A FAILURE OF IMAGINATION: VLMS FAIL TO USE VISUAL REPRESENTATIONS FOR TEXT, BUT SELF-DISTILLATION CAN HELP

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language (VL) models are typically built by fine-tuning a language model to attend to visual features in service of a Visual Question-Answering objective. We ask whether this alignment changes how language models represent *text*—especially on tasks where visualization would help, like geometry questions posed in words. We distinguish *weak* multimodality (the model can answer questions about an input image) from *strong* multimodality (the model's text representations themselves become vision-aligned, improving performance on text-only tasks with latent visual structure). To test for strong multimodality, we construct a task of shape-counting problems with paired equivalent text and image specifications. This task is trivial when the image is provided, but requires visualization when specified in text as line coordinates. We compare instruction-tuned Qwen-2.5 and Mistral models against their vision-aligned counterparts on the text variant of the task. We find that the multimodal models do no better in this setting, despite strong performance on the image version of the task. Probing experiments corroborate this: vision-aligned decoder embeddings carry no more visual structure than their text-only counterparts. Finally, we provide an optimization objective that distills in-context visual embeddings into text embeddings. This alignment induces gains on the text variant of the task without explicit task supervision, even on samples more complex than seen during alignment. This suggests that VQA-style alignment leaves available cross-modal transfer untapped.

1 INTRODUCTION

We study the paradigm of vision language alignment used in the Qwen (Bai et al., 2025b;a), LlaVA (Liu et al., 2023), and Pixtral (Agrawal et al., 2024). These models begin with a strong language decoder that is then aligned with a pretrained vision encoder on a visual question answering task. Since the visual task feedback is used to update the language backbone, we ask whether it teaches the language decoder to build meaningful visual representations of text when useful.

We distinguish between *weak multimodality* and *strong multimodality*. A weakly multimodal model can operate on input from a different modality (vision) and can answer questions about that input. However, when it comes to input that is purely in text, a weakly multimodal model will behave exactly like the language-only model. By contrast, a strongly multimodal model can form representations for inputs provided in text that are vision-aligned and consequently perform better when the tasks are more naturally represented visually (for example, front-end code and edits, or solving geometry problems specified in words). Informally, strongly multimodal models treat all modalities as part of the same cognitive system rather than as separate input pipelines that share a decoder. We call this capability of a strongly multimodal model "visualization" in this work. We argue that strong multimodality is desirable because it would utilize data from a modality (say vision) to improve performance on another modality (text).

To study visualization in a tractable way, we propose a synthetic task that has the following properties (i) For each instance, there is a way to equivalently specify the question in text or image; and (ii) The problem is much easier when specified visually. Our task consists of code that draws shapes and the language model is asked questions about the resulting image (see section 3). We evaluate Qwen

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.

2.5 models (Bai et al., 2025b) and Pixtral / Mistral Nemo 12B (Agrawal et al., 2024; Mistral AI, 2024), which provide a clean minimal pair. We isolate the contribution of vision-language training to text-only reasoning by comparing vanilla text decoders to the corresponding VL-aligned text decoders. For both the model families, we find that vision alignment provides essentially no benefit on the text task, despite the image version being significantly easier for the same models. Linear and Contrastive probes on the decoder's hidden states confirm that the visual structure in both models' decoder representations are equally bad.

Finally, we address whether visualization – beyond what text training alone provides– can be instilled at all. We propose an alignment method based on self-distillation: the model is trained to predict, from text, the embeddings it would have obtained given the image. Applied to our synthetic data (where paired samples are available) *with no explicit task supervision*, this yields substantial improvements on the text version of the task across difficulty levels. This provides a proof of concept that alignment objectives aimed at strong multimodality, by targeting text representations directly, can yield gains on this task that VQA-style alignment does not.

Contributions. Our contributions are three-fold:

- A new task that can be equivalently specified visually and in text, but is much easier to solve using visual representations. This allows us to evaluate the same models on the vision-specified version of the task and the text-specified version of the task while controlling for difficulty. Additionally, it also allows us to compare vision-aligned text decoders to pure text decoders with the upper-bound context of vision informed performance.
- We present behavioral and representational evidence that for two widely used VL model families (Qwen 2.5 VL, Pixtral), the vision-aligned decoder does not construct visual representations for tasks provided in text, even when such representations are useful.
- To demonstrate that stronger multimodality can be instilled in multimodal models through better alignment methods, we provide a self distillation based method to vision-align a text student to learn to use visual representations for textual tasks. With this method, we provide a proof-of-concept model that operates on the text version of our task but with vision-informed representations – demonstrating improvements across the difficulties of the task.

2 Related Work

2.1 Non-text Representations in Pure Language Models

Sharma et al. (2024), Patel & Pavlick (2022), Abdou et al. (2021), and Ilharco et al. (2021) explore the question of whether visual grounding (similar to our notion of strong multimodality) emerges in vanilla language models. Sharma et al. (2024) operationalize this in a code generation setting – where they make the language models generate images from descriptions via code. Ilharco et al. (2021) and Abdou et al. (2021) study whether language models learn visual relationships from their text training corpus – Ilharco et al. (2021) train mappings from BERT representations of concrete words to representations from a vision model (Faster R-CNN) and find nontrivial structural alignment while Abdou et al. (2021) do this by comparing embeddings to the CIELAB colour space. Finally, Patel & Pavlick (2022) study grounded representations in language models (GPT-2 and GPT-3). Our work asks a complementary question: what does vision-language alignment provide *beyond* language-only training. Lepori et al. (2026) show that language models presented with a random walk over a latent state-space topology come to encode that topology in their hidden states (replicating Park et al. (2025)), yet largely fail to use those representations.

2.2 Visual Reasoning

Xu et al. (2026) and Wang et al. (2025) both study vision-language models on geometry tasks. Xu et al. (2026) investigate math problems that they can equivalently transcribe in images, text, and both together. They observe that the tested models (Qwen-3 (Bai et al., 2025a)) simply do best on the text-only variants of the problems. Wang et al. (2025) likewise report that the contribution of visual information diminishes on harder problems. These papers study whether visual information

in-context can be effectively leveraged by vision-language models. By contrast, we study visual representations in purely textual input. Our setup shares motivation with MentisOculi (Zeller et al., 2026), which designs tasks that require visual reasoning. They demonstrate that visual reasoning in models that can interleave text and image generation is limited by error accumulation. Our work differs in task specification: nearly all of their tasks include an image in the input, and on the one exception (RushHour) the models they test do *worse* with visual context than without. Therefore, this setting does not allow us to answer our more specific question: *In tasks where appropriate, does the vision-aligned text decoder perform better than a comparable text decoder with no vision training?*. Finally, Xu et al. (2025) criticize existing vision-language benchmarks on the grounds that they can be solved from the text descriptions alone.

2.3 Methods: Self-Distillation to Imagine Missing Modalities

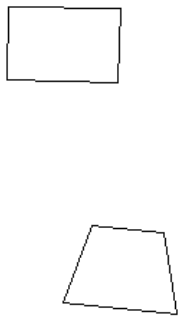
Yang et al. (2025) provide a latent continuous reasoning methodology. For tasks where one can generate intermediate helper images for reasoning as training data, they propose training a multimodal model to generate the image encoding of the intermediate helper images instead. This is followed by finetuning on the task. Additionally, the idea of imitating a missing modality has been explored by Li et al. (2024) and Zheng et al. (2022).

3 The Shape Counting Task

```
img = Image.new('RGB', (128, 255), color=(255, 255, 255))
draw = ImageDraw.Draw(img)

draw.line([109.36, 158.61, 118.10, 215.79])
draw.line([59.62, 153.53, 109.36, 158.61])
draw.line([1.03, 0.00, 79.01, 1.54])
draw.line([79.01, 1.54, 77.98, 53.53])
draw.line([118.10, 215.79, 39.51, 207.77])
draw.line([39.51, 207.77, 59.62, 153.53])
draw.line([77.98, 53.53, 0.00, 51.99])
draw.line([0.00, 51.99, 1.03, 0.00])
img.show()
```

(a) An example code snippet that would draw a configuration of a rectangle and a trapezium.



(b) Output image with one trapezium and one rectangle.

