Neuro-Symbolic Visual Dialog

Adnen Abdessaied[*], **Mihai Bâce**, **Andreas Bulling**
Institute for Visualization and Interactive Systems (VIS)
University of Stuttgart, Germany
adnen.abdessaied@vis.uni-stuttgart.de
mihai.bace@vis.uni-stuttgart.de
andreas.bulling@vis.uni-stuttgart.de

Abstract

We propose Neuro-Symbolic Visual Dialog (NSVD)[1] —the first method to combine deep learning and symbolic program execution for multi-round visually-grounded reasoning. NSVD significantly outperforms existing purely-connectionist methods on two key challenges inherent to visual dialog: long-distance co-reference resolution as well as vanishing question-answering performance. We demonstrate the latter by proposing a more realistic and stricter evaluation scheme in which we use *predicted* answers for the full dialog history when calculating accuracy. We describe two variants of our model and show that using this new scheme, our best model achieves an accuracy of 99.72% on CLEVR-Dialog —a relative improvement of more than 10% over the state of the art —while only requiring a fraction of training data. Moreover, we demonstrate that our neuro-symbolic models have a higher mean first failure round, are more robust against incomplete dialog histories, and generalise better not only to dialogs that are up to three times longer than those seen during training but also to unseen question types and scenes.

1 Introduction

Modelled after human-human communication, visual dialog involves reasoning about a visual scene through multiple question-answering rounds in natural language (Das et al., 2019). Its multi-round nature gives rise to one of its unresolved key challenges: co-reference resolution (Kottur et al., 2018; Das et al., 2019). That is, as dialogs unfold over time, questions tend to include more and more pronouns, such as "it", "that", and "those" that have to be resolved to the appropriate previously-mentioned entities in the scene. Co-reference resolution is profoundly challenging (Das et al., 2019; Hu et al., 2017), even for models specifically designed for this task (Kottur et al., 2018). Existing models follow a purely connectionist approach and suffer from several limitations: first, they require large amounts of training data, which is prohibitive for most settings. Second, these models are not explainable, making it difficult to troubleshoot their logic when co-references are incorrectly resolved. Finally, current models lack generalisability, in particular for real-world dialogs that include incomplete or inaccurate dialog histories, longer dialogs than those seen during training, or unseen question types. While neuro-symbolic hybrid models have proven effective as a more robust, explainable, and data-efficient alternative, e.g. for VQA (Yi et al., 2018), video QA (Yi et al., 2020), or commonsense reasoning (Arabshahi et al., 2021), they have not yet been explored for visual dialog.

We fill this gap by proposing Neuro-Symbolic Visual Dialog (NSVD) —the first neuro-symbolic method geared towards visual dialog. Our method combines three novel contributions to disentangle vision and language understanding from reasoning: First, it introduces two different program generators: a caption and a question program generator, the former of which induces a program from the caption to initialise the knowledge base of the executor at the beginning of each dialog. Second, a question program generator that predicts a program in each round using not only the current question but also the dialog history. We describe two variants of this generator: one that uses a question encoder to concatenate the caption and question-answer pairs of previous rounds to encode the dialog history, as well as one that stacks them. Third, a symbolic executor with a dynamic knowledge base keeps track of all entities mentioned in the dialog.

NSVD also addresses another limitation of existing models that was "hidden" by the dominant evaluation scheme used so far: vanishing question-answering performance over the course of the dia-

[*]Corresponding Author.
[1]Project page: https://perceptualui.org/publications/abdessaied22_coling/

log. Adopted from VQA (Antol et al., 2015), the dominant scheme assumes that the model has full access to the dialog history, in particular all *ground truth* answers. We argue that this assumption is overly optimistic and overestimates real-world performance on the visual dialog task. We instead propose a more realistic and stricter evaluation scheme in which prediction in the current round is conditioned on previous *predicted* answers. This scheme better represents real-world dialogs in which communication partners rarely know whether their previous answers were correct or not.

Through extensive experiments on CLEVR-Dialog (Kottur et al., 2019), we show that our models are significantly better at resolving co-references and at maintaining performance over many rounds. Using our stricter evaluation scheme, we still achieve an accuracy of 99.72% while requiring only a fraction of the training data. Our results further suggest that NSVD has a higher mean First Failure Round, is more robust to incomplete dialog histories, and generalises better to dialogs that are up to three times longer than those seen during training as well as to unseen question types and scenes. The contributions of our work are threefold: (1) We introduce the first neuro-symbolic visual dialog model that is more robust again incomplete histories, is significantly better at resolving co-references and at maintaining performance over more rounds on CLEVR-Dialog. (2) We contribute a new Domain Specific Language (DSL) for CLEVR-Dialog that we augment with ground truth caption and question programs. (3) We unveil a fundamental limitation of the dominant evaluation scheme for visual dialog models and propose a more realistic and stricter alternative that better represents real-world dialogs.

2 Related Work

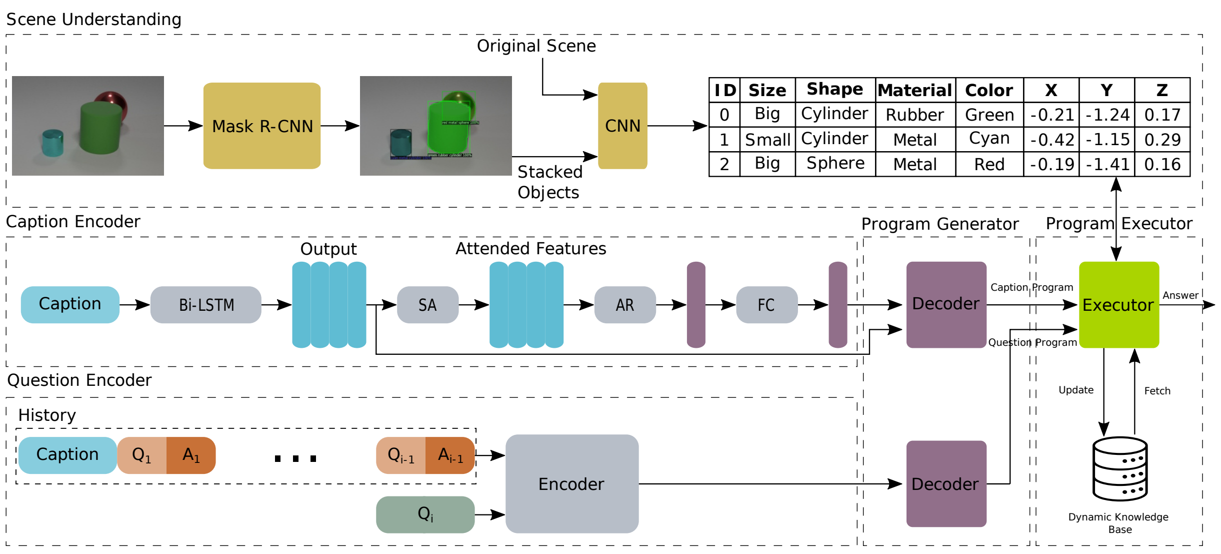
Neuro-Symbolic Models and Reasoning. Many works have used end-to-end connectionist models on the CLEVR dataset (Johnson et al., 2017a) with varying degrees of success (Johnson et al., 2017b; Hu et al., 2017; Perez et al., 2018; Hudson and Manning, 2018). NS-VQA (Yi et al., 2018) was one of the first neuro-symbolic models on CLEVR, achieving a near-perfect test accuracy. Mao et al. proposed NS-CL, a neuro-symbolic VQA network that, in contrast to NS-VQA, learned simply by looking at images and reading question-answer pairs. In parallel, other works explored neuro-symbolic models for mono-modal conversational settings (Williams et al., 2017; Suhr et al., 2018; Arabshahi et al., 2021). Recently, Andreas et al. introduced a method that represents dialog states as a dataflow graph to better deal with co-references. Although a number of works have demonstrated the significant potential of neuro-symbolic methods for mono-modal task-oriented dialog settings or tasks at the intersection of computer vision and natural language processing, we are the first to use them for the multi-modal visual dialog task.

Visual Dialog. Das et al. introduced early visual dialog models that used an encoder-decoder approach to rank a set of possible answers. Others explored explicit reasoning based on the dialog structure (Zheng et al., 2019; Niu et al., 2019; Gan et al., 2019). However, these models focused mainly on the real-world VisDial dataset (Das et al., 2019). Although popular, this dataset is not well-suited to study a key challenge of visual dialog, i.e. co-reference resolution, because it lacks complete annotation of all images and dialogs. Several works have focused on co-reference resolution in videos (Ramanathan et al., 2014; Rohrbach et al., 2017) and 3D data (Kong et al., 2014). Kottur et al. introduced CLEVR-Dialog – a fully-annotated diagnostic dataset for multi-round visual reasoning with a grammar grounded in the CLEVR scene graphs. More recently, Shah et al. introduced models that build on the Memory, Attention, and Composition (MAC) network (Hudson and Manning, 2018). Although their models achieved promising results on CLEVR-Dialog, they are computationally and memory inefficient (Shah et al., 2020). While the neuro-symbolic approach has significant potential to address these shortcomings, it has not yet been explored for visual dialog.

3 Method

Our method consists of four components (see Figure 1): a scene understanding method, a program generator with caption and question encoders and a decoder, and a symbolic program executor with a dynamic knowledge base.

Scene Understanding. We used a pre-trained Mask R-CNN (He et al., 2017) to predict segmentation masks and attributes (colour, shape, material, size) for each entity in the visual scene. We then learned the 3D coordinates of each segment paired with the original image using a ResNet-34 (He



ID	Size	Shape	Material	Color	X	Y	Z
0	Big	Cylinder	Rubber	Green	-0.21	-1.24	0.17
1	Small	Cylinder	Metal	Cyan	-0.42	-1.15	0.29
2	Big	Sphere	Metal	Red	-0.19	-1.41	0.16

Figure 1: Overview of our method: first, a structured scene representation is created. Then, a caption program is induced and run by our executor to initialise its knowledge base. At each subsequent round, the question and the history are used to induce a program that answers the question and updates the dynamic knowledge base.

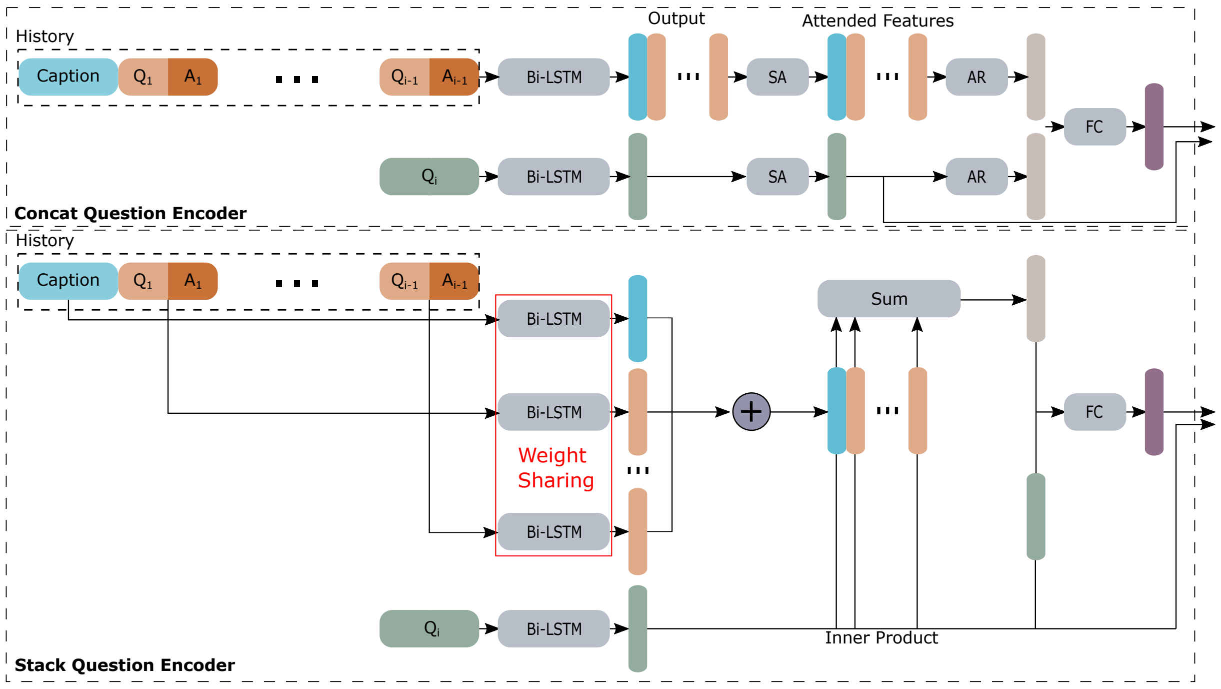


Figure 2: **Top:** The concatenative encoder ("concat") takes the question and the concatenated previous rounds as input and outputs a latent vector and a question representation to the decoder. **Bottom:** The stacking encoder ("stack") takes the question and the previous rounds as input and attends to each round separately. Then, it outputs a latent vector and a question representation to the decoder.

et al., 2016). The Mask R-CNN was pre-trained on the CLEVR-mini dataset (Yi et al., 2018).

