A Survey on Neural-symbolic Systems

Dongran Yu, Bo Yang, Dayou Liu and Hui Wang

Abstract—In recent years, neural systems have demonstrated superior perceptual intelligence through highly effective learning, but their reasoning capabilities remain poor. In contrast, symbolic systems have exceptional cognitive intelligence through efficient reasoning, but their learning capabilities are poor. In this case, an ideal intelligent system–a neural-symbolic system–with high perceptual and cognitive intelligence through powerful learning and reasoning capabilities gains a growing interest in the research community. Combining the fast computation ability of neural systems and the powerful expression ability of symbolic systems, neural-symbolic systems can perform effective learning and reasoning in multi-domain tasks, demonstrating concurrent perception and cognition capabilities in intelligent systems. This paper surveys the latest research in neural-symbolic systems along four dimensions: the necessity of combination, technical challenges, methods, and applications. This paper aims to help advance this emerging area of research by providing researchers with a holistic and comprehensive view, highlighting the state of art and identifying the opportunities.

Index Terms—Deep Learning, Neural Network, Symbolic, Neural-Symbolic System, Logic, Knowledge Graph, Interpretability.

1 Introduction

Perception and cognition[1] are two important processes of intelligent systems including humans. Many real-world intelligent systems such as clinical decision support systems and autonomous driving systems require both perception and cognition capabilities. Turing Award winner Yoshua Bengio pointed out in the special lecture of NIPS 2019 that deep learning needs system 1 to system 2 transformation, where system 1 represents intuitive, fast, unconscious, non-linguistic, and habitual; system 2 represents slow, logical, sequential, conscious, linguistic, algorithmic, planning, and reasoning. This comment boosted public interest in a research area, *neural-symbolic systems*, a milestone in the history of artificial intelligence (AI).

Fig. 1 provides an overview of the history of AI. In the period from the 1950s to the 1990s, the symbolism was prevailing with logical reasoning, and knowledge engineering was the mainstream AI technology. This period was followed by connectionism with neural networks and machine learning as the mainstream AI technologies. It can be seen from the figure that the whole development of artificial intelligence includes two major doctrines: symbolism (resulting in symbolic systems) and connectionism (resulting in neural systems). Symbolism studies knowledge representation, logical reasoning, and search algorithms for problem-solving. Symbolic systems include expert systems (deriving conclusions from input data based on rules from experts), constraint solvers (solving an optimization problem that must satisfy some logical conditions), and planning systems (searching for a series of actions from some initial state values to achieve a given goal). Connectionism studies induction, learning, and neural networks that can automatically acquire models of data. The neural system includes multilayer perceptron, convolution neural networks (which are good at dealing with image problems), and long short-term memory networks (which are good at dealing with time sequence questions).

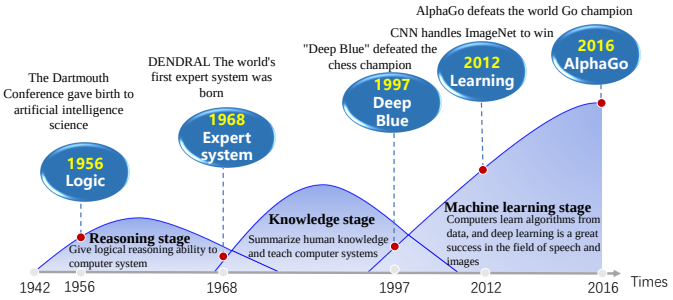


Fig. 1. The brief history of artificial intelligence.

A comparison of neural and symbolic systems in terms of their advantages and disadvantages is presented in Table 1. Symbolic systems are good at reasoning, a deductive process from general to special. The main advantages are: (1) We can apply knowledge to build models. (2) The reasoning process is transparent with interpretability. (3) Knowledge-based models are not limited to specific tasks so symbolic systems generally have strong generalization capabilities. The main disadvantages are: (1) Symbolic systems are ineffective at processing unstructured data (such as images). (2) The reasoning process is vulnerable to noisy samples (the model can not with effective reasoning ability from noisy data). (3) The inference process is slow.

Correspondingly, neural systems are mainly good at learning, an inductive process from special to general. The main advantages are: (1) Neural systems are effective at processing unstructured data. (2) The learning process is insensitive to noise (Even if the training data is noisy, the model can still learn valid information

- *B. Yang (corresponding author) and D. Liu are with the School of Computer Science and Technology, Jilin University, Changchun, Jilin 130012, China, and the Key Laboratory of Symbolic Computation and Knowledge Engineer (Jilin University), Ministry of Education, Changchun, Jilin 130012, China.*
  E-mail: ybo@jlu.edu.cn
- *D. Yu is with School of Artificial Intelligence, Jilin University, Changchun, Jilin 130012, China, and the Key Laboratory of Symbolic Computation and Knowledge Engineer (Jilin University), Ministry of Education, Changchun, Jilin 130012, China.*
  E-mail: yudran@foxmail.com
- *H. Wang is with School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast.*

1. Perception is the organization, identification, and interpretation of sensory information to represent and understand the presented information. Cognition is the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses. (Wikipedia)

and has learning ability). (3) The learning process is fast. The main disadvantages are: (1) Neural system's performance relies on large amounts of training data. (2) The learning process is black-box, so it is impossible for people to intuitively understand the process of model decision-making. (3) The learned model is limited to specific tasks, so neural systems generally have weak generalization abilities. Through the above summary and comparison, we can see that the symbolic system and the neural system are complementary.

The symbolic system makes good use of knowledge, and the neural system makes good use of data. However, people need both knowledge and data to make decisions. There is a growing interest among AI researchers in combining symbolic and neural systems to merge the benefits of both systems [1] , [2], [3], [4], [5], [6], [7], [8], [9]. Fig. 2 provides a conceptual comparison of neural-symbolic systems with neural and symbolic systems from three dimensions – efficiency, generalization, and interpretability. Firstly, in terms of model efficiency, neural-symbolic systems are fast thus facilitating reasoning over large-scale data. For example, in a large-scale knowledge graph reasoning task, due to a large number of triples, traditional symbolic methods would not be suitable as they have exponentially computational complexity in the size of the knowledge graph; however, neural-symbolic systems would be suitable as they have polynomial computational complexity from neural networks [10], [11]. Secondly, in terms of model generalization, neural-symbolic systems do not solely rely on large labeled data, and thus they have strong generalization capabilities. For example, the long tail problem with sparse data exists in the real world. A neural-symbolic system can apply background knowledge to make up for the lack of training data, which enables the model to achieve the expected effect with decent generalization ability. Thirdly, in terms of the model's interpretability, the neural-symbolic system can embody a transparent reasoning process and hence is interpretable. This is particularly useful in some applications such as medical image analysis when people need not only a decision but also the reason and the decision-making process. The neural-symbolic system has become an important approach for explainable artificial intelligence.

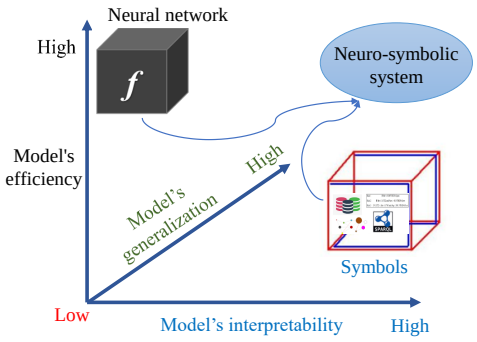


Fig. 2. The advantages of neural-symbolic systems in model efficiency, generalization and interpretability.

Given the above advantages, neural-symbolic systems have attracted soaring attention in the AI research community and have been applied to many fields, such as computer vision and natural language processing. More specifically, they have been used for visual relationship detection (VRD) [12] [13], knowledge graph reasoning [10], [11], zero-shot learning (ZSL) [14], [15], [16], few-shot learning (FSL) [17], [18] ,and semantic parsing [19], [20], etc. They have huge potentials for further development. Symbolic systems and neural systems are completely different in problem solving and representation – for example, the former adopts logical representation and the latter adopts feature representation. Therefore, the main research challenge for neural-symbolic systems is to combine these two systems together effectively.

In order to facilitate readers to have a comprehensive understanding of neural-symbolic systems, this paper surveys the research and applications of neural-symbolic systems, with particular attention to the past three years. There exist two surveys of neural-symbolic systems [4], [21]. Compared with these works, the main contributions of this paper are listed in 3 aspects. Firstly, concerning the writing style, Besold et.al. (2017) [4] analyzed the purpose and theoretical basis of neural-symbolic learning and reasoning from the perspectives of cognitive science, artificial intelligence, and psychology. Calegari et al. (2020) [21] introduces a classification of relevant research works and representative methods from the perspective of explainable artificial intelligence (XAI). However, after a systematic investigation, this paper comprehensively surveys neural-symbolic systems, covering the necessity of combining the neural system with the symbolic system, the difficulties, the solutions of difficulties in existing work, and the applications. The paper also analyzes the development trend of neural-symbolic systems. Secondly, from the coverage range perspective, Calegari et al. (2020) focus on reviewing relevant research works on neural-symbolic systems published before 2018, and Besold et. al. (2017) focuses on reviewing earlier research works in this field published before 2017. However, this paper highlights the cutting-edge developments in neural-symbolic systems such as neuro-variational inference [10], [11] and neural Markov logic network [22]. Thirdly, in terms of the representation of symbols, the references [4], [21] only paid attention to logic symbols, and this paper will summarize and analyze two symbols – logic and knowledge graph.

The rest of this survey is organized as follows. In Section 1, we provide an overview of neural-symbolic systems. Section 2 introduces the types and representations of the symbol. In Section 3, we categorize the different methods of neural-symbolic systems. We summarize the main applications of the neural-symbolic system in Section 4. Section 5 discusses the problems and challenges faced in the neural-symbolic system, and Section 6 concludes this survey.

2 Overview of Neural-Symbolic Systems

At present, the organic combination of learning and reasoning has become a key challenge in the field of artificial intelligence and machine learning. Models with both learning and reasoning capabilities provide accurate prediction results based on input data, and provide an interpretable decision-making process for the results. The neural-symbolic system is a hybrid model that applys the high-efficiency of connectionism (the neural system) and the generalization of symbolism (the symbolic system) to effectively integrate learning and reasoning. The neural-symbolic system has achieved excellent performance in different fields of logical reasoning, such as from classic propositional logic to probabilistic logic, deductive logic and inductive logic [23], [24], [25], [26], [27], [28]. Fig. 3 is the integrated schematic diagram of the two systems. In the figure: The green rectangle represents the symbolic system, which is close to the ground truth from the perspective of logical reasoning. The information processing

TABLE 1
Advantages and disadvantages of neural system and symbolic system

Systems	Advantages	Disadvantages
Symbolic system (Good at deductive reasoning)	Strong generalization ability Interpretability Knowledge driven	Not good at handling unstructured data Weak robustness Slowly reasoning
Neural system (Good at inductive learning)	Good at handling unstructured data Strong robustness Fastly learning	Weak generalization ability No interpretability Requiring a large amount of label data

unit is the symbols. The input is usually structured data (such as databases, logic rules, and knowledge graphs, etc.). After training, it obtains the solution space of search algorithm for a specific task and outputs the reasoning results; The blue rectangle represents the neural system. It is close to the true value from the perspective of data relevance. The information processing unit is the feature vector of the entity. The input is usually unstructured data (such as images, videos, texts, etc.). After training, the model learns a neural network model for a specific task and outputs predicted results; The outer blue box represents the integration of the two systems, which has both advantages of the symbolic system and the neural system. The input is usually a mixture of structured data and unstructured data. After fusion training, the model aquires both perception and cognition.

