How VLMs Plan What's Next

Anonymous authors
Paper under double-blind review

Abstract

When a vision-language model describes an image, it names one object at a time and it rarely repeats itself, yet how the model plans this order is not obvious. We characterize the mechanism a VLM uses to *decide where to look* before it looks. The mechanism has four main parts. First, *binding heads* match the words of the question against the image and turn them into pointers to every matching image region. Second, the model binds each written answer to the region it came from, and turns it into a *progress* signal: a cursor, the position in the image up to which the list has been written. Third, a *select* operation chooses the nearest unwritten pointer after the cursor, or stops generating. Finally, *gaze heads* read the image at the chosen region and return a language-free concept, which *language heads* and the final MLPs turn into a word in the language of the question. We validate each part through causal patching in Qwen3-VL models on a synthetic visual retrieval task, pairing donor and receiver runs so that every alternative explanation predicts a different answer. We further show that the mechanism holds across model sizes, input formats and languages.

1 Introduction

How does a vision-language model decide which part of an image to describe next? There is an inherent mismatch between image and text modalities. Nothing in a two-dimensional image imposes a one-dimensional order on its contents. Somewhere inside the model, a plan must turn the image into a sequence of words. In this work, we characterize the mechanism inside a VLM that chooses this order while it describes an image. We study this problem in a controlled environment where images and prompts have enough structure for us to distinguish between several possible strategies.

Prior work has found Gaze Heads, a small set of attention heads whose image attention follows the region the model is describing, and redirecting them redirects the description (Gandikota & Bau, 2026). They show where the model looks, but only after the choice is made. Studies of binding in VLMs show that when a prompt names an object, the model retrieves a content-free index of its position and later uses that index to fetch its features (Assouel et al., 2026; Prakash et al., 2026). What happens once the model starts writing is much less clear. VLMs list the objects of a grid in a fixed order (Assouel et al., 2026), but how that order is controlled has been left open. We still do not know how a VLM plans one answer after another, what form that plan takes, or what sets it. A transformer makes this harder, because it has no recurrent state. What one token computes at layer l reaches later tokens only at layer $l+1$ or above, so a plan cannot simply be handed from one answer to the next without running out of layers.

To find the plan, we design a task that allows us to control each part of the mechanism on its own and see what changes. Each image is a strip of panels, each panel holds a shape and a letter, and the question asks for the letter inside a named shape. A patch that changes the answer could have moved the request, the image layout, the chosen answer region, or the answer itself. So we pair every donor run with a receiver run in which each of these four predicts a different letter. The receiver's answer then tells us exactly what the patched state carried.

We find that the plan is small and explicit (Figure 1), and three things about it surprised us. First, the model computes a pointer to the answer region early, at the word in the user prompt that names the shape. A few *binding heads* do this: their queries carry the request, their image keys carry the layout, and the two can be independently controlled. Second, *progress* is a place in the image, "done up to panel 4". When the model writes a letter, attention heads bind that letter to the panel it came

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

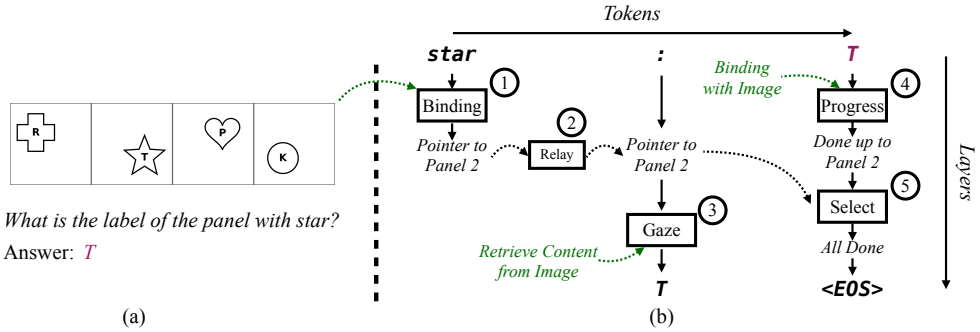


Figure 1: **How a VLM plans what to describe next.** (a) Task setup: We provide an image strip with each panel containing a shape and a letter. We ask the model to retrieve the label of the panel that contains the queried shape. (b) The mechanism, with tokens across and layers down. The model solves the task early, at the word `star` in the user prompt, well before it answers: (1) binding heads match the request against the image and write a pointer to the matching panel. (2) Relay heads carry that pointer to the "`:`" token, where the answer has to be produced, and (3) gaze heads use it to retrieve the letter from that panel. (4) Once `T` is written, the model records its progress, the position in the image it has covered so far, and (5) selects what is left after it. Here nothing is left, so it ends its turn. With several matching panels, steps 3 to 5 repeat at every answer.

from, and that panel becomes the cursor. It rebuilds this state at every answer from the letter it has just written, rather than carrying it over from the last answer. It then selects the nearest unwritten candidate after the cursor, or stops. Third, a smaller set of *gaze heads*, chosen by causal effect, does the reading, and reads the pointer better than the published Gaze Heads. They read the pointers written by binding heads, fetch what is drawn in that visual region, and return a concept that does not depend on the language of the question. A few *language heads* and the last MLPs turn these gaze outputs into a word. The same planning mechanism holds across model sizes, image layouts, natural photographs, and four languages.

2 Related Work

Binding and pointers: When a prompt introduces several entities, language models tag each one with an abstract code tied to the order of mention, and a later attention step follows that code back to the entity it needs (Feng & Steinhardt, 2024; Dai et al., 2024; Prakash et al., 2026; 2024). LLMs also contain filter heads that carry the rule that selects the matching items in their queries (Sen Sharma et al., 2026). VLMs often fail to keep the features of several objects apart (Campbell et al., 2024). Image information reaches the answer either through the question tokens (Zhang et al., 2025b; Basu et al., 2024) or directly from the image tokens (Neo et al., 2025; Kaduri et al., 2025; Gandikota & Bau, 2026), and the route depends on the task (Salazar et al., 2026). On a synthetic task, several works have found attention heads retrieve a pointer through binding, which is later used by another set of heads to retrieve the information (Saravanan et al., 2025; Kang et al., 2026; Wang & Xu, 2026; Cui et al., 2026; Assouel et al., 2026). We find similar binding mechanisms, and extend this study to how a VLM keeps track of its answers and how it chooses the next object when the question matches several regions.

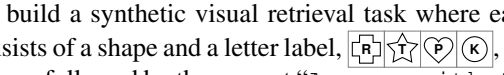
Keeping track of progress: Language models plan beyond the next token (Pal et al., 2023; Lindsey et al., 2025; Hanna & Ameisen, 2026) and keep a running state of their own output: a character count (Gurnee et al., 2025), a count of list items at the separators (Hasani et al., 2026), and the suppression of answers already given (Yan & Jia, 2025; Merullo et al., 2024). VLMs seem to work serially only through the text they generate (Budny et al., 2025) and judge when to stop by comparing that text with the image (Yue et al., 2024). They list the objects of a grid in raster order (Assouel et al., 2026). A VLM trained to point keeps an internal record of the objects it has counted (Haputhanthri et al., 2026). We find such a record in a general VLM that writes only letters, and it is neither a count nor a set of given answers but a place in the image, set by the last answer, from which the next region is chosen.

Image reading heads: Several studies find small sets of heads that attend from the text to the relevant part of the image. Some select heads by where they look (Kang et al., 2025; Gandikota & Bau, 2026) while others rank heads by their causal effect (Golovanevsky et al., 2025; Schaumlöffel et al., 2026; Liu et al., 2026b). We find that heads chosen by causal effect read the pointer better than heads chosen by attention and are more causally sufficient.

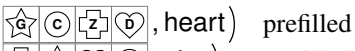
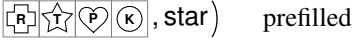
Concept and output language: Multilingual language models hold concepts in their middle layers in a form close to their dominant language, often English (Wendler et al., 2024; Schut et al., 2025; Wu et al., 2025). Concept and output language can be patched apart (Dumas et al., 2025) with a small set of attention heads (Zhang et al., 2025a; Lasnier et al., 2026; Liu et al., 2026a). We find that when the concept comes from an image, the heads that read the image return language-independent information, and separate heads and the last MLPs translate it into the question's language.

3 Setup and Notation

Model: We primarily study the language backbone of the Qwen3-VL-8B-Instruct model (Bai et al., 2025), which has 36 layers of 32 attention heads. The vision encoder runs once per image and its output is cached and never edited, so it plays no part in our interventions. We show that the findings generalize across model sizes in Appendix F.

Dataset: We build a synthetic visual retrieval task where each image is a strip of blank panels. Each panel consists of a shape and a letter label, , as shown in Figure 1a. The user input contains the image followed by the prompt "`Answer with the letter only. What is the letter inside the star?`". We prefill the assistant turn with "`Answer:`", so the next token is the letter "`T`". Four tokens will matter throughout. The word that names the shape, here "`star`", is the *marker token*. The *colon token* "`:`" produces the first answer letter, and in the list task each later *comma token* produces the next answers. Each letter the model has already written is a *label token*. Throughout the paper, we use the notation $\big($, star$\big)$ to define the input image and prompt. Shapes and letters are placed at random, and the question names a shape rather than a panel, so the model must find the shape in the image before it can read the letter. The 8B model answers our synthetic task correctly in 98.5% of 2,000 questions (Appendix A.6).

Intervention: Every experiment pairs two runs that differ in both the image and the question:

Donor: $\big($, heart$\big)$ prefilled "`Answer:`" $\rightarrow$ `D`
Receiver: $\big($, star$\big)$ prefilled "`Answer:`" $\rightarrow$ `T`

We record internal activations of the donor run and patch them into the receiver run, one component at a time, to read the causal effect on the receiver's next token. The pairs are built so that the question, the layout, and the letters all differ, and so that each possible answer names what the patch moved:

- `T`, the receiver's own letter: the patch had no effect.
- `D`, the donor's letter: the patch copied the whole answer.
- `K`, the receiver's letter in panel 4, where the donor found its answer: the patch moved the location the model reads from.
- `P`, the receiver's letter inside its own heart: the patch moved the request, and the donor's shape was found on the receiver's image.
- `R`, the receiver's letter in panel 1, where the donor's image has the star: the patch moved the layout, so the receiver's question was resolved on the donor's layout and the letter was read from the receiver's image.

We patch at three levels: the residual vector at one layer, the output of one MLP, and the output of single attention heads. For heads we also patch the queries, the keys, and the values separately. Heads are never hand-picked. We patch one head at a time, for all 1,152 heads, on 200 discovery pairs, and rank the heads by how much the patch raises the logit of the target letter over the receiver's own letter. We then patch the top K heads together for K in 1, 2, 4, ..., 128 on 100 validation pairs

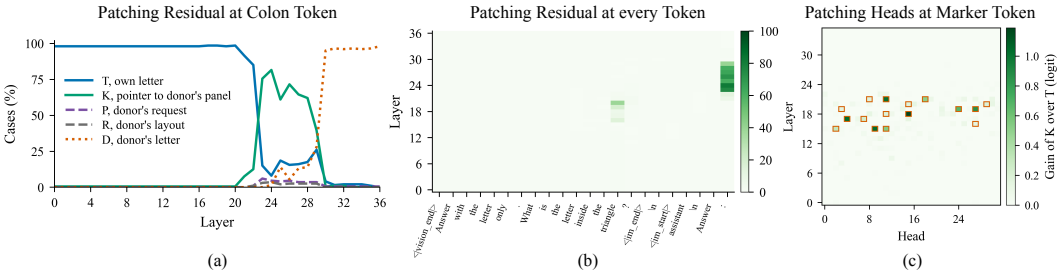


Figure 2: **The pointer is computed at the marker token by a few attention heads.** (a) Patching the donor's residual at the colon token into the receiver, one layer at a time. Up to layer 20 the receiver keeps its own letter `T`; from layers 23-29 it answers `K`, receiver's letter at the donor's answer location, so the colon token holds a pointer to a panel; from layer 30 it copies the donor's letter `D`. (b) The same patch at every text token and every layer, tracking the logit of `K`: the pointer appears only at the marker token, around layer 20, and at the colon token, from layer 23. (c) The output of one attention head at the marker token replaced with the donor's, for every head; each cell is the mean gain of `K` over `T`, and the 16 heads selected by the search sit in layers 15-21 and are outlined.

to choose K, and report every result on 500 held-out pairs that played no part in either choice, against the same number of random heads from the same layers (Appendix A).

Task types: In our dataset, some inputs have one answer and some have a list. In the list task the asked shape sits in multiple panels. We ask "`What is the letter inside each {shape}?`", and the model lists the letters from left to right, for example $\left(\boxed{\text{E}}\boxed{\text{M}}\boxed{\text{U}}\boxed{\text{V}}\boxed{\text{P}}\boxed{\text{A}}, \text{heart}\right) \rightarrow$ `E, U, P, A` (Appendix A.3). We use the list task to study how the model tracks its progress through the list and selects the next panel. Appendix A describes the image design, token positions, pairs, and splits in more detail.

4 Vision-Language Binding: VLMs Solve a Task via Pointers

Similar to language models (Prakash et al., 2024; Feng & Steinhardt, 2024; Dai et al., 2024; Prakash et al., 2026), we find that vision-language models, when solving our retrieval task, also use pointers (Assouel et al., 2026; Cui et al., 2026) to the corresponding image regions. To show this, we use the donor and receiver of Section 3, which answer from panels 4 and 2, and patch the entire residual vector at the colon token from the donor into the receiver, one layer at a time. We then track which of the five letters of Section 3 the receiver answers with (Figure 2a). Up to layer 20 the receiver's answer is unchanged in 98% of cases. From layers 23-29 the receiver answers with `K`, its own letter in panel 4, where the donor found its answer, in up to 81.5% of cases, with the peak at layer 24. From layer 30 on it copies the donor's letter `D` in 95% of cases or more. This shows that in layers 23-29 the colon token holds a pointer to a panel and not a letter. The model uses this pointer to retrieve the letter from the image, and writes the final answer into the residual stream after these layers. In this section, we investigate where and how this pointer is created.

4.1 Pointers are Calculated as Early as the User Prompt Tokens

Instead of patching at the "`:`" token alone, we patch at every token in the user prompt and the assistant prompt, at every layer, to track the pointer signal. Figure 2b shows how much the logit of `K`, the receiver's letter at the donor's panel, rises when the patch is applied at each token and layer. The pointer moves in only two places: at the marker token "`star`", and at the colon token "`:`". To investigate further, we patch the output of one attention head at a time at the marker token. We rank the heads by how much each patch raises the logit of `K`. We find that the top 16 attention heads, at layers 15-21, causally calculate the pointer (Figure 2c): patching the outputs of these 16 heads moves the receiver to `K` in 85.5% of cases, while the same number of random heads from the same layers does so in at most 0.5%. We call them *binding heads*.

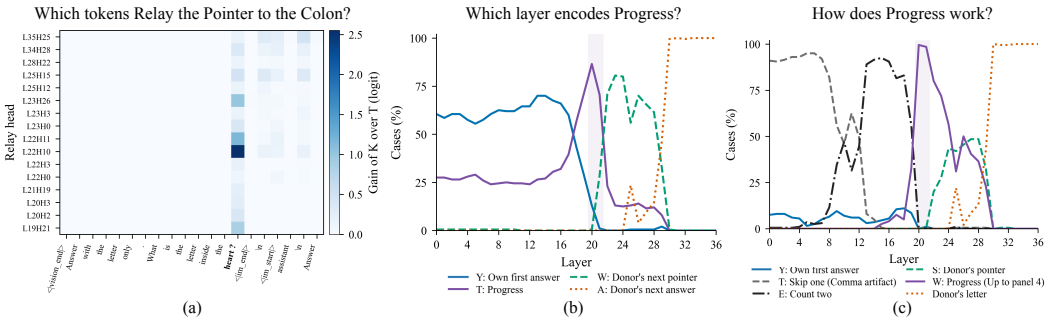


Figure 3: (a) **Relay heads transfer the pointer from marker to colon.** When each relay head may read only one token at a time, the pointer is transferred mostly when the relay heads at the colon token read the marker token. Each cell is the mean gain of `K` over `T` on 500 held-out pairs. (b) **Progress is encoded in layers 20-21**. When patching the donor's comma-token residual into the receiver's colon-token residual, one layer at a time: at layers 20-21 the receiver skips to its next candidate, `T`, in 86.5% of pairs. From layers 22-28 it takes the donor's next panel pointer, `W`, and from layer 30 the donor's letter, `A`. (c) **Progress is an "Up to panel x" signal**. With a more targeted setup, we find at layers 20 and 21 the model answers with the first candidate "after panel 4", `W`, in 99.5% and 98.5% of pairs. The shaded band marks layers 20-21.

We further investigate how the binding heads calculate the pointer. For this, we patch the queries and the keys of all heads at the marker token separately (Appendix Figure 7). As with the outputs, we rank every head by each edit and find a set of query heads and a set of key heads, in the same layers as the binding heads. Patching the donor's queries of the 8 query heads makes the receiver answer with "`P`", the letter inside its own heart, in 76.5% of cases. In other words, the queries act as a proxy for the language instruction "I need to look for hearts". The request comes from the first MLP. Replacing its output at the marker token with the donor's moves the request in 98.0% of cases, and no other MLP moves it in more than 7.5% (Appendix B.3). Letting the 16 key heads read the donor's image keys makes the receiver answer with `R` in 90.5% of cases, where the pointer is calculated with the receiver's request of star but with the donor's layout. So the keys of the image tokens act like "I am the panel with the star". Patching both the queries and the keys of these heads moves the pointer itself, `K`, in 94.0% of cases. When the binding heads read the donor's image values instead, the answer is unchanged in 94.5% of cases, so they write a place and not what is there (Appendix B.4). In every case the letter is still read from the receiver's own image. So the pointer computed at the marker token is used later, at the colon token. How does it get there?

4.2 Relay Heads Carry the Pointer to the Colon Token

Attention is the only way to move information between tokens, so some heads at the colon token must fetch the pointer from the marker (Figure 3a). We investigate this by patching, one head at a time, the values that a head at the colon token reads from the marker token and the tokens after it. The head keeps its own query and keys. We find 16 attention heads, mostly in layers 19-23, where swapping these values moves the donor's pointer into the receiver: the receiver answers `K` in 87.0% of cases (Appendix Figure 10a). We call them the *relay heads*.

To show that relay heads are the channel of pointer flow from the marker to the colon token, we run two interventions. First, when the relay heads cannot read the text at the colon token, the receiver keeps its own answer in only 8.0% of cases, against 97.5% when the same number of random heads is blocked (Appendix B.6). The model still reads a letter from the image, but without the relay it no longer knows which panel. Second, we build a run in which only the binding heads' outputs at the marker token are replaced with the donor's, everything else being the receiver's own. When the relay heads of an unedited receiver read the values of this run, the pointer moves in 87.5% of cases, as often as with the donor's own values. Giving that run the receiver's values back at the relay heads restores the receiver's answer `T` in 99.5% of cases (Appendix B.5). This shows that the binding heads calculate the pointer at the marker token, and the relay heads transport it to the colon token, where the model reads the right letter from the image.

5 PROGRESS AND SELECT: HOW A VLM PLANS THE NEXT ANSWER

The previous section shows how a VLM solves a retrieval task with one unique answer. How does the model solve the same task when several panels are correct? This is closer to how a VLM is used for image captioning, where the model names several regions of an image one after another. On a list, the 24 query, key, and binding heads at the marker token put 72% of their image attention on the four matching panels, a probe on their output reads each candidate panel with 94% accuracy, and the first answer is the leftmost candidate (Appendix B.8). To solve the list task the model has to do two more things beyond binding-based task solving: keep track of how far it has come, and pick the next panel. We call the first signal progress and the second step select. In this section we show how the model encodes its progress, and how it uses that progress to select the next answer or to stop.