Figure 1: A simple example of the paired data sample: the code snippet (left) when compiled generates the image (right). The task is simply to count shapes of each kind given either the code snippet (for the text-variant of the task) or the rendered image (for the image variant).

We are concerned with problems that can be completely specified in text but would benefit a lot from intermediate computations in a visual representation space. To that end, we use a synthetic data setting for the experiments in this work. We describe it here briefly and provide the full data generation process in Appendix A.1. We procedurally generate python code that when run would output a set of shapes (specifically, rectangles and trapeziums). This allows us to generate samples of varying difficulties that deterministically specify an image but do so entirely in text; for an illustrative example, see Figure 1. To generate an item, we sample the number of rectangles and trapeziums in the scene, rotate them randomly, and sample their positions in the image. We derive from the shapes a list of edges with exact coordinates that can be turned into a python program (see Figure 1a). Crucially, we shuffle the order in which edges appear in the program. This leaves the rendered image unchanged but forces the model to compose edges that may be separated by long stretches of code.

4 VISION-LANGUAGE MODELS USE POOR VISUAL REPRESENTATIONS

Scope and Models. This study examines a popular vision-language alignment paradigm, in which a pretrained language model is coupled to a vision encoder through a translation MLP block and trained under some alignment objective (usually visual question answering). To isolate the effect of alignment on the language model's representations, we require *minimal pairs* of models, that is, the text decoder before visual adaptation and the updated weights after visual adaptation. These requirements leave us with the Qwen-VL family (Bai et al., 2025b;a), Mistral / Pixtral (Mistral AI, 2024; Agrawal et al., 2024) and LLaVA (Liu et al., 2023). Furthermore, we need the VLM to do better on the image variant of the task than the text variant to conduct the study. This would validate that our synthetic data is in-distribution and the vision encoder has the features required for the image variant of the task. LLaVA fails this criterion but we present its results in Section 4.1 for completeness. The analysis in this work centres around Qwen 2.5 and Pixtral models.

4.1 BEHAVIORAL EXPERIMENTS

The task is to count the number of shapes of each type given the source code. This requires the model to group edges into shapes and identify the angles in it. However, when the rendered image is available, the distinction (between rectangles and trapeziums) is obvious using heuristics in the image. We measure the performance of the model(s) on the task based on just the image; this establishes our upper-bound baseline. We then compare text-only models to vision-aligned text models on different answer modes (No-CoT, ICL-No-CoT, and ICL-CoT). In the NO-COT condition, the model is provided just with the description of the task, and is asked to output the answer immediately. In the ICL-NO-COT condition, the model is provided with the task description as well as examples of the task (i.e. in-context learning, or ICL), and is asked to output the answer immediately, Finally, in the ICL-COT condition, the model is provided with examples and is allowed to output whatever it wants before providing the answer. For the image variant of the task, the model is simply asked to look at the image and count the number of shapes of each type (for the full prompts, see Appendix A.2). For all ICL setups, we use 3 ICL examples. We calculate the accuracy of the task strictly: the model is given a point only if it gets the count of both the shapes right.

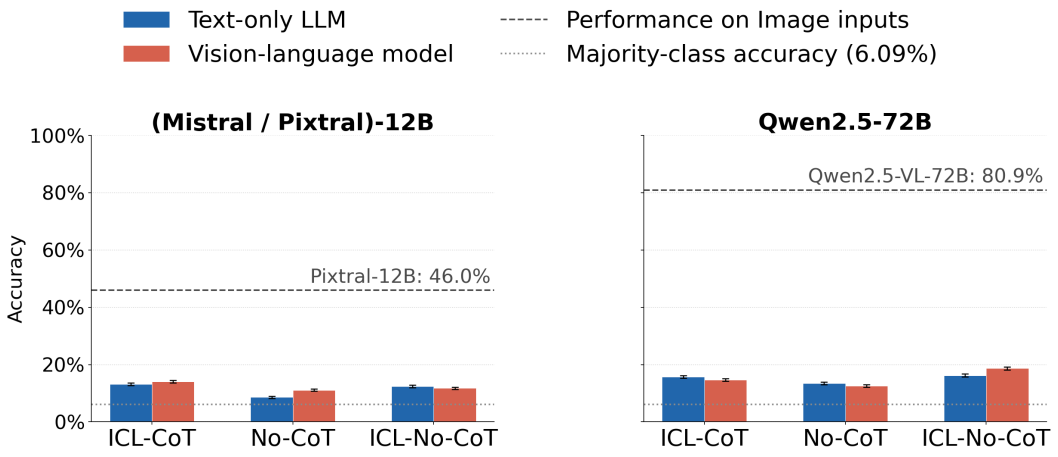


Figure 2: When the problem is provided in text, for all the settings tested—with and without in-context examples (ICL), and with and without chain of thought (CoT)—the performance of the multimodal models is similar to their text-only counterparts, and far below the image ceiling. This indicates that vision alignment did not transfer visualization capabilities to the text decoder. Error bars are based on the 95% confidence interval on accuracy estimates using the normal approximation to the binomial distribution.

Results. The results are shown in Figure 2. When the task is provided visually, the models' performance sets a high ceiling. Neither of the text decoders come close to this ceiling on the text variant of the task. Most importantly, the vision-aligned decoder performs similarly to the text

decoder. While ICL or COT don't seem to help our task, they improve the model's ability to simply count the total number of shapes (without identifying them). The models can often group vertices by looking at closed-loops during CoT (see Appendix A.7). Therefore, we restrict to rectangles and trapeziums as they have the same number of vertices and distinguishing them requires *some* estimation of the angle. Additionally, we ran these experiments on LLaVA and the corresponding text backbone Llama-Vicuna (Zheng et al., 2023). The vision-informed version of the task had an accuracy of 0.06 and the text variant performances were 0.07 for the multimodal decoder and 0.09 for the vanilla decoder. As the vision-informed performance is slightly worse, we exclude LLaVA from the remaining study.

4.2 Limited Evidence for Visual Representations in the Models' Embeddings

The results in the shape-counting task suggest that vision-language aligned language decoders are not significantly better than their language-only counterparts. We next test if the internal representations of the vision-aligned models have more visual structure than those of the text-only models. We use both linear probes and contrastive alignment probes (for details, see Appendix A.4.1 and Appendix A.4.2).

Linear Probes The probe is trained to predict the number of times a particular type of shape (rectangle or trapezium) appears in the sample. This is a 9-class classification problem (an image can have $0-8$ rectangles). For each layer, we train separate probes on the mean-pooled representations for rectangle count and trapezium count.

Contrastive Probes Our dataset consists of paired image-text instances (x_i, y_i) where x_i is a rendered image and y_i is the code that produces it. We measure the alignment between the text model's embeddings of the code and the embeddings of the corresponding image produced by DiNOv2 (Oquab et al., 2023), a pure vision encoder. We also conduct this experiment against a SigLip (Zhai et al., 2023) vision encoder. We fit linear projections to the InfoNCE (Oord et al., 2018) (contrastive) loss popularized by CLIP (Radford et al., 2021). We refer to these as *contrastive probes*. We extract a single vector for each: the CLS token from DINOv2 for x_i, and the mean-pooled hidden state at layer ℓ of the text model for y_i. Two linear projections $f_\theta : \mathbb{R}^{d_{\text{img}}} \to \mathbb{R}^{1536}$ and $f_\phi : \mathbb{R}^{d_{\text{txt}}} \to \mathbb{R}^{1536}$ map these into a shared space, where they are L2-normalized. The dimensionality of the shared space is that of DinoV2-Giant:

$$\tilde{z}_i^{\text{img}} = \frac{f_\theta(x_i)}{\|f_\theta(x_i)\|_2}, \qquad \tilde{z}_i^{\text{txt}} = \frac{f_\phi(y_i)}{\|f_\phi(y_i)\|_2}.$$

