Title
Close the Loop of Neural Perception, Grammar Parsing, and Symbolic Reasoning

Author
Li, Qing

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Close the Loop of Neural Perception, Grammar Parsing, and Symbolic Reasoning

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Qing Li

2022

ABSTRACT OF THE DISSERTATION

Close the Loop of Neural Perception, Grammar Parsing, and Symbolic Reasoning

by

Qing Li
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2022
Professor Song-Chun Zhu, Chair

Despite the recent remarkable advances in deep learning, we are still far from building machines with human-like general intelligence, for instance, understanding the world in a fast, structured, and generalizable way. The dominant stream in contemporary AI hopes to achieve human-level performance via purely data-driven methods, *i.e.*, fitting deep neural networks on a massive amount of training data. However, these methods are often trapped in a dilemma of "big data, small tasks", and are hard to interpret and generalize.

In this dissertation, we seek a unified framework for general intelligence by integrating connectionism and symbolism in a neuro-symbolic system. We argue that (i) **Neural Network** is excellent at imitating human perception from raw signals, (ii) **Grammar** provides a universal approach to construct a holistic structured representation of the world, and (iii) **Symbolic Reasoning** forms a principled basis to incorporate commonsense knowledge and perform complex reasoning. Therefore, we propose a neural-symbolic framework by using grammar as the bridge to connect neural networks and symbolic reasoning. The learning of such a neural-symbolic framework mimics human's ability to learn from failures via abductive reasoning and requires very little supervision. We have developed benchmarks, algorithms,

and practices, across vision and language, from synthetic environments to real-world scenarios, to realize such a unified framework. We hope such a unified framework can contribute to the long-term goal of building general artificial intelligence like humans.

The dissertation of Qing Li is approved.

Ying Nian Wu

Tao Gao

Hongjing Lu

Song-Chun Zhu, Committee Chair

University of California, Los Angeles

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I give my first and deepest gratitude to my advisor, Professor Song-Chun Zhu, for introducing me to the field of neural-symbolic AI, showing me the most visionary pictures of AI, and challenging me to purse the most fundamental and ambiguous goals. Without his insightful ideas and constructive suggestions, this dissertation would have been nowhere near possible.

I feel very fortunate to work with Professor Ying Nian Wu, who is the kindest person I have ever met. He showed me what a pure researcher is and how to be a scientist devoted to the fundamental problems. His pursuit for simplicity and elegance of research encourages me to develop simple but effective models all the time.

My committee members, Prof. Hongjing Lu and Prof. Tao Gao, also gave me tremendous help along the journey. They show me the fantastic world of human cognition and provide refreshing views of true intelligence.

I feel very grateful to work with the very talented people at VCLA. Among all the students I collaborated with, I thank Siyuan Huang the most. Siyuan has always been a great support for all my research and also helped me addressed many non-research problems. I thank Yining Hong for her excellent work in math word problems and vision-language understanding. I thank Yixin Chen for his pioneering efforts in human-robot communication. I thank Daniel Ciao and Ran Gong for their hard work on math word problems.

I enjoyed the time at UCLA so much. People here gave me a feeling of home and brought tons of joy to my life through parties, travels, sports, and food. I thank my friends Feng Gao, Baoxiong Jia, Yuxing Qiu, Tao Yuan, Chi Zhang, Sirui Xie, XiaoJian Ma, Yizhou Zhao, Xu Xie, Tengyu Liu, *etc.* I played with them and learned from them.

Finally, I thank my parents for their selfless love and support to me. I thank my girlfriend, Ziqi Zhu, for her company and support along this wonderful journey. I thank my cats, Momo and Pipi, for bringing countless joy and comfort through the hard times of pandemic.

VITA

2018-2022	Graduate Research Assistant, Department of Statistics, UCLA.
2021-2022	Dissertation Year Fellowship, UCLA.
2021	Research Intern, Google Research.
2021	Research Intern, Microsoft.
2020	Applied Scientist Intern, Amazon.
2018	M.S. in Electronic Engineering, University of Science and Technology of China.
2015	B.E. in Automation, University of Science and Technology of China.

PUBLICATIONS

Yixin Chen, **Qing Li**, Deqian Kong, Yik Lun Kei, Siyuan Huang, Yixin Zhu, Song-Chun Zhu. "*YouRefIt: Embodied Reference Understanding with Language and Gesture*." ICCV'21.

Yining Hong, **Qing Li**, Siyuan Huang, Song-Chun Zhu. "*VLGrammar: Grounded Grammar Induction of Vision and Language*." ICCV'21.

Qing Li, Siyuan Huang, Yining Hong, Song-Chun Zhu. "*A HINT from Arithmetic: On Systematic Generalization of Perception, Syntax, and Semantics*." ICLR'21 MATH-AI Workshop.

Yining Hong, **Qing Li**, Daniel Ciao, Siyuan Huang, Song-Chun Zhu. "*Learning by Fixing: Solving Math Word Problems with Weak Supervision.*" AAAI'21.

Yining Hong, **Qing Li**, Ran Gong, Daniel Ciao, Siyuan Huang, Song-Chun Zhu. "*SMART: A Situation Model for Algebra Story Problems via Attributed Grammar.*" AAAI'21.

Qing Li, Boqing Gong, Yin Cui, Dan Kondratyuk, Xianzhi Du, Ming-Hsuan Yang, Matthew Brown. "*Towards a Unified Foundation Model: Jointly Pre-Training Transformers on Unpaired Images and Text.*" Under review, arXiv'21.

Qing Li, Siyuan Huang, Yining Hong, Song-Chun Zhu. "*A Competence-aware Curriculum for Visual Concepts Learning via Question Answering.*" ECCV'20.

Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, Song-Chun Zhu. "*Closed Loop Neural-Symbolic Learning via Integrating Neural Perception, Grammar Parsing, and Symbolic Reasoning.*" ICML'20. (**Best paper award** in ICML'20 Workshop on Bridge Between Perception and Reasoning: Graph Neural Networks & Beyond.)

Nilavra Bhattacharya, **Qing Li**, Danna Gurari. "*Why Does a Visual Question Have Different Answers?*" ICCV'19.

Danna Gurari, **Qing Li**, Chi Lin, Yinan Zhao, Anhong Guo, Abigale J. Stangl, and Jeffrey P. Bigham. "*VizWiz-Priv: A Dataset for Recognizing the Presence and Purpose of Private Visual Information in Images Taken by Blind People.*" CVPR'19.

CHAPTER 1

Introduction

Humans demonstrate superior capabilities at learning and reasoning over the physical world. Specifically, humans can (i) quickly learn to recognize unseen objects and events from a limited amount of data, (2) infer the relationships between individual observations and construct a holistic understanding of the world, (3) perform complex reasoning over the perceived environments to solve a variety of cognitive tasks, and (4) generalize the learned concepts to novel domains and environments. How can we build a machine that possesses similar capabilities of learning, understanding, and reasoning like humans?

Recent advances in deep learning have achieved remarkable results on perceptual tasks such as image classification, speech recognition, and machine translation. However, it is widely recognized that there still exists a huge gap between perception and cognition to be bridged, in order to develop truly intelligent systems. Highly cognitive tasks such as reasoning, planning, and explaining are typically associated with symbolic systems which do not scale to the high-dimensional signals from the physical world.

To tackle this challenge and build human-like general intelligence, we seek a unified framework by integrating neural network, grammar, and logic in a neuro-symbolic system, as exemplified by Fig. 1.1. This unified framework combines the strengths of deep learning with symbolic approaches, by using the former to learn low-dimensional representations which significantly reduces the search space for symbolic approaches. Another justification for such a neuro-symbolic framework is related to human learning. While far from fully understood, there is an increasing body of evidence that similar mechanisms combining low-

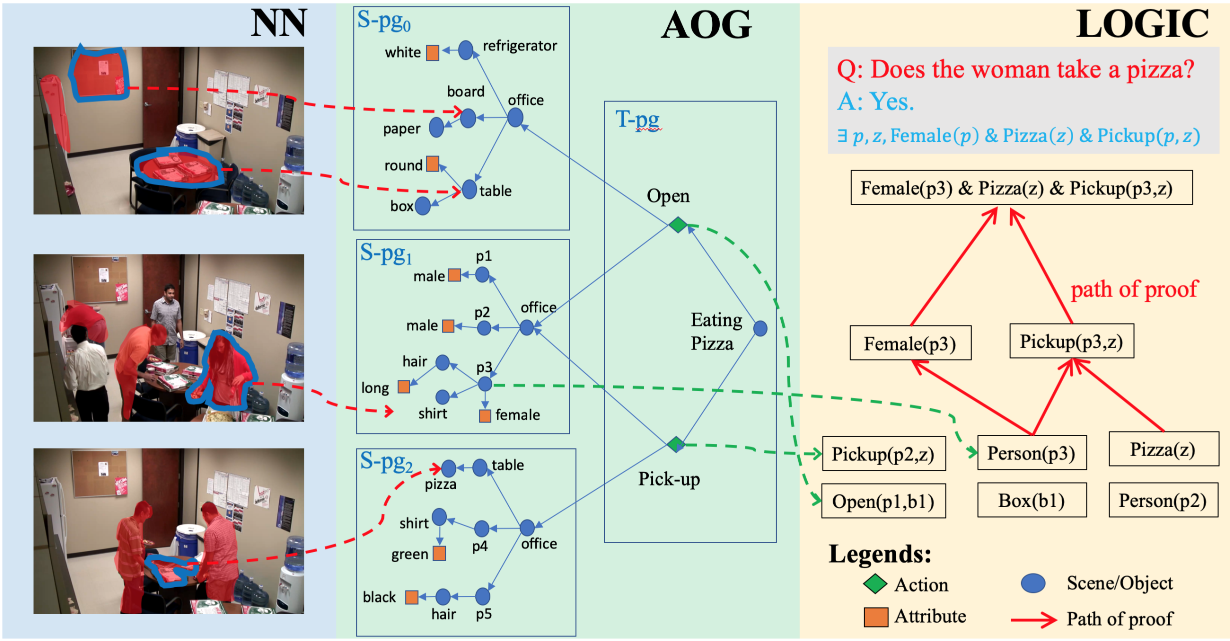


Figure 1.1: **Human-like understanding and reasoning via a neuro-symbolic framework.** This framework consists of three layers: (i) **Neural Network (NN)** is used as a perception module to recognize objects, attributes, and actions from the video; (ii) **Grammar (AOG)** is adopted to organize the outputs of the neural network in a hierarchical structure; (iii) **Logic** is a formal method to perform reasoning and answer a specific question.

level perception with high-level cognition are at play in the human brain.

Specifically, we describe the background, problems, challenges, and our proposed benchmarks and algorithms to study this direction in the following sections.

1.1 A Tale of Recognition and Cognition

Both pattern recognition and cognitive reasoning have a long history in the field of artificial intelligence. Next, we briefly discuss each of them, respectively.

A modern definition of pattern recognition [BN06] is: "The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories." For example, how to classify an image is a popular task of pattern recognition in the context of computer vision. Pattern recognition originated from statistics and has undergone substantial development over the past few decades. Modern approaches to pattern recognition are mainly the use of machine learning, particularly deep learning [LBH15], due to the increased availability of big data and a new abundance of processing power.

Reasoning is the capability of drawing logical conclusions from existing information, with the aim of seeking the truth of an unknown statement. In the field of psychology and philosophy, reasoning is normally considered to be a distinguishing ability possessed by humans and highly related to humans' mind and cognition [MS17]. In the past decades, tremendous efforts have been made by the community of artificial intelligence to realize human-like automated reasoning in an artificial system, such as automated theorem proving [Fit12] and expert systems [Jac86]. An expert system is a computer system designed to emulate the decision-making ability of a human expert by reasoning through bodies of knowledge, represented mainly as if-then rules rather than through conventional procedural code.

Although both pattern recognition and automated reasoning have been well studied separately and are able to solve plenty of tasks on their own, either of them alone is insufficient to truly achieve human-like general intelligence. Therefore, researchers have devoted a lot of efforts to integrating pattern recognition and automated reasoning, which is also known as neural-symbolic integration [GBD15, BGB17b]. Neural-symbolic integration seeks to integrate principles from neural-networks learning and logical reasoning. It is an interdisciplinary field involving components of knowledge representation, neuroscience, machine learning, and cognitive science [GBD15]. Neural-symbolic integration provides an AI system capable of bridging lower-level information processing (for perception and pattern recognition) and higher-level abstract knowledge (for reasoning and explanation). Such neural-symbolic systems have shown promise in various applications, such as fault diagnosis, computational biology, training and assessment in simulators, and software verification [BGB17b].

However, it is very challenging to seamlessly integrate pattern recognition and automated reasoning in a desirable way [BGB17b]. When building neural-symbolic models, we have to conciliate the methodologies of distinct areas – namely neural networks and logical reasoning – in order to combine the respective advantages and circumvent the shortcomings and limitations. In particular, it involves how principles of symbolic computation can be implemented by connectionist mechanisms and how sub-symbolic computation can be described and analyzed in logical terms.

1.2 Closing the Loop of Recognition and Reasoning

In this dissertation, we discuss the integration of pattern recognition and automated reasoning from a unique viewpoint and close the loop of recognition and cognition from a statistical learning perspective. We first introduce a novel benchmark to study this problem, then discuss the representations, modeling, and learning, and conclude with several real-world applications.

1.2.1 Benchmark: A HINT from Arithmetic

Humans possess a versatile mechanism for learning concepts [FS16]. Take the arithmetic examples in Fig. 3.1: When we master concepts like digits and operators, we not only know how to recognize, write, and pronounce them—what these concepts mean at the *perceptual* level, but also know how to compose them into valid expressions—at the *syntactic* level, and how to calculate the results by reasoning over these concepts—at the *semantic* level. Learning concepts heavily rely on these three-level interweaving meanings. Such observation also conforms with the classic view of human cognition, which postulates at least three distinct levels of organizations in computation systems [Pyl84, FP88].

Crucially, a unique property of human concept learning is its systematic generalization. Once we master the syntax of arithmetic using short expressions, we can parse novel, long expressions. Similarly, once we master operators' semantics using small numbers, we can apply them to large numbers. This property corresponds to the classic idea of the *system-*

aticity (interpolation) and *productivity* (extrapolation) in cognition: An infinite number of representations can be constructed from a finite set of primitives, just as the mind can think an infinite number of thoughts, understand an infinite number of sentences, or learn new concepts from a seemingly infinite space of possibilities [LUT17, Mar18, Fod75].

To examine the versatile human-like capabilities of concept learning with a focus on systematic generalization, we take inspiration from arithmetic and introduce a new benchmark HINT, Handwritten arithmetic with INTegers [LHH21]. The task of HINT is intuitive and straightforward: Machines take as input images of handwritten expressions and predict the final results of expressions, restricted in the integer space. The task of HINT is also challenging: Concepts in HINT, including digits and operators, are learned in a weakly-supervised manner. Using final results as the only supervision, machines are tasked to learn the three-level meanings simultaneously—perception, syntax, and semantics of these concepts—to correctly predict the results. Since there is no supervision on any intermediate values or representations, the three-level meanings are presumably intertwined during learning. To provide a holistic and rigorous test on whether learning machines can generalize the learned concepts, we introduce a carefully designed evaluation scheme instead of using a typical i.i.d. test split. This new scheme includes five subsets, focusing on generalization capabilities (*i.e.*, interpolation and extrapolation) at different levels of meanings (*i.e.*, perception, syntax, and semantics).

1.2.2 Representation, Modeling, and Learning

To build a system that is able to integrate recognition and reasoning, we need to consider various aspects including representation, modeling, and learning.

1.2.2.1 Representation: Connectionism v.s. Symbolism

First, what representation should we adopt to bridge recognition and reasoning? Basically, we can choose the representation from two paradigms: connectionism or symbolism. Central to connectionism is *distributed representations* [Hin84]. In a connectionist network, a distributed representation indicates that some concept or meaning is represented by a pattern of activity across a number of processing units, a.k.a. neurons. We usually adopt fixed-dimensional continuous vectors as distributed representations and thus can implement the whole system as an end-to-end neural network. Such an end-to-end neural network is a homogeneous model and can be optimized very fast via GPUs in practice. It is easy to transfer neural models and algorithms to other domains because they usually require little domain knowledge. Distributed representations also make the learning of neural networks robust to noisy inputs and robustness is a highly desirable property in pattern recognition, because real-world signals, like images and speeches, usually include a lot of noise. However, distributed representations have been shown to be insufficient for cognitive reasoning tasks that require systematic generalization [LB18]. Another disadvantage of distributed representations is that they make the internal structure of a trained network very difficult to interpret, because the meaning is associated with a group activity of neurons, instead of single ones. The uninterpretable nature of distributed representations makes it nearly impossible to inject prior domain knowledge into the neural network as well as hard to diagnose wrong predictions from the model.

The other choice of representation is a *physical symbol system* [NS07] (also called a formal system) adopted by symbolism. A physical symbol system takes physical patterns (symbols), combines them into structures (expressions), and manipulates them (using processes) to produce new expressions. In contrast to distributed representations, each symbol alone in a symbol system represents an atomic concept or meaning and more complicated concepts are formed by combining multiple symbols in a certain syntax. Besides, symbol systems are more interpretable and support stronger abstraction and generalization than distributed representations. However, building a symbol system for a domain requires strong domain-specific knowledge and the built system is usually fragile and inflexible.

The physical symbol system hypothesis [NS07] claims that "a physical symbol system has the necessary and sufficient means for general intelligent action." This claim implies both that human thinking is a kind of symbol manipulation (because a symbol system is necessary for intelligence) and that machines can be intelligent (because a symbol system is sufficient for intelligence). This hypothesis is a core part of AI research in the last century, but it has been criticized strongly by various parties. A common critical view is that the hypothesis seems appropriate for higher-level intelligence such as playing chess, but less appropriate for commonplace intelligence such as vision.

In this dissertation, we adopt a symbol system as the internal representation. Particularly, the atomic symbols are grounded in the input raw signals, and the syntax and the semantics of the symbol system are learned from the provided examples.

1.2.2.2 Neural-Grammar-Symbolic Model

In this dissertation, we propose a novel Neural-Grammar-Symbolic (NGS) model [LHH20a] for the integration of pattern recognition and automated reasoning. Particularly, we introduce a grammar parsing model to bridge neural perception and logical reasoning. Next, we will briefly discuss the proposed NGS model in the context of HINT.

Neural Perception A neural network is used as a perception module that maps a high-dimensional input to a normalized probability distribution of the hidden symbolic sequence. The distributed representation learned by the neural network makes the model robust to noise in the raw inputs.

Grammar Parsing While neural networks are powerful at modeling the mapping from raw inputs to atomic symbols. Grammar is a natural choice to model the compositional and recursive properties in sequence data. A grammar model is supposed to parse a sequence of symbols into a structured representation like a parse tree.

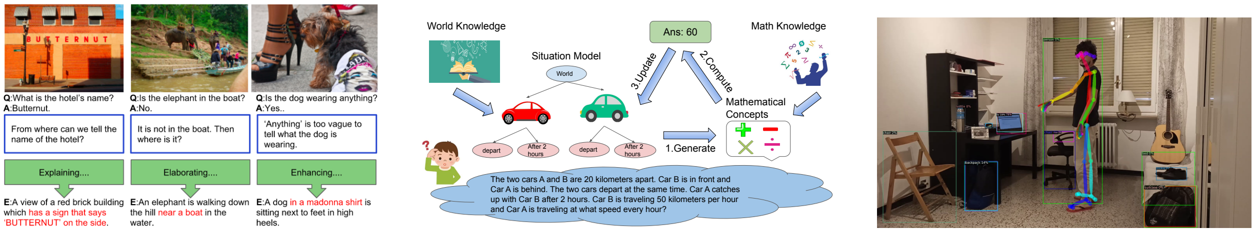
Symbolic Reasoning Given the structured representation, a symbolic reasoning module performs deterministic inference using the background knowledge to infer the final predictions. The inference rules generate a reasoning path that leads to the predicted output from the structured representation and the used background knowledge.

1.2.2.3 Learning by Deduction-Abduction

Tasks like HINT usually provide weak supervision for learning, which means that we only observe the raw inputs and the final outputs, with the intermediate symbolic representations being hidden. In the proposed NGS model, the model spaces for different modules are heterogeneous, *e.g.*, the perception module is a continuous neural network using distributed representations while the reasoning module might use discrete logic or programs. Therefore, it is infeasible to perform an end-to-end optimization for such a heterogeneous model.

To address this optimization issue, we derive a general learning framework from a probabilistic perspective and it turns out that the key to learning a heterogeneous model with weak supervision is to perform statistical sampling from the posterior distribution of the intermediate symbolic representations given the raw inputs and the final supervision in the maximum likelihood estimation.

Sampling from three heterogeneous spaces is not easy. The first natural choice is to use rejection sampling. The rejection sampling method first generates a sample from a candidate distribution formed by the model output and then decides whether or not to keep the sample based on the posterior probability of this sample. This method is conceptually simple but very inefficient in practice. The latent space of the intermediate symbolic representations is very large and most of it has zero probability in the posterior distribution. Thus the optimization is very time-consuming since it requires generating a huge number of samples over the latent space, in the hope that some samples may be lucky enough to hit the non-



(a) Visual Reasoning (b) Math Word Problems (c) Reference Understanding

Figure 1.2: A wide range of applications of closing the loop of recognition and reasoning, across vision and language.

zero regions in the posterior distribution. In practice, the learning based on the rejection sampling converges slowly or even fails to converge without pre-training the neural perception module. We also find out that when only considering the learning of the neural perception, the rejection sampling method coincides with the REINFORCE algorithm, which is one of the popular policy gradient methods of reinforcement learning and widely used in previous neural-symbolic models.

Inspired by the human ability to learn from failures via abductive reasoning [Mag09, Zho19a], we propose a novel *deduction-abduction* [LHH20a, LHH21] strategy to coordinate the learning of three heterogeneous modules in the proposed model. Specifically, during learning, the system first performs greedy deduction over these modules to propose an initial, rough solution, which is likely to produce a wrong result. Abduction over the three heterogeneous spaces is then applied in a top-down manner to search the initial solution's neighborhood, which updates the solution to explain the ground-truth result better. This revised solution provides *pseudo* supervision on the intermediate values and representations, which are then used to train each module individually. The deduction-abduction strategy makes the learning much more efficient than the rejection sampling method. We prove that the multi-step abduction process behaves as a Metropolis-Hastings sampler for the posterior distribution of the intermediate symbolic representations.

1.2.3 Various Applications

To demonstrate the practical values of the proposed framework for closing the loop of recognition and reasoning, we apply it for several applications in different domains across vision and language, including visual reasoning [LHH20c, LFY18], math word problems [HLC21, HLG21], embodied reference understanding [CLK21], and grounded grammar induction [HLZ21], as exemplified in Fig. 1.2.

1.3 Contributions

This dissertation aims to close the loop of recognition and reasoning from a statistical learning perspective and it is mainly addressed from the aforementioned three perspectives: benchmarks, models, and applications.

The contributions of our work can be summarized as follows:

- **A new benchmark:** Handwritten arithmetic with INTegers (HINT). This benchmark is simple and effective to study various aspects of the integration of recognition and reasoning. It provides us a test-bed to lay the theoretical foundation for closing the loop of recognition and reasoning from a statistical learning perspective.

- **A new model:** Neural-Grammar-Symbolic (NGS). We propose a grammar parsing module to bridge neural perception and symbolic reasoning. The proposed NGS model is an implementation of a symbol system with combinatorial syntactic and semantic structures, which is arguably a necessary and sufficient means of general intelligence.

- **A new learning strategy:** Deduction-Abduction. Inspired by the human ability to learn from failures, we derive a novel deduction-abduction strategy to conciliate the joint optimization of three heterogeneous modules, which makes the learning much faster and more data-efficient.

- **Applications:** visual reasoning, math word problems, embodied reference understanding, and grounded grammar induction. We apply the proposed model and learning method for these applications and obtain promising results compared with the prior methods.

In the following chapters, we introduce more details about these contributions. In the last chapter, we conclude this dissertation by summarizing our work and discussing potential directions for future research in this exciting area.

CHAPTER 2

Bridging Neural Perception and Symbolic Reasoning via Grammar Parsing

The goal of neural-symbolic computation is to integrate the connectionist and symbolist paradigms. Prior methods learn the neural-symbolic models using reinforcement learning (RL) approaches, which ignore the error propagation in the symbolic reasoning module and thus converge slowly with sparse rewards. In this chapter, we address these issues and close the loop of neural-symbolic learning by (1) introducing the **grammar** model as a *symbolic prior* to bridge neural perception and symbolic reasoning, and (2) proposing a novel **back-search** algorithm which mimics the top-down human-like learning procedure to propagate the error through the symbolic reasoning module efficiently. We further interpret the proposed learning framework as maximum likelihood estimation using Markov chain Monte Carlo sampling and the back-search algorithm as a Metropolis-Hastings sampler. The experiments are conducted on two weakly-supervised neural-symbolic tasks: (1) handwritten formula recognition on the newly introduced HWF dataset; (2) visual question answering on the CLEVR dataset. The results show that our approach significantly outperforms the RL methods in terms of performance, converging speed, and data efficiency.

2.1 Introduction

Integrating robust connectionist learning and sound symbolic reasoning is a key challenge in modern Artificial Intelligence. Deep neural networks [LBH15, LB95, HS97] provide us

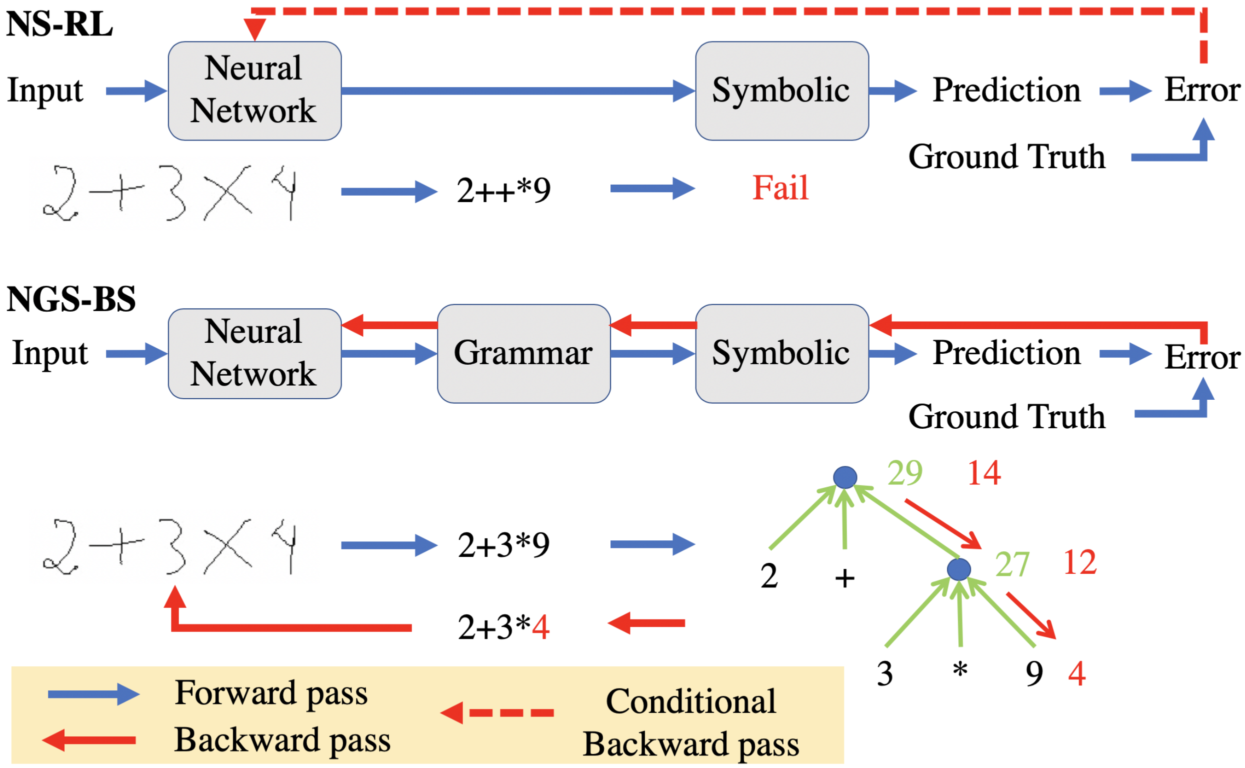


Figure 2.1: Comparison between the original neural-symbolic model learned by REINFORCE (NS-RL) and the proposed neural-grammar-symbolic model learned by back-search (NGS-BS). In NS-RL, the neural network predicts an invalid formula, causing a failure in the symbolic reasoning module. There is no backward pass in this example since it generates zero reward. In contrast, NGS-BS predicts a valid formula and searches a correction for its prediction. The neural network is updated using this correction as the pseudo label.

powerful and flexible representation learning that has achieved state-of-the-art performances across a variety of AI tasks such as image classification [KSH12, SLJ15, HZR16], machine translation [SVL14], and speech recognition [GMH13]. However, it turns out that many aspects of human cognition, such as systematic compositionality and generalization [FP88, Mar98, FL02, CS14, Mar18, LB18], cannot be captured by neural networks. On the other hand, symbolic reasoning supports strong abstraction and generalization but is fragile and inflexible. Consequently, many methods have focused on building neural-symbolic models to combine the best of deep representation learning and symbolic reasoning [Sun94, GLG08, BGH09, BGB17b, YWG18].

