Detecting and Preventing Hallucinations in Large Vision Language Models

Anisha Gunjal[†], **Jihan Yin**[†], **Erhan Bas**[*]
Scale AI
{anisha.gunjal,jihan.yin,erhan.bas}@scale.com

Abstract

Instruction tuned Large Vision Language Models (LVLMs) have made significant advancements in generalizing across a diverse set of multimodal tasks, especially for Visual Question Answering (VQA). However, generating detailed responses that are visually grounded is still a challenging task for these models. We find that even the current state-of-the-art LVLMs (InstructBLIP) still contain a staggering 30 percent of hallucinatory text in the form of non-existent objects, unfaithful descriptions, and inaccurate relationships. To address this, we introduce **M-HalDetect**[‡], a **M**ultimodal **Hal**lucination **Detect**ion Dataset that can be used to train and benchmark models for hallucination detection and prevention. M-HalDetect consists of 16k fine-grained labels on VQA examples, making it the first comprehensive multi-modal hallucination detection dataset for detailed image descriptions. Unlike previous work that only consider object hallucination, we additionally annotate both entity descriptions and relationships that are unfaithful. To demonstrate the potential of this dataset for preference alignment, we propose fine-grained Direct Preference Optimization, as well as train fine-grained multi-modal reward models and evaluate their effectiveness with best-of-n rejection sampling. We perform human evaluation on both DPO and rejection sampling, and find that they reduce hallucination rates by 41% and 55% respectively, a significant improvement over the baseline.

1 Introduction

Large language models (LLMs) have transformed the AI landscape in recent years, scaling their training data to trillions of tokens and their parameter count to hundreds of billions[5, 17, 24]. This has unlocked powerful emergent behaviors, and seen widespread adoption through the use of chat agents such as ChatGPT.

Recently, advances in multi-modal Visual Question Answering (VQA) models have seen adoption of these large language models as the backbone for image to language generation, resulting in LVLMs[15, 6, 29]. While this has led to strides in overall VQA performance, it brings along the same challenges that plague these LLMs - a significant one being the propensity to generate hallucinations.

In language models, hallucinations occur when the model produces inaccurate or misleading factual information that cannot be supported by existing knowledge stores[9, 3]. In the context of VQA for LVLMs, hallucinations can manifest as responses containing references or descriptions of the input image that are incorrect[11]. It is essential to address and mitigate these hallucinations to enhance the reliability and accuracy of LVLMs in various real life production usecases.

[†]Equal Contribution
[*]Contributions made while working at Scale AI
[‡]Code and dataset will be publicly released

Preprint. Under review.

Figure 1: Example Annotation from the M-HalDetect Dataset. The sub-sentences of text generated by multimodal LM are tagged into categories: *Accurate, Inaccurate*, and *Analysis*.

As multimodal hallucinations are hard to programmatically detect and optimize against, we first build a diverse human-labeled dataset using VQA responses from InstructBLIP to improve detection of such hallucinations. Figure 1 illustrates an annotated example sampled from the dataset.

We then train reward models of various densities on this dataset for hallucination detection. These reward models can be used to optimize a generative VQA model, but we choose to focus on reward model quality over RL optimization in this paper, using best-of-n rejection sampling as an approximation of RL performance [2][16]. We also directly optimize InstructBLIP with fine-grained DPO (FDPO), a novel variation of DPO[20] in which we directly leverage fine grained annotation information from individual examples, rather than collecting relative preference signals from pairs of texts. Both methods show significant gains in reducing hallucination rates from InstructBLIP, and show the usefulness of our data not just for evaluating hallucination detection, but also for training to prevent hallucinations.

2 Related Work

Large Vision Language Models (LVLMs) play a crucial role in tasks such as generating text from images[10] and multimodal in-context learning[1]. Recent work has focused on utilizing instruction tuning techniques to enhance the zero-shot performance of instruction-aware LVLMs across different vision-language tasks[15, 6]. These approaches utilize GPT-4 to generate multimodal instruction tuning datasets[15] where the image context is provided to GPT-4 through symbolic representations of the image such as captions and object bounding boxes. Others combine datasets across various multi-modal tasks [6] with hand-crafted instructions, a method that has found success in training traditional LLMs[27]. This achieves state of the art performance in a variety of multi-modal tasks.

Nevertheless, a significant challenge associated with LVLMs has emerged: the propensity to generate hallucinations when generating textual output. It is essential to address and mitigate these hallucinations to enhance the reliability and accuracy of LVLMs in production usecases.

Hallucination Analysis in LVLMs Recent research [11] undertakes a systematic analysis of object hallucinations in large vision-language models (LVLMs). This work investigated the impact of various factors on hallucination occurrences, including the object distribution in the visual instruction data and the co occurance probabilities of objects with the ground truth objects in the image. To effectively evaluate the presence of hallucinations, they introduced a novel evaluation metric called POPE, which involves generating polling questions about the generated text. They find that InstructBLIP has the lowest object hallucination rates among recent LVLMs. We choose to focus on InstructBLIP for this reason, and the fact that it has state of the art performance on various multi-modal benchmarks. Another relevant contribution by Liu et al. [14] is the introduction of the LRV dataset. This dataset contains positive and negative instructions specifically designed to enhance the robustness of LVLMs against hallucination and inconsistent text generation. Furthermore, they proposed a method called GAVIE, which leverages GPT-4 to assist in evaluating preferred answer generations.

These studies collectively contribute to the understanding and mitigation of hallucination-related challenges in LVLMs, by providing evaluation metrics, datasets, and evaluation methods that enhance

the reliability and consistency of text generation in multimodal models. Our work extends the scope of the previous works by not only considering hallucinations on the pressence of objects, but also on descriptions of objects such as relative positioning or attributes. We also consider hallucinations on complex object reasoning.

Aligning to Human Preferences Despite having strong zero-shot performance on classical language benchmark datasets, pre-trained LLMs still struggle to produce longer form generations on par with those written by real humans. Supervised fine-tuning on demonstration data written by humans is not enough to bridge this gap. Recent works have focused on using Reinforcement Learning with Human Feedback (RLHF) to address this problem[23, 24, 18, 17].

RLHF uses RL, typically Proximal Policy Optimization[22], to optimize a policy model with rewards from a reward model. This reward model is typically trained on preference pairs of same-prompt generations, often sourced from the base policy model. This preference is usually given by humans, though attempts have been made to use more traditional metrics such as BLEU[19] and ROUGE[8] as proxies. Using human preferences is more effective in aligning LLMs to human preferences[23], though sees mixed results in hallucination prevention. Ouyang et al. [18] found that RLHF helps smaller (6B) language models reduce their hallucination rate, while having the opposite effect on larger models (175B). In this paper, we will focus on relatively smaller multi-modal models (7B) that can be more accessible to end users.

