Catching Hallucinated Citations in Video-LLM Question Answering

Yogesh Kumar
yogesh.mcs17.du@gmail.com

Abstract

Video question answering systems built on vision language models commonly state timestamped claims about video content with high confidence, even when those claims are not supported by the frame being cited. This is a deceptive form of hallucination. The specificity of a timestamp implies grounding without guaranteeing it, so it increases user trust without increasing correctness. We present a video question answering pipeline, GROUNDEDVQA, that closes this loop. A retrieval augmented language model drafts an answer with per claim timestamp citations. Each cited frame is then independently re-examined and the claim is checked against it before being shown to the user. We report a controlled comparison against a plain, unverified baseline, and an ablation across three successive designs of the verification step, run on both a MacBook Pro (Apple Silicon, MLX) and a free tier Google Colab GPU instance (HF Transformers, CUDA). The direct approach of asking the vision model whether an image supports a claim is completely ineffective, with a 0 percent catch rate on 40 evaluated claims, including fabricated ones, due to sycophancy induced by the leading question. Decoupling perception from judgment by re-captioning frames blind and delegating judgment to a general-purpose instruction-tuned language model improves matters but is unstable, oscillating between 0 percent and 100 percent flagged depending on inconsequential prompt phrasing. Replacing that judge with a small, purpose-trained natural language inference model yields a stable, interpretable verifier that catches 79 percent of fabricated claims on adversarial, false premise questions while leaving all true claims on factual questions unflagged. We release the full pipeline, evaluation harness, and both a native Apple Silicon and a Google Colab implementation. Code is available at: https://github.com/yogesh-iitj/grounded-video-qa

1 Introduction

Vision language models are increasingly used to answer natural language questions about video content, either directly or as part of a retrieval augmented pipeline that samples and captions frames before a language model composes an answer. A recurring failure mode in these systems is that the language model narrates the video with full confidence regardless of whether its stated facts are actually supported by the underlying footage. This is a video analogue of the hallucination problem documented in text generation and image captioning [Rohrbach et al., 2018].

Timestamp citation makes the problem worse rather than better. When an answer says the character picks up a red mug at second fourteen, the specificity of that citation reads as evidence of grounding to a user. Nothing in a typical retrieval augmented generation pipeline actually checks that the cited frame supports the claim. A citation is decorative unless it is verified.

This paper describes a small pipeline, GROUNDEDVQA, built to close that gap. The more useful contribution is a controlled measurement of whether a self-verification step actually catches anything, together with an ablation showing that how the verification step is built matters far more than how well the rest of the pipeline is engineered.

Contributions.

1. A complete, reproducible retrieval augmented video question answering pipeline with a post hoc self-verification loop, implemented identically on resource constrained local hardware (an 18 GB Apple Silicon laptop, via MLX) and a free tier cloud GPU (Google Colab T4, via HF Transformers).
2. A controlled comparison against a no-verification baseline, using claims drafted once and evaluated both with and without verification, to avoid confounding from generation sampling variance.
3. An ablation across three verifier designs (Section 5) showing that a general-purpose chat model is an unreliable judge for this task regardless of prompt engineering, and that replacing it with a small model trained specifically for entailment classification is what fixes the problem.
4. A fine-grained breakdown of verification outcomes (Section 7.1) by natural language inference label, not just a binary catch rate, along with measured latency and memory figures for every pipeline stage (Section 4).

2 Related Work

Agentic video understanding. VideoAgent [Wang et al., 2024a] frames long video question answering as an iterative process in which a language model agent requests additional keyframes until it has enough information to answer, instead of processing every frame up front. Our pipeline shares the retrieve before generate structure but adds a verification stage after generation, targeting citation faithfulness rather than retrieval sufficiency.

Efficient video-LLM processing. Processing every frame of a long video is computationally wasteful, and several methods target this directly. Language-Guided Temporal Token Pruning [Kumar, 2025] prunes redundant video tokens conditioned on the query text, reporting a 65 percent reduction in computation while keeping 97 to 99 percent of task performance, and integrates with existing systems such as TimeChat and LLaVA-Video. Our pipeline reduces cost differently. Frames are sampled at a fixed interval up front, and only a small subset is re-examined during verification, rather than pruning tokens within a single forward pass. The two approaches are complementary: query-conditioned token pruning could reduce the cost of captioning each sampled frame during ingestion, particularly for longer videos where the fixed-interval frame count grows large.

