Probabilistic Neural-symbolic Models for Interpretable Visual Question Answering

Ramakrishna Vedantam [*1] **Karan Desai** [2] **Stefan Lee** [2] **Marcus Rohrbach** [1] **Dhruv Batra** [1 2] **Devi Parikh** [1 2]

Abstract

We propose a new class of probabilistic neural-symbolic models, that have symbolic functional programs as a latent, stochastic variable. Instantiated in the context of visual question answering, our probabilistic formulation offers two key conceptual advantages over prior neural-symbolic models for VQA. Firstly, the programs generated by our model are more understandable while requiring less number of teaching examples. Secondly, we show that one can pose counterfactual scenarios to the model, to probe its beliefs on the programs that could lead to a specified answer given an image. Our results on the CLEVR and SHAPES datasets verify our hypotheses, showing that the model gets better program (and answer) prediction accuracy even in the low data regime, and allows one to probe the coherence and consistency of reasoning performed.

1. Introduction

Building flexible learning and reasoning machines is a central challenge in Artificial Intelligence (AI). Deep representation learning (LeCun et al., 2015) provides us powerful, flexible function approximations that have resulted in state-of-the-art performance across multiple AI tasks such as recognition (Krizhevsky et al., 2012; He et al., 2016), machine translation (Sutskever et al., 2014), visual question answering (Agrawal et al., 2015), speech modeling (van den Oord et al., 2016), and reinforcement learning (Mnih et al., 2015). However, many aspects of human cognition such as systematic compositional generalization (*e.g.*, understanding that "John loves Mary" could imply that "Mary loves John") (Lake et al., 2017; Lake & Baroni, 2018) have proved harder to model.

Symbol manipulation (Newell & Simon, 1976), on the other hand lacks flexible learning capabilities but supports strong generalization and systematicity (Lake & Baroni, 2018). Consequently, many works have focused on building neural-symbolic models with the aim of combining the best of representation learning and symbolic reasoning (Bader & Hitzler, 2005; Evans et al., 2018; Valiant, 2003; Yi et al., 2018; Yin et al., 2018).

As we scale machine learning to machine reasoning (Andreas et al., 2017; 2016; Bottou, 2011; Weston et al., 2016) a natural desire is to provide guidance to the model in the form of instructions. In such a context, symbols are more intuitive to specify than say the parameters of a neural network. Thus, a promising method for interpretable reasoning models is to specify the reasoning plan symbolically and learn to execute it using deep learning.

This neural-symbolic methodology has been extensively used to model reasoning capabilities in visual question answering (VQA) (Andreas et al., 2016; Hu et al., 2017; Mao et al., 2019; Mascharka et al., 2018; Johnson et al., 2017; Yi et al., 2018) and to some extent in reinforcement learning (Andreas et al., 2017; Das et al., 2018). Concretely, in the VQA task one is given an image $\mathbf{i}$, a question $\mathbf{x}$ ("*Is there a cylinder left of a cube?*"), for which we would like to provide an answer a (`yes`). In addition, one may also optionally be provided a program $\mathbf{z}$ for the question that specifies a reasoning plan. For *e.g.*, one might ask a model to apply the `filter[cube]` operator, follwed by `relate[left]`, and then `And` the result together with `filter[cylinder]` to predict the answer, where each operator is parameterized as a neural network (Figure 1).

The scope of this current work is to provide a probabilistic framework for such neural-symbolic models. This provides two benefits: firstly, given a limited number of teaching examples of plans / programs for a given question / context, we show one can better capture the association between the question and programs to provide more understandable and legible program explanations for novel questions. Inspired by Dragan et al. (2013), we call this notion data-efficient legibility, since the model's program (actions) in this case need to be legible, *i.e.* clearly convey the question (goal specification) the model has been given.

[*] Part of this work was done when R.V. was at Georgia Tech. [1]Facebook AI Research [2]Georgia Tech. Correspondence to: Ramakrishna Vedantam <ramav@fb.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

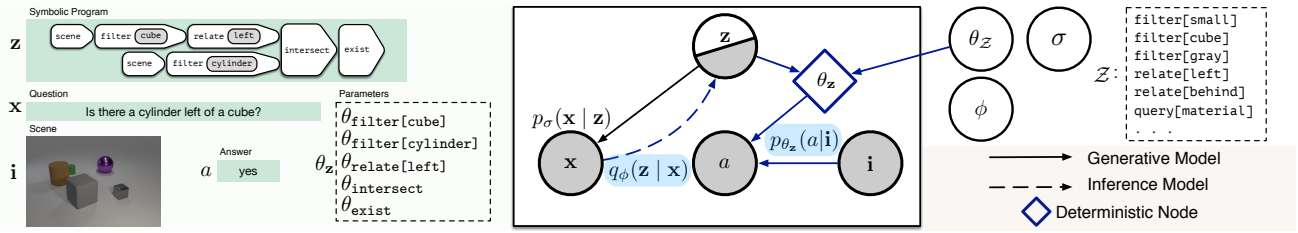

Figure 1. **Probabilistic Neural Symbolic Models for VQA:** We show our novel probabilistic neural-symbolic model (**Prob-NMN**) in plate notation (right). Given image $\mathbf{i}$, programs $\mathbf{z}$ are a (partially observed) latent variable executed on the image to generate an answer a using parameters $\theta_{\mathbf{z}}$ (left). For VQA, we infer $\mathbf{z}$ given question $\mathbf{x}$ to generate an answer. Inference on $\mathbf{z}$ given a and $\mathbf{i}$ tests coherence of reasoning patterns. Baseline non-probabilsitic neural symbolic models (**NMNs**) capture (subset of) terms and edges shown in blue, and are less data-efficient and less interpretable reasoning models than our (probabilistic) proposal.

Secondly, the formulation makes it possible to probe deeper into the model's reasoning capabilities. For *e.g.*, one can test if the reasoning done by the system is: 1) coherent: programs which lead to similar answers are consistent with each other and 2) sensitive: tweaking the answer meaningfully changes the underlying reasoning plan.