DPO has emerged recently as a viable alternative to RLHF for preference alignment, enabling optimization of the policy model directly without needing to train a reward model and sample rewards through reinforcement learning[20]. It has shown comparable performances with RLHF in summarization and chatbot usecases on language models, and maintains strong performance in higher temperature sampling. At the same time, it avoids the unstable and brittle process of training models with RL[7].

Fine-grained Preferences A limitation of both RLHF and DPO is their lack of fine-grained interpretability regarding what makes one generation more preferred than the other. Recent research has made significant progress in leveraging fine-grained user preferences to improve the performance and interpretability of reward models. For example, Wu et al. [28] utilize fine-grained human feedback to train multiple reward models at different density levels. These reward models covered passage level preferences as in the traditional RLHF setting, but also sentence level and sub-sentence level preferences in the form of error identification. [12] employs process supervision, providing human feedback on step-level accuracy for more robust rewards.

To extend this fine-grained feedback mechanism into the multimodal domain, we introduce a new dataset for multimodal faithfulness detection. Our dataset comprises 16,000 image and detailed description pairs, annotated at the sub-sentence level to indicate the accuracy of the generated descriptions. This dataset aims to facilitate the improvement of multimodal models by incorporating fine-grained feedback into their training process.

Similarly to [28], we train sub-sentence and sentence level reward models on this dataset. As the quality of the final policy model through RLHF is limited by the quality of the reward model, we ignore RL optimization in this paper and instead focus on evaluating the reward model. We also modify the DPO loss to directly optimize InstructBLIP over fine-grained annotations in individual generations. Both methods show demonstrable progress in lowering hallucination rates.

3 M-HalDetect : Multimodal Hallucination Detection Dataset

Dataset Description In this section, we introduce the M-HalDetect dataset that incorporates fine-grained annotations for identifying hallucinations in detailed image descriptions generated by LVLMs. The dataset comprises of image-description pairs sampled from 4,000 images taken from the *val2014* split of the Common Objects in Context (COCO) dataset [13]. The dataset is divided into a training set with 3,200 images and a development set with 800 images.

We choose to utilize the validation set of COCO to avoid potential training data regurgitation from LVLMs trained on the COCO training set. This is roughly 10% of the original COCO validation set, so it should not have too much of an impact on additional testing with the rest of the validation set.

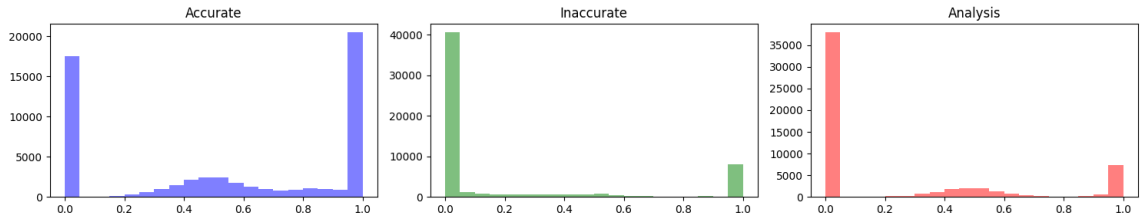

Figure 2: Class-wise Label Density histogram: We show the fine-grained dataset label distribution by computing the percentage of a sentence annotated into each category. The x-axis represents the percentage of sentence that is annotated into a class (Accurate/Inaccurate/Analysis) and the y-axis represents the frequency of such sentences in the dataset.

To generate responses, we prompt InstructBLIP[6] with each image and a randomly selected question from a pool of instructions for describing an image. An exhaustive list of question prompts is listed in Appendix B.4. We sample four responses using nucleus sampling from InstructBLIP with a temperature value set to 1.0. This creates 16k image-prompt-response triplets, split between 12800 samples in the *train* split and 3200 samples in the *val* split.

Dataset Categories The annotation process involves categorizing different segments of each response into three categories: (i) Accurate, (ii) Inaccurate, and (iii) Analysis. We also include an Unsure category for ambiguous cases. We define the classes as follows:

- **Accurate** Objects exist in the image, their descriptions are accurate according the image, and any described relationships can be accurately inferred from the image.
- **Inaccurate** Objects do not exist in the image or their descriptions are inaccurate. Furthermore, if the analysis about the image is not plausible, it is also marked as Inaccurate.
- **Analysis** Scene or object analysis including complex reasoning or interpretations about the image.
- **Unsure** This category is reserved as a last resort if annotators cannot make a judgment about the sentence segment into one of the above three categories.

We provide fine-grained annotations for these 3 categories on the detailed descriptions of images generated by the LVLM. The annotations are provided at sub-sentence level - i.e. one sentence can comprise of multiple segments from different classes. Consider, for example, a sentence describing an object present in the image, but using the wrong size modifier. In this case, the sentence would mostly be accurate, but the size would be marked inaccurate.

To make the annotation process user-friendly, we allow a leeway to the annotators to miss a few words in the annotations if there are too many segments in a sentence to be annotated. The unmarked words in a sentence are by default considered as "Accurate". In our analysis, we noticed that sometime annotators skip annotating punctuation, connector words, or introductory sub-sentences such as "The image features" (illustrated in 1).

Dataset Collection To collect the annotations, we employed Scale AI's RAPID[21] labeling tool and involved 10 randomly selected human annotators. These annotators had to qualify by passing a training course with a minimum accuracy of 85% on the example tasks to be selected for the final tagging task. The annotators are presented with an image and four responses about the image generated by InstructBLIP. Their task is to annotate segments of the sentence into one the categories. An example annotation task is illustrated in Figure 1. More details about the dataset can be found in Appendix 7.

Class-wise density distribution For each sentence in the dataset (train split), we compute densities in the form of number of words in each sentence annotated into each of the three classes. This is illustrated in a historgram at Figure 2, where the *x-axis* represents the class presence within the sentence and the *y-axis represents* represents the number of sentences.

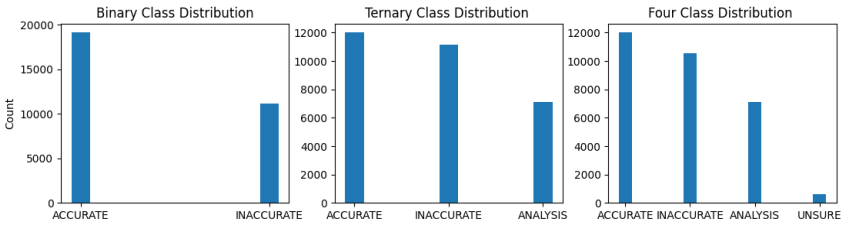

Figure 3: Class-wise Label Distribution

We can see that of the three classes, the Accurate class's densities are the least polar, while the Inaccurate class's densities are the most polar with some slight bias towards lower densities. This indicates that the sentences with inaccuracies are either fully inaccurate, or contain just a few words that are inaccurate. This matches up with the Accurate class's slight bias towards higher densities, implying that most mixed-label sentences with inaccuracies tend to comprise of inaccurate and accurate material, not analysis.

