WHAT FEATURES DO VISION-LANGUAGE MODELS ACTUALLY USE?
A LANGUAGE-DRIVEN PROBE OF VISUAL FEATURE IMPORTANCE

ESE 5460: PRINCIPLES OF DEEP LEARNING, EXTRA CREDIT PROJECT

ABSTRACT. This project tries to figure out what visual features LLaVA-1.6 is actually learning to rely on, by testing whether its descriptions hold up under image perturbations that selectively remove parts of the scene. 75 images from MS-COCO are split into three groups (simple, cluttered, and symbolic) and for each one LLaVA is asked to describe what visual features matter most. The image is then perturbed three ways, masking the main object grey, cropping to just the object, and zooming into an irrelevant corner. Consistency between before and after descriptions is measured. Simple images score 0.800, cluttered 0.747, symbolic 0.613. The clearest result shows up in the symbolic group where masking the clock face or sign text barely changes the description (0.88 consistency), but removing the background context causes it to fall apart (0.44). This suggests LLaVA is identifying symbolic objects from scene context and shape rather than reading their actual content. The described features seem to reflect category knowledge more than what's visible in the specific image.

1. INTRODUCTION

The core question is what features a VLM actually learns to use. LLaVA can produce fluent descriptions of what it's "looking at", things like "the brown fur," "the rectangular shape," "the numbers on the clock face," but those descriptions don't necessarily reflect what's driving the output. The model might be relying on background context or learned category priors while generating text that sounds like it came from closely examining the object. Removing things from the image and watching what changes in the output is a direct way to test that.

Professor Chaudhari suggested using LLaVA as both the subject and the evaluator: ask it to describe features, change the image, then ask whether its descriptions are consistent. The model is grading itself, which is a real limitation, but there's no obvious external ground truth for what the "correct" feature description should be. What ended up being just as important as the perturbation design was getting the prompts right. Small changes to how the feature elicitation question was worded, or how the consistency evaluator was asked to respond, had a noticeable effect on the numbers. Prompt engineering turned out to be a real part of the experiment, not just setup.

1.1. Contributions

(1) A perturbation-based pipeline that tests three image editing strategies and measures whether LLaVA's feature descriptions hold up across each one.

(2) A record of prompt iteration across both the feature elicitation and consistency evaluation steps, including a structured format that was tried and reverted after it made results worse.

(3) Evidence that symbolic objects (clocks, TVs, signs) are probably identified from scene context and shape rather than their actual symbolic content, since masking them barely changes descriptions but removing background context does.

2. BACKGROUND

LLaVA-1.6 is a vision-language model built by combining a CLIP vision encoder with a Mistral-7B language backbone. The version used here is llava-hf/llava-v1.6-mistral-7b-hf. How the two components fit together is worth understanding because it affect what the perturbation results actually mean.

On the vision side, CLIP ViT-L/14 splits the input image into 14x14 pixel patches and embeds each one as a vector via a transformer encoder. These aren't raw pixel values; they're learned representations that carry texture, shape, and context information. CLIP was pretrained contrastively on image-text pairs so the embeddings already have semantic content. Those patch vectors then go through a two-layer MLP that maps them into Mistral's embedding space. This projection layer is the main thing LLaVA actually trains during instruction tuning. After that, image tokens and text tokens get concatenated into one flat sequence and fed

into Mistral-7B for autoregressive generation. There's no separate cross-attention between the visual and language parts. It's all processed together.

This matters for the experiment because there's no clean way to verify which patch tokens actually drove a given output. With a CNN you can run GradCAM and get a spatial heatmap. With a decoder-only transformer processing a flat mix of image and text tokens, attribution is genuinely hard. The generated description might reflect what the model actually saw, or it might just reflect what the model learned to expect objects of that type to look like. That's the ambiguity this experiment is trying to get at.

The other relevant detail is what masking actually does at the token level. Replacing pixels with grey doesn't remove those patch tokens from the sequence. The masked region still produces vectors, they just encode a uniform grey patch instead of the original content. The model still sees where in the frame the grey region sits, what shape it is, and all the surrounding context. This is why grey-masking the main object is a weaker intervention than it sounds. The model can still infer a lot from the residual silhouette and background.

