



TUM Data Innovation Lab
Munich Data Science Institute (MDSI)
Technical University of Munich
&
PreciTaste GmbH

Final report of project:
**Visual Question Answering using Large
Language Models**

Authors	Nattapol Choojirawong, Hsuan-Yu Yao, Ziyue Zhang
Mentor(s)	M.Sc. Mathias Sundholm, M.Sc. Sebastian Freytag
Project lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

Feb 2024

Abstract

Visual Question Answering (VQA) is a prominent vision-language task that is highly applicable in various real-world scenarios. However, it poses significant challenges, necessitating the comprehension of user queries in natural language, the retrieval of information from images or videos, and the provision of accurate responses in natural language. Recently, with the assistance of Large Language Models (LLMs), numerous solutions have emerged. However, many of these solutions depend on multimodal models and entail significant computational costs for fine-tuning. Consequently, establishing a VQA system within limitations presents a notable challenge. To tackle this issue, we propose methods that leverage textual information as a conduit to link visual and textual domains, enabling zero-shot VQA tasks without the need for end-to-end training. Our model explores two distinct approaches: 1) utilizing a detection model and a captioning model to establish a connection between image and text. In this phase, we demonstrate the feasibility of leveraging textual representations instead of vectors to connect the two domains, thereby identifying methods to mitigate the hallucination issue. 2) combining the Img2LLM model and the stochastic captioning model to provide both local and general information to the model and assess its performance. Additionally, we introduce a new metric for VQA evaluation, incorporating semantic understanding rather than strictly matching to ensure fairness in the evaluation process. Code is available at <https://github.com/Nattapolchoo/Visual-Question-Answering-Using-LLM>.

Contents

1	Introduction	1
2	Related Work	1
3	Attempt at Training VQA Model	2
4	Methods: Language-mediated VQA Model	3
4.1	Mistral with Image-to-Text Vision Models	3
4.1.1	Image-to-Caption	4
4.1.2	Image-to-BoundingBoxes	4
4.2	Original Img2LLM	6
4.3	Our implementation of Img2LLM with stochastic captioning	8
4.3.1	Stochastic Caption	8
4.3.2	Our Img2LLM implementation	9
5	Experiments and Results	10
5.1	Dataset	10
5.2	Evaluation Metrics	11
5.2.1	Standard VQA evaluation	11
5.2.2	Cosine Similarity	12
5.3	Experiments	13
5.3.1	Setup	13
5.3.2	Limitations	14
5.4	Results	14
5.4.1	Performance comparison	14
5.4.2	Effect of Cosine Similarity Metrics	17
5.4.3	Analysis	19
6	Future Work	20
7	Conclusion	20
	Appendix	23
A	System message	23
B	Prompts	23
B.1	Image-to-Caption	23
B.2	Image-to-BoundingBoxes	23
C	Template question	24
D	Setup	24

1 Introduction

Visual Question Answering (VQA) [1] is a prominent vision-language task with a broad range of real-world applications, such as a visual chatbot. In the food industry, real-time monitoring of inventory levels is essential for addressing inquiries from users or staff. Use cases are quite easy to think of, but building a VQA system for one specific use case is challenging as the task involves comprehending user queries formulated in natural language and understanding the image or video content sufficiently to provide accurate responses. Advancements in Large Language Models (LLMs) and Vision-language Pre-training models make it feasible to tackle this task by integrating information from both the visual and textual domains.

In this collaborative project between TUM Data Innovation Lab and PreciTaste, an AI and sensor technology company specializing in solutions for the gastronomy and baking industry, we explore a VQA system utilizing Large Language Models (LLMs) capable of accurately answering natural language questions about input images. The LLM employed is Mistral 7B[7], a text generation model hosted on their servers. Despite the limitation of the hosted LLM, as opposed to an on-hand model, we successfully constructed our VQA system through language-mediated VQA models, addressing the core challenge of transferring visual information into the text space via text instead of vectors.

Our primary contributions can be summarized as follows:

- We introduce a language-mediated VQA model by connecting the instance detection model to the provided LLM via the captioning model, then analyzing performance and identifying causes of hallucination issues.
- We propose a VQA model with zero training, leveraging Img2LLM with stochastic captioning to extract question-relevant and general information from the image.
- We analyze the performance using the VQA validation dataset and introduce new metrics for a fair evaluation of the results.

2 Related Work

Multimodal models are utilized in most of the VQA models, which align visual features and text embedding space. One support of them is CLIP [12], which trains an image encoder and a text encoder jointly on image-text pairs and proves that natural language provides a border source of supervision for visual concept training. A multimodal mixture of an encoder-decoder model from BLIP[8] offers more flexibility and better performance on a wide range of downstream tasks, including VQA. Other methods like LLaVA [9] and LLaVA-Plus[10] extend instruction-tuning to the multimodal space, which paves the way towards building a general-purpose visual assistant, turning the VQA task into a multimodal instruction-following agent.

Language-mediated VQA is a VQA paradigm that uses natural language as the intermediate image representation rather than the vectorized representation. In this paradigm, the VQA model requires no training to align the representation. Moreover, the VQA system is modular, which allows the LLM to be easily upgraded without the need to retrain the alignment layer or fine-tune the LLM again. The image is first converted into language descriptions using a vision transformer, then fed the descriptions, possibly with other in-context exemplars, to a frozen LLM. PNP-VQA [16] generates image-relevant captions in order to provide relevant information to a QA-specific LLM. Img2LLM [6] achieves a zero-shot setting without the need for QA-specific LLM by generating image-relevant captions and multiple QA pairs as an in-context exemplar for the image.

Vision Models take images as input. Their outputs can be categorized into the following three types: caption, bounding boxes, and embedding. We investigated the capability of the LLM to understand images when combined with different vision models. **Image-to-caption** models output a textual description from a given image. The image captioning task lies at the intersection of computer vision and natural language processing. Based on OpenCLIP, CoCa [18] is a minimalist design of an image-text encoder-decoder model pre-trained jointly with contrastive loss and captioning loss. Another pretraining framework for unifying vision-language understanding and generation is BLIP [8], which utilizes large-scale noisy web data by bootstrapping the captions. **Image-to-bounding boxes** models, whose primary use case is in the object detection task, identify the position, boundary, and class label of objects in an image. Despite the small model size of YOLOS [4], which is a Vision Transformer (ViT) trained using the Detection Transformer (DETR) loss, it is one of the most popular models for object detection tasks in the Hugging Face community. In an open-vocabulary setting, OWL-ViT [11] is a zero-shot detector that attains strong performance on text-conditioned objection detection. **Image-to-embedding** models are mostly pre-trained models for downstream tasks. An embedding is a high-dimensional vector representation capturing several features of an image. CLIP ViT-L/14 [12] learns image representations from scratch by pretraining to predict matching image-text pairs.

3 Attempt at Training VQA Model

Firstly, PreciTaste provided us with Mistral 7B [7] as an LLM by hosting it on one of their servers. As such, all of the VQA models we tried in this project used Mistral 7B as an LLM. So, our LLM is always fixed to Mistral 7B, but our choice of the vision transformer model is flexible.

Our first idea for this project is to follow in the footsteps of LLaVA [9] by recreating the LLaVA architecture with Mistral 7B (Fig 1).

