A Bayesian approach to Visual Question Answering

Gursimran Singh
Department of Computer Science
The University of British Columbia
msimar@cs.ubc.ca

Saeid Naderiparizi
Department of Computer Science
The University of British Columbia
saeidnp@cs.ubc.ca

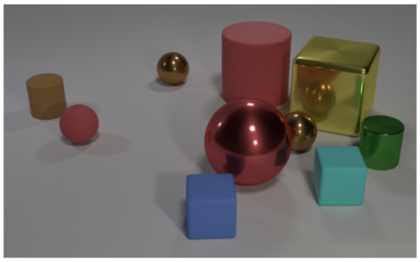
Setareh Cohan
Department of Computer Science
The University of British Columbia
setarehc@cs.ubc.ca

Abstract

Visual question answering (VQA) is a complex task involving perception and reasoning. Typical approaches which use black-box neural networks, do well on perception but fail to generalize due to lack of compositional reasoning ability. Recent promising approaches [1, 2] learn latent representations of queries in the form of programs, facilitating efficient reasoning. On similar lines, we propose to take both image and question into their latent spaces, guided by the intuition that VQA task is much easier in the latent space. We use a generative model of images and questions and perform inference independently on them using inference compilation and importance sampling respectively. Finally, we use a symbolic solver to arrive at an answer. Hence we use an approach which disentangles reasoning from perception. We show the application of our approach on sort-of-CLEVR dataset and achieve state of the art performance.

1 Introduction

The task of visual question answering (VQA) involves developing an artificially intelligent system which can answer questions about images [4, 5]. This is a fairly complex task which involves visual perception (understanding the image and its contents), language understanding (understanding the question) and deductive reasoning (inferring the correct answer from facts) [6]. Typical neural network-based approaches use a feed-forward neural network as a black-box function approximator, trained by maximizing the likelihood of the correct answer given the question and the image [7]. Although successful, these approaches have been argued to exploit dataset-specific biases rather than structured reasoning. This has been corroborated by poor generalization on CLEVR dataset, a diagnostic dataset containing queries ranging from simple perception tasks to complex logical and spatial reasoning tasks[8] (fig. 1 shows a sample from this dataset).



Q: How many objects are either small cylinders or metal cubes?
A: 3

Figure 1: A sample image and a question and its answer from CLEVR dataset[3].

[1] proposes to use a custom neural network architecture, by assembling modules, which induces a specific chain of reasoning required to answer a particular question. In order to build the custom architecture, a program or plan is inferred separately from the question using a seq-to-seq neural

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

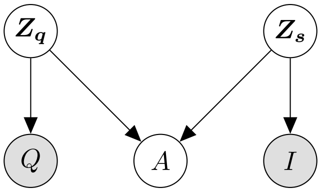


Figure 2: The graphical model for image, question and answer generation process. Z_s and Z_q refer to the scene representation and latent variables of question generation, respectively. Also, Q, A, and I are the program, question, answer and image, respectively.

network. Once we have the custom architecture, we can expect the custom neural network to execute the specific reasoning chain on the image to obtain the answer. Another approach [2] uses a neural network to directly infer the structured scene representation of the image as well. This allows them to obtain the answer by executing the program directly on the scene representation using a deterministic symbolic executor. The symbol manipulation serves as a strong prior and allows more interpretable and robust reasoning. It replaces the neural network where composition and reasoning are hard to model. However, we argue that this can be extended further to completely disentangle perception and reasoning.

We set up a VQA task in a fully Bayesian setting and treat question latents, scene representation and the answer as latent variables fig. 2. Also, we assume the knowledge of the generative process of images, given the images-latents, and the questions, given the question-latents is available. Since the two processes are independent of each other, we can frame this as two independent discrete-latent inference problems. Once, we know the question and image latents, the task of visual question answering simply generates to running a deterministic symbolic solver.