DSL for CLEVR-Dialog. Because CLEVR-Dialog implements its own grammar and vocabulary, we designed a novel domain-specific language (DSL) for it by implementing a collection of deterministic functions in Python that our symbolic executor can run over a CLEVR scene. In previous works (Johnson et al., 2017b; Yi et al., 2018; Mao et al., 2019), these functional modules shared the same input/output interface and were arranged one after another to predict the answer. Instead, we followed a stricter approach by executing only one function that expects a *different* number of input arguments to answer a particular question. The full

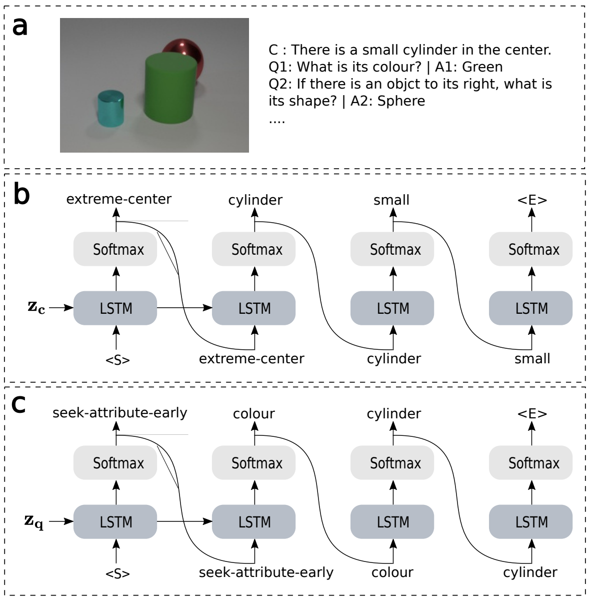


Figure 3: **a:** An example of a CLEVR-Dialog instance with a caption and two rounds. **b:** Inference sample of the caption program generation. **c:** Inference sample of the question program generation on the first round.

list of our functions, their arguments, and expected output can be found in Appendix A.1.

Program Generation. Semantic parsing methods were shown to be effective for mapping sentences to logical forms via a knowledge base or a program (Guu et al., 2017; Liang et al., 2013; Suhr et al., 2018). We adopted this approach and used a sequence-to-sequence model with an encoder-decoder structure to generate the programs. Although we used the same decoder, we propose two types of encoders that differ in the way they encode the dialog history (Figure 1).

Caption Encoder. The caption encoder first embeds the caption tokens into a 300-dim. space to give $\{\mathbf{w_{c_j}}\}_{j=1}^{n_c}$ that are then fed into a bi-directional LSTM (Hochreiter and Schmidhuber, 1997). The self-attended (Vaswani et al., 2017) LSTM outputs $\mathbf{C} = [\mathbf{c_1}, ..., \mathbf{c_{n_c}}]$ are reduced following:

$$\mathbf{a} = \mathrm{softmax}(\mathrm{MLP}(\mathbf{C}))$$

$$\mathbf{\bar{z}_c} = \sum_{i=1}^{n_c} a_i \mathbf{c_i}.$$

Finally, the latent vector $\mathbf{z_c} = \mathrm{Linear}(\mathbf{\bar{z}_c})$ and the LSTM output $\mathbf{C}$ are passed to the decoder.

Question Encoders. To generate the question program at the current round i, we use not only the question Q_i but also the history $H_i = [C, Q_1, A_1, .., Q_{i-1}, A_{i-1}]$ of previous question-answer pairs including the caption. We propose two different encoders based on how the question interacts with the history.

– Concat Encoder: The concat encoder is similar in structure to the caption encoder. The caption and the question-answer pairs of previous rounds are concatenated to form the history H_i. Then, the tokens of the current question Q_i and history H_i are embedded into a 300-dim. space to give $\{\mathbf{w_{q_j}^{(i)}}\}_{j=1}^{n_q}$ and $\{\mathbf{w_{h_j}^{(i)}}\}_{j=1}^{n_h}$, respectively, which are then processed by two separate bi-directional LSTMs (Figure 2). The reduced attended question and history features $\mathbf{\bar{z}_q}$ and $\mathbf{\bar{z}_h}$ are obtained in a similar manner to $\mathbf{\bar{z}_c}$. Finally, $\mathbf{\bar{z}_q}$ and $\mathbf{\bar{z}_h}$ are concatenated and linearly transformed to produce the question latent vector $\mathbf{z_q} = \mathrm{Linear}([\mathbf{\bar{z}_q}, \mathbf{\bar{z}_h}])$. Similarly, $\mathbf{z_q}$ and the question LSTM output $\mathbf{Q_i} = [\mathbf{q_1}, ..., \mathbf{q_{n_q}}]$ are passed to the decoder.

– Stack Encoder: The approach of concatenating the question-answer pairs to form the history suffers from two main drawbacks. First, since the history is processed by an LSTM, its encoding becomes inefficient, in particular for later rounds as the LSTM tends to forget crucial information that was mentioned in the first rounds. Second, this approach does not scale well for longer dialogs as it becomes computationally and memory demanding especially because we use self-attention (Vaswani et al., 2017) at a later stage in the concat encoder. To overcome these limitations, we introduce the stack encoder that separately encodes each question-answer pair in order to equally preserve the information from all previous rounds. The question Q_i and each previous round $R_{j<i}$, including the caption, are embedded then processed by separate bidirectional LSTMs (Figure 2). The last hidden states are used as feature representations of the question and previous rounds, i.e.

$$\mathbf{q} = [\overrightarrow{\mathbf{h}_{Q_i}}, \overleftarrow{\mathbf{h}_{Q_i}}] \text{ and } \mathbf{r_{j<i}} = [\overrightarrow{\mathbf{h}_{R_{j<i}}}, \overleftarrow{\mathbf{h}_{R_{j<i}}}],$$

where $\overrightarrow{\mathbf{h}_{(.)}}$ and $\overleftarrow{\mathbf{h}_{(.)}}$ are the bi-directional LSTM's last forward and backward hidden states, respectively. $\mathbf{\bar{z}_h}$ is obtained by applying an inner-product attention between the question and history features:

$$\mathbf{a} = \mathrm{softmax}(\mathbf{q^T H}),$$

$$\mathbf{H} = [\mathbf{r_0}, ..., \mathbf{r_{i-1}}].$$

Finally, $\mathbf{q}$ and $\mathbf{\bar{z}_h} = \sum_{j=1}^{i-1} a_j \mathbf{r_j}$ are concatenated and linearly transformed to produce the latent question vector $\mathbf{z_q} = \mathrm{Linear}([\mathbf{q}, \mathbf{\bar{z}_h}])$. Similarly, $\mathbf{z_q}$

and the question LSTM output $\mathbf{Q} = [\mathbf{q_1}, ..., \mathbf{q_{n_q}}]$ are passed to the decoder.

Decoder. We use the same decoder architecture to generate all the caption as well as the question programs. First, the ground truth program sequence Y_i of the i-th dialog round is embedded into a 300-dim. space to give $\{\mathbf{w^{(i)}_{y_j}}\}_{j=1}^{n_y}$ which are then processed by a simple LSTM whose hidden states are initialised by the encoder latent vector, i.e. $\mathbf{z_c}$ or $\mathbf{z_q}$. The output $\mathbf{P}$ of the LSTM is used with the encoder output, i.e. $\mathbf{C}$ or $\mathbf{Q}$, to generate a context vector $\mathbf{\Delta}$ following:

$$\mathbf{A} = \text{softmax}(\mathbf{Q^T P}),$$
$$\mathbf{\Delta} = \mathbf{A^T Q}.$$

Finally, the context vector $\mathbf{\Delta}$ is concatenated with the program output and the result is mapped to the program vocabulary dimension followed by a softmax function to obtain a distribution for the current program token y_j, i.e.

$$p(y_j|Y_{[1:j-1]}; Q_i, H_i,) \sim \text{softmax}(\text{Linear}(\tanh([\mathbf{P}, \mathbf{\Delta}]))),$$

where $Y_{[1:j-1]}$ is the sequence of previous ground truth program tokens. For training, we follow the teacher forcing strategy by Williams and Zipser. For inference, we first start with the `<S>` and sequentially generate the next program token until we reach the end token `<E>`. Figure 3 illustrates an example of the CLEVR-Dialog dataset alongside the generated programs, i.e. the program `extreme-centre(cylinder, small)` was generated from the caption "*There is a small cylinder in the center*" to initialize our executor and its knowledge base and the program `seek-attribute-early(colour, cylinder)` was generated to answer the first question of the dialog "*What is its colour?*". Our full DSL grammar and further concrete examples can be found in the Appendices A.1 and A.8, respectively.

Executor. We add a dynamic knowledge base to the symbolic executor to keep track of the previously-mentioned entities in the dialog. It is initialised at the beginning of each dialog by executing the caption program. For instance, by executing the caption program `extreme-centre(cylinder, small)`, the executor searches for the centre entity satisfying the function's arguments and stores it in the knowledge base under the handle `small-cylinder`. The executor interacts with its knowledge base via two main operations:

`fetch`. The `fetch`-operation is performed when executing a function that requires co-reference resolution. Given a set of attributes, the executor fetches the appropriate entity in the knowledge base by searching the stored handles. For example `seek-attribute-early(colour, cylinder)` first searches the previously stored handles and fetches the corresponding entity (in our example that is the cylinder mentioned in the caption with the handle `small-cylinder`) and then queries its colour to answer the question.

`update`. The `update`-operation is performed after each question function. We differentiate between four update types:

1. *Handle update:* If a fetched entity is referenced by a new attribute, its handle in the knowledge base should be updated accordingly. If the colour of the previous cylinder is red, then its handle changes from `small-cylinder` to `small-cylinder-red`.

2. *Conversation subject update:* If the question program addresses a new entity, the latter becomes the new conversation subject. In our example, the conversation subject is still the small red cylinder. However, the question program `exist-obj-exclude-early(colour, small, cylinder)` searches for other potential entities that share the same colour as the previous small cylinder. If there is one, it becomes the new conversation subject.

3. *Seen entities update:* Each time a new entity is addressed, the executor saves it in its knowledge base together with the appropriate handle.

4. *Groups update:* Some questions refer to a group of entities, e.g. `count-attribute(red)` counts all red entities in the scene. These sets might be relevant for subsequent questions, e.g. `count-attribute-group(large)` counts how many of the previous red entities are large.

4 Experiments

We modified the publicly available code for CLEVR-Dialog (Kottur et al., 2019) to generate datasets with ground truth caption and question programs required to train our program generators. Similar to (Kottur et al., 2019), we used the 70,000 training and $15,000$ validation CLEVR images as our visual groundings when generating the dialogs. We left out the CLEVR test images because they lack ground truth scene annotations. For each image, we generated five dialogs each consisting of $L = 10$ question-answer rounds as in (Kottur et al., 2019). We used $1,000$ training images and their corresponding dialogs to create a validation set and tested our models and the baselines on the dialogs generated using the CLEVR validation images.

Performance Evaluation. Alongside the answer accuracy, the First Failure Round (FFR), i.e. the number of dialog rounds necessary for a model to make its first mistake, is commonly used to evaluate visual dialog models. Although popular, this metric has one major limitation as it only allows us to compare the performance of models across datasets with the same dialog length but not across datasets with different ones. Thus, we propose the *Normalised First Failure Round* (NFFR) $\in [0, 1]$ as an improvement and use it alongside the answer accuracy to assess the performance of all models. See Appendix A.2 for more details.

History during Evaluation. One key limitation in the way that visual dialog models are currently evaluated is the use of *ground truth* answers when calculating the correctness of an answer in any given round (Kottur et al., 2018; Das et al., 2019; Shah et al., 2020). The problem of this approach is that it leads to overly-optimistic performance that do not reflect the true capabilities of the models in real-world scenarios: in real-world dialogs, full information on which previous answers were correct or not is typically not available. We instead propose to condition the generation of the current answer on all previous *predicted* answers. This new evaluation scheme is geared to the visual dialog task, better represents real-world use, and is stricter. We call these evaluation schemes “Hist. + GT” and “Hist. + Pred.”, respectively.

5 Results

Visual Dialog Performance. After validating our implemented logic (see Appendix A.4), we compared the performance of our models with the visual dialog MAC networks introduced by Shah et al. (2020). We limited our comparison to their top three performing models given that these outperformed the previous state of the art (Kottur et al., 2018) by 30% in accuracy. Furthermore, we compared our models to the Hybrid Code Networks (HCN) (Williams et al., 2017) that also operate on symbolic dialog state representation but follow a different approach to parse programs than our generative one. They represent programs as templates in an action space and select the one with the highest probability during inference. This action space might become intractable if the DSL has many functions and arguments.

Model	Hist. + GT		Hist. + Pred.	
	Acc.	NFFR ↑	Acc.	NFFR ↑
MAC-CQ	97.34‡	0.92	41.10	0.15
+ CAA	97.87‡	0.94	89.39‡	0.75
+ MTM	97.58‡	0.92	70.39‡	0.46
HCN	75.88	0.34	74.42‡	0.32
NSVD-concat	99.59‡	0.98	99.59‡	0.98
NSVD-stack	**99.72‡**	**0.99**	**99.72‡**	**0.99**

Table 1: Performance comparison of our models with the state of the art on CLEVR-Dialog *test*. Results are shown for both “Hist. + GT” and “Hist. + Pred.” Our proposed models are highlighted in grey; best performance is in bold. ‡ represents $p < 0.00001$ compared to the second best score in the respective column.

