THE NEURO-SYMBOLIC CONCEPT LEARNER: INTERPRETING SCENES, WORDS, AND SENTENCES FROM NATURAL SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose the Neuro-Symbolic Concept Learner (NS-CL), a model that learns visual concepts, words, and semantic parsing of sentences without explicit supervision on any of them; instead, our model learns by simply looking at images and reading paired questions and answers. Our model builds an object-based scene representation and translates sentences into executable, symbolic programs. To bridge the learning of two modules, we use a neuro-symbolic reasoning module that executes these programs on the latent scene representation. Analog to the human concept learning, given the parsed program, the perception module learns visual concepts based on the language description of the object being referred to. Meanwhile, the learned visual concepts facilitate learning new words and parsing new sentences. We use curriculum learning to guide searching over the large compositional space of images and language. Extensive experiments demonstrate the accuracy and efficiency of our model on learning visual concepts, word representations, and semantic parsing of sentences. Further, our method allows easy generalization to new object attributes, compositions, language concepts, scenes and questions, and even new program domains. It also empowers applications including visual question answering and bidirectional image-text retrieval.

1 INTRODUCTION

Humans are capable of learning visual concepts by jointly understanding vision and language (Fazly et al., 2010; Chrupała et al., 2015; Gauthier et al., 2018). Take the example shown in Figure 1(I). Imagine that a human with no prior knowledge of colors, is presented with the images of the red and green cubes paired with the questions and answers. She can easily identify the differences in certain component of the visual appearance (e.g. color in this case) of the objects and align the component to the corresponding word in the question and answer (`Red` and `Green`). Other single-object level attributes (e.g. shape) can also be learned in a similar fashion. Starting from there, a human is also able to inductively learn the correspondence between visual concepts and word semantics (e.g. spatial relations and referential expressions) Figure 1(II), and unravel compositional logic from complex questions assisted by the learned visual concepts Figure 1(III) (Abend et al., 2017).

Motivated by the way humans jointly perceive visual concepts and understand languages through a joint reasoning process (Gauthier et al., 2018), we propose a neuro-symbolic concept learner (NS-CL) that jointly learns visual perception, words, and semantic language parsing based on a visual question answering (VQA) setup. We employ a visually-grounded semantic parser for translating questions into executable programs, and a neural-based perception module that extracts object-level representations from the scene. A symbolic program executor then reads out the perceptual representation of objects, classifies their attributes/relations and executes the program to obtain an answer.

Our model learns from natural supervision signals (i.e. VQA pairs) through a curriculum-based approach. It starts from learning representations of individual objects from short questions (e.g., What's the color of the cylinder?) on simple scenes ($\leq$3 objects). It then learns relational concepts by leveraging object-level concepts to interpret object referrals (e.g., Is there a box right of a cylinder?). Finally, the model iteratively adapts to more complex scenes and compositional questions.

Our model recovers an object-based representation for visual scenes. Meanwhile, based on the learned visual representations, we propose a visually-grounded semantic parsing approach to resolve

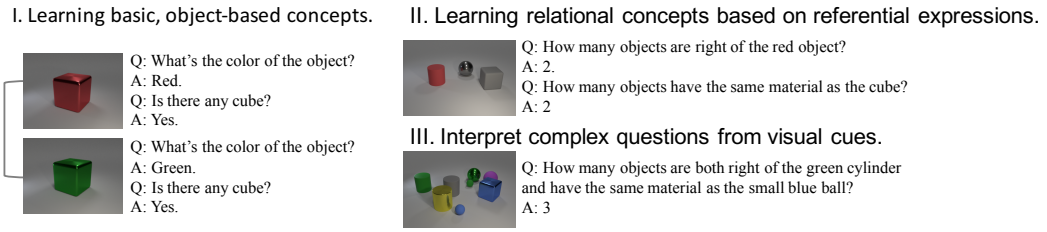


Figure 1: Humans learn visual concepts, words, and semantic parsing jointly and incrementally. **I.** Learning visual concepts (red *vs*. green) starts from looking at simple scenes, reading simple questions, and reasoning over contrastive examples (Fazly et al., 2010). **II.** Afterwards, we can interpret referential expressions based on the learned object-based concepts, and learn relational concepts (e.g., on the right of, the same material as). **III** Finally, we can interpret complex questions from visual cues by exploiting the compositional structure.

the correspondence between word semantics and visual concepts. It also learns the semantic parsing of sentences requiring *zero* program annotations.

These disentangled and structural representations enables an interpretable and robust reasoning for VQA. Beyond showing a state-of-the-art performance on the CLEVR VQA dataset, our neuro-symbolic approach naturally supports combinatorial generalization w.r.t. the complexity of the scene and the programs (e.g., the depth of the program tree). We also propose solutions to the visual compositional generalization (CLEVR-CoGenT (Johnson et al., 2017a)) and the incremental learning of concepts. The learned visual concepts can be easily applied into other domains such as image-caption retrieval by only changing the program specification.

2 Related work

Our model is related to research on joint learning from visual data and natural language. In particular, there are many papers that learn visual concepts from descriptive visually-grounded languages, such as image-captioning or visually-grounded question-answer pairs (Kiros et al., 2014; Mao et al., 2016; Vendrov et al., 2016; Ganju et al., 2017), dense language descriptions for scenes (Johnson et al., 2016), video-captioning (Donahue et al., 2015) and video-text alignment (Zhu et al., 2015).

Visual question answering (VQA) stands out as it requires understanding both visual content and language. The state-of-the-art approaches usually use neural attentions (Malinowski & Fritz, 2014; Chen et al., 2015; Yang et al., 2016; Xu & Saenko, 2016). Beyond question answering, Johnson et al. (2017a) proposed the CLEVR (VQA) dataset to diagnose reasoning models. CLEVR contains synthetic visual scenes and questions generated from latent programs. Table 1 compares our model with state-of-the-art visual reasoning models (Andreas et al., 2016; Suarez et al., 2018; Santoro et al., 2017) along four directions: visual features, semantics, inference, and the requirement of extra labels.

For visual representation, Johnson et al. (2017b) encoded visual scenes into a convolutional feature map for program operators. Mascharka et al. (2018); Hudson & Manning (2018) used attention as intermediate representations for transparent program execution. Recently, Yi et al. (2018) explored an interpretable, object-based visual representation for visual reasoning. It performs well, but requires fully-annotated scenes during training. Our model also adopts an object-based visual representation, but the representation is learned only based on natural supervision (questions and answers).