Our probabilistic model treats functional programs ($\mathbf{z}$) as a stochastic latent variable. This allows us to share statistics meaningfully between questions with associated programs and those without corresponding programs. This aids data-efficient legibility. Secondly, probabilistic modeling brings to bear a rich set of inferential techniques to answer conditional queries on the beliefs of the model. After fitting a model for VQA, one can sample $\mathbf{z} \sim p(\mathbf{z}|\mathbf{i}, a)$, to probe if the reasoning done by the model is coherent (*i.e.* multiple programs leading to answer `yes` are coherent) and sensitive to a different answer (a) (say, `no`).

Given an image ($\mathbf{i}$), we build a model for $p(\mathbf{x}, \mathbf{z}, a|\mathbf{i})$ (see Figure 1); where the model factorizes as: $p(\mathbf{x}, \mathbf{z}, a|\mathbf{i}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(a|\mathbf{z}, \mathbf{i})$. The implied generative process is as follows. First we sample a program $\mathbf{z}$, which generates questions $\mathbf{x}$. Further, given a program $\mathbf{z}$ and an image $\mathbf{i}$ we generate answers a. Note that based on the symbolic program $\mathbf{z}$, we dynamically instantiate parameters of a neural network $\theta_{\mathbf{z}}$ (these are deterministic, given $\mathbf{z}$), by composing smaller neural-modules for each symbol in the program. This is similar to prior work on neural-symbolic VQA (Hu et al., 2017; Johnson et al., 2017) using neural module networks (NMNs). In comparison to prior works, our probabilistic formulation (shorthand Prob-NMN) leads to better semi-supervised learning, and reasoning capabilities[1].

Our technical contribution is to formulate semi-supervised learning with this deep generative neural-symbolic model using variational inference (Jordan et al., 1999). First, we derive variational lower bounds on the evidence for the model for the semi-supervised and supervised cases, and show how this motivates the semi-supervised objectives used in previous work with discrete structured latent spaces (Miao & Blunsom, 2016; Yin et al., 2018). Next, we show how to learn program execution, *i.e.* $p(a|\mathbf{z}, \mathbf{i})$ jointly with the remaining terms in the model by devising a stage-wise optimization algorithm inspired by (Johnson et al., 2017).

Contributions. First, we provide tractable algorithms for training models with probabilistic latent programs that also learn to execute them in an end-to-end manner. Second, we take the first steps towards deriving useful semi-supervised learning objectives for the class of structured sequential latent variable models (Miao & Blunsom, 2016; Yin et al., 2018). Third, our approach enables interpretable reasoning systems that are more legible with less supervision, and expose their reasoning process to test coherence and sensitivity. Fourth, our system offers improvements on the CLEVR (Johnson et al., 2017) as well as SHAPES (Andreas et al., 2016) datasets in the low question program supervision regime. That is, our model answers questions more accurately and with legible (understandable) programs than adaptations of prior work to our setting as well as baseline non-probabilistic variants of our model.

2. Methods

We first explain the model and its parameterization, then detail the training objectives along with a discussion of a stage-wise training procedure.

Let $\mathbf{i} \in \mathbb{R}^{U \times V}$ be an input image, $\mathbf{x}$ be a question, which is comprised of a sequence of words $(x_1, \cdots, x_t)$, where each $x_t \in \mathcal{X}$, where $\mathcal{X}$ is the vocabulary comprising all words, similarly, $a \in \mathcal{A}$ be the answer (where $\mathcal{A}$ is the answer vocabulary), and $\mathbf{z}$ be the prefix serialization of a program. That is, $\mathbf{z} = (z_1, \cdots, z_T) \ni z_t \in \mathcal{Z}$, where $\mathcal{Z}$ is

[1]Note that this model assumes independence of programs from images, which corresponds to the weak sampling assumptions in concept learning (Tenenbaum, 1999), one can handle question premise, *i.e.* that people might ask a specific set of questions for an image in such a model by reparameterizing the answer variable to include a relevance label.

the program token vocabulary.

The model we describe below assumes $\mathbf{z}$ is a latent variable that is observed only for a subset of datapoints in $\mathcal{D}$. Concretely, the programs $\mathbf{z}$ express (prefix-serializations of) tree structured graphs. Computations are performed given this symbolic representation by instantiating, for each symbol $z \in \mathcal{Z}$ a corresponding neural network (Figure 1, right) with parameters θ_z (Figure 1). That is, given a symbol in the program, say `find[green]`, the model instantiates parameters $\theta_{\texttt{find[green]}}$. In this manner, the mapping $p(a|\mathbf{i}, \mathbf{z})$ is operationalized as $p(a|\mathbf{i}; \theta_{\mathbf{z}})$, where $\theta_{\mathbf{z}} = \{\theta_{z_t}\}_{t=1}^{T}$.

Our model factorizes as $p(\mathbf{x}, \mathbf{z}, a|\mathbf{i}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(a|\mathbf{i}; \theta_{\mathbf{z}})$ (see Figure 1), where parameters $\theta_{\mathbf{z}}$ are a deterministic node in the graph instantiated given the program $\mathbf{z}$ and $\theta_{\mathcal{Z}}$ (which represents the concatenation of parameters across all tokens in $\mathcal{Z}$). In addition to the generative model, we also use an inference network $q_\phi(\mathbf{z}|\mathbf{x})$ to map questions to latent structured programs. Thus, the generative parameters in the model are $\{\theta_{\mathcal{Z}}, \sigma\}$, and inference parameters are ϕ.

2.1. Parameterization

The terms $p(\mathbf{z})$, $p_\sigma(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are all parameterized as LSTM neural networks (Hochreiter & Schmidhuber, 1997). The program prior $p(\mathbf{z})$ is pretrained using maximum likelihood on programs simulated using the syntax of the program or a held-out dataset of programs. The prior is learned first and kept fixed for the rest of the training process. Finally, the parameters θ_z for symbols z parameterize small, deep convolutional neural networks which optionally take as input an attention map over the image (see Appendix for more details).

For reference, previous non-probabilistic, neural-symbolic models for VQA (Hu et al., 2017; Johnson et al., 2017; Mascharka et al., 2018) model $q_\phi(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{a}|\mathbf{i}; \theta_{\mathbf{z}})$ terms from our present model (see blue arrows in Figure 1).