Table 1 shows the performance of our models and the baselines on the test split using both evaluation schemes (“Hist. + GT” and “Hist. + Pred.”). As the table shows, our models achieve new state-of-the-art performance with *NSVD-stack* topping with an overall accuracy of 99.72% and a NFFR $= 0.99$. The high NFFR demonstrates our models’ ability to answer correctly across all rounds of the dialogs with only few failures in between. More fine-grained evaluations (e.g. on individual rounds, question categories and types) are available in the Appendix A.5. While Table 1 shows results obtained when training on the entire dataset, our method achieves the same performance when trained on only 20% of the data, while the performance of other methods deteriorate significantly with less data as shown in Appendix A.6.

History Length vs Co-reference Distance. CLEVR-Dialog provides co-reference distances for each question, i.e. the number of rounds between the current and earlier mention of an entity in a question. A co-reference distance of 1 means that the co-referent was mentioned in the previous question while a co-reference distance of 10 means that the question at round 10 refers to an entity in the caption. “All” and “None” mean that the question either depends on all previous rounds or is

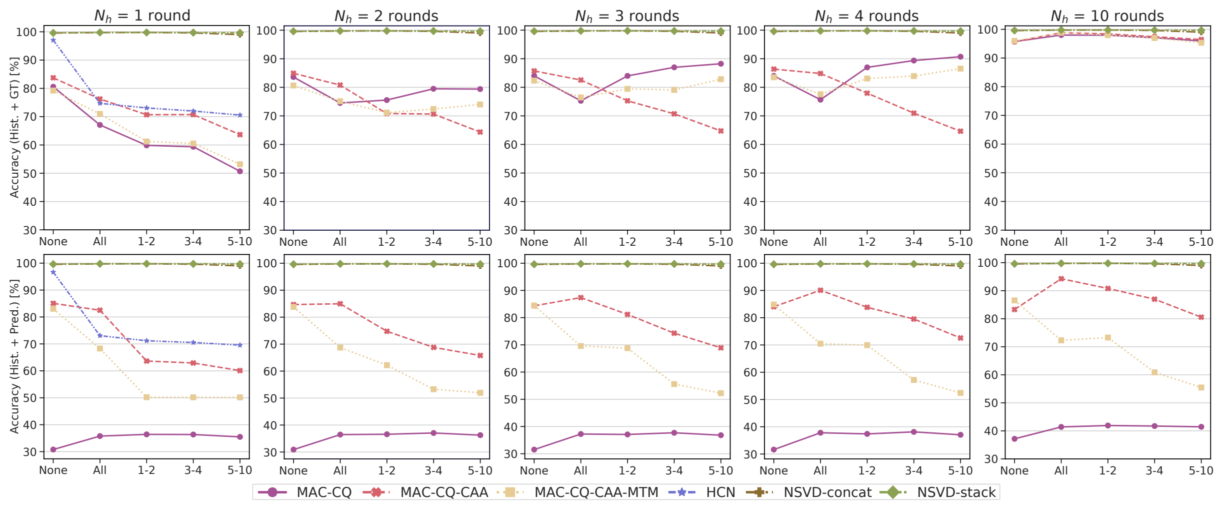


Figure 4: Robustness for different co-reference distance bins and varying number of rounds in the history. All models were trained with full histories. Independent of the evaluation scheme, the performance of our models is only slightly affected by incomplete histories across all bins. In contrast, performance of the baselines deteriorates when the full history is not available, especially for questions with large co-reference distances.

stand-alone, i.e. it does not depend on the history.

To assess performance with respect to the co-reference distance, we evaluated accuracy on different co-reference distance bins. In our evaluations, we further limited the histories to the last N_h question-answer rounds to assess the robustness of the models to incomplete dialog histories. All of the models were trained with histories containing all previous rounds except for HCN (Williams et al., 2017) that only uses the last round.

Our models consistently achieve a performance of over 99% across all co-reference distance bins, independent of the evaluation scheme (Figure 4). Furthermore, their performance is only slightly affected by incomplete dialog histories. Contrarily, performance of all connectionist baselines deteriorates quickly without access to the complete history. This deterioration is more conspicuous when the "Hist. + Pred." evaluation scheme is used (second row of Figure 4). However, their performance is consistent with the difficulty levels of the co-reference distance bins, i.e. the accuracy decreases with increasing co-reference distance. In contrast, this behaviour is not reflected by the popular evaluation approach "Hist. + GT" (first row, middle three plots of Figure 4). The most likely reason for this is that, as is currently common practice when evaluating visual dialog models, the ground truth answers of all previous rounds are used for prediction.

Generalisation to Unseen Scenes and Attributes. In previous experiments, our training and validation sets had similar distributions both in the number of objects (between three and 10) as well their sizes, shapes, colours, and materials. To further test generalisability, we created a new training set consisting of 1500 images in which we restricted the type of objects to small, rubber cubes and spheres with the colours grey, red, or blue. We kept the number of objects in this dataset between three and 10. For testing, we generated three datasets consisting of 1000 images each in which we allowed all CLEVR object classes (cubes, spheres and cylinders) and materials (rubber, metal) to appear. However, we excluded the training colours and increased the number of objects $N_{objects}$ in each one to 10, 15, and 20, respectively. Finally, we generated three fine-tuning datasets containing 1500 images each in a similar way to the testing ones. Figure 5 illustrates some examples of our new images. As in (Kottur et al., 2019), all dialogs had a length of 10 rounds.

We can see that the purely-connectionist models outperform the neuro-symbolic ones without fine-tuning in all scene complexities (Table 2). This outcome is expected since these models rely on a Mask-RCNN to understand the scenes. By increasing their complexities, i.e. more objects and attributes, the Mask-RCNN fails to accurately reconstruct these scenes which is reflected by the poor test accuracies. However, after fine-tuning, neuro-symbolic models are the best performing with our best model *NSVD-stack* scoring 99.33%, 70.62%, and 64.95% on the test datasets with 10,

	Model	$N_{objects}=10$				$N_{objects}=15$				$N_{objects}=20$			
		Without FT		**After FT**		**Without FT**		**After FT**		**Without FT**		**After FT**	
		Acc.	**NFFR ↑**	**Acc.**	**NFFR ↑**	**Acc.**	**NFFR ↑**	**Acc.**	**NFFR ↑**	**Acc.**	**NFFR ↑**	**Acc.**	**NFFR ↑**
Hist. + GT	MAC-CQ	$38.52^\dagger$	0.14	53.49^*	0.21	$\mathbf{36.87}^*$	**0.13**	48.89^*	0.19	$\mathbf{36.12}^*$	**0.13**	47.44^*	0.18
	+ CAA	$37.72^\ddagger$	0.14	$53.35^\dagger$	0.21	$36.82^\dagger$	0.13	$48.67^\dagger$	0.18	$35.52^\ddagger$	0.13	$47.43^\dagger$	0.18
	+ MTM	$\mathbf{38.59}^*$	**0.14**	52.62	0.21	$36.22^\ddagger$	0.13	47.41	0.17	36.01^*	0.13	46.54	0.17
	HCN	19.59	0.11	$73.07^\ddagger$	0.30	14.42	0.11	$56.65^\ddagger$	0.22	12.33	0.11	$53.14^\ddagger$	0.19
	NSVD-concat	25.05^*	0.12	$99.32^\ddagger$	0.97	18.51^*	0.11	$70.59^\ddagger$	0.44	$15.67^\ddagger$	0.11	$64.82^\ddagger$	0.38
	NSVD-stack	$24.95^\ddagger$	0.12	$\mathbf{99.33}^*$	**0.98**	$18.45^\ddagger$	0.11	$\mathbf{70.62}^*$	**0.44**	15.65^*	0.11	$\mathbf{64.95}^*$	**0.38**
Hist. + Pred.	MAC-CQ	$\mathbf{37.74}^*$	**0.14**	$52.26^\dagger$	0.21	36.32^*	0.12	47.58^*	0.18	$35.36^\ddagger$	0.13	$46.30^\dagger$	0.18
	+ CAA	$37.70^\ddagger$	0.13	$51.36^\dagger$	0.21	$\mathbf{36.76}^*$	0.13	$47.08^\ddagger$	0.18	$\mathbf{35.57}^*$	**0.13**	$45.56^\ddagger$	0.17
	+ MTM	$36.29^\ddagger$	0.14	50.58	0.20	$35.62^\ddagger$	**0.14**	45.67	0.17	$33.98^\ddagger$	0.13	44.02	0.17
	HCN	19.50	0.11	$71.55^\ddagger$	0.29	14.40	0.11	$55.55^\ddagger$	0.21	12.26	0.11	$51.95^\ddagger$	0.19
	NSVD-concat	25.05^*	0.12	$99.32^\ddagger$	0.97	18.51^*	0.11	$70.59^\ddagger$	0.44	$15.67^\ddagger$	0.11	$64.82^\ddagger$	0.38
	NSVD-stack	$24.95^\ddagger$	0.12	$\mathbf{99.33}^*$	**0.98**	$18.45^\ddagger$	0.11	$\mathbf{70.62}^*$	**0.44**	15.65^*	0.11	$\mathbf{64.95}^*$	**0.38**

Table 2: Results when training on simple scenes and testing on more complex ones. Best results in bold. ‡, †, and $*$ represent $p < 0.00001$, $p < 0.05$ and $p \geq 0.05$ compared to the second best score in each column, respectively.

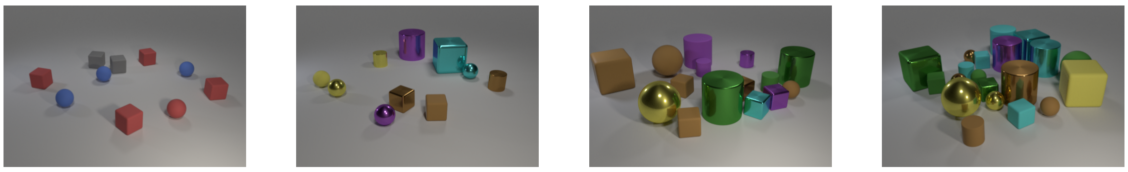


Figure 5: **From left to right:** Samples of a training image, fine-tuning image with 10 objects, fine-tuning image with 15 objects, and fine-tuning image with 20 objects.

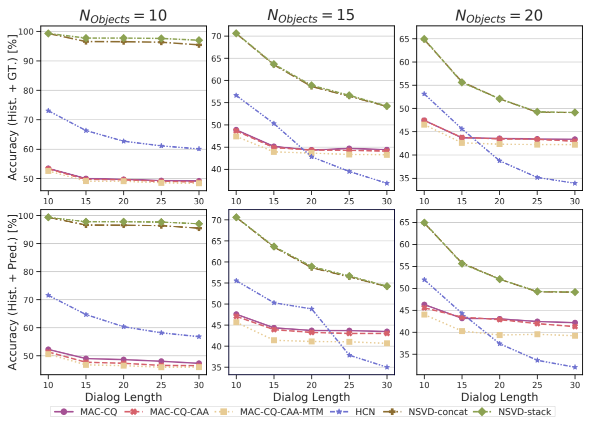


Figure 6: Answer accuracy for different dialog lengths and scene complexities. Our models generalise better to longer dialogs *without* the need for any fine-tuning.

15, and 20 objects, respectively.

Generalisation to Longer Dialogs. Typically, evaluation of visual dialog models is limited to dialogs having the same length as the training data, i.e. $L = 10$ (Kottur et al., 2018; Das et al., 2019; Shah et al., 2020). In order to assess the generalisation capabilities of our models to longer dialogs, we used the testing images of the previous experiment to generate three dialog datasets with increasing numbers of rounds, i.e. $L = 15$, 20, 25, and 30, respectively. That is, our testing datasets for this experiment not only contain dialogs that are up to three times the length of the training dialogs but also visual scenarios never seen during training. Finally, we evaluated the best fine-tuned models of the previous experiment on this data *without* fine-tuning them again on longer dialogs. Figure 6 shows that our models generalise better across all datasets for both evaluation schemes. As expected, the performance of all models decreases with longer dialogs and more complex scenes. However, our models suffer less and still significantly outperform all baselines in all test scenarios. Our best model *NSVD-stack* achieves an overall answer accuracy of 97.02%, 54.25%, and 49.14% on the longest dialogs ($L = 30$) that are created from scenes with 10, 15, and 20 objects, respectively.