In the original LLaVA architecture, the input question X_q is embedded using frozen LLM to get the textual embedding H_q , and the image X_v is embedded using frozen vision encoder to get the image embedding Z_v . Then, the Z_v goes through a trainable projection layer (simple linear layer, in the case of LLaVA), which is trained to align the image embedding space to the text embedding space, resulting in H_v . Then, both H_v and H_q are concat and sent to the LLM to get the result. In the original paper, the authors use

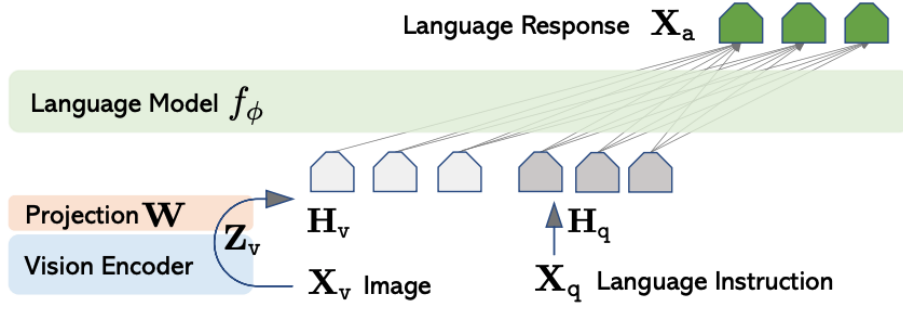


Figure 1: Image from the paper [9]: LLaVA original architecture. The projection layer W is trained to align the image embedding Z_v to the textual embedding in order for LLM to understand the visual feature.

Vicuna [3] as their LLM and CLIP ViT-L/14 [12] as their vision encoder.

In our design, we wish to implement LLaVA with the same vision encoder and Mistral 7B as our LLM. We would only change the LLM to keep the architecture as close to LLaVA as possible. Then, we would train a simple linear layer as a projection layer and evaluate the performance. Our end goal would be to use multiple vision encoders and then train the projection layer for each vision encoder. Each vision encoder will be used for different tasks based on the input question.

Unfortunately, while implementing the model, we found that it takes around 8 seconds for Mistral to respond to the API request with an answer when using text embedding as an input. It would require an extremely long time to train the projection layer, even with a small dataset. We would have to wait 8 seconds to get the output from Mistral and calculate the loss for each training data, ignoring other processing time from other parts of the model. This response time makes it impossible for us to train the projection layer and create a VQA model this way. Thus, we have to find other approaches to create a VQA model.

4 Methods: Language-mediated VQA Model

The Language-mediated VQA model uses natural language as an intermediate representation of the image rather than the vectorized representation, which means that we do not have to train the projection layer to align the representation. However, natural language, such as the caption of an image, cannot convey as much information inside the image. For this reason, many papers (Img2LLM [6], PNP-VQA [16], and PICa [17]) have tried to improve the performance of the VQA model using different prompt engineering methods to give more image information to the LLM.

4.1 Mistral with Image-to-Text Vision Models

Without the need to align multimodal embeddings, a naive approach to combine Mistral with a vision model is to feed the vision model output in textual form as input to Mistral.

We implement two methods using different image-to-text vision models: image-to-caption and image-to-bounding boxes. It is straightforward to feed the caption information to the LLM because it is already represented in natural language. Regarding the bounding box information, we are able to convey the bounding box coordinates in textual form after using some prompt engineering. We include the prompts in Appendix B.

To test the performance, we design four types of questions to ask the LLM, including:

- Description: “What is in this image?”
- Counting (Yes/No): “How many OBJECTs are there in this image?”
- Location: “Where is the OBJECT in this image?”, “What is in the top left corner of this image?”
- Irrelevant question: “What is the OBJECT doing in this image?” (OBJECT is not in the image)

where the word OBJECT is replaced with a specific object of interest in the image.

4.1.1 Image-to-Caption

First, we input the image to the vision model CoCa [18] to generate a caption. Then, we concat the caption with the user question to form the input prompt to Mistral (Fig 2). In our observation, this model can correctly answer the first three types of questions when the provided vision model’s caption contains valuable information for the question. However, for the irrelevant question, the LLM starts to make up stories about nonexistent objects due to a lack of information.



Caption: three tomatoes on a wooden table in front of a blurry background.

Question: What is in this image?

Three tomatoes are visible on a wooden table.

The background appears to be blurred, making it difficult to see any other objects or details in the image.

Question: What is the dog doing in this image?

The dog appears to be sitting next to the three tomatoes on the wooden table. It’s not clear what it is doing, as there isn’t enough context or information provided in the image alone.

Figure 2: Example of caption from CoCa [18] and Mistral’s answers to the descriptive and irrelevant questions.

4.1.2 Image-to-BoundingBoxes

In the previous approach, the vision model’s caption can provide Mistral with general information about the whole image. However, with the bounding box information instead,

Mistral can obtain more information regarding the local features, such as the location and number of a specific object in a complicated scene.

Each bounding box of an object is represented by a tuple of 4 numbers, indicating the 2-dimensional coordinates of the top left and bottom right corners of the bounding box. When constructing the prompt, we wrap the information, including the class label and the coordinates of each bounding box, into one sentence. To make it more understandable for the LLM, we simplify the information by normalizing the coordinates, using the center point coordinates for each bounding box, and explicitly explaining the meaning of each value. To teach the LLM how to read the bounding box coordinates, we refer to and redesign the instruction message used in LLaVA [9] and attach it as an instruction, referred to as “system message”, at the beginning of the prompt (Fig 3 and Appendix A).



There is a remote at location (0.18, 0.2). There is a remote at location (0.55, 0.28). There is a cat at location (0.26, 0.55). There is a remote at location (0.18, 0.2). There is a cat at location (0.77, 0.41).
Question: What is in this image?

This image contains two remotes and two cats. One remote is located near the top-left corner of the image, while the other is located near the bottom-right corner. Both remotes are positioned slightly above the centerline of the image. The first cat is located near the middle of the image, with its head pointing towards the top-left corner. The second cat is located near the bottom-right corner of the image, with its tail pointing towards the top-left corner.

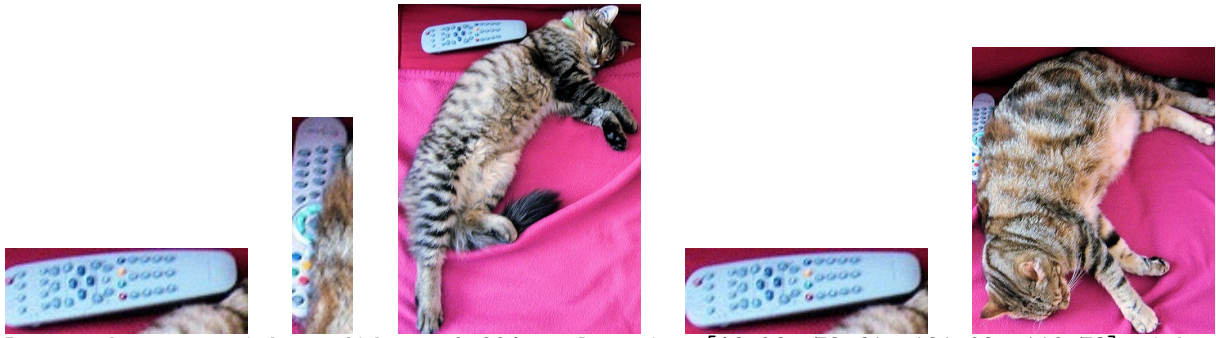
Figure 3: Example of the bounding boxes information detected by yolos-tiny [4] and the descriptive question, followed by the answer from Mistral.

