

## A Definition of MLP Neurons

Our definition of MLP neurons as circuit components follows existing literature (Nikankin et al., 2024), and is provided for completeness. In the VLMs we analyze, the MLP layer of a transformer block is implemented by a Gated MLP (Liu et al., 2021). This MLP at layer  $l$  can be described by the following equations:

$$\mathbf{h}_{post}^l = \sigma(\mathbf{h}_{in}^l \mathbf{W}_{gate}^{l\top}) \circ (\mathbf{h}_{in}^l \mathbf{W}_{in}^{l\top}) \quad (6)$$

$$\mathbf{h}_{out}^l = \mathbf{h}_{post}^l \mathbf{W}_{out}^l \quad (7)$$

where  $\mathbf{h}_{in}^l, \mathbf{h}_{out}^l \in \mathbb{R}^d$ ,  $\mathbf{h}_{post}^l \in \mathbb{R}^{d_{mlp}}$  are the MLP input, MLP output and post-non-linearity representations of the MLP block, respectively.  $\mathbf{W}_{in}^l, \mathbf{W}_{out}^l \in \mathbb{R}^{d_{mlp} \times d}$ ,  $\mathbf{W}_{gate}^l \in \mathbb{R}^{d_{mlp} \times d}$  are weight matrices, and  $\sigma$  is a non-linearity function,  $\circ$  is Hadamard product, and biases are omitted.





In this formulation, we define the activation of the  $n^{th}$  MLP neuron in layer  $l$  as the  $n^{th}$  scalar value in the intermediate representation  $\mathbf{h}_{post}^l$ . This scalar is multiplied with the  $n^{th}$  row of  $\mathbf{W}_{out}^l$ , to produce the neuron’s contribution to the output of the entire MLP block.

## B Experimental and Technical Details

### B.1 Prompt examples

Table 2 presents prompt examples for each task in both input modalities.

Table 2: Textual and Visual Prompt Examples per task

Task Name	Textual Prompt Example	Visual Prompt Example
Object Counting	“Sequence: book tree book cup cup ball tree. How many "tree" are in the sequence? Answer in a single number”	 “How many "tree" are in the image? Answer in a single number.”
Arithmetic	“Question: 10*48. What is the result of the given arithmetic calculation? Answer in a single number”	10 × 48 = “What is the result of the given arithmetic calculation? Answer in a single number.”
Spatial Ordering	“In a scene with four objects arranged horizontally, there is a green object, a yellow object, a yellow object and a cyan object. What is the color of the third object from the left? Answer in a single word.”	 “What is the color of the third object from the left? Answer in a single word.”
Factual Recall	“Consider Dennis Rodman. What sport does this athlete play? Answer in a single word.”	 “What sport does this athlete play? Answer in a single word.”
Sentiment Analysis	“ “A child runs through a field of daisies, laughter echoing in the sunshine, feeling the ...”. Is this scene happy, sad, or neutral? Answer in a single word.”	 “Is this scene happy, sad, or neutral? Answer in a single word.”

The amount of prompts per task is shown in Table 3. The amount of prompts for the same task and modality varies between models due to tokenization considerations (e.g. all object colors in the spatial ordering task must be tokenized to a single token for positional alignment purposes—prompts that contain multi-token object colors are filtered out).

Table 3: **Prompt amounts for each model, task and modality.**

Task	Qwen2-7B-VL		Pixtral-12B		Gemma-3-12B	
	L	V	L	V	L	V
<b>Counting</b>	1524	383	1524	334	1524	383
<b>Arithmetic</b>	1000	1000	1000	1000	1000	1000
<b>Spatial Ordering</b>	1865	1925	472	497	1865	1925
<b>Factual Recall</b>	416	1265	454	1265	453	1265
<b>Sentiment Analysis</b>	232	245	237	245	222	245

The full list of prompts will be published.

## B.2 Tasks Generation Details

In this section we describe in detail the prompt generation process for each task and modality in our dataset.

**Object Counting:** We use a base list of 30 possible object types (e.g. “banana”, “fork”, “book”). Each textual prompt contains seven randomly sampled objects drawn from up to four different object types—a configuration chosen to balance task difficulty while maintaining high VLM accuracy. For the visual prompts, we use SD3-XL (Podell et al., 2023) to generate images containing the same amount of objects described in the text. Due to limitations in image generation models’ ability to produce exact object counts, we manually verify each image, and filter out images that don’t match the required object counts.

**Arithmetic:** For the textual variant, each prompt consists of two two-digit operands, ensuring positional alignment between different prompts (e.g., the second position always includes the singles digit of the first operand). This consistency is important because all analyzed models tokenize numbers into separate digits. For the visual variant, we create white-background images (75×338 resolution) displaying the identical arithmetic calculations in large black font centered in the image, maintaining a simple presentation that focuses entirely on the calculation itself.

