CoMet: Context and Multiplicity Decomposition for Multimodal Uncertainty Estimation

Sanghyuk Chun **William Yang** **Amaya Dharmasiri** **Olga Russakovsky**

Princeton University

Abstract

Uncertainty estimation has been a long-standing challenge in AI models; it amounts to "knowing what you don't know," and metacognition is notoriously difficult even for humans (cf. the Dunning-Kruger effect). Although it is still far from solved even in simpler classification systems, tackling it in multimodal large language models (MLLMs) is becoming increasingly important. Within MLLMs, uncertainty can stem from any of the diverse sources as well as from their relationships, and further can stem from the unbounded answers in the open-ended setting. To tackle the issues, we propose CoMet, an MLLM uncertainty estimation method by decomposing uncertainty into a context-specific term and a multiplicity-specific term. The former captures ambiguity induced by the given context (*e.g.*, task or prompt), while the latter captures how many plausible answers determined by the context remain compatible with the given input. We train a lightweight post-hoc uncertainty module to estimate these quantities, which enables efficient uncertainty estimation without autoregressive answer generation or repeated sampling. Experiments on various open-ended multimodal benchmarks, hallucination detection, and multiple-choice visual question answering benchmarks show that CoMet consistently improves uncertainty estimation over existing baselines while remaining efficient in practice. Code is available at `https://github.com/princetonvisualai/comet_uncertainty`

1 Introduction

Multimodal large language models (MLLMs) are increasingly used in complex and open-ended settings where inputs are diverse and outputs are not always verifiable. Consequently, reliable uncertainty estimation becomes critical for downstream decision-making: beyond solely producing an answer, a model should also assign a score that reflects the ambiguity of the prediction for a given input. This score can be used for self-evaluation (whether a prediction is correct) [1–5], abstention (whether a model can correctly answer the given input) [6–9], self-improvement [10], or hallucination detection [11–13]. However, uncertainty estimation in MLLMs is challenging due to two fundamental properties: the diversity of uncertainty sources and the open-ended nature of the answer space.

First, uncertainty can arise from each modality as well as from their many-to-many relationship. Let us take a vision-language task as an example. Visual uncertainty can arise from low image quality [14, 15], *e.g.*, a blurry image is inherently more uncertain for a model. The text modality can also be intrinsically ambiguous when the text itself maps to multiple semantic meanings [16], *e.g.*, "What color is the mouse?" may refer either to the animal or to the computer equipment. Moreover, even when both the input image and the input text are individually certain, the many-to-many nature of the input-output correspondence (*i.e.*, their multiplicity [17]) can still introduce ambiguity [18–20]. For example, an image with multiple objects and a question "What is in this image?" can be mapped to multiple plausible textual descriptions [21]. As a result, uncertainty in multimodal tasks is inherently compositional, and a faithful uncertainty estimate should distinguish different sources of ambiguity.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

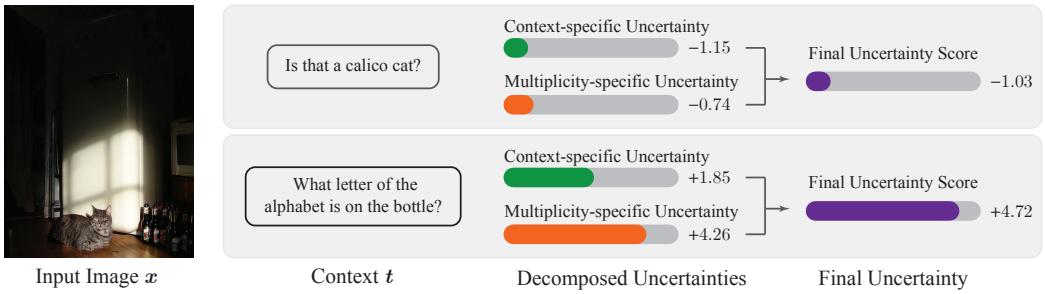


Figure 1: **Multimodal uncertainty decomposition.** Even for the same image x, uncertainty could vary by the given context t. We decompose the uncertainty of a multimodal input into two components: (1) **Context-specific uncertainty** quantifies how broadly the context t defines the answer candidates (*e.g.*, "is that a calico cat?" has two valid answer candidates, "yes" and "no", whereas "what letter of the alphabet is on the bottle?" has 26 candidates) (2) **Multiplicity-specific uncertainty** quantifies how many answer candidates (*e.g.*, "yes" and "no") remain plausible with the input x (*e.g.*, the cat certainly corresponds to a tabby cat, and therefore "no" is the only plausible answer, whereas the bottles are small and not well-lit, thus increasing rather than reducing uncertainty)

Second, the set of valid answers is unbounded in open-ended settings. Unlike closed-set tasks, where uncertainty can be measured over a fixed candidate set using quantities such as entropy [22] or logit confidence [23], open-ended tasks do not provide an explicit answer space to compare against. Instead, recent studies on LLM uncertainty quantification often utilize the generative capability of LLMs, for example, by asking a model to assess its confidence [1, 3] or approximating the open-ended answer space with a finite set of generated answers [12, 10]. However, these approaches rely heavily on generation performance, are sensitive to generation hyperparameters, and introduce an additional inference cost. In practice, these methods often require additional uncertainty-aware instruction tuning [24, 25] on the target LLM, which further increases training and deployment cost.

We aim to estimate the predictive uncertainty for MLLMs by considering both (1) the compositional nature of multimodal uncertainty and (2) the unbounded answer space of open-ended settings, without relying on the LLM generation capability at inference time or expensive model fine-tuning. To address these challenges, we propose a novel decomposition of predictive uncertainty into a context-specific term and a multiplicity-specific term. The first term captures the ambiguity induced by context t, where t determines the set of answer candidates. For example, in visual question answering (VQA) tasks, t is a text question. Fig. 1 shows an example: the question "What letter of the alphabet is on the bottle?" (26 candidates) induces a higher uncertainty than "Is that a calico cat?" (2 candidates). The second term captures how many candidates remain plausible with the given input x. For example, the cat in the image is a tabby cat; therefore, only "No" remains plausible when x is given.

We construct our uncertainty measure using an answer candidate y for the given context t and a new binary variable m. A candidate answer y is matched ($m = 1$) if it is semantically valid when question t is asked about image x. For example, let x be an image of a cat and t be the question "What is in this image?". The answer candidate $y =$"a cat" is matched ($m = 1$), while $y =$"a dog" is not ($m = 0$). Using this variable, we define a match-conditioned posterior $p(y \mid x, t, m = 1)$ and quantify its uncertainty using entropy. However, in open-ended settings, standard discrete entropy is not directly applicable because the answer space is not explicitly enumerable and can be viewed as continuous. To tackle the issue, we employ Rényi-2 entropy, a generalized entropy measure. Our derivation represents uncertainty in terms of matching probability $p(m = 1 \mid x, t, y)$ (*i.e.*, "is y a plausible answer of x, t?") without autoregressive answer generation or repeated sampling from $p_\theta(y \mid x, t)$. This avoids sensitivity to various generation hyperparameters and design choices.

In practice, we estimate the matching probability $p(m = 1 \mid x, t, y)$ using an MLLM-as-verifier strategy and approximate $p(y \mid t)$ with answers collected from visual instruction tuning datasets. This provides a finite candidate set without autoregressive candidate generation. Using these two ingredients, we train lightweight post-hoc uncertainty modules that predict our uncertainty formulation from an input x and a context t. Once trained, our uncertainty estimation module requires only x and t at inference time, without access to the candidate-answer set or any generation process.

We evaluate our uncertainty estimation framework, named **Co**ntext and **M**ultiplicity D**e**compositi**o**n (CoMEt) for Uncertainty estimation, on diverse open-ended multimodal benchmarks, including VQA v2 [26], VizWiz [27], and OK-VQA [28]. We additionally evaluate our method on a hallucination

detection benchmark [29] and multiple choice VQA benchmarks, such as MMMU [30], MMMU Pro [31], and MMStar [32]. We validate CoMet across multiple backbones, including Qwen3VL 2B, 4B, 8B Instruct [33], and InternVL3.5-1B [34]. Across these settings, CoMet more reliably identifies inputs that are likely to be incorrect than existing baselines. Importantly, our method remains generation-free at inference time, making it more efficient in practice than existing methods.

Contributions. First, we introduce a new binary matching m for an input x, context t, and answer candidate y, define a match-conditioned posterior over answer candidates (Sec. 3.1), and quantify its uncertainty with Rényi-2 entropy (Sec. 3.2). This approach yields a novel decomposition into a *context-specific term*, which quantifies how broadly the context defines the answer candidate space, and a *multiplicity-specific term*, which quantifies how many answer candidates remain plausible with the observed input. Second, we develop a practical estimator for the proposed uncertainty by employing an MLLM verifier to estimate matching probabilities (Sec. 4.1), approximating $p(y \mid t)$ by collecting answer candidates from visual instruction tuning datasets (Sec. 4.2), and training lightweight post-hoc heads to predict uncertainty from (x, t) alone (Sec. 4.3). We refer to the proposed pipeline as CoMet. Third, CoMet outperforms existing baselines across open-ended VQA, hallucination detection, and MCQ benchmarks, while we can take advantage of our decomposed components, *e.g.*, interpretable information about different sources of ambiguity (Sec. 5.3).

2 Related Work

Sampling-based uncertainty for M/LLM. Sampling-based methods estimate uncertainty by aggregating multiple outputs, *e.g.*, quantify the output consistency [10] or their semantic meaning [12, 35]. In MLLM, the diverse outputs can also be generated by perturbing input pixels [11]. While these methods can estimate meaningful uncertainty quantities, they require multiple generations, which limits their practicality. We estimate uncertainty efficiently without generation.

M/LLM-as-uncertainty estimator. Instead of generating samples repeatedly, a line of study attempts to let an LLM directly quantify its uncertainty. For example, Kadavath et al. [1] derives uncertainty by the probability of the "True" token when a user asks whether the generated output is correct. As another example, Tian et al. [3] lets a model predict its uncertainty in a verbalized form, *e.g.*, "confidence is 8/10", along with its generated output. These methods empirically show their effectiveness, but they remain tied to model-specific confidence proxies or generation behavior. On the other hand, our decomposed uncertainty formulation reduces the reliance on the model capability.

In Appendix A, we provide a more comprehensive discussion, including methods that require expensive fine-tuning and methods that decompose uncertainty. In summary, CoMet enables efficient uncertainty estimation inference using a lightweight post-hoc module, and our decomposition provides a theoretically motivated perspective that differs from existing uncertainty decompositions.

3 Context and Multiplicity Decomposition for Uncertainty Estimation

In this section, we introduce a novel decomposition of multimodal uncertainty into context-specific and multiplicity-specific terms. Our key idea is to introduce a binary variable, matching m, which indicates whether an input x and an answer candidate y are semantically compatible under a context t. We then derive a match-conditioned posterior and quantify its uncertainty using Rényi-2 entropy. Our formulation has two advantages: it provides a theoretically motivated uncertainty decomposition, and it can be applied to open-ended settings where the answers cannot be explicitly enumerated.

3.1 Matching probability to posterior

Let x denote the input, y the answer candidate, and t the context that specifies the space of y, such as a task description or a prompt. x, y and t can be continuous. In this paper, we assume the simplest scenario, where x is an image, t is a text question or instruction, and y is an answer candidate.

For a given context t, let $p(y \mid t)$ denote a context-conditioned distribution over answer candidates. For example, if t is the question "What is this animal?", y may be "a cat", "a dog", or "a calico cat". Note that we define y as an answer candidate of the given context t, therefore, it is not yet tied to a particular input x (*i.e.*, $p(y \mid x, t) = p(y \mid t)$). To connect y with an input x, we introduce a binary

variable m, where $m = 1$ indicates that candidate y is a plausible answer for input x and context t. For example, let x be an image of a cat and t be the question "What is this animal?". The candidate $y =$"a cat" is matched ($m = 1$), while $y =$"a dog" is not ($m = 0$).

Our goal is to quantify the uncertainty over plausible answers for a given input x and context t. We consider the distribution of answer candidates conditioned on $m = 1$, *i.e.*, $\pi_y := p(y \mid x, t, m = 1)$:

$$\pi_y = \frac{p(m = 1 \mid x, t, y)p(y \mid x, t)}{\int p(m = 1 \mid x, t, y')p(y' \mid x, t)dy'} = \frac{p(m = 1 \mid x, t, y)p(y \mid t)}{\int p(m = 1 \mid x, t, y')p(y' \mid t)dy'}, \tag{1}$$

The equation holds because we assume that a candidate y is drawn using t alone without relating it to a particular x. Therefore, before conditioning on $m = 1$, the candidate distribution satisfies $p(y \mid t) = p(y \mid x, t)$. Eq. (1) represents the match-conditioned posterior π_y with two quantities: (1) the matching probability $p(m = 1 \mid x, t, y)$, which measures whether candidate y is a plausible answer for the given x and t, and (2) the answer candidate distribution of the given t, *i.e.*, $p(y \mid t)$.

Our match-conditioned posterior formulation has two practical advantages. First, given $p(y \mid t)$, Eq. (1) enables the estimation of the posterior π_y by answer verification (*e.g.*, asking whether the answer is correct [1]) over all answer candidates. Therefore, our formulation does not need to estimate the answer distribution through autoregressive generation, which is sensitive to generation prompts, sampling hyperparameters, and decoding algorithms, and may produce semantically equivalent paraphrases. Second, $p(y \mid t)$ does not depend on the specific input x or the model. We can approximate it directly from answers to similar questions in an existing image-question-answer dataset. This approximated $p(y \mid t)$ can be reused across inputs and models.

3.2 Uncertainty Decomposition

Our goal is to quantify the uncertainty of the match-conditioned posterior, Eq. (1). If y is discrete and its distribution is explicitly given (*i.e.*, in closed-set settings), we can easily compute its information entropy by $-\sum_y \pi_y \log \pi_y$. For example, in classification, this sum is taken over the target classes. However, in open-ended settings, the answer space is not explicitly enumeratable. Also, if we model the answer space as continuous, we need another form of entropy defined for continuous space.

To mitigate this problem, we use a generalized notion of entropy, specifically Rényi-2 entropy, defined by $H_2(p(x)) := -\log \int_x p(x)^2 dx$. Rényi entropy is a generalized version of entropy, which is defined by $H_\alpha(p(x)) := \lim_{\gamma \to \alpha} \frac{1}{1-\gamma} \log \int p(x)^\gamma dx$; if $\alpha = 1$, it is the same as Shannon's entropy in discrete cases and differential entropy in continuous cases. The case of $\alpha = 2$, also known as collision entropy or Simpson diversity index, is particularly suitable in our case because it quantifies how concentrated the distribution is. We provide a more detailed discussion in Appendix B.1.

Let $Z_1(x, t) := \mathbb{E}_{y' \sim p(\cdot|t)}\big[p(m = 1 \mid x, t, y')\big]$ and $Z_2(x, t) := \mathbb{E}_{y' \sim p(\cdot|t)}\big[p(m = 1 \mid x, t, y')^2\big]$. We assume that $p(y \mid t)$ is a uniform distribution over a restricted region Ω_t, *i.e.*, $p(y \mid t) = \frac{1}{\int_{\Omega_t} dy} := \frac{1}{K_t}$. Then Rényi-2 entropy of Eq. (1) can be computed as follows:

$$\begin{aligned} u(x, t) &:= -\log \int \pi_y^2 dy = -\log \int \frac{p(m = 1 \mid x, t, y)^2 p(y \mid t)^2}{Z_1(x, t)^2} dy = -\log \left[\frac{1}{K_t} \frac{Z_2(x, t)}{Z_1(x, t)^2} \right] \\ &= \underbrace{\log K_t}_{:= u_t(t)} + \underbrace{2 \log Z_1(x, t) - \log Z_2(x, t)}_{:= u_{x|t}(x|t)}, \end{aligned} \tag{2}$$

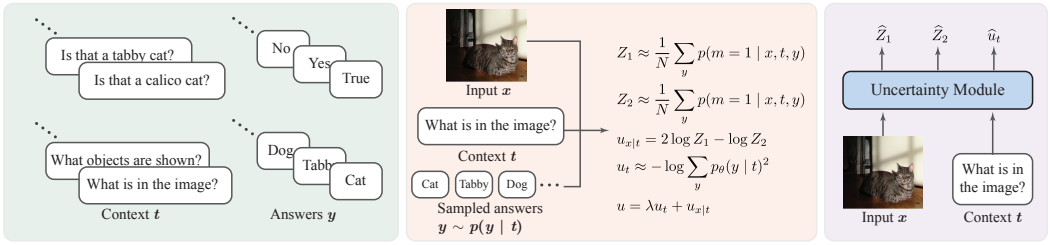
where the more detailed derivation is in Appendix B.2. In Eq. (2), the uncertainty is decomposed by (1) the context-specific term $u_t(t)$ and (2) the multiplicity-specific term $u_{x|t}(x \mid t)$. More specifically, $u_t(t)$ captures how broad the plausible answer space for the given context $p(y \mid t)$ is:

$$u_t(t) = \log K_t = -\log p(y \mid t) \underset{\text{(If } p(y \mid t) \text{ is uniform)}}{=} -\log \mathbb{E}_{Y \sim p(\cdot|t)}[p(y \mid t)] = H_2(p(y \mid t)). \tag{3}$$

In other words, if the context t is ambiguous to answer, we naturally have a high uncertainty. For example, the context "describe this image" (K_t is almost infinitely large) has a higher uncertainty than the context "Is there a cat in this image? Answer yes or no" (K_t is 2).