2.2. Learning

We assume access to a dataset $\mathcal{D} = \{\mathbf{x}^n, \mathbf{z}^n\} \cup \{\mathbf{x}^m, a^m, \mathbf{i}^m\}$ where m indexes the visual question answering dataset, with questions, corresponding images and answers, while n indexes a teaching dataset, which provides the corresponding programs for a question, explaining the steps required to solve the question. For data-efficient legibility, we are interested in the setting where $N < M$ *i.e.* we have few annotated examples of programs which might be more expensive to specify.

Given this, learning in our model consists of estimating parameters $\{\theta_{\mathcal{Z}}, \sigma, \phi\}$. We do this in a stage-wise fashion (shown below) (c.f. Johnson et al. (2017)):

Stage-wise Optimization.

1) Question Coding: Optimizing parameters $\{\sigma, \phi\}$ to learn a good code for questions $\mathbf{x}$ in the latent space $\mathbf{z}$.
2) Module Training: Optimizing parameters $\theta_{\mathcal{Z}}$ for learning to execute symbols z using neural networks θ_z.
3) Joint Training: Learning all the parameters of the model $\{\theta_{\mathcal{Z}}, \sigma, \phi\}$.
We describe each of the stages in detail below, and defer the reader to the Appendix for a more details.

Question Coding. We fit the model on the evidence for questions $\mathbf{x}$ and programs $\mathbf{z}$, marginalizing answers a, *i.e.* $\sum_n \log p(\mathbf{x}^n, \mathbf{z}^n) + \sum_m \log p(\mathbf{x}^m)$. We lower-bound the second term, optimizing the evidence lower bound (ELBO) with an amortized inference network q_ϕ (c.f. Kingma & Welling (2014); Rezende et al. (2014)):

$$\sum_m \log p(\mathbf{x}^m) \geq \sum_m \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^m)}[\log p_\sigma(\mathbf{x}^m|\mathbf{z}) - \beta \log q_\phi(\mathbf{z}|\mathbf{x}^m) + \beta \log p(\mathbf{z})] = \mathcal{U}_{qc}^{\beta} \tag{1}$$

where the lower bound holds for $\beta > 1$. In practice, we follow prior work in violating the bound, using $\beta < 1$ to scale the contribution from $\mathbb{KL}\left(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)$ in Equation (1). This has been shown to be important for learning meaningful representations when decoding sequences (c.f. Alemi et al. (2018); Bowman et al. (2016); Miao & Blunsom (2016); Yin et al. (2018)). For more context, Alemi et al. (2018) provides theoretical justifications for why this is desirable when decoding sequences which we discuss in more detail in the Appendix.

The $\mathcal{U}_{qc}$ term in Equation (1) does not capture the semantics of programs, in terms of how they relate to particular questions. For modeling legible programs, we would also like to make use of the labelled data $\{\mathbf{x}^n, \mathbf{z}^n\}$ to learn associations between questions and programs, and provide legible explanations for novel questions $\mathbf{x}$. To do this, one can factorize the model to maximize $\mathcal{L} = \sum_n \log p(\mathbf{x}^n|\mathbf{z}^n) + \log p(\mathbf{z}^n)$. While in theory given the joint, it is possible to estimate $p(\mathbf{z}|\mathbf{x})$, this is expensive and requires an (in general) intractable sum over all possible programs. Ideally, one would like to reuse the *same* variational approximation $q_\phi(\cdot)$ that we are training for $\mathcal{U}_{qc}$ so that it learns from both labelled as well as unlabelled data (c.f. Kingma et al. (2014)). We prove the following lemma and use it to construct an objective that makes use of q_ϕ, and relate it to the evidence.

Lemma 1. *Given observations $\{\mathbf{x}^n, \mathbf{z}^n\}$ and $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p_\sigma(\mathbf{x}|\mathbf{z})$, let z_t, the token at the t^{th} timestep in a sequence $\mathbf{z}$ be distributed as a categorical with parameters π_t. Let us denote $\Pi = \{\pi_t\}_{t=1}^{T}$, the joint random variable over all π_t. Then, the following is a lower bound on the joint evidence $\mathcal{L} = \sum_n \log p(\mathbf{z}^n) + \log p_\sigma(\mathbf{x}^n|\mathbf{z}^n)$:*

$$\mathcal{L} \geq \sum_{n=1}^{N} \log q_\phi(\mathbf{z}^n|\mathbf{x}^n) + \log p_\sigma(\mathbf{x}^n|\mathbf{z}^n) - \mathbb{KL}\left(q(\Pi|\mathbf{z}^n, \mathbf{x}^n)||p(\Pi)\right) \tag{2}$$

where $p(\Pi)$ is a distribution over the sampling distributions implied by the prior $p(\mathbf{z})$ and $q(\Pi|\mathbf{x}^n, \mathbf{z}^n) = q(\pi_0)\prod_{t=1}^{T} q(\pi_t|\mathbf{x}^n, z_0^n, \cdots, z_{t-1}^n)$, where each $q(\pi_t|\cdot)$ is a delta distribution on the probability simplex.

See Appendix for a proof of the result. This is an extension of the result for a related graphical model (with discrete $\mathbf{z}$ observations) from (Kingma et al., 2014; Keng, 2017) to the case where $\mathbf{z}$ is a sequence.

In practice, the bound above not useful, as the proof assumes a delta posterior $q(\Pi|\mathbf{z}, \mathbf{x})$ which makes the last KL term ∞. This means we have to resort to learning only the first two terms in Lemma 1 as an approximation:

$$\mathcal{L} \approx \sum_{n=1}^{N} \alpha \log q_\phi(\mathbf{z}^n|\mathbf{x}^n) + \log p_\sigma(\mathbf{x}^n|\mathbf{z}^n) \quad (3)$$

where $\alpha > 1$ is a scaling on $\log q_\phi$, also used in Kingma & Welling (2014); Miao & Blunsom (2016) (see Appendix for more details and a justification).

