Closed Loop Neural-Symbolic Learning via Integrating Neural Perception, Grammar Parsing, and Symbolic Reasoning

Anonymous Authors[1]

Abstract

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 paper, 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. The experiments are conducted on two weakly-supervised neural-symbolic tasks: (1) handwritten formula recognition on the CROHME 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.

1. Introduction

Integrating robust connectionist learning and sound symbolic reasoning is a key challenge in modern Artificial Intelligence. Recently, the neural-symbolic paradigm has been extensively explored in many tasks, such as visual question answering (Yi et al., 2018; Vedantam et al., 2019; Mao et al., 2019) and semantic parsing (Liang et al., 2016; Yin et al., 2018), often with weak supervision. 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. These methods have been proved to be time-consuming because they require generating a large number of samples over the latent space of symbolic representations with sparse rewards.

To model the recursive compositionality in a sequence of

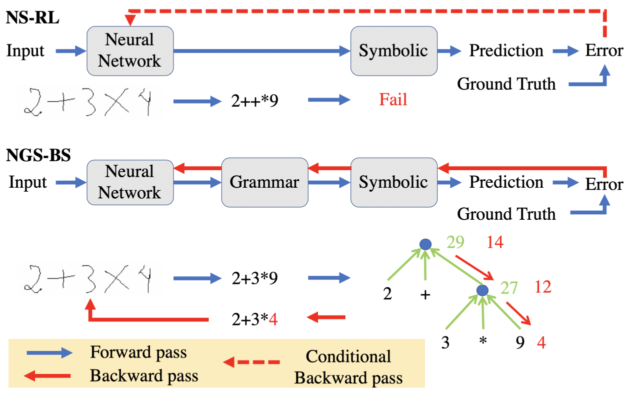

Figure 1. Comparison between the original neural-symbolic model learned by REINFORCE (NS-RL) and the proposed neural-grammar-symbolic model learned by back-search (NGS-BS). In NS-RL, the neural network predicts an invalid formula, causing a failure in the symbolic reasoning module. There is no backward 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.

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 1.

To make the neural-symbolic learning more efficient, we propose a novel **back-search** strategy which mimics human's ability to diagnose and correct the samples that cannot generate desired outputs. 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 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.

We conduct experiments on two weakly-supervised tasks: (1) handwritten formula recognition on the CHROME dataset; (2) visual question answering on the CLEVR dataset. 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. We present in this paper a summary of our full paper (Li et al., 2020), which is also included in the supplementary material for more details.

2. Neural-Grammar-Symbolic Model (NGS)

2.1. Inference

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:

$$p_\theta(z|x) = softmax(\phi_\theta(z,x)) \tag{1}$$

$$= \frac{\exp(\phi_\theta(z,x))}{\sum_{z'}\exp(\phi_\theta(z',x))}, \tag{2}$$

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. 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 $\alpha \in \Sigma$.
- $S \in V$ 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. In neural-symbolic tasks, the objective of parsing is to find the most probable z that can be accepted by the grammar:

$$\hat{z} = \arg\max_{z \in L(G)} p_\theta(z|x) \tag{3}$$

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

Traditional grammar parsers can only work on discrete symbols. Qi et al. (2018) 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 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:

$$\hat{y} : \hat{z} \wedge \Delta \models \hat{y} \tag{4}$$

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

$$\hat{y} = f(\hat{z}; \Delta), \tag{5}$$

where f denotes inference rules under the domain Δ.

2.2. Learning

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.

2.2.1. 1-STEP BACK-SEARCH (1-BS)

As shown in Figure 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 is defined as a 3-tuple (A, α_A, p), where $A \in V \cup \Sigma$ is the current visiting node. α_A is the expected value on this node thus $y = f(\hat{z}(A \to \alpha_A); \Delta))$, where $\hat{z}(A \to \alpha_A)$ denotes that the sub-tree of A is replaced by α_A. p is the visiting priority, which reflects the potential of changing the current node or its child nodes to correct the wrong answer.

