The Bayesian Babes Present...
Bayesian Visual Question Answering
Replication and Analysis

Ryan Marten
1003813279
University of Toronto
ryan.marten@mail.utoronto.ca

Conor Vedova
1004405672
University of Toronto
conor.vedova@mail.utoronto.ca

Ben Prystawski
1004240877
University of Toronto
ben.prystawski@mail.utoronto.ca

Mark Abdullah
1004002802
University of Toronto
mark.abdullah@mail.utoroto.ca

Abstract

We implement a Bayesian framework for Visual Question answering and test how generalizable it is. The framework is based on inferring latent representations of images and questions, then using a deterministic symbolic solver to answer the question given both latent representations. Their system relies on detailed knowledge about the structure of the dataset they used, so we investigate whether it is robust to varying the wordings of the questions and a few pixels in the images. We find that question inference is somewhat robust to small changes (synonym word substitutions) to existing question types, but is unable to infer latents on related but unseen questions. Further, we were unable to reproduce the results for image inference, with the main difficulty of training an adequate VAE to encode images. Our implementation code can be found on Github [1].

1 Introduction

Visual question-answering is a challenging machine learning task requiring multiple competencies. To perform well, models need to interpret both images and questions and understand how they relate to each other. While several end-to-end neural network architectures have been designed to solve this problem [13], they struggle to learn the intended abilities [5]. This is due to the networks learning statistical shortcuts to answering questions, rather than learning an understanding of visual and linguistic reasoning.

Singh et al. [14] propose a Bayesian setup for the VQA task in order to explicitly disentangle the visual perception and language understanding tasks, with the goal of increased interpretability and generalizability. The Bayesian approach is applied to a simple toy dataset, sort-of-CLEVR [13], which consists of variations of colored shapes on a white background. However, Singh et al. [14] require a rigid probabilistic model that has knowledge of the structure of the questions and images in the sort-of-CLEVR dataset, so it is not a general-purpose framework. They leave open the challenge of applying this approach to more varied and complex images and questions. We investigate how robust the approach is even when limited to the simple domain of sort-of-CLEVR images and questions by perturbing the wordings of questions and measuring the effect on performance.

[1] https://github.com/RyanMarten/BayesianVisualQuestionAnswering

2 Background and Related Work

2.1 VQA: Visual Question Answering

Visual Question Answering is the task of taking an input image and an input natural language question about the image and outputting a natural language answer to the question [2]. This has been a highly active area of research and it is one of the leading tasks in multi-modal machine intelligence [6, 11]. A general approach is based on deep embeddings of the image and question followed by an fully connected neural network, yielding a final distribution over possible answers. Flaws in the datasets allow this simple approach achieve high accuracies. After studying the distribution of their dataset, it was found that most questions are answered in with "yes/no", a single word or a number [2]. This is exacerbated by current metrics that enable models to leverage biases in the distributions of datasets. This motivates new approaches to the VQA task that build in prior structure or knowledge that encourages interpretable reasoning.

2.2 VAE: Variational Autoencoder

VAEs are a class of autoencoders that introduce a Gaussian probability distribution over the latent representation variables [7]. They enable us to encode data into a latent space which represents a meaningful distribution over possible data points.

2.3 Probabilistic Programming

Higher-order probabilistic programming languages (HOPPLs) are computational frameworks for representing distributions and performing inference [15]. We chose to use *PyProb* [2] because defining the generative model is as simple as directly replicating the sort-of-CLEVR dataset code. Importantly, *PyProb* also implements a convenient API for training an inference compilation network and selecting the network to used during importance sampling. [3]. Generally, probabilistic programming languages make it easier to write Bayesian models and contain ready-to-use implementations of popular inference methods, such as Markov chain Monte Carlo (MCMC).

2.4 Importance Sampling

Importance sampling is a method used to make inferences on an unknown distribution easier. In importance sampling for MCMC, samples are generated from a biased distribution that centers around a more meaningful area of interest, and then these samples are re-weighted to correct for the biased sampling.