Given a batch of N such pairs sampled uniformly from the dataset, we optimize the symmetric InfoNCE objective with temperature $\tau = 0.5$:

$$\mathcal{L} = -\frac{1}{2N}\sum_{i=1}^{N}\left[\log\frac{\exp(\tilde{z}_i^{\text{img}}\cdot\tilde{z}_i^{\text{txt}}/\tau)}{\sum_{j=1}^{N}\exp(\tilde{z}_i^{\text{img}}\cdot\tilde{z}_j^{\text{txt}}/\tau)} + \log\frac{\exp(\tilde{z}_i^{\text{img}}\cdot\tilde{z}_i^{\text{txt}}/\tau)}{\sum_{j=1}^{N}\exp(\tilde{z}_j^{\text{img}}\cdot\tilde{z}_i^{\text{txt}}/\tau)}\right]. \tag{1}$$

We report bidirectional top-1 batch retrieval accuracy: instance i is correct in the image $\to$ text direction if $\arg\max_j \tilde{z}_i^{\text{img}}\cdot\tilde{z}_j^{\text{txt}} = i$, and analogously for text. We report the average of the two. Because both projections are linear, the retrieval accuracy can be treated as a proxy for the similarity in embedding structure between the two sets which we interpret as degree of visualization behavior. We argue that if the text model is doing computations that "look like" visualization at some layer then the representations at that level should have structure similar to the corresponding image embeddings from a good image encoder. We choose DiNo-v2 (Oquab et al., 2023) for the image embeddings as it is a *pure* image encoder that was never aligned or supervised by text. We want to compare the degree of "visualization" between vision-aligned text models (Qwen-2.5VL, Pixtral) and their non-vision-aligned counterparts (Qwen-2.5, Mistral-Nemo). On our dataset, we know there is perfect correspondence between the text and the image data therefore text embeddings with visual structure should be easier to align with corresponding image embeddings. See Figure 4 for results and Appendix A.4.2 for experimental details. To support these results we do the experiment with SigLip as the image encoder and see consistent results (see Appendix A.5).

Probe Controls To rule out easy-to-exploit shortcuts in both probing setups we do a random model control. For the controls, all the probes are retrained on top of embeddings calculated by a randomly-initialized language decoder. Low performance on these control probes implies that the

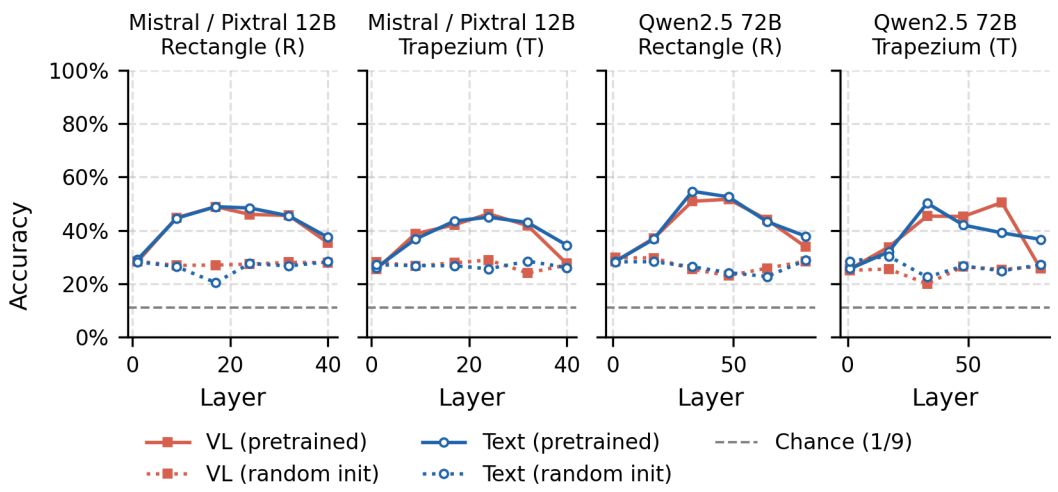


Figure 3: Vision-alignment does not add more visual structure to the embeddings of the multimodal model, compared to the text-only model. While probes trained on the embeddings from both classes of models show above-chance accuracy, the performance for the two classes of models is very similar.

probes measure properties of the pretrained checkpoints beyond what can be inferred simply from surface-level statistics in the embeddings.

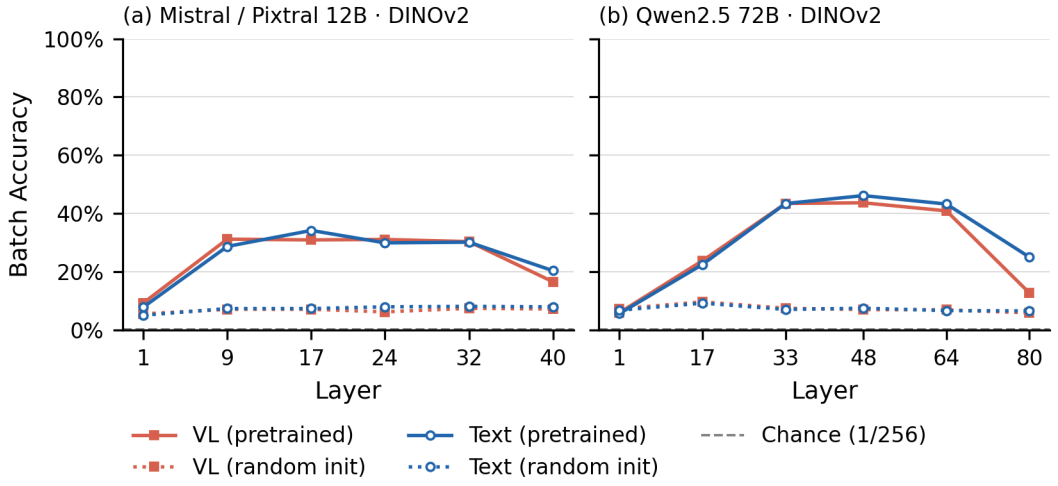


Figure 4: The trends on learnt retrieval corroborate that the multimodal decoder does not have better visual representations of textual inputs than the text-only model. Both the embedding sets get significantly above-chance and above-control retrieval accuracy against DiNo embeddings – at least in the middle layers. However, the VL embeddings closely track text embeddings.

Results. Both the linear probes (Figure 3) and the contrastive probes (Figure 4) show that, when the input is provided in text, the multimodal models typically do not encode visual structure better than their vanilla text counterparts. We see that representational quality with respect to visualization peaks in the middle layers and then falls back towards the end, a finding consistent with Skean et al. (2025). We also note that all the probes significantly outperform the controls. These findings together with the behavioral findings suggest that, for these models, the VQA based vision-alignment paradigm does not add visual structure beyond what text training already provides.

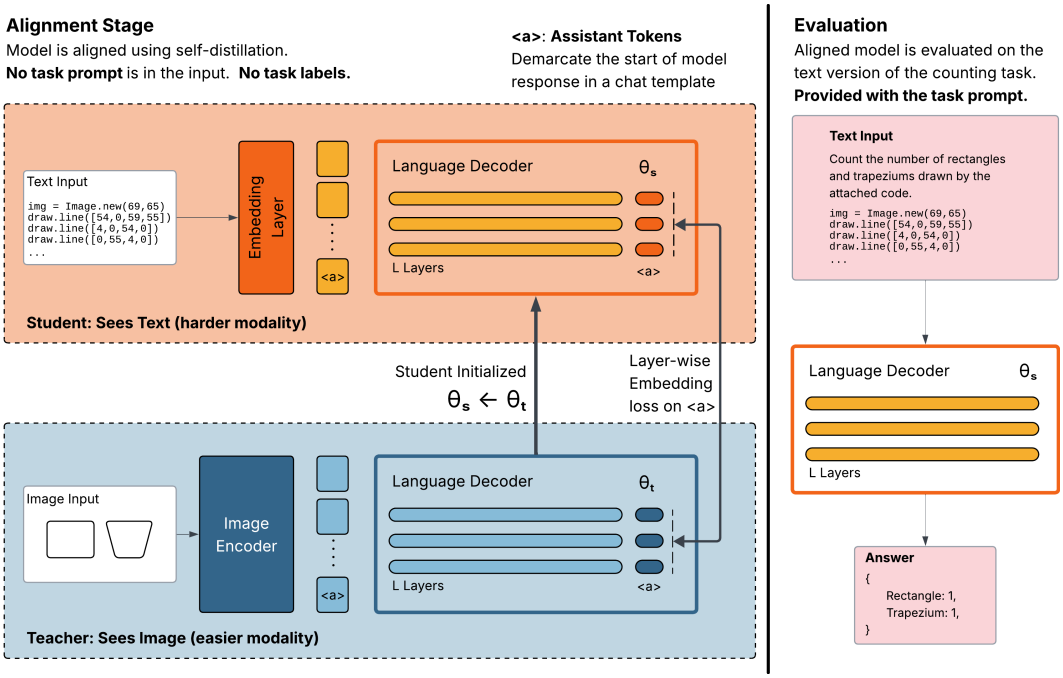


Figure 5: Given paired text-image data our alignment method trains the model to predict from text (code) the embeddings it would've had given the image. During alignment (left) we provide no task context or answer labels. During evaluation (right) the model performs the shape counting task.

