

HalluVision at SHROOM-Vision: Hallucination Span Detection with a Classifier over a Vision-Language Backbone

Mohammed Sohail Pasha Qureshi¹, Surya Charan¹, Alapan Kuila²

¹ Dr. K.V. Subba Reddy Institute of Technology, Kurnool, India

²Indian Institute of Information Technology Design and Manufacturing Kurnool, Kurnool
sohail.shaa786@gmail.com, charansurya927@gmail.com , alapan.cse@iiitk.ac.in

Abstract

We present our system for the SHROOM-Visions 2026 shared task on fine-grained hallucination detection in vision-language model (VLM) outputs. Our approach uses three token-level internal signals from Qwen2.5-VL-3B-Instruct: hidden representations, the change in token confidence when the image is removed, and predictive uncertainty. A lightweight classifier combines these signals to predict hallucinated spans and their confidence, while LoRA enables parameter-efficient adaptation of the VLM backbone. Our submission achieves IoU scores of 0.36–0.42 and Spearman correlations of 0.41–0.46 on English, French, and Italian, ranking 10th in both French and Italian. We additionally analyze continued-training experiments that improved our local validation scores but consistently degraded official test performance. Our analysis attributes this discrepancy primarily to threshold tuning on a small validation split, which was only partially mitigated by subsequent adjustments. Increasing the backbone size from 3B to 7B parameters did not resolve the issue. These findings highlight the sensitivity of hallucination detection systems to validation size and decision-threshold selection.

1 Introduction

Vision-language models (VLMs) (Zhang et al., 2024) are increasingly used for image description, visual question answering, and caption generation. Despite their fluency, VLMs can produce hallucinations (Gunjal et al., 2024), such as inventing objects, miscounting entities, misreading text, or mischaracterizing objects that are actually present (Li et al., 2023). Automatically detecting such errors is important for assessing the visual grounding of VLMs. Figure 1 illustrates an example of a hallucinated description.

The SHROOM-Visions 2026 shared task¹ for-

¹<https://helsinki-nlp.github.io/shroom/>



Figure 1: A schematic example of a labeled hallucination (not an actual dataset item, constructed here to illustrate the annotation format).

mulates hallucination detection as a fine-grained span-localization problem. Given an image, a question, and a model-generated answer, systems must identify hallucinated character spans, assign each span a hallucination type (invention, mischaracterization, OCR error, miscounting, or other), and predict a confidence score. Performance is measured using span Intersection-over-Union (IoU) and Spearman correlation between predicted and gold confidence scores across English, French, Italian, and Chinese.

Our approach exploits internal signals of a VLM to identify potentially hallucinated tokens without requiring full model retraining. Specifically, we use the change in token confidence when the image is removed, token-level predictive entropy, and changes in token representations across layers. These signals are provided to a lightweight supervised classifier while keeping the backbone largely frozen. We further investigate parameter-efficient fine-tuning with LoRA adapters (Xu et al., 2026), allowing limited adaptation of the backbone at a substantially lower training cost. Our submitted system uses the LoRA-based variant, while we also evaluate the frozen-probe setting and additional continued-training experiments. This approach is related to recent work on internal signals for reference-free VLM evaluation (Roy Dipta et al.,

2025).

The remainder of the paper is organized as follows. Section 2 discusses related work, Section 3 describes the task and dataset, and Section 4 presents our method. Section 5 describes the experimental setup, Section 6 reports the results and analysis, Section 7 discusses our findings, Section 8 presents limitations, and Section 9 concludes the paper.

2 Related Work

Recent work on vision-language hallucination detection (Chen et al., 2024; Huo et al., 2025) has explored internal model signals as indicators of unreliable generation. Self-consistency (Taubenfeld et al., 2025) and semantic-entropy (Melamed, 1997) methods use variation or uncertainty across model generations to identify unreliable outputs (Kuhn et al., 2023), while reference-free approaches use repeated or contrastive model evaluations without external references (Roy Dipta et al., 2025). Our approach builds on this line of work by using token-level internal signals—grounding contrast, predictive entropy, and hidden representations—as features for supervised span-level hallucination detection. For efficient adaptation of large VLMs, we use Low-Rank Adaptation (LoRA), which trains a small set of low-rank parameters (Li et al., 2025) while keeping the backbone frozen (Hu et al., 2022). We further combine LoRA with 4-bit quantization (Dettmers et al., 2023) following QLoRA (Dettmers et al., 2023) to reduce memory requirements during fine-tuning.

