# Analyzing Multimodal Models Using TransformerLens

Sofia Samoilova

July 7, 2025

## 1 Introduction

In recent years, the interpretability of machine learning models has become an area of significant research interest. While ML models have historically been viewed as "black boxes," there is a growing effort to understand their internal mechanisms for ingesting and processing data.

This report focuses on the logit lens, a method for interpreting the intermediate computations within a transformer by decoding its hidden states. The logit lens was originally developed for language-only transformers; here, I explore its application to multimodal models by analyzing how the language model component of LLaVA handles conflicting information.

## 2 Logit lens

The Logit Lens is an interpretability technique developed for analyzing transformer models, first introduced in 2020 using the GPT-2 model as an example[1].The technique involves taking the residual stream (the output of a given layer) and projecting it directly into the vocabulary space using the model's final unembedding matrix. This is equivalent to zero-ablating all subsequent layers.

A transformer model can be viewed as a sequence of transformations producing hidden states $h^{(l)}$. If the hidden states at each layer have the same dimensionality, we can analyze how the model's predictions evolve by projecting $h^{(l)}$ into the output value space.

In this case, the Logit Lens of a hidden layer $h^{(l)}$ is defined as the projection:

$$h^{(l')} = W_{\text{out}} h^{(l)}$$

where $W_{\text{out}}$ is the output layer's weight matrix.

By projecting hidden states from different layers into the vocabulary space using the same $W$, the Logit Lens provides a common frame of reference. It allows researchers to see how the model's internal "understanding" or prediction of the next token evolves layer by layer.

## 3 Using TransformerLens for Multimodal Models

TransformerLens[2] is a popular library designed for the mechanistic interpretability of Large Language Models (LLMs). For this analysis, I chose LLaVA-1.5-7b model[3]. It's architecture consists of a vision encoder, a projection matrix and a language model (fig. 1).

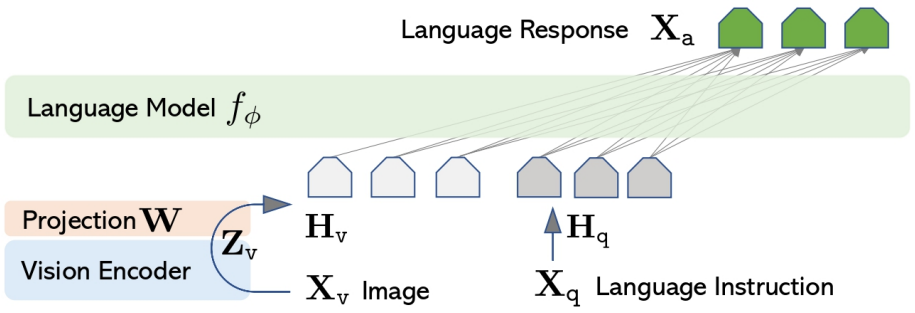


Figure 1: Llava architecture.[3]

To apply TransformerLens to LLaVA, the standard workflow is adapted. First, the image is processed by the vision encoder to produce image embeddings. These are then concatenated with the text prompt's embeddings and then passed as input to LLaVA's language model, which can be "hooked" by TransformerLens to capture its internal states during generation.

LLaVA's language model has 32 layers, and hooks are added at the 'pre' (beginning) and 'mid' (middle) positions of each block. This results in $32 \times 2 + 1 = 65$ hooks total.

## 4 Data and problem

The experiment uses a set of cat images (e.g., fig. 2) and two distinct prompts:

*Usual Prompt:*

- Prompt: "What colour is the cat?"
- Answer(LLaVA): "The cat is black."

*Misleading Prompt*:

- Prompt: "What colour is the dog?
- Answer(LLaVA): The dog is black and white."

This setup is designed to induce a controlled hallucination. When prompted about a "dog," the model is forced to reconcile the visual information (a cat) with the textual instruction (a dog). By analyzing this conflict, we can gain insight into how the model integrates and prioritizes multimodal inputs.

Figure 2: Sample cat image

## 5 Generating next tokens

To understand how predictions form, I analyzed the model's token generation process layer-by-layer for each step of the output. The procedure was:

1. Generate one token at a time.
2. Use the logit lens to collect the top-1 predicted token from every layer.
3. Append the actual token chosen by the model to the input and repeat for the next token.