5 Teaching the Model to Visualize

The problem, as identified, is that while vision-language models can pay attention to vision features to perform question-answering tasks (including the one we propose), they don't seem to construct internal representations that capture the visual structure of the problem. To encourage cross-modality transfer, we propose a methodology based on self-distillation. The intuition is that we will make the student model (initialized as a copy of the V+L teacher model) have the same intermediate embeddings *without* the image as the teacher calculates *with* the image. The Qwen 2.5 VL model treats vision features the same as token embeddings during language decoding i.e. vision patch embeddings are concatenated with text context (details about the Qwen architecture are in Appendix A.3). This allows us to get meaningful transfer by literally equating the embeddings as opposed to some projection or KL-divergence loss which is typical in distillation literature. We illustrate this in Figure 5.

Setup Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote a dataset of paired image–text examples, where x_i is an image and $y_i = (y_i^1, \ldots, y_i^{T_i})$ is a tokenized text sequence drawn from a vocabulary $\mathcal{W}$. We assume access to a pretrained vision–language model (VLM) f_θ comprising a vision encoder $\Phi : \mathcal{X} \to \mathbb{R}^{M \times d}$ that maps an image to a sequence of M visual embeddings, a text embedding layer $\mathcal{E} : \mathcal{W} \to \mathbb{R}^d$, and an L-layer transformer decoder $\mathcal{T}$ operating on sequences in $\mathbb{R}^d$, whose ℓ-th layer produces hidden states in $\mathbb{R}^d$ for $\ell \in \{1, \ldots, L\}$.

Teacher and Student We initialize two copies of the pretrained f_θ. The *teacher* hereafter referred to as f_{θ_T} retains the full architecture and is kept frozen throughout training. For the *student*, we discard the vision encoder; the student has to operate on text tokens only. Its parameters θ_S are trainable and initialized from θ.

Input Construction The teacher consumes the image-embeddings followed by the assistant token while the student consumes the text embeddings followed by the assistant token. We denote the assistant token as $\langle a \rangle$

$$z_T^{(i)} = \Phi_{\theta_T}(x_i) \,\|\, \mathcal{E}_{\theta_T}(\langle a \rangle), \quad z_S^{(i)} = \mathcal{E}_{\theta_S}(y_i) \,\|\, \mathcal{E}_{\theta_S}(\langle a \rangle) \tag{2}$$

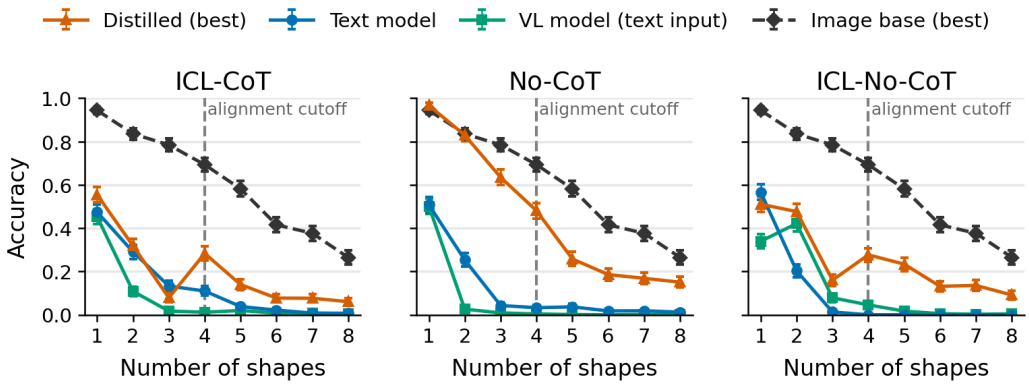


Figure 6: In all settings the distilled model outperforms the base models on across-shape accuracy even beyond the alignment cutoff, that is, the maximum number of shapes the distilled model has seen in a single sample. All the subplots represent the same set of samples – just with different prompts. All the models in this experiment are 7B models (Qwen 2.5 Instruct and Qwen 2.5 VL Instruct).

Where $\|$ denotes sequence concatenation along the token axis and $\mathcal{E}$, the embedding layer, that applies to each token independently.

Distillation objective. For each example we perform a single forward pass of the teacher and student decoders on their respective input embedding sequences and extract the hidden states at the assistant generation position across all L transformer layers:

$$h_T^{(i,\ell)} \;=\; \mathcal{T}_{\theta_T}^{(\ell)}\Big(z_T^{(i)}\Big)[-1], \quad h_S^{(i,\ell)} \;=\; \mathcal{T}_{\theta_S}^{(\ell)}\Big(z_S^{(i)}\Big)[-1], \tag{3}$$

for $\ell = 1, \ldots, L$, where $\mathcal{T}^{(\ell)}(\cdot)[p]$ returns the ℓ-th layer hidden state at position p and we use $[-1]$ to indicate the last in the sequence (which refers to the representation of $\langle a \rangle$ token). The student is trained to match the teacher's representations at this position via the layerwise ℓ_2 reconstruction loss. Note that for our experiments we try two variants of the reconstruction loss: vanilla MSE and normalized MSE; $\hat{h}$ in the following equation represents h or $\frac{h}{\|h\|_2}$, respectively. We find that normalized MSE works better and report the corresponding results in Figure 6.

$$\mathcal{L}(\theta_S) \;=\; \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}\left[\frac{1}{L\,d}\sum_{\ell=1}^{L}\big\|\hat{h}_S^{(i,\ell)} \;-\; \mathrm{sg}\big(\hat{h}_T^{(i,\ell)}\big)\big\|_2^2\right], \tag{4}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator, reflecting that the teacher is frozen. Note that the teacher conditions on x_i via $\Phi(x_i)$ while the student conditions on y_i; the loss therefore forces θ_S to encode, from y_i, the internal state the multimodal teacher would have formed from x_i.

5.1 Experiment

Alignment and Length Generalization We experiment with a Qwen2.5-VL-7B-Instruct model; we use a 7B model, as opposed to the 72B or 12B models used earlier, because of compute constraints. Crucially, during alignment, neither the student nor the teacher is provided with a specification of the task (an instruction to count the number of shapes) or task labels (the counts). Therefore, any transfer comes from the teacher's task-agnostic processing of the image. These image-conditioned representations are informative about properties such as shape counts, even though the teacher is never prompted to attend to them; this is precisely the kind of general visual structure we aim to transfer. To test for generalization, we only align the model on samples corresponding to a maximum of 4 shapes per image (and code). Then, as in section 4.1, we evaluate on a completely held-out set all the way up till 8 shapes, this time with prompts and task specification. The experiment demonstrates: 1. Task agnostic alignment 2. Length generalization of alignment. The results are shown in Figure 6. Note that all of them represent performance on the same set of samples and only the instruction prompts

are different for the text inputs. The image performance curve is on the same instructions prompt for all the subplots. We provide the different instructions used in Appendix A.2. See Appendix A.4.3 for other details.

Results Figure 6 shows that just the objective of predicting image-informed embeddings from the same model can show very strong gains on the text version of the task. The distillation model usually outperforms the vanilla text decoder and the multimodal decoder. It is important to note that the task was not specified during alignment – this suggests that image encoder *does* process the images in a generic way that is useful for the task *post-facto* and hence learning to imitate *that* computation allows our language model to visualize. The performance gains continue beyond the alignment cutoff, suggesting the stronger visual representations imparted by our distillation method generalize to samples that are harder to process than the ones used for alignment. We note that the gains in the ICL settings are modest, but this experiment is only intended as a proof-of-concept.