3 Task and Data

Each SHROOM-Visions instance consists of an image, a natural-language prompt, and a response generated by a vision-language model. Training instances contain zero or more hallucinated spans, each specified by character-level start and end offsets, a hallucination category, and an annotator confidence score. The five categories used in the shared task are summarized in Table 1.

The dataset (Vázquez et al., 2026; Mickus et al., 2026b) covers English, French, Italian, and Chinese. The training split provides gold span annotations, while the test split is evaluated through the shared-task submission server. Systems are evaluated using character-level Intersection-over-Union (IoU) for span localization and Spearman correlation for predicted confidence. Scores are first

Label	Description
Invention	An entity, object, property, or event that is not present in the image.
Mischaracterization	Something present in the image but described incorrectly.
OCR error	Misreading text that is visible in the image.
Miscounting	Incorrectly reporting the quantity of visible items.
Other	Any hallucination not covered by the above (used sparingly).

Table 1: Hallucination category labels used by the shared task, adapted from the task’s underlying dataset description (Mickus et al., 2026a).

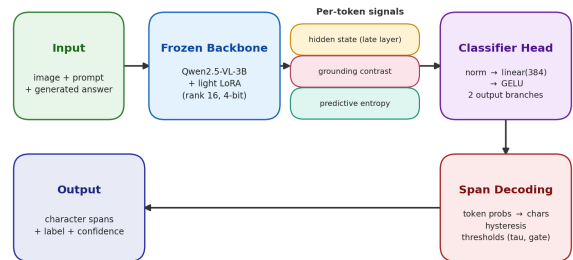


Figure 2: End-to-end pipeline. The top row runs left to right, from raw input through the frozen backbone to the classifier head; the classifier’s output then flows into span decoding (bottom right), which produces the final character-level output (bottom left).

averaged within each language and then across the four languages to obtain the overall score. Figure 1 illustrates a miscounting example, where the image contains two sheep but the generated response incorrectly reports three.

4 Method

4.1 Overview

Our pipeline consists of three stages, illustrated in Figure 2. First, the image, prompt, and response are processed by the vision-language backbone in a single forward pass. Second, per-token features are extracted from the response. Finally, a lightweight classifier predicts a hallucination probability and category for each token. Token-level predictions are subsequently converted into character-level spans using threshold-based decoding.

4.2 Backbone and per-token features

We use Qwen2.5-VL-3B-Instruct (Qwen Team, 2025) as the backbone. The image, prompt, and response are combined into a single chat-formatted input and processed with hidden-state extraction

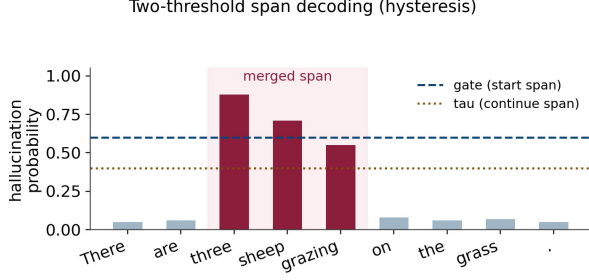


Figure 3: Two-threshold (hysteresis) span decoding on a toy example. The token *three* clears the higher *gate* threshold and starts a span; *sheep* and *grazing* fall below the gate but above the lower *tau* threshold, so they extend the same span instead of being dropped.

Layer	Shape / detail
LayerNorm	over backbone hidden size H
Linear + GELU	$H \rightarrow 384$
Hallucination head	$384 \rightarrow 1$ (sigmoid at inference)
Category head	$384 \rightarrow 5$ (softmax at inference)
Total parameters	≈ 1.9 million

Table 2: Classifier head architecture. H is the backbone’s hidden size (2048 for the 3B backbone, 3584 for the 7B backbone).

enabled. For each response token, we use its hidden representation from a late layer, approximately 90

For the frozen-probe setting, we additionally derive two token-level signals. The first is *grounding-contrast*, defined as the difference between the token log-probability with the original image and with the image replaced by a blank placeholder. The second is the model’s *predictive entropy* over the vocabulary. Grounding-contrast measures the extent to which a token’s prediction depends on the visual input, while entropy captures predictive uncertainty.

4.3 Classifier head

The classifier consists of layer normalization, a linear projection to 384 dimensions with GELU activation (Hendrycks and Gimpel, 2016), and two output branches for hallucination probability and hallucination category. Its architecture is shown in Table 2.