In this particular setting, we arguably are supposed to have better *interpretability*, due to interpretable manipulation of discrete latent variables, better *generalization*, due to composition ability of symbolic solvers, and *less data hungry* due to Bayesian setting and incorporation of generative prior. In order to incorporate these properties, one important design is to treat latent variables as discrete instead of continuous, which makes the problem hard due to high variance of gradients, and non-differentiability of models makes credit assignment of rewards hard. However, we argue that despite being difficult, the current setting of the problem is more interesting to solve. More generally, the setting described in fig. 2 applies to the problem of performing compositional and complex queries on images/ videos/ text/ and many other domains. A progress towards an approach like this will lead to an interpretable, and generalized query solving on latent spaces. Moreover, we show in experiments that we can achieve state-of-art performance in one of the datasets viz sort-of-CLEVR. However, its part of future work to completely analyze this approach, and its limitations by testing it on more datasets.

We have organized the report as follows. We discuss the background and related work in section 2. The detailed explanation of the problem, challenges, and strategy for inference is discussed in section 3. section 4 contains implementation details of our experiments. Additionally, we present a comparison of our approach with existing state-of-the-art methods in table 1. We present the related work in section 5. Since this is an ongoing effort to solve this hard discrete inference problem, we comprehensibly discuss future work in section 6. Finally, we conclude the report in section 7.

2 Background

2.1 Likelihood-Ratio Gradient Estimator

Consider a probabilistic model defined by latents Z and observed variables X. In order to perform an inference task efficiently, we need a high-quality proposal distribution. In variational inference, we use a parameterized family of proposals $q(X; \lambda) = \prod_{x \in X} q(x, \lambda_x)$ and learn the parameters of the proposal by maximizing the evidence lower bound (ELBO).

$$L(\lambda) = \mathbb{E}_{q(X;\lambda)} \left[\log \frac{p(Y, X)}{q(X; \lambda)} \right] \tag{1}$$

In the general case, when $q(x, \lambda)$ is non-reparameterizable, the gradient of the above expectation can be written as

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_{q(X;\lambda)} \left[\nabla_\lambda \log q(X;\lambda) \left(\log \frac{p(Y,X)}{q(X;\lambda)} \right) \right] \tag{2}$$

The above gradient estimator is amenable to Monte-Carlo estimation and we can compute an unbiased estimate of the gradient. However, the gradient estimator exhibits high variance and is often useless in many practical applications. Various gradient reduction techniques are used to reduce the variance of the estimator.

2.1.1 Variance Reduction

The central idea in variance reduction is based in the observation that the expectation of score function $\nabla_\lambda \log q(X;\lambda)$ under proposal distribution is zero. Hence, we can subtract a constant multiple of the score function and still get an unbiased estimate of the gradient. The Monte-Carlo estimator of the gradient can be written as follows:

$$\hat{\nabla}_\lambda \mathcal{L}(\lambda) := \frac{1}{L} \sum_{l=1}^{L} \nabla_\lambda \log q(X^l;\lambda) \left(\log W^l - \hat{b} \right) \tag{3}$$

where $W^l = p(Y^l, X^l)/q(X^l)$, $X^l \sim q(X)$ and $\hat{b}$ is the value that minimizes the variance of the estimator. There are many possible ways to obtain optimal $\hat{b}$ and these techniques come under the name of *baselines* or *control variates*. For instance, decaying average baseline is constructed by maintaining a running average of recent samples of $\log W^l$. Another approach is to construct W^l using only the Markov blanket of a particular latent variable appearing in a particular trace. This technique is called *Rao-Blackwellization*.

2.2 Inference Compilation

Inference compilation [9] is a method of amortized inference in universal Probabilistic Programming Languages. It consists of a recurrent neural network which gets trained to provide better proposals for posterior distributions $p(X|Y)$ of a generative model. The objective function for the network is defined as

$$\begin{aligned} \mathcal{L}(\phi) &= \mathbb{E}_Y \left[KL(p(X|Y), q(X|Y;\phi)) \right] \\ &= \mathbb{E}_{p(X,Y)} \left[-\log q(X|Y;\theta) \right] + \text{const.} \end{aligned}$$

Where q can be factorized to proposal distributions for each of the random variables x_i in X and the type of these proposals is determined by the type of the corresponding distribution in the generative model. For training the model, we generate samples (X, Y) from the joint distribution $p(X, Y)$ by running the generative model forward. This procedure network can be thought of as a supervised online learning task. Note that in this setting we have access to an infinite amount of data, therefore, the model can be as complex as we like since overfitting is not defined here. Moreover, unlike variational inference, inference compilation lets us have almost any kind of generative model i.e. it can include discrete random variables or can have branches based on the value of random variables.