**Spatial Ordering:** For the visual variant, the images are taken from the CLEVR dataset (Johnson et al., 2017). Each image depicts a scene with four colored objects. We select this specific object count as it provides an optimal difficulty tradeoff—scenes with three objects prove too easy for models, while scenes with five objects result in significantly lower model accuracy. The textual variant is generated out of these scenes. Each textual prompt depicts a CLEVR scene with four colored objects, mentioning only their colors without shapes, to maintain positional alignment (as different shapes (e.g. “sphere”) get tokenized into varying numbers of tokens).

**Sentiment Analysis:** We use ChatGPT (Hurst et al., 2024) to generate 120 scene descriptions for each sentiment—happy, sad, or neutral, and manually drop scene descriptions that don’t convey the target sentiment. To maintain positional alignment in the textual variant, we truncate each description to exactly 20 tokens and pad the description with “...”. Scene descriptions shorter than 20 tokens are filtered out. We use a scene token length of 20 since most scenes are longer than it, and we found it to convey the sentiment of the scene well enough in all cases. For the corresponding visual prompts, we use Flux1-Schnell (Labs, 2024) to generate images based on these scene descriptions, conducting additional manual filtering to exclude any images that don’t adequately match the described scene or fail to convey the intended sentiment.

**Factual Recall:** We use three prompt templates from Hernandez et al. (2023):

1. "Consider [X]. What sport does this athlete play? Answer in a single word."

- 856 2. "Consider [X]. Which country is this landmark in? Answer in a single word."  
857 3. "Consider [X]. What instrument does this person play? Answer in a single word."

858 For the textual variant, we select entities (to replace the [X] in the prompt, e.g. "Michael Jordan")  
859 that, when combined with the prefix "Consider [X]", result in exactly 5 tokens to ensure positional  
860 alignment. We use this number as it results in the highest number of possible entities in the given  
861 dataset. For the visual variant, we collect entity images with public licenses from Wikimedia  
862 Commons. We rank 10 potential images per entity using CLIP (Radford et al., 2021) scores and  
863 select the best match for each entity. We resize and center-crop each image (to 256x256 resolution).  
864 We manually review the images to remove any that contain "shortcut" hints (e.g., any image of a  
865 basketball player holding a basketball), to ensure models rely on factual recall rather than visual cues.  
866 In both variants, the counterfactual prompts for each template are sampled only from within the same  
867 template.

### 868 B.3 Circuit Discovery and Evaluation

869 For our circuit discovery experiments (detailed in Section 2 and Section 4), we allocate each modality-  
870 specific task dataset with a 75/25 split: 75% of the prompts for discovery and 25% of the prompts for  
871 faithfulness evaluation. We divide prompts between the discovery and evaluation subsets such that all  
872 possible answers are distributed in the same ratio between the subsets.

873 When answers span multiple tokens (e.g. in the arithmetic task, where answers can be comprised of  
874 multiple digits), we measure the patching effect (Equation (1)) solely on the first token. We verify  
875 that the answers for each prompt and counterfactual prompt ( $r$  and  $r'$ ) differ in the analyzed first  
876 token.

877 Our circuit discovery experiments use our fork of the TransformerLens library (Nanda & Bloom,  
878 2022), in which we implement patching code for VLMs. This fork is available in our code release.

### 879 B.4 Compute Resources

880 Our experiments were conducted on an NVIDIA L40 node equipped with 8 GPUs, each containing  
881 48GB of memory. Peak memory consumption occurred during circuit discovery operations on the  
882 Gemma-3-12B-Instruct model, that required the parallel use of 4 GPUs. The complete experimental  
883 suite, including studies not featured in the final paper, consumed roughly 200–300 GPU hours.

## 884 C Task Accuracies

885 In Table 4 we report the accuracies of models on each of the tasks. We observe that across most tasks,  
886 VLMs achieve higher accuracy in the textual variant than the visual variant, motivating our analysis  
887 into this performance gap. This is compatible with earlier results on closed-source VLMs (Fu et al.,  
888 2024).

Table 4: Model Accuracies for textual and visual task variants.