The head contains approximately 1.9 million trainable parameters. We optimize a weighted binary cross-entropy loss for token-level hallucination probabilities and a cross-entropy loss for category prediction. Character-level hallucination probabilities are mapped to tokens by averaging

Hyperparameter	Value
LoRA rank / alpha	16 / 32
LoRA dropout	0.15
Learning rate	5×10^{-5}
Weight decay	0.05
Optimizer	AdamW
Gradient accumulation	8 steps
Probe layer	$\approx 0.9 \times \text{depth}$
Quantization	4-bit NF4, double quant

Table 3: Main training hyperparameters used for LoRA fine-tuning of the 3B backbone.

the probabilities over the characters covered by each token. The category loss is applied only to tokens with non-trivial hallucination probability, and the hallucination loss is weighted more strongly to account for the large number of non-hallucinated tokens.

4.4 From token probabilities to character spans

Because evaluation is performed at the character level, token predictions are converted into character spans using the tokenizer’s offset mapping. Consecutive high-probability tokens are merged into a single span, as illustrated in Figure 3.

We use two thresholds: a higher *gate* threshold required to initiate a span and a lower *tau* threshold used to extend an existing span. This hysteresis reduces fragmentation caused by individual low-confidence tokens within an otherwise continuous hallucinated span. Both thresholds are tuned independently for each language on a held-out portion of the training data.

4.5 Parameter-efficient fine-tuning

Our primary submission extends the frozen-probe system with parameter-efficient fine-tuning. LoRA adapters with rank 16 are applied to the query, key, value, output, gate, up, and down projection matrices. The base model remains frozen and is quantized to 4-bit NF4 with double quantization and bfloat16 computation. The LoRA adapters and classifier head are trained jointly using gradient checkpointing. The main training configuration is given in Table 3.

Practical training constraints. Training was performed on a single T4 GPU with limited session duration. To support interrupted training, we periodically saved the best-performing LoRA adapter and classifier head based on validation performance and resumed subsequent sessions from the latest

Language	IoU	Cor	Cor+Label
English	0.364	0.410	0.308
French	0.422	0.458	0.362
Italian	0.376	0.438	0.353
Chinese	0.40*	0.5116	0.4148

Table 4: Official leaderboard scores for our primary submission. Cor+Label is the organizers’ combined ranking metric, which additionally weighs category-label agreement. *Chinese IoU is an internal estimate; IoU is not reported on the official leaderboard for this language.

checkpoint. This checkpoint-and-resume procedure enabled continued training within the available compute budget.

5 Experimental Setup

We use the official SHROOM-Visions training data across all four languages, holding out a stratified portion as a validation split (so all four languages stay represented) and training on the rest. Our first LoRA checkpoint, which turned into our main submission, used a small validation split of around 8% of the training data. After we saw the overfitting pattern described in Section 6, a later run bumped this up to 20% and also split the validation data into two separate halves: one used only to pick which checkpoint to keep, and one kept aside only for tuning the decoding thresholds at inference time.

All training and inference ran on a single T4 GPU (16 GB) for the 3B backbone, and on two T4 GPUs (with automatic model-parallel placement) for the 7B backbone attempt described in Section F.

6 Results

6.1 Primary submission

Table 4 and Figure 4 show our official leaderboard scores for our best submitted system – a LoRA fine-tuned Qwen2.5-VL-3B checkpoint we call v9 in the version history below.

On the official leaderboard, this puts us 10th out of the participating teams for both French and Italian, and lower for English, which was our weakest language the whole way through development. For Chinese, we placed 9th, with a confirmed Cor of 0.5116 and Cor+Label of 0.4148.

6.2 Qualitative observations across languages

English consistently produced the lowest scores across our model versions. Manual inspection suggests two possible factors: English responses were

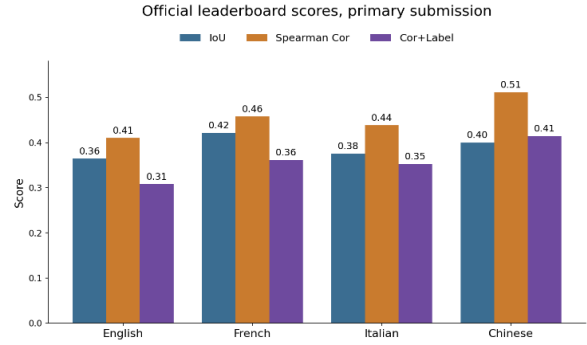


Figure 4: Official leaderboard scores by language for our primary submission.

generally longer, increasing the likelihood of span fragmentation or over-merging, while the backbone’s stronger confidence in English may reduce the discriminative range of grounding-contrast and entropy signals. Chinese showed greater variation in these signals and achieved comparatively stronger performance. These observations are based on limited manual inspection and require systematic validation.