The $solve(\cdot|\Delta, G)$ function is an error propagation function which aims at computing an expected value α_B from its parent's expected value α_A given the rules. Therefore $f(\hat{z}(B \to \alpha_B); \Delta)) = f(\hat{z}(A \to \alpha_A); \Delta)) = y$. Accordingly, the priority for this change is defined as the probability ratio:

$$p(B \to \alpha_B) = \begin{cases} \frac{1-p(B)}{p(B)}, & \text{if } B \notin \Sigma \\ \frac{p(\alpha_B)}{p(B)}, & \text{if } B \in \Sigma \;\&\; \alpha_B \in \Sigma. \end{cases} \tag{6}$$

If $B \in \Sigma$ and $\alpha_B \notin \Sigma$, 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. 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.

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

```
1:  Input: $\hat{z}, S, y$
2:  $q = PriorityQueue()$
3:  $q.push(S, y, 1)$
4:  while $A, \alpha_A, p = q.pop()$ do
5:      if $A \in \Sigma$ then
6:          $z^* = \hat{z}(A \to \alpha_A)$
7:          return $z^*$
8:      for $B \in child(A)$ do
9:          $\alpha_B = solve(B, A, \alpha_A|\Delta, G)$
10:         $q.push(B, \alpha_B, p(B \to \alpha_B))$
11: return $\varnothing$
```

2.2.2. Multi-step Back-Search (m-BS)

We extend the 1-step back-search to a multi-step back-search (m-BS) by incorporating a RandomWalk function, 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 traverse the whole solution space with non-zero probabilities.

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

```
1:  Hyperparameters: $T$, $\lambda$
2:  Input: $\hat{z}, y$
3:  $z^{(0)} = \hat{z}$
4:  for $t \leftarrow 0$ to $T-1$ do
5:      $z^* = $ 1-BS$(z^t, y)$
6:      draw $u \sim \mathcal{U}(0,1)$
7:      if $u \le \lambda$ and $z^* \neq \varnothing$ then
8:          $z^{t+1} = z^*$
9:      else
10:         $z^{t+1} = $ RandomWalk$(z^t)$
11: return $z^T$
12:
13: function RandomWalk$(z^t)$
14:     sample $z^* \sim g(\cdot|z^t)$
15:     compute acceptance ratio $a = min(1, \frac{p_\theta(z^*|x)}{p_\theta(z^t|x)})$
16:     draw $u \sim \mathcal{U}(0,1)$
17:     $z^{t+1} = \begin{cases} z^*, & \text{if } u \le a \\ z^t, & \text{otherwise.} \end{cases}$
```

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

$$g(z_1|z_2) = Poisson(d(z_1, z_2); \beta), \tag{7}$$

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

$$r(z^t, z^*) = \frac{p_\theta(z^*|x)}{p_\theta(z^t|x)}. \tag{8}$$

Notably, the m-BS algorithm serves as a Metropolis-Hastings sampler which samples from the posterior distribution of symbolic representations given the final results.

3. Experiments and Results

3.1. Handwritten Formula Recognition

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, and the computed results of the formulas is treated as outputs. 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. We report the calculation accuracy (*i.e.*whether the calculation of predicted formula yields to the correct result) on the CROHME dataset.

Learning Curve. Figure 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.

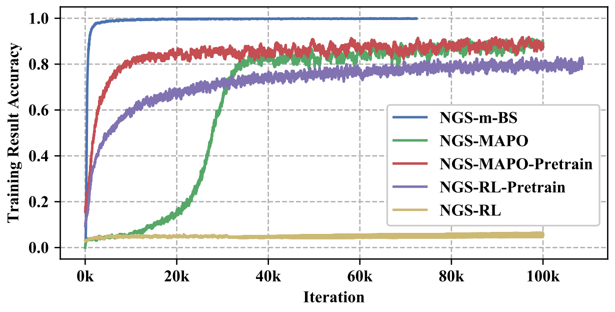

Figure 2. The learning curves of calculation accuracy.

Data Efficiency. Table 1 shows 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 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**

3.2. Neural-Symbolic Visual Question Answering

Following (Yi et al., 2018), 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. We report the answer accuracy on the CLEVR dataset (Johnson et al., 2017).

