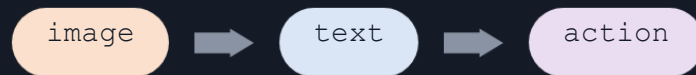


From seeing to doing

Vision-language models (VLMs), early vs. late fusion, and how a VLM becomes a vision-language-action model (VLA)



Three questions for today

1

What is a VLM?

How an image becomes a sequence of tokens that a language model can read.

slides 3-7

2

Where do the modalities meet?

Early, middle and late fusion, each with an annotated figure.

slides 8-14

3

How does a VLM become a VLA?

Keep the backbone, replace the text output with robot actions.

slides 15-20

Assumes: transformers, self-attention and next-token prediction.

A VLM images and associated text, and generates text

$$p(y \mid \text{image}, \text{prompt}) = \Pi_t p(y_t \mid y_{<t}, \text{image}, \text{prompt})$$

The answer y is generated one token at a time, conditioned on the image and the prompt. One model, many tasks:

Captioning

“Describe this photo.”

Visual question answering (VQA)

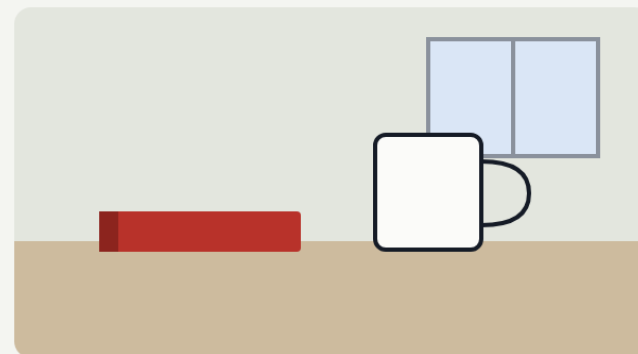
“What’s on the left of the cup?”

Document reading

“What is the total on this receipt?”

Grounding

“Where is the notebook?” → a box

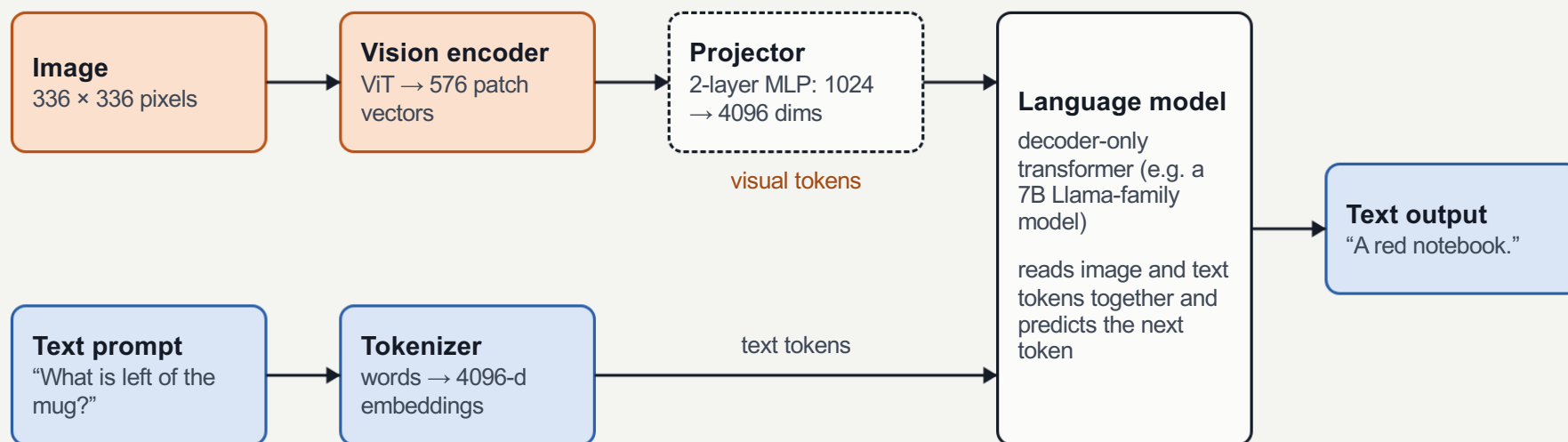


What is left of the cup?

A red notebook.

