Confidence-as-Context: Persistent Uncertainty Signals for Autoregressive Generation

Xueyan Li, Jonas Geiping
Max Planck Institute, ETH Zurich

Abstract

Modern autoregressive language models generate rich, step-wise signals of uncertainty over each token that correlate with calibration and correctness, yet these signals are transient and typically discarded before the next decoding step. We explore whether exposing this information back to the model can guide generation. We propose *entropy-guided interleaving*, a family of training and decoding schemes that interleave per-token confidence cues directly into the token stream. Specifically, we have three variations: **Full-INTL** (interleave across the entire sequence), **Ans-INTL** (interleave only in the answer span), and **Reason-INTL** (interleave only within marked reasoning segments). We provide an evaluation protocol on GSM8K and BIG-Bench-Mistake (reasoning subset), along with attention-pattern and loss-dynamics observations, to understand how models make use of confidence signals. This work introduces a simple, general mechanism to make internal uncertainty explicitly available for guiding downstream generation.[1]

1 Introduction

Large language models generates rich, step-wise uncertainty signals such as entropy and probability distribution over the full vocabulary that correlate with calibration, truthfulness, and error modes. Prior work leverages these signals for self-verification or critique–correct loops (Miao et al., 2023; Gou et al., 2024), but systematic studies show that models often struggle to reliably locate their own reasoning faults (Huang et al., 2024), even if they can sometimes repair them once highlighted (Tyen et al., 2024). In parallel, uncertainty has been used for control and detection: entropy-aware decoding (Zhang et al., 2024), detectors over internal representations (Li and Goldwasser, 2021; Zhou et al., 2021; Orgad et al., 2025), and confidence signals used for routing or hallucination detection (Chuang et al., 2025; Shelmanov et al., 2025). Yet across existing works, the uncertainty computed at each step is typically consumed immediately, to gate a sampler or trigger a detector, and then discarded, leaving subsequent generations blind to the full history of past confidence. We ask a simple question: if models were persistently reminded of how uncertain they were at each prior step, could they condition their future choices on that history? To investigate, we introduce *entropy-guided interleaving*, a family of training/decoding schemes that insert quantized confidence cues (derived from next-token uncertainty) directly into the sequence as lightweight "confidence tokens." Intuitively, this turns ephemeral uncertainty into an explicit recurrent signals in the context window so that every future token is produced in view of all prior uncertainties. We study three practical variants: (i) Full-INTL interleaves confidence alongside every token, (ii) Ans-INTL interleaves only in the answer span to focus uncertainties where generation happens, and (iii) Reason-INTL interleaves within marked reasoning segments to regularize chain-of-thought without altering answer decoding. Our main contributions are

- We identify and formalize the gap between one-shot uses of uncertainty (for detection/routing) and *persistent* conditioning on uncertainty history, proposing a method to expose past confidence to future decoding steps.
- We present *entropy-guided interleaving*, a simple, model-agnostic mechanism that inserts discretized confidence cues into the token stream, with three variants usable at both training and inference.
- We provide an evaluation protocol and diagnostics (attention flow and loss dynamics) on GSM8K and BIG-Bench Mistake to probe whether and how models *use* confidence cues during generation.

[1] Preprint. Work in progress.

2 Related Work

Self-verification and correction. There are mixed evidence on whether models can check and correct their own reasoning. Several works propose zero-shot or tool-aided self-verification and critique-correct loops (Miao et al., 2023; Gou et al., 2024), yet systematic studies find that today's LLMs often fail at locating reasoning errors (Huang et al., 2024), even when they can fix them once pointed out(Tyen et al., 2024). New benchmarks target step-level verifiers and critique-correct ability to evaluate how well LLMs can self-correct (Lin et al., 2024; Jacovi et al., 2024).

Uncertainty and hallucination detection. Internal signals (entropy, logits, hidden states) carry information about truthfulness and confidence. Recent work formalizes entropy-based controls at decoding time (Zhang et al., 2024), builds detectors over internal representations (Li and Goldwasser, 2021), and shows that models often "know" more about correctness than they reveal in their outputs (Zhou et al., 2021; Orgad et al., 2025). Surveys and evaluations consolidate these techniques and highlight generalization limits.