Connections to other objectives. To our knowledge, two previous works (Miao & Blunsom, 2016; Yin et al., 2018) have formulated semi-supervised learning with discrete (sequential) latent variable models. While Miao & Blunsom (2016) write the supervised term as $\log p_\phi(\mathbf{z}|\mathbf{x})$, Yin et al. (2018) write it as $\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\sigma(\mathbf{x}|\mathbf{z})$. The lemma above provides a clarifying perspective on both the objectives, firstly showing that p_ϕ should actually be written as q_ϕ (and suggests an additional p_σ term), and that the objective from Yin et al. (2018) is actually a part of a loose lower bound on the evidence for $\mathcal{L}$ providing some justification for the intuition presented in Yin et al. (2018)[2].

We next explain the evidence formulation for the full graphical model; and then introduce the module training and joint training steps.

Module and Joint Training. For the full model (including the answers a), the evidence is $\mathcal{L} + \mathcal{U}_f$, where $\mathcal{U}_f = \sum_{m=1}^{M} \log p(\mathbf{x}^m, a^m|\mathbf{i}^m)$ and $\mathcal{L} = \sum_{n=1}^{N} \log p(\mathbf{x}^n, \mathbf{z}^n)$. Similar to the previous section, one can derive a variational lower bound (Jordan et al., 1999) on $\mathcal{U}_f$ (c.f. (Vedantam et al., 2018; Suzuki et al., 2017)):

$$\mathcal{U}_f \geq \sum_{m=1}^{M} E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^m)}[\log p(a^m|\mathbf{i}^m; \theta_\mathbf{z}) + \log p_\sigma(\mathbf{x}^m|\mathbf{z})] - \mathbb{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}^m)||\log p(\mathbf{z})\right) \quad (4)$$

Module Training. During module training, first we optimize the model only w.r.t the parameters responsible for neural execution of the symbolic programs, namely $\theta_\mathcal{Z}$. Concretely, we maximize:

$$\max_{\theta_\mathcal{Z}} \sum_{m=1}^{M} E_{z\sim q(\mathbf{z}|\mathbf{x}^m)}[\log p_{\theta_z}(a^m|\mathbf{z}, \mathbf{i}^m)] \quad (5)$$

The goal is to find a good initialization of the module parameters, say $\theta_{\texttt{find[green]}}$ that binds the execution to the computations expected for the symbol `find[green]` (namely the neural module network).

Joint Training. Having trained the question code and the neural module network parameters, we train all terms jointly, optimizing the complete evidence with the lower bound $\mathcal{L} + \mathcal{U}_f$. We make changes to the above objective (across all the stages), by adding in scaling factors α, β and γ for corresponding terms in the objective, and write out the KL term (Equation (4)), subsuming it into the expectation:

$$\mathcal{L} + \mathcal{U}_f \approx \sum_{m=1}^{M} E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^m)}[\gamma \log p(a^m|\mathbf{i}^m; \theta_\mathbf{z}) + \log p_\sigma(\mathbf{x}^m|\mathbf{z}) - \beta \log q_\phi(\mathbf{z}|\mathbf{x}^m) + \beta \log p(\mathbf{z})] + \sum_{n=1}^{N} [\alpha \log q_\phi(\mathbf{z}^n|\mathbf{x}^n) + \log p_\sigma(\mathbf{x}^n|\mathbf{z}^n)] \quad (6)$$

where $\gamma > 1$ is a scaling factor on the answer likelihood, which has fewer bits of information than the question. For answers a, which have probability mass functions, $\gamma > 1$ still gives us a valid lower bound[3]. The same values of β and α are used as in question coding (and for the same reasons explained above).

The first term, with expectation over $\mathbf{z} \sim q_\phi(\cdot)$ is not differentiable with respect to ϕ. Thus we use the REINFORCE (Williams, 1992) estimator with a moving average baseline to get a gradient estimate for ϕ (see Appendix for more details.). We take the gradients (where available) for updating the rest of the parameters.

2.3. Benefits of Three Stage Training

In this section, we outline the difficulties that arise when we try to optimize Equation (6) directly, without following the three stage procedure. Let us consider question coding – if we do not do question coding independently of the answer, learning the parameters θ_z of the neural module network becomes difficult, especially when $N << M$ as the mapping $p(a|\mathbf{z}, \mathbf{i})$ is implemented using neural modules $p_{\theta_z}(a|\mathbf{i})$. This optimization is discrete in the program choices, which hurts when q_ϕ is uncertain (or has not converged). Next, training the joint model without first running module training is possible, but trickier, because the gradient from an untrained neural module network would pass

[2] A promising direction for obtaining tighter bounds could be to change the parameterization of the variational $q(\Pi)$ distribution. Overall, learning of q_ϕ is challenging in the structured, discrete space of sequences and a proper treatment of how to train this term in a semi-supervised setting is important for this class of models.

[3] Similar scaling factors have been found to be useful in prior work (Vedantam et al., 2018) (Appendix A.3) in terms of shaping the latent space.

Algorithm 1 Prob-NMN Training Procedure

Given: $\mathcal{D} = \{\mathbf{x}^n, \mathbf{z}^n\}_{n=1}^N \cup \{\mathbf{x}^m, a^m, \mathbf{i}^m\}_{m=1}^M, p(\mathbf{z})$
Initialize: $\theta_{\mathcal{Z}}$, σ, ϕ
Set: $\beta < 1, \alpha > 1, \gamma > 1$
Question Coding
Estimate σ, ϕ optimizing Equation (1) + Equation (3)
Module Training
Estimate $\theta_{\mathcal{Z}}$ optimizing Equation (5)
Joint Training
Estimate $\theta_{\mathcal{Z}}$, σ, ϕ optimizing Equation (6)

Note: Updates to ϕ from Equation (1) and Equation (4) happen via. score function estimator, and the path derivative. Updates to $\theta_{\mathcal{Z}}$, σ are computed using the gradient.

into the $q_\phi(\mathbf{z}|\mathbf{x})$ inference network, adding noise to the updates. Indeed, we find that inference often deteriorates when trained with REINFORCE on a reward computed from an untrained network (Table 1).

3. Related Work

We first explain differences with other discrete structured latent variable models proposed in the literature. We then connect our work to the broader context of research in reasoning and visual question answering (VQA) and conclude by discussing interpretability in the context of VQA.