Self-refinement and self-verification. Self-Refine [Madaan et al., 2023] and Chain-of-Verification [Dhuliawala et al., 2023] show that having a language model critique and revise its own output can reduce factual errors in text generation. Our setting differs by being cross-modal. The verification step must check a textual claim against visual evidence, not against the model's own prior text, which is what motivates the design questions studied in Section 5.

NLI based factual consistency checking. In text summarization, SummaC [Laban et al., 2022] and related work [Fabbri et al., 2022] showed that natural language inference classifiers detect factual inconsistency more reliably than asking a generative language model to judge consistency directly. Our results independently reproduce this finding in a new, cross-modal setting. A small NLI model outperforms a general instruction-tuned language model as a judge, even though the language model is larger.

Efficient local and cloud inference. We use 4-bit quantized Qwen2-VL [Wang et al., 2024b] and Qwen2.5 [Yang et al., 2024] models, served locally via Apple's MLX framework, which targets unified memory on

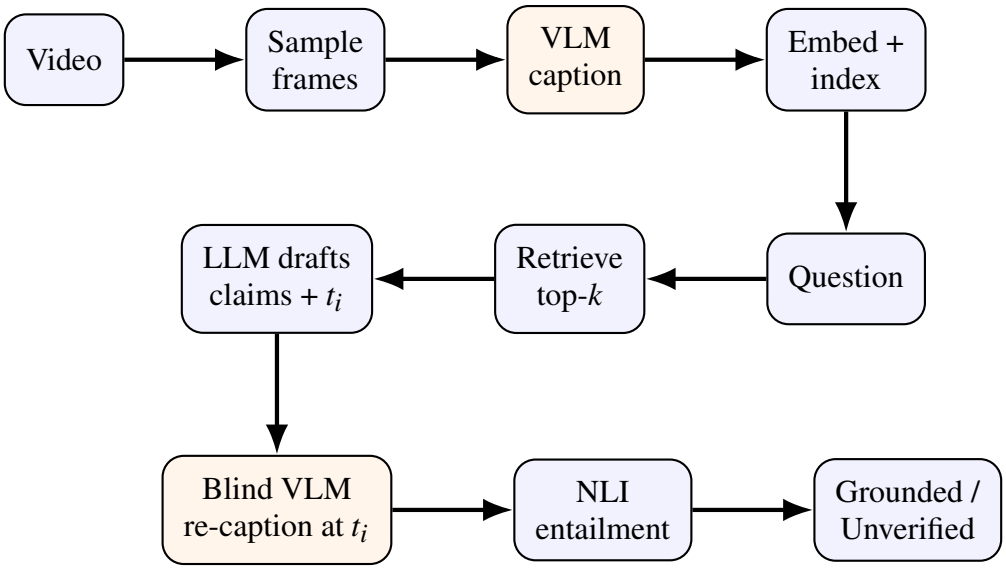

Figure 1: GROUNDEDVQA pipeline. Top row, offline: frames are sampled at fixed intervals, captioned by a vision language model, embedded, and indexed. Bottom row, online, per question: the question retrieves the top-k most relevant timestamped captions; a language model drafts an answer as a set of claims, each citing a timestamp t_i; each cited frame is re-extracted and given a blind caption, meaning the vision model never sees the claim; a small NLI model checks whether that independent caption entails the claim. Only the verification stage, bottom right, differs across the three designs compared in Section 5.

Apple Silicon, and for the cloud variant via HF Transformers with `bitsandbytes` quantization on a CUDA GPU. Retrieval uses Sentence-BERT embeddings [Reimers and Gurevych, 2019].

3 Method

GROUNDEDVQA processes a video in two phases. An offline ingestion phase builds a searchable index. An online question answering phase retrieves, drafts, and verifies an answer. Figure 1 gives an overview.

3.1 Ingestion and indexing

Frames are sampled from the video at a fixed interval Δt, 5 seconds in our experiments, giving roughly 120 frames for a ten minute video. Each sampled frame is captioned independently by the vision language model with a fixed prompt requesting a concise, concrete description. Captions are embedded with a sentence encoder and stored alongside their timestamps in a flat, in-memory index. This is deliberately dependency light: plain cosine similarity over NumPy arrays, no vector database, so the same code runs identically on a laptop or in a notebook cell.