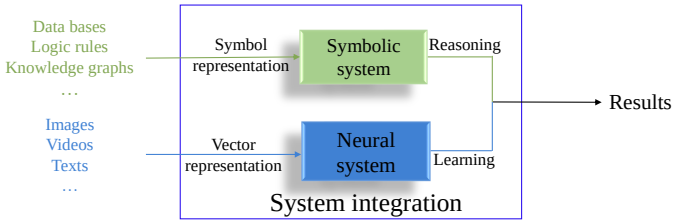


Fig. 3. Integration of neural and symbolic systems. The symbolic system generally use domain knowledge representation (such as first-order logic and knowledge graph, etc.) to formalize the problem to be solved, through reasoning tools (such as logic programs, etc.) realize automatic machine reasoning; The goal of the neural system is to learn a model from training samples so that the model can be generalized to unseen data.

2.1 Taxonomy of neural-symbolic systems methods

From the perspective of combination mode, this paper classifies the existing related work into two categories: heavy-reasoning light-learning and heavy-learning light-reasoning. The first category is based on the symbolic system with a supplementary neural system. The input data is represented as symbols, and the related technology of the neural system is introduced into the symbolic system during the computing process [28], [29], [30], [31], [24], [32], [33], which makes reasoning more efficient. For example, in the method based on Statistical Relational Learning (SRL) [34], the neural network is used to approximate the search algorithm to realize the efficient reasoning of the model. The second category is dominated by the neural system and supplemented by the symbolic system, which uses feature vectors as input data, and introduces the prior knowledge of symbolic representation into the neural system in the process of solving [35], [36], [13], [17], boosting the learning process. For example, the method of regularization take symbols as the constraints of the neural network to guide the learning of the model. In addition to the two main methods above, there are other neural-symbolic system methods, such as abductive learning [27]. According to the method of taxonomy in this paper, the classification of the existing neural-symbolic system is shown in Table 2. Specifically, Table 2 summarizes and analyzes the existing main work from six aspects: representative methods, combination modes, specific methods, representations of symbols, problems, and applications. The specific details and practical effects of these methods will be described in Section 3.

2.2 Heavy-reasoning light-learning methods

The method of heavy-reasoning light-learning mainly adopts the methods of symbolic system to solve the problems in machine reasoning, and introduces neural network to assist in solving the problems [28], [29], [30], [31], [24], [32], [33]. The basic principle of this method is shown in Fig. 4. Firstly, the unstructured training data X is put into the neural network model, the obtained data is transformed to symbolical representation, then it is put into the reasoning solver. Finally, the reasoning result is output.

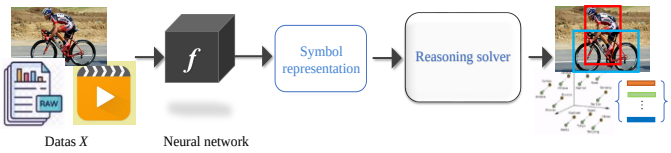


Fig. 4. Schematic diagram of heavy-reasoning light-learning. It is to introduce machine learning to logical reasoning, and mainly solves the problem through logical reasoning.

2.3 Heavy-learning light-reasoning methods

The method of heavy-learning light-reasoning mainly applies methods of the neural system to solve the problems in machine learning, and introduces symbolic knowledge in the training process [35], [36], [13], [17]. The basic principle of this method is shown in Fig. 5. The training data includes unstructured data X and structured symbolic knowledge S. After the learning of the neural network, the output can be used to make predictions $P(Y|X,S)$.

2.4 Abductive learning

Abductive learning is a balanced combination method of the neural system and symbolic system. To illustrate, the degree of participation of neural system and symbolic system in the process

TABLE 2
Classification of neural and symbolic systems

Representative methods	Combination modes	Specific methods	Representations of symbol	Problems(Shortages)	Applications
DeepProLog [24](2018)	Heavy-reasoning light-learning	Statistical Relational Learning	First-order logic and Boolean representation	Poor robustness	Classification
pLogicNet [10](2019)				Slowly reasoning	Knowledge graph reasoning
ExpressGNN [11](2020)					
RNM [37](2020)			First-order logic and numerical representation		Classification
NMLN [22](2019)				Poor robustness	Knowledge graph reasoning
NLIL [38](2020)				No interpretability and poor robustness	Image recognition and Knowledge graph reasoning
SL [36](2018)	Heavy-learning light-reasoning	Regularization		Requiring a large amount of label data	Classification
SBR [39](2018)					
LENSR [12](2019)			Propositional logic and numerical representation	Weak generalization ability and requiring a large amount of label data	Visual relationship detection
HDNN [40](2016)			First-order logic and numerical representation		Classification
CA-ZSL [41](2019)			Knowledge graph and numerical representation		Zero-shot learning
LSFSL [18](2019)					Few-shot learning
SEKB-ZSR [14](2018)		Knowledge transfer			Zero-shot learning
DGP [15](2019)					
KGTN [17](2020)					Few-shot learning
GABL [42](2021)	Alternation of learning and reasoning	Abductive learning	First-order logic and Boolean representation	Requiring a large amount of label data	Image recognition

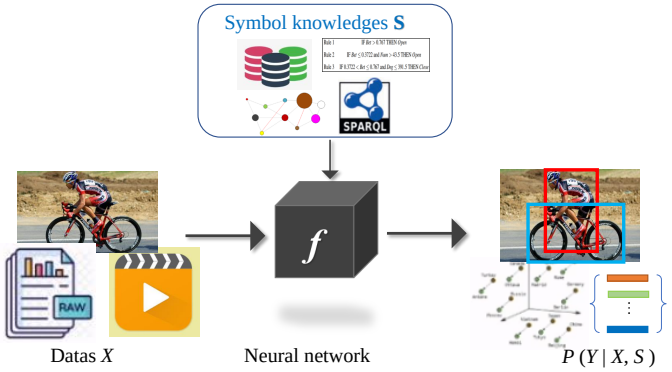


Fig. 5. Schematic diagram of heavy-learning light-reasoning. It introduces logical reasoning technology to machine learning, and the main body relies on machine learning to solve problems.

of solving problems is equal. It embeds the deductive inverse method into the induction process of machine learning, and the basic principle is shown in Fig. 6. In the learning process of the model, the original data is put into the classifier to obtain pseudo-labels; Then the model converts the pseudo-labels into symbolic representation, which is acceptable by the knowledge reasoning system, such as A and ¬ B and ¬ C, etc. Next, the model minimizes the inconsistency between the symbolic representation and the knowledge base (KB) to revise pseudo-labels, and output the deductive label. For example, ¬ C is revised to C, this is the result of deduction; Finally, a new classifier is trained with the deductive label and the original data to replace the original classifier. The above learning process is an iterative process until the classifier no longer changes or the pseudo-labels are consistent with the knowledge base.

3 Types and Representations of the Symbol

3.1 Types of the symbol

The symbol is mainly divided into two categories: logic and knowledge graph. Logic can be categorized into first-order logic and propositional logic, etc.

3.1.1 *First-order logic*

First-order logic (FOL) is a formal language [43]. It describes knowledge in the form of rules (formulas) and has strong expression ability. The language of first-order logic consists of 4 types of elements, connectors and, quantifiers. 4 types of elements:

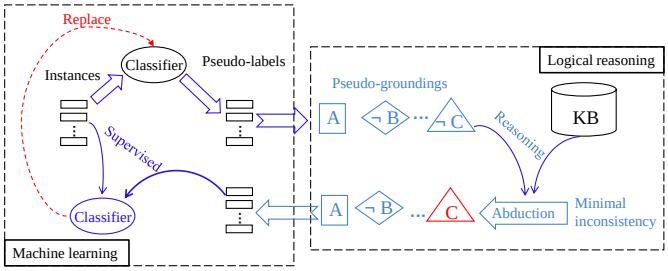


Fig. 6. Framework of abductive learning. It combines machine learning and logical reasoning iterative learning, and the two technologies together to solve problems. The figure is from reference [27].

constants, variables, functions, predicates. Constants represent objects in the domain of interest (for example, father(a,b), a=Bob, b=Mara, a and b are constant). Variables range over the objects in the domain (for example, father(x,y), x is the father of y, and the variable x is limited to the scope of the father class). Functions represent mappings from tuples of objects to objects. Predicates represent relations among objects in the domain or attributes of objects. The connector includes "$\wedge$", "$\vee$", "$\neg$" and "$\Rightarrow$". The quantifier includes the universal quantifier "$\forall$" and the existential quantifier "$\exists$". Generally, when using first-order logic, there is the term *atom*, which means applying a predicate to logical variable (for example, father(x,y)). First-order logic is the combination of atoms through connectors, and the expression can be written in the following form:

$$B_1(x) \wedge B_2(x) \wedge \cdots \wedge B_n(x) \Rightarrow H(x) \tag{1}$$

where $B_1(x)$, $B_2(x)$, $\cdots$, $B_n(x)$ represents the rule body, which is composed of multiple atoms. "$\Rightarrow$" represents the implication symbol. $H(x)$ represents the rule head and is the result derived from the rule's body. When variables in the first-order logic are replaced by constants, it is called *grounding*. For example, a is a constant, and the atom $H(a)$ after the grounding is called the ground atom.

The first-order logic calculates the True and False of each formula through assignment, which usually assigns Boolean variables for ground atoms. *A possible world* that assigns True or False to all ground atoms and is not unique. A formula is satisfiable if and only if there is at least one world in which it is True.

3.1.2 Propositional logic

The propositional logic is a declarative sentence, which includes either True or False. A declarative sentence consistent with the fact is a true sentence, while a declarative sentence inconsistent with the fact is a false sentence. The connectors between propositions are the same as the first-order logic, which is "$\wedge$", "$\vee$", "$\neg$" and "$\Rightarrow$". The only difference between the two logical expressions is: the first-order logic is in the form of variable and predicate (for example, father(x,y)), while the expression of propositional logic is in the form of constant and predicate (for example, a and b are constants, father(a,b)). The following is the expression of propositional logic:

$$P \Rightarrow Q \tag{2}$$

Where P represents the antecedent (condition) and Q represents the consequent (conclusion).

3.1.3 Knowledge graph

The knowledge graph is a natural symbol. It is not only a semantic network to describe entity relationships, but also a formal description framework for general semantic knowledge. It is stored in the form of triples (subject, predicate, object) and displayed in the form of a table or graph. This paper mainly represents the knowledge graph in the form of a graph.

The knowledge graph is a directed and labeled graph. Nodes represent semantic symbols (concepts), and anything can be used as nodes, such as animals, computers, and people. Edges connect node pairs and express the semantic relationship between them, such as the food chain relationship between animals, the network connection relationship between computers, friend relationship between people, etc. The formal representation of knowledge graph: $G = (H,R,T)$, Where $H = \{h_1,h_2,...,h_n\}$ represents the set of head entities, and n represents the number of head entities; $T = \{t_1,t_2,...,t_m\}$ represents the set of tail entities, and m represents the number of tail entities; $R = \{r_1,r_2,...,r_k\}$ represents the set of relationships, and k represents the number of relationships. Fig. 7 shows an example of the knowledge graph. In the example, a triplet (cat, attribute, paw), nodes (head entity and tail entity) are cat and claw, and the relationship is an attribute. This triple means "The attribute of a cat is paw." The knowledge graph contains a large number of common sense, rules, domain knowledge, and semantic relationships between entities, and is an indispensable basic resource for artificial intelligence applications.

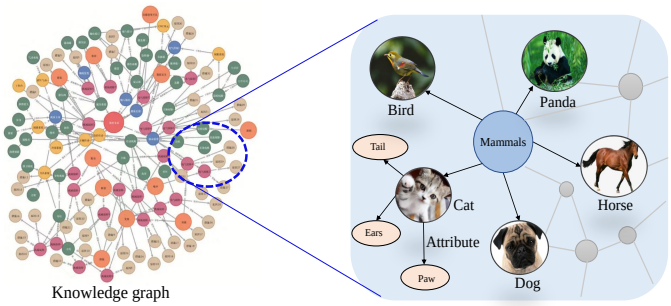


Fig. 7. An example of a knowledge graph.