As there is a high concentration of sentences that are fully categorized into one of the classes, we consider using sentence level representation of annotations as one of the reward model baselines. More details on the generation are deferred to Section 4.1.

During our analysis, we observed that utilizing the LLaVA-150k[15] dataset's questions for generating detailed descriptions resulted in less diverse responses, potentially due to the influence of this dataset during the instruction-tuning phase of InstructBLIP. To address this, we added in our own prompts to improve generation diversity.

Further details on dataset generation, diverse prompts, and examples can be found in the Appendix 7.

Annotation Agreement Study To assess the subjectivity of this annotation task, we sample 10 tasks from the dataset, each having one image and four responses sampled from InstructBLIP. We compute agreement over these tasks by finding the number of words with the same annotation over the total number of words averaged over each task. For the first study, two authors annotate this subset and we compute agreement between the two researchers. We find a Researcher Agreement score of 83.03%. Second, we perform a agreement study between each researcher and annotators on the same subset. We find an agreement score of 81.1%. A confusion matrix for both studies in shown in Appendix A.2.

We performed qualitative analysis on the disagreements between the researchers or annotators and found a pattern in labelling differences rooting mostly between the classes (i) Accurate and Analysis, and (ii) Accurate and Inaccurate. The different interpretations of the image is attributed mainly to the subjectivity of this task or ambiguity in the descriptions. In addition, disparities in annotation can emerge when a single attribute of a phrase is incorrect. In such instances, some annotators might opt to flag the specific attribute as erroneous, while others could decide to label the entire phrase as incorrect. Attributing to these challenges in the dataset and the subjectivity of this task, we can expect a reward model trained on this dataset to have a ceiling classification performance around the same range.

4 Method

4.1 Multimodal Reward Model

We implement a multimodal reward model for detecting the presence of hallucinations generated by LVLMs. Specifically, we reuse the InstructBLIP weights and architecture, swapping the final embedding layer with a classification head. We do this as initializing the reward model from the generative model weights improves training robustness and reward generalization in later RL[30]. InstructBLIP consists of an image encoder that extracts image features, a linear mapping layer that projects these features, an instruction-aware attention layer called a QFormer that attends instructions over the projected image features, and a frozen pretrained decoder that takes the QFormer output and

Type	Method	RM Score	Human Eval
Baseline	InstructBLIP (T=0)	-0.97	0.71
DPO	IA Finetune Qformer (T=0)	$\mathbf{-0.48}$	$\mathbf{0.83}$
DPO	IA Finetune Qformer (T=1)	-0.72	0.75
DPO	IA Finetune 3 layers (T=0)	-0.87	N/A
DPO	DA Finetune Qformer (T=0)	-0.85	0.70
DPO	DA Finetune Qformer (T=1)	-1.03	0.58
DPO	DA Finetune 3 layers (T=0)	-0.87	N/A
RS	Best of 64	$\mathbf{-0.26}$	$\mathbf{0.87}$
RS	Worst of 64	-1.76	0.53
RS	Best of 16	-0.36	0.82

Table 1: Results of reward model and human eval scores. The RM score is the average log probability of the passage not containing hallucinations, while the human eval score is the percentage of content that was truthful. A perfect RM score would be 0, and a perfect human eval score would be 1.

instruction as input. For this paper, we choose to use Vicuna[25] as the frozen decoder following the original InstructBLIP.

For each image-text pair, we run one forward pass similar to [12], and set target class labels at the token concluding each sentence, masking out all other indices in the sequence. We identify these sentences using the Natural Language Toolkit[4]. We optimize with cross entropy loss.

Sentence-level Reward Prediction

We condense the M-HalDetect sub-sentence labels into sentence-level labels for a more structured reward format - this makes it more straightforward to run rejection sampling and train with RL, without worrying about segment identification. For each sentence, if there is any segment that is inaccurate, we label the entire sentence as inaccurate. Furthermore, if a sentence has a segment with the "unsure" category, we merge that sentence into the inaccurate class. We experiment with two levels of label granularity with this dataset:

- **Binary Classification**: Condense Analysis and Accurate classes into the Accurate class. In this setting we have two classes: `Accurate` and `Inaccurate`
- **Ternary Classification**: In this setting, we have three classes: `Accurate`, `Inaccurate` and `Analysis`.

The dataset distribution is visualized in Figure 3.

Segment-level Reward Prediction

We also train a finer-grained reward model that make hallucination judgments on segments of sentences as opposed to entire sentences. This can provide less noisy signal when training on annotations, especially with longer compound sentences and hallucinations isolated to small portions of a sentence. We train on this data in a similar fashion to the sentence level rewards, by labeling the end token index of each span or segment of annotated text into its corresponding label. We then mask out every other index in the sequence. As a baseline, we assume perfect localization of the annotation segments as an upper bound for the performance of this method. Future works can consider training a segment predictor in parallel with the reward model, to detect when hallucinations start and end. Since we do not do this, we cannot use this reward model for rejection sampling, and evaluate purely on classification metrics over the test set. Similar to sentence-level reward prediction baselines, we also experiment with the binary and ternary variants of the segment-level reward prediction models.

For both sentence-level and segment-level reward models, we try freezing different parts of the model, while always fine-tuning the reward head. We try freezing the entire decoder, as InstructBLIP does in its original instruction tuning. We also try freezing all but the top 3 layers of the decoder, all but the

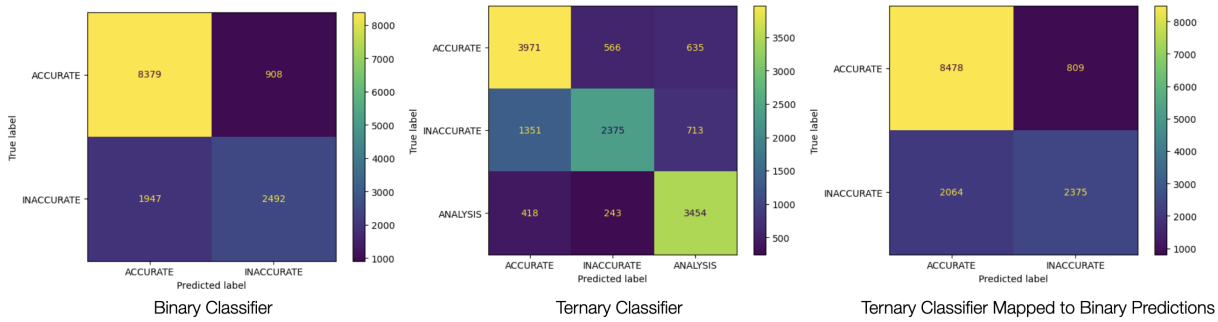

Figure 4: Confusion Matrix comparison between Binary and Ternary Classifiers. The rightmost plot represents the binary classifier labels derived from the ternary classifier by merging the Accurate and Analysis class.

