Open-Vocabulary Detection, Grounding & Segmentation

W4 | TUESDAY, SEPTEMBER 15, 2026

SEMINAL

OWL-ViT

Minderer et al. · 2022

FRONTIER

Qwen3-VL-Seg

Yao et al. · 2026

Group Member: Arihanth Jayavijayan, Malhar Jadhav, and En-Chao Liu

Georgia Tech

Outline

01 **Detection, Segmentation, and Grounding**
Open Vocabulary vs Language Understanding

02 **Seminal: OWL-ViT**
Follow-Up Works

03 **Frontier: Qwen3-VL-Seg**
Conclusions & Our take

Presentation Overview

How do we reuse broad visual semantics for a new spatial output?

Seminal Paper · 2022

OWL-ViT

Contrastive image–text model
→ object-level predictions

Text queries → scores + boxes

Backbone Bridge

Qwen3-VL

Interpret an instruction
and ground its target

Grounded box + features

Frontier Paper · 2026

Qwen3-VL-Seg

Reuse MLLM grounding
→ box-guided mask decoder

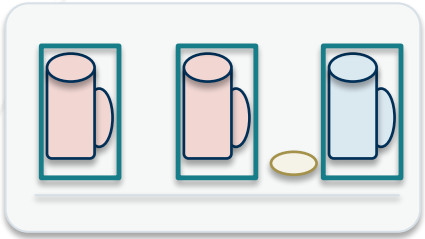
Referring instruction → masks

Same principle: reuse pretrained knowledge. Different language interface and spatial output.

Detection, Segmentation, and Grounding

Detection

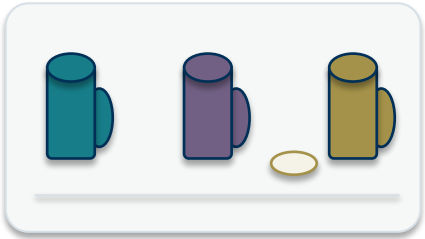
“mug”



Find all matching instances; output boxes.

Instance segmentation

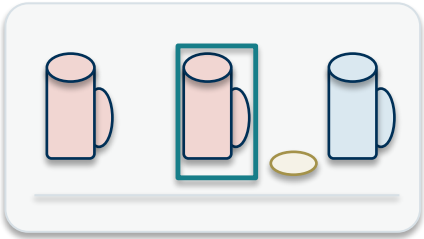
No referring expression



Separate object instances; output individual masks.

Referring grounding

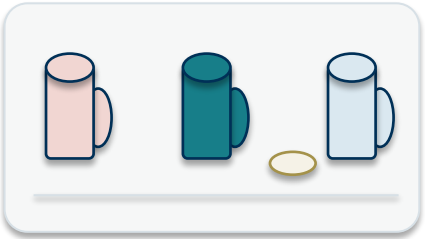
“red mug beside the plate”



Resolve the expression; output the target box.

Referring segmentation

“red mug beside the plate”

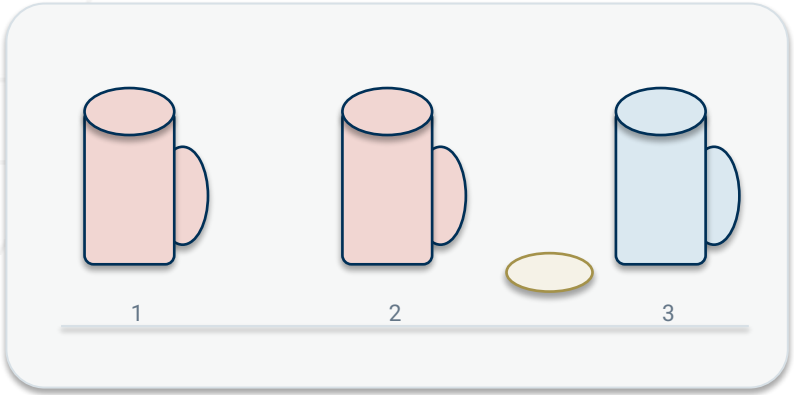


Resolve the expression; output the target mask.

Two evaluation axes: choose the intended instance, then represent its location precisely.

Open Vocabulary ≠ Full Query Understanding

Adding nouns is different from binding attributes, relations, order, and negation



Same objects. Different target selection.

Category
“mug”

Attribute binding
“red mug”

Relation
“the red mug beside the plate”

Relation + negation
“the red mug not beside the plate”

Novel-category AP does not establish reliable compositional grounding.