When we take a look into the multiplicity-specific term $u_{x|t}(x \mid t)$, we have the following relationship:

$$u_{x|t}(x \mid t) = 2 \log Z_1(x, t) - \log Z_2(x, t) = -\log \left(\frac{\mathrm{Var}_{Y \sim p(\cdot|t)}[p(m = 1 \mid x, t, Y)]}{\{\mathbb{E}_{Y \sim p(\cdot|t)}[p(m = 1 \mid x, t, Y)]\}^2} + 1 \right), \tag{4}$$



(a) Data collection for $p(y \mid t)$ (b) Training target estimation using $p(y \mid t)$ (c) Uncertainty estimation

Figure 2: **Overview of the proposed CoMet framework.** (a) We construct an answer candidate distribution $p(y \mid t)$ based on text clustering on the Cambrian dataset [36] (Sec. 4.2). (b) Using the MLLM-based matching probability estimator (Sec. 4.1), we estimate the uncertainty targets from the constructed dataset. (c) We train a lightweight uncertainty module using the constructed dataset and the estimated targets (Sec. 4.3).

which implies that $u_{x|t}(x \mid t)$ can be represented in a form related to coefficient of variation (or relative standard deviation) of the matching probability over the distribution $p(y \mid t)$. In other words, if the matching probabilities $p(m = 1 \mid x, t, y)$ are very similar for all the possible y (*i.e.*, how many y can be matched to x with the given t), then we have a high uncertainty value. We provide a more detailed discussion related to u_t and $u_{x|t}$ in Appendix B.2. We also show that we can have a similar formulation with Eq. (2) even with discrete entropy. Details are provided in Appendix B.3.

4 Practical Uncertainty Estimation

Uncertainty estimation with our formulation, Eq. (2), requires two quantities: the matching probability $p(m = 1 \mid x, t, y)$ and the context-conditioned answer candidate distribution $p(y \mid t)$. For the matching probability, we adopt an MLLM-as-verifier strategy. Estimating $p(y \mid t)$ is more challenging. We construct a answer candidate set that approximates $p(y \mid t)$, without autoregressive candidate generation, avoiding its generation-related sensitivity and cost. Using these two ingredients, we train lightweight post-hoc uncertainty modules that predict our uncertainty formulation from an input x and a context t. Once the module is trained, the uncertainty can be estimated from x and t alone without requiring any generation process or sampling process from an MLLM, which makes the overall inference efficient. The overview of our framework CoMet is illustrated in Fig. 2.

4.1 Verification as matching probability

Our decomposition requires a matching probability function $p(m = 1 \mid x, t, y)$. We can employ a parameterized matching probability module, which is a binary classifier that takes a (x, t, y) triplet as input. However, we found that training a reliable matching probability module is difficult and does not generalize well to arbitrary open-ended settings. More detailed discussions are in Appendix C.1.

Instead of training a separate matching probability module, we employ an MLLM as a verifier of matching. For example, we let an MLLM take "`Image:{x}, Question:{t}, Proposed Answer:{y}. Is this answer correct? Answer Yes or No.`", then compute the probability of "`Yes`" (the full prompt can be found in Appendix C.2). This simple technique is known to be effective in estimating self-correctness despite its simplicity [1, 37]. We additionally employ affine transform parameters (β, γ) to calibrate the matching probability as follows:

$$p(m = 1 \mid x, t, y) = \text{sigmoid}\left(\beta \log \frac{p(\text{“Yes”} \mid x, t, y)}{p(\text{“No”} \mid x, t, y)} + \gamma\right), \tag{5}$$

If $\beta = 1, \gamma = 0$, Eq. (5) becomes $\frac{p(\text{“Yes”}|x,t,y)}{p(\text{“Yes”}|x,t,y)+p(\text{“No”}|x,t,y)}$. In our experiments, we calibrate β and γ from a validation dataset. See more details in Appendix C.3.

4.2 Approximating the context-conditioned answer candidate distribution

We also require $p(y \mid t)$ for our uncertainty estimates. One possible approach is to approximate it using a language model, *i.e.*, $p(y \mid t) \approx p_\theta(y \mid t)$. However, this approach is sensitive to generation hyperparameters, such as prompts, sampling hyperparameters and decoding algorithms. Also, it is

computationally expensive as we observed in our experiment. Instead, we efficiently approximate $p(y \mid t)$ from existing image-question-answer datasets without autoregressive generation.

We first collect all question texts and their corresponding answers from the Cambrian dataset [36]. Then, we perform a simple greedy clustering on the questions. After clustering, we treat all the answers from the same cluster as the answer candidates of the questions in the cluster. For example, assume that we have a question "what is in this image?" and its answer "a cat", and another question-answer pair "what is in this photo?" and "a bird" in a cluster. In this case, both "a cat" and "a bird" are treated as answer candidates of the question "what is in this image?". Compared to sampling-based methods [12] constructing a finite answer set by repeated generation, we construct the set by reusing answers in existing datasets. This avoids dependence on generation prompts and decoding hyperparameters, while grounding candidate construction in observed question–answer pairs. We describe the details of the collection procedure in Appendix C.4.

4.3 Training objective

Now, we approximate the quantities Z_1, Z_2 (Eq. (4)), u_t (Eq. (3)), and u (Eq. (2)) defined in Sec. 3.2 using $p(m = 1 \mid x, t, y)$ in Sec. 4.1 and $p(y \mid t)$ in Sec. 4.2.

First, we approximate $u_{x|t} = 2 \log Z_1(x, t) - \log Z_2(x, t)$ by Monte Carlo approximation using a finite answer candidate set $\mathcal{Y}_t$ that approximates $p(y \mid t)$ as follows:

$$\widehat{Z}_1(x,t) = \frac{1}{|\mathcal{Y}_t|} \sum_{y' \in \mathcal{Y}_t} p(m = 1 \mid x, t, y') \quad \widehat{Z}_2(x,t) = \frac{1}{|\mathcal{Y}_t|} \sum_{y' \in \mathcal{Y}_t} p(m = 1 \mid x, t, y')^2. \tag{6}$$

Estimating $u_t(t)$ is more challenging because we cannot access the true $p(y \mid t) = \frac{1}{K_t}$ distribution, hence K_t is unknown. Instead, we employ a practical approximation using a tractable density estimator $p_\theta(y \mid t)$, *i.e.*, an autoregressive LM, based on Eq. (3) as follows:

$$u_t(t) = H_2(p(y \mid t)) \approx \widehat{u}_t(t) = -\log \sum_{y' \in \mathcal{Y}_t} p_\theta(y' \mid t)^2, \tag{7}$$

where $p_\theta(y' \mid t)$ can be computed using the autoregressive nature of LMs. Since this surrogate breaks the uniform assumption of $p(y \mid t)$, Eq. (7) is not exactly the same as the original $u_t(t)$. To control the effect of the approximation error, we add a control hyperparameter λ as follows:

$$\widehat{u}(x,t) = \lambda \widehat{u}_t(t) + 2 \log \widehat{Z}_1(x,t) - \log \widehat{Z}_2(x,t). \tag{8}$$

Using the collected data samples described in Sec. 4.2, we optimize a light-weight Transformer head that estimates $\widehat{Z}_1(x,t)$, $\widehat{Z}_2(x,t)$ and $\widehat{u}_t(t)$ using a regression loss. After training the module, we additionally employ a light-weight adaptation to the attention pooling layers and the last linear layers of the uncertainty module on a small number of data samples related to the target benchmark. More details of the model architecture and the objective functions are described in Appendix C.5.

5 Experiments

In this section, we evaluate CoMet for its capability to improve MLLM uncertainty estimation. We first evaluate CoMet on open-ended multimodal understanding, unanswerability/hallucination detection, and multiple-choice question (MCQ) settings (Sec. 5.2). Next, we examine the effectiveness of our proposed decomposition formulation by quantitative and qualitative studies (Sec. 5.3).

5.1 Experimental setup

Benchmarks. We evaluate CoMet on open-ended VQA benchmarks, including VQA v2 [26], VizWiz [27], and OK-VQA [28]. For VQA v2, we randomly subsample 10k samples from the dataset to reduce the inference cost of sampling or generation-based methods (the original VQA v2 consists of 214k samples). We additionally consider two complementary reliability tasks: unanswerability detection on VizWiz and hallucination detection on HallusionBench [29]. Finally, we evaluate CoMet on MCQ VQA benchmarks, including MMMU (test split) [30], MMMU Pro [31], and MMStar [32]. We describe more details of each benchmark in Appendix D.1.

Evaluation metrics. We report standard uncertainty estimation metrics, such as AUROC, AUPR, and FPR95. Following common practice, we treat uncertainty estimation as a binary classification problem where the uncertainty score is used to distinguish correct predictions from incorrect ones. For open-ended VQA whose evaluation is based on a soft VQA score, we use a threshold to make the prediction binary. We describe the details of our evaluation metrics in Appendix D.2.

Implementation details. Using the data collection process described in Sec. 4.2, we collect 200,000 image-question pairs, where each pair is associated with 32 plausible answers. We split this dataset into 199k training samples and 1k validation samples. Model selection is performed on the 1k validation split, and the hyperparameter search is performed on the MMMU validation split (900 samples). In our hyperparameter study, we set $\beta = 0.2$ and $\gamma = -0.1$ for Eq. (5) across all backbones. For Eq. (8), we use $\lambda = 0.25$ for all backbones except Qwen3VL-4B-Instruct, for which $\lambda = 1.0$ performs best. For adaptation, we use 900 samples of the MMMU validation split [30] and 10k randomly selected samples from VQA v2 [26], without overlap with our selected 10k test split.

We evaluate four MLLM backbones: Qwen3VL-2B-Instruct, Qwen3VL-4B-Instruct, Qwen3VL-8B-Instruct [33], and InternVL3.5-1B [34]. For each backbone, we use the selected backbone for matching probability estimator (Eq. (5)), and train a separate uncertainty module while keeping the original MLLM weight fixed. The full implementation details are provided in Appendix D.3.

Comparison methods. We compare CoMet against four groups of methods. (1) *Generation-based* that generate an answer y for a given input x and context t, and then estimate confidence from the generated output. **Negative log-likelihood (NLL)** measures the likelihood of the generated answer under $p(y \mid x, t)$, **P(Correct)** [1] estimates confidence from the token probability of "Yes" when the model is asked a follow-up question, *e.g.*, "Is this answer correct?". **Verbalized confidence** [3] asks the model to produce its confidence in text, *e.g.*, "confidence is 6/10". (2) *Sampling-based* that generate multiple answers for the same input and estimate uncertainty from their consistency. We consider **consistency entropy** and **consistency maximum probability**, both computed from the empirical answer distribution induced by repeated sampling [10]. (3) *Perturbation-based*: we compare against **VL-Uncertainty** [11], which derives MLLM uncertainty from consistency of generated outputs with perturbed images. (4) *Candidate-based baselines*: for MCQ VQA, we additionally report **candidate entropy** and **candidate maximum probability**, computed over the explicit answer options. More details of the methods are provided in Appendix D.4.

5.2 Main results

Open-ended VQA. We first evaluate CoMet on open-ended VQA benchmarks, including VQA v2, VizWiz, and OK-VQA. The first part of Tab. 1 and Tab. E.1 show that CoMet consistently outperforms existing baselines across all three benchmarks and all backbones. For example, on the Qwen3VL-2B backbone, CoMet improves the average score by +0.100 AUROC over the strongest baseline, P(Correct). Importantly, CoMet is generation-free at inference time, yet substantially outperforms MLLM-specific uncertainty estimation methods requiring repeated generation, *e.g.*, VL Uncertainty [11], with a large gap (*e.g.*, +0.135 average AUROC).

Error detection. We next evaluate whether the proposed uncertainty estimates transfer to related reliability tasks, such as unanswerability detection on VizWiz and hallucination detection on HallusionBench. They are complementary to open-ended VQA because they require uncertainty estimates to identify predictions that are unsupported or unreliable, rather than rank answer confidence. As shown in the second part of Tab. 1, CoMet achieves the best average performance across all backbones. The full results in Tab. E.2 show that CoMet achieves either the best or second-best performance on all sub-benchmarks. This supports that our decomposed uncertainty formulation can be extended to more general uncertainty use cases.

MCQ VQA. Finally, we evaluate CoMet on MCQ VQAs, including MMMU, MMMU Pro, and MMStar. Although our formulation is designed for open-ended settings rather than closed-set MCQ tasks, Tab. 1 and Tab. E.3 show that CoMet remains highly competitive in this structured setting. In most cases, CoMet outperforms the comparison methods; for some cases, such as InternVL3.5-1B, CoMet achieves the second-best performance after consistency-based baselines. We presume that this is because MCQ benchmarks provide an explicit candidate set, making repeated sampling particularly effective for estimating answer instability by reducing the search space. Nevertheless, CoMet remains competitive without repeated answer generation, suggesting that the proposed decomposed uncertainty captures semantic ambiguity that transfers beyond open-ended settings.

Table 1: **Main comparison results.** We report AUROC (AUC), AUPRC (AP), and FPR@95 of uncertainty estimation methods on three settings: open-ended multimodal understanding, error detection, and MCQ VQA. We report the average score for each setting; the full results are in Appendix E.1. "GF" denotes generation-free.

		Qwen3VL-2B-Instruct			Qwen3VL-4B-Instruct			Qwen3VL-8B-Instruct			InternVL3.5-1B		
	GF	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓
Open-ended multimodal understanding benchmarks (Average)													
Consist. (Ent) [10]	✘	.624	.593	.918	.661	.464	.899	.657	.548	.904	.495	.411	.959
Consist. (Max) [10]	✘	.623	.593	.914	.660	.464	.899	.655	.547	.904	.495	.411	.959
NLL	✘	.569	.545	.963	.491	.357	.974	.475	.417	.970	.576	.433	.949
Verbalized Conf [3]	✘	.591	.592	.842	.562	.378	.896	.642	.598	.787	.639	.546	.840
P(Correct) [1]	✘	.685	.648	.863	.712	.536	.837	.714	.652	.766	.667	.532	.897
VL Uncertainty [11]	✘	.650	.646	.852	.654	.605	.829	.654	.665	.829	.637	.535	.869
CoMET (Ours)	✔	**.785**	**.784**	**.707**	**.784**	**.631**	**.717**	**.799**	**.728**	**.701**	**.734**	**.647**	**.771**
Unanswerability/hallucination detection benchmarks (Average)													
Consist. (Ent)	✘	.586	.519	.886	.599	.438	.877	.614	.442	.886	.493	.441	.926
Consist. (Max)	✘	.584	.519	.892	.598	.434	.904	.610	.441	.886	.493	.441	.926
NLL	✘	.506	.465	.961	.547	.377	.943	.499	.353	.962	.514	.432	.950
Verbalized Conf	✘	.599	.567	.953	.548	.397	.930	.612	.503	.911	.615	.560	.954
P(Correct)	✘	.672	.619	.817	.648	.474	.800	.635	.467	.844	.649	.553	.812
VL Uncertainty	✘	.534	.584	.955	.522	.380	.979	.522	.376	.975	.561	.526	.884
CoMET (Ours)	✔	**.751**	**.693**	**.710**	**.705**	**.540**	**.761**	**.701**	**.572**	**.765**	**.732**	**.649**	**.767**
Multiple choice question (MCQ) visual question answering (VQA) benchmarks (Average)													
Candidate (Ent)	✔	.719	.801	.799	.680	.676	.861	.675	.646	.846	.583	.657	.921
Candidate (Max)	✔	.708	.793	.812	.673	.664	.880	.669	.635	.867	.575	.651	.928
Consist. (Ent)	✘	.775	.899	.562	.709	.813	**.638**	.714	.791	.658	**.680**	**.804**	**.767**
Consist. (Max)	✘	.766	.893	.586	.706	.807	.678	.709	.781	.688	.674	.798	.781
NLL	✘	.191	.556	.996	.247	.446	.996	.301	.452	.980	.357	.558	.994
Verbalized Conf	✘	.617	.773	.933	.540	.676	.911	.560	.640	.899	.531	.695	.892
P(Correct)	✘	.511	.688	.970	.729	.776	.763	.733	.771	.724	.589	.716	.911
VL Uncertainty	✘	.802	.909	.482	.721	.812	.660	.722	.789	.687	.587	.715	.902
CoMET (Ours)	✔	**.842**	**.929**	**.438**	**.782**	**.843**	.655	**.783**	**.825**	**.654**	.660	.772	.851

Table 2: **Impact of the proposed CoMET uncertainty decomposition.** We compare CoMET trained with various combinations of uncertainty components in AUROC scores on each benchmark. Qwen3VL-8B-Instruct is used for the backbone, and the MMMU validation split is used for the actual hyperparameter selection.

u_t	$u_{x\|t}$	MMMU val	VQA v2	VizWiz	OK-VQA	MMMU Test	MMMU Pro	MMStar	Avg
✔	✘	.751	.691	.691	.625	.716	.649	.677	.686
✘	✔	.837	.834	.734	.735	.753	.707	**.857**	.780
✔	✔	**.842**	**.836**	**.815**	**.746**	**.770**	**.726**	.853	**.798**

Table 3: **Sensitivity to the choice of verifier.** We compare various verifier models for CoMET with the Qwen3VL-2B backbone. VW and HB denote VizWiz and HallusionBench; (U) denotes unanswerable detection.