5.1 LOCALIZING THE PROGRESS SIGNAL INSIDE VLMS

Since the language backbone is autoregressive, the only tokens that can carry progress are the ones after at least one answer has been written: the label tokens and the comma tokens. We start with the comma token, by copying the residual vector of the donor's comma token into the receiver's colon token, one layer at a time. The donor has written one letter and the receiver none: so the receiver has no label token of its own to read, and whatever it skips it learned from the copied state. The two runs ask for different shapes and share no letter; the only thing they share is the panel the donor has just answered from, which is also the receiver's first panel:

Donor: (, star) prefilled "`Answer: X,`"
Receiver: (, heart) prefilled "`Answer:`"

With no edit, the receiver says `Y`. If the patched signal represents progress past the first answer, the receiver should naturally move onto its second answer, `T`. If it is a pointer to the next answer, the receiver should output the label at panel 6, `W`. Finally, if the state already carries the donor's next answer as a letter rather than a pointer, the receiver says `A`, the letter of the star in the donor's panel 6. Figure 3b shows the four curves. Up to layer 17 the receiver mostly keeps `Y`. At layers 20-21 it says `T` in 86.5% of cases: it has skipped a panel it never wrote, because a run about a different shape with different letters had written from that panel. From layer 22 the donor's next place `W` comes in, and from layer 30 the donor's letter `A`. So the comma token holds progress at layers 20-21, before it holds the next place, and progress is about a panel, not about a shape or a letter.

5.2 PROGRESS IS AN ORDINAL POSITION: "DONE UP TO HERE"

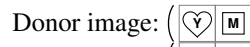
This raises a natural question: what is the nature of the progress signal? Is it a count ("two answers done"), a set of pointers ("panels 1 and 4 done"), or an ordinal position ("done up to panel 4")? We use a patching setup in which each type of signal predicts a different letter: the donor has written two letters, from panels 2 and 4, and its next answer is at panel 5. The receiver has hearts in panels 1, 2, 3, and 6, so its own first answer, panel 1, is not a panel the donor has written:

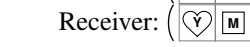
Donor: (, star) prefilled "`Answer: X, V,`"
Receiver: (, heart) prefilled "`Answer:`"

If progress is a count, "two done", the receiver skips two hearts and says `E`. If it is a set of pointers, "panels 2 and 4 done", the receiver outputs the first heart that is not either in panel 2 or 4, `Y`. If the progress is an ordinal position, "done up to panel 4", the receiver answers with the first heart after panel 4 and says `W`. Figure 3c shows the receiver's next letter at every layer. Through layer 7 the receiver says `T` in 91 to 95% of pairs. This is not progress: a comma token by itself says "one letter was written", before it has read anything (Appendix C.2). From layers 13-18 it says `E` in 81.5 to 92.5% of pairs: by layer 13 the comma has counted the written letters, so the early state is a count. At layer 20 the receiver says `W` in 99.5% of pairs, 98.5% at layer 21, and 56.5 to 80.0% through layer 24. It does not keep its own answer: `Y`, the outcome a set of pointers would produce, occurs in at most 1.0% of pairs in this band. So at the layers where the previous subsection found progress, the signal is an ordinal position: the list is done up to panel 4, and the next answer starts after it. We call this position the cursor.

5.3 The Cursor Is Set by the Last Label Token

Where does the cursor come from? The label tokens are the first tokens that know what has been written. We take a receiver that has written two letters and let one of its label tokens see a different image, in which the letter it wrote sits one panel further to the right:

Donor image: 

Receiver: 

prefilled "`Answer: Y, T,`" → `E`

The donor image is the receiver's image with the letters of panels 3 and 4 exchanged, so the shapes, the candidates, and the written letters are all unchanged; only the panel that holds `T` has moved. We let the binding heads at the last label token, `T`, compare their own queries with the donor image's keys, where `T` sits in panel 4, and change nothing else. If the cursor is the location bound at the last written letter, it moves from panel 3 to panel 4, and the receiver skips panel 4 and says `W`. This happens in 85.5% of pairs with the 16 binding heads, in 86.0% with the 16 heads found at the label token, and in 0% with 16 random heads from the same layers; the receiver repeats a written letter in at most 2.5% (Appendix Figure 15). The same edit at the first label token, with a donor image in which `Y` sits in panel 4 instead, leaves the answer at `E` in 100% of pairs with the binding heads and 97% with the 16 label heads. So the cursor is set by the last label token, and an earlier label token rebound to a panel ahead of the cursor does not move it. When the last comma cannot read the label token, the model writes the same letter again in 66.5% of cases, while cutting its read of the colon leaves 97.4% unchanged (Appendix C.4). When the next candidate holds the same letter that was just written, the model writes it again in 93.5% of lists (Appendix C.3).

The edit is the binding edit of Section 4, applied to a written letter instead of the asked shape, and 7 of the 16 heads found at the label token are binding heads (Appendix C.3). So binding happens twice, largely through the same heads. At the marker token they bind the asked shape to its panels. At a label token they bind the letter just written to the panel it came from, and by layer 21 the comma token has turned that panel into the cursor.

5.4 Select: The Next Candidate After the Cursor, or None

With the candidates and the cursor both at the comma token, the choice can be read off the model with no edit, by writing letters out of order (Appendix Figure 16). With the second heart written and nothing else, the model says the third in 91.5% of lists and goes back to the first in 3%. With the third written it says the fourth in 64% and goes back to the second in 35.5%. With the third and then the first written it says the second in 91.5%. So select starts at the cursor and takes the nearest unwritten candidate, preferring the right; when nothing lies to the right it goes back. The written places count as done: with the second and then the first written the model says the third in 86.5% of cases, and when the comma cannot read the second letter's token it writes that letter again in 43.9% (Appendix C.5). Gaze heads given a donor query that points at an already written place are followed in 0.0% of cases, against 88.7% for an unwritten candidate, so the cursor also vetoes a read (Appendix D.3).

The choice happens at the comma token in layers 20-25. We hide the next panel from every text token from layer l on and sweep l (Appendix Figure 19). Hidden from the first layer, the model moves on to the following candidate in 85.6% of cases, so the other candidates are there to fall back on. Hidden from layer 20 on, 71.1%; from 21 on, 34.0%; from 25 on, 6.2%. Probes on the comma's residual give the same window with no edit: the candidate set is readable at 88% per panel at layer 17 and 97% at 21, the next place at 85% at layer 18 and 99% at 20, with the written panel scored below the other candidates throughout (Appendix C.7). The comma token finds the candidates again by itself: a donor set swapped into the marker changes the next letter in 26.4% of cases, and only when the comma's own view of the image is cut does it reach 72.5% (Appendix C.6).

The cursor acts on this choice through the comma's residual. When progress is moved at the label token, holding the comma's residual at its own values at layer 23 or later brings the answer back in 97.4% of cases, holding it at layer 19 changes nothing, and holding the gaze heads' queries brings it back in 98.5% (Appendix C.7). No small set of heads makes this choice. It is spread over layers 20-25, and the gaze heads read its result (Appendix C.8). The same step decides whether to stop. At the last letter the model chooses between a comma and the end of its turn, and the outputs of the

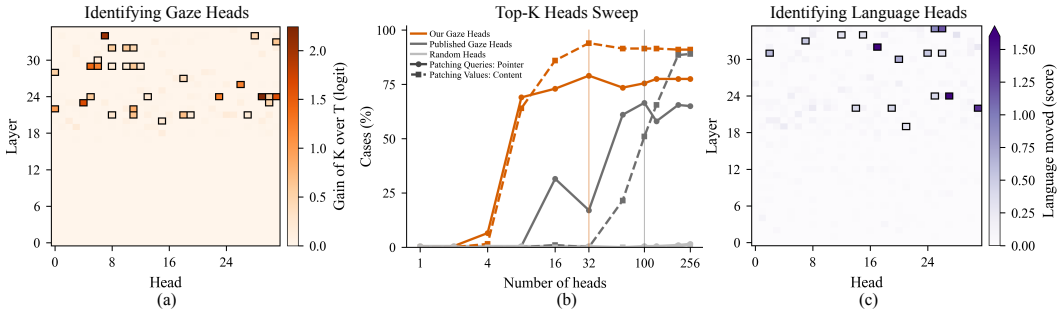


Figure 4: (a) **Gaze heads exist after layer 20.** Pointer logit gain when one head's query at the colon token is patched. The 32 gaze heads are outlined, all in layers 20-34. (b) **Our gaze heads vs published Gaze Heads (Gandikota & Bau, 2026).** Solid lines replace the queries and count the receiver's letter at the donor's panel, K. Dashed lines let the heads read the donor's image values and count the donor's letter at the receiver's panel, C. The vertical lines mark our 32 heads and the top 100 published heads. Our 32 heads beat the top 100 published heads, and random heads stay at or below 1.5%. (c) **Language heads set the language.** Scoring the language change of a word when patching attention head outputs from a different language donor. The 16 language heads are outlined, in layers 19-35. The color scale is capped at 1.6; L24H27 scores 5.05.

heads found there make a finished list go on in 100.0% of cases, against 1.0% for random heads, and a continuing list stop in 96.4%, against 0.0% (Appendix C.5).

6 Visual Retrieval from Pointers and Language Translation

Through binding, progress tracking and selecting, VLMs write pointers to the corresponding image regions that contain the answer. In this section, we investigate how the model uses these pointers to retrieve the visual information and translate that to text.

6.1 Gaze Heads are the Visual Pointer Readers

Attention heads are the only components that can look back at the image, and a head that reads the pointer must build its query from it. To identify the heads that read the pointer and execute the retrieval, we use the donor and receiver of Section 3. At the "**:**", we patch the queries of every attention head from the donor's colon token. We rank the heads by how much the receiver moves toward its own letter at the donor's location, K (Figure 4a). Patching only the queries of the top 32 heads moves the answer to the donor's location in 79.0% of cases, against 0.5% for the same number of random heads from the same layers. We call them the *gaze heads*. Moving the pointer at the marker token, with the binding heads' outputs of Section 4.1, no longer changes the answer when the gaze heads' queries are held at their own values: the receiver keeps its own letter in 98.0% of cases, while holding random heads instead lets the moved pointer through in 85.5%.

We then let the image be seen only by the marker token and by these 32 heads at the colon token, and hide it from every other head at every other token. The model still answers correctly, with "T", in 96.0% of cases, against 3.5% when random heads may see in their place. Their queries carry the pointer, the image keys match it by place, and the image values bring the letter. When patching the values of the gaze heads from the donor's image, the model outputs the donor's label at the receiver's answer location, "C", in 94.0% of cases (Appendix D.3). This is the opposite of the binding heads, whose image values leave the answer unchanged (Section 4.1).

Our gaze heads resemble the published Gaze Heads (Gandikota & Bau, 2026). Unlike our discovery of the gaze heads via patching, Gaze Heads were discovered through observing the attention patterns when describing comic panels. We find that our gaze heads are a smaller subset of the published Gaze Heads, yet causally more effective. Patching the queries of the top 100 published Gaze Heads reaches 66.5% compared to 79.0% for our 32 gaze heads. Similarly, reading the donor's image values through the top 100 published Gaze Heads gives the donor's letter in 51.0% against 94.0% with our gaze heads (Figure 4b). Fewer heads, chosen by their causal effect, read the pointer better

than a longer list chosen by attention behavior. The top 100 published Gaze Heads contain 3 of the 16 binding heads, 4 of the 16 relay heads, and none of the 16 language heads (Appendix D.2).

6.2 Language Heads Translate Concept Information into the User's Language

The gaze heads read the region at the pointer, but it is unclear if they output the meaning of the visual concept or the word for the concept. To ask what they return, we flip our task and ask "What is the shape with the label X" in English, German, French, or Chinese. It answers with the right shape in the question's language in 100.0% of cases (Appendix A.5). This setup allows us to study the VLM's behavior when multiple answers can be correct depending on the language of the user prompt (e.g. 'circle' vs 'cercle').

We take the collective LogitLens token distribution of the 32 heads and find that they are nearly the same for an image with the same prompt under two languages, with a cosine of 0.95, and different for two images under one language, 0.50. The top logits of gaze heads are always English or Chinese, irrespective of the user prompt's language. Replacing the gaze heads' outputs with those of a donor that asks in another language about another image makes the receiver name the donor's shape in its own language in 81.7% of cases, and never changes the language; random heads move it in 0.0% of cases. So the gaze heads return the concept, not the word (Appendix D.4).

To find what component adds the language, we take a donor that asks in language A about image X and the receiver in language B about image Y, and track the token of the receiver's answer in the donor's language A, a word neither run produces. We keep the top 16 heads of this search, in layers 19-35, and call them the *language heads*. Replacing their outputs moves the language in 100.0% of cases without ever copying the donor's shape, against 0.0% for random heads (Figure 4c), and removing them returns 98% of German runs and 82% of French runs to English (Appendix D.7). However, language heads are not the components that actually translate the word. With the last three MLPs, layers 33-35, held at the receiver's values, the language moves in only 5.5% of cases. We can use the MLPs as independent translators: adding the output direction of an English noun at layer 26 of a German, French, or Chinese run makes the model say that noun in the question's language in 74.7% of cases (Appendix D.6). So an answer ends with a translation step. The gaze heads return a concept, the language heads add the language of the question, and the late MLPs join the two into a word.

The heads found on panel strips work unchanged on other images (Appendix E). On a blank canvas with scattered shapes, the gaze heads' queries move the answer in 90.0% of cases, and a pilot search there selects 16 gaze heads, all of them among our 32. On COCO photographs with natural list questions, a comma that cannot read the word just written names that object again in 79.6% of lists, and setting the language heads to English turns 96.5% of German answers English. The 32B model holds the same cursor, in 96% of 50 pairs at layer 48 of 64 (Appendix F).

7 Conclusion

We have traced how a vision-language model decides what to describe next. Binding heads at the word that names the shape match the question against the image and compute a pointer to every matching region. Relay heads carry the pointer to the colon token, where gaze heads read the image at the pointed region and return a concept that does not depend on the language of the question. Language heads and the last MLPs turn that concept into a word in the question's language. When several regions match, the model binds each written answer to the panel it came from, largely through the same heads, and the comma token keeps a cursor, the place in the image up to which the list has been written. From the cursor the model selects the nearest unwritten region, or stops.

The plan is rebuilt at every answer rather than carried from one answer to the next. A transformer has no recurrent state, and a value computed at one token reaches later tokens only in later layers, so a plan handed from answer to answer would run out of layers. Instead, the model binds each letter it writes to the image again in the middle layers, and the comma reads the cursor from that letter. The text the model has written is its record of where it has been. Much of interpretability work has mostly asked what a model knows when it predicts one token. Long generations ask what it intends to do next, and our results suggest that this intention is written down inside the model, one answer at a time, where we can read it.

AI USE STATEMENT

We used generative AI tools in four ways. First, to polish the writing and for spell check. Second, we used AI agents to scale up experiments that we designed and wrote ourselves, and to propose alternative hypotheses, which we then tested and ruled out in the paper. Third, we used them to search for related papers; we read the papers and wrote the related work ourselves. Fourth, we used them to clean up the structure of our codebase. We reviewed all AI-assisted work, and we take responsibility for the final content of this paper, including its text, claims, and code.

REPRODUCIBILITY STATEMENT

We release our source code, which generates the images and prompts, runs every experiment, and builds every figure and table from the saved results. The appendix gives the implementation details of every experiment: the model and its settings, the images and prompts, the donor-receiver pairs and splits, each intervention, and how heads are chosen (Appendix A), with the details specific to each section in Appendices B to F. We use only open-weight models, the Qwen3-VL family, and no closed-source model is used anywhere, including for judging or scoring answers.

REFERENCES

Rim Assouel, Declan Iain Campbell, Yoshua Bengio, and Taylor Whittington Webb. Visual symbolic mechanisms: Emergent symbol processing in vision language models. In *International Conference on Learning Representations*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Samyadeep Basu, Martin Grayson, Cecily Morrison, Besmira Nushi, Soheil Feizi, and Daniela Massiceti. Understanding information storage and transfer in multi-modal large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7400–7426, 2024.

Nicholas Budny, Kia Ghods, Declan Campbell, Raja Marjieh, Amogh Joshi, Sreejan Kumar, Jonathan D. Cohen, Taylor W. Webb, and Thomas L. Griffiths. Visual serial processing deficits explain divergences in human and vlm reasoning. *arXiv preprint arXiv:2509.25142*, 2025.

Declan Campbell, Sunayana Rane, Tyler Giallanza, Nicolò De Sabbata, Kia Ghods, Amogh Joshi, Alexander Ku, Steven M. Frankland, Thomas L. Griffiths, Jonathan D. Cohen, and Taylor W. Webb. Understanding the limits of vision language models through the lens of the binding problem. In *Advances in Neural Information Processing Systems*, volume 37, pp. 113436–113460, 2024.

Kelly Cui, Nikhil Prakash, Shoval Messica, Ayush Raina, David Bau, Antonio Torralba, and Tamar Rott Shaham. The dual mechanisms of spatial variable binding in vision-language models. *arXiv preprint arXiv:2603.22278*, 2026.

Qin Dai, Benjamin Heinzerling, and Kentaro Inui. Representational analysis of binding in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17468–17493, 2024. doi: 10.18653/v1/2024.emnlp-main.967.

Clément Dumas, Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Separating tongue from thought: Activation patching reveals language-agnostic concept representations in transformers. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31822–31841, 2025.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? In *International Conference on Learning Representations*, 2024.

Rohit Gandikota and David Bau. Gaze heads: How VLMs look at what they describe, June 2026. URL https://arxiv.org/abs/2606.14703.

Michal Golovanevsky, William Rudman, Vedant Palit, Carsten Eickhoff, and Ritambhara Singh. What do VLMs NOTICE? a mechanistic interpretability pipeline for Gaussian-noise-free text-image corruption and evaluation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11462–11482. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.571. URL https://aclanthology.org/2025.naacl-long.571/.

Wes Gurnee, Emmanuel Ameisen, Isaac Kauvar, Julius Tarng, Adam Pearce, Chris Olah, and Joshua Batson. When models manipulate manifolds: The geometry of a counting task. Transformer Circuits Thread, 2025. URL https://transformer-circuits.pub/2025/linebreaks/index.html. arXiv:2601.04480.

Michael Hanna and Emmanuel Ameisen. Latent planning emerges with scale. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H0B7pDTT0M.

Udith Haputhanthri, Declan Campbell, Rim Assouel, Jonathan D. Cohen, and Taylor W. Webb. Binding visual features point by point. *arXiv preprint arXiv:2605.25427*, 2026.

Hosein Hasani, Amirmohammad Izadi, Fatemeh Askari, Mobin Bagherian, Sadegh Mohammadian, Mohammad Izadi, and Mahdieh Soleymani Baghshah. Understanding counting mechanisms in large language and vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Omri Kaduri, Shai Bagon, and Tali Dekel. What's in the image? a deep-dive into the vision of vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14549–14558, 2025.

Raphi Kang, Hongqiao Chen, Georgia Gkioxari, and Pietro Perona. Linear mechanisms for spatiotemporal reasoning in vision language models. In *International Conference on Learning Representations*, 2026.

Seil Kang, Jinyeong Kim, Junhyeok Kim, and Seong Jae Hwang. Your large vision-language model only needs a few attention heads for visual grounding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Théo Lasnier, Armel Randy Zebaze, Djamé Seddah, Rachel Bawden, and Benoît Sagot. Translation heads: Disentangling meaning from language in LLM-based machine translation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=JKSDWukEgN.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. Transformer Circuits Thread, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Xin Liu, Qiyang Song, Qihang Zhou, Haichao Du, Shaowen Xu, Wenbo Jiang, Weijuan Zhang, and Xiaoqi Jia. Focusing on language: Revealing and exploiting language attention heads in multilingual large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32195–32203, 2026a. doi: 10.1609/aaai.v40i38.40492.

Xiwei Liu, Yulong Li, Xinlin Zhuang, Xuhui Li, Zhixiang Lu, Haolin Yang, Imran Razzak, and Yutong Xie. Grounding isn't knowing: Do VLMs need object localization for spatial reasoning? *arXiv preprint arXiv:2608.23074*, 2026b.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Talking heads: Understanding inter-layer communication in transformer language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 61372–61418, 2024.

Clement Neo, Luke Ong, Philip H. S. Torr, Mor Geva, David Krueger, and Fazl Barez. Towards interpreting visual information processing in vision-language models. In *International Conference on Learning Representations*, 2025.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pp. 548–560. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.conll-1.37. URL https://aclanthology.org/2023.conll-1.37/.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *International Conference on Learning Representations*, 2024.

Nikhil Prakash, Natalie Shapira, Arnab Sen Sharma, Christoph Riedl, Yonatan Belinkov, Tamar Rott Shaham, David Bau, and Atticus Geiger. Language models use lookbacks to track beliefs. In *International Conference on Learning Representations*, 2026.

Israfel Salazar, Stella Frank, Dan Oneata, Desmond Elliott, and Constanza Fierro. Pathways of visual information flow in vision-language models. *arXiv preprint arXiv:2607.03358*, 2026.

Darshana Saravanan, Makarand Tapaswi, and Vineet Gandhi. Investigating mechanisms for in-context vision language binding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2025.