Discrete Structured Latent Variables. Previous works have applied amortized variational inference (Kingma & Welling, 2014; Rezende et al., 2014) to build discrete, structured latent variable models, where the observations as well as the latents are either sequences (Miao & Blunsom, 2016) or tree structured programs (Yin et al., 2018). While Yin et al. (2018) consider the problem of parsing text into programs, Miao & Blunsom (2016) generate (textual) summaries using a latent variable (Miao & Blunsom, 2016). In contrast, our joint model is richer since in addition to parsing, we also learn to decode a latent program into answers by executing neural modules $\theta_{\mathbf{z}}$. Finally, as discussed in Section 2, our derivation for the lower bound on the question-program evidence provides some understanding of the objectives used in these prior works for semi-supervised learning (Yin et al., 2018; Miao & Blunsom, 2016).

Visual Question Answering and Reasoning. A number of approaches have studied visual question answering, motivated to study multi-hop reasoning (Hu et al., 2017; 2018; Hudson & Manning, 2018; Johnson et al., 2017; Perez et al., 2018; Santoro et al., 2017; Yi et al., 2018). Some of these works build in implicit, non-symbolic inductive biases to support compositional reasoning into the network (Hudson & Manning, 2018; Hu et al., 2018; Perez et al., 2018), while others take a more explicit symbolic approach (Yi et al., 2018; Hu et al., 2017; Johnson et al., 2017; Mascharka et al., 2018; Yi et al., 2018). Our high level goal is centered around providing legible explanations and reasoning traces for programs, and thus, we adopt a symbolic approach. Even in the realm of symbols, different approaches utilize different kind of inductive biases in the mapping from symbols (programs) to answer. While Yi et al. (2018) favor an approach that represents objects in a scene with a vectorized representation, and compute various operations as manipulations of the vectors, other works take a more modular approach (Andreas et al., 2016; Hu et al., 2018; Johnson et al., 2017; Mascharka et al., 2018) where a set of symbols $\{z\}$ intsantiate neural networks with parameters $\theta_{\{z\}}$. We study the latter approach since it is arguably more general and could conceivably transfer better to other tasks such as planning and control (Das et al., 2018), lifelong learning (Gaunt et al., 2017; Valkov et al., 2018) *etc.*

Different from all these prior works, we provide a probabilistic scaffolding that embeds previous neural-symbolic models, which we conceptualize should lead to better data-efficient legibility and the ability to debug coherence and sensitivity in reasoning. We are not aware of any prior work on VQA where it is possible to reason about coherence or sensitivity of the reasoning performed by the model.

Interpretable VQA. Given its importance as a scene understanding task, and as a general benchmark for reasoning, there has been a lot of work in trying to interpret VQA systems and explain their decisions (Das et al., 2016; Lu et al., 2016; Park et al., 2018; Selvaraju et al., 2017). Interpretability approaches typically either perform some kind of explicit attention (Bahdanau et al., 2015) over the question or the image (Lu et al., 2016) to explain with a heat map the regions or parts of the question the model used to arrive at an answer. Some other works develop post-hoc attribution techniques (Mudrakarta et al., 2018; Selvaraju et al., 2017) for providing explanations. In this work, we are interested in an orthogonal notion of interpretability, in terms of the legibility of the reasoning process used by the network given symbolic instructions for a subset of examples. More similar to our high-level motivation are approaches which take a neural-symbolic approach, providing explanations in terms of programs used for reasoning about a question (Andreas et al., 2016; Hu et al., 2018), optionally including a spatial attention over the image to localize the function of the modules (Andreas et al., 2016; Hu et al., 2017; Mascharka et al., 2018). In this work we augment the legibility of the programs/reasoning from these approaches.

4. Experimental Setup

Dataset. We report our results on the CLEVR (Johnson et al., 2017) dataset and the SHAPES datasets (Andreas et al., 2016). The CLEVR dataset has been extensively used as a benchmark for testing reasoning in VQA models in various prior works (Hu et al., 2017; 2018; Hudson & Man-

ning, 2018; Johnson et al., 2017; Perez et al., 2018; Santoro et al., 2017) and is composed of 70,000 images and around 700K questions, answers and functional programs in the training set, and 15,000 images and 150K questions in the validation set. We choose first 20K examples from CLEVR v1.0 validation set and use it as our val set. We report results on CLEVR v1.0 test set using the CLEVR evaluation server on EvalAI (Yadav et al., 2019). The longest questions in the dataset are of length 44 while the longest programs are of length 25. The question vocabulary has 89 tokens while the program vocabulary has 40 tokens, with 28 possible answers.

We investigate our design choices on the smaller SHAPES dataset proposed in previous works (Andreas et al., 2016; Hu et al., 2017) for visual question answering. The dataset is explicitly designed to test for compositional reasoning, and contains compositionally novel questions that the model must demonstrate generalization to at test time. Overall there are 244 unique questions with `yes`/`no` answers and 15,616 images (Andreas et al., 2016). The dataset also has annotated programs for each of the questions. We use train, val, and test splits of 13,568, 1,024, and 1,024 $(\mathbf{x}, \mathbf{z}, \mathbf{i}, \mathbf{a})$ triplets respectively. The longest questions in the dataset are of length 11 and shortest are of length 4, while the longest programs are of length 6 and shortest programs are of length 4. The size of the question vocabulary $\mathcal{X}$ is 14 and the program vocabulary $\mathcal{Z}$ is 12.

Training. On SHAPES, to simulate a data-sparse regime, we restrict the set of question-aligned programs to 5, 10, 15, or 20% of unique questions – such that even at the highest level of supervision, programs for 80% of unique questions have never been seen during training. We train our program prior using a set of 1848 unique programs simulated from the syntax (more details of this can be seen in the Appendix).

In general, we find that performance at a given amount of supervision can be quite variable. In addition, we also find question coding, and module training stages tend to show a fair amount of variance across multiple runs. To make fair comparisons, for every experiment, we run question coding across 5 different runs, pick the best performing model, and then run module training (updating $\theta_{\mathcal{Z}}$) across 10 different runs. Next, we run the best model from this stage for joint training (sweeping across values of $\gamma \in \{1, 10, 100\}$). Finally, at the end of this process we have the best model for an entire training run. We repeat this process across five random datasets and report mean and variance at a given level of supervision.