Task	Qwen2-7B-VL		Pixtral-12B		Gemma-3-12B	
	L	V	L	V	L	V
Counting	79.3%	70.9%	49.3%	66.1%	89.3%	73.4%
Arithmetic	99.2%	67.6%	25.7%	22.7%	99.0%	87.5%
Color Ordering	86.8%	74.2%	77.4%	80.6%	76.1%	46.0%
Factual Recall	73.5%	68.1%	83.4%	45.0%	80.7%	66.5%
Sentiment Analysis	97.4%	92.6%	98.3%	69.7%	98.7%	92.6%

## 889 D Additional Circuit Discovery Results

890 In Table 5 we report the circuit size, in a percentage of the model’s component count. As described in  
891 Section 4, the circuits are chosen to be the minimum-sized circuit with a faithfulness of over 80%.  
892 The faithfulness achieved by each circuit on each task and modality is reported in Table 6.

Table 5: Circuit sizes, in percentage of components out of the model’s whole component set.

Task	Qwen2-7B-VL		Pixtral-12B		Gemma-3-12B	
	L	V	L	V	L	V
Counting	5%	8%	7%	20%	5%	20%
Arithmetic	1%	3%	6%	7%	2%	5%
Color Ordering	2%	8%	5%	20%	2%	7%
Factual Recall	2%	8%	5%	20%	5%	20%
Sentiment Analysis	2%	10%	20%	20%	10%	20%

Table 6: Circuit faithfulness scores for textual and visual task variants.

Task	Qwen2-7B-VL		Pixtral-12B		Gemma-3-12B	
	L	V	L	V	L	V
Counting	82.9%	82.0%	81.4%	82.7%	88.0%	89.7%
Arithmetic	86.0%	81.9%	82.6%	80.3%	80.9%	83.9%
Color Ordering	83.6%	81.2%	82.4%	84.1%	82.8%	81.3%
Factual Recall	82.6%	82.7%	80.7%	94.6%	82.3%	92.8%
Sentiment Analysis	88.2%	80.4%	92.4%	80.7%	85.3%	82.1%

Furthermore, in Figure 10, Figure 11, Figure 12, Figure 13, Figure 14 we present the patching importance scores for each model and task, summed per position and layer. These figures are complementary to Figure 3.

## E Back-Patching Ablations and Extended Results

### E.1 Iterative Back-patching

Extending the back-patching experiments in Section 5, we explore if additional processing of visual data tokens yields further improvements. To do that, we perform back-patching iteratively (back-patching more than once). After identifying the optimal layers  $l_{src}$ ,  $l_{dst}$  and layer window size for each model and task, we perform back-patching iteratively. Namely, after each patching of the window centered at  $l_{dst}$ , we continue the forward pass to calculate the window centered at  $l_{src}$ , and re-patch the window centered at  $l_{dst}$ . This process is repeated up to 10 times. As shown in Figure 8, performance degrades after the first back-patching iteration across all but one task and model pairs, with each subsequent iteration further reducing model accuracy. We attribute this decline to representations becoming increasingly out-of-distribution with each iteration, making them progressively less compatible with the model’s learned parameters.

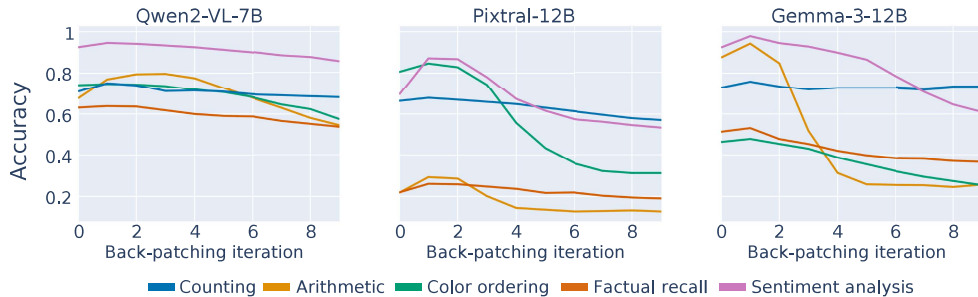


Figure 8: **Iterative back-patching results.** Applying back-patching several times in the highest-performing settings leads to a decrease in accuracy after the first back-patching application.

## E.2 Best Layers For Back-Patching

For each model and task, Table 7 shows the values of  $l_{src}$ ,  $l_{dst}$  and the layer window size that lead to the highest back-patching results, reported in Table 1.

Table 7: **Best back-patching settings.** Each cell presents the  $l_{src}$ ,  $l_{dst}$  and layer window size (in parenthesis) that leads to the highest back-patching accuracy.

