Learning to Parse and Reconstruct: Bidirectional Modeling of Question-to-Program Mapping

Zeying Duan, Youtian Du*, Yuanlin Chang, Bowen Lin, Weijia Wu

MOE Key Laboratory for Intelligent Networks and Network Security, School of Automation Science and Engineering, Interdisciplinary Research Center of Frontier Science and Technology, Xi'an Jiaotong University, Xi'an, China
duanzeying@stu.xjtu.edu.cn, duyt@mail.xjtu.edu.cn, {changyl, linbowen, 2201412011}@stu.xjtu.edu.cn

Abstract

Neuro-symbolic learning has emerged as a promising paradigm for interpretable visual reasoning, where mapping natural language questions to executable programs plays a central role. However, most existing methods focus exclusively on the forward program generation from questions while overlooking the reverse process of reconstructing questions from programs. In this paper, we propose BiPaR (Bidirectional Parsing and Reconstruction), a Transformer-based framework that jointly models both program parsing and question reconstruction within a unified architecture. Unlike previous approaches that only perform forward parsing, BiPaR introduces reverse program-to-question reconstruction as a powerful auxiliary signal, which improves program generation quality and accelerates convergence, particularly under limited supervision. We further provide a theoretical analysis showing how reverse reconstruction facilitates faster optimization during training. The bidirectional modeling makes BiPaR well-suited for both supervised and semi-supervised learning scenarios. We present two architectural variants: BiPaR-Full, which employs encoder-decoder Transformers for both modules; and BiPaR-DOnly, a lightweight variant that employs a decoder-only structure for question reconstruction, reducing model complexity. Experiments on CLEVR and a GQA subset demonstrate that BiPaR significantly outperforms standard Transformer baselines. Furthermore, in the semi-supervised learning setting, BiPaR achieves notable improvements by leveraging additional questions without program annotations.

Introduction

Multimodal reasoning, such as visual question answering (VQA) (Antol et al. 2015; Qian et al. 2022), presents a fundamental challenge for artificial intelligence systems. While deep neural networks (DNNs) have achieved remarkable success across various multimodal reasoning benchmarks, they often function as black boxes-producing answers without revealing the underlying reasoning process. To address this limitation, the neural-symbolic paradigm (Andreas et al. 2016b) seeks to bridge the gap between connectionist and symbolic approaches by structuring the reasoning process through interpretable, program-like representations. A prominent line of research in this area involves using neural networks to map input questions into executable programs, which are subsequently executed by a symbolic reasoning engine to derive answers.

In neural-symbolic VQA tasks, a critical step is parsing natural language questions into symbolic programs, typically represented as sequences of operations or operator graphs, that can be executed over structured visual representations to infer answers (Chen and Zhao 2022). A representative class of parsing approaches is neural module networks (NMNs) (Andreas et al. 2016b), which decompose questions into linguistic substructures and instantiate corresponding modular networks. Building on the original NMN framework, several variants have been proposed, including the meta module network (MMN) (Chen et al. 2021) and Transformer module network (TMN) (Yamada et al. 2024). Another line of work adopts an end-to-end learning paradigm to train program generators directly from question-answer pairs (Johnson et al. 2017b). These generators, often implemented using LSTMs or Transformers (Vaswani et al. 2017), eliminate the need for explicit linguistic grammar rules but require large amounts of question-program annotations for supervised training. In addition, some approaches leverage the answering results as weak supervision to jointly improve both the program generator and the executor (Mao et al. 2019).

In this work, we focus exclusively on generating programs, i.e., operator sequences, from natural language questions. While most prior methods concentrate solely on question-to-program translation, we propose leveraging an additional learning signal: reconstructing the input question from the predicted program. This reconstruction signal provides critical feedback that improves the quality of program generation. It is particularly effective during the early stages of training when program predictions are inaccurate. Diverse and inconsistent reconstructions expose flaws in the generated programs, encouraging the model to avoid semantically incorrect or spurious outputs. Conversely, accurate program predictions support faithful question reconstructions, which can reinforce correct question-program mappings. This asymmetry reflects a general principle: correct solutions are narrow, whereas errors are diverse.

This paper introduces Bidirectional Parsing and Reconstruction (BiPaR), a Transformer-based framework that

*Corresponding author.

jointly models question-to-program parsing and program-to-question reconstruction. We present two architectural variants: BiPaR-Full, which employs encoder-decoder Transformers for both program generation and question reconstruction; and BiPaR-DOnly, a lightweight alternative that adopts a decoder-only architecture for the reconstruction module. Unlike previous approaches that focus exclusively on forward parsing, BiPaR incorporates reverse reconstruction as a powerful auxiliary signal to enhance learning and accelerate convergence, particularly under limited supervision. Moreover, the design is well-suited for semi-supervised settings, where only a small number of question-program pairs are available alongside a large corpus of unlabeled questions. In summary, our contributions are threefold: (1) We propose BiPaR, a novel framework for program parsing that incorporates bidirectional parsing and reconstruction. (2) We provide a theoretical analysis showing how reverse question reconstruction accelerates training. (3) Experimental results on CLEVR and a subset of GQA demonstrate that BiPaR significantly outperforms the strong Transformer baseline in both convergence speed and final accuracy; in the semi-supervised setting, with 10K question-program pairs and 50K unlabeled questions, BiPaR improves performance by 37.9% on CLEVR and 15.9% on the GQA subset.