With CLEVR, we report results when we train on 1000 question-program supervision pairs (this is 0.143% of all question-program pairs), along with the rest of the question, image answer pairs (dropping the corresponding programs). Similar to SHAPES, we report our results across 20 different choices of the subset of 1000 questions to estimate statistical significance. In general, the CLEVR dataset has question lengths with really large variance, and we select the subset of 1000 questions with length at most 40. We do this to stabilize training (for both our method as well as the baseline) and also to simulate a realistic scenario where an end user would not want to annotate really long programs. For each choice of a subset of 1000 questions, we run our entire training pipeline (across the question coding, module training and joint training stages) and report results on the top 15 runs out of the 20 (for all comparisons reported in the paper), based on question coding accuracy (see metrics below).

Metrics. For question coding, we report the accuracy of the programs predicted by the $q_\phi(\mathbf{z}|\mathbf{x})$ model (determined with exact string match), since we are interested in the legibility of the generated programs. In module training, we report the VQA accuracy obtained by the model, and finally, in joint training, we report both, VQA and program prediction accuracy since we are interested in both, getting the right answers and the legibility of the model's reasoning trace.

Prior. On SHAPES, we train the program prior $p(\mathbf{z})$ using programs simulated using syntax (more details in the Appendix). On CLEVR, we train using all the programs in the training set, while restricting the number of paired questions and programs to 1000 (as explained above).

Baselines. We compare against adaptation of a state-of-the-art semi-supervised learning approach proposed for neural module networks by Johnson et al. (2017). Johnson et al. fit the terms corresponding to $q_\phi(\mathbf{z}|\mathbf{x})$ and $p(a|\mathbf{i}; \theta_{\mathbf{z}})$ in our model. Specifically, in question coding, they optimize $\max_\phi \sum_n \log q_\phi(\mathbf{x}^n|\mathbf{z}^n)$ (where n indexes data points with associated questions, *i.e.* the approach does not make use of "unlabelled" data). Next, in module training they optimize $\max_{\theta_{\mathcal{Z}}} \sum_{m=1}^{M} E_{z\sim q(\mathbf{z}|\mathbf{x}^m)} [\log p(a^m|\mathbf{i}^m; \theta_{\mathbf{z}})]$. In joint training, they optimize the same objective with respect to the parameters $\theta_{\mathcal{Z}}$ as well as ϕ, using the REINFORCE gradient estimator: $\nabla_\phi = \frac{1}{M} \sum_m [\log p(a|\mathbf{i}; \theta_{\mathbf{z}}) - B] \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}^m)$, where B is a baseline. In contrast, we follow Algorithm 1 and maximize the corresponding evidence at every stage of training. We also report other baselines which are either ablations of our model or deterministic variants of our full model. See Appendix for the exact setting of the hyperparameters.

5. Results

SHAPES results. We focus on evaluating approaches by varying the fraction of question-aligned programs (which we denote $\%\mathbf{x} \leftrightarrow \mathbf{z}$) in Table 1. To put the numbers in context, a baseline LSTM + image model, which does not use module networks, gets an accuracy of 63.0% on Test

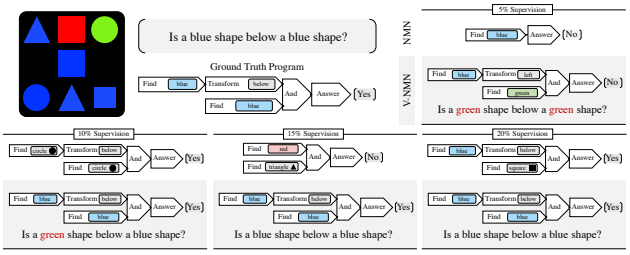

Figure 2. **Data-efficient Legibility:** Image with the corresponding question, ground truth program, and ground truth answer (top left). Predicted programs and answers of different models (NMN in white, Prob-NMN in gray) and reconstructed question (Prob-NMN) for variable amount of supervision. We find Prob-NMN finds the right answer as well as the program more often than NMN (Johnson et al., 2017).

(see Andreas et al. (2016); Table 2). This indicates that SHAPES has highly compositional questions which is challenging to model via conventional methods for Visual Question Answering (Agrawal et al., 2015). Overall, we make the following observations:

- **Our Prob-NMN approach consistently improves performance in data-sparse regimes.** While both methods tend to improve with greater program supervision, Prob-NMN quickly outpaces NMN (Johnson et al., 2017), achieving test VQA accuracies 30-35% points higher for $>5\%$ program supervision. Notably, both methods perform similarly poorly on the test set given 5% program supervision, suggesting this may be too few examples to learn compositional reasoning. Similarly, we see that the program prediction accuracy is also significantly higher for our approach at the end of joint training, meaning that Prob-NMN is right for the right reasons.
- **Our question coding stage greatly improves initial program prediction.** Our Prob-NMN approach to question coding gets approximately double the question coding (program prediction) accuracy as the NMN approach (col 1, question coding). This means that it effectively propagates groundings from question-aligned programs during the coding phase. Thus the initial programs produced by our approach are more legible at a lower amount of supervision than NMN. Further, we also see improved VQA performance after the module and joint training stages which are based on predicted programs.
- **Successful joint training improves program prediction.** In general, we find that the accuracies obtained on program prediction deteriorate when the module training stage is weak (row 1). On the other hand, higher program prediction accuracies generally lead to better module training, which further improves the program prediction performance.

Figure 2 shows sample programs for each model. We see the limited supervision negatively affects NMN program prediction, with the 5% model resorting to simple `Find[X]`$\rightarrow$`Answer` structures. Interestingly, we find that the mistakes made by the Prob-NMN model, *e.g.*, green in 5% supervision (top-right) are also made when reconstructing the question (also substituting green for blue). Further, when the token does get corrected to blue, the question also eventually gets reconstructed (partially correctly (10%) and then fully (15%)), and the program produces the correct answer. This indicates that there is high fidelity between the learnt question space, the answer space and the latent space.