Open examples: LLaVA · Flamingo · PaliGemma · Qwen-VL. Chat assistants such as GPT-6, Gemini and Claude also accept images.

Three parts: encoder, projector, language model

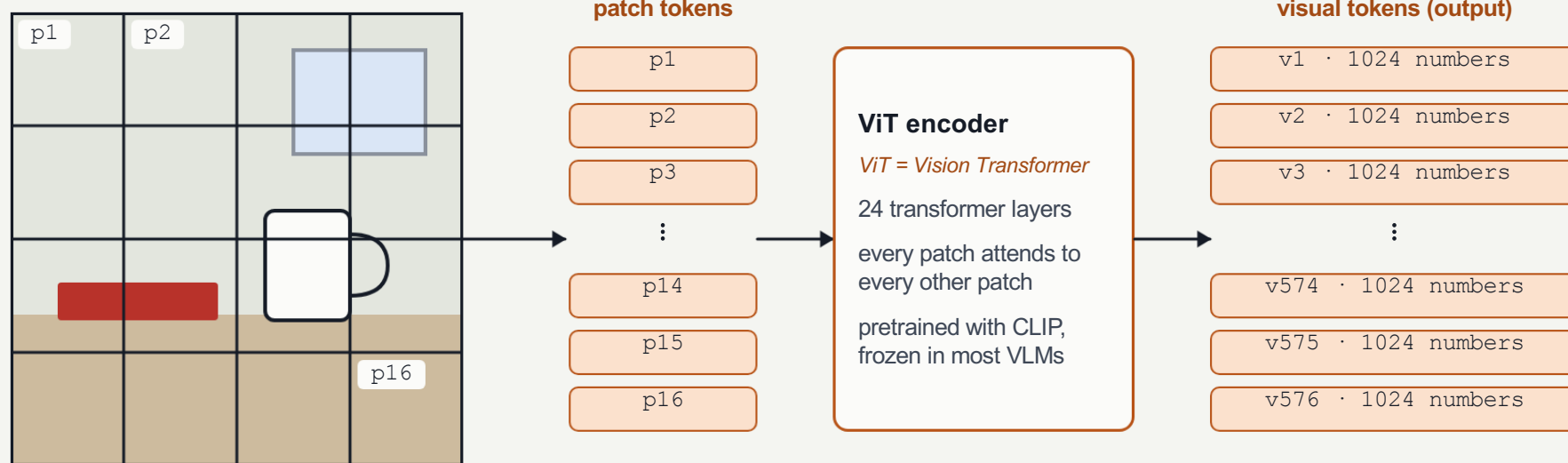


Only the projector is new. The encoder and the language model are pretrained separately, then connected.

ViT = Vision Transformer; MLP = multi-layer perceptron, a small fully-connected network.
Orange = vision · blue = language · dashed border = trained from scratch.

Values are for LLaVA-1.5 (CLIP ViT-L/14 at 336 px, 7B language model).