There are two types of approaches in semantic sentence parsing for visual reasoning: implicit programs as conditioned neural operations (e.g., conditioned convolution and dual attention) (Perez et al., 2017; Hudson & Manning, 2018) and explicit programs as sequences of symbolic tokens (Johnson et al., 2017b; Mascharka et al., 2018). Explicit programs gain better interpretability, but usually require extra supervision such as ground-truth program annotations for training. This restricts their application. We propose a visually-grounded semantic parsing approach to parse questions in natural languages into explicit programs with *zero* program annotations. Given the semantic parsing of questions into programs, Yi et al. (2018) proposes a purely symbolic executor for the inference of the answer in the logic space. Compared with theirs, we propose an quasi-symbolic executor for VQA.

Our work is also related to learning interpretable and disentangled representations for visual scenes using neural networks. Kulkarni et al. proposed convolutional inverse graphics networks for learning and inferring pose of faces, while Yang et al. (2015) learned disentangled representation of pose of chairs from images. Wu et al. (2017) proposes the neural scene de-rendering franework as an inverse process of any rendering process. Siddharth et al. (2017); Higgins et al. (2018) learned disentangled

Models	Visual Features	Semantics	Extra Labels		Inference
			# Prog.	Attr.	
FiLM (Perez et al., 2017)	Convolutional	Implicit	0	No	Feature Manipulation
IEP (Johnson et al., 2017b)	Convolutional	Explicit	700K	No	Feature Manipulation
MAC (Hudson & Manning, 2018)	Attentional	Implicit	0	No	Feature Manipulation
Stack-NMN (Hu et al., 2018)	Attentional	Implicit	0	No	Attention Manipulation
TbD (Mascharka et al., 2018)	Attentional	Explicit	700K	No	Attention Manipulation
NS-VQA (Yi et al., 2018)	Object-Based	Explicit	0.2K	Yes	Symbolic Execution
NS-CL	Object-Based	Explicit	0	No	Symbolic Execution

Table 1: Comparison with other frameworks on the CLEVR VQA dataset, w.r.t. visual features, implicit or explicit semantics and supervisions.

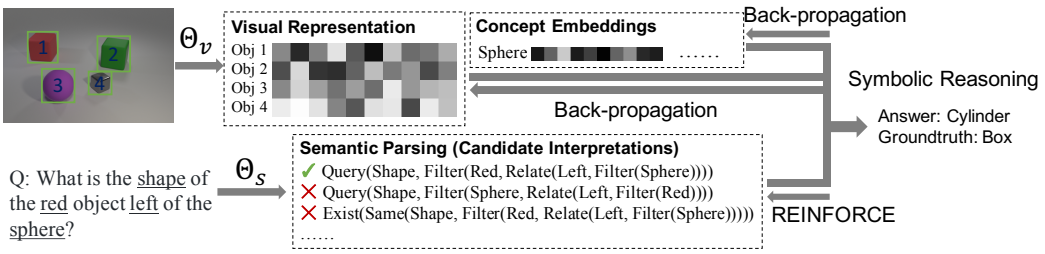


Figure 2: We propose to use neural symbolic reasoning as a bridge to jointly learn visual concepts, words, and semantic parsing of sentences.

representations using deep generative models. In contrast, we propose an alternative interpretable and disentangled representations learning approach through joint reasoning with language.

3 NEURO-SYMBOLIC CONCEPT LEARNER

We present our neuro-symbolic concept learner, which uses a symbolic reasoning process to learn visual concepts, words, and semantic parsing of sentences without explicit annotations for any of them. We first use a visual perception module to construct an object-based representation for a scene, and run a semantic parsing module to translate a question into an executable program. We then apply a quasi-symbolic program executor to infer the answer based on the scene representation. We use paired images, questions, and answers to jointly train the visual and language modules.

Shown in Figure 2, given an input image, the visual perception module detects objects in the scene and extracts a deep, latent representation for each of them. The semantic parsing module translates an input question in natural language into an executable program in a domain specific language (DSL). The generated programs have a hierarchical structure of symbolic, functional modules, each fulfilling a specific operation over the scene representation. The explicit program semantics enjoys compositionality, interpretability, and generalizability.

The program executor executes the program upon the derived scene representation and answers the question. Our program executor works in a symbolic and deterministic manner. This feature ensures a transparent execution trace of the program. Our program executor has a fully differentiable design w.r.t. the visual representations and concept embeddings, which can be updated using standard back-propagation during training.

3.1 MODEL DETAILS

Visual perception. Shown in Figure 2, given the input image, we use a Mask R-CNN pipeline (He et al., 2017) to generate object proposals for all objects. The bounding box for each single object paired with the original image is then sent to a ResNet-34 (He et al., 2015) to extract the region-based (by RoI Align) and image-based features respectively. We concatenate them to represent each object. Here, the inclusion of the representation of the full scene adds the contextual information, which is essential for the inference of relative attributes such as size or spatial position.

Concept quantization. Visual reasoning requires determining an object's attributes (e.g., its color or shape). We assume each visual attribute (e.g., shape) contains a set of possible visual concept (e.g., `Cube`). In NS-CL, visual attributes are implemented as neural operators (Nagarajan & Grauman, 2018), mapping the object representation into an attribute-specific embedding space. Figure 3 shows

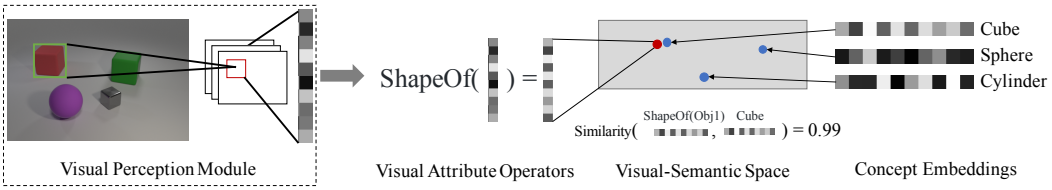


Figure 3: We treat attributes such as `Shape` and `Color` as neural operators. The operators map object representations into a visual-semantic space. We use similarity-based metric to classify objects.