Language Evaluations We evaluate our distilled artifact on unrelated language benchmarks to check whether the distillation has catastrophic effects on language modeling. The distilled models show little to no degradation, See Appendix A.6.

6 Discussion

We test minimal pairs of language-only and vision-and-language models, on a task that can be equivalently specified in both text and image, but where the image version is substantially easier. We demonstrate that the standard vision-language alignment paradigm like in Qwen 2.5-VL and Pixtral do not make the language modeling backbone meaningfully better at text tasks that benefit from visualization. We conduct this analysis at two levels: behavioral and representational. Behaviorally, the vision-aligned text decoder and the vanilla text decoder perform comparably poorly on the text variant of our shape-counting task, lagging significantly behind the performance of the multimodal model when it is given the rendered image. Our representational probes show some evidence of visual structure emerging in the middle layers of both classes of models, but, crucially, the vision-aligned decoders' embeddings are typically on par with or worse than those of the vanilla text models. Together, these findings suggest that for models we study the VQA-based vision-alignment paradigm does not transfer vision priors into the language decoder's text-processing pathway: it only teaches the model to *attend* to visual features when they are present, and not to *think* visually about text.

A natural objection to this conclusion is that text training alone may already saturate whatever visual structure a language model can plausibly acquire. Prior work on emergent grounding in pure language models (Abdou et al., 2021; Sharma et al., 2024; Patel & Pavlick, 2022) has shown that nontrivial perceptual structure emerges in model that have only been trained on text. We address this objection with our distillation experiment. By aligning the Qwen-2.5-VL-7B text decoder so that, given a textual input, it reconstructs the hidden states the same model would produce given the corresponding image—without any task supervision—we induce measurable improvements on the text variant of our task. These gains length-generalize beyond the shape counts seen during alignment training. We interpret these results as a proof of concept for vision alignment that bleeds into the text pathways of a language model allowing it to use vision-informed representations for text tasks that benefit.

We view our work as a call to review how vision-language alignment training is usually done i.e. visual alignment of a language model is mostly seen as a way for it to process images. Our results suggest that this does not, by itself, provide strong multimodality. If our goal is to get strong multimodality then alignment strategies that make the language model better at text tasks like the self distillation we demonstrate deserve more attention

7 Limitations

The results demonstrated in this work including the proposed alignment methodology has been conducted on a single synthetic setup. Future work should expand the scope to other evaluative setups where the image-based performance is significantly better than the equivalently specified text-based performance cleanly. Furthermore, the distillation results are provided as a proof of concept. We leave adapting to scale of data for production-ready models to future work.

8 AI DISCLOSURE

LLMs were used to help code and manage the experiments reported in this paper i.e. for implementing methods. The written code was verified by the authors for correctness. They were also used to polish the document, including the figures and writing. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. Can language models encode perceptual structure without grounding? a case study in color. In *Proceedings of the 25th conference on computational natural language learning*, pp. 109–132, 2021.

Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, et al. Pixtral 12b. *arXiv preprint arXiv:2410.07073*, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Gabriel Ilharco, Rowan Zellers, Ali Farhadi, and Hannaneh Hajishirzi. Probing contextual language models for common ground with visual representations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5367–5377, 2021.

Michael A. Lepori, Tal Linzen, Ann Yuan, and Katja Filippova. Language models struggle to use representations learned in-context, 2026. URL https://arxiv.org/abs/2602.04212.

Mingcheng Li, Dingkang Yang, Yuxuan Lei, Shunli Wang, Shuaibing Wang, Liuzhen Su, Kun Yang, Yuzheng Wang, Mingyang Sun, and Lihua Zhang. A unified self-distillation framework for multimodal sentiment analysis with uncertain missing modalities. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 10074–10082, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Mistral AI. Mistral NeMo, July 2024. URL https://mistral.ai/news/mistral-nemo/. Accessed: 2026-07-02.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. ICLR: In-context learning of representations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pXlmOmlHJZ.

Roma Patel and Ellie Pavlick. Mapping language models to grounded conceptual spaces. In *International conference on learning representations*, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Pratyusha Sharma, Tamar Rott Shaham, Manel Baradad, Stephanie Fu, Adrian Rodriguez-Munoz, Shivam Duggal, Phillip Isola, and Antonio Torralba. A vision check-up for language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14410–14419, 2024.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann Lecun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *International Conference on Machine Learning*, pp. 55854–55875. PMLR, 2025.

Yuandong Wang, Yao Cui, Yuxin Zhao, Zhen Yang, Yangfu Zhu, and Zhenzhou Shao. Mathsight: A benchmark exploring have vision-language models really seen in university-level mathematical reasoning? *arXiv preprint arXiv:2511.23112*, 2025.

Weiye Xu, Jiahao Wang, Weiyun Wang, Zhe Chen, Wengang Zhou, Aijun Yang, Lewei Lu, Houqiang Li, Xiaohua Wang, Xizhou Zhu, et al. Visulogic: A benchmark for evaluating visual reasoning in multi-modal large language models. *arXiv preprint arXiv:2504.15279*, 2025.

Yige Xu, Yongjie Wang, Zizhuo Wu, Kaisong Song, Jun Lin, and Zhiqi Shen. Do vision-language models truly perform vision reasoning? a rigorous study of the modality gap. *arXiv preprint arXiv:2604.16256*, 2026.

Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens. *arXiv preprint arXiv:2506.17218*, 2025.

Jana Zeller, Thaddäus Wiedemer, Fanfei Li, Thomas Klein, Prasanna Mayilvahanan, Matthias Bethge, Felix Wichmann, Ryan Cotterell, and Wieland Brendel. Mentisoculi: Revealing the limits of reasoning with mental imagery. *arXiv preprint arXiv:2602.02465*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Wu Zheng, Mingxuan Hong, Li Jiang, and Chi-Wing Fu. Boosting 3d object detection by simulating multimodality on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13638–13647, 2022.

A Appendix

A.1 Data Generation Protocol

Each sample consists of a small scene of non-overlapping shapes placed along a random walk on a 2D plane, together with a shuffled list of the edge lengths of those shapes. The generation procedure is summarized in Algorithm 1; we describe its components here.

Shapes and directions. We draw shapes from the set $\mathcal{S} = \{\texttt{rectangle}, \texttt{trapezium}\}$. A rectangle is parameterized by two side lengths (a, b), and an isosceles trapezium by its top length t, bottom length b, and height h. For each shape we deterministically construct its vertices at the origin via the helpers RectVertices and TrapVertices, and connect them into a closed polygon with CyclicEdges.

Placements are connected by unit steps in one of four cardinal directions, $\mathcal{D} = \{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}\}$, with associated unit vectors $\{\mathbf{e}_d\}_{d\in\mathcal{D}}$. Successive shapes are placed at positions $p_1, p_2, \ldots$ where $p_{i+1} = p_i + \rho\,\mathbf{e}_{c_i}$ and $\rho > 0$ controls the spacing.

Sampling procedure. Given a sample, we first draw the number of shapes $n \sim \mathrm{Uniform}\{1, \ldots, N_{\max}\}$, their types $s_1, \ldots, s_n \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}(\mathcal{S})$, and the step directions $c_1, \ldots, c_{n-1} \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}(\mathcal{D})$ between successive placements. For each shape i we draw its dimensions uniformly from $[\ell_{\min}, \ell_{\max}]$ (and additionally $h_i \in [h_{\min}, h_{\max}]$ for trapezia), construct its vertices at the origin, and apply a rotation by $\theta_i \sim \mathrm{Uniform}(\Theta)$ followed by a translation to p_i. Letting $R(\theta)$ denote the 2×2 rotation matrix, each edge (u, v) of shape i contributes the rotated, translated edge $\big(R(\theta_i)u + p_i,\ R(\theta_i)v + p_i\big)$ to the global edge set E.

Output. Given the final edge set E, we sample a random permutation σ of its elements and return the sequence of edge lengths $\mathcal{P} = (\ell(e_{\sigma(1)}), \ldots, \ell(e_{\sigma(|E|)}))$ together with a rendered image of the scene. The shuffling forces the text version to require the model to compose edges separated by other lines of code.