From pixels to patch tokens

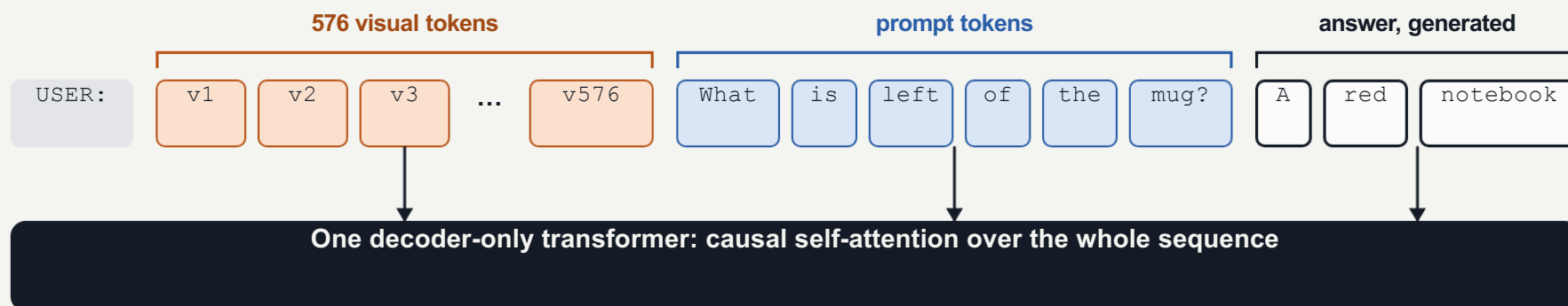


1 Cut the 336 px image into 14×14 px patches: $24 \times 24 = 576$ (4×4 drawn here).

2 Flatten each patch ($14 \cdot 14 \cdot 3 = 588$ numbers), project it linearly and add a position embedding.

3 Out come 576 vectors of 1024 numbers each: the visual tokens.

Image tokens and words share one sequence



- 1 Visual tokens live in the same embedding space as words, so the LLM needs no new layer type.
- 2 Causal mask: every prompt and answer token can attend to all 576 image tokens before it.
- 3 The next-token loss is applied to the answer tokens only; the image and prompt are context.

Train in stages, unfreezing one part at a time

Stage 0 · Pretrain the parts

Vision encoder **trained**

Projector not yet

Language model **trained**

Data: web image–text pairs for the encoder (CLIP used 400M); text for the LLM.

Goal: two strong models, trained separately.



Stage 1 · Align

Vision encoder  frozen

Projector **trained**

Language model  frozen

Data: ~558K image–caption pairs (LLaVA-1.5).

Goal: map visual features into the LLM's word-embedding space.



Stage 2 · Instruction-tune

Vision encoder  frozen

Projector **trained**

Language model **trained**

Data: ~665K image-grounded conversations and VQA examples (LLaVA-1.5).

Goal: follow instructions about images.

Where do image and text first meet?

Terms vary between papers. Here, fusion means how deep into the network the two streams first exchange information.

← more cross-modal interaction, more compute

cheaper; embeddings can be reused →

network input

network output

Early fusion

Merge the tokens before layer 1. One transformer processes both.

LLaVA · PaliGemma · Fuyu · Chameleon

Middle fusion

Separate streams, joined by cross-attention inside the network.

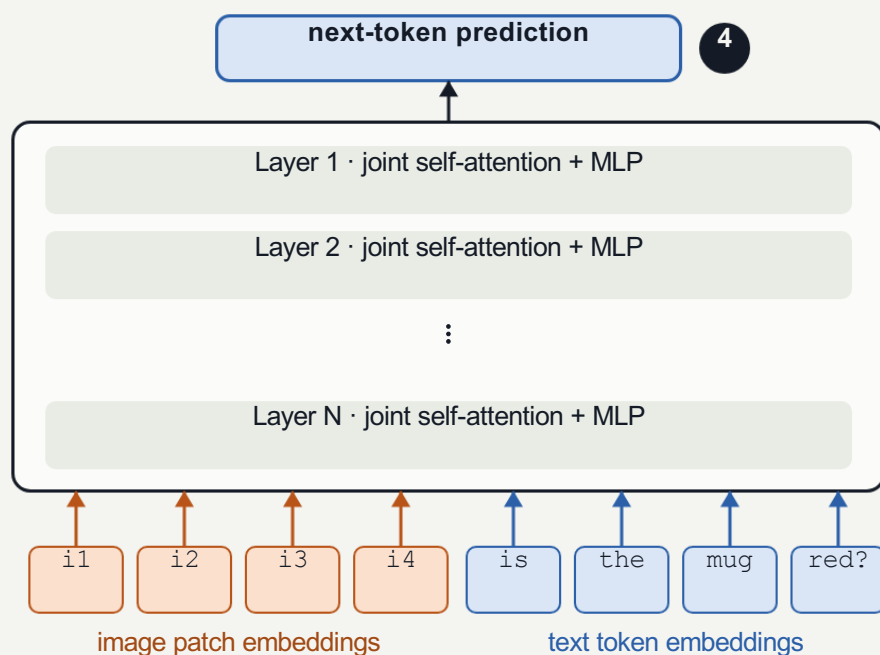
Flamingo · IDEFICS · Llama 3.2 Vision

Late fusion

Encode each modality fully, then compare or combine two vectors.