Recently, this neural-symbolic paradigm has been extensively explored in the tasks of the visual question answering (VQA) [YWG18, VDL19, MGK19], vision-language navigation [AWT18, FHC18], embodied question answering [DDG18, DGL18], and semantic parsing [LBL16a, YZH18], often with weak supervision. Concretely, for these tasks, neural networks are used to map raw signals (images/questions/instructions) to symbolic representations (scenes/programs/actions), which are then used to perform symbolic reasoning/execution to generate final outputs. Weak supervision in these tasks usually provides pairs of raw inputs and final outputs, with intermediate symbolic representations unobserved. Since symbolic reasoning is non-differentiable, previous methods usually learn the neural-symbolic models by policy gradient methods like REINFORCE. The policy gradient methods generate samples and update the policy based on the generated samples that happen to hit high cumulative rewards. No efforts are made to improve each generated sample to increase its cumulative reward. Thus the learning has been proved to be time-consuming because it requires generating a large number of samples over a large latent space of symbolic representations with sparse rewards, in the hope that some samples may be lucky enough to hit high rewards so that such lucky samples can be utilized for updating the policy. As a result, policy gradients methods converge slowly or even fail to converge without pre-training the neural networks on fully-supervised data.

To model the recursive compositionality in a sequence of symbols, we introduce the **grammar** model to bridge neural perception and symbolic reasoning. The structured symbolic representation often exhibits compositional and recursive properties over individual symbols in it. Correspondingly, the grammar models encode *symbolic prior* about composition rules, thus can dramatically reduce the solution space by parsing the sequence of symbols into valid sentences. For example, in the handwritten formula recognition problem, the grammar model ensures that the predicted formula is always valid, as shown in Figure 2.1.

To make the neural-symbolic learning more efficient, we propose a novel **back-search** strategy which mimics human's ability to learn from failures via abductive reasoning [Mag09, Zho19a]. Specifically, the back-search algorithm propagates the error from the root node to the leaf nodes in the reasoning tree and finds the most probable *correction* that can generate the desired output. The correction is further used as a pseudo label for training the neural network. Figure 2.1 shows an exemplar backward pass of the back-search algorithm. We argue that the back-search algorithm makes a first step towards closing the learning loop by propagating the error through the non-differentiable grammar parsing and symbolic reasoning modules. We also show that the proposed multi-step back-search algorithm can serve as a Metropolis-Hastings sampler which samples the posterior distribution of the symbolic representations in the maximum likelihood estimation in subsubsection 2.3.2.3.

We conduct experiments on two weakly-supervised neural-symbolic tasks: (1) handwritten formula recognition on the newly introduced HWF dataset (Hand-Written Formula), where the input image and the formula result are given during training, while the formula is hidden; (2) visual question answering on the CLEVR dataset. The question, image, and answer are given, while the functional program generated by the question is hidden. The evaluation results show that the proposed Neural-Grammar-Symbolic (NGS) model with back-search significantly outperforms the baselines in terms of performance, convergence speed, and data efficiency. The ablative experiments also demonstrate the efficacy of the multi-step back-search algorithm and the incorporation of grammar in the neural-symbolic model.

2.2 Related Work

Neural-symbolic Integration. Researchers have proposed to combine statistical learning and symbolic reasoning in the AI community, with pioneer works devoted to different aspects including representation learning and reasoning [Sun94, GLG08, MDK18], abductive learning [DZ17, DXY19, Zho19a], knowledge abstraction [HOT06, BGH09], knowledge transfer [FFG89, YCX09], *etc.* Recent research shifts the focus to the application of neural-symbolic integration, where a large amount of heterogeneous data and knowledge descriptions are needed, such as neural-symbolic VQA [YWG18, VDL19, MGK19, LFY18, LTJ18, LHH20c], semantic parsing in Natural Language Processing (NLP) [LBL16a, YZH18], math word problem [LC19, LSR19] and program synthesis [EG18, KMP18, MDK18]. Different from previous methods, the proposed NGS model considers the compositionality and recursivity in natural sequences of symbols and brings together the neural perception and symbolic reasoning module with a grammar model.

Grammar Model. Grammar model has been adopted in various tasks for its advantage in modeling compositional and recursive structures, like image parsing [TCY05, HZ05, ZM07, ZZ11, FD18], video parsing [GSS09, QJZ18, QJH20], scene understanding [HQZ18, HQX18, QZH18, JQZ18, CHY19], and task planning [XLE18]. By integrating the grammar into the neural-symbolic task as a symbolic prior for the first time, the grammar model ensures the desired dependencies and structures for the symbol sequence and generates valid sentences for symbolic reasoning. Furthermore, it improves the learning efficiency significantly by shrinking the search space with the back-search algorithm.

Policy Gradient. Policy gradient methods like REINFORCE [Wil92] are the most commonly used algorithm for the neural-symbolic tasks to connect the learning gap between neural networks and symbolic reasoning [MTS18, MGK19, AKL17, DGL18, BHD18, GPL17a]. However, original REINFORCE algorithm suffers from large sample estimate variance, sparse rewards from cold start and exploitation-exploration dilemma, which lead to unstable learning dynamics and poor data efficiency. Many papers propose to tackle this problem [LBL16a, GPL17a, LNB18b, WZG18, ALS19a]. Specifically, [LBL16a] uses iterative maximum likelihood to find pseudo-gold symbolic representations, and then add these representations to the REINFORCE training set. [GPL17a] combines the systematic beam search employed in maximum marginal likelihood with the greedy randomized exploration of REINFORCE. [LNB18b] proposes Memory Augmented Policy Optimization (MAPO) to express the expected return objective as a weighted sum of an expectation over the high-reward history trajectories, and a separate expectation over new trajectories. Although utilizing positive representations from either beam search or past training process, these methods still cannot learn from negative samples and thus fail to explore the solution space efficiently. On the contrary, we propose to diagnose and correct the negative samples through the back-search algorithm under the constraint of grammar and symbolic reasoning rules. Intuitively speaking, the proposed back-search algorithm traverses around the negative sample and find a nearby positive sample to help the training.

2.3 Neural-Grammar-Symbolic Model (NGS)

In this section, we will first describe the inference and learning algorithms of the proposed neural-grammar-symbolic (NGS) model. Then we provide an interpretation of our model based on maximum likelihood estimation (MLE) and draw the connection between the proposed back-search algorithm and Metropolis-Hastings sampler. We further introduce the task-specific designs in section 2.4.

2.3.1 Inference

In a neural-symbolic system, let x be the input (*e.g.* an image or question), z be the hidden symbolic representation, and y be the desired output inferred by z. The proposed NGS model combines neural perception, grammar parsing, and symbolic reasoning modules efficiently to perform the inference.

Neural Perception. The neural network is used as a perception module which maps the high-dimensional input x to a normalized probability distribution of the hidden symbolic representation z:

$$p_\theta(z|x) = softmax(\phi_\theta(z,x)) \tag{2.1}$$
$$= \frac{\exp(\phi_\theta(z,x))}{\sum_{z'} \exp(\phi_\theta(z',x))}, \tag{2.2}$$

where $\phi_\theta(z,x)$ is a scoring function or a negative energy function represented by a neural network with parameters θ.

Grammar Parsing. Take z as a sequence of individual symbols: $z = (z_1, z_2, ..., z_l), z_i \in \Sigma$, where Σ denotes the vocabulary of possible symbols. The neural network is powerful at modeling the mapping between x and z, but the recursive compositionality among the individual symbols z_i is not well captured. Grammar is a natural choice to tackle this problem by modeling the compositional properties in sequence data.

Take the *context-free grammar* (CFG) as an example. In formal language theory, a CFG is a type of formal grammar containing a set of production rules that describe all possible sentences in a given formal language. Specifically, a context-free grammar G in Chomsky Normal Form is defined by a 4-tuple $G = (V, \Sigma, R, S)$, where

- V is a finite set of non-terminal symbols that can be replaced by/expanded to a sequence of symbols.
- Σ is a finite set of terminal symbols that represent actual words in a language, which cannot be further expanded. Here Σ is the vocabulary of possible symbols.
- R is a finite set of production rules describing the replacement of symbols, typically of the form $A \to BC$ or $A \to \alpha$, where $A, B, C \in V$ and $\alpha \in \Sigma$. A production rule replaces the left-hand side non-terminal symbols by the right-hand side expression. For example, $A \to BC|\alpha$ means that A can be replaced by either BC or α.
- $S \in V$ is the start symbol.

Given a formal grammar, *parsing* is the process of determining whether a string of symbolic nodes can be accepted according to the production rules in the grammar. If the string is accepted by the grammar, the parsing process generates a parse tree. A parse tree represents the syntactic structure of a string according to certain CFG. The root node of the tree is the grammar root. Other non-leaf nodes correspond to non-terminals in the grammar, expanded according to grammar production rules. The leaf nodes are terminal nodes. All the leaf nodes together form a sentence.

In neural-symbolic tasks, the objective of parsing is to find the most probable z that can be accepted by the grammar:

$$\hat{z} = \arg \max_{z \in L(G)} p_\theta(z|x) \tag{2.3}$$

where $L(G)$ denotes the language of G, i.e., the set of all valid z that accepted by G.

Traditional grammar parsers can only work on symbolic sentences. [QJZ18] proposes a generalized version of Earley Parser, which takes a probability sequence as input and outputs the most probable parse. We use this method to compute the best parse $\hat{z}$ in Equation 2.3.

Symbolic Reasoning. Given the parsed symbolic representation $\hat{z}$, the symbolic reasoning module performs deterministic inference with $\hat{z}$ and the domain-specific knowledge Δ. Formally, we want to find the entailed sentence $\hat{y}$ given $\hat{z}$ and Δ:

$$\hat{y} : \hat{z} \ \wedge \ \Delta \models \hat{y} \tag{2.4}$$

Since the inference process is deterministic, we re-write the above equation as:

$$\hat{y} = f(\hat{z}; \Delta), \tag{2.5}$$

where f denotes complete inference rules under the domain Δ. The inference rules generate a reasoning path $\hat{\tau}$ that leads to the predicted output $\hat{y}$ from $\hat{z}$ and Δ. The reasoning path $\hat{\tau}$ has a tree structure with the root node $\hat{y}$ and the leaf nodes from $\hat{z}$ or Δ.

2.3.2 Learning

It is challenging to obtain the ground truth of the symbolic representation z, and the rules (*i.e.* grammar rules and the symbolic inference rules) are usually designed explicitly by human knowledge. We formulate the learning process as a weakly-supervised learning of the neural network model θ where the symbolic representation z is missing, and the grammar model G, domain-specific language Δ, the symbolic inference rules f are given.

2.3.2.1 1-step back-search (1-BS)

As shown in Figure 2.1, previous methods using policy gradient to learn the model discard all the samples with zero reward and learn nothing from them. It makes the learning process inefficient and unstable. However, humans can learn from the wrong predictions by *diagnosing* and *correcting* the wrong answers according to the desired outputs with top-down reasoning. Based on such observation, we propose a 1-step back-search (1-BS) algorithm which can *correct* wrong samples and use the corrections as pseudo labels for training. The 1-BS algorithm closes the learning loop since the error can also be propagated through the non-differentiable grammar parsing and symbolic reasoning modules. Specifically, we find the most probable correction for the wrong prediction by back-tracking the symbolic reasoning tree and propagating the error from the root node into the leaf nodes in a top-down manner.

The 1-BS algorithm is implemented with a priority queue as shown in Algorithm 1. The 1-BS gradually searches down the reasoning tree $\hat{\tau}$ starting from the root node S to the leaf nodes. Specifically, each element in the priority queue represents a valid change, defined as a 3-tuple (A, α_A, p):

- $A \in V \cup \Sigma$ is the current visiting node.
- α_A is the expected value on this node, which means if the value of A is changed to α_A, $\hat{z}$ will execute to the ground-truth answer y, *i.e.* $y = f(\hat{z}(A \to \alpha_A); \Delta))$.
- p is the visiting priority, which reflects the potential of changing the value of A.

Formally, the priority for this change is defined as the probability ratio:

$$p(A \to \alpha_A) = \begin{cases} \frac{1-p(A)}{p(A)}, & \text{if } A \notin \Sigma \\ \frac{p(\alpha_A)}{p(A)}, & \text{if } A \in \Sigma \ \& \ \alpha_A \in \Sigma. \end{cases} \tag{2.6}$$

where $p(A)$ is calculated as Equation 2.1,if $A \in \Sigma$; otherwise, it is defined as the product of the probabilities of all leaf nodes in A. If $A \in \Sigma$ and $\alpha_A \notin \Sigma$, it means we need to correct the terminal node to a value that is not in the vocabulary. Therefore, this change is not possible and thus should be discarded.

The error propagation through the reasoning tree is achieved by a $solve(B, A, \alpha_A | \Delta, G)$ function, which aims at computing the expected value α_B of the child node B from the expected value α_A of its parent node A, *i.e.*, finding α_B satisfying $f(\hat{z}(B \to \alpha_B); \Delta)) = f(\hat{z}(A \to \alpha_A); \Delta)) = y$. Please refer to the *supplementary material* for some illustrative examples of the 1-BS process.

In the 1-BS, we make a greedy assumption that only one symbol can be replaced at a time. This assumption implies only searching the neighborhood of $\hat{z}$ at one-step distance. In subsubsection 2.3.2.3, the 1-BS is extended to the multi-step back-search algorithm, which allows searching beyond one-step distance.

2.3.2.2 Maximum Likelihood Estimation

Since z is conditioned on x and y is conditioned on z, the likelihood for the observation (x, y) marginalized over z is:

$$p(y|x) = \sum_z p(y, z|x) = \sum_z p(y|z) p_\theta(z|x). \tag{2.7}$$

The learning goal is to maximize the observed-data log likelihood $L(x, y) = \log p(y|x)$.

Algorithm 1 1-step back-search (1-BS)

```
Input: ẑ, S, y
q = PriorityQueue()
q.push(S, y, 1)
while A, α_A, p = q.pop() do
   if A ∈ Σ then
      z* = ẑ(A → α_A)
      return z*
   end if
   for B ∈ child(A) do
      α_B = solve(B, A, α_A | Δ, G)
      q.push(B, α_B, p(B → α_B))
   end for
end while
return ∅
```

By taking derivative, the gradient for the parameter θ is given by

$$\begin{aligned}
\nabla_\theta L(x,y) &= \nabla_\theta \log p(y|x) \\
&= \frac{1}{p(y|x)} \nabla_\theta p(y|x) \\
&= \sum_z \frac{p(y|z)p_\theta(z|x)}{\sum_{z'} p(y|z')p_\theta(z'|x)} \nabla_\theta \log p_\theta(z|x) \\
&= \mathbb{E}_{z\sim p(z|x,y)}[\nabla_\theta \log p_\theta(z|x)],
\end{aligned} \tag{2.8}$$

where $p(z|x,y)$ is the posterior distribution of z given x,y. Since $p(y|z)$ is computed by the symbolic reasoning module and can only be 0 or 1, $p(z|x,y)$ can be written as:

$$\begin{aligned}
p(z|x,y) &= \frac{p(y|z)p_\theta(z|x)}{\sum_{z'} p(y|z')p_\theta(z'|x)} \\
&= \begin{cases} 0, \quad \text{for} \quad z \notin Q \\ \frac{p_\theta(z|x)}{\sum_{z'\in Q} p_\theta(z'|x)}, \quad \text{for } z \in Q \end{cases}
\end{aligned} \tag{2.9}$$

where $Q = \{z : p(y|z) = 1\} = \{z : f(z;\Delta) = y\}$ is the set of z that generates y. Usually Q is a very small subset of the whole space of z.

Equation 2.9 indicates that z is sampled from the posterior distribution $p(z|x, y)$, which only has non-zero probabilities on Q, instead of the whole space of z. Unfortunately, computing the posterior distribution is not efficient as evaluating the normalizing constant for this distribution requires summing over all possible z, and the computational complexity of the summation grows exponentially.

Nonetheless, it is feasible to design algorithms that sample from this distribution using Markov chain Monte Carlo (MCMC). Since z is always trapped in the modes where $p(z|x, y) = 0$, the remaining question is how we can sample the posterior distribution $p(z|x, y)$ efficiently to avoid redundant random walk at states with zero probabilities.

2.3.2.3 m-BS as Metropolis-Hastings Sampler

Algorithm 2 m-step back-search (m-BS)

Hyperparameters: T, λ
Input: $\hat{z}, y$
$z^{(0)} = \hat{z}$
for $t \leftarrow 0$ to $T-1$ **do**
    $z^* = \text{1-BS}(z^t, y)$
    draw $u \sim \mathcal{U}(0, 1)$
    if $u \leqslant \lambda$ and $z^* \neq \varnothing$ **then**
        $z^{t+1} = z^*$
    else
        $z^{t+1} = \textsc{RandomWalk}(z^t)$
    end if
end for
return z^T

function $\textsc{RandomWalk}(z^t)$
    sample $z^* \sim g(\cdot|z^t)$
    compute acceptance ratio $a = min(1, \frac{p_\theta(z^*|x)}{p_\theta(z^t|x)})$
    draw $u \sim \mathcal{U}(0, 1)$
    $z^{t+1} = \begin{cases} z^*, & \text{if } u \leqslant a \\ z^t, & \text{otherwise.} \end{cases}$
end function

In order to perform efficient sampling, we extend the 1-step back search to a multi-step back search (m-BS), which serves as a Metropolis-Hastings sampler.

A Metropolis-Hastings sampler for a probability distribution $\pi(s)$ is a MCMC algorithm that makes use of a proposal distribution $Q(s'|s)$ from which it draws samples and uses an acceptance/rejection scheme to define a transition kernel with the desired distribution $\pi(s)$. Specifically, given the current state s, a sample $s' \neq s$ drawn from $Q(s'|s)$ is accepted as the next state with probability

$$A(s, s') = min\left\{1, \frac{\pi(s')Q(s|s')}{\pi(s)Q(s'|s)}\right\}. \tag{2.10}$$

Since it is impossible to jump between the states with zero probability, we define $p'(z|x, y)$ as a smoothing of $p(z|x, y)$ by adding a small constant ϵ to $p(y|z)$:

$$p'(z|x, y) = \frac{[p(y|z) + \epsilon]p_\theta(z|x)}{\sum_{z'}[p(y|z') + \epsilon]p_\theta(z'|x)} \tag{2.11}$$

As shown in Algorithm 2, in each step, the m-BS proposes 1-BS search with probability of λ ($\lambda < 1$) and random walk with probability of $1 - \lambda$. The combination of 1-BS and random walk helps the sampler to traverse all the states with non-zero probabilities and ensures the Markov chain to be ergodic.

Random Walk: Defining a Poisson distribution for the random walk as

$$g(z_1|z_2) = Poisson(d(z_1, z_2); \beta), \tag{2.12}$$

where $d(z_1, z_2)$ denotes the edit distance between z_1, z_2, and β is equal to the expected value of d and also to its variance. β is set as 1 in most cases due to the preference for a short-distance random walk. The acceptance ratio for sampling a z^* from $g(\cdot|z^t)$ is $a = min(1, r(z^t, z^*))$, where

$$
\begin{aligned}
r(z^t, z^*) &= \frac{q(z^*)(1-\lambda)g(z^t|z^*)}{q(z^t)(1-\lambda)g(z^*|z^t)} \\
&= \frac{p_\theta(z^*|x)}{p_\theta(z^t|x)}.
\end{aligned}
\tag{2.13}
$$

1-BS: While proposing the z^* with 1-BS, we search a z^* that satisfies $p(y|z^*) = 1$. If z^* is proposed, the acceptance ratio for is $a = min(1, r(z^t, z^*))$, where

$$
\begin{aligned}
r(z^{(t)}, z^*) &= \frac{q(z^*)[0 + (1-\lambda)g(z^t|z^*)]}{q(z^t)\cdot[\lambda + (1-\lambda)g(z^*|z^{(t)})]} \\
&= \frac{1+\epsilon}{\epsilon}\cdot\frac{p_\theta(z^*|x)}{p_\theta(z^t|x)}\cdot\frac{(1-\lambda)g(z^t|z^*)}{\lambda+(1-\lambda)g(z^*|z^t)}.
\end{aligned}
\tag{2.14}
$$

$q(z) = [p(y|z)+\epsilon]p_\theta(z|x)$ is denoted as the numerator of $p'(z|x, y)$. With an enough small ϵ, $\frac{1+\epsilon}{\epsilon} \gg 1$, $r(z^t, z^*) > 1$, we will always accept z^*.

Notably, the 1-BS algorithm tries to transit the current state into a state where $z^* =$ 1-BS(z^t, y), making movements in directions of increasing the posterior probability. Similar to the gradient-based MCMCs like Langevin dynamics [DK86, WT11], this is the main reason that the proposed method can sample the posterior efficiently.

2.3.2.4 Comparison with Policy Gradient

Since grammar parsing and symbolic reasoning are non-differentiable, most of the previous approaches for neural-symbolic learning use policy gradient like REINFORCE to learn the neural network. Treat $p_\theta(z|x)$ as the policy function and the reward given z, y can be written as:

$$
r(z, y) = \begin{cases} 0, \text{ if } f(z;\Delta) \neq y. \\ 1, \text{ if } f(z;\Delta) = y. \end{cases}
\tag{2.15}
$$

The learning objective is to maximize the expected reward under current policy p_θ:

$$R(x,y) = \mathbb{E}_{z\sim p_\theta(z|x))}\, r(z,y) = \sum_z p_\theta(z|x) r(z,y). \tag{2.16}$$

Then the gradient for θ is:

$$\begin{aligned}\nabla_\theta R(x,y) &= \sum_z r(z,y) p_\theta(z|x) \nabla_\theta \log p_\theta(z|x) \\ &= \mathbb{E}_{z\sim p_\theta(z|x))}[r(z,y)\nabla_\theta \log p_\theta(z|x)].\end{aligned} \tag{2.17}$$

We can approximate the expectation using one sample at each time, and then we get the REINFORCE algorithm:

$$\begin{aligned}\nabla_\theta &= r(z,y)\nabla_\theta \log p_\theta(z|x),\, z \sim p_\theta(z|x) \\ &= \begin{cases} 0, & \text{if } f(z;\Delta) \neq y. \\ \nabla_\theta \log p_\theta(z|x), & \text{if } f(z;\Delta) = y. \end{cases}\end{aligned} \tag{2.18}$$

Equation 2.18 reveals the gradient is non-zero only when the sampled z satisfies $f(z;\Delta) = y$. However, among the whole space of z, only a very small portion can generate the desired y, which implies that *the REINFORCE will get zero gradients from most of the samples*. This is why the REINFORCE method converges slowly or even fail to converge, as also shown from the experiments in section 2.4.

2.4 Experiments and Results

2.4.1 Handwritten Formula Recognition

2.4.1.1 Experimental Setup

Task definition. The handwritten formula recognition task tries to recognize each mathematical symbol given a raw image of the handwritten formula. We learn this task in a weakly-supervised manner, where raw image of the handwritten formula is given as input data x, and the computed results of the formulas is treated as outputs y. The symbolic representation z that represent the ground-truth formula composed by individual symbols is hidden. Our task is to predict the formula, which could further be executed to calculate the final result.

HWF Dataset. We generate the HWF dataset based on the CROHME 2019 Offline Handwritten Formula Recognition Task[1]. First, we extract all symbols from CROHME and only keep ten digits (0∼9) and four basic operators (+,−,×, ÷). Then we generate formulas by sampling from a pre-defined grammar that only considers arithmetic operations over single-digit numbers. For each formula, we randomly select symbol images from CROHME. Overall, our dataset contains 10K training formulas and 2K test formulas.

Evaluation Metrics. We report both the calculation accuracy (*i.e.* whether the calculation of predicted formula yields to the correct result) and the symbol recognition accuracy (*i.e.* whether each symbol is recognized correctly from the image) on the synthetic dataset.

Models. In this task, we use LeNet [LeC15] as the neural perception module to process the handwritten formula. Before feeding into LeNet, the original image of an formula is pre-segmented into a sequence of sub-images, and each sub-image contains only one symbol. The symbolic reasoning module works like a calculator, and each inference step computes the parent value given the values of two child nodes (left/right) and the operator. The $solve(B, A, \alpha_A)$ function in 1-step back-search algorithm works in the following way for mathematical formulas:

- If B is A's left or right child, we directly solve the equation $\alpha_B \bigoplus child_R(A) = \alpha_A$ or $child_L(A) \bigoplus \alpha_B = \alpha_A$ to get α_B, where $\bigoplus$ denotes the operator.
- If B is an operator node, we try all other operators and check whether the new formula can generate the correct result.

[1] `https://www.cs.rit.edu/~crohme2019/task.html`

We conduct experiments by comparing the following variants of the proposed model:

- **NGS-RL**: learning the NGS model with REINFORCE.
- **NGS-MAPO**: learning the NGS model by Memory Augmented Policy Optimization (MAPO) [LNB18b], which leverages a memory buffer of rewarding samples to reduce the variance of policy gradient estimates.
- **NGS-RL-Pretrain**: NGS-RL with LeNet pre-trained on a small set of fully-supervised data.
- **NGS-MAPO-Pretrain**: NGS-MAPO with pre-trained LeNet.
- **NGS-m-BS**: learning the NGS model with the proposed m-step back-search algorithm.

2.4.1.2 Results and Analyses

Learning Curve. Figure 2.2 shows the learning curves of different models. The proposed NGS-m-BS converges much faster and achieves higher accuracy compared with other models. NGS-RL fails without pre-training and rarely improves during the entire training process. NGS-MAPO can learn the model without pre-training, but it takes a long time to start efficient learning, which indicates that MAPO suffers from the cold-start problem and needs time to accumulate rewarding samples. Pre-training the LeNet solves the cold start problem for NGS-RL and NGS-MAPO. However, the training curves for these two models are quite noisy and are hard to converge even after 100k iterations. Our NGS-m-BS model learns from scratch and avoids the cold-start problem. It converges quickly with nearly perfect accuracy, with a much smoother training curve than the RL baselines.

Back-Search Step. Figure 2.3 illustrates the comparison of the various number of steps in the multi-step back-search algorithm. Generally, increasing the number of steps will increase the chances of correcting wrong samples, thus making the model converge faster. However, increasing the number of steps will also increase the time consumption of each iteration.

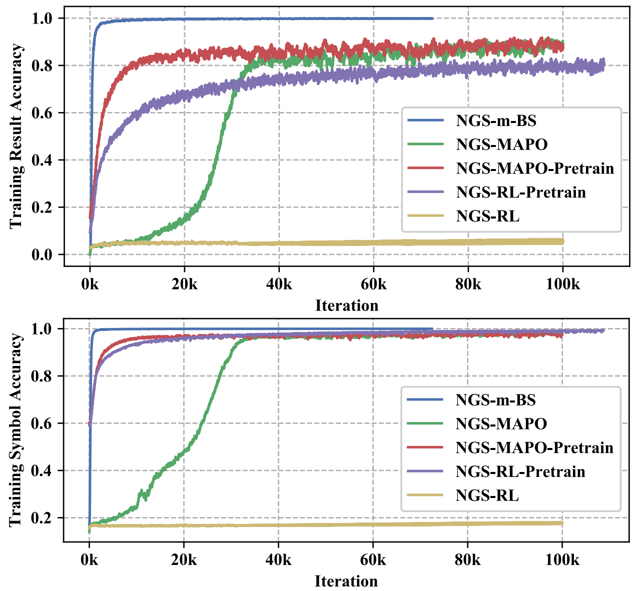


Figure 2.2: The learning curves of the calculation accuracy and the symbol recognition accuracy for different models.