Related Work

Neuro-Symbolic Learning for VQA

By combining the statistical strengths of learning with the logical rigor of reasoning, neuro-symbolic learning has emerged as a prominent direction in visual reasoning research (Besold et al. 2021). Neuro-symbolic VQA methods can be broadly categorized into three types: grammar-based approaches, end-to-end learning, and large language model (LLM)-based methods. Andreas et al. (2016b,a) proposed neural module networks (NMNs), which rely on syntactic parsing to convert questions into executable programs. Each operator in the program is associated with a predefined neural module, enabling step-by-step execution. This line of work laid the foundation for neuro-symbolic visual reasoning. However, NMNs face challenges when handling complex natural language questions due to their dependence on hand-crafted parsing rules. Moreover, their predefined operator modules often suffer from limited scalability and generalization. To address these limitations, researchers extended the NMN framework (Chen et al. 2021; Gupta et al. 2019).

To further enhance scalability and accuracy, end-to-end learning frameworks based on deep neural networks, such as LSTMs, were introduced. These models directly translate natural language questions into executable programs, supervised by the final VQA answer (Hu et al. 2017; Johnson et al. 2017b; Liang et al. 2024; Dong and Lapata 2018). In terms of visual representation, Yi et al. (2018) and Mao et al. (2019) employed object detection techniques to obtain structured representations of visual scenes, such as object attributes or scene graphs, for enhancing program execution. More recently, Transformer-based architectures have attracted increasing attention in the field. Zhao et al. (2021) proposed ProTo, a program-guided Transformer that integrates operator representations and program structures into the Transformer encoding process. Yamada et al. (2024) developed Transformer module networks (TMNs), which use stacked Transformer layers to build neural modules capable of executing individual operators either sequentially or in parallel. Addressing the link between answer prediction and interpretability, Xue, Qian, and Xu (2024) developed a program-guided variational causal reasoning network, which encodes both the program and visual inputs via a Transformer to form a program execution vector for answer prediction using a perceptron. While most existing neuro-symbolic approaches rely heavily on supervised training data, recent efforts aim to reduce this dependency (Kamali, Barezi, and Kordjamshidi 2025). Gupta and Kembhavi (2023) introduced VISPROG, a compositional visual reasoning framework that leverages LLMs and a few in-context examples to generate complex programs without explicit training, thus alleviating the need for large-scale supervision. Beyond the VQA task, neuro-symbolic program induction has also been extended to other modalities (Gao et al. 2023).

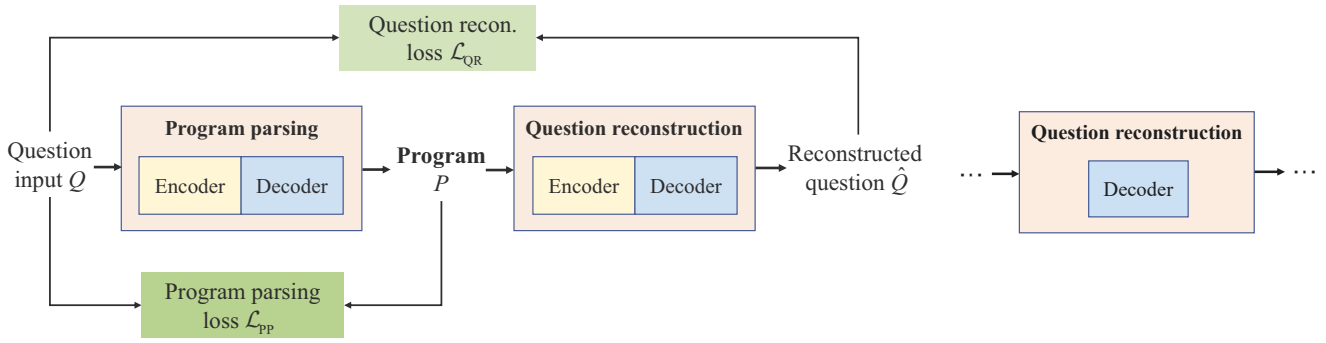
Weakly Supervised Neuro-Symbolic Learning

Neuro-symbolic learning typically requires large amounts of labeled data, such as question-program pairs in program generation tasks, leading to significant annotation costs. Weakly supervised learning with limited supervision has become a critical direction for enabling practical applications. To address this, various approaches have been proposed. Goldman et al. (2018) proposed using abstract representations for both natural language utterances and programs under closed-world assumptions. This abstraction substantially reduces the search space of potential programs and mitigates the issue of spurious programs caused by noisy supervision. Saha, Joty, and Hoi (2022) explore a different strategy in the context of machine reading comprehension, where only the final answer is provided as supervision. Tian et al. (2022) introduced a weakly supervised neural-symbolic learning model for cognitive tasks by sampling plausible reasoning paths with a back-search algorithm, which serve as pseudo-labels to guide the neural network. For the NLP reasoning tasks, Liu, Lu, and Mou (2023) learn symbolic latent structures by reinforcement learning or its relaxation.

Methodology

Problem of Program Parsing

In the neuro-symbolic learning paradigm, given a natural language question Q, the task of program parsing aims to predict a corresponding executable program P, represented as a sequence of operators. For instance, given an input question "Are there more big green things than large purple shiny cubes?", the model is expected to generate a program such as "[] scene [0] filter_size large [1] filter_color green [2] count $\cdots$ [8] count [3 9] greater_than". In this example, each element corresponds to either an operator name (e.g., filter_size), an attribute (e.g., large) or the index of a previously executed operator (e.g., [0]). Executing the generated



(a) BiPaR-Full (b) BiPaR-DOnly

Figure 1: The proposed BiPaR framework. In (b), only the question reconstruction module is shown for the BiPaR-DOnly variant, while all other components remain the same as in (a) BiPaR-Full.

program over the visual modality produces an answer in the complete VQA pipeline. This work addresses the program parsing problem using a bidirectional approach that jointly models parsing and reconstruction.