Generalisation to Unseen Questions Types. Similar to prior works (Johnson et al., 2017a; Yi et al., 2018; Mao et al., 2019), we addressed the generalisability to new scenes and object combinations in our previous experiments. However, generalisability to unseen questions remains unexplored. To address this, we created two splits (AA and BB) on CLEVR-Dialog as follows: we first split the CLEVR validation images into two disjoint halves A and B. We then split the question types into split A and split B. We randomly assigned half of the question types in each category to split A and the other half to split B to prevent biasing either one to a particular question category. For each image

	Model	Without FT		FT on split BB	
		Acc.	**NFFR ↑**	**Acc.**	**NFFR ↑**
Hist. + GT	MAC-CQ	$36.12^{‡}$	0.14	$40.33^{†}$	0.16
	+ CAA	$35.09^{†}$	0.14	40.36^{*}	0.16
	+ MTM	34.73	0.15	35.09	0.13
	HCN	$47.67^{‡}$	0.14	$70.43^{‡}$	0.27
	NSVD-concat	$64.07^{‡}$	0.24	$99.44^{‡}$	0.96
	NSVD-stack	$\mathbf{71.55}^{‡}$	**0.28**	$\mathbf{99.51}^{†}$	**0.97**
Hist. + Pred.	MAC-CQ	$35.09^{‡}$	0.13	$39.53^{‡}$	0.15
	+ CAA	$36.40^{‡}$	0.14	$37.72^{‡}$	0.15
	+ MTM	6.19	0.09	7.03	0.09
	HCN	$46.91^{‡}$	0.13	$68.82^{‡}$	0.25
	NSVD-concat	$64.07^{‡}$	0.24	$99.44^{‡}$	0.96
	NSVD-stack	$\mathbf{71.55}^{‡}$	**0.28**	$\mathbf{99.51}^{†}$	**0.97**

Table 3: Results when training on split AA and testing on split BB. Best results in bold. ‡, †, and $*$ represent $p < 0.00001$, $p < 0.05$ and $p \geq 0.05$ compared to the second best score in each column, respectively.

in both splits, we generated five dialogs consisting of 10 rounds as in (Kottur et al., 2019). Split AA contains a training and a validation set based on $6,000$ and 1500 images, respectively. Split BB has a fine-tuning, a validation, and a test set generated on 2000, 500, and 5000 images, respectively. The desired behaviour for a model that generalises well is to perform well on split BB when only trained on split AA.

NSVD-concat and *NSVD-stack* achieve an accuracy of 64.07% and 71.55%, respectively, when tested on split BB without fine-tuning, thereby significantly outperforming all baselines (Table 3). However, low NFRR values indicate that first failures occur early on in the dialog. After fine-tuning all models on a small amount of data from split BB, our models achieve accuracies and NFRR values comparable to previous experiments. In stark contrast, purely-connectionist baselines' performance only improves by a small margin with the highest jump of 5.26% being achieved by *MAC-CQ-CAA*. This shows an impressive data efficiency of the neuro-symbolic models that, in contrast to the data-hungry purely-connectionist baselines, are able to learn and adapt efficiently from a very small amount of data. More details regarding the training data efficiency can be found in the Appendix A.6.

6 Conclusion and Future Work

We proposed Neuro-Symbolic Visual Dialog (NSVD) —the first hybrid method to combine deep learning and symbolic program execution for multi-round visual reasoning —and a new, stricter and more realistic evaluation scheme for visual dialog. Our method outperforms state-of-the-art purely-connectionist baselines on CLEVR-Dialog and sets a new near-perfect test accuracy of 99.72%. Furthermore, NSVD has a higher mean First Failure Round, is more robust to incomplete dialog histories, and generalises better to longer dialogs and to unseen question types and scenes. These performance improvements are not to be seen in isolation of the stricter supervision our models have as they require a supervised fine-tuning of a Mask-RCNN in addition to a supervised training of the program parsers. Finally, additional evaluations show that our models generalise to other scene domains (Appendix A.7) and we expect them to even generalise to naturalistic datasets as well if they provide the necessary supervision requirements for our models similar to the VQA scenario (Wang et al., 2017; Gan et al., 2017). To the best of our knowledge, such datasets for visual dialog do not yet exist.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments, Jiayuan Mao for providing us with the Minecraft data, and Dominike Thomas for paper editing support. M. Bâce was funded by a Swiss National Science Foundation (SNSF) Early Postdoc Mobility Fellowship (No. 199991). A. Bulling was funded by the European Research Council (ERC; grant agreement 801708).

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Semantic machines et al. task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of the International Conference on Computer Vision*, pages 2425–2433.

Forough Arabshahi, Jennifer Lee, Mikayla Gawarecki, Kathryn Mazaitis, Amos Azaria, and Tom M.

Mitchell. 2021. Conversational neuro-symbolic commonsense reasoning. In *Proceedings of the Conference on Artificial Intelligence*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, Stefan Lee, Jose M.F. Moura, Devi Parikh, and Dhruv Batra. 2019. Visual Dialog. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(5):1242–1256.

Chuang Gan, Yandong Li, Haoxiang Li, Chen Sun, and Boqing Gong. 2017. VQS: Linking Segmentations to Questions and Answers for Supervised Attention in VQA and Question-Focused Semantic Segmentation. In *Proceedings of the International Conference on Computer Vision*, pages 1829–1838.

Zhe Gan, Yu Cheng, Ahmed Kholy, Linjie Li, Jingjing Liu, and Jianfeng Gao. 2019. Multi-step reasoning via recurrent dual attention for visual dialog. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1051–1062.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the International Conference on Computer Vision*, pages 2961–2969.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the International Conference on Computer Vision*, pages 804–813.

Drew A Hudson and Christopher D Manning. 2018. Compositional attention networks for machine reasoning. In *Proceedings of the International Conference on Learning Representations*.

Justin Johnson, Li Fei-Fei, Bharath Hariharan, C. Lawrence Zitnick, Laurens Van Der Maaten, and Ross Girshick. 2017a. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1988–1997.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017b. Inferring and executing programs for visual reasoning. In *Proceedings of the International Conference on Computer Vision*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? text-to-image coreference. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3558–3565.

Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the European Conference on Computer Vision*, pages 153–169.

Satwik Kottur, José M.F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2019. Clevr-dialog: A diagnostic dataset for multi-round reasoning in visual dialog. In *Proceeding of the Conference of the North American Chapter of the Association for Computational Linguistics*, volume 1, pages 582–595.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *Proceedings of the International Conference on Learning Representations*, pages 1–28.

Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu, and Ji-Rong Wen. 2019. Recursive visual attention in visual dialog. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Conference on Artificial Intelligence*.

Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. 2014. Linking people in videos with "their" names using coreference resolution. In *Proceedings of the European Conference on Computer Vision*, pages 95–110.

Anna Rohrbach, Marcus Rohrbach, Siyu Tang, Seong Joon Oh, and Bernt Schiele. 2017. Generating descriptions with grounded and co-referenced people. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 4979–4989.

Muhammad Shah, Shikib Mehri, and Tejas Srinivasan. 2020. Reasoning Over History: Context Aware Visual Dialog. In *Proceedings of the International*

Workshop on Natural Language Processing Beyond Text, pages 75–83. Association for Computational Linguistics.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistic*, pages 2238–2249.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. 2017. Explicit knowledge-based reasoning for visual question answering. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1290–1296.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 665–677.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280.

Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. 2017. Neural Scene De-rendering. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. 2020. CLEVRER: collision events for video representation and reasoning. In *Proceedings of the International Conference on Learning Representations*.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. 2018. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *Advances in Neural Information Processing Systems*.

Zilong Zheng, Wenguan Wang, Siyuan Qi, and Song-Chun Zhu. 2019. Reasoning visual dialogs with structural and partial observations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

A Appendix

A.1 The CLEVR-Dialog DSL

Our Domain Specific Language (DSL) for CLEVR-Dialog is depicted in Table 4. We present each function with its expected argument types, output, and knowledge base operations. The argument types are further defined in Table 5. We use the variables `attr`, `attr_obj_1`, `attr_obj_2` and `attr_i` for `i = 1,..,4` to denote the set of possible CLEVR attributes, i.e. colour, material, shape, and size. Furthermore, the variable `pos` denote one possible position, i.e. `right`, `left`, `front`, or `behind`. Finally, the variable `num` denotes the set of possible numerical values , i.e. between 0 and N, where N is the maximum number of objects in the scene. If not explicitly stated otherwise, we assume $N = 10$.

A.2 Normalised First Failure Round

The Normalised First Failure Round NFFR is calculated as

$$\text{NFFR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{L+1}\sum_{j=1}^{L}\Delta_j^{(i)}\delta_{\text{pred}_j^{(i)},\text{gt}_j^{(i)}}\alpha_j^{(i)},$$

where N is the total number of dialogs, L is the length of each dialog, and $\text{pred}_j^{(i)}$ and $\text{gt}_j^{(i)}$ are the predicted and ground truth answers at round j of dialog i, respectively. Furthermore, for each round j of every dialog i, we define $\Delta_j^{(i)}$, $\delta_{\text{pred}_j^{(i)},\text{gt}_j^{(i)}}$, and $\alpha_j^{(i)}$ as:

$$\Delta_j^{(i)} = \begin{cases} j & \text{if} \quad j \leq L \\ L+1 & \text{if} \quad j = L \wedge \delta_{\text{pred}_j^{(i)},\text{gt}_j^{(i)}} = 1 \end{cases},$$

$$\delta_{\text{pred}_j^{(i)},\text{gt}_j^{(i)}} = \begin{cases} 1 & \text{if} \quad \text{pred}_j^{(i)} \neq \text{gt}_j^{(i)} \\ 0 & \text{otherwise} \end{cases},$$

$$\alpha_j^{(i)} = \begin{cases} 0 & \text{if } \exists k < j \text{ s.t. } \delta_{\text{pred}_k^{(i)},\text{gt}_k^{(i)}} = 1 \\ 1 & \text{otherwise} \end{cases}.$$

By definition, we set the NFFR of a model to be $L + 1$ if it correctly answers all L dialog rounds.

A.3 Training Details

Our Models. In order to encode our raw data, we generated two different vocabularies from the training data: one that handles the captions, questions, and answers in form of natural language and another that deals with ground truth caption and question programs. For encoding, we pad all captions, questions, and programs to a maximum length of $n_q = 21$, $n_c = 16$, and $n_y = 6$, respectively. Then, the corresponding tokens are transformed into a 300-dim. space using either the encoder text embedding or the decoder program embedding. We

	Func. Name	Func. Args.	Func. Out.	fetch	update			
					Handle	Conv. Subj.	Seen Objs.	Groups
Caption Programs	count-att	attr	none	✗	✓	✗	✓	✓
	extreme-right	⊎ [attr_1,..,attr_4]	none	✗	✓	✓	✓	✗
	extreme-left	⊎ [attr_1,..,attr_4]	none	✗	✓	✓	✓	✗
	extreme-behind	⊎ [attr_1,..,attr_4]	none	✗	✓	✓	✓	✗
	extreme-front	⊎ [attr_1,..,attr_4]	none	✗	✓	✓	✓	✗
	extreme-centre	⊎ [attr_1,..,attr_4]	none	✗	✓	✓	✓	✗
	unique-obj	⊎ [attr_1,..,attr_4]	none	✗	✓	✓	✓	✗
	obj-relation	attr_obj_1,pos,attr_obj_2	none	✗	✓	✓	✓	✗
Question Programs	count-all	-	num	✗	✗	✗	✗	✓
	count-other	-	num	✗	✗	✓	✓	✗
	count-all-group	-	num	✗	✗	✗	✗	✗
	count-attribute	attr	num	✗	✓	✓	✓	✓
	count-attribute-group	attr	num	✗	✓	✓	✓	✓
	count-obj-rel-imm	pos	num	✗	✗	✓	✓	✓
	count-obj-rel-imm-2	pos	num	✗	✗	✓	✓	✓
	count-obj-rel-early	pos,attr	num	✓	✓	✓	✓	✓
	count-obj-exclude-imm	attr_type	num	✗	✗	✓	✓	✓
	count-obj-exclude-early	attr_type,attr	num	✓	✗	✓	✓	✓
	exist-other	-	yes/no	✗	✗	✗	✓	✓
	exist-attribute	attr	yes/no	✗	✓	✗	✓	✓
	exist-attribute-group	attr	yes/no	✗	✓	✓	✓	✓
	exist-obj-rel-imm	pos	yes/no	✗	✗	✓	✓	✓
	exist-obj-rel-imm2	pos	yes/no	✗	✗	✓	✓	✓
	exist-obj-rel-early	pos,attr	yes/no	✓	✓	✓	✓	✓
	exist-obj-exclude-imm	attr_type	yes/no	✗	✗	✓	✓	✓
	exist-obj-exclude-early	attr_type,attr	yes/no	✓	✗	✗	✗	✓
	seek-attr-imm	attr_type	attr	✗	✓	✗	✗	✗
	seek-attr-imm2	attr_type	attr	✗	✓	✗	✗	✗
	seek-attr-early	attr_type,attr	attr	✓	✓	✓	✓	✗
	seek-attr-sim-early	attr_type,attr	attr	✓	✓	✓	✓	✗
	seek-attr-rel-imm	attr_type	attr	✗	✓	✓	✓	✗
	seek-attr-rel-early	attr_type,pos,attr	attr	✓	✓	✓	✓	✗

Table 4: Our Domain Specific Language (DSL) for CLEVR-Dialog. We present each function with its expected argument types, output, and knowledge base operations. Given a set of n arguments, the ⊎ operator selects a subset of $m \leq n$ possible ones, e.g. ⊎[attr_1,..,attr_4] = [attr_2,attr_3].

```
attr_1 ∈ COLOURS=[blue,brown,cyan,grey,green,purple,red,yellow],
attr_2 ∈ MATERIALS=[rubber,metal],
attr_3 ∈ SHAPES=[cube,cylinder,sphere],
attr_4 ∈ SIZES=[large,small],
attr,attr_obj_1,attr_obj_2 ∈ ⋃{COLOURS,MATERIALS,SHAPES,SIZES},
attr_type ∈ [colour,material,shape,size],
pos ∈ [right,left,front,behind],
num ∈ [0,1,2,3,4,5,6,7,8,9,10].
```

Table 5: Argument types of our DSL.

trained the caption and question program generators separately. We used a 2-layered bi-directional LSTM with a hidden size of 256 in both caption and question encoders. In the decoder, we used a 2-layered LSTM with a hidden size of 512. We fixed the batch size to 64 and used the Adam optimiser (Kingma and Ba, 2014) with a learning rate of 7×10^{-4} to train for *one* epoch. Every 2000 iteration, we validated the models based on the program accuracy in order not to select one that follows a flawed logic, i.e. a wrong program, to predict an answer.