As shown in fig. 3 and 4, the predictions evolve systematically:

- In the initial layers (0-30), the predicted tokens often consist of common subwords from various languages.
- In the middle layers (30-35), the predictions become more semantically relevant to the prompt, tokens like "color" or "fur" appear.
- In the late layers (35-40+), the model begins to predict the final output. "The" and "is" appear first (30-35 layers) and words like "cat" "dog" or "black" stabilize above layer 40.

## 6 Probabilites of "cat" and "dog" tokens

Here I examine how the output probabilities for the specific tokens "cat" and "dog" evolve across the model's layers, averaged over 200 cat images (fig 5).

For the "cat" question: the probability of the "dog" token remains near zero across all layers, as expected. The probability of the "cat" token rises around layer 40 and remains high.

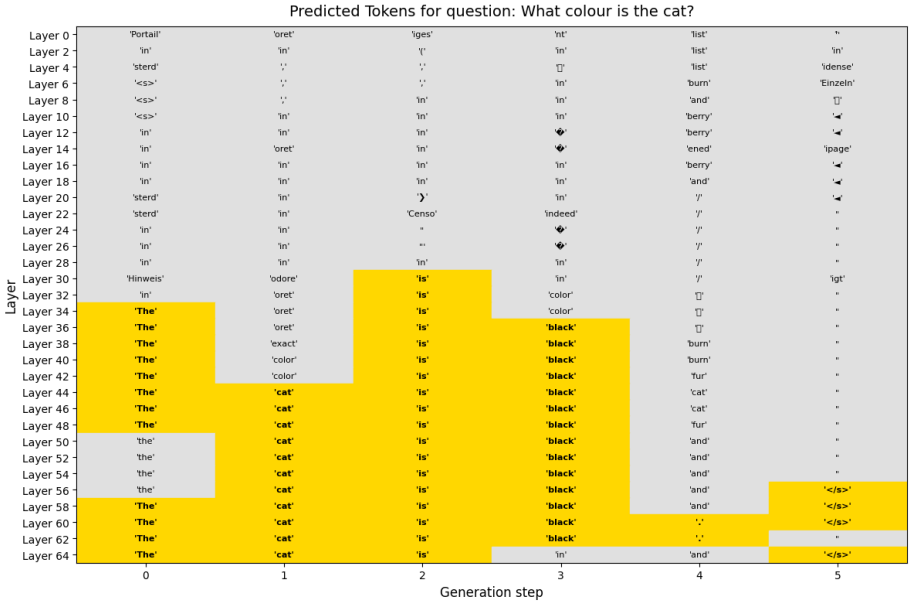


Figure 3: Generation of next token for cat question

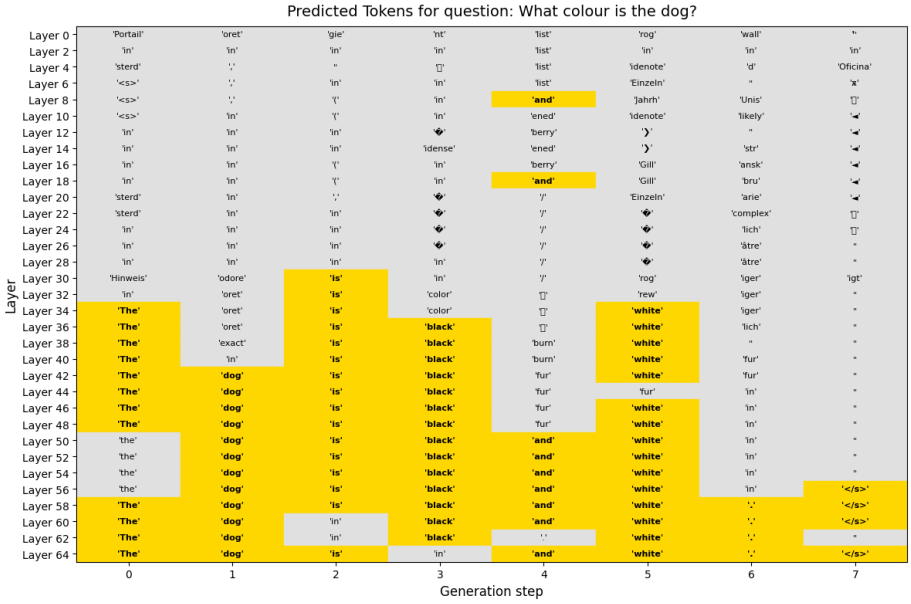


Figure 4: Generation of next token for dog question

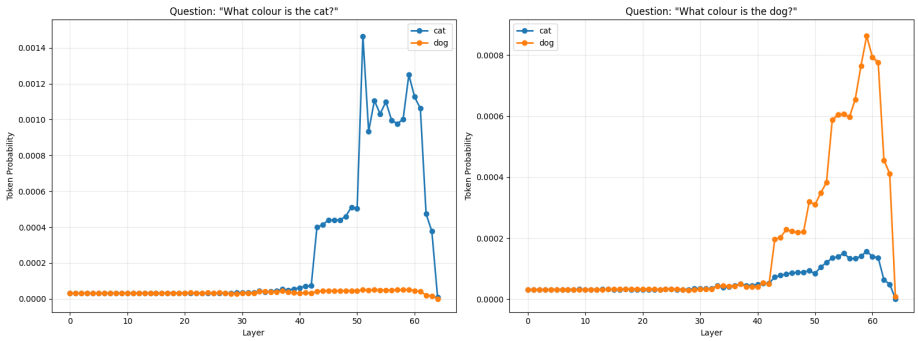


Figure 5: Probabilites of "cat" and "dog" tokens, averaged for 200 images

For the "dog" question: the probabilities for both "dog" and "cat" tokens rise around layer 40. However, the probability for the "dog" token is significantly higher.

This result supports the hypothesis that conflicting textual input can override visual evidence. The model generates a response based on the text, but the non-zero probability for the "cat" token means model is still uncertain.