Bidirectional Parsing and Reconstruction

We represent a natural language question as a sequence of tokens: $Q = (t_1, t_2, \cdots, t_{T_q})$, where each t_i is a token obtained through tokenization of the original input question. An operator, which serves as the basic unit in an executable program, can be represented as: $\text{Oper} = \text{Oper\_name}([i], \text{arg}_1, \text{arg}_2, \cdots)$, where Oper_name is the name of the operator, $[i]$ is the index of a previously executed operator in the program, and each arg_k is an argument representing an object attribute such as name, color, or material. A program is then defined as a sequence of operators: $P = (\text{Oper}_1, \text{Oper}_2, \cdots)$, where Oper_k denotes the k-th operator in the program. To enable sequential processing within a deep neural network, we reformulate each operator into the following structure: $\text{Oper} = ([i], \text{Oper\_name}, \text{arg}_1, \text{arg}_2, \cdots)$. For fine-grained modeling, the program can also be flattened into a sequence of atomic tokens: $P = (p_1, p_2, \cdots, p_{T_p})$, where each p_i is an element of some operator Oper.

We formulate the bidirectional learning framework in BiPaR as the composition of a forward program parser g and a backward question reconstructor h:

$$\begin{cases} g : \mathcal{Q} \mapsto \mathcal{P} \\ h : \mathcal{P} \mapsto \hat{\mathcal{Q}} \end{cases} \quad \text{with } h \circ g \approx \text{Id}_{\mathcal{Q}} \tag{1}$$

where Id_Q denotes the identity mapping on the question space $\mathcal{Q}$, $\mathcal{P}$ represents the space of executable programs, and $\hat{\mathcal{Q}}$ denotes the reconstructed question space. The mapping g serves as the program parser, which translates a given question $Q \in \mathcal{Q}$ into a program $P \in \mathcal{P}$. Conversely, the mapping h functions as the question reconstructor, which attempts to recover an approximation $\hat{Q} \in \hat{\mathcal{Q}}$ of the original question from the program P. The objective of the composition $h \circ g$ is to approximate the identity function $\text{Id}_{\mathcal{Q}}$, ensuring that the reconstructed question $\hat{Q}$ remains both semantically and syntactically close to the original input question Q.

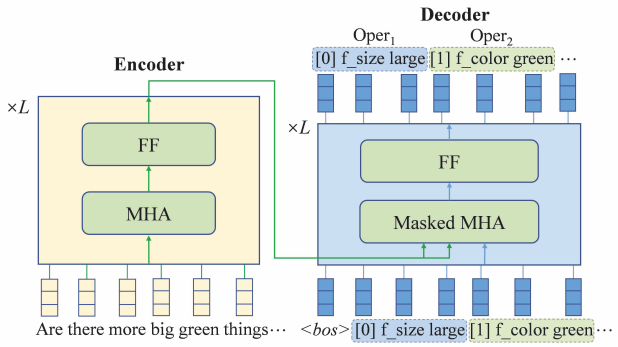


Figure 2: The program parsing module in BiPaR. Due to space limitations, 'filter_size' and 'filter_color' are abbreviated as 'f_size' and 'f_color', respectively, in the diagram.

As illustrated in Fig. 1, BiPaR comprises two primary modules: the program parsing module and the question reconstruction module, which are responsible for mapping questions to programs and reconstructing questions from programs, respectively. These two modules instantiate the abstract mappings g and h defined in Eq. (1), and are implemented using Transformer architectures due to their unified structure and strong empirical performance. Given an input question Q, the program parsing process is formulated as:

$$P = \Phi_{\text{pp}}(Q; \Theta_{\text{pp}}) \tag{2}$$

where Φ_{pp} denotes the program parsing module implemented with a Transformer architecture, parameterized by Θ_{pp}. The question reconstruction module then takes the generated program P as input and attempts to recover the original question:

$$\hat{Q} = \Phi_{\text{qr}}(P; \Theta_{\text{qr}}) \tag{3}$$

where $\hat{Q}$ is the reconstructed question and Φ_{qr} is also implemented using a Transformer architecture, parameterized by Θ_{qr}.

Program parsing module. In our implementation, the program parsing module Φ_{pp} adopts an encoder-decoder

Transformer architecture, which is well-suited for capturing complex dependencies in question-to-program translation tasks. The encoder comprises a stack of L identical Transformer layers. For each input token t_i in the input question, its representation is formed by summing the content embedding e_i^c and a sinusoidal positional encoding e_i^p, yielding a vectorial input $z_i = e_i^c + e_i^p$. The encoder processes the input sequence $(z_1, z_2, \cdots, z_{T_q})$ to produce a sequence of contextualized hidden representations $H^q = (h_1^q, h_2^q, \cdots, h_{T_q}^q)$, where each h_i^q corresponds to the token t_i. The decoder then generates the program $P = (p_1, p_2, \cdots, p_{T_p})$ in an auto-regressive manner, conditioned on both the previously generated tokens and the encoder outputs H^q.