Even after the simplification, it is still tricky for the LLM to understand the coordinate system. In addition, Mistral still has a hallucination problem and makes up irrelevant and wrong statements, even with enough information for the LLM to answer the question.

To mitigate the hallucination problem, we develop two approaches to enhance the communication between the image-to-bounding box model and the LLM.

Our first approach is to provide more context to the LLM by adding a caption to each image patch cropped by the bounding box. After the bounding box is detected, we cut the image based on this bounding box and use the model blip-image-captioning-base [8] to generate a caption for this image patch. We concat the caption to the bounding box location in the prompt (Fig 4). This raises a problem that the object in the cropped image may not be recognized successfully due to the small image size or weird size ratio, resulting in an incorrect caption and thus misleading the LLM.

Another approach is to leverage text-conditional object detection. Unlike previous objection detection models we have attempted, a characteristic of OWL-ViT [11] is that it simultaneously takes an image and object classes to detect as input, where the classes are



Detected remote with confidence 0.994 at location [46.96, 72.61, 181.02, 119.73] with caption a cat is laying on a couch with a remote control device Detected remote with confidence 0.975 at location [340.66, 79.19, 372.59, 192.65] with caption a bunch of white flowers Detected cat with confidence 0.984 at location [12.27, 54.25, 319.42, 470.99] with caption a cat laying on a pink blanket Detected remote with confidence 0.922 at location [41.66, 71.96, 178.7, 120.33] with caption a cat is laying on a red couch Detected cat with confidence 0.914 at location [342.34, 21.48, 638.64, 372.46] with caption a cat sleeping on a pink couch

Question: What is in this image?

Answer: There are two cats lying on couches. One cat is lying on a pink blanket while the other is sleeping on a pink couch. On one side of the room, there is a remote control device sits on a table. On the opposite side of the room, there is another remote control device sitting on a coffee table. Additionally, there are several white flowers arranged on a table nearby.

Figure 4: Example of cropped images, bounding boxes coordinates with captions from blip-image-captioning-base [8] and answer from Mistral to the descriptive question. Five cropped images are on top, with two indicating the same remote. This is because yolostiny [4] produces five bounding boxes, two overlapping and detecting the same remote.

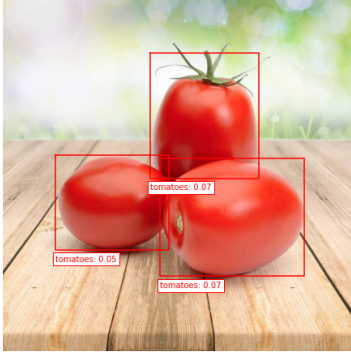
specified in a list of texts. This property allowed us to combine the LLM with the vision model in the following way:

1. After receiving the user question, the LLM extracts objects to look for from the question and formulates them into a list.
2. The list acts as the input for the vision model to generate bounding boxes.
3. The detected bounding boxes are returned to the LLM, enabling it to answer the user’s question based on the information.

The advantage of this model is that the objects to detect are conditioned on the user’s interest. There is no need for the vision model to detect everything in an image. Therefore, the message delivered from OWL-ViT [11] to Mistral contains less unrelated and mislabeled information, making Mistral focus more on the question the user is interested in. For example, without the keywords list, the vision model would also detect the wooden table (Fig 5).

4.2 Original Img2LLM

Img2LLM [6] established a VQA model without any pretraining and relied solely on the pretrained model. Facing the computation limitation, we opted to initially emulate the



Question: How many tomatoes do you detect in this image?
 Keywords: [tomatoes]
 Detected tomatoes with confidence 0.066 at location [373.69, 137.45, 648.23, 455.77]. Detected tomatoes with confidence 0.052 at location [134.07, 397.0, 421.52, 637.24]. Detected tomatoes with confidence 0.071 at location [398.16, 404.45, 764.06, 703.3].
 Answer: 3

Figure 5: Example of list of keywords extracted by Mistral, bounding boxes detected by OWL-ViT [11] based on the keywords, and answer from Mistral to the question about the number of tomatoes.

procedures of Img2LLM before seeking further enhancements. The Img2LLM’s hypothesis is that LLMs are usually pretrained using generative or denoising objectives on language modeling tasks. Therefore, LLMs are unfamiliar with the task of question and answering and can fail to utilize the contextual information we give to generate the answer. By including examples of relevant question-and-answer pairs in the prompt, it not only demonstrates the QA task but also provides contexts of the image to the LLM.

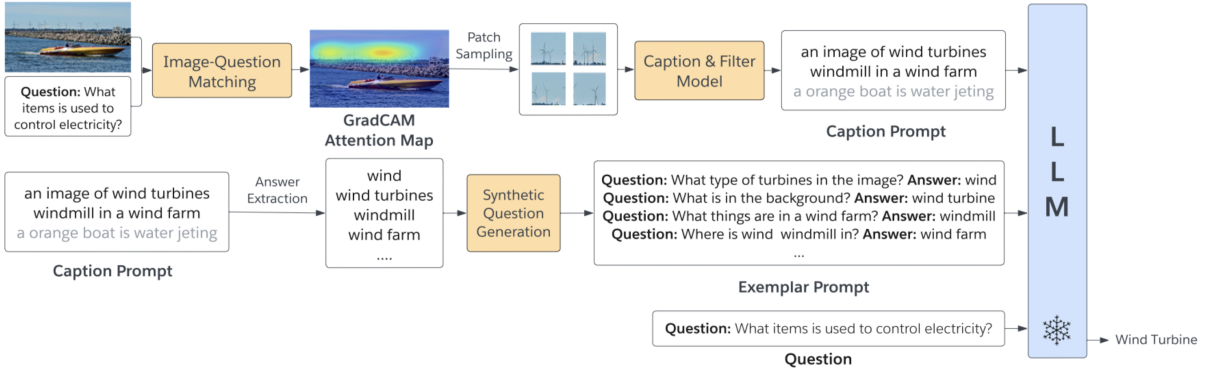


Figure 6: Img2LLM original architecture [6]. The top pipeline generates the question-relevant captions as a Caption Prompt. The bottom pipeline generates the Question-Answer pairs as an Exemplar Prompt. The Caption Prompt and Exemplar Prompt are concat with the Input Question and sent to the LLM.

The original Img2LLM architecture contains two parts (Fig: 6). The **top pipeline** creates captions relevant to the input image and question. To achieve this, the image is first equally divided into K patches. Then the image goes through the Image-grounded Text Encoder (ITE) in BLIP [8], which assigns a similarity score $sim(v, q)$ to any pair of image v and question q . The ITE uses a vision transformer to encode the image and a textual encoder to attend to the image features using cross-attention. However, the cross-attention matrix A (indicative of patch importance) can contain many redundancies, and many attention heads can be pruned with little performance loss, suggesting that some

attention scores are uninformative [2]. The paper uses a variation of GradCAM, a feature-attribution interpretability technique, to filter out uninformative patches and extract only relevant patches. This is done by multiplying the cross-attention scores element-wise with the gradient of the similarity score w.r.t the cross-attention score $\partial \text{sim}(v, q) / \partial A$. The relevance of the i^{th} image patch takes the average over all attention heads and sum over all textual tokens w.r.t the i^{th} column of attention score.

Then, the caption is generated using BLIP’s image captioning network branch. The subset of image patches is sampled with probability proportional to the patch relevance and then sent to the branch to generate the caption with top-k sampling.