CLIP · ALIGN · SigLIP

Early fusion: one transformer sees everything

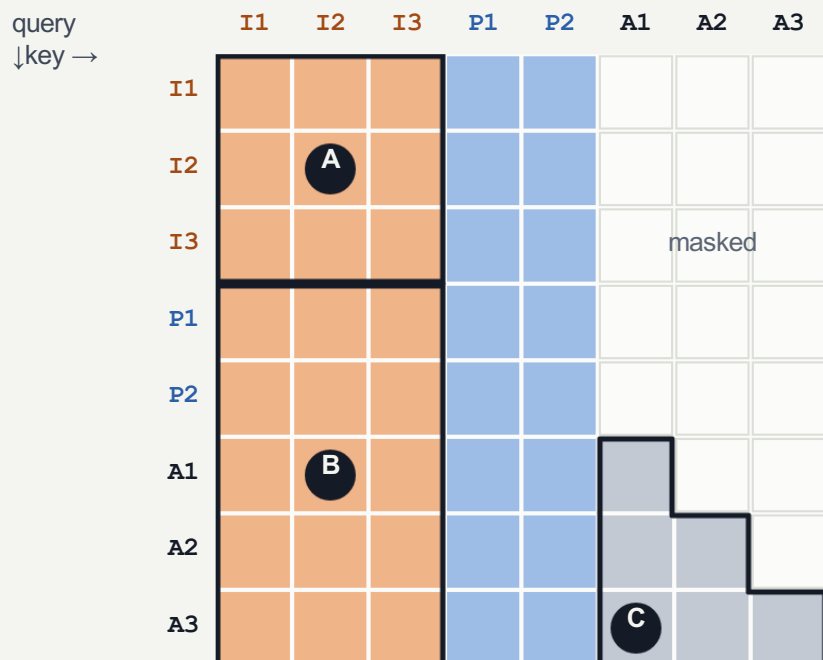


- 1 Before layer 1, image patch embeddings and text token embeddings are concatenated into one sequence.
- 2 In every layer, self-attention mixes all tokens: text looks at patches, and patches can look at text.
- 3 One set of transformer weights handles both modalities.
- 4 A single head predicts the next text token.

Examples. Fuyu-8B feeds raw patches through one linear layer into the decoder, with no image encoder. Chameleon turns images into discrete tokens mixed freely with text. LLaVA and PaliGemma are early fusion from the LLM's point of view.

+ fine-grained cross-modal reasoning – cost grows with (image + text tokens)²

The attention mask shows where fusion happens



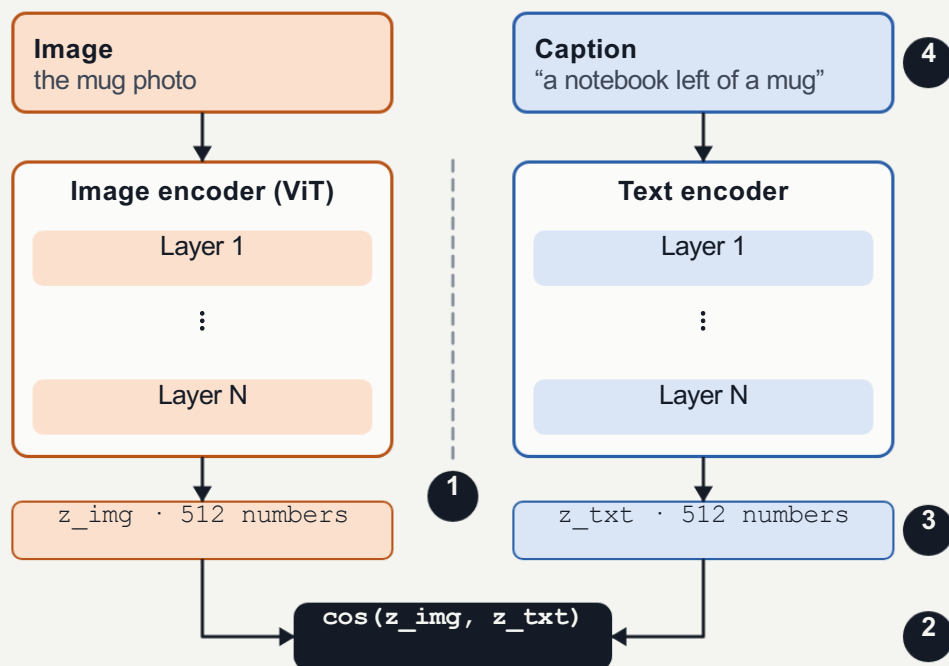
Row = query token. Column = key token it may attend to.
Blank = masked.