Question reconstruction module. This module Φ_{qr} aims to reconstruct a natural language question $\hat{Q}$ from a given program P, serving as the reverse process of program parsing. We implement two variants of BiPaR depending on the reconstruction strategy. In the BiPaR-Full variant, the module adopts a standard encoder-decoder Transformer architecture. The input program tokens are embedded and combined with sinusoidal positional encodings to form input vectors, which are then encoded into a hidden representation H^p. The decoder generates the reconstructed question $\hat{Q} = (\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_{T_q})$ in an auto-regressive manner, conditioned on H^p. In BiPaR-DOnly, the module consists of a lightweight decoder-only Transformer. It directly translates the vectorial representation of the program P into the reconstructed question $\hat{Q}$, without an explicit encoder stage.

Training Paradigm

Supervised learning. In the supervised learning setting, the objective is to learn the optimal parameters Θ_{pp} and Θ_{qr} for the program parsing and question reconstruction modules, respectively, using N_{L} annotated question-program pairs $X_{\text{L}} = \{(Q^{(i)}, P^{(i)})\}_{i=1}^{N_{\text{L}}}$. For program parsing, the model is trained to minimize the cross-entropy loss:

$$\mathcal{L}_{\text{PP}} = -\sum_{i=1}^{N_{\text{L}}} \sum_{t=1}^{T_{p,i}} \log \Pr(p_t^{(i)} | \hat{p}_{<t}^{(i)}, Q^{(i)}; \Theta_{\text{pp}}) \tag{4}$$

where $p_t^{(i)}$ denotes the ground-truth token at time step t in the i-th program, and $\hat{p}_{<t}^{(i)}$ represents the previously generated tokens. For question reconstruction, we minimize the following loss to encourage semantic consistency between the original question and its reconstructed counterpart:

$$\mathcal{L}_{\text{QR}} = -\sum_{i=1}^{N_{\text{L}}} \sum_{t=1}^{T_{q,i}} \log \Pr(t_t^{(i)} | \hat{t}_{<t}^{(i)}, P_i; \Theta_{\text{qr}}) \tag{5}$$

where $t_t^{(i)}$ is the ground-truth token in the i-th question, and $\hat{t}_{<t}^{(i)}$ denotes the previously generated tokens before step t in the reconstructed question. The final training objective jointly optimizes both components:

$$\mathcal{L} = \lambda \mathcal{L}_{\text{PP}} + (1 - \lambda) \mathcal{L}_{\text{QR}} \tag{6}$$

where $\lambda \in [0, 1]$ balances the contributions of program parsing and question reconstruction.

Online semi-supervised learning. In the semi-supervised learning setting, BiPaR leverages both labeled question-program pairs X_{L} and unlabeled questions $X_{\text{U}} = \{Q^{(i)}\}_{i=N_{\text{L}}+1}^{N_{\text{L}}+N_{\text{U}}}$ to enhance program parsing. We adopt the following strategy for online semi-supervised learning: For each unlabeled question $Q^{(i)} \in X_{\text{U}}$, we generate a pseudo program $\widetilde{P}^{(i)}$ using a pre-trained program parser $\widetilde{\Phi}_{\text{pp}}(\cdot; \widetilde{\Theta}_{\text{pp}})$ trained on X_{L}, i.e., $\widetilde{P}^{(i)} = \widetilde{\Phi}_{\text{pp}}(Q^{(i)}; \widetilde{\Theta}_{\text{pp}})$, and use it for question reconstruction while keeping the pseudo program fixed throughout training. The overall training objective for the online semi-supervised setting is:

$$\mathcal{L}' = \lambda' \mathcal{L}'_{\text{PP}} + (1 - \lambda') \mathcal{L}'_{\text{QR}} \tag{7}$$

where $\mathcal{L}'_{\text{PP}}$ measures the difference between the predicted and pseudo programs, $\mathcal{L}'_{\text{QR}}$ evaluates the reconstruction error between the input and reconstructed questions, and λ' balances the two terms.

Theoretical Analysis

This subsection analyzes the impact of question reconstruction on model training. Let Q denote the input question and $P^\star$ the corresponding ground-truth program. Let $\hat{P} = g(Q)$ and $\hat{Q} = h(\hat{P})$ be the predicted program and reconstructed question, respectively, where g and h are defined in Eq. (1). We begin by defining the conditional variance of the predicted program $\hat{P}$ given Q as:

$$\text{Var}(\hat{P}|Q) \triangleq \mathbb{E}\|\hat{P} - P^\star\|^2 = \text{tr}(\Sigma_P) \tag{8}$$

where Σ_P is the covariance matrix of $\hat{P}$. Here we assume, for simplicity, that $P^\star \approx \mathbb{E}[\hat{P}|Q]$. Assuming the question reconstruction function h is locally differentiable around $P^\star$, we perform a first-order Taylor expansion of $h(\hat{P})$:

$$\hat{Q} = h(\hat{P}) \approx h(P^\star) + J_h(P^\star)(P - P^\star) \tag{9}$$

where $J_h(P^\star) = \frac{\partial h}{\partial P}|_{P^\star}$ is the Jacobian matrix of h evaluated at $P^\star$. Under this approximation, the conditional covariance of the reconstructed question $\hat{Q}$ given Q is:

$$\text{Cov}(\hat{Q}|Q) \approx J_h(P^\star) \Sigma_P J_h(P^\star)^T \tag{10}$$

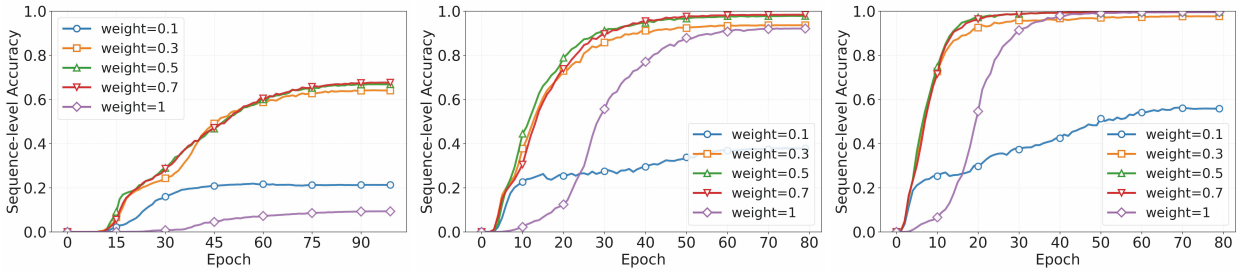
Taking the trace on both sides gives the conditional variance of the reconstructed question:

$$\begin{aligned} \text{Var}(\hat{Q}|Q) \triangleq \text{tr}(\text{Cov}(\hat{Q}|Q)) &\approx \text{tr}(J_h \Sigma_P {J_h}^T) \\ &\leq \|J_h\|_F^2 \lambda_{\max}(\Sigma_P) \\ &\leq \|J_h\|_F^2 \text{tr}(\Sigma_P) \end{aligned} \tag{11}$$

where $\lambda_{\max}(\Sigma_P)$ denotes the largest eigenvalue of Σ_P, and $\|\cdot\|_F$ is the Frobenius norm. Combining the above, we obtain:

$$\text{Var}(\hat{Q}|Q) \lesssim \|J_h(P^\star)\|_F^2 \text{Var}(\hat{P}|Q) \tag{12}$$

This analysis reveals that the reconstruction variance $\text{Var}(\hat{Q}|Q)$ is significantly amplified when the prediction variance $\text{Var}(\hat{P}|Q)$ is large and the question reconstruction function h is highly sensitive in the neighborhood of $\hat{P}$ (i.e., has a large $\|J_h\|_F^2$). Therefore, during early training, when the predicted program $\hat{P}$ is far from $P^\star$, the discrepancy between $\hat{Q}$ and Q serves as an effective amplified error signal, enabling faster convergence via stronger gradient feedback.



(a) Training set size: 10K (b) Training set size: 30K (c) Training set size: 50K

Figure 3: The influence of λ in Eq. (6) for CLEVR.

Experiments

Datasets and Metrics

We conduct experiments on two public datasets: CLEVR (Johnson et al. 2017a) and GQA (Hudson and Manning 2019), both annotated with functional programs. (1) CLEVR is a synthetic visual question answering (VQA) dataset consisting of images made up of simple geometric shapes, accompanied by approximately one million automatically generated questions. These questions span various tasks, including querying object attributes, comparing attributes, checking for existence, counting, and performing integer comparisons. The official split provides 699,989 training questions and 149,991 validation questions. (2) GQA is a large-scale dataset for real-world visual reasoning and compositional question answering. It comprises 22 million diverse reasoning questions generated from 113,018 real-world images, each paired with a functional program that represents its semantic structure. GQA primarily evaluates reasoning abilities such as object and attribute recognition, relational and spatial reasoning, logical inference, and comparison. For our experiments, we use a subset of GQA, randomly sampling 700,000 questions from the training split and 150,000 from the validation split.

We evaluate the methods using both sequence-level and token-level accuracy. Sequence-level accuracy assesses whether the entire predicted program sequence exactly matches the gold reference sequence. A prediction is considered correct only if all tokens match the ground truth in both content and order. Formally, it is defined as:

$$Seq\_Acc = \frac{1}{N}\sum_{i=1}^{N} I(\hat{P}^{(i)} = P^{(i)}) \tag{13}$$

where $\hat{P}^{(i)}$ and $P^{(i)}$ denote the predicted program and ground-truth program, respectively, for the i-th input question. Token-level accuracy measures the proportion of individual tokens that exactly match their gold counterparts at the same position across all sequences:

$$Tok\_Acc = \frac{1}{M}\sum_{i=1}^{N}\sum_{t=1}^{T_{p,i}} I(\hat{p}_t^{(i)} = p_t^{(i)}) \tag{14}$$