3 Proposed Method

3.1 Problem Formulation

VQA in the Bayesian setting can be visualized using the graphical model in fig. 2. In this model, we have two observed variables, the image and the question, and three unobserved variables, the question-latent, the image-latent, and the answer. The graphical model corresponds nicely to an intuitive understanding of the visual question answering task. Having observed the question-latent and image-latent, both the question-answer and image-answer pairs are independent. It encodes nicely that the question-latent and the image-latent must be sufficient to generate an answer. Similarly,

Algorithm 1 Generative model for images

```
procedure IMAGE-GEN(colors)
    objects $\leftarrow$ empty_list
    num_objects = length(colors)
    for $i$ in $0$:num_objects do
        while True do
            $x_i \sim$ Categorical$(0,$ image_size$)$
            $y_i \sim$ Categorical$(0,$ image_size$)$
            if $(x_i, y_i)$ is valid then
                break
        shape$_i \sim$ Bernoulli(shapes)
        color$_i =$ colors$_i$
        Add object$($color-id $= i,$ center $= (x_i, y_i),$ shape $=$ shape$_i)$ to objects
    image = Render(objects)
    return image
```

Algorithm 2 Generative model for questions

```
procedure QUESTION-GEN(colors)
    c$_i \sim$ Categorical$($length(colors)$)$
    t$_i \sim$ Categorical$(2)$
    st$_i \sim$ Categorical$(3)$
    tmp$_i \sim$ Categorical$(2)$
    question = question$($type $=$ t$_i$, subtype $=$ st$_i$, template $=$ tmp$_i$, color $=$ c$_i)$
    return question
```

the question-latent and image-latent are independent, which means we can separately sample each of these. Hence, we can perform inference on image-latent and question-latent independently and the answer can be obtained deterministically using a symbolic solver. An overview of the inference process is illustrated in fig. 3. Details of the framework will be discussed further in the report.

3.2 Generative Model

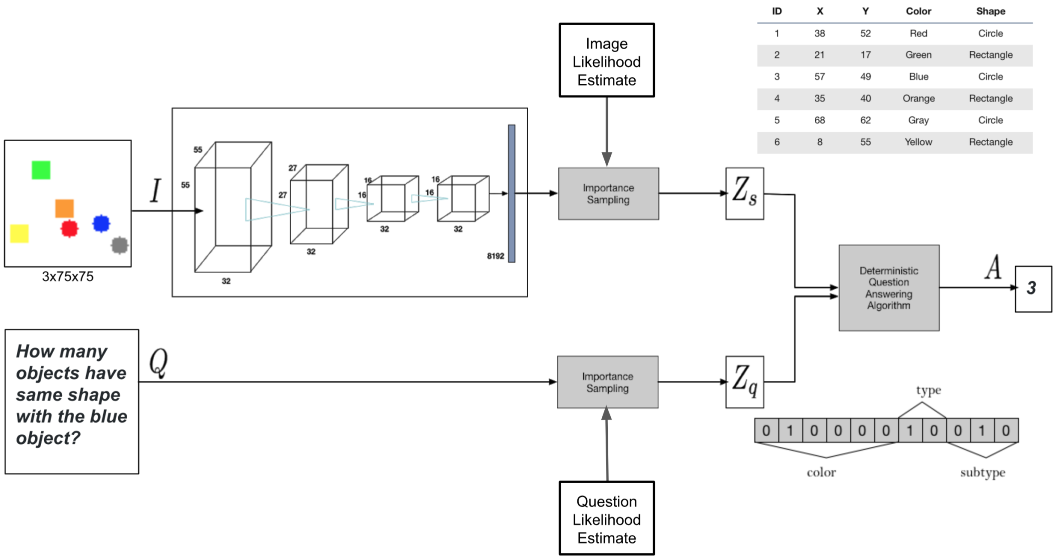
We use the generative models of Sort-of-CLEVR dataset which can be written as independent HOPPL programs. We show the generative model for images algorithm 1 and algorithm 2 respectively.

Image The generative model for images takes the list of colors as input and for each color, it samples an object's location and type. Finally, it renders (link function) an image consisting of all the sampled objects. Additionally, it has a rejection sampling loop, which checks that a sampled center does not overlap any previous objects centers.