6.3 Category-level error patterns

The system performed best on *invention* and *mis-characterization*, which are the most frequent categories in the training data. *Miscounting* and *OCR errors* were more challenging, likely due to their lower frequency and their reliance on specific visual evidence such as object counts and text recognition. Category-aware sampling or reweighting could improve performance on these underrepresented categories, but we leave this for future work.

7 Conclusion and Future Work

We presented a hallucination span detection system for SHROOM-Visions 2026 based on internal VLM signals, a lightweight classifier, and LoRA-based parameter-efficient fine-tuning. The system achieved competitive performance on French and Italian while providing a baseline across the evaluated languages. Our experiments also revealed a mismatch between validation and official test performance, highlighting the sensitivity of threshold tuning to small validation sets. Future work will focus on validating the revised evaluation procedure, improving grounding-contrast using multiple image crops, addressing underrepresented OCR and miscounting categories through category-aware sampling, and evaluating the frozen-probe approach across different VLM backbones.

8 Limitations

Our validation methodology, while it got better over the course of the shared task, was still limited by the overall size of the official training set for some languages, and by how much time we had to fully re-run the fixed methodology against the official leaderboard. Our system also inherits whatever blind spots the Qwen2.5-VL backbone has: hallucination types the model isn't systematically aware of in its own generations may be harder for it to flag when reading someone else's text, since the internal signals we use are a property of the backbone's own uncertainty, not an independent ground truth.

Finally, we did all our development on a small number of consumer-tier GPUs with limited memory and session length, which capped our batch size, image resolution, how many training epochs we could run, and, in the end, how thoroughly we could validate each design decision within the shared task's timeline.

Ethical Considerations

This system is meant to flag potentially inaccurate content in AI-generated image descriptions, not to generate such content itself. The main risk we see with this kind of work is a false sense of security: a detector scoring what we report in Section 6 will miss a meaningful chunk of real hallucinations, and will sometimes flag correct text as hallucinated too (Section 6.5 goes into the category-level gaps). So it should be treated as a filtering aid, not a guarantee of factual correctness.

We didn't collect any new data involving human subjects for this work – all training and evaluation data comes from the officially released SHROOM-Visions dataset, and we made no attempt to identify or de-anonymize anyone who might appear in the underlying images.

References

- Xuwei Chen, Ziqiao Ma, Xuejun Zhang, Sihan Xu, Jianing Yang, David F Fouhey, Joyce Chai, and Shengyi Qian. 2024. Multi-object hallucination in vision language models. *Advances in Neural Information Processing Systems*, 37:44393–44418.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.
- Anisha Gunjal, Jihan Yin, and Erhan Bas. 2024. Detecting and preventing hallucinations in large vision language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18135–18143.
- Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.
- Fushuo Huo, Wenchao Xu, Zhong Zhang, Haozhao Wang, Zhicheng Chen, and Peilin Zhao. 2025. Self-introspective decoding: Alleviating hallucinations for large vision-language models. In *International Conference on Learning Representations*, volume 2025, pages 24272–24295.
- Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations (ICLR)*.
- Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 292–305.
- Zhangming Li, Qinghao Hu, Yiqun Chen, Peisong Wang, Yifan Zhang, and Jian Cheng. 2025. Lorada: Low-rank direct attention adaptation for efficient llm fine-tuning. In *EMNLP (Findings)*, pages 12638–12655.
- I Dan Melamed. 1997. Measuring semantic entropy. In *Tagging Text With Lexical Semantics: Why, What, and How?*
- Timothee Mickus, Claudio Savelli, Eduardo Calò, Emilio Raimond, Stella Frank, Hengyu Luo, Flavio Giobergia, Vincent Segonne, Chuyuan Li, Aman Sinha, Lorenzo Vaiani, Jörg Tiedemann, and Raúl Vázquez. 2026a. Can humans dream of electric sheep? human-written samples for fine-grained vision-and-language hallucination benchmarking. *Preprint*.
- Timothee Mickus, Claudio Savelli, Eduardo Calò, Emilio Raimond, Stella Frank, Hengyu Luo, Flavio Giobergia, Vincent Segonne, Chuyuan Li, Aman Sinha, Lorenzo Vaiani, Jörg Tiedemann, and Raúl Vázquez. 2026b. Can humans dream of electric sheep? human-written samples for fine-grained vision-and-language hallucination benchmarking. *Preprint*, arXiv:2608.01021.
- Qwen Team. 2025. Qwen2.5-vl technical report. *arXiv preprint*.