Verifier	VQA v2	VW	OK-VQA	VW (U)	HB	MMMU	MMMU Pro	MMStar	Avg
Qwen3VL-8B	.825	.782	.739	.779	.672	.874	.815	**.830**	.789
InternVL3.5-1B	.815	.741	.716	.756	.695	.873	.827	.815	.780
Qwen3VL-2B	**.833**	**.783**	**.740**	**.788**	**.715**	**.875**	**.834**	.819	**.798**

5.3 Analysis

Effect of decomposed uncertainty targets. Tab. 2 shows that training a model with only u_t or $u_{x|t}$ performs consistently worse than using both, *e.g.*, 0.798 average AUROC with both vs. 0.780 or 0.686 with each term alone. This shows that the two terms capture complementary uncertainty signals. In particular, u_t can be effective when dataset-level context priors are informative, while $u_{x|t}$ becomes important when the visual input is necessary to resolve the ambiguity (*e.g.*, VQA v2 [26]). In Appendix E.2, we additionally compare CoMET with unified entropy targets without decomposition; CoMET achieves 0.798 average AUROC, while the unified entropy target achieves 0.729. These results support the effectiveness of our multimodal uncertainty decomposition.

Table 4: **Effect of candidate construction.** We compare candidates collected from the Cambrian dataset with candidates generated by LLM. N denotes the number of questions. Other details are the same as Tab. 3.

Source	N	VQA v2	VW	OK-VQA	VW (U)	HB	MMMU	MMMU Pro	MMStar	Avg
Cambrian	3k	**.835**	.741	.727	.751	.707	**.879**	.831	.813	.786
LLM	3k	.829	.764	.728	.770	**.715**	.878	.834	**.822**	.792
Cambrian	200k	.833	**.783**	**.740**	**.788**	**.715**	.875	**.834**	.819	**.798**

Table 5: **Impact of adaptation dataset.** The adaptation results with various datasets are shown. For Cambrian, the adaptation and the pre-training datasets are from the same source. Other details are the same as Tab. 3.

Adaptation dataset	VQA v2	VW	OK-VQA	VW (U)	HB	MMMU	MMMU Pro	MMStar	Avg
No adaptation	.580	.648	.609	.753	.572	.546	.513	.518	.592
Cambrian	.638	.714	.663	.734	.684	.641	.642	.762	.685
MMMU + VQA v2	**.833**	**.783**	**.740**	**.788**	**.715**	**.875**	**.834**	**.819**	**.798**

Table 6: **Comparison with direct post-hoc training methods.** We compare CoMet after adaptation with uncertainty modules directly trained using BCE or ranking loss on the same adaptation data. All methods use the same frozen MLLM features (Qwen3VL-2B-Instruct) and uncertainty-module architecture. "All layers" updates the entire uncertainty module; "Last layer" updates only the attention pooling layer and last linear layer.

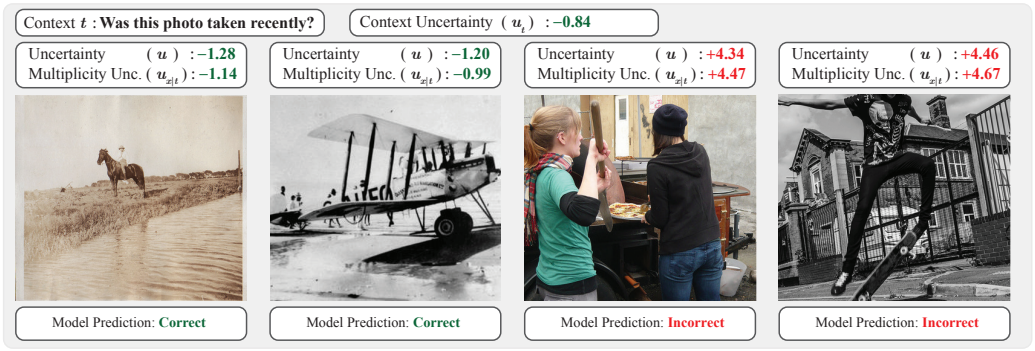
Method	Adaptation	VQA v2	VW	OK-VQA	VW (U)	HB	MMMU	MMMU Pro	MMStar	Avg
Direct (BCE)	All layers	.825	.772	.706	.773	.595	.859	**.837**	.805	.772
Direct (Ranking)	All layers	.791	.763	.694	**.793**	.680	.852	.834	.791	.775
CoMet + Adapt.	Last layer	**.833**	**.783**	**.740**	.788	**.715**	**.875**	.834	**.819**	**.798**

Sensitivity to the MLLM-as-verifier. We examine how the choice of verifier affects CoMet while keeping the Qwen3VL-2B backbone and all other settings fixed. We observe that in their individual matching decisions, the verifiers differ substantially: InternVL3.5-1B and Qwen3VL-8B disagree with Qwen3VL-2B on 35.0% and 19.3% of collected candidates from Sec. 4.2. Despite these differences, Qwen3VL-2B, Qwen3VL-8B, and InternVL3.5-1B verifiers achieve comparable average AUROCs of 0.798, 0.789, and 0.780, respectively (Tab. 3). Aggregating matching probabilities over $p(y \mid t)$ may explain this limited sensitivity. We provide more detailed discussion in Appendix E.3.

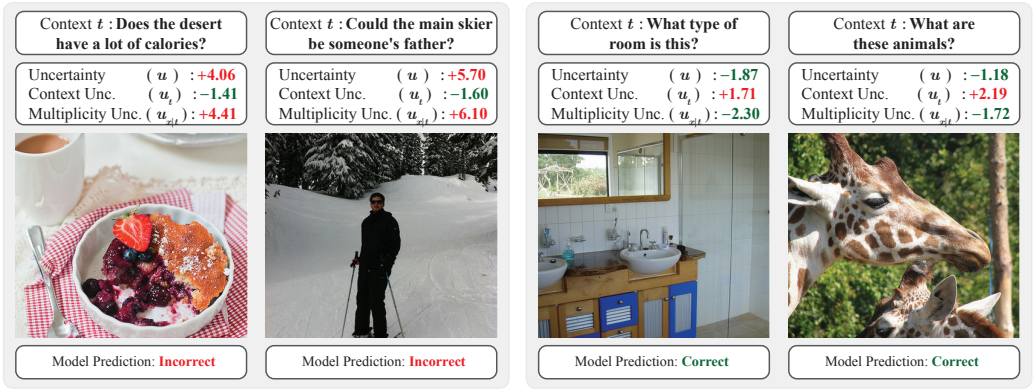
Effect of candidate construction. We compare our dataset-based candidate construction strategy introduced in Sec. 4.2 with generation-based construction. We generate 28 candidates for each question using Qwen3-8B [38]; note that we need to design a dedicated candidate-generation prompt and in-context examples, where we use our dataset-based candidates as the in-context samples – Details are in Appendix E.3. Using the same 3k questions, LM-generated candidates achieve 0.792 average AUROC, compared with 0.786 for dataset-based candidates, showing that CoMet remains effective under the alternative strategy (Tab. 4). However, LM construction for 3k questions requires 84k generations, requiring 250 L40 GPU-hours. Meanwhile, our dataset-based method only takes fewer than 2.5 L40 GPU-hours for 200k questions. As shown in Tab. 4 and Appendix E.3, increasing the number of questions or candidates per question tends to improve average performance. In summary, our candidate construction strategy provides a more scalable candidate-construction mechanism without carefully designed generation prompts or hyperparameters.

Impact of adaptation dataset. We examine how the source of adaptation data affects the downstream performance of CoMet. Tab. 5 compares the results without adaptation, with adaptation samples constructed from the pre-training dataset only, and with adaptation on held-out samples from downstream tasks. Without adaptation, CoMet achieves 0.592 average AUROC; Cambrian-only adaptation improves it to 0.685 without using downstream benchmark data, comparable to VL-Uncertainty (0.678). Appendix E.4 shows more detailed results with various adaptation datasets.

Comparison with direct post-hoc predictors. Finally, we compare CoMet with the uncertainty module trained on the same adaptation dataset. We train uncertainty modules with BCE or the proposed ranking adaptation loss in Eq. (C.2). Here, the comparison methods update the entire uncertainty module parameters, while CoMet adapts only the pooling and last linear layer. CoMet achieves 0.798 average AUROC, outperforming BCE (0.772) and ranking (0.775) post-hoc modules. The result supports that the effectiveness of CoMet is not explained by adaptation alone.



(a) Samples with the same context but different images. Sorted by its uncertainty (right is more uncertain)



(b) Samples with low context unc. but high multiplicity unc. **(c) Samples with high context unc. but low multiplicity unc.**

Figure 3: **Visualization of uncertainties of various samples.**

Controlled analysis. We examine how u_t and $u_{x|t}$ change by altering the input x or the context t. As expected, making x more ambiguous increases $u_{x|t}$ while leaving u_t fixed. When we intervene in t to allow more answers, u_t increases. The detailed discussion can be found in Appendix E.6.

Visualization of uncertain samples. Fig. 3 visualizes examples with the estimated uncertainty u and its decomposed components u_t and $u_{x|t}$. Fig. 3 (a) shows that the uncertainty can vary through $u_{x|t}$ even when the context t is fixed; Fig. 1 shows the complementary case where the image is fixed but the context changes. Since the question is a yes/no question, it yields low u_t. However, when the image does not provide enough visual evidence to determine "when the photo was taken", $u_{x|t}$ becomes high, as shown in the rightmost examples. Fig. 3(b) and (c) show cases where u_t and $u_{x|t}$ behave differently. For example, the question "What are these animals?" induces a broad answer space, but the image clearly identifies the animals as giraffes, resulting in high u_t but low $u_{x|t}$.

We additionally provide more visualizations of the learned uncertainty in Appendix E.8.

Additional analyses. We provide further analyses in the appendix. Appendix E.5 studies the effect of λ (for Eq. (8)) and (β, γ) (for Eq. (5)). Appendix E.7 reports runtime comparisons, where sampling-based baselines [10, 11] are 2.8 to 5.9 times slower than CoMet in the setting.

6 Conclusion

We introduced CoMet, a decomposition-based MLLM uncertainty estimation framework that separates context-specific ambiguity from multiplicity-specific uncertainty. By combining an MLLM-as-verifier strategy with lightweight post-hoc uncertainty heads, CoMet enables efficient uncertainty estimation in open-ended multimodal settings without autoregressive answer generation or repeated sampling. Experiments across open-ended multimodal understanding, error detection, and MCQ VQA benchmarks show that this decomposition yields more reliable uncertainty estimates than existing baselines while preserving practical inference efficiency. We hope that our decomposed formulation offers a useful step toward understanding and estimating uncertainty in MLLMs.

Acknowledgements

This work was partially supported by the Princeton Francis Robbins Upton Fellowship (SC), NSF Athena #2112562 (OR), and NSF CAREER #2145198 (OR). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF). We are also grateful to Allison Chen, Salma Abdel Magid, Tyler Zhu, Arnold Caleb Asiimwe, and Song Park for insightful discussions and valuable feedback that helped shape this work.

References

[1] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[2] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL `https://openreview.net/forum?id=8s8K2UZGTZ`.

[3] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5433–5442, 2023.

[4] Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=gjeQKFxFpZ`.

[5] Wenyi Xiao, Xinchi Xu, and Leilei Gan. Vl-calibration: Decoupled confidence calibration for large vision-language models reasoning. In *Association for Computational Linguistics (ACL)*, 2026.

[6] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Association for Computational Linguistics (ACL)*, pages 784–789, 2018.

[7] Spencer Whitehead, Suzanne Petryk, Vedaad Shakib, Joseph Gonzalez, Trevor Darrell, Anna Rohrbach, and Marcus Rohrbach. Reliable visual question answering: Abstain rather than answer incorrectly. In *European Conference on Computer Vision (ECCV)*, pages 148–166. Springer, 2022.

[8] Hanning Zhang, Shizhe Diao, Yong Lin, Yi Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. R-tuning: Instructing large language models to say 'i don't know'. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 7113–7139, 2024.

[9] Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics (TACL)*, 13:529–556, 2025.

[10] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[11] Ruiyang Zhang, Hu Zhang, and Zhedong Zheng. VL-Uncertainty: Detecting hallucination in large vision-language model via uncertainty estimation. *arXiv preprint arXiv:2411.11919*, 2024.

[12] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

[13] Seongheon Park and Sharon Li. Glsim: Detecting object hallucinations in lvlms via global-local similarity. *Advances in Neural Information Processing Systems (NeurIPS)*, 38:37747–37776, 2025.

[14] Joshua C Peterson, Ruairidh M Battleday, Thomas L Griffiths, and Olga Russakovsky. Human uncertainty makes classification more robust. In *International Conference on Computer Vision (ICCV)*, pages 9617–9626, 2019.

[15] Yichun Shi and Anil K Jain. Probabilistic face embeddings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6902–6911, 2019.

[16] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *International Conference on Learning Representations (ICLR)*, 2015.

[17] Sanghyuk Chun and Olga Russakovsky. Position: Multiplicity is an inevitable and inherent challenge in multimodal learning. In *International Conference on Machine Learning (ICML)*, 2026.

[18] Sanghyuk Chun, Seong Joon Oh, Rafael Sampaio De Rezende, Yannis Kalantidis, and Diane Larlus. Probabilistic embeddings for cross-modal retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[19] Sanghyuk Chun. Improved probabilistic image-text representations. In *International Conference on Learning Representations (ICLR)*, 2024.

[20] Sanghyuk Chun, Wonjae Kim, Song Park, and Sangdoo Yun. Probabilistic language-image pre-training. In *International Conference on Learning Representations (ICLR)*, 2025.

[21] Sanghyuk Chun, Wonjae Kim, Song Park, Minsuk Chang, and Seong Joon Oh. ECCV Caption: Correcting false negatives by collecting machine-and-human-verified image-caption associations for MS-COCO. In *European Conference on Computer Vision (ECCV)*, 2022.

[22] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[23] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[24] Sanyam Kapoor, Nate Gruver, Manley Roberts, Katherine Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew G Wilson. Large language models must be taught to know what they don't know. *Advances in Neural Information Processing Systems*, 37:85932–85972, 2024.

[25] Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training LMs to reason about their uncertainty. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://openreview.net/forum?id=ASQ649zdHm`.

[26] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6904–6913, 2017.

[27] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3608–3617, 2018.

[28] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3195–3204, 2019.

[29] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385, 2024.

[30] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567, 2024.

[31] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. Mmmu-pro: a more robust multi-discipline multimodal understanding benchmark (2025). In *Association for Computational Linguistics (ACL)*, 2025.

[32] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems (NeurIPS)*, 37:27056–27087, 2024.

[33] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[34] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

[35] Lukas Aichberger, Kajetan Schweighofer, Mykyta Ielanskyi, and Sepp Hochreiter. Improving uncertainty estimation through semantically diverse language generation. In *International Conference on Learning Representations (ICLR)*, 2025.

[36] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:87310–87356, 2024.

[37] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision (ECCV)*, pages 366–384. Springer, 2024.

[38] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[39] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, pages 1321–1330. PMLR, 2017.

[40] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059. PMLR, 2016.

[41] Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*, 2020.

[42] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[43] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:9694–9705, 2021.

[44] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.

[45] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, pages 19730–19742. PMLR, 2023.

[46] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4971–4980, 2018.

[47] Remi Cadene, Corentin Dancette, Matthieu Cord, and Devi Parikh. Rubi: Reducing unimodal biases for visual question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[48] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language Processing (EMNLP-IJCNLP)*, pages 4069–4082, 2019.

[49] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning (ICML)*, 2020.

[50] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:34892–34916, 2023.

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[53] Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights. In *International Conference on Learning Representations (ICLR)*, 2021.

[54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research (JMLR)*, 15(1):1929–1958, 2014.

[55] Ziang Zhou, Tianyuan Jin, Jieming Shi, and Qing Li. Steerconf: Steering llms for confidence elicitation. *arXiv preprint arXiv:2503.02863*, 2025.

[56] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[57] Yibo Li, Miao Xiong, Jiaying Wu, and Bryan Hooi. Conftuner: Training large language models to express their confidence verbally. *arXiv preprint arXiv:2508.18847*, 2025.

[58] Anita Kriz, Elizabeth Laura Janes, Xing Shen, and Tal Arbel. Prompt4trust: A reinforcement learning prompt augmentation framework for clinically-aligned confidence calibration in multimodal large language models. In *International Conference on Computer Vision Workshop (ICCVW)*, pages 1320–1329, 2025.

[59] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:24824–24837, 2022.

[60] Changdae Oh, Seongheon Park, To Eun Kim, Jiatong Li, Wendi Li, Samuel Yeh, Xuefeng Du, Hamed Hassani, Paul Bogdan, Dawn Song, and Sharon Li. Uncertainty quantification in llm agents: Foundations, emerging challenges, and opportunities. In *Association for Computational Linguistics (ACL)*, 2026.

Appendix

A More Related Work

Uncertainty estimation. Uncertainty estimation has been studied in deep neural networks. Early works in closed-set tasks (*e.g.*, classification) focused on predicting uncertainty scores based on their prediction probability, *e.g.*, entropy [22] or softmax confidence score [23]. These methods have demonstrated effectiveness on calibration [39] or out-of-distributed data detection [23]. However, if the answer space is not explicitly given, these methods are no longer applicable.

Improving uncertainty estimation via fine-tuning. Although using M/LLM as uncertainty estimator empirically shows meaningful uncertainty estimation, these methods need additional fine-tuning of the backbone LM [24, 5] to achieve a proper uncertainty estimate in practice. This introduces additional training cost and may degrade the generalizability of the original M/LLMs. CoMET aims to estimate uncertainty of a frozen MLLM by employing lightweight post-hoc uncertainty heads, resulting in efficient uncertainty estimation.