Question In case of questions, we sample a question type, a question subtype, and a template. It uses this information to pick a question from a predefined bank of questions available. Next, it samples the color of the object which the question asks about, and replaces the object of interest in the question template, with this object.

3.3 Likelihood

Image We define a likelihood on the output of the generative model algorithm. A basic naive likelihood is obtained by defining Gaussian distributions on individual pixels of the image. However, the resulting distribution turns out to be too peaky (a Dirac distribution). So, if we do importance sampling, the weights will end up being zero, unless our sample hits the mode. To relax this problem, We use Approximate Bayesian Computation (ABC) method [10] [11]. We instead define a kernel $K(\cdot, \cdot)$ which measures the similarity of two observed variables. For each sampled latent X_i we render the corresponding image Y_i and use $K(Y, Y_i)$ as a proxy to the likelihood. Although using ABC method is expected to resolve the problem of highly peaked distributions, the covariance of the kernels remains as a hyper-parameter to be determined.



ID	X	Y	Color	Shape
1	38	52	Red	Circle
2	21	17	Green	Rectangle
3	57	49	Blue	Circle
4	35	40	Orange	Rectangle
5	68	62	Gray	Circle
6	8	55	Yellow	Rectangle

Figure 3: Dataflow diagram to infer an answer

Question Similarly, for questions, we use ABC to calculate the likelihood of a candidate Q^* given a sampled latent Z_q. To this end, Q is first obtained using Z_q, and the embedding vectors of both Q and Q^* are calculated. The similarity between these embedding vectors is used to approximate the likelihood $P(Q^*|Z_q)$ in this case. To obtain the embedding vectors of questions, we can use a pre-trained RNN encoder. However, we found that an existing trained embedding such as Word2Vec [12] works well for our case. We calculate the embedding vector of each question as the average of Word2Vec embeddings of all of its words.

3.4 Inference

Inference corresponds to identifying the latents, having observed the image and the question. As discussed in the introduction, the structure of our graphical model helps us to simplify the inference process. It means that the VQA problem breaks down nicely into two separate and independent inference problems - question-latent-to-question and image-latent-to-image. Specifically, we use importance sampling, variational inference and inference compilation for images. In case of questions, we simply use importance sampling to approximate the posterior.

3.5 Obtaining the Answer

We generate an answer in a deterministic fashion using the image and question latents. We can still take advantage of the uncertainty information in the question and image latents and generate a distribution for the answer. Given the image and question latents $p(\boldsymbol{Z_s}|I)$ and $p(\boldsymbol{Z_q}|Q)$, we sample from these and compute the $p(A|Q,I)$ as follows

$$p(A|Q,I) = \int\int p(A,\boldsymbol{Z_s},\boldsymbol{Z_q}|Q,I)d\boldsymbol{Z_s}d\boldsymbol{Z_q} = \int\int p(A|\boldsymbol{Z_s},\boldsymbol{Z_q})p(\boldsymbol{Z_s}|I)p(\boldsymbol{Z_q}|Q)d\boldsymbol{Z_s}d\boldsymbol{Z_q} \tag{4}$$

$$\approx \frac{1}{n}\sum_{i=1}^{n} p(A|\boldsymbol{Z_s}^{(i)},\boldsymbol{Z_q}^{(i)}),\ \boldsymbol{Z_s}^{(i)} \sim p(\boldsymbol{Z_s}|I), \boldsymbol{Z_q}^{(i)} \sim p(\boldsymbol{Z_q}|Q) \tag{5}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \delta(A = f_{ans}(\boldsymbol{Z_s}^{(i)},\boldsymbol{Z_q}^{(i)})),\ \boldsymbol{Z_s}^{(i)} \sim p(\boldsymbol{Z_s}|I), \boldsymbol{Z_q}^{(i)} \sim p(\boldsymbol{Z_q}|Q) \tag{6}$$

where f_{ans} is used to infer A from $\boldsymbol{Z_q}$ and $\boldsymbol{Z_s}$ deterministically.

