Food-related Visual Question Answering

Caroline Jin

Abstract
There are few applications available to help the blind with activities involving sight. Particularly, blind people face situations, in which they do not know whether the food they eat is safe or not. To achieve better human-computer interactions, my research focuses on Visual Question Answering Models, which respond to questions about a food-image. Researches studying Visual Question Answering face challenges of how to best integrate computer vision with natural language processing. Some researchers propose combining image models with question models while others claim models analyzing the relationship between questions and images. My approach is creating a VQA model composed of a convolutional neural network (CNN) and recurrent neural network (RNN). I trained and evaluated my model on questions about the identity of the food item with about a 60%. To expand the answering capabilities of my VQA model, I created a graphical user interface that enables the model to be continuously trained by users.

Keywords
Deep Learning — Computer Vision — Natural Language Processing

Contents

1. Introduction

Blind people suffer greatly in situation where visual information is needed, particularly what food to eat. When going to the groceries, a blind person may not be sure if the product he wants to buy contains an ingredient he is allergic to. To alleviate this situation, there are currently applications that can inform blind people about their food. TapTapSee [1] and CamFind [2] are mobile applications that enable users to take a picture of an object and have it described through a voice over. The next step for these visual applications is to better human-computer interactions, meaning the user can naturally speak to the system to retrieve a more comprehensive response. Visual Question Answering (VQA) enables more information about the image to be gathered by asking specific questions about it.

Unlike image recognition, in which models like ImageNet and ResNet have achieved high accuracy, there is no set of VQA methods that have shown extremely high-performance. For the general VQA dataset, researchers have experimented with using simple classifiers such as K-Nearest Neighbors [3]. Researchers also proposed a Multinomial Compact Bilinear pooling model, which projected image and question feature vectors at random and convolved these vectors into a Fourier space [4]. For my project, I created two separate models for extracting image and text features and then inputting these features to a classifier like a neural network [5]. My model followed a similar structure to the Neural-Image-QA models [6, 3], which used a convolutional neural network (CNN) and a recurrent neural network (RNN) with a long-short term memory layer (LSTM). However, instead of using GoogLeNet for my CNN [7], I used the VGG16 model [8].

I tested my VQA model on identity questions that had only one answer. Thus, the testing metric I used was simply the accuracy of my model. My model performed with an accuracy of 60%.

2. Methods

There are currently publicly available VQA datasets, such as COCO [9], that are focused on a broader audience. The VizWiz dataset [10] has images taken and questions asked by blind people. This dataset contains 20,000 training image/question pairs, 200,000 training answer/answer confidence pairs, 3,173 image/question pairs, and 31,730 validation answer/answer confidence pairs. I focused on the 7,000 food images in this dataset. Using the subset of food images, I

Figure 1. This picture is a sample of images that I retrieved from my Python script.

created a machine learning model that can answer question about pictures of food.

The VQA model was composed of four parts:

1. Image feature extraction: pre-trained CNN
2. Question feature extraction: natural language processing model with long-short term memory (LSTM)
3. Combined feature: element-wise concatenation
4. Final classifier: two-layer neural network

2.1 Current Packages and Software Used

I programmed all the data extractions and models Jupyter Notebook in Python 3 [11]. For data collection, I organize and formatted the data more easily with pandas Dataframe [12]. For natural language processing, I stored my vectors using NumPy [13]. My image model and question model required the TensorFlow [14] and keras packages [15]. I used Python's pickle file library to save and access my data and a JSON file to store my model.

2.2 Data Collection

Initial Dataset: VizWiz Once I collected the data, I cleaned up the raw text, so I could easily extract the text features. Cleaning up the raw text involved tokenizing each question string using python's string methods. I transformed all the words to lowercase. In this way, when transforming into vectors, the words 'when' and 'When' would have the same vector. I also removed strings that were unnecessary. For instance, punctuation, such as '?' and '.', were not needed to understand the context of the question. I found that the users often included 'please' and 'thank you' to show their gratification for the authors of the VizWiz dataset. However, these words of appreciation did not contribute much to the meaning of the question, so I removed those words as well. The result was a string that was concise.