Georgia Tech

OWL-ViT

- Minderer et al. (ECCV 2022)

Georgia Tech

Works Before OWL-ViT

Same bottleneck: broad semantics are plentiful; category-specific bounding boxes are expensive

ViLD
Distill a teacher

CLIP crop embedding → Region-feature distillation → Detector + text classifier

Detic
Use cheaper labels

Image-level class label → Classify a large proposal → Expanded detector vocabulary

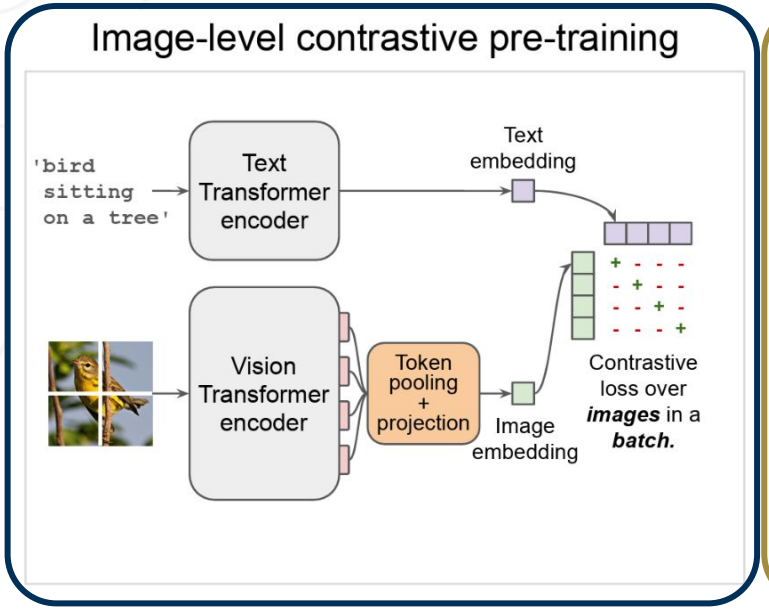
GLIP
Unify as grounding

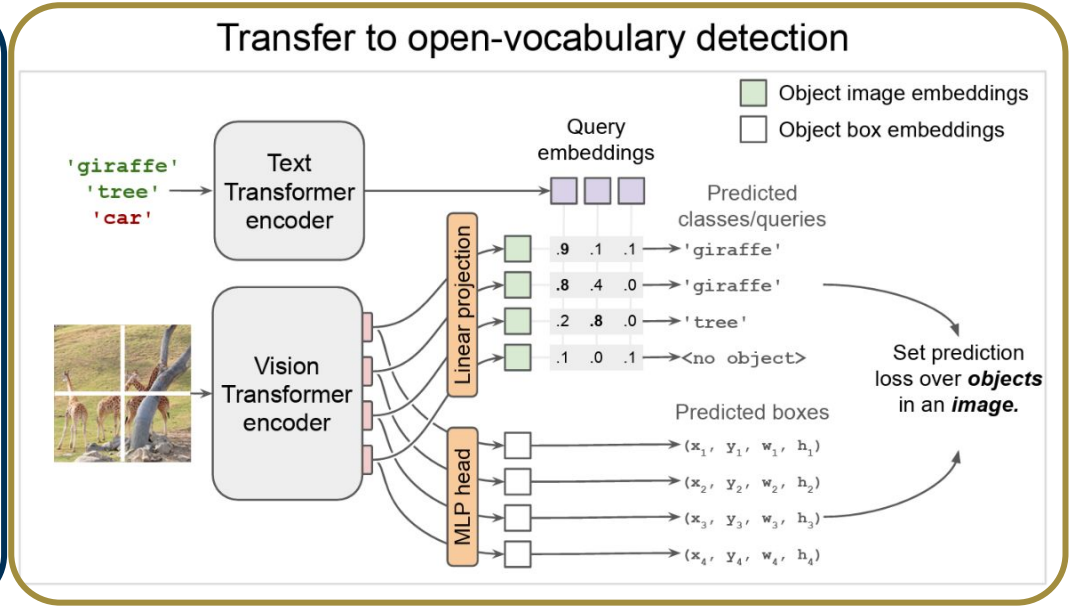
Class names / phrases → Deep image–text fusion → Region–word alignment

OWL-ViT asks: can the pretrained dual encoder itself become the detector?