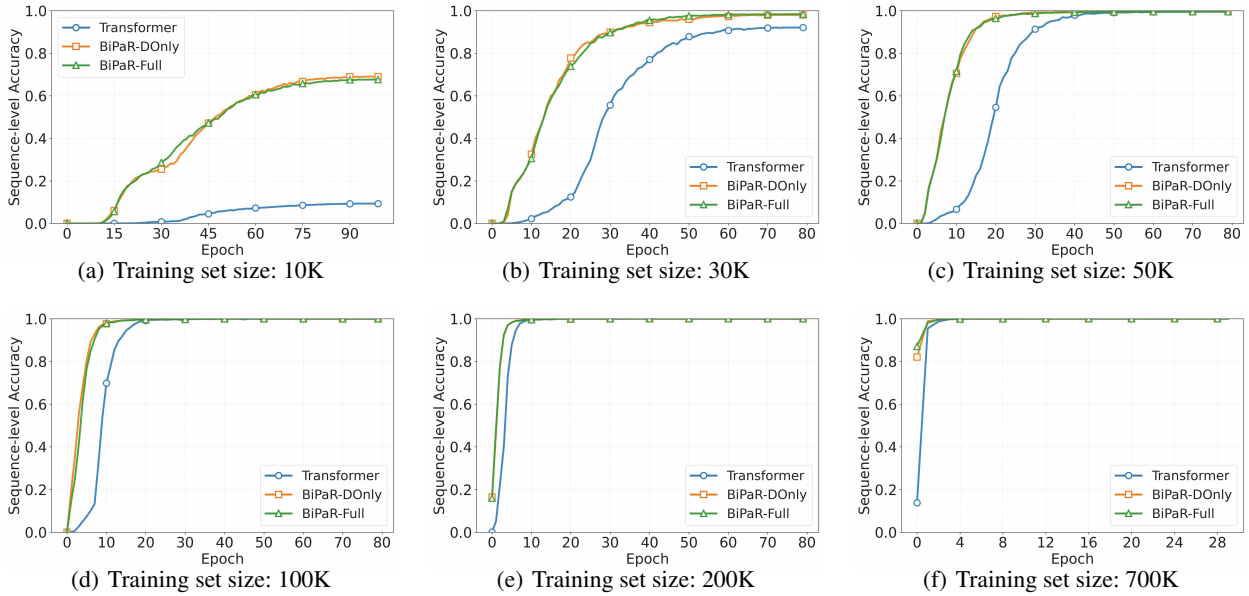
where $\hat{p}_t^{(i)}$ and $p_t^{(i)}$ denote the t-th token in the predicted program and ground-truth program, respectively, for the i-th input question, $T_{p,i}$ is the length of $P^{(i)}$, and M is the total number of tokens across all N programs.

Experimental Results on CLEVR

For CLEVR, the BiPaR model consists of six encoder layers and six decoder layers, each incorporating multi-head self-attention with eight heads. Input sequences are tokenized using a pretrained tokenizer, with identical vocabulary sizes for both the source and target languages. All sequences are truncated or padded to a fixed maximum length of 128 tokens to ensure consistent input lengths. The embedding dimension is set to 64, and the hidden dimension of the feed-forward network is 256. The model is initialized using Xavier initialization and optimized with the Adam optimizer, employing an initial learning rate of 1.0×10^{-4} and momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.98$, following standard practices in Transformer-based models. A cosine annealing learning rate scheduler is applied to progressively reduce the learning rate over 100 training epochs. All results are reported as the average accuracy over five independent runs. Experiments were conducted on a Linux system equipped with an NVIDIA RTX 4090 GPU (24GB VRAM), using Python 3.9.21, PyTorch 2.6.0 with CUDA 11.6, and TorchAudio 0.13.1.

As illustrated in Fig. 3, we first evaluate the impact of the weight coefficient λ in Eq. (6) using three training set sizes: 10K, 30K, and 50K. We examine five values of λ: 0.1, 0.3, 0.5, 0.7, and 1.0, and report the sequence-level accuracy of BiPaR-Full on the validation set. As the weight λ increases, the relative contribution of the question reconstruction objective diminishes. The results indicate that BiPaR achieves optimal performance when $\lambda = 0.7$, while both smaller and larger values result in decreased accuracy. Accordingly, we adopt $\lambda = 0.7$ for subsequent supervised learning experiments with both BiPaR-Full and BiPaR-DOnly variants.

We conducted experiments with six training set sizes: 10K, 30K, 50K, 100K, 200K, and 700K (corresponding to 699,989 samples). Evaluation was performed on the validation set comprising 149,991 questions. Fig. 4 presents the sequence-level accuracy results for BiPaR-Full, BiPaR-DOnly, and the Transformer baseline (Xue, Qian, and Xu 2024). From the figure, we observe that both variants of BiPaR achieve the similar performance and significantly outperform the Transformer baseline in terms of convergence



(a) Training set size: 10K (b) Training set size: 30K (c) Training set size: 50K

(d) Training set size: 100K (e) Training set size: 200K (f) Training set size: 700K

Figure 4: The results of program generation on CLEVR across different sizes of training sets.

Size	CLEVR						GQA					
	Sequence-level Accuracy			Token-level Accuracy			Sequence-level Accuracy			Token-level Accuracy		
	TransF.	DOnly	Full	TransF.	DOnly	Full	TransF.	DOnly	Full	TransF.	DOnly	Full
10K	0.093	0.691	0.676	0.915	0.988	0.987	0.095	0.587	0.579	0.892	0.962	0.961
30K	0.921	0.980	0.984	0.998	1.000	1.000	0.517	0.771	0.770	0.961	0.982	0.982
50K	0.995	0.998	0.996	1.000	1.000	1.000	0.553	0.791	0.794	0.975	0.984	0.984
100K	1.000	1.000	1.000	1.000	1.000	1.000	0.658	0.813	0.811	0.982	0.986	0.986
200K	1.000	1.000	1.000	1.000	1.000	1.000	0.762	0.825	0.825	0.985	0.987	0.987
700K	1.000	1.000	1.000	1.000	1.000	1.000	0.818	0.833	0.832	0.987	0.988	0.987