- A** Image ↔ image. Patches refine each other, as inside a ViT.
- B** Text → image. Every prompt and answer token can read every patch, in every layer. This column is what early fusion buys.
- C** Answer is causal. Each answer token sees the prefix and earlier answer tokens only.

This is a prefix-LM mask (PaliGemma): image and prompt attend both ways. A purely causal VLM such as LLaVA also masks image → prompt.

Cost: 576 image + 50 text tokens → $626^2 \approx 392\text{K}$ attention scores per head, per layer.

Late fusion: two towers, one comparison



- 1 Two separate towers.** No information crosses the dashed line until the very end.
- 2 Fusion is one number:** a cosine similarity, or a small head on the two concatenated vectors.
- 3 One vector per input.** Embeddings can be computed once and indexed, so one text query can search millions of images.
- 4 The cost.** Relations such as "left of", and counts, are hard to recover from two squeezed vectors.

Examples. CLIP, ALIGN and SigLIP are dual encoders trained this way. CLIP's image tower is the vision encoder that early-fusion VLMs build on.

CLIP learns by matching pairs in a batch

caption embeddings $z_{\text{txt}} \rightarrow$

	a dog	a mug	a bike	a tree
photo: dog	.31	.12	.08	.10
photo: mug	.11	.29	.09	.07
photo: bike	.06	.10	.33	.12
photo: tree	.09	.08	.13	.30

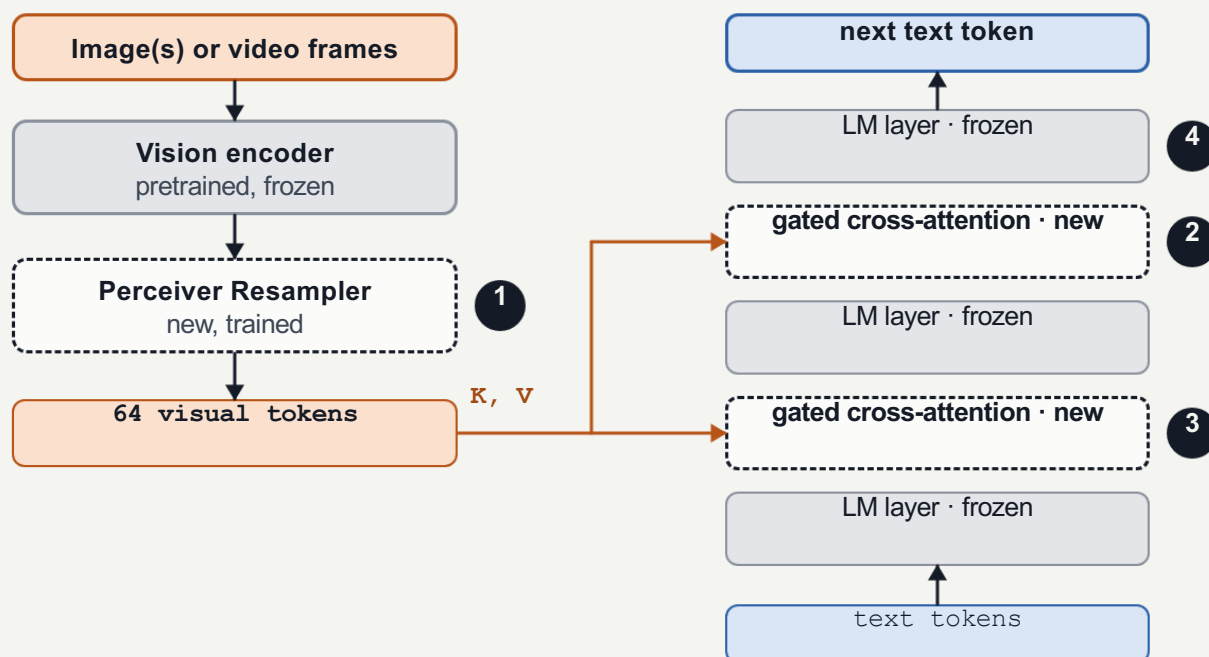
image embeddings $z_{\text{img}} \downarrow$

Illustrative cosine similarities, $N = 4$.

Dark cells = true pairs (positives). All other cells = negatives.