The **bottom pipeline** creates question-and-answer pairs. The question-relevant captions generated from the top pipeline are given to the spaCy parser to extract noun phrases, verb phrases, adjective phrases, numbers, and boolean-type words to be used as answers. Then, the synthetic questions can be generated given the captions and answers using either template-based or neural-based (e.g., T5-large model) question generation method. The question templates are designed specifically for each part of speech (POS) type for the template-based question generation. (Appendix C)

For their best-performing model, the model selects n most frequent answers from the captions and their corresponding questions as QA pairs. The questions are generated using neural question generation. Then, the prompt is arranged in order of Instruction + Captions + QA pairs before being sent to the LLM. The prompt is formatted as

```

Please reason the answers of question according to the contexts.
Contexts: [all captions]
Question: [question 1]
Answer: [answer 1]
.
.
Question: [input question]
Answer:

```

Their model with 6.7B parameters has the best accuracy at 57.6 on the VQAv2 validation dataset. We look specifically at the model with 6.7B parameters since it has the closest number of parameters to Mistral with 7.3B parameters.

4.3 Our implementation of Img2LLM with stochastic captioning

4.3.1 Stochastic Caption

The paper [15] shows the strength of using stochastic captions with respect to the improvement of VQA model performance. We decided to combine the stochastic captions with the captions generated from the Img2LLM to improve the model’s performance. To generate stochastic captions, we use the vision transformer *CoCa-ViT-L-14-laion2B-s13B-b90k* [14] to output captions of an image with top-k sampling, or top-k most probable captions. We hypothesize that the stochastic captions allow us to provide the LLM with

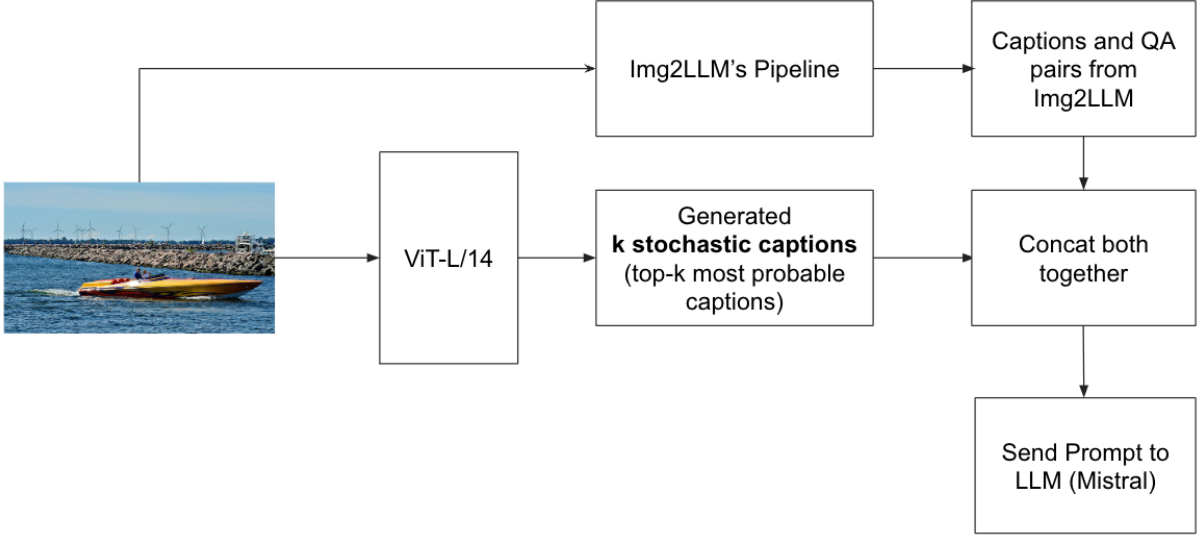


Figure 7: Img2LLM combined with stochastic captioning. The vision transfer ViT-L/14 generates top-k most probable captions as the **stochastic captions** from an image. Then, the stochastic captions are concat with the Captions and QA pairs from Img2LLM before being sent to Mistral.

a greater variety of descriptions and maximal image coverage. Moreover, it can help provide the information needed to the LLM if the Img2LLM captions, generated from ITE and GradCAM, are focused in the wrong spot (generating incorrect patch relevance). In the prompt, the stochastic captions are concat after the captions generated from ITE and GradCAM (Fig: 7).

4.3.2 Our Img2LLM implementation

For our implementation of the Img2LLM, we tried to follow the original implementation and their best configuration as closely as possible. However, there are some parts that are different, and it should be noted that using the original Img2LLM best configuration does not guarantee the best result for our model.

For the bottom pipeline, we use the `en_core_web_lg` from spaCy to extract the answers from captions. Moreover, the original paper finetuned a pretrained T5-large model on five textual QA datasets to generate questions from answers while we decided to use the T5-base fine-tune on SQuAD for Question Generation from huggingface instead [13].

Furthermore, we changed the LLM to Mistral 7B, which comes with complications. The LLM used in the original paper can have as many as 2048 maximum token lengths as input. Meanwhile, Mistral 7B can only accept 1024 maximum token lengths. This means that Mistral cannot get as much context and information from the prompt as the original Img2LLM’s LLM model, which causes our model to not perform as well as the original model. We also notice that, on average, questions generated via neural provide more context but longer token lengths than those generated via the template. We tried to send as much information as possible to the LLM by reducing the number of QA pairs by one

and kept reducing until we no longer got the error from Mistral.

In order to experiment with the model, our model looks at the configuration file to determine if the question should be generated with neural or template question generation and how many question-relevant captions, stochastic captions, and QA pairs should be included in the prompt. The set of parameters that we experiment on will be discussed later in Section 5.3.

To provide clarity, an illustrative example of the entire pipeline is presented. As depicted in Figure 8b, GradCAM highlights the relevant portion of the image corresponding to the question, represented as a heatmap. Subsequently, captions are generated based on the specific region of interest within the image. Additionally, stochastic captioning is utilized to generate captions based on the overall context of the raw image.

The example in Figure 8 briefly demonstrates how the prompts are generated. Our model extracts keywords from the captions, highlighted in shading, and subsequently generates questions based on these keywords, forming Question-Answer pairs. To enhance clarity, QA pairs generated using the template QA selection strategy are depicted in black, while those generated via the neural method are highlighted in blue. Additionally, the user question is concatenated to the end of the prompt, denoted in red. It is important to note that within our model, only one of these methods is employed.

5 Experiments and Results

5.1 Dataset

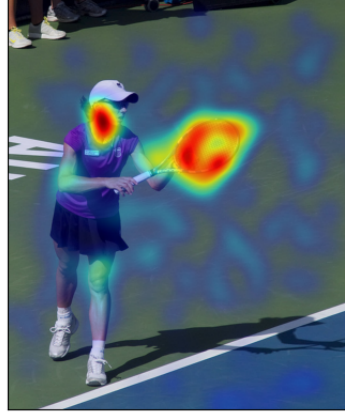
We decided to conduct our experiments on the VQAv2 [5] validation dataset, which contains 40,504 COCO images and at least 3 questions (5.4 on average) per image, 214,354 total. Each question has 10 ground truth answers annotated by humans. However, considering the computation limitation, we extract a smaller subset by randomly selecting 100 images with more than five corresponding questions and then randomly collecting 5 questions per image, resulting in 500 questions. The experiment results shown in Section 5.4 are obtained on this extracted data subset.