Timothy Schaumlöffel, Martina G. Vilas, and Gemma Roig. Mechanisms of object localization in vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Lisa Schut, Yarin Gal, and Sebastian Farquhar. Do multilingual LLMs think in English? *arXiv preprint arXiv:2502.15603*, 2025.

Arnab Sen Sharma, Giordano Rogers, Natalie Shapira, and David Bau. LLMs process lists with general filter heads. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=iPFlJESrsh.

Zeyu Wang and Xinming Xu. Knowing isn't always saying: When do spatial encodings reach answers in vision-language models? *arXiv preprint arXiv:2608.22916*, 2026. Accepted to EMNLP 2026.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in English? on the latent language of multilingual transformers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15366–15394, 2024.

Zhaofeng Wu, Xinyan Velocity Yu, Dani Yogatama, Jiasen Lu, and Yoon Kim. The semantic hub hypothesis: Language models share semantic representations across languages and modalities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=FrFQpAgnGE.

Tianyi Lorena Yan and Robin Jia. Promote, suppress, iterate: How language models answer one-to-many factual queries. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16111–16134. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.815. URL https://aclanthology.org/2025.emnlp-main.815/.

Zihao Yue, Liang Zhang, and Qin Jin. Less is more: Mitigating multimodal hallucination from an EOS decision perspective. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11766–11781. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.633. URL https://aclanthology.org/2024.acl-long.633/.

Hongbin Zhang, Kehai Chen, Xuefeng Bai, Xiucheng Li, Yang Xiang, and Min Zhang. Exploring the translation mechanism of large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 106539–106579, 2025a. doi: 10.52202/085713-3554.

Zhi Zhang, Srishti Yadav, Fengze Han, and Ekaterina Shutova. Cross-modal information flow in multimodal large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b.

A Implementation Details

This appendix gives everything the experiments of the main paper share: the model and its settings, the images and prompts of the two tasks, the pairs and splits, how an edit or a cut is run, how heads are chosen, and how reliable the model is with no edit.

A.1 Model and Settings

All experiments use Qwen3-VL-8B-Instruct at revision `0c351dd0`, loaded in 32-bit floating point with the eager attention implementation, so that every attention weight and every intermediate state can be read and edited. TensorFloat-32 is turned off. Decoding is greedy. The language model has 36 layers, a hidden size of 4,096, and 32 query heads of width 128 per layer, with 8 key-value groups, so four query heads share each group of keys and values. No weight is ever changed. Every edit is made during a forward pass. The vision encoder runs once per image, and its output is cached and never edited.

For each run we cache the keys and values of every token up to the last image token. The text tokens after the image are then recomputed in one forward pass on top of this cache, and every edit is applied inside that pass. With no edit, this pass reproduces the letter scores of an ordinary forward pass to within 7×10^{-5}. Several edited copies of one run are processed as one batch.

A.2 The Task With One Answer

Images. Each image is a strip of 1,536 by 384 pixels holding four square panels of 384 pixels, each with a 3-pixel gray border on a white background. Each panel holds one marker: a shape outline with a 6-pixel black stroke and white fill, about 106 pixels across (140 for the star, whose inside is smaller), with one bold capital letter in 48-point DejaVu Sans Bold at its center. The center of the marker is drawn uniformly inside the panel, at least 78 pixels from each edge, and the donor and the receiver of a pair share marker positions. The shapes are star, square, circle, triangle, diamond, plus, and heart, and each image uses four different shapes. Letters come from a pool of 21 capital letters (Appendix A.6). The image becomes 576 image tokens, a grid of 48 by 12, so each panel is 12 tokens wide.

Prompt. The user turn holds the image followed by the text `Answer with the letter only. What is the letter inside the {shape}?` There is no system prompt. After the chat template's assistant tag we append `Answer:`, so the prompt ends with the colon and the next token is the answer. Every shape name is one token, and every answer is the single token made of a space and the letter. The prompts of a donor and a receiver have the same length and differ at exactly one position, the shape word. The image tokens sit at positions 4 to 579, the shape word at position 593, and the colon at position 601. The tokens after the shape word are the question mark, the end-of-turn token, a newline, the assistant tag (two tokens), a newline, `Answer`, and the colon.

Pairs and splits. In each pair the donor and the receiver use the same four shapes in different orders, ask for different shapes, and use eight different letters. A pair is kept only if the five outcome letters of Section 3 are all different, in both directions. We use 200 donor-receiver pairs to rank heads (the discovery pairs), 100 pairs to fix the number of heads (the validation pairs), and 500 held-out pairs to report every result.

Scoring. After an edit we take the most likely next token over the full vocabulary and score it as one of the outcome letters or as a miss. Where a table also reports *Gaze*, it counts the cases in which the published Gaze Heads (Gandikota & Bau, 2026) look at the expected panel: for each of the top 100 heads of that ranking we sum its attention from the colon over the image tokens of each panel, add the sums over the 100 heads, and take the panel with the largest total. That ranking is never refit.

A.3 The Task With Many Answers

Images. Each image is one row of six panels of 384 by 384 pixels. Every panel holds one marker drawn as above but larger, so that the letter sits clear of the outline (the circle has a radius of 72

pixels and the star of 100). Marker centers are uniform in 106 to 278 pixels on both axes and are shared by all images of a pair.

Prompt. The user turn holds the image followed by `Answer with the letters only, separated by a comma and a space. What is the letter inside each {shape}?` The assistant turn starts with `Answer:`. The model writes `K, U, E`: a letter token, a comma token, a letter token. So the last comma chooses the next letter, and the last letter chooses between a comma and the end of the turn. Letters that we write for the model are ordinary tokens of the assistant turn. Throughout the appendix we write $r_1 < r_2 < r_3 < r_4$ for the receiver's four candidate panels.

Which shapes. We ask for stars, hearts, or triangles, which the model lists reliably; it does not list squares and pluses reliably. The other panels hold any of the remaining shapes except the ones the model confuses with the asked shape in a preliminary check (no triangle or plus when stars are asked, no diamond or heart when triangles are asked). In the same check, with the asked shape in five of six panels the model often listed the sixth panel too (27 of 96 images), so we use four.

Pairs. Each pair has one receiver and several donors that share the asked shape, the marker centers, and the written letter. The receiver has the asked shape at four places, which cycle over all 15 choices of four panels out of six. The main donor has the asked shape at r_2 and r_4 and the receiver's first letter written at r_2; all its other letters are new, so a donor letter in the answer is a copied read. For the first answer the same donor carries a new letter in place of the shared one. For stopping, a donor has the asked shape at r_1 only. The supporting experiments add donors with the asked shape at r_1 and r_3 (the moved set), at r_3 and r_4, and at r_1 and a panel that holds another shape in the receiver. The progress experiments of Section 5 draw their own pairs at random from the same shapes and letters, with the panels stated in each subsection, and keep a pair only if both unedited runs give the expected next letter.

Splits and counting. As in the single-answer task, we use 200 discovery pairs to rank heads, 100 validation pairs to fix the number of heads, and 500 held-out pairs to report results. The head sets of Sections 4 and 6 are always tested first, unchanged, and only then searched again. A case is counted when the unedited runs it needs give the right next token. Unedited, the receiver's second letter is right in 97.1% of pairs and the donor's in 100.0%, the first letters in 95.1% and 99.1%, and the one-place donor ends its turn in 95.7%.

Hiding a panel. To hide a panel from a token we remove that token's attention to the image tokens of the panel, in all heads of the chosen layers. The instruction tokens before the shape word also read the image, so "every text token" means every token after the image.

A.4 Edits, Cuts, and How Heads Are Chosen

All edits act while the receiver processes one token. Donor states are recorded from the donor's own unedited run at the same token position, which is possible because the two prompts align token by token. Everything that is not named stays the receiver's own, and all later computation runs normally on the edited state.

Residual swap. At layer l we replace the residual vector of the token with the donor's vector at the same boundary. Boundary l is the input to layer l, and boundary 36 is the input to the final normalization.

Query edit. For one head we replace the query vector of the token, after the query normalization and the rotary position encoding, with the donor's. The two runs share positions, so the position encoding is identical.

Key-read edit. For one head we recompute only its attention scores to the image tokens, using the receiver's query and the donor's image keys from the key-value group that head reads. Its scores to the text tokens, its values, and all other heads are unchanged. Because four query heads share each

group of keys, this edit changes what one head reads without changing the keys the other three heads see.

Value edit. For a head processing the colon or a comma, let α_s be its attention weight on source token s and v_s the value it reads there. We replace the head's output o with $o + \sum_{s \in S} \alpha_s (v_s^{\text{new}} - v_s)$, where S is the set of source tokens named in the experiment and v^{new} comes from the donor or from an induced run. The attention weights are the receiver's own. For the gaze heads the same edit is applied to the image tokens.

Output edit. We replace the output of a head at the token, taken before the layer's shared output projection, with the donor's. An MLP edit replaces the output of one MLP at the token.

Cuts. To cut a connection we set the attention score from a token, for the chosen heads, to the chosen source tokens, to the most negative value before the softmax. The head's remaining attention is renormalized by the softmax as usual. To hold a state at its own value we record it from the unedited receiver and write it back on top of the edit.

Choosing heads. For each attention edit we first apply the edit to one head at a time, for all 1,152 heads, on the ranking pairs. The score of a head is the change in the gap between the logit of the target letter and the logit of the receiver's own letter, averaged over cases and both directions. We then apply the edit to the top K heads together on the second set of pairs for K in 1, 2, 4, ..., 128 (256 for the gaze heads). The rule, fixed before these runs, takes the smallest K for which the edit moves the letter in at least 80% of cases, or the best K if none does; on the single-answer task the Gaze count must reach 80% as well. The chosen heads are then run once on the held-out pairs.

Controls. Every edit is repeated on a random set of heads of the same size, with each random head drawn from the same layer as the head it replaces and outside the selected set. Every cut is repeated on matched random heads or on a neighboring token.

A.5 The Language Tasks

Section 6.2 needs answers that differ between languages, which a letter does not. Images are strips of four panels as above. In the *colored-shape* task each panel holds one filled shape, all four shapes differ, and exactly one is cyan. The other three take distinct colors from orange, green, pink, yellow, red, and purple. The question asks for the shape of the cyan marker: in English `What shape is the cyan marker? Reply with one shape word only.`, in German `Welche Form hat die cyanfarbene Markierung? Antworte nur mit einem Wort für die Form.`, in French `Quelle est la forme du marqueur cyan ? Réponds avec un seul mot de forme.`, and in Chinese the direct translation. In the *object* task each panel holds a shape outline with a small drawn object inside it (a color emoji from the Noto Color Emoji font, 96 pixels), the asked shape is always the circle, and the questions are `What is inside the circle? Answer with one word only.` and its translations. No prompt names a language, and the assistant turn is empty, so the last prompt token is the newline after the assistant tag. It is the same token in every language and every image, and all edits act there or at generated tokens. Answers are generated greedily for up to eight tokens and matched against lists of accepted words per concept and language.

Objects. We started from 24 objects whose names differ in all four languages and kept an object only if the model named it correctly at all four panel positions in all four languages. Twenty-one passed; the crescent moon (read as a croissant in French), the top hat, and the sneaker failed. We use 16 of the 21.

Images, splits, and cases. Each task has 12 development images, used to rank heads and to compute the per-language averages and vectors, and 100 held-out images. Concepts and panel positions are balanced. Each held-out receiver is paired with three donor images of a different concept. Receivers ask in German, French, or Chinese and donors in any other language, which gives 2,700 cases per edit over nine ordered language pairs. The number of language heads, the layers of the held MLPs, and the boundary and scale of the language vector were fixed in an earlier study on

comic images (Appendix E.2) before any run on these tasks. An edit is applied at every generated token, and when it uses a donor the donor run is advanced in step by feeding it the receiver's generated tokens. To *set* a head to a language we overwrite its output with the average of its output at the last prompt token under that language over the 12 development images; nothing from the test image is involved.

A.6 How Reliable Is the Task?

Before any edit, we ask about every shape in every image, which gives 2,000 questions over 500 images. We read the answer in two ways: the next token after `Answer:`, and free generation with no prefix. The next token after `Answer:` is correct in 98.5% of the 2,000 questions, free generation with no prefix in 98.5%, and the published Gaze Heads look at the asked panel in 98.2%, with the final pool of 21 letters.

We first ran this check with all 26 letters. The next token was correct in 98.0% of 2,000 questions when ignoring letter case. Of the 41 errors, 25 were letters that do not appear in the image at all, led by Q read as O (8), J as A (4), L as I (3), and P as D (3), and the Gaze Heads were on the correct panel in 97.6% of them. The model was looking at the right place and misreading the letter. Only 0.8% of the 2,000 answers were the letter of another panel. We removed Q, O, I, L, and J from the letter pool for every experiment in this paper. On the language tasks, the model answers the unedited colored-shape questions in the question's language in 100.0% of runs and the object questions in 99.0%.

B Binding: Details and Supporting Experiments

This appendix supports Section 4: the residual swap at every token, the head screens behind the binding heads, the MLP that writes the request, what the binding heads add to the residual stream, the relay heads, and how the same heads behave on a list.

B.1 The Residual Swap at Every Token

Section 4.1 tracks the pointer by swapping the residual vector at every token and every layer. Figure 6 shows this map for all five outcome letters. Tokens before the shape word never change the answer. After the shape word, no single token changes the answer until the colon. The state that travels between the two is spread over several tokens, which is why Section 4 replaces the values of all of them together when it looks for the relay heads.

The marker token carries two things in turn (Figure 5). Up to layer 15 the swap moves the request: the receiver answers the donor's question on its own image, `P`. Around layer 20 it moves the place, `K`, in 39.0% of the 500 held-out pairs, and then the effect fades, because the relay heads have already read the marker by then. At the colon the place holds from layers 23-29, with its peak of 81.5% at layer 24, and from layer 30 the swap copies the donor's letter.

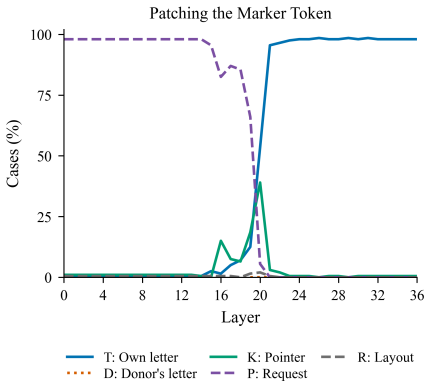


Figure 5: **At the marker token, the request comes before the place.** Swapping the marker token's residual at one layer moves the request, `P`, up to layer 15, and the pointer, `K`, at layer 20 (39.0% of 500 held-out pairs).

B.2 Head Screens and the Number of Heads

Figure 7 shows the single-head scores of the query, key-read, and output edits at the marker token over all 36 layers and 32 heads. The three edits pick out heads in the same band of middle layers, and many heads rank high in all three. Table 1 gives every edit on the held-out cases, Table 2 lists the selected heads, and Figure 8 reports every K on the pairs used to fix it. The key-read, output, and relay edits pass the rule at 16 heads and stay flat beyond it. The query edit passes at no K: eight heads reach the letter rate but fall short on the Gaze count, so the rule

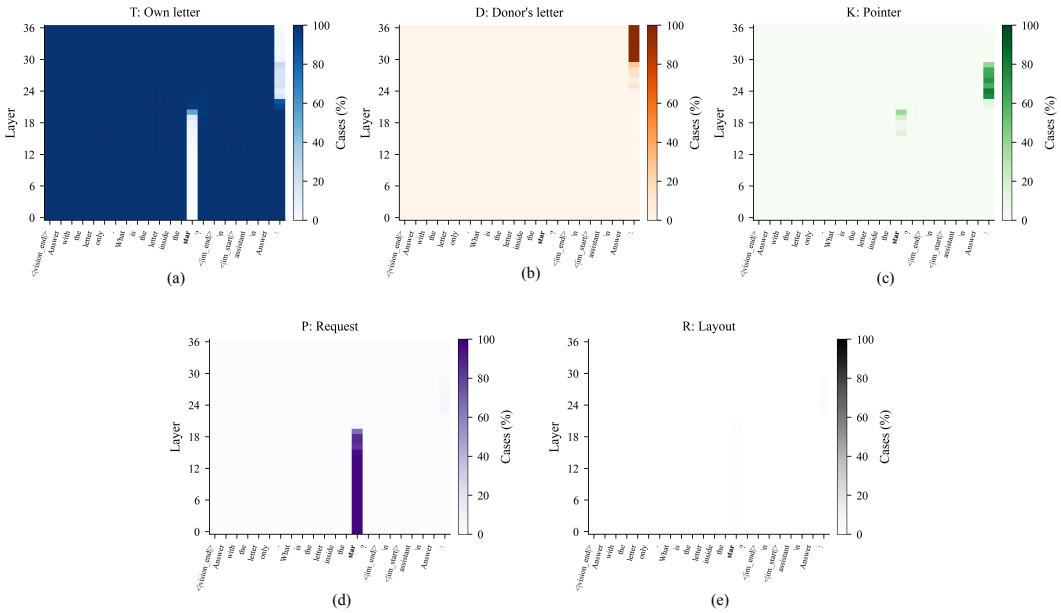


Figure 6: **Only the marker token and the colon token change the answer.** Each cell replaces the receiver's residual vector at one text token and one layer with the donor's, and counts the cases that answer with one outcome letter of Section 3: (a) `T`, the receiver's own letter; (b) `D`, the donor's letter; (c) `K`, the receiver's letter at the donor's panel; (d) `P`, the receiver's letter inside the donor's shape; (e) `R`, the receiver's letter where the donor's layout puts the asked shape. The marker token is shown in bold. The request, `P`, sits at the marker token up to layer 19; the pointer, `K`, sits at the marker token around layer 20 and at the colon token from layers 23-29; the donor's letter, `D`, comes through the colon token from layer 30. 500 held-out pairs.

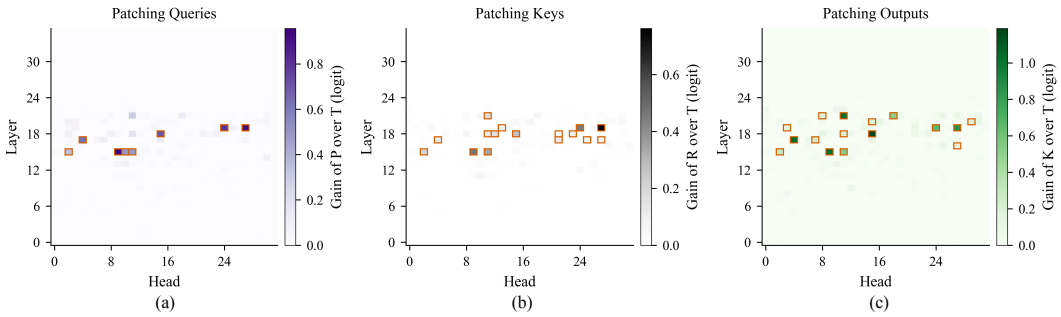


Figure 7: **Queries carry the request, keys carry the layout, outputs carry the pointer.** Each cell is one attention head at the marker token, patched from the donor on its own and averaged over the 200 discovery pairs. (a) The head's query is replaced with the donor's, scored as the gain of `P`, the receiver's letter inside the donor's shape, over `T`. (b) The head reads the donor's image keys, scored as the gain of `R`, the receiver's letter where the donor's layout puts the asked shape, over `T`. (c) The head's output is replaced with the donor's, scored as the gain of `K`, the pointer, over `T`. The heads each search selects are outlined: 8 query heads, 16 key heads, and 16 output heads, all in layers 15-21. The 16 output heads are the binding heads of the main text.

falls back on the best K, which is eight, and adding more heads makes the edit worse. The binding heads are the top 16 heads of the output search of Section 4.1. The query and key-read edits have their own searches, which select 8 query heads and 16 key heads, and each edit is reported with its own heads. The union of the three sets is 24 heads in layers 15-21, and the experiments that use this union, such as the rebind of Appendix C.3, say 24 binding heads.

Table 1: **Each edit moves one variable.** Every row copies one kind of state from the donor into the receiver at the marker token, or at the colon for the relay values. *Letter* is the percent of cases whose next token is the letter that the named variable predicts, and *Gaze* the percent in which the untouched published Gaze Heads move to the matching panel. 500 held-out pairs.