- 1 A batch holds N image–caption pairs. Each image and each caption is encoded on its own (late fusion).
- 2 Compute all $N \times N$ similarities. The diagonal holds N positives; the other $N^2 - N$ cells are negatives.
- 3 Loss: cross-entropy over each row (which caption fits this image?) and each column (which image fits this caption?), averaged.
- 4 CLIP used $N = 32,768$ and 400M web pairs. Zero-shot classification: pick the caption “a photo of a {class}” with the highest score.

Middle fusion: cross-attention inside the LM



- 1 The resampler turns any number of patches into a fixed 64 tokens.
- 2 Text tokens are the queries; the 64 visual tokens are keys and values. Images never enter the text sequence.
- 3 Output is scaled by $\tanh(\alpha)$, with $\alpha = 0$ at the start, so training begins from the unchanged LM.
- 4 The pretrained LM layers stay frozen, which keeps its language skills.

Examples. Flamingo (2022), IDEFICS, Llama 3.2 Vision.

Early, middle and late fusion side by side

	Early	Middle (cross-attention)	Late
Streams meet	At the input of the main transformer	Inside the LM, at added cross-attention layers	After both encoders finish
Interaction	Every layer, every token pair	Every k-th layer; text reads image	One score, or a small head
Sequence cost	High: ~576 extra tokens per image	Low: 64 visual keys; text length unchanged	Minimal: one vector per input
Reuse embeddings?	No: recompute per question	Image features only	Yes: index once, search fast
Good at	Detailed QA, OCR, spatial grounding	Many images per prompt; LM kept intact	Retrieval, zero-shot classification
Examples	LLaVA, PaliGemma, Fuyu, Chameleon	Flamingo, IDEFICS, Llama 3.2 Vision	CLIP, ALIGN, SigLIP

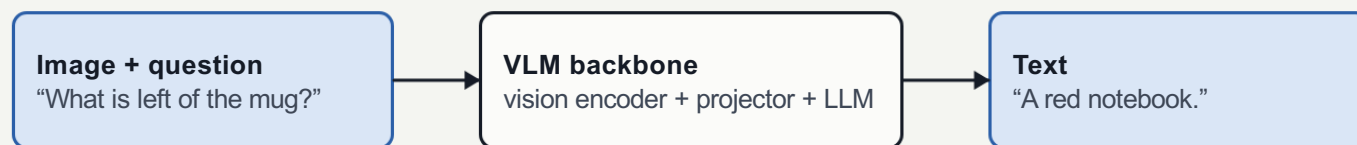
Robots need to know **where** things are and **how** they relate. That is why today's VLAs are built on early-fusion VLMs.

A VLA is a VLM whose output is robot actions.

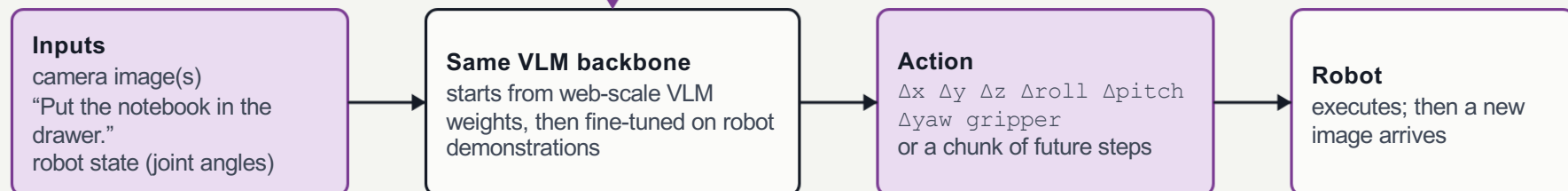
Same pretrained backbone, same early fusion. Robot state is added as an input, and an action head is added as the output.