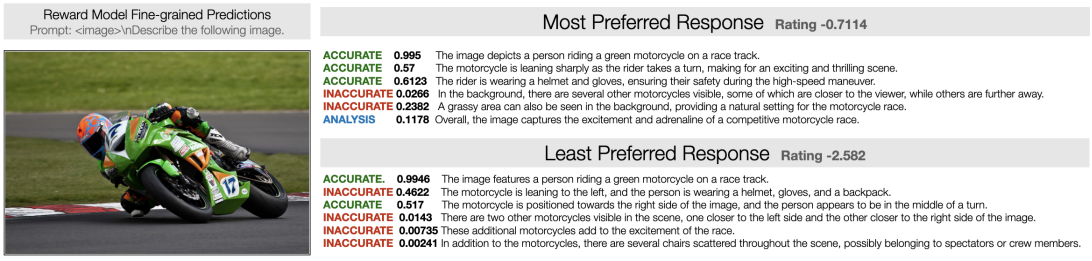

Figure 5: Rejection Sampling using Reward Model predictions to rate 64 responses sampled from InstructBLIP. Scores for sampled responses are computed using an average per sentence confidence predicted by the reward model head.

top 1, and unfreezing the entire decoder for training. Further hyperparameters and details on training can be found in Appendix A.3.

Rejection Sampling

We use the trained reward models to perform rejection sampling on the generations of InstructBLIP to promote selection of less hallucinatory responses. We do this on the passage level, computing reward scores for the whole generation at once. We calculate the reward score by averaging the non-hallucination log probabilities of each sentence. This represents the normalized log probability of the entire passage containing no hallucinations. We compute rejection sampling in a best-of-n and worst-of-n setting, for $n = 16, 64$, to study the ability of the reward model in selecting the best generations from InstructBLIP, and the variance in quality between generations. While this prohibitively slows down inference in production, this can be used as an approximation of best case performance when optimizing the source generative model with RL.

As we train two types of sentence level reward models (binary and ternary, including the analysis class), we experiment with using both models for reward scoring. We found in our initial experiments that although the binary reward model is able to penalize hallucinations with low scores, it tends to give very high scores towards the analysis class. We theorize that it is much easier to detect non-hallucinogenic analysis over factual descriptions, and as a result the binary reward model scores are biased towards generations that contain more subjective analysis rather than objective descriptions. This is not a problem with the ternary reward model, as analysis has been split into its own class, so we choose to use the best performing ternary reward model for rejection sampling moving forward.

4.2 Direct Preference Optimization

While we train a reward model to show the potential of optimizing against hallucinations with RL, we also directly optimize InstructBLIP using FDPO to reduce hallucinations.

Since M-HalDetect does not contain the traditional preference pairs used in DPO and RLHF, we explicitly segment each generation into sequences of preferred, dispreferred, and neutral chunks. We then reuse the DPO loss in increasing the likelihoods of preferred chunks while decreasing the likelihood of dispreferred chunks, each regularized by the original likelihood from the base model for the corresponding chunk, while neutral chunks are ignored. Similar to [28], this should give stronger signal during training in reducing hallucinatory generations as compared to using pairs of likelihoods over entire generations.

Recall the loss used in DPO, with π_{ref} as the reference model, π_θ as the policy model, x being the input, y_w being the preferred generation, and y_l being the dispreferred generation.

$$\mathcal{L}_{\text{DPO}}\left(\pi_\theta; \pi_{\text{ref}}\right) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta\left(y_w \mid x\right)}{\pi_{\text{ref}}\left(y_w \mid x\right)} - \beta\log\frac{\pi_\theta\left(y_l \mid x\right)}{\pi_{\text{ref}}\left(y_l \mid x\right)}\right)\right]$$

Since we don't have preferences over pairs of generations, but spans of fine-grained preferences throughout each generation, our FDPO loss can be modeled as

$$\mathcal{L}_{\text{FDPO}}\left(\pi_\theta; \pi_{\text{ref}}\right) = -\mathbb{E}_{(x,y,c)\sim\mathcal{D}}\left[\log\sigma\left(\beta k\right)\right]$$

$$k = \begin{cases} -r & c = 0 \\ r & c = 1 \\ -\infty & c > 1 \end{cases}, \qquad r = \log\frac{\pi_\theta\left(y \mid x\right)}{\pi_{\text{ref}}\left(y \mid x\right)}$$

with sample segments x, y, c being drawn from the dataset. Here, x is the entire input up until the start of the current segment, y is the generated segment, and c is the class of the current segment, with $c = 1$ being the preferred class, $c = 0$ being the dispreferred class, and all other classes being ignored. Since segments are non-overlapping, we can run a single forward pass for each sample to calculate the loss of all segments within the sample all at once.

This formulation allows us to categorize each class into positive, negative, or neutral signal, the latter of which will be ignored during training. We run ablations on including the analysis class as either a negative or neutral class when optimizing InstructBLIP with FDPO. While always fine-tuning the final unembedding layer of the decoder, we also study the effects of fine-tuning just the QFormer and fine-tuning just the top 3 layers of the decoder. We also explored fine-tuning the entire decoder in FDPO, but early experiments showed a heavy propensity towards overfitting and instable training, so we chose not to pursue that option. the We use $\beta = 0.5$ for all our FDPO experiments, and train for a maximum of 5 epochs with $lr = 10^{-6}$, warmup ratio of .03, and a cosine scheduler.

4.3 Evaluation

Recent works in multimodal LLMs[15, 14] commonly use GPT-4 as a human proxy to evaluate the helpfulness, relevance, accuracy, and the level of detail for responses generated from LMs. Specifically, GPT-4 is prompted to give a preference rating to a LM generation and that is compared with preference rating that GPT-4 generates on it's own. This metric enables automatic evaluation without depending on human evaluators.

However, using GPT-4 as a refree to score quality of responses is plagued with systematic bias such as senstitivity to positioning/ordering of responses in the context [26]. Furthermore, the OpenAI API for GPT-4 does not support image inputs which requires image context to be provided in the form of captions and object bounding boxes. In several cases, this symbolic image input can be insufficient to represent the object interactions and leads to incorrect evaluations. We performed a qualitative analysis on GPT-4's performance on LLaVA-150k's detail subset and noted that GPT-4 gave frequent inaccurate scores and explanations, failing to detect hallucinations while incorrectly penalizing correct generations. For this reason, we do not use GPT-4 for automatic evaluation of generation quality.