Learning Curve. Figure 3 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.

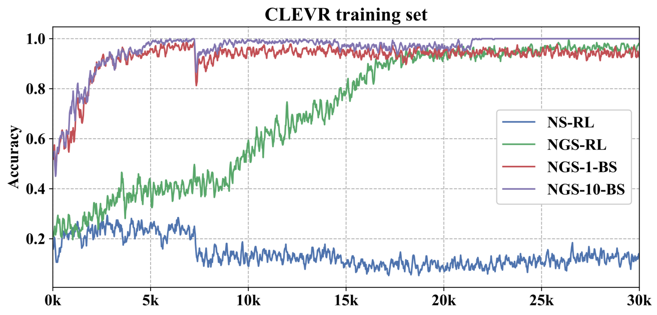

Figure 3. The learning curves of answer accuracy.

Data Efficiency Table 2 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. The 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**

4. 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.

References

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2901–2910, 2017.

Li, Q., Huang, S., Hong, Y., Chen, Y., Wu, Y. N., and Zhu, S.-C. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *International Conference on Machine Learning (ICML)*, 2020.

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

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

Qi, S., Jia, B., and Zhu, S.-C. Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction. *ICML*, 2018.

Vedantam, R., Desai, K., Lee, S., Rohrbach, M., Batra, D., and Parikh, D. Probabilistic neural-symbolic models for interpretable visual question answering. *arXiv preprint arXiv:1902.07864*, 2019.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018.

Yin, P., Zhou, C., He, J., and Neubig, G. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. *arXiv preprint arXiv:1806.07832*, 2018.

Supplementary Materials (Full Paper) for Closed Loop Neural-Symbolic Learning via Integrating Neural Perception, Grammar Parsing, and Symbolic Reasoning

Anonymous Authors[1]

Abstract

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 paper, 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 CROHME 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.

1. Introduction

Integrating robust connectionist learning and sound symbolic reasoning is a key challenge in modern Artificial Intelligence. Deep neural networks (LeCun et al., 2015a; 1995; Hochreiter & Schmidhuber, 1997) provide us powerful and flexible representation learning that has achieved state-of-the-art performances across a variety of AI tasks such as image classification (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016), machine translation (Sutskever et al., 2014), and speech recognition (Graves et al., 2013). However, it turns out that many aspects of human cognition, such as systematic compositionality and generalization (Fodor et al., 1988; Marcus, 1998; Fodor & Lepore, 2002; Calvo & Symons, 2014; Marcus, 2018; Lake & Baroni, 2018), 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 (Sun, 1994; Garcez et al., 2008; Bader et al., 2009; Besold et al., 2017; Yi et al., 2018).

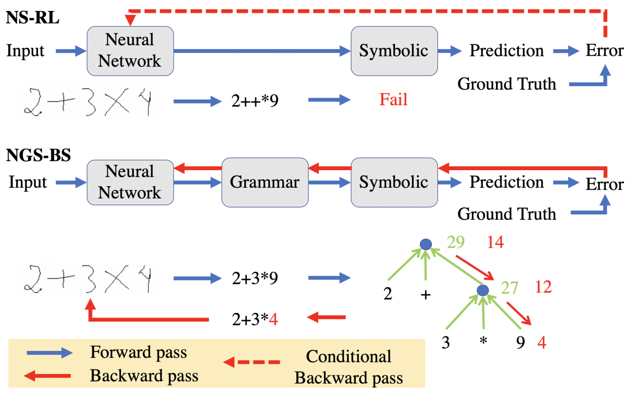

Figure 1. Comparison between the original neural-symbolic model learned by REINFORCE (NS-RL) and the proposed neural-grammar-symbolic model learned by back-search (NGS-BS). In NS-RL, the neural network predicts an invalid formula, causing a failure in the symbolic reasoning module. There is no backward 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.

Recently, this neural-symbolic paradigm has been extensively explored in the tasks of the visual question answering (VQA) (Yi et al., 2018; Vedantam et al., 2019; Mao et al., 2019), vision-language navigation (Anderson et al., 2018; Fried et al., 2018), embodied question answering (Das et al., 2018a;b), and semantic parsing (Liang et al., 2016; Yin et al., 2018), 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 1.