2.5 Inference Compilation

Inference compilation is a method of inference for probabilistic models that involves a computationally expensive ‘compilation’ step that allows for more efficient ‘inference’ step [10]. The cost of the compilation step is amortized over all future inference steps. During the compilation step, a neural network learns to suggest better samples from a proposal distribution, after being trained on data from a generative model specific using a probabilistic programming language. During the inference step, these improved samples are used in an importance sampling method to quicker infer the target distribution by sampling areas of interest more frequently.

3 Formal Description

Our model takes in two inputs: images from sort-of-CLEVR and questions. The images are two-dimensional and consist of six objects, each with a different colour. Objects can be either circular or rectangular and are located anywhere on the 75×75 image. There are 36 possible questions about the image. Half of the questions are relational, meaning they require the model to parse two images, such as “What shape is the object that is farthest from the orange object?”. The other half of questions are non-relational, meaning do not require knowledge of relationships between objects. They only

[2]https://github.com/pyprob/pyprob

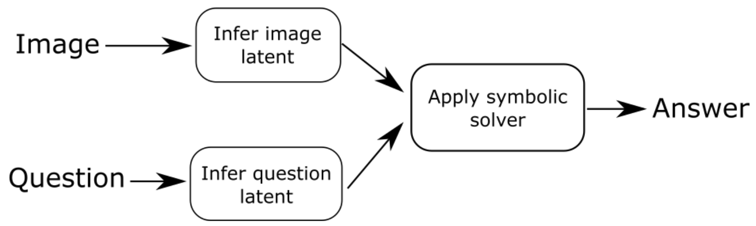

Figure 1: Overview of Bayesian Visual Question Answering task. Latent representations are inferred from images and questions independently, then fed into a symbolic solver which uses them to output an answer.

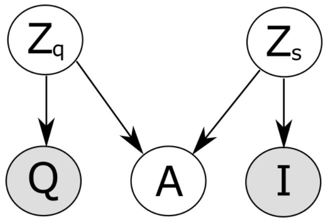

Figure 2: Graphical model of the elements of the VQA task. Shaded circles represent observed variables and white circles represent latent variables. Z_q and Z_s are question and image latents respectively, while Q is the question, I is the image, and A is the answer.

involve one object ("What shape is the green object?"). The overall structure of this model is shown in Figure 1, while a graphical model representing the latent variables is shown in Figure 2. The latent variable image was initially introduced by Singh et al. [14].

We know the structure of the questions and images, so we can explicitly represent them with latent variables. Image latents consist of tables whose rows contain the colour, X and Y positions, and shape of each object in the image. Question latents are binary strings that uniquely specify the colour of the object being asked about and the specific type of question. If we know both the question and image latents, it is easy to answer the question using a series of if statements and simple algorithms that produce the answer.

Therefore, the main difficulty of our problem is inferring the image latents from the images and inferring the question latents from the questions. Singh et al. [14] do this inference using generative models of images and questions as described in Algorithm 1 and Algorithm 2, respectively.

3.1 Inferring Question Latents

We infer the correct question latent using importance sampling, which involves sampling from one distribution then re-weighting the samples by the ratio between their probability in the distribution we sample from and their probabilities in the distribution we want to approximate. In particular, we sample from the prior distribution described in the generative model of questions, then re-weight based on the likelihood of the given question under that question latent. We compute this likelihood by averaging word2vec [12] embeddings for the observed question and a particular candidate question, then using the likelihood of the candidate question under a Gaussian distribution with a diagonal covariance matrix whose diagonal contains 0.001 at every value. In practice, this results in one image latent (the true image latent) with extremely high probability and all other image latents with extremely low probability, so we can simply take one sample and treat it as the inferred latent.