New Batch of Food Images The VizWiz dataset had very few examples of each type of food. For instance, there were only two images of apples. I used Microsoft Azure's APIs for data collection. One of these APIs was the Bing Web Search API. I used this API to write a Python script that automatically gathered data when I type in command a keyword, such as 'cookie' as shown in Figure 1. After retrieval, I manually removed images that were not the specific food type or were not foods at all. When obtaining images of a soda can or bottle, I removed images that were the logo of the soda company. Through this process, I had between 70-100 images per food category and saved these images under a folder.

2.3 Image Feature Extraction

The major component of my model is the convolutional neural network (CNN). Instead of constructing my own CNN, I used a pre-trained CNN, offered by the keras package [15]. The model I used was the VGG16 model [8]. A snippet of the code is shown in Figure 2. The model required the input sizes of the images to be 224 pixels by 224 pixels by 3 rgb values. Thus, I resized all of the images in my dataset to the aforementioned size. I kept most of the layers, except for the last few layers to fit my data. I modified the last four layers into the following: a flattening layer to reduce the dimensions of the output, a dense layer with a relu activation function, a dropout layer of a 0.2 rate, and another dense layer with the same parameters.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792	input_1[0][0]
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block1_conv2[0][0]
block2_conv1 (Conv2D)	(None, 112, 112, 128	73856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 112, 112, 128	147584	block2_conv1[0][0]
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block2_conv2[0][0]

Figure 2. The code snippet shows the first few layers in the VGG16 model. The output shape refers to the magnitude of the vector being outputted. The number of parameters represent how many internal variables must be trained by the model. The last column points to the previous layer. The blue boxes highlights the convolutional layers, and the red boxes highlight the pooling layers.

I decided to focus on image identification for my image model as I thought it would help answer the 'what is this' question. I took my image dataset from both the VizWiz dataset and those samples retrieved from bing and had a total of 4660 images. I trained my model on 80% of these images until I reached an accuracy of 91% on the training images. On the testing images, my model achieved an accuracy of 80% on 36 different food categories. Figure 3 shows how well my model predicted for one image. I decided to train my model until a training accuracy of 90% was reached to ensure that my modified VGG16 model sufficiently learned to identify each image while it did not lead to overfitting on only identifying.

2.4 Question Feature Extraction

Generating Questions and Preprocessing Instead of analyzing a wide range of questions, I focused on answering three forms of questions: 'what is this,' 'is this a/an [food],' and 'are these [food]s.' I generated these questions based on the food labels of each image and established rules that followed correct grammar and spelling. These rules included using 'an' instead of 'a' for words that started with a vowel

Figure 3. The output represents the modified VGG16 model's predictions of the top five classes of one image.

and changing the ends of words from singular to plural form, such as 'jerky' to 'jerkies.' Preprocessing these raw questions involved two steps. The first step was creating a dictionary of each word corresponding to its numerical ID. The second step was placing these numerical IDs into a length-50 vector. I set the vector to length 50 to ensure that long questions could be processed.

I saved these vectors in the order corresponding with the 4660 images. Each question type contained 4660 questions, and these questions were stored in a pickle file. When combining each of my questions to its corresponding answer and image file ID, I created a data table using pandas.

Question Model: Natural Language Processing Model My question model contained four layers: an input for the preprocessed questions, an embedding layer, a long-short term memory layer, and a dense layer, which merged the question model to the image model.

2.4.1 Embedding Layer

The purpose of the embedding layer is to determine the 'meaning' of the word. This layer converts words into dense vectors where a vector represents the projection of the word into a continuous vector space. The word learned from the question is placed in the vector space based on the words that surround the word when it is used. Words that are in the same category, such as 'cookie' and 'cupcake,' are placed closer to each other. The embedding layer in my question model restricted the size of vocabulary to 2000 words and represented the words with a length-100 dense vector.

2.4.2 Long-short Term Memory Layer (LSTM)

The long-short term memory layer determines the dependencies of each word in the question. Each cell in this layer is composed of three parts: the input gate, the forget gate, and the output gate. The input gate decides to what extent input values x can be flowed into the cell using an activation function. The forget gate determines how long x should remain in the cell using a recurrent activation function. The output gate considers the extent to which x should be computed in the output. In my question model, the LSTM had an output with a dimensionality of 100, a tanh for the activation function, and a sigmoid function for the recurrent activation. The layer also had a dropout rate, the rate at which neurons are being deactivated in a layer, of 0.2 and a recurrent dropout rate of 0.2.