Uncertainty decomposition. A classical line of work decomposes predictive uncertainty into aleatoric uncertainty, which captures data uncertainty (how data itself is uncertain), and epistemic uncertainty (how the trained model is uncertain) [22, 40, 41]. Our decomposition is fundamentally different in its target and motivation. Rather than separating data noise from model uncertainty, we study semantic predictive uncertainty in open-ended multimodal settings, where the answer space is not explicitly enumerable and multiple semantically valid answers may exist for the same input and context. In our formulation, the context-specific term measures how broadly the context t defines the plausible answer space, while the multiplicity-specific term measures how many of those context-plausible answers remain compatible with the observed input x. Thus, our decomposition is not an aleatoric-epistemic decomposition of model uncertainty, but a semantic decomposition of answer-space uncertainty induced by context and input-answer compatibility. We argue that this distinction is important for MLLMs, because uncertainty can remain high even with a fixed frozen model, not because the model parameters are uncertain, but because the context induces many valid answers or the input supports multiple compatible answers under the context.

Concurrent with our work, Xiao et al. [5] proposed a decoupled multimodal uncertainty estimation framework that combines perturbation-based visual grounding confidence [11] with reasoning-side internal confidence based on token entropy. Unlike our method, their formulation is built on heuristic confidence proxies rather than theoretically motivated quantity. For example, there is no specific reason of why the uncertainty should be decomposed into those two quantities, and why each quantity can be measured by the proposed quantities. Furthermore, their approach focuses on reasoning-oriented MLLMs and requires backbone fine-tuning based on GRPO [42], whereas our method estimates uncertainty for general-purpose MLLMs using lightweight post-hoc modules on top of a frozen backbone.

B More discussions on uncertainty decomposition

B.1 Why Rényi entropy?

As we discussed in the main paper, when the output space is unknown or continuous, the standard discrete Shannon entropy is not directly applicable. One may consider the differential entropy ($H_d := \mathbb{E}_{Y\sim\pi_y}[-\log \pi_Y]$) as an alternative, but it is not aligned with our desired property. More specifically, differential entropy measures the spread of a continuous density with respect to a chosen reference measure. In open-ended answer spaces, any continuous representation of answers is largely arbitrary, so differential entropy would depend on representational choices rather than directly reflecting the multiplicity of semantic answers.

Assume an open-ended answer distribution has one dominant region of plausible answers with a tail of low-probability variants (illustrated in Fig. B.1). In this case, even though we fix the total probability mass assigned to the tail, by enlarging the tail support, the differential entropy is increased without bound (See Eq. (B.1)). Instead, a proper uncertainty estimate should more directly reflect the

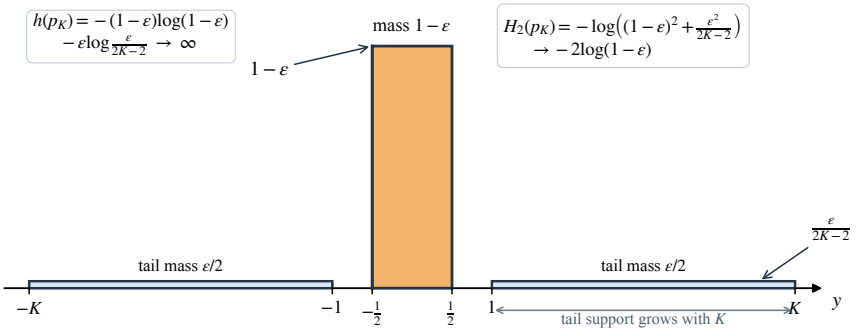


Figure B.1: **A counterexample for differential entropy.** The distribution p_K has one dominant region of mass $1-\epsilon$ and diffuse tails with total mass ϵ spread over growing support. Differential entropy diverges as $K \to \infty$, even though the high-probability semantic region remains unchanged, whereas Rényi-2 remains bounded and converges to $-2\log(1-\epsilon)$.

concentration of probability mass on the core plausible answers.

$$H(p(x)) = -\int p(x)\log p(x)dx = -(1-\varepsilon)\log(1-\varepsilon) - \varepsilon\log\frac{\varepsilon}{2k-2} \quad \text{if } k \to \infty, H \to \infty \tag{B.1}$$

On the other hand, Rényi-2 entropy remains bounded even when its tail spreads over an increasingly wide support (See Eq. (B.2)); whereas differential entropy diverges in the same setting. Therefore, it better reflects the concentration of probability mass on the core plausible answers.

$$H_2(p(x)) = -\log\int p(x)^2 dx = -\log((1-\varepsilon)^2 - \frac{\varepsilon^2}{2k-2}) \quad \text{if } k \to \infty, H_2 \to -2\log(1-\varepsilon) \tag{B.2}$$

B.2 The full derivation of uncertainty decomposition with Rényi-2 entropy

We denote $s(x,t,y) := p(m=1 \mid x,t,y)$ for simplicity. We also assume that $p(y \mid t)$ is a uniform distribution over a restricted region Ω, namely $p(y \mid t) = \frac{1}{\int_\Omega dy} = \frac{1}{K_t}$. Now, we define the first and the second moments of the matching probability over $p(y \mid t)$ as follows:

$$\begin{aligned} Z_1(x,t) &:= \mathbb{E}_{Y\sim p(\cdot|t)}\big[s(x,t,Y)\big] = \int s(x,t,y)p(y \mid t)dy \\ Z_2(x,t) &:= \mathbb{E}_{Y\sim p(\cdot|t)}\big[s(x,t,Y)^2\big] = \int s(x,t,y)^2 p(y \mid t)dy = \frac{1}{K_t}\int s(x,t,y)^2 dy \end{aligned} \tag{B.3}$$

From the definition of Rényi-2 entropy, we can derive the following decomposition:

$$\begin{aligned} u(x,t) := H_2(\pi_y) &= -\log\int \pi_y^2 dy = -\log\int \frac{s(x,t,y)^2 p(y \mid t)^2}{Z_1(x,t)^2}dy \\ &= -\log\left[\frac{1}{K_t^2}\frac{1}{Z_1(x,t)^2}\int s(x,t,y)^2 dy\right] = -\log\left[\frac{1}{K_t}\frac{Z_2(x,t)}{Z_1(x,t)^2}\right] \\ &= \underbrace{\log K_t}_{:=u_t(t)} + \underbrace{2\log Z_1(x,t) - \log Z_2(x,t)}_{:=u_{x|t}(x|t)} \\ &= \underbrace{\log K_t}_{:=u_t(t)} + \Big(\underbrace{-\log\left[\frac{\mathrm{Var}(s)}{(\mathbb{E}(s))^2}+1\right]}_{:=u_{x|t}(x|t)}\Big) = \underbrace{\log K_t}_{:=u_t(t)} + \Big(\underbrace{-\log\left[\mathrm{CV}(s)^2+1\right]}_{:=u_{x|t}(x|t)}\Big), \end{aligned} \tag{B.4}$$

where $\mathrm{CV}(\cdot)$ denotes the coefficient of variance (also known as normalized root-mean-square deviation), which is defined as follows:

$$\mathrm{CV}(s) = \frac{\sqrt{\mathrm{Var}(s)}}{\mathbb{E}(s)} = \sqrt{\frac{\mathbb{E}[s^2]}{\mathbb{E}[s]^2} - 1} = \sqrt{\frac{Z_2}{Z_1^2} - 1}. \tag{B.5}$$

Eq. (B.4) implies that the uncertainty of an input x with a given task t is given by the summation of (1) how the output space is complex ($\log K_t$), and (2) the negative coefficient of variance (also known as normalized root-mean-square deviation) of the matching probability $s(x, t, y)$. Namely, it measure how the matching probability between the input x and the answer set $\mathcal{Y}$ under the context t is distributed. Therefore, $u_{x|t}$ measures the degree of many-to-many relationships or multiplicity [17] of the matching between x and y. For example, if we have more plausible answers compatible to the given x under t, the uncertainty will be higher.

B.3 Derivation of Shannon's entropy in a discrete case

Now, we assume that the answer space y is discrete and explictly enumeratable. In this case, we can derive the following equation from the definition of discrete entropy:

$$\begin{aligned} H(\pi_y) &= -\sum_y \pi_y \log \pi_y = -\sum_y \pi_y (\log s + \underbrace{\log p(y \mid t)}_{\text{constant}} - \log Z_1(x,t)) \\ &= \log K_t - \sum_y \pi_y \log s + \sum_y \pi_y \log \underbrace{Z_1(x,t)}_{=\mathbb{E}[s]} = \log K_t - \sum_y \pi_y \log s + \log(\mathbb{E}_{Y\sim p(\cdot|t)}[s]) \end{aligned} \tag{B.6}$$

Note that $\sum_y \pi_y \log \mathbb{E}[s] = \log \mathbb{E}[s] \sum_y \pi_y = \log \mathbb{E}[s]$, because $\mathbb{E}[s]$ is no longer a function of y. Here, we rewrite $-\sum_y \pi_y \log s$ as follows:

$$-\sum_y \pi_y \log s = -\frac{1}{Z_1} \sum_y s \log s \cdot p(y \mid t) = -\frac{1}{Z_1} \mathbb{E}_{Y\sim p(\cdot|t)}[s \log s]. \tag{B.7}$$

Therefore, we have $H(\pi_y) = \log K_t + \log Z_1 - \frac{1}{Z_1}\mathbb{E}_{Y\sim p(\cdot|t)}[s \log s]$. We approximate this value using a second-order Taylor expansion. Let $\mu = \mathbb{E}[s]$ and $s = \mu + \delta$, where $\mathbb{E}[\delta] = 0, \mathbb{E}[\delta^2] = \sigma^2$. Now, apply the Taylor expansion to $f(s) = s \log s$ and take the expectation as follows:

$$\begin{aligned} \mathbb{E}_{Y\sim p(\cdot|t)}[s \log s] &= \mathbb{E}[f(\mu + \delta)] \approx \mathbb{E}\left[\mu \log \mu + (1 + \log \mu)\delta + \frac{\delta^2}{2\mu}\right] \\ &= \mu \log \mu + \mathbb{E}[\delta](1 + \log \mu) + \frac{\mathbb{E}[\delta^2]}{2\mu} = \mu \log \mu + \frac{\sigma^2}{2\mu} = Z_1 \log Z_1 + \frac{Z_2 - Z_1^2}{2Z_1}. \end{aligned} \tag{B.8}$$

Substituting this into Eq. (B.6), we have:

$$\begin{aligned} H(\pi_y) &= \log K_t + \log Z_1 - \frac{1}{Z_1}\mathbb{E}_{Y\sim p(\cdot|t)}[s \log s] \\ &= \log K_t + \log Z_1 - \log Z_1 - \frac{1}{2}\frac{Z_2}{Z_1^2} = \log K_t - \frac{1}{2}\left(\frac{Z_2}{Z_1^2} - 1\right) \end{aligned} \tag{B.9}$$

Interestingly, Eq. (B.9) is very similar to the formulation derived from Rényi-2 entropy in Appendix B.2 (the constant term is omitted):

$$H(\pi_y) \approx \log K_t - \frac{1}{2}\frac{Z_2}{Z_1^2} = \underbrace{\log K_t}_{:=u'_t(t)} + \big(\underbrace{-\frac{1}{2}\frac{\mathrm{Var}(s)}{(\mathbb{E}[s])^2}}_{:=u'_{x|t}(x|t)}\big) \tag{B.10}$$

$$H_2(\pi_y) = \log K_t - \log \frac{Z_2}{Z_1^2} = \underbrace{\log K_t}_{:=u_t(t)} + \big(\underbrace{-\log\left[\frac{\mathrm{Var}(s)}{(\mathbb{E}(s))^2} + 1\right]}_{:=u_{x|t}(x|t)}\big) \tag{B.11}$$

Both imply the same intuition: the uncertainty of the posterior π_y is decomposed by (1) the task-dependent uncertainty value $\log K_t$, *i.e.*, how the possible outcome space is large and complex, and (2) the uncertainty value by input and task, related to the normalized standard deviation of the matching probability $p(m = 1 \mid x, t, y)$ over all the possible outcome y for a given task t, $\mathrm{CV}_{Y\sim p(\cdot|t)}[s(x, t, y)]$. The Shannon information entropy is directly related to the normalized standard deviation, while Renyi entropy is related to the log of the value.

C Method details

C.1 Details of MLLM-as-verifier

Our decomposition requires a matching probability function $p(m = 1 \mid x, t, y)$. We can employ a parameterized matching probability module, which is a binary classifier that takes a (x, t, y) triplet as input. This strategy is popular in vision-language models (VLMs) training, *e.g.*, image-text matching (ITM) loss [43–45] or directly optimizing the loglikelihood of the matching probability between vision-language embeddings [18–20]. However, we found that training a reliable matching probability module is difficult for two reasons. First, it easily suffers from the context shortcut similar to the text-shortcut problem in early visual-question answering (VQA) tasks [46–49]. Also, we need a large-scale (x, t, y) triplet dataset to train a matching probability module; however, we empirically found that a matching probability module trained on a triplet dataset, such as visual instruction tuning dataset [50, 36] does not generalize to arbitrary open-ended settings.

For example, we can achieve this by using a contrastive loss [51], *i.e.*, treating the answers corresponding to the original x, t as positives and all the other answers from the other x, t in the mini-batch as negatives. However, we found that 84% the questions in the Cambrian dataset are unique, *i.e.*, if we apply contrastive learning to this dataset, the learned module is highly motivated to simply determine the matching based on t, while ignoring x.

Table C.1: **Unique answers per question in Cambrian dataset [36].**

# Unique answers	1	2	3–5	6–10	11–20	21–50	>50
# samples	811,806	120,364	22,470	7,038	2,615	945	250

As a probing example, we train a simple MLP that estimates whether the given x, t and y are matched upon CLIP-L/14 features [51]. When we test the module on the separated validation split with the same distribution with the training dataset, it shows 0.97 AUROC; however, when we mix the questions across the validation dataset, so each test question now can appear with random answers, the AUROC drops significantly to 0.51, which is almost random. In other words, it is very difficult to train matching probability module because the text short between t and y could happen.

C.2 Prompt for verification-based matching probability

Fig. C.1 shows the prompt template for our MLLM-as-verifier matching probability.

Prompts for matching probability verification

```
Image: {x}
Question: {t}
Proposed Answer: {y}
Only answer Yes if the image clearly supports the answer.
If the image is missing, unclear, or the answer is not visually confirmed,
answer No.
Answer Yes or No.
```

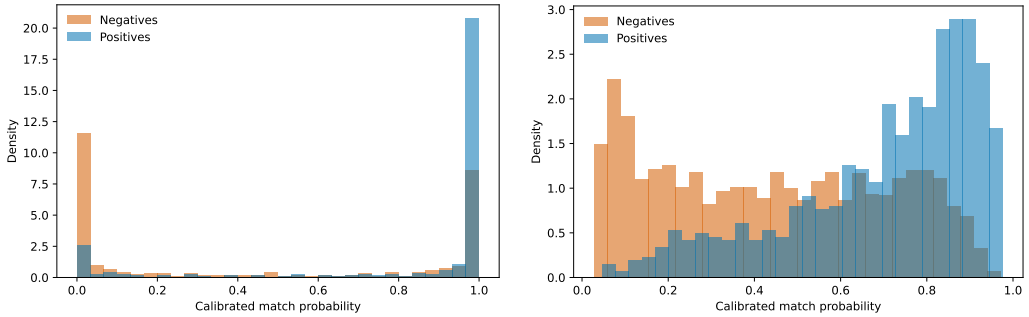
Figure C.1: **Prompt for MLLM-based matching probability.**

Using this template, we measure the $p(y = \text{“Yes”} \mid x, t)$ and $p(y = \text{“No”} \mid x, t)$ to compute the matching probability using Eq. (5).

C.3 Matching probability calibration

Recall that Eq. (5) needs affine transform variables, β and γ. This affine transform does not change the order of matching probabilities (because this is a monotonic increasing function), but it calibrates the degree of matching probability; smaller β makes the distribution of matching probability smoother, and γ controls a “reference point” for matching probability (*i.e.*, it shifts the distribution).

We apply a simple optimization to get a calibrated affine transform. We first binarize the distribution using a threshold, *i.e.*, $m = 1$ if $p(m) > 0.5$, otherwise 0, and search for β, γ that achieves the best matching accuracy in the validation dataset. We use the MMMU validation split for the calibration; because MMMU is a MCQ dataset, there exist positive samples (where $m = 1$) and negative samples (where $m = 0$) for each x and t. We achieve the best matching accuracy with $\beta = 0.2$ and $\gamma = -0.1$ for Qwen3VL-2B-Instruct. We use the same value for all the other backbones, such as Qwen3VL-4B and 8B models, and the InternVL model.



(a) Histogram before calibration

(b) Histogram after calibration

Figure C.2: **Matching probability histogram before and after calibration.** We visualize the matching probability distributions in the MMMU [30] validation split. The blue bars denote the matching probability distribution for the triplet $(x, t, y+)$ where $y+$ is the corresponding answer among the given multiple-choice questions. The orange bars denote the matching probability when a negative answer is given.

Fig. C.2 shows the distribution of matching probabilities on the MMMU validation split before and after calibration, *i.e.*, before calibration means $\beta = 1, \gamma = 0$ and after calibration means $\beta = 0.2$ and $\gamma = -0.1$. As we discussed previously, the calibration makes the overall matching probability distribution smoother.

In a later section, we will show that this calibration strategy achieves a better uncertainty estimation in the MMMU validation split (where we optimize the hyperparameter) as well as the target benchmark – See Tab. E.9 for more details.