To make the neural-symbolic learning more efficient, we propose a novel **back-search** strategy which mimics human's ability to diagnose and correct the samples that cannot generate desired outputs. 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 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 3.2.3.

We conduct experiments on two weakly-supervised neural-symbolic tasks: (1) handwritten formula recognition on the CHROME dataset, 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. Related Work

Neural-symbolic Computation. 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 (Sun, 1994; Garcez et al., 2008), knowledge abstraction (Hinton et al., 2006; Bader et al., 2009), knowledge transfer (Falkenhainer et al., 1989; Yang et al., 2009), *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. For example neural-symbolic VQA (Yi et al., 2018; Vedantam et al., 2019; Mao et al., 2019), semantic parsing in Natural Language Processing (NLP) (Liang et al., 2016; Yin et al., 2018), math word problem (Lample & Charton, 2019; Lee et al., 2019) and program synthesis (Evans & Grefenstette, 2018; Kalyan et al., 2018; Manhaeve et al., 2018). 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 (Zhao & Zhu, 2011), video parsing (Gupta et al., 2009; Qi et al., 2018), scene understanding (Huang et al., 2018; Jiang et al., 2018), and task planning (Xie et al., 2018). 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 shrinks the search space greatly during the back-search algorithm, thus improve the learning efficiency significantly.

Policy Gradient. Policy gradient methods like REINFORCE (Williams, 1992) are the most commonly used algorithm for the neural-symbolic tasks to connect the learning gap between neural networks and symbolic reasoning (Mascharka et al., 2018; Mao et al., 2019; Andreas et al., 2017; Das et al., 2018b; Bunel et al., 2018; Guu et al., 2017). 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 (Liang et al., 2016; Guu et al., 2017; Liang et al., 2018; Wang et al., 2018; Agarwal et al., 2019). Specifically, Liang et al. (2016) uses iterative maximum likelihood to find pseudo-gold symbolic representations, and then add these representations to the REINFORCE training set. Guu et al. (2017) combines the systematic beam search employed in maximum marginal likelihood with the greedy randomized exploration of REINFORCE. Liang et al. (2018) 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.

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 4.

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:

$$p_\theta(z|x) = softmax(\phi_\theta(z,x)) \tag{1}$$

$$= \frac{\exp(\phi_\theta(z,x))}{\sum_{z'}\exp(\phi_\theta(z',x))}, \tag{2}$$

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:

$$\hat{z} = \arg\max_{z \in L(G)} p_\theta(z|x) \tag{3}$$

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 discrete symbols. Qi et al. (2018) 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 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:

$$\hat{y} : \hat{z} \wedge \Delta \models \hat{y} \tag{4}$$

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

$$\hat{y} = f(\hat{z}; \Delta), \tag{5}$$

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.

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.

3.2.1. 1-STEP BACK-SEARCH (1-BS)

As shown in Figure 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 is defined as a 3-tuple (A, α_A, p), where $A \in V \cup \Sigma$ is the current visiting node. α_A is the expected value on this node thus $y = f(\hat{z}(A \to \alpha_A); \Delta))$, where $\hat{z}(A \to \alpha_A)$ denotes that the sub-tree of A is replaced by α_A. p is the visiting priority, which reflects the potential of changing the current node or its child nodes to correct the wrong answer.

The $solve(\cdot|\Delta, G)$ function is an error propagation function which aims at computing an expected value α_B from its parent's expected value α_A given the rules. Therefore $f(\hat{z}(B \to \alpha_B); \Delta)) = f(\hat{z}(A \to \alpha_A); \Delta)) = y$. Accordingly, the priority for this change is defined as the probability ratio:

$$p(B \to \alpha_B) = \begin{cases} \frac{1-p(B)}{p(B)}, & \text{if } B \notin \Sigma \\ \frac{p(\alpha_B)}{p(B)}, & \text{if } B \in \Sigma \;\&\; \alpha_B \in \Sigma. \end{cases} \tag{6}$$

If $B \in \Sigma$ and $\alpha_B \notin \Sigma$, 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. 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 3.2.3, the 1-BS is extended to the multi-step back-search algorithm, which allows searching beyond one-step distance.