2.5 Combined Feature and Neural Network

To combine the image feature and question feature, I initially started with elementwise addition. This layer involved reshaping the output vectors of both image and question models to be the same size; the resulting reshaped size was (36,). However, after training the model, it tended to overfit on specific types of questions. The model tended to answer 'what is this' questions with 'yes' or 'no' or tended to answer 'is this a/an [food]' with the identity of the food item. I used elementwise concatenation; this operation pasted the output vector of the image model and the output vector of the question model. The combined output was used as the input for the final classifier, which was a three-layer neural network, consisting of the input, a fully-connected layer, and output.

dense_2 (Dense)	(None, 36)	36900	dropout_1[0][0]
dense_3 (Dense)	(None, 50)	5050	lstm_1[0][0]
concatenate_1 (Concatenate)	(None, 86)	0	dense_2[0][0] dense_3[0][0]
dense_4 (Dense)	(None, 100)	8700	concatenate_1[0][0]
dense_5 (Dense)	(None, 38)	3838	dense_4[0][0]

Figure 4. This output shows the last few layers in my vqa model. 'dense2' and 'dense3' refers to the output of the image and question model respectively. 'concatenate1' performs the elementwise concatenation. 'dense4' and 'dense5' are the two layers in my neural network. See Fig. 2 for description of each column.

Reference to Figure 5.

3. Results and Discussion

3.1 Training

Training the model in an efficient manner was challenging due to the need of high computing power to train my model on large data sample. Before I trained, I mixed the three forms of questions in a new dataset. I divided my data: 64% training, 16% validation, and 20% evaluation. I trained for a total of around 30 epochs until I reached around 70% accuracy for training and 60% accuracy for validation. My final accuracy when evaluating on the testing data was around 60% on 400 food-question pairs. This is comparable to the 54% accuracy obtained by Antol et al. [3] for open-ended questions.

3.2 Graphical User Interface for Continued Training

As I moved on to expand my dataset to other type of questions, there were so many possible food image-question pairs that I needed to tackle. I decided the best way to approach the problem was to create an interface that allows users to input their