3.2 Representations of the symbol

Representation of the symbol is an important part of the neural-symbolic system. It converts symbols into numerical representation to participate in the calculation of the model.

3.2.1 First-order logic

The first-Order logic usually uses numerical representation [12], [35], [44], [45], [46], [10], [11], [22], [37], which mainly adopts t-norm fuzzy logic [47], logic tensor network (LTN) [48], and Markov logic network (MLN) [49] to convert logical truth values into the interval [0,1] and perform uncertainty reasoning.

(1) **t-norm fuzzy logic.** t-norm fuzzy logic extends the first-order logic from Boolean $\{0,1\}$ to continuous and differentiable interval [0,1]. Its calculation process mainly includes two steps: grounding the first-order logic formula and calculating quantifiers.

Grounding first-order logic formula. Construct an expression tree for each ground first-order logic formula, and use fuzzy logic operation units to replace the basic logical connectors ($\neg$, $\wedge$, $\vee$, $\Rightarrow$) according to a certain logical semantics, which is

to perform algebraic calculations. Table 3 lists three algebraic operation methods corresponding to the connector ¬, ∧, ∨, ⇒, Where a and b are constants whose domains are in the interval [0,1].

Calculating quantifiers. For the first-order logic of the universal quantifier "∀" limited variables, the average of t-norm values of all variables within the limited range is usually calculated according to the limitation. For the first-order logic that the variable is limited by the quantifier "∃", it usually takes the value of the largest t-norm value of all the variables in a limited range. When multiple quantifiers appear, the corresponding calculations are performed in order of the outside to the inside. For example, the expression of first-order logic is $\forall x_1 \forall x_2 \neg A(x_1,x_2) \wedge B(x_2)$, the variables in the expression are x_1 and x_2, and the predicates are $\neg A$ and B. When the input data (constant) is $D = \{D_1, D_2\}$, the calculation formula for the output value φ is as follows:

$$\varphi(D,A,B) = \frac{1}{|D_1||D_2|} \sum_{x_1 \in D_1} \sum_{x_2 \in D_2} [1 - A(x_1,x_2)] B(x_2) \tag{3}$$

(2) **Logic tensor network.** Logic tensor network is a popular representation method of learning relationship (predicate). It is an extension of neural tensor networks (NTN) [50]. Its main idea is to encode real logic into the deep tensor network. In this symbolic representation method, each element in the first-order logic needs to be instantiated. The instantiated calculation is denoted as G. The instantiation of related terms and clauses is defined as follows [48]:

Definition 1 The instantiated first-order logic is converted to a truth vector to meet the following conditions:

1. $G(c) \in R^n$ for every constant symbol $c \in C$;
2. $G(f) \in R^{n \cdot \alpha(f)} \to R^n$ for every $f \in F$;
3. $G(P) \in R^{n \cdot \alpha(p)} \to [0,1]$ for every $p \in P$.

G is inductively extended to all the closed terms and clauses, as follows:

$$G(f(t_1,...,t_m)) = G(f)(G(t_1),...,G(t_m)) \tag{4}$$

$$G(P(t_1,...,t_m)) = G(P)(G(t_1),...,G(t_m)) \tag{5}$$

$$G(\neg P(t_1,...,t_m)) = 1 - G(P(t_1,...,t_m)) \tag{6}$$

$$G(\varphi_1 \vee \cdots \vee \varphi_k) = \mu(G(\varphi_1),...,G(\varphi_k)) \tag{7}$$

Where n represents the number of attributes of the constant, $\alpha(.)$ represents the number of variables contained in the function or relationship, t_i represents the variables in the first-order logic, and m represents the number of variables, $\varphi_i (i = 1,2,...,k)$ represents the atom in the first-order logic, and μ represents the above calculation of t-norm.

The main idea of the logic tensor network is to input all constant vectors into the network and calculate the scores of related functions and predicates, and then calculates the probability of a rule established through the t-norm. Specifically, if f is an m-ary function, $v_i \in R^n (1 \le i \le m)$ is an instantiated real value vector, and the function score $G(f)(v_1,...,v_m)$ can be written as the following calculation form:

$$G(f)(v_1,...,v_m) = \boldsymbol{M_f v} + N_f \tag{8}$$

Where $\boldsymbol{M_f}$ is $n \times mn$ matrix, $\boldsymbol{v} = ({v_1}^{\mathrm{T}},...,{v_m}^{\mathrm{T}})^{\mathrm{T}}$ is the mn-ary matrix obtained after v_i splice, and N_f is an n-dimensional vector.

If P is an m-ary predicate, the predicate score $G(P)$ is calculated as follows:

$$G(P) = \sigma(\mu_P{}^{\mathrm{T}} \tanh({v}^{\mathrm{T}} \boldsymbol{W_P}^{[1:k]} v + \boldsymbol{V_P} v + B_P)) \tag{9}$$

Where $\boldsymbol{W_P}^{[1:k]}$ represents the 3-dimensional tensor of $R^{mn \times mn \times k}$, k represents the number of types of variables, $\boldsymbol{V_P}$ represents the $R^{k \times mn}$ matrix, B_P represents the R^k vector, and σ represents the sigmoid function. After the function and the predicate are predicted by the neural network, the clauses of the first-order logic formula are merged one by one through the t-norm.

Fig. 8 shows a specific example $Smokes(x) \Rightarrow Cough(x)$, which means "Smoking causes cough.". The calculated process is as follows. First, rewrite the implication as disjunction; Then set up two neural networks $G(\neg\text{Smokes})$ and $G(\text{Cough})$ for the two predicates and input embedding of the constant v, which can obtain a score of two predicates. Finally, the t-norm is used to calculate the probability of the whole rule based on the score. It is worth noting that during the training process, the logic tensor network will build a sub-network for each predicate, so the training speed will be slower.

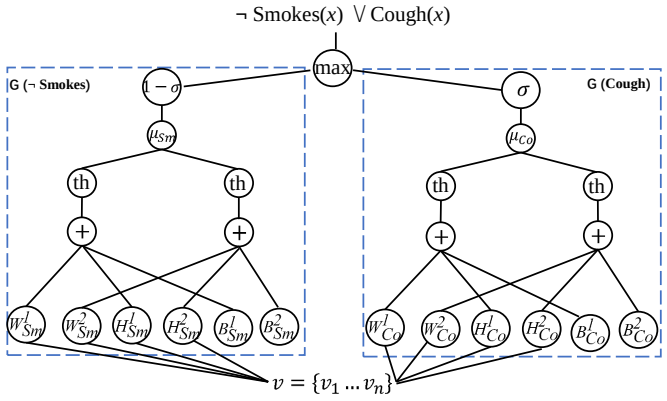


Fig. 8. Logic tensor network.

(3) **Markov logic network.** Markov logic network combines logic rules with probability graph model to give first-order logic the ability to express uncertainty. The definition of Markov logic network is as follows [49]:

Definition 2 Markov logic network is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic, and w_i represents weight of F_i and is a real number. Together with a finite set of constants $C = \{c_1, c_2, ..., c_n\}$, it defines a ground MLN:

(1) Any ground atom in MLN corresponds to a binary node in the ground Markov logic network. If a ground atom is true, the corresponding binary node takes the value 1 and 0 otherwise.

(2) Having a feature for any ground formula in MLN. The feature is 1 if the ground formula is true and 0 otherwise. The weight of the feature is the w_i associated with F_i.

(3) Ground Markov logic network obtained by replacing the variable with the constant set C, which is called grounding Markov logic network.

Nodes in the Markov logic network are generated by all ground atoms in ground Markov logic network, and edges are generated by the relationship between the ground atoms (Edge appears between two nodes if the corresponding ground atoms appear together in at least one grounding of formula in MLN.). Given the same Markov logic network and different constant sets C, it can form different ground Markov logic networks. The scale of the ground Markov logic network is determined by the size

TABLE 3
The algebraic operations corresponding to logical connectives in t-norm fuzzy logics

Operation \ t-norm	Product	Minimize	Łukaseiwicz
a ∧ b	a · b	min(a,b)	max(0,a+b-1)
a ∨ b	a+b-a · b	max(a,b)	min(1,a+b)
¬ a	1-a	1-a	1-a
a ⇒ b	min(1,b/a)	a≤ b ? 1 : b	min(1,1-a+b)

of the constant set C. The probability distribution over a possible world x specified by ground MLN is given by:

$$P(X=x)=\frac{1}{Z}exp\{\sum_i w_i n_i(x)\}=\frac{1}{Z}\prod_i \psi_i(x_{\{i\}})^{n_i(x)} \tag{10}$$

Where Z represents the partition function, w_i represents the weight of rule, $n_i(x)$ represents the number of times that the value of the rule F_i is true, ψ_i represents the potential function and $x_{\{i\}}$ represents clique of graph.

The following introduces a simple Markov logic network example. Table 4 shows the two rules (F_1,1.5), (F_2,1.1) of this example [51].

Given the constant set $C=\{A,B\}$, then generated ground Markov logic network is shown in Fig. 9.

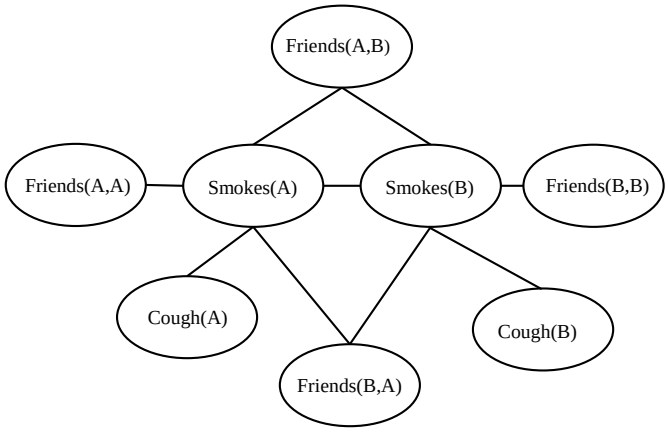
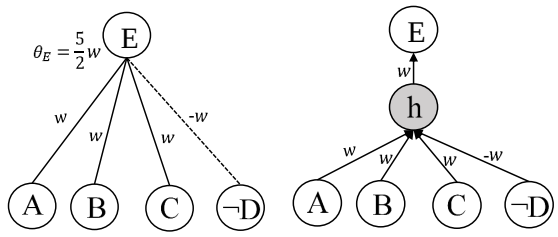


Fig. 9. Ground Markov logic network.

3.2.2 Propositional logic

The knowledge-based artificial neural network(KBANN) [3] and connectionist inductive learning and logic programming(CILP) [8] are two early neural-symbolic methods, which encode propositional logic into the structure of the neural network.

KBANN's method of constructing a neural network is: Firstly, constructing the conjunctive rule set into a graph in the form of "and-or tree". Then fill in the hidden nodes in the graph and generating connections for nodes that have no edges between adjacent layers. Given a rule $(A\wedge B\wedge C\wedge\neg D\Rightarrow E)$, Fig. 10(a) is obtained according to the above method, nodes represent predicates (There are no hidden nodes in this example), edges represent the connector between predicates, the solid line represents the edge of positive weight, the dotted line represents the edge of negative weight, and positive and negative weight is determined by the



(a) KBANN (b) CILP

Fig. 10. Symbolic representations of KBANN and CILP.

symbol of the predicate. w means positive weight, $-w$ means negative weight, and KBANN usually sets w to 4. Threshold θ means bias, $\theta=(P-\frac{1}{2})w$. P means the number of positive weight edges in a rule.