Task	Qwen2-7B-VL	Pixtral-12B	Gemma-3-12B
Counting	20 → 15 (3)	28 → 23 (5)	36 → 5 (3)
Arithmetic	11 → 7 (3)	37 → 10 (5)	32 → 5 (1)
Color Ordering	18 → 15 (1)	32 → 22 (5)	34 → 24 (1)
Factual Recall	11 → 6 (5)	32 → 18 (5)	34 → 20 (5)
Sentiment Analysis	15 → 5 (5)	37 → 19 (5)	23 → 3 (5)

These optimal layers match our hypothesis across models: replacing visual data token representations at model layers when they are less textually-aligned, with more textually-aligned representations, yields the highest accuracy improvements. This pattern persists even in Gemma-3-12B, where visual data tokens demonstrate high alignment with textual analogs throughout most of the model (as shown in Figure 7): tasks showing the greatest accuracy gains from back-patching (counting, arithmetic, and sentiment analysis, as detailed in Table 1) benefit from patching in the earliest layers where the alignment of tokens with textual analogs is relatively low.

## E.3 Back-Patching Control Experiments

This section presents our back-patching control experiment results across models and tasks. The accuracy increases from back-patching visual data tokens (reported in Table 1) can either stem from earlier textually-aligned representations or from additional compute. To isolate the effect of adding computational layers, we conduct the same back-patching analysis on analogous textual prompts. In this control experiment, the alignment with textual tokens is inherent, so any accuracy increase must stem from added compute. Across each back-patching setting—model, task and layer window size—we check for how many values of  $l_{src}$  and  $l_{dst}$  is the accuracy increase caused by back-patching larger than the control accuracy increase. Our findings (Figure 9) show that in most settings, visual back-patching yields greater accuracy improvements than the control for a majority of layer values. A notable exception occurs in factual recall tasks for Qwen2-VL-7B and Gemma-3-12B, where back-patching shows minimal accuracy gains and is outperformed by the control in over half the cases. These results support our hypothesis that improved visual performance stems primarily from the early transition of visual tokens into textually-aligned representations, rather than from increased computational depth.

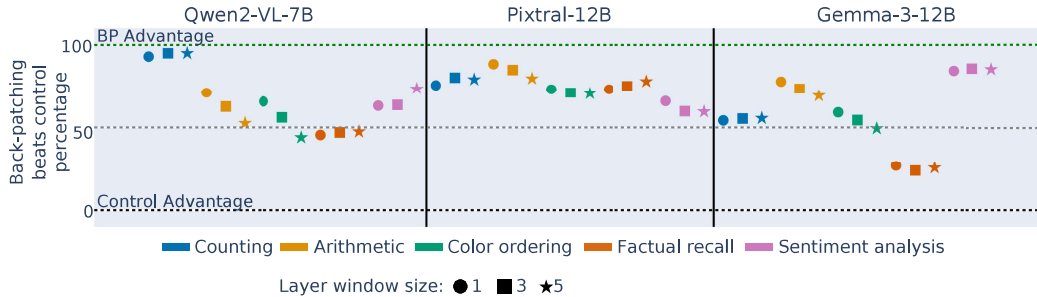
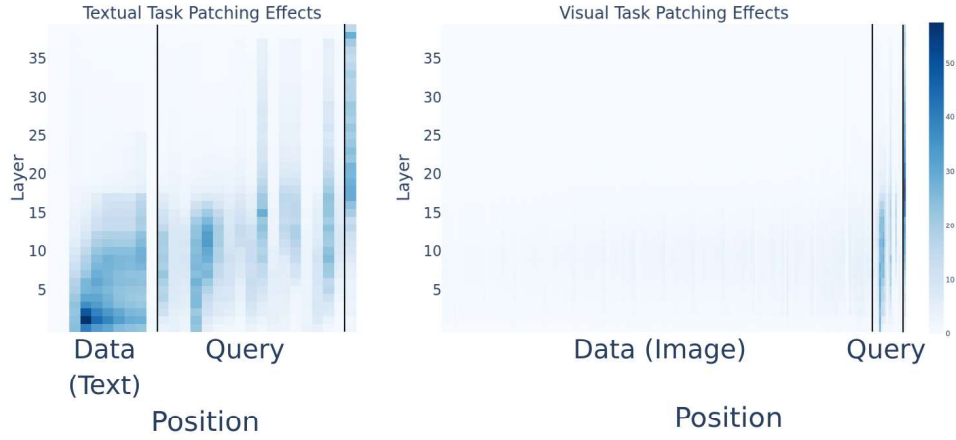