Algorithm 1 Sample generation

Require: $N_{\max}$, $[\ell_{\min}, \ell_{\max}]$, $[h_{\min}, h_{\max}]$, ρ, Θ
$n \sim \mathrm{Uniform}\{1, \ldots, N_{\max}\}$
$s_i \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}(\mathcal{S})$ for $i = 1, \ldots, n$
$c_i \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}(\mathcal{D})$ for $i = 1, \ldots, n-1$
$p_1 \leftarrow \mathbf{0}$; $\quad E \leftarrow \emptyset$
for $i = 1, \ldots, n$ **do**
  $\theta_i \sim \mathrm{Uniform}(\Theta)$
  if $s_i = \texttt{rectangle}$ **then**
    $(a_i, b_i) \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}([\ell_{\min}, \ell_{\max}])^2$
    $V_i \leftarrow \textsc{RectVertices}(a_i, b_i)$
  else ▷ $s_i = \texttt{trapezium}$
    $(t_i, b_i) \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}([\ell_{\min}, \ell_{\max}])^2$
    $h_i \sim \mathrm{Uniform}([h_{\min}, h_{\max}])$
    $V_i \leftarrow \textsc{TrapVertices}(t_i, b_i, h_i)$
  end if
  $E_i \leftarrow \textsc{CyclicEdges}(V_i)$
  $E \leftarrow E \cup \left\{ \left(R(\theta_i)\, u + p_i,\ R(\theta_i)\, v + p_i \right) : (u, v) \in E_i \right\}$
  if $i < n$ **then**
    $p_{i+1} \leftarrow p_i + \rho \cdot \mathbf{e}_{c_i}$
  end if
end for
$\sigma \sim \mathrm{Uniform}(\mathrm{Permutations}(E))$
$\mathcal{P} \leftarrow \left(e_{\sigma(1)}, \ldots, e_{\sigma(|E|)} \right)$
return $(\mathcal{P}, \textsc{Render}(\mathcal{P}))$

A.2 Different Prompts Used

A.2.1 ICL + COT

This is the prompt we use for the setting where the model is provided with ICL samples and is allowed to do chain of thought.

User Prompt

Your task is to analyze the snippet of code in detail and identify the shapes it draws.
Follow these steps:
1. Parse coordinates: List each draw.line([x1, y1, x2, y2]) with start and end points.
2. Group connected lines: Identify which lines connect to form closed polygons.
3. Classify the shape: Decide if it is a rectangle, trapezium, triangle, or another polygon.
4. Locate the shape: Describe its approximate position (e.g., "top-left", "center-right").
5. Visual reasoning: Consider how the shapes would look on the canvas.
6. Count shapes: Report how many of each type of shape exist.
7. Summarize:
- Give a natural-language description of all the shapes and their layout.
- Provide a dictionary that lists the shape names and how many times each occurs.
Format your final answer like this:
Analysis:
<your reasoning and description here>
Shapes:
"Rectangle": <count>,
"Trapezium": <count>,
Important note: You will be given images that have either rectangles or trapeziums. No other shape.
Here are few examples with their right answers:

Figure 1:
‘‘‘
img = Image.new(’RGB’, (69, 65), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([54.50, 0.00, 59.00, 55.00], fill=’black’, width=1)
draw.line([4.50, 0.00, 54.50, 0.00], fill=’black’, width=1)
draw.line([0.00, 55.00, 4.50, 0.00], fill=’black’, width=1)
draw.line([59.00, 55.00, 0.00, 55.00], fill=’black’, width=1)
img.show()
‘‘‘
The image generated by the above code has the following shapes:
Shapes:
"Trapezium": 1,
Figure 2:
‘‘‘
img = Image.new(’RGB’, (75, 85), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([3.25, 0.00, 65.19, 2.76], fill=’black’, width=1)
draw.line([61.94, 75.69, 0.00, 72.93], fill=’black’, width=1)
draw.line([65.19, 2.76, 61.94, 75.69], fill=’black’, width=1)
draw.line([0.00, 72.93, 3.25, 0.00], fill=’black’, width=1)
img.show()
‘‘‘
The image generated by the above code has the following shapes:
Shapes:
"Rectangle": 1,
Figure 3:
‘‘‘
img = Image.new(’RGB’, (147, 257), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([137.00, 247.00, 44.00, 247.00], fill=’black’, width=1)
draw.line([88.00, 0.00, 88.00, 90.00], fill=’black’, width=1)
draw.line([88.00, 90.00, 0.00, 90.00], fill=’black’, width=1)
draw.line([115.50, 190.00, 137.00, 247.00], fill=’black’, width=1)
draw.line([0.00, 90.00, 0.00, 0.00], fill=’black’, width=1)
draw.line([44.00, 247.00, 65.50, 190.00], fill=’black’, width=1)
draw.line([0.00, 0.00, 88.00, 0.00], fill=’black’, width=1)
draw.line([65.50, 190.00, 115.50, 190.00], fill=’black’, width=1)
img.show()
‘‘‘
The image generated by the above code has the following shapes:
Shapes:
"Rectangle": 1,
"Trapezium": 1,
Figure 4:
‘‘‘
img = Image.new(’RGB’, (128, 225), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([109.36, 158.61, 118.10, 215.79], fill=’black’, width=1)
draw.line([59.62, 153.53, 109.36, 158.61], fill=’black’, width=1)
draw.line([1.03, 0.00, 79.01, 1.54], fill=’black’, width=1)
draw.line([79.01, 1.54, 77.98, 53.53], fill=’black’, width=1)
draw.line([118.10, 215.79, 39.51, 207.77], fill=’black’, width=1)
draw.line([39.51, 207.77, 59.62, 153.53], fill=’black’, width=1)
draw.line([77.98, 53.53, 0.00, 51.99], fill=’black’, width=1)
draw.line([0.00, 51.99, 1.03, 0.00], fill=’black’, width=1)
img.show()

```
The image generated by the above code has the following shapes:
Shapes:
"Rectangle": 1,
"Trapezium": 1,
Now, Do whatever analysis you need to understand the shapes drawn by the code snippet below:
```

A.2.2 ICL + No COT

This is the prompt for the setting where the model is provided with ICL examples and is asked to not do chain of thought.

User Prompt

```
Your task is to analyze the snippet of code in detail and identify the shapes it draws. Specifically, you need to count the number of shapes of each kind.
The answer should be in the form of a dictionary with shape names and counts.
Important note: You will be given images that have either rectangles or trapeziums. No other shape.
Here are few examples with their right answers:
Figure 1:
‘‘‘
img = Image.new('RGB', (69, 65), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([54.50, 0.00, 59.00, 55.00], fill='black', width=1)
draw.line([4.50, 0.00, 54.50, 0.00], fill='black', width=1)
draw.line([0.00, 55.00, 4.50, 0.00], fill='black', width=1)
draw.line([59.00, 55.00, 0.00, 55.00], fill='black', width=1)
img.show()
‘‘‘
The image generated by the above code has the following shapes:
Shapes:
"Trapezium": 1,
Figure 2:
‘‘‘
img = Image.new('RGB', (75, 85), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([3.25, 0.00, 65.19, 2.76], fill='black', width=1)
draw.line([61.94, 75.69, 0.00, 72.93], fill='black', width=1)
draw.line([65.19, 2.76, 61.94, 75.69], fill='black', width=1)
draw.line([0.00, 72.93, 3.25, 0.00], fill='black', width=1)
img.show()
‘‘‘
The image generated by the above code has the following shapes:
Shapes:
"Rectangle": 1,
Figure 3:
‘‘‘
img = Image.new('RGB', (147, 257), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([137.00, 247.00, 44.00, 247.00], fill='black', width=1)
draw.line([88.00, 0.00, 88.00, 90.00], fill='black', width=1)
draw.line([88.00, 90.00, 0.00, 90.00], fill='black', width=1)
draw.line([115.50, 190.00, 137.00, 247.00], fill='black', width=1)
draw.line([0.00, 90.00, 0.00, 0.00], fill='black', width=1)
draw.line([44.00, 247.00, 65.50, 190.00], fill='black', width=1)
```