The method of CILP for constructing a neural network is to use the rule body in the rule set as the input unit of the neural network, and the rule head as the output unit, and learn the mapping function from the rule body to the rule head through the neuron h. According to the above rule set, the symbolic representation obtained by CILP is shown in Fig. 10(b). where w is network weight to be learned, and its sign depends on the non of proposition unit.

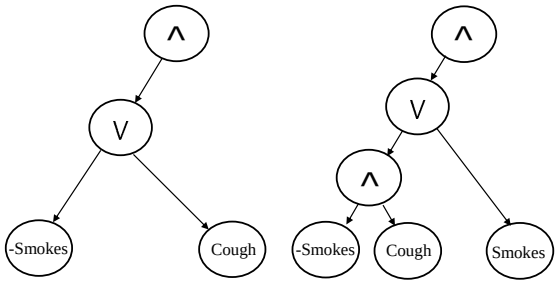
With the prevalence of representation learning, logical formulas are usually converted into directed acyclic graphs, and graph neural network (GNN) [52] is used to learn the vector representation of logical formulas. Conjunctive Normal Form (CNF) and deterministic-Decomposable Negation Normal Form (d-DNNF) [53], [54] are representatives of the transformation of logical formulas into directed acyclic graphs. For CNF, the construction process of the directed acyclic graph is: first, rewrite the logical formula into CNF, and then take the conjunctive symbol as the root node and the proposition as the leaf node. d-DNNF is more complicated than CNF, and it is more difficult to construct a directed acyclic graph of d-DNNF. It is usually use the c2d [55] method. For example, the directed acyclic graph CNF and d-DNNF corresponding to the logical formula $Smokes\Rightarrow Cough$ are shown in Fig. 11(a) and Fig. 11(b), respectively.

3.2.3 Representation of Knowledge graph

Representation of knowledge graph is to embed discrete symbols (entities, attributes, relationships, etc.) from the knowledge graph into a low-dimensional vector space to obtain distributed representation [56], [57], [58], [59], [60], [61], [62], [63]. These methods mainly include translation-based models [56] [57], [58] and multiplicative models [59], [60], [61].

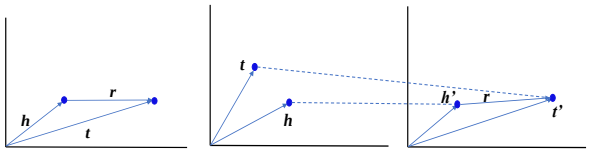
TABLE 4
An instance of a Markov logical network

Proposition	First-order logic	Weight
Smoking causes cough.	$F1: \forall x, Smokes(x) \Rightarrow Cough(x)$	1.5
If two people are friends, either both smoke or neither does.	$F2: \forall x \forall y, Friends(x,y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$	1.1



(a) CNF (b) d-DNNF

Fig. 11. Directed acyclic graph of CNF and d-DNNF.



(a) TransE (b) TransR

Fig. 12. Compute graphs of vector space. Figure (a) shows that head entity h and relationship r are summed to obtain tail entity t in the same vector space; Figure (b) shows that head entity h and tail entity t are projected into r space, and then performing the same calculation process as in figure (a).

(1) **Translation-based models.**

In the knowledge graph, the transformation-based model transforms the head entity to the tail entity by the relation to learn the vector representation of the entity. For example, TransE [56] maps the entities and relations into the same vector space, and it forces the added embedding of head entity h and relation r to be close to the embedding of the corresponding tail entity t. TransR [57] method improves the drawbacks of TransE–fail to solve the symmetric relationship. The triple (h,r,t) and the triple (t,r,h) are symmetrical, but h and t are not the same entity. First, TransR transforms the head entity h and the tail entity t into the relation vector space through the relation transformation matrix $\boldsymbol{M_r}$ to obtain h' and t', and then TransR performs the same vector calculation as TransE. For given triple (h,r,t), the formal definition of TransE is as shown in Eq.(11). See Fig.12(a) for the diagram of vector space calculation. The formal definition of TransR is as shown in Eq.(12). See Fig.12(b) for the diagram of vector space calculation.

$$h + r = t \tag{11}$$

$$\boldsymbol{M_r} h + r = \boldsymbol{M_r} t \tag{12}$$

(2) **Multiplicative models.** Multiplicative models produce entity and relation embedding via tensor inner product. The RESCAL [59] adopts matrix decomposition to learn the vector representation of entities and relationships. The triple (h,r,t) is represented as an element in entity relationship tensor X^{nmn}, where n is the number of entities, and m is the number of relationships. Suppose the entity relationship tensor is $\boldsymbol{X} = \{X_1, X_2, ..., X_m\}$, and each relationship matrix $\boldsymbol{M_i}$ is a square matrix with the specific matrix decomposition formula is as in Eq.(13). Since the RESCAL method calculates the tensor inner product for all entity pairs in the knowledge graph, a large number of parameters are required in the calculation bringing huge computational costs. In order to ease this problem, the DisMult [60] method is brought up. To reduce parameters, the DisMult method replaces the relational matrix $\boldsymbol{M_i}$ in the RESCAL method with a diagonal matrix.

$$\boldsymbol{X_i} = h \boldsymbol{M_i} t \tag{13}$$

The optimization goal of the above method is designing a scoring function according to the calculation formula of vector space in order to effectively learn the vector representation of entities and relations. This representation can express the semantic information of entities and relations with the effective calculation of the semantic relationship between entities and relations [64].

4 Methods of Neural-Symbolic System

According to the taxonomy of methods of the neural-symbolic system described in section 1, this section describes methods of the specific implementation of the neural-symbolic system in detail.

4.1 Heavy-reasoning light-learning methods

Heavy-reasoning light-learning methods are mainly based on statistical relational learning [34], which combines logic and probability to capture complexity and uncertainty. Statistical relation learning methods are categorized into ProbLog-based models, MLN-based models, ILP-based models. As shown in Table 5. Both ProbLog and MLN use the existing rule sets to calculate the probability and introduce uncertainty reasoning. ILP automatically induces new logic rules from the data for model learning and reasoning.

4.1.1 ProbLog-based models

Probabilistic logic programming (ProbLog) [65] is an extension of the probabilistic logic language. It is a method of statistical relational learning with reasoning ability. Specifically, its rules and atoms have given probabilities, so that the inference task is no longer binary classification of "yes" or "no", but values to improves the expressive ability of the model. To enable the model to have both reasoning and learning ability, Robin, et al. [24] proposed a model that organically combines probability, logic, and deep learning based on ProbLog, called DeepProLog. This model proposes a framework that combines general neural networks with

TABLE 5
Representatives based on the probabilistic logical model

Category	Problem solved	Representative work
ProbLog-based	Capturing complex relationships and Characterizing uncertainty	DeepProLog [24]
MLN-based	Characterizing uncertainty and improving reasoning efficiency	pLogicNet [10],ExpreeGNN [11]
ILP-based	Cpaturing new knowledge and solving robust	∂ILP [28],NLIL [38]

probabilistic logic in a certain way for the first time. It has the advantage of stronger expressive ability, and it can be trained end-to-end based on samples.

DeepProLog is a kind of probabilistic programming language, combining with deep learning in the way of "neural predicates". Specifically, input entity pairs (data in this paper are 2-ary predicate), the classifier of the neural network predicts the probability that the entity pair belongs to all the predicates, and then it uses the predicted result for the inference of the probabilistic program. During the training, DeepProLog utilizes a tensor calculation graph, which is optimized by gradient descent.

The main advantages of DeepProlog are: (1) it is a programming language supporting neural networks and machine learning, with semantics definition ability; (2) It combines symbol and neural network to improve model's learning and reasoning ability. Its deficiency is that it needs to instantiate predicates before constructing a network (calculated graph).

4.1.2 MLN-based models

MLN is an undirected graph, which maps first-order logic formula to Markov network, training parameters of reasoner, and implementing uncertainty inferencing. In MLN, the cost of learning and inference increases exponentially with the size of ground MLN. Given the joint probability distribution of MLN in section 3.2.1 (see Equation 10), if the data set is relatively large, the size of the ground MLN will be relatively large, and the calculation of partition function Z will be intractable. Therefore, when solving large-scale practical problems, methods of precise inference of statistical relational learning are no longer being feasible. Although a variety of approximate reasoning methods have been proposed [66], [67], [68], [69], [70], [71], the computational cost is still high. Due to outstanding ability of neural networks in optimizing learning, researchers have proposed new research ideas, using neural networks to replace traditional approximate reasoning of MLN (such as variational methods). The following idea takes knowledge graph reasoning task as an example for description.

Though MLN effectively uses rules, it is difficult to infer on large-scale knowledge graphs. Neural networks can effectively infer, but cannot directly use rule knowledge. Under this circumstance, researchers have successively proposed probabilistic Logic Neural Networks (pLogicNet) [10] and ExpressGNN [11]. The main idea of the two models is to model the problem of reasoning in knowledge graph (triple completion problem) as an inferred problem of hidden variables in probability graph. Both ideas adopt a combination of variational EM and neural network to approximate inference. The basic process is as follows:

Firstly, available rules are modeled as an undirected graph represented by MLN, such as factor graph as shown in Fig. 13. The circular nodes in the figure represent ground atoms in all rules, and the square nodes represent factors (a rule corresponds to a factor node). If a ground atom appears in a rule, add an undirected edge between the corresponding circular node and square node; Then, determine observed variables and hidden variables in an undirected graph. Triples in knowledge graph correspond to observed variables in an undirected graph (corresponding to white nodes), and triples nodes in a knowledge graph are hidden variables (corresponding to the gray nodes); Finally, calculating the probability that any query hidden variable is true according to the joint probability distribution of undirected graph.

ExpressGNN improves inference network of pLogicNet, using graph neural network (GNN) to replace flattened embedding table and adding a tunable part to the embedding of entity to alleviate the problem that isomorphic nodes have the same embedding. The specific framework of ExpressGNN is shown in Fig. 13. The Boolean random variables in MLN are all ground atoms contained in ruleset, and each ground atom corresponds to a triple of form of (subject, predicate, object). The white ground atoms corresponding to triples appearing in the knowledge graph as observed variables (known facts) of MLN. The triplet corresponding to the gray ground atom does not appear in the knowledge graph as a hidden variable (unknown fact) to be queried. Applying variational EM algorithm for the joint probability distribution of MLN, calculate the probability that latent variable is true, and complete knowledge graph.

The training process of ExpressGNN is interactive: Rule's weights learned in symbolic space are used as known information in continuous space to assist in predicting each unknown fact; In addition, ExpressGNN uses a neural network to parameterize variational posterior, called neuro-variational inference, which greatly improves the efficiency of inference. Fig.14 illustrates the process of neuro-variational inference : Entity embedding learned by GNN is the input relation prediction network, and the output probability of that triplet is true. θ_1 represents parameters of GNN, θ_2 represents the parameters of inferred network(relation prediction network), θ_c represents tunable factor, and r_k represents the k-th relation.

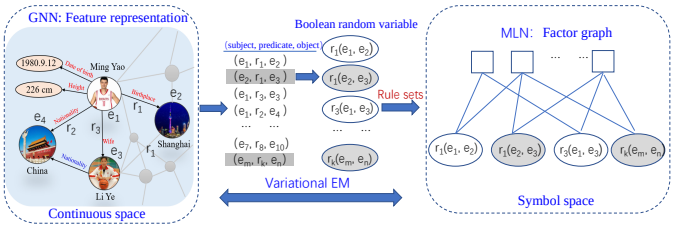


Fig. 13. The framework of ExpressGNN model.The model contains continuous space and symbolic space, and the variational EM algorithm acts as a bridge connecting continuous space and symbolic space.