Edit (token)	Variable moved	Letter	Gaze
Queries of 8 heads (marker)	request	76.5%	75.0%
Image-key reads of 16 heads (marker)	layout	90.5%	88.0%
Both, 17 heads (marker)	place	94.0%	94.0%
Outputs of 16 heads (marker)	place	85.5%	78.5%
Values read by 16 heads (colon), induced run	place	87.5%	83.5%
Same, donor's own values	place	87.0%	88.0%
Same, receiver's values returned	none	99.5%	100.0%
Output of the first MLP (marker)	request	98.0%	100.0%
Strongest random-head control		0.5%	

Table 2: **Selected heads.** L and H give the layer and the head. The binding heads are the union of the first three rows.

Edit	Selected heads
Queries	L15H9, L19H27, L19H24, L18H15, L17H4, L15H10, L15H11, L15H2
Image-key reads	L19H27, L15H9, L19H24, L15H11, L18H15, L15H2, L21H11, L18H12, L17H4, L17H21, L18H11, L18H23, L19H13, L17H25, L18H21, L17H27
Head outputs	L18H15, L15H9, L17H4, L21H11, L19H27, L19H24, L15H11, L21H18, L15H2, L17H7, L20H29, L18H11, L20H15, L21H8, L16H27, L19H3
Values at the colon (relay heads)	L22H10, L35H25, L22H11, L23H26, L25H15, L19H21, L34H28, L23H0, L20H2, L20H3, L23H3, L25H12, L28H22, L21H19, L22H3, L22H0
Queries at the colon (gaze heads)	L24H29, L34H7, L23H4, L24H31, L29H5, L24H23, L26H26, L29H6, L22H0, L22H11, L21H18, L21H11, L33H31, L32H10, L29H11, L24H5, L24H30, L32H8, L28H0, L34H28, L23H30, L32H11, L27H18, L29H12, L21H8, L24H13, L30H6, L29H8, L20H15, L29H10, L21H27, L21H19
Queries, comic images	L15H2, L15H9, L15H11, L17H4, L18H15, L19H24, L19H27, L21H11
Image-key reads, comic images	L15H2, L15H9, L15H11, L17H4, L17H21, L18H15, L19H24, L19H27

Queries that carry the place. The query edit is the least clean of the three. With the 8 selected heads the receiver answers the donor's question on its own image in 76.5% of cases, but in 9.0% it answers at the donor's *location*. A query that moves the place must already say where to look, not only what to look for. The place forms inside the marker token while these heads run (Figure 5), and the queries of some of them carry part of it. Measured one head at a time, two heads in layer 19 (L19H24 and L19H27) move the request almost alone, while heads in layers 15, 17, 18, and 21 (L15H9, L15H10, L17H4, L18H15, L21H11) move the place about as much as the request. When heads are ranked instead by how much a query moves the request *minus* how much it moves the place, the edit needs 128 heads to pass, and then moves the request in 94.5% of cases and the place in 0.0%. Once the heads that mix in the place are ranked low, the query carries the request alone.

B.3 The First MLP Writes the Request

The queries of the query heads carry the request, so something must write the request into the marker token before the binding layers. We replace the output of one MLP at a time at the marker token, for all 36 layers. Only the first MLP moves the request: with its output taken from the donor, the receiver answers the donor's question in 98.0% of cases, and no other MLP moves it in more than 7.5% (Figure 9). The shape word is one token, and the first MLP is where that token's identity becomes a state the binding heads can use.

B.4 What the Binding Heads Write

The binding heads could add a code for the place, or the content at that place. We take each head set apart into queries, keys, and values, replacing one part of the heads' input from the donor at a time. When the 17 query and key heads keep their own queries and keys but read the donor's image

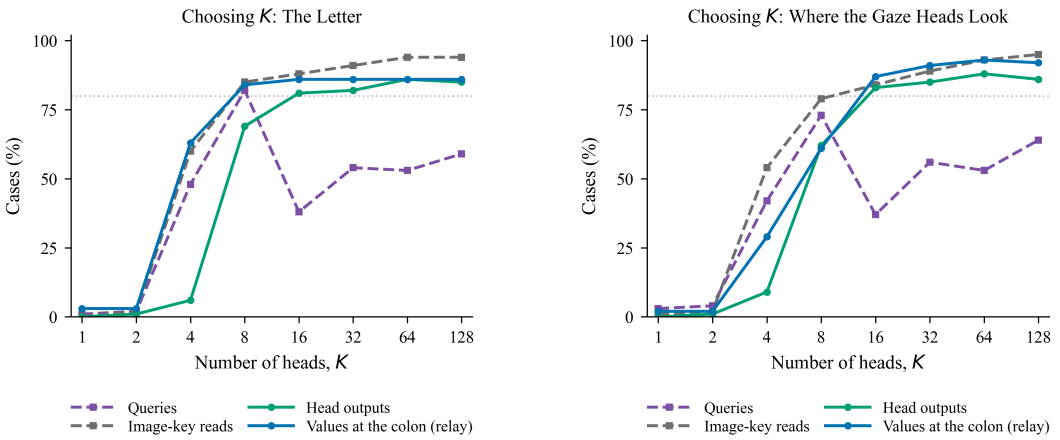


Figure 8: **Choosing the number of heads.** The top K heads of each single-head search are edited together on the 100 validation pairs: queries, image-key reads, and outputs at the marker token, and the values read at the colon (relay). (a) Cases whose next token is the letter the edit predicts. (b) Cases in which the published Gaze Heads look at the matching panel. The dotted line marks the 80% rule. The key reads, the outputs, and the relay values first pass it on both readouts at 16 heads; the query edit passes at no K, and the rule takes its best, 8 heads (82% and 73%).

values, the answer is unchanged in 95.0% of the 500 held-out pairs (the 16 binding heads: 94.5%), while queries, image keys, and values together move the place in 85.0%. Random heads change the answer in 0.0% of cases under any of these edits. What the binding heads add to the residual stream is a code for where, not for what is there. The gaze heads do the opposite: their image values carry the letter (Appendix D.3).

B.5 Relay: Which Tokens Carry the Place to the Colon

The relay heads carry the state made at the marker. To check that the relay heads move the place computed at the marker token, and not something else in the donor's prompt, we build an *induced* run: the receiver with only the binding heads' outputs replaced at the marker token, and nothing else. Its prompt, image, and every other state are the receiver's. When the relay heads of an unedited receiver read the values of this run, the place moves in 87.5% of cases, as often as with the donor's own values (87.0%). Giving the induced run the receiver's values back restores the receiver's answer in 99.5%. Figure 10a shows the single-head screen that found the 16 relay heads.

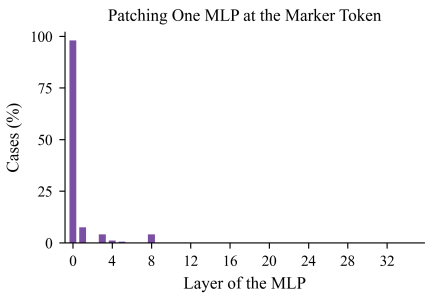


Figure 9: **The first MLP writes the request.** Replacing one MLP's output at the marker token moves the request, `P`, in 98.0% of 500 held-out pairs for the first MLP and in at most 7.5% for any other.

Which source tokens matter. The relay heads read values from every text token between the marker and the colon. Figure 11 installs the induced run's values from all sources but one, or from one source only. Leaving out the marker token costs the most, leaving out the question mark costs a little, and leaving out any other token costs nothing. No single token is enough on its own: the marker alone moves the place in 42.0% of cases and the question mark alone in 11.0%. So the relay heads read the place mostly from the marker itself and partly from the tokens after it (Figure 10b). On comic images with a longer prompt the question mark was the largest single source instead (Appendix E.1), so which token holds the copy depends on the prompt, while the relay heads do not.

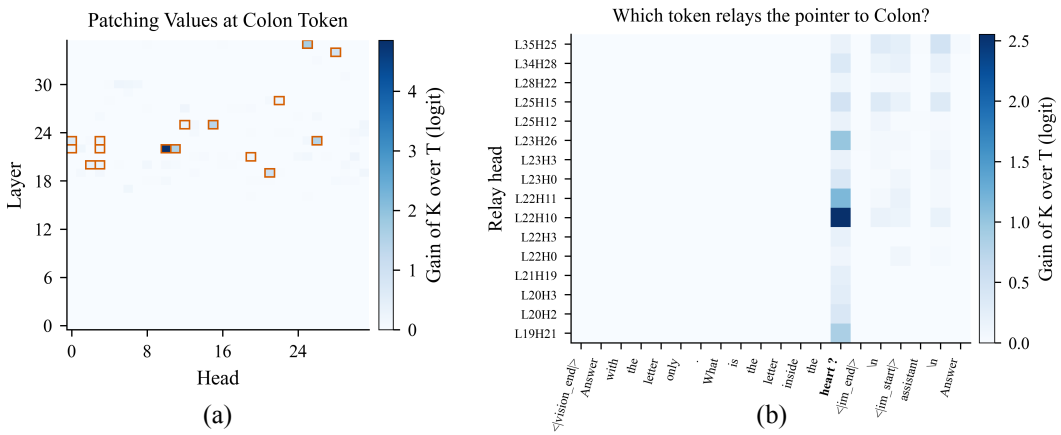


Figure 10: **Relay heads carry the pointer from the marker token to the colon token.** (a) At the colon token, one attention head at a time reads the donor's values from the marker token and the tokens after it, while its own query and keys stay unchanged. Each cell is the mean gain of K over T on the 200 discovery pairs. The 16 relay heads selected by the search are outlined, in layers 19-35. (b) Each relay head alone reads the donor's value at one text token only, with its query, keys, and attention unchanged. Each cell is the mean gain of K over T on the 500 held-out pairs, and the marker token and the "?" after it are shown as one column. With all 16 relay heads reading only the marker token the receiver answers K in 47.0% of cases, and reading only the "?" in 15.5%.

Each relay head, one source token at a time. Figure 11 uses the induced run. Figure 3a and Figure 10b use the donor's own values instead. At the colon, one relay head at a time reads the donor's value at a single text token after the image, while its query, keys, and attention weights stay the receiver's own, for every text token before the colon. Each cell is the mean gain of the logit of K over T, relative to the unedited receiver, on the 500 held-out pairs. The marker token and the question mark after it are drawn as one column, the sum of their two gains. With all 16 relay heads reading the donor's value at the marker token alone, the receiver answers K in 47.0% of cases, and at the question mark alone in 15.5%; at every other token it does so in at most 0.5%.

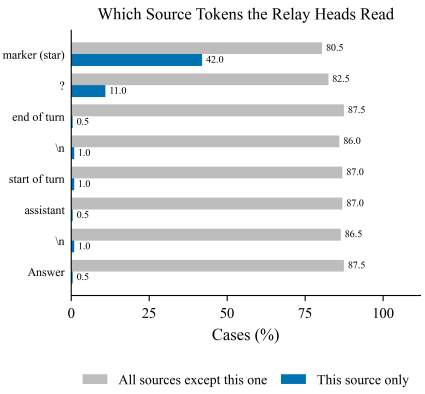


Figure 11: **The relay heads read the place mostly from the marker token.** The relay heads read the induced run's values from every source token but one (gray) or from one token only (blue). Percent of 500 held-out pairs whose answer moves to the donor's place.

B.6 Cuts: All Conditions

Table 3 reports every cut in the receiver's own run with its control. Three points add to the main text. First, the pointer reaches the colon by two routes. Cutting only the colon's direct view of the marker token leaves the answer in 98.0% of cases, and cutting only the view that the tokens in between have of it leaves 85.0%. Cutting both leaves 24.5%. Second, binding is spread over more heads than the smallest set that can move it. Blinding the 24 binding heads at the marker token leaves the answer in 77.0% of cases, and blinding every head there leaves 39.0%; when the marker and every later token are blind, 31.5% survive, because the instruction tokens before the marker also read the image. Third, when a cut breaks the pointer the wrong answers lean toward the leftmost panel: 37.9% of the wrong answers under the two pointer cuts are the letter of the leftmost panel, against one in four by chance.

Table 3: **All cuts, single-answer task.** *Letter* is the percent of cases that keep the receiver's correct answer. *Gaze* is the percent of cases in which the published Gaze Heads stay on the correct panel. 500 held-out pairs.

Connections removed	Letter	Gaze
No cut	98.0%	99.0%
The 16 key-read heads cannot see the image at the marker	80.5%	75.5%
same number of random heads	99.0%	99.0%
The 24 binding heads cannot see the image at the marker	77.0%	68.5%
No head can see the image at the marker	39.0%	41.0%
No head can see the image from the marker to the token before the colon	31.5%	34.5%
control: the text before the marker cannot see the image	98.0%	100.0%
Only the marker sees the image	4.5%	28.0%
Later tokens cannot read the marker	24.5%	27.5%
control: they cannot read the token before it	98.0%	99.0%
only the colon cannot read it	98.0%	99.5%
only the tokens in between cannot read it	85.0%	92.5%
Relay heads cannot read the text at the colon	8.0%	20.5%
same number of random heads	97.5%	99.0%

B.7 Relay Heads: Further Checks

Queries and keys route, values carry. Replacing the relay heads' queries at the colon with the donor's leaves the answer in 81.5% of cases, replacing the keys of the text they read in 97.5%, and both together in 97.0%, while the values alone move the place in 87.0%. Queries alone are the least clean of the three, because the late relay heads build their queries from a residual stream that already holds the pointer.

One place, not the set. On the list task the relay heads reading the donor's values at the colon move the first answer to the donor's first place in 78.5% of cases (random heads 0.0%). To ask whether the message is one place or the set of the donor's places, we then hide the donor's first place from every text token in every layer. If the set had traveled, the receiver would go on to the donor's other place. It does so in 8.4% of cases. It falls back to its own first place in 31.9%, goes to the place after the hidden one in 14.7%, and still reads the hidden place in 30.9%, since hiding a panel from the text is never complete. With every head reading the donor's values the counts are 48.7%, 22.5%, and 4.7%. The relay carries one place, the first.

The colon also binds on its own. On the list task the relay heads are not the only path. When they cannot read the text at the colon, the first answer survives in 73.3% of cases, against 8.0% on the single-answer task, because the colon of the list prompt binds the question to the image itself. With the marker token blind to the image the first answer survives in 72.8% of cases, and with the colon blind in 99.5%. Either token can bind the question on its own, and the marker's route is the stronger one.

B.8 Binding on a List

The binding heads find every candidate, and write the set flat. With four panels holding the asked shape, the binding heads at the marker token spread their image attention over all of them: 72% of it falls on the four candidate panels, and 26% of that on the panel the model will answer from first. We take the binding heads' summed contribution to the residual stream at the marker on the held-out pairs of the list task, fit a ridge probe on half of them, and read it on the other half. A probe for "panel p has the asked shape" reads each panel with 94% accuracy. A probe for "which panel is first" reads it in 100% of pairs, but a probe that sees only the six candidate scores already reads it in 96%, so the first place is the leftmost candidate and no separate signal. In the candidate directions, centered, the first candidate scores $+0.29$, the other candidates $+0.25$, and the fillers -0.53. The set is written flat, and Section 5.4 follows it to the answer token, where one place is chosen.

The binding heads' outputs set where a list starts. We use the binding heads' outputs to move a whole list. The donor asks about the same shape on another image with the asked shape at two places $d_1 < d_2$ that differ from the receiver's four, and letters that differ everywhere. At every generation step we copy the outputs of the 24 binding heads from the donor into the receiver at the marker token, at the colon, and at the last comma, edit nothing else, and let the receiver write its list freely, on the held-out pairs whose unedited lists are correct. The first letter moves to the receiver's letter at d_1 in 60.7% of pairs, and the marker alone moves it in 55.5%, so the place enters at the marker; the answer tokens edited alone leave the list unchanged in 86.7%, and random heads at all three tokens never give the donor's places. The rest of the list is decided again at each comma: only 21.4% of lists follow the donor's places to the end, and in 34.1% the second letter is the receiver's own next candidate after d_1, because the comma binds the receiver's own question again through more heads than the 24 we edit. The donor's image keys, read by the same heads at the same three tokens, move the list in 4.0% of pairs. So the binding heads' outputs set where a list starts, and each answer token decides again where it goes.

The token map on the list task. Swapping the receiver's residual vector at one token and one layer, for every token after the image, gives the same two tokens as the single-answer task for the first answer (the marker and the colon). For the second answer, a donor set swapped into the marker changes the next letter in 20.1% of cases, and the same set swapped into the comma in 78.4%, because the comma binds the question and re-reads the written letter from the image (Appendix C.6).

C Progress and Select: Details and Supporting Experiments

This appendix supports Section 5: the pairs of the progress experiments, the state the comma holds before the cursor forms, the heads that set the cursor at the label token, the two routes from the label token to the comma, the rule that select follows, how the comma finds the candidates again, what can be read from the comma's residual with no edit, and the searches for selection heads. Figure 12 draws the whole mechanism on a list with two answers.

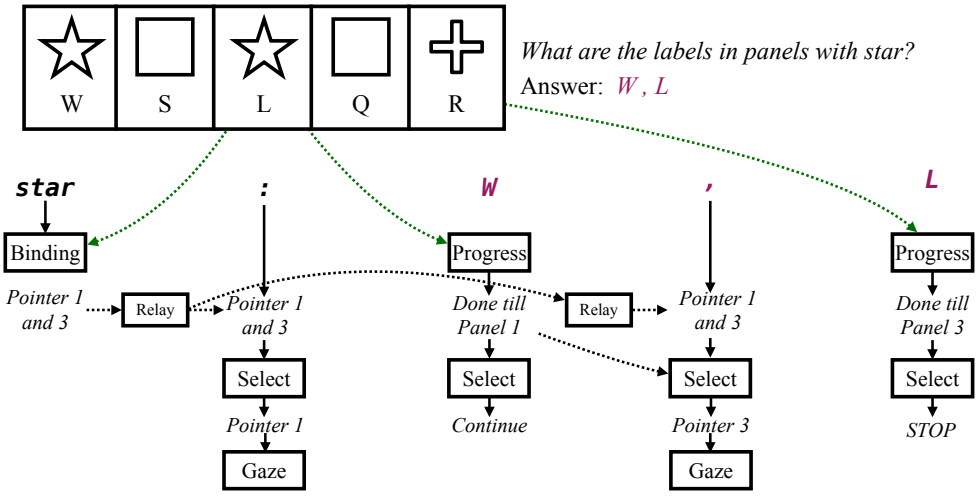


Figure 12: **The mechanism on a list with two answers.** The image has stars in panels 1 and 3, and the model answers `W, L`. At the marker token `star`, binding heads write pointers to panels 1 and 3, and relay heads carry them to the colon token. There, select picks panel 1 and gaze heads read its letter, `W`. At the label token `W`, progress records that the list is done up to panel 1, and a candidate is left, so the list continues. At the comma token, relay heads bring the pointers to panels 1 and 3 again, select uses the progress to pick panel 3, and gaze heads read `L`. At the label token `L`, progress is done up to panel 3 and no candidate is left, so the model stops.

C.1 Implementation Details

The three experiments of Sections 5.1 to 5.3 use their own pairs, drawn at random with the images and prompt of Appendix A.3. The donor and the receiver ask for different shapes drawn from star, heart, and triangle, the filler panels hold shapes the model does not confuse with the asked one, and the twelve letters of a pair are all different. The candidate panels are fixed by the design: in the skip experiment the donor's written panel is the receiver's first candidate, drawn from panels 1, 2, or 3; in the cursor experiment the donor has its shape at panels 2, 4, and 5 and has written panels 2 and 4, and the receiver has its shape at panels 1, 2, 3, and 6; in the rebind experiment the receiver has its shape at panels 1, 3, 4, and 6 and has written panels 1 and 3. A pair is kept only if both unedited runs give the expected next letter. Each experiment is run on 500 held-out pairs. The residual swap copies the donor's comma vector into the receiver's colon at one boundary, for every boundary from 0 to 36, and reads the receiver's next token. The rebind edit is the key-read edit of Appendix A.4 applied at a label token, with the donor image's keys.

C.2 The State Before the Cursor

Section 5.2 finds that a comma's residual copied into a colon makes the colon skip one candidate in the early layers, count the written letters in the middle layers, and start after the last written panel from layer 20. Three paper-scale runs show what the early state is.

The early state says "one letter was written", whichever letter it was. We take the held-out receivers of the list task and run each image twice more, with r_1 written and with r_2 written, then copy the residual of that comma at one boundary into the colon of the run with nothing written (Figure 13a). With the donor that wrote r_1 the colon says r_2 in 65.8% of cases already at boundary 14 and in 96.4% at boundary 21. With the donor that wrote r_2 the colon also says r_2 in 75.1% of cases at boundary 19, and r_3 in only 2.1%; from boundary 23 it says r_3 in 82.4%, rising to 92.7% at 32. Up to boundary 19 the copied state says that one letter has been written, and the colon skips its first candidate whichever letter that was. From boundary 20 it says which panel was written, and the colon starts after it.