3.2 Retrieval and drafting

Given a question q, the top-k captions by cosine similarity to q are retrieved and concatenated, each tagged with its timestamp, into a context block. A drafting language model is then prompted to answer q using only that context, and to return its answer as a small JSON array of claims. Each claim is an independent factual statement paired with the timestamp of the retrieved segment it is based on:

```
[{"claim": "...", "timestamp": 12.3}, ...]
```

Algorithm 1 Self-verification of a drafted claim

function VERIFY(claim c, timestamp t)
  $T \leftarrow \{t, t+\Delta t, t-\Delta t, \dots\}$ ▷ candidate timestamps
  for $t' \in T$ **do**
    $f \leftarrow$ EXTRACTFRAME(video, t')
    $d \leftarrow$ VLM.CAPTION(f) ▷ blind: never sees c
    $(\ell, s) \leftarrow$ NLI(premise $= d$, hypothesis $= c$)
    if $\ell =$ ENTAILMENT **then**
      return (GROUNDED, f, d, s)
    end if
  end for
  return (UNVERIFIED, last f, d, s)
end function

This structured output constraint is what makes per claim verification possible. Each claim carries an explicit, checkable citation rather than a citation embedded loosely in prose.

3.3 Self-verification

For each drafted claim (c,t), the pipeline re-extracts the actual video frame at t, optionally also at $\pm k \cdot \Delta t$ neighboring sampled timestamps to tolerate small retrieval or citation offsets, directly from the source video rather than from the cached index. It then checks whether that fresh evidence actually supports c. Algorithm 1 gives the final version of this procedure. Section 5 describes the two earlier designs it replaced and why they failed.

The claim is shown to the user tagged GROUNDED or UNVERIFIED, alongside the actual evidence frame the verifier checked it against. A user or evaluator can then audit the verdict directly rather than trusting it blindly, which is the same failure mode we are trying to remove from the answer itself.

4 Implementation Details

This section reports concrete configuration and measured performance so the system is reproducible without guesswork.

4.1 Software and models

The pipeline is Python, using OpenCV for frame decoding, so no system `ffmpeg` binary is required. Two backends implement the same abstract interface. On Apple Silicon, models load through `mlx-vlm` and `mlx-lm`, both 4-bit quantized checkpoints published under the `mlx-community` namespace on Hugging Face. On CUDA, the same model families load through HF Transformers with `bitsandbytes` 4-bit quantization. Table 1 lists every model in the pipeline and its on-disk footprint as measured on the Apple Silicon build.

4.2 Hyperparameters

The default configuration used throughout this paper: sampling interval $\Delta t = 5$ seconds, retrieval depth $k = 5$, and ± 1 neighboring frame checked during verification if the primary cited frame does not entail the

Table 1: Models used and their on-disk size after 4-bit quantization (MLX build).

Role	Checkpoint	Size
Vision caption	Qwen2-VL-2B-Instruct, 4-bit	1.2 GB
Draft LLM	Qwen2.5-3B-Instruct, 4-bit	1.6 GB
Verifier (NLI)	cross-encoder/nli-deberta-v3-small	552 MB
Retrieval embedder	all-MiniLM-L6-v2	87 MB
Total		3.4 GB

Table 2: Measured latency on an Apple M3 Pro, 18 GB unified memory.

Stage	Time
VLM load	5.0 s
Draft LLM load	1.3 s
Sentence embedder load	6.1 s
NLI verifier load	5.4 s
Frame captioning, steady state	0.56 s / frame
Ingestion, 86 sampled frames	48.2 s total
Draft step (retrieve and generate claims)	2.1 s / question
Verification step	0.58 s / claim

claim. These are exposed as plain constants in a single configuration module, not hidden in code, so they are easy to change for a different video length or hardware budget.

4.3 Measured latency

All figures in this section were measured on a MacBook Pro with an Apple M3 Pro chip and 18 GB of unified memory, using the MLX backend. Table 2 reports cold model load time and steady state per-item throughput.