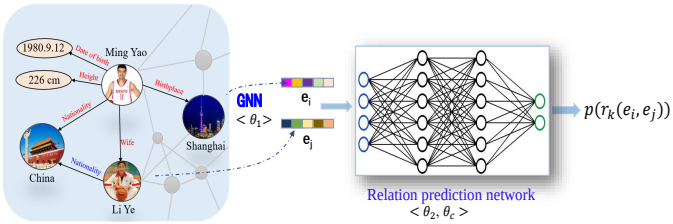


Fig. 14. Schematic diagram of neural variation inference process. This process is included in the inference process of the symbolic space MLN in Figure 13, which replaces variational posterior calculation of the traditional EM algorithm.

4.1.3 ILP-based models

The precondition of the above method is that training rules are known. ProbLog-based models input rules into the neural network for training; MLN-based methods model rule as a probability graph, and the potential function of the probability graph is manually designed. But only relying on rules constructed by experts can't completely capture the knowledge implicit in data. For this reason, researchers have begun to explore ways that can automatically capture rules from data. Marra et al. [22], [37] extended MLN and designed a general neural network to automatically learn rules and MLN's potential functions from original data. In addition to methods of automatically learning rules MLN-based, some scholars have proposed differentiable ILP based on the traditional ILP method, which combines neural network and logic [72] [28], [73], [74], [75]. For example, ∂ILP [28] is a forward reasoning method, which mainly uses set of predefined templates to construct rules, and it applies rules multiple times on background data to infer new facts for evaluation. Even when encountering noisy data, ∂ILP still learns effective FOL.

In order to be computationally feasible, the current method expresses the chain rule as a Horn clause and controls the length of search, the number of relationships, and entities. This has caused the problem of the limited expressive power of complex rules. To solve these problems, Yang et al. [38] proposed neural logic inductive learning (NLIL), which can reasoning complex logic rules(such as tree and conjunctive rules, etc.), explaining the patterns hidden in data through learned FOL, and applies it to target detection tasks. NLIL extends the multi-hop reasoning framework for general ILP problems and is a differentiable ILP model. NLIL implements a divide-and-conquer strategy, decomposing the search space into 3 subspaces in a hierarchical manner. Each subspace adopts an attention mechanism for effective search, calculating the score of a rule parameter into the weighted summation of the attention network. Specifically, the process of rule generation is mainly to run the Transformer model, which consists of a stack of 3 Transformers. Each Transformer is a multi-head attention module that generates a set of weights. Applying generated weights and hierarchical structure executes multi-hop reasoning.

4.2 Heavy-learning light-reasoning

Heavy-learning light-reasoning methods are mainly divided into regularization models and knowledge transfer models. Regularization models [12], [13], [76], [39], [77], [18], [78] add symbols in the form of regular terms to the objective function of training model and as a kind of prior knowledge to guide model's training. The knowledge transfer models [14], [15], [17] establish connection between the visual space and the semantic space, transfer symbolic knowledge of the semantic space to the visual space, and assist model's learning.

4.2.1 Regularization models

According to types of symbols, regularization models can be divided into logic-based models and knowledge graph-based models.

(1) **Regularization models of logic-based**

Diligenti and Xu et al. [39], [36] proposed semantic-based regularization (SBR) and semantic loss (SL) methods respectively. They use logical knowledge as a constraint of the hypothesis space. If the corresponding logical specification or logical theory is violated, the model will be punished. SBR is a method of learning from constraints. This method combines classic machine learning (with continuous feature representation learning ability) and statistical relation learning (with advanced semantic knowledge reasoning ability) to solve problems such as multi-task optimization and classification. The SL method combines the automatic reasoning technology of propositional logic with existing deep learning architecture, which feeds the output of the neural network into loss function as a constraint of the learnable neural network. The propositional logic is encoded into the loss function of the neural network by a training algorithm to improving the learning ability of the neural network. SL adopts the marginal probability distribution of the target atom to define the regular, and uses arithmetic circuit [79] to evaluate the effectiveness of the model.

Hu et al. [40] proposed a general framework called harnessing deep neural networks with logic rules (HDNN), where neural network models include CNN and DNN, etc. HDNN adopts the idea of knowledge distillation(Guide the learning of small-scale networks through a large-scale and well-trained network) to design teacher networks and student networks. The teacher network can learn information from labeled data and logic rules (unlabeled data). The student network is close to the teacher network, so that structured information encoded by logic rules can restrict learning of the student network. The framework of the model is shown in Fig. 15. Where the rule knowledge distillation includes the student network $p_\theta(y|x)$ and the teacher network $q(y|x)$. The input of the teacher network is labeled data and unlabeled data (logic rules), and the input of the student network is labeled data. The red dotted line represents the integrated coded rule information, that is, by balancing the output of the teacher network and the student network, and updating the parameter θ of the student network with real labels, the teacher network $q(y|x)$ of the coding rule information is approached to the student network $p_\theta(y|x)$, so that the teacher network $q(y|x)$ simulates the regularized output of the student network $p_\theta(y|x)$, and its objective function is shown in Eq.(14). The teacher network construction part uses soft logic to encode the first-order logic, and it is to give the first-order logic probability value and perform the numerical calculation. $q(y|x)$ is projected to the subspace structure constrained by the rule through $p_\theta(y|x)$, and the construction process needs to meet the following conditions: (1) The probability distribution $q(y|x)$ of the teacher network should be as close as possible to the probability distribution $p_\theta(y|x)$ of the student network; (2) The teacher network should obey the rules as much as possible. The formal representation of the two conditions is shown in Eq.(15). The whole model is iteratively trained. The difference from the

original knowledge distillation model is that the teacher network and the student network can learn at the same time.

$$\theta^{(t+1)} = \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{n=1}^{N}(1-\pi)l(y_n,\sigma_\theta(x_n)) + \pi l(s_n{}^{(t)},\sigma_\theta(x_n)) \tag{14}$$

$$\begin{cases} \min_{q,\xi\geq 0} KL(q(Y|X)\|p_\theta(Y|X)) + C\sum_{l,gl}\xi_{l,gl} \\ \lambda_l(1-E_q[r_{l,gl}(X,Y)]) \leq \xi_{l,gl} \\ gl = 1,...,G_l, l = 1,...,L \end{cases} \tag{15}$$

In Eq.(14), π is the limitation parameter calibrating the relative importance of the two objectives; x_n represents training data, y_n represents label of training data; l denotes the loss function selected according to specific applications (e.g., the cross entropy loss for classification); $s_n^{(t)}$ is the soft prediction vector of $q(y|x)$ on x_n at iteration t; $\sigma_\theta(x)$ represents output of $p_\theta(y|x)$; the first term is the student network, and the second term is the teacher network. In Eq.(15), $\xi_{l,gl} \geq 0$ is the slack variable for respective logic constraint; C is the regularization parameter; l is index of rule; gl is the index of ground rule; λ_l is the weight of the rule.

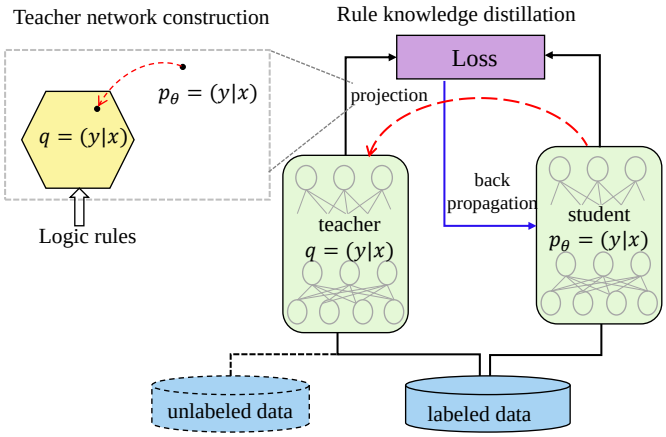


Fig. 15. The overview of HDNN framework. At each iteration, the teacher network is obtained by projecting the student network to a rule-regularized subspace (red dashed arrow); and the student network is updated to balance between emulating the teacher's output and predicting the true labels (black/blue solid arrows). The figure is from reference [40].

Xie et al. [12] integrated propositional logic into the relationship detection model and proposed a logic embedding network with semantic regularization (LENSR) to improving the relationship detection ability of the deep model. The process of LENSR mainly includes the following steps: (1) Apply the visual relationship detection model for each image to predict the probability distribution of the relation predicate; (2) The prior propositional logic formula related to the sample image is expressed as a directed acyclic graph by d-DNNF, and then use graph convolutional network to learn its probability distribution; (3) Design the objective function for the above two distributions to align. Fig.16 shows a schematic diagram of the model, which uses a propositional logic of the form $P \Rightarrow Q$. In this example, the predicate P represents $wear(person, glasses)$, and the predicate Q represents $in(glasses, person)$. On the left is the ground truth of the input image and the corresponding propositional logic (prior knowledge). The d-DNNF's directed acyclic graph of the propositional logic is sent to the Embedder q to obtain embeddings of the real propositional logic $q(F_x)$. On the right is the relation label predicted by the convolution model. The predicted labels are combined into a conjunctive normal form $h(x) = \wedge p_i$ to construct d-DNNF's directed acyclic graph, which is sent to the Embedder q to obtain embeddings of the prediction formula $q(h(x))$. The optimization goal of the model is shown in Eq.(16). L_{task} represents the loss of a specific task, and L_{logic} represents the loss of propositional logic, that is, the distance between vector $q(Fx)$ and vector $q(h(x))$, which is calculated in two norms.

$$L = L_{task} + \lambda L_{logic} \tag{16}$$

$$L_{logic} = \|q(f) - q(\wedge p_i)\|_2 \tag{17}$$

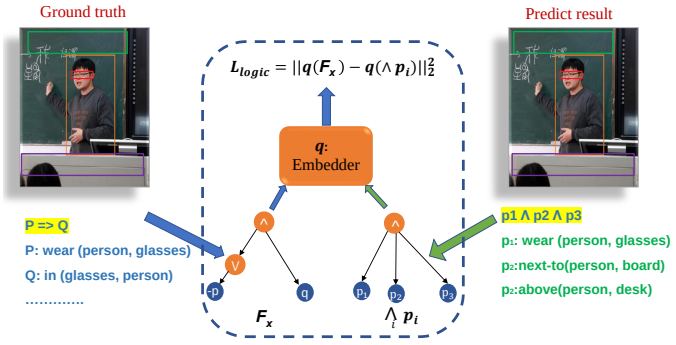


Fig. 16. LENSR model. The GCN-based embedder q projects logic graph to the vector space, satisfying the distribution of the predicted result is as close to distribution of the real label as possible. we use this embedding space to form logic losses that regularize deep neural networks for a target task.