Same backbone, different output

VLM · answers questions



VLA · controls a robot



same weights, then fine-tuned

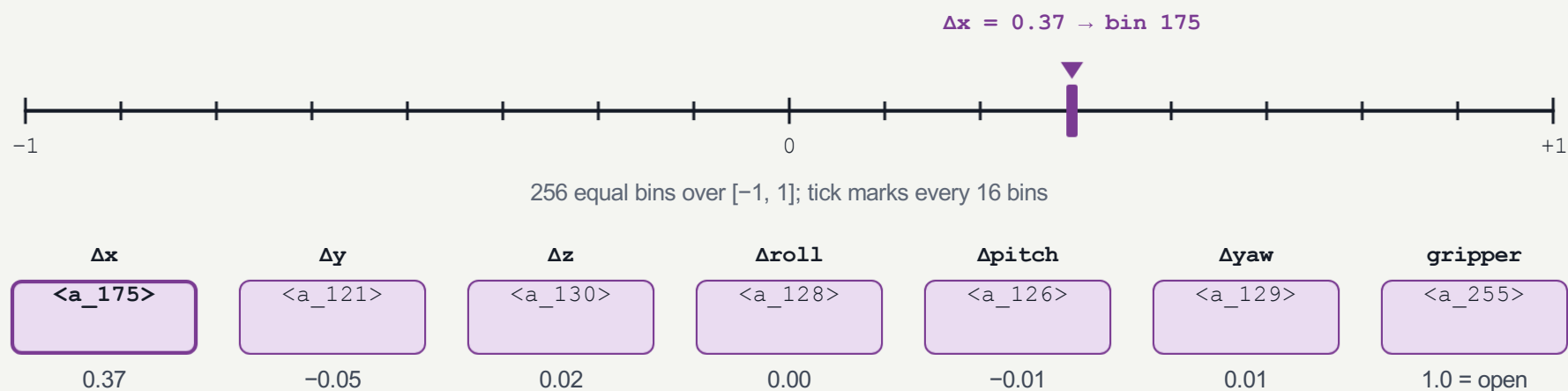
1 New input: robot state (proprioception).

2 Robot data, e.g. Open X-Embodiment: 1M+ trajectories, 22 robot types.

3 New output: 7 numbers per step, or a chunk.

4 Runs in a loop: 1–3 Hz (RT-2) up to 50 Hz (π0).

Turning a motor command into words



The LLM generates these 7 tokens left to right. Each is then mapped back to its bin's centre value.

Bins. RT-2 splits each dimension into 256 uniform bins. OpenVLA sets each range from the 1st–99th percentile of training actions.

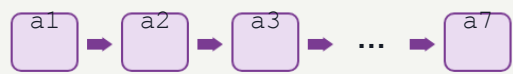
Vocabulary. RT-2 (PaLI-X) reuses existing number tokens. OpenVLA overwrites the 256 least-used tokens of the Llama tokenizer.

Resolution. One bin is $1/256 \approx 0.4\%$ of the range. That is fine for pick-and-place, but coarse for fast, dexterous motion.

Two ways to produce actions

Discrete action tokens

RT-2 · OpenVLA



one token at a time, 7 tokens per action step

Loss: cross-entropy over bins, same as text.

Speed: RT-2 (55B) runs at 1–3 Hz; OpenVLA (7B) at about 6 Hz on one RTX 4090.

Strength: no change to the VLM architecture.

Continuous action expert

π_0 (Physical Intelligence)