Reusing confidence signals. Most relevant to this paper, some recent studies expose confidence back to the model as a recurrent input such as using confidence tokens for routing (Chuang et al., 2025), or using an auxiliary heads trained to quantify uncertainty for downstream checking (Shelmanov et al., 2025). However, these methods use uncertainty values once for each generation step, and discard them for subsequent generations. Our approach reuse all confidence signals throughout so that all subsequent generations are aware of each of previous steps' uncertainties.

Overall, prior works establish the promise and limits of self-verification, and the utility of uncertainty signals for control and detection. Our contribution targets the underexplored recurrent usage of uncertainty signals: feeding step-wise confidence signals into the generative context itself to test whether models can condition on their own uncertainty while decoding.

3 Methods

This section formalizes our *entropy-guided interleaving* framework. We introduce notation, define the three variants (Full-/Ans-/Reason-INTL), and describe how training and inference are carried out, including sequence construction, masking, and next-token loss functions.

3.1 Preliminaries and Notation

Let $\mathcal{V}$ be the base vocabulary and $\mathcal{V}_c$ a disjoint set of reserved *confidence tokens* ($\mathcal{V} \cap \mathcal{V}_c = \emptyset$) which are typically each model's reserved and unused special tokens. Given an input sequence $x_{1:T} \in \mathcal{V}^T$, an autoregressive LM p_θ defines the next-token distribution

$$p_\theta(\cdot \mid x_{\le t}) \in \Delta^{|\mathcal{V}|} \quad \text{for } t = 1, \dots, T.$$

From each next-token distribution we compute a scalar *confidence/uncertainty* value $c_t \in \mathbb{R}_{\ge 0}$. We use two families with entropy based representation c_t^{H} and maximum-probability based representation c_t^{H}:

$$\begin{aligned} c_t^{\mathrm{H}} &= H\big(p_\theta(\cdot \mid x_{\le t})\big) \\ &= -\sum_{v \in \mathcal{V}} p_\theta(v \mid x_{\le t}) \log p_\theta(v \mid x_{\le t}), \end{aligned} \tag{1}$$

$$c_t^{\max} = \max_{v \in \mathcal{V}} p_\theta(v \mid x_{\le t}) \tag{2}$$

A discretization process $Q : \mathbb{R}_{\ge 0} \to \mathcal{V}_c$ maps c_t to a token $\hat{c}_t = Q(c_t)$. We parameterize Q by bin edges $\mathbf{b} = (b_0 < \cdots < b_K)$ and an index-to-token map $\sigma : \{1, \dots, K\} \to \mathcal{V}_c$:

$$Q(c) = \sigma(k(c)), \tag{3}$$

$$k(c) = \min\big\{k \in \{1, \dots, K\} : c \in [b_{k-1}, b_k)\big\}. \tag{4}$$

In practice we use $K = 10$ bins with (i) uniform *entropy bins* with dedicated IDs in $\mathcal{V}_c$, and (ii) *max-probability bins* mapped to either $\mathcal{V}_c$ or compact numeral token IDs $\hat{c}_t$. We quantize $p_{\max,t}$ to an integer in $\{1, \dots, 10\}$ and reuse the *existing* single-token IDs for the numerals 1–10. We use nearest-integer rounding with clamping:

$$q_t = \mathrm{clip}\big(\lfloor 10\, p_{\max,t} + 0.5 \rfloor,\ 1,\ 10\big), \tag{5}$$

$$\hat{c}_t = \mathrm{ID}_{\mathrm{tok}}(\mathtt{str}(q_t)). \tag{6}$$

Example: $p_{\max,t} = 0.73 \Rightarrow q_t = 7 \Rightarrow \hat{c}_t =$ the tokenizer's ID for the string "7".

Interleaving operator. Let $m_t \in \{0, 1\}$ indicate whether position t should be interleaved with a confidence token. Define the cumulative sum $M_t = \sum_{i=1}^{t} m_i$ and the index mapping from original token positions to new positions after inserting confidence tokens