Shubhashis Roy Dipta, Tsung-Yi Wu, and Shivang Tripathi. 2025. Vc-inspector: Advancing reference-free evaluation of video captions with factual analysis. *arXiv preprint arXiv:2509.16538*.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20090–20111.

Raúl Vázquez, Aman Sinha, Chuyuan Li, Artem Shelmanov, Artem Vazhentsev, Claudio Savelli, Eduardo Calò, Emilio Raimond, Stella Frank, Hengyu Luo, Flavio Giobergia, Vincent Segonne, Lorenzo Viani, Jörg Tiedemann, and Timothee Mickus. 2026. [Overview of shroom-vision 2026: A shared task on hallucination detection in large vision-language models](#). *Preprint*, arXiv:2608.25662.

Lingling Xu, Haoran Xie, S Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2026. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jingyi Zhang, Jiaying Huang, Sheng Jin, and Shijian Lu. 2024. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644.

A Prompt Template

For every item, we build a single chat turn with the image and the response text to be checked, using the model’s native chat template. The system role stays at its default; the user turn has the image followed by the original prompt text (or a generic “Describe the image.” instruction when the original prompt isn’t available for a given item), and the assistant turn has the response text whose tokens we pull features from. We don’t add any extra instruction asking the model to judge hallucinations directly – our classifier head, not the backbone’s own language modeling head, makes the final call.

B Reproducibility Details

For completeness, here are some extra implementation details not covered in the main text. All experiments used a fixed random seed for data splitting, so the same stratified validation examples were held out across every training run in Section 6 – this keeps the version comparison in Figure 5 fair, rather than confounded by different validation sets.

We trained the classifier head with early stopping based on the validation Cor+Label score from Section 5, checked at fixed intervals rather than after every optimizer step – this cuts evaluation

overhead and smooths out the noise you get from scoring a small held-out split.

We’re releasing our training and inference code, along with the exact prompt templates for every language, at the repository linked in the abstract, so the checkpoint-and-resume training loop from Section 4.5 and the span-decoding procedure in Appendix D can both be reproduced without needing the same compute environment we used.

Setting	3B run	7B run
GPUs	1 × T4	2 × T4
Device placement	single device	automatic split
Gradient accum.	8	16
Val. split size	8%	20%, disjoint

Table 5: Configuration differences between the 3B primary submission and the exploratory 7B run described in Section F.

C Worked Example of Span Decoding

To make Algorithm 1 concrete, here’s a trace through a single toy sentence with $gate = 0.6$ and $tau = 0.35$. Say the model produces a response with the words “The cat ate three fish today,” and the classifier head assigns these per-token hallucination probabilities (the gold error here is a miscounting hallucination, since the image only shows one fish):

Token	p_k	Action	In span?
“The”	0.04	below tau	no
“cat”	0.11	below tau	no
“ate”	0.22	below tau	no
“three”	0.81	start span ($\geq gate$)	yes
“fish”	0.47	extend span ($\geq tau$)	yes
“today”	0.09	below tau, close span	no

Walking through Algorithm 1: the decoder moves past “The,” “cat,” and “ate” without doing anything, since none of them clear tau. At “three” (0.81), the probability clears gate, so a new span opens. At “fish” (0.47), the probability drops below gate but stays above tau, so the span keeps going instead of closing. At “today” (0.09), the probability falls below tau, so the span closes. The resulting span covers the characters in “three fish,” with its label set to the majority category among its two tokens (miscounting) and its confidence set to the mean of 0.81 and 0.47, which is 0.64.

This example also shows why a single threshold would have struggled here: a gate-only decoder at 0.6 would have flagged just “three” and missed “fish” completely, splitting one hallucination into a

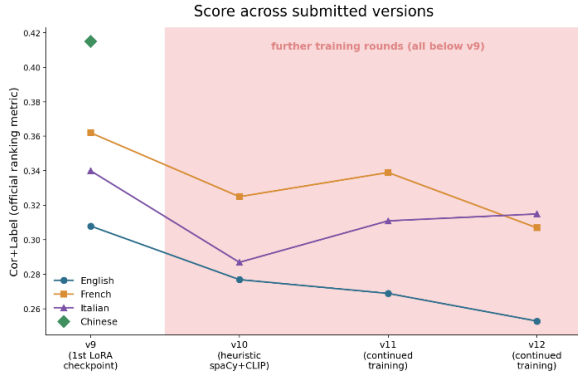


Figure 5: Cor+Label score across our four submitted versions. v9 is our best-performing checkpoint on the official leaderboard; every subsequent attempt scored lower on at least two of the three languages shown.

shorter, incomplete span and lowering IoU against the gold annotation, which covers both words.