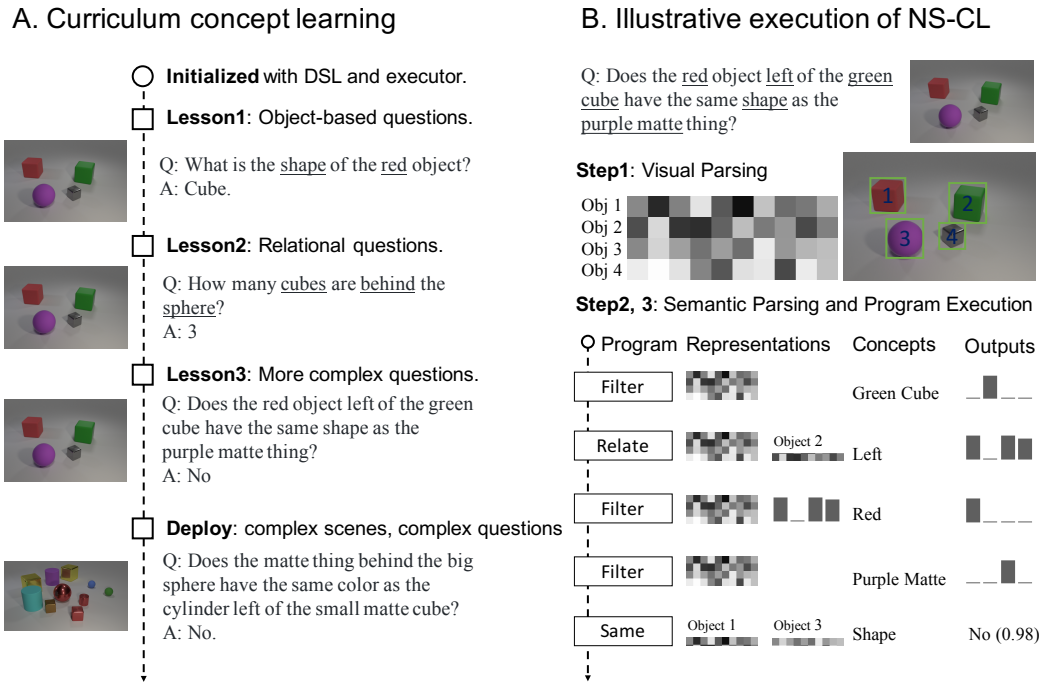


Figure 4: **A.** Demonstration of the curriculum learning of visual concepts, words, and semantic parsing of sentences by watching images and reading paired questions and answers. Scenes and questions of different complexities are illustrated to the learner in an incremental manner. **B.** Illustration of our neuro-symbolic inference model for VQA. The perception module begins with parsing visual scenes into object-based deep representations, while the semantic parser parse sentences into executable programs. A symbolic execution process bridges two modules.

an inference of the shape of an object. Visual concepts that belong to the shape attribute, including `Cube`, `Sphere` and `Cylinder`, are represented as vectors in this shared space. These concept vectors are also learned along the process. We measure the cosine distances $\langle \cdot, \cdot \rangle$ between these vectors to decide the shape of the object. Specifically, we compute the probability that an object o_i is a cube by $\sigma\left(\langle \text{ShapeOf}(o_i), v_{\texttt{Cube}} \rangle - \gamma \right) / \tau$, where $\text{ShapeOf}(\cdot)$ denotes the neural operator, $v_{\texttt{Cube}}$ the concept embedding of `Cube` and σ the Sigmoid function. γ and τ are scalar constants for scaling and shifting the values of similarities. We classify relational concepts (e.g., `Left`) between a pair of objects similarly, except that we concatenate the visual representations for both objects to form a representation of the pair.

DSL and semantic parsing. The semantic parsing module translates a natural language question into an executable program with a hierarchy of primitive operations, represented in a domain-specific language (DSL) designed for VQA. The DSL covers a set of fundamental operations for visual reasoning, such as filtering out objects with certain concepts or query the attribute of an object. The operations share the same input and output interface, and thus can be compositionally combined to form programs of any complexity. We include a complete specification of the DSL used by our framework in the Appendix A.

Our semantic parser generates the hierarchies of latent programs in a sequence to sequence manner (Sutskever et al., 2014). We use an bidirectional GRU (Cho et al., 2014) to encode an input question, which outputs a fixed-length embedding of the question. A decoder based on GRU cells is applied to the embedding, and recovers the hierarchy of operations as the latent program.

Quasi-symbolic program execution. Given the latent program recovered from the question in natural language, a symbolic program executor executes the program and derives the answer based on the object-based visual representation. Our program executor is a collection of deterministic functional modules designed to realize all logic operations specified in the DSL. Figure 4(B) shows an illustrative execution trace of a program.

To make the execution differentiable w.r.t. visual representations, we represent the intermediate results in a probabilistic manner. Specifically, a set of objects is represented by a real-valued mask over all objects in the scene. Each element, $\texttt{Mask}_i \in [0, 1]$ represents the probablity that the i-th object of the scene belongs to the set. For example, shown in Figure 4(B), the first `Filter` operation outputs a mask of length 4 (there are in total 4 objects in the scene), with each element representing the probability that the corresponding object is selected out (i.e., the probability that each object is a green cube). The output "mask" on the objects will be fed into the next module (`Relate` in this case) as input and the execution of programs continues. The last module outputs the final answer to the question. We refer interested readers to Appendix C for the technical implementation of all VQA operators.

3.2 Training paradigm

Optimization objective. The optimization objective of NS-CL is composed of two parts: concept learning and language understanding. Our goal is to find the optimal parameters Θ_v of the visual perception module and Θ_s of the semantic parsing module, to maximize the likelihood of answering the question Q correctly:

$$\Theta_v, \Theta_s \leftarrow \arg\max_{\Theta_v, \Theta_s} \mathbb{E}_P[\Pr[A = \text{Executor}(\text{Perception}(S; \Theta_v), P)]], \tag{1}$$

where P denotes the program, A the answer, and S the scene. The expectation is taken over $P \sim \text{SemanticParse}(Q; \Theta_s)$.

Recall the program executor is fully differentiable w.r.t. the visual representation. We compute the gradient w.r.t. Θ_v as $\nabla_{\Theta_v} \mathbb{E}_P[D_{\text{KL}}(\text{Executor}(\text{Perception}(S; \Theta_v), P) \| A)]$. We use REINFORCE (Williams, 1992) to optimize the semantic parser Θ_s via $\nabla_{\Theta_s} = \mathbb{E}_P[r \cdot \log \Pr[P | \text{SemanticParse}(Q; \Theta_s)]]$, where the reward $r = 1$ if the answer is correct and 0 otherwise. We also use off-policy search to reduce the variance of REINFORCE, the detail of which can be found at Appendix B.

Curriculum visual concept learning. Motivated by human concept learning as in Figure 1, we employ a curriculum learning approach to help joint optimization. We heuristically split the training samples into four stages (Figure 4(A)): first, learning object-level visual concepts; second, learning relational questions; third, learning more complex questions with perception modules fixed; fourth, joint fine-tuning of all modules. Empirical experiments show that this is essential to the learning of our neuro-symbolic concept learner. We include more technical details in Appendix D.

4 Experiments

We demonstrate the following advantages of our NS-CL. First, it learns visual concepts with remarkable accuracy; second, it allows data-efficient visual reasoning on the CLEVR dataset (Johnson et al., 2017a); third, it generalizes well to new attributes, visual composition, and language domains.

We train NS-CL on 5K images ($< 10\%$ of CLEVR's 70K training images). We generate 20 questions for each image for the entire curriculum learning process.