$$\pi(t) = t + M_t \quad \text{for } t = 1, \dots, T.$$

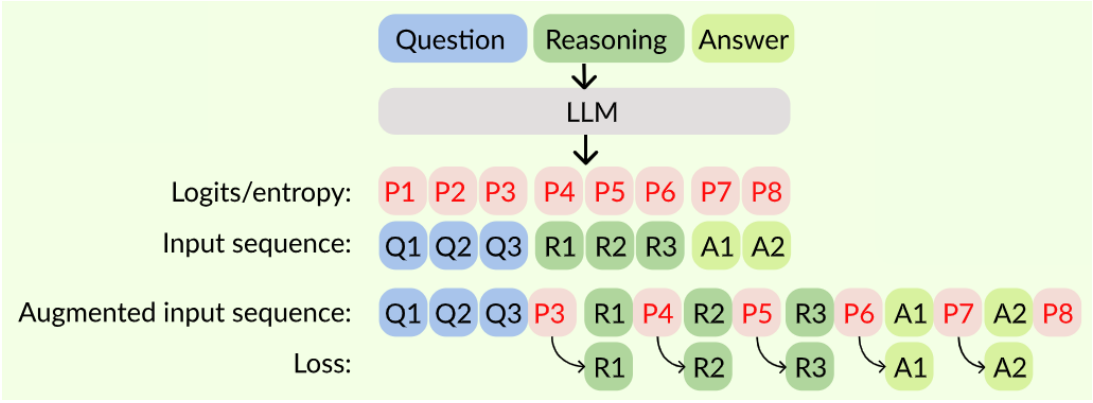

Figure 1: Confidence values are inserted after each token during fine-tuning so that all past confidence values are utilized during all future generations.

The interleaved sequence $y_{1:(T+M_T)} \in (\mathcal{V} \cup \mathcal{V}_c)^{T+M_T}$ has $y_{\pi(t)} = x_t$, and if $m_t = 1$ then $y_{\pi(t)-1} = \hat{c}_t$ else no confidence token is inserted before x_t. Thus confidence tokens, when present, appear *immediately before* their associated content token.[2]

3.2 Variants of Entropy-Guided Interleaving

We instantiate m_t based on which part of the sequence should expose confidence:

Full interleaving (Full-INTL). Set $m_t = 1$ for all $t \in \{1, \dots, T\}$. Every position is augmented, i.e., $[\hat{c}_1, x_1, \hat{c}_2, x_2, \dots, \hat{c}_T, x_T]$. Sequence length is effectively doubled.

Answer-only interleaving (Ans-INTL). Assume a standard supervision layout with question/prompt tokens $\mathcal{Q}$ (label -100) and answer tokens $\mathcal{A}$ (labeled) such that $\mathcal{Q} \cup \mathcal{A} = \{1, \dots, T\}$ and $\mathcal{Q} \cap \mathcal{A} = \emptyset$. Set $m_t = \mathbb{1}[t \in \mathcal{A}]$, i.e., only answer tokens are preceded by $\hat{c}_t$.

Marker-based reasoning interleaving (Reason-INTL). Let the prompt be segmented as $[\mathcal{Q} \mid \mathcal{R} \mid \mathcal{A}]$ using a marker or known delimiter for the reasoning region $\mathcal{R}$. Set $m_t = \mathbb{1}[t \in \mathcal{R}]$; reasoning tokens are interleaved, question and answer remain plain.

3.3 Two-Pass Training and Objective

We use a two-pass procedure per example:

Pass 1 (confidence extraction, no gradients). Run teacher-forcing on the original $x_{1:T}$ to compute $c_t = g\big(p_\theta(\cdot \mid x_{\le t})\big)$ for the chosen signal ($g \in \{\text{entropy}, \max\}$). Discretize to $\hat{c}_t = Q(c_t)$.

Pass 2 (learning with interleaved inputs). Build y via the interleaving operator using the variant-specific m_t. Feed y to the same model p_θ and compute the standard next-token cross-entropy *only* at the original-token positions. Concretely, define a supervision mask

$$w_s = \begin{cases} 1, & \text{if } s = \pi(t) \text{ for some } t \in \{1, \dots, T-1\} \\ 0, & \text{otherwise} \end{cases}$$

and labels ℓ_s with the usual one-step shift. If $s = \pi(t)$, set $\ell_s = x_{t+1}$; otherwise $\ell_s = -100$ (ignored). The loss is the same as standard cross entropy loss but only for original token positions

$$\mathcal{L}(\theta) = -\sum_{s=1}^{|y|-1} w_s \log p_\theta\big(\ell_s \mid y_{\le s}\big).$$

The confidence tokens never receive direct prediction loss. Causal masks and positional indices are constructed on y as usual. Gradients flow only in Pass 2.