Data Efficiency. Table 2.1 and Table 2.2 show the accuracies on the test set while using various percentage of training data. All models are trained with 15K iterations. It turns out the NGS-m-BS is much more data-efficient than the RL methods. Specifically, when only using 25% of the training data, NGS-m-BS can get a calculation accuracy of 93.3%, while NGS-MAPO only gets 5.1%.

Table 2.1: The calculation accuracy on the test set using various percentage of training data.

Model	25%	50 %	75 %	100%
NGS-RL	0.035	0.036	0.034	0.034
NGS-MAPO	0.051	0.095	0.305	0.717
NGS-RL-Pretrain	0.534	0.621	0.663	0.685
NGS-MAPO-Pretrain	0.687	0.773	0.893	0.956
NGS-m-BS	**0.933**	**0.957**	**0.975**	**0.985**

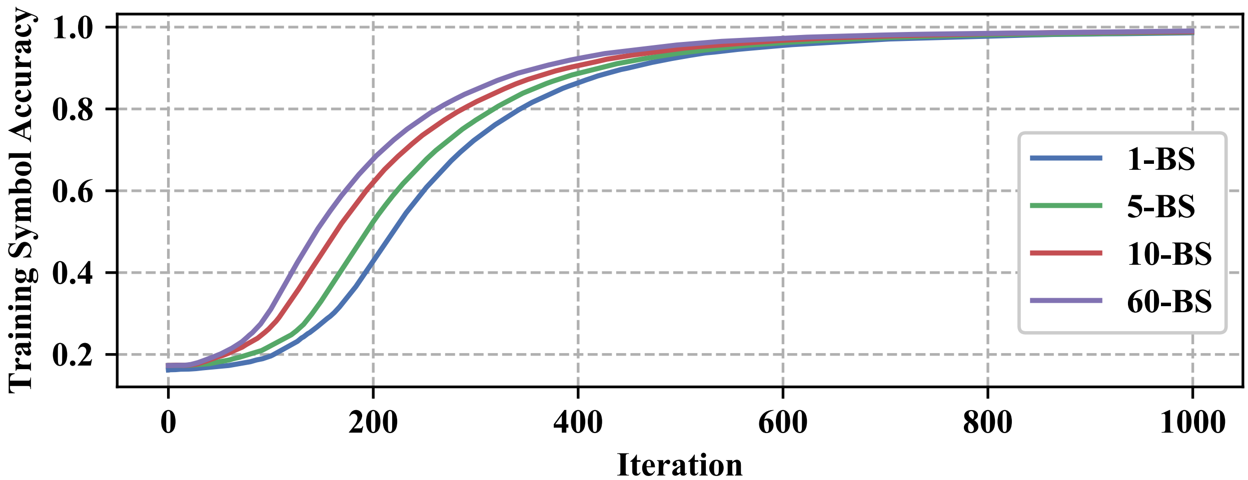


Figure 2.3: The training curves of NGS-m-BS with different steps.

Table 2.2: The symbol recognition accuracy on the test set using various percentage of training data.

Model	25%	50 %	75 %	100%
NGS-RL	0.170	0.170	0.170	0.170
NGS-MAPO	0.316	0.481	0.785	0.967
NGS-RL-Pretrain	0.916	0.945	0.959	0.964
NGS-MAPO-Pretrain	0.962	0.983	0.985	0.991
NGS-m-BS	**0.988**	**0.992**	**0.995**	**0.997**

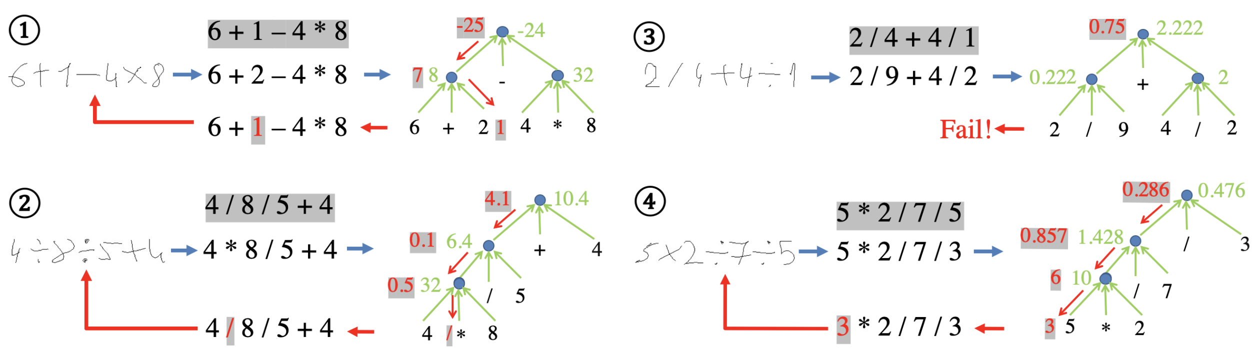


Figure 2.4: Examples of correcting wrong predictions using the one-step back-search algorithm.

Qualitative Results. Figure 2.4 illustrates four examples of correcting the wrong predictions with 1-BS. In the first two examples, the back-search algorithm successfully corrects the wrong predictions by changing a digit and an operator, respectively. In the third example, the back-search fails to correct the wrong sample. However, if we increase the number of search steps, the model could find a correction for the example. In the fourth example, the back-search finds a spurious correction, which is not the same as the ground-truth formula but generates the same result. Such spurious correction brings a noisy gradient to the neural network update. It remains an open problem for how to avoid similar spurious corrections.

2.4.2 Neural-Symbolic Visual Question Answering

2.4.2.1 Experimental Setup

Task. Following [YWG18], the neural-symbolic visual question answering task tries to parse the question into functional program and then use a program executor that runs the program on the structural scene representation to obtain the answer. The functional program is hidden.

Dataset. We evaluate the proposed method on the CLEVR dataset [JHM17a]. The CLEVR dataset is a popular benchmark for testing compositional reasoning capability of VQA models in previous works [JHV17, VDL19]. CLEVR consists of a training set of 70K images and ∼700K questions, and a validation set of 15K images and ∼150K questions. We use the VQA accuracy as the evaluation metric.

Models. We adopt the NS-VQA model in [YWG18] and replace the attention-based seq2seq question parser with a Pointer Network [VFJ15]. We store a dictionary to map the keywords in each question to the corresponding functional modules. For example, "red"→"filter color [red]", "how many"→ "count", and "what size" → "query size" *etc.* Therefore, the Pointer Network can point to the functional modules that are related to the input question. The grammar model ensures that the generated sequence of function modules can form a valid program, which indicates the inputs and outputs of these modules can be strictly matched with their forms. We conduct experiments by comparing following models: **NS-RL**, **NGS-RL**, **NGS-1-BS**, **NGS-m-BS**.

2.4.2.2 Results and Analyses

Learning Curve. Figure 2.5 shows the learning curves of different model variants. NGS-BS converges much faster and achieves higher VQA accuracy on the test set compared with the RL baselines. Though taking a long time, NGS-RL does converge, while NS-RL fails. This fact indicates that the grammar model plays a critical role in this task. Conceivably, the latent functional program space is combinatory, but the grammar model rules out all invalid programs that cannot be executed by the symbolic reasoning module. It largely reduces the solution space in this task.

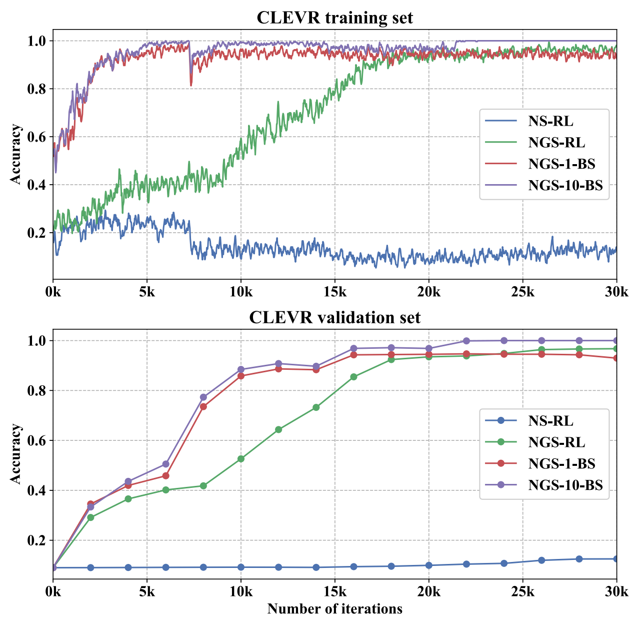


Figure 2.5: The learning curves of different model variants on training and validation set of the CLEVR dataset.

Back-Search Step. As shown in Figure 2.5, NGS-10-BS performs slightly better than the NGS-1-BS, which indicates that searching multiple steps does not help greatly in this task. One possible reason is that there are more ambiguities and more spurious examples compared with the handwritten formula recognition task, making it less efficient to do the m-BS. For example, for the answer "yes", there might be many possible programs for this question that can generate the same answer given the image.

Data Efficiency Table 2.3 shows the accuracies on the CLEVR validation set when different portions of training data are used. With less training data, the performances decrease for both NGS-RL and NGS-m-BS, but NGS-m-BS still consistently obtains higher accuracies.

Table 2.3: The VQA accuracy on the CLEVR validation set using different percentage of training data. All models are trained 30k iterations.

Model	25%	50 %	75 %	100%
NS-RL	0.090	0.091	0.099	0.125
NGS-RL	0.678	0.839	0.905	0.969
NGS-m-BS	**0.873**	**0.936**	**1.000**	**1.000**

2.5 Conclusions

In this work, we propose a neural-grammar-symbolic model and a back-search algorithm to close the loop of neural-symbolic learning. We demonstrate that the grammar model can dramatically reduce the solution space by eliminating invalid possibilities in the latent representation space. The back-search algorithm endows the NGS model with the capability of learning from wrong samples, making the learning more stable and efficient. One future direction is to learn the symbolic prior (*i.e.* the grammar rules and symbolic inference rules) automatically from the data.

CHAPTER 3

A HINT from Arithmetic: On the Integration and Generalization of Perception, Syntax, and Semantics

In this chapter, we introduce a synthetic benchmark that is specially designed to study the problem of closing the loop of recognition and reasoning. Inspired by humans' remarkable ability to master arithmetic and generalize to unseen problems, we present a new dataset, HINT, to study machines' capability of learning generalizable concepts at three different levels: *perception*, *syntax*, and *semantics*. In particular, concepts in HINT, including both digits and operators, are required to learn in a weakly-supervised fashion: Only the final results of handwriting expressions are provided as supervision. Learning agents need to reckon how concepts are perceived from raw signals such as images (*i.e.*, perception), how multiple concepts are structurally combined to form a valid expression (*i.e.*, syntax), and how concepts are realized to afford various reasoning tasks (*i.e.*, semantics). With a focus on systematic generalization, we carefully design a five-fold test set to evaluate both the *interpolation* and the *extrapolation* of learned concepts. To tackle this challenging problem, we propose a neural-symbolic system by integrating neural networks with grammar parsing and program synthesis, learned by a novel deduction–abduction strategy. In experiments, the proposed neural-symbolic system demonstrates strong generalization capability and significantly outperforms end-to-end neural methods like RNN and Transformer. The results also indicate the significance of *recursive priors* for extrapolation on syntax and semantics. An additional preliminary few-shot study also indicates that the proposed neural-symbolic system can learn new concepts with limited examples.

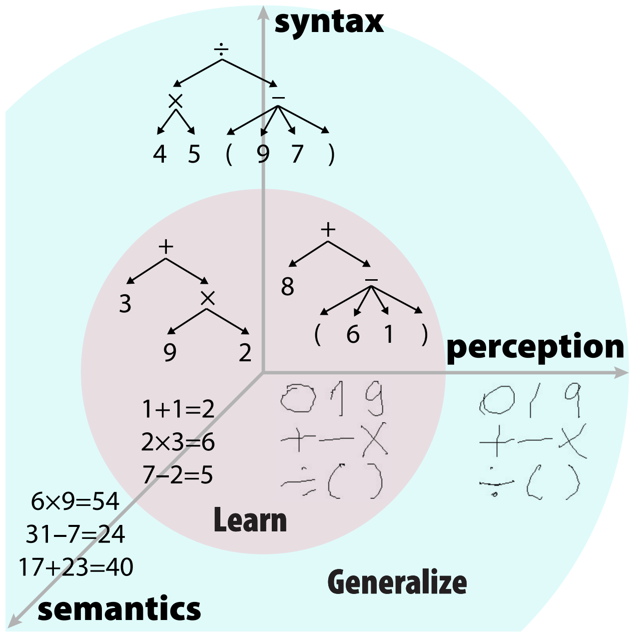


Figure 3.1: **Concept learning and generalization at three different levels.** A learning agent needs to simultaneously master (i) **perception**, how concepts are perceived from raw signals such as images, (ii) **syntax**, how multiple concepts are structurally combined to form a valid expression, and (iii) **semantics**, how concepts are realized to afford various reasoning tasks.

3.1 Introduction

Humans possess a versatile mechanism for learning concepts [FS16]. Take the arithmetic examples in Fig. 3.1: When we master concepts like digits and operators, we not only know how to recognize, write, and pronounce them—what these concepts mean at the *perceptual* level, but also know how to compose them into valid expressions—at the *syntactic* level, and how to calculate the results by reasoning over these concepts—at the *semantic* level. Learning concepts heavily rely on these three-level interweaving meanings. Such observation also conforms with the classic view of human cognition, which postulates at least three distinct levels of organizations in computation systems [Pyl84, FP88].

Crucially, a unique property of human concept learning is its systematic generalization. Once we master the syntax of arithmetic using short expressions, we can parse novel, long expressions. Similarly, once we master operators' semantics using small numbers, we can apply them over novel, large numbers. This property corresponds to the classic idea of the *systematicity* (interpolation) and *productivity* (extrapolation) in cognition: An infinite number of representations can be constructed from a finite set of primitives, just as the mind can think an infinite number of thoughts, understand an infinite number of sentences, or learn new concepts from a seemingly infinite space of possibilities [LUT17, Mar18, Fod75].

To examine the versatile human-like capabilities of concept learning with a focus on systematic generalization, we take inspiration from arithmetic and introduce a new benchmark HINT, Handwritten arithmetic with INTegers. The task of HINT is intuitive and straightforward: Machines take as input images of handwritten expressions and predict the final results of expressions, restricted in the integer space. The task of HINT is also challenging: Concepts in HINT, including digits and operators, are learned in a weakly-supervised manner. Using final results as the only supervision, machines are tasked to learn the three-level meanings simultaneously—perception, syntax, and semantics of these concepts—to correctly predict the results. Since there is no supervision on any intermediate values or representations, the three-level meanings are presumably intertwined during learning. To provide a holistic and rigorous test on whether learning machines can generalize the learned concepts, we introduce a carefully designed evaluation scheme instead of using a typical i.i.d. test split. This new scheme includes five subsets, focusing on generalization capabilities (*i.e.*, interpolation and extrapolation) at different levels of meanings (*i.e.*, perception, syntax, and semantics).

We evaluate popular state-of-the-art deep learning methods, such as GRU [CGC14] and Transformer [VSP17], on HINT. Our experiment shows that such end-to-end neural networks' performance drops significantly on examples requiring interpolation and extrapolation, even though these models can very well fit the training set. This finding echoes the long-standing arguments against connectionist models, which are believed to lack systematic generalization prevailing in human cognition [LB18, FP88].

Inspired by the superb generalization capability demonstrated in symbolic systems with combinatorial structure [FP88] and recent advances in neural-symbolic integration [LHH20a, MGK19, YWG18, MDK18], we propose an ANS system to approach the HINT challenge.

The proposed ANS system integrates the learning of perception, syntax, and semantics in a principled framework; see an illustration in Fig. 3.3. Specifically, we first utilize ResNet-18 [HZR16] as a perception module to translate a handwritten expression into a symbolic sequence. This symbolic sequence is then parsed by a transition-based neural dependency parser [CM14], which encodes the syntax of concepts. Finally, we adopt *functional programs* to realize the semantic meaning of concepts, thus view learning semantics as program induction [EWN20].

It is infeasible to perform an end-to-end optimization for our model since syntactic parsing and semantic reasoning are non-differentiable. Inspired by prior arts on abductive learning [LHH20a, Zho19a, DXY19], we derive a novel *deduction-abduction* strategy to coordinate the learning of different modules. Specifically, during learning, the system first performs greedy deduction over these modules to propose an initial, rough solution, which is likely to produce a wrong result. A one-step abduction over perception, syntax, and semantics is then applied in a top-down manner to search the initial solution's neighborhood, which updates the solution to explain the ground-truth result better. This revised solution provides *pseudo* supervision on the intermediate values and representations, which are then used to train each module individually.

Evaluated on HINT, ANS exhibits strong systematic generalization with an overall accuracy of 72%, outperforming end-to-end neural methods by nearly 33 percents. Experiments also show the strong generalization of ANS relies on its underlying *symbol system* [FP88] encoded with *recursive* priors, which facilitate the extrapolation on syntax and semantics. A preliminary study of few-shot learning further demonstrates that ANS can quickly learn new concepts with limited examples, obtaining an average accuracy of 62% on four new concepts with a hundred training examples.

3.2 Related Work

3.2.1 Three Levels of Concept Learning

The surge of deep neural networks [LBH15] in the last decade has significantly advanced the accuracy of **perception learning** from raw signals across multiple modalities, such as image classification from image pixels [HZR16, KSH12] and automatic speech recognition from audio waveforms [PCZ19, HDY12, GMH13].

The goal of **syntax analysis** is to understand the compositional and recursive structures in various tasks, such as natural language parsing [CM14, KK18], image and video parsing [TCY05, ZM07, ZZ11, GSS09, QJZ18, QJH20, JCH20], scene understanding [HQZ18, HQX18, QZH18, JQZ18, CHY19, YLF20], task planning [XLE18, LZS18, EGL19, LZZ19, ZZZ20c], and abstract reasoning [ZGJ19b, ZJG19, ZZZ20b, EMQ20, EQZ19, EKS18]. There exist two major structural types: constituency structures [KK18] and dependency structures [CM14]. Constituency structures use phrase structure grammar to organize input tokens into nested constituents, whereas dependency structures show which tokens depend on which other tokens.

Semantics of concepts essentially describe its causal effect. There are two primary semantic representations in symbolic reasoning. The first is *logic* [Llo12, MDK18], which regards the semantic learning as inductive logic programming [MD94, EG18]—a general framework to induce first-order logic theory from examples. The other representation is *program*, which treats the semantic learning as inductive program synthesis [KKT15, LST15, BGB17a, DUB17, ERS18, EMS18]. Recently, [EWN20] release a neural-guided program induction system, *DreamCoder*, which can efficiently discover interpretable, reusable, and generalizable knowledge across a wide range of domains.

However, aforementioned literature tackles *only one or two levels* of concept learning and usually requires *direct* supervision on model outputs. In contrast, we offer a more holistic perspective that addresses all three levels of concept learning, *i.e.*, perception, syntax, and semantics, taking one step closer to realize a versatile mechanism of concept learning under weak supervision. The design of three-level concept learning echoes a newly proposed challenge, HALMA [XMY21], but with a simpler setting of no interaction with the environments.

3.2.2 Systematic Generalization

The central question in systematic generalization is: How well can a learning agent perform in unseen scenarios given limited exposure to the underlying configurations [Gre93]? This question is also connected to the Language of Thought Hypothesis [Fod75]: The systematicity, productivity, and inferential coherence characterize compositional generalization of concepts [LST15]. As a prevailing property of human cognition, systematicity poses a central argument against connectionist models [FP88]. Recently, there have been several works to explore the systematic generalization of deep neural networks in different tasks [LB18, BMN18, KSS19, GLB19, XMY21]. By going beyond traditional i.i.d. train/test split, the proposed HINT benchmark well-captures the characteristics of systematic generalization across different aspects of concepts w.r.t. perception, syntax, and semantics.

3.2.3 Neural-Symbolic Integration

Researchers have proposed to combine statistical learning and symbolic reasoning, with pioneer efforts devoted to different directions, including representation learning and reasoning [Sun94, GLG08, MDK18], abductive learning [LHH20a, DXY19, Zho19a], knowledge abstraction [HOT06, BGH09], *etc*. There also have been recent works on the application of neural-symbolic methods, such as neural-symbolic visual reasoning and program synthesis [YWG18, MGK19, LHH20c, PMS16], semantic parsing [LBL16a, YZH18], and math word problems [LC20, LSR20]. Current neural-symbolic approaches often require a perfect domain-specific language, including both the syntax and semantics of the targeted domain. In comparison, the proposed model relaxes such a strict requirement and enables the learning of syntax and semantics.

3.3 The HINT Benchmark

Task Definition The task of HINT is intuitive and straightforward: It is tasked to predict the final results of handwritten arithmetic expressions in a weakly-supervised manner. Only the final results are given as supervision; all intermediate values and representations are latent, including symbolic expressions, parse trees, and execution traces.

Data Generation The data generation process follows three steps; see Fig. 3.2 for an illustration. First, we extract handwritten images from CROHME[1] to obtain primitive concepts, including digits $0 \sim 9$, operators $+, -, \times, \div$, and parentheses (,). Second, we randomly sample *prefix* expressions and convert them to *infix* expressions with necessary parentheses based on the operator precedence; we only allow single-digit numbers in expressions. These symbolic expressions are fed into a solver to calculate the final results. Third, we randomly sample handwritten images for symbols in an expression and concatenate them to construct final handwritten expressions. We only keep the handwritten expressions as input and the corresponding final results as supervision; all intermediate results are discarded.

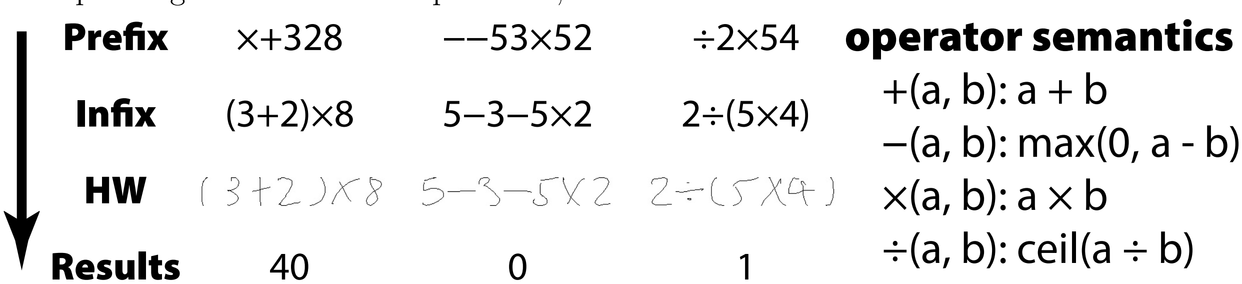


Figure 3.2: **Illustrations of the data generation pipeline.**

Train and Evaluation To rigorously evaluate how well the learned concepts are systematically generalized, we replace the typical i.i.d. train/test split with a carefully designed

[1] `https://www.cs.rit.edu/~crohme2019/`

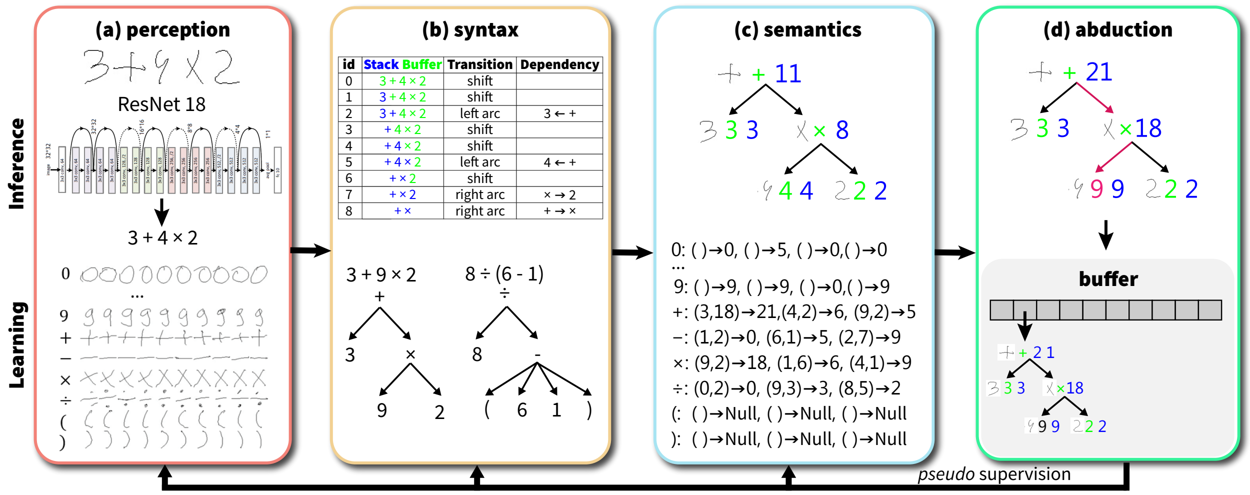


Figure 3.3: **The Arithmetic Neural-Symbolic model (ANS).** ANS consists of three modules for perception, syntax, and semantics. During inference, the model performs greedy deduction over three modules and directly proposes a solution. During learning, the proposed solution is further revised by performing abduction based on the ground-truth supervision. The updated solution is stored in a buffer, providing *pseudo* supervisions to train three modules individually. Each node in the solution tree is an (image, symbol, value) triplet.

evaluation scheme: (i) all handwritten images in the test set are unseen in training, (ii) at most 1,000 samples are generated for each number of operators in expressions, (iii) limit the maximum number of operators to 10 and the maximum values to 100 in the training set:

$$D_{train} \subset \mathcal{D}_{train} = \{(x, y) : |x| \leqslant 10, \max(v) \leqslant 100\}, \tag{3.1}$$

where x is the handwritten expression, $|x|$ its number of operators, y the final result, and v all the intermediate values generated when calculating the final result.

We carefully devise the test set to evaluate different generalization capabilities (*i.e.*, interpolation and extrapolation) on different levels of meanings (*i.e.*, perception, syntax and semantics). Specifically, the test set is composed of five subsets, formally defined as:

$$
\begin{aligned}
&D_{test} = D_{test}^{(1)} \cup D_{test}^{(2)} \cup D_{test}^{(3)} \cup D_{test}^{(4)} \cup D_{test}^{(5)}, \text{where} \\
&D_{test}^{(1)} = D_{train}, \\
&D_{test}^{(2)} \subset \mathcal{D}_{train} \backslash D_{train}, \\
&D_{test}^{(3)} \subset \{(x, y) : |x| \leqslant 10, \max(v) > 100\}, \\
&D_{test}^{(4)} \subset \{(x, y) : |x| > 10, \max(v) \leqslant 100\}, \\
&D_{test}^{(5)} \subset \{(x, y) : |x| > 10, \max(v) > 100\}.
\end{aligned}
\tag{3.2}
$$

All above subsets requires generalization on perception of learned concepts. $D_{test}^{(1)}$ requires no generalization on either syntax or semantics, $D_{test}^{(2)}$ requires interpolation on both syntax and semantics, $D_{test}^{(3)}$ requires interpolation on syntax and extrapolation on semantics, $D_{test}^{(4)}$ requires extrapolation on syntax and interpolation on semantics, and $D_{test}^{(5)}$ requires extrapolation on both syntax and semantics.

In total, the training and test set includes 11,170 and 48,910 samples, respectively. Subsets in the test set are balanced to be 23%, 23%, 22%, 16%, and 16%.

3.4 A Neural-Symbolic Approach

Below we first describe a general framework from a probabilistic perspective for learning the Hint task as a neural-symbolic approach. This general framework implies a *symbol system* with combinatorial syntactic and semantic structures, initially introduced by [FP88], as a feasible representation of the human mind. Such a symbol system provides a principled integration of perception, syntax, and semantics. Guided by this general framework, we next provide a concrete instantiation of such a neural-symbolic system and introduce a novel deduction-abduction strategy to learn it with weak supervision; see Fig. 3.3 for overview.