Baselines. We trained the purely-connectionist baselines using the official code [2] we obtained from the authors of Shah et al. (2020). We used the same hyperparameters and training scheme, i.e. we trained the models for a maximum of 25 epochs and used early stopping when the validation accuracy did not improve for five consecutive epochs. For the HCN models, we used a publicly available codebase[3] for training with the same hyper-parameters as in (Williams et al., 2017).

Runtime Analysis. We conducted all of our experiments on a single NVIDIA Tesla V100 GPU. The training runtimes for one epoch when using a batch size of 64 are illustrated in Figure 7. As can be seen, all the purely-connectionist baselines need more than seven hours to complete one epoch. Although NSVD-concat concatenates the previous dialog rounds to form the history similarly to the baselines, it is more computationally efficient as it only needs circa 22 minutes to complete one epoch. Finally, HCN needs around 19 minutes to complete one epoch and NSVD-stack only around 14 which solidifies our aforementioned efficiency claims.

[2]https://github.com/ahmedshah1494/clevr-dialog-mac-net/tree/dialog-macnet

[3]https://github.com/jojonki/Hybrid-Code-Networks

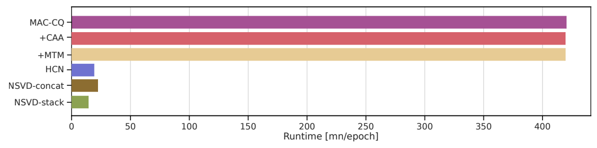


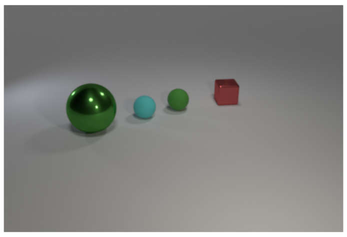
Figure 7: Training runtime comparison of the models. The training was conducted on a single NVIDIA Tesla V100 GPU with batch size 64.

Model	**Prog. Acc.**		**Executor Acc.**
	Caption	**Question**	
Caption-Net	99.79	-	99.99
NSVD-concat	-	99.87	
NSVD-stack	-	99.99	

Table 6: Quantitative analysis of our models' logic. The high program accuracies demonstrate that our models follow the implemented logic to predict the correct answer, i.e. they do not execute false programs that by chance might lead to a correct prediction. In addition, when tested with the ground truth scene annotations and programs, our executor reaches an answer-accuracy of 99.99% showcasing its flawless logic.

A.4 Quantitative Analysis of (our) Logic.

To quantify the logical capabilities of our models, we measured the caption and question program accuracies on the test split. As we can see from Table 6, our model achieves 99.79% caption program accuracy and our best question generator reaches the 99.99% accuracy mark. That is, they do not follow a flawed logic, i.e. a wrong program, to predict the correct answer. Furthermore, we evaluated the logic of our program executor by measuring its answer accuracy when provided with the ground truth programs and scene annotations of the test split. The last column of Table 6 shows that it reaches 99.99% answer accuracy indicating that its logic is close to flawless. The reason why it does not reach the 100% mark is that some scene captions cannot be uniquely interpreted leading to potential



Scene

```
C  : There is a small round thing sitting in the center of the view.
Q1 : If there is a thing in front of it, what size is it? | A1: Small.
Q2 : If there is a thing on the right side of the earlier sphere, what material is it? | A2: Metal.
Q3 : What material is that sphere? | A3: Rubber.
Q4 : What color is the earlier shiny thing? | A4: Red.
Q5 : Are there things present to the right of itself in the picture? | A5: No.
Q6 : How about to its behind? | A6: No.
Q7 : If there is a thing right of the above rubber thing, what shape is it? | A7: Cube.
Q8 : How many other things have its shape? | A8: 0.
Q9 : What color is the previous matte thing? | A9: Green.
Q10: Any cylinders? | A10: No.
```

Dialog

```
Pred. 1 : Large
Pred. 2 : Rubber
Pred. 3 : Rubber
Pred. 4 : Green
Pred. 5 : Yes
Pred. 6 : Yes
Pred. 7 : Sphere
Pred. 8 : 2
Pred. 9 : Green
Pred. 10: No
```

Predictions

Figure 8: Example of a situation where a caption cannot be uniquely interpreted. The program `extreme-centre(cylinder, small)` induced from the caption "there is a small round thing sitting in the centre of the view" does not lead to a unique initialisation of our executor's knowledge base as there are *two* small spheres in the centre of the scene. By our logic, we consider it to be the cyan one. Incorrectly initialising the knowledge base leads to confusion when answering the subsequent questions. The blue and red colours indicate a match or a mismatch between the predicted answer and the ground truth, respectively.

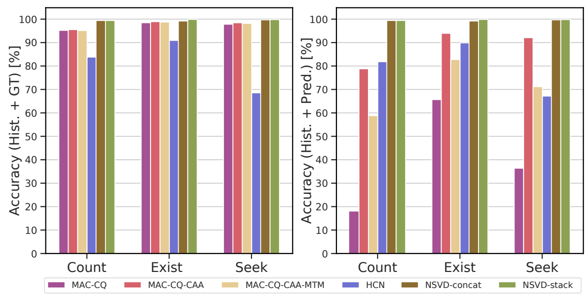


Figure 9: Detailed performance comparison for the different question categories (`Count`, `Exist`, `Seek`). Independent of the evaluation approach, our models achieve new state-of-art results on all question categories. These improvements are statistically significant with $p < 0.0001$ in all categories.

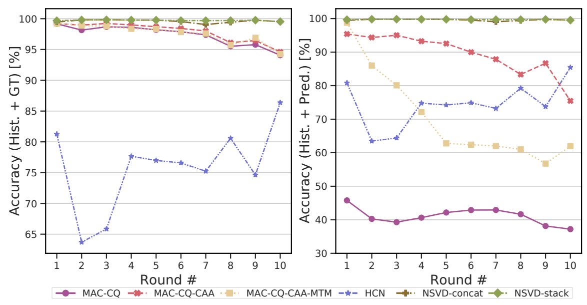


Figure 10: Accuracy for different question rounds. Our models significantly outperform the baselines with $p < 0.0001$ in all question rounds.

confusions when answering the subsequent questions. Figure 8 illustrates a concrete example of such a case. The caption "there is a small round thing sitting in the centre of the view" induces the program `extreme-centre(cylinder, small)`. However, this can be interpreted in two different ways since there are *two* small spheres in the centre of the scene, i.e. the cyan and the green ones. Therefore, the performance of our executor at answering the following dialog questions depends on which object is considered as the central one. By our logic, we consider it to be the cyan one. The subsequent questions, ground truth answers and predictions are also shown in Figure 8.

A.5 Fine-grained Evaluations

When looking at the performance for different question rounds (Figure 10), we can see that while the performance of the baselines starts to deteriorate quickly with longer dialogs, our models achieve a consistently-high performance across all rounds. The inability to maintain performance becomes even more apparent (especially for the purely-connectionist baselines) when using the stricter "Hist. + Pred." evaluation scheme. The best baseline, *MAC-CQ-CAA*, suffers from a drop of 8.48% in accuracy and 0.19 in NFFR. The same observation can be made when analysing performance for the different CLEVR-Dialog question categories (`Count`, `Exist`, and `Seek`). Our models outperform all baselines for all categories and both evaluation schemes (Figure 9). These improvements are statistically significant with $p < 0.00001$.

Figure 11 illustrates the performance of our models on individual question types compared to the baselines. As can be seen, our models outperform the baselines on almost all question types with *NSVD-stack* topping them all with an accuracy of over 98% for all types. As seen from previous experiments, the gap between our models and the baseline becomes ever more conspicuous when the "Hist. + Pred." evaluation scheme is deployed (bottom table of Figure 11).

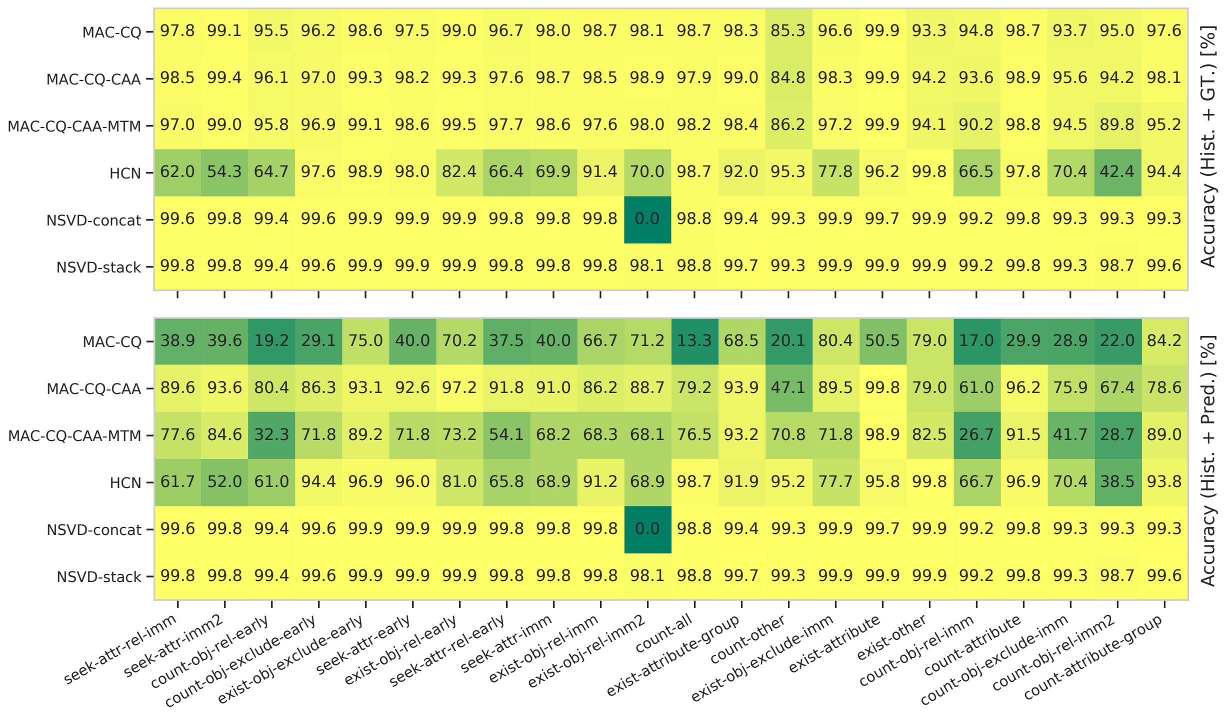


Figure 11: Performance comparison on the individual question types. **Top:** Models were evaluated following the "Hist. + GT" evaluation scheme. **Bottom:** Models were evaluated following the "Hist. + Pred." evaluation scheme

A.6 Data Efficiency

To study the data efficiency of our models further, we trained them and the baselines on $20\%, 40\%, 60\%, 80\%$, and 100% of the available training data. Not to bias the data towards any specific question type, we used the same distribution of question types to construct the reduced training sets. After training, we evaluated the models on the test split using both evaluation schemes. Our models not only outperform all baselines for the same reduced dataset but also in the extreme case of training the baselines with 100% of the data and our models with only 20% (Figure 13). While the performance of the neuro-symbolic models is only slightly affected by the size of the training data, the connectionist baselines' performance deteriorates with less data. This deterioration becomes even more significant when the stricter "Hist. + Pred." evaluation scheme is used.

Model	Hist. + GT		Hist. + Pred.	
	Acc.	**NFFR** ↑	**Acc.**	**NFFR** ↑
MAC-CQ	64.30*	0.27	59.96‡	0.24
+ CAA	64.28‡	0.27	57.69‡	0.23
+ MTM	61.55‡	0.25	52.04‡	0.20
HCN	47.31	0.14	46.50	0.14
NSVD-concat	91.57‡	0.76	91.57‡	0.76
NSVD-stack	**92.46**‡	**0.83**	**92.46**‡	**0.83**

Table 7: Performance comparison on Minecraft-Dialog *test*. Results are shown for both "Hist. + GT" and "Hist. + Pred." Our proposed models are highlighted in grey; best performance is in bold. ‡ and $*$ represents $p < 0.00001$ and $p \geq 0.05$ compared to the second best score in the respective column, respectively.

A.7 Generalisation to Other Scene Domains

In this experiment, we show that our method could be extended to a new reasoning testbed. Contrarily to CLEVR, the new scenes are grounded in Minecraft and are, as can be seen in Figure 12, drastically different in terms of context and scene constellations. Specifically, they comes with more entities (12 vs 3 in CLEVR) that have drastically different visual appearances. These entities can be grouped in a hierarchical manner (e.g. "a cow" and "a wolf" are both "animals" whereas "an animal" and "a human" are both "creatures").