To combat these limitations, we use human evaluation to evaluate the hallucination rates of our rejection sampling and DPO generations. Following the same labeling instructions as the M-HalDetect, we annotate the generations into accurate, inaccurate, and analysis spans. To save on resources, we select the ternary reward model with the best f1 score for rejection sampling, and filter down DPO candidates to ones with the highest reward score from this reward model. For generations from our DPO model, we use temperature=1 and nucleus sampling.

A common tradeoff between reducing hallucinations is a reduction in helpfulness. Consider, for example, a model that outputs nothing - it does not hallucinate, yet it is not helpful either. To avoid this potential bias in our evaluation, we choose to measure the hallucination rate as the number of inaccurate words divided by the number of total words, excluding analysis segments, to calculate what percentage of useful objective content contained hallucinations.

5 Results

5.1 Classification Metrics

Type	Method	Sentence-level Prediction		Segment-level Prediction	
		Accuracy	**F1 Score**	**Accuracy**	**F1 Score**
Binary	Finetune Decoder	0.792	**0.7837**	**0.8392**	**0.8322**
Binary	FT-Decoder 3 layers	0.785	0.7691	0.8373	0.8261
Binary	FT-Decoder 1 layer	0.7841	0.774	0.8308	0.8165
Binary	Finetune Qformer	**0.7922**	0.7822	0.8341	0.8161
Ternary	Finetune Decoder	**0.714**	**0.708**	**0.772**	**0.7693**
Ternary	FT-Decoder 3 layers	0.7041	0.6964	0.7629	0.7537
Ternary	FT-Decoder 1 layer	0.7025	0.7002	0.7608	0.7546
Ternary	Finetune Qformer	0.688	0.627	0.747	0.7397

Table 2: Baseline Reward Model Results: We compare the development set performance of `sentence-level` and `segment-level` reward prediction models. Comparison is done with Accuracy and F-1 score across binary and ternary label granularities.

We evaluate the multimodal reward models (Sentence-level and Segment-level) using the development split of the M-HalDetect Dataset. We report *Accuracy* and *F-1 Score* for each of the baselines. We consider the following baselines to report the classification metrics. All baselines are initialized with pre-trained InstructBLIP weights and differ by the components of the architecture that are fine-tuned.

- **Finetune Decoder**: The Vicuna Decoder and the Reward Model head is finetuned.
- **FT-Decoder 3 layers**: The last 3 layers of Vicuna Decoder and the Reward Model head is finetuned, while keeping everything else frozen.
- **FT-Decoder 1 layer**: The final layer of Vicuna Decoder and the Reward Model head is finetuned, while keeping everything else frozen.
- **Finetune Qformer**: The InstructBLIP Qformer is finetuned along with the Reward Model head.

The results are summarized in Table 5.1. For `Binary Classification`, the fine-tuned decoder outperforms or is at par with all the other baselines. However, the performance gap between the fully fine-tuned decoder and partially fine-tuned models is not very significant. A similar trend is seen for `Ternary Classification`, but we observe a significant drop in performance for the finetuned Qformer. We theorize that this may be caused by fine-tuning a random initialized classification head at the end of the model at the same time as the QFormer present towards the start of the model. Improvements can be made on this to fine-tune the classification head first, before fine-tuning the QFormer, but we leave that to future work due to resource constraints.

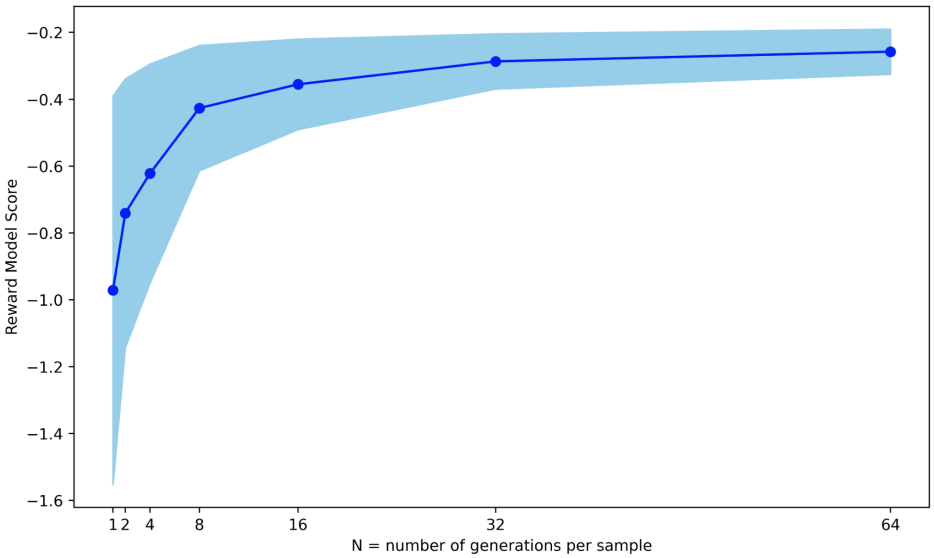

Figure 6: Reward model score means and variances as n increases in best-of-n rejection sampling. We see diminishing returns as we increase n.

5.2 Human Evaluation

Reward Model Rejection Sampling

Figure 5 illustrates an example of rejection sampling using fine-grained feedback from the reward model. The reward model is accurately able to flag hallucinatory sentences which predict presence of incorrect objects such as motorcycles and chairs. Furthermore, it is also able to flag sentences that generate analysis about non-existent objects such as "additional motorcycles add to the excitement of the race".

As we can see in table 4, rejection sampling significantly improves the factual rate of InstructBLIP's outputs, with diminishing returns as the number of generations to sample increases. On the other hand, the worst generations of InstructBLIP can be extremely poor, with an almost 50% hallucination rate! While we cannot conduct human evaluation ablations on sample size for rejection sampling due to resource constraints, we can see in figure 6 that we get exponentially diminishing returns as the sample size increases.

This shows the effectiveness of the data in reducing hallucinations, as with even just best of 16, rejection sampling being able to cut hallucination rates in half. As rejection sampling is shown to be an approximation of performance when optimizing the model through RL, we invite others to extend our work in optimizing InstructBLIP with reward models trained on our dataset.

Fine-grained DPO

To save on labeling resources, we first do a preliminary analysis of our different FDPO trained models, scoring generations using our sentence level ternary reward model with the best F1 score. We find that fine-tuning the Qformer performs better than fine-tuning the top 3 layers of the decoder, and that increasing the temperature reduces performance. We thus run human eval on FDPO models that fine-tune the QFormer, and study the effects of temperature on hallucination potency.