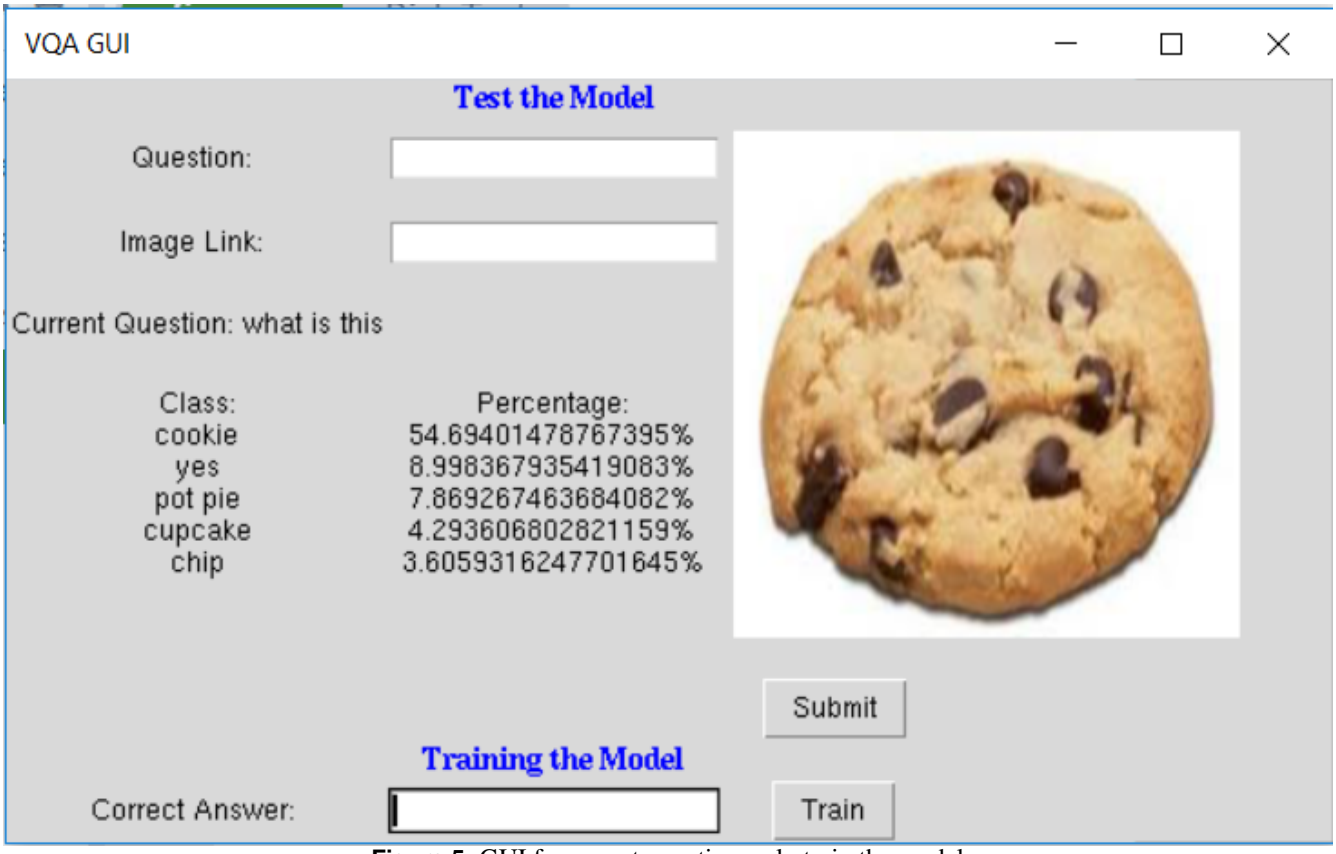

Figure 5. GUI for users to continuously train the model

respective food image-question pairs and have the model train on those. That way I can improve my model performance.

I composed my GUI of two parts: testing and training. The testing section displayed image of a food item to the right. In this part, the users would type a question about the image into the question-input box and press the 'Submit' button. The GUI would output the top five answers the model predicted the answer to be. If the answer was not in the top five choices or the percentage for the answer was too low, users would input the correct answer and press the 'Train' button so that the model would learn similar forms of the inputted image-question pairs. Users could also input their own image from online and repeat the process mentioned above (See Fig. 5).

4. Conclusions and Open Questions

My research focused on three forms of questions: 'what is this,' 'is this a/an [food],' and 'are these [food]s.' Future research can work to expand the question-answering abilities of the vqa model beyond the questions about identification. More specific questions, such as 'what brand are these pretzels,' would retrieve more information about the food item. Furthermore, these questions would provide an improved conversation between the blind person and computer.

In its current form, my GUI is targeted toward three forms of questions. To expand the usefulness of the GUI, future researchers can add new sections into the GUI. Such sections include new questions and new words used. Moreover, the user would be better informed with a section that shows a live training of the model as outputted by the TensorFlow's methods 'fit' method [14].

5. Acknowledgments

I would like to thank the following people for helping me with my research project: Mr. White, my research lab director, for providing guidance over the course of my research, and Dr. Shah for introducing the VQA challenge to me.

References

[1] Taptapsee app: How to Use.

[2] Camfind: Visual Search.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[4] 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. *CoRR*, abs/1606.01847, 2016.

[5] Kushal Kafle and Christopher Kanan. Visual question answering: Datasets, algorithms, and future challenges. *CoRR*, abs/1610.01465, 2016.

[6] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. *CoRR*, abs/1505.01121, 2015.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[8] Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, Denis Teplyashin, Karl Moritz Hermann, Mateusz Malinowski, Matthew Koichi Grimes, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. The streetlearn environment and dataset. *CoRR*, abs/1903.01292, 2019.

[9] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samuel White, and Tom Yeh. Vizwiz: Nearly real-time answers to visual questions. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2010.

[11] Project Jupyter. The jupyter notebook. `https://jupyter.org/index.html`, February 2018.

[12] pandas. pandas dataframe. `https://pandas.pydata.org/pandas-docs/stable/index.html`, April 2019. v.0.24.2.

[13] Travis Oliphant. Numpy. `https://www.numpy.org/`. v.1.16.3.

[14] Apache License 2.0. Tensorflow. `https://www.tensorflow.org/`. v.1.13.1.

[15] François Chollet. Keras. `https://www.tensorflow.org/guide/keras`. v.2.1.6.