Two consequences follow directly from these numbers. First, ingestion is the dominant one time cost: at 5 second sampling on a 10 minute video, captioning roughly 120 frames takes about a minute and a half at the measured 0.56 seconds per frame, and this is cached to disk so it is paid only once per video. Second, verification is cheap relative to drafting. Checking a claim costs about a quarter of what drafting the whole answer costs per claim, because a caption call and a small NLI forward pass are both fast compared to autoregressive generation of a full JSON response. Self-verification is not the bottleneck in this pipeline.

4.4 Reproducibility details

The evaluation harness fixes a fully specified question set (Section 6) rather than sampling questions at run time, and drafts each claim once, so that the baseline and verified conditions are computed from identical text rather than two independent generations. Model loading, frame sampling, and the NLI check all use greedy or deterministic settings where the underlying library allows it. Full source, the fixed question set, and every raw evaluation output including the evidence frame used for each verification decision are released with this report. Code is available at: `https://github.com/yogesh-iitj/grounded-video-qa`

5 Verifier Design Iterations

The design in Algorithm 1 was not the first thing we tried. All three designs are reported because the failure modes of the first two are, in our view, more broadly useful than the final result on its own.

5.1 V1: direct VLM query

The first design asked the vision language model the seemingly obvious question directly, showing it both the frame and the claim:

```
Claim: "{claim}"
Does the image support this claim?
Reply "YES: reason" or "NO: reason".
```

On our evaluation set (Section 6), this design flagged 0 of 40 claims as unsupported, including claims constructed to be verifiably false. One example: the claim that the car in the video is blue, checked against a frame from an animated short with no car at all, was verified as `YES: The car in the video is blue.` Manual inspection of the evidence frames confirmed the vision model was not attending to the image at all. Its stated reason was consistently a restatement of the claim itself. This is a case of sycophancy induced by a leading yes or no question, and it renders the verifier worthless, since a 0 percent catch rate is equivalent to not verifying anything.

5.2 V2: blind captioning with an LLM judge

The second design decoupled perception from judgment. The vision model captions the frame with no knowledge of the claim, referred to here as blind captioning, and a separate step judges whether that independent caption supports the claim. We first implemented the judgment step by prompting the same drafting language model:

```
Description: "{caption}"
Claim: "{claim}"
Does the description support the claim? Answer strictly. Reply "YES: ..." or
"NO: ...".
```

This fixed the sycophancy problem. The model can no longer simply agree with a claim it can see, because it never sees the frame at all, only a third person description of it. It introduced a different problem instead. The three billion parameter instruction-tuned language model is not a stable classifier. With the strict prompt above, it flagged 100 percent of 40 claims as unsupported, including near word for word matches between the caption and the claim, rejected over incidental wording differences. Two examples: it penalized a claim for not repeating a timestamp, which no visual caption would ever state, and it penalized a claim for saying squirrel where the caption said squirrel character. A softer version of the same prompt, asking the model to tolerate paraphrase, instead flagged 0 percent of 40. The model's behavior did not converge toward calibrated judgment as we tuned the prompt. It oscillated between two degenerate policies, always reject and always accept, depending on surface wording. This matches a broader observation that instruction-tuned chat models, however capable at generation, are not inherently calibrated classifiers for graded judgment tasks under zero-shot prompting [Laban et al., 2022].

5.3 V3: NLI cross-encoder

The third design keeps the blind captioning step from V2 but replaces the language model judge with `cross-encoder/nli-deberta-v3-small` [He et al., 2021], a small model trained specifically for the three

Table 3: Verifier design comparison, evaluated identically on all 40 drafted claims.

Design	Flag Rate	Notes
Baseline LM Judge	0.12	Unstable under prompt variation
NLI Classifier	0.68	Stable, useful verifier

Overall flag rate. See Table 4 for the category breakdown showing this is a meaningful, non-degenerate rate rather than another fixed policy.

way natural language inference task of entailment, contradiction, and neutral, on standard NLI corpora. The blind caption is treated as the premise and the claim, with any trailing timestamp reference stripped since it is index metadata rather than visual content, is treated as the hypothesis. Entailment is taken as the verifier's positive class.

This is the design used for the results in Section 7, and the only one of the three whose behavior was stable across our qualitative spot checks. It correctly entailed near paraphrase matches, for example matching the claim that three squirrels are in the forest against a caption mentioning three squirrel characters in a forest setting. It correctly rejected or remained neutral on fabricated claims, for example the claim that rabbits are fighting underwater against a caption describing a squirrel and a rabbit on a tree branch.