Following (Yi et al., 2018), were rendered $10,000$ using the generation tool provided by Wu et al. (2017). The scenes consist of three to six objects in a 2D plane that are sampled from 12 entities with 4 different facing directions. Finally, we filtered out scenes that contain fully-occluded objects. We used $5,273$ images for training, $1,500$ for validation, and $1,000$ for testing. Furthermore, we adapted the dialog generation tool provided by Kottur et al. (2019) to be able to account for the

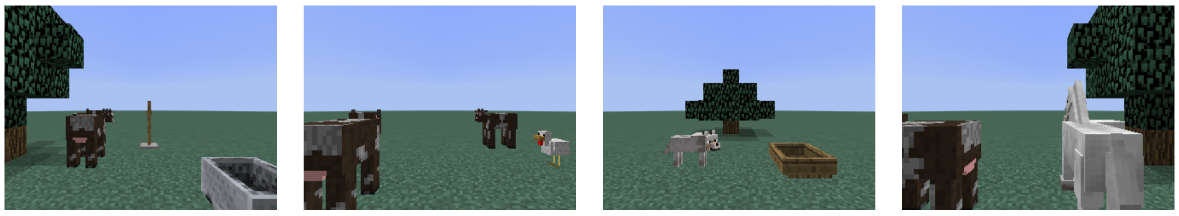


Figure 12: Sample images from the Minecraft dataset.

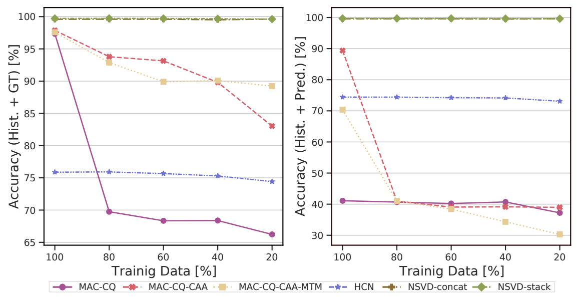


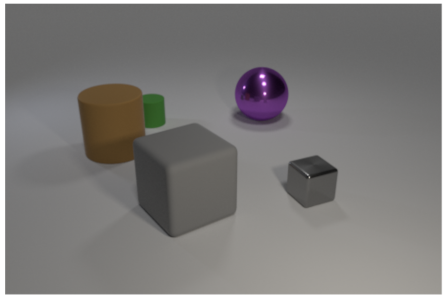
Figure 13: Accuracy when trained on limited amounts of data ($20\%, 40\%, 60\%, 80\%$, and 100% of the overall training data) and evaluated on the test split following the “Hist. + GT” and “Hist. + Pred.” approaches.

different scene properties.

Similar to previous experiments, we generated five dialogs for every image consisting of 10 rounds each. We call this dataset Minecraft-Dialog. The results are summarised in Table 7. We compared our models to the same baselines as in previous experiments in terms of accuracy as well as NFRR. Once again, our neuro-symbolic models managed to significantly outperform all the baselines by achieving 92.64% and 91.57% accuracies for *NSVD-stack* and *NSVD-concat*, respectively, while maintaining high NFRR values.

Contrarily, the best connectionist model achieved test accuracies of 64.30% and 59.96% using the “Hist. + GT” and “Hist. + Pred.” evaluation schemes, respectively. Finally, HCN achieved its best performance of 47.31 accuracy and 0.14 NFRR when the “Hist. + GT” evaluation scheme was used. Compared to CLEVR-Dialog (Table 1), the performance of our models witnessed a drop in this experiment which is attributed to the difficulty of the Minecraft scenes that come with heavy-occluded and diverse objects. However, the promising results of our models showcase that they indeed can generalise to new scene domains other then CLEVR.

A.8 Input/Output Samples



Scene

```
C   : There is a block to the right of all object.
Q1  : If there is an object behind it, what is its shape? | A1:  Cylinder.
Q2  : And material? | A2: Rubber.
Q3  : What about size? | A3: Large.
Q4  : And colour? | A4: Brown.
Q5  : What is the count of objects the previous block has to its front? | A5: 1.
Q6  : How many other objects have the same size as that large object? | A6: 2.
Q7  : Does the picture have other objects that share the same color with the aforementioned cylinder? | A7:No.
Q8  : What color is the aforementioned cube? | A8: Grey.
Q9  : If there is an object in front of it , what is its material? | A9: Rubber.
Q10: Does the previous brown object have objects to its left? | A10: No.
```

Dialog

```
P0 :extreme-right(cube)
P1 :seek-attr-rel-imm(shape, behind)
P2 :seek-attr-imm2(material)
P3 :seek-attr-imm2(size)
P4 :seek-attr-imm2(colour)
P5 :count-obj-rel-early(front, cube)
P6 :count-obj-exclude-early(size, large)
P7 :exist-obj-exclude-early(color, cylinder)
P8 :seek-attr-early(color, cube)
P9 :seek-attr-rel-imm(material, front)
P10:exist-obj-rel-early(left, brown)
```

GT programs

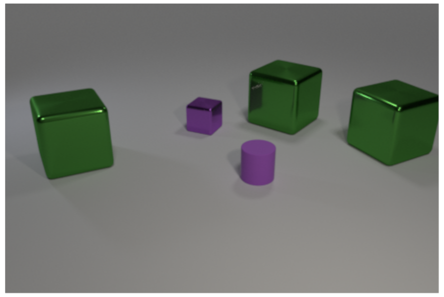
```
P0 :extreme-right(cube)                         Pred. 0 : Init.
P1 :seek-attr-rel-imm(shape, behind)            Pred. 1 : Cylinder
P2 :seek-attr-imm2(material)                    Pred. 2 : Rubber
P3 :seek-attr-imm2(size)                        Pred. 3 : Large
P4 :seek-attr-imm2(colour)                      Pred. 4 : Brown
P5 :count-obj-rel-early(front, cube)            Pred. 5 : 1
P6 :count-obj-exclude-early(size, large)        Pred. 6 : 2
P7 :exist-obj-exclude-early(color, cylinder)    Pred. 7 : No
P8 :seek-attr-early(color, cube)                Pred. 8 : Grey
P9 :seek-attr-rel-imm(material, front)          Pred. 9 : Rubber
P10:exist-obj-rel-early(left, brown)            Pred. 10: No
```

Pred. programs & answers
(NSVD-Stack)

```
P0 :extreme-right(cube)                         Pred. 0 : Init.
P1 :seek-attr-rel-imm(shape, behind)            Pred. 1 : Cylinder
P2 :seek-attr-imm2(material)                    Pred. 2 : Rubber
P3 :seek-attr-imm2(size)                        Pred. 3 : Large
P4 :seek-attr-imm2(colour)                      Pred. 4 : Brown
P5 :count-obj-rel-early(front, cube)            Pred. 5 : 1
P6 :count-obj-exclude-early(size, large)        Pred. 6 : 2
P7 :exist-obj-exclude-early(color, cylinder)    Pred. 7 : No
P8 :seek-attr-early(color, cube)                Pred. 8 : Grey
P9 :seek-attr-rel-imm(material, front)          Pred. 9 : Rubber
P10:exist-obj-rel-early(left, brown)            Pred. 10: No
```

Pred. programs & answers
(NSVD-Concat)

Figure 14: Inference example of our models on a test instance. Both of our models executed correct programs to predict the answers. The blue colour indicates a match between the predicted program/answer and the ground truth.



Scene

```
C   : There is a rubber thing in front of a large thing in the view.
Q1  : How many other things are in the image? | A1:  3.
Q2  : What color is the above large thing? | A2: Green.
Q3  : What is the shape of the previous matte thing? | A3: Cylinder.
Q4  : What material is the previous big thing? | A4: Metal.
Q5  : If there is a thing to the left of that matte thing , what is its color? | A5: Purple.
Q6  : Does it have things to to the right of itself? | A6: Yes.
Q7  : And to its left? | A7: Yes.
Q8  : If there is a thing behind the earlier purple thing , what shape is it? | A8: Cube.
Q9  : If there is a thing in front of the previous purple thing , what is its shape? | A9: Cube.
Q10: What number of red things are present? | A10: 0.
```

Dialog

```
P0  :obj-relation(rubber, large, front)
P1  :count-other()
P2  :seek-attr-early(color, large)
P3  :seek-attr-early(shape, rubber)
P4  :seek-attr-early(material, large)
P5  :seek-attr-rel-early(color, left, rubber)
P6  :exist-obj-rel-imm(right)
P7  :exist-obj-rel-imm2(left)
P8  :seek-attr-rel-early(shape,behind,purple)
P9  :seek-attr-rel-early(shape,front,purple)
P10:count-attribute(red)
```

GT programs

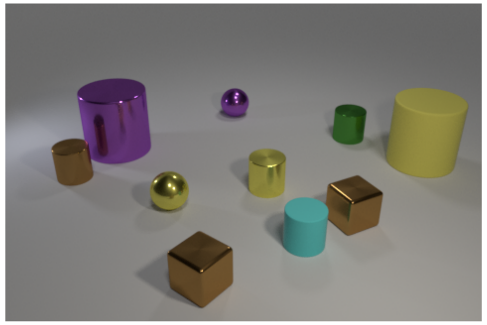
```
P0  :obj-relation(rubber, large, front)            Pred. 0  : Init.
P1  :count-other()                                 Pred. 1  : 3
P2  :seek-attr-early(color, large)                 Pred. 2  : Green
P3  :seek-attr-early(shape, rubber)                Pred. 3  : Cylinder
P4  :seek-attr-early(material, large)              Pred. 4  : Metal
P5  :seek-attr-rel-early(color, left, rubber)      Pred. 5  : Purple
P6  :exist-obj-rel-imm(right)                      Pred. 6  : Yes
P7  :exist-obj-rel-imm2(left)                      Pred. 7  : Yes
P8  :seek-attr-rel-early(shape,behind,purple)      Pred. 8  : Cube
P9  :seek-attr-rel-early(shape,front,purple)       Pred. 9  : Cube
P10:count-attribute(red)                           Pred. 10: 0
```

Pred. programs & answers
(NSVD-Stack)

```
P0  :obj-relation(rubber, large, front)            Pred. 0  : Init.
P1  :count-other()                                 Pred. 1  : 3
P2  :seek-attr-early(color, large)                 Pred. 2  : Green
P3  :seek-attr-early(shape, rubber)                Pred. 3  : Cylinder
P4  :seek-attr-early(material, large)              Pred. 4  : Metal
P5  :seek-attr-rel-early(color, left, rubber)      Pred. 5  : Purple
P6  :exist-obj-rel-imm(right)                      Pred. 6  : Yes
P7  :count-obj-rel-imm2(left)                      Pred. 7  : 1
P8  :seek-attr-rel-early(shape,behind,purple)      Pred. 8  : Cube
P9  :seek-attr-rel-early(shape,front,purple)       Pred. 9  : Cube
P10:count-attribute(red)                           Pred. 10: 0
```

Pred. programs & answers
(NSVD-Concat)

Figure 15: Inference example of our models on a test instance. While *NSVD-stack* answered all questions correctly, *NSVD-concat* failed at round 7. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.



Scene

C : The view has a cyan thing on the left side of a block.
Q1 : What is the count of balls , if present? | A1: 2.
Q2 : What is the size of that block? | A2: Small.
Q3 : If there is a thing in front of the previous cyan thing, what is its material? | A3: Metal.
Q4 : What color is the above cube? | A4: Brown.
Q5 : What is the size of the above metallic thing? | A5: Small.
Q6 : Does it have things to in front of itself in the image? | A6: No.
Q7 : And to its left? | A7 : Yes.
Q8 : If there is a thing in front of the previous brown thing, what material is it? | A8: Rubber.
Q9 : What shape is the above shiny thing? | A9: Cube.
Q10: How many things are in the scene? | A10: 10.

Dialog

```
P0 :obj-relation(cyan, cube, left)
P1 :count-attribute(sphere)
P2 :seek-attr-early(size, cube)
P3 :seek-attr-rel-early(material,front,cyan)
P4 :seek-attr-early(color, cube)
P5 :seek-attr-early(size, metal)
P6 :exist-obj-rel-imm(front)
P7 :exist-obj-rel-imm2(left)
P8 :seek-attr-rel-early(material,front,brown)
P9 :seek-attr-early(shape, metal)
P10:count-all()
```

GT Programs

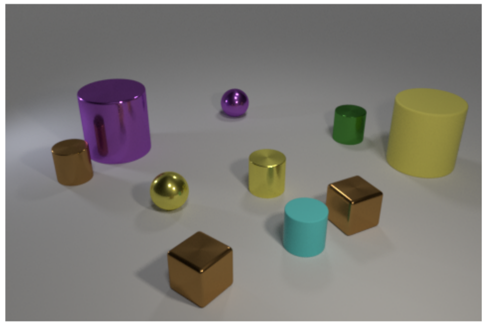
```
P0 :obj-relation(sphere, cube, left)                    Pred. 0 : Init
P1 :count-attribute(sphere)                             Pred. 1 : 0.
P2 :seek-attr-early(size, cube)                         Pred. 2 : Small.
P3 :seek-attr-rel-early(material,front,small)           Pred. 3 : None.
P4 :seek-attr-early(color, cube)                        Pred. 4 : Gray.
P5 :seek-attr-early(size, rubber)                       Pred. 5 : Small.
P6 :exist-obj-rel-imm(front)                            Pred. 6 : No.
P7 :count-obj-rel-imm2(left)                            Pred. 7 : 0.
P8 :seek-attr-rel-early(material,front,rubber)          Pred. 8 : None.
P9 :seek-attr-early(shape, rubber)                      Pred. 9 : Cube.
P10:count-all()                                         Pred. 10: 4.
```

Pred. programs & answers
(NSVD-Stack)

```
P0 :obj-relation(sphere, cube, left)                    Pred. 0 : Init
P1 :count-attribute(sphere)                             Pred. 1 : 0.
P2 :seek-attr-early(size, cube)                         Pred. 2 : Small.
P3 :seek-attr-rel-early(material,front,gray)            Pred. 3 : None.
P4 :seek-attr-early(color, cube)                        Pred. 4 : Gray.
P5 :seek-attr-early(size, sphere)                       Pred. 5 : Small.
P6 :exist-obj-rel-imm(right)                            Pred. 6 : Yes.
P7 :exist-obj-rel-imm2(left)                            Pred. 7 : No.
P8 :seek-attr-rel-early(material,front,gray)            Pred. 8 : None.
P9 :seek-attr-early(shape, rubber)                      Pred. 9 : Cube.
P10:count-all()                                         Pred. 10: 4.
```

Pred. programs & answers
(NSVD-Concat)

Figure 16: Inference example of our models on a test instance with 10 objects before fine-tuning. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.