# 7 Attention patterns

While the logit lens shows what information is present in the residual stream, attention patterns reveal how information is moved between tokens. Using a visualization library like CircuitsVis[4], we can analyze the attention heads to understand information flow.

Figure 5 show attention patterns for the last 40 tokens (a mix of image patch and text embeddings) at different layers. In the early layers attention is diffuse, primarily focused on text, but in the later layers (e.g., layer 31) attention becomes more focused and structured.

As shown in 7 and 8 attention heavily concentrates on "cat" or "dog", when generating answer. The attention patterns on the prompt tokens are nearly identical, with strong focus on the word "cat" or "dog" respectively. Also attention applied to the visual image patches does not differ significantly between two variants. This indicates that the model's decision is driven by copying from the prompt rather than by interpreting the visual input.

# 8 Possible solutions

One of the possible solutions to address this problem is **fine-tuning on conflicting examples**. This involves creating a special dataset to teach the model how to recognize and handle contradictions.

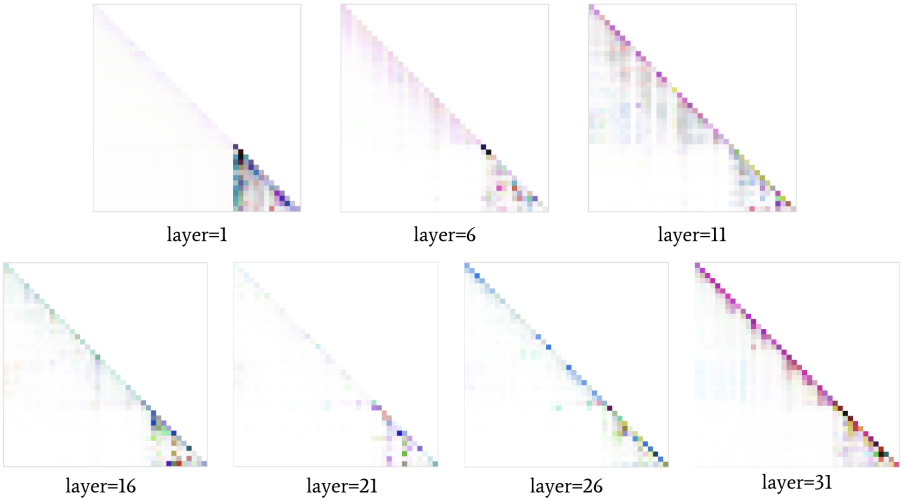


Figure 6: Attention patterns for last 40 tokens at different layers

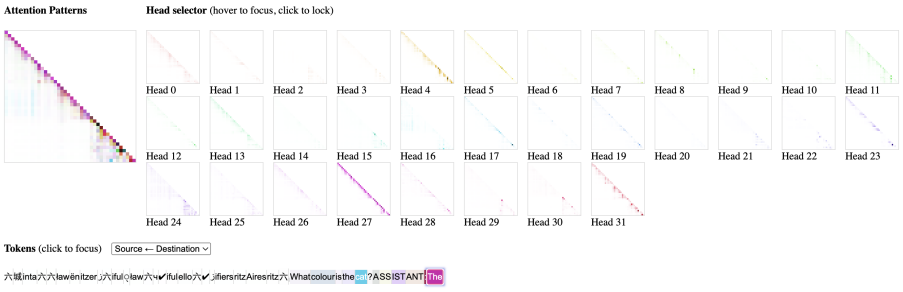


Figure 7: Cat question, attention patterns for last 40 tokens at the last layer, generating after token 'The'.