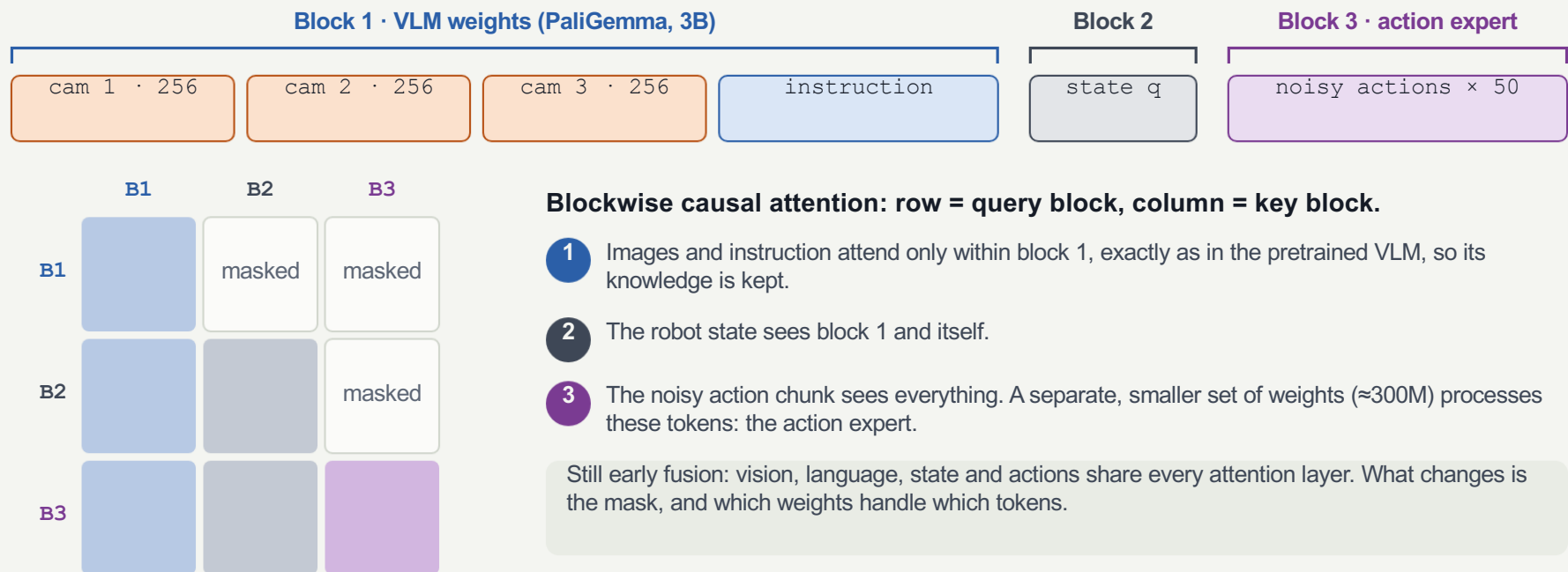
noise → 10 denoising steps → 50-step action chunk

Loss: flow matching (a diffusion-like technique) that learns to turn noise into actions.

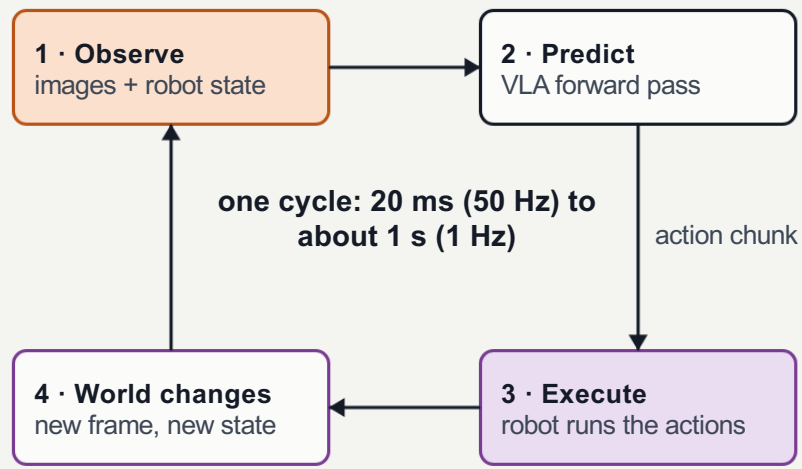
Speed: one pass gives 50 future actions, for control at up to 50 Hz.

Strength: smooth, precise, dexterous motion (e.g. folding laundry).

π_0 : early fusion with a blockwise mask



A VLA runs in a loop, which is hard



RT-2 predicts one step per cycle. $\pi 0$ predicts a chunk of 50 steps and executes several of them before looking again.



Latency

Billions of parameters must answer within one control cycle.



Data

Robot demos number in the millions; web image-text pairs in the billions.



Embodiment

Arms, grippers and cameras differ; one model must handle many robots.



Safety and evaluation

Errors act on the physical world; real-robot tests are slow and costly.

SUMMARY

Three things to remember

- 1 A **VLM** turns image patches into tokens that a language model reads alongside text.
- 2 **Fusion** is where the streams meet: early (joint attention from layer 1), middle (cross-attention) or late (compare two vectors). Early fusion gives the fine-grained grounding robots need.
- 3 A **VLA** keeps the VLM backbone and changes the output: discrete action tokens (RT-2, OpenVLA) or a continuous action expert (π_0).

Further reading: CLIP (Radford et al., 2021) · Flamingo (Alayrac et al., 2022) · LLaVA (Liu et al., 2023) · RT-2 (Brohan et al., 2023) · OpenVLA (Kim et al., 2024) · π_0 (Black et al., 2024)