3.4.1 A General Framework

Given a neural-symbolic system, let $x \in \Omega_x$ denote the input (images of handwritten expression in the HINT dataset), $s \in \Omega_s$ the symbolic expression, $pt \in \Omega_t$ the parse tree of the symbolic expression, $v \in \Omega_e$ the execution trace, and $y \in \Omega_y$ the output. During learning, (x, y) are observed but (s, pt, v) are latent. The likelihood of the observation (x, y) marginalized over (s, pt, v) can be decomposed as:

$$p(y|x;\Theta) = \sum_{s,pt,v} p(s,pt,v,y|x;\Theta) = \sum_{s,pt,v} p(s|x;\theta_p)p(pt|s;\theta_s)p(v|pt;\theta_l)p(y|v), \tag{3.3}$$

where (i) $s|x$ denotes the process of perceiving symbols from raw signals, guided by the perceptual model θ_p of learned concepts; (ii) $pt|s$ denotes the process of parsing the symbolic expression into a parse tree, guided by the syntactic model θ_s; (iii) $v|pt$ denotes the process of reasoning over the parse tree, guided by the semantic model θ_l; and (iv) $y|v$ is a deterministic process: If the final output of v equals to y, $p(y|v) = 1$, otherwise 0.

From a maximum likelihood prospective, the learning objective is to maximize the observed-data log likelihood $L(x, y) = \log p(y|x)$. Take the derivative of L w.r.t. $\theta_p, \theta_s, \theta_l$, we have: (see *supp* for detailed derivation)

$$\begin{aligned}
\nabla_{\theta_p} L(x,y) &= \mathbb{E}_{s,pt,v \sim p(s,pt,v|x,y)}[\nabla_{\theta_p} \log p(s|x;\theta_p)], \\
\nabla_{\theta_s} L(x,y) &= \mathbb{E}_{s,pt,v \sim p(s,pt,v|x,y)}[\nabla_{\theta_s} \log p(pt|s;\theta_s)], \\
\nabla_{\theta_l} L(x,y) &= \mathbb{E}_{s,pt,v \sim p(s,pt,v|x,y)}[\nabla_{\theta_l} \log p(v|pt;\theta_l)],
\end{aligned} \tag{3.4}$$

where $p(s, pt, v|x, y)$ is the posterior distribution of (s, pt, v) given (x, y). Since $p(y|v)$ can only be 0 or 1, $p(s, pt, v|x, y)$ can be rewritten as:

$$p(s,pt,v|x,y) = \frac{p(s,pt,v,y|x;\Theta)}{\sum_{s',pt',v'} p(s',pt',v',y|x;\Theta)} = \begin{cases} 0, & \text{for} \quad s,pt,v \notin Q \\ \frac{p(s,pt,v|x;\Theta)}{\sum_{s',pt',v' \in Q} p(s',pt',v'|x;\Theta)}, & \text{for } s,pt,v \in Q \end{cases} \tag{3.5}$$

where $Q = \{(s, pt, v) : p(y|v) = 1, s \in \Omega_s, pt \in \Omega_t, v \in \Omega_v\}$ is the set of (s, pt, v) that generates y. Usually, Q is a very small subset of the entire space of (s, pt, v), *i.e.*, $Q \subseteq \Omega_s \times \Omega_t \times \Omega_v$, where $\times$ denotes the Cartesian product.

$p(s, pt, v|x, y)$ is a highly-sparse distribution in which most points has zero probability.

Since taking expectation w.r.t. this posterior distribution is intractable, we use Monte Carlo sampling to approximate it. Therefore, the learning procedure for an example (x, y) can be depicted as following:

1. sample $\hat{s}, \hat{pt}, \hat{v} \sim p(s, pt, v|x, y)$;
2. use $(x, \hat{s})$ to update the perception module (θ_p);
3. use $(\hat{s}, \hat{pt})$ to update the parsing module (θ_s);
4. use $(\hat{pt}, \hat{v})$ to update the reasoning module (θ_l).

3.4.2 Instantiation: Arithmetic Neural-Symbolic (ANS)

The general framework of the desired neural-symbolic system described above is agnostic to the choice of functions and algorithms. Below we delineate a learnable implementation, named ANS, capable of learning generalizable concepts in arithmetic on the proposed Hint dataset.

3.4.2.1 Perception: Neural Network (NN)

The role of the perception module is to map a handwritten expression x into a symbolic expression s. Since disentangling visual symbols from handwritten expressions is trivial in this domain , we assume the input as a sequence of handwritten images, where each image contains one symbol. We adopt a standard ResNet-18 [HZR16] as the perception module to map each handwritten image into a probability distribution over the concept space Σ. Formally,

$$p(s|x;\theta_p) = \prod_i p(w_i|x_i;\theta_p) = \prod_i \texttt{softmax}(\phi(w_i, x_i;\theta_p)), \tag{3.6}$$

where $\phi(s, x; \theta_p)$ is a scoring function parameterized by a NN with parameters θ_p. Since learning such an NN from scratch is prohibitively challenging, the ResNet-18 is pre-trained unsupervisedly [VVG20] on unlabeled handwritten images.

3.4.2.2 Syntax: Dependency Parsing

To parse the symbolic sequence into a parse tree, we adopt a greedy transition-based neural dependency parser [CM14], commonly used for parsing natural language sentences. The transition-based dependency parser relies on a state machine that defines the possible transitions to parse the input sequence into a dependency tree; see panel (b) of Fig. 3.3. The learning process induces a model to predict the next transition in the state machine based on the transition history. The parsing process constructs the optimal sequence of transitions for the input sequence. A dependency parser for arithmetic expressions is essentially approximating the Shunting-yard algorithm.

In our parser, a *state* $c = (\alpha, \beta, A)$ consists of a *stack* α, a *buffer* β, and a set of *dependency arcs* A. The initial state for a sequence $s = w_0 w_1 ... w_n$ is $\alpha = [\texttt{Root}], \beta = [w_0 w_1 ... w_n], A = \varnothing$. A state is regarded as terminal if the buffer is empty *and* the stack only contains the node `Root`. The parse tree can be derived from the dependency arcs A. Let α_i denote the i-th top element on the stack, and β_i the i-th element on the buffer. The parser defines three types of transitions between states:

- Left-Arc: add an arc $\alpha_1 \to \alpha_2$ to A and remove α_2 from the stack α. Precondition: $|\alpha| \geqslant 2$.
- Right-Arc: add an arc $\alpha_2 \to \alpha_1$ to A and remove α_1 from the stack α. Precondition: $|\alpha| \geqslant 2$.
- Shift: move β_1 from the buffer β to the stack α. Precondition: $|\beta| \geqslant 1$.

The goal of the parser is to predict a transition sequence from an initial state to a terminal state. As the parser is greedy, it attempts to predict one transition from $\mathcal{T} = \{\text{Left-Arc}, \text{Right-Arc}, \text{Shift}\}$ at a time, based on the current state $c = (\alpha, \beta, A)$. The features for a state c contains following three elements: (i) The top three words on the stack and buffer: $\alpha_i, \beta_i, i=1,2,3$; (ii) The first and second leftmost/rightmost children of the top two words on the stack: $lc_1(\alpha_i), rc_1(\alpha_i), lc_2(\alpha_i), rc_2(\alpha_i), i=1,2$; (iii) The leftmost of leftmost/rightmost of rightmost children of the top two words on the stack: $lc_1(lc_1(\alpha_i)), rc_1(rc_1(\alpha_i)), i=1,2$. We use a special `Null` token for non-existent elements. Each element in the state representation is embedded to a d-dimensional vector $e \in R^d$, and the full embedding matrix is denoted as $E \in R^{|\Sigma| \times d}$, where Σ is the concept space. The embedding vectors for all elements in the state are concatenated as its representation: $c = [e_1 \ e_2 ... e_n] \in R^{nd}$. Given the state representation, we adopt a two-layer feed-forward NN to predict a transition.

3.4.2.3 Semantics: Program Synthesis

Algorithm 3 Learning by Deduction-Abduction

Input: Training set $D = \{(x_i, y_i) : i = 1, 2, ..., N\}$
Initial Module: perception $\theta_p^{(0)}$, syntax $\theta_s^{(0)}$, semantics $\theta_l^{(0)}$
for $t \leftarrow 0$ to T **do**
  Buffer $\mathcal{B} = \varnothing$
  for $(x, y) \in D$ **do**
    $ct = \textsc{Deduce}(x, \theta_p^{(t)}, \theta_s^{(t)}, \theta_l^{(t)})$
    $ct^* = \textsc{Abduce}(ct, y)$
    $\mathcal{B} = \mathcal{B} \cup \{ct^*\}$
  end for
  $\theta_p^{(t+1)}, \theta_s^{(t+1)}, \theta_l^{(t+1)} = \text{learn}(\mathcal{B}, \theta_p^{(t)}, \theta_s^{(t)}, \theta_l^{(t)})$
end for
return $\theta_p^{(T)}, \theta_s^{(T)}, \theta_l^{(T)}$
function $\textsc{Deduce}(x, \theta_p, \theta_s, \theta_l)$
  sample $\hat{s} \sim p(s|x; \theta_p), \hat{pt} \sim p(pt|\hat{s}; \theta_s), \hat{et} = f(\hat{pt}; \theta_l)$
  return $ct = (x, \hat{s}, \hat{pt}, \hat{et})$
end function

Inspired by recent advances in program synthesis [EWN20, BGB17a, DUB17], we adopt *functional programs* to represent the semantics of concepts and view learning as program induction. The semantics of a concept is treated as a function, mapping certain inputs to an output. Learning semantics is equivalent to searching for a program that approximates this unknown function. Compare to purely statistical approaches, symbolic programs exhibit better generalizability and interpretability, and the learning is also more sample-efficient.

(1) $\texttt{0}$; (2) $\texttt{inc}$: $a \rightarrow a+1$; (3) $\texttt{dec}$: $a \rightarrow a-1$; (4) $\texttt{if}$: $(a,b,c) \rightarrow$ (if a is $\texttt{0}$) b (else) c.

$+(a,b): \texttt{if}\ b\ a\ (+\ \texttt{inc}(a)\ \texttt{dec}(b))$

To learn semantics as programs, we start from DreamCoder [EWN20], a machine learning system that can efficiently synthesize interpretable, reusable, and generalizable programs across a wide range of domains. DreamCoder embodies a wake-sleep Bayesian program induction approach to progressively learn multiple tasks in a domain, given a set of primitives and input-out pairs for each task. For arithmetic reasoning, the Peano axioms [Pea89] define four primitives: (1) $\texttt{0}$; (2) $\texttt{inc}$: $a \rightarrow a+1$; (3) $\texttt{dec}$: $a \rightarrow \max(0, a-1)$; (4) $\texttt{if}$: $(a,b,c) \rightarrow$ (if a is 0) b (else) c. Any arithmetic function can be provably composed of these four primitives. This set of primitives is augmented with a recursion primitive, $\texttt{Y}$-combinator (*a.k.a.*, fixed-point combinator). The $\texttt{Y}$-combinator enables the derivation of recursive functions and is the crux of extrapolating to large numbers.

The semantics of concepts in HINT, including digits, operators, and parentheses, are all represented as programs composed from these primitives $L = \{\texttt{0}, \texttt{inc}, \texttt{dec}, \texttt{if}, \texttt{Y}\}$. During inference, these programs are used for reasoning to obtain the results. The learning for a concept c is to find a program ρ_c to maximize the following objective:

$$\rho_c = \arg\max_{\rho} p(\rho|D_c, L) \propto p(D_c|\rho)\ p(\rho|L), \tag{3.7}$$

where D_c denotes the input-output pairs of the concept c for program induction, $p(D_c|\rho)$ the likelihood of the program ρ explaining D_c, and $p(\rho|L)$ the prior of ρ under the library L, which defines a generative model over programs. The maximization in Eq. (3.7) is achieved by a stochastic search process guided by a neural network, which is trained to approximate the posterior distribution $p(\rho|D_c, L)$.

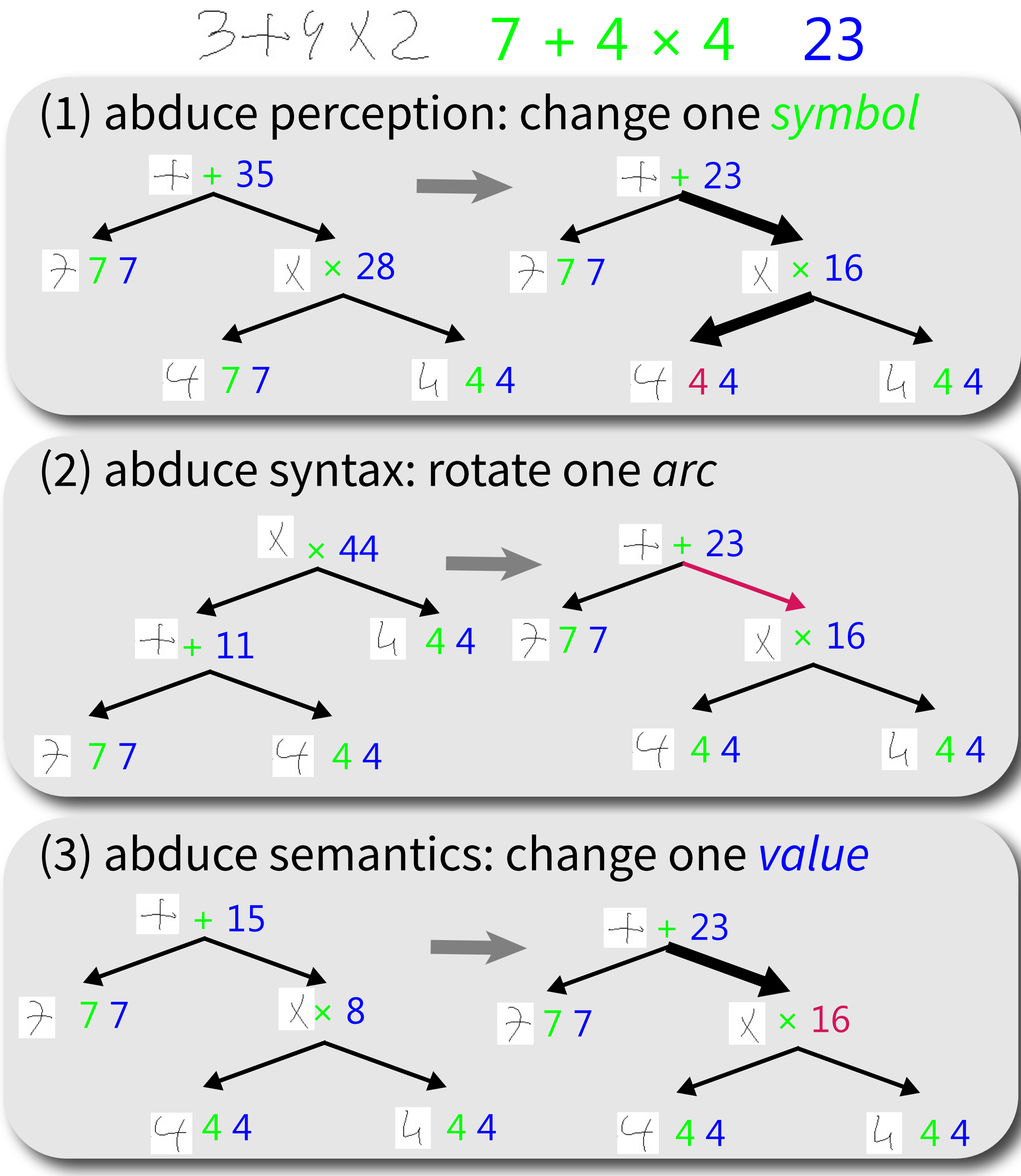


Figure 3.4: **Abduction over perception, syntax, and semantics.** Each node in the solution tree is a triplet of (image, symbol, value). Parts revised during abduction are highlighted in red.

3.4.2.4 Learning by Deduction-Abduction

In Section 3.4.1, we derive a general learning procedure for such a neural-symbolic system. The key is to perform efficient sampling from the posterior distribution $p(s, pt, et|x, y)$. Algorithm 3 provides an overview of the proposed learning algorithm. In short, we generalize the back-search algorithm in [LHH20a] to a *deduction-abduction* strategy to enable efficient sampling from the posterior distribution of perception, syntax, and semantics.

Deduction For a given example (x, y), we first perform greedy deduction from x to obtain a candidate solution of a compound tree $ct = (x, \hat{s}, \hat{pt}, \hat{et})$. This process is likely to produce a wrong result, thus requiring a separate abduction process to further correct it, detailed below.

Abduction To find a revised solution ct^* that can reach the goal y, we search the neighbors of ct in a top-down manner by performing abduction over perception (s), syntax (pt), and semantics (et), as detailed in Algorithm 4 and illustrated in Fig. 3.4. Our abduction strategy generalizes the perception-only, one-step back-search algorithm described in [LHH20a] to all three levels. The Solve function and the priority used in the top-down search are similarly to the ones in [LHH20a]. The abduction can also be extended to multiple steps, but we only use one step for lower computation overhead. The above deduction-abduction strategy likely behaves as a Metropolis-Hastings sampler for the posterior distribution [LHH20a].

3.5 Experiments and Results

3.5.1 Experimental Setup

Training Both the ResNet-18 and the dependency parser in the proposed ANS model are trained by an Adam optimizer [KB15a] with a learning rate of 10^{-4} and a batch size of 512. The program synthesis module is adapted from DreamCoder [EWN20].

Algorithm 4 Abduction

```
function ABDUCE(ct, y)
    Q=PriorityQueue()
    Q.push(root(ct), y, 1.0)
    while A, y_A, p = Q.pop() do
        A = (i, w, v, arcs)                          ▷ (image, symbol, value, arcs)
        if A.v == y_A then
            return A
        end if
                                                     ▷ Abduce perception
        for w' ∈ Σ do
            A' = A(w → w')
            if A'.v == y_A then
                Q.push(A', y_A, p(A'))
            end if
        end for
                                                     ▷ Abduce syntax
        for arc ∈ arcs do
            A' = rotate(A, arc)
            if A'.v == y_A then
                Q.push(A', y_A, p(A'))
            end if
        end for
                                                     ▷ Abduce semantics
        A' = A(v → y_A)
        Q.push(A', y_A, p(A'))
                                                     ▷ Top-down search
        for B ∈ children(A) do
            y_B = SOLVE(B, A, y_A | θ_l(A.w))
            Q.push(B, y_B, p(B))
        end for
    end while
end function
```

Evaluation Metric We evaluate the models with the accuracy of final results. Note that a predicted result is considered correct when it *exactly* equals to the ground-truth.

Baselines For end-to-end NN baselines, the task of HINT is formulated as a sequence-to-sequence problem: The input is an expression sequence, and the output is a sequence of digits, which is then converted to an integer as the predicted result. We test two popular seq2seq models: (1) BiGRU: the encoder is a bi-directional GRU [CGC14] with three layers, and the decoder is a one-layer GRU; (2) TRAN: a Transformer model [VSP17] with three encoder-layers, three decoder-layers, and four attention heads for each layer. Before being fed into these models, the handwritten expressions are processed by the same ResNet-18 used in ANS. We test models with varied numbers of layers and report ones with the best results. To speed up the convergence, we train all models with a simple curriculum from short expressions to long ones.[2]

3.5.2 Neural-Symbolic v.s. End-to-End Neural Networks

We compare the performance of the proposed neural-symbolic model ANS with end-to-end neural baselines on Hint. As shown in Table 3.1, both BiGRU and TRAN obtain high accuracy on the test subset 1, which indicates that they can generalize over perception very well. However, their performances drop significantly on the test subsets 2∼5, which require systematic generalization over syntax and semantics. Notably, their accuracy is less than 10% on test subsets 3 and 5 that involve larger numbers compared to the training set. This result indicates that the pure neural models do not learn the semantics of concepts in a generalizable way and fail to extrapolate to large numbers. In contrast, the proposed ANS model consistently outperforms BiGRU and TRAN by at least 30 absolute percent across all test subsets 2∼5. This superb performance demonstrates the strong systematic generalization of ANS, including both interpolation and extrapolation w.r.t. syntax and semantics.

How do models extrapolate? Among the generalization capability, we are particularly interested in extrapolation. Based on the experimental results, we firmly believe that the key is *recursion*. In ANS, the extrapolation on syntax is achieved by the transition system of the dependency parser, which recursively applies transition actions to parse arbitrarily long expressions. The extrapolation on semantics is realized by the recursion primitive, *i.e.*, `Y`-combinator. It allows programs to represent recursive functions, which can decompose large

[2]Please refer to the *supp* for the code, experimental logs, and detailed settings.

Table 3.1: The performance comparison of ANS and end-to-end neural networks, *i.e.*, GRU (BiGRU) and Transformer (TRAN).

Input	Model	Test Accuracy (%)					
		Overall	1	2	3	4	5
Symbol (Embedding)	BiGRU	49.71	97.05	63.67	11.58	52.41	12.57
	TRAN	34.58	98.31	29.79	2.91	26.39	2.76
	ANS	**88.36**	**99.26**	**97.56**	**84.66**	**87.65**	**65.37**
Image (ResNet-18)	BiGRU	39.39	87.02	46.17	6.51	40.44	6.47
	TRAN	32.95	87.31	30.74	2.67	31.17	2.55
	ANS	**71.97**	**89.10**	**84.29**	**66.77**	**68.19**	**40.73**

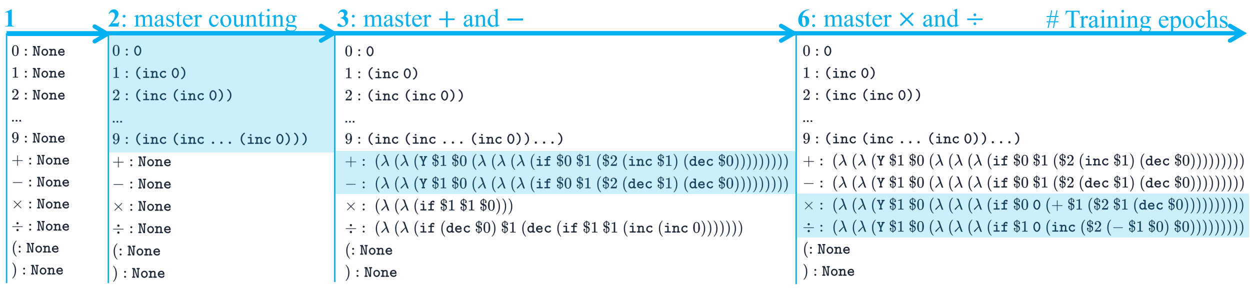


Figure 3.5: **The evolution of semantics in ANS from initial primitives** {`0`,`inc`,`dec`,`if`,`Y`}. The programs representing the semantics of concepts are denoted by lambda calculus (*a.k.a.*. λ-calculus) with De Bruijn indexing. Note that there might be different yet functionally-equivalent programs to represent the same semantics of concepts. Here, we only show one possibility for each concept.

numbers into smaller ones by recursively invoking themselves. For BiGRU, although the recurrent structure in its hidden cells serves as a recursive prior on syntax, no such prior in its representation for semantics. This deficiency explains why BiGRU would achieve a decent accuracy (40.44%) on the test subset 3 (extrapolation only on syntax) but a much lower accuracy (6.51%) on the test subset 4 (extrapolation only on semantics). Taken together, these observations strongly imply that the recursive prior on task-specific representations is the crux of extrapolation, which is also in line with the recent analysis of Graph Neural Network, where it successfully extrapolates algorithmic tasks due to the *task-specific non-linearities* in the architecture or features [XLZ20a, XLZ20b].

3.5.3 Ablation Study

Table 3.2 shows an ablation study on the proposed ANS model. In general, providing the ground-truth meaning of concepts can ease the learning and lead to higher test accuracy. Among the three levels of concepts, perception is the hardest to learn since the handwriting images possess a large variance in terms of the visual appearance. The syntax and semantics are relatively easier to learn, since the recursive prior of the transition-based dependency parser and `Y`-combinator fits the task well.

Table 3.2: **Ablation study on ANS.** ✓indicates that the ground-truth labels are given during training. For each setting (row), we perform three experiments with different random seeds and report the results of the model with the highest *training* accuracy.

Training Setting			**Test Accuracy (%)**					
Per.	**Syn.**	**Sem.**	**Overall**	**1**	**2**	**3**	**4**	**5**
			71.97	89.10	84.29	66.77	68.19	40.73
		✓	86.44	94.53	91.62	89.58	78.22	71.18
	✓		80.14	92.51	90.16	71.32	84.27	56.27
✓			88.36	99.26	97.56	84.66	87.65	65.37
✓	✓		97.81	100.00	100.00	96.66	100.00	90.97
✓		✓	95.84	99.60	98.23	98.09	91.50	88.20
	✓	✓	88.93	94.30	92.19	90.06	82.99	80.88

Fig. 3.5 illustrates the typical pattern of the evolution of semantics in ANS. This pattern is highly in accord with how children learn arithmetic in developmental psychology [CFF99]: The model first masters the semantics of digits as `counting`, then learns $+$ and $-$ as recursive counting, and finally it figures out how to define $\times$ and $\div$ based on the learned programs for $+$ and $-$. Crucially, $\times$ and $\div$ are impossible to be correctly learned before mastering $+$ and $-$. The model is endowed with such an incremental learning capability since the program induction module allows the semantics of concepts to be built compositionally from those learned earlier [EWN20].

3.5.4 Few-shot Concept Learning

We further conduct a preliminary study of few-shot learning to demonstrate the ANS's potential in learning new concepts with limited examples. As shown in Table 3.3, we define four new concepts with common semantics. Their visual appearances are denoted by four unseen handwritten symbols $\{\alpha, \beta, \gamma, \phi\}$, and their syntax is decided by their precedence (*i.e.*, 1 is for $\{+, -\}$ and 2 is for $\{\times, \div\}$). We randomly sample a hundred examples from short to long expressions for training each new concept and fine-tune the ANS model on the new training data.

Table 3.3 shows the test accuracy for each new concept. The proposed ANS model obtains a decent performance with an average overall accuracy of 61.92%. Concepts with more complex semantics ($\{\gamma, \phi\}$) are generally harder to learn than those with simpler semantics ($\{\alpha, \beta\}$).

Table 3.3: **Few-shot concept learning with ANS.**

Per.	**Syn.**	**Sem.**	**Test Accuracy** (%)					
			Overall	**1**	**2**	**3**	**4**	**5**
α	1	$\max(x, y)$	64.08	70.91	81.98	70.79	50.56	40.66
β	1	$\min(x, y)$	72.45	85.45	83.93	81.82	65.91	40.22
γ	2	$(x+y)/2$	56.73	76.36	70.09	61.80	41.94	27.47
ϕ	2	$xy-(x+y)$	54.40	76.36	68.81	41.35	56.04	22.09
avg.	-	-	61.92	77.27	76.20	63.94	53.61	32.61

3.6 Conclusions and Discussions

In this chapter, we take inspiration from how humans learn arithmetic and present a new challenge for the machine learning community, Hint, which serves as a minimal yet complete benchmark for studying systematic generalization of concepts w.r.t. perception, syntax, and semantics. Additionally, we propose a neural-symbolic system, Arithmetic Neural-Symbolic (ANS), to approach this challenge. ANS integrates recent efforts from the disciplines of neural networks, grammar parsing, and program synthesis. One potential future work is to extend our model to other domains and applications.

Extending to other domains. To extend our model to other domains with varieties of semantics, such as visual reasoning [JHM17a, HM19] and question answering [RZL16], we may consider injecting contexts into the semantics of concepts and capture their inherent stochastic nature with probabilistic programs [Gha15, CGH17, GXG18, BCJ19, HBM20].

CHAPTER 4

Competence-aware Curriculum Learning

Humans can progressively learn visual concepts from easy to hard questions. To mimic this efficient learning ability, we propose a competence-aware curriculum for visual concept learning in a question-answering manner. Specifically, we design a neural-symbolic concept learner for learning the visual concepts and a multi-dimensional Item Response Theory (mIRT) model for guiding the learning process with an adaptive curriculum. The mIRT effectively estimates the concept difficulty and the model competence at each learning step from accumulated model responses. The estimated concept difficulty and model competence are further utilized to select the most profitable training samples. Experimental results on CLEVR show that with a competence-aware curriculum, the proposed method achieves state-of-the-art performances with superior data efficiency and convergence speed. Specifically, the proposed model only uses **40% of training data** and converges **three times faster** compared with other state-of-the-art methods.