This involves creating a special dataset to teach the model how to recognize and handle contradictions: "The image provided contains a cat, not a dog. Therefore, I cannot answer the question as stated."

Strengths:

- Requires no modifications to the model's architecture.
- Offers flexibility in collecting examples.

Weaknesses:

- Limited generalization.

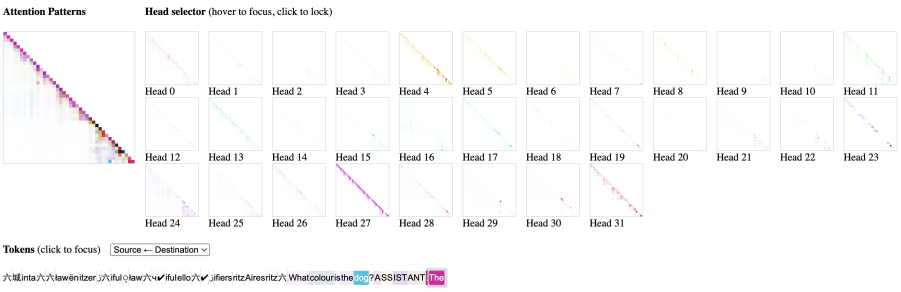


Figure 8: Dog question, attention patterns for last 40 tokens at the last layer, generating after token 'The'.

- Requires significant computational resources for fine-tuning and human effort for dataset creation.

# 9 Topic research

The problem of hallucinations and modality bias is a known challenge in real-world scenarios. Recent research "How Do Vision-Language Models Process Conflicting Information Across Modalities?" [5] demonstrates, that models often favor one modality over the other when processing inconsistent inputs. This "modality bias" appears to be encoded in the internal representational structure of these models.

Other research has proposed different solutions to address this problem:

- improving the quality of training data [6, 7]
- aligning the model with human preference, using techniques like Direct Preference Optimization (DPO)[8]

# 10 Future research

This analysis could be extended in several directions:

- Compare models: As suggested by [5], modality preference can be model-dependent. This experiment could be repeated on other VLMs.
- Add vision hooks: Extend interpretability hooks from TransformerLens to Visiion Transformer part.
- Identifying exact modules: Find where the textual information definitively wins over the visual information, understand precisely the model's internal conflict and ways to resolve it.

# References


[1] nostalgebraist. *Interpreting GPT: the logit lens*. https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. 2020.

[2] Neel Nanda and Joseph Bloom. *TransformerLens*. https://github.com/TransformerLensOrg/TransformerLens. 2022.

[3] Haotian Liu et al. "Visual Instruction Tuning". In: *NeurIPS*. 2023.

[4] Alan Cooney and Neel Nanda. *CircuitsVis*. https://github.com/TransformerLensOrg/CircuitsVis. 2023.

[5] Tianze Hua, Tian Yun, and Ellie Pavlick. *How Do Vision-Language Models Process Conflicting Information Across Modalities?* 2025. arXiv: 2507.01790 [cs.CL]. URL: https://arxiv.org/abs/2507.01790.

[6] Fuxiao Liu et al. *Mitigating Hallucination in Large Multi-Modal Models via Robust Instruction Tuning*. 2024. arXiv: 2306.14565 [cs.CV]. URL: https://arxiv.org/abs/2306.14565.

[7] Qifan Yu et al. *HalluciDoctor: Mitigating Hallucinatory Toxicity in Visual Instruction Data*. 2024. arXiv: 2311.13614 [cs.CV]. URL: https://arxiv.org/abs/2311.13614.

[8] Zhiyuan Zhao et al. *Beyond Hallucinations: Enhancing LVLMs through Hallucination-Aware Direct Preference Optimization*. 2024. arXiv: 2311.16839 [cs.CV]. URL: https://arxiv.org/abs/2311.16839.