3.4 Inference

At decoding time we mirror the training-time interleaving pattern:

- **Full-INTL.** After sampling (or selecting) x_t, compute c_t, emit $\hat{c}_t = Q(c_t)$, then condition the next step on $[\dots, \hat{c}_t, x_t]$.
- **Ans-INTL.** Use the plain prompt. During answer generation, after each newly produced answer token x_t (i.e., $t \in \mathcal{A}$ as it unfolds), compute/insert $\hat{c}_t$ and continue decoding.

[2] Placing $\hat{c}_t$ immediately before x_t keeps the supervision/readout alignment simple; swapping the order is straightforward and does not change the loss definition below.

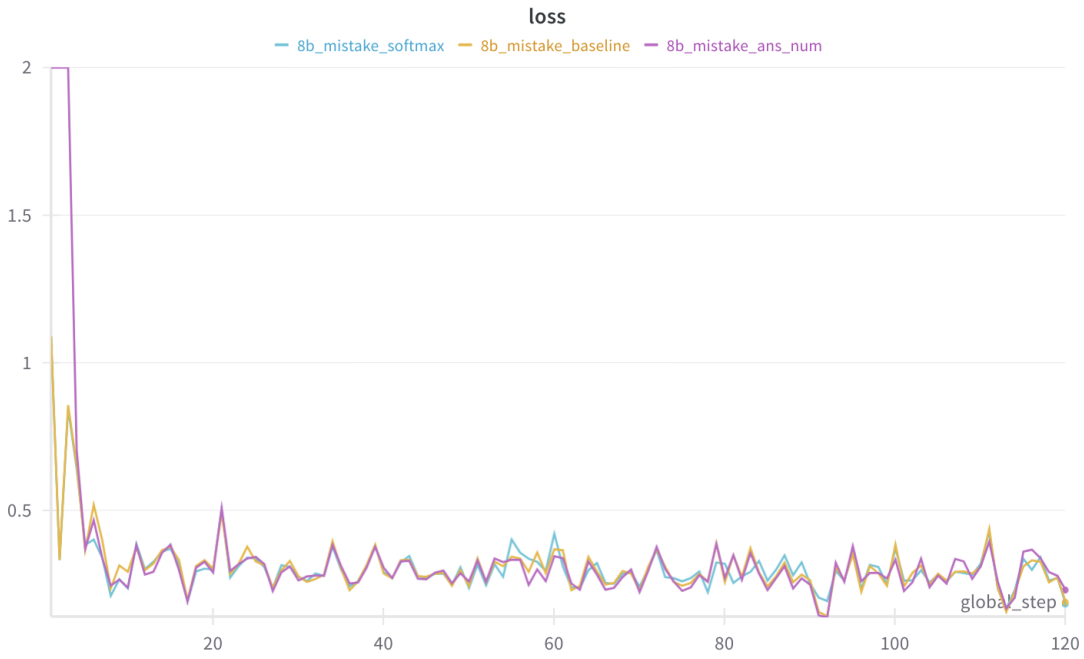

Figure 2: Training loss over three runs using Big-Bench-Mistake. Baseline indicates finetuning without additional uncertainty tokens. Softmax indicates inserting p_{max} values mapped to reserved unused special tokens. Ans num indicates inserting p_{max} values mapped to existing integer tokens.

- **Reason-INTL.** Given a prompt with an explicit reasoning context $\mathcal{R}$, compute and insert $\hat{c}_t$ for $\mathcal{R}$ *only* before answer decoding. The answer $\mathcal{A}$ is generated without further interleaving.

The user-visible text is obtained by stripping $\hat{c}$-tokens from the final sequence. During evaluation, kv-caching mechanism incrementally cache two tokens (original token and uncertainty token) at once, in contrast to the standard caching of one additional token at a time.

4 Experiments

We finetune and test on GSM8K (Cobbe et al., 2021) and Big-Bench Mistake's multistep arithmetic subset (Tyen et al., 2024) using Llama-3.1-1B and Llama-3.1-8B (Touvron et al., 2023). During finetuning, the batch size is 8, learning rate is 1e-4, number of warmup steps is 64 using cosine scheduler.

5 Results

Loss dynamics. Figure 2 compares training curves for three regimes: (i) a plain baseline (no uncertainty tokens), (ii) interleaving with reserved confidence tokens, and (iii) interleaving with numeric tokens reused from the base vocabulary. If the interleaved signals conveyed incremental predictive information, we would expect lower cross-entropy relative to the baseline under the same labels and optimizer. Instead, after a short transient the three trajectories collapse to nearly the same loss and develop in a highly correlated manner, suggesting that the additional tokens are functionally ignored by the model during learning.