3.2 Inferring Image Latents

The first step of inferring image latents from images is to create a proposal distribution to sample from. This proposal distribution is created using inference compilation, which is described above in the related works section 2. This proposal distribution returns image latents when sampled. These image latents are then used to generate the actual images. During inference, we perform importance sampling using these proposed images and the input image. The likelihood metric that we use here must be well-defined. If we assume that the images' pixels are each distributed by a Gaussian, we get

Algorithm 1: Generative Model of Images

```
objects ← []
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
        end
    end
    shape$_i \sim$ Bernoulli(shapes)
    color$_i \sim$ colors$_i$
    Add object(color-id=$i$, center=$(x_i, y_i)$, shape=shape$_i$) to objects
end
image = Render(objects)
return image
```

Algorithm 2: Generative Model of Questions

```
$c_i \sim$ Categorical(length(colors))
$t_i \sim$ Categorical(2)
$st_i \sim$ Categorical(3)
$q$ = question(type=$t_i$, subtype=$st_i$, color=$c_i$)
return $q$
```

a far-too-peaky distribution. Instead, we need to have some better notion of measuring likelihood. The original paper uses a pre-trained variational autoencoder to encode both the input image and the proposed image into a semantically meaningful latent space. Within this latent space, two encoded objects should be close if the original images are similar in nature. We encode both the input image and the proposed image. Then, we look at the likelihood of the mean of the latent representation of the proposed image under the Gaussian distribution defined by the mean and log-sigma attained from encoding the input image.

3.3 Symbolic Solver

After image latents and question latents are inferred, they are passed into a *symbolic solver*. This solver is a function that first identifies the relevant question type based on its latent representation, then uses a simple algorithm for solving that type of question. It returns the answer to the question as specified by its latent representation.

4 Demonstration

In this section, we report how the Bayesian VQA model performs when we vary the images and questions we give it. Singh et al. [14] report 100% accuracy on the sort-of-CLEVR dataset, but as we show below, this is only because they hard-code knowledge about the structure of the images and questions that prevents the model from generalizing to new questions effectively.

4.1 Variational Autoencoder

As the original paper suggests, we implement a variational autoencoder to measure the likelihood of proposed images against the input image. The reason we use a variational autoencoder here is because comparing similarities between images is difficult. The natural method of assuming each pixel is represented by a Gaussian distribution is far too peaky. Rather, we embed each image into some semantically meaningful latent space where we can compare the embeddings easily. Variational autoencoders are perfect for this process because their latent space is incentivized to follow a Gaussian

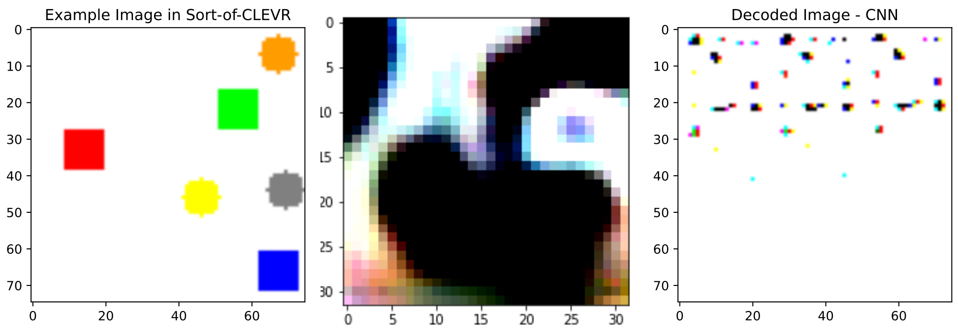

Figure 3: Example image from sort-of-CLEVR (leftmost) and reconstructions from two VAEs: the pre-trained VAE (centre), and the generic CNN VAE (right)

distribution so that the VAE must find some meaningful encoding for the data. This meaningful encoding implies that two embedded objects that are close to each other (using euclidean distance) within the latent space are similar to each other. So, if two images are close to each other within the embedded space, then we would assign a high likelihood to the proposed image.

We designed a multitude of variational autoencoders. However, we were not able to achieve one that faithfully reconstructs images because of the limits on hardware and the difficulty in finding a proper architecture for the dataset. We tested a couple of architectures which are described below.

Figure 3 also presents an example of a reconstructed image of a pre-trained VAE using ResNet layers, trained on CIFAR10 [8], as well as an example of an image reconstructed by a VAE with mostly convolutional layers. The corresponding images are the centre and right images, respectively. As one can see, these VAEs do not perform well with the data provided.