ViLD · ICLR 2022 Detic · ECCV 2022 GLIP · CVPR 2022

Approach – Seminal: OWL-ViT





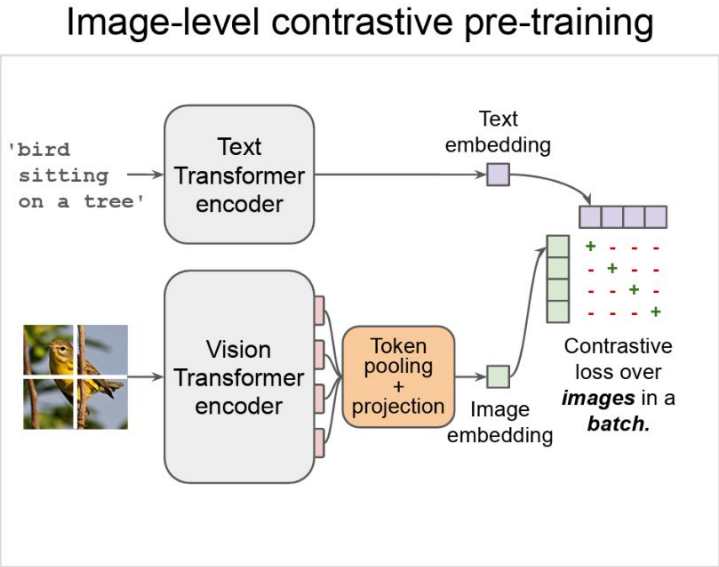
STAGE 1 **Global image–text semantics**

STAGE 2 **Object-level scores + boxes**

Georgia Tech

Stage 1: Image-Text Contrastive Pretraining

Pipeline: Data → independent encoders → global embeddings → paired-image/text supervision



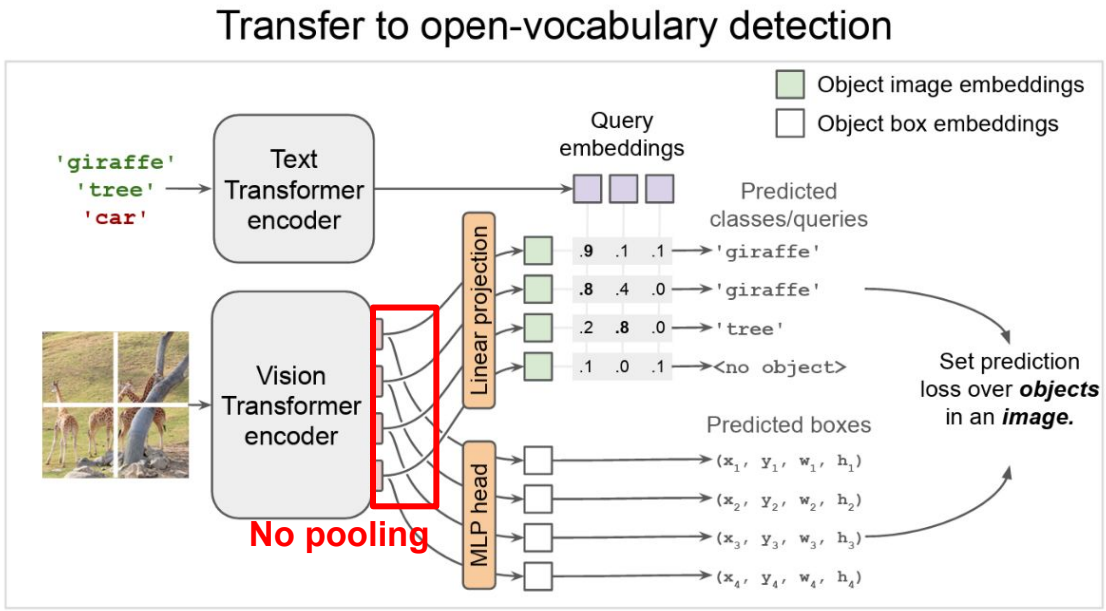
- **Data**
  Matched image–caption pairs; no boxes
- **Global Output**
  One image vector + one text vector per pair
- **Loss**
  Pairwise similarity → row / column softmax CE

CLIP-style objective; public CLIP weights are also used.

Learn reusable semantics

- Contrastive training on image–text pairs.
- Use pooled representations at this stage.
- Public CLIP checkpoints are an alternative.