(2) **Regularization models of knowledge graph-based**

Luo et al. [41] proposed a context-aware zero-shot recognition method (CA-ZSL) to solve the problem of zero-shot detection. It builds a model based on deep learning and conditional random fields(CRF) and uses the semantic relationship between classes to assist in identifying objects of unseen classes. The framework of the model is shown in Fig.17. The individual objects and pairwise features are extracted from the image. The instance-level zero-shot inference module adopts individual objects features to generate a unary potential function, and the relationship inference module uses pairwise features and relationship knowledge graphs to generate binary potential function. According to the conditional random field model constructed by the two potential functions, the label of the unseen objects is reasoned. The model integrates knowledge graphs into the calculation of the binary potential function of the conditional random field to constrain learning of the model. The optimization goal of the model is to maximize the joint probability distribution of the conditional random field, as shown in Eq.(18). Where θ represents the unary potential function, ψ represents the binary potential function, c_i represents the class, and B_i represents the object region in the image, γ represents balance factor, N is a number of objects.

$$P = (c_1...c_N|B_1...B_N) \propto exp(\sum_i \theta(c_i|B_i) + \gamma\sum_{i\neq j}\psi(c_i,c_j|B_i,B_j)) \tag{18}$$

Li et al. [18] proposed large-scale few-shot learning (LSFSL) to solve the task of few-shot image recognition. The large-scale is that the training samples of the data set contain more source class samples and fewer target classes, that is, more samples of seen classes and fewer samples of unseen classes. This method models a class hierarchical graph (the semantic relationship graph between the source class and the target class) and learns transferable visual

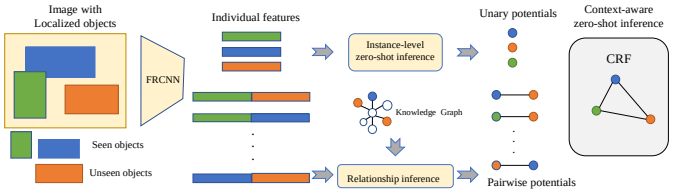


Fig. 17. The overall pipeline of CA-ZSL. The features of individual objects as well as pairwise are extracted from image and input into an instance-level zero-shot inference module and a relationship inference module respectively. Combined with the knowledge graph, the unary potential function and binary potential function of CRF are generated respectively to predict the labels of objects. The figure is from reference [41].

features that integrate the semantic relationship between the source class and the target class to realize the recognition of the unseen class. The specific implementation process of LSFSL is shown in Fig.18. The convolution neural network extracts the features of the input image and inputs it into the hierarchical prediction network. In this network, two steps are performed: Firstly, image features go through three different full connection layers(FC) to extract three levels of label features (class or superclass); Secondly, the label features of the three layers are merged as the feature representation of the first superclass layer of the class hierarchy graph. The label features of the first two layers of fusion are used as the feature representation of the second superclass layer of the class hierarchy graph. The feature of the leaf node in the class hierarchical graph directly uses the label feature of the first layer of the hierarchical network. Finally, the obtained image features of different levels and the semantic features of the corresponding class hierarchical graph are calculated by cross-entropy for inference. The model uses the class hierarchical graph to constrain the process of extracting features by the convolutional neural network so that the convolutional neural network can undergo multiple learning to obtain transferable visual features. The loss function is shown in Eq.(19), where L_{cls} is the cross-entropy loss, the predicted distribution of the l_i-th layer class obtained in the first step of p_{l_i}, $p_{l_i}{}'$ represents the predicted distribution of the l_i-th layer class finally obtained, and λ_i is a hyperparameter.

$$L(x,Y,\Theta) = L_{cls}(y_{l_i}, p_{l_i}) + \sum_{i=2}^{n+1} \lambda_i L_{cls}(y_{l_i}, p_{l_i}{}') \tag{19}$$

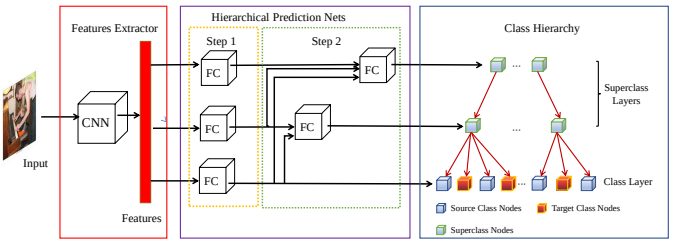


Fig. 18. The overview of LSFSL model. This model constructs a tree-structured class hierarchy, encoding semantic relations between source and target classes. It integrates the prior knowledge from the class hierarchy to learn transferable features. "FC" represents the fully connected network. This figure comes from reference [18].

4.2.2 *Knowledge transfer models*

Knowledge transfer integrates the knowledge graph that represents semantic information into the neural network model, and it makes up for the lack of data by transferring semantic knowledge. Knowledge transfer models are mainly used to solve zero-shot learning and few-shot learning tasks [80], [81].

(1) **Zero-shot learning**

zero-shot learning aims to apply seen classes to the training model, which makes the model with the ability to predict unseen classes, that is, there are no samples of new classes in the training set.

In visual tasks, considering that semantic information can make up for the deficiencies of visual data, researchers have successively proposed a zero-shot recognition model that combines semantic representation and knowledge graphs. They are SEKB-ZSR (zero-shot recognition via semantic embeddings) [14] and DGP (dense graph propagation module) [15]. Both models use the semantic classifier weights of the combined knowledge graph for all seen and unseen classes to constrain learning of the visual classifier weights, which can achieve knowledge transfer. The basic principles are: In the visual space, the model adopts a convolutional neural network to extract the visual features of images, and learns the visual classifier of the seen class; in the semantic space, the graph convolution neural network is used to learn the node features of the combined knowledge graph for all seen and unseen classes, and the class semantic classifier is obtained; In turn, the weights of the class semantic classifier are used to supervise the learning process of the visual classifier weights for realizing the transfer of semantic knowledge of new classes.

DGP improves SEKB-ZSR. To address the over-smoothing problem of the graph convolution network, the 6-layer graph convolution in the SEKB-ZS model is reduced to 2-layers. To enhance the connection between nodes in the hierarchical knowledge graph, the attention mechanism is used to calculate the connection weights between the nodes. The principle of DGP is shown in Fig. 19. The model is mainly divided into two parts: the visual part and the semantic part. The visual part is the ResNet module, which mainly uses CNN to extract the features of images to perform the classification. The semantic part is the dense graph propagation module, which learns semantic features of classes.

Specifically, the hierarchical knowledge graph is used as the input of the dense graph propagation module, and the initial feature is the word vector of the class. In the process of performing graph convolution, the module adopts two aggregation modes of ancestor propagation and offspring propagation respectively for update the feature of nodes, and performs classification to obtain the weight of the semantic classifier. The image is input the ResNet module, and the model performs CNN to extract visual features of the image for classification to obtain the weight of the visual classifier. To the ResNet module can obtain the ability to recognize unseen classes, the model uses the semantic classifier weights to supervise the visual classifier weights obtained during the training process. The entire model is trained end-to-end. The loss function is the similarity of the weights of the two module classifiers, as shown in Eq.(20). Where M represents the number of classes, P represents the dimension of the weight, $W_{i,j}$ represents the visual classifier weights, $W'_{i,j}$ represents the semantic classifier weights.

$$L = \frac{1}{2M} \sum_{i=1}^{M} \sum_{j=1}^{P} (W_{i,j} - W'_{i,j})^2 \tag{20}$$

(2) **Few-shot learning**

Few-shot learning aims to learn unseen classes from very few

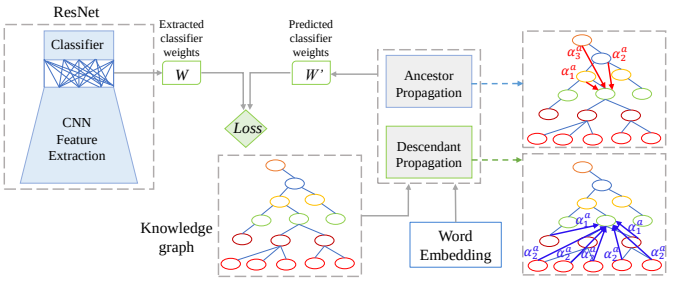


Fig. 19. DGP is trained to predict classifier weights W for each node/class in a graph. The weights for the training classes are extracted from the final layer of a pre-trained ResNet. The graph is constructed from a knowledge graph and each node is represented by a vector that encodes semantic class information(it is the classes word embedding in this paper). The network consists of two phases, a descendant phase where each node receives knowledge form its descendants and an ancestor phase, where it receives knowledge from its ancestors. The figure is from reference [15].

training samples(The training data contains only a few samples of the novel class.) so that the model can recognize the novel classes.

Chen et al. [17] found that transferring the correlation information between classes helps to learn new concepts. They utilized knowledge graphs to model the correlations between seen and unseen classes and combines with neural networks to propose a knowledge graph transfer network model (KGTN) for the problem of few-shot classification. As shown in Fig.20, KGTN mainly includes three parts: feature extraction module, knowledge graph transfer module, and prediction module.

The feature extraction module uses CNN to extract the feature vector of images. The knowledge graph transfer module models the classifier weights as a graph, and it uses a gated graph neural network (GGNN) to learn the representation of nodes. The nodes of the graph represent seen or unseen classes, the edges represent the semantic relevance between them, and the weight w_{init} on the edges represents the learnable parameters, which are randomly initialized before training. When training the model, for the $t = 0$ to $t = T - 1$ iteration, each node aggregates information from neighboring nodes. After T iterations, obtain the final weight w^*, w^* has captured the correlation between the seen and the unseen classes. The prediction module calculates the similarity between the weight w^* updated by GGNN and the image feature to predict the probability distribution of the label.

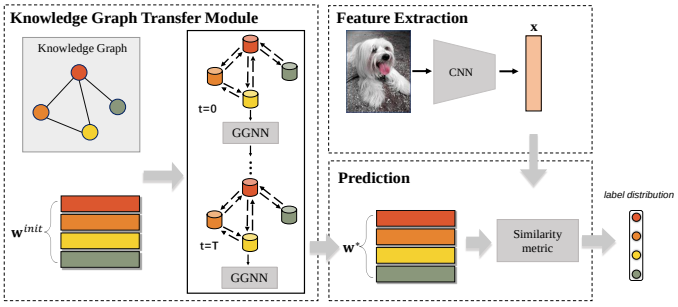


Fig. 20. Illustration of the KGTN model. It incorporates the prior knowledge of category correlation and makes use of the interaction between category classifier weights, which can better learn the classifier weights of unknown categories. The figure is from reference [17].

5 Application Areas

(1) **Visual relationship detection and image recognition**

Visual relationship detection and image recognition only use visual features to train the model, resulting in a relatively weak model. Recently, the emergence of a neural-symbolic system inspired scholars to introduce external knowledge–integrate image features with knowledge of logical symbol representation to improve the detection performance and robustness of the model. Marszalek and Forestier et al. [82], [83] suggested adopting methods based on symbolic knowledge to improve object detection capabilities, but their model only uses simple inclusion relations as constraints. Zhu and Nyga et al. [16], [84] suggested using the knowledge base, and MLN as the knowledge representation, to learn the scoring function and predict the relationship between the input image and the person. For example, input an image of a horse, and the model can predict that its relation with people is "ridable". Donadello et al. [13] combines neural networks with first-order logic to design a logic tensor network (LTN). When the presence of logical constraints, it can not only make effective reasoning from noisy but also use logical formulas to reasoning and describe characteristics of data. Therefore, LTN provides the interpretability of image recognition. Xie et al. [12] used the propositional logic between image object regions to enhance the detection ability of the deep model. For example, propositional logic is formed between the object and the object in the image through the spatial relationship, "people on horse", where "people" and "horse" are objects, "on" is a relation. This method models the propositional logic as a graph structure and learns the semantic representation of regions of an object by GNN as a constraint for the whole object detection task, which improves the detection ability of the model.

(2) **Knowledge graph reasoning**

The knowledge graph may contain incorrect, incomplete, or duplicate records. It is usually necessary to perform link prediction (completion), attribute classification, and delete duplicate records to improve the quality of the knowledge graph to applying better in other fields. For example, image recognition and public opinion analysis, etc. Qu and Zhang et al. [[10], [11]] modeled knowledge graph reasoning as a probabilistic inference problem and adopted the EM algorithm to solve it, which greatly improved the inference efficiency of the model. Marra et al. [22] assumed that the probability distribution expression of MLN was unknowable, and used neural networks to automatically learn the expression from logical rules, thereby solving the limitations of manual rulemaking and realizing end-to-end knowledge graph reasoning.