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

```
1:  Input: $\hat{z}, S, y$
2:  $q = PriorityQueue()$
3:  $q.push(S, y, 1)$
4:  while $A, \alpha_A, p = q.pop()$ do
5:      if $A \in \Sigma$ then
6:          $z^* = \hat{z}(A \to \alpha_A)$
7:          return $z^*$
8:      for $B \in child(A)$ do
9:          $\alpha_B = solve(B, A, \alpha_A|\Delta, G)$
10:         $q.push(B, \alpha_B, p(B \to \alpha_B))$
11: return $\varnothing$
```

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:

$$p(y|x) = \sum_z p(y, z|x) = \sum_z p(y|z) p_\theta(z|x). \tag{7}$$

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

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

$$\begin{aligned} \nabla_\theta L(x, y) &= \nabla_\theta \log p(y|x) \\ &= \frac{1}{p(y|x)} \nabla_\theta p(y|x) \\ &= \sum_z \frac{p(y|z) p_\theta(z|x)}{\sum_{z'} p(y|z') p_\theta(z'|x)} \nabla_\theta \log p_\theta(z|x) \\ &= \mathbb{E}_{z \sim p(z|x,y)} [\nabla_\theta \log p_\theta(z|x)], \end{aligned} \tag{8}$$

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:

$$p(z|x,y) = \frac{p(y|z)p_\theta(z|x)}{\sum_{z'} p(y|z')p_\theta(z'|x)} = \begin{cases} 0, & \text{for } z \notin Q \\ \frac{p_\theta(z|x)}{\sum_{z' \in Q} p_\theta(z'|x)}, & \text{for } z \in Q \end{cases} \tag{9}$$

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 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.

3.2.3. m-BS as Metropolis-Hastings Sampler

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

```
1: Hyperparameters: $T$, $\lambda$
2: Input: $\hat{z}, y$
3: $z^{(0)} = \hat{z}$
4: for $t \leftarrow 0$ to $T-1$ do
5:     $z^* = $ 1-BS$(z^t, y)$
6:     draw $u \sim \mathcal{U}(0,1)$
7:     if $u \le \lambda$ and $z^* \neq \varnothing$ then
8:         $z^{t+1} = z^*$
9:     else
10:        $z^{t+1} = $ RandomWalk$(z^t)$
11: return $z^T$
12:
13: function RandomWalk$(z^t)$
14:     sample $z^* \sim g(\cdot|z^t)$
15:     compute acceptance ratio $a = min(1, \frac{p_\theta(z^*|x)}{p_\theta(z^t|x)})$
16:     draw $u \sim \mathcal{U}(0,1)$
17:     $z^{t+1} = \begin{cases} z^*, & \text{if } u \le a \\ z^t, & \text{otherwise.} \end{cases}$
```

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

$$A(s,s') = min\left\{1, \frac{\pi(s')Q(s|s')}{\pi(s)Q(s'|s)}\right\}. \tag{10}$$

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 $p(y|z)$:

$$p'(z|x,y) = \frac{[p(y|z) + \epsilon]p_\theta(z|x)}{\sum_{z'}[p(y|z') + \epsilon]p_\theta(z'|x)} \tag{11}$$

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

$$g(z_1|z_2) = Poisson(d(z_1, z_2); \beta), \tag{12}$$

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

$$r(z^t, z^*) = \frac{q(z^*)(1-\lambda)g(z^t|z^*)}{q(z^t)(1-\lambda)g(z^*|z^t)} = \frac{p_\theta(z^*|x)}{p_\theta(z^t|x)}. \tag{13}$$

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

$$r(z^{(t)}, z^*) = \frac{q(z^*)[0 + (1-\lambda)g(z^t|z^*)]}{q(z^t) \cdot [\lambda + (1-\lambda)g(z^*|z^{(t)})]} = \frac{1+\epsilon}{\epsilon} \cdot \frac{p_\theta(z^*|x)}{p_\theta(z^t|x)} \cdot \frac{(1-\lambda)g(z^t|z^*)}{\lambda + (1-\lambda)g(z^*|z^t)}. \tag{14}$$

$q(z) = [p(y|z) + \epsilon]p_\theta(z|x)$ 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 $z^* = \text{1-BS}(z^t, y)$, making movements in directions of increasing the posterior probability. Similar to the gradient-based MCMCs like Langevin dynamics (Duane & Kogut, 1986; Welling & Teh, 2011), this is the main reason that the proposed method can sample the posterior efficiently.