We try two variations of FDPO across the three classes - one that ignores analysis (IA), and one that disprefers analysis (DA), merging it with the inaccurate class. Interestingly, marking analysis as a negative class does not impact hallucination rates in a significant way when training with FDPO, and may actually worsen hallucination rates at higher temperatures. We suspect that part of the reason this might happen is that InstructBLIP's generations often have the last sentence being subjective analysis of the image, followed by an end of sequence token. Pushing down the likelihoods of generating this

sentence increases the likelihood of the generation being lengthened, potentially inducing additional hallucinations as the model runs out of accurate content to describe.

On the other hand, we see that ignoring analysis in FDPO training almost cuts hallucination rates in half. Even sampling at high temperature, generations still on average contain less hallucinations than the baseline InstructBLIP model sampled at 0 temperature, where it would have the least propensity to hallucinate. This is slightly better than best-of-16 rejection sampling, and almost as good as best-of-64 rejection sampling. We are confident that with more careful hyperparameter selection, the FDPO trained InstructBLIP model can reduce its hallucination rate even further, which we leave up to future work.

6 Conclusion

We introduce M-HalDetect, a novel multi-modal fine-grained hallucination detection dataset for benchmarking and training LVLMs to produce more truthful generations. We train fine-grained multi-modal reward models to perform rejection sampling against InstructBLIP. We also modify DPO to perform fine-grained training on M-HalDetect, avoiding the need for preference pairs and optimizing directly on individual annotations. Both methods cut the effective hallucination rate in InstructBLIP's generations by half, showing their continued capabilities in the multi-modal domain, and the usefulness of the dataset in catching and reducing hallucinations.

While we show strong performance with rejection sampling, doing so for production usecases is prohibitively slow for inference. The next step would be to optimize a generative model, perhaps InstructBLIP, using our trained reward models to create a higher quality LVLM for instruction aware VQA.

For modern day applications of training large models with fine-grained feedback through RLHF or DPO, training typically takes place over multiple iterations of model training and feedback collection. In this paper, we only perform one cycle of collecting response feedback and training. Indeed, when analyzing some of the responses, we can see hints of overfitting to our training objective - image descriptions are slightly more generic than before, and the preciseness of descriptions may have gone down. Future work can extend our dataset and methods to also account for descriptiveness and informativeness, training multiple reward models for optimizing a more robust final model.

7 Acknowledgments

We thank Sean Hendryx for his feedback and support through the internal development of this paper.

References

[1] Jean-Baptiste Alayrac et al. "Flamingo: a visual language model for few-shot learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23716–23736.

[2] Yuntao Bai et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback". In: *arXiv preprint arXiv:2204.05862* (2022).

[3] Yejin Bang et al. *A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity*. 2023. arXiv: 2302.04023 [cs.CL].

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[5] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[6] Wenliang Dai et al. *InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning*. 2023. arXiv: 2305.06500 [cs.CV].

[7] Logan Engstrom et al. "Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO". In: *CoRR* abs/2005.12729 (2020). arXiv: 2005.12729. URL: https://arxiv.org/abs/2005.12729.

[8] Kavita Ganesan. *ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks*. 2018. arXiv: 1803.01937 [cs.IR].

[9] Ziwei Ji et al. “Survey of Hallucination in Natural Language Generation”. In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38.

[10] Chunyuan Li. “Large Multimodal Models: Notes on CVPR 2023 Tutorial”. In: *arXiv preprint arXiv:2306.14895* (2023).

[11] Yifan Li et al. “Evaluating object hallucination in large vision-language models”. In: *arXiv preprint arXiv:2305.10355* (2023).

[12] Hunter Lightman et al. “Let’s Verify Step by Step”. In: *arXiv preprint arXiv:2305.20050* (2023).

[13] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312.

[14] Fuxiao Liu et al. “Aligning Large Multi-Modal Model with Robust Instruction Tuning”. In: *arXiv preprint arXiv:2306.14565* (2023).

[15] Haotian Liu et al. “Visual instruction tuning”. In: *arXiv preprint arXiv:2304.08485* (2023).

[16] Reiichiro Nakano et al. “WebGPT: Browser-assisted question-answering with human feedback”. In: *CoRR* abs/2112.09332 (2021). arXiv: 2112.09332.

[17] OpenAI. “GPT-4 Technical Report”. In: (2023). arXiv: 2303.08774 [cs.CL].

[18] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[19] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2002, pp. 311–318.

[20] Rafael Rafailov et al. “Direct preference optimization: Your language model is secretly a reward model”. In: *arXiv preprint arXiv:2305.18290* (2023).

[21] *Scale AI Rapid Portal*. https://scale.com/docs/how-rapid-works. Accessed: 2023-07-23.

[22] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347.

[23] Nisan Stiennon et al. “Learning to summarize from human feedback”. In: *CoRR* abs/2009.01325 (2020). arXiv: 2009.01325.

[24] Hugo Touvron et al. “Llama 2: Open Foundation and Fine-Tuned Chat Models”. In: (2023). arXiv: 2307.09288 [cs.CL].

[25] *Vicuna*. https://github.com/lm-sys/FastChat. Accessed: 2023-07-23.

[26] Peiyi Wang et al. “Large language models are not fair evaluators”. In: *arXiv preprint arXiv:2305.17926* (2023).

[27] Jason Wei et al. “Finetuned Language Models Are Zero-Shot Learners”. In: *CoRR* abs/2109.01652 (2021). arXiv: 2109.01652.

[28] Zeqiu Wu et al. “Fine-Grained Human Feedback Gives Better Rewards for Language Model Training”. In: *arXiv preprint arXiv:2306.01693* (2023).

[29] Qinghao Ye et al. “mPLUG-Owl: Modularization Empowers Large Language Models with Multimodality”. In: (2023). arXiv: 2304.14178 [cs.CL].

[30] Rui Zheng et al. *Secrets of RLHF in Large Language Models Part I: PPO*. 2023. arXiv: 2307.04964 [cs.CL].