(3) **Few-shot learning and zero-shot learning**

The main challenge for few-shot learning and zero-shot learning is the shortage of training samples, which leads to poor training performances and model over-fitting. The shortage of training samples are caused by the difficulty of obtaining samples or the high cost of labeling, a common phenomenon in many academic fields. In recent years, many algorithms for this problem have emerged. Wang et al. [14] use the idea of knowledge transfer to learn the relationship between seen and unseen classes in the knowledge graph, and they integrate this idea into the learning process of visual features to enhance the classification and generalization capabilities of the model. Kampffmeyer et al. [15] adopts enhanced graph convolution network to learn the semantic representation of the knowledge graph as a constraint of the image classifier, improving zero-shot classification ability. Luo et

al. [41] uses the semantic relationship between seen and unseen classes in the knowledge graph to design a context-aware zero-shot learning method based on the conditional random field. Li et al. [18] propose a knowledge transfer algorithm with class hierarchy, which models the relationships between classes as a tree structure, using the semantic relationships between the source class and the target class to help the model learn more transfer features and achieve large-scale few-shot learning tasks. Sikka et al. [85] integrate common sense knowledge into deep neural networks, and he uses logical knowledge as a new neural-symbolic loss function to constrain visual semantic features, which learns information of unseen class in model learning to improve the ability of zero-shot learning. Altszyler et al. [86] incorporate description logic rules into the neural network framework for multi-domain dialogue recognition tasks, so the model can recognize labels of unseen classes without giving new training data.

(4) **Intelligent question answering**

Intelligent question answering of complex questions is one of the main applications of neural-symbolic reasoning in natural language processing. The task of intelligent question answering is: given a question (maybe a combined question), the model should infer the answer from the context composed of text and images.

For non-synthetic question answering tasks of open-domain text, Gupta et al. [87] extend the neural module networks (NMN) [88]. They introduce reasoning module for text to perform symbolic reasoning (such as arithmetic, sorting, and counting) on numbers and dates in a probabilistic or differentiable way, which can output logic parsing of question and intermediate decisions. It is a neural-symbolic method of explain an end-to-end model.

Hudson et al. [89] propose a fully differentiable network model (MAC) with cyclic memory, attention, and composition functions. The MAC provides strong prior conditions for iterative reasoning, turns the black box architecture to transparency, and supports interpretability and structured learning. The core idea is to decompose the image and the question into sequential units, input the recurrent network for sequential reasoning, and store the result in the memory unit to calculate the final answer together with the question.

For visual question answering tasks, Hudson et al. [90] propose the neural state machine (NSM). NSM uses the supervised training method to construct a probabilistic scene graph based on the concepts in the image, and then performs sequential reasoning on the probabilistic scene graph, answering questions or discovering new conclusions.

(5) **Semantic parsing**

Semantic parsing is to map natural language sentences to logical forms representing the same information. In tasks such as event recognition and machine translation, etc. It usually faces problems such as noise data and missing observations. To solve these problems, it is necessary to introduce external knowledge to fill data or guide model learning.

Tran and Poon et al. [91], [92] model domain common sense with MLN and use probabilistic inference methods for query. Sun et al. [93] learn neural semantic parser and trained a model-agnostic model based on meta-learning to improve the predictive ability of language question-answering tasks in the case of limited simple rules.

(6) **Reinforcement learning**

Deep reinforcement learning is a research hotspot of artificial intelligence and has been applied in many fields. The current deep reinforcement learning has some limitations, especially the lack of reasoning ability. To solve this problem, Garnelo et al. [94] propose a deep symbolic reinforcement learning method (DSRL), which uses the Q algorithm to learn related symbolic priors. In the original deep reinforcement learning, the state is represented as a high-dimensional vector. In DSRL, the state is represented as a low-dimensional symbol, and then the symbol representation is mapped to the corresponding action by maximizing the expected reward value for end-to-end training. DSRL integrates symbolic prior knowledge into the learning process of the agent, enhancing the agent's generalization ability.

Garcez et al. extend DSRL and propose a symbolic reinforcement learning method with common sense(SRL+CS) [95]. This method improves both the learning phase and decision-making phase based on the DRSL method. In the learning phase, to fully distribute the reward, the model no longer set a fixed calculation formula to update the Q value, but update according to the interaction between the agent and the object. In the decision-making stage, to fully aggregate the Q values, the model assigns an importance weight for each Q function according to the distance between the object and the agent.

6 Problems and challenges of neural-symbolic system

The above paper introduces the current research status and research methods of neural-symbolic system in detail. On this basis, this section attempts to analyze the current the neural-symbolic system that what the main challenges are still faced, and what the research direction of the future is.

(1) **Inference method**

When modeling a neural-symbolic system from the perspective of the probability graphs, there are still problems with the reasoning in the calculation stage. For example, in the MLN-based model, if the number of logical rules is large and the size of the entity is large, the number of instantiations will increase exponentially, resulting in a rapid decline in the speed of model inference. For example, if the number of entities is n, and the instantiation of an m-ary predicate requires n^m ways. For this problem, although some improved methods have been proposed [70], [71], there are still limitations. For example, approximate inference, which is usually used, improves the speed of inference at the expense of reducing data usage and sacrificing inference accuracy. Therefore, combining the characteristics of the deep learning model, designing the selected method of rule instantiation, and the corresponding fast reasoning algorithm for statistical relation learning models such as MLN, etc., is a present task to be dealt with.

(2) **Automatic construction of rule**

The rules used in the above neural-symbolic system model are usually constructed manually by domain experts. This construction method is time-consuming, laborious, and not scalable. How to achieve end-to-end learn rules to describe prior knowledge from the data is also a challenge for the neural-symbolic system. At present, people explore and extend ILP-based methods to solve the automatic construction of the rule, but it has problems such as complex reasoning processes, founding simple rules(single-chain rules). Therefore, the automatic construction of rules end-to-end is also a direction that the neural-symbolic system needs to pay attention to in the future.

(3) **Symbolic representation learning**

Good symbolic representation can make seemingly complex learning tasks easier and more efficient. For example, in the zero-shot image classification task, the introduction of symbolic knowledge is essential to improve the classification ability of the model. If learned symbolic representation contains less required semantic information, the classification model will not be competent for complex classification tasks. At present, most symbolic representation methods cannot handle predicates with strong similarity (similar in semantics but different literally). If the two predicates have similar meanings, but the logical formulas are different, the current symbolic representation learning methods cannot capture the same semantics, which damages the model's reasoning ability. Therefore, how to design a more robust and efficient symbolic representation learning method is a significant challenge for the neural-symbolic system. With the development of graph representation learning, nodes are mapped into low-dimensional, dense, and continuous vectors, which can be flexibly embedded in various learning and reasoning tasks. A lot of symbolic knowledge can be modeled as directed or undirected graphs of heterogeneous, multi-relational, and even multi-modal. How to develop and use heterogeneous graph representation learning methods to solve the challenges faced by neural-symbolic systems is also a direction worth exploration.

(4) **Application field expansion**

At present, the neural-symbolic system is mainly applied in the fields of computer vision and natural language processing, resulting in effective achievements. In addition, some works have also explored how to use the neural-symbolic system to solve the interpretability problem of the recommender system [96], [97], [98] and the classification problem [99]. Therefore, a very natural idea is that the neural-symbolic system can effectively solve the problems in which areas, and how to design corresponding models and methods for the commonality and characteristics(particularity).

7 Summary

The neural-symbolic system is considered a bridge between symbolism and connectionism and has received more and more attention in recent years. In this context, this paper surveys the neural-symbolic system's research status, application, and development trend. In short, this paper introduces the overall framework of the neural-symbolic system and analyzes the types and representations of symbols in detail. Taking the combination method as the standard, we classify the existing neural-symbolic systems and elaborate on various neural-symbolic systems' fundamental principles, with the usual methods' technical details. The paper summarizes the main applications of the neural-symbolic system. Although various methods are different in principle and implementation technology, they all demonstrate the necessity, feasibility, and advantages of the combination of symbolic system and neural system. We believe that the neural-symbolic system that organically integrates perception and cognition, learning, and reasoning is the primary trend in developing artificial intelligence. It has significant theoretical and application value and is worthy of more in-depth research and discussion.

Acknowledgments

This research was partly supported by Foundation item: National Natural Science Foundation of China (61876069); Jilin Province Key Scientific and Technological Research and Development Project under Grant Nos. 20180201067GX and 20180201044GX; and Jilin Province Natural Science Foundation (20200201036JC).

References

[1] A. Perotti, G. Boella, S. Colombo Tosatto, A. S. d'Avila Garcez, V. Genovese, and L. van der Torre, "Learning and reasoning about norms using neural-symbolic systems," in *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012*, 2012, pp. 1023–1030.

[2] A. d. Garcez, T. R. Besold, L. De Raedt, P. Földiak, P. Hitzler, T. Icard, K.-U. Kühnberger, L. C. Lamb, R. Miikkulainen, and D. L. Silver, "Neural-symbolic learning and reasoning: contributions and challenges," in *2015 AAAI Spring Symposium Series*, 2015.

[3] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artificial intelligence*, vol. 70, no. 1-2, pp. 119–165, 1994.

[4] T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, D. Lowd, P. M. V. Lima *et al.*, "Neural-symbolic learning and reasoning: A survey and interpretation," *arXiv preprint arXiv:1711.03902*, 2017.

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

[6] N. Kaur, G. Kunapuli, T. Khot, K. Kersting, W. Cohen, and S. Natarajan, "Relational restricted boltzmann machines: A probabilistic logic learning approach," in *International Conference on Inductive Logic Programming*. Springer, 2017, pp. 94–111.

[7] G. Sourek, V. Aschenbrenner, F. Zelezny, S. Schockaert, and O. Kuzelka, "Lifted relational neural networks: Efficient learning of latent relational structures," *Journal of Artificial Intelligence Research*, vol. 62, pp. 69–100, 2018.

[8] A. S. A. Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Applied Intelligence*, vol. 11, no. 1, pp. 59–77, 1999.

[9] P. Dragone, S. Teso, and A. Passerini, "Neuro-symbolic constraint programming for structured prediction," *arXiv preprint arXiv:2103.17232*, 2021.

[10] M. Qu and J. Tang, "Probabilistic logic neural networks for reasoning," *arXiv preprint arXiv:1906.08495*, 2019.

[11] Y. Zhang, X. Chen, Y. Yang, A. Ramamurthy, B. Li, Y. Qi, and L. Song, "Efficient probabilistic logic reasoning with graph neural networks," *arXiv preprint arXiv:2001.11850*, 2020.

[12] Y. Xie, Z. Xu, M. S. Kankanhalli, K. S. Meel, and H. Soh, "Embedding symbolic knowledge into deep networks," *arXiv preprint arXiv:1909.01161*, 2019.

[13] I. Donadello, L. Serafini, and A. D. Garcez, "Logic tensor networks for semantic image interpretation," *arXiv preprint arXiv:1705.08968*, 2017.

[14] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6857–6866.

[15] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing, "Rethinking knowledge graph propagation for zero-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 487–11 496.

[16] Y. Zhu, A. Fathi, and L. Fei-Fei, "Reasoning about object affordances in a knowledge base representation," in *European conference on computer vision*. Springer, 2014, pp. 408–424.

[17] R. Chen, T. Chen, X. Hui, H. Wu, G. Li, and L. Lin, "Knowledge graph transfer network for few-shot recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 575–10 582.

[18] A. Li, T. Luo, Z. Lu, T. Xiang, and L. Wang, "Large-scale few-shot learning: Knowledge transfer with class hierarchy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7212–7220.

[19] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision," *arXiv preprint arXiv:1611.00020*, 2016.

[20] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. B. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," *arXiv preprint arXiv:1810.02338*, 2018.

[21] R. Calegari, G. Ciatto, and A. Omicini, "On the integration of symbolic and sub-symbolic techniques for xai: A survey," *Intelligenza Artificiale*, vol. 14, no. 1, pp. 7–32, 2020.