4.1 Visual concept learning

Classification-based concept evaluation. Our model treats attributes as neural operators that maps latent object representations into an attribute-specific embedding space (Figure 3). We evaluate the concept quantization of objects in the CLEVR validation split. Table 2 shows that our model achieves near perfect classification accuracy ($\sim 99\%$) for 8 out of 9 attributes, suggesting it effectively learns generic concept representations. The result for spatial relations is relatively lower. This is because CLEVR does not have direct queries on the spatial relation between objects, so spatial relation concepts can only be learned indirectly.

Count-based concept evaluation. The SOTA methods do not provide interpretable representation on individual objects (Johnson et al., 2017a; Hudson & Manning, 2018; Mascharka et al., 2018) . To evaluate the visual concepts learned by such models, we generate a synthetic question set. The

	Color	Material	Shape	Size	Spatial Relation	Same Color	Same Material	Same Shape	Same Size
NS-CL	99.4	99.7	98.7	99.9	93.9	99.7	99.3	99.0	99.9

Table 2: We evaluate the learned visual concepts by reading out the concept classification for each object in the CLEVR validation split (e.g., classify whether each object is red). The table shows the average classification accuracy for each category of concepts. NS-CL learns visual concepts effectively and efficiently by only reading visually-grounded QA pairs.

	Visual	Overall	Color	Material	Shape	Size
IEP	Convolutional	90.6	91.0	90.0	89.9	90.6
MAC	Attentional	95.9	98.0	91.4	94.4	94.2
TbD (hres.)	Attentional	96.5	96.6	92.2	95.4	92.6
NS-CL	Object-Based	**98.7**	**99.0**	**98.7**	**98.1**	**99.1**

Table 3: We also evaluate the learned visual concepts using a diagnostic question set containing simple questions as "How many `red` objects are there?". NS-CL outperforms both convolutional and attentional baselines. The suggested object-based visual representation and symbolic reasoning approach perceives better interpretation of visual concepts.

Model	Prog. Anno.	Overall	Count	Compare Numbers	Exist	Query Attribute	Compare Attribute
Human	N/A	92.6	86.7	86.4	96.6	95.0	96.0
NMN	700K	72.1	52.5	72.7	79.3	79.0	78.0
N2NMN	700K	88.8	68.5	84.9	85.7	90.0	88.8
IEP	700K	96.9	92.7	98.7	97.1	98.1	98.9
DDRprog	700K	98.3	96.5	98.4	98.8	99.1	99.0
TbD	700K	**99.1**	**97.6**	**99.4**	**99.2**	**99.5**	**99.6**
RN	0	95.5	90.1	93.6	97.8	97.1	97.9
FiLM	0	97.6	94.5	93.8	99.2	99.2	99.0
MAC	0	98.9	97.2	**99.4**	**99.5**	99.3	**99.5**
NS-CL	0	**98.9**	**98.2**	99.0	98.8	**99.3**	99.1

Table 4: Our model outperforms all baselines using no program annotations. It also achieves comparable results with baselines trained by full program annotations such as TbD (Mascharka et al., 2018), using less than 10% of the training images and 15% of the training questions.

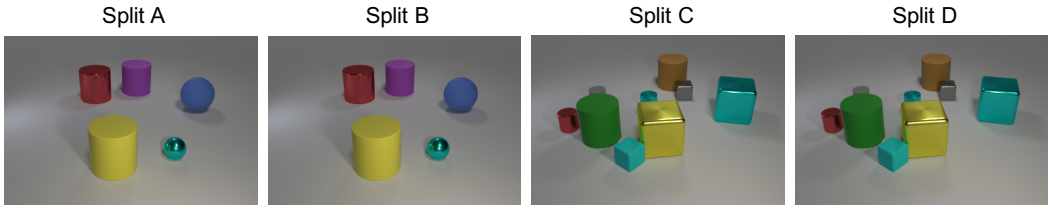
diagnostic question set contains simple questions as the following form: "How many `red` objects are there?". We evaluate the performance on all concepts appeared in the CLEVR dataset.

Table 3 summarizes the results compared with strong baselines, including methods based on convolutional features (Johnson et al., 2017b) and those based on neural attentions (Mascharka et al., 2018; Hudson & Manning, 2018). Our approach outperforms IEP by a significant margin (8%) and attentional baselines by $> 2\%$, suggesting object-based visual representations and symbolic reasoning helps interpreting visual concepts.

4.2 Data-efficient and interpretable visual reasoning

NS-CL jointly learns visual concepts, words and semantic parsing by watching images and reading paired questions and answers. It can be directly applied to VQA.

Table 4 summarizes results on the CLEVR validation split. Our model achieves the state-of-the-art performance among all baselines using zero program annotations, including MAC (Hudson & Manning, 2018) and FiLM (Perez et al., 2017). Our model achieves comparable performance with the strong baseline TbD-Nets (Mascharka et al., 2018), whose semantic parser is trained using 700K programs in CLEVR (ours need 0). The recent NS-VQA model from Yi et al. (2018) achieves better performance on CLEVR; however, their system requires annotated visual attributes and program traces during training, while our NS-CL needs no extra labels.



Q: What's the shape of the big yellow thing?

Q: What size is the cylinder that is left of the cyan thing that is in front of the big sphere?

Q: What's the shape of the big yellow thing?

Q: What size is the cylinder that is left of the cyan thing that is in front of the gray cube?

Figure 5: Samples collected from four splits in Section 4.3 for illustration. Models are trained on split A but evaluated on all splits for testing the combinatorial generalization.

Our model also recovers the underlying programs of questions accurately ($> 99.9\%$ accuracy). Our visual perception module is pre-trained on ImageNet (Deng et al., 2009). Without pre-training, the concept learning accuracies drop by 0.2% on average and the QA accuracy drops by 0.5%. Our model can also detect ambiguous or invalid programs and indicate exceptions. Please see Appendix E for more details.

4.3 Generalization to new attributes and compositions

Generalize to new visual compositions. The CLEVR-CoGenT dataset is designed to evaluate models' ability to generalize to new visual compositions. It has two splits: Split A only contains gray, blue, born and yellow cubes, but red, green, purple, and cyan cylinders; split B imposes the opposite color constraints on cubes and cylinders. If we directly learn visual concepts on split A, it overfits to classify shapes based on the color, leading to a poor generalization to split B.

Our solution is based on the idea of seeing attributes as operators. Specifically, we jointly train the concept embeddings (e.g., `Red`, `Cube`, etc.) as well as the semantic parser on split A, keeping pre-trained, frozen attribute operators. As we learn distinct representation spaces for different attributes, our model achieves an accuracy of 98.8% on split A and 98.9% on split B.