The early skip is tied to panel 1. In the skip experiment of Section 5.1 the donor's written panel is the receiver's first candidate, and it is panel 1, 2, or 3. Split by that panel (Figure 13b), the receiver skips its first candidate in the early layers in 81% of pairs when the shared panel is panel 1, in 5% when it is panel 2, and in 3% when it is panel 3. At layers 20-21 the three groups skip in 80%, 82%, and 75%. So the early lean falls on panel 1, not on a count of candidates, and the cursor at layers 20-21 does not depend on which panel was written.

Count, then place, then the donor's choice. With the standard donor of Appendix A.3, which has the asked shape at r_2 and r_4 and has written the receiver's first letter at r_2, a count predicts r_2, the written panel predicts r_3, and the donor's own next place predicts r_4 (Figure 13c). The colon says r_2 in 86.5% at boundary 19, r_3 in 68.4% at boundary 20 and 53.9% at 21, r_4 in 77.7% at boundary 22 and 87.6% at 24, and the donor's letter in 99.5% at boundary 30. This is the order of Section 5.2, on a pair whose donor has written one letter.

C.3 The Heads That Set the Cursor

The label-token screens. We screen every head at the label token the same way as at the marker (Figure 14): for each of the 1,152 heads, on the discovery pairs of the list task, the gain in progress when the head reads the donor's image keys, the gain when its output is the donor's, and, in the receiver's own run, the gain toward repeating the written letter when the head is silenced. The heads that move progress sit in the layers of the binding heads, and the strongest of them are binding heads: 8 of the top 24 of the key-read screen and 8 of the top 24 of the output screen belong to the 24. Silencing single heads at the label token pushes the model toward writing the letter again; the top heads of that screen sit in layers 15-34, and 7 of its top 24 are binding heads. The same silencing screen at the comma finds only late heads, in layers 19-35, and none of them is a binding head: the comma reads the cursor and does not write it.

The key-read edit with one letter written. The 16 heads that the key-read screen selects at the label token move the next answer to the letter after the donor's panel in 74.7% of held-out cases, the

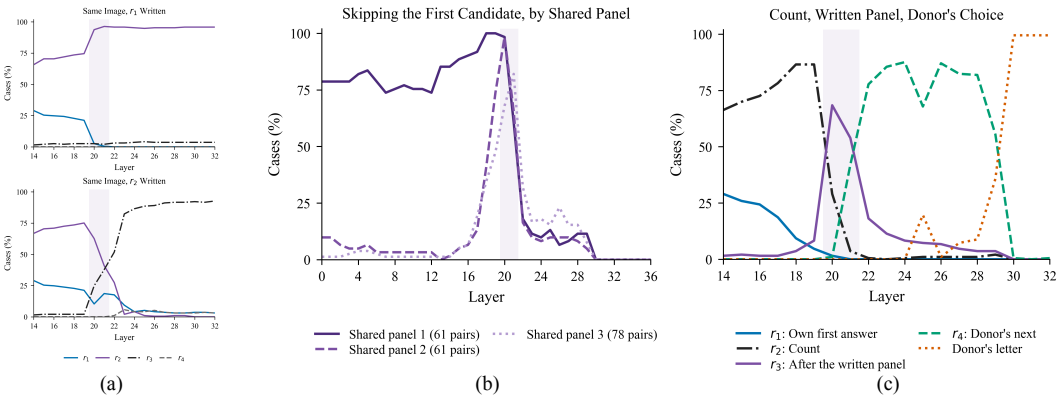


Figure 13: **The state the comma holds before the cursor forms.** The residual of a comma is copied into a colon that has written nothing, at one layer, and the colon's next letter is read. (a) The same image, with r_1 written (top) and with r_2 written (bottom), on the held-out receivers. With r_1 written the colon says r_2 in 65.8% of cases at layer 14 and 96.4% at layer 21. With r_2 written it also says r_2, in 75.1% at layer 19, and says r_3 from layer 23 (82.4%). Up to layer 19 the copied state says only that one letter was written; from layer 20 it starts after the written panel. (b) The skip experiment of Section 5.1, split by the shared first candidate. Averaged over layers 0-17, the receiver skips it in 81% of pairs when the shared panel is panel 1, 5% for panel 2, and 3% for panel 3; at layers 20-21 it skips in 80%, 82%, and 75%. (c) The standard donor, which has written the receiver's first letter at r_2 and has its own next candidate at r_4: a count, r_2, at layer 19 (86.5%), the panel after the written one, r_3, at layer 20 (68.4%), the donor's own choice, r_4, at layer 24 (87.6%), and the donor's letter from layer 30. The shaded band marks layers 20-21.

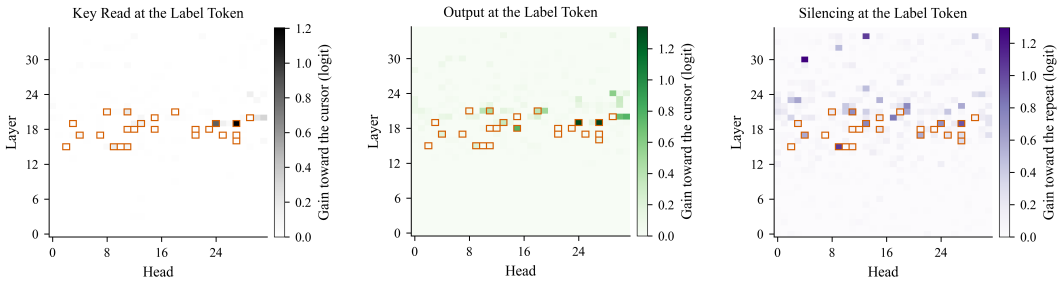


Figure 14: **The heads that set the cursor at the label token are binding heads.** One head at a time at the label token of the list task. (a) The head reads the donor's image keys and (b) the head's output is replaced with the donor's; each cell is the mean gain of the cursor's letter on the discovery pairs. (c) The head is silenced in the receiver's own run; each cell is the mean gain toward repeating the written letter on the discovery pairs. The 24 binding heads, the union of the three sets of Figure 7, are outlined. Eight of the top 24 heads of (a) and of (b) are binding heads.

24 binding heads of the single-answer task in 49.5%, and random heads in 0.0% (Table 4). Their outputs from the donor do the same in 48.5% with the 64 heads the output screen selects and 29.9% with the binding heads. Every new search on the list task finds mostly heads of the matching set of Sections 4 and 6: 18 of the 32 heads found at the marker token are binding heads, 10 of the 16 found at the colon are relay heads, 7 of the 16 found at the label token are binding heads, and 15 of the 16 found at the comma are gaze heads. The 64 heads found at the last letter mix 3 binding, 10 relay, 7 gaze, and 1 language head.

All arms of the rebind. Figure 15 gives every arm of Section 5.3. Rebinding the last label token to panel 4 moves the next answer past panel 4 with the 16 label-token heads (86.0%), the 24 binding heads (85.5%), and all 1,152 heads (68.0%), and never with 24 random heads from the binding heads' layers; the rest still answer from panel 4 (11.5%, 12.0%, and 30.0%). The receiver repeats a

Table 4: **Each edit on the list task, with the heads of Sections 4 and 6 and with the heads a new search finds.** Percent of the held-out list cases in which the edit moves the expected variable.

Edit	Heads of Sections 4 and 6, unchanged	Random heads	Heads found here (K)
First answer: values read at the colon	78.5%	0.0%	84.3% (16)
First answer: marker outputs	53.4%	0.5%	70.2% (32)
First answer: marker image key reads	9.9%	0.0%	11.0% (16)
First answer: colon queries	46.1%	0.0%	53.4% (128)
Label token: image key reads (cursor moves)	49.5%	0.5%	74.7% (16)
Label token: outputs (cursor moves)	29.9%	1.0%	48.5% (64)
Last comma: queries (next place moves)	88.7%	0.0%	84.0% (16)

written letter in at most 2.5% of pairs in any arm. Rebinding the first label token moves the answer in at most 3.0% and leaves it at panel 4 in 97.0% to 100.0%.

The cursor is a place, not a letter. If the cursor were the written letter rather than its panel, a second copy of that letter in the image would be skipped too. We put the written letter into r_3 as well, so that after "W, r_2," the next candidate holds W again. The model writes W again in 93.5% of lists, against 96.5% when r_3 holds a new letter. The letter adds only a weak pull; what is crossed off is the panel.

The label token needs nothing from the question. A label token that cannot read the marker token still sets the cursor: the next answer is unchanged in 96.9% of cases. When the comma cannot read the marker token, 72.7% survive, because the comma also uses the marker to find the candidates (Appendix C.6).

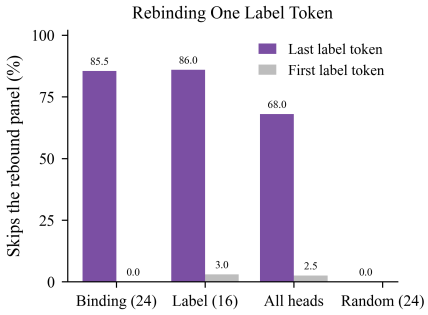


Figure 15: **The cursor is set by the last label token.** Pairs that skip the rebound panel, for each head set. The random heads were tested at the last label token only.

C.4 Two Routes From the Label Token to the Comma

The comma needs the cursor. When the last comma cannot read the label token, the model writes the letter again in 66.5% of cases, while blocking its read of the colon instead leaves 97.4% unchanged (Table 5). The comma can learn where the last letter came from in two ways: by reading the label token, which has bound the letter to its panel, or by finding the letter in the image itself. Either is enough on its own. With the label token blind to the image the next answer is kept in 88.7% of cases, with the comma blind in 96.9%, and with both blind in 17.0%.

The first route carries the cursor, and it is spread over many heads. Copying the donor's values of the label token, as read by every head at the comma, moves the next letter in only 17.0% of cases while the comma can still look at the image. Once the comma's own look at the image is cut, except through the gaze heads, the same edit moves the answer off the receiver's own place in 99.0%: to the donor's cursor in 40.7% and to the donor's own next place in 58.2%. Giving the comma the donor's image keys instead moves it in 10.8%. The comma reads the label token in two ranges of layers, and either is enough: with the read cut in layers 0-24 the answer survives in 96.9% of cases, and with the read cut in layers 20-35 in 96.9%. Cutting the 64 highest-ranked heads' read of the label token gives 16.5% repeats, and 64 random heads 0.0%. A single-head screen for this value edit, scored with the comma's own look cut, selects 8 heads that do not reach the 80% gate: on the held-out cases they move the cursor in 63.9% with the cut (random heads 11.9%) and in 5.7% without it. Two of the eight are relay heads and none is a gaze head. So no small set of heads relays the cursor; the pointer's relay heads carry none of it.

Table 5: **Cuts in the receiver's own run, second answer.** Percent of held-out cases with one letter written.

Cut	Own answer kept	Repeats the written letter
No cut	100.0%	0.0%
Marker token cannot see the image	67.5%	0.0%
Later tokens cannot read the marker token	65.5%	0.0%
control: later tokens cannot read the token before the marker	99.0%	0.0%
Relay heads cannot read the text at the comma	66.0%	0.0%
random heads cannot read the text at the comma	97.4%	0.0%
Label token cannot see the image	88.7%	4.6%
Comma cannot see the image	96.9%	0.0%
Label token and comma cannot see the image	17.0%	0.0%
Comma cannot read the label token	26.3%	66.5%
control: comma cannot read the colon	97.4%	0.0%
Only the marker, the label token, and the gaze heads at the comma see the image	83.5%	0.0%
Only the marker and the gaze heads at the comma see the image	42.3%	0.0%

C.5 Select: The Rule, the Count, and Stopping

Out of order. Figure 16 writes letters out of order, with no edit, and reads the next letter of the held-out receivers. Table 6 adds cuts. With every written letter visible the model says the next unsaid candidate after the last written one in every case. When the comma cannot read an earlier written letter, that place is written again in 43.9% of cases after "r_2, r_1" and in 51.1% after "r_3, r_2, r_1"; the rest still skip it, so part of the record is carried forward by the later label tokens. Select starts at the cursor and takes the nearest unwritten candidate, preferring the right, and every label token contributes to what counts as done.

Table 6: **Letters written out of order, with cuts.** Percent of receivers whose next letter comes from each place, when the comma cannot read one of the written letters.

Written so far	Cut at the comma	Next unsaid place	A place already said
r_1	no cut	100.0%	0.0%
r_2, r_1	no cut	100.0%	0.0%
r_2, r_1	comma cannot read r_2	53.2%	43.9%
r_2, r_1	comma cannot read r_1	73.4%	4.0%
r_3, r_2, r_1	no cut	100.0%	0.0%
r_3, r_2, r_1	comma cannot read r_3	47.3%	51.1%
r_3, r_2, r_1	comma cannot read r_2	72.3%	27.1%
r_3, r_2, r_1	comma cannot read any letter but the last	39.4%	59.0%
r_3, r_1	no cut	100.0%	0.0%
r_1, r_3	no cut	100.0%	0.0%
r_2	no cut	100.0%	0.0%

What looks like a count is the record of earlier places. A receiver that has written only r_3 says r_2 in 35.5% of cases and r_4 in most of the rest. The residual of the comma from a run on the same image that wrote r_1, r_2, r_3, copied at boundary 22, makes it say r_4 in 98.5% of cases, and the values of the third letter from that run, read by the relay heads, in 89.3% (all heads 81.2%). The last written token carries the earlier places because it read the earlier label tokens.

Stopping. At the last letter the model chooses between a comma and the end of its turn. A donor that has written its only candidate stops, and a receiver with candidates left goes on. The outputs of the 64 heads found at the last letter make a finished list go on in 100.0% of cases and a continuing list stop in 96.4%, against 1.0% and 0.0% for the same number of random heads, and the values they read do the latter in 81.3%. The binding heads' outputs from a run that still has candidates make a finished list go on in 95.9% of cases and the relay heads' values in 84.5%, while no small set of the earlier heads makes a continuing list stop. In the receiver's own run a finished list goes on when the later tokens cannot read the marker token (100.0%) and when the relay heads cannot read the text at the last letter (74.6%, random heads 1.0%), and a continuing list never stops under any cut we tried.

After it goes on, a finished list adds only letters of its own image. The 64 heads found at the comma for the next place (below) make a finished list go on in 99.5% of cases, but random heads do so in 29.5%, and make a continuing list stop in 33.7% against 0.0%. On the single-answer task, with the letter written after `Answer:`, the model ends its turn in 100.0% of runs under every cut, so there the instruction decides the stop; only the list task makes the model decide from the image and its own words.

C.6 The Candidates Are Found Again at the Comma

A donor with the asked shape at r_1 and r_3 and the same written letter at r_1 differs from the receiver only in the set of candidates, and a moved set makes the receiver skip r_2. Replacing the outputs of the 32 heads found at the marker token does this in 26.4% of cases, against 0.5% for random heads. It does it in 72.5% when the comma keeps only the gaze heads' view of the image, and in 96.9% when the label token and the comma also read the donor's image keys. The comma's fresh look usually wins. In the receiver's own run, cutting the comma's own view of the image keeps 86.5% of answers right, and cutting every later token's view of the marker token keeps 65.3%.

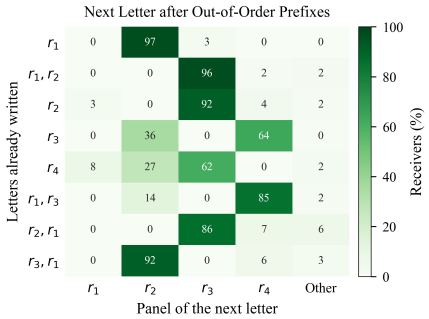


Figure 16: **Select starts at the last written panel.** Letters are written out of order, with no edit. Each row is a prefix of written letters, and each cell the percent of held-out receivers whose next letter comes from that candidate panel.

The fresh look already knows what is done. In the panel directions learned at the marker token (Appendix B.8), the binding heads' contribution at the label token scores the written panel at $+0.41$ and the next panel at -0.13, with the next below the written in 85% of held-out pairs: a written place has the sign of a candidate when it is recorded. The same directions applied to the comma's contribution score the written panel at -0.20 and the next at $+0.34$, with the written panel below the fillers in 76% of pairs and below the other candidates in 81%. The label token records the place, and the comma subtracts it.

C.7 Reading the Comma's State

Probes by boundary. On the held-out pairs of the list task we record the residual of the last comma at every boundary, fit a ridge probe on half of them, and read it on the other half (Figure 17a). "Panel p has the asked shape" is read per panel with 88% accuracy at boundary 17 and 97% at 21, from a chance level of two thirds. "Which panel was written" names the written place in 91% of pairs at boundary 12 and 100% at 18. "Which panel is next" names the next place in 85% at boundary 18 and 99% at 20, and in its centered scores the written panel sits below the other candidates at every boundary. The flat starts of the curves are the majority baselines, since the first places of a list are usually the leftmost panels. The colon behaves the same way with nothing written: its candidate probe peaks at boundaries 20-23.

Attention by layer. From the recorded attention of the same runs, the binding heads at the comma put 51% of their image attention on the next place at layer 21 and 4% on the written place, while at layers 15-19 their shares on the written place, the next place, and the other candidates are all small and alike. The gaze heads at layer 20 give 23% to the next place and 22% to each other candidate; by layer 23 the shares are 60% and 8% (Figure 17b). At the marker token the binding heads put 72% of their image attention on the candidate panels and 26% of that on the one chosen first; at the answer token the gaze heads put 44% of their attention to candidate panels on the chosen one for the first answer and 64% for the second.

Holding the comma's residual after the cursor is moved. On the held-out pairs with one letter written, the 16 label-token heads read the donor's image keys at the label token, which moves the cursor: the receiver says r_3 in 74.7% of cases (the 24 binding heads: 47.9%). We then hold the comma's residual at one boundary at the receiver's own values, on top of the edit (Figure 18). Held at boundary 19 or earlier, the edit keeps its effect in 72.2% of cases; held at boundary 23 or later,

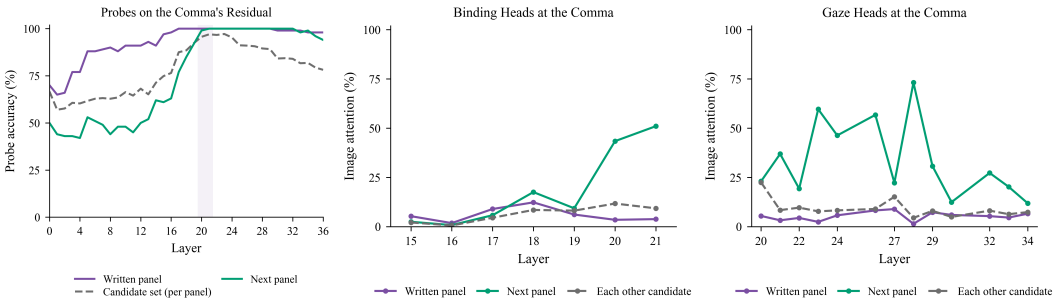


Figure 17: **What the comma's residual holds, with no edit.** (a) Ridge probes on the residual of the last comma, trained on half of the held-out pairs of the list task and read on the other half. The written panel is named in 91% of pairs at layer 12 and 100% at 18. Each panel's candidate status is read with 88% accuracy at layer 17 and 97% at 21, from a chance level of two thirds. The next panel is named in 85% at layer 18 and 99% at 20. (b) Image attention of the binding heads at the comma, by layer: at layer 21 they put 51% of it on the next panel and 4% on the written panel. (c) Image attention of the gaze heads at the comma, by layer: at layer 20 they give 23% to the next panel and 22% to each other candidate, and by layer 23 they give 60% and 8%.

the receiver says its own r_2 again in 97.4% (binding heads 99.0%). The hold alone changes nothing (100.0% own answers at every boundary tried), and holding the gaze heads' queries at the comma instead of the residual brings the answer back in 98.5%. So the cursor set at the label token acts through the comma's residual up to boundary 23 and through what the gaze heads read from it.