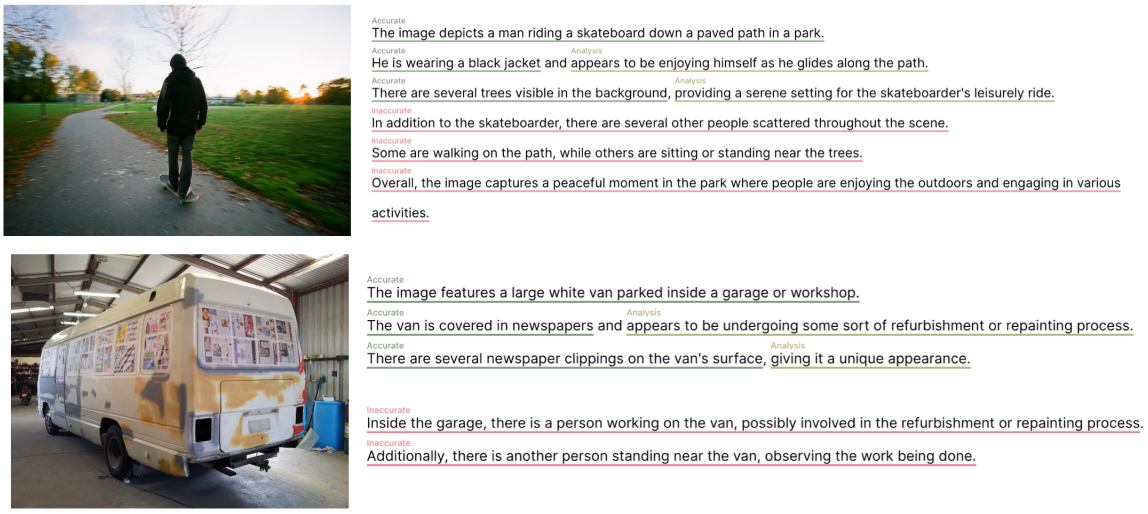

Figure 7: Example Annotations

A Data Annotation

A.1 Annotation Portal

We use Scale AI's RAPID Annotation Portal [21]. The annotators are provided with an image, question, and LM-generated detailed description of the image. For each sentence, the annotators mark parts of the sentence into appropriate cateogies: Accurate, Inaccurate, Analysis, Unsure. This is illustrated in Figure 8

A.2 Annotation Examples

We present some examples from the M-HalDetect dataset in Figure 7.

A.3 Researcher Agreement

Figure 11 illustrates the class-level analysis of researcher agreement concerning the annotation task. Differing from human agreement, this assessment was conducted by two authors of the paper who possess expertise in the field of Natural Language Processing (NLP) and a comprehensive understanding of the practical use of the trained reward models derived from this dataset. The study involves a comparison of independent annotations provided by two researchers for a consistent set of 10 images.

Due to fine-grained nature of the annotation, there are some disagreements or subjectivity for annotating individual words, especially between the accurate and inaccurate classes.

B Training Details

B.1 Training Hyperparameters

We train all models for 10 epochs with a batch size of 16 per device with a learning rate of 2e-5. The training is done with fsdp full_shard_auto_wrap mode.

B.2 Binary Classification Training Logs

In this experiment, the classifier predicts a sentence into one of two classes: Accurate and Inaccurate. Training logs are shown in 9, 10 . All models are trained for 10 epochs. Fine-tuning the entire decoder

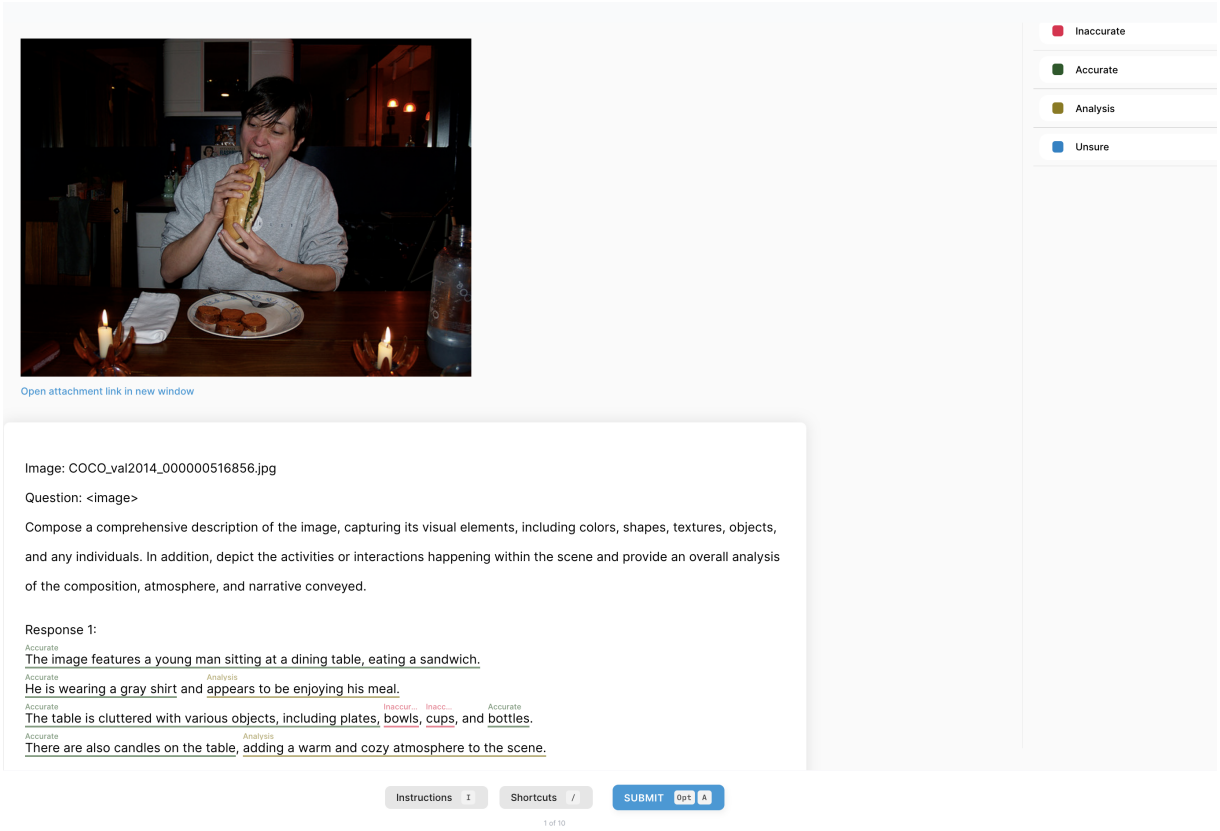

Figure 8: Scale AI RAPID Portal used for annotation.

Figure 9: Binary Classification: Sentence-level model Training and Evaluation loss, Evaluation F-1 Score and Accuracy.

Figure 10: Binary Classification: Segment-level model Training and Evaluation loss, Evaluation F-1 Score and Accuracy.