Stage 2: Detection Fine-tuning

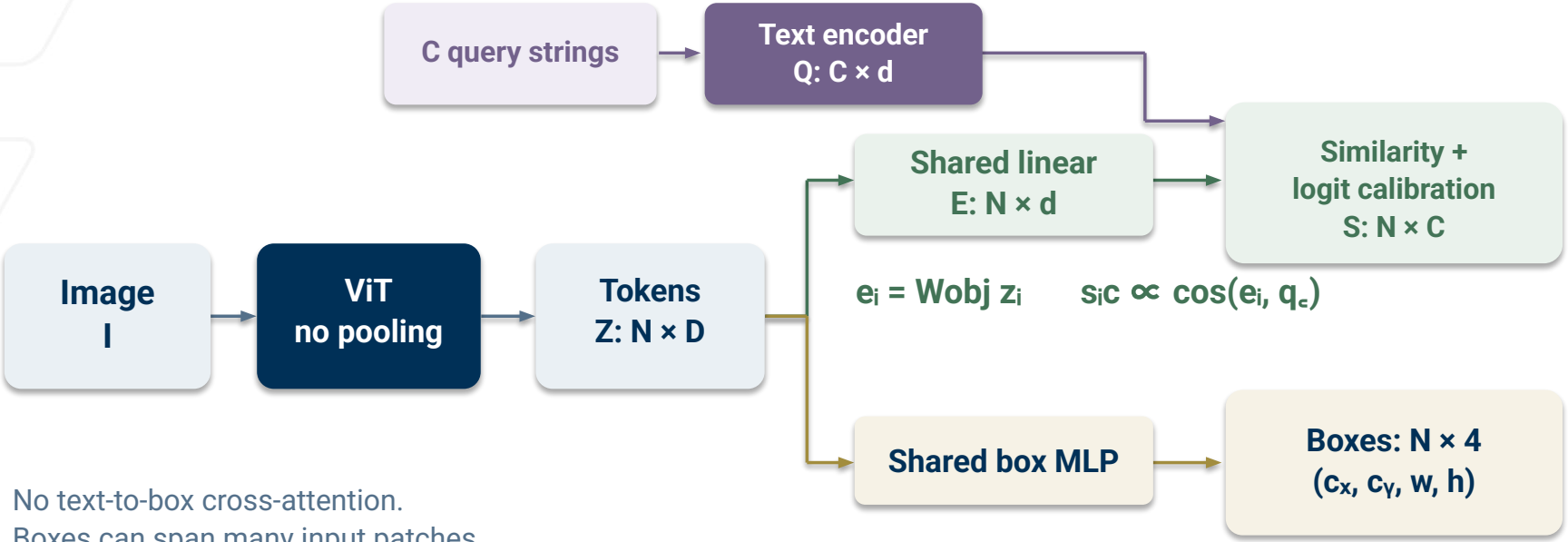


Learn object-level predictions

- Supervise with boxes + class names.
- Fine-tune both image and text encoders.
- Match objects, then train scores and boxes.

Stage 2: Forward Pass: Two Branches from the Same Token

The classifier uses text; the box branch predicts coordinates from image features



No text-to-box cross-attention.
Boxes can span many input patches.

Stage 2: Detection Labels Supervise Both Branches

Similarity is the scoring operation. The stage-2 objective is supervised set prediction

Training data

Image + annotated boxes + class names

Candidate 17	→	**“giraffe” + GT box**
Candidate 95	→	**“tree” + GT box**
Candidate 41	→	**Unmatched**

DETR-style one-to-one assignment

Class branch

Query targets → sigmoid focal loss

Box branch

Matched coordinates → L1 + GIoU

Same semantic interface; different supervision: image–caption pairs → annotated objects.

Inference: Change the Vocabulary, Not the Classifier Architecture

New query strings become new classifier vectors; no new detection head is required

Training query examples
“dog” “car” “person”

No ground-truth boxes. No Hungarian matching.
Independent text embeddings can be cached.

Test-time query
“alpaca”

→

Text encoder
q = E_text(“alpaca”)

→

Score every candidate embedding against the new query

Image-derived queries are also possible: use a reference object instead of a name.
This exploits the shared embedding interface, not an added reasoning module.

“Zero-shot” means no localized training labels for that category—not no prior exposure.

Result: Simple Transfer Reaches Held-Out Detection Categories

LVIS v1.0 val · rare-category box annotations removed from OWL-ViT detection training

LVIS-base detection training

Model / image-text pretraining	AP	Rare AP
ViLD-ens · EffNet-b7 / ALIGN	29.3	26.3
OWL-ViT · ViT-L/14 / CLIP	**34.7**	**25.6**

Different protocol: Objects365 + Visual Genome

OWL-ViT · ViT-L/14 / CLIP	**34.6**	**31.2**

Takeaway: competitive novel-category transfer—not proof of compositional grounding.

Ablations: Fine-tuning Can Improve Familiar Classes But Hurt New Ones

The transfer recipe matters—not just the small detection heads

ΔAP relative to the baseline adaptation recipe

Change to text-encoder training	LVIS rare zero-shot	OpenImages in-distribution
Same LR as image encoder	−8.5	+0.4
Freeze text encoder	−2.3	+1.2

Baseline: text learning rate = image learning rate / 100

Our reading: optimize the recipe for transfer, not only for in-distribution detection.