Scene

```
C   : The view has a cyan thing on the left side of a block.
Q1  : What is the count of balls , if present? | A1: 2.
Q2  : What is the size of that block? | A2: Small.
Q3  : If there is a thing in front of the previous cyan thing, what is its material? | A3: Metal.
Q4  : What color is the above cube? | A4: Brown.
Q5  : What is the size of the above metallic thing? | A5: Small.
Q6  : Does it have things to in front of itself in the image? | A6: No.
Q7  : And to its left? | A7 : Yes.
Q8  : If there is a thing in front of the previous brown thing, what material is it? | A8: Rubber.
Q9  : What shape is the above shiny thing? | A9: Cube.
Q10: How many things are in the scene? | A10: 10.
```

Dialog

```
P0 :obj-relation(cyan, cube, left)
P1 :count-attribute(sphere)
P2 :seek-attr-early(size, cube)
P3 :seek-attr-rel-early(material,front,cyan)
P4 :seek-attr-early(color, cube)
P5 :seek-attr-early(size, metal)
P6 :exist-obj-rel-imm(front)
P7 :exist-obj-rel-imm2(left)
P8 :seek-attr-rel-early(material,front,brown)
P9 :seek-attr-early(shape, metal)
P10:count-all()
```

GT Programs

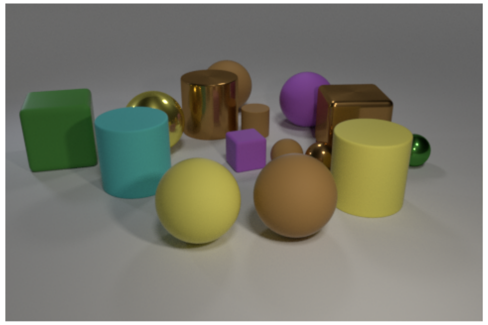
```
P0 :obj-relation(cyan, cube, left)              Pred. 0 : Init
P1 :count-attribute(sphere)                     Pred. 1 : 2.
P2 :seek-attr-early(size, cube)                 Pred. 2 : Small.
P3 :seek-attr-rel-early(material,front,cyan)    Pred. 3 : Metal.
P4 :seek-attr-early(color, cube)                Pred. 4 : Brown.
P5 :seek-attr-early(size, metal)                Pred. 5 : Small.
P6 :exist-obj-rel-imm(front)                    Pred. 6 : No.
P7 :exist-obj-rel-imm2(left)                    Pred. 7 : Yes.
P8 :seek-attr-rel-early(material,front,brown)   Pred. 8 : Rubber.
P9 :seek-attr-early(shape, metal)               Pred. 9 : Cube.
P10:count-all()                                 Pred. 10: 10.
```

Pred. programs & answers
(NSVD-Stack)

```
P0 :obj-relation(cyan, cube, left)              Pred. 0 : Init
P1 :count-attribute(sphere)                     Pred. 1 : 2.
P2 :seek-attr-early(size, cube)                 Pred. 2 : Small.
P3 :seek-attr-rel-early(material,front,cyan)    Pred. 3 : Metal.
P4 :seek-attr-early(color, cube)                Pred. 4 : Brown.
P5 :seek-attr-early(size, metal)                Pred. 5 : Small.
P6 :exist-obj-rel-imm(front)                    Pred. 6 : No.
P7 :exist-obj-rel-imm2(left)                    Pred. 7 : Yes.
P8 :seek-attr-rel-early(material,front,brown)   Pred. 8 : Rubber.
P9 :seek-attr-early(shape, metal)               Pred. 9 : Cube.
P10:count-all()                                 Pred. 10: 10.
```

Pred. programs & answers
(NSVD-Concat)

Figure 17: Inference example of our models on a test instance with 10 objects after fine-tuning. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.



Scene

```
C  : There are 2 green things in the scene.
Q1 : How many of them are cubes? | A1: 1.
Q2 : If there is a thing right of it, what is its color? | A2: Yellow.
Q3 : And material? | A3: Metal.
Q4 : And size? | A4: Large.
Q5 : What material is the aforementioned block? | A5: Rubber.
Q6 : Does the scene have other things that share the same material with the previous metal thing? | A6: Yes.
Q7 : Does the image have other things that share the same shape with the aforementioned rubber thing? | A7 : Yes.
Q8 : What is the shape of the previous yellow thing? | A8: Sphere.
Q9 : What size is the above rubber thing? | A9: Large.
Q10: What number of things are present to its right? | A10: 14.
```

Dialog

```
P0  :count-att(green)
P1  :count-attribute-group(cube)
P2  :seek-attr-rel-imm(color, right)
P3  :seek-attr-imm2(material)
P4  :seek-attr-imm2(size)
P5  :seek-attr-early(material, cube)
P6  :exist-obj-exclude-early(material, metal)
P7  :exist-obj-exclude-early(shape, rubber)
P8  :seek-attr-early(shape, yellow)
P9  :seek-attr-early(size, rubber)
P10:count-obj-rel-imm(right)
```

GT Programs

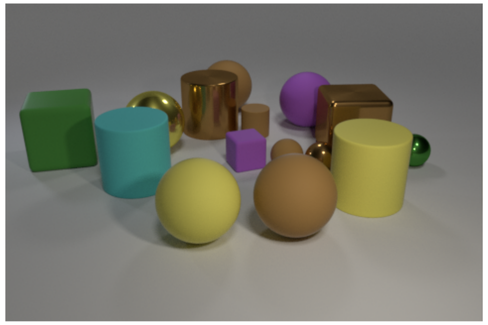
```
P0  :count-att(sphere)                              Pred. 0 : Init
P1  :count-attribute-group(cube)                    Pred. 1 : 0.
P2  :seek-attr-rel-imm(color, right)                Pred. 2 : Error.
P3  :seek-attr-imm2(material)                       Pred. 3 : Error.
P4  :seek-attr-imm2(size)                           Pred. 4 : Error.
P5  :seek-attr-early(material, cube)                Pred. 5 : Error.
P6  :exist-obj-exclude-early(material,rubber)       Pred. 6 : Error.
P7  :exist-obj-exclude-early(shape, rubber)         Pred. 7 : Error.
P8  :seek-attr-early(shape, small)                  Pred. 8 : Error.
P9  :seek-attr-early(size, rubber)                  Pred. 9 : Error.
P10:count-obj-rel-imm(right)                        Pred. 10: Error.
```

Pred. programs & answers (NSVD-Stack)

```
P0  :count-att(cube)                                Pred. 0 : Init
P1  :count-attribute-group(cube)                    Pred. 1 : 4.
P2  :seek-attr-rel-imm(color, right)                Pred. 2 : Error.
P3  :seek-attr-imm2(material)                       Pred. 3 : Error.
P4  :seek-attr-imm2(size)                           Pred. 4 : Error.
P5  :seek-attr-early(material, cube)                Pred. 5 : Error.
P6  :exist-obj-exclude-early(material,shpere)       Pred. 6 : Error.
P7  :exist-obj-exclude-early(shape, rubber)         Pred. 7 : Error.
P8  :seek-attr-early(shape, sphere)                 Pred. 8 : Error.
P9  :seek-attr-early(size, rubber)                  Pred. 9 : Error.
P10:count-obj-rel-imm(right)                        Pred. 10: Error.
```

Pred. programs & answers (NSVD-Concat)

Figure 18: Inference example of our models on a test instance with 15 objects before fine-tuning. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively. `Error` means that the predicted program encountered an exception during execution.



Scene

C : There are 2 green things in the scene.
Q1 : How many of them are cubes? | A1: 1.
Q2 : If there is a thing right of it, what is its color? | A2: Yellow.
Q3 : And material? | A3: Metal.
Q4 : And size? | A4: Large.
Q5 : What material is the aforementioned block? | A5: Rubber.
Q6 : Does the scene have other things that share the same material with the previous metal thing? | A6: Yes.
Q7 : Does the image have other things that share the same shape with the aforementioned rubber thing? | A7 : Yes.
Q8 : What is the shape of the previous yellow thing? | A8: Sphere.
Q9 : What size is the above rubber thing? | A9: Large.
Q10: What number of things are present to its right? | A10: 14.

Dialog

```
P0 :count-att(green)
P1 :count-attribute-group(cube)
P2 :seek-attr-rel-imm(color, right)
P3 :seek-attr-imm2(material)
P4 :seek-attr-imm2(size)
P5 :seek-attr-early(material, cube)
P6 :exist-obj-exclude-early(material, metal)
P7 :exist-obj-exclude-early(shape, rubber)
P8 :seek-attr-early(shape, yellow)
P9 :seek-attr-early(size, rubber)
P10:count-obj-rel-imm(right)
```

GT Programs

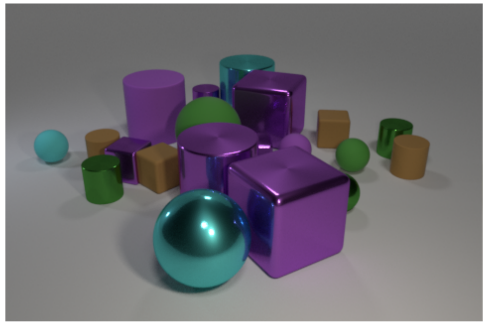
```
P0 :count-att(green)                           Pred. 0 : Init
P1 :count-attribute-group(cube)                Pred. 1 : 1.
P2 :seek-attr-rel-imm(color, right)            Pred. 2 : Cyan.
P3 :seek-attr-imm2(material)                   Pred. 3 : Rubber.
P4 :seek-attr-imm2(size)                       Pred. 4 : Large.
P5 :seek-attr-early(material, cube)            Pred. 5 : Rubber.
P6 :exist-obj-exclude-early(material, metal)   Pred. 6 : Yes.
P7 :exist-obj-exclude-early(shape, rubber)     Pred. 7 : Yes.
P8 :seek-attr-early(shape, yellow)             Pred. 8 : Cylinder.
P9 :seek-attr-early(size, rubber)              Pred. 9 : Large.
P10:count-obj-rel-imm(right)                   Pred. 10: 13.
```

Pred. programs & answers
(NSVD-Stack)

```
P0 :count-att(green)                           Pred. 0 : Init
P1 :count-attribute-group(metal)               Pred. 1 : 1.
P2 :seek-attr-rel-imm(color, right)            Pred. 2 : Cyan.
P3 :seek-attr-imm2(material)                   Pred. 3 : Rubber.
P4 :seek-attr-imm2(size)                       Pred. 4 : Large.
P5 :seek-attr-early(material, cube)            Pred. 5 : Rubber.
P6 :exist-obj-exclude-early(material, metal)   Pred. 6 : Yes.
P7 :exist-obj-exclude-early(shape, rubber)     Pred. 7 : Yes.
P8 :seek-attr-early(shape, yellow)             Pred. 8 : Cylinder.
P9 :seek-attr-early(size, rubber)              Pred. 9 : Large.
P10:count-obj-rel-imm(right)                   Pred. 10: 13.
```

Pred. programs & answers
(NSVD-Concat)

Figure 19: Inference example of our models on a test instance with 15 objects after fine-tuning. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.