We precalculate and store intermediate results, such as the question-relevant and stochastic captions, to speed up the evaluation process. Overall, it takes around 20 minutes to predict answers for all 500 questions in our dataset with the template question generation method on our device with an Apple M1 Pro chip and 32 GB RAM. It takes more than 2 hours when using the neural question generation method. During the process, around 14 GB of memory storage is occupied.

The VQAv2 dataset contains many question types. Figure 9 shows the question type distribution on our extracted dataset. In addition, the VQAv2 dataset categorizes the answers into three groups: yes/no, other, and number. An analysis of our models' performance on different answer types can be found in Section 5.4.1.



(a) raw image



(b) heatmap

Question:

What kind of racket is she using?

Img2LLM Captions:

a woman is wearing sunglasses holding a tennis racket and a tennis racket, woman playing tennis in sunglasses holding a racket in her hand...

Stochastic Captions:

a woman in a purple shirt is playing tennis, a female tennis player is hitting the ball...

Prompt:

Please reason the answer of the questions according to the given contexts. Contexts: Captions

Question: What item is this in this picture? Answer: woman

Question: What item is this in this picture? Answer: a tennis racket

Question: Which action is being taken in this picture? Answer: holding

Question: Who is swinging a tennis racket and a tennis ball on a tennis court?

Answer: a woman

Question: What kind of racket is she using?

Answer:

Figure 8: An illustration depicting workflow of our model. (a) is the input image, (b) denotes the heatmap of relevant image areas to the question. Img2LLM and stochastic captions are combined to form a unified context (depicted in the figure as Captions). QA pairs are generated exclusively from the captions’ keywords (depicted in black shading). Two QA selection strategies, template and neural, are denoted by black and blue, respectively. The input question is appended at the end of the prompt and marked in red.

5.2 Evaluation Metrics

To properly assess the performance of our model, we employ evaluation methodologies provided by the VQA dataset and further address instances of false negatives through additional means.

5.2.1 Standard VQA evaluation

Following the recommendations outlined in the VQA dataset [1], we adopt an evaluation metric robust to inter-human variability in answer phrasing. The formula utilized is presented as follows:

$$Acc(ans) = \min\left\{\frac{\#humans \text{ that said } ans}{3}, 1\right\}$$

Specifically, for each question, we select 9 human-provided answers from the 10 available options and ascertain the degree of agreement between our proposed answer and the

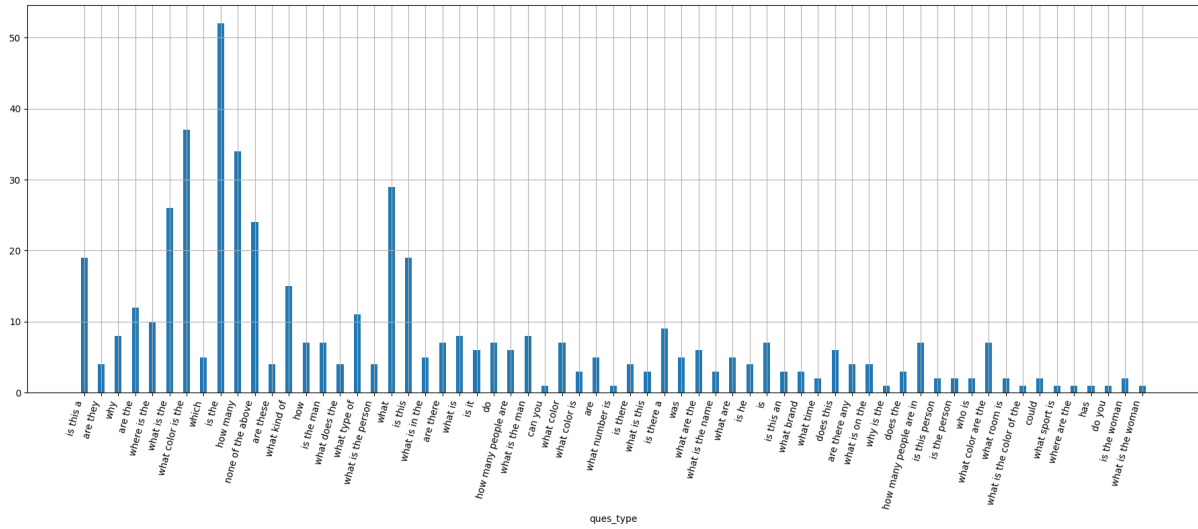


Figure 9: Question type distribution on our dataset. Among the 500 questions we use for our dataset, the question type “is the” has the highest frequency.

ground truth response. The accuracy is then computed as an average over 10 sets of responses. To ensure the accuracy of evaluation, preprocessing steps are applied to the result answer prior to comparison, including:

- Lowercasing all characters.
- Removing periods, except where occurring as decimal points.
- Converting number words to digits.
- Removing articles (e.g., a, an, the).
- Correcting contractions by adding apostrophes where missing (e.g., dont to don’t).
- One problem that comes from using Mistral is that Mistral sometimes generates QA pairs after the real answer as part of the answer. We need to remove all the Mistral-generated QA pairs after the answer.
- Replacing all punctuation marks (excluding apostrophes and colons) with space characters. Apostrophes are retained to avoid incorrect pluralization of possessives (e.g., girl’s to girls), while colons, often denoting time, are preserved (e.g., 2:50 pm). In instances where commas occur between digits, no space is inserted, ensuring continuity (e.g., converting 100,978 to 100978). (Note that this preprocessing step is also applied to ground truth answers.)

5.2.2 Cosine Similarity

Despite the implementation of preprocessing steps to enhance the fairness and reliability of our evaluation metrics, we must acknowledge the existence of several limitations. One prominent issue is the existence of false negative instances stemming from strict matching



(a) image

Question: **What is the slatted object?**

Ground Truth Answer: “bench”, “bench”, “bench”, “bench”, “bench”, “bench”, “bench”, “maybe”, “bench”.

Result Answer: **wooden bench**

Figure 10: Example of false negative caused by strict match, where “wooden bench” represents a correct answer but is classified as wrong by strictly matching metrics.

criteria. Like the one shown below:

In the provided example, “wooden bench” represents a correct and more specific answer than the ground truth answer “bench”. Consequently, categorizing the answer as false under strict matching criteria would be unfair, which gives rise to an evaluation metric capable of comprehending the semantic meaning of answers for a fair assessment of our system’s performance.

The evaluation metric we propose involves computing the cosine similarity between words, drawing inspiration from the methodology outlined in CLIP [12]. Initially, we vectorize the preprocessed result and ground truth answers using spaCy. Subsequently, we use the cosine similarity method from `sklearn` to calculate the cosine similarity between the vectors. Rather than adhering strictly to a binary matching criterion, we introduce a threshold parameter to gauge the degree of similarity between two examples.

5.3 Experiments

5.3.1 Setup

To generate exemplary question-and-answer pairs, we first extract keywords from question-relevant captions as answers. Then, questions are generated based on the n most frequent answers selected from the set of extracted keywords.

In our experiments, we compare different variants of the question generation methods:

- **template.** Questions are sampled from a template pool based on the POS type for each selected answer.
- **neural.** We leverage the T5-base model [13] to generate questions using the generated captions and selected answers.

In addition, we investigate the impact of the following hyperparameters:

- number of question-relevant captions: [5, 15, 25]
- number of exemplar question-and-answer pairs: [10, 20, 30]
- number of stochastic captions, which depend only on the image: [0, 5, 15, 25]

and keep all the other hyperparameters fixed (Appendix D). Note that the stochastic captions are appended after the question-relevant captions in Img2LLM.