Table 3 + Appendix A1.9 / Table A5; R26+B/32, 70k steps

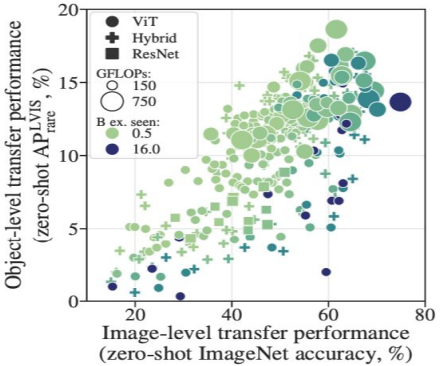
More Pretraining Helps—Only If Transfer Keeps Up

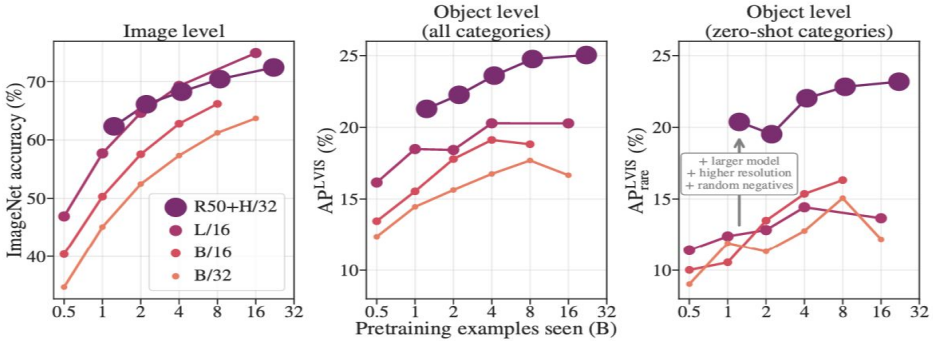
Figure 3 asks whether image-level performance predicts downstream detection quality

01 **Across architectures** High image-level accuracy is not sufficient for strong detection transfer.

02 **For a fixed small model** Detection can peak while image-level accuracy still improves.

03 **When capacity and tuning scale** Larger models and better fine-tuning can extend detection gains.

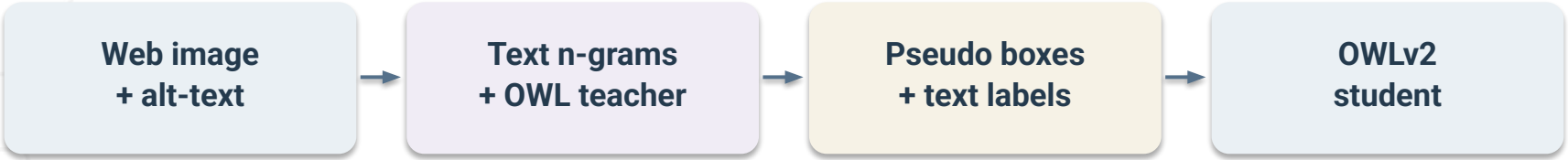




So what? Select pretraining and model size using the downstream transfer metric.

Follow-Up Work: OWLv2 + OWL-ST

OWLv2 changes training efficiency. OWL-ST is the self-training recipe that feeds it



OWLv2: spend less compute per example

Drop low-variance patches; select likely objects; pack more images with larger mosaics.

OWL-ST: expand detection supervision

Learn from web-scale pseudo annotations. Optional fine-tuning on human-labeled boxes.

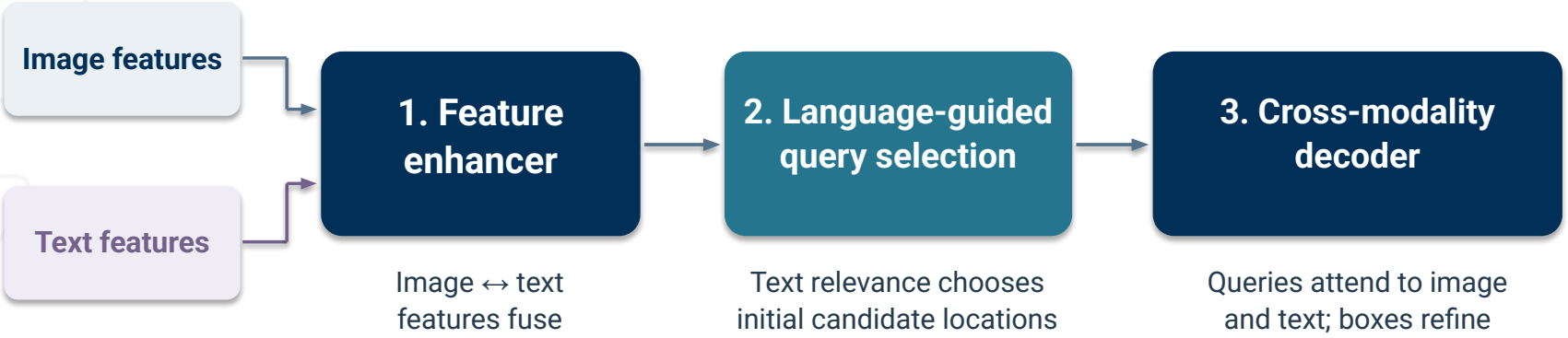
The open-vocabulary interface stays simple; the amount of localization training grows.