Scene

```
C   : There is a matte object to the right of all objects in the view.
Q1  : If there is an object in front of it, what is its color? | A1: Brown.
Q2  : What about size? | A2: Small.
Q3  : What size is the earlier rubber object? | A3: Small.
Q4  : Does the previous brown object have objects to its behind? | A4: Yes.
Q5  : If there is an object to the left of the earlier brown object, what is its color? | A5: Purple.
Q6  : Are there objects present left of the above brown object? | A6: Yes.
Q7  : How many of them are blocks? | A7 : 1.
Q8  : What is its material? | A8: Metal.
Q9  : What size is it? | A9: Small.
Q10: How many other objects are in the view? | A10: 4.
```

Dialog

```
P0  :extreme-right(rubber)
P1  :seek-attr-rel-imm(color, front)
P2  :seek-attr-imm2(size)
P3  :seek-attr-early(size, rubber)
P4  :exist-obj-rel-early(behind, brown)
P5  :seek-attr-rel-early(color, left, brown)
P6  :exist-obj-rel-early(left, brown)
P7  :count-attribute-group(cube)
P8  :seek-attr-imm(material)
P9  :seek-attr-imm(size)
P10:count-other()
```

GT Programs

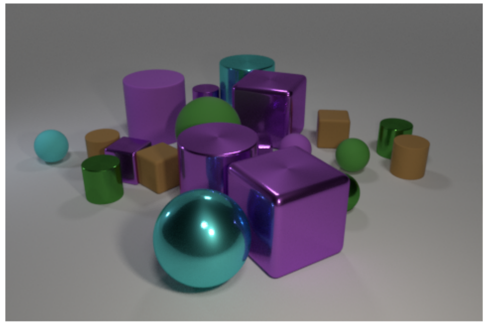
```
P0  :extreme-right(rubber)                          Pred. 0  : Init
P1  :seek-attr-rel-imm(color, front)                Pred. 1  : Gray.
P2  :seek-attr-imm2(size)                           Pred. 2  : Small.
P3  :seek-attr-early(size, rubber)                  Pred. 3  : Small.
P4  :exist-obj-rel-early(behind, rubber)            Pred. 4  : Yes.
P5  :seek-attr-rel-early(color, left, rubber)       Pred. 5  : Blue.
P6  :exist-obj-rel-early(left, rubber)              Pred. 6  : Yes.
P7  :count-attribute-group(cube)                    Pred. 7  : 2.
P8  :seek-attr-imm(material)                        Pred. 8  : Rubber.
P9  :seek-attr-imm(size)                            Pred. 9  : Small.
P10:count-other()                                   Pred. 10: 0.
```

Pred. programs & answers
(NSVD-Stack)

```
P0  :extreme-right(rubber)                          Pred. 0  : Init
P1  :seek-attr-rel-imm(color, front)                Pred. 1  : Gray.
P2  :seek-attr-imm2(size)                           Pred. 2  : Small.
P3  :seek-attr-early(size, rubber)                  Pred. 3  : Small.
P4  :exist-obj-rel-early(behind, cube)              Pred. 4  : Yes.
P5  :seek-attr-rel-early(color, left, red)          Pred. 5  : Blue.
P6  :exist-obj-rel-early(left, gray)                Pred. 6  : Yes.
P7  :count-attribute-group(cube)                    Pred. 7  : 2.
P8  :seek-attr-imm(material)                        Pred. 8  : Rubber.
P9  :seek-attr-imm(size)                            Pred. 9  : Small.
P10:count-other()                                   Pred. 10: 0.
```

Pred. programs & answers
(NSVD-Concat)

Figure 20: Inference example of our models on a test instance with 20 objects before fine-tuning. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.



Scene

```
C   : There is a matte object to the right of all objects in the view.
Q1  : If there is an object in front of it, what is its color? | A1: Brown.
Q2  : What about size? | A2: Small.
Q3  : What size is the earlier rubber object? | A3: Small.
Q4  : Does the previous brown object have objects to its behind? | A4: Yes.
Q5  : If there is an object to the left of the earlier brown object, what is its color? | A5: Purple.
Q6  : Are there objects present left of the above brown object? | A6: Yes.
Q7  : How many of them are blocks? | A7 : 1.
Q8  : What is its material? | A8: Metal.
Q9  : What size is it? | A9: Small.
Q10: How many other objects are in the view? | A10: 4.
```

Dialog

```
P0  :extreme-right(rubber)
P1  :seek-attr-rel-imm(color, front)
P2  :seek-attr-imm2(size)
P3  :seek-attr-early(size, rubber)
P4  :exist-obj-rel-early(behind, brown)
P5  :seek-attr-rel-early(color, left, brown)
P6  :exist-obj-rel-early(left, brown)
P7  :count-attribute-group(cube)
P8  :seek-attr-imm(material)
P9  :seek-attr-imm(size)
P10:count-other()
```

GT Programs

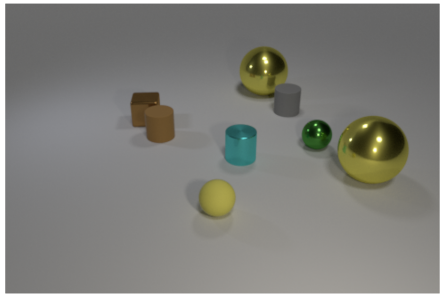
```
P0  :extreme-right(rubber)                      Pred. 0  : Init
P1  :seek-attr-rel-imm(color, front)            Pred. 1  : Green.
P2  :seek-attr-imm2(size)                       Pred. 2  : Small.
P3  :seek-attr-early(size, rubber)              Pred. 3  : Small.
P4  :exist-obj-rel-early(behind, brown)         Pred. 4  : Yes.
P5  :seek-attr-rel-early(color, left, brown)    Pred. 5  : Green.
P6  :exist-obj-rel-early(left, brown)           Pred. 6  : Yes.
P7  :count-attribute-group(cube)                Pred. 7  : 6.
P8  :seek-attr-imm(material)                    Pred. 8  : Rubber.
P9  :seek-attr-imm(size)                        Pred. 9  : Small.
P10:count-other()                               Pred. 10: 0.
```

Pred. programs & answers
(NSVD-Stack)

```
P0  :extreme-right(rubber)                      Pred. 0  : Init
P1  :seek-attr-rel-imm(color, front)            Pred. 1  : Green.
P2  :seek-attr-imm2(size)                       Pred. 2  : Small.
P3  :seek-attr-early(size, rubber)              Pred. 3  : Small.
P4  :exist-obj-rel-early(behind, brown)         Pred. 4  : Yes.
P5  :seek-attr-rel-early(color, left, brown)    Pred. 5  : Green.
P6  :exist-obj-rel-early(left, brown)           Pred. 6  : Yes.
P7  :count-attribute-group(metal)               Pred. 7  : 11.
P8  :seek-attr-imm(material)                    Pred. 8  : Rubber.
P9  :seek-attr-imm(size)                        Pred. 9  : Small.
P10:count-other()                               Pred. 10: 0.
```

Pred. programs & answers
(NSVD-Concat)

Figure 21: Inference example of our models on a test instance with 20 objects after fine-tuning. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.



Scene

```
C   : No other green object except for exactly one.
Q1  : If there is an object to the left of it, what is its color? | A1:  Gray.
Q2  : What about material? | A2: Rubber.
Q3  : How about shape? | A3: Cylinder.
Q4  : What material is the aforementioned green object? | A4: Metal.
Q5  : Are there objects present behind the above gray object? | A5: Yes.
Q6  : What is the count of objects the earlier green object has to its behind? | A6: 4.
Q7  : Are there purple objects? | A7: No.
Q8  : If there is an object to the left of that cylinder, what is its material? | A8: Metal.
Q9  : If there is an object on the right side of it, what size is it? | A9: Small.
Q10: Are there objects present on the left side of the above green object? | A10: Yes.
```

Dialog

```
P0  :unique-obj(green)
P1  :seek-attr-rel-imm(color, left)
P2  :seek-attr-imm2(material)
P3  :seek-attr-imm2(shape)
P4  :seek-attr-early(material, green)
P5  :exist-obj-rel-early(behind, gray)
P6  :count-obj-rel-early(behind, green)
P7  :exist-attribute(purple)
P8  :seek-attr-rel-early(material,left,cylinder)
P9  :seek-attr-rel-imm(size, right)
P10:exist-obj-rel-early(left, green)
```

GT programs

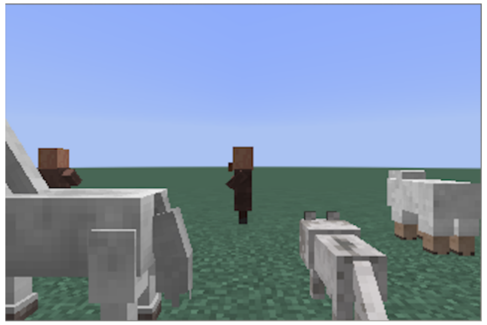
```
P0  :unique-obj(green)                              Pred. 0  : Init.
P1  :seek-attr-rel-imm(color, left)                 Pred. 1  : Gray
P2  :seek-attr-imm2(material)                       Pred. 2  : Rubber
P3  :seek-attr-imm2(shape)                          Pred. 3  : Cylinder
P4  :seek-attr-early(material, green)               Pred. 4  : Metal
P5  :exist-obj-rel-early(behind, gray)              Pred. 5  : Yes
P6  :count-obj-rel-early(behind, green)             Pred. 6  : 4
P7  :exist-other()                                  Pred. 7  : Yes
P8  :seek-attr-rel-early(material,left,cylinder)    Pred. 8  : Metal
P9  :seek-attr-rel-imm(size, right)                 Pred. 9  : Small
P10:exist-obj-rel-early(left, green)                Pred. 10: Yes
```

Pred. programs & answers (NSVD-Stack)

```
P0  :unique-obj(green)                              Pred. 0  : Init.
P1  :seek-attr-rel-imm(color, left)                 Pred. 1  : Gray
P2  :seek-attr-imm2(material)                       Pred. 2  : Rubber
P3  :seek-attr-imm2(shape)                          Pred. 3  : Cylinder
P4  :seek-attr-early(material, green)               Pred. 4  : Metal
P5  :seek-attr-rel-early(color, left, gray)         Pred. 5  : Yellow
P6  :seek-attr-early(size, green)                   Pred. 6  : Small
P7  :exist-other()                                  Pred. 7  : Yes
P8  :seek-attr-rel-early(material,left,cylinder)    Pred. 8  : Metal
P9  :seek-attr-rel-imm(size, right)                 Pred. 9  : Small
P10:seek-attr-early(size, green)                    Pred. 10: Small
```

Pred. programs & answers (NSVD-Concat)

Figure 22: Inference example of our models on a test instance of split BB before fine-tuning. While *NSVD-stack* failed at answering only one question, *NSVD-concat* failed at 4 rounds. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.



Scene

```
C   : The view has a filly.
Q1  : If present, how many humans are in the view? | A1: 2.
Q2  : If there is a thing to the left of the aforementioned horse , what is its class? | A2: Villager.
Q3  : Does the aforementioned mare have things to its front? | A3: No.
Q4  : What direction is the above villager facing | A4: Backward.
Q5  : What is the direction of the previous mare? | A5: Left.
Q6  : If there is a thing on the right side of it, what is its direction? | A6 : Left.
Q7  : What about class? | A7 : Villager.
Q8  : Are there other things that are of same direction as that object facing backward? | A8: Yes.
Q9  : What is the nature of that filly? | A9: Animal.
Q10: What is the count of other things that share its direction? | A10: 1
```

Dialog

```
P0 :unique-obj(horse)
P1 :count-attribute(human)
P2 :seek-attr-rel-early(class, left, horse)
P3 :exist-obj-rel-early(front, horse)
P4 :seek-attr-early(direction, villager)
P5 :seek-attr-early(direction, horse)
P6 :seek-attr-rel-imm(direction, right)
P7 :seek-attr-imm2(class)
P8 :exist-obj-exclude-early(direction, backward)
P9 :seek-attr-early(nature, horse)
P10:count-obj-exclude-imm(direction)
```

GT Programs

```
P0 :unique-obj(horse)                            Pred. 0 : Init
P1 :count-attribute(human)                       Pred. 1 : 2.
P2 :seek-attr-rel-early(class, left, horse)      Pred. 2 : Villager.
P3 :exist-obj-rel-early(front, horse)            Pred. 3 : No.
P4 :seek-attr-early(direction, villager)         Pred. 4 : Backward.
P5 :seek-attr-early(direction, horse)            Pred. 5 : Left.
P6 :seek-attr-rel-imm(direction, right)          Pred. 6 : Left.
P7 :seek-attr-imm2(class)                        Pred. 7 : Villager.
P8 :exist-obj-exclude-early(direction, backward) Pred. 8 : Yes.
P9 :seek-attr-early(nature, horse)               Pred. 9 : Animal.
P10:count-obj-exclude-imm(direction)             Pred. 10: 1.
```

Pred. programs & answers
(NSVD-Stack)

```
P0 :unique-obj(horse)                            Pred. 0 : Init
P1 :count-attribute(human)                       Pred. 1 : 2.
P2 :seek-attr-rel-early(class, left, horse)      Pred. 2 : Villager.
P3 :exist-obj-rel-early(front, horse)            Pred. 3 : No.
P4 :seek-attr-early(direction, villager)         Pred. 4 : Backward.
P5 :seek-attr-early(direction, horse)            Pred. 5 : Left.
P6 :seek-attr-rel-imm(direction, right)          Pred. 6 : Left.
P7 :seek-attr-imm2(class)                        Pred. 7 : Villager.
P8 :exist-obj-exclude-early(direction, backward) Pred. 8 : Yes.
P9 :seek-attr-early(nature, horse)               Pred. 9 : Animal.
P10:count-obj-exclude-imm(direction)             Pred. 10: 1.
```

Pred. programs & answers
(NSVD-Concat)

Figure 23: Inference example of our models on a test instance of Minecraft-Dialog. The blue and red colours indicate a match or a mismatch between the predicted program/answer and the ground truth, respectively.