5.3.2 Limitations

As mentioned in Section 4.3.2, one of the problems we met while implementing the model is that we cannot provide as much context to the LLM as the original implementation of Img2LLM. For models using stochastic captioning, when increasing the number of question-and-answer pairs and captions per image, the prompt they generate starts to exceed the maximum token length at some point (Fig 11). Moreover, the questions generated using the neural-based method contain more context but are usually longer than the template questions. The prompt consisting of neural-generated questions can hold even fewer question-and-answer pairs (Fig 11b). Under these conditions, we cannot include as many question-and-answer pairs as we intended.

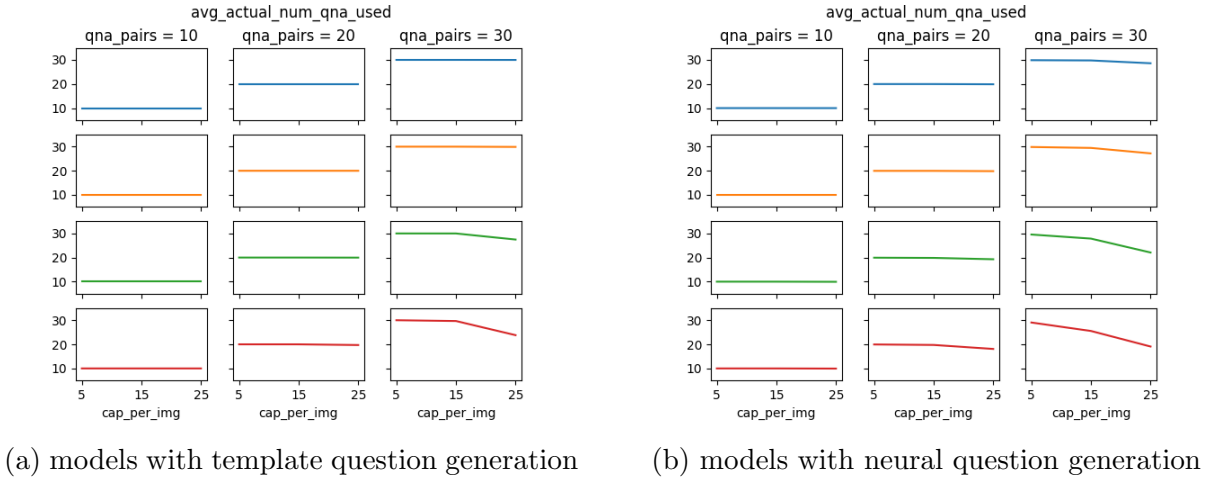


Figure 11: Average number of actual question-and-answer pairs used with respect to the configured number of question-relevant captions and QA pairs. The configured number of question-and-answer pairs is fixed in each column, and the horizontal axis in each line chart represents the number of question-relevant captions. The blue, yellow, green, and red lines represent models using 0, 5, 15, and 25 stochastic captions, respectively.

5.4 Results

5.4.1 Performance comparison

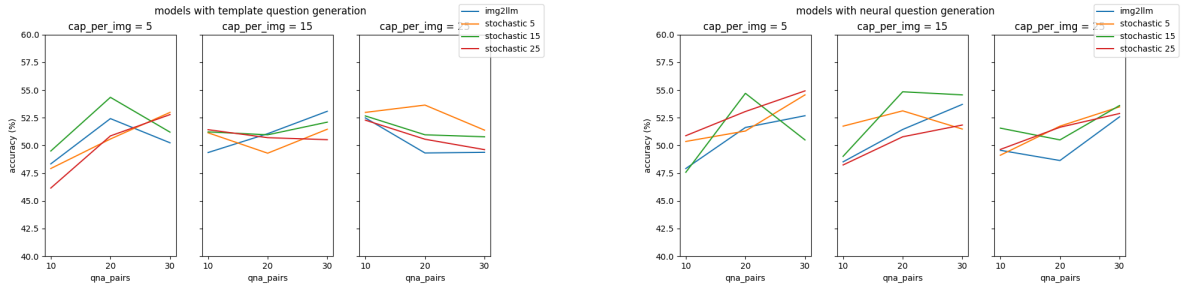
Our implementation model vs. original Img2LLM

Among the configurations listed in Section 5.3.1, the model using 5 captions per image, 30 question-and-answer pairs, 25 stochastic captions, and leveraging the neural-based question generation method achieves **the highest accuracy on our dataset with an accuracy of 54.92**. In comparison, the original Img2LLM with 6.7B parameters reaches an accuracy 57.6 on the VQAv2 validation set.

Despite sharing similar model sizes, our implemented model with Mistral 7B does not outperform the original Img2LLM 6.7B. This may be due to several reasons, such as the previously mentioned maximum token length and the size of our dataset. Note that while the performance of Img2LLM is measured on the VQAv2 validation set, as mentioned in Section 5.1, our model is measured on the extracted subset, which has a much smaller size. Therefore, the results could be biased and noisy. Nevertheless, we can still see if the stochastic captioning method can improve the performance of our Img2LLM model.

Effect of hyperparameters

Generally, there is no significant increase in accuracy either when increasing the number of question-and-answer pairs and captions or when adding stochastic captions (Fig 12), which goes against our intuition. However, when using the neural question generation method, models with fewer question-relevant captions and more stochastic captions seem to perform better.



(a) models with template question generation

(b) models with neural question generation

Figure 12: Performance of models using different numbers of captions, question-and-answer pairs, and stochastic captions. With the number of captions fixed in each line chart, the horizontal axis shows the number of question-and-answer pairs, and the vertical axis shows the accuracy. More models using neural question are able to reach 55.0

To analyze the impact of stochastic captioning, we look into the performance of the models using none and 25 stochastic captions on all of the three answer types, respectively (Table 1). We first evaluate the performance of our Img2LLM models with 5 question-relevant captions and 30 question-and-answer pairs. Either with the template or neural-based question generation methods, after appending 25 stochastic captions, the performance improves in all answer types.

Table 1: Accuracy of models with/without stochastic captioning and template/neural question generation on different answer types. Here, the correctness of a predicted answer is defined as having at least one matching ground truth answer. Note that this is different from the VQA evaluation metric.

answer type	yes/no	other	number
total number	203	241	56
Img2LLM, template	86.7	32.0	42.9
Img2LLM, neural	84.7	39.0	37.5
Img2LLM + Stochastic Captioning, template	89.2	34.9	48.2
Img2LLM + Stochastic Captioning, neural	88.2	41.1	46.4

Question generation methods

When using more question-and-answer pairs, the models with neural-generated questions almost consistently outperform the ones with template questions (Fig 13). In addition, the top 5 highest accuracies in our experiments are all achieved by models using the neural question generation method. We hypothesize that this is because the neural-based method generates fewer duplicate questions than the template-based method. The neural-generated questions are more diverse and thus provide more context to Mistral. Therefore, using neural questions can increase the accuracy.