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:

$$r(z,y) = \begin{cases} 0, \text{ if } f(z;\Delta) \neq y. \\ 1, \text{ if } f(z;\Delta) = y. \end{cases} \tag{15}$$

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

$$R(x,y) = \mathbb{E}_{z\sim p_\theta(z|x))}\, r(z,y) = \sum_z p_\theta(z|x) r(z,y). \tag{16}$$

Then the gradient for θ is:

$$\begin{aligned}\nabla_\theta R(x,y) &= \sum_z r(z,y) p_\theta(z|x) \nabla_\theta \log p_\theta(z|x) \\ &= \mathbb{E}_{z\sim p_\theta(z|x))}[r(z,y)\nabla_\theta \log p_\theta(z|x)]. \end{aligned} \tag{17}$$

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

$$\begin{aligned}\nabla_\theta &= r(z,y)\nabla_\theta \log p_\theta(z|x), z \sim p_\theta(z|x) \\ &= \begin{cases} 0, & \text{if } f(z;\Delta) \neq y. \\ \nabla_\theta \log p_\theta(z|x), & \text{if } f(z;\Delta) = y. \end{cases}\end{aligned} \tag{18}$$

Equation 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 4.

4. Experiments and Results

4.1. Handwritten Formula Recognition

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.

Synthetic Dataset. We generate our synthetic dataset based on 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. 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 (LeCun et al., 2015b) as the neural perception module and define a simple formula grammar that only considers arithmetic operations over single-digit numbers. The symbolic reasoning 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.

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) (Liang et al., 2018), 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.

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

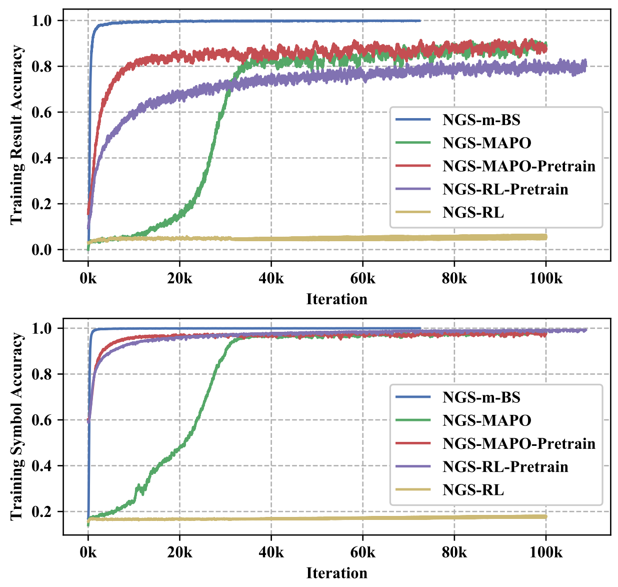

Figure 2. The learning curves of the calculation accuracy and the symbol recognition accuracy for different models.

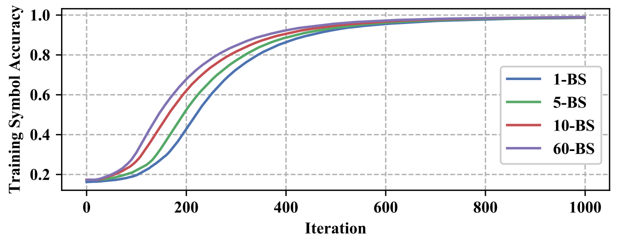

Figure 3. The training curves of NGS-m-BS with different steps.

4.1.2. Results and Analyses

Learning Curve. Figure 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 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.