draw.line([0.00, 0.00, 88.00, 0.00], fill='black', width=1)
draw.line([65.50, 190.00, 115.50, 190.00], fill='black', width=1)
img.show()
‘‘‘
The image generated by the above code has the following shapes:
Shapes:
"Rectangle": 1,
"Trapezium": 1,
Figure 4:
‘‘‘
img = Image.new('RGB', (128, 225), color=(255, 255, 255))
draw = ImageDraw.Draw(img)
draw.line([109.36, 158.61, 118.10, 215.79], fill='black', width=1)
draw.line([59.62, 153.53, 109.36, 158.61], fill='black', width=1)
draw.line([1.03, 0.00, 79.01, 1.54], fill='black', width=1)
draw.line([79.01, 1.54, 77.98, 53.53], fill='black', width=1)
draw.line([118.10, 215.79, 39.51, 207.77], fill='black', width=1)
draw.line([39.51, 207.77, 59.62, 153.53], fill='black', width=1)
draw.line([77.98, 53.53, 0.00, 51.99], fill='black', width=1)
draw.line([0.00, 51.99, 1.03, 0.00], fill='black', width=1)
img.show()
‘‘‘
The image generated by the above code has the following shapes:
Shapes:
"Rectangle": 1,
"Trapezium": 1,
Respond with only the answer dictionary for the code snippet below:

A.2.3 Prompt with only the task description

This is the prompt where the prompt is only provided with the task description.

User Prompt

Your task is to analyze the snippet of code in detail and identify the shapes it draws. Sometimes you will have the image it draws.
Specifically, you need to count the number of shapes of each kind.
The answer should be in the form of a dictionary with shape names and counts.
Important note: You will be given images that have either rectangles or trapeziums. No other shape.
Stick to this output format:
"Rectangle": <count>,
"Trapezium": <count>

A.2.4 Prompt for image input

User Prompt

Your task is to analyse the image and count the number of shapes of each kind in it.
The answer should be in the form of a dictionary with shape names and counts.
Important note: You will be given images that have rectangles and / or trapeziums. No other shape.
Look at each shape in this image and analyse whether it is a rectangle or a trapezium and tell me the counts.
Format your answer as a dictionary like so:
'Rectangle': <count>,

'Trapezium': <count>,

A.3 Summarising the Qwen 2.5 Architecture

We provide a summary description of the Qwen 2.5 VL models here to help with the intuition behind some design decisions for the distillation objective specified in Section 5.

The Qwen 2.5 VL models (Bai et al., 2025b) consist of three main components (i) the vision encoder which is a ViT with some efficiency modifications (ii) an MLP-based vision-language merger component (iii) Large Language Model which starts off as the Qwen 2.5 LM. An inference forward pass goes through the following: the image is processed by the ViT and the corresponding features are passed through the MLP merger block, following this the patch vectors are simply put in the context of the language model like the embedding of any other token. The Language Model decoder attends to the image information and the text information auto-regressively.

The vision encoder is pre-trained using a CLIP-style objective first. In the second-stage, all the components including the vision encoder, the MLP-merger, and the language model are kept unfrozen and trained for a visual question answering task. This is followed by a third stage to adapt the model to work with long-context videos.

A.4 Experiment Hyper-parameters

A.4.1 Linear-Probe Layer Sweep Details

Hyperparameters for the linear probe from Section 4.2.

- **Layers swept:** We conduct the probing experiments on evenly spaced 6 layers for each of the models.
- **Inference:** bfloat16 with FlashAttention-2 under HuggingFace tensor parallelism (`torchrun`, one process per GPU, all visible GPUs).
- **Pooling:** mean-pooling.
- **Optimizer:** Adam, learning rate 10^{-3} (PyTorch defaults otherwise); no weight decay, no gradient clipping.
- **Schedule:** at most 100 epochs with early stopping on validation loss (patience 10).
- **Batch size:** 32 (train/val/test); training batches are *not* shuffled.
- **Precision / hardware:** probe trains in fp32 on a single GPU.
- **Data:** $12{,}000$ synthetic scenes, at most 8 shapes, sizes in $[20, 100]$ px, random rotation, 10 px padding, shuffled description lines; sequential (not random) 80/10/10 train/validation/test split. Text embeddings only — no images are used.
- **Metrics:** macro-averaged accuracy, precision, and recall per head.

A.4.2 Contrastive Text–Image Probe Details

Hyperparameters for the contrastive probe from Section 4.2.

- **Text embeddings:** extracted exactly as in Appendix A.4.1 (same models, same six layers, same random re-initialization, masked-mean pooling, bfloat16 tensor-parallel inference).
- **Image embeddings:** frozen DINOv2-giant, final-layer CLS token (dimension 1536); the image-side layer is *not* swept. Images are rendered from the same scene descriptions, so each sample yields an aligned (text, image) pair. For SigLIP, the image embedding and shared projection dimension is 1152.
- **Probe:** one linear projection per modality; shared projection dimension $= \min(d_{\text{text}}, d_{\text{image}})$), with no hidden layers or activations.

- **Loss:** symmetric InfoNCE on ℓ_2-normalized projections — cross-entropy over the in-batch similarity matrix, averaged over the text→image and image→text directions — with a fixed temperature of 0.5.
- **Optimizer:** Adam, learning rate 10^{-3}; no early stopping, no gradient clipping.
- **Schedule:** 10 epochs.
- **Batch size:** 256; batches drawn by a repeated-shuffle sampler with sample multiplier 1.5 (each "epoch" covers 1.5 shuffled passes; no sample appears twice within a batch). The same sampler is used for validation and test, so evaluation metrics average over $\sim$1.5 shuffled passes of the split.
- **Precision / hardware:** probe trains in fp32 on a single GPU.
- **Data:** same generation settings and sequential 80/10/10 split as Appendix A.4.1 ($12{,}000$ samples, $9{,}600$ train).
- **Evaluation:** in-batch retrieval accuracy — within each batch of 256 pairs, the fraction of samples whose highest cosine similarity is their true counterpart, averaged over both retrieval directions (chance $= 1/256$).

A.4.3 Distillation Training Details

Hyperparameters for the distillation experiment from Section 5.

- Optimizer: AdamW, learning rate 5×10^{-5} (selected from a sweep over $\{10^{-6}, 5 \times 10^{-6}, 10^{-5}, 5 \times 10^{-5}\}$); PyTorch defaults otherwise ($\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01).
- Schedule: cosine decay with linear warmup over the first $1{,}000$ steps, decaying over the estimated total number of training steps; stepped per optimizer step.
- Gradient clipping: global norm 1.0 (computed collectively across FSDP shards).
- Epochs: We early-stop based on performance on a held-out validation set. The reported results are for the 10th epoch of a 40 epoch run.
- Batch size: 4
- Precision: full fp32 (no mixed precision).

The best model across sweeps was picked based on performance on a held-out validation set of 2000 samples. The test set consists of $6{,}000$ held-out samples.

A.5 Contrastive Probes With a SigLip as an Image Encoder

We support the results of the contrastive probes in Section 4.2 by rerunning them with a different image encoder as the "image reference". Specifically, we use SigLIP (Zhai et al., 2023).

A.6 LM Evaluations

These evaluations were run using the LM-Eval harness (Gao et al., 2023). We run the commonsense 0 shot setting, which consists of evaluating on Multiple-Choice questions on Arc challenge (Clark et al., 2018), Hellaswag (Zellers et al., 2019), PIQA (Bisk et al., 2019), and Winogrande Sakaguchi et al. (2019). We also run a 5-shot ICL setting on MMLU (Hendrycks et al., 2021). See results in Table 1.

	VL	Text	Distilled
Commonsense 0 Shot (Accuracy)	60.33%	59.39%	59.10%
MMLU (macro avg)	72.49%	74.66%	70.14%

Table 1: There is minimal to no degradation from the initialized VL checkpoint due to the distillation.