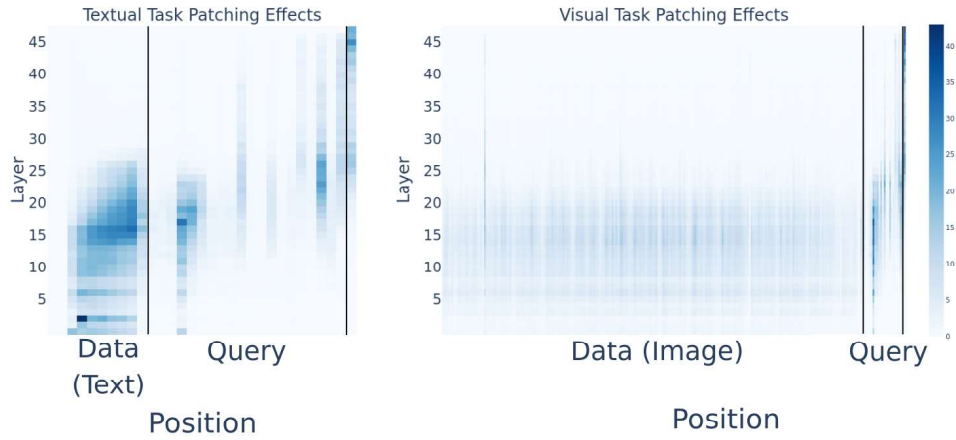
Figure 9: **Back-patching vs control results.** We measure the percentage of  $l_{src}$ ,  $l_{dst}$  values for which the control accuracy increase is lower than the back-patching accuracy increase. In most model, task and layer window size combinations, back-patching shows higher accuracy increase (above the 50% line) compared to the control results.



(a) Patching effects for Qwen-7B-VL for the textual (left) and visual (right) counting task.



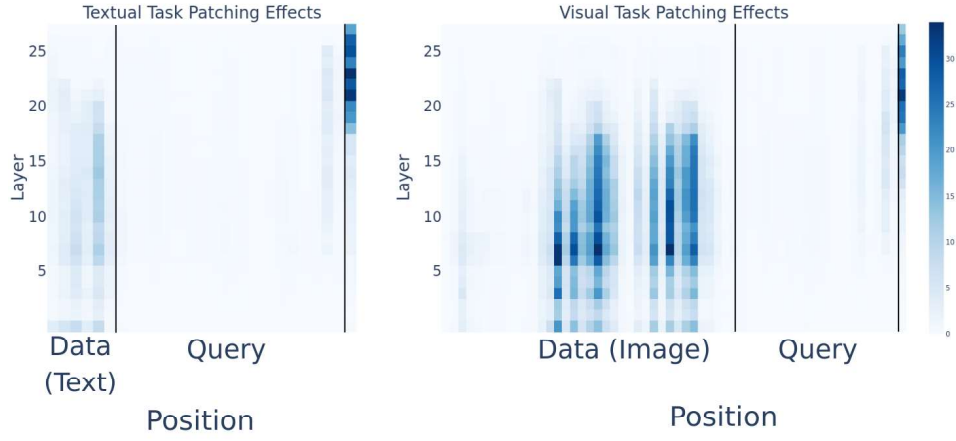
(b) Patching effects for Pixtral-12B for the textual (left) and visual (right) counting task.



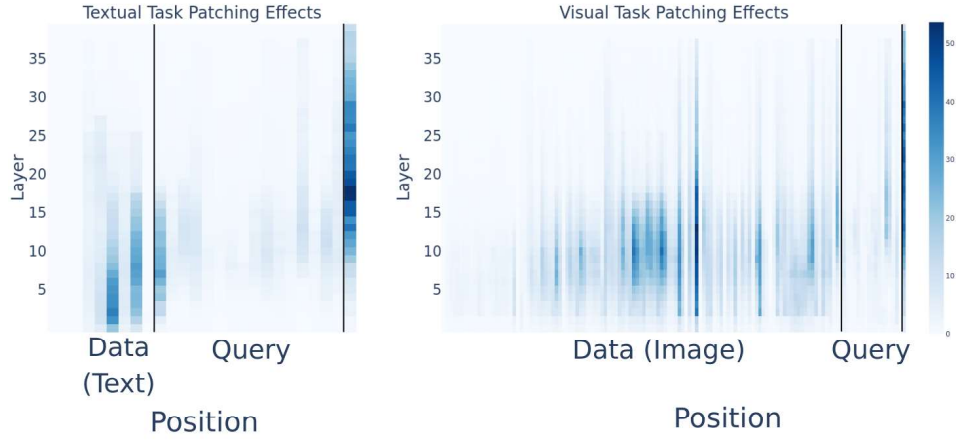
(c) Patching effects for Gemma-3-12B for the textual (left) and visual (right) counting task.

Figure 10: Patching effects for the counting task. The vertical lines split between data, query and generation positions.