4.1 Introduction

Humans excel at learning visual concepts and their compositions in a question-answering manner [FAS10, CKA15, GLT18, ZCN17, ZRH20], which requires a joint understanding of vision and language. The essence of such learning skill is the superior capability to connect linguistic symbols (words/phrases) in question-answer pairs with visual cues (appearance/geometry) in images. Imagine a person without prior knowledge of colors is presented with two contrastive examples in Figure 4.1-I. The left images are the same except for color,

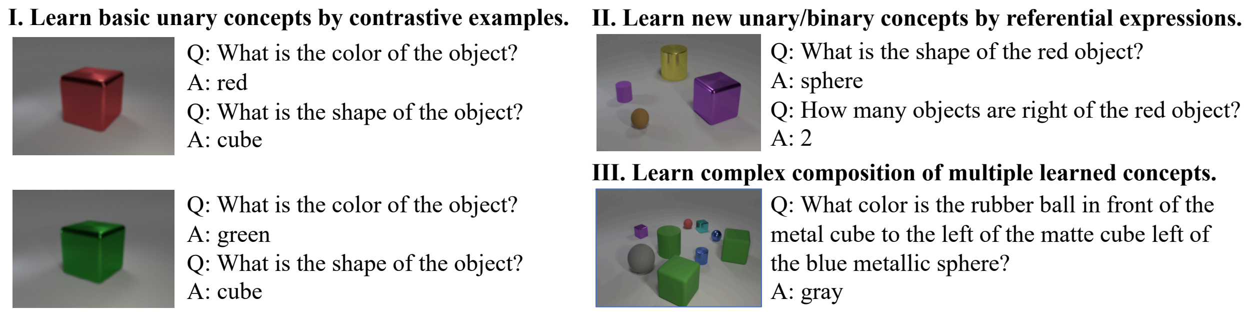


Figure 4.1: The incremental learning of visual concepts in a question-answering manner. Three difficulty levels can be categorized into I) unary concepts from simple questions, II) binary (relational) concepts based on the learned concepts, and III) compositions of visual concepts from comprehensive questions.

and the right question-answer pairs differ only in the descriptions about color. By assuming that the differences in the question-answer pairs capture the differences in appearances, he can learn the concept of color and the appearance of specific colors (*i.e.*, red and green). Besides learning the basic unary concepts from contrastive examples, compositional relations from complex questions consisting of multiple concepts can be further learned, as shown in Figure 4.1-II and -III.

Another crucial characteristic of the human learning process is to start *small* and learn *incrementally*. More specifically, the human learning process is well-organized with a curriculum that introduces concepts progressively and facilitates the learning of new abstract knowledge by exploiting learned concepts. A good curriculum serves as an experienced teacher. By ranking and selecting examples according to the learning state, it can guide the training process of the learner (student) and significantly increase the learning speed. This idea is originally examined in animal training as *shaping* [Ski58, Pet04, KD09] and then applied to machine learning as *curriculum learning* [Elm93, BLC09, GBM17, GHZ18, PSL14].

Inspired by the efficient curriculum, Mao *et al.* [MGK19] proposes a neural-symbolic approach to learn visual concepts with a *fixed* curriculum. Their approach learns from image-question-answer triplets and does not require annotation on images or programs generated from questions. The model is trained with a manually-designed curriculum that includes four stages: (1) learning unary visual concepts; (2) learning relational concepts; (3) learning more complex questions with visual perception fixed; (4) joint fine-tuning all modules. They select questions for each stage by the depths of the latent programs. Their curriculum heavily relies on the manually-designed heuristic that measures the question difficulty and discretizes the curriculum. Such heuristic suffers from three limitations. First, it ignores the variance of difficulties for questions with the same program depths, where different concepts might have various difficulties. Second, the manually-designed curriculum relies on strong human prior knowledge for the difficulties, while such prior may conflict with the inherent difficulty distribution of the training examples. Last but most importantly, it neglects the progress of the learner that evolves along with the training process. More specifically, the order of training samples in the curriculum is nonadjustable based on the model state. This scheme is in stark contrast to the way that humans learn – by *actively* selecting learning samples based on our current learning state, instead of *passively* accepting specific training samples. A desirable learning system should be capable of automatically adjusting the curriculum during the learning process without requiring any prior knowledge, which makes the learning procedure more efficient with less data redundancy and faster convergence speed.

To address these issues and mimic human ability in adaptive learning, we propose a **competence-aware** curriculum for visual concept learning via question answering, where competence represents the capability of the model to recognize each concept. The proposed approach utilizes multi-dimensional Item Response Theory (mIRT) to estimate the **concept difficulty** and **model competence** at each learning step from accumulated model responses. Item Response Theory (IRT) [Bak01, BK04] is a widely adopted method in psychometrics that estimates the human ability and the item difficulty from human responses on various items. We extend the IRT to a mIRT that matches the compositional nature of visual reasoning, and apply variational inference to get a Bayesian estimation for the parameters in mIRT. Based on the estimations of concept difficulty and model competence, we further define a continuous adaptive curriculum (instead of a discretized fixed regime) that selects the most profitable training samples according to the current learning state.

More specifically, the learner can filter out samples with either too naive or too challenging questions. These questions bring either negligible or sharp gradients to the learner, which makes it slower and harder to converge.

With the proposed competence-aware curriculum, the learner can address the aforementioned limitations brought by a fixed curriculum with the following advantages:

1. The concept difficulty and the model competence at each learning step can be inferred effectively from accumulated model responses. It enables the model to distinguish difficulties among various concepts and be aware of its own capability for recognizing these concepts.
2. The question difficulty can be calculated with the estimated concept difficulty and model competence without requiring any heuristics.
3. The adaptive curriculum significantly contributes to the improvement of learning efficiency by relieving the data redundancy and accelerating the convergence, as well as the improvement of the final performance.

We explore the proposed method on the CLEVR dataset [JHM17a], an artificial universe where visual concepts are clearly defined and less correlated. We opt for this synthetic environment because there is little prior work on curriculum learning for visual concepts and there lacks a clear definition of visual concepts in real-world setting. CLEVR allows us to perform controllable diagnoses of the proposed mIRT model in building an adaptive curriculum. section 4.5 further discusses the potentials and challenges of generalizing our method to other domains such as real-world images and natural language processing.

Experimental results show that the visual concept learner with the proposed competence-aware curriculum converges three times faster and consumes only 40% of the training data while achieving similar or even higher accuracy compared with other state-of-the-art models. We also evaluate individual modules in the proposed method and demonstrate their efficacy in section 4.4.

4.2 Related Work

4.2.1 Neural-symbolic Visual Question Answering

Visual question answering (VQA) [MF14, TML14b, QWL15b, JHM17a, GLL17] is a popular task for gauging the capability of visual reasoning systems. Some recent studies [ARD15, ARD16, HAR17, JHM17b, YGL20] focus on learning the neural module networks (NMNs) on the CLEVR dataset. NMNs translate questions into programs, which are further executed over image features to predict answers. The program generator is typically trained on human annotations. Several recent works target on reducing the supervision or increasing the generalization ability to new tasks in NMNs. For example, Johnson *et al*. [JHM17b] replaces the hand-designed syntactic parsers by a learned program generator. Neural-Symbolic VQA [YWG18] explores an object-based visual representation and uses a symbolic executor for inferring the answer. Neural-symbolic concept learner [MGK19] uses a symbolic reasoning process and manually-defined curriculum to bridge the learning of visual concepts, words, and the parsing of questions without explicit annotations. In this work, we build our model on the neural-symbolic concept learner [MGK19] and learn an adaptive curriculum to select the most profitable training samples.

Learning-by-asking (LBA) [MGF17] proposes an interactive learning framework that allows the model to actively query an oracle and discover an easy-to-hard curriculum. LBA uses the expected accuracy improvement over candidate answers as an informativeness measure to pick questions. However, it is costly to compute the expected accuracy improvement for sampled questions since it requires to process all the questions and images through a VQA model. Moreover, the expected accuracy improvement cannot help to learn which specific component of the question contributes to the performance, especially while learning from the answers with little information such as “yes/no”. In contrast, we select questions by explicitly modeling the difficulty of visual concepts, combined with model competence to infer the difficulty of each question.

4.2.2 Curriculum Learning and Machine Teaching

The competence-aware curriculum in our work is related to *curriculum learning* [BLC09, SAJ10, TFL16, GBM17, Sac16, PSL14, GHZ18, PSN19] and *machine teaching* [Zhu15, ZSZ18, LDH17, DHP19, MCV19, Fan18, Wu18]. *Curriculum learning* is firstly proposed by Bengio *et al.* [BLC09] and demonstrates that a dataset order from easy instances to hard ones benefits learning process. The measures of hardness in curriculum learning approaches are usually determined by hand-designed heuristics [SAJ10, TFL16, Sac16, MGK19]. Graves *et al.* [GBM17] explore learning signals based on the increase rates in prediction accuracy and network complexity to adjust data distributions along with training. Self-paced learning [Kum10, Jia14, Jia15, Sac16] quantifies the sample hardness by the training loss and formulates curriculum learning as an optimization problem by jointly modeling the sample selection and the learning objective. These hand-designed heuristics are usually task-specific without any generalization ability to other domains.

Machine teaching [Zhu15, ZSZ18, LDH17] introduces a teacher model that receives feedback from the student model and guides the learning of the student model accordingly. Zhu *et al.* [Zhu15, ZSZ18] assume that the teacher knows the ground-truth model (*i.e.*, the Oracle) beforehand and constructs a minimal training set for the student model. The recent works *learning to teach* [Fan18, Wu18] break this strong assumption of the existence of the oracle model and endow the teacher with the capability of learning to teach via a reinforcement learning framework.

Our work explores curriculum learning in visual reasoning, which is highly compositional and more complex than tasks studied before. Different from previous works, our method requires neither hand-designed heuristics nor an extra teacher model. We combine the idea of *competence* with curriculum learning and propose a novel mIRT model that estimates the concept difficulty and model competence from accumulated model responses.

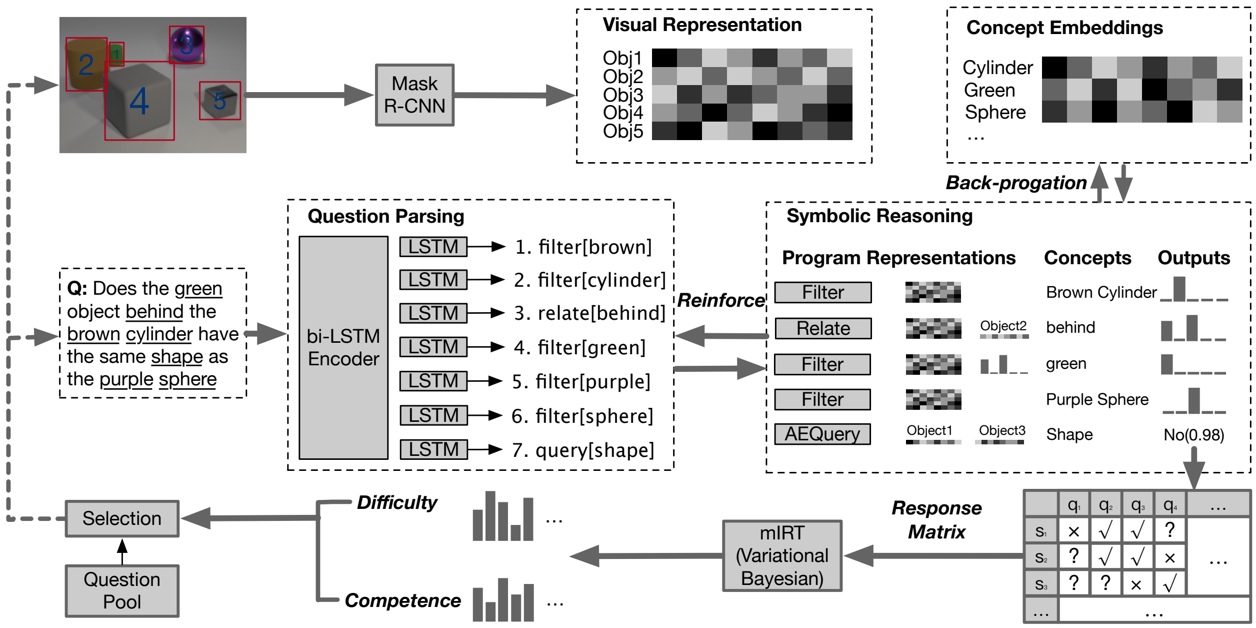


Figure 4.2: The overview of the proposed approach. We use neural symbolic reasoning as a bridge to jointly learn concept embeddings and question parsing. The model responses in the training process are accumulated to estimate concept difficulty and model competence at each learning step with mIRT. The estimations help to select appropriate training samples for the current model. In the response matrix,'✓' or '✕' denotes that the snapshot predicts a correct or wrong answer, and '?' means the snapshot has no response to this question.

4.3 Methodology

In this section, we will discuss the proposed competence-aware curriculum for visual concept learning, as also shown in Figure 4.2. We first describe a neural-symbolic approach to learn visual concepts from image-question-answer triplets. Next, we introduce the background of IRT model and discuss how we derive a mIRT model for estimating concept difficulty and model competence. Finally, we present how to select training samples based on the estimated concept difficulty and model competence to make the training process more efficient.

4.3.1 Neural-Symbolic Concept Learner

We briefly describe the neural-symbolic concept learner. It uses a symbolic reasoning process to bridge the learning of visual concepts and the semantic parsing of textual questions without any intermediate annotations except for the final answers. We refer readers to [MGK19, YWG18] for more details on this model.

Scene Parsing. A scene parsing module develops an object-based representation for each image. Concretely, we adopt a pre-trained Mask R-CNN [HGD17] to generate object proposals from the image. The detected bounding boxes with the original image are sent to a ResNet-34 [HZR16] to extract the object-based features.

Concept Embeddings. By assuming each visual attribute (*e.g.*, shape) contains a set of visual concepts (*e.g.*, cylinder), the extracted visual features are embedded into concept spaces by learnable neural operators of the attributes.

Question Parsing. The question parsing module translates a question in natural language into an executable program in a domain-specific language designed for VQA. The question parser generates the latent program from a question in a sequence-to-sequence manner. A bi-directional LSTM is used to encode the input question into a fixed-length representation. The decoder is an attention-based LSTM, which produces the operations in the program step-by-step. Some operations take concepts as their parameters, such as *Filter[Cube]* and *Relate[Left]*. These concepts are selected from the concepts appearing in the question by the attention mechanism.

Symbolic Reasoning. Given the latent program, the symbolic executor runs the operations in the program with the object-based image representation to derive an answer for the input question. The execution is fully differentiable with respect to the concept embeddings since the intermediate results are represented in a probabilistic manner. Specifically, we keep an attention mask on all object proposals, with each element in the mask denoting the probability that the corresponding object contains certain concepts. The attention mask is fed into the next operation, and the execution continues. The final operation predicts an answer to the question. We refer the readers to the supplementary materials for more details and examples of the symbolic execution process.

Joint Optimizing. We formulate the problem of jointly learning the question parser and the concept embeddings without the annotated programs. Suppose we have a training sample consisting of image I, question Q, and answer A, and we do not observe the latent program l. The goal of training the whole system is to maximize the following conditional probability:

$$p(A|I,Q) = \mathbb{E}_{l \sim p(l|Q)} \; [p(A|l,I)], \tag{4.1}$$

where $p(l|Q)$ is parametrized by the question parser with the parameters θ_l and $p(A|l,I)$ is parametrized by the concept embeddings θ_e (there are no learnable parameters in the symbolic reasoning module). Considering the expectation over the program space in Eq. 4.1 is intractable, we approximate the expectation with Monte Carlo sampling. Specifically, we first sample a program $\hat{l}$ from the question parser $p(l|Q;\theta_l)$ and then apply $\hat{l}$ to obtain a probability distribution over possible answers $p(A|\hat{l},I;\theta_e)$.

Recalling the program execution is fully differentiable w.r.t. the concept embeddings, we learn the concept embeddings by directly maximizing $\log p(A|\hat{l},I;\theta_e)$ using gradient descent and the gradient $\nabla_{\theta_e} \log p(A|\hat{l},I;\theta_e)$ can be calculated through back-propagation. Since the hard selection of $\hat{l}$ through Monte Carlo sampling is non-differentiable, the gradients of the question parser cannot be computed by back-propagation. Instead we optimize the question parser using the REINFORCE algorithm [Wil92]. The gradient of the reward function J over the parameters of the policy is:

$$\nabla J(\theta_l) = \mathbb{E}_{l \sim p(l|Q;\theta_l)} \left[\nabla \log p\left(l|Q;\theta_l\right) \cdot r \right], \tag{4.2}$$

where r denotes the reward. Defining the reward as the log-probability of the correct answer and again, we rewrite the intractable expectation with one Monte Carlo sample $\hat{l}$:

$$\nabla J(\theta_l) = \nabla \log p\left(\hat{l}|Q;\theta_l\right) \cdot [\log p(A|\hat{l},I;\theta_e) - b], \tag{4.3}$$

where b is the exponential moving average of $\log p(A|\hat{l},I;\theta_e)$, serving as a simple baseline to reduce the variance of gradients. Therefore, the update to the question parser at each learning step is simply the gradient of the log-probability of choosing the program, multiplied by the probability of the correct answer using that program.

4.3.2 Background of Item Response Theory (IRT)

Item response theory (IRT) [Bak01, BK04] was initially created in the fields of educational measurement and psychometrics. It has been widely used to measure the latent abilities of subjects (*e.g.*, human beings, robots or AI models) based on their responses to items (*e.g.*, test questions) with different levels of difficulty. The core idea of IRT is that the probability of a correct response to an item can be modeled by a mathematical function of both individual ability and item characteristics. More formally, if we let i be an individual and j be an item, then the probability that the individual i answers the item j correctly can be modeled by a logistic model as:

$$p_{ij} = c_j + \frac{1 - c_j}{1 + e^{-a_j(\theta_i - b_j)}}, \tag{4.4}$$

where θ_i is the latent ability of the individual i and a_j, b_j, c_j are the characteristics of the item j. The item parameters can be interpreted as changing the shape of the standard logistic function: a_j (the discrimination parameter) controls the slope of the curve; b_j (the difficulty parameter) is the ability level, it is the point on θ_i where the probability of a correct response is the average of c_j (min) and 1 (max), also where the slope is maximized; c_j (the guessing parameter) is the asymptotic minimum of this function, which accounts for the effects of guessing on the probability of a correct response for a multi-choice item. Equation 4.4 is often referred to as the three-parameter logistic (3PL) model since it has three parameters describing the characteristics of items. We refer the readers to [Bak01, BK04, ER13] for more background and details on IRT.

4.3.3 Multi-dimensional IRT using Model Responses

Traditional IRT is proposed to model the human responses to several hundred items. However, datasets used in machine learning, especially deep neural networks, often consist of hundreds of thousands of samples or even more. It is costly to collect human responses for large datasets, and more importantly, human responses are not distinguishable enough to estimate the sample difficulties since samples in machine learning datasets are usually straightforward for humans. Lalor *et al*. [LWY16, LWY19] empirically shows on two NLP tasks that IRT models can be fit using machine responses by comparing item parameters learned from the human responses and the responses from an artificial crowd of thousands of machine learning models.

Similarly, we propose to fit IRT models with accumulated model responses (*i.e.*, the predictions of model snapshots) from the training process. Considering the compositional nature of visual reasoning, we propose a multi-dimensional IRT (mIRT) model to estimate the concept difficulty and model competence (corresponding to the subject ability in original IRT), from which the question difficulty can be further calculated.

Formally, we have C concepts, M model snapshots saved from all time steps, and N questions. Let $\Theta = \{\theta_{ic}\}_{i=1..M}^{c=1...C}$, where θ_{ic} is the i-th snapshot's competence on the c-th concept, and $B = \{b_c\}^{c=1...C}$, where b_c is the difficulty of the c-th concept, $\mathcal{Q} = \{q_{jc}\}_{j=1...N}^{c=1...C}$, where q_{jc} is the number of the c-th concept in the j-th question and g_j is the probability of guessing the correct answer to the j-th question, $\mathcal{Z} = \{z_{ij}\}_{i=1...M}^{j=1...N}$, where $z_{ij} \in \{0, 1\}$ be the response of the i-th snapshot to the j-th question (1 if the model answers the question correctly and 0 otherwise). The probability that the snapshot i can correctly recognize the concept c is formulated by a logistic function:

$$p_{ic}(\theta_{ic}, b_c) = \frac{1}{1 + e^{-(\theta_{ic} - b_c)}}. \tag{4.5}$$

Then the probability that the snapshot i answers the question j correctly is calculated as:

$$p(z_{ij}=1|\theta_i,B)=g_j+(1-g_j)\prod_{c=1}^{C}p_{ic}^{q_{jc}}. \tag{4.6}$$

The probability that the snapshot i answers the question j incorrectly is:

$$p(z_{ij}=0|\theta_i,B)=1-p(z_{ij}=1|\theta_i,B). \tag{4.7}$$

The total data likelihood is:

$$p(\mathcal{Z}|\Theta,B)=\prod_{i=1}^{M}\prod_{j=1}^{N}p(z_{ij}|\theta_i,B). \tag{4.8}$$

This formulation is also referred to as conjunctive multi-dimensional IRT [Rec85, Rec09].

4.3.4 Variational Bayesian Inference for mIRT

The goal of fitting an IRT model on observed responses is to estimate the latent subject abilities and item parameters. In traditional IRT, the item parameters are usually estimated by Marginal Maximum Likelihood (MML) via an Expectation-Maximization (EM) algorithm [BA81], where the subject ability parameters are randomly sampled from a normal distribution and marginalized out. Once the item parameters are estimated, the subject abilities are scored by maximum a posterior (MAP) estimation based on their responses to items. However, the EM algorithm is not computational efficient on large datasets. One feasible way for scaling up is to perform variational Bayesian inference on IRT [NNM16, LWY19]. The posterior probability of the parameters in mIRT can be written as:

$$p(\Theta,B|\mathcal{Z})=\frac{p(\mathcal{Z}|\Theta,B)p(\Theta)p(B)}{\int_{\Theta,B}p(\Theta,B,\mathcal{Z})}, \tag{4.9}$$

where $p(\Theta), p(B)$ are the priors distribution of Θ and B. The integral over the parameter space in Eq 4.9 is intractable. Therefore, we approximate it by a factorized variational distribution on top of an independence assumption of Θ and B:

$$q(\Theta, B) = \prod_{i=1,c=1}^{M,C} \pi_{ic}^{\theta}\left(\theta_{ic}\right) \prod_{c=1}^{C} \pi_{c}^{b}\left(b_{c}\right), \tag{4.10}$$

where π_{ic}^{θ} and π_{c}^{b} denote Gaussian distributions for model competences and concept difficulties, respectively. We adopt the Kullback-Leibler divergence (KL-divergence) to measure the distance of p from q, which is defined as:

$$D_{\mathrm{KL}}(q \| p) := \mathbb{E}_{q(\Theta,B)} \log \frac{q(\Theta, B)}{p(\Theta, B|\mathcal{Z})}, \tag{4.11}$$

where $p(\Theta, B|\mathcal{Z})$ is still intractable. We can further decompose the KL-divergence as:

$$D_{\mathrm{KL}}(q \| p) = \mathbb{E}_{q(\Theta,B)} \left[\log \frac{q(\Theta, B)}{p(\Theta, B, \mathcal{Z})} + \log p(\mathcal{Z})\right]. \tag{4.12}$$

In other words, we also have:

$$\log p(\mathcal{Z}) = D_{\mathrm{KL}}(q \| p) - \mathbb{E}_{q(\Theta,B)} \log \frac{q(\Theta, B)}{p(\Theta, B, \mathcal{Z})} \tag{4.13}$$

$$= D_{\mathrm{KL}}(q \| p) + \mathcal{L}(q). \tag{4.14}$$

As the log evidence $\log p(\mathcal{Z})$ is fixed with respect to q, maximizing the final term $\mathcal{L}(q)$ minimizes the KL divergence of q from p. And since $q(\Theta, B)$ is a parametric distribution we can sample from, we can use Monte Carlo sampling to estimate this quantity. Since the KL-divergence is non-negative, $\mathcal{L}(q)$ is an evidence lower bound (ELBO) of $\log p(\mathcal{Z})$. By maximizing the ELBO with an Adam optimizer [KB15b] in Pyro [BCJ18], we can estimate the parameters in mIRT.

4.3.5 Training Samples Selection Strategy

The proposed model can estimate the question difficulty for the current model competence without looking at the ground-truth images and answers. It facilitates the active selection for future training samples. More specifically, we can easily calculate the probability that the model answers a given question correctly from Eq. 4.5 and Eq. 4.6 (without guessing) using estimated Θ and b. This probability serves as an indicator of the question difficulty for the learner in each stage. The higher the probability, the easier the question. To select appropriate training samples, we rank the questions and filter out the hardest questions by setting a probability lower bound (LB) and the easiest questions by a probability upper bound (UB). Algorithm 5 summarizes the overall training process. We will discuss the influence of LB and UB on the learning process in Section 4.4.5.

Algorithm 5 Competence-aware Curriculum Learning

Initialization: the training set $\mathcal{D} = \{(I_j, Q_j, A_j)\}_{j=1}^{N}$, concept difficulty $B^{(0)}$, model competence $\Theta^{(0)}$, concept learner $\phi^{(0)}$, accumulated responses $\mathcal{Z} = \{\}$

for $t = 1$ to T **do**

$\Theta^{(t)}, B^{(t)} = \arg\max_{\Theta,B} \mathcal{L}(q; \Theta^{(t-1)}, B^{(t-1)}, \mathcal{Z})$

$\mathcal{D}^{(t)} = \{(I, Q, A) : \mathrm{LB} \leqslant p(Q; \Theta^{(t)}, B^{(t)}) \leqslant \mathrm{UB}\}$

$\phi^{(t)}, \mathcal{Z}^{(t)} = \mathrm{Train}(\phi^{(t-1)}, \mathcal{D}^{(t)})$

$\mathcal{Z} = \mathcal{Z} \cup \mathcal{Z}^{(t)}$

end for

4.4 Experiments

4.4.1 Experimental Setup

Dataset. We evaluate the proposed method on the CLEVR dataset [JHM17a], which consists of a training set of 70k images and ∼700k questions, and a validation set of 15k images and

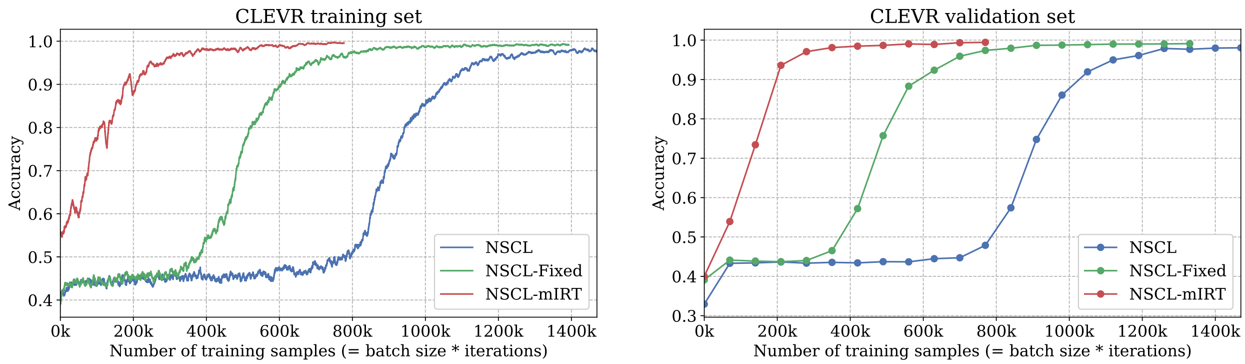


Figure 4.3: The learning curves of different model variants on the CLEVR dataset.

∼150k questions. The proposed model selects questions from the training set during learning, and we evaluate our model on the entire validation set.

Models. To analyze the performance of the proposed approach, We conduct experiments by comparing with several model variants:

- **FiLM-LBA**: the best model from [MGF17].
- **NSCL**: the neural-symbolic concept learner [MGK19] without using any curriculum. Questions are randomly sampled from the training set.
- **NSCL-Fixed**: NSCL following a manually-designed discretized curriculum.
- **NSCL-mIRT**: NSCL following a continuous curriculum built by the proposed mIRT estimator.

Please refer to the supplementary materials for detailed model settings and learning techniques during training.

4.4.2 Training Process & Model Performance

Figure 4.3 shows the accuracies of the model variants at different timesteps on the training set (left) and validation set (right). Notably, the proposed NSCL-mIRT converges almost 2 times faster than NSCL-Fixed and 3 times faster than NSCL (*i.e.*, 400k v.s. 800k v.s. 1200k). Although NSCL-mIRT spends extra time to estimate the parameters of the mIRT model, such time cost is negligible compared to other time spent in training (less than 1%).