Finally, the N2NMN approach (Hu et al., 2017) evaluates their question-attention based module networks in the fully unsupervised setting, getting to 96.19% on TEST. However, the programs in this case become non-compositional (see Section 3), as the model leaks information from questions to answers via attention, meaning that programs no longer carry the burden of explaining the observed answers. This makes the modules illegible. In general, our approach makes the right independence assumptions ($a \perp \mathbf{x}|\mathbf{i}, \mathbf{z}$) which helps legibility to emerge, along with our careful design of the three stage optimization procedure.

In the Appendix, we show additional comparisons to a deterministic variant of our model (with $\beta = 0$), and study the effect of optimizing the true ELBO.

CLEVR Results. Our results on the CLEVR dataset (Johnson et al., 2017) reflect similar trends as the results on SHAPES (Table 2). As explained in Section 4, we work with 1000 supervised question program examples from the CLEVR dataset (0.143% of all question program pairs). With this, at the end of question coding, Prob-NMN gets to an accuracy of 93.15 $\pm$ 8.61, while the baseline NMN approach gets to an accuracy of 62.47 $\pm$ 9.82. These gains for the Prob-NMN model are reflected in module training, where the Prob-NMN approach gets to an accuracy of 94.42 $\pm$ 3.77, while the baseline is at 79.26 $\pm$ 4.03. At the end of joint training these improve marginally to 95.52 $\pm$ 4.15 and 79.38 $\pm$ 4.21 respectively. Crucially, this is achieved with a program generation accuracy of 93.87 $\pm$ 8.73 by our approach compared to a baseline accuracy of 63.08 $\pm$ 9.91. Thus, the programs generated by the Prob-NMN model are more legible than those by the semi-supervised NMN approach (Johnson et al., 2017). Finally, the same trends are reflected on the CLEVR test set as well, with Prob-NMN outperforming NMN. We refer to the Appendix for more details of the training regime and architectural choices [4].

Coherence and Sensitivity in Reasoning. Next we show that the probabilistic formulation can be used to check a model's coherence in reasoning (see Figure 3, top). Given the image and the answer `yes`, we observe that one is able to generate multiple, diverse reasoning patterns which lead

[4]We find the question reconstruction accuracy is close to zero, since the model often decodes to a semantically equivalent question string which an exact match does not recover, thus omit that number from Table 2."

Table 1. Results (in percentage) on the SHAPES dataset with varying amounts of question-program supervision ($\%\mathbf{x} \leftrightarrow \mathbf{z}$), $\beta = 0.1$.

	$\%\mathbf{x} \leftrightarrow \mathbf{z}$	Validation During Training Stages					Test Accuracy
		[I] Question Coding		[II] Module Training	[III] Joint Training		
		(Reconstruction)	(Program Prediction)	(VQA Accuracy)	(Program Prediction)	(VQA Accuracy)	(VQA Accuracy)
NMN (Johnson et al., 2017)	5	-	9.28±1.91	61.56±3.59	0.0 ±0.0	63.08±0.78	60.06±3.88
Prob-NMN(Ours)		62.86±5.31	17.12±5.28	54.55±11.31	28.12 ±28.12	72.50±8.35	71.95±11.15
NMN (Johnson et al., 2017)	10	-	24.30±2.39	60.31±3.37	6.25 ±10.83	66.51±4.10	61.99±0.96
Prob-NMN(Ours)		83.60±5.57	60.18±9.56	75.80±3.62	90.62±6.98	96.86±2.48	94.53±2.06
NMN (Johnson et al., 2017)	15	-	47.67±5.02	69.47±9.87	0.0 ±0.0	62.43±0.49	61.32±2.36
Prob-NMN(Ours)		95.86±0.20	84.85±6.25	90.57±3.44	95.31 ±5.18	98.40±1.63	97.02±0.84
NMN (Johnson et al., 2017)	20	-	58.37±3.30	66.17±7.02	43.75 ±43.75	80.68±18.00	78.59±19.27
Prob-NMN(Ours)		96.10±0.27	90.22±1.63	91.81±1.58	96.87 ±5.41	99.43±0.61	96.97±1.30

Table 2. Results (in percentage) on the CLEVR dataset with 0.143 % $\mathbf{x} \leftrightarrow \mathbf{z}$, $\beta = 0.1$. Validation metrics are calculated on a set of 20K examples out of CLEVR v1.0 validation split, across 15 random seeds. Test metrics are calculated on CLEVR v1.0 test split, and correspond to the best performing checkpoint across 15 random seeds.

	$\%\mathbf{x} \leftrightarrow \mathbf{z}$	Validation During Training Stages					Test Accuracy
		[I] Question Coding		[II] Module Training	[III] Joint Training		
		(Reconstruction)	(Program Prediction)	(VQA Accuracy)	(Program Prediction)	(VQA Accuracy)	(VQA Accuracy)
NMN (Johnson et al., 2017)	0.143	-	62.47±9.82	79.26±4.03	63.08 ±9.91	79.38±4.21	75.71
Prob-NMN(Ours)		-	93.15±8.61	94.42±3.77	93.87 ±8.73	95.52±4.15	97.73

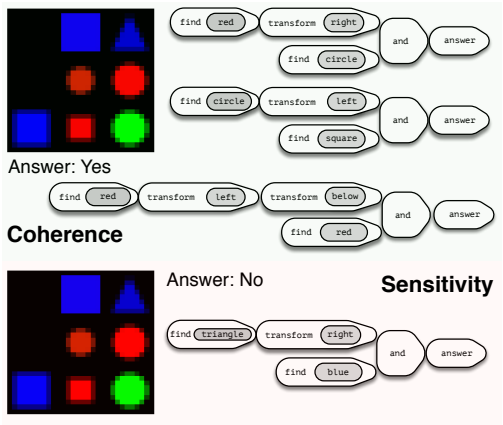

Figure 3. **Coherence and Sensitivity**: Image and a correponding answer are specified, and the model is asked to produce programs it believes should lead to the particular answer for the given image. One can notice that the generated programs are consistent with each other, and evaluate to the specified answer. Results are shown for a Prob-NMN model trained with 20% program supervision on SHAPES.

to the answer, by sampling $\mathbf{z} \sim p(\mathbf{z}|a, \mathbf{i})$, showing a kind of systematicity in reasoning (Lake et al., 2017). On the other hand, when we change the answer to `no` (see Figure 3, bottom), keeping the image the same, we observe that the reasoning pattern changes meaningfully, yielding a program that evaluates to the desired answer. See Appendix for a description of how we do the sampling and results on CLEVR.