Generalize to new visual concepts. We expect the process of concept learning can takes place in an incremental manner: having learned 7 different colors, humans can learn the 8-th color incrementally and efficiently. To this end, we build a synthetic split of the CLEVR dataset to replicate the setting of incremental concept learning. Split A contains only images without any purple objects, while split B contains images with at least one purple object. We train all the models on split A first, and finetune them on 100 images from split B. We report the final QA performance on split B's validation set. All models use a pre-trained semantic parser on the full CLEVR dataset.

Our model performs a 93.9% accuracy on the QA test in Split B, outperforming the convlutional baseline IEP (Johnson et al., 2017b) and the attentional baseline TbD (Mascharka et al., 2018) by 4.6% and 6.1% respectively. The acquisition of `Color` operator brings more efficient learning of new visual concepts.

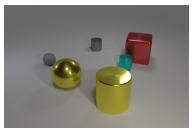
4.4 Combinatorial generalization to new scenes and questions

Having learned visual concepts on small-scale scenes (containing only few objects) and simple questions (only single-hop questions), we humans can easily generalize the knowledge to larger-scale scenes and to answer complex questions. To evaluate this, we split the CLEVR dataset into four parts: **Split A** contains only scenes with less than 6 objects, and questions whose latent programs having a depth less than 4; **Split B** contains scenes with less than 6 objects, but arbitrary questions; **Split C** contains arbitrary scenes, but restricts the program depth being less than 4; **Split D** contains arbitrary scenes and questions. Figure 5 shows some illustrative samples.

As VQA baselines are unable to count a set of objects of arbitrary size, for a fair comparison, all programs containing the "count" operation over > 6 objects are removed from the set. For methods using explicit program semantics, the semantic parser are pre-trained on the full dataset and fixed. Methods with implicit program semantics (Hudson & Manning, 2018) learn an entangled representation for perception and reasoning, and cannot trivially generalize to more complex programs. We only use the training data from the Split A and then quantify the generalization ability on other three splits. Shown in Table 5, our NS-CL leads to almost-perfect generalization to larger scenes and more complex questions, outperforming all baselines by at least 4% in QA accuracy.

Model	Program	Visual Representation	Train	Test			
				Split A	Split B	Split C	Split D
MAC	Implicit	Attentional	Split A	97.3	N/A	92.9	N/A
IEP	Explicit	Convolutional	Split A	96.1	92.1	91.5	90.9
TbD(hres.)	Explicit	Attentional	Split A	98.8	94.5	94.3	91.9
NS-CL	Explicit	Object-Based	Split A	**98.9**	**98.9**	**98.7**	**98.8**

Table 5: We test the combinatorial generalization w.r.t. the number of objects in scenes and the complexity of questions (i.e. the depth of the program trees). We makes four split of the data containing various complexities of scenes and questions. Our object-based visual representation and explicit program semantics enjoys the best (and almost-perfect) combinatorial generalization compared with strong baselines.



Caption: There is a big yellow cylinder in front of a gray object.

(a) An illustrative pair of image and caption in our synthetic dataset.

Model	Retrieval Accuracy
IEP	95.5
TbD	**97.0**
NS-CL	96.9

(b) Image-caption retrieval accuracy on a subset of data. Our model archives comparable results with VQA baselines.

Model	Retrieval Accuracy
CNN-LSTM	68.9
NS-CL	**97.0**

(c) Image-caption retrieval accuracy on the full dataset. Our model outperforms baselines and requires no extra training or finetuning of the visual perception module.

Table 6: To validate the transferrablity of the learned visual concepts, we introduce a new simple DSL for image-caption retrieval. Due to the difference between VQA and caption retrieval, VQA baselines are only able to infer the result on a partial set of data. The learned object-based visual concepts can be directly transffered into the new domain for free.

4.5 Extending to other program domain

The learned visual concepts can also be used in other domains such as image retrieval. With the visual scenes fixed, the learned visual concepts can be directly transferred into the new domain. We only need to learn the semantic parsing of natural language into the new DSL.

We build a synthetic dataset for image retrieval and adopt a DSL from scene graph–based image retrieval (Johnson et al., 2015). The dataset contains only simple captions: "There is an <object A> <relation> <object B>." (e.g., There is a box right of a cylinder). The semantic parser learns to extract corresponding visual concepts (e.g., `box`, `right`, and `cylinder`) from the sentence. The program can then be executed on the visual representation to determine if the visual scene contains such relational triples.

For simplicity, we treat retrieval as classifying whether a relational triple exists in the image. This functionality cannot be directly implemented on the CLEVR VQA program domain, because questions such as "Is there a box right of a cylinder" can be ambiguous if there exists multiple cylinders in the scene. Due to the entanglement of the visual representation with the specific DSL, baselines trained on CLEVR QA can not be directly applied to this task. For a fair comparison with them, we show the result in Table 6b on a subset of the generated image-caption pairs where the underlying programs have no ambiguity regarding the reference of object B. A separate semantic parser is trained for the VQA baselines, which translates captions into a CLEVR QA-compatible program (e.g., `Exist(Filter(Box, Relate(Right, Filter(Cylinder))).`

Table 6c compares our NS-CL against typical image-text retrieval baselines on the full image-caption dataset. Without any annotations of the sentence semantics, our model learns to parse the captions into the programs in the new DSL. It outperforms the CNN-LSTM baseline by 30%.

5 Conclusion

We presented a method that jointly learns visual concepts, words, and semantic parsing of sentences from natural supervision. The proposed framework, NS-CL, learns by looking at images and reading paired questions and answers, without any explicit supervision such as class labels for objects. Our model learns visual concepts with remarkable accuracy. Based upon the learned concepts, our model achieves good results on question answering, and more importantly, generalizes well to new visual compositions, new visual concepts, and new domain specific languages.

REFERENCES

Omri Abend, Tom Kwiatkowski, Nathaniel J Smith, Sharon Goldwater, and Mark Steedman. Bootstrapping language acquisition. *Cognition*, 164:116–143, 2017.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *NAACL-HLT*, 2016.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*, 2015.

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

Grzegorz Chrupała, Akos Kádár, and Afra Alishahi. Learning language through pictures. *arXiv preprint arXiv:1506.03694*, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. A probabilistic computational model of cross-situational word learning. *Cognitive Science*, 34(6):1017–1063, 2010.

Siddha Ganju, Olga Russakovsky, and Abhinav Gupta. What's in a question: Using visual questions as a form of supervision. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 6422–6431. IEEE, 2017.

Jon Gauthier, Roger Levy, and Joshua B Tenenbaum. Word learning and the acquisition of syntactic–semantic overhypotheses. *arXiv preprint arXiv:1805.04988*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matthew Botvinick, Demis Hassabis, and Alexander Lerchner. Scan: learning abstract hierarchical compositional visual concepts. In *ICLR*, 2018.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. In *ICLR*, 2018.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *CVPR*, 2015.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017a.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, 2017b.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

M Malinowski and M Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*, 2014.