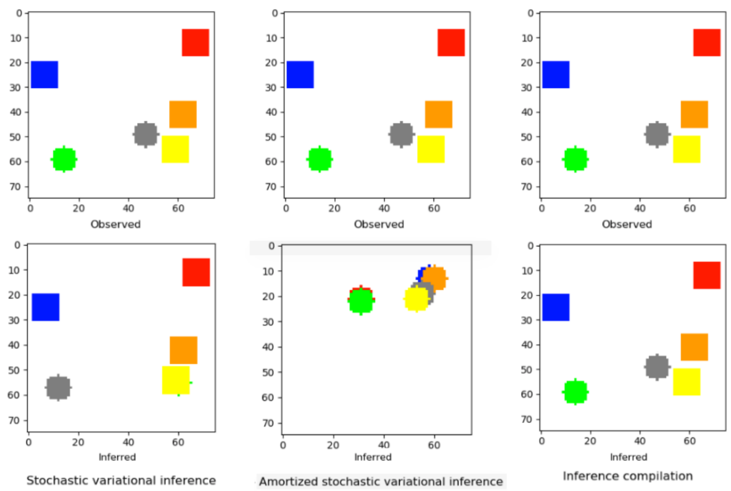


Figure 4: Samples of different inference algorithms - The leftmost charts show the observed image (top) and inferred image (bottom) when we train the inference network on a single image. We see that it kinds of work but does not work perfectly. The chart in the middle shows inference results when we train the inference network in an amortized manner. As seen in the image, the inference network simply learns to maximize the white area. The rightmost plots show inference results of inference compilation. As seen in the inferred image, it achieves perfect reconstruction of the observed image.

4 Experiments

4.1 Setup

As explained in section 3, we run separate experiments on two independent inference problems - image and question. Since our generative models do not correspond to a finite graphical model, we write them in higher order probabilistic language (HOPPL) which allows automatic inference and symbolic reasoning. In particular, we use both *Pyro* [13], which is good for doing variational inference and *PyProb*, which supports inference compilation in a variety of models. We run all our experiments on an 8-core machine and GTX-GeForce-TitanX GPU having 16 Gb of internal memory. For the likelihood of images, we trained a Variational Auto Encoder (VAE) on the images which has a 64-dimensional hidden space and the encoder predicts mean and diagonal covariance matrix of a Normal distribution. For questions, we use the word2vec likelihood, which we explain in section 3. For evaluation, we use both visual inspection and quantification. In order to make sure that inference networks are trained, we check whether negative ELBO (or surrogate loss with variance reduction) decrease substantially and converge to a minimum. To test the quality of the proposal, we take random samples and visually inspect them in relation to the observed image. We use a similar procedure in case of questions. To test the quality of the answers, we test our system on a held-out test set. We calculate the accuracy, i.e the number of correctly predicted answers. We discuss the individual techniques, challenges, results, and conclusions in the following subsections.

4.2 Renderer

In our generative model, we convert the latents of images using a deterministic computation process, called renderer. Originally, we planned to use the CLEVR dataset, however, the renderer it uses takes around 5 seconds per image. Since we use a generative model in the variational inference and inference compilation, we need to generate samples in an online manner. Although after some optimization, we were able to reduce the generation time to 2s but this was still slow and it turned out to be a rate-determining step in running experiments. Hence, we used a simplified version of CLEVR renderer, known as Sort-of-CLEVR. This renderer uses OpenCV library to render images and provided a speedup by a factor of 15k. In case of questions, the renderer is a pretty straightforward template-based system which is pretty fast for experiments.

Model	Relational questions	Non-relational questions
Relational Networks (Santoro etal)	96%	98%
CNN + LSTM (Santoro etal)	60%	99%
Our approach	100%	100%
Our approach (with ground truth question-latents)	100%	100%
Our approach (with ground truth image-latents)	100%	100%

Table 1: Accuracy of answers obtained by various methods.

4.2.1 Sort-of-CLEVR

Sort-of-CLEVR[14] consists of images of 2D colored shapes along with questions and answers. Each image has a total of 6 objects, which can be either a square or a circle. There are 6 colors of red, blue, green, orange, yellow and gray to unambiguously identify each object. In this dataset, there are two types of questions: relational and non-relational. Relational questions ask about objects in relation to a particular object of interest, while non-relational questions ask questions only about the object of interest. Each question type consists of three subtypes each of which can take the object of interest using its color as input. For the question generation, for each subtype, we added two different templates by restating the question differently to increase the number of different questions in total.