C.4 Dataset collection details

For dataset collection, we first perform the text clustering described in Fig. C.3 to get plausible answers for each question. We compute the similarity between texts using the CLIP-L/14 text encoder [51]. After that, we randomly select 200,000 rows from the Cambrian dataset. For each example (x_i, t_i, y_i^+) assigned to the prototype p_i, we construct a candidate set $\mathcal{Y}_i$ containing the original answer and prototype positives (*i.e.*, answers corresponding to the other questions in the same cluster). Due to the computation complexity, we restrict the size of $\mathcal{Y}_i$ to 32 in our experiments.

Then, we extract the following features for our training: (1) $f_\theta(x_i, t_i)$: the MLLM feature of the input pair x_i and t_i, (2) $f_\theta(t_i)$: the MLLM feature of the input t_i, (3) $\log p(\text{“Yes”} \mid x_i, t_i, y_i+)$ for Eq. (5) – this value is computed for all candidates $y_i \in \mathcal{Y}_i$; we also extract the log probability of “No”, and (4) $\log p(y_i \mid t_i)$ for all candidates $y_i \in \mathcal{Y}_i$. These values are used to estimate the targets $\widehat{Z}_1(x, t)$, $\widehat{Z}_2(x, t)$ and $\widehat{u}(x, t)$, and for input features of the uncertainty estimation modules F_{Z_1}, F_{Z_2} and F_{u_t}. In practice, we pre-extracted features and re-use the pre-computed values for training to reduce the computation cost due to the MLLM inference.

C.5 Training and architecture details

We employ three neural uncertainty estimation modules $F_{Z_1}(x, t)$, $F_\Delta(x, t)$ and $F_{u_t}(t)$ that estimate $\log \widehat{Z}_1(x, t)$, $\Delta = \log \widehat{Z}_1(x, t) - \log \widehat{Z}_2(x, t)$ and $\widehat{u}_t(t)$, respectively. Here, Δ is always nonnegative because $0 \leq p(m = 1 \mid x, t, y) \leq 1$. We use a lightweight Transformer architecture for the uncertainty modules. We use a Transformer for estimating $F_{Z_1}(x, t)$ and $F_\Delta(x, t)$ by taking the MLLM features of (x, t) and t ($f_\theta(x, t)$ and f_θ(t), respectively) for their input. $F_{u_t}(t)$ is also a

```python
def greedy_clustering(dataset, sim_fn, thres):
    """dataset: (x, t, y) triplets
       sim_fn: text similarity function (higher is more similar) """
    # Collect answers by question/context.
    candidates = {}
    for x, t, y in dataset:
        candidates[t].append(y)

    # Process frequent questions first.
    candidates = sorted(candidates.items(), key=lambda item: -len(item[1]))

    clusters, cluster_assignments = [], {}
    for t, answers in candidates:
        # Greedy cluster assignment to the first similar cluster.
        assigned = False
        for cluster in clusters:
            if sim_fn() > thres:
                cluster_assignments[cluster].extend(answers)
                assigned = True
                break

        if not assigned:
            # Ensure that the order of clusters is sorted by the frequency.
            clusters.append(t)
            cluster_assignments[t] = answers
    return cluster_assignments
```

Figure C.3: **Greedy clustering Python pseudo-code.**

Transformer, which takes $f_\theta(t)$ as its input. These modules are all 3-layer multi-head Transformers [52] with a hidden dimension of 768 and 8 attention heads. The outputs are pooled by an attention pooling, and then fed into a linear head to estimate the target value; $F_{Z_1}(x,t)$ and $F_\Delta(x,t)$ share the Transformer backbone, but use different attention pooling modules and the linear projection. Since Z_1 and Z_2 are the first and second momentum of the probability $\in [0,1]$, $F_{Z_1} \leq 0$ and $F_\Delta \geq 0$; we use a softplus operation to ensure their value range. This is a similar technique to enforce positive variance values in probabilistic embeddings [18–20].

We use a mixture of the pooled features as the input of our module. More specifically, F_{Z_1} takes $f(x,t)$, the MLLM feature of multimodal input (x,t), and $f(t)$, the MLLM feature of text-only input (t). We use four pooled feature types for these features: (1) the last output token – which is also known as the "end-of-sentence (EOS)" token [51], (2) the mean of the last eight output tokens, (3) the mean of the last 32 output tokens, and (4) the mean of all output tokens. We pre-extract these values from the dataset and cache them to avoid expensive MLLM forward operations. These four features are fed into the Transformer; F_{Z_1} and F_Δ take 16 token features as their input (four features from each $f(x,t)$, $f(t)$, $f(x,t) - f(t)$, and $f(x,t) \times f(t)$), while F_{u_t} takes four tokens as input.

During training, we optimize the following objective function using a regression loss ℓ (*e.g.*, ℓ_1 loss) on collected data samples described in Sec. 4.2:

$$\ell(F_{Z_1}(x,t), \log \widehat{Z}_1(x,t)) + \ell(F_\Delta(x,t), \Delta) + \ell(\lambda F_{u_t}(t) + F_{Z_1}(x,t) + F_\Delta(x,t), \hat{u}(x,t)). \quad \text{(C.1)}$$

Since our training dataset has a different distribution from the actual target benchmarks, we found that the modules trained on our collected training dataset often suffer from the generalizability problem. To mitigate this issue, we additionally employ a lightweight adaptation to a small number of data samples related to the target benchmark. We only updated the last linear layer to estimate each scalar value (hidden dimension $\rightarrow$ 1) and the attention pooling parameters during this adaptation process. Let $\mathcal{I}^+$ denote the set of samples (x_i, t_i, y_i) that are correctly predicted by an MLLM and $\mathcal{I}^-$ denote the set of incorrectly predicted samples. The adaptation loss is defined as follows:

$$\mathbb{E}_{i\in\mathcal{I}^+, j\in\mathcal{I}^-}\left[\text{softplus}(u'_i - u'_j)\right], \quad \text{where } u'_i := \lambda F_{u_t}(t_i) + F_{Z_1}(x_i,t_i) + F_\Delta(x_i,t_i). \quad \text{(C.2)}$$

Eq. (C.2) calibrates the predicted uncertainty value using the actual prediction of an MLLM by enlarging the gap between the uncertainty of correctly predicted samples and that of incorrect samples. For our experiments, we use 900 multiple-choice question VQA samples from the MMMU dataset [30] and 10k open-ended VQA samples for the adaptation from the VQA v2 dataset [26].

Table D.1: **Overview of evaluation benchmarks.** The detailed statics of the datasets are shown. We report the number of samples for each dataset and its answer type. We also report the performance of Qwen3VL-2B, -4B, -8B-Instruct models and InternVL3.5-1B model on each dataset. For open-ended VQAs, we report the original VQA score and separately report the threshold-based accuracy used in our experiments in the brackets. †: The original VQA v2 has 214,354 images, but we randomly select 10k samples for our evaluation. ‡: These samples are used for the adaptation. *: These samples are used for model selection and hyperparameter tuning.

	Dataset stats		Performance			
	# Samples	Answer type	Qwen 2B	Qwen 4B	Qwen 8B	Intern 1B
VQA v2 (Validation) [26]	10,000†	Open-ended	62.2 (61.6)	82.8 (82.5)	71.6 (71.3)	76.5 (76.0)
VQA v2 (Adaptation)	10,000‡	Open-ended	-	-	-	-
Vizwiz (Validation) [27]	4319	Open-ended	38.9 (37.2)	46.4 (45.0)	40.8 (39.3)	41.2 (39.0)
OK-VQA (Validation) [28]	5046	Open-ended	46.1 (49.3)	61.2 (64.8)	55.4 (58.5)	54.9 (59.0)
HallusionBench [29]	1129	Yes / No	40.4	61.2	61.5	45.6
MMMU (Test) [30]	10500	MCQ	26	43.4	47.5	37.4
MMMU (Validation)	900‡*	MCQ	-	-	-	-
MMMU (Dev)	150*	MCQ	-	-	-	-
MMMU Pro [31]	1730	MCQ	20.6	32.2	35.9	28
MMStar [32]	1500	MCQ	41.5	48.3	49.3	39.1

D More Experimental Setup

D.1 Evaluation dataset details

We report the details of each dataset in Tab. D.1. Note that the uncertainty estimation methods compared in this paper keep the backbone MLLM weight frozen; therefore, their performance on each dataset are identical each other. In the main paper, we use 10,000 randomly sampled VQA v2 samples for reporting the VQA v2 results to reduce the expensive inference cost by generation-based methods († in the table). We also use randomly selected (but mutually exclusive) 10k samples from VQA v2 and 900 MMMU validation samples for our lightweight adaptation (‡ in the table). Finally, 150 MMMU dev samples and 900 MMMU validation samples are used for model selection and hyperparameter tuning along with the 1k validation split from our collected samples (Sec. 4.2).

In the table, the performances are measured by (1) a soft VQA score [26] for open-ended VQAs and (2) accuracy for the other tasks. Here, we additionally report the threshold-based accuracy for open-ended VQA tasks, which will be described in the following subsection.

D.2 Evaluation metric details

We use the following three metrics as our main metrics: AUROC, AUPRC and FPR95 (False Positive Rate when the True Positive Rate is 95%), which are standard metrics to measure the performance of uncertainty estimates [23].

These metrics are defined based on a binary classifier where the input is uncertainty score and the output is whether prediction is correct or incorrect. However, for open-ended VQA tasks, we use a soft score, named VQA score [26] as $\min\left(\frac{m(y)}{3}, 1\right)$, where $m(y)$ is the number of human reference answers that match the model prediction y after normalization. The main numbers in Tab. D.1 are the average VQA score over the samples.

To measure the uncertainty estimation performance of each method, we make the soft VQA score to a binary 0, 1 value using a threshold, *i.e.*, if the score is larger than threshold τ, the score becomes 1, otherwise 0. We report the accuracy measured by this in Tab. D.1 (numbers in brackets).

D.3 Implementation details

We optimize the uncertainty modules using AdamP [53] with a learning rate of 10^{-5}, a batch size of 64, weight decay of 0, and a dropout ratio [54] of 0.1. We optimize the module for 50 epochs and choose the best model based on the validation AUROC using the 1k held-out split. For hyperparameters, we search over λ in $\{0.25, 0.5, 1.0\}$ based on the MMMU validation split (MMMU

Prompts for comparison methods

[Loglikelihood]

```
Image: {x}
Question: {t}
Proposed Answer: {y}
```

[Verbalized Confidence]

```
Image: {x}
Question: {t}
Proposed Answer: {y}
How likely is the possible answer to be correct? Briefly explain, then write
only one score from 0 to {confidence_scale} inside <confidence></confidence>
tags.
```

[P(Correct)]

```
Image: {x}
Question: {t}
Proposed Answer: {y}
Is the proposed answer correct?
(A) True
(B) False
The proposed answer is:
```

Figure D.1: **Prompts for comparison methods.**

val AUROC − MMMU val FPR95). Based on this, $\lambda = 0.25$ was selected as the best hyperparameter except for Qwen3VL-4B, where its best λ is 1.0. During training, for each iteration, we randomly sample 16 samples from 32 samples of plausible answers. (described in Sec. 4.2).

For the lightweight adaptation stage, we search over batch size in $\{128, 256, 512\}$ and learning rate in $\{10^{-3}, 5 \times 10^{-4}, 10^{-4}, 5 \times 10^{-5}\}$ for each trained uncertainty module. Adaptation is also run for up to 50 epochs, and the best checkpoint is selected based on development-set AUROC.

D.4 Comparison methods details

Negative log-likelihood (NLL). NLL uses the likelihood of the generated answer under the backbone model as a confidence proxy. Lower likelihood corresponds to higher uncertainty. This is a natural decoder-based baseline but is often sensitive to output length and tokenization.

$$\ell(x, t, y) := -\sum_{i=1}^{|y|} \log p_\theta(y_i \mid x, t, y_{<i}). \tag{D.1}$$

In our experiments, however, we found that NLL actually does not serve as a proper uncertainty estimate; it usually shows a very low AUROC score, which means that higher likelihood corresponds to higher uncertainty. This is contradictory to our intuition, and it supports that naively using MLLM output as uncertainty estimate may not work very well in practice.

P(Correct). Following Kadavath et al. [1], we ask the model a follow-up verification question about its own answer and use the probability of the affirmative response as a confidence score (See Fig. D.1 for the prompt). This baseline is closely related to our verifier, but it is applied to the model's final generated answer rather than to a candidate-answer bank used to construct uncertainty moments.

Verbalized confidence. We also compare against verbalized confidence [3], where the model is prompted to explicitly state a confidence value in natural language. This baseline tests whether self-reported confidence alone is sufficient for reliable uncertainty estimation.

VL-Uncertainty. VL-Uncertainty [11] measures uncertainty from the stability of model outputs under multimodal perturbations. It is a strong multimodal baseline, but it is substantially more expensive than our single-pass estimator because it requires multiple perturbed forward passes and repeated generation. In our implementation, we use five perturbation rounds, pairing image blur

radius $\{0.6, 0.8, 1.0, 1.2, 1.4\}$ with question-rephrasing temperatures $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. For each perturbed input, we sample an answer with temperature 1.0, top-$p = 0.9$, and at most 32 new tokens, cluster the sampled answers into semantic groups, and use the entropy of the cluster-count distribution as the uncertainty score.

Consistency-based baselines. Following Wang et al. [10], we sample multiple answers for the same input and build an empirical answer distribution from the normalized outputs. We then compute either the entropy of this distribution or its maximum probability. These baselines are effective when short answers repeat reliably, but they become less stable when valid answers can be paraphrased in many ways. In our experiments, we sample five answers with temperature 1.0, top-$p = 0.9$, and at most 32 new tokens. Then we count the normalized output to make a multinormal distribution; entropy and maximum probability are used for uncertainty score.

Candidate-based baselines. For MCQ benchmarks, we additionally report candidate-based baselines computed over the explicit answer options. More specifically, we compute the log probability of each candidate (Eq. (D.1)) and use the value as the logit of the candidate; then we get a softmax probability using the logit values. These baselines are not candidate-free, but they provide a useful reference point in settings where the answer set is fully specified at test time.

We consider SteerConf [55] as well, which is a prompt-based baseline that queries the model multiple times under different confidence styles (*e.g.*, from very cautious to very confident). SteerConf extracts an answer-confidence pair from each response, and then aggregates these responses into a calibrated uncertainty estimate. When we applied SteerConf in our settings, it only produces a valid score when the model follows the required output format reliably enough across several steering levels. Concretely, many responses do not contain a parsable answer-confidence pair, therefore, it often fails to predict uncertainty score; we therefore did not compare SteerConf with CoMet.

We also did not compare CoMet with methods that require additional fine-tuning (or LoRA adaptation [56]) on the model, *e.g.*, Kapoor et al. [24], ConfTuner [57], Prompt4Trust [58], RLCR [25], or VL-Calibration [5]. We aim to keep the original weight of MLLM fixed and measure the uncertainty of the given MLLM without any model steering. These methods may improve calibration by changing the model itself, often with extra supervision, optimization, and hyperparameter tuning, and are therefore not directly comparable to a frozen-weight post-hoc uncertainty estimator.

Finally, it is worth noting that CoMet does not need to generate y, but only takes x and t as its input for uncertainty estimation (Appendix C.5) during inference time, resulting in more efficient uncertainty estimation than the comparison methods.

E More Experiment Results

E.1 Full results

We show the full results of open-ended multimodal understanding benchmarks (Tab. E.1), unanswerability/hallucination detection benchmarks (Tab. E.2), and multiple choice question (MCQ) visual question answering (VQA) benchmarks (Tab. E.3).

E.2 Decomposed pre-training targets vs. unified target

We compare CoMet with unified entropy targets without decomposition, but using the same candidates and verifier outputs. We construct a candidate distribution by applying a softmax to the MLLM-as-verifier results, $s_y = \log \frac{p(\text{Yes}|x,t,y)}{p(\text{No}|x,t,y)}$ over answer candidates for each question. Then, we train the same uncertainty module to regress either Shannon entropy $-\sum_y p_y \log p_y$ or Rényi-2 entropy $-\log \sum_y p_y^2$ as a single target, where $p_y = \mathrm{softmax}(s_y)$. Tab. E.4 shows that CoMet achieves 0.798 average AUROC, outperforming Shannon (0.729) and Rényi-2 (0.735). These results support the benefit of component-wise pre-training over directly regressing a single entropy target.

E.3 More discussion on MLLM-as-verifier and candidate construction

More discussion on MLLM-as-verifier. As we discussed in the main paper, the gap due to the choice of verifiers is not as significant as their disagreements. For example, InternVL3.5-1B and

Table E.1: **Open-ended VQA results.** We report AUROC (AUC), AUPRC (AP), and FPR@95 of uncertainty estimation methods on open-ended VQA benchmarks. "GF" denotes generation-free.