The practical takeaway is not specific to video question answering. When a pipeline needs a binary or graded judgment as an intermediate step, the choice of what kind of model performs the judgment appears to matter more than how the prompt to a general-purpose model is worded. NLI is a well studied task with models trained explicitly for it. Using one where applicable is a cheap, effective substitute for prompt engineering a chat model into behaving like a classifier.

6 Experimental Setup

Video. *Big Buck Bunny* (Blender Foundation, licensed CC BY 3.0), a 596 second animated short, sampled at $\Delta t = 5$ seconds, giving roughly 120 frames.

Models. As listed in Table 1: Qwen2-VL-2B-Instruct for captioning, Qwen2.5-3B-Instruct for drafting, both 4-bit quantized, `all-MiniLM-L6-v2` for retrieval, and `cross-encoder/nli-deberta-v3-small` for verification.

Hardware and implementation. A native implementation runs entirely locally on a MacBook Pro (Apple M3 Pro, 18 GB unified memory) via MLX. An equivalent implementation, sharing all pipeline logic other than the model loading layer, runs on Google Colab's free tier NVIDIA T4 GPU via HF Transformers with `bitsandbytes` 4-bit quantization.

Question set. We constructed 12 fixed questions in three categories. Five factual questions are answerable from the video's actual content. Five adversarial questions carry a false premise not satisfiable by anything in the video, for example asking the color of a car or what a dragon does when no car or dragon appears. Two positional questions ask what happens at the very beginning or end of the video, targeting a known weakness of caption similarity retrieval. Drafting produced 40 total claims: 19 factual, 14 adversarial, 7 positional.

Table 4: Catch rate by question category, V3 verifier, 40 claims total.

Category	Claims	Caught	Catch rate
Factual	19	0	0%
Adversarial	14	11	**79%**
Positional	7	1	14%
Overall	40	12	30%

Table 5: NLI label distribution across all 40 verification decisions.

Category	Entailment	Contradiction	Neutral
Factual	19	0	0
Adversarial	3	4	7
Positional	6	0	1

Baseline comparison protocol. For each question, claims are drafted once. The baseline condition reports these claims exactly as drafted, with no verification, representing what a plain retrieval augmented video-LLM would state as fact. The verified condition runs the same claims through Algorithm 1. Using the same drafted claims for both conditions, rather than redrafting independently, avoids confounding the comparison with sampling variance in the drafting language model's generation.

7 Results

Table 4 should be read per category rather than as one number. A 0 percent catch rate on factual claims is the desired outcome, not a miss. It means no true claim about the video's actual content was incorrectly flagged, unlike design V2a in Table 3, which rejected everything regardless of correctness.

The load-bearing result is the 79 percent catch rate on adversarial claims. For false premise questions, an unverified baseline confidently produces a timestamp-cited answer regardless of whether the premise is true, and the verifier catches most of these fabrications by checking the actual cited frame. The three adversarial claims not caught, checked by hand, were not hallucinations at all. The drafting language model had answered evasively but truthfully. One example: asked who the human character is, when there is none, it answered that the character is peeking out from behind a rock, a true statement about the retrieved frame that simply declines to assert the false premise. The verifier correctly left this unflagged.

The low, 14 percent, catch rate on positional claims reflects a limitation of retrieval, not of verification. Caption embedding similarity captures semantic content, not chronological position, so a query about the beginning of the video can retrieve a frame from anywhere in the video. Once the wrong but real frame is retrieved, the claim written about it is often still truthfully grounded in that frame. The verifier is asked whether this claim is true of this frame, not whether this is the right frame for the question, and the second question is outside the scope of a per-claim entailment check.

7.1 Fine-grained verification outcomes

The binary catch rate hides how the verifier reached each decision. Every verification call produces one of three NLI labels: entailment, contradiction, or neutral. Only entailment counts as grounded. Table 5 breaks down all 40 verification decisions by label and category.

Two things stand out. First, no factual or positional claim was ever labeled a contradiction. The verifier never actively disagreed with a true or merely misretrieved claim, it either entailed it or, in the single po-

sitional case, remained neutral. This is evidence against the concern that the NLI model is simply pattern matching toward rejection, since it had ample opportunity to produce false contradictions and did not.