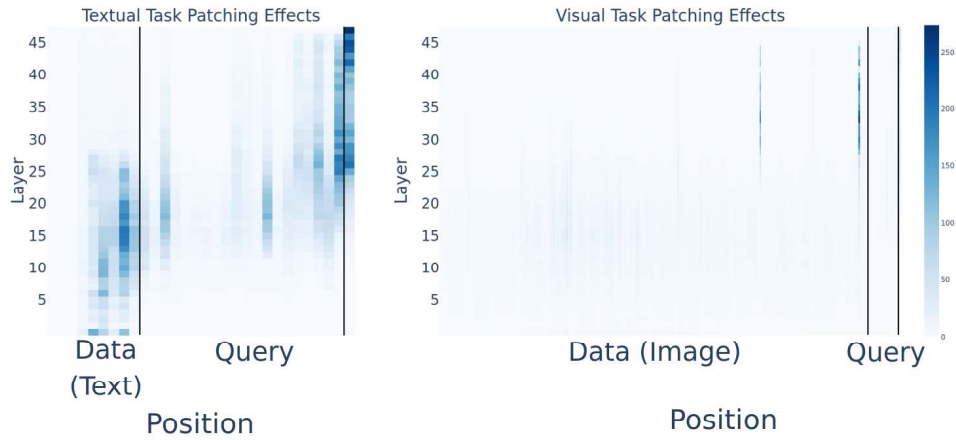




(a) Patching effects for Qwen-7B-VL for the textual (left) and visual (right) arithmetic task.

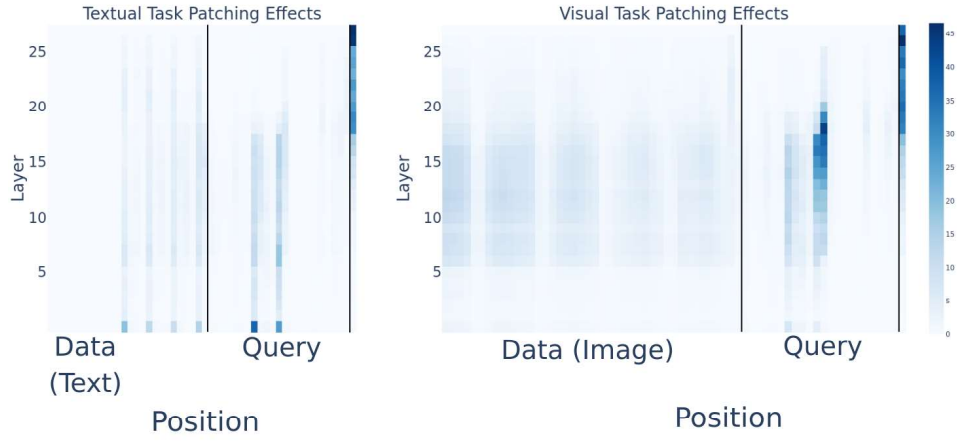


(b) Patching effects for Pixtral-12B for the textual (left) and visual (right) arithmetic task.

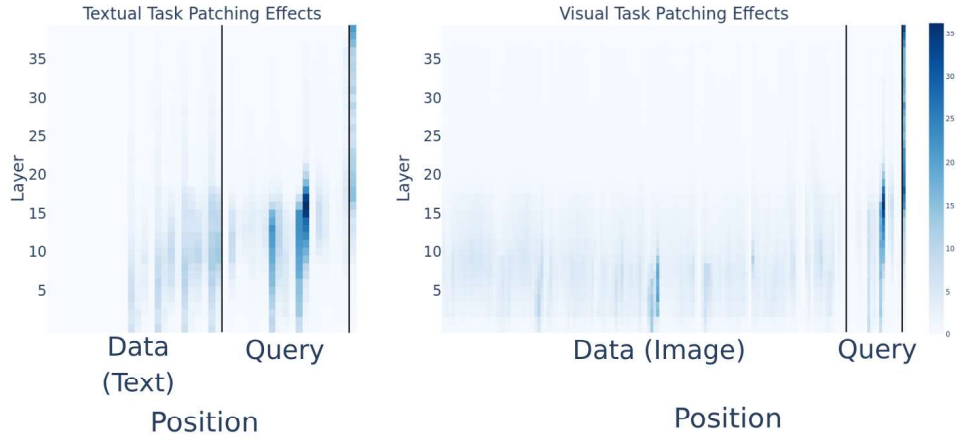


(c) Patching effects for Gemma-3-12B for the textual (left) and visual (right) arithmetic task.