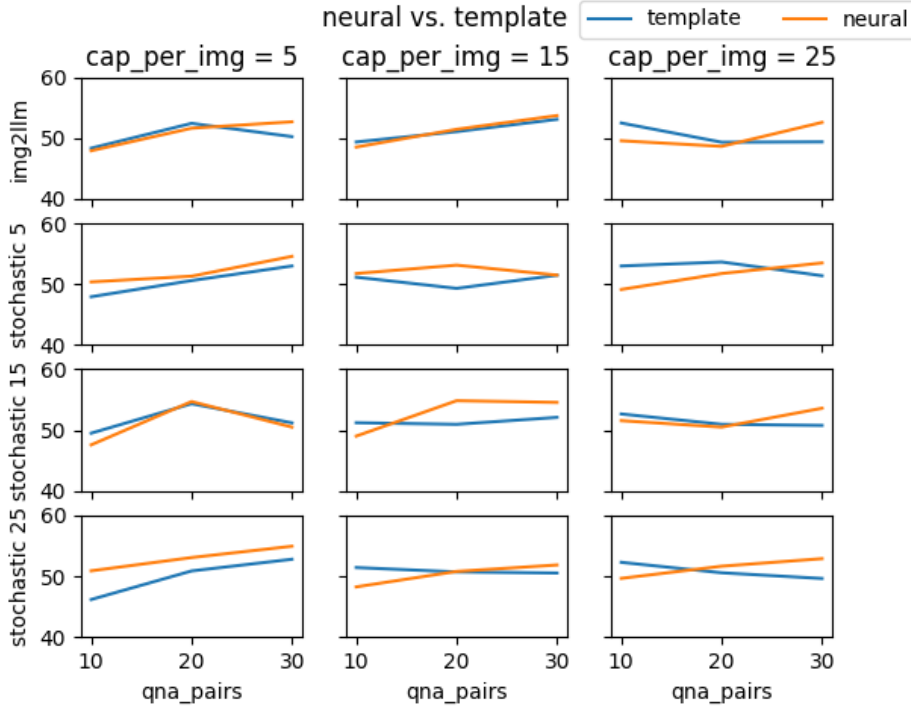


Figure 13: Performance of models using template vs. neural-based question generation methods. Different line colors indicate different question-generation methods. With the number of captions fixed in each column, the four rows from top to bottom represent models using 0, 5, 15, and 25 stochastic captions, respectively. The horizontal axis shows the number of question-and-answer pairs, and the vertical axis shows the accuracy.

5.4.2 Effect of Cosine Similarity Metrics

In this section, we illustrate the impact of cosine similarity through several examples.

Advantages: Solve part of misjudgment issues:

A. When the result answer conveys the same idea as the ground truth answers



(a) harbor

Question: **What purpose has the light-house?**

Ground Truth Answer: “**safety**”, “for boats to see shore at night”, “guide”, “to protect ships”, “to warn ships of rocks”, “scouting”, “signal”, “direction”, “to guide ships away from rocky coasts”, “to alert of dangerous places”

Result Answer: **to guide ships safely into the harbor**

cosine similarity (result, “safety”)=0.877



(b) boat

Question: **What color is the boat?**

Ground Truth Answer: “black”, “red white black”, “black red and white”, “black and white”, “red white and black”, “white black and red”, “black white and red”, “black red yellow”, “black red tan white”, “black and red and white”

Result Answer: **black**

cosine similarity (“black”, “black red and white”)=0.912

Figure 14: Example of misjudgment solved by cosine similarity. In the first example, the result answer has the same meaning as the ground truth answer. The second example indicates the case when the result answer is part of the ground truth answer.

For Figure 14a and the question “What purpose has the lighthouse?”, although the exact wording differs, both the ground truth answer “safety” and our system’s answer “to guide ships safely into the harbor” convey the notion of **safe**. The latter should be deemed as a valid response. Despite not being an exact match semantically, the cosine similarity between the resulting answer and the keyword “safety” is high, measured at 0.877.

B. When the result answer is part of the ground truth answer

For the image 14b, although the boat is predominantly red, black, and white, the predominant color is black. In this sense, our output “black” is also a valid response. While this might be deemed incorrect under strict matching criteria, it remains an acceptable answer when considering cosine similarity metrics.

Limitations: Inability to Distinguish Words of Similar Type

While cosine similarity effectively mitigates certain misjudgment issues, it also introduces challenges stemming from high similarity scores for words of the same type. For example, numerical values often exhibit elevated similarity, as demonstrated by:

$$\text{cosine_similarity}('1', '2') = 0.883$$

Similar trends are observed with colors and genders:

$$\text{cosine_similarity}('white', 'blue') = 0.780, \text{ cosine_similarity}('man', 'woman') = 0.740$$

Limitations: Inability to distinguish between different tenses of the same stem.

While cosine similarity can capture part of semantic meaning, it is unable to distinguish between different tenses of the same word stem. As we can see one example as follows:

$$\text{cosine_similarity}('rain', 'rained') = 0.654$$

For this aspect, manual modifications are necessary, and we hope that alternative methods can address this issue in the future.

Drawback: Increased Likelihood of General Answers Being Deemed Correct

Another concern that arises after introducing cosine similarity for evaluation is that general answers are more likely to be considered correct, which is not as desirable. This outcome is less desirable, as exemplified by the following scenario: when prompted with the question “what is the food called?” for an image, one would anticipate a more specific response detailing the dish, such as **salad, deviled eggs, ham, rice, and brussels sprouts**, rather than a simplistic term like “meal”. Despite the high cosine score between the result answer “meals” and one of the ground truth answers “lunch”, such a response fails to capture the specificity expected in this context.

Effect:

As previous examples show the inaccuracy if we solely rely on the similarity score for evaluation, we need to discuss their correctness based on question and answer types. For the answers that are marked as incorrect by VQA evaluation but have a high similarity score, we divide them into three parts:

- **Number-related questions:** mark as wrong, use purely VQA evaluation metrics
- **Color-related questions:** if the result answer is part of ground truth answers, we mark it as correct. If not, mark it as wrong.
- **Others:** mark as correct, rely on similarity metrics

In the end, we get an accuracy improvement of approximately 11.6% with cosine similarity.



(a) meal

Question: **What is the food called?**

Ground Truth Answer: “**salad deviled eggs ham rice and brussel sprouts**”, “ham”, “lunch”, “meat”, “rice”, “dinner”, “dinner”, “steak and rice”, “dinner”, “rice”

Result Answer: **meals**

cosine similarity(“meals”, “meat”)=0.614

cosine similarity(“meals”, “lunch”) = 0.775

Figure 15: Example of a response, where the resulting answer exhibits less specificity than expected, yet is classified as correct according to cosine similarity metrics.

5.4.3 Analysis

The primary challenge encountered in our initial attempt, as documented in Section 4, pertains to the **Hallucination issue**. This issue arises when language models generate irrelevant words for items not present in the image under consideration. Importantly, our model offers a viable solution to mitigate the Hallucination issue.



(a) women with kite

Question: **What color shoes are the girls wearing?**

Ground Truth Answer: “no shoes”, “n a”, “no shoe”, “none”, “pink”, “none”, “there are no shoes”, “none”, “pink”, “barefoot”

Result Answer: **not specified**

Figure 16: Example of a challenging question, where an irrelevant object in the image is queried. Our model can respond correctly instead of guessing a random color.

For the question “What color shoes are the girls wearing?”, posed in Figure 16a. This question is deemed challenging because, as depicted, the girls are not wearing any shoes. With the help of GradCAM, our model adeptly refrains from providing a false answer regarding the color of shoes. Consequently, the Hallucination issues encountered in our initial phase are effectively addressed.