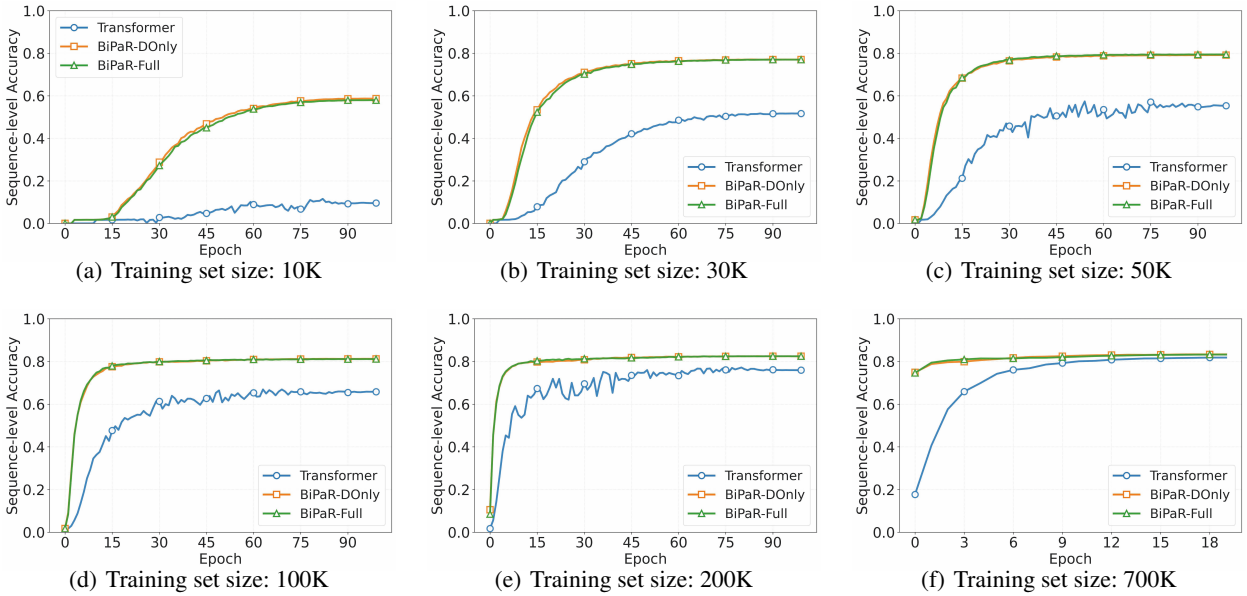
Table 1: The accuracy of BiPaR for CLEVR and the GQA subset. Due to space limitations, "BiPaR-DOnly","BiPaR-Full", and "Transformer" are abbreviated as "DOnly", "Full" and "TransF.", respectively.

speed, particularly with smaller training sets. For instance, with the 30K training set, BiPaR reaches approximately 0.9 accuracy within 30 epochs, whereas the Transformer only attains this level at the end of training. The learning curves indicate that incorporating question reconstruction substantially accelerates convergence compared to the baseline, which only employs forward program parsing. For larger training sets, both our approach and the baseline achieve comparable final accuracy close to 1.0. Table 1 reports the quantitative results of both sequence-level and token-level accuracy at convergence across the six training sets. We find that the BiPaR approach consistently improves inference accuracy compared to the Transformer baseline.

Experimental Results on GQA

For the GQA benchmark, the BiPaR model adopts a Transformer architecture with 12 encoder layers and 12 decoder layers, each equipped with multi-head self-attention comprising eight heads. All sequences are truncated or padded to a fixed maximum length of 128 tokens to ensure uniform input lengths. The embedding dimension is set to 256, and the hidden dimension of the feed-forward network is 1024. The model is initialized using Xavier initialization and optimized with the AdamW optimizer, employing an initial learning rate of 5.0×10^{-5} and momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.98$, following standard practices in Transformer-based models. A cosine annealing learning rate scheduler is applied to progressively reduce the learning rate over 100 training epochs. All results are reported as the average accuracy over five independent runs.

We evaluate BiPaR's convergence speed and final performance on GQA using six training set sizes: 10K, 30K, 50K, 100K, 200K, and 700K, randomly sampled from the training subset introduced above. Inference is conducted on the validation set consisting of 150,000 questions. Fig. 5 presents the sequence-level accuracy results on the validation set for BiPaR-Full, BiPaR-DOnly, and the Transformer baseline (Xue, Qian, and Xu 2024). As shown in the figure,



(a) Training set size: 10K (b) Training set size: 30K (c) Training set size: 50K

(d) Training set size: 100K (e) Training set size: 200K (f) Training set size: 700K

Figure 5: The results of program generation on the subset of GQA across different sizes of training sets.

both variants of BiPaR achieve comparable performance and consistently outperform the Transformer baseline in terms of convergence speed across all training set sizes. For instance, with the 30K training set, BiPaR reaches the Transformer's final accuracy in fewer than 15 epochs and further achieves a significantly higher final performance. When the training set size is increased to 700K, the Transformer baseline attains accuracy close to that of BiPaR, but its convergence remains considerably slower. These results on GQA further demonstrate that question reconstruction provides a valuable learning signal for program parsing. Table 1 reports the quantitative results of both sequence-level and token-level accuracy of all methods trained on the six training sets. The results further confirm the consistent effectiveness of BiPaR.

Results for Semi-supervised Learning

As shown in Table 2, we evaluate the proposed BiPaR under a semi-supervised learning setting in this subsection. Specifically, we use 10K question-program pairs as labeled data for supervised learning (denoted as SL(10K) in the table), and augment the training set with additional unlabeled questions (without program annotations) of varying sizes: 5K, 30K, and 50K. These semi-supervised configurations are referred to as SSL(10K+5K), SSL(10K+30K), and SSL(10K+50K), respectively. We empirically set $\lambda' = 0.7$ in Eq. (7) to reduce the influence of potentially noisy supervision derived from the predicted program. Table 2 presents the results on the validation set under the semi-supervised learning setting. We observe that incorporating additional unlabeled questions consistently improves the program parsing performance of BiPaR compared to training with labeled data alone. For example, compared to the supervised-only baseline, BiPaR-Full achieves relative gains in sequence-level accuracy of 29.6%, 36.2%, and 37.9% on CLEVR, and 11.4%, 14.7%, and 15.9% on the GQA subset, under the three semi-supervised configurations, respectively.