Data Efficiency. Table 1 and Table 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 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**

Table 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**

Qualitative Results. Figure 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.

Please refer to the *supplementary material* for more implementation details of the models and more qualitative results.

4.2. Neural-Symbolic Visual Question Answering

4.2.1. Experimental Setup

Task. Following (Yi et al., 2018), 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 (Johnson et al., 2017a). The CLEVR dataset is a popular benchmark for testing compositional reasoning capability of VQA models in previous works (Johnson et al., 2017b; Vedantam et al., 2019). CLEVR consists of a training set of 70K images and $\sim$700K questions, and a validation set of 15K images and $\sim$150K questions. We use the

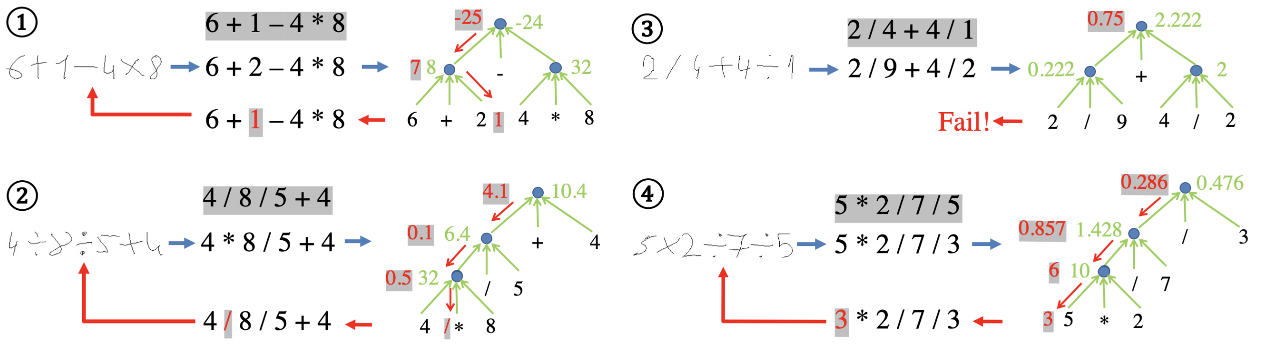

Figure 4. Examples of correcting wrong predictions using the one-step back-search algorithm.

Table 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**

VQA accuracy as the evaluation metric.

We adopt the NS-VQA model in (Yi et al., 2018) and replace the attention-based seq2seq question parser with a Pointer Network (Vinyals et al., 2015). 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**.

4.2.2. Results and Analyses

Learning Curve. Figure 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.

Back-Search Step. As shown in Figure 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.

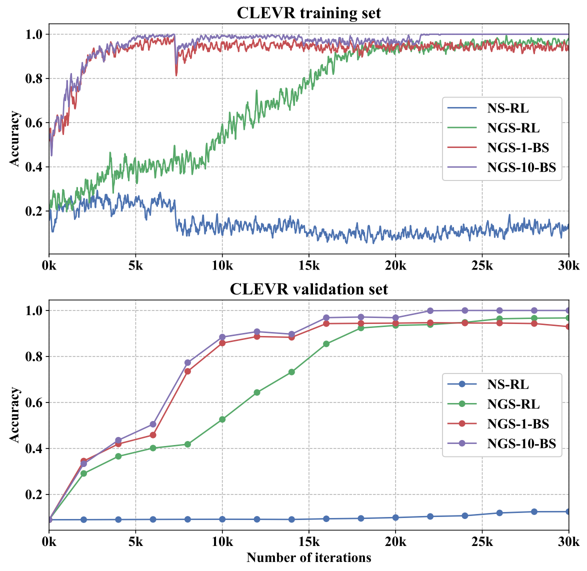

Figure 5. The learning curve of different model variants on training and validation set of the CLEVR dataset.

Data Efficiency Table 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.

Please refer to the *supplementary material* for more implementation details of the models and more qualitative results.

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.

References

Agarwal, R., Liang, C., Schuurmans, D., and Norouzi, M. Learning to generalize from sparse and underspecified rewards. *arXiv preprint arXiv:1902.07198*, 2019.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. 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.