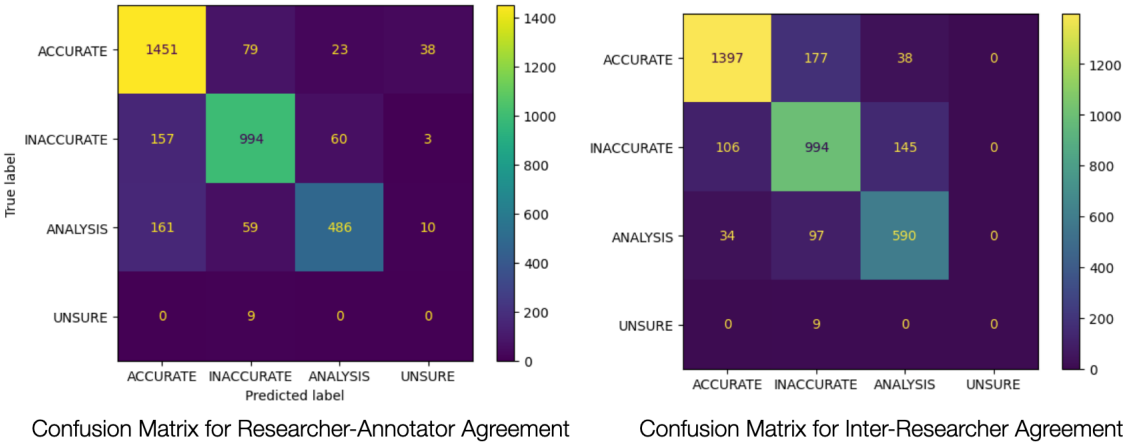

Figure 11: Confusion Matrix for class-wise researcher agreement scores for the M-Haldetect dataset's annotation task.

Figure 12: Ternary Classification: Sentence-level model Training and Evaluation loss, Evaluation F-1 Score and Accuracy.

Figure 13: Ternary Classification: Segment-level model Training and Evaluation loss, Evaluation F-1 Score and Accuracy.

model (orange) leads to over-fitting as compared to fine-tuning only last few layers of the decoder. Freezing the entire decoder and fine-tuning only the reward model head has the lowest performance.

B.3 Ternary Classification Training Logs

The training curves and evaluation metrics for training ternary reward model classifiers is shown in 12, 13. In this experiment, the classifier predicts a sentence into one of three classes: Accurate, Inaccurate, and Analysis.

B.4 Data Scaling Analysis

To study the effects of data scaling on the performance of our reward model, we ablate the amount of training data used, comparing the differences in validation accuracy and F-1. We perform this

LLaVA based question prompts

'Can you describe the main features of this image for me?\n<image>'
'<image>\nDescribe the following image.'
'<image>\nWhat are the key elements in this picture?'
"<image>\nWhat's happening in the scene?"
'Write a detailed description of the given image.\n<image>'
'Can you elaborate on the elements of the picture provided?\n<image>'
'Describe the following image.\n<image>'
'<image>\nWhat do you see happening in this image?'
'<image>\nCan you elaborate on the elements of the picture provided?'
'<image>\nCan you describe the main features of this image for me?'
"<image>\nWhat is this photo about'?"
'Explain the visual content of the image in great detail.\n<image>'
'Analyze the image in a comprehensive and detailed manner.\n<image>'
'What do you see happening in this image?\n<image>'
'What are the key elements in this picture?\n<image>'
'<image>\nExplain the visual content of the image in great detail.'
"What is this photo about'?\n<image>"
'<image>\nAnalyze the image in a comprehensive and detailed manner.'
'<image>\nWrite a detailed description of the given image.'
'<image>\nWhat do you think is going on in this snapshot?'
'What do you think is going on in this snapshot?\n<image>'
"What's happening in the scene?\n<image>"

Custom question prompts

''<image>\nPlease provide a detailed description of the image. Describe the visual elements, colors, shapes, textures, and any objects or people present along with the overall mood or atmosphere portrayed in the image.'
'<image>\nPlease provide a detailed description of the image, including its visual elements, such as colors, shapes, textures, objects, and people.'
'<image>\nProvide an intricate description of the image, capturing its visual elements, including colors, shapes, textures, objects, and any people present.'
'<image>\nDelve into the details of the image and compose a comprehensive description, incorporating its visual aspects like colors, shapes, textures, objects, and individuals.'
'<image>\nCraft an elaborate depiction of the image, highlighting its visual components such as colors, shapes, textures, objects, and the presence of any individuals.'
'<image>\nCompose a detailed account of the image, encompassing its visual characteristics, like colors, shapes, textures, objects, and any human subjects, by paying careful attention to the specifics.'
'<image>\nCompose a comprehensive description of the image, capturing its visual elements, including colors, shapes, textures, objects, and any individuals. In addition, depict the activities or interactions happening within the scene and provide an overall analysis of the composition, atmosphere, and narrative conveyed.'
'Please provide a detailed description of the image. Describe the visual elements, colors, shapes, textures, and any objects or people present along with the overall mood or atmosphere portrayed in the image.\n<image>'
'Please provide a detailed description of the image, including its visual elements, such as colors, shapes, textures, objects, and people.\n<image>'
'Provide an intricate description of the image, capturing its visual elements, including colors, shapes, textures, objects, and any people present.\n<image>'
'Delve into the details of the image and compose a comprehensive description, incorporating its visual aspects like colors, shapes, textures, objects, and individuals.\n<image>'
'Craft an elaborate depiction of the image, highlighting its visual components such as colors, shapes, textures, objects, and the presence of any individuals.\n<image>'
'Compose a detailed account of the image, encompassing its visual characteristics, like colors, shapes, textures, objects, and any human subjects, by paying careful attention to the specifics.\n<image>'
'Compose a comprehensive description of the image, capturing its visual elements, including colors, shapes, textures, objects, and any individuals. In addition, depict the activities or interactions happening within the scene and provide an overall analysis of the composition, atmosphere, and narrative conveyed.\n<image>'
'Give a detailed description of the image.\n<image>Write a detailed description of the given image.\n<image>'

Figure 14: List of prompts used for detail generation responses from InstructBLIP.

analysis on the reward model that fine-tunes last 3 layers of the InstructBLIP's LM decoder. Table B.4 shows that as the dataset size for reward model training is gradually increased from a quarter to half, the performance of the model experiences over 2 percent increase in it's F1-Score. However, beyond the half dataset size, further increments in data do not lead to substantial performance improvements and begins to saturate.

Datasize	Accuracy	F1 Score
Full Dataset	0.7489	0.7414
Half Dataset	0.7474	0.7387
Quarter Dataset	0.7375	0.7144

Table 3: Dataset Scaling: Increasing the dataset size for reward model training gives a performance boost as size increases from a quarter to half but saturates thereafter.

C Question prompts for dataset generation

Figure 14 lists the description generation-related question prompts that we use for generating data. We generate data with two sets of questions. The first set is derived from the data generation method used in LLaVA dataset [15]. The second set is a custom list of questions drafted by the authors. Question Prompts are passed to InstructBLIP [6] to sample responses.