However, we chose the pre-trained VAE and used the encoder to embed the input image and proposed images. The likelihood of each proposed image was then calculated as the likelihood of the embedded image given the distribution is the Gaussian distribution with mean and log-sigma attained from encoding the input image. This likelihood was then used for importance sampling.

4.2 Differently-Worded Equivalent Questions

We tested how robust the word2vec likelihood method is by designing semantically equivalent questions with minor differences in wording. For instance, instead of “Is the red object on the left?” we asked “Is the red object on the left-hand side?” and tested whether the model infers the same question latent. The model infers the likelihood of a question latent given a question in text based on the distance between the average word2vec embedding of the “canonical” question for that latent and the average embedding of the given question text. Therefore, they rely on the assumption that the mean word2vec embeddings for semantically equivalent questions will be similar. ***Appendix A*** contains the full list of original questions and question variants we used.

We find that the model recovers the true question latent 25 out of 36 times, or 69.4%. This is much better than chance, since there are 36 possible questions, but also much worse than the 100 percent accuracy in inferring question latents achieved when only the canonical questions for each latent were used [14].

4.3 Unseen Questions

Another way to test how well this model generalizes is to ask it questions that it was not explicitly designed to answer. We created 6 questions that are similar, but semantically different from the questions used by Singh et al. [14]. For instance, instead of asking “What shape is the object furthest from the green object?” we ask “What colour is the object furthest from the green object?” then infer a question latent and feed it into the symbolic solver along with a randomly-chosen image latent. The full list of new questions, along with accuracy for each question, is reported in ***Appendix B***.

We sampled 1000 image latents from the generative model of images, then used those latents along with the inferred latents for the new questions. By generating and feeding in image latents directly, we assume that we can perfectly infer the correct image latents in this experiment. This is suitable for our purposes since we are interested in how well the model can generalize to different *questions*, not different images. Averaging across image latents, the model got $5.9/21$ questions right, or 28%. When it got a question right, it did so largely by chance because the symbolic solver contains no algorithms to answer the new questions correctly.

This illustrates a major flaw in the model: if we want it to answer questions other than the 36 we explicitly designed it to answer, we have to hand-design the question latents and add code to the symbolic solver to support those question types. The framework has no way of flexibly generalizing to new question types.

5 Discussion and Limitations

In our model comparison and analysis, there are many factors that must be taken into consideration. A major factor to consider is that this model is given the generative process of the data whereas other models have to learn it. This can limit our methods for comparing models if not taken into consideration. When we consider the experiment setup in the original paper, 100% accuracy is expected. This is potentially due to a lack of complete accuracy measurement performed by the original authors. Additional experiments should have been performed such as attempting to remove parts of the generative model or increase the variability in the data set to include data outside the generative space of our generative model, as we have done in our experiments. The authors claim their model has the ability to generalize, but when tested on new questions and images, the model does not perform well.

The choices of likelihood functions in the original paper include word2vec similarity and similarity in the VAE latent space. These were not well discussed and justified, making it hard to implement then as little detail was provided. It is a critical part of the analysis which we believe should be explained in more detail as it is also an interesting part of the model. This shows how hard a task inference can be even when given complete knowledge of the generative process. The idea of comparing a proposal and observations is non-trivial. This is a limitation of this approach.

While the scope of this model is limited, it does provide an interesting probabilistic approach that may be useful for developing higher-level models. The explicit representation of image and question latents makes it easy to disentangle whether a particular testing example was mis-identified due to a visual error, a linguistic error, or both.

Future work might make this framework more general by incorporating Bayesian Program Induction, which is a method for inferring the program that generated given data [9, 4]. By inferring generative models of images and questions and the structure of their latent representations rather than hard-coding them, we might be able to apply this Bayesian approach to a wider range of visual question answering tasks.