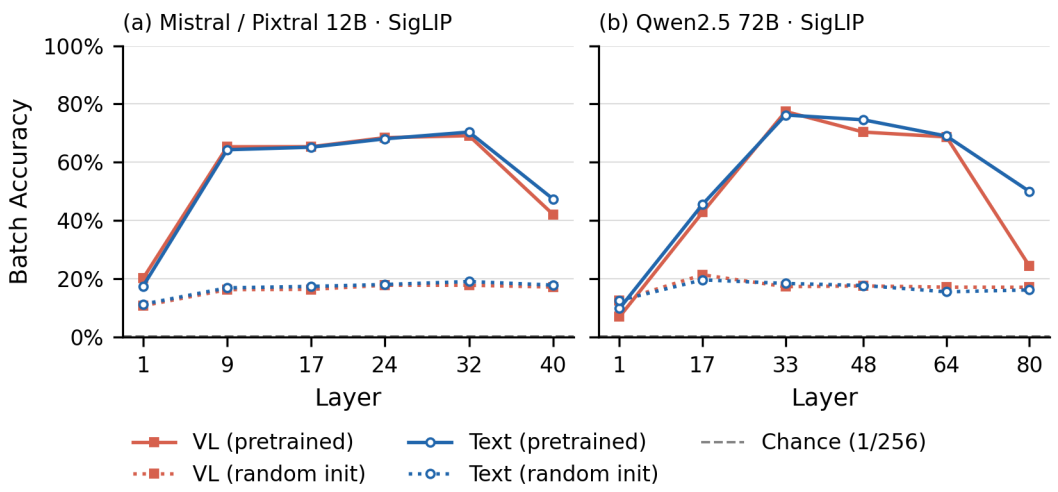


Figure 7: We observe the same trend as in Figure 4 i.e. the text embeddings from the text-only and the VL models show similar propensity to align with the corresponding image embeddings. The absolute numbers are higher than seen in Figure 4 because siglip image embeddings are pre-trained with text supervision and it is not a "pure" image encoder.

A.7 How often do models get the total number of shapes right?

We note that models can get the total number of shapes right without identifying them by performing a closed-loop analysis as previously mentioned in Section 4.1. They can look for cycles in the vertices and group them to deduce shapes. For our largest model (Qwen 2.5 72B), the COT setting really helps in getting the shape count variant right. See Figure 8.

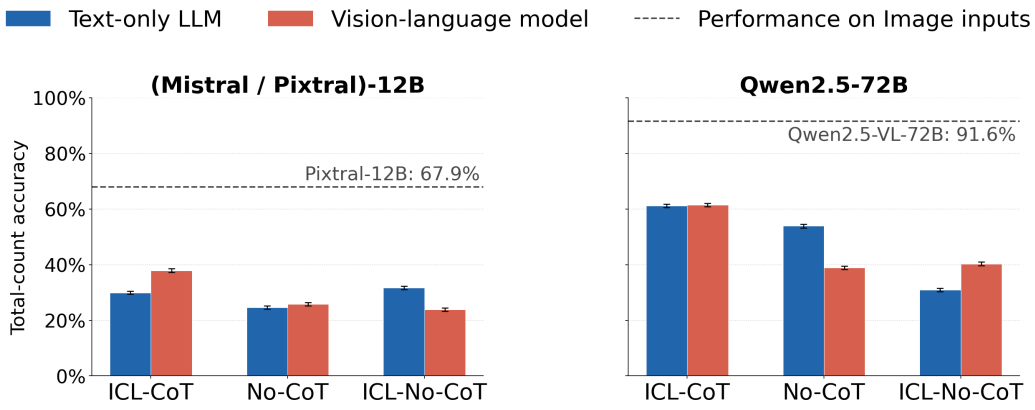
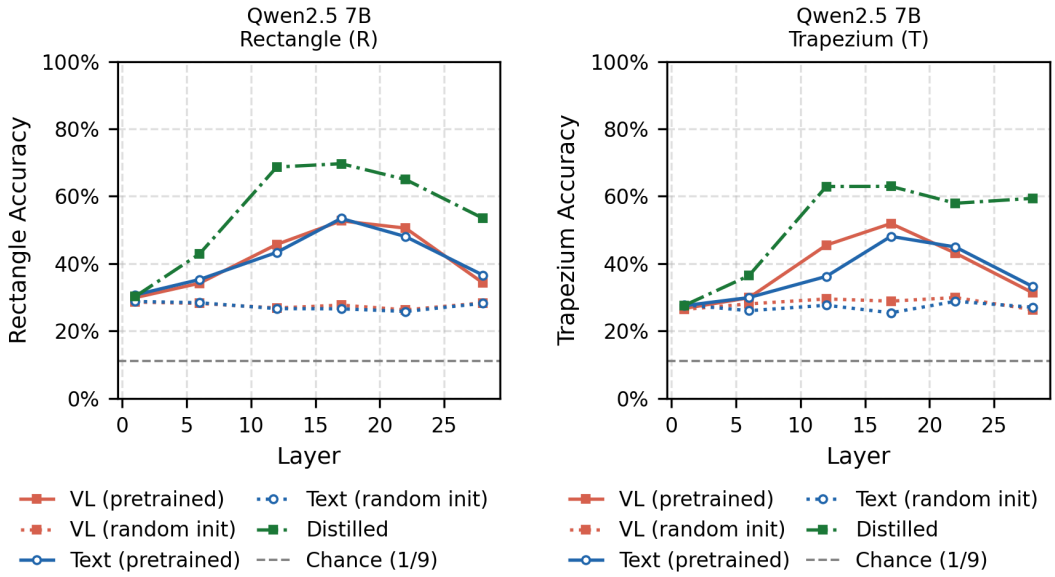


Figure 8: The models perform much better at the easier sub-task of getting the total number of shapes in a sample.

A.8 Probe Results on Distilled Artifacts

We run the probe experiments from Section 4.2 on our distilled artifacts from Section 5. The distillation objective, in some sense, is a literalisation of our strong multimodality principle as an optimization objective. Therefore, if the probes are measuring the right proxy, we should see a strong difference in trends for the distilled model on those experiments compared to the release models. We present these results in Figure 9 and Figure 10 and see that the distilled model's representations not only perform better on the probing task but also lend themselves better for the contrastive (retrieval) task against an unrelated image encoder. This is consistent with the probes tracking the property the distillation targets, though in-distribution fine-tuning may also raise probe accuracy.



(a) Linear probe on rectangles

(b) Linear probe on trapeziums

Figure 9: The distilled model representations are better across layers.

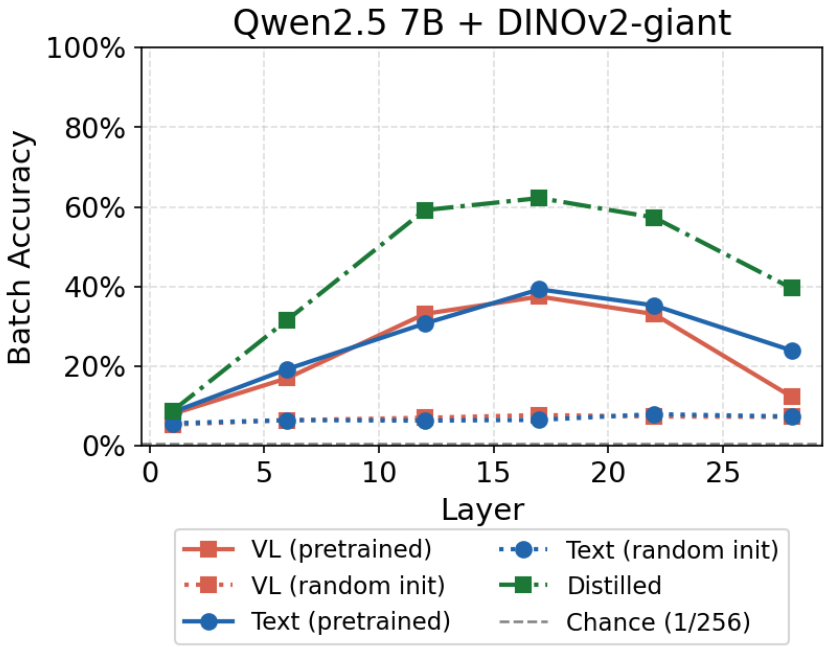


Figure 10: The distilled model representations are better across layers.