		Qwen3VL-2B-Instruct			Qwen3VL-4B-Instruct			Qwen3VL-8B-Instruct			InternVL3.5-1B		
Method	GF	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓
VQA v2 (10k) [26]													
Consist. (Ent)	✘	.624	.471	.925	.677	.266	.896	.684	.423	.891	.535	.249	.956
Consist. (Max)	✘	.620	.470	.914	.678	.266	.896	.680	.421	.891	.535	.249	.956
NLL	✘	.531	.389	.991	.425	.145	.995	.450	.253	.991	.547	.211	.986
Verbalized Conf	✘	.545	.419	.913	.608	.218	.877	.651	.460	.808	.623	.350	.846
P(Correct)	✘	.600	.454	.913	.724	.361	.795	.740	.554	.743	.610	.274	.982
VL Uncertainty	✘	.630	.520	.849	.684	.355	.776	.665	.448	.836	.623	.339	.878
CoMET (Ours)	✔	**.833**	**.765**	**.609**	**.844**	**.524**	**.644**	**.836**	**.664**	**.664**	**.773**	**.542**	**.709**
VizWiz (val) [27]													
Consist. (Ent)	✘	.594	.697	.908	.640	.656	.906	.634	.701	.904	.475	.590	.959
Consist. (Max)	✘	.596	.697	.908	.639	.656	.906	.634	.701	.904	.475	.590	.959
NLL	✘	.567	.669	.942	.497	.543	.975	.427	.547	.980	.566	.575	.975
Verbalized Conf	✘	.606	.723	.823	.531	.564	.895	.656	.773	.727	.655	.751	.786
P(Correct)	✘	.756	.827	.786	.718	.734	.839	.715	.796	.734	.734	.772	.839
VL Uncertainty	✘	.662	.767	.837	.641	.687	.842	.658	.761	.806	.679	.746	.854
CoMET (Ours)	✔	**.783**	**.849**	**.732**	**.793**	**.813**	**.696**	**.815**	**.865**	**.663**	**.752**	**.813**	**.759**
OK-VQA [28]													
Consist. (Ent)	✘	.655	.611	.921	.666	.471	.895	.653	.520	.918	.477	.393	.963
Consist. (Max)	✘	.654	.610	.921	.663	.470	.895	.652	.519	.918	.477	.393	.963
NLL	✘	.609	.577	.955	.553	.384	.953	.548	.452	.938	.615	.512	.887
Verbalized Conf	✘	.621	.635	.790	.548	.352	.916	.619	.561	.826	.641	.535	.888
P(Correct)	✘	.698	.663	.892	.693	.514	.876	.689	.606	.821	.658	.551	.870
VL Uncertainty	✘	.660	.651	.869	.667	.524	.817	.650	.568	.853	.608	.519	.876
CoMET (Ours)	✔	**.740**	**.738**	**.781**	**.715**	**.557**	**.813**	**.746**	**.656**	**.776**	**.677**	**.586**	**.844**

Table E.2: **Unanswerability and hallucination detection results.** Details are the same as Tab. E.1.

		Qwen3VL-2B-Instruct			Qwen3VL-4B-Instruct			Qwen3VL-8B-Instruct			InternVL3.5-1B		
Method	GF	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓
VizWiz (unanswerable) [27]													
Consist. (Ent)	✘	.608	.408	.852	.628	.437	.823	.656	.443	.852	.465	.307	.933
Consist. (Max)	✘	.607	.409	.864	.628	.436	.852	.654	.443	.852	.465	.307	.933
NLL	✘	.570	.383	.942	.526	.329	.944	.413	.272	.982	.560	.341	.935
Verbalized Conf	✘	.615	.470	.961	.553	.360	.939	.621	.521	.946	.727	.575	.952
P(Correct)	✘	.751	.576	.712	.660	.421	.764	.600	.397	.839	.788	.594	**.628**
VL Uncertainty	✘	.651	.450	.910	.545	.367	.957	.543	.359	.950	.693	.481	.860
CoMET (Ours)	✔	**.788**	**.596**	**.641**	**.767**	**.556**	**.641**	**.797**	**.635**	**.689**	**.808**	**.641**	**.628**
HallusionBench [29]													
Consist. (Ent)	✘	.564	.631	.920	.569	.439	.932	.572	.442	.920	.522	.575	.919
Consist. (Max)	✘	.562	.630	.920	.567	.431	.957	.566	.438	.920	.521	.574	.919
NLL	✘	.443	.548	.981	.567	.425	.943	.585	.434	.943	.468	.524	.966
Verbalized Conf	✘	.583	.664	.945	.543	.435	.921	.604	.486	.876	.503	.545	.956
P(Correct)	✘	.594	.661	.921	.636	**.527**	**.836**	**.670**	**.536**	.848	.510	.512	.995
VL Uncertainty	✘	.417	.717	1.000	.500	.393	1.000	.500	.393	1.000	.429	.571	.909
CoMET (Ours)	✔	**.715**	**.791**	**.779**	**.644**	.523	.881	.605	.510	**.841**	**.656**	**.658**	**.906**

Qwen3VL-8B disagree with Qwen3VL-2B on 35.0% and 19.3% of collected candidates, but the average uncertainty estimation AUROC does not drop significantly. One possible explanation is that our uncertainty targets aggregate matching probabilities over $p(y \mid t)$, rather than directly using individual verification results. When we aggregate verification results for each question, the disagreement rates with Qwen3VL-2B become 28.0% for InternVL3.5-1B and 22.6% for Qwen3VL-8B, narrowing the gap between the two verifiers. These results support the limited sensitivity to the choice of the verifier model.

Table E.3: **Multiple Choice Question (MCQ) VQA results.** Details are the same as Tab. E.1. Note that Candidate (Ent) and (Max) use the candidate set information, while the other methods do not.

		Qwen3VL-2B-Instruct			Qwen3VL-4B-Instruct			Qwen3VL-8B-Instruct			InternVL3.5-1B		
		AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓	AUC ↑	AP ↑	FPR ↓
MMMU Test[30]													
Candidate (Ent)	✔	.732	.859	.812	.700	.692	.880	.692	.661	.856	.559	.648	.920
Candidate (Max)	✔	.717	.850	.832	.694	.680	.896	.685	.647	.872	.552	.642	.933
Consist. (Ent)	✘	.804	.926	.511	.707	.794	**.669**	.695	.740	**.741**	**.638**	**.751**	**.845**
Consist. (Max)	✘	.796	.922	.557	.705	.790	.690	.691	.732	.744	.634	.746	.855
NLL	✘	.155	.575	.997	.271	.431	.995	.337	.432	.977	.386	.539	.990
Verbalized Conf	✘	.666	.832	.934	.521	.621	.950	.552	.586	.923	.521	.654	.940
P(Correct)	✘	.430	.680	.981	.743	.775	.769	.702	.696	.830	.580	.691	.903
VL Uncertainty	✘	.837	.941	.404	.696	.770	.727	.697	.730	.762	.579	.692	.915
CoMet (Ours)	✔	**.875**	**.956**	**.350**	**.777**	**.819**	.688	**.770**	**.783**	**.725**	.630	.730	.880
MMMU Pro [31]													
Candidate (Ent)	✔	.725	.903	.775	.672	.803	.853	.680	.784	.807	.598	.770	.934
Candidate (Max)	✔	.715	.898	.800	.659	.786	.887	.672	.770	.850	.583	.763	.929
Consist. (Ent)	✘	.769	.928	.612	.696	.847	**.670**	.708	**.831**	**.695**	**.662**	**.814**	**.884**
Consist. (Max)	✘	.757	.921	.635	.694	.845	.700	.705	.826	.700	.648	.801	.916
NLL	✘	.192	.659	.996	.293	.554	.995	.359	.569	.968	.439	.673	.993
Verbalized Conf	✘	.626	.834	.946	.493	.713	.961	.564	.677	.925	.521	.741	.928
P(Correct)	✘	.522	.781	.970	.711	.828	.758	.704	.802	.760	.555	.747	.935
VL Uncertainty	✘	.815	.949	.448	.703	.839	.730	.714	.825	.709	.652	.810	.873
CoMet (Ours)	✔	**.834**	**.953**	**.442**	**.739**	**.851**	.778	**.726**	.811	.794	.610	.771	.931
MMStar [32]													
Candidate (Ent)	✔	.701	.643	.810	.669	.533	.849	.654	.492	.874	.590	.554	.909
Candidate (Max)	✔	.692	.632	.803	.667	.524	.858	.651	.489	.879	.592	.548	.921
Consist. (Ent)	✘	.753	.844	.564	.723	.797	.574	.740	.800	.539	**.741**	**.848**	**.574**
Consist. (Max)	✘	.746	.837	.567	.719	.786	.644	.732	.785	.622	.739	.847	**.574**
NLL	✘	.226	.434	.995	.177	.352	.999	.206	.356	.996	.247	.462	.999
Verbalized Conf	✘	.559	.651	.919	.607	.694	.821	.564	.658	.850	.551	.691	.808
P(Correct)	✘	.581	.604	.958	.732	.725	.763	.793	.815	.581	.634	.710	.896
VL Uncertainty	✘	.754	.838	.593	.765	.827	.524	.754	.811	.590	.529	.644	.917
CoMet (Ours)	✔	**.819**	**.878**	**.523**	**.830**	**.859**	**.501**	**.853**	**.881**	**.444**	.739	.815	.743

Table E.4: **Unified entropy target vs. decomposed target for pre-training.** Using the same answer candidates and MLLM-as-verifier results, we compare the decomposed pre-training target by CoMet with the unified target by Shannon's entropy and Rényi-2 entropy. We use the Qwen3VL-2B backbone for all experiments and use the same dataset, architecture, and adaptation. The other details are the same as Tab. 3.

Object	VQA v2	VW	OK-VQA	VW (U)	HB	MMMU	MMMU Pro	MMStar	Avg
Shannon Ent.	.747	.685	.651	.721	.626	.806	.779	.820	.729
Rényi-2 Ent.	.748	.747	.677	.784	.538	.821	.774	.789	.735
CoMet	**.833**	**.783**	**.740**	**.788**	**.715**	**.875**	**.834**	**.819**	**.798**

More discussion on LLM-generated candidate construction. As we mentioned in the main paper, LLM-based answer candidate generation is not straightforward for vision-related questions. When an LLM is directly given a question such as "What is in this image?", it often refuses to answer, *e.g.*, "I currently cannot view or analyze images," rather than producing plausible visual answers. Therefore, we need to design a dedicated candidate-generation prompt. Fig. E.1 shows our dedicated prompt with in-context samples. We use the candidates constructed by our dataset-based method as the in-context examples. Using this setup, we generate 28 candidates for each question using Qwen3-8B.

The impact of the candidate size. Tab. E.5 and Tab. E.6 show the impact of the size of answer candidates. Tab. E.5 shows that increasing the number of questions from 3k to 200k improves average AUROC from 0.786 to 0.798. When the number of candidates per question changed from 4 to 32, the average performance tends to improve (0.785 to 0.798) as shown in Tab. E.6.

Prompts for candidate construction study

```
You are constructing plausible candidate answers for a visual question.
Generate exactly one short answer phrase.  Do not explain.  Do not output
multiple answers.  Question:  {t}, Known valid answers (examples, not
exhaustive):  {in-context examples}.  Generate one additional plausible
answer, preferably different from the examples above.  Answer:
```

Figure E.1: **Prompts for candidate construction study.**

Table E.5: **Impact of the number of candidates.** CoMet results by varying the number of pre-training answer candidates are shown. Each question has 32 candidates. Qwen3VL-2B-Instruct is used for the backbone.

# questions	VQA v2	VW	OK-VQA	VW (U)	HB	MMMU	MMMU Pro	MMStar	Avg
3k	.835	.741	.727	.751	.707	.879	.831	.813	.786
5k	.828	.800	.738	.795	.680	.874	.811	.804	.791
10k	.828	.774	.731	.797	.715	.863	.794	.782	.785
20k	.831	.775	.734	.793	.700	.873	.806	.808	.790
25k	.831	.774	.738	.788	.702	.874	.811	.809	.791
100k	.823	.741	.721	.711	.697	.872	.816	.819	.775
200k	.833	.783	.740	.788	.715	.875	.834	.819	.798

Table E.6: **Impact of the number of candidates per question.** K denotes the number of candidate answers for each question. The details are the same as Tab. E.5.

K	VQA v2	VW	OK-VQA	VW (U)	HB	MMMU	MMMU Pro	MMStar	Avg
4	.832	.779	.722	.776	.681	.877	.818	.795	.785
8	.812	.772	.709	.791	.615	.866	.822	.809	.775
16	.835	.778	.734	.777	.681	.877	.833	.816	.791
32	.833	.783	.740	.788	.715	.875	.834	.819	.798

E.4 Impact of adaptation

Tab. E.7 shows the impact of our lightweight adaptation method. When we use the Qwen3VL-8B-Instruct backbone, even without adaptation, CoMet already provides a meaningful uncertainty signal compared to strong baselines, such as Verbalized confidence and P(Correct) (average AUROC 0.609 vs. 0.604 and 0.598, respectively). By adapting CoMet on the MMMU validation split, CoMet ties with VL uncertainty, the strongest baseline in average AUROC. By adding VQA samples for the adaptation, the uncertainty estimation by CoMet is significantly improved.

Importantly, the effect depends on the adaptation source. MMMU-only adaptation specializes the uncertainty head to MCQ evaluation (MMMU, MMMU-Pro, and MMStar), while open-ended VQA performances are somewhat worse than the original (*e.g.*, OK-VQA AUROC 0.621 $\rightarrow$ 0.593 after the MMMU-only adaptation). In contrast, VQA-only adaptation significantly improves open-ended VQA tasks, but its MCQ performance is somewhat worse than that of the MMMU-only adaptation. Combining VQA and MMMU adaptation gives the best average performance, suggesting that the two sources provide complementary calibration signals.

In the main paper, we show that using the same samples for pre-training and adaptation also works. More specifically, we evaluate the model's answers on Cambrian using the same VQA scoring protocol and convert the scores to binary correctness labels for adaptation.

E.5 Parameter study

Tab. E.8 and Tab. E.9 show the parameter study results on λ and (β, γ) using the Qwen3VL-8B-Instruct model. We select the best hyperparameter based on the MMMU validation split. In the tables, we observe that our hyperparameter choice performs the best on the MMMU validation split as well as the average score.

Table E.7: **Impact of the adaptation strategy.** Measured by the Qwen3VL-8B-Instruct backbone.

Method	VQA v2	VizWiz	OK-VQA	MMMU Test	MMMU Pro	MMStar	Avg
CoMet (No Adapt)	.503	.676	.621	.671	.659	.526	.609
CoMet (MMMU only adapt)	.654	.730	.593	.772	.731	.863	.724
CoMet (VQA only adapt)	.826	.816	.726	.702	.678	.804	.759
CoMet (VQA + MMMU adapt)	.836	.815	.746	.770	.726	.853	.791
Consist (ent)	.624	.594	.655	.732	.725	.701	.672
Verbalized Confidence	.545	.606	.621	.666	.626	.559	.604
P(Correct)	.600	.756	.698	.430	.522	.581	.598
VL uncertainty	.630	.662	.660	.837	.815	.754	.726

Table E.8: **Impact of λ.** We use Qwen3VL-8B-Instruct as the backbone. Hyperparameters are selected on the MMMU validation split (the first column).

λ	MMMU val	VQA v2	VizWiz (val)	OK-VQA	MMMU Test	MMMU Pro	MMStar	Average
0	.837	.834	.734	.735	.753	.707	**.857**	.780
0.25	**.842**	.836	.815	.746	.770	**.726**	.853	**.798**
0.5	.822	**.841**	.796	.741	**.772**	.715	.835	.789
1	.828	.839	**.821**	**.751**	.767	.707	.852	.795

Table E.9: **Impact of β, γ.** Details are the same as Tab. E.8.

β/γ	MMMU val	VQA v2	VizWiz (val)	OK-VQA	MMMU Test	MMMU Pro	MMStar	Average
0.2/-0.1	**.842**	.836	**.815**	**.746**	**.770**	.726	**.853**	**.798**
1/0	.835	**.837**	.796	**.746**	.760	.736	.832	.792

E.6 Controlled analysis of the uncertainty components

Visual intervention. We first modify only the visual input x while fixing context t. Specifically, for each image-question pair, we keep the question unchanged and progressively degrade the image using Gaussian blur with radii 0, 2, 4, 8, and 16. Then, we measure how u_t and $u_{x|t}$ change. Tab. E.10a shows that when we perturb the input image x, only $u_{x|t}$ increases monotonically, while u_t remains fixed. This is because we might have lower confidence in a low-quality image, which broadens the possible answer set of the given x, t pair.

Context intervention. We modify the breadth of the answer space induced by t. Note that $u_{x|t}$ is also dependent on t; therefore, solely testing u_t without changing $u_{x|t}$ is not possible. Instead, we construct a toy example where the direction of the uncertainty is predictable.

Let $s_i = p(m = 1 \mid x, t, y_i)$ and $p(y \mid t) = \frac{1}{K}$. Then we have:

$$u_{x|t} = -\log(\mathrm{CV}[S]^2 + 1) = -\log \frac{\mathbb{E}[S^2]}{\mathbb{E}[S]^2} = -\log \left[K \frac{\sum_{i=1}^{K} s_i^2}{(\sum_{i=1}^{K} s_i)^2} \right]. \tag{E.1}$$

Now, let's assume that exactly r of the K context-plausible candidates are plausible with the given input (*i.e.*, $s_i = 1$ or 0 and only r scores are 1). For example, assume we have a question "what color is this car among blue, dark blue, and red?" and a dark blue-ish car image. In this case, $K = 3$ (blue, dark blue, red) and $r = 2$ (blue, dark blue). Under this simplified assumption, $\sum_{i=1}^{K} s_i^2 = r$ and $(\sum_{i=1}^{K} s_i)^2 = r^2$, thus, $u_{x|t} = \log r - \log K$. By assuming $r = 1$, *i.e.*, only one plausible answer. We have $u_{x|t} = -\log K$. Namely, in this setting, $u_{x|t}$ becomes smaller (or more certain) by enlarging K. Note that this is not a realistic setting, but a controlled experiment to verify the behavior of each module. We will explore a more realistic setting later.