6 Conclusion

The method proposed by Singh et al. [14] is an interesting departure from current VQA models and is a promising direction for more interpretable and generalizable visual question answering. In particular, seperating inference of the image latents and questions latents intially to gain explicit distributions over image and question concepts allows for easier investigation into the model's understanding and reasoning (or lack thereof). The idea of disentangling vision and language has been presented before, [1] and there is still lots to explore.

The simplest Bayesian setup however provides many challenges that the authors do not fully address. First, the setup is not very robust to changes in the input image and questions, shown by our experiments in section 4. Second, the non-trivial choices in setup are not fully justified, such as the use of a VAE for encoding images when caclulating likelihoods of proposals versus the observed image. For future work, a more principled method of comparing this method to other VQA models on simplistic toy datasets should be devised and adjustments to the method should be made so it can be tested on the real-world data found in common VQA benchmarks.

References

[1] S. Amizadeh, H. Palangi, O. Polozov, Y. Huang, and K. Koishida. Neuro-symbolic visual reasoning: Disentangling "visual" from "reasoning", 2020.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[3] A. G. Baydin, L. Shao, W. Bhimji, L. Heinrich, L. F. Meadows, J. Liu, A. Munk, S. Naderiparizi, B. Gram-Hansen, G. Louppe, M. Ma, X. Zhao, P. Torr, V. Lee, K. Cranmer, Prabhat, and F. Wood. Etalumis: Bringing probabilistic programming to scientific simulators at scale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC19), November 17–22, 2019*, 2019.

[4] K. Ellis, C. Wong, M. Nye, M. Sable-Meyer, L. Cary, L. Morales, L. Hewitt, A. Solar-Lezama, and J. B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv preprint arXiv:2006.08381*, 2020.

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

[6] K. Kafle and C. Kanan. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20, 2017.

[7] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[8] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[9] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[10] T. A. Le, A. G. Baydin, and F. Wood. Inference compilation and universal probabilistic programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 1338–1348, Fort Lauderdale, FL, USA, 2017. PMLR.

[11] S. Manmadhan and B. C. Kovoor. Visual question answering: a state-of-the-art review. *Artificial Intelligence Review*, 53(8):5705–5745, 2020.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[13] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

[14] G. Singh, S. Naderiparizi, and S. Cohan. A bayesian approach to visual question answering. 2018.

[15] J.-W. van de Meent, B. Paige, H. Yang, and F. Wood. An introduction to probabilistic programming, 2018.

Appendix A: Canonical Questions and their Variants

Original Question	Variant Question	Accuracy
What shape is the [colour] object?	What is the shape of the [colour] thing?	6/6
Is the [colour] object on the left?	Is the [colour] object on the left-hand side?	2/6
Is the [colour] object on the top?	Is the [colour] object on the top of the screen?	2/6
What shape is the object closest to the [colour] object?	What shape is the object with the least distance to the [colour] object?	6/6
What shape is the object furthest from the [colour] object?	What shape is the thing that has the greatest distance to the [colour] thing?	0/6
How many objects are the same shape as the [colour] object?	How many objects have the same geometric form as the [colour] object?	6/6

Table 1: Original and variant questions

Table 1 contains both the original question types and variant question types. The variable [colour] can take on the values {red, green, blue, orange, gray, yellow}, making a total of 36 possible original questions (and thus 36 possible variant questions). The "Accuracy" column reports the fraction of colours for which the model infers the right question latent

Appendix B: Unseen Questions

New Question	Accuracy
What colour is the object closest to the [colour] object?	0.0/6
Is the [colour] object on the right?	2.9/6
Is the [colour] object on the bottom?	2.8/6
What shape is the most common?	0.0/1
How many objects are on the left side of the screen?	0.2/1
How many objects are on the top of the screen?	0.0/1

Table 2: New questions used to evaluate the generalization of the Bayesian VQA model.

Table 2 contains a list of all of the new questions we used to evaluate the model, along with the average accuracy on those questions. There are 6 possible variants of the first three questions, one for each colour, while there are no variants for the last three questions. Accuracies are computed across 1000 image latents randomly sampled from the generative model of images.