[22] G. Marra and O. Kuželka, "Neural markov logic networks," *arXiv preprint arXiv:1905.13462*, 2019.

[23] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," *arXiv preprint arXiv:1904.12584*, 2019.

[24] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt, "Deepproblog: Neural probabilistic logic programming," *arXiv preprint arXiv:1805.10872*, 2018.

[25] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou, "Neural logic machines," *arXiv preprint arXiv:1904.11694*, 2019.

[26] G. Marra, F. Giannini, M. Diligenti, and M. Gori, "Integrating learning and reasoning with deep logic models," *arXiv preprint arXiv:1901.04195*, 2019.

[27] Z.-H. Zhou, "Abductive learning: towards bridging machine learning and logical reasoning," *Science China Information Sciences*, vol. 62, no. 7, pp. 1–3, 2019.

[28] R. Evans and E. Grefenstette, "Learning explanatory rules from noisy data," *Journal of Artificial Intelligence Research*, vol. 61, pp. 1–64, 2018.

[29] P. Z. Dos Martires, V. Derkinderen, R. Manhaeve, W. Meert, A. Kimmig, and L. De Raedt, "Transforming probabilistic programs into algebraic circuits for inference and learning," 2019.

[30] A. Kalyan, A. Mohta, O. Polozov, D. Batra, P. Jain, and S. Gulwani, "Neural-guided deductive search for real-time program synthesis from examples," *arXiv preprint arXiv:1804.01186*, 2018.

[31] K. M. Ellis, L. E. Morales, M. Sablé-Meyer, A. Solar Lezama, and J. B. Tenenbaum, "Library learning for neurally-guided bayesian program induction," 2018.

[32] X. Si, M. Raghothaman, K. Heo, and M. Naik, "Synthesizing datalog programs using numerical relaxation," *arXiv preprint arXiv:1906.00163*, 2019.

[33] M. Bošnjak, T. Rocktäschel, J. Naradowsky, and S. Riedel, "Programming with a differentiable forth interpreter," in *International conference on machine learning*. PMLR, 2017, pp. 547–556.

[34] D. Koller, N. Friedman, S. Džeroski, C. Sutton, A. McCallum, A. Pfeffer, P. Abbeel, M.-F. Wong, C. Meek, J. Neville *et al.*, *Introduction to statistical relational learning*. MIT press, 2007.

[35] W. W. Cohen, F. Yang, and K. R. Mazaitis, "Tensorlog: Deep learning meets probabilistic dbs," *arXiv preprint arXiv:1707.05390*, 2017.

[36] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Broeck, "A semantic loss function for deep learning with symbolic knowledge," in *International conference on machine learning*. PMLR, 2018, pp. 5502–5511.

[37] G. Marra, M. Diligenti, F. Giannini, M. Gori, and M. Maggini, "Relational neural machines," *arXiv preprint arXiv:2002.02193*, 2020.

[38] Y. Yang and L. Song, "Learn to explain efficiently via neural logic inductive learning," *arXiv preprint arXiv:1910.02481*, 2019.

[39] M. Diligenti, M. Gori, and C. Sacca, "Semantic-based regularization for learning and inference," *Artificial Intelligence*, vol. 244, pp. 143–165, 2017.

[40] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing, "Harnessing deep neural networks with logic rules," *arXiv preprint arXiv:1603.06318*, 2016.

[41] R. Luo, N. Zhang, B. Han, and L. Yang, "Context-aware zero-shot recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 709–11 716.

[42] L.-W. Cai, W.-Z. Dai, Y.-X. Huang, Y.-F. Li, S. Muggleton, and Y. Jiang, "Abductive learning with ground knowledge base."

[43] H. B. Enderton, *A mathematical introduction to logic*. Elsevier, 2001.

[44] F. Arabshahi, J. Lee, M. Gawarecki, K. Mazaitis, A. Azaria, and T. Mitchell, "Conversational neuro-symbolic commonsense reasoning," *arXiv preprint arXiv:2006.10022*, 2020.

[45] A. Cropper, "Playgol: Learning programs through play," *arXiv preprint arXiv:1904.08993*, 2019.

[46] S. Dumancic, T. Guns, W. Meert, and H. Blockeel, "Learning relational representations with auto-encoding logic programs," *arXiv preprint arXiv:1903.12577*, 2019.

[47] V. Novák, I. Perfilieva, and J. Mockor, *Mathematical principles of fuzzy logic*. Springer Science & Business Media, 2012, vol. 517.

[48] L. Serafini and A. d. Garcez, "Logic tensor networks: Deep learning and logical reasoning from data and knowledge," *arXiv preprint arXiv:1606.04422*, 2016.

[49] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[50] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems*, 2013, pp. 926–934.

[51] P. Domingos and D. Lowd, "Unifying logical and statistical ai with markov logic," *Communications of the ACM*, vol. 62, no. 7, pp. 74–83, 2019.

[52] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[53] A. Darwiche, "On the tractable counting of theory models and its application to truth maintenance and belief revision," *Journal of Applied Non-Classical Logics*, vol. 11, no. 1-2, pp. 11–34, 2001.

[54] A. Darwiche and P. Marquis, "A knowledge compilation map," *Journal of Artificial Intelligence Research*, vol. 17, pp. 229–264, 2002.

[55] A. Darwiche, "New advances in compiling cnf to decomposable negation normal form," in *Proc. of ECAI*. Citeseer, 2004, pp. 328–332.

[56] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[57] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[58] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[59] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Icml*, 2011.

[60] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[61] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International conference on machine learning*. PMLR, 2016, pp. 2071–2080.

[62] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[63] X. Jiang, Q. Wang, and B. Wang, "Adaptive convolution for multi-relational learning," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 978–987.

[64] L. Zhiyuan, S. Maosong, L. Yankai, and X. Ruobing, "Knowledge representation learning: a review," *Journal of Computer Research and Development*, vol. 53, no. 2, p. 247, 2016.

[65] L. De Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic prolog and its application in link discovery." in *IJCAI*, vol. 7. Hyderabad, 2007, pp. 2462–2467.

[66] P. Singla and P. Domingos, "Discriminative training of markov logic networks," in *AAAI*, vol. 5, 2005, pp. 868–873.

[67] L. Mihalkova and R. J. Mooney, "Bottom-up learning of markov logic network structure," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 625–632.

[68] P. Singla and P. Domingos, "Memory-efficient inference in relational domains," in *AAAI*, vol. 6, 2006, pp. 488–493.

[69] H. Poon and P. Domingos, "Sound and efficient inference with probabilistic and deterministic dependencies," in *AAAI*, vol. 6, 2006, pp. 458–463.

[70] T. Khot, S. Natarajan, K. Kersting, and J. Shavlik, "Learning markov logic networks via functional gradient boosting," in *2011 IEEE 11th international conference on data mining*. IEEE, 2011, pp. 320–329.

[71] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor, "Hinge-loss markov random fields and probabilistic soft logic," *arXiv preprint arXiv:1505.04406*, 2015.

[72] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek, "Fast rule mining in ontological knowledge bases with amie + +," *The VLDB Journal*, vol. 24, no. 6, pp. 707–730, 2015.

[73] A. Campero, A. Pareja, T. Klinger, J. Tenenbaum, and S. Riedel, "Logical rule induction and theory learning using neural theorem proving," *arXiv preprint arXiv:1809.02193*, 2018.

[74] T. Rocktäschel and S. Riedel, "End-to-end differentiable proving," *arXiv preprint arXiv:1705.11040*, 2017.

[75] A. Payani and F. Fekri, "Inductive logic programming via differentiable deep neural logic networks," *arXiv preprint arXiv:1906.03523*, 2019.

[76] P. Minervini, T. Demeester, T. Rocktäschel, and S. Riedel, "Adversarial sets for regularising neural link predictors," *arXiv preprint arXiv:1707.07596*, 2017.

[77] T. Chen, R. Chen, L. Nie, X. Luo, X. Liu, and L. Lin, "Neural task planning with and–or graph representations," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 1022–1034, 2018.

[78] Z. Liu, Z. Jiang, and F. Wei, "Od-gcn object detection by knowledge graph with gcn," *arXiv preprint arXiv:1908.04385*, 2019.

[79] A. Darwiche, "Sdd: A new canonical representation of propositional knowledge bases," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[80] J. Zhong, W. Haoran, Y. Yunlong, and P. Yanwei, "A decadal survey of zero-shot image classification," *SCIENTIA SINICA Informationis*, vol. 49, no. 10, pp. 1299–1320, 2019.

[81] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[82] M. Marszalek and C. Schmid, "Semantic hierarchies for visual object recognition," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–7.

[83] G. Forestier, C. Wemmert, and A. Puissant, "Coastal image interpretation using background knowledge and semantics," *Computers & Geosciences*, vol. 54, pp. 88–96, 2013.

[84] D. Nyga, F. Balint-Benczedi, and M. Beetz, "Pr2 looking at things—ensemble learning for unstructured information processing with markov logic networks," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3916–3923.

[85] K. Sikka, J. Huang, A. Silberfarb, P. Nayak, L. Rohrer, P. Sahu, J. Byrnes, A. Divakaran, and R. Rohwer, "Zero-shot learning with knowledge enhanced visual semantic embeddings," *arXiv preprint arXiv:2011.10889*, 2020.

[86] E. Altszyler, P. Brusco, N. Basiou, J. Byrnes, and D. Vergyri, "Zero-shot multi-domain dialog state tracking using descriptive rules," *arXiv preprint arXiv:2009.13275*, 2020.

[87] N. Gupta, K. Lin, D. Roth, S. Singh, and M. Gardner, "Neural module networks for reasoning over text," *arXiv preprint arXiv:1912.04971*, 2019.

[88] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural module networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 39–48.

[89] D. A. Hudson and C. D. Manning, "Compositional attention networks for machine reasoning," *arXiv preprint arXiv:1803.03067*, 2018.

[90] ——, "Learning by abstraction: The neural state machine," *arXiv preprint arXiv:1907.03950*, 2019.

[91] S. D. Tran and L. S. Davis, "Event modeling and recognition using markov logic networks," in *European Conference on Computer Vision*. Springer, 2008, pp. 610–623.

[92] H. Poon and P. Domingos, "Unsupervised semantic parsing," in *Proceedings of the 2009 conference on empirical methods in natural language processing*, 2009, pp. 1–10.

[93] Y. Sun, D. Tang, N. Duan, Y. Gong, X. Feng, B. Qin, and D. Jiang, "Neural semantic parsing in low-resource settings with back-translation and meta-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8960–8967.

[94] M. Garnelo, K. Arulkumaran, and M. Shanahan, "Towards deep symbolic reinforcement learning," *arXiv preprint arXiv:1609.05518*, 2016.

[95] A. d. Garcez, A. R. R. Dutra, and E. Alonso, "Towards symbolic reinforcement learning with common sense," *arXiv preprint arXiv:1804.08597*, 2018.

[96] A. Menk, L. Sebastia, and R. Ferreira, "Recommendation systems for tourism based on social networks: A survey," *arXiv preprint arXiv:1903.12099*, 2019.

[97] Y. Zhu, Y. Xian, Z. Fu, G. de Melo, and Y. Zhang, "Faithfully explainable recommendation via neural logic reasoning," *arXiv preprint arXiv:2104.07869*, 2021.

[98] Z. Fu, Y. Xian, Y. Zhu, S. Xu, Z. Li, G. de Melo, and Y. Zhang, "Hoops: Human-in-the-loop graph reasoning for conversational recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2415–2421.

[99] A. Daniele and L. Serafini, "Neural networks enhancement through prior logical knowledge," *arXiv preprint arXiv:2009.06087*, 2020.