The behavior differs only in the first few steps of fine-tuning. The numeric-token variant starts with a noticeably higher loss, consistent with an initial distribution shift caused by introducing frequent "word–number–word–number" patterns that the pretrained tokenizer/model do not treat as neutral separators. The reserved-token variant has a smaller starting loss, indicating that isolating confidence IDs from the natural-language vocabulary is less confusing. However, these differences disappear rapidly. After a small number of updates, all runs converge to indistinguishable losses, and step-to-step fluctuations are tightly correlated. This convergence pattern implies that the optimizer quickly finds a solution where inserted tokens are ignored rather than utilized.

Attention behavior. The attention heatmap in Figure 3 makes the failure reason clear. Confidence-token columns form low-attention "gaps" between

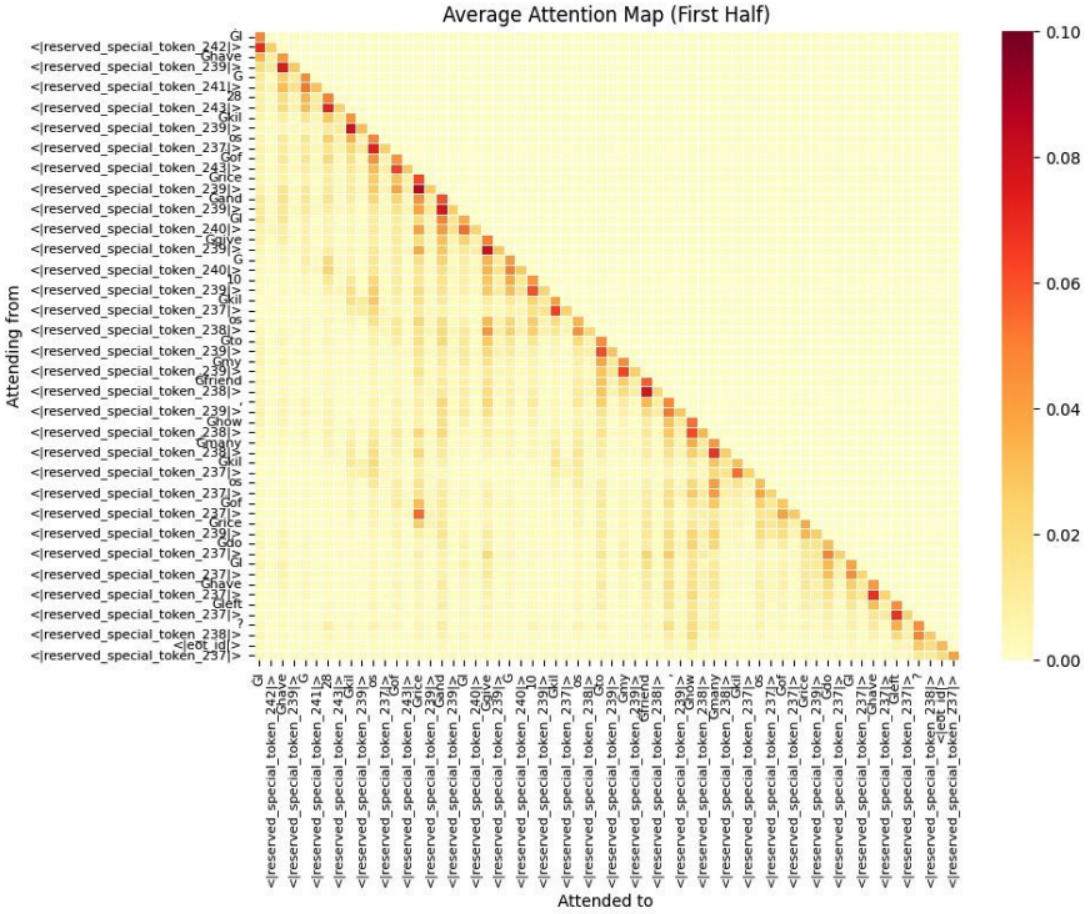

Figure 3: Average cross-layer attention for a single training sequence (Full-INTL). Columns are input positions after interleaving. Alternating light vertical bands align with confidence-token positions, while darker bands align with textual tokens. The pattern indicates that across layers, tokens predominantly attend to textual positions and systematically skip the interleaved uncertainty tokens. We observe both low incoming and low outgoing attention at confidence-token columns, consistent with the loss curves: the model learns to treat these tokens as ignorable delimiters rather than informative context.

content tokens. Because our training objective places loss only on original token predictions, gradients encourage attention to flow through positions that directly reduce next-token error (text tokens) and place minimal pressure to read from, or write to, the confidence tokens. The resulting solution is a consistent "skip pattern": content tokens attend to nearby content, while the interleaved tokens are ignored. This behavior is stable across layers and emerges within the first few optimization steps.