4.3 Image Inference

4.3.1 Stochastic Variational Inference

In order to perform inference in an amortized manner, we train a parameterized proposal distribution. Since our latent variables are non-reparameterizable, we use the likelihood-ratio gradient estimator eq. 3. For our proposal network, we use four CNN layers with 32 filters and kernel sizes of (4,5,2,3). The resultant embedding is passed through three separate fully connected layers to generate the latent variables X, Y and type of objects for each color. For training, we use stochastic gradient descent using an Adam optimizer with a learning rate of 0.001. We use a batch size of 64 and 20 particles. We used the gradient estimator eq. 3 in our experiment which helps reduce the variance of gradient using *Rao-blackwellization* and *decaying-average-baseline*. We found that variance reduction makes a huge difference in the speed and effectiveness of training. We ran the experiments in two settings - single-image and amortized fig. 4. In the single-image setting, we used the same image for optimizing the proposal network. We found that it does not work consistently but still does an okay job in inferring the correct location and types of objects in an image. As shown in the figure (leftmost), it completely misses one object but is able to correctly identify locations of five objects. Also, it misrepresents the green object and places a grey object in its place. On the other hand, we found that the amortized inference network fig. 4 (middle) simply learns to place objects one over the other and tries to maximize the white area. We posit that this is due to sparse rewards in the image space. The odds of sampling the correct configuration of all latents is rare. The sub-optimal configurations, though more likely, have troubles with credit assignment to specific latents. Hence, on average, the network just learns to collapse in a sub-optimal solution.

4.3.2 Inference Compilation

In the case of inference compilation, we use a similar inference network as described in the section 4.3.1. While training the inference network, we use the batch size of 10 and run it for 40K steps. For optimization, we use an Adam optimizer for stochastic gradient descent with a learning rate of 0.0001. During test time, we use the trained inference network as a guide and get samples using 100 importance weighted 100 traces. We show our results in the table 1. The row 4 of the table, shows the accuracy of answers using the image-latents obtained by inference compilation and ground-truth question-latents. The high accuracy obtained suggests the quality of our latents obtained after importance sampling. The fact that only 100 importance weighted samples were required to get good latents, suggests the quality of our proposals.

4.4 Question Inference

In case of questions, we have 2 different types, 3 subtypes, 2 templates and 6 colors which adds up to $2 * 3 * 2 * 6 = 72$ distinct questions. Since this space of latents is small, we can ideally enumerate this for inferring each question. However, we found that importance sampling with 100 samples works good or us. Specifically, given a question-latent $\boldsymbol{Z}$, we get a question Q using the generative model. Then, to embed questions Q^* and Q, we take the average Word2Vec embeddings of the words. Our measure of similarity between Q^* and Q is then defined as the log-likelihood of a Normal distribution with a mean of embedded Q^* and standard deviation (σ = 0.0001) given embedded Q. In table 1, row 5 corresponds to the accuracy of answers using inferred question-latents and ground truth image-latents. The high accuracy depicts the quality of inferred question latents.

4.5 Answer Inference

As explained in section 3, we obtain an answer by sampling the question-latent and image-latent and running them through a deterministic answer solver algorithm 3. However, since our distributions of image latents and question latents are peaked, we obtain a peaked distribution in the answer as well. We report the final accuracy of our system in row 3 in table 1. Also, we report the accuracy of the state-of-the-art approach, relational networks [14] in row 1. We also report the accuracy of a baseline deep learning model which uses a CNN+LSTM for VQA task in row 2. As can be seen in the table, our approach beats the state-of-the-art method by 3%. This can be attributed to the fact that we disentangle perception (inferring latents of an image) and reasoning (arriving at an answer). Since neural networks are good for perception, we use a CNN to infer the latent variables of the image. For reasoning, to arrive at an answer, we do not use a neural network and instead use a deterministic symbolic solver which has a better composition and act as an effective prior. On the other hand, previous state-of-the-art approach [14], required a specialized deep learning system to both learn the perception and also reason to arrive at the answer.