To approximate this controlled case, we select "how many" questions from VQA v2 whose majority answer is supported by at least nine human annotations. We change the question to "`{original question}. Answer with a single integer from 0 to {K-1}`", and vary K.

Tab. E.10b shows the results. As we expected, u_t is monotonically increasing because the plausible answer space becomes larger, and $u_{x|t}$ is monotonically decreasing as we derived previously. The results are computed using 525 samples.

Table E.10: **Intervention on x and t.** We intervene on VQA v2 samples to control u_t and $u_{x|t}$. (a) used the full 10k evaluation split of VQA v2. For (b) and (c), we use 525 "how many questions" and 1,809 "how many" and "what color" questions. Qwen3VL-2B is used for the backbone.

(a) **Intervetion on x.** We intervene on the visual input x by adding stronger Gaussian blur.

Radius	0 (clean)	2.0	4.0	8.0	16.0
u_t	0.79	0.79	0.79	0.79	0.79
$u_{x\|t}$	-2.39	-1.69	-1.42	-1.27	-0.87
u	-2.19	-1.50	-1.22	-1.07	-0.68

(b) **Intervention on t with an extreme case.**

K	u_t	$u_{x\|t}$	u
20	1.99	-1.81	-1.31
30	2.04	-1.90	-1.39
40	2.09	-2.25	-1.73

(c) **More realistic t control.**

K	u_t	$u_{x\|t}$	u
2	-0.55	-2.99	-3.13
4	-0.39	-2.40	-2.50
8	0.02	-1.83	-1.82
-	1.79	-2.47	-2.02

Table E.11: **Additional parameters and memory by CoMet.** Bfloat16 is used, excluding activations.

Backbone	Backbone param/memory (B/GiB)	CoMet param/memory (M/MiB)	Increase (%)
Qwen3VL-2B	2.13 / 3.96	50.4 / 96.2	+2.4%
Qwen3VL-4B	4.44 / 8.27	52.4 / 99.9	+1.2%
Qwen3VL-8B	8.77 / 16.33	58.3 / 111.2	+0.7%
InternVL3.5-1B	1.06 / 1.98	46.5 / 88.7	+4.4%

Again, this is an unrealistic setting and does not represent natural open-ended settings. As the candidate set expands, it is likely to include additional plausible answers compatible with the input. For example, let's assume a "what color" question. Increasing the size of the answer space might lead to confusing answers, such as blue, dark blue, navy blue, and so on. In other words, in practice, r is a function of K, i.e., $r(K)$, and increases with K, and $\log \frac{r(K)}{K}$ can be increased.

To examine a less restrictive and more natural controlled setting, we select "what color" and "how many" questions from VQA v2 whose normalized majority answer is supported by at least three human annotations. We select 1,809 samples from the 10k VQA v2 evaluation set. For each question, we construct nested answer sets of sizes $K = 2, 4, 8$. Color distractors are sampled from a fixed color vocabulary, while count distractors are sampled from neighboring non-negative integers. In this more natural construction, expanding K may also alter the distribution of matching probabilities.

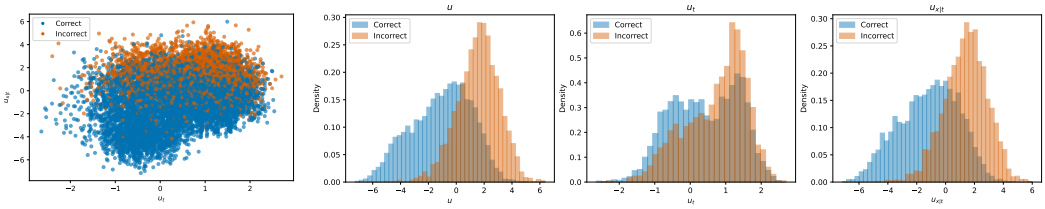
For example, given the question "How many men are wearing hats?" with the ground-truth answer "1" and $K = 8$, we construct: "`How many men are wearing hats? Choices: A. 3, B. 4, C. 1, D. 6, E. 7, F. 2, G. 0, H. 5. Answer with the option letter only.`" When $K = 2, 4$, we simply use a subset of the above example with fewer choices and shuffled order.

We also evaluate the original open-ended question as a reference ("$-$" in the table). Note that because the original open-ended questions do not have an explicitly specified K, this result is not directly comparable with the controlled K experiment, which is similar to MCQ settings.

Tab. E.10c shows the average u_t, $u_{x|t}$, and u in the selected samples. There are three interesting findings in this table. First, on average, altering K from 2 to 8 to have a broader answer space significantly increases the context-related uncertainty from -0.546 to 0.018. We also observe that 96.6% of samples show larger u_t when $K = 8$ compared to $K = 2$. Second, the original open-ended t has u_t of 1.786, which is much larger than restricted contexts. This is expected because open-ended questions might have a larger number of plausible answers compared to questions limiting the number of answers. Finally, $u_{x|t}$ increases as K increases. In natural open-ended settings, expanding $p(y \mid t)$ often introduces semantically related variants that remain compatible with x, rather than only incompatible candidates. Therefore, we generally expect $u_{x|t}$ to increase because the relative mass of compatible answers can increase with K. This differs from the simplified $r = 1$ case, where every additional candidate is assumed to be incompatible.

E.7 Runtime analysis

We first show the additional parameter size and memory consumption by CoMet in Tab. E.11. In terms of the parameter size, CoMet adds 46.5M to 58.3M parameters depending on the backbone, corresponding to a relative overhead of 0.7% to 4.4% depending on the backbone. This shows that our uncertainty module is lightweight.



(a) u_t **vs.** $u_{x|t}$. (b) **Histogram of each uncertainty**

Figure E.2: **Statistics of uncertainty values.** We plot the statistics related to u, u_t, and $u_{x|t}$ on the VQA v2 dataset using the Qwen3VL-2B-Instruct model. Blue samples denote correct samples, while orange samples denote incorrect samples.

We measure the actual runtime by various uncertainty estimation methods. Note that the multimodal understanding benchmarks used in our experiments are usually sufficient with very short generated tokens (*e.g.*, "A" for MCQ VQA tasks or "A cat" for open-ended VQA tasks). Therefore, we set the maximum generated tokens to 4,096 to examine the scenario when we need to generate a longer-context output; this could include the reasoning trace [59], or long context generation.

Tab. E.12 shows the results. We randomly select 100 samples from the VQA v2 dataset and measure the runtime for uncertainty estimation methods. In this experiment, CoMET shows the most efficient inference time uncertainty estimation. Specifically, compared to sampling-based methods, such as consistency (we generate five diverse samples) and VL Uncertainty (we randomly generate five noises), CoMET performs much faster inference, but achieves significantly better uncertainty estimation performance as shown in the main results.

Table E.12: **Runtime Analysis.** We report the average throughput of uncertainty estimation methods on the sampled VQA v2 dataset using the Qwen3VL-2B-Instruct model with CoMET. The number of maximum generation tokens is set to 4096 in this experiment. Runtime is measured on a single RTX 3090 GPU.

Methods	Throughput (s)
Consistency	34.87
NLL	5.95
Verbalized Conf	6.10
P(Correct)	6.01
VL Uncertainty	16.33
CoMET (Ours)	**5.88**

Training takes about 3 L40 GPU hours after cache extraction. Cache extraction took 5.5 hours on 40 L40 GPUs (220 GPU-hours) for 200k samples. For evaluation, consistency or VL uncertainty takes more than a day with 1 L40 GPU when we have about 10k samples (*e.g.*, VQA v2 or MMMU test); meanwhile, because CoMET can re-use the cached features, once features are cached, it takes less than a minute to evaluate; cache extraction takes about 3-4 hours in the same setting.

E.8 Visualization

Fig. E.2 shows the statistics of uncertainties on VQA v2 datasets using the Qwen3VL-2B-Instruct model. Blue and orange samples correspond to correct and incorrect samples, respectively.

For the qualitative visualizations, we use CoMET with the Qwen3VL-8B-Instruct model. We visualize the samples having high and low uncertainty in each uncertainty term: Fig. E.3 shows the samples with our proposed uncertainty, Fig. E.4 shows the samples with context-specific uncertainty u_t, and Fig. E.5 shows the samples with multiplicity-specific uncertainty $u_{x|t}$.

In addition, we illustrate the samples having the same x but different t in Fig. E.6. As shown in the figure, the level of uncertainty can be changed a lot by choosing different context. For example, "Is this food ready to eat" can be answered easily, but "Is this pizza homemade?" is somewhat hard to answer; we need more information to answer the question.

We also visualize the samples having the same context (question) t but different input (image x) in Fig. E.7. Even with the same t (therefore, having the same u_t), the degree of uncertainty could vary a lot as shown in the figure.

Finally, we report the samples with opposing u_t and $u_{x|t}$, *e.g.*, u_t is high but $u_{x|t}$ is low, and vice versa. Fig. E.8 shows the samples. For example, even though a question can be answered in just two answers ("yes" or "no"; so u_t is very low), it is somewhat difficult to answer the question "Are all people about the same age?" in the first example, resulting in a high uncertainty value due to $u_{x|t}$.

F Discussion and Limitation

Limitation in theory. In theory, we use the uniform assumption on $p(y \mid t)$. Although we believe that it is reasonable and not strong assumptions, this does not generalize to case where $p(y \mid t)$ is not a uniform distribution. However, our assumption can be viewed as a relaxed version of those cases; furthermore, the empirical evidence supports that our formulation is suitable in many scenarios, including multimodal understanding, error detection, and multimodal MCQ tasks.

Limitation in method design. Although our formulation is model-agnostic and can be generalized to any form of multimodal tasks, our actual method design has some limitations.

The first limitation of CoMet in terms of method design is the approximation of $p(y \mid t)$, or specifically K_t. However, we believe that the approximation is one of the most reasonable and sound ways considering that the true $p(y \mid t)$ is impossible to access. We also use a control parameter λ to control the approximation error induced by this approximation; our experiments show that $\lambda = 1$ (*i.e.*, using the approximation without any control) actually performs well on average (Tab. E.8); we chose $\lambda = 0.25$ because it performs the best in the hyperparameter selection criteria (*i.e.*, MMMU validation), but in practice, we believe that $\lambda = 1$ is also acceptable, considering the empirical evidence in our parameter study.

Another limitation is the quality of the matching probability module, *i.e.*, MLLM-as-verifier. As shown in Fig. C.2, the MLLM-as-verifier is not a perfect estimator; the quality of matching probability estimator can degrade the performance of CoMet as shown in Appendix C.1. This makes the framework simple and broadly applicable, but the resulting uncertainty targets can inherit calibration errors or semantic biases from the verifier. We partially address this issue through validation-based calibration, but a more principled calibration of semantic matching probabilities remains an interesting direction for future work.

Potential extensions. Although this paper focuses on open-ended multimodal VQA, the proposed formulation is not limited to this setting. More broadly, X can denote an observed input, T can denote the task context or environment specification that constrains the valid outputs, and Y can denote a candidate response or action. For example, in agentic AI settings, uncertainty may arise from ambiguity in the user request, the environment state, or the set of valid actions [60]. In this case, X can denote the observed state or user input, T can denote the task context or environment specification, and Y can denote a candidate response or action. Similarly, for reasoning models, the context t can include an intermediate reasoning trace that define the space of acceptable outputs. We leave a systematic study of these broader settings to future work.

G Broader impacts

This work aims to provide a better multimodal predictive uncertainty of MLLMs, which potentially supports safer deployment in high-stakes scenarios. We do not anticipate direct negative impacts specific to CoMet beyond those of uncertainty-aware MLLMs in general.

We use VizWiz [27] to evaluate both open-ended VQA and unanswerability detection. Its images and spoken questions were contributed by blind users seeking help with everyday visual tasks; the reference answers were collected from crowd workers. A question may be unanswerable from the submitted image even though the user still needs an answer. If an assistive system gives an incorrect answer with low uncertainty, the user may act on information they cannot easily verify. If it abstains when the image does contain enough information, the user may have to retake the image or ask someone else for help. Evaluating such a system with blind and low-vision users would help determine when abstention is useful and what assistance to offer next.

These errors also have different consequences in other high-stakes settings. For example, a falsely confident answer about a medication label could lead someone to take the wrong dose, while unnecessary abstention could delay access to instructions that were legible. Our benchmarks measure how well uncertainty scores distinguish correct from incorrect predictions; they do not establish when it is safe to act on an answer or when a system should abstain. Before using CoMet for such decisions, both types of error should be evaluated under the intended conditions of use, with an alternative way to obtain or verify an answer when needed.

Final uncertainty u

Most certain

Prediction: Correct
u: −6.4965
u_t: 0.1047
$u_{x|t}$: −6.5227
Context t: Is this indoors or outdoors?
Input x:



Prediction: Correct
u: −5.8762
u_t: 0.1136
$u_{x|t}$: −5.9046
Context t: Is this taken during day or night?
Input x:



Prediction: Correct
u: −5.0620
u_t: −0.1937
$u_{x|t}$: −5.0136
Context t: Is this water or snow?
Input x:



Prediction: Correct
u: −4.8878
u_t: 1.4556
$u_{x|t}$: −5.2517
Context t: How many people?
Input x:



Prediction: Correct
u: −4.4962
u_t: 2.1177
$u_{x|t}$: −5.0256
Context t: What sport is this?
Input x:



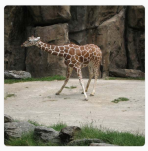
Prediction: Correct
u: −4.4546
u_t: 0.3690
$u_{x|t}$: −4.5469
Context t: How many zebras are in the photo?
Input x:



Prediction: Correct
u: −4.4534
u_t: 1.5107
$u_{x|t}$: −4.8311
Context t: How many men are pictured?
Input x:



Prediction: Correct
u: −4.4210
u_t: −1.0236
$u_{x|t}$: −4.1651
Context t: Is this a giraffe?
Input x:



Prediction: Correct
u: −4.3760
u_t: 0.5957
$u_{x|t}$: −4.5249
Context t: Is this real or a model?
Input x:



Prediction: Correct
u: −4.3511
u_t: −0.1960
$u_{x|t}$: −4.3021
Context t: Is this daytime or night time?
Input x:



Prediction: Correct
u: −4.3007
u_t: 1.7420
$u_{x|t}$: −4.7362
Context t: What sport is being played?
Input x:



Prediction: Correct
u: −4.2206
u_t: −0.9810
$u_{x|t}$: −3.9754
Context t: Is there an animal in this photo?
Input x:



Most uncertain

Prediction: Incorrect
u: 7.4382
u_t: 1.5318
$u_{x|t}$: 7.0553
Context t: How many calories?
Input x:



Prediction: Incorrect
u: 7.1284
u_t: 0.8759
$u_{x|t}$: 6.9094
Context t: How many train windows?
Input x:



Prediction: Incorrect
u: 7.0486
u_t: −0.6632
$u_{x|t}$: 7.2144
Context t: Do these people exercise a lot?
Input x:



Prediction: Correct
u: 6.9888
u_t: 1.4068
$u_{x|t}$: 6.6371
Context t: How many windows are there?
Input x:



Prediction: Correct
u: 6.7517
u_t: −0.6885
$u_{x|t}$: 6.9238
Context t: Are these children now grown up?
Input x:



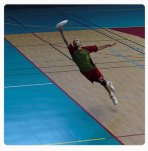
Prediction: Correct
u: 6.7162
u_t: 1.6327
$u_{x|t}$: 6.3081
Context t: How many clay jars are there?
Input x:



Prediction: Incorrect
u: 6.7016
u_t: 1.2308
$u_{x|t}$: 6.3939
Context t: How many inch of snow is there?
Input x:



Prediction: Incorrect
u: 6.6497
u_t: −0.9218
$u_{x|t}$: 6.8802
Context t: Will he catch the frisbee?
Input x:



Prediction: Incorrect
u: 6.5530
u_t: −0.4423
$u_{x|t}$: 6.6635
Context t: Are they married?
Input x:



Prediction: Incorrect
u: 6.5447
u_t: 0.5199
$u_{x|t}$: 6.4147
Context t: How much longer until midnight?
Input x:



Prediction: Incorrect
u: 6.5433
u_t: 1.4658
$u_{x|t}$: 6.1768
Context t: How many bunches of fruit are hanging?
Input x:



Prediction: Incorrect
u: 6.4931
u_t: 1.2433
$u_{x|t}$: 6.1823
Context t: How tall is the clock?
Input x:



Figure E.3: **Visualization of the certain/uncertain samples by the proposed uncertainty scores u.**