Hiding a panel from a layer on. Figure 19 hides the next candidate from every text token from layer l on. Hiding a panel that holds no candidate, from every text token except the marker token, moves the answer to the following candidate in at most 1.0% of cases at any layer. Three more results support its reading. Hiding the first candidate the same way makes the model start from r_2 in 36.1% of cases when it is hidden from the first layer, and in 8.9% when it is hidden from layer 22 on, with the fall in layers 16-22. The first place is harder to hide than the second, because it is the leftmost and its content also reaches the text through the image tokens of later panels. A copied pointer holds one place: we copy the colon state of a donor whose candidates are r_2 and r_4 into the receiver at boundary 24 and hide r_2 from the colon from that boundary on. The receiver goes back to its own first candidate in 75.3% of cases and to the donor's other candidate in 3.1%.

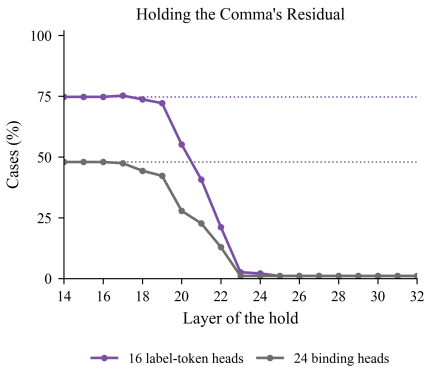


Figure 18: **The cursor acts through the comma's residual up to layer 23.** Pairs that still answer from the moved cursor when the comma's residual is held at one layer. Dotted lines: the edit alone.

C.8 The Searches for Selection Heads

We searched three times for heads that make the choice. The first search ranks every head in layers 20-25 by the effect of replacing its output at the last comma with the donor's, scored for the next place and for the cursor, on the discovery pairs, and sweeps the top sets on the validation pairs. No size reaches the 80% gate for either outcome; the best size for the next place, 64 heads, reaches 44% there. On the held-out pairs those heads move the next place in 61.9% (random heads 0.0%). Their queries move it in 40.7% and the cursor in 21.6%, their image keys in 0.0%, and their text values in 5.7%. Fourteen of the 64 are gaze heads, and with the gaze heads' queries held at the receiver's values the output edit moves nothing (0.0%), so these heads act by writing what the gaze heads read. Replacing the outputs of the gaze heads themselves copies the donor's letter in 100.0% of cases, which is the read and not the choice.

The two other searches need no donor. We rank the heads of layers 16-25 at the comma by the drop in the margin between the next letter and the best other candidate when the head's output is zeroed, and by the shift of the head's image attention from the other candidates onto the next place. The top heads of the two rankings overlap (L21H11, L20H29, L20H30, L21H18, L23H26 in both). On the held-out pairs, zeroing the top 4, 8, 16, 32, or 64 heads of either ranking leaves the next letter unchanged in 87.1% to 99.0% of cases; copying their outputs from the donor moves the next place in at most 33.5% (the top 64 by margin drop) against at most 0.5% for the same number of random heads, and with the gaze heads' queries held every such edit leaves the answer unchanged in 97.9% or more. Select is spread over many heads of layers 20-25, whose joint result the gaze heads read.

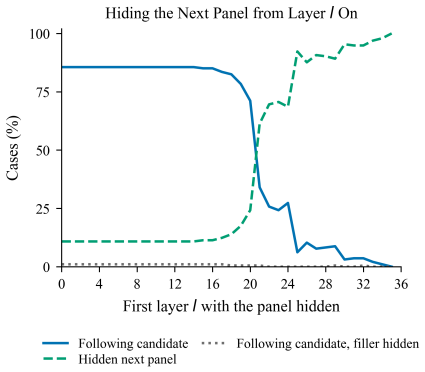


Figure 19: **The next panel is chosen in layers 20-25.** Lists that move on to the following candidate when the next panel is hidden from layer l on. Hiding a panel with no candidate stays at or below 1.0%.

D Reading and Language: Details and Supporting Experiments

This appendix supports Section 6: the two screens that looked for the heads that read the pointer, the comparison with the published Gaze Heads, the checks on the gaze heads' keys and values, the read on a list and the places it refuses, what the gaze heads bring back, the object task, and every language experiment.

D.1 The Two Screens, and Which Heads They Select

A head that reads the pointer must build its query from it, so the main text ranks every head by a query swap at the colon. We also ran a second screen: each head, one at a time, is blinded to every image token, and the score is the drop in the margin of the receiver's own letter over the best of the other letters in its image. Figure 20b reports both screens for every K on the pairs used to fix it. The swap screen rises steeply to eight heads and is flat from 16 to 256, so the part of the pointer's effect that it does not move is not a matter of missing heads. The blinding screen finds no set of heads whose blinding breaks the answer: even 256 blinded heads leave it intact in almost every case, and the heads it ranks highest have no effect in any held-out test (Table 7). The read at the colon has many possible routes, and removing a large share of them is not enough to stop it: blinding the 32 gaze heads still leaves the right answer in 86.5% of the 500 held-out pairs, and blinding the top 200 published Gaze Heads leaves 97.0%. When the usual gaze heads cannot see, other heads read in their place. The gaze heads are therefore sufficient, and the pointer acts only through their queries, but they are not the only heads that can read.

Table 7: **The heads that read the pointer, found by patching, against the published Gaze Heads and the blinding screen.** All edits act at the colon only. Percent of the 500 held-out pairs. The published ranking played no part in the search.

Head set	Queries swapped: donor's place	Pointer moved, queries held: own kept	Donor's image values: donor's letter	Blinded: own lost	Only these heads see: own kept
32 gaze heads found by patching	79.0%	98.0%	94.0%	13.5%	96.0%
same number of random heads	0.5%	7.0%	0.0%	1.5%	3.5%
Published Gaze ranking, top 100	66.5%	94.5%	51.0%	3.5%	96.0%
Published Gaze ranking, top 200	65.5%	96.0%	88.5%	3.0%	96.5%
Heads from the blinding screen	7.5%	10.5%	0.0%	3.5%	–

D.2 Our Gaze Heads and the Published Gaze Heads

Figure 20a compares three rankings of all 1,152 heads: the swap screen, the Gaze score of Gandikota & Bau (2026) measured on our task (the attention a head places, from the colon, on the image tokens of the asked panel, averaged over 2,000 questions), and the original Gaze ranking from comic strips. The patched gaze heads agree best with the Gaze score measured on the same task. The original and re-measured rankings share 57 of their top 100 heads, so the ranking itself shifts somewhat with the task, and 24 of the 32 gaze heads are in the original top 100.

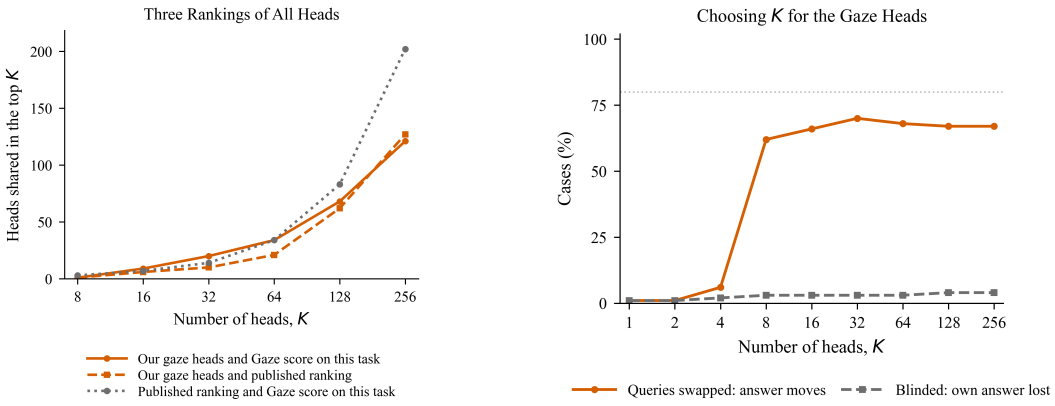


Figure 20: **Finding the gaze heads, and how they compare with the published ranking.** (a) Heads shared in the top K of three rankings of all 1,152 heads: our query-swap screen, the Gaze score measured on our task, and the published Gaze Heads ranking. From $K = 16$ to 128 our ranking shares more heads with the Gaze score on this task than with the published ranking (20 against 10 in the top 32). (b) The top K heads of the two screens at the colon, edited together on the 100 validation pairs. Swapping the queries moves the answer to the donor's place in 62% of cases at $K = 8$ and in 66% to 70% from 16 to 256 heads, below the 80% rule, so the rule takes its best, 32 heads. Blinding up to 256 heads loses the receiver's answer in at most 4%.

Effect against the number of heads. Figure 4b edits the top K heads of three rankings of all 1,152 heads together at the colon, for K in 1, 2, 4, 8, 16, 32, 64, 100, 128, 200, and 256, on the 500 held-out pairs. The rankings are ours (the swap screen), the published Gaze Heads ranking of Gandikota & Bau (2026), used as published, and a random order of all heads, drawn anew for every pair and direction. Each K gets two edits in separate runs. In the query edit the heads' queries are replaced with the donor's, and a case counts when the receiver answers with its own letter at the donor's panel. In the value edit the heads keep their own queries and keys and read the donor's image values, and a case counts when the receiver answers with the donor's letter at the receiver's asked panel. Our 32 gaze heads reach 79.0% with the query edit and 94.0% with the value edit; the top 100 published heads reach 66.5% and 51.0%, and the top 256 reach 65.0% and 89.0%. Random heads stay at or below 1.5% at every K.

Every head set against the published Gaze Heads. The published list was made by watching where heads look while the model describes comic strips, with no edit and no task of ours. Table 8 counts how many heads of each set in this paper are in it. The published list is close to a superset of our gaze heads: 24 of the 32 are in its top 100 and 30 in its top 200. It holds almost none of the others. Of the 24 binding heads, 3 are in the top 100, which is what chance gives; of the 16 relay heads, 4; and of the 16 language heads, 0. The same holds for the searches on the list task: the heads found at the last comma, where the next place is read, are published Gaze Heads (11 of 16 in the top 100), and the heads found at the marker token and at the label token mostly are not. When the Gaze score is measured again on our task it ranks more binding heads highly (9 of 24 in its top 100), because on these images a binding head also looks at the shape that will be read. So the published Gaze Heads are the heads of one step, the read. Binding, relay, and the choice of language are done by other heads, which attention to the described region does not reveal.

D.3 Gaze Heads: Keys, Values, and the Read on a List

Keys match, values carry. Gaze heads reading the donor's image keys with their own queries and values leave the answer in 91.0% of the 500 held-out pairs; the donor's image values give the donor's letter at the receiver's place in 94.0%, and keys and values together in 91.5%. Their queries carry the pointer, their keys on the image tokens match it by place, and their values bring the letter.

Table 8: **The published Gaze Heads are the heads of the read step.** Number of heads of each set that are in the published ranking of Gaze Heads and in the Gaze score measured again on our task. The last column is the number expected if the set were drawn at random from the 1,152 heads.

Head set	Heads	Layers	In published top 100	In published top 200	In re-measured top 100	Expected by chance in a top 100
Binding heads	24	15-21	3	7	9	2.1
Relay heads	16	19-35	4	5	3	1.4
Gaze heads	32	20-34	24	30	29	2.8
Language heads	16	19-35	0	0	0	1.4
List task: outputs at the marker token	32	14-21	6	12	14	2.8
List task: key reads at the label token	16	15-24	3	7	8	1.4
List task: queries at the last comma	16	21-35	11	15	14	1.4
List task: outputs at the last letter (stop)	64	18-35	11	17	13	5.6

The same gaze heads read every answer of a list. In the list task the pointer to the next place forms at the last comma, and the 32 gaze heads, unchanged, read it. Their queries from the donor move the second answer to the donor's next place in 88.7% of cases (random heads 0.0%), and with their queries held at the receiver's values a moved cursor changes the answer in only 2.1%. At the colon of the list task their queries move the first answer in 46.1% of cases with the 32 heads and 53.4% with the 128 heads a new search selects there, because the colon of the list prompt has more than one place to choose from (Table 4).

The read refuses a written place, and a wrong shape. We point the gaze heads, with the donor's queries, at a place the receiver should not read. When the place holds the asked shape and is not yet written, the answer goes there in 88.7% of cases. When it holds another shape, it goes there in 27.6% of cases, and in 87.3% when the gaze heads also read the donor's image keys for that panel, so the shape check runs through the image keys the gaze heads see. When the place is already written, the answer goes there in 0.0% of cases, and the donor's keys do not change this (16.1%). The refusal of a written place goes through the label token: when the comma cannot read that label token, the gaze heads are followed in 94.6% of these cases, and with the donor's image keys as well in 100.0%. When the label token itself cannot see the image the refusal stands (5.4%), because the comma can bind the letter to the image by itself (Appendix C.4). The cursor does not only move the choice; it vetoes a read.

D.4 What the Gaze Heads Bring Back

Setup. We use the 32 gaze heads unchanged, at the last prompt token of the language tasks, which is the token that produces the first answer word. Their *summed output* is the sum over these heads of the head's output after the output projection. We read it against the vocabulary by multiplying it with the output embedding, with no normalization. For the patch, the receiver asks in language B about image Y and the donor asks in language $A \neq B$ about one of three images X with a different concept. We replace the outputs of a head set with the donor's at every generated token, with the donor run advanced on the receiver's tokens. The answer is scored as the donor's concept in the receiver's language, the receiver's concept in the donor's language, the donor's word, or no change, on the 2,700 cases whose receiver asks in German, French, or Chinese.

Are the gaze heads the same heads on this task? The gaze heads were found on the letter task, with another prompt and another answer token. We therefore repeat a single-head search on the 12 development images of the colored-shape task: one head's output is replaced with the donor's, and the score is the gain of the donor's concept word over the receiver's, both in the receiver's language. This search ranks 14 of the 32 gaze heads in its top 32 and 26 in its top 128.

Results. The summed output of the gaze heads is nearly the same for one image under two languages (cosine 0.95) and differs between two images under one language (0.50). Its top word against the output vocabulary is the English name of the shape in 78.0% of 100 images when the question is German and in 67.0% when it is French, and otherwise the Chinese name; it is never the German or French name. Replacing the outputs of the 32 gaze heads with the donor's makes the receiver name the donor's shape in its own language in 81.7% of the 2,700 cases, change the language in 0.0%, and keep its answer in 14.9%. The 16 language heads move the language in 100.0% and the shape in 0.0%, and 32 random heads move the shape in 0.0% and the language in 0.1%. The gaze heads move the concept and never the language, and the language heads of Section 6.2 do the opposite.

The top 32 heads of the search on this task move the concept in every case, so the gaze heads found on the letter task are most, but not all, of what carries the concept here. With a donor in the receiver's own language the gaze heads move the concept in 86.6% of cases, and replacing them at the first generated token only moves it in 67.7%.

D.5 The Object Task

With four shapes, a readout that names the shape could be a coincidence of a small vocabulary. The object task has sixteen concepts, and the model answers its unedited questions in the right language with the right object in 99.0% of runs. The first two measurements hold for the same 32 gaze heads. Their summed output has cosine 0.96 for one image under two languages and 0.36 for two images under one language. Over 400 readouts, its top word is the English name of the object in the circle in 195, the start of that English name in 132, the Chinese name in 8, and the German or French name in 0. The 16 language heads move the language in 87.6% of cases and the object in 0.0%, so the separation of concept and language holds on both tasks.

What changes is who carries the concept. Replacing the outputs of the 32 gaze heads with those of a donor that asks in another language about another object makes the receiver name the donor's object in 9.8% of cases; in 15.5% the receiver names the object that its own image holds at the donor's asked panel, so part of what these heads pass on is still a place, which later heads use to read the image again. The heads that the same search finds on this task carry the object: replacing its top 32 heads moves the object in 86.1% of cases and the language in 0.0%, against 0.0% for random heads. That search ranks 14 of the 32 gaze heads in its top 32 and 28 in its top 128, and 21 of its top 32 heads are also in the top 32 of the search on the colored-shape task. A letter or a plain shape is read by the heads that read the pointer; a drawn object needs further heads after them.

D.6 Language Heads: All Conditions, Both Tasks

Table 9 reports every language experiment on both tasks. The two tasks select largely the same heads: 10 of the 16 are shared, and the two strongest heads are the same in both. The object task is the harder one for almost every edit; its answers are often several tokens long, and replacing the heads at the first generated token only gives 78.4% against 87.6% at every token. Holding the last three MLPs, layers 33-35, at the receiver's values leaves the language in place in both tasks (5.5% and 5.0% moved), holding MLPs 28-32 leaves 10.7% and 4.4%, while holding the other attention heads of layers 33-35 barely matters (91.3% and 79.7%). So the late MLPs, and not other late heads, write the word.

A single vector sets the language. The mean difference between the summed outputs of the 16 heads under a language and under English, added to the residual at boundary 24, does the same job as the per-head averages. On held-out images it switches English runs to French in 100.0% of cases, to Chinese in 100.0%, and to German in 75.0% on the colored-shape task; subtracted from German and French runs, it returns English in 100.0%. It also works on text-only prompts with no image, whose nouns never appear in any image: 65.0% to French, 100.0% to Chinese, and 55.0% to German.

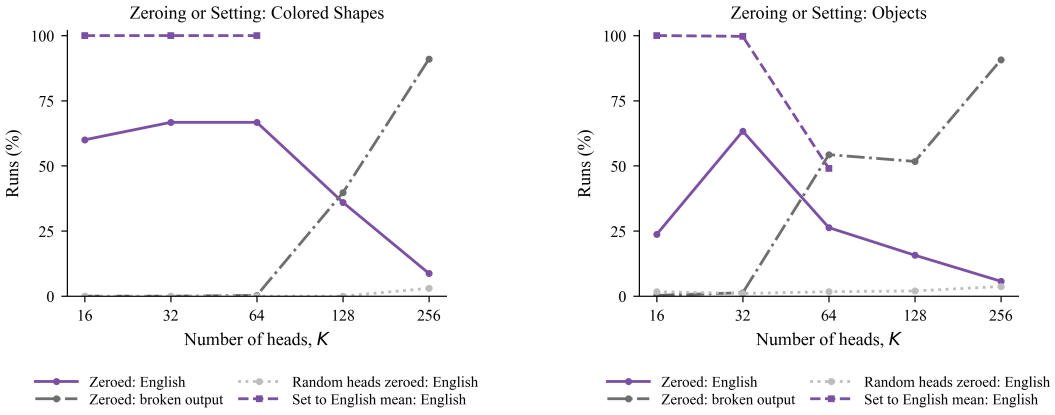
The late MLPs accept concepts they have not been shown. Adding the vocabulary direction of a new English noun to the residual at boundary 26 of a German, French, or Chinese run produces that noun in the question's language in 74.7% of cases on the colored-shape task and 67.0% on the object task, with the English word in 4.6%. The same direction added late, after the language heads have acted, gives the English word instead (69.2%) and the translated word in 3.9%. The translation step at the end of an answer takes any concept that reaches it.

D.7 Zeroing the Heads, and Setting Them

Figure 21 compares zeroing the language heads with setting them to their English averages. Setting 16 heads returns English in every run of both tasks. Zeroing depends on the language of the question. On the colored-shape task, zeroing 16 heads returns English in 98% of German runs, 82% of French runs, and 0% of Chinese runs. Zeroing 32 or 64 heads brings French to 100% and leaves Chinese at 0%, and zeroing more breaks the output. Random heads never return English. So for German and

Table 9: **Every language experiment on both tasks.** 100 held-out images and 2,700 cross-image cases per task. Success means the receiver names its own concept in the donor's or target language.

Experiment	Colored shapes	Objects in shapes
Answers in the question's language, no edit	100.0%	99.0%
Language heads replaced from the donor, every generated token	100.0%	87.6%
first generated token only	90.7%	78.4%
copied donor words	0.0%	1.3%
Random heads	0.0%	0.5%
Lowest-ranked heads	0.0%	0.3%
Donor in the same language: word unchanged	100.0%	91.1%
Heads replaced, MLPs 33-35 held	5.5%	5.0%
Heads replaced, MLPs 28-32 held	10.7%	4.4%
Heads replaced, other attention of layers 33-35 held	91.3%	79.7%
Heads set to their English mean (no donor): English	100.0%	100.0%
Heads set to another language's mean: that language	77.3%	85.2%
random heads given the same means	0.0%	0.0%
Heads zeroed: English	60.0%	23.7%
Language vector, held-out images: French / Chinese / German	100.0 / 100.0 / 75.0%	15.0 / 100.0 / 36.0%
Language vector, text-only prompts: French / Chinese / German	65.0 / 100.0 / 55.0%	0.0 / 100.0 / 7.1%
Vector subtracted from German and French runs: English	100.0%	100.0%
New noun added at boundary 26: named in the question's language	74.7%	67.0%



(a) (b)

Figure 21: **Zeroing the language heads, against setting them.** Runs asked in German, French, or Chinese on the 100 held-out images (300 runs), with the top K language-search heads edited at every generated token. Solid purple: the heads are zeroed and the answer turns English; dash-dot gray: zeroing breaks the output; dotted light gray: random heads zeroed; dashed purple: the heads are set to their English average and the answer turns English (run up to 64 heads). (a) Colored shapes: 16 heads set to English give English in 100.0% of runs, and 16 heads zeroed in 60.0%. (b) Objects: 100.0% and 23.7%.