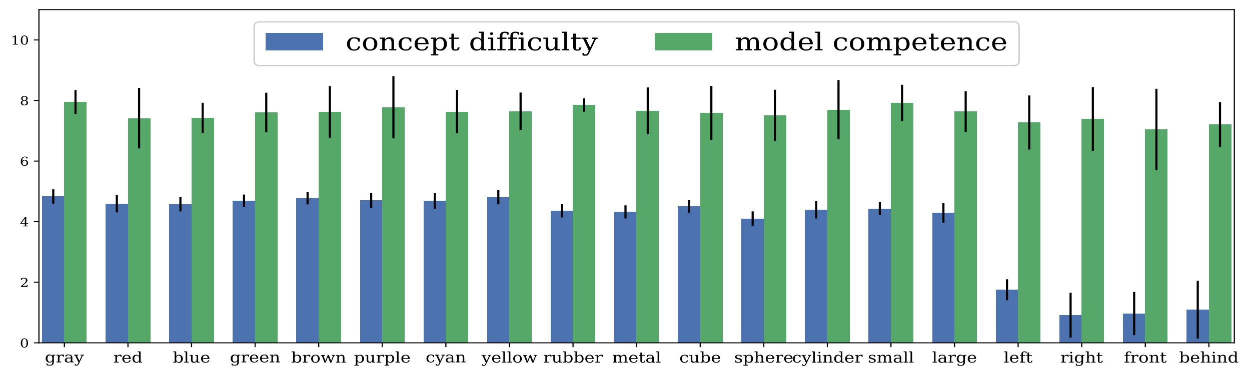


Figure 4.4: The estimated concept difficulty and model competence at the final iteration.

From Table 4.1, we can see that NSCL-mIRT consistently outperforms FilM-LBA at various iterations, which demonstrates the preeminence of mIRT in building an adaptive curriculum.

Besides, NSCL-mIRT consumes less than 300k unique questions for training when it converges. It indicates that NSCL-mIRT saves about 60% of the training data, which largely eases the data redundancy problems. It provides a promising direction for designing a data-efficient curriculum and helping current data-hungry deep learning models save time and money cost during data annotation and model training.

Moreover, NSCL-mIRT obtains even higher accuracy than NSCL and NSCL-Fixed. This indicates that the adaptive curriculum built by the multi-dimensional IRT model not only remarkably increases the speed of convergence and reduces the data consumption during the training process, but also leads to better performance, which also verifies the hypothesis made by Bengio *et al*. [BLC09].

4.4.3 Multi-dimensional IRT

The estimated concept difficulty and model competence after converging is shown in Figure 4.4 for studying the performance of the mIRT model. Several critical observations are: (1) The spatial relations (*i.e.*, left/right/front/behind) are the easiest concepts. It satisfies our intuition since the model only needs to exploit the object positions to determine their spatial relations without dealing with appearance. The spatial relations are learned during the late

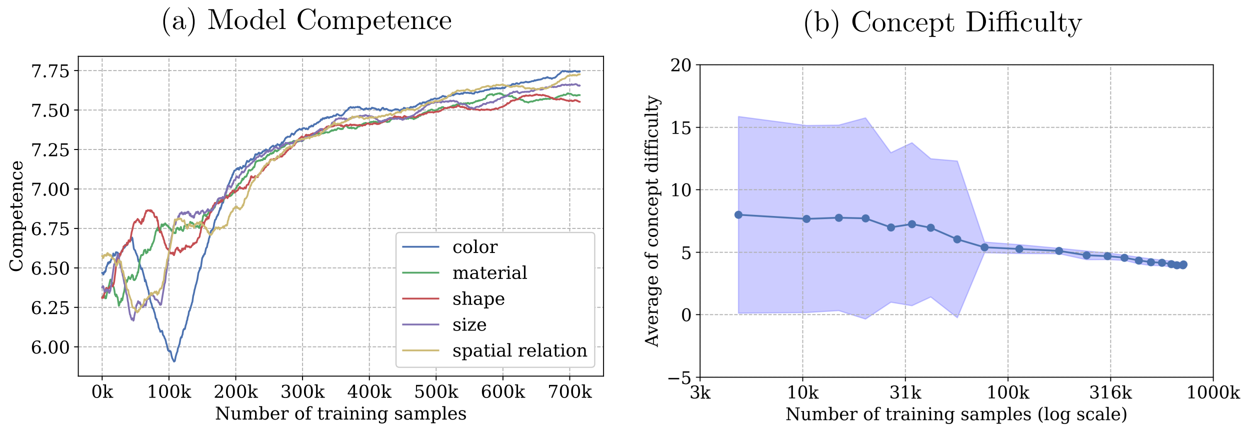


Figure 4.5: (a) The estimated model competence at various iterations for different attributes. The value for each attribute type is averaged from the visual concept it contains. (b) The estimated concept difficulty at various iterations. The shaded area represents the variance of the estimations.

stages since they appear more frequently in complex questions to connect multiple concepts. (2) Colors are the most difficult concepts. The model needs to capture the subtle differences in the appearance of objects to distinguish eight different colors. (3) The model competence scores surpass the concept difficulty scores for all the concepts. This result corresponds to the nearly perfect accuracy ($>99\%$) on all questions and concepts.

Figure 4.5a shows the estimation of the model competence for each attribute type at various iterations. We can observe that model competence consistently increases throughout the training. Figure 4.5b shows the estimations of the concept difficulty at different learning steps. As the training progresses, the estimations become more stable with smaller variance since more model responses are accumulated.

4.4.4 Concept Learner

We apply the count-based concept evaluation metric proposed in [MGK19] to measure the performance of the concept learner, which evaluates the visual concepts on synthetic questions with a single concept such as "How many *red* objects are there?" Table 4.2 presents the results by comparing with several state-of-the-art methods, which includes methods based on neural module network with programs (IEP [JHM17a]) and neural attentions without

Table 4.1: The VQA accuracy of different models on the CLEVR validation set at various iterations. NSCL and NSCL-Fixed continue to improve with longer training steps, which is not shown for space limit.

Models	70k	140k	280k	420k	630k	700k
FiLM-LBA [MGF17]	51.2	**76.2**	92.9	94.8	95.2	97.3
NSCL	43.3	43.4	43.3	43.4	44.5	44.7
NSCL-Fixed	44.1	43.9	44.0	57.2	92.4	95.9
NSCL-mIRT	**53.9**	73.4	**97.1**	**98.5**	**98.9**	**99.3**

Table 4.2: The accuracy of the visual attributes of different models. Please refer to the supplementary materials for detailed performance on each visual concept (*i.e.*, "gray" and "red" in color attribute).

Model	Overall	Color	Material	Shape	Size
IEP [JHM17a]	90.6	91.0	90.0	89.9	90.6
MAC [HM18]	95.9	98.0	91.4	94.4	94.2
NSCL-Fixed [MGK19]	98.7	99.0	98.7	98.1	99.1
NSCL-mIRT	**99.5**	**99.5**	**99.7**	**99.4**	**99.6**

Table 4.3: Comparisons of the VQA accuracy on the CLEVR validation set with other models.

Model	Overall	Count	Cmp Num.	Exist	Query Attr.	Cmp Attr.
Human	92.6	86.7	86.4	96.6	95.0	96.0
IEP [JHM17a]	96.9	92.7	98.7	97.1	98.1	98.9
FiLM [PSV17]	97.6	94.5	93.8	99.2	99.2	99.0
MAC [HM18]	98.9	97.2	99.4	99.5	99.3	99.5
NSCL [MGK19]	98.9	98.2	99.0	98.8	99.3	99.1
NS-VQA [YWG18]	**99.8**	**99.7**	**99.9**	**99.9**	**99.8**	**99.8**
NSCL-mIRT	99.5	98.9	99.0	99.7	99.7	99.6

Table 4.4: The VQA accuracy on CLEVR validation set with different LBs and UBs in the question selection strategy. Both LB and UB are in log scale.

(LB,UB)	70k	140k	210k	280k	560k	770k
(-10, 0)	44.39	52.01	63.04	73.5	97.93	99.01
(-5, 0)	53.75	69.55	82.44	95.31	98.92	99.27
(-3, 0)	51.38	55.97	58.33	65.11	69.57	70.01
(-5, -0.5)	42.06	52.67	80.46	95.54	98.41	99.06
(-5, -0.75)	**53.91**	**73.42**	**93.6**	**97.07**	99.04	**99.50**
(-5, -1)	44.57	63.65	82.95	94.38	**99.15**	99.48

programs (MAC [HAR17]). Our model achieves nearly perfect performance across visual concepts and outperforms all other approaches. This means the model can learn visual concepts better with an adaptive curriculum. Our model can also be applied to the VQA. Table 4.3 summarizes the VQA accuracy on the CLEVR validation split. Our approach achieves comparable performance with state-of-the-art methods.

4.4.5 Question Selection strategy

The question selection strategy is controlled by two hyper-parameters: the lower bound (LB) and upper bound (UB). We conduct experiments by learning with different LBs and UBs, and Table 4.4 shows the VQA accuracy at various iterations. It reveals that the proper lower bound can effectively filter out too hard questions and accelerate the learning at the early stage of the training, as shown in the first three rows. Similarly, a proper upper bound helps to filter out too easy questions at the late stage of the training when the model has learned most concepts. Please refer to the supplementary material for the visualization of selected questions at various iterations.

4.5 Conclusions and Discussions

We propose a competence-aware curriculum for visual concepts learning via question answering. We design a multi-dimensional IRT model to estimate concept difficulty and model competence at each training step from the accumulated model responses generated by different model snapshots. The estimated concept difficulty and model competence are further used to build an adaptive curriculum for the visual concept learner. Experiments on the CLEVR dataset show that the concept learner with the proposed competence-aware curriculum converges three times faster and consumes only 40% of the training data while achieving similar or even higher accuracy compared with other state-of-the-art models.

In the future, our work can be potentially applied to *real-world images* like GQA [HM19] and VQA-v2 [GKS17] datasets, by explicitly modeling the relationship among visual concepts. However, there are still unsolved challenges for real-world images. Specifically, compared with synthetic images in CLEVR, real-world images have a much larger vocabulary of visual concepts. For example, as shown in [AHB18], there are over 2,000 visual concepts in MSCOCO images. Usually, these concepts are automatically mined from image captions and scene graphs. Thus some of them are highly correlated like "huge" and "large", and some of them are very subjective like "busy" and "calm". Such a large and noisy vocabulary of visual concepts is challenging for the mIRT model since current visual concepts are assumed to be independent. It also requires a much longer time to converge when maximizing the ELBO to fit the mIRT model with more concepts. A potential solution is to consider the hierarchical structure of visual concept space and correlations among the concepts and incorporate commonsense knowledge to handle subjective concepts.

More importantly, the competence-aware curriculum can be adapted to other domains that possess compositional structures such as natural language processing. Specifically, in neural machine translation task [SVL14, BCB15], mIRT can be used to model the difficulty and competence of translating different words/phrases and build a curriculum to increase learning speed and data efficiency. mIRT can also be used in the task of semantic parsing [DL16, LBL16b, LNB18a] that transforms natural language sentences (*e.g.*, instructions or queries) into logic forms (*e.g.*, lambda-calculus or SQL). The difficulty and competence of different logic predicates can also be estimated by the mIRT model.

CHAPTER 5

Case Study: Solving Math Word Problems with Weak Supervision

Previous neural solvers of math word problems (MWPs) are learned with full supervision and fail to generate diverse solutions. In this work, we address this issue by introducing a *weakly-supervised* paradigm for learning MWPs. Our method only requires the annotations of the final answers and can generate various solutions for a single problem. To boost weakly-supervised learning, we propose a novel *learning-by-fixing* (LBF) framework, which corrects the misperceptions of the neural network via symbolic reasoning. Specifically, for an incorrect solution tree generated by the neural network, the *fixing* mechanism propagates the error from the root node to the leaf nodes and infers the most probable fix that can be executed to get the desired answer. To generate more diverse solutions, *tree regularization* is applied to guide the efficient shrinkage and exploration of the solution space, and a *memory buffer* is designed to track and save the discovered various fixes for each problem. Experimental results on the Math23K dataset show the proposed LBF framework significantly outperforms reinforcement learning baselines in weakly-supervised learning. Furthermore, it achieves comparable top-1 and much better top-3/5 answer accuracies than fully-supervised methods, demonstrating its strength in producing diverse solutions.

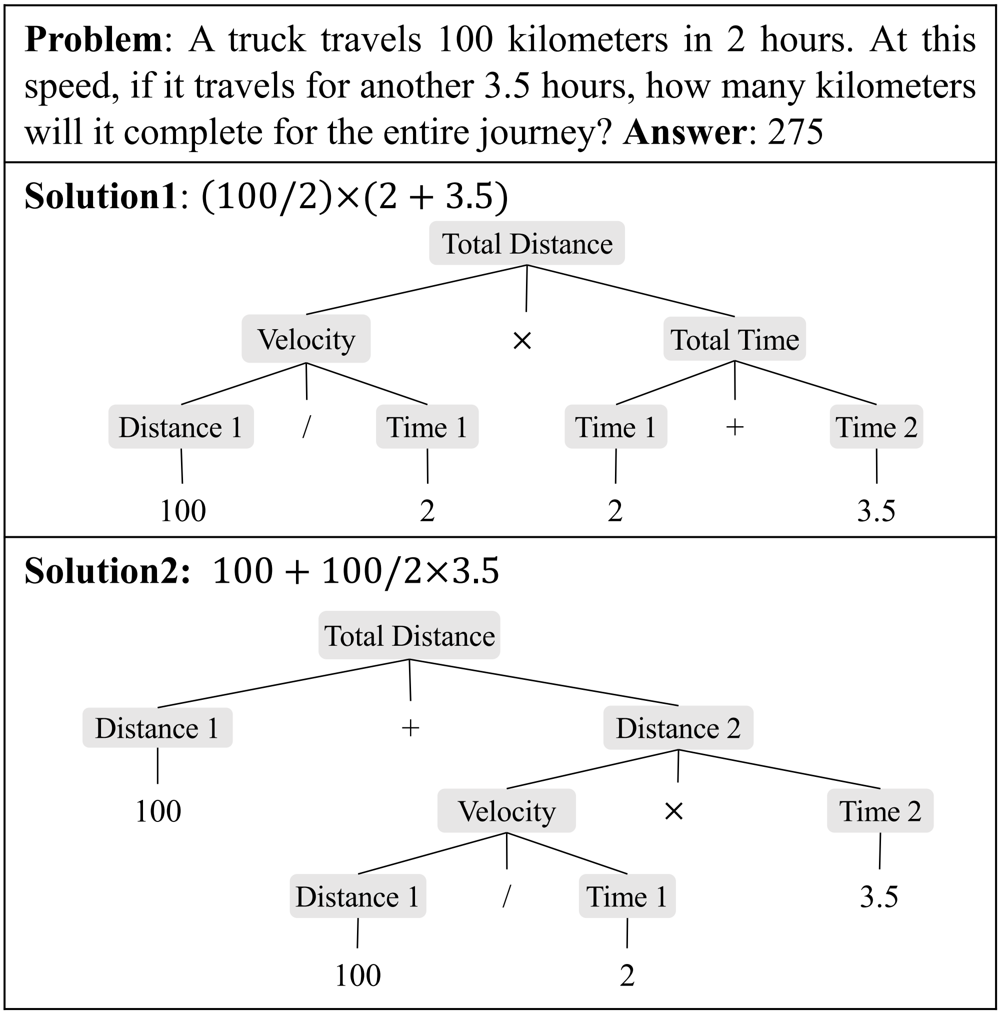


Figure 5.1: Exemplar MWP with multiple solutions.

5.1 Introduction

Solving math word problems (MWPs) poses unique challenges for understanding natural-language problems and performing arithmetic reasoning over quantities with commonsense knowledge. As shown in Figure 5.1, a typical MWP consists of a short narrative describing a situation in the world and asking a question about an unknown quantity. To solve the MWP in Figure 5.1, a machine needs to extract key quantities from the text, such as ”100 kilometers” and ”2 hours”, and understand the relationships between them. General mathematical knowledge like ”distance = velocity × time” is then used to calculate the solution.

Researchers have recently focused on solving MWPs using neural-symbolic models [LYD17, WLS17, HLL18, WWC18, XS19]. These models usually consist of a neural perception module (*i.e.*, Seq2Seq or Seq2Tree) that maps the problem text into a solution expression or tree, and a symbolic module which executes the expression and generates the final answer. Training these models requires the full supervision of the solution expressions.

However, these fully-supervised approaches have three drawbacks. First, current MWP datasets only provide one solution for each problem, while there naturally exist multiple solutions that give different paths of solving the same problem. For instance, the problem in Figure 5.1 can be solved by "$(100/2) \times (2+3.5)$" if we first calculate the speed and then multiply it by the total time; alternatively, we can solve it using "$100 + 100/2 \times 3.5$" by summing the distances of the first and second parts of the journey. The models trained with full supervision on current datasets are forced to fit the given solution and cannot generate diverse solutions. Second, annotating the expressions for MWPs is time-consuming. However, a large amount of MWPs with their final answers can be mined effortlessly from the internet (*e.g.*, online forums). How to efficiently utilize these partially-labeled data without the supervision of expressions remains an open problem. Third, current supervised learning approaches suffer from the train-test discrepancy. The fully-supervised learning methods optimize expression accuracy rather than answer accuracy. However, the model is evaluated by the answer accuracy on the test set, causing a natural performance gap.

To address these issues, we propose to solve the MWPs with *weak supervision*, where only the problem texts and the final answers are required. By directly optimizing the answer accuracy rather than the expression accuracy, learning with weak supervision naturally addresses the train-test discrepancy. Our model consists of a tree-structured neural model similar to [XS19] to generate the solution tree and a symbolic execution module to calculate the answer. However, the symbolic execution module for arithmetic expressions is non-differentiable with respect to the answer accuracy, making it infeasible to use back-propagation to compute gradients. A straightforward approach is to employ policy gradient methods like REINFORCE [Wil92] to train the neural model. The policy gradient methods explore the solution space and update the policy based on generated solutions that happen to hit the correct answer. Since the solution space is large and incorrect solutions are abandoned with zero reward, these methods usually converge slowly or fail to converge.

To improve the efficiency of weakly-supervised learning, we propose a novel *fixing* mechanism to learn from incorrect predictions, which is inspired by the human ability to learn from failures via abductive reasoning [Mag09, Zho19b]. The fixing mechanism propagates the error from the root node to the leaf nodes in the solution tree and finds the most probable *fix* that can generate the desired answer. The fixed solution tree is further used as a pseudo label to train the neural model. Figure 5.2 shows how the fixing mechanism corrects the wrong solution tree by tracing the error in a top-down manner.

Furthermore, we design two practical techniques to traverse the solution space and discover possible solutions efficiently. First, we observe a positive correlation between the number of quantities in the text and the size of the solution tree (the number of leaf nodes in the tree), and propose a *tree regularization* technique based on this observation to limit the range of possible tree sizes and shrink the solution space. Second, we adopt a *memory buffer* to track and save the discovered fixes for each problem with the fixing mechanism. All memory buffer solutions are used as pseudo labels to train the model, encouraging the model to generate more diverse solutions for a single problem.

In summary, by combining the fixing mechanism and the above two techniques, the proposed **learning-by-fixing** (LBF) method contains an exploring stage and a learning stage in each iteration, as shown in Figure 5.2. We utilize the fixing mechanism and tree regularization to correct wrong answers in the exploring stage and generate fixed expressions as pseudo labels. In the learning stage, we train the neural model using these pseudo labels.

We conduct comprehensive experiments on the Math23K dataset [WLS17]. The proposed LBF method significantly outperforms the reinforcement learning baselines in weakly-supervised learning and achieves comparable performance with several fully-supervised methods. Furthermore, our proposed method achieves significantly better answer accuracies of all the top-3/5 answers than fully-supervised methods, illustrating its advantage in generating diverse solutions. The ablative experiments also demonstrate the efficacy of the designed algorithms, including the fixing mechanism, tree regularization, and memory buffer.

5.2 Related Work

5.2.1 Math Word Problems

Recently, there emerges various question-answering tasks that require human-like reasoning abilities [QWL15a, TML14a, ZGJ19a, DWD19, HWJ19, ZGF20, ZZZ20a, LHH20c, YJD20]. Among them, solving mathematical word problems (MWPs) is a fundamental and challenging task.

Previous studies of MWPs range from traditional rule-based methods [Fle85, Bak07, YYG10], statistical learning methods [KZB14, ZDC15, MB16, RR17, HSL16], semantic-parsing methods [SWL15, KHS15, HSL17] to recent deep learning methods [LYD17, WLS17, HLL18, RKH18, WWC18, WZZ19, CC19, XS19, ZWL20].

In particular, Deep Neural Solver (DNS) [WLS17] is a pioneering work that designs a Seq2seq model to solve MWPs and achieves promising results. [XS19] propose a tree-structured neural solver to generate the solution tree in a goal-driven manner. All these neural solvers learn the model with full supervision, where the ground-truth intermediate representations (e.g., expressions, programs) are given during training. To learn the solver with less supervision, [KHS15] use a discriminative model to solve MWPs in a weakly-supervised way. They utilize separate modules to extract features, construct expression trees, and score the likelihood, which is different from the current end-to-end neural solvers. [UCC16], [ZDC15], and [KZB14] use mixed supervision, where one dataset has only annotated equations, and the other has only final answers. However, for the set with final answers, they also depend on pre-defined equation templates. [CLY20] apply a neural-symbolic reader on MathQA[AGL19], which is a large-scale dataset with fully-specified operational programs. They have access to the ground truth programs for a small fraction of training samples at the first iterations of training.

Unlike these methods, the proposed LBF method requires only the supervision of the final answer and generates diverse solutions by keeping a memory buffer. Notably, it addresses the sparse reward problem in policy gradient methods using a fixing mechanism that propagates error down a solution tree and finds the most probable fix.

5.2.2 Neural-Symbolic Learning for NLP

Neural-symbolic learning has been applied to solve NLP tasks with weak supervision, such as semantic parsing and program synthesis [LBL16a, GPL17b, LNB18b, ALS19b, LHH20c]. Similar to MWP, they generate intermediate symbolic representations with a neural network and execute the intermediate representation with a symbolic reasoning module to get the final result. Typical approaches for such neural-symbolic models use policy gradient methods like REINFORCE since the symbolic execution module is non-differentiable. For example, Neural Symbolic Machines [LBL16c] combines REINFORCE with a maximum-likelihood training process to find good programs. [GPL17b] augment reinforcement learning with the maximum marginal likelihood so that probability is distributed evenly across consistent programs. Memory Augmented Policy Optimization (MAPO) [LNB18b] formulates its learning objective as an expectation over a memory buffer of high-reward samples and a separate expectation outside the buffer, which helps accelerate and stabilize policy gradient training. Meta Reward Learning [ALS19b] uses an auxiliary reward function to provide feedback beyond a binary success or failure. Since these methods can only learn from sparse successful samples, they suffer from cold start and inefficient exploration of large search spaces. Recently, [DZ17], [DXY19], and [Zho19a] introduce abductive learning, which states that human misperceptions can be corrected via abductive reasoning. In this work, we follow the abductive learning method [LHH20b] and propose a novel fixing mechanism to learn from negative samples, significantly accelerating and stabilizing the weakly-supervised learning process. We further design the tree regularization and memory buffer techniques to efficiently shrink and explore the solution space.

5.3 Weakly-Supervised MWPs

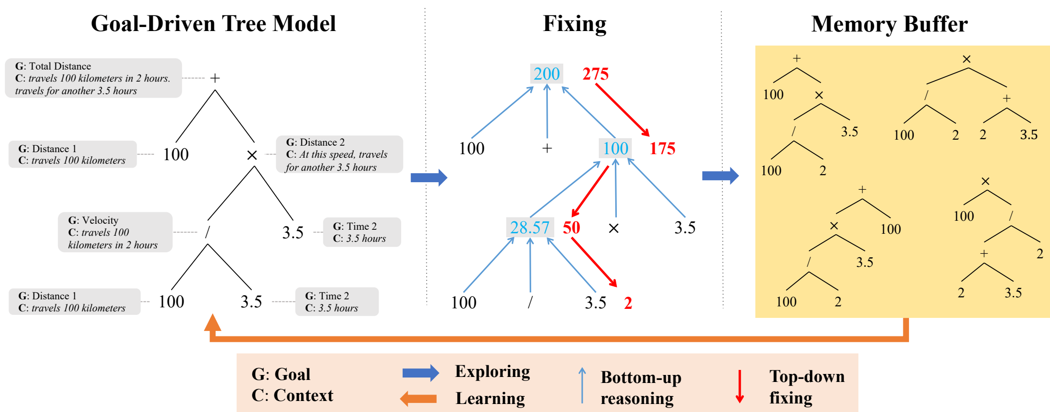


Figure 5.2: Overview of our proposed learning-by-fixing (LBF) method. It shows the process for learning the example in Figure 5.1. LBF works by iteratively exploring the solution space and learning the MWP solver. Exploring: the problem first goes through the GTS module and produces a tentative solution using tree regularization. Then the fixing mechanism diagnoses this solution by propagating the correct answer in a top-down manner. The fixed solution is then added to the memory buffer. Learning: all solutions in the memory buffer are used as pseudo labels to train the GTS module using a cross-entropy loss function.

In this section, we define the weakly-supervised math word problems and describe the goal-driven tree model originated from [XS19]. Then we introduce the proposed learning-by-fixing method, as also shown in Figure 5.2.

5.3.1 Problem Definition

A math word problem is represented by an input problem text P. The machine learning model with parameters θ requires to translate P into an intermediate expression T, which is executed to compute the final answer y. In fully-supervised learning, we learn from the ground truth expression T and the final answer y. The learning objective is to maximize the data likelihood $p(T,y|P;\theta)=p_\theta(T|P)p(y|T)$, where computing y given T is a deterministic process. In contrast, in the weakly-supervised setting, only P and y are observed, while T is hidden. In other words, the model is required to generate an unknown expression from the problem text. The expression is then executed to get the final answer.

5.3.2 Goal-driven Tree-Structured Model

A problem text P consists of words and numeric values. The model takes in problem text P and generates a solution tree T. Let V^{num} denote the ordered list of numeric values in P according to their order in the problem text. Generally, T may contain constants $V^{con} = \{1, 2, \pi\}$, mathematical operators $V^{op} = \{+, -, \times, \div, \wedge\}$, and numeric values V^{num} from the problem text P. Therefore, the target vocabulary of P is denoted as $\Sigma = V^{op} \cup V^{con} \cup V^{num}$ and it varies between problems due to different V^{num}.

To generate the solution tree, we adopt the goal-driven tree-structured neural model (GTS) [XS19], which first encodes the problem text into its goal and then recursively decomposes it into sub-goals in a top-down manner.

Problem Encoding. Each word of the problem text is encoded into a contextual representation. Specifically, for a problem $P = w_1 w_2 ... w_n$, each word w_i is first converted to a word embedding $\mathbf{w}_i$. Then the sequence of embeddings is inputted to a bi-directional GRU [CVG14] to produce a contextual word representation: $\mathbf{h}_i = \overrightarrow{\mathbf{h}_i} + \overleftarrow{\mathbf{h}_i}$, where $\overrightarrow{\mathbf{h}_i}, \overleftarrow{\mathbf{h}_i}$ are the hidden states of the forward and backward GRUs at position i, respectively.

Solution Tree Generation. The tree generation process is designed as a preorder tree traversal (root-left-right). The root node of the solution tree is initialized with a goal vector $\mathbf{q}_0 = \overrightarrow{\mathbf{h}_n} + \overleftarrow{\mathbf{h}_0}$.

For a node with goal $\mathbf{q}$, we first derive a context vector $\mathbf{c}$ by an attention mechanism to summarize relevant information from the problem:

$$a_i = \text{softmax}(\mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{q}, \mathbf{h}_i])) \tag{5.1}$$

$$\mathbf{c} = \sum_i a_i \mathbf{h}_i \tag{5.2}$$

where $\mathbf{v}_a$ and $\mathbf{W}_a$ are trainable parameters. Then the goal $\mathbf{q}$ and the context $\mathbf{c}$ are used to predict the token of this node from the target vocabulary Σ. The probability of token t is defined as:

$$s(t|\mathbf{q}, \mathbf{c}) = \mathbf{w}_n^\top \tanh(\mathbf{W}_s[\mathbf{q}, \mathbf{c}, \mathbf{e}(t)]) \tag{5.3}$$

$$p(t|\mathbf{q}, \mathbf{c}) = \text{softmax}(s(t|\mathbf{q}, \mathbf{c})) \tag{5.4}$$

where $\mathbf{e}(t)$ is the embedding of token t:

$$\mathbf{e}(t) = \begin{cases} \mathbf{M}_{op}(t) & \text{if } t \in V^{op} \\ \mathbf{M}_{con}(t) & \text{if } t \in V^{con} \\ \mathbf{h}_{loc(t,P)} & \text{if } t \in V^{num} \end{cases} \tag{5.5}$$

where $\mathbf{M}_{op}$ and $\mathbf{M}_{con}$ are two trainable embeddings for operators and constants, respectively. For a number token, its embedding is the corresponding hidden state $\mathbf{h}_{loc(t,P)}$ from the encoder, where $loc(t, P)$ is the index of t in the problem P. The predicted token $\hat{t}$ is:

$$\hat{t} = \arg\max_{t \in \Sigma} p(t|\mathbf{q}, \mathbf{c}) \tag{5.6}$$

If the predicted token is a number token or constant, the node is terminated and its goal is realized by the predicted token; otherwise, the predicted token is an operator and the current goal is decomposed into left and right sub-goals combined by the operator. Please refer to the *supplementary material* for more details about the goal decomposition process.