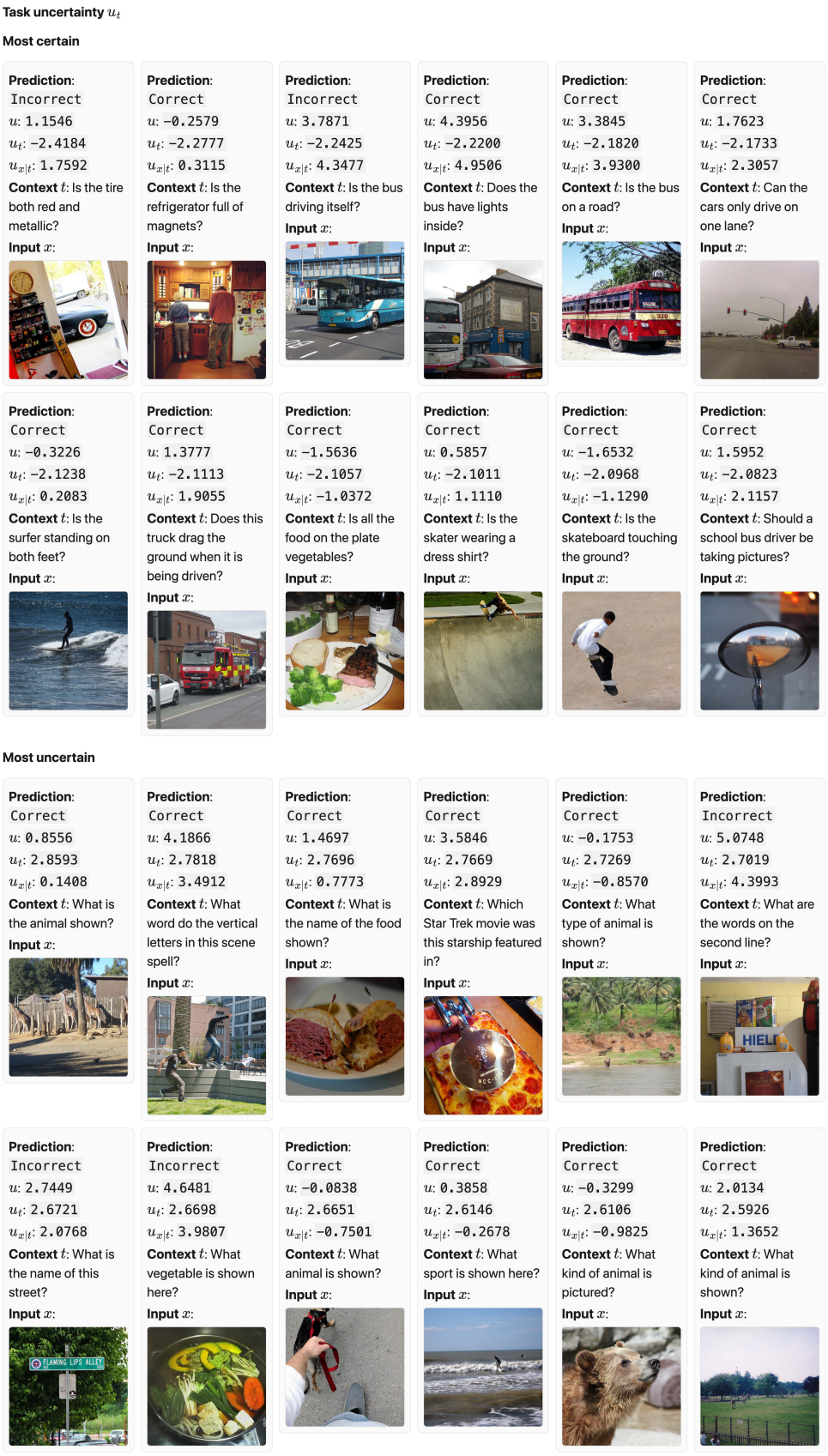


Figure E.4: **Visualization of the certain/uncertain samples by the context-specific uncertainty scores u_t.**

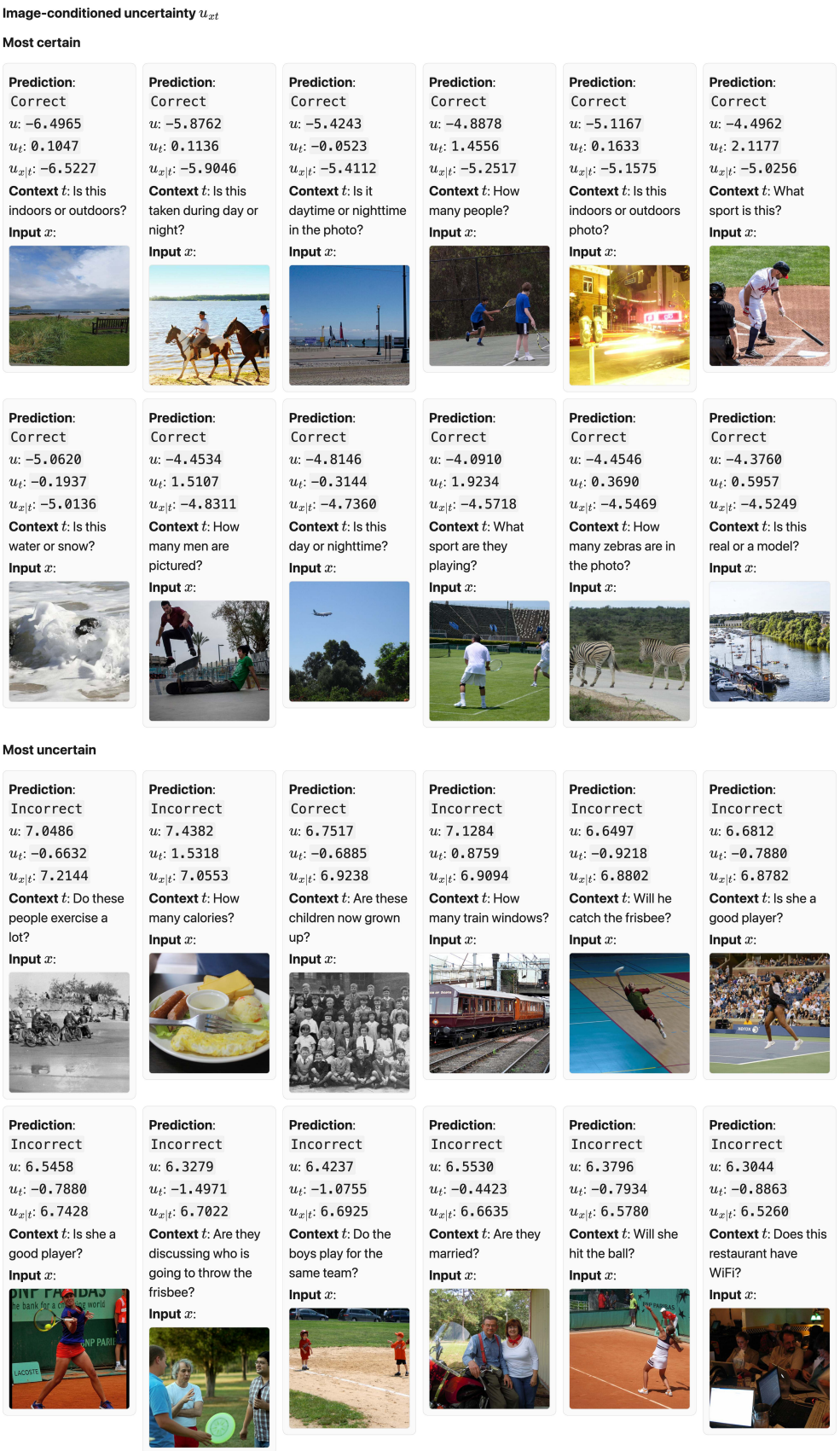


Figure E.5: **Visualization of the certain/uncertain samples by the multiplicity-specific uncertainty scores** $u_{x|t}$.

Same Image, Different Questions by u



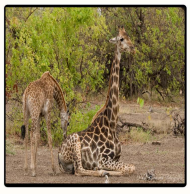
Prediction	u	u_t	$u_{x\|t}$	Context t
Correct	-5.4243	-0.0523	-5.4112	Is it daytime or nighttime in the photo?
Incorrect	4.7103	-1.5009	5.0856	Is it summer?



Prediction	u	u_t	$u_{x\|t}$	Context t
Correct	-3.8697	0.3690	-3.9619	How many zebras are in the photo?
Incorrect	4.9631	0.6115	4.8102	How many stripes on the zebra on the right?



Prediction	u	u_t	$u_{x\|t}$	Context t
Correct	-2.5025	-1.3541	-2.1640	Is the man skateboarding?
Incorrect	5.8100	1.2954	5.4861	How tall is the wave in this picture?



Prediction	u	u_t	$u_{x\|t}$	Context t
Correct	-2.7855	1.8233	-3.2413	How many animals?
Correct	-0.7946	-1.6813	-0.3743	Is the giraffe behind a fence?
Incorrect	0.5428	-0.9541	0.7813	Is this giraffe at the zoo?
Incorrect	5.0513	-0.5124	5.1794	How tall is the giraffe?



Prediction	u	u_t	$u_{x\|t}$	Context t
Correct	-2.8732	-1.0869	-2.6015	Is this a pickup truck?
Incorrect	4.5726	1.3015	4.2472	What company made the white car?



Prediction	u	u_t	$u_{x\|t}$	Context t
Correct	-2.4693	-1.3881	-2.1223	Is the food ready to eat?
Correct	1.3160	-0.5751	1.4597	Is this a special plate for this type of dessert?
Incorrect	4.8152	-1.3319	5.1481	Is this pizza homemade?



Prediction	u	u_t	$u_{x\|t}$	Context t
Correct	-1.4730	0.5779	-1.6175	How many busses can be seen?
Incorrect	4.8565	-0.9150	5.0853	Is it cold out?
Incorrect	5.7690	1.3067	5.4423	What number of seats are on the bus?

Figure E.6: **Uncertainty can vary by context t even with the same input x.**

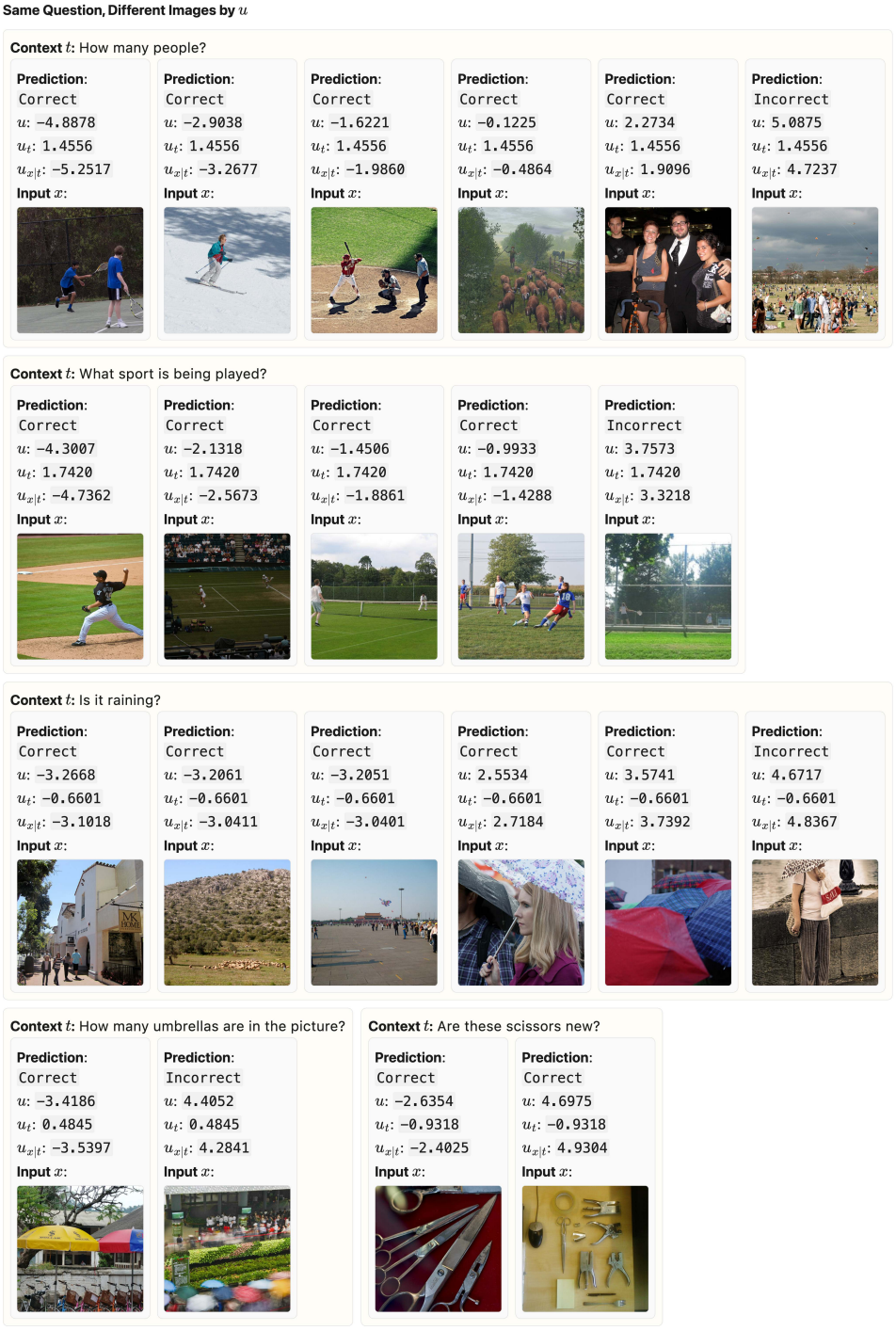


Figure E.7: **Uncertainty can vary by the relationship between $x \mid t$ and the set of plausible answers y (*i.e.*, their multiplicity), even with the same input t.**

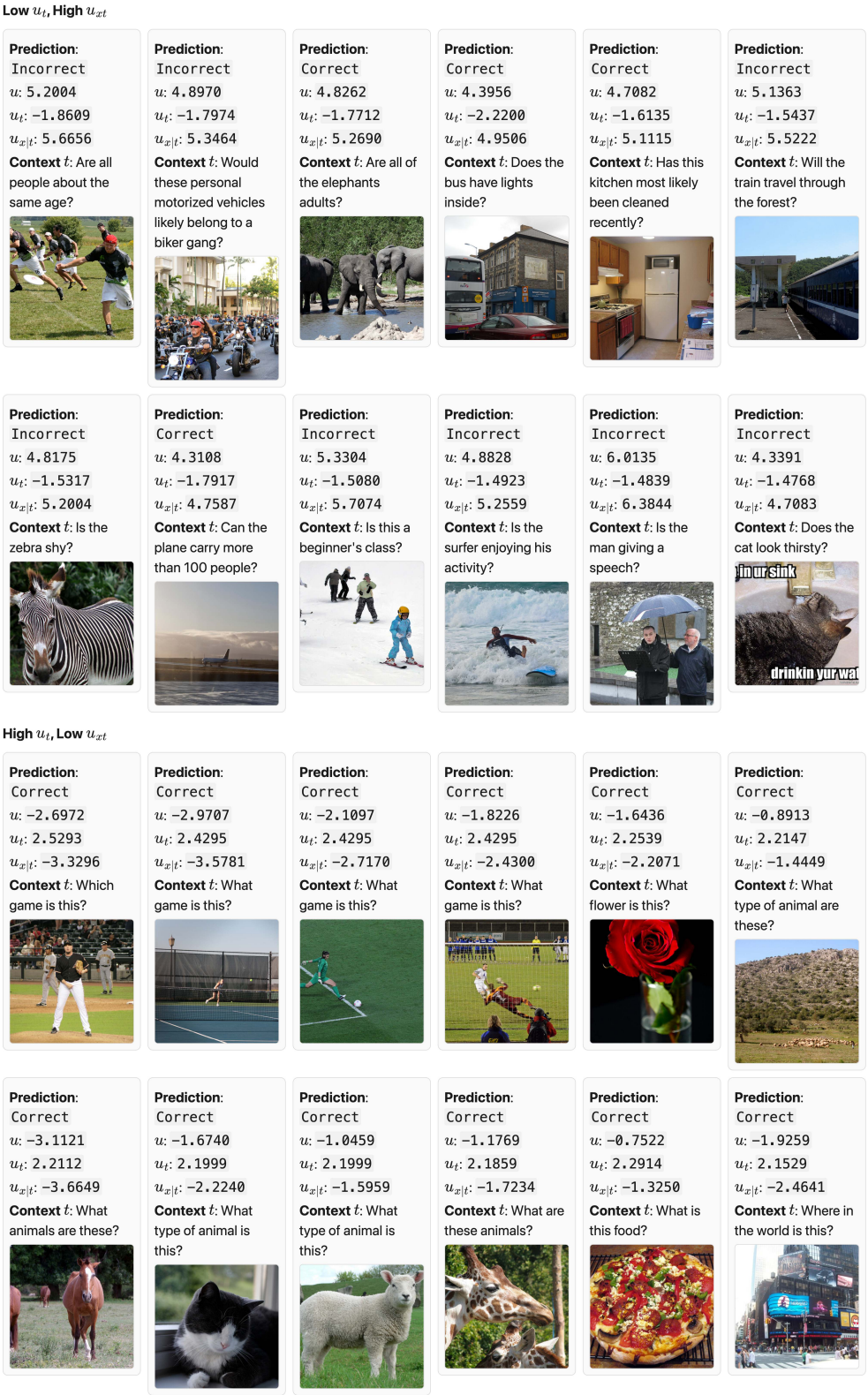


Figure E.8: **Samples with opposing uncertainty components.** Top: low u_t and high $u_{x|t}$. Bottom: high u_t and low $u_{x|t}$. These examples illustrate cases where task-only and image-conditioned uncertainty provide conflicting signals.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately explain our contribution and scope: uncertainty decomposition in MLLMs.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix F

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Appendix B.2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Appendix D.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release our code upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Sec. 5.1 and D

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: First, the comparison methods are training-free. Sampling-based baselines exhibit some randomness, but comparing them multiple times is time-consuming. CoMet uses a very lightweight post-hoc module, and we perform model selection based on the separated validation splits. We did not conduct multiple different experiments due to the computational limit.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix E.7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conduct our research with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix G.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all the assets used in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: Since we did not release our code, there is no new released asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.