Junhua Mao, Jiajing Xu, Kevin Jing, and Alan L Yuille. Training and Evaluating Multimodal Word Embeddings with Large-Scale Web Annotated Images. In *Proc. of NIPS*, 2016.

David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *CVPR*, 2018.

Tushar Nagarajan and Kristen Grauman. Attributes as operators: Factorizing unseen attribute-object compositions. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017.

Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017.

N Siddharth, T. B. Paige, J.W. Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. Learning disentangled representations with semi-supervised deep generative models. In *NIPS*, 2017.

Joseph Suarez, Justin Johnson, and Fei-Fei Li. Ddrprog: A clevr differentiable dynamic reasoning programmer. *arXiv:1803.11361*, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 2000.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *Proc. of ICLR*, 2016.

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

Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *CVPR*, 2017.

Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pp. 451–466. Springer, 2016.

Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *NIPS*, 2015.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *CVPR*, 2016.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Kohli Pushmeet, and Joshua B Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in neural information processing systems*, 2018.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

Supplementary Material

A CLEVR DOMAIN-SPECIFIC LANGUAGE AND IMPLEMENTATIONS

We begin the appendix with the introduction to a domain-specific language designed for the CLEVR VQA dataset (Johnson et al., 2017a).

Operation Token	Signature	Description
`Filter-Scene`	Concept $\longrightarrow$ ObjectSet	Filter out a set of objects having the attribute Concept (e.g., red) from the scene.
`Filter`	ObjectSet, Concept $\longrightarrow$ ObjectSet	Filter out a set of objects having the attribute Concept (e.g., red) from the input object set.
`Relate`	Object, Concept $\longrightarrow$ ObjectSet	Filter out a set of objects that have the relation Concept (e.g., left) with the input object.
`AERelate`	Object, Attribute $\longrightarrow$ ObjectSet	Filter out a set of objects that have the same attribute value (e.g., same color) with the input object.
`Intersection`	ObjectSet, ObjectSet $\longrightarrow$ ObjectSet	Take the intersection of two object sets.
`Union`	ObjectSet, ObjectSet $\longrightarrow$ ObjectSet	Take the union of two object sets.
`Query`	Object, Attribute $\longrightarrow$ Concept	Query the attribute (e.g., color) of the input object.
`AEQuery`	Object, Object, Attribute $\longrightarrow$ Bool	Query if two input objects have the same attribute value (e.g., same color).
`Count`	ObjectSet $\longrightarrow$ Integer	Query the number of objects in a set.
`LessThan`	ObjectSet, ObjectSet $\longrightarrow$ Bool	Query if the number of objects in the first input set is less than the one of the second set.
`GreaterThan`	ObjectSet, ObjectSet $\longrightarrow$ Bool	Query if the number of objects in the first input set is greater than the one of the second set.
`Equal`	ObjectSet, ObjectSet $\longrightarrow$ Bool	Query if the number of objects in the first input set is the same as the one of the second set.

Table 7: The domain specific language for CLEVR VQA.

We note that some function takes Object as its input instead of ObjectSet. These functions requires the uniqueness of the referral object. For example, to answer the question "What's the color of the red object?", there should be one and only one red object in the scene. During the program execution, the object sets will be implicitly casted to the single object (if valid) using a special operation called `Unique`.

We adopt a quasi-symbolic implementation of object sets. Denote the set of all objects in the scene as a set S, we represent a subset $T \subseteq S$ by assigning an real value for each object $\in S$, denoting the probability that the specific object belongs to T. We call such representations "masks" on all objects.

B SEMANTIC PARSING

We adopt a visually-grounded semantic parsing approach for the joint learning of visual concepts, word semantics and semantic parsing of sentences. We consider three types of visually-grounded words:

Object-level concepts This group includes red, blue, cube, sphere, big, small, rubber, material, etc.

Relation-level concepts This group includes front, behind, left and right.

Visual attributes This group includes color, shape, material and size.

Here we add a small note for reproducibility: in the CLEVR dataset, some object-level concepts have synonyms (e.g., box is a synonym of cube). We also consider their synonyms as independent concepts.

The goal of the language module is to learn the word semantics of concept words and the semantic parsing of sentences of different forms. Consider the sentence "What is the color of the cube right of the red matte object?". To exploit the grounded nature of natural language questions, in our DSL, we treat visual concepts and attributes such as `Color` or `Cube` in the above case as parameters of program operations, instead of part of the program token as in Johnson et al. (2017b) or Mascharka et al. (2018). We impose the restriction that all such parameters in the program must be chosen from the set of concept words appeared in the sentence.

Another pre-processing to the sentence is to group consecutive concept words into a group and treat it as a single concept. The intuition behind this grouping is, the latent programs of CLEVR questions usually contains multiple consecutive `Filter` tokens. In the program execution, we want to fuse all such `Filter`s into a single `Filter` operation that takes multiple concepts as its parameter.

Consider again the sentence "What is the color of the cube right of the red matte object?". We encode the sentence as: "What is the <attribute 1> of the <concept 1> <relational concept 1> of the <concept 2>", where each concept is marked with a special label indicating their type. The expected parsing result of this sentence is:

```
Query(<attribute 1>, Filter(<concept 1>,
      Relate(<relational concept 1>, Filter-Scene(<concept 2>) )
) ).
```

We generate the hierarchical program structure using a sequence-to-sequence-like model based on the encoder-decoder architecture. We fill in the parameters of each token using the attention on the input sequence, chosen from all concept words in the sentence. Algorithm 1 illustrates the algorithm outline of the semantic parsing.

Algorithm 1: String-to-Tree Semantic parser.

Function `parse` $(f_s, \{c_i\})$ **:**
- $program \leftarrow$ `EmptyProgram`();
- $program.op \leftarrow$ `OpDecoder`(f_s);
- **if** *program requires concept parameter* **then**
  - $program.concept \leftarrow$ `ConceptDecoder`$(f_s, \{c_i\})$;
- **for** $i = 0, 1, \cdots$ number of parameters **do**
  - $program.param[i] \leftarrow$ `parse` (`ParamDecoder`$_i(f_s, program.op)$,$\{c_i\}$);
- **return** $program$

Shown in Algorithm 1, we first map each word into a word embedding space of dimension 256. We concatenate the word embedding with a positional embedding of dimension 128. The semantic parser begins with a two-layer GRU with hidden dimension $256 * 2$ (bidirectional). The function `Parse` starts from the state encoded by the GRU, and works recursively to generate the hierarchical program layout. Parameter decoder 0 and 1 are individual GRU cells. `ConceptDecoder` performs attention using current state f_s over all encoded feature of concept words by the GRU. We select (sample) the concept word as parameters based on the attention weights.