Answer Calculation. The generated solution tree is transformed into a reasoning tree $\hat{T}$ by creating auxiliary non-terminal nodes in place of the operator nodes to store the intermediate results, and the original operator nodes are attached as child nodes to the corresponding auxiliary nodes. Then the final answer $\hat{y}$ is calculated by executing $\hat{T}$ to the value of the root node in a bottom-up manner.

5.3.3 Learning-by-Fixing

5.3.3.1 Fixing Mechanism

Drawing inspiration from humans' ability to correct and learn from failures, we propose a fixing mechanism to correct the wrong solution trees via abductive reasoning following [LHH20b] and use the fixed solution trees as pseudo labels for training. Specifically, we find the most probable fix for the wrong prediction by back-tracking the reasoning tree and propagating the error from the root node into the leaf nodes in a top-down manner.

The key ingredient in the fixing mechanism is the 1-step fix (1-FIX) algorithm which assumes that only one symbol in the reasoning tree can be substituted. As shown by the 1-Fix function in Algorithm 6, the 1-step fix starts from the root node of the reasoning tree and gradually searches down to find a fix that makes the final output equal to the ground-truth. The search process is implemented with a priority queue, where each element is defined as a fix-tuple (A, α_A, p):

- A is the current visiting node.
- α_A is the expected value on this node, which means if the value of A is changed to α_A, $\hat{T}$ will execute to the ground-truth answer y.
- p is the visiting priority, which reflects the probability of changing the value of A.

In 1-FIX, error propagation through the solution tree is achieved by a *solve* function, which aims at computing the expected value of a child node from its parent's expected value.

Supposing B is A's child node and α_A is the expected value of A, the $solve(B, A, \alpha_A)$ function works as following:

- If B is A's left or right child, we directly solve the equation $\alpha_B \bigoplus child_R(A) = \alpha_A$ or $child_L(A) \bigoplus \alpha_B = \alpha_A$ to get B's expected value α_B, where $\bigoplus$ denotes the operator.
- If B is an operator node, we try to replace B with all other operators and check whether the new expression can generate the correct answer. That is, $child_L(A)\ \alpha_B\ child_R(A) = \alpha_A$ where α_B is now an operator. If there is no α_B satisfying this equation, the solve function returns none.

Please refer to the *supplementary material* for the definition of the visiting priority as well as the illustrative example of the 1-FIX process.

To search the neighbors of $\hat{T}$ within multi-step distance, we extend the 1-step fix to multi-step by incorporating a RandomWalk function. As shown in Algorithm 6, if we find a fix by 1-FIX, we return this fix; otherwise, we randomly change one leaf node in the reasoning tree to another symbol within the same set (*e.g.*, operators V^{op}) based on the probability in Equation 5.4. This process will be repeated for certain iterations until it finds a fix for the solution.

5.3.3.2 Solution Space Exploration

Tree Regularization While [LHH20b] assumes the length of the intermediate representation is given, the expression length is unknown in weakly-supervised learning. Thus, the original solution space is infinite since the predicted token decides whether to continue the generation or stop. Therefore, it is critical to shrink the solution space, *i.e.*, control the size of the generated solution trees. If the size of the generated solution tree varies a lot from the target size, it would be challenging for the solution or its fix to hit the correct answer. Although the target size is unknown, we observe a positive correlation between the target

Algorithm 6 Fixing Mechanism

```
Input: reasoning tree \hat{T}, ground-truth answer y
T^{(0)} = \hat{T}
for i ← 0 to m do
    T* = 1-Fix(T^{(i)}, y)
    if T* ≠ ∅ then
        return T*
    else
        T^{(i+1)} = RandomWalk(T^{(i)})
    end if
end for
return ∅

function 1-Fix(T, y)
q = PriorityQueue(), S = the root node of T
q.push(S, y, 1)
while (A, α_A, p) = q.pop() do
    if A ∈ Σ then
        T* = \hat{T}(A → α_A)
        return T*
    end if
    for B ∈ child(A) do
        α_B = solve(B, A, α_A)
        if not (B ∈ Σ and α_B ∉ Σ) then
            q.push(B, α_B, p(B → α_B))
        end if
    end for
end while
return ∅
```

size and the number of quantities in text. Regarding this observation as a tree size prior, we design a tree regularization algorithm to generate a solution tree with a target size and regularize the size in an empirical range. Denote the size of a solution tree $\mathrm{Size}(T)$ as the number of leaf nodes including quantities, constants, and operators. The prior range of $\mathrm{Size}(T)$ given the length of the numeric value list $\mathrm{len}(V^{num})$ is defined as:

$$
\begin{aligned}
&\mathrm{Size}(T) \in [\mathrm{minSize}(T), \mathrm{maxSize}(T)]\\
&\mathrm{minSize}(T) = a_{min}\mathrm{len}(V^{num}) + b_{min}\\
&\mathrm{maxSize}(T) = a_{max}\mathrm{len}(V^{num}) + b_{max}
\end{aligned}
\tag{5.7}
$$

where $a_{min}, b_{min}, a_{max}, b_{max}$ are the hyperparameters. The effect of these hyperparameters will be discussed in Table 5.2.

We further propose a *tree regularization* algorithm to decode a solution tree with a given size. To generate a tree of a given size l, we design two rules to produce a prefix-order expression during the preorder tree decoding:

1. The number of operators cannot be greater than $\lfloor l/2 \rfloor$.
2. Except the l-th position, the number of numeric values (quantities and constants) cannot be greater than the number of operators.

These two rules are inspired by the syntax of prefix notation (a.k.a, normal Polish notation) for mathematical expressions. The rules shrink the target vocabulary Σ in Equation 5.6 so that the tree generation can be stopped when it reaches the target size. Figure 5.3 shows illustrative examples of the tree regularization algorithm.

With tree regularization, we can search the possible fixes within a given range of tree size $[\text{minSize}(T), \text{maxSize}(T)]$ for each problem.

Memory Buffer. We adopt a memory buffer to track and save the discovered fixes for each problem. The memory buffer enables us to seek multiple solutions for a single problem and use all of them as pseudo labels for training, which encourages diverse solutions. Formally, given a problem P and its buffer β, the learning objective is to minimize the negative log-likelihood of all fixed expressions in the buffer:

$$J(P, \beta) = -\sum_{T^* \in \beta} \log p(T^*|P) \tag{5.8}$$

5.3.4 Learning-by-Fixing Framework

The complete learning-by-fixing method is described in Algorithm 7. In the exploring state, we use the fixing mechanism and tree regularization to discover possible fixes for the wrong

$V^{num} = \{100, 2, 3.5\}$
$V^{op} = \{+, -, \times, \div, \wedge\}$
$V^{con} = \{1, 2, \pi\}$

Target size $l = 5$

$\times$	②	V^{op}
$\div$	N/A	$V^{op} \cup V^{num} \cup V^{con}$
100	①	$V^{num} \cup V^{con}$
2	①	$V^{num} \cup V^{con}$
3.5	①	$V^{num} \cup V^{con}$

Prefix: $\times \div$ 100 2 3.5

Target size $l = 7$

$\times$	②	V^{op}
$\div$	N/A	$V^{op} \cup V^{num} \cup V^{con}$
100	N/A	$V^{op} \cup V^{num} \cup V^{con}$
2	N/A	$V^{op} \cup V^{num} \cup V^{con}$
$+$	②	V^{op}
2	①	$V^{num} \cup V^{con}$
3.5	①	$V^{num} \cup V^{con}$

Prefix: $\times \div$ 100 2 $+$ 2 3.5

Figure 5.3: Tree regularization for the problem in Figure 5.1 given different target sizes. The three columns are the generated tokens, the effective rules, and the target vocabularies shrunk by the rules, respectively.

trees generated by the neural network, and put them into a buffer. In the learning stage, we train the model with all the solutions in the memory buffer by minimizing the loss function in Equation 5.8.

Algorithm 7 Learning-by-Fixing

Input: training set $\mathcal{D} = \{(P_i, y_i)\}_{i=1}^N$
memory buffer $\mathcal{B} = \{\beta_i\}_{i=1}^N$, the GTS model θ
for $P_i, y_i, \beta_i \in (\mathcal{D}, \mathcal{B})$ **do**
  ▷*Exploring*
  $\hat{T}_i$ = GTS $(P; \theta)$
  $T_i^* = m$-FIX$(\hat{T}_i, y_i)$
  if $T_i^* \neq \varnothing$ and $T_i^* \notin \beta_i$ **then**
    $\beta_i \leftarrow \beta_i \cup \{T_i^*\}$
  end if
  ▷*Learning*
  $\theta = \theta - \nabla_\theta J(P_i, \beta_i)$
end for

5.4 Experimental Results

5.4.1 Experimental Setup

Dataset. We evaluate our proposed method on the Math23K dataset [WLS17]. It contains 23,161 math word problems annotated with solution expressions and answers. For the weakly-supervised setting, we only use the problems and final answers and discard the expressions. We do cross-validation following the setting of [XS19].

Evaluation Metric. We evaluate the model performance by answer accuracy, where the generated solution is considered correct if it executes to the ground-truth answer. Specifically, we report answer accuracies of all the top-1/3/5 predictions using beam search. It evaluates the model's ability to generate multiple possible solutions.

Models. We conduct experiments by comparing our methods with variants of weakly-supervised learning methods. Specifically, we experiment with two inference models: Seq2Seq with bidirectional Long Short Memory network (BiLSTM) [WSC16] and GTS [XS19], and train with four learning strategies: REINFORCE, MAPO [LNB18b], LBF, LBF-w/o-M (without memory buffer). MAPO is a state-of-the-art method in semantic parsing task that extends the REINFORCE with augmented memory. Both models are also trained with the tree regularization algorithm. We also compare with the fully-supervised learning methods to demonstrate our superiority in generating diverse solutions. In the ablative studies, we analyze the effect of the proposed tree regularization and the length of search steps in fixing mechanism.

5.4.2 Comparisons with State-of-the-art

Table 5.1 summarizes the answer accuracy of different weakly-supervised learning methods and the state-of-the-art fully-supervised approaches. The proposed learning-by-fixing framework significantly outperforms the policy gradient baselines like REINFORCE and MAPO, on both the Seq2seq and the GTS models. It demonstrates the strength of our proposed LBF method in weakly-supervised learning. The GTS-LBF-fully model is trained by initializing the memory buffer with all the ground-truth expressions. It demonstrates that by extending to the fully-supervised setting, our model maintains the top-1 accuracy while significantly improving solutions' diversity. We believe that learning MWPs with weak supervision is a promising direction. It requires fewer annotations and allows us to build larger datasets with less cost.

Model		**Accuracy**(%)
Fully-Supervised		
Retrieval		47.2
Classification		57.9
LSTM		51.9
CNN		42.3
DNS		58.1
Seq2seqET		66.7
Stack-Decoder		65.8
T-RNN		66.9
GTS		74.3
Graph2Tree		**74.8**
GTS-LBF-fully		74.1
Weakly-Supervised		
Seq2seq	REINFORCE	1.2
	MAPO	10.7
	LBF-w/o-M	44.7
	LBF	43.6
GTS	REINFORCE	15.8
	MAPO	20.8
	LBF-w/o-M	58.3
	LBF	**59.4**

Table 5.1: Answer accuracy on the Math23K dataset. We compare variants of models with our LBF method.

5.4.3 Convergence Speed

Figure 5.4 shows the learning curves of different weakly-supervised learning methods for the GTS model. The proposed LBF method converges significantly faster and achieves higher accuracy compared with other methods. Both the REINFORCE and MAPO take a long time to start improving, which indicates the policy gradient methods suffer from the cold-start and need time to accumulate rewarding samples.

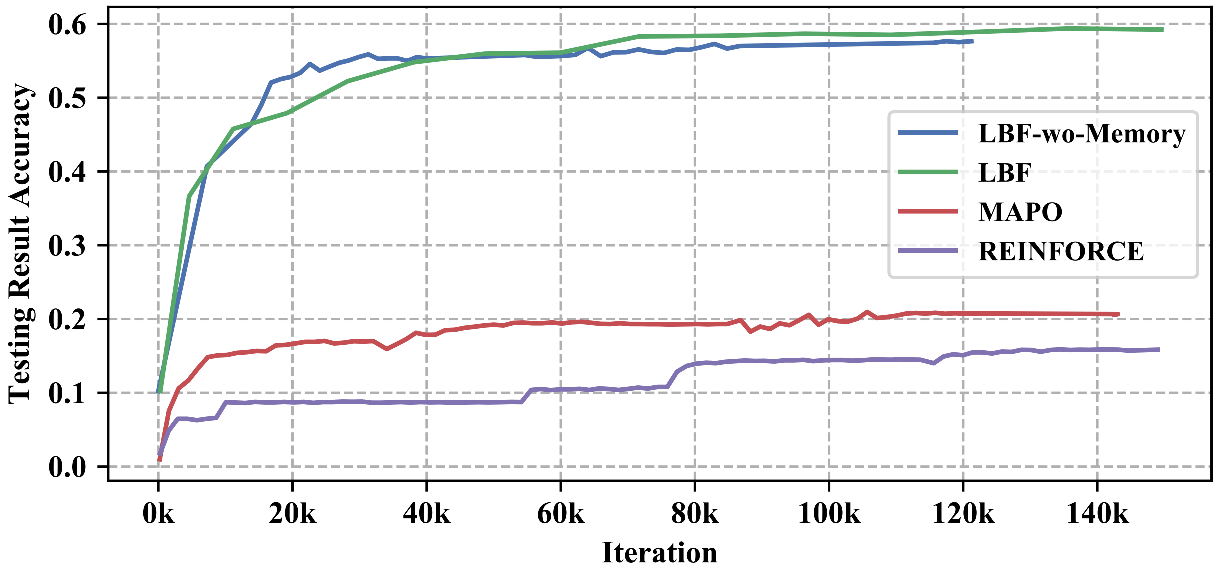


Figure 5.4: The learning curves of the GTS model using different weakly-supervised learning methods.

5.4.4 Diverse Solutions with Memory Buffer

To evaluate the ability to generate diverse solutions, we report the answer accuracies of all the top-1/3/5 solutions on the test set using beam search, denoted as Acc@1/3/5, as shown in Table 5.2. In the weakly-supervised scenario, GTS-LBF achieves slightly better Acc@1 accuracy and much better Acc@3/5 accuracy than GTS-LBF-w/o-M. In the fully supervised scenario, GTS-LBF-fully achieves comparable Acc@1 accuracy and much better Acc@3/5 accuracy than the original GTS model. Particularly, GTS-LBF-fully outperforms GTS by 21% and 26% in terms of Acc@3/5 accuracy. It reveals the efficacy of the memory buffer in encouraging diverse solutions in both weakly-supervised learning and fully-supervised learning.

Model	Tree Size	Acc@1	Acc@3	Acc@5
Fully Supervised				
GTS		**74.3**	42.2	30.0
GTS-LBF-fully		74.1	**63.4**	**56.3**
Weakly Supervised				
GTS-LBF-w/o-M	[1,+∞)	∼0	∼0	∼0
	[2n-1,2n+1]	55.3	26.2	19.3
	[2n-1,2n+3]	58.3	27.7	20.3
	[2n-3,2n+5]	56.7	27.7	20.6
GTS-LBF	[1,+∞)	∼0	∼0	∼0
	[2n-1,2n+1]	56.7	45.3	39.1
	[2n-1,2n+3]	**59.4**	**49.6**	**45.2**
	[2n-3,2n+5]	57.6	49.3	45.2

Table 5.2: Answer accuracies of all the top-1/3/5 solutions decoded using beam search, denoted as Acc@1/3/5.

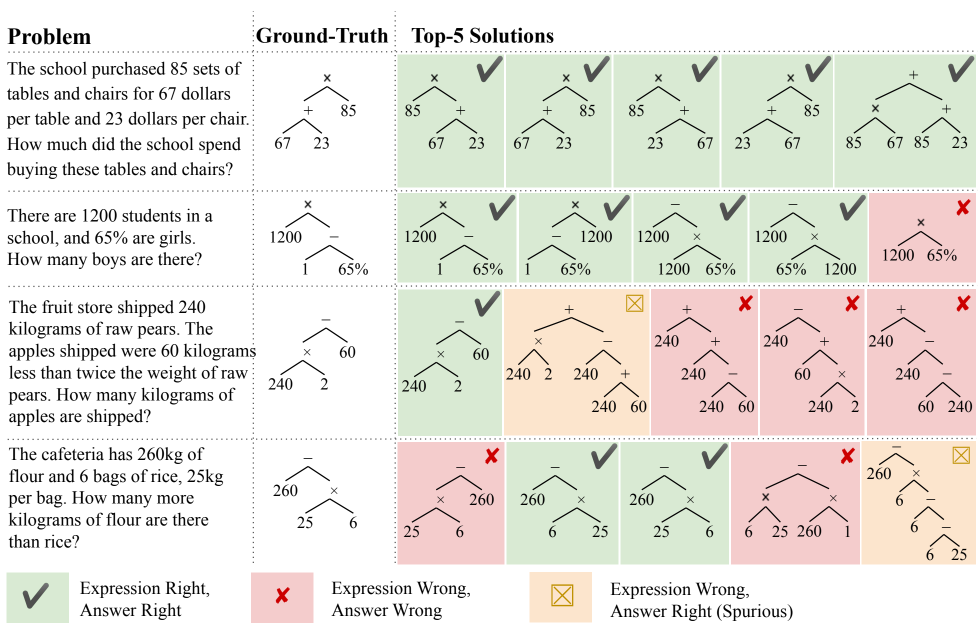


Figure 5.5: Qualitative results on the Math23K dataset. We visualize the solution trees generated by our method.

5.4.5 Qualitative Analysis

We visualize several examples of the top-5 predictions of GTS-LBF in Figure 5.5. In the first example, the first solution generated by our model is to sum up the prices of a table and a chair first, and then multiply it by the number of pairs of tables and chairs. Our model can also produce another reasonable solution (the fifth column) by deriving the prices of tables and chairs separately and then summing them up.

One caveat for the multiple solutions is that some solutions have different solution trees but are equivalent by switching the order of numeric values or subtrees, as shown in the first four solutions of the first problem in Figure 5.5. In particular, multiplication and addition are commutative, and our model learns and exploits this property to generate equivalent solutions with different tree structures.

	Right	Wrong	Spurious
Acc@1	58.6	40.6	0.56
Acc@3	49.3	50.4	0.27
Acc@5	44.9	54.8	0.32

Table 5.3: Human evaluation on the generated solutions (%).

The first solution to the fourth problem in Figure 5.5 is a typical error case of our model due to the wrong prediction of the problem goal. Another failure type is the spurious solutions, which are correct but not meaningful answers, such as the second solution of the third problem in Figure 5.5. To test how frequent the spurious solutions appear, we randomly select 500 examples from the test set, and ask three human annotators to determine whether each generated expression is right, wrong, or spurious. Table 5.3 provides the human evaluation results, and it shows that spurious solutions are rare in our model.

5.4.6 Ablative Analyses

Tree Regularization. We test different choices of the hyperparameters defined by Equation 5.7 in tree regularization. As shown in Table 5.2, the model without tree regularization,

$i.e.$, tree size $\in [1, +\infty)$, fails to converge and gets nearly 0 accuracy. The best range for the solution tree size is $[2n-1, 2n+3]$, where $n = \text{len}(V^{num})$. We provide an intuitive interpretation of this range: for a problem with n quantities, (1) $n-1$ operators are needed to connect n quantities, which leads to the lower bound of tree size to $2n-1$; (2) in certain cases, the constants or quantities are used more than once, leading to a rough upper bound of $2n+3$. Therefore, we use $[2n-1, 2n+3]$ as the default range in our implementations. Empirically, this range covers 88% of the lengths of the given ground-truth expressions in the Math23K dataset, providing an efficient prior for tree size.

Number of Search Steps Table 5.4 shows the comparison of various step lengths in the m-FIX algorithm. In most cases, increasing the step length improves the chances of correcting wrong solutions, thus improving the performance.

Models \ Steps	**1**	**10**	**50 (default)**	**100**
Seq2seq-LBF-w/o-M	41.9	43.4	44.7	**47.8**
Seq2seq-LBF	43.9	**45.7**	43.6	44.6
GTS-LBF-w/o-M	51.2	54.6	**58.3**	57.8
GTS-LBF	52.5	55.8	59.4	**59.6**

Table 5.4: Accuracy (%) using various search steps.

5.5 Conclusions and Discussions

In this work, we propose a weakly-supervised paradigm for learning MWPs and a novel learning-by-fixing framework to boost the learning. Our method endows the MWP learner with the capability of learning from wrong solutions, thus significantly improving the answer accuracy and learning efficiency. The fixing mechanism endows the MWP learner with the capability of learning from wrong solutions, thus significantly improving the answer accuracy and learning efficiency. The tree regularization efficiently shrinks and explores the solution space by limiting the tree size within an empirical range. The memory buffer encourages the model to learn diverse solutions for each problem. One future direction of the proposed model is to prevent generating equivalent or spurious solutions during training, possibly by making the generated solution trees more interpretable with semantic constraints.

CHAPTER 6

Conclusion

This dissertation introduces our contributions to closing the loop of recognition and reasoning to seek a unified framework for artificial general intelligence. To study this area, we take inspiration from how humans learn arithmetic and present a new benchmark, HINT, which serves as a minimal yet complete benchmark for studying the systematic generalization of concepts w.r.t. perception, syntax, and semantics.

To solve tasks like HINT, we propose a new Neural-Grammar-Symbolic model, which uses grammar parsing to bridge neural perception and symbolic reasoning. The proposed NGS model is a realization of a symbol system with combinatorial syntactic and semantic structures, which is arguably a necessary and sufficient means of general intelligence. However, it is very challenging to optimize such a heterogeneous model using weak supervision.

To address this optimization issue, we derive a general learning framework from a probabilistic perspective and the key to successful learning is to perform efficient sampling from the posterior distribution of the intermediate symbolic representations given the raw inputs and the final supervision in the maximum likelihood estimation. Inspired by the human ability to learn from failures via abductive reasoning, we propose a novel deduction-abduction strategy to coordinate the learning of three heterogeneous modules in the proposed model. The deduction-abduction strategy makes the learning much more efficient than previous methods. We also prove that the multi-step abduction process behaves as a Metropolis-Hastings sampler for the posterior distribution of the intermediate symbolic representations.

The proposed framework for the integration of recognition and reasoning is potentially useful for a wide range of applications, such as visual reasoning, math word problems, and grounded grammar induction. In this dissertation, we present a case study of solving math word problems with weak supervision. Experimental results demonstrate that the proposed framework outperforms the baselines by a large margin.

In the end, we summarize several fundamental and practical research directions inspired by this dissertation:

More Efficient Optimization Although the proposed learning strategy significantly outperforms the existing baselines, there still exists a large room for improvement to obtain human-level performance w.r.t. data efficiency and learning speed. A potential solution might be to perform the optimization in a continuous space using a homogeneous model and then project the learned model into a symbol system that has stronger generalization capability.

More Generalizable Representation To achieve general artificial intelligence, a fundamental obstacle to be addressed is how to help the machine learn from fewer examples and achieve strong generalizations to novel scenarios. Machine learning researchers have proposed various algorithms such as meta-learning and zero-shot learning to address this problem. However, most of them can only be generalized to in-distribution data. A promising future direction is to explore more generalizable representation from raw signals, through learning a generative model to establish the relationships between raw observations and underlying hidden representations.

More Real Applications Most of this dissertation focuses on laying the theoretical foundations for closing the loop of recognition and reasoning and we envision that the proposed framework can be transferred to a wide range of domains. For example, this framework might enable a cognitive robot to process visual signals efficiently, communicate with humans using gestures and dialogues, and collaborate with humans by inferring the human minds and actions in the future.

REFERENCES

[AGL19] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. "MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms." In *NAACL-HLT*, 2019.

[AHB18] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. "Bottom-up and top-down attention for image captioning and visual question answering." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6077–6086, 2018.

[AKL17] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular multitask reinforcement learning with policy sketches." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 166–175. JMLR. org, 2017.

[ALS19a] Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. "Learning to generalize from sparse and underspecified rewards." *arXiv preprint arXiv:1902.07198*, 2019.

[ALS19b] Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. "Learning to Generalize from Sparse and Underspecified Rewards." In *ICML*, 2019.

[ARD15] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. "Neural Module Networks." *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39–48, 2015.

[ARD16] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. "Learning to Compose Neural Networks for Question Answering." In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[AWT18] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018.

[BA81] R Darrell Bock and Murray Aitkin. "Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm." *Psychometrika*, 1981.

[Bak01] Frank B Baker. *The basics of item response theory.* ERIC, 2001.

[Bak07] Yefim Bakman. "Robust Understanding of Word Problems with Extraneous Information." In *arxiv.General Mathematics*, 2007.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *ICLR*, 2015.

[BCJ18] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. "Pyro: Deep Universal Probabilistic Programming." *Journal of Machine Learning Research*, 2018.

[BCJ19] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. "Pyro: Deep universal probabilistic programming." *Journal of Machine Learning Research*, **20**(1):973–978, 2019.

[BGB17a] Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. "Deepcoder: Learning to write programs." In *International Conference on Learning Representations (ICLR)*, 2017.

[BGB17b] Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. "Neural-symbolic learning and reasoning: A survey and interpretation." *arXiv preprint arXiv:1711.03902*, 2017.

[BGH09] Sebastian Bader, Artur S d'Avila Garcez, and P Hitzler. "Extracting propositional rules from feed-forward neural networks by means of binary decision diagrams." In *Proceedings of the 5th International Workshop on Neural-Symbolic Learning and Reasoning, NeSy*, volume 9, pp. 22–27. Citeseer, 2009.

[BHD18] Rudy Bunel, Matthew Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. "Leveraging grammar and reinforcement learning for neural program synthesis." *arXiv preprint arXiv:1805.04276*, 2018.

[BK04] Frank B Baker and Seock-Ho Kim. *Item response theory: Parameter estimation techniques*. CRC Press, 2004.

[BLC09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning." In *International Conference on Machine Learning (ICML)*, 2009.

[BMN18] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. "Systematic generalization: what is required and can it be learned?" In *International Conference on Learning Representations (ICLR)*, 2018.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[CC19] Ting-Rui Chiang and Yun-Nung Chen. "Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems." *ArXiv*, **abs/1811.00720**, 2019.

[CFF99] Thomas P Carpenter, Elizabeth Fennema, M Loef Franke, Linda Levi, and Susan B Empson. "Children's mathematics." *Cognitively Guided*, 1999.

[CGC14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555*, 2014.

[CGH17] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. "Stan: a probabilistic programming language." *Grantee Submission*, **76**(1):1–32, 2017.

[CHY19] Yixin Chen, Siyuan Huang, Tao Yuan, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. "Holistic++ scene understanding: Single-view 3D holistic scene parsing and human pose estimation with human-object interaction and physical commonsense." In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[CKA15] Grzegorz Chrupała, Akos Kádár, and Afra Alishahi. "Learning language through pictures." *Association for Computational Linguistics(ACL)*, 2015.

[CLK21] Yixin Chen, Qing Li, Deqian Kong, Yik Lun Kei, Song-Chun Zhu, Tao Gao, Yixin Zhu, and Siyuan Huang. "Yourefit: Embodied reference understanding with language and gesture." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1385–1395, 2021.

[CLY20] Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, D. Song, and Quoc V. Le. "Neural Symbolic Reader: Scalable Integration of Distributed and Symbolic Representations for Reading Comprehension." In *ICLR*, 2020.