6. Discussion and Conclusion

In this work, we discussed a probabilistic, sequential latent variable model for visual question answering, that jointly learns to parse questions into programs, reasons about abstract programs, and learns to execute them on images using modular neural networks. We demonstrate that the probabilistic model endows the model with desirable properties for interpretable reasoning systems, such as the reasoning being clearly legible given minimal number of teaching examples, and the ability to probe into the reasoning patterns of the model, by testing their coherence (how consistent are the reasoning patterns which lead to the same decision?) and sensitivity (how sensitive is the decision to the reasoning pattern?). We test our model on the CLEVR dataset as well as a dataset of compositional questions about SHAPES and find that handling stochasticity enables better generalization to compositionally novel inputs.

References

Agrawal, A., Lu, J., Antol, S., Mitchell, M., Lawrence Zitnick, C., Batra, D., and Parikh, D. VQA: Visual question answering. In *Intl. Conf. on Computer Vision*, May 2015.

Alemi, A. A., Poole, B., Fischer, I., Dillon, J. V., Saurous, R. A., and Murphy, K. Fixing a broken ELBO. In *ICML*, 2018.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural

module networks. In *CVPR*, 2016.

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In *ICML*, 2017.

Bader, S. and Hitzler, P. Dimensions of neural-symbolic integration - a structured survey. November 2005.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Bottou, L. From machine learning to machine reasoning. February 2011.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *Proc. Conf. Natural Language Learning (CoNLL)*, 2016. URL http://arxiv.org/abs/1511.06349.

Das, A., Agrawal, H., Lawrence Zitnick, C., Parikh, D., and Batra, D. Human attention in visual question answering: Do humans and deep networks look at the same regions? In *Proc. Empirical Methods in Natural Language Processing*, 2016.

Das, A., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. Neural modular control for embodied question answering. In *Conference on Robot Learning*, 2018.

Dragan, A. D., Lee, K. C. T., and Srinivasa, S. S. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE International Conference on Human-robot Interaction*, HRI '13, pp. 301–308, Piscataway, NJ, USA, 2013. IEEE Press.

Evans, R., Saxton, D., Amos, D., Kohli, P., and Grefenstette, E. Can neural networks understand logical entailment? February 2018.

Gaunt, A. L., Brockschmidt, M., Kushman, N., and others. Lifelong perceptual programming by example. In *ICLR*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hochreiter, S. and Schmidhuber, J. Long Short-Term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

Hu, R., Andreas, J., Rohrbach, M., Darrell, T., and Saenko, K. Learning to reason: End-to-End module networks for visual question answering. In *CVPR*, 2017.

Hu, R., Andreas, J., Darrell, T., and Saenko, K. Explainable neural computation via stack neural module networks. In *ECCV*, 2018.

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

Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Inferring and executing programs for visual reasoning. In *Intl. Conf. on Computer Vision*, 2017.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *J. of Machine Learning Research*, 37(2):183–233, November 1999.

Keng, B. Semi-supervised learning with variational autoencoders, September 2017. Accessed: 2019-1-14.

Kingma, D. P. and Welling, M. Auto-Encoding variational bayes. In *ICLR*, 2014.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

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

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behav. Brain Sci.*, 40:e253, January 2017.

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

Lu, J., Yang, J., Batra, D., and Parikh, D. Hierarchical Question-Image Co-Attention for visual question answering. In *NIPS*, 2016.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., and Wu, J. The Neuro-Symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019.

Mascharka, D., Tran, P., Soklaski, R., and Majumdar, A. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *CVPR*, 2018.

Miao, Y. and Blunsom, P. Language as a latent variable: Discrete generative models for sentence compression. In *Proc. Empirical Methods in Natural Language Processing*, 2016. URL http://arxiv.org/abs/1609.07317.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518:529, February 2015.

Mudrakarta, P. K., Taly, A., Sundararajan, M., and Dhamdhere, K. Did the model understand the question? In *Proc. ACL*, 2018.

Newell, A. and Simon, H. A. Computer science as empirical inquiry: Symbols and search. *Commun. ACM*, 19(3):113–126, March 1976.

Park, D. H., Hendricks, L. A., Akata, Z., Rohrbach, A., Schiele, B., Darrell, T., and Rohrbach, M. Multimodal explanations: Justifying decisions and pointing to the evidence. In *CVPR*, 2018.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. In *NIPS*, 2017.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Intl. Conf. on Computer Vision*, 2017.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014.

Suzuki, M., Nakayama, K., and Matsuo, Y. Joint multimodal learning with deep generative models. In *ICLR Workshop*, 2017.

Tenenbaum, J. B. j. B. *A Bayesian framework for concept learning*. PhD thesis, Massachusetts Institute of Technology, 1999.

Valiant, L. G. Three problems in computer science. *J. ACM*, 50(1):96–99, January 2003.

Valkov, L., Chaudhari, D., Srivastava, A., Sutton, C., and Chaudhuri, S. HOUDINI: Lifelong learning as program synthesis. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *NIPS*. 2018.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A generative model for raw audio. September 2016.

Vedantam, R., Fischer, I., Huang, J., and Murphy, K. Generative models of visually grounded imagination. In *ICLR*, 2018.

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merrienboer, B., Joulin, A., and Mikolov, T. Towards AI-Complete question answering: A set of prerequisite toy tasks. In *ICLR*, 2016.

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

Yadav, D., Jain, R., Agrawal, H., Chattopadhyay, P., Singh, T., Jain, A., Singh, S. B., Lee, S., and Batra, D. Evalai: Towards better evaluation systems for ai agents. 2019.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. B. Neural-Symbolic VQA: Disentangling reasoning from vision and language understanding. In *NIPS*, 2018.

Yin, P., Zhou, C., He, J., and Neubig, G. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proc. ACL*, 2018.