6 Limitations and Future Work

Several factors likely contribute to the model learning to ignore uncertainty tokens during fine-tuning:

- **Objective mismatch and easy local minima.** With standard next-token cross-entropy applied only at original token positions, the shortest path to reducing loss is to bypass the confidence tokens. Treating them as inert delimiters constitutes a shallow but effective local minimum that the optimizer reaches quickly.

- **Teacher-forcing eliminates the need to react to uncertainty.** Under teacher forcing, the prefix is always the ground-truth sequence. The model never experiences states where past predictions are wrong. Consequently, there is little gradient signal to condition future decisions on past uncertainty, because uncertainty never changes the target.

- **Data scarcity and signal-to-noise.** The BIG-Bench Mistake colored objects subset contains only 300 examples before train/eval splits. The discretized uncertainty tokens add high complexity without sufficient increase in supervision data, making it difficult for the model to

discover a stable, useful mapping from confidence tokens to improved predictions.

Several methods can be explored to target the aforementioned limitations.

- **Pre-train then RL**. Pretrain the interleaving mechanism on large generic corpora with synthetic confidence labels (from a frozen teacher) before task finetuning on GSM8K/BBH-Mistake. Alternatively, generate answer sequences with uncertainties and reward based on correctness.
- **Compress the history.** Replace per-step tokens with compact summaries such as rolling mean entropy or streak length of low-confidence emitted at sentence boundaries or every few tokens, to reduce total sequence length.
- **Parallel cross attention.** Feed confidence tokens only through a special transformer layer and combine with attention for textual tokens.

7 Conclusion

We set out to test a simple premise: if step-wise uncertainty is predictive of downstream correctness, then persistently exposing a model to its own past uncertainties should help it make better future decisions. We introduced *entropy-guided interleaving*, a family of lightweight mechanisms that discretize per-step entropy or maximum probability into "confidence tokens" and interleave them with text during both training and decoding. Despite the current limitations, we raise some interesting directions for future work.

References

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. 2025. Learning to Route LLMs with Confidence Tokens. ArXiv:2410.13284 [cs].

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. ArXiv:2110.14168 [cs].

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. ArXiv:2305.11738 [cs].

Yushi Hu, Hang Hua, Zhengyuan Yang, Weijia Shi, Noah A. Smith, and Jiebo Luo. 2023. Promptcap: Prompt-guided task-aware image captioning.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large Language Models Cannot Self-Correct Reasoning Yet. ArXiv:2310.01798 [cs].

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. 2024. A Chain-of-Thought Is as Strong as Its Weakest Link: A Benchmark for Verifiers of Reasoning Chains. ArXiv:2402.00559 [cs].

Chang Li and Dan Goldwasser. 2021. MEAN: Multi-head entity aware attention networkfor political perspective detection in news media. In *Proceedings of the Fourth Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 66–75, Online. Association for Computational Linguistics.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object- semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, page 121–137.

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. 2024. CriticBench: Benchmarking LLMs for Critique-Correct Reasoning. ArXiv:2402.14809 [cs].

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. SelfCheck: Using LLMs to Zero-Shot Check Their Own Step-by-Step Reasoning. ArXiv:2308.00436 [cs].

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2025. LLMs Know More Than They Show: On the Intrinsic Representation of LLM Hallucinations. ArXiv:2410.02707 [cs].

Artem Shelmanov, Ekaterina Fadeeva, Akim Tsvigun, Ivan Tsvigun, Zhuohan Xie, Igor Kiselev, Nico Daheim, Caiqi Zhang, Artem Vazhentsev, Mrinmaya Sachan, Preslav Nakov, and Timothy Baldwin. 2025. A Head to Predict and a Head to Question: Pre-trained Uncertainty Quantification Heads for Hallucination Detection in LLM Outputs. ArXiv:2505.08200 [cs].