[CM14] Danqi Chen and Christopher D Manning. "A fast and accurate dependency parser using neural networks." In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[CS14] Paco Calvo and John Symons. *The architecture of cognition: Rethinking Fodor and Pylyshyn's systematicity challenge.* MIT Press, 2014.

[CVG14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *EMNLP*, 2014.

[DDG18] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. "Embodied question answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2054–2063, 2018.

[DGL18] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. "Neural modular control for embodied question answering." *arXiv preprint arXiv:1810.11181*, 2018.

[DHP19] Sanjoy Dasgupta, Daniel Hsu, Stefanos Poulis, and Xiaojin Zhu. "Teaching a black-box learner." In *ICML*, 2019.

[DK86] Simon Duane and John B Kogut. "The theory of hybrid stochastic algorithms." *Nuclear Physics B*, **275**(3):398–420, 1986.

[DL16] Li Dong and Mirella Lapata. "Language to logical form with neural attention." *ACL*, 2016.

[DUB17] Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdelrahman Mohamed, and Pushmeet Kohli. "Robustfill: Neural program learning under noisy i/o." In *Proceedings of International Conference on Machine Learning (ICML)*, 2017.

[DWD19] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. "DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs." In *NAACL-HLT*, 2019.

[DXY19] Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. "Bridging Machine Learning and Logical Reasoning by Abductive Learning." In *Advances in Neural Information Processing Systems*, pp. 2811–2822, 2019.

[DZ17] Wang-Zhou Dai and Zhi-Hua Zhou. "Combining logical abduction and statistical induction: Discovering written primitives with human knowledge." In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[EG18] Richard Evans and Edward Grefenstette. "Learning explanatory rules from noisy data." *Journal of Artificial Intelligence Research*, **61**:1–64, 2018.

[EGL19] Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. "A tale of two explanations: Enhancing human trust by explaining robot behavior." *Science Robotics*, **4**(37), 2019.

[EKS18] Mark Edmonds, Feng Kubricht, James, Colin Summers, Yixin Zhu, Brandon Rothrock, Song-Chun Zhu, and Hongjing Lu. "Human Causal Transfer: Challenges for Deep Reinforcement Learning." In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci)*, 2018.

[Elm93] Jeffrey L Elman. “Learning and development in neural networks: The importance of starting small.” *Cognition*, 1993.

[EMQ20] Mark Edmonds, Xiaojian Ma, Siyuan Qi, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. “Theory-based Causal Transfer: Integrating Instance-level Induction and Abstract-level Structure Learning.” In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[EMS18] Kevin M Ellis, Lucas E Morales, Mathias Sablé-Meyer, Armando Solar Lezama, and Joshua B Tenenbaum. “Library learning for neurally-guided bayesian program induction.” In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[EQZ19] Mark Edmonds, Siyuan Qi, Yixin Zhu, James Kubricht, Song-Chun Zhu, and Hongjing Lu. “Decomposing Human Causal Learning: Bottom-up Associative Learning and Top-down Schema Reasoning.” In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci)*, 2019.

[ER13] Susan E Embretson and Steven P Reise. *Item response theory.* Psychology Press, 2013.

[ERS18] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Joshua B Tenenbaum. “Learning to infer graphics programs from hand-drawn images.” In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[EWN20] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. “Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning.” *arXiv preprint arXiv:2006.08381*, 2020.

[Fan18] Yang Fan et al. “Learning to teach.” *ICLR*, 2018.

[FAS10] Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. “A probabilistic computational model of cross-situational word learning.” *Annual Meeting of the Cognitive Science Society (CogSci)*, 2010.

[FD18] Abram L Friesen and Pedro M Domingos. “Submodular field grammars: representation, inference, and application to image parsing.” In *Advances in Neural Information Processing Systems*, 2018.

[FFG89] Brian Falkenhainer, Kenneth D Forbus, and Dedre Gentner. “The structure-mapping engine: Algorithm and examples.” *Artificial intelligence*, **41**(1):1–63, 1989.

[FHC18] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. "Speaker-follower models for vision-and-language navigation." In *Advances in Neural Information Processing Systems*, pp. 3314–3325, 2018.

[Fit12] Melvin Fitting. *First-order logic and automated theorem proving*. Springer Science & Business Media, 2012.

[FL02] Jerry A Fodor and Ernest Lepore. *The compositionality papers*. Oxford University Press, 2002.

[Fle85] Charles R. Fletcher. "Understanding and solving arithmetic word problems: A computer simulation." *Behavior Research Methods, Instruments, & Computers*, **17**:565–571, 1985.

[Fod75] Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

[FP88] Jerry A Fodor, Zenon W Pylyshyn, et al. "Connectionism and cognitive architecture: A critical analysis." *Cognition*, **28**(1-2):3–71, 1988.

[FS16] Chaz Firestone and Brian J Scholl. "Cognition does not affect perception: Evaluating the evidence for "top-down" effects." *Behavioral and Brain Sciences*, **39**, 2016.

[GBD15] Artur d'Avila Garcez, Tarek R Besold, Luc De Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. "Neural-symbolic learning and reasoning: contributions and challenges." In *2015 AAAI Spring Symposium Series*, 2015.

[GBM17] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. "Automated Curriculum Learning for Neural Networks." In *International Conference on Machine Learning (ICML)*, 2017.

[Gha15] Zoubin Ghahramani. "Probabilistic machine learning and artificial intelligence." *Nature*, **521**(7553):452–459, 2015.

[GHZ18] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R. Scott, and Dinglong Huang. "CurriculumNet: Weakly Supervised Learning from Large-Scale Web Images." *ArXiv*, **abs/1808.01097**, 2018.

[GKS17] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. "Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6904–6913, 2017.

[GLB19] Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. "Permutation equivariant models for compositional generalization in language." In *International Conference on Learning Representations (ICLR)*, 2019.

[GLG08] Artur SD'Avila Garcez, Luis C Lamb, and Dov M Gabbay. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.

[GLL17] Chuang Gan, Yandong Li, Haoxiang Li, Chen Sun, and Boqing Gong. "Vqs: Linking segmentations to questions and answers for supervised attention in vqa and question-focused semantic segmentation." In *ICCV*, pp. 1811–1820, 2017.

[GLT18] Jon Gauthier, Roger Levy, and Joshua B Tenenbaum. "Word learning and the acquisition of syntactic-semantic overhypotheses." *Annual Meeting of the Cognitive Science Society (CogSci)*, 2018.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE, 2013.

[GPL17a] Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. "From language to programs: Bridging reinforcement learning and maximum marginal likelihood." *arXiv preprint arXiv:1704.07926*, 2017.

[GPL17b] Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. "From Language to Programs: Bridging Reinforcement Learning and Maximum Marginal Likelihood." In *ACL*, 2017.

[Gre93] Ulf Grenander. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993.

[GSS09] Abhinav Gupta, Praveen Srinivasan, Jianbo Shi, and Larry S Davis. "Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos." In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2012–2019. IEEE, 2009.

[GXG18] Hong Ge, Kai Xu, and Zoubin Ghahramani. "Turing: A language for flexible probabilistic inference." In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[HAR17] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. "Learning to Reason: End-to-End Module Networks for Visual Question Answering." *International Conference on Computer Vision (ICCV)*, pp. 804–813, 2017.

[HBM20] Steven Holtzen, Guy Van den Broeck, and Todd Millstein. "Scaling exact inference for discrete probabilistic programs." *Proceedings of the ACM on Programming Languages*, **4**(OOPSLA):1–31, 2020.

[HDY12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. "Deep neural networks for acoustic modeling in speech recognition." *Signal Processing Magazine*, **29**(6):82–97, 2012.

[HGD17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[Hin84] Geoffrey E Hinton. "Distributed representations." 1984.

[HLC21] Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. "Learning by fixing: Solving math word problems with weak supervision." In *AAAI Conference on Artificial Intelligence*, 2021.

[HLG21] Yining Hong, Qing Li, Ran Gong, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. "Smart: A situation model for algebra story problems via attributed grammar." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 13009–13017, 2021.

[HLL18] Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. "Neural Math Word Problem Solver with Reinforcement Learning." In *COLING*, 2018.

[HLZ21] Yining Hong, Qing Li, Song-Chun Zhu, and Siyuan Huang. "Vlgrammar: Grounded grammar induction of vision and language." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1665–1674, 2021.

[HM18] Drew A Hudson and Christopher D Manning. "Compositional Attention Networks for Machine Reasoning." In *International Conference on Learning Representations (ICLR)*, 2018.

[HM19] Drew A Hudson and Christopher D Manning. "GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering." *CVPR*, 2019.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation*, **18**(7):1527–1554, 2006.

[HQX18] Siyuan Huang, Siyuan Qi, Yinxue Xiao, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. "Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation." In *Advances in Neural Information Processing Systems*, 2018.

[HQZ18] Siyuan Huang, Siyuan Qi, Yixin Zhu, Yinxue Xiao, Yuanlu Xu, and Song-Chun Zhu. "Holistic 3D scene parsing and reconstruction from a single RGB image." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 187–203, 2018.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." *Neural computation*, **9**(8):1735–1780, 1997.

[HSL16] Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. "How well do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation." In *ACL*, 2016.

[HSL17] Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. "Learning Fine-Grained Expressions to Solve Math Word Problems." In *EMNLP*, 2017.

[HWJ19] Yining Hong, Jialu Wang, Yuting Jia, Weinan Zhang, and Xinbing Wang. "Academic Reader: An Interactive Question Answering System on Academic Literatures." *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[HZ05] Feng Han and Song-Chun Zhu. "Bottom-up/top-down image parsing by attribute graph grammar." In *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2005.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[Jac86] Peter Jackson. "Introduction to expert systems." 1986.

[JCH20] baoxiong Jia, Yixin Chen, Siyuan Huang, Yixin Zhu, and Song-Chun Zhu. "LEMMA: A Multi-view Dataset for LEarning Multi-agent Multi-task Activities." In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[JHM17a] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[JHM17b] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Fei Fei Li, C. Zitnick, and Ross Girshick. "Inferring and Executing Programs for Visual Reasoning." In *International Conference on Computer Vision (ICCV)*, 2017.

[JHV17] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. "Inferring and executing programs for visual reasoning." In *ICCV*, 2017.

[Jia14] Lu Jiang et al. "Self-paced learning with diversity." In *NIPS*, 2014.

[Jia15] Lu Jiang et al. "Self-paced curriculum learning." In *AAAI*, 2015.

[JQZ18] Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. "Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars." *International Journal of Computer Vision*, **126**(9):920–941, 2018.

[KB15a] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In *International Conference on Learning Representations (ICLR)*, 2015.

[KB15b] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In *International Conference on Learning Representations (ICLR)*, 2015.

[KD09] Kai A Krueger and Peter Dayan. "Flexible shaping: How learning in small steps helps." *Cognition*, 2009.

[KHS15] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. "Parsing Algebraic Word Problems into Equations." *Transactions of the Association for Computational Linguistics*, **3**:585–597, 2015.

[KK18] Nikita Kitaev and Dan Klein. "Constituency parsing with a self-attentive encoder." In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[KKT15] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. "Picture: A probabilistic programming language for scene perception." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[KMP18] Ashwin Kalyan, Abhishek Mohta, Oleksandr Polozov, Dhruv Batra, Prateek Jain, and Sumit Gulwani. "Neural-guided deductive search for real-time program synthesis from examples." *arXiv preprint arXiv:1804.01186*, 2018.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[KSS19] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. "Measuring compositional generalization: A comprehensive method on realistic data." In *International Conference on Learning Representations (ICLR)*, 2019.

[Kum10] M Pawan Kumar et al. "Self-paced learning for latent variable models." In *NIPS*, 2010.

[KZB14] Nate Kushman, Luke Zettlemoyer, Regina Barzilay, and Yoav Artzi. "Learning to Automatically Solve Algebra Word Problems." In *ACL*, 2014.

[LB95] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series." *The handbook of brain theory and neural networks*, **3361**(10):1995, 1995.

[LB18] Brenden M. Lake and Marco Baroni. "Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks." In *ICML*, 2018.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature*, **521**(7553):436–444, 2015.

[LBL16a] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision." *arXiv preprint arXiv:1611.00020*, 2016.

[LBL16b] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. "Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision." In *ACL*, 2016.

[LBL16c] Chen Liang, Jonathan Berant, Quoc V. Le, Kenneth D. Forbus, and Ni Lao. "Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision." In *ACL*, 2016.

[LC19] Guillaume Lample and François Charton. "Deep learning for symbolic mathematics." *arXiv preprint arXiv:1912.01412*, 2019.

[LC20] Guillaume Lample and François Charton. "Deep learning for symbolic mathematics." In *International Conference on Learning Representations (ICLR)*, 2020.

[LDH17] Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B Smith, James M Rehg, and Le Song. "Iterative machine teaching." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2149–2158. JMLR. org, 2017.

[LeC15] Yann LeCun et al. "LeNet-5, convolutional neural networks." *URL: http://yann. lecun. com/exdb/lenet*, **20**:5, 2015.

[LFY18] Qing Li, Jianlong Fu, Dongfei Yu, Tao Mei, and Jiebo Luo. "Tell-and-answer: Towards explainable visual question answering using attributes and captions." *EMNLP*, 2018.

[LHH20a] Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. "Closed Loop Neural-Symbolic Learning via Integrating Neural Perception, Grammar Parsing, and Symbolic Reasoning." In *Proceedings of International Conference on Machine Learning (ICML)*, 2020.

[LHH20b] Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun. Zhu. "Closed Loop Neural-Symbolic Learning via Integrating Neural Perception, Grammar Parsing, and Symbolic Reasoning." In *International Conference on Machine Learning (ICML)*, 2020.

[LHH20c] Qing Li, Siyuan Huang, Yining Hong, and Song-Chun Zhu. "A Competence-aware Curriculum for Visual Concepts Learning via Question Answering." *ECCV*, 2020.

[LHH21] Qing Li, Siyuan Huang, Yining Hong, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. "A hint from arithmetic: On systematic generalization of perception, syntax, and semantics." *arXiv preprint arXiv:2103.01403*, 2021.

[Llo12] John W Lloyd. *Foundations of logic programming.* Springer Science & Business Media, 2012.

[LNB18a] Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. "Memory Augmented Policy Optimization for Program Synthesis and Semantic Parsing." In *NIPS*, July 2018.

[LNB18b] Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V. Le, and Ni Lao. "Memory Augmented Policy Optimization for Program Synthesis and Semantic Parsing." In *NeurIPS*, 2018.

[LSR19] Dennis Lee, Christian Szegedy, Markus N Rabe, Sarah M Loos, and Kshitij Bansal. "Mathematical Reasoning in Latent Space." *arXiv preprint arXiv:1909.11851*, 2019.

[LSR20] Dennis Lee, Christian Szegedy, Markus N Rabe, Sarah M Loos, and Kshitij Bansal. "Mathematical Reasoning in Latent Space." In *International Conference on Learning Representations (ICLR)*, 2020.

[LST15] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. "Human-level concept learning through probabilistic program induction." *Science*, **350**(6266):1332–1338, 2015.

[LTJ18] Qing Li, Qingyi Tao, Shafiq Joty, Jianfei Cai, and Jiebo Luo. "VQA-E: Explaining, elaborating, and enhancing your answers for visual questions." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 552–567, 2018.

[LUT17] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. "Building machines that learn and think like people." *Behavioral and Brain Sciences*, **40**, 2017.

[LWY16] John P. Lalor, Hao Wu, and Hong Yu. "Building an Evaluation Scale using Item Response Theory." *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[LWY19] John P Lalor, Hao Wu, and Hong Yu. "Learning Latent Parameters without Human Response Patterns: Item Response Theory with Artificial Crowds." *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[LYD17] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. "Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems." *ArXiv*, **abs/1705.04146**, 2017.

[LZS18] Hangxin Liu, Yaofang Zhang, Wenwen Si, Xu Xie, Yixin Zhu, and Song-Chun Zhu. "Interactive Robot Knowledge Patching using Augmented Reality." In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2018.

[LZZ19] Hangxin Liu, Chi Zhang, Yixin Zhu, Chenfanfu Jiang, and Song-Chun Zhu. "Mirroring without Overimitation: Learning Functionally Equivalent Manipulation Actions." In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[Mag09] Lorenzo Magnani. *Abductive cognition: The epistemological and eco-cognitive dimensions of hypothetical reasoning*, volume 3. Springer Science & Business Media, 2009.

[Mar98] Gary F Marcus. "Rethinking eliminative connectionism." *Cognitive psychology*, **37**(3):243–282, 1998.

[Mar18] Gary F Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press, 2018.

[MB16] Arindam Mitra and Chitta Baral. "Learning To Use Formulas To Solve Simple Arithmetic Problems." In *ACL*, 2016.

[MCV19] Farnam Mansouri, Yuxin Chen, Ara Vartanian, Xiaojin Zhu, and Adish Singla. "Preference-Based Batch and Sequential Teaching: Towards a Unified View of Models." In *NeurIPS*, 2019.

[MD94] Stephen Muggleton and Luc De Raedt. "Inductive logic programming: Theory and methods." *The Journal of Logic Programming*, **19**:629–679, 1994.

[MDK18] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. "Deepproblog: Neural probabilistic logic programming." In *Advances in Neural Information Processing Systems*, pp. 3749–3759, 2018.

[MF14] Mateusz Malinowski and Mario Fritz. "A Multi-world Approach to Question Answering About Real-world Scenes Based on Uncertain Input." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[MGF17] Ishan Misra, Ross B. Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. "Learning by Asking Questions." *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[MGK19] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision." *International Conference on Learning Representations (ICLR)*, 2019.

[MS17] Hugo Mercier and Dan Sperber. *The enigma of reason*. Harvard University Press, 2017.

[MTS18] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. "Transparency by design: Closing the gap between performance and interpretability in visual reasoning." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4942–4950, 2018.

[NNM16] Prathiba Natesan, Ratna Nandakumar, Tom Minka, and Jonathan D Rubright. "Bayesian prior choice in IRT estimation using MCMC and variational Bayes." *Frontiers in psychology*, 2016.

[NS07] Allen Newell and Herbert A Simon. "Computer science as empirical inquiry: Symbols and search." In *ACM Turing award lectures*, p. 1975. 2007.

[PCZ19] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. "Specaugment: A simple data augmentation method for automatic speech recognition." In *Interspeech*, 2019.

[Pea89] Giuseppe Peano. *Arithmetices principia: Nova methodo exposita*. Fratres Bocca, 1889.

[Pet04] Gail B Peterson. "A day of great illumination: BF Skinner's discovery of shaping." *Journal of the experimental analysis of behavior*, 2004.

[PMS16] Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. "Neuro-symbolic program synthesis." In *International Conference on Learning Representations (ICLR)*, 2016.

[PSL14] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. "Curriculum learning of multiple tasks." *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5492–5500, 2014.

[PSN19] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabás Póczos, and Tom M. Mitchell. "Competence-based Curriculum Learning for Neural Machine Translation." In *North American Chapter of the Association for Computational Linguistics(NAACL-HLT)*, 2019.

[PSV17] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. "FiLM: Visual Reasoning with a General Conditioning Layer." In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[Pyl84] Zenon W Pylyshyn. "Computation and cognition: Towards a foundation for cognitive science.", 1984.

[QJH20] Siyuan Qi, Baoxiong Jia, Siyuan Huang, Ping Wei, and Song-Chun Zhu. "A Generalized Earley Parser for Human Activity Parsing and Prediction." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[QJZ18] Siyuan Qi, Baoxiong Jia, and Song-Chun Zhu. "Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction." *ICML*, 2018.

[QWL15a] Hang Qi, Tianfu Wu, M. Lee, and S. Zhu. "A Restricted Visual Turing Test for Deep Scene and Event Understanding." *ArXiv*, **abs/1512.01715**, 2015.

[QWL15b] Hang Qi, Tianfu Wu, Mun-Wai Lee, and Song-Chun Zhu. "A restricted visual turing test for deep scene and event understanding." *arXiv preprint arXiv:1512.01715*, 2015.

[QZH18] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. "Human-centric indoor scene synthesis using stochastic grammar." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5899–5908, 2018.

[Rec85] Mark D Reckase. "The difficulty of test items that measure more than one ability." *Applied psychological measurement*, 1985.

[Rec09] Mark D Reckase. "Multidimensional item response theory models." In *Multidimensional item response theory*. 2009.

[RKH18] Benjamin Robaidek, Rik Koncel-Kedziorski, and Hannaneh Hajishirzi. "Data-Driven Methods for Solving Algebra Word Problems." *ArXiv*, **abs/1804.10718**, 2018.

[RR17] Subhro Roy and Dan Roth. "Unit Dependency Graph and Its Application to Arithmetic Word Problem Solving." In *AAAI*, 2017.

[RZL16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "Squad: 100,000+ questions for machine comprehension of text." In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[Sac16] Mrinmaya Sachan et al. "Easy questions first? a case study on curriculum learning for question answering." In *ACL*, 2016.

[SAJ10] Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. "From baby steps to leapfrog: How less is more in unsupervised dependency parsing." In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 751–759. Association for Computational Linguistics, 2010.

[Ski58] Burrhus F Skinner. "Reinforcement today." *American Psychologist*, 1958.

[SLJ15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[Sun94] Ron Sun. *Integrating rules and connectionism for robust commonsense reasoning.* John Wiley & Sons, Inc., 1994.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks." In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[SWL15] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. "Automatically Solving Number Word Problems by Semantic Parsing and Reasoning." In *EMNLP*, 2015.

[TCY05] Zhuowen Tu, Xiangrong Chen, Alan L Yuille, and Song-Chun Zhu. "Image parsing: Unifying segmentation, detection, and recognition." *International Journal of computer vision*, 2005.

[TFL16] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. “Learning the curriculum with bayesian optimization for task-specific word representation learning.” *ACL*, 2016.

[TML14a] K. Tu, M. Meng, M. Lee, T. E. Choe, and S. Zhu. “Joint Video and Text Parsing for Understanding Events and Answering Queries.” *IEEE MultiMedia*, **21**:42–70, 2014.

[TML14b] Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song-Chun Zhu. “Joint video and text parsing for understanding events and answering queries.” *IEEE MultiMedia*, 2014.

[UCC16] Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen tau Yih. “Learning from Explicit and Implicit Supervision Jointly For Algebra Word Problems.” In *EMNLP*, 2016.

[VDL19] Ramakrishna Vedantam, Karan Desai, Stefan Lee, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. “Probabilistic neural-symbolic models for interpretable visual question answering.” *arXiv preprint arXiv:1902.07864*, 2019.

[VFJ15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. “Pointer Networks.” In *NIPS*, 2015.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[VVG20] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. “Scan: Learning to classify images without labels.” In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[Wil92] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning.” *Machine learning*, 1992.

[WLS17] Yan Wang, Xiaojiang Liu, and Shuming Shi. “Deep Neural Solver for Math Word Problems.” pp. 845–854, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[WSC16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean.

"Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." *ArXiv*, **abs/1609.08144**, 2016.

[WT11] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics." In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

[Wu18] Lijun Wu et al. "Learning to teach with dynamic loss functions." In *NeurIPS*, 2018.

[WWC18] Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. "Translating Math Word Problem to Expression Tree." In *EMNLP*, 2018.

[WZG18] Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. "Mathdqn: Solving arithmetic word problems via deep reinforcement learning." In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[WZZ19] Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. "Template-Based Math Word Problem Solvers with Recursive Neural Networks." *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**(01):7144–7151, Jul 2019.

[XLE18] Xu Xie, Hangxin Liu, Mark Edmonds, Feng Gao, Siyuan Qi, Yixin Zhu, Brandon Rothrock, and Song-Chun Zhu. "Unsupervised Learning of Hierarchical Models for Hand-Object Interactions." In *ICRA*, 2018.

[XLZ20a] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "How neural networks extrapolate: From feedforward to graph neural networks." *arXiv preprint arXiv:2009.11848*, 2020.

[XLZ20b] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "What can neural networks reason about?" In *International Conference on Learning Representations (ICLR)*, 2020.

[XMY21] Sirui Xie, Xiaojian Ma, Peiyu Yu, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. "HALMA: Humanlike Abstraction Learning Meets Affordance in Rapid Problem Solving." *arXiv preprint arXiv:2102.11344*, 2021.

[XS19] Zhipeng Xie and Shichao Sun. "A Goal-Driven Tree-Structured Neural Model for Math Word Problems." In *IJCAI*, 2019.

[YCX09] Qiang Yang, Yuqiang Chen, Gui-Rong Xue, Wenyuan Dai, and Yong Yu. "Heterogeneous transfer learning for image clustering via the social web." In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th international joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pp. 1–9. Association for Computational Linguistics, 2009.

[YGL20] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. "Clevrer: Collision events for video representation and reasoning." *ICLR*, 2020.

[YJD20] Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. "ReClor: A Reading Comprehension Dataset Requiring Logical Reasoning." *ArXiv*, **abs/2002.04326**, 2020.

[YLF20] Tao Yuan, Hangxin Liu, Lifeng Fan, Zilong Zheng, Tao Gao, Yixin Zhu, and Song-Chun Zhu. "Joint Inference of States, Robot Knowledge, and Human (False-)Beliefs." In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2020.

[YWG18] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding." In *Advances in Neural Information Processing Systems*, 2018.

[YYG10] Ma Yu-hui, Zhou Ying, Cui Guang-zuo, Ren Yun, and Huang Rong-huai. "Frame-Based Calculus of Solving Arithmetic Multi-Step Addition and Subtraction Word Problems." *2010 Second International Workshop on Education Technology and Computer Science*, **2**:476–479, 2010.

[YZH18] Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. "StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing." *arXiv preprint arXiv:1806.07832*, 2018.

[ZCN17] Quanshi Zhang, Ruiming Cao, Ying Nian Wu, and Song-Chun Zhu. "Mining object parts from cnns via active question-answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 346–355, 2017.

[ZDC15] Lipu Zhou, Shuaixiang Dai, and Liwei Chen. "Learn to Solve Algebra Word Problems Using Quadratic Programming." In *EMNLP*, 2015.

[ZGF20] Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Nian Ying Wu, B. Joshua Tenenbaum, and Song-Chun Zhu. "Dark, Beyond Deep: A Paradigm Shift to Cognitive AI with Humanlike Common Sense." *Engineering*, 2020.

[ZGJ19a] Chi Zhang, F. Gao, Baoxiong Jia, Yixin Zhu, and S. Zhu. "RAVEN: A Dataset for Relational and Analogical Visual REasoNing." *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5312–5322, 2019.

[ZGJ19b] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. "RAVEN: A Dataset for Relational and Analogical Visual rEasoNing." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[Zho19a] Zhi-Hua Zhou. "Abductive learning: towards bridging machine learning and logical reasoning." *Science China Information Sciences*, **62**:1–3, 2019.

[Zho19b] Zhi-Hua Zhou. "Abductive learning: towards bridging machine learning and logical reasoning." *Science China Information Sciences*, **62**:1–3, 2019.

[Zhu15] Xiaojin Zhu. "Machine teaching: An inverse problem to machine learning and an approach toward optimal education." In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[ZJG19] Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. "Learning Perceptual Inference by Contrasting." In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[ZM07] Song-Chun Zhu, David Mumford, et al. "A stochastic grammar of images." *Foundations and Trends® in Computer Graphics and Vision*, **2**(4):259–362, 2007.

[ZRH20] Quanshi Zhang, Jie Ren, Ge Huang, Ruiming Cao, Ying Nian Wu, and Song-Chun Zhu. "Mining Interpretable AOG Representations from Convolutional Networks via Active Question Answering." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[ZSZ18] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. "An overview of machine teaching." *arXiv preprint arXiv:1801.05927*, 2018.

[ZWL20] Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Jie Shao, and Ee-Peng Lim. "Graph-to-Tree Learning for Solving Math Word Problems." *ACL 2020*, 2020.

[ZZ11] Yibiao Zhao and Song-Chun Zhu. "Image parsing with stochastic scene grammar." In *Advances in Neural Information Processing Systems*, pp. 73–81, 2011.

[ZZZ20a] Wenhe Zhang, Chi Zhang, Yixin Zhu, and S. Zhu. "Machine Number Sense: A Dataset of Visual Arithmetic Problems for Abstract and Relational Reasoning." *ArXiv*, **abs/2004.12193**, 2020.

[ZZZ20b] Wenhe Zhang, Chi Zhang, Yixin Zhu, and Song-Chun Zhu. "Machine Number Sense: A Dataset of Visual Arithmetic Problems for Abstract and Relational Reasoning." In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[ZZZ20c] Zhenliang Zhang, Yixin Zhu, and Song-Chun Zhu. "Graph-based Hierarchical Knowledge Representation for Robot Task Transfer from Virtual to Physical World." In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2020.