Off-policy program search. To tackle the optimization in a non-smooth program space, we apply an off-policy program search process(Sutton et al., 2000) to assist the REINFORCE for training the semantic parser. Denote $\mathbb{P}(s)$ as the set of all valid programs in the CLEVR dataset with parameters chosen from concepts in the sentence s. We want to compute the gradient w.r.t. Θ_s, which requires summing over all possible $p \in \mathbb{P}(s)$. In REINFORCE, we approximate it via Monte Carlo sampling.

An alternative solution is to exactly compute the gradient. Note in Equation 1, only the set of programs $\mathbb{Q}(s)$ leading to correct answer will contribute to the gradient estimation. With the perception module fixed, the set $\mathbb{Q}$ can be determined by an off-policy exhaustive search. In the third stage, we search for the set $\mathbb{Q}$ offline and compute the exact gradient $\nabla\Theta_s$ during training. An intuitive explanation of the off-policy search is that, we search over all possible programs and find the ones leading to the correct answer. We use $\mathbb{Q}(s)$ as the program annotation for the question.

Spurious program suppression. However, directly using $\mathbb{Q}(s)$ as the annotations for the supervision as $\ell = \sum_{p \in \mathbb{Q}(S)} -\log \Pr(p)$ can be problematic, due to the spuriousness or ambiguity of the programs. This comes from two aspects: 1) intrinsic ambiguity: two programs are different but equivalent. For example $\texttt{color-equal-query}(\texttt{filter}(\texttt{cube}), \texttt{filter}(\texttt{sphere}))$ is equivalent to $\texttt{exist}(\texttt{filter}(\texttt{sphere}, \texttt{color-equal-relate}(\texttt{filter}(\texttt{cube}))))$. 2) extrinsic spuriousness: the program is incorrect, but may lead to correct answer under a specific scene. For example, although the correct understanding might be "the red object behinds the cube", interpret the sentence as "the red object behinds the sphere" may also locate the correct red object being referred to. To suppress such spurious programs, we change the loss function as: $\ell = \sum_{p \in \mathbb{Q}} \Pr[p] \cdot (-\log \Pr[p])$. The key observation is that, given a sufficiently large set of scenes, a program can be identified as spurious if there exists at least one scene where the program leads to a wrong answer. As the training goes, such spurious programs will get less update due to the sample importance term $\Pr[p]$ which weights the likelihood maximization term.

C PROGRAM EXECUTION

Table 8 summarizes the implementation of some representative operators.

Opeartor	Signature	Implementation
Filter-Scene	$\texttt{Concept} \to \texttt{Mask}^o$	$\texttt{Mask}_i^o = \texttt{Concept}_i$
Filter	$\texttt{Mask}, \texttt{Concept} \to \texttt{Mask}^o$	$\texttt{Mask}_i^o = \min(\texttt{Mask}_i, \texttt{Concept}_i)$
Relate	$\texttt{Mask}, \texttt{RConcept} \to \texttt{Mask}^o$	$\texttt{Mask}_i^o = \sum_j (\texttt{Mask}_j \cdot \texttt{RConcept}_{j,i})$
Intersection	$\texttt{Mask}^1, \texttt{Mask}^2 \to \texttt{Mask}^o$	$\texttt{Mask}_i^o = \min(\texttt{Mask}_i^1, \texttt{Mask}_i^2)$
Union	$\texttt{Mask}^1, \texttt{Mask}^2 \to \texttt{Mask}^o$	$\texttt{Mask}_i^o = \max(\texttt{Mask}_i^1, \texttt{Mask}_i^2)$
Unique	$\texttt{Mask} \to \texttt{Mask}^o$	$\texttt{Mask}_i^o = \mathbf{1}[i = \arg\max \texttt{Mask}]$
Query	$\texttt{Mask}, \texttt{Attr} \to \texttt{Concept}^o$	$\texttt{Concept}^o = \arg\max_{c \in \texttt{Attr}} \sum \texttt{Mask}_i \cdot c_i$
Count	$\texttt{Mask} \to \texttt{Integer}$	$\texttt{Integer} = \sum_i \texttt{Mask}_i$
Count	$\texttt{Mask} \to \texttt{Bool}$	$\texttt{Bool} = \max_i \texttt{Mask}_i$

Table 8: Our program executor takes an quasi-symbolic way to execute programs.

We implement a set of generic operations for VQA. The LHS in the signature column are the inputs while the RHS the output. Each `Mask` is a real-valued vector denoting an object set. Throughout this section, we consider a scene containing n objects.

Take `Filter` as an example, it takes the current set of objects, represented by a vector `Mask` of length n, with each value $\texttt{Mask}_i$ representing the probability that the i-th object belongs to the set and a concept word (e.g., `Cube`) as input. We compute a probability vector `Concept` that $\texttt{Concept}_i$ denotes the probability that object i is a `Cube`. The output of the filter operation is still a object set, represented by a vector $\texttt{Mask}^{(o)}$, where $\texttt{Mask}_i^{(o)} = \min(\texttt{Mask}_i, \texttt{Concept}_i)$.

Sharing a similar spirit, the `Relate` operation is implemented by computing an $n \times n$ matrix `RConcept` based on the input relational concept (e.g., `Left`), where $\texttt{RConcept}_{j,i}$ denotes the probability that object i is left of object j in the scene.

Unique is a special operation use for transforming an object set into a single object. During the inference, this operation is simply choosing the object i having the highest score: $i = \arg\max_j \texttt{Mask}_j$. During the training, this operation is implemented by taking the softmax.

To query the attribute (e.g., `Shape`) of an object, we first enumerate all concepts belongs to this attribute (e.g., `Cube`, `Sphere`, `Cylinder`), and computes the probability that the object is of each kind of shapes. During the inference, we answer the query by the most confident prediction, while during the training, we take the softmax.

D CURRICULUM LEARNING SETUP

During the whole training process, we gradually add more visual concepts and more complex question examples into the model. Summarized in Figure 4(A), in general, the whole training process is split into 4 stages. First, we only use questions from lesson 1 to let the model learn object-level visual concepts. Second, we train the model to parse simple questions (with depth ≤ 4) and to learn relational concepts. Here we freeze the neural operators and embeddings of object-level concepts. Third, we fix all perception modules. The model gets trained on the full question set (lesson 3), learning to understand questions of different complexities and various format. Finally, at the last stage, we finetune all modules.