French the heads are needed for the language, and English is what the model falls back to without them. Chinese runs stay Chinese with the heads silent, yet they switch to English, German, or French when the same heads are set to those averages (Figure 22). The heads can therefore overrule the language, and for Chinese some other part of the model also carries it. Leaving English is the hardest direction, and it needs 32 heads instead of 16.

What keeps Chinese runs Chinese. On the 12 development images we zero the 16 heads in a Chinese run and copy one more component from a run on the same image asked in English, so that the two runs differ only in language, and score how far the first answer token moves toward the English word. No single head moves it far. Copying the whole residual of the last prompt token closes most of the gap from boundary 4 on, so the token already knows the language of the conversation a few layers in, before any of the 16 heads act, and MLPs 22-35 each move the answer

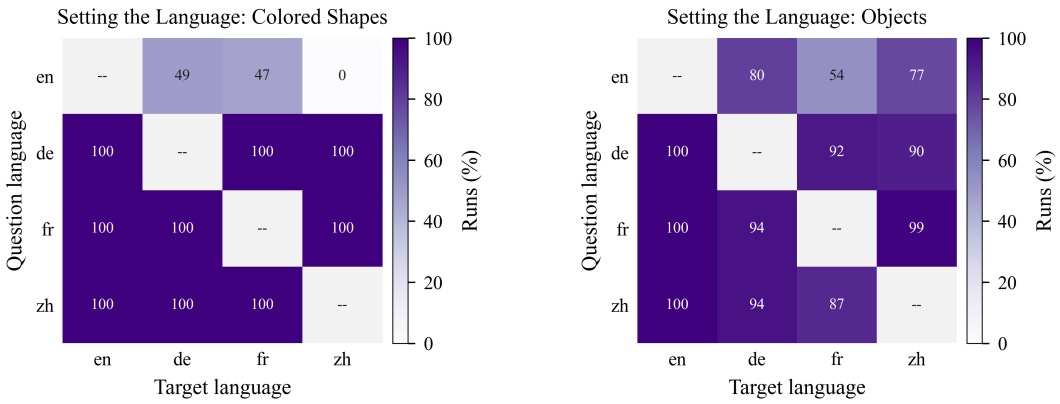


Figure 22: **Choosing the language with no donor.** The top 16 language heads are set to their average output under the target language (columns) in runs asked in the question language (rows). Each cell is the percent of the 100 held-out images whose answer names its own concept in the target language. (a) Colored shapes. (b) Objects. Leaving English is the hardest direction.

when copied. On held-out images (Table 10), zeroing the 16 heads together with the 64 heads that rank highest in this search returns English in 90% of Chinese runs, while the same 64 heads zeroed alone, or 64 random heads added to the 16, return English in 0%. Setting the early residual to its English average has the same effect: with the 16 heads silent it returns English, and alone it does not. So Chinese is carried twice, once by the 16 heads and once by an early trace in the residual stream that a wider set of heads and the late MLPs read. French has a weaker second copy, and German has none that we can detect.

Table 10: **What keeps Chinese runs Chinese.** Percent of runs that answer in English, colored-shape task, all 100 held-out images; edits to heads at every generated token and to the residual at the last prompt token. The extra heads are searched separately for each language.

Edit	German	French	Chinese
16 language heads zeroed	98	82	0
and 16 more heads found with those silent	100	100	0
and 64 more heads found with those silent	100	100	90
and 64 random heads	92	85	0
The 64 more heads zeroed alone	0	12	0
16 heads zeroed, residual at boundary 4 set to the English average	100	100	31
at boundary 8	100	100	89
at boundary 16	100	100	100
Residual at boundary 8 set to the English average alone	0	0	0

D.8 The Color Question: Language Heads and MLPs

We investigate further how the visual concepts retrieved by the gaze heads are converted into language, extending Section 6.2. We vary slightly the shape setting by having each image have four panels each containing a unique colored shape and without labels or text. In this setup, the model is instructed to describe the color of a specified shape {X}; the shape names replace the labels as the marker in the instruction prompt. The question, the shape name and the prefill are all written in one language, and the model answers with the color word in that language. For example, in French, (, heart_{fr}) prefilled "`Réponse :`" $\rightarrow$ `rouge`.

We document the prompts and color concepts used in each language in Table 11 and Table 12 respectively.

Table 11: **The task instruction in each language.** The user turn is the question followed by the single-word instruction, and the assistant turn is prefilled with the answer prefix.

	Question	Instruction	Prefill
English	What is the color of the {shape}?	Answer with a single word.	`Answer:`
French	Quelle est la couleur de la forme « {shape} » ?	Réponds par un seul mot.	`Réponse :`
Chinese	{shape}是什么颜色?	请用一个词回答。	答案:
German	Welche Farbe hat die Form „{shape}“?	Antworte mit einem einzigen Wort.	`Antwort:`

Table 12: **The eight color concepts in each language.** A readout counts the first token of the word; forms in parentheses are language equivalents.

Concept	English	French	Chinese	German
red	red	rouge	红色	rot
blue	blue	bleu (bleue)	蓝色	blau
green	green	vert (verte)	绿色	grün
yellow	yellow	jaune	黄色	gelb
purple	purple	violet (violette)	紫色	lila (violett)
black	black	noir (noire)	黑色	schwarz
pink	pink	rose	粉色 (粉红色, 粉)	rosa (pink)
gray	gray (grey)	gris (grise)	灰色 (灰)	grau

Donor: (, heart_{fr}) prefilled “`Réponse :`” → `rouge`

Receiver: (, $\text{diamond}_{\text{de}}$) prefilled “`Antwort:`” → `lila`

Language heads. We use causal mediation with a contrastive setup to measure the effect of individual components on the translation mechanism. A donor and receiver ask the question in two different languages about two different images. The two images show the same four shapes in different panels, colored with eight different colors, and the donor and the receiver ask about different shapes, so that the receiver's concept, the donor's concept, and the receiver's concept at the donor's panel are three different words. We use 160 contrastive pairs per language pair combination, balanced over markers, concepts and panels, and all 12 ordered pairs of English, French, Chinese and German. All edits are made at the colon of the prefill, and scores are computed over the logit distribution of the receiver's output with the edit.

Figure 24 shows the head search on this question.

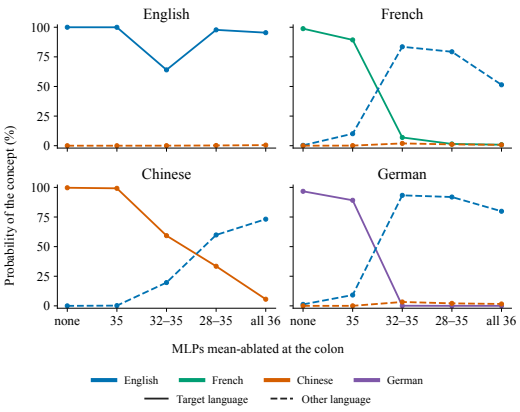
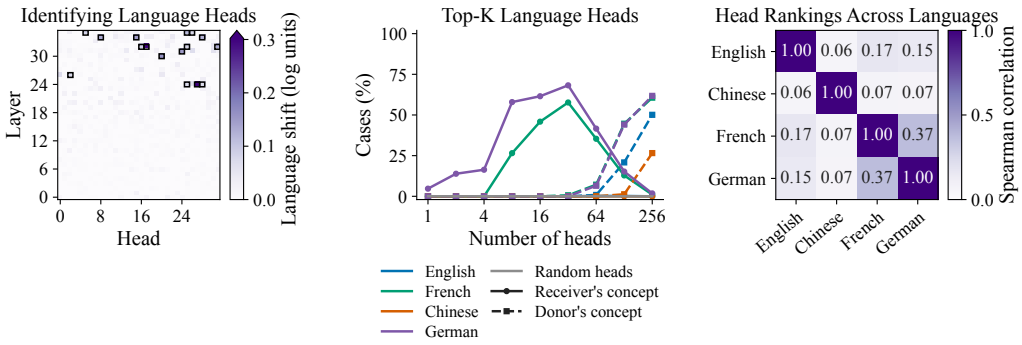


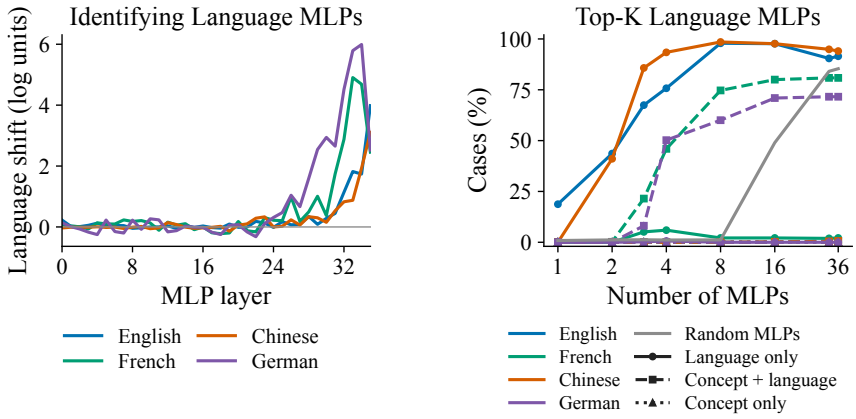
Figure 23: **Without the late MLPs, German and French answers fall back to English.** The outputs of the top MLPs at the colon are set to their mean over prompts in all languages, with no donor, and we read the probability of the correct color word in the target language, the language of the question (solid), and in English and Chinese (dashed). 21 prompts per language.

MLPs. We repeat the MLP search on the color question, replacing the output of one MLP at the colon with the donor's (Figure 25a) and then the top k MLPs together (Figure 25b). Ranked over all languages and both questions, the top 4 MLPs are layers 32-35 and the top 8 are layers 28-35. These late MLPs are also needed without any donor. Setting their outputs to their mean over prompts in all languages sends German and French answers back to English (Figure 23).



(a) (b) (c)

Figure 24: **Language heads on the color question.** (a) Language shift of each head, averaged over the 12 ordered pairs of English, French, Chinese and German: the output of one head at the colon is replaced with the donor's, and we measure how far the receiver's color words move from its own language toward the donor's (log units). The top 16 heads are outlined (b) The top k heads patched together, by receiver language. Heads are added in the order of a ranking over all languages. (c) Spearman correlation between the head rankings of each receiver language, over all 1,152 heads.



(a) (b)

Figure 25: **MLPs performing translation.** (a) Language shift when the output of one MLP at the colon is replaced with the donor's, by receiver language, averaged over the three donor languages. (b) The top k MLPs patched together, ranked over all languages and both questions (the top 4 are layers 32-35, the top 8 layers 28-35): grouped by donor language and averaged over its three receivers.

E Other Images and Prompts

Every image in the main paper is a row of panels with one prompt wording. This appendix asks whether the heads found there do the same job when the image or the prompt changes. In every experiment below the binding, relay, gaze, and language heads are used *exactly as found* on the panel images, with no new search unless we say so.

E.1 Comic Images With a Longer Prompt

We first ran the single-answer study on a different version of the task, and its held-out results remain valid as a test of robustness. There, each panel held a comic drawing under a white header that carried the marker, the letters used all 26 capitals, and the prompt was `Each panel has one shape marker containing one letter. What is`

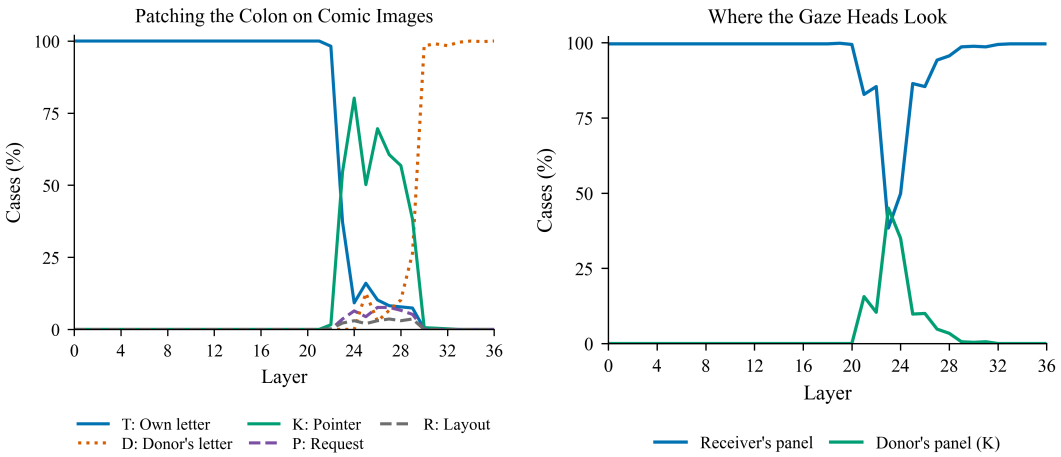


Figure 26: **The colon sweep on comic images.** The receiver's residual vector at the colon token is replaced with the donor's at one layer, as in Figure 2a, on 500 cases with comic drawings under the markers and the longer prompt. (a) The receiver's next letter: the pointer, `K`, peaks at 80.2% at layer 24, and the donor's letter, `D`, comes through from layer 30. (b) The panel the published Gaze Heads look at, with no head edited: at layer 23 they look at the donor's panel in 45.0% of cases.

`the letter inside the {shape} shape?` with the same `Answer:` prefix. Heads were ranked on 100 pairs, the number of heads was fixed on 50 pairs, and results are from the remaining 100 pairs (200 cases); the residual swap used all 500 cases.

The same steps appear. At the colon, the place outcome peaks at 80.2% at boundary 24 (Figure 26). At the marker token, 8 query heads move the request in 97.5% of cases, 8 key-read heads move the layout in 90.0%, and 32 output heads move the place in 85.5%, with the published Gaze Heads following in 99.0%, 88.0%, and 84.5%. Sixteen relay heads carry the place to the colon: values from the induced run move it in 85.5% of cases, the donor's own values in 91.5%, and restoring the receiver's values returns the receiver's answer in 100.0%. The first MLP moves the request in 99.0%. The strongest matched control reaches 0.0% for any attention edit. The heads selected on the comic images overlap heavily with the heads selected on the blank panels (Table 2), and the comic heads, used unchanged on the blank panels, still move each variable: the queries in 67.0% of 200 cases, the key reads in 81.0%, the outputs in 85.0%, and the relay values in 87.5%.

E.2 Language Heads on Comic Images

We first ran the language experiments with the colored markers drawn over comic strips, and the sizes used in Section 6.2 were fixed there. Those results, from 100 held-out images and 2,700 cases, agree with the plain-panel results. Replacing the 16 heads moved the language in 99.2% of cases with 0% copied words, random and lowest-ranked heads gave 0%, holding MLPs 33-35 left 0.3%, and the language vector switched 95%, 100%, and 49% of held-out images to French, Chinese, and German. Zeroing the heads returned English in 100% of German and French runs, and Chinese runs kept their language there too (Appendix D.7).

E.3 Images With No Panels

A reader may wonder whether the panels do some of the work: they give the model a small number of slots, and a place could simply be a slot number. This appendix removes them. The shapes sit at random places on one white square canvas with no borders (Figure 27), and we run the experiments of Sections 4 to 6 again with the head sets of the paper unchanged.

Implementation details. One-answer and language images are 768×768 pixels with four shapes, and list images are 960×960 with six. Both sizes give the same number of image tokens as the

matching panel image, or close to it. Centers are drawn at random with a minimum distance of 250 pixels, so shapes never touch. Shapes, letters, prompts, pairs, splits, and the outcome letters are those of the panel task. A donor and its receiver share the centers, as they share the panels in the main task. For the Gaze count, a place is the box around a shape with a margin of 24 pixels, and image tokens outside every box count for no place. The head sets are tested on the full held-out sets: 200 one-answer cases, of which the model answers 198 unedited, and 200 list pairs. The new searches reported below are pilots that rank heads on 20 pairs, fix their number on 10, and test on 20, and the language task uses 20 held-out images.

Order of a list. With no panels, "left to right" is no longer the order of a list. On 60 images with random centers the model names the right set of letters in 98.3% of them, but in left-to-right order in only 8.3%. It reads like text: rows from the top, and left to right inside a row. A simple rule, which takes the topmost remaining shape and everything within 160 pixels below it as a row, reproduces 79.7% of those lists. The list pairs therefore use only layouts on which this rule gives the same order for row heights of 60, 160, and 260 pixels, for the six shapes and for every subset of the four asked places. On such layouts the rule matches 100.0% of 24 lists, and r_1 to r_4 are the places in this order. With these pairs the unedited runs are as reliable as on panels (97.0%, 92.5%, and 98.0% of 200 held-out pairs usable for the next answer, the first answer, and stopping).

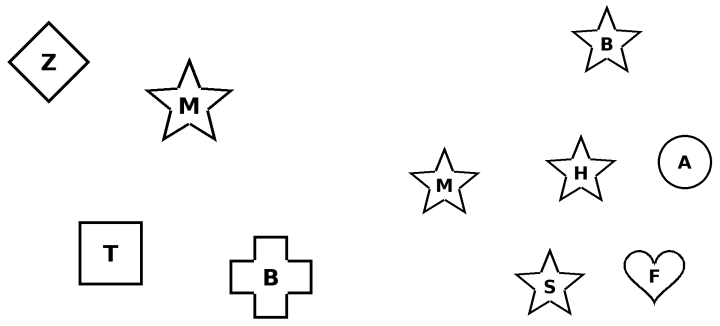


Figure 27: **Images with no panels.** Left: a one-answer image. Right: a list image in which the asked shape, a star, sits at four places.

The heads carry over unchanged. Figure 28 puts the two kinds of image side by side. Every edit of the one-answer task works with the heads of the paper, and most of them work better than on panels. The residual swap at the colon gives a place in 93.5% of cases, the binding heads' outputs move the place in 97.5%, the relay heads in 95.5%, and the gaze heads' queries in 90.0%, against 79.0% on panels. The gaze heads are still all the model needs to see with (98.0%), and a moved pointer still acts only through them (99.0%). The list task tells the same story: the gaze heads read the next place at the comma (86.6% of 194 cases), the relay heads deliver the first place (83.2% of 185), a comma that cannot read the label token repeats its letter (78.4%), the binding heads at the label token move the cursor (43.8%, against 49.5% on panels), and a run whose marker token is blind never stops (99.5% of 196). Gaze heads pointed at a written place are followed in 0.0% of 71 cases, and at a panel of another shape in 37.7% of 183. So a place is not a panel number. The same heads compute, carry, and read places on a continuous canvas.

A new search agrees. The pilot search for gaze heads on these images selects 16 heads, all 16 of them among the 32 gaze heads of the paper. On the language task the pilot search selects 16 heads, 12 of them the language heads of the paper. Replacing their outputs moves the language in 100.0% of 540 cases with 0.0% copied words (random heads 0.0%), holding the late MLPs leaves 8.3%, and setting the heads to the average of another language switches 77.8% of runs with 16 heads and 100.0% with 32. Patching the 32 gaze heads of the paper moves the concept in 71.5% of cases and the language in 0.0%.