D Span-Decoding Procedure

Algorithm 1 lays out the two-threshold decoding procedure shown in Figure 3. It runs once per response and is linear in the number of tokens.

Algorithm 1: two-threshold span decoding

Input: token probabilities $p_1, \dots, p_K$, category predictions $c_1, \dots, c_K$, character offsets for each token, thresholds $gate > \tau$

Output: list of character spans with label and confidence

1. Set $k \leftarrow 1$, $spans \leftarrow \emptyset$.
 2. While $k \leq K$:
 - a. If $p_k \geq gate$: start a new span at token k .
 - i. Extend the span through consecutive tokens while $p_k \geq \tau$.
 - ii. Set the span’s character range to the min start / max end offset of its tokens, its label to the majority category among its tokens, and its confidence to the mean of p_k over its tokens.
 - iii. Append the span to $spans$; advance k past the span.
 - b. Else: advance k by 1.
 3. Return $spans$.
-

Table 6: Pseudocode for the span-decoding procedure used to turn per-token predictions into the final character-level spans submitted to the shared task.

E What did not work: the submission version history

Figure 5 plots the Cor+Label score for every version we submitted, from our first LoRA checkpoint (v9) through three follow-up attempts. We think this figure tells a more useful story than any single number in Table 4.

v9 is the checkpoint described in Section 4, trained with the small 8% validation split. v10 is the training-free alternative we describe in Section G, included here as a point of comparison rather than a step in the same training lineage. v11 and v12 are two more rounds of continued LoRA training, resumed from progressively later checkpoints. Each one clearly improved our own held-out validation score at the time, sometimes by a lot. Once submitted, though, both scored *lower* than v9 on the official test set across most languages.

We think the main cause is the size and construction of our initial validation split. An 8% split, for some languages, is well under 200 examples. On top of that, the decoding thresholds from Section 4.4 were tuned on this same small split at every evaluation step during training. So the validation score we watched during training was actually picking up two overfitting effects stacked on top of each other: the classifier and adapters fitting the validation examples, and then the thresholds fitting the same examples a second time. Neither of these shows up in the training loss curve, which kept decreasing smoothly through every one of our continued-training runs.

F Scaling to a 7B backbone

Since our very first checkpoint scored noticeably higher than any of our continued-training attempts, we tried scaling up to a larger, 7B-parameter version of the same backbone family, on the reasoning that a bigger model might have more head-room before it starts overfitting a small validation split. This attempt taught us two more practical lessons.

First, the larger model didn’t fit into a single T4’s memory during checkpoint loading, even at 4-bit precision, so we had to split it automatically across two GPUs instead of loading it onto one device. It’s an easy detail to overlook when you scale up model size but keep the same hardware tier.

Second, and more importantly: even after we switched to the fixed validation methodology (the 20% split, separated into disjoint model-selection and threshold-tuning halves, described in Section 5), the 7B run still didn’t clearly beat our original v9 checkpoint in the training time we had left before the deadline. A single training epoch at this scale, on our hardware, takes long enough that we only got through a small fraction of one epoch across several resumed sessions. We don’t take this as proof that a bigger backbone can’t help – we

read it more as evidence that, given our time and compute budget, fixing the validation methodology had to come first before scaling up was even worth trying, and that fix used up most of the time we had left.

G A lightweight, training-free alternative

Alongside the LoRA-based system, we also tried a much lighter approach that needs no GPU training at all. We used a syntactic parser to pull out candidate factual claims from the response text, then compared each claim against the image with a general-purpose image-text similarity model, flagging low-similarity claims as likely hallucinations. This approach (v10 in Figure 5) is fast and easy to deploy, but its ceiling was clearly lower than our internal-signal system – it scored below v9 on every language we tested. We’re reporting it mainly to note that, for this task and under our time constraints, reading a model’s own internal signals beat a purely external, similarity-based heuristic, even though the heuristic is a lot simpler to build and needs no labeled training data for the detector itself.