6 Future Work

In our implementation of Img2LLM (Section 4.3.2), we use Mistral 7B as the LLM since it is the LLM that PreciTaste can provide to us. However, Mistral 7B can only accept half of the maximum token length as input. We hypothesize that this leads to lower performance of Img2LLM since we cannot provide as much context from the prompt to Mistral. On the other hand, it might be possible to increase the model’s performance by using LLM with a higher maximum token length. We leave it to future work to see how the performance would be affected if we used the LLM with the same maximum token length as the LLM of the original Img2LLM or higher.

As described in Section 5.2.2, we introduce cosine similarity as an alternative evaluator for the VQA task, addressing limitations associated with the standard VQA evaluation method, which requires strictly matching to the grounded truth answer for correctness determination. This results in many false negative instances. While cosine similarity offers a more loose matching criterion, allowing for answers with similar semantic meaning to be deemed correct, it also introduces a propensity for false positive instances. Further research on cosine similarity as VQA evaluation is required to understand its impact on each type of answer, and further improvement is needed for it to be used as an actual evaluation metric.

Lastly, we want to point out one limitation of our model. Our model cannot keep the context from the previous question and answer. Therefore, the user cannot query about the answer, and the conversation does not flow as well as using ChatGPT-4. For the model to understand the context of the whole conversation, the model might need to retain the previous conversation in the prompt. However, this is out of the scope of our research and could be another research topic.

7 Conclusion

In this paper, we propose an improvement to the Img2LLM, the Language-Mediated VQA model, by adding stochastic captions. Img2LLM creates question-relevant captions using the Image-grounded Text Encoder and GradCAM and uses the generated caption to create QA pairs. These QA pairs provided the context of the Question-and-Answer task to the model, as well as the context of the image, allowing the model to make predictions for zero-shot VQA tasks without the need for QA-specific LLM. With the introduction of stochastic captioning, we hypothesize that it could help provide even more context to the LLM, especially when the Img2LLM fails to correctly identify the relevant patches and produce question-relevant captions. Due to the limitation of Mistral 7B, we cannot achieve the same performance as the original paper. However, experiments with stochastic captions and neural question generation had higher performance than those without, proving that stochastic captions help improve performance. Moreover, we propose a new VQA evaluation technique using cosine similarity, which is designed to be less strict than the standard VQA evaluation, allowing answers with similar meanings to be evaluated as correct.

References

- [1] Aishwarya Agrawal et al. *VQA: Visual Question Answering*. 2016. arXiv: 1505.00468 [cs.CL].
- [2] Yuchen Bian et al. “On Attention Redundancy: A Comprehensive Study”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Kristina Toutanova et al. Online: Association for Computational Linguistics, June 2021, pp. 930–945. DOI: 10.18653/v1/2021.naacl-main.72. URL: <https://aclanthology.org/2021.naacl-main.72>.
- [3] Wei-Lin Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. Mar. 2023. URL: <https://lmsys.org/blog/2023-03-30-vicuna/>.
- [4] Yuxin Fang et al. *You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection*. 2021. arXiv: 2106.00666 [cs.CV].
- [5] Yash Goyal et al. “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.
- [6] Jiaxian Guo et al. *From Images to Textual Prompts: Zero-shot VQA with Frozen Large Language Models*. 2023. arXiv: 2212.10846 [cs.CV].
- [7] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL].
- [8] Junnan Li et al. *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*. 2022. arXiv: 2201.12086 [cs.CV].
- [9] Haotian Liu et al. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV].
- [10] Shilong Liu et al. *LLaVA-Plus: Learning to Use Tools for Creating Multimodal Agents*. 2023. arXiv: 2311.05437 [cs.CV].
- [11] Matthias Minderer et al. *Simple Open-Vocabulary Object Detection with Vision Transformers*. 2022. arXiv: 2205.06230 [cs.CV].
- [12] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].
- [13] Manuel Romero. *T5 (base) fine-tuned on SQUAD for QG via AP*. <https://huggingface.co/mrm8488/t5-base-finetuned-question-generation-ap>. 2021.
- [14] Christoph Schuhmann et al. “LAION-5B: An open large-scale dataset for training next generation image-text models”. In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022. URL: <https://openreview.net/forum?id=M3Y74vmsMcY>.
- [15] Alvin De Jun Tan and Bingquan Shen. *Tackling VQA with Pretrained Foundation Models without Further Training*. 2023. arXiv: 2309.15487 [cs.CV].
- [16] Anthony Meng Huat Tiong et al. *Plug-and-Play VQA: Zero-shot VQA by Conjoining Large Pretrained Models with Zero Training*. 2023. arXiv: 2210.08773 [cs.CV].

- [17] Zhengyuan Yang et al. *An Empirical Study of GPT-3 for Few-Shot Knowledge-Based VQA*. 2022. arXiv: 2109.05014 [cs.CV].
- [18] Jiahui Yu et al. *CoCa: Contrastive Captioners are Image-Text Foundation Models*. 2022. arXiv: 2205.01917 [cs.CV].

Appendix

A System message

We redesigned the system message used in LLaVA and used it to instruct Mistral to read bounding box information. The system message is prepended before the object detection model’s output to form the prompt.

You are an AI visual assistant that can analyze a single image. Each specific object location within the image is given in the form of a point, represented as two floating numbers ranging from 0 to 1. The first value corresponds to the distance between the object and the left edge, and the second value corresponds to the distance between the object and the top edge. That is, the closer the first value is to 0, the closer the object is to the left; the closer the first value is to 1, the closer the object is to the right. The closer the second value is to 0, the closer the object is to the top; the closer the second value is to 1, the closer the object is to the bottom. Using the provided information, describe the scene in a detailed manner. Instead of directly mentioning the coordinates, utilize this data to explain the scene using natural language. Include details like object counts, position of the objects, relative position between the objects. When using the information from the coordinates, directly explain the scene, and do not mention that the information source is the coordinates. Always answer as if you are directly looking at the image.

B Prompts

B.1 Image-to-Caption

In the prompt to Mistral, we first state the caption and then append the question behind with the word “Question:” in between.

[caption] Question: [question]

B.2 Image-to-BoundingBoxes

Given the object detection model output, we use the words “There is a” and “at location” to wrap the object class label and bounding box coordinates into one sentence for each detected bounding box.

[system message]
There is a [class] at location [bounding box].
Question: [question]

To combine the information about the bounding box and the caption of the cropped image, we concat the caption to the bounding box location through the words “with the caption” in the prompt of our approach for improvement.

[system message]
Detected [class] with confidence [confidence level] at location [bounding box].

with caption [caption]
Question: [question]

C Template question

Using template questions, the model uses spaCy to extract the POS of the answer. Depending on the POS, the model randomly selects one of the questions for that specific POS.

For **Noun**, the question could be

- What item is this in this picture?
- What item is that in this picture?

For **Verb**, the question could be

- What action is being done in this picture?
- Why is this item doing in this picture?
- Which action is being taken in this picture?
- What action is item doing in this picture?
- What action is item performing in this picture?

For **Adjective**, the question could be

- How to describe one item in this picture?
- What is item’s ADJ TYPE in this picture?
- What is the ADJ TYPE in this picture?

D Setup

In our model, some of the parameters are always fixed. For the Img2LLM model that we implement, the image is always split into **20 patches**, and we always select the top k **most frequent** answers to generate the QA pairs. We fixed the parameters for querying Mistral as:

Parameter	Value
Max token	50
Repetition	1.3
Temp	0.2
Top p	0.9

Table 2: Fixed parameters for Mistral