We select questions for each lesson in the curriculum learning by their depth of the latent program layout. For eaxmple, the program "`Query(Shape, Filter-Scene(Red))`" has the depth of 2, while the program "`Query(Shape Filter(Cube, Relate(Left, Filter-Scene(Red))))`" has the depth of 4. We fuse the consecutive `Filter` operations into a single one, taking a group of concepts as input. We also fuse `Attribute-Equal-Query` and `Query`, `Less-Than` and `Count` into a same operator. We compute the depth based on the fused program. Thus, the maximum depth of all programs is 8 on the CLEVR dataset.

We now present the split of our curriculum learning lessons.

For lesson 1, we introduce only programs of depth 2. It contains three types of questions: querying an attribute of the object, querying the existence of a certain type of objects, count a certain type of objects, and querying if two objects have the same attribute (e.g., of the same color). These questions are almost about fundamental object-based visual concepts.

For lesson 2, we introduce programs of depth less than 4, containing a number of relation-typed questions such as querying the attribute of an object that is left of another object. We found that in the original CLEVR dataset, all `Relate` operations are followed by a `Filter` operation. This setting degenerates the performance of the learning of relational concepts such as `Left`. Thus, we manually add another question type: `Count(Relate(Filter-Scene()))` (e.g., "What's the number of objects that are left of the cube?") to facilitate the learning of relational concepts.

For lesson 3, we use the full set of CLEVR question set.

E ABLATION STUDY, VISUALIZATION AND DISCUSSION

Program accuracy. We also evaluate how well our model recovers the underlying programs of questions. Due to the intrinsic equivalence of different programs, we evaluate the accuracy of programs by executing them on the ground-truth annotations of objects. Invalid or ambiguous programs are also considered as incorrect. Our semantic parser archives $> 99.9\%$ QA accuracy on the validation split.

Impacts of the ImageNet pre-training. The only extra supervision of the visual perception module comes from the pre-training of the perception modules on ImageNet Deng et al. (2009). To quantify the influence of this pre-training, we conduct ablation experiments where we randomly initialize the perception module following He et al. (2015). The classification accuracies of the learned concepts almost remain the same except for `Shape`. The classification accuracy of `Shape` drops from 98.7 to 97.5 on the validation set while the overall QA accuracy on the CLEVR dataset drops to 98.2. We speculate that large-scale image recognition dataset can provide prior knowledge of shape.

Interpretability and robustness. Another appealing benefit is that our reasoning model enjoys full interpretability. As a side product, our system is able to detect ambiguity or invalidity of the programs and throw out exceptions. As an example, shown in 6, the question "What's the color of the cylinder?" can be ambiguous if there are multiple cylinders or even invalid if there is no cylinder.

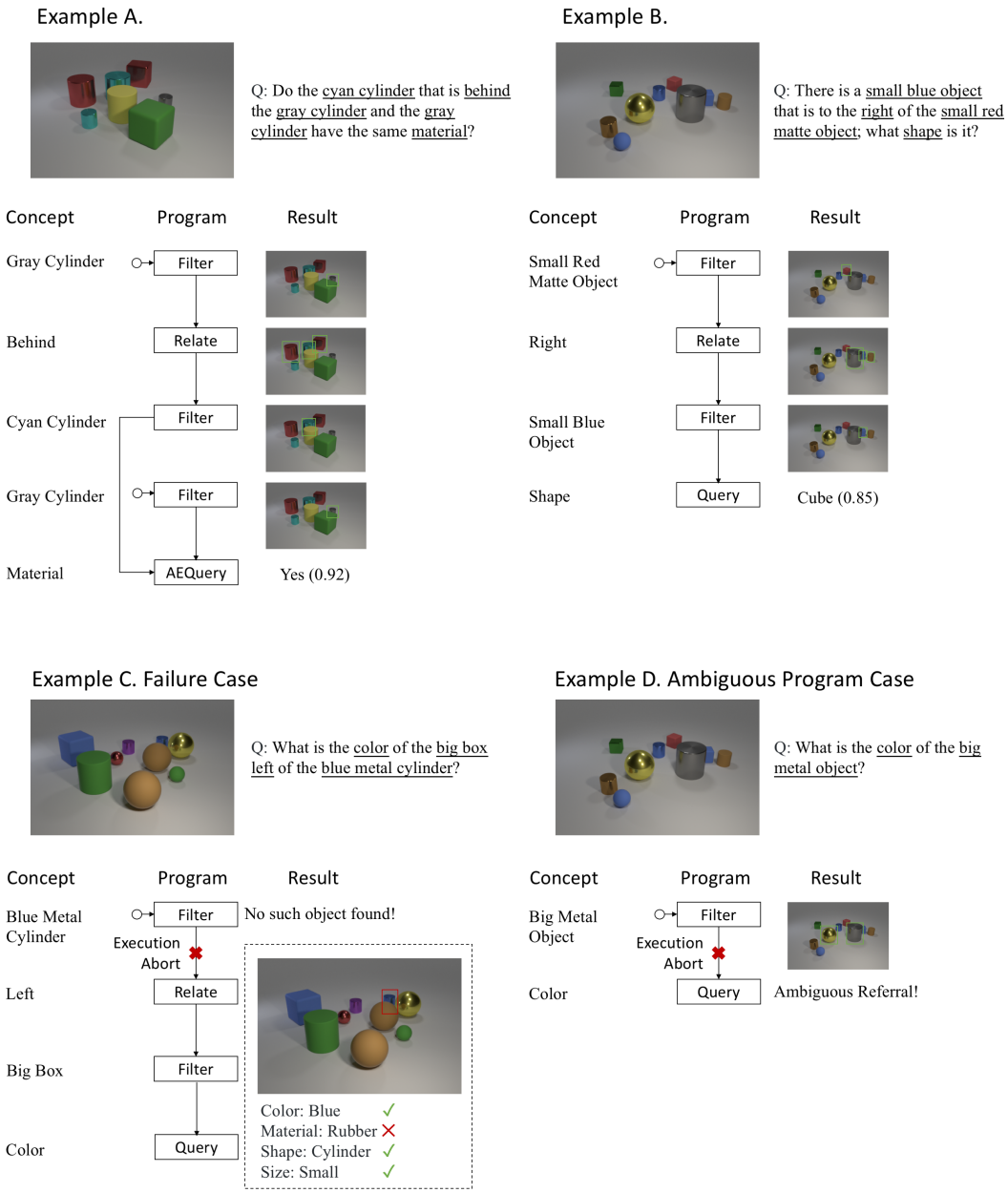


Figure 6: Visualization of the execution trace generated by our Neuro-Symbolic Concept Learner on the CLEVR dataset. Example A and B are successful executions that generates correct answers. In example C, the execution aborts at the first operator. To inspect the reason why the execution engine fails to find the corresponding object, we can read out the visual representation of the object, and locate the error source as the misclassification of the object material. Example D shows how our symbolic execution engine can detect invalid or ambiguous programs during the execution by performing sanity checks.