Scaling Open-Vocabulary Object Detection · NeurIPS 2023

Georgia Tech

Follow-Up Work: Grounding DINO

A related DETR-family branch—not a direct replacement in the OWL training pipeline



OWL: text mainly scores image-only candidates.
Grounding DINO: text also affects where candidates are formed.

DINO here is a DETR detector—not the DINO / DINOv2 self-supervised vision encoder.

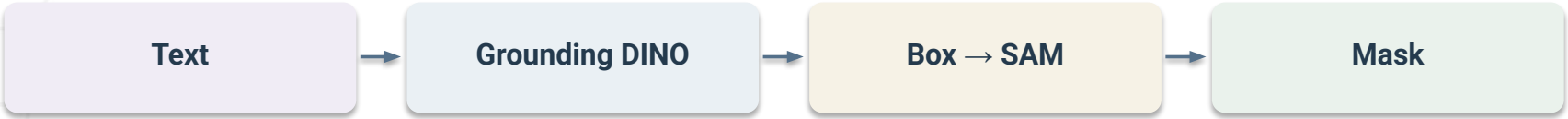
Grounding DINO · ECCV 2024

Georgia Tech

Masks Already Exist: Two Bridges Beyond Box Prediction

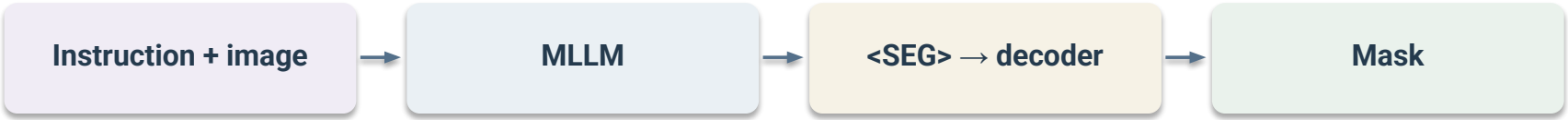
Grounded SAM

Compose specialists



LISA

Use an MLLM interface



LISA already targets reasoning segmentation; both routes already produce pixel masks.

Next question: integrate strong grounding and dense masks without a separate heavy segmenter.

Grounded SAM · 2024

LISA · CVPR 2024

Georgia Tech

Next: Turn Grounded Intent Into the Target's Pixels

"Segment the red mug beside the plate."

Interpret the instruction and ground the target → **Use the grounded signals to predict a precise mask**

1. Does it select the intended instance?

2. Does it recover the target pixels?

Qwen3-VL-Seg

- **Yao et al., 2026**
  Tongyi Lab, Alibaba Group

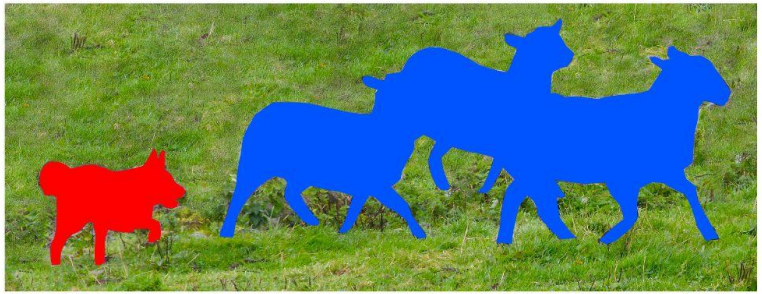
Unlocking Open-World Referring Segmentation with Vision-Language Grounding

From language-grounded boxes to pixel-level masks

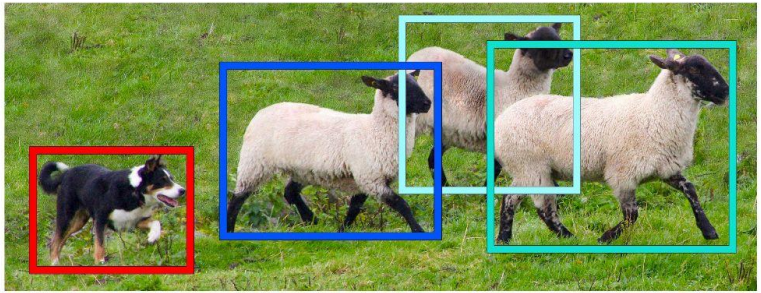
Georgia Tech

What is Image Segmentation?