Everything ran in 4-bit NF4 quantization to fit on a T4, with greedy decoding to keep outputs deterministic across runs.

3. RELATED WORK

The approach here is closest in spirit to LIME (Ribeiro et al., 2016), which explains model predictions by generating perturbed versions of the input and seeing how the output changes. For images, LIME masks out superpixel regions at random and fits a local linear model to figure out which ones the classifier actually relied on. This project does something structurally similar (perturb the image, measure the change in output) but the perturbations are semantically chosen rather than arbitrary, and the output being measured is a natural-language description rather than a class probability. That difference matters. LIME tells you which pixels moved a number. This project asks whether what the model says it cares about matches what it actually responds to.

The hallucination literature is more directly relevant. Li et al. (2023) introduced POPE to test VLM hallucination by asking binary yes/no questions about object presence. The key finding was that models tend to hallucinate objects that statistically co-occur with things that are actually in the image. A dining table prompts claims about nearby food even when there isn't any. That co-occurrence bias is probably part of what's going on with the symbolic results in this project. LLaVA describing clock hands that have been greyed out isn't necessarily reading the image; it might just know that clocks have hands. POPE asks a binary question with a verifiable answer. What this project asks is harder to pin down, consistency of natural language descriptions doesn't have a clean ground truth.

Jing et al. (2023) proposed FAITHSCORE specifically to evaluate whether VLM descriptions are grounded in what's actually in the image. The method decomposes a caption into individual atomic claims and verifies each one against the image using a separate model. It's more rigorous: atomic decomposition, external verifier, no self-evaluation problem. The question it's asking is essentially the same one this project is asking.

BERTScore (Zhang et al., 2020) compares two text outputs by embedding them and computing cosine similarity at the token level, using contextual representations from a pretrained language model. It's a better consistency metric than asking LLaVA to evaluate its own outputs, because it removes the model from the loop entirely. Using LLaVA as the evaluator means the same model that generated the descriptions is also judging whether they changed. That's circular, and there's no way to fully audit it. BERTScore isn't perfect either, but at least it doesn't have that problem. Swapping it in is the most obvious extension of this work.

4. APPROACH

4.1. Dataset

75 images from MS-COCO val2017, 25 per group. Simple: one annotated object. Cluttered: multiple. Symbolic: contains at least one of stop sign, parking meter, clock, or TV. The original intent was to have a dedicated signage group (street signs, stop signs, text-bearing objects) to specifically test whether LLaVA was

reading the text content or just recognizing the object shape. MS-COCO doesn't have enough pure signage images to fill 25 slots cleanly, so the category got broadened to symbolic, pulling in clocks and TVs as objects that also have readable content. The core test is same: if the model is actually using what's written or displayed on these objects, masking that content should change the description.

One problem worth naming: the symbolic group was assigned by checking whether any symbolic category appeared in annotations, not whether it was the main thing in the frame. A few images ended up being mostly about a dog or person who happened to be near a TV in the background. That noise probably pulled some symbolic scores toward cluttered. Ideally you'd filter to images where the symbolic object is the dominant annotated element, but doing that properly would have needed manual review.

4.2. Feature Prompt

Two versions of the feature prompt were tried. The first was structured, asking LLaVA to output a fixed format:

> *"First state the main object in this image in the format 'Object: <n>'. Then on a new line, list the most important visual features in the format 'Features: <feature1>, <feature2>, <feature3>'. Be specific: include colour, shape, texture, and location where relevant. List at most 5 features."*

The idea was that comma-separated feature lists would be easier to compare and that label extraction would be trivial. It turned out to make things worse. Consistency scores dropped across all groups, probably because keyword lists are brittle to compare. Something like 'white, rectangular, smooth' versus 'white, boxy, matte surface' looks like disagreement even when they mean the same thing. Free-form prose gives the evaluator more room to match meaning across paraphrases, so the structured prompt got reverted:

> *"What is the main object? What features or regions of this image are most important for identifying the main object? Be specific about which visual elements (colour, shape, texture, location) matter most."*

Pulling the object label out of free-form output uses a trigger-phrase heuristic: scan for phrases like 'main object is' or 'shows a' and grab the next three words. Falls back to the first eight words when nothing matches. Not perfect, but it works well enough that the labels fed into the consistency evaluator are usually sensible, and the two-step consistency prompt catches obvious object mismatches anyway.