Algorithm 3 Question answering

```
procedure ANSWER-GEN(image_latents, question_latents)
    objects ← image_latents
    target ← objects[color]
    drive (type, subtype, color) from question_latents
    if type == 'non-relational' then
        if subtype == shape query then
            return shape(target)
        else if subtype == horizontal location query then
            if $X(target) < image\_size/2$ then
                return 'yes'
            else
                return 'no'
        else
            if $Y(target) < image\_size/2$ then
                return 'yes'
            else
                return 'no'
    if type == 'relational' then
        if subtype == closest query then
            obj ← closest(target, objects)
            return obj
        else if subtype == furthest query then
            obj ← furthest(target, objects)
            return obj
        else
            cnt ← count(target, objects)
            return cnt
```

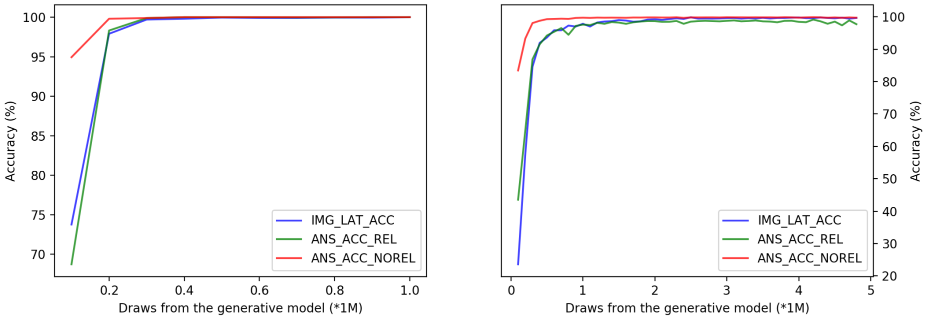


Figure 5: Accuracy of the model while training the inference compilation network. IMG_LAT_ACC is the accuracy of finding the groundtruth latent variables of the observed image, ANS_ACC_REL is the accuracy of the answers to the relational questions and ANS_ACC_REL is the accuracy of the answers to the non-relational questions. The accuracy is computed based on a test set of 2000 images and 10 relational and 10 non-relational questions for each image. Left: posterior samples obtained by performing importance sampling with 100 samples. Right: posterior samples obtained by drawing one sample from the proposal. The left plot is cropped after 1 million samples since it reaches perfect accuracy after that point.

5 Related work

Visual Reasoning VQA is a good means of gauging our progress on visual reasoning and perception. A number of neural network-based approaches [15, 16, 17] use a CNN and an RNN to encode the image and the question respectively. These embeddings are concatenated and are fed into a multi-level perceptron which samples an answer from the discrete distribution over the vocabulary. The whole network is trained end-to-end by maximizing the likelihood of the correct answer. Current leading approaches selectively attend to the image [18], question [19] or co-attend to both [20]. However, recent works [1, 8] show that such works exploit dataset biases rather than doing actual visual reasoning. Particularly, [3] proposed a diagnostic dataset (CLEVR) of synthetically generated images, questions, and answers to benchmark the progress of exiting VQA models on various types of queries on images ranging from simple perception tasks to complex tasks involving logical reasoning, spatial reasoning, etc. Subsequent proposed models which combine attention and construct question specific architecture are known to perform well on various reasoning benchmarks [21, 22]. In both approaches, the central idea is to transform the space of inputs into a structured latent representation which is more amenable to visual question answering. Although many of these approaches perform well on CLEVR dataset [3], it can be argued that they have limited generalization power. This is due to the fact that they learn direct mappings (point estimates in latent space) without any quantization of uncertainty and hence the latent space will be unstructured. [23] models the program as a latent variable in the question generation process guided by the intuition that many possible questions have the same underlying structured latent program. They use probabilistic inference and model the uncertainty associated with a program which results in more interpretable latent space. However, their approach does not model images as a generative process and instead uses module networks to perform feed-forward inference to obtain the answer.

Program Induction Program induction or searching is the field in which we synthesize a program given some specification like IO examples, mathematical constraints, language description, or a question (in our case). [1, 2] uses a simple seq-to-seq LSTM-based network to go from the text of a question to the program used for answering the question. Recent approaches [24] propose a two-step approach where a specification encoder, encodes the question and a program generator uses a recursive neural network to incrementally generate program trees. [25] uses a Bayesian approach to automatically synthesize single dimensional samplers. They first specify a prior over the grammar and compute the likelihood of a generated candidate using summary statistics. We will use a similar approach for our question generative model except the fact that we learn to generate a program using a language specification instead of IO pairs.