What the panels give us. One tool of the paper needs the panels. Section 5.4 hides one panel from the text tokens and watches the model fall to the next one. On a blank canvas this does not hide

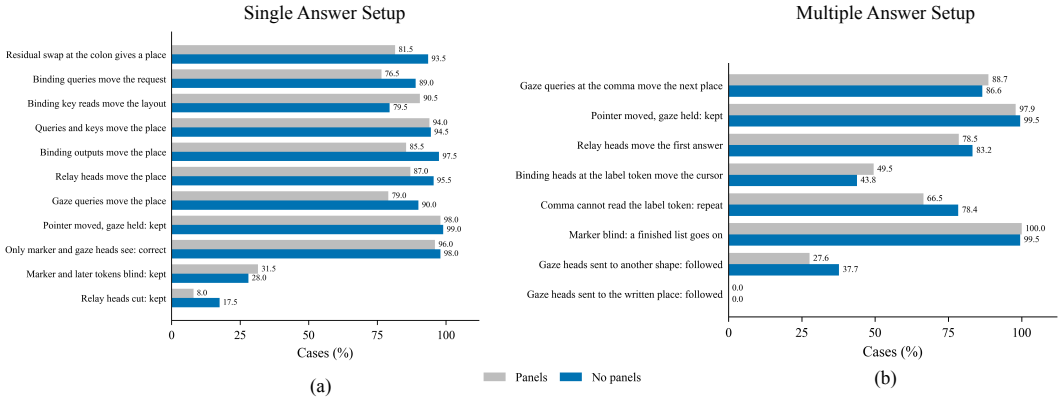


Figure 28: **The same heads, on images with no panels.** Every bar uses the head sets selected on panel images, with no new search. (a) The single-answer tests: the 500 held-out pairs on panels (gray) and 200 cases with no panels (blue). (b) The list tests, on the held-out list cases whose unedited runs are right. Random heads from the same layers move the answer in 0.0% of one-answer and list cases.

anything: with the box around r_2 hidden from every text token in every layer, the model still names the letter of r_2 in 100.0% of 19 pilot cases, and when every image token closer to r_2 than to any other shape is hidden, in 84.2%. The vision encoder spreads what a shape holds over image tokens far from it when nothing separates the shapes. A panel keeps a shape's content inside its own image tokens, which is what makes a place something we can hide.

E.4 Other Prompt Wordings and Layouts

We change the prompt or the layout, one at a time, and run the held-out batteries of Sections 4 to 6 with the heads of the paper. Two wordings: *prompt A* asks `Which letter is inside the {shape}? Reply with one letter only.` for one answer and `What are the letters inside every {shape}?` for a list; *prompt B* asks `Look at the {shape}. What letter is written in it? Answer with the letter only.` and `List the letter inside each {shape}.` Two layouts with the paper's prompt: a *grid*, four panels in a 2×2 grid of 768 by 768 pixels and six panels in a 2×3 grid of 1,152 by 768, read in row-major order; and a *column*, the panels stacked top to bottom (384 by 1,536 and 384 by 2,304 pixels). A fifth variant draws the four markers with a white fill and a black outline over a COCO photograph resized to 768 by 768, with a place defined as the 248-pixel box around a marker. The one-answer battery ran on 60 held-out pairs for prompt A and the photographs, 40 for the grid and the column, and 30 for prompt B; the list task ran on the same pairs, counting the cases whose unedited runs are right.

Table 13 gives the single-answer battery. The heads of the paper, found on the row images and one wording, move the place on every layout and on photographs in 85% to 98% of cases with the outputs, the joint query and key edit, and the relay values, against at most 1.2% for random heads, and the relay heads remain the necessary channel: cutting them keeps 10% to 20% of answers. Prompt B is the weak case. The model's own accuracy drops (86.7%), and every edit lands in 48% to 62% of cases, because the heads were chosen on a prompt in which the marker token sits last before the answer, and in prompt B it sits early. The photographs transfer best of all (97% to 98%).

Table 14 gives the gaze heads and the list task. On every variant the gaze heads' queries move the answer to the donor's place, a moved pointer is blocked when their queries are held, their image values bring the donor's letter, and they are all the model needs to see with, against at most 3.8% for random heads. On the lists, the relay values move the first answer, the binding heads at the label token move the cursor, the gaze heads' queries at the comma move the next answer, a comma that cannot read the label token repeats its letter (control 0.0%), and the label token and comma blind together lose the answer. The column reads as reliably as a row: with no edit the model gets the second letter right in 98.5% of 200 held-out pairs, the first in 99.0%, and stops in 86.0%.

Table 13: **The single-answer battery with the paper's heads on other prompts and layouts.** Percent of cases whose next token is the expected letter; the cuts give the percent of cases that keep the receiver's own answer. Cases are 2 × pairs: 120 for prompt A and the photographs, 60 for prompt B, and 80 for the grid and the column; the row task is the paper's held-out set.

Edit	Row, paper	Prompt A	Prompt B	Grid	Column	Photographs
Binding queries: donor's request	76.5%	73.3%	48.3%	67.5%	72.5%	90.8%
Binding image keys: donor's layout	90.5%	83.3%	61.7%	75.0%	75.0%	98.3%
Queries and keys: donor's place	94.0%	90.0%	56.7%	93.8%	92.5%	96.7%
Binding outputs: donor's place	85.5%	80.0%	53.3%	93.8%	85.0%	96.7%
Relay values from the induced run: donor's place	87.5%	79.2%	58.3%	93.8%	86.2%	98.3%
receiver's values restored: own answer	99.5%	99.2%	93.3%	98.8%	96.2%	100.0%
Random heads, any edit	0.5%	0.0%	0.0%	1.2%	1.2%	0.0%
No cut: own answer	98.0%	99.2%	86.7%	97.5%	97.5%	100.0%
Relay heads cannot read the text at the colon: own kept	8.0%	9.2%	10.0%	15.0%	21.2%	20.0%
Later tokens cannot read the marker: own kept	24.5%	30.8%	25.0%	25.0%	23.8%	18.3%
Binding heads blind at the marker: own kept	77.0%	89.2%	93.3%	77.5%	82.5%	78.3%
Gaze heads blind at the colon: own kept	86.5%	97.5%	91.7%	96.2%	95.0%	100.0%

Table 14: **Gaze heads and the list task with the paper's heads on other prompts and layouts.** Percent of cases. Top: the 32 gaze heads, with cases = 2 × pairs (120 for prompt A, 60 for prompt B, 80 for the grid and the column). Bottom: the list task, on the cases whose unedited runs are right; the row of valid cases gives these counts (first answer / second answer). A dash marks a test not run on that variant.

Test	Row, paper	Prompt A	Prompt B	Grid	Column
Queries swapped: donor's place	79.0%	83.3%	76.7%	82.5%	73.8%
Pointer moved, gaze queries held: own kept	98.0%	98.3%	93.3%	97.5%	96.2%
Donor's image values: donor's letter	94.0%	94.2%	95.0%	90.0%	90.0%
Only these heads see the image: own kept	96.0%	90.0%	91.7%	96.2%	96.2%
Random heads: queries swapped / only these see	0.5% / 3.5%	0.0% / 3.3%	0.0% / 1.7%	1.2% / 3.8%	0.0% / 3.8%
List task, valid cases (first / second)	191 / 194	58 / 58	– / 30	38 / 33	39 / 39
First answer: binding outputs at the marker: donor's first place	53.4%	58.6%	–	71.1%	51.3%
First answer: relay values at the colon: donor's first place	78.5%	93.1%	–	97.4%	76.9%
First answer: gaze queries at the colon: donor's first place	46.1%	53.4%	–	60.5%	51.3%
Second answer: binding heads at the label token read donor keys: cursor moves	49.5%	62.1%	56.7%	39.4%	38.5%
Second answer: binding outputs at the label token: cursor moves	29.9%	43.1%	36.7%	60.6%	35.9%
Second answer: gaze queries at the comma: donor's next place	88.7%	–	70.0%	–	82.1%
Random heads, every edit	≤1.0%	0.0%	≤3.3%	0.0%	≤2.6%
Cut: comma cannot read the label token: repeats	66.5%	72.4%	83.3%	72.7%	82.1%
control: comma cannot read the colon: repeats	0.0%	0.0%	0.0%	0.0%	0.0%
Relay heads cannot read the text at the comma: not own	34.0%	41.4%	43.3%	30.3%	46.2%
Label token and comma both blind: not own	83.0%	62.1%	70.0%	39.4%	56.4%

E.5 Natural Images and Natural Questions

The list task of the paper has one shape word and one panel per answer. Here the image is a photograph and the question names a category. We take COCO val2014 with its instance annotations and ask, after the image, `What animals are in this image? Answer with the names only, separated by a comma and a space.`, and the same for vehicles, for furniture (chair, couch, bed, dining table), and for objects (all 80 categories, on images with at least three present), with the assistant turn starting `Answer:`. A category counts if its boxes cover at least 1% of the image, images are resized to multiples of 32 pixels so that each image token is a 32 by 32 cell, and a category's region is the union of its boxes. We ran 304 animal, 300 vehicle, 300 furniture, and 400 object images. A case is valid when the model's first two items name two different annotated categories, A then B, and the rebuilt prefix `Answer: A,` gives B's first token again; 64.4% of the 1,304 images are valid (26.3% are lost because the first two items were not two annotated categories, 8.5% because the list had fewer than two items, 0.7% because a region was smaller than one token, and 0.1% because the rebuilt prefix changed the next token). The written answer A is now several tokens, and "the label token" means all of them.

Table 15 gives the results with the paper's heads. Progress on natural images is carried by the written word and read by the comma: when the comma cannot read the tokens of A, the model names A again in 79.6% of the 840 lists, and when it cannot read the colon instead, 98.0% are unchanged. The written word's own look at the image is not needed, because the comma finds the object again by itself, as in Appendix C.4: with A's tokens blind to their own region 94.2% of answers are kept,

Table 15: **Natural images and natural questions, with the paper's heads.** COCO val2014, 840 valid lists. A is the first item the model wrote and B the second. Each cell is the percent of valid lists in that group.

Test	Animals (222)	Vehicles (194)	Furniture (214)	Objects (210)	All (840)
Comma cannot read A's tokens: names A again	89.2%	70.6%	82.2%	75.2%	79.6%
control: comma cannot read the colon: unchanged	99.5%	99.5%	95.8%	97.1%	98.0%
A's tokens blind to A's region: unchanged	98.6%	95.9%	93.9%	88.1%	94.2%
A's tokens blind to the image: unchanged	93.7%	91.8%	89.7%	67.6%	85.7%
24 binding heads blind at A: unchanged	95.9%	94.3%	95.3%	74.8%	90.1%
24 random heads blind at A: unchanged	99.5%	100.0%	98.1%	94.3%	98.0%
Comma blind to the image: unchanged	73.0%	70.1%	91.1%	39.5%	68.6%
A's tokens and the comma both blind: unchanged	69.8%	67.5%	82.2%	31.0%	62.7%
Gaze heads at the comma: more attention per token on B than on A	95.5%	86.1%	76.6%	75.7%	83.6%
Binding heads at A: more attention per token on A than on B	69.4%	71.1%	86.4%	92.4%	79.9%

with them blind to the whole image 85.7%, with the 24 binding heads blind at A 90.1% (24 random heads 98.0%), with the comma blind 68.6%, and with A and the comma both blind 62.7%. The gaze heads at the comma look at the object named next: their attention per image token is higher on B's region than on A's in 83.6% of lists, by a median factor of 2.6, and the binding heads at A's last token look at A more than at B in 79.9%, by a median factor of 2.1. The language heads carry the output language here too: with a German question, the first token is German in 44.0% of 904 runs (the rest are names spelled the same in both languages, such as *Giraffe* or *Bus*), and setting the 16 language heads to their English averages turns 96.5% of those runs English, while 16 random heads set the same way leave 97.0% German. Hiding B's region from every text token does not remove B on a photograph (70% of lists are unchanged even when it is hidden from the first layer), for the reason given in Appendix E.3, so the layer window of select cannot be read off natural images with region masking.

F Model Sizes

The main paper studies the 8B model. Here we run the same pipeline on the 2B, 4B, and 32B models of the Qwen3-VL-Instruct family, with no head carried over: every head set is searched again on each model, with the rule of Appendix A.4, and tested on that model's own held-out pairs.

F.1 Implementation Details

The 2B language model has 28 layers of 16 query heads, with 2 query heads per key-value group; the 4B and 8B have 36 layers of 32 heads with 4 per group; the 32B has 64 layers of 64 heads with 8 per group. The images, prompts, pairs, splits, edits, and controls are those of Appendix A, and the layer bands of the language protocol scale with the depth of the model. The 2B, 4B, and 8B run in 32-bit floating point on one GPU. The 32B does not fit one GPU in 32-bit precision, so its weights are split across two 80 GB GPUs and its matrix products use TensorFloat-32, which changes the letter scores of an unedited run by less than the parity tolerance of Appendix A; everything else is unchanged. Because a 32B forward pass is about ten times slower, the 32B was run at development scale: 20 held-out pairs (40 cases) for the single-answer battery and the gaze heads, and 50 pairs for each progress experiment. The list task of Section 5.4 and the language tasks were not run on the 32B. With no edit, the next token after `Answer:` is right in 93.4% of 2,000 single-answer questions on the 2B, 96.7% on the 4B, and 98.5% on the 8B; on the list task both runs of a pair are right on the second letter in 48.0% of held-out pairs on the 2B and 92.5% on the 4B (8B: 97.1%), so the 2B's list results rest on fewer cases.

F.2 Binding and Relay

Table 16 repeats the single-answer battery of Section 4 at every size. The key-read edit, the output edit, and the relay values move the place at every size, and restoring the receiver's values at the relay heads restores the answer. The query edit is weaker below 8B (37.5% on the 2B and 59.0% on the 4B) and strong on the 32B (87.5% of 40). Two cuts differ by size. On the 2B the binding heads are necessary (blinding them at the marker keeps 12.0% of answers) and so are the relay heads (cutting

them keeps 3.0%), while on the 32B cutting its 16 relay heads leaves every answer intact, so its relay set is sufficient (85.0%) but not the only route. Random heads never exceed 1.5% at any size.

Table 16: **The single-answer battery at every size.** Each model's own heads, found with the rule of Appendix A.4. Percent of cases whose next token is the expected letter; the cuts give the percent of cases that keep the receiver's own answer. the 500 held-out pairs on the 2B, 4B, and 8B, and 40 cases on the 32B.

Edit	2B	4B	8B	32B
Binding queries: donor's request	37.5%	59.0%	76.5%	87.5%
Binding image keys: donor's layout	71.0%	82.0%	90.5%	95.0%
Queries and keys: donor's place	84.0%	96.0%	94.0%	100.0%
Binding outputs: donor's place	86.0%	93.5%	85.5%	92.5%
Relay values from the induced run: donor's place	84.5%	95.5%	87.5%	85.0%
receiver's values restored: own answer	91.0%	98.0%	99.5%	100.0%
Random heads, any edit	$\leq$1.0%	$\leq$1.5%	$\leq$0.5%	0.0%
No cut: own answer	93.0%	97.0%	98.0%	100.0%
Binding heads blind at the marker: own kept	12.0%	57.5%	77.0%	100.0%
Later tokens cannot read the marker: own kept	10.5%	24.5%	24.5%	32.5%
Relay heads cannot read the text at the colon: own kept	3.0%	28.0%	8.0%	100.0%
random heads: own kept	93.5%	97.0%	97.5%	100.0%

F.3 Gaze Heads

Table 17 repeats the tests of Section 6. The query-swap search selects 16 heads on the 2B (no K passed the gate, so the rule took the best), 128 on the 4B (a plateau from 16 heads on), 32 on the 8B, and 64 on the 32B (passing at 0.95 on the selection pairs), in the last third of the layers at every size. At every size the selected heads' queries move the answer to the donor's place, a moved pointer is blocked when their queries are held, and their image values bring the donor's letter. The 2B's 16 gaze heads are also necessary: blinding them loses 74.5% of the answers, where the 4B, 8B, and 32B keep theirs. The published 8B Gaze ranking, applied to the other sizes by head index, moves the answer in 56.0% of cases on the 4B and 10.0% on the 2B, so the ranking is a property of the 8B and the searched heads are needed at each size.

Table 17: **The gaze heads at every size.** The heads found by the query-swap search on each model. Percent of cases: the 500 held-out pairs on the 2B, 4B, and 8B, and 40 cases on the 32B.

Test	2B (16 heads)	4B (128 heads)	8B (32 heads)	32B (64 heads)
Queries swapped: donor's place	63.5%	75.5%	79.0%	90.0%
Pointer moved at the marker, queries held: own kept	82.0%	92.0%	98.0%	100.0%
Donor's image values: donor's letter	44.5%	63.0%	94.0%	97.5%
Only these heads and the marker see the image: own kept	84.5%	75.5%	96.0%	95.0%
These heads blind at the colon: own kept	25.5%	95.5%	86.5%	100.0%
Random heads, queries swapped	0.0%	0.0%	0.5%	0.0%

F.4 Progress

Table 18 repeats the three experiments of Sections 5.1 to 5.3, with the residual copied at every boundary of each model. The order of Section 5.2 reappears at every size, scaled by depth: a "one letter was written" phase, then a count, then a cursor after the last written panel, then the donor's letter. On the 32B the cursor reaches 96% of 50 pairs at layer 48 of 64, and the shared-panel skip peaks at 98% at layer 47, which is layer 26 in 8B units; the 32B has no early skip from the comma embedding (12% or less through layer 40), and the visited panel itself is never named (2% or less). On the 4B the cursor reaches 53% at layer 23 and the count 65.5% at layer 20, but the visited panel itself is named in 49% of pairs at layer 22, and on the 2B the visited panel is named in up to 41% of pairs (layer 18) against a cursor of at most 22%. So the 8B and 32B hold a clean "done up to here" state, and the smaller models hold the count and the visited panel about as often as the position after it. Rebinding the last label token to panel 4 with each model's own binding heads moves the cursor in 23.5% of pairs on the 4B, 10.5% on the 2B, and 26.0% of 50 on the 32B, against 85.5% on the

8B, with random heads at 0.0% and the first label token at most 7.0% at every size. On the 2B and 4B list task, a comma that cannot read the label token repeats the written letter in 41.6% of cases on the 4B and answers with a letter of another panel in most of the rest, and the 2B breaks rather than repeats (6.3% repeats); both keep 81% or more of their answers with the label token blind or the comma blind alone and lose them with both blind (4.9% and 12.5% kept), so the two routes of Appendix C.4 exist at every size.

Table 18: **Progress at every size.** The three experiments of Section 5 with the residual copied at every boundary of each model, and the rebind with each model's own binding heads. Percent of pairs, with the layer of the peak; 500 held-out pairs at the 2B, 4B, and 8B, and 50 at the 32B.

Reading of the copied comma state	2B (28 layers)	4B (36 layers)	8B (36 layers)	32B (64 layers)
Skip experiment: skips the shared first candidate	78 (17)	74 (20)	90.5 (20, 21)	98 (47)
Cursor experiment: "one written", skip one	87 to 99 (3 to 7)	97 (7)	97 (6)	82 to 92 (34 to 40)
Cursor experiment: count of two	100 (8 to 13)	65.5 (20)	95.5 (15)	100 (42)
Cursor experiment: the visited panel itself	41 (18)	49 (22)	$\leq$3 (20 to 24)	$\leq$2
Cursor experiment: first candidate after the visited panel	22 (20)	53 (23)	97.5 (21)	96 (48)
Donor's letter, from layer	22	30	30	56
Rebind, last label token, binding heads: past panel 4	10.5	23.5	85.5	26.0
all heads	13.5	23.5	68.0	34.0
random heads	0.0	0.0	0.0	0.0
Rebind, first label token, binding heads / all heads	0.0 / –	0.0 / 7.0	0.0 / 2.5	0.0 / 6.0

F.5 LANGUAGE

Table 19 repeats the colored-shape task on the 2B and 4B. On the 4B the picture is the 8B's: 16 language heads found in the same way move the language in 93.0% of 2,700 cases with 2.6% copied words, holding MLPs 33-35 leaves 10.5%, zeroing the heads returns 54% of German and French runs to English and no Chinese run, and the language vector switches runs in 32%, 48%, and 92% (German, French, Chinese). The gaze heads' summed output is less separated on the smaller models: cosine 0.88 across languages of one image against 0.71 across images on the 4B, and 0.87 against 0.78 on the 2B, where the 8B gives 0.95 against 0.50. On the 2B the 16 heads move the language in 23.7% of cases and copy the donor's word in 55.7%, so concept and language are not yet separate there.

Table 19: **The colored-shape language task on the 2B, 4B, and 8B.** 100 held-out images; 2,700 cross-language cases for the head edits.

Experiment	2B	4B	8B
Answers in the question's language, no edit	97.5%	98.2%	100.0%
Gaze heads' summed output: cosine across languages / across images	0.87 / 0.78	0.88 / 0.71	0.95 / 0.50
16 language heads replaced from the donor: own concept, donor's language	23.7%	93.0%	100.0%
copied donor words	55.7%	2.6%	0.0%
MLPs of the last three layers held	0.4%	10.5%	5.5%
Heads zeroed: German and French runs to English	76%	54%	90%
Heads zeroed: Chinese runs kept Chinese	7%	100%	100%
Language vector: to German / French / Chinese	75 / 75 / 100%	32 / 48 / 92%	75 / 100 / 100%