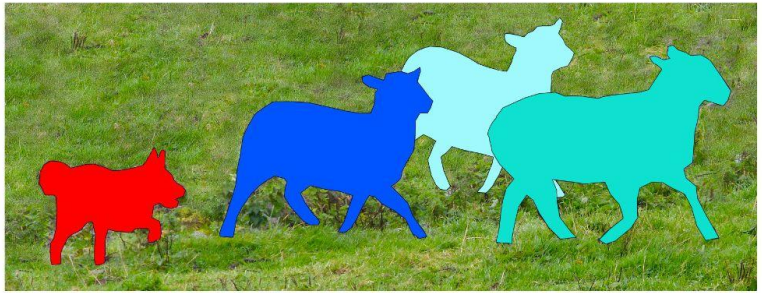
Image Recognition



Semantic Segmentation



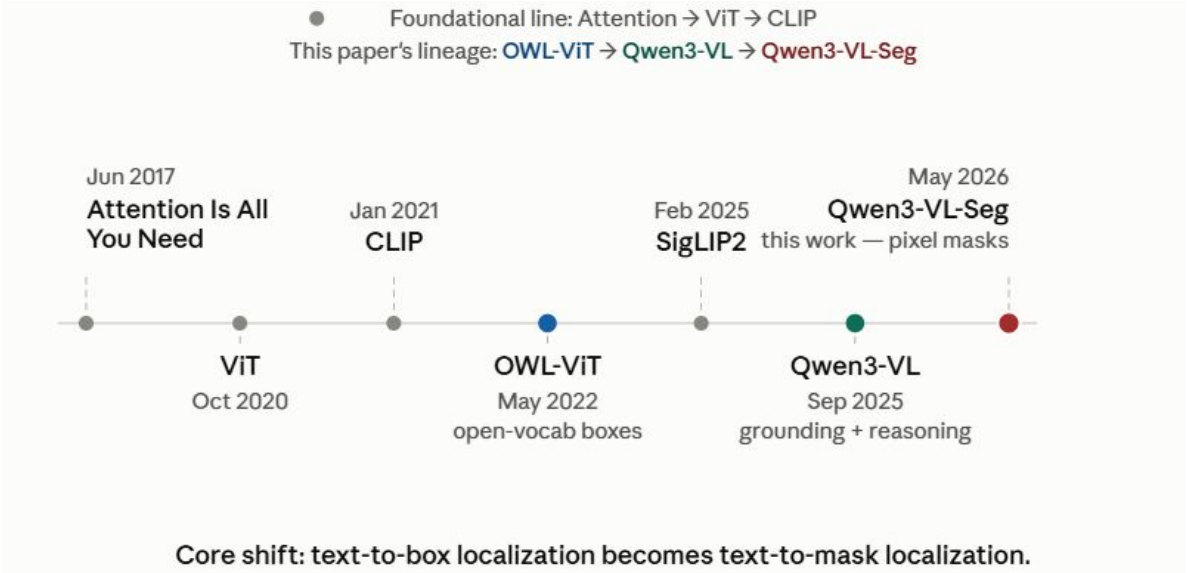
Object Detection



Instance Segmentation

Georgia Tech

Where This Paper Sits?

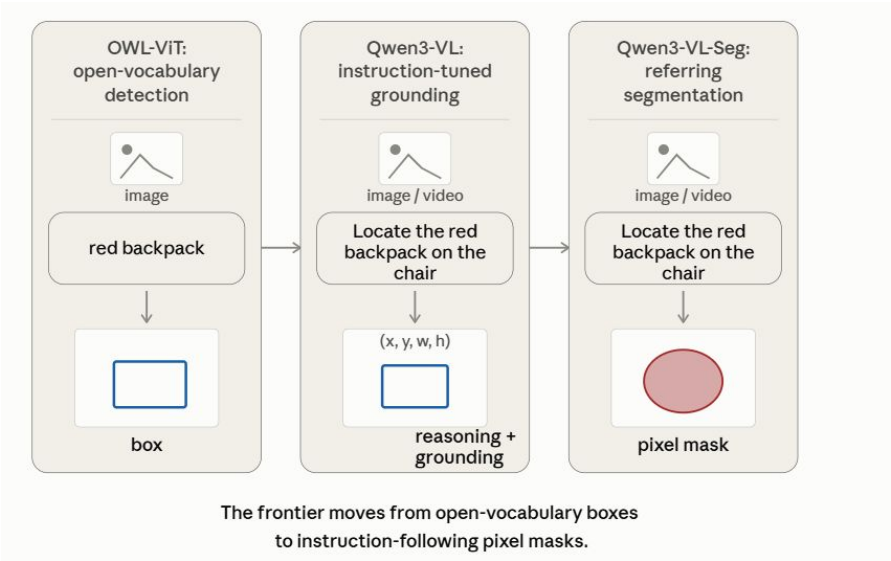


Georgia Tech

Where This Paper Sits?