4.3. Perturbations

Three versions were generated per image from the COCO segmentation mask for the largest annotated object:

- **mask_important:** Replaces the object with 128-grey, leaving the background intact. 128 is neutral mid-grey to avoid introducing any color signal. Falls back to greying the central 50% if no mask is available.
- **crop_object:** Crops to the bounding box of the object with 10% padding and upscales to fill the full frame. BILINEAR interpolation was used because NEAREST produced visible pixel artifacts on tight crops.
- **zoom_irrelevant:** Zooms into whichever corner is farthest from the object centroid. The goal is to show the model a region it should have no reason to focus on.

4.4. Consistency Scoring

LLaVA was run on the original and all three perturbed versions of each image. The original and perturbed descriptions were then compared using a two-step prompt: first check whether the identified objects are the same (if the mask caused the model to switch from dog to cow, that's already a NO), then check whether the same visual features are being emphasized. Single-word YES or NO output.

An earlier version of the consistency prompt asked for a one-sentence explanation after the verdict. That caused parsing failures when the explanation got generated before the YES/NO word. Dropping the explanation and requiring a single word fixed it. A simpler prompt before that (just 'do these describe the same

features') was returning UNCLEAR too often and missing object-level switches. The two-step version solved both.

YES = 1.0, NO = 0.0, UNCLEAR = 0.5. Circularity is the obvious problem: LLaVA is scoring LLaVA. Swapping in BERTScore would be the most straightforward fix.

5. EXPERIMENTAL RESULTS

5.1. Overall Results

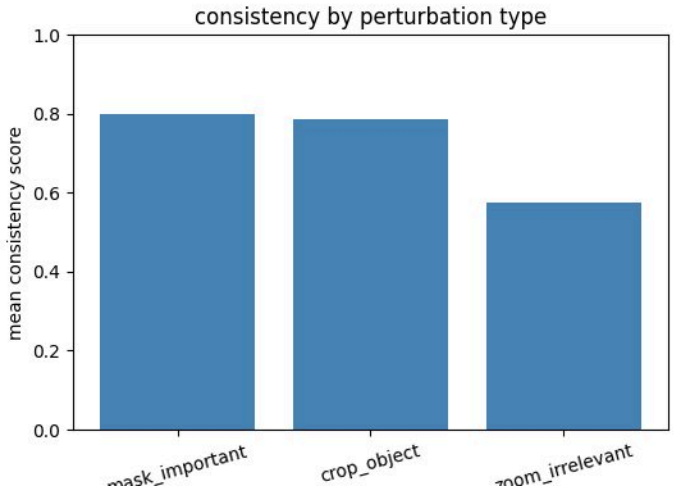
The aggregate numbers in Table 1 are fine, but the per-condition breakdown in Table 2 is where things get interesting.

TABLE 1. Aggregate consistency scores by group.

Group	Images	Mean Score	% YES	% NO	UNCLEAR
simple	25	0.800	80.0%	20.0%	0%
cluttered	25	0.747	74.7%	25.3%	0%
symbolic	25	0.613	61.3%	38.7%	0%

TABLE 2. Mean consistency score by group and perturbation condition.

Condition	simple	cluttered	symbolic
mask_important	0.80	0.72	0.88
crop_object	1.00	0.92	0.44
zoom_irrelevant	0.60	0.60	0.52



5.2. Key Findings

crop_object for simple hitting 1.00 is about as clean a result as you can get. Every single simple image produced the same description when cropped to just the object. That makes sense: for a single-object scene, the object basically is the image. mask_important for simple is 0.80, which also makes sense for the same reason. There's enough background for the model to reconstruct what the object was even when the object itself is greyed out.