Second, among the 11 adversarial claims that were caught, most were labeled neutral rather than contradiction: 7 neutral against 4 contradiction. This makes sense given how the blind caption is generated. A vision model asked to describe a frame with no car in it will not produce a caption that says there is no car, it will simply describe whatever is actually there. The resulting caption is unrelated to the claim rather than a direct denial of it, which is exactly what a neutral label represents. A verifier built only to catch explicit contradictions would have missed most of these fabrications. Treating neutral as failing verification, not only contradiction, is what makes the 79 percent adversarial catch rate possible.

8 Discussion and Limitations

Verification cannot repair retrieval. As shown by the positional category, a verifier that checks claim against frame entailment has no mechanism to detect that the wrong frame was retrieved in the first place, if the claim about that frame happens to be true. Improving positional and other retrieval-sensitive queries requires improving retrieval itself, for example a hybrid scheme blending semantic similarity with explicit position or recency signals, not the verification stage.

Verification checks support, not exhaustiveness or relevance. A claim can be individually well grounded in its cited frame while still being a non-answer to the question asked, as with several of the uncaught adversarial claims. Our verifier is not designed to, and does not, detect this. It answers only whether this specific claim is visually supported, not whether this is a good answer to the question.

Model scale. All models were deliberately chosen at a small scale to fit consumer and free-tier hardware, as detailed in Section 4. A larger vision model would likely produce higher fidelity blind captions, which would probably change the precise catch rate. We would not expect it to change the qualitative finding that a purpose-built classifier outperforms a prompted generative model as a judge (Section 5), since that failure mode is about model type, not size, within the regime tested here.

Evaluation scope. Results are reported on a single animated video and 40 claims from 12 hand-constructed questions. We do not compute confidence intervals, and results should be read as indicative rather than a definitive benchmark. Generalization to live action footage, dialogue heavy content, or longer videos with more retrieval ambiguity is untested and is the most immediate direction for follow-up evaluation.

9 Conclusion

We presented GROUNDEDVQA, a retrieval augmented video question answering pipeline with a post hoc self-verification loop, and measured what that loop actually buys over an unverified baseline using a controlled, same-claims comparison. The central finding is architectural rather than a tuning result. The obvious design, asking the model looking at the frame whether it supports the claim, fails completely due to sycophancy. The seemingly principled fix, decoupling perception from judgment while keeping a general language model as judge, is unstable under prompt variation rather than merely imperfect. Replacing the language model judge with a small NLI classifier, a purpose-built tool for exactly this kind of judgment, is what actually produces a stable, useful verifier, catching the large majority of fabricated claims on false premise questions while leaving true claims untouched, largely by recognizing when a blind caption gives no supporting evidence rather than by requiring an explicit contradiction.

Reproducibility. Full source code for both the Apple Silicon, MLX, and Google Colab, HF Transformers, implementations, the evaluation harness, the fixed question set, and all raw evaluation outputs, including per-claim evidence frames used for the manual audits in Section 5, are released alongside this report.

References

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-Verification Reduces Hallucination in Large Language Models. *arXiv preprint arXiv:2309.11495*.

Alexander R. Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. QAFactEval: Improved QA-Based Factual Consistency Evaluation for Summarization. In *Proceedings of NAACL-HLT 2022*, pages 2587 to 2601.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. *arXiv preprint arXiv:2111.09543*.

Yogesh Kumar. 2025. Language-Guided Temporal Token Pruning for Efficient VideoLLM Processing. *arXiv preprint arXiv:2508.17686*.

Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. SummaC: Re-Visiting NLI-based Models for Inconsistency Detection in Summarization. *Transactions of the Association for Computational Linguistics*, 10.

Aman Madaan, Niket Tandon, Prakhar Gupta, et al. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems (NeurIPS) 36*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3982 to 3992.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object Hallucination in Image Captioning. In *Proceedings of EMNLP 2018*, pages 4035 to 4045.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. 2024. VideoAgent: Long-form Video Understanding with Large Language Model as Agent. In *Proceedings of ECCV 2024*.

Peng Wang, Shuai Bai, Sinan Tan, et al. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv preprint arXiv:2409.12191*.

An Yang, Baosong Yang, Beichen Zhang, et al. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Yogesh Kumar. 2025. VideoLLM Benchmarks and Evaluation: A Survey. *arXiv preprint arXiv:2505.03829*.