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Peter Chen, and Tony Mak. 2024. LLMs cannot find reasoning errors, but can correct them given the error location. ArXiv:2311.08516 [cs].

Shimao Zhang, Yu Bao, and Shujian Huang. 2024. EDT: Improving Large Language Models' Generation by Entropy-based Dynamic Temperature Sampling. ArXiv:2403.14541 [cs].

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404, Online. Association for Computational Linguistics.

A Hyperparameters

All fine-tuning hyperparameters and setups are given in this section for replication.

Table 1: Hyperparameters for Q-only and question + text-based vision OKVQA fine-tuning

LLM	T5-Large	Flan-T5-Large	Flan-T5-XL
Batch size	2		
Gradient accumulation	16		
Start learning rate	6e-5		
LR scheduler	Linear decay at 2e-8/step		
Precision	fp32		
Epoch	6		
LoRA r	NA	NA	8
LoRA α	NA	NA	32
LoRA dropout	NA	NA	0.1

Table 2: Hyperparameters for pre-training on Conceptual Captions

LLM	Flan-T5-Large	Flan-T5-XL
Batch size	64	
Gradient accumulation	2	
Start learning rate	3e-4	
LR scheduler	Constant	
Precision	fp32	
Epoch	10	
LoRA r	NA	8
LoRA α	NA	32
LoRA dropout	NA	0.1

B Image Caption Quality

The image caption is an important component of in-context examples since it describes the context to each question. Since it is not possible to include image encoding directly to GPT-3.5, we need to use text-based vision. Three visual models are used to generate captions: Oscar+ (Li et al., 2020), BLIP2 (Li et al., 2023) and PromptCap (Hu et al., 2023).

Table 3: Hyperparameters for fine-tuning visual-language models on OKVQA without documents.

LLM	Flan-T5-Large	Flan-T5-XL
Batch size	2	
Gradient accumulation	16	
Start learning rate	1e-4	
LR scheduler	Linear decay at 2e-8/step	
Precision	fp32	
Epoch	6	
LoRA r	NA	8
LoRA α	NA	32
LoRA dropout	NA	0.1

Table 4: Hyperparameters for fine-tuning InstructBLIP-Flan-T5-XL-FrDPR on OKVQA.

LLM	Flan-T5-XL
Batch size	1
Gradient accumulation	32
Retriever learning rate	1e-5
LLM learning rate	6e-5
Precision	bf16
Epoch	4
LoRA r	8
LoRA α	32
LoRA dropout	0.1

Table 5 shows that Oscar+ captions are generally more detailed than that of BLIP2 despite BLIP2 achieving higher COCO Caption performance (Li et al., 2023, 2020). BLIP2's short caption style likely aligns better with COCO's ground truth. BLIP2 also has better text-recognition ability. BLIP2's better caption translates to better in-context performance. The baseline model that uses Oscar+ captions and performs worse than using BLIP2 captions (59.32 vs 59.55) in Table **??**.

However, both Oscar+ and BLIP2 do not add enough relevant information that can help answer the question. In contrast, PromptCap provides captions that include details about what the question is asking about. As seen in Table 5, PromptCap describes the key object of interest in more detail. For example, when the question asks about birds on TV, PromptCap states that the birds are geese, whereas the other models do not include details about the birds. PromptCap also occasionally directly answers the question. This advantage is reflected in its better performance in in-context testing. It scores 1.2 better than BLIP2 (Table **??**.)

Table 5: Examples of image captions by Oscar+, BLIP2 and PromptCap.

Question	What profession would you say this guy has?
Oscar+	a man is working on a motorcycle in front of a tent.
BLIP2	a man working on a motorcycle.
PromptCap	a man in blue overalls working on a motorcycle.
Question	The birds on the television derive their name from what country?
Oscar+	a cat sitting in front of a television watching birds
BLIP2	a cat sitting on a television
PromptCap	a cat sitting in front of a tv with a picture of geese on it.
Question	Where is this building located?
Oscar+	a large building with a clock tower on top of it.
BLIP2	a building with a clock tower.
PromptCap	a building with a clock tower in england.
Question	What does the blue p represent?
Oscar+	a black box sitting next to a brick wall.
BLIP2	a parking meter and a brick wall.
PromptCap	a parking meter with a blue p on it.
Question	What is the name of the beer?
Oscar+	a bottle of beer next to a plate of food
BLIP2	a beer and food
PromptCap	a bottle of kingfish beer and a plate of food.