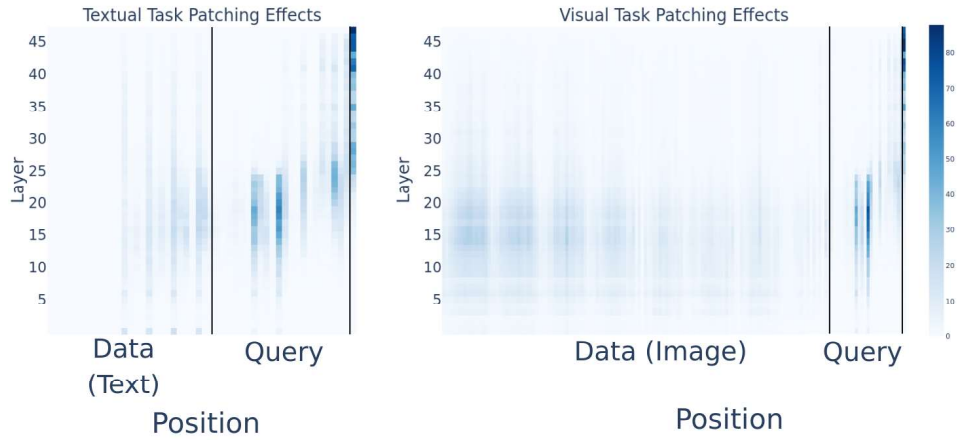
Figure 11: Patching effects for the arithmetic task. The vertical lines split between data, query and generation positions.



(a) Patching effects for Qwen-7B-VL for the textual (left) and visual (right) spatial ordering task.



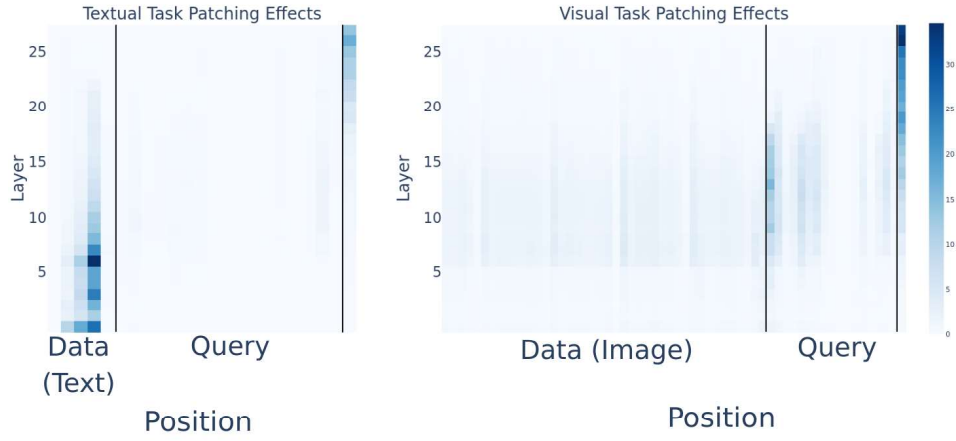
(b) Patching effects for Pixtral-12B for the textual (left) and visual (right) spatial ordering task.



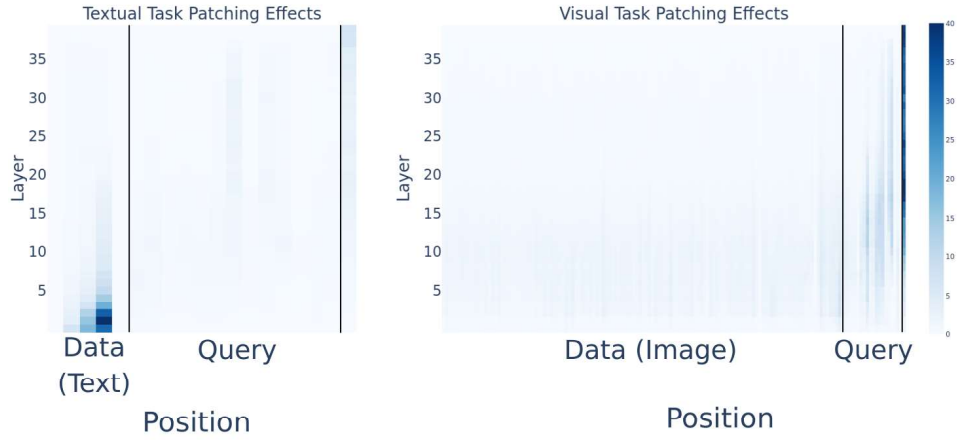
(c) Patching effects for Gemma-3-12B for the textual (left) and visual (right) spatial ordering task.

Figure 12: Patching effects for the spatial ordering task. The vertical lines split between data, query and generation positions.