	CLEVR		GQA	
	DOnly	Full	DOnly	Full
SL (10K)	0.691	0.676	0.587	0.579
SSL(10K+5K)	0.848	0.876	0.641	0.645
SSL(10K+30K)	0.899	0.921	0.655	0.664
SSL(10K+50K)	0.919	0.932	0.668	0.671

Table 2: The sequence-level accuracy of BiPaR with semi-supervised learning settings. Due to space limitations, "BiPaR-DOnly" and "BiPaR-Full" are abbreviated as "DOnly" and "Full", respectively.

Conclusion

We introduced BiPaR, a bidirectional framework that unifies program parsing and question reconstruction to enhance program generation from questions in neural-symbolic learning tasks. Two variants were proposed: BiPaR-Full and BiPaR-DOnly. By incorporating the reverse program-to-question reconstruction as an auxiliary signal, BiPaR not only improves the quality of generated programs but also accelerates training, as supported by our theoretical analysis. Experimental results on CLEVR and a subset of GQA demonstrate that BiPaR significantly outperforms the strong Transformer baseline in terms of both convergence speed and final accuracy. Furthermore, BiPaR exhibits strong potential in semi-supervised settings by effectively leveraging unlabeled questions. In future work, we plan to extend BiPaR to broader multimodal reasoning tasks, such as full-scale GQA.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (62576272, T2341003) and the Fundamental Research Funds for the Central Universities (xzy012023029).

References

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016a. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016b. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 39–48.

Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C. L.; and Parikh, D. 2015. VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2425–2433.

Besold, T. R.; Bader, S.; Bowman, H.; Domingos, P.; Hitzler, P.; Kühnberger, K.-U.; Lamb, L. C.; Lima, P. M. V.; de Penning, L.; Pinkas, G.; et al. 2021. Neural-symbolic learning and reasoning: A survey and interpretation 1. In *Neuro-symbolic Artificial Intelligence: The State of the Art*, 1–51. IOS press.

Chen, S.; and Zhao, Q. 2022. Rex: Reasoning-aware and grounded explanation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15586–15595.

Chen, W.; Gan, Z.; Li, L.; Cheng, Y.; Wang, W.; and Liu, J. 2021. Meta module network for compositional visual reasoning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 655–664.

Dong, L.; and Lapata, M. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 731–742.

Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; and Neubig, G. 2023. PAL: Program-aided language models. In *International Conference on Machine Learning*, 10764–10799. PMLR.

Goldman, O.; Latcinnik, V.; Naveh, U.; Globerson, A.; and Berant, J. 2018. Weakly-supervised semantic parsing with abstract examples. In *the 56th Annual Meeting of the Association for Computational Linguistics*, 1809–1819.

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

Gupta, T.; and Kembhavi, A. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14953–14962.

Hu, R.; Andreas, J.; Rohrbach, M.; Darrell, T.; and Saenko, K. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 804–813.

Hudson, D. A.; and Manning, C. D. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6700–6709.

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

Johnson, J.; Hariharan, B.; Van Der Maaten, L.; Hoffman, J.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017b. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2989–2998.

Kamali, D.; Barezi, E. J.; and Kordjamshidi, P. 2025. NeSyCoCo: A neuro-symbolic concept composer for compositional generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 4184–4193.

Liang, L.; Sun, G.; Qiu, J.; and Zhang, L. 2024. Neural-symbolic videoQA: Learning compositional spatio-temporal reasoning for real-world video question answering. *arXiv preprint arXiv:2404.04007*.

Liu, X.; Lu, Z.; and Mou, L. 2023. Weakly supervised reasoning by neuro-symbolic approaches. In *Compendium of Neurosymbolic Artificial Intelligence*, 665–692. IOS Press.

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

Qian, T.; Chen, J.; Chen, S.; Wu, B.; and Jiang, Y.-G. 2022. Scene graph refinement network for visual question answering. *IEEE Transactions on Multimedia*, 25: 3950–3961.

Saha, A.; Joty, S.; and Hoi, S. C. 2022. Weakly supervised neuro-symbolic module networks for numerical reasoning over text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 11238–11247.

Tian, J.; Li, Y.; Chen, W.; Xiao, L.; He, H.; and Jin, Y. 2022. Weakly supervised neural symbolic learning for cognitive tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 5888–5896.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Xue, D.; Qian, S.; and Xu, C. 2024. Integrating neural-symbolic reasoning with variational causal inference network for explanatory visual question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 7893–7908.

Yamada, M.; D'Amario, V.; Takemoto, K.; Boix, X.; and Sasaki, T. 2024. Transformer module networks for systematic generalization in visual question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yi, K.; Wu, J.; Gan, C.; Torralba, A.; Kohli, P.; and Tenenbaum, J. 2018. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. *Advances in Neural Information Processing Systems*, 31.

Zhao, Z.; Samel, K.; Chen, B.; et al. 2021. Proto: Program-guided Transformer for program-guided tasks. *Advances in Neural Information Processing Systems*, 34: 17021–17036.