120)
15. This is also known as top-1 accuracy, as proposed in [39]. Cross-entropy results could have been reported here as is common with the use of softmax, although it is worth noting that, of course, the loss function used by LTN is different.

    这也被称为top-1准确度，如在[39]中提出。可以在此报告交叉熵结果 ，这是使用softmax的常见做法，不过值得注意的是，LTN使用的损失函数当然与之不同。 [↑](#footnote-ref-130)
16. There are two possible approaches here: either each item is labeled with one multi-hot encoding or each item is labeled with several one-hot encodings. The latter approach was used in this example.

    There are two possible approaches here: either each item is labeled with one multi-hot encoding or each item is labeled with several one-hot encodings. The latter approach was used in this example. [↑](#footnote-ref-147)
17. Training should usually not focus on outliers, as optimizers would struggle to generalize and tend to get stuck in local minima. However, when querying , we wish to be more careful about the interpretation of our statement. See also 3.1.3

    训练通常不应关注异常值，因为优化器在泛化和陷入局部最小值方面可能会遇到困难。然而，在查询 时，我们希望更加小心地解释我们的陈述。参见3.1.3 。 [↑](#footnote-ref-154)
18. Notice the use of the same number of examples for each of these variables as they are supposed to match one-to-one due to the use of Diag.

    注意，由于使用了 Diag，这些变量的每个示例数量 是相同的，因为它们应该一一对应。 [↑](#footnote-ref-166)
19. https://www.kaggle.com/quantbruce/real-estate-price-prediction

    https://www.kaggle.com/quantbruce/real-estate-price-prediction [↑](#footnote-ref-177)
20. Intuitively, the smooth equality is , where is the Euclidean distance between and . It produces a 1 if the distance is zero; as the distance increases, the result decreases exponentially towards 0 . In case an exponential decrease is undesirable, one can adopt the following alternative equation: .

    直观上，平滑等式是 ，其中 是 和 之间的欧几里得距离。如果距离为零，则产生 1；随着距离的增加，结果指数减少至 0。如果指数减少不希望出现，可以采用以下替代方程： 。 [↑](#footnote-ref-194)
21. Here, learning refers to Section 3.2 which is optimizing using the satisfaction of the knowledge base as an objective.

    在这里，学习指的是第3.2节，即使用知识库的满意度作为目标进行优化。 [↑](#footnote-ref-234)
22. We use the notation .

    我们使用表示法 。 [↑](#footnote-ref-242)
23. https://github.com/logictensornetworks/logictensornetworks

    https://github.com/logictensornetworks/logictensornetworks [↑](#footnote-ref-265)