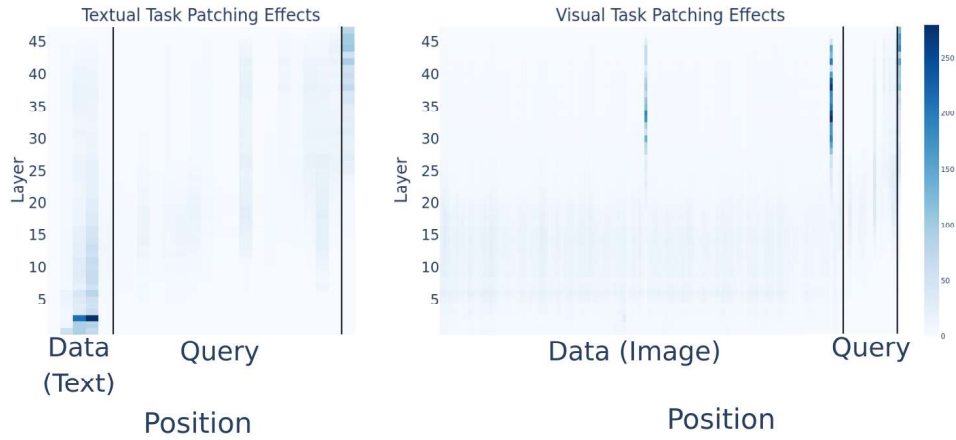




(a) Patching effects for Qwen-7B-VL for the textual (left) and visual (right) factual recall task.



(b) Patching effects for Pixtral-12B for the textual (left) and visual (right) factual recall task.

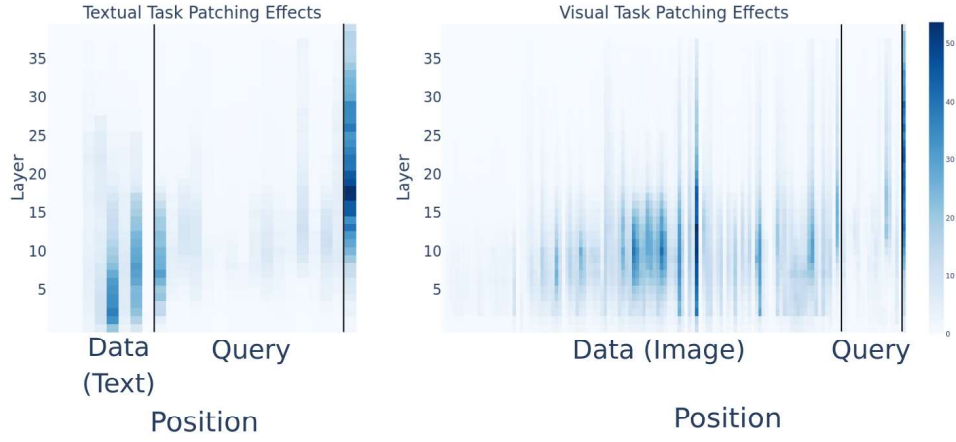


(c) Patching effects for Gemma-3-12B for the textual (left) and visual (right) factual recall task.

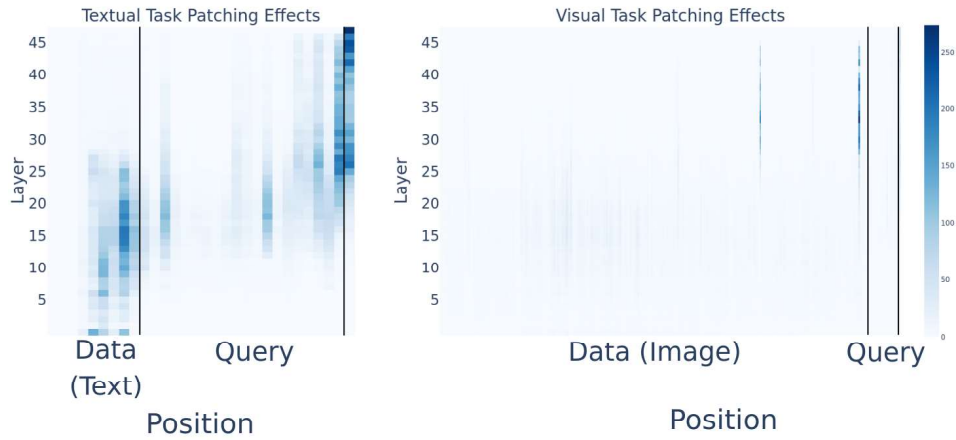
Figure 13: Patching effects for the factual recall task. The vertical lines split between data, query and generation positions.



(a) Patching effects for Qwen-7B-VL for the textual (left) and visual (right) sentiment analysis task.



(b) Patching effects for Pixtral-12B for the textual (left) and visual (right) sentiment analysis task.



(c) Patching effects for Gemma-3-12B for the textual (left) and visual (right) sentiment analysis task.

Figure 14: Patching effects for the sentiment analysis task. The vertical lines split between data, query and generation positions.