The symbolic group is the main thing worth looking at. The aggregate (0.613) is the lowest of the three, but what's actually going on is a split between conditions. mask_important for symbolic is 0.88, the highest single number in all of Table 2, including higher than simple. Grey out the clock or the stop sign and LLaVA

barely notices. crop_object for symbolic is 0.44, the lowest number in the table. So within the same group of images, one perturbation scores near-perfect and the other is basically a coin flip. That 0.44-point gap is what the whole project is pointing at.

zoom_irrelevant came in at 0.60 for both simple and cluttered, 0.52 for symbolic. The simple number was more surprising to me then the symbolic one. I assumed zooming into a random corner of a clean single-object image would barely affect anything, but it dropped meaningfully from the 1.00 crop baseline. The model seems to get confused by unusual framing regardless of how simple the scene is. For symbolic, the zoom result (0.52) being higher than crop (0.44) makes sense if the irrelevant corner occasionally happens to contain background context that still anchors the object type.

6. DISCUSSION

The symbolic gap is hard to explain if the model is actually reading the clock face or the stop sign text. If it were, masking those out should hurt consistency more then removing the background does. What the results suggest instead is that LLaVA is using scene context and the residual shape of the masked region to figure out the object type, and then generating a description of what that category typically looks like from memory. A clock stays roughly circular even when its face is greyed out. A TV stays rectangular. That silhouette plus the surrounding room is apparently enough to identify it, and from there the described features (things like 'clock face with numbers,' 'rectangular screen') seem to be confabulated from category knowledge rather than actually read from the image.

This connects to what Li et al. found with POPE: models report co-occurring attributes based on statistical patterns from training, regardless of whether those things are actually present. The symbolic result here might be the same effect applied to feature descriptions rather than object existence.

The symbolic classification is the biggest methodological issue. Any image with a clock or TV in the annotations gets classified as symbolic, even if the clock is a tiny thing in the background of an image that's clearly about something else. I ran into this during the experiment and didn't have time to fix it. The noisier the symbolic group is, the harder it is to interpret the symbolic numbers specifically. Filtering to images where the symbolic object is the largest-area annotated element would be a real improvement.

The other main issue is the evaluator. Having LLaVA grade its own consistency is circular, and there's no easy way to audit whether it's applying the same standard across conditions. FAITHSCORE (Jing et al., 2023) handles this more carefully by decomposing descriptions into individual claims and checking each one; BERTScore (Zhang et al., 2020) would at least remove the model from the loop entirely. Both are more principled than what's done here.

The blender (image 284282) is the clearest individual example of what the experiment is picking up on. The masked version (blender completely replaced with grey) still produced a description mentioning a 'cylindrical shape with narrow base,' a 'predominantly white' color, and a 'clear pitcher on top.' The clear pitcher part is the giveaway: that's a detail that was greyed out, but the model described it anyway because that's what blenders have. At 25 images per group the numbers are noisy, but cases like this make it feel like the experiment is catching something real.

REFERENCES

Lin, T.Y. et al. (2014). Microsoft COCO: Common Objects in Context. ECCV. https://arxiv.org/abs/1405.0312

Wolf, T. et al. (2020). HuggingFace Transformers. EMNLP. https://arxiv.org/abs/1910.03771

Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., & Artzi, Y. (2020). BERTScore: Evaluating Text Generation with BERT. ICLR. https://arxiv.org/abs/1904.09675

Liu, H. et al. (2024). LLaVA-NeXT: Improved reasoning, OCR, and world knowledge. HuggingFace: llava-hf/llava-v1.6-mistral-7b-hf. https://huggingface.co/llava-hf/llava-v1.6-mistral-7b-hf

Ribeiro, M.T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. KDD. https://arxiv.org/abs/1602.04938

Li, Y., Du, Y., Zhou, K., Wang, J., Zhao, W.X., & Wen, J.R. (2023). Evaluating Object Hallucination in Large Vision-Language Models. EMNLP. https://arxiv.org/abs/2305.10355

Jing, L., Li, R., Chen, Y., & Du, X. (2023). FAITHSCORE: Fine-grained Evaluations of Hallucinations in Large Vision-Language Models. EMNLP Findings. https://arxiv.org/abs/2311.01477