Andreas, J., Klein, D., and Levine, S. 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.

Bader, S., Garcez, A. S. d., and Hitzler, P. 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.

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

Bunel, R., Hausknecht, M., Devlin, J., Singh, R., and Kohli, P. Leveraging grammar and reinforcement learning for neural program synthesis. *arXiv preprint arXiv:1805.04276*, 2018.

Calvo, P. and Symons, J. *The architecture of cognition: Rethinking Fodor and Pylyshyn's systematicity challenge*. MIT Press, 2014.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2054–2063, 2018a.

Das, A., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. Neural modular control for embodied question answering. *arXiv preprint arXiv:1810.11181*, 2018b.

Duane, S. and Kogut, J. B. The theory of hybrid stochastic algorithms. *Nuclear Physics B*, 275(3):398–420, 1986.

Evans, R. and Grefenstette, E. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Falkenhainer, B., Forbus, K. D., and Gentner, D. The structure-mapping engine: Algorithm and examples. *Artificial intelligence*, 41(1):1–63, 1989.

Fodor, J. A. and Lepore, E. *The compositionality papers*. Oxford University Press, 2002.

Fodor, J. A., Pylyshyn, Z. W., et al. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., and Darrell, T. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pp. 3314–3325, 2018.

Garcez, A. S., Lamb, L. C., and Gabbay, D. M. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE, 2013.

Gupta, A., Srinivasan, P., Shi, J., and Davis, L. S. 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.

Guu, K., Pasupat, P., Liu, E. Z., and Liang, P. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Huang, S., Qi, S., Zhu, Y., Xiao, Y., Xu, Y., and Zhu, S.-C. 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.

Jiang, C., Qi, S., Zhu, Y., Huang, S., Lin, J., Yu, L.-F., Terzopoulos, D., and Zhu, S.-C. 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.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2901–2910, 2017a.

Johnson, J., Hariharan, B., Van Der Maaten, L., Hoffman, J., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Inferring and executing programs for visual reasoning. In *ICCV*, 2017b.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lake, B. M. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2018.

Lample, G. and Charton, F. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*, 2019.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521 (7553):436–444, 2015a.

LeCun, Y. et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20:5, 2015b.

Lee, D., Szegedy, C., Rabe, M. N., Loos, S. M., and Bansal, K. Mathematical reasoning in latent space. *arXiv preprint arXiv:1909.11851*, 2019.

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

Liang, C., Norouzi, M., Berant, J., Le, Q. V., and Lao, N. Memory augmented policy optimization for program synthesis and semantic parsing. In *NeurIPS*, 2018.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pp. 3749–3759, 2018.

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

Marcus, G. F. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282, 1998.

Marcus, G. F. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press, 2018.

Mascharka, D., Tran, P., Soklaski, R., and Majumdar, A. 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.

Qi, S., Jia, B., and Zhu, S.-C. Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction. *ICML*, 2018.

Sun, R. *Integrating rules and connectionism for robust commonsense reasoning*. John Wiley & Sons, Inc., 1994.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Vedantam, R., Desai, K., Lee, S., Rohrbach, M., Batra, D., and Parikh, D. Probabilistic neural-symbolic models for interpretable visual question answering. *arXiv preprint arXiv:1902.07864*, 2019.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *NIPS*, 2015.

Wang, L., Zhang, D., Gao, L., Song, J., Guo, L., and Shen, H. T. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8 (3-4):229–256, 1992.

Xie, X., Liu, H., Edmonds, M., Gao, F., Qi, S., Zhu, Y., Rothrock, B., and Zhu, S.-C. Unsupervised learning of hierarchical models for hand-object interactions. In *ICRA*, 2018.

Yang, Q., Chen, Y., Xue, G.-R., Dai, W., and Yu, Y. 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.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018.

Yin, P., Zhou, C., He, J., and Neubig, G. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. *arXiv preprint arXiv:1806.07832*, 2018.

Zhao, Y. and Zhu, S.-C. Image parsing with stochastic scene grammar. In *Advances in Neural Information Processing Systems*, pp. 73–81, 2011.