Structural Scene Representation - Structured scene representation is the field of identifying latent variables which explain various factors of variation in an image. [26] proposes a variational autoencoder (VAE) based technique to infer the pose and lightening of a face. [27] trained an encoder to output a structured scene representation which is trained using the renderer as a decoder. While these approaches use feed-forward neural networks, our approach uses probabilistic inference for scene de-rendering, which is similar in spirit to [28]. However, it uses a β-VAE framework which uses neural networks to approximate both the generative and inference networks.

6 Future Work and Discussion

We demonstrated an application of a Bayesian VQA system. We split the problem as two separate inference problems over discrete latent variables. Inference compilation worked well to invert the graphical model and get a decent proposal. However, we faced challenges with importance sampling due to peaked distribution. As part of future work, we would like to define better likelihood which attributes high likelihood to images with close object placements.

In case of questions, we were simply able to use importance sampling to get perfect latents. It was due to the small space of latents. However, as part of future work, we would like to define more complex questions and treat question-latent as a general program in a domain specific language. This would allow us to express more powerful questions, at the cost of complicating the inference.

Another interesting line of future work is to extend this framework on more complicated CLEVR dataset [3]. Since the two datasets are similar, conceptually we should be able to extend this framework and get results for CLEVR. However, in CLEVR we have more complicated questions which require us to model them using programs. Also, the perception task will be hard since the images are 3D rendered objects.

7 Conclusion

We implemented a VQA system in the Bayesian setting. We also achieved state-of-the-art performance on sort-of-CLEVR dataset. Previous approaches try to force neural networks to perform both reasoning and perception and hence are less generalizable, require more data, and perform worse than ours. We used a neural network for what they are good at i.e perception and symbolic reasoning for what its good for, i.e reasoning. In this sense, we took the best of both the worlds. However, we do not claim to solve the general problem of VQA. Our approach assumes the knowledge of generative models, which are not readily available for many practical settings. Also, we show results only on sort-of-CLEVR dataset, which consists of simple images and questions. As part of the further work, one would want to extend this framework and evaluate its performance on other datasets.

References

[1] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross B Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, pages 3008–3017, 2017.

[2] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *arXiv preprint arXiv:1810.02338*, 2018.

[3] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[5] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in neural information processing systems*, pages 1682–1690, 2014.

[6] Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018.

[7] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Vqa: Visual question answering. *International Journal of Computer Vision*, 123(1):4–31, 2017.

[8] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, volume 1, page 3, 2017.

[9] Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. *arXiv preprint arXiv:1610.09900*, 2016.

[10] George Karabatsos, Fabrizio Leisen, et al. An approximate likelihood perspective on abc methods. *Statistics Surveys*, 12:66–104, 2018.

[11] SA Sisson, Y Fan, and MA Beaumont. Overview of approximate bayesian computation. *arXiv preprint arXiv:1802.09720*, 2018.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[13] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *arXiv preprint arXiv:1810.09538*, 2018.

[14] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[15] Mengye Ren, Ryan Kiros, and Richard Zemel. Image question answering: A visual semantic embedding model and a new dataset. *Proc. Advances in Neural Inf. Process. Syst*, 1(2):5, 2015.

[16] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in neural information processing systems*, pages 2296–2304, 2015.

[17] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*, 2016.

[18] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4995–5004, 2016.

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

[20] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016.

[21] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

[22] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pages 2397–2406, 2016.

[23] Anonymous. Probabilistic neural-symbolic models for interpretable visual question answering. In *Submitted to International Conference on Learning Representations*, 2019. under review.

[24] Jacob Devlin, Rudy R Bunel, Rishabh Singh, Matthew Hausknecht, and Pushmeet Kohli. Neural program meta-induction. In *Advances in Neural Information Processing Systems*, pages 2080–2088, 2017.

[25] Yura Perov and Frank Wood. Automatic sampler discovery via probabilistic programming and approximate bayesian computation. In *Artificial General Intelligence*, pages 262–273. Springer, 2016.

[26] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.

[27] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *Proc. CVPR*, volume 2, 2017.

[28] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.