OWL-ViT

- Image + text query
- Open-vocabulary object detection
- **Output:** bounding box



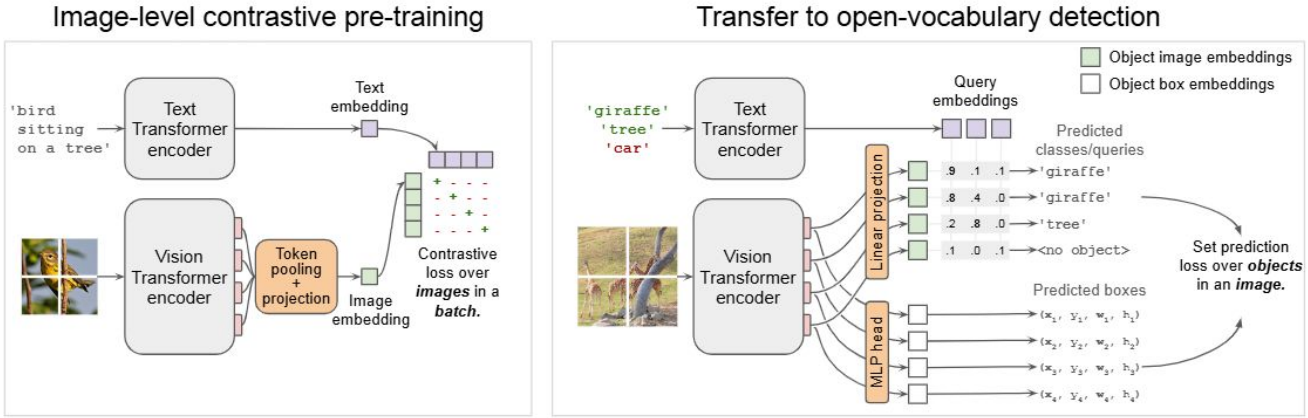
Qwen3-VL-Seg

- Image + referring instruction
- Open-world referring segmentation
- **Output:** precise pixel mask

Qwen3-VL

- Image/video + natural language instruction
- Strong multimodal reasoning and grounding
- **Output:** text, boxes, points, reasoning

OWL-ViT Recap: From Language to Open-Vocabulary Boxes



What OWL-ViT established:

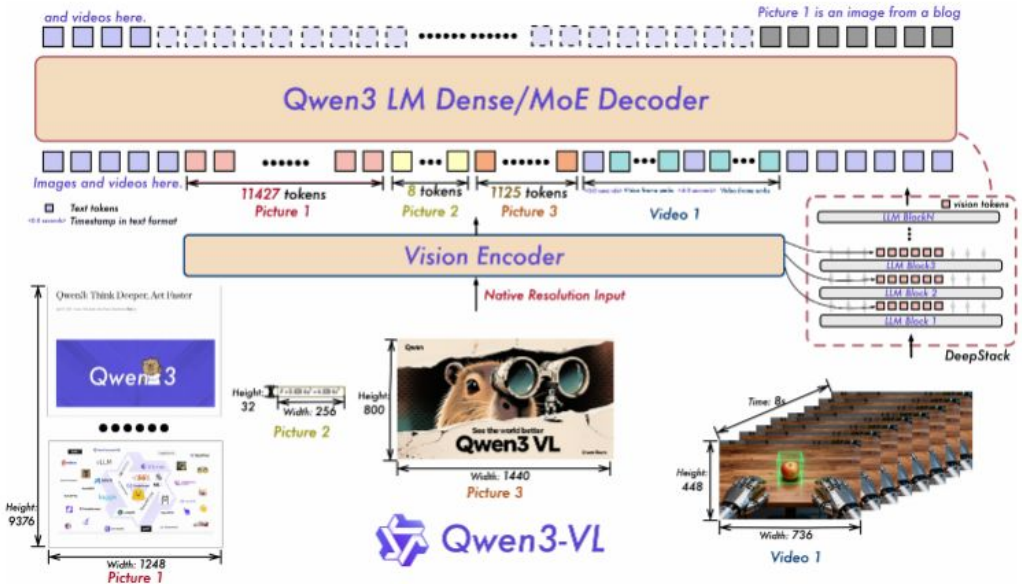
- Text embeddings act as **open-vocabulary** object queries
- Image tokens are matched against those queries
- Each image token predicts:
  - object/query **similarity**
  - **bounding-box coordinates**
- Detection is **no longer restricted to a fixed class vocabulary during inference**

Output: semantic localization at the bounding-box level

What is still missing? Exact pixel-level object boundaries.

Georgia Tech

Qwen3-VL Recap: From Multimodal Reasoning to Grounded Object Localization



What Qwen3-VL established:

- Jointly reasons over **visual content and natural-language instructions**
- Supports **fine-grained visual grounding**
- Can understand complex referring expressions, not just class names
  - *"the cup beside the laptop"*
  - *"the person wearing red behind the car"*
- Generates **spatial coordinates autoregressive as part of the language output**
- Can localize objects using:
  - bounding boxes
  - points / spatial coordinates
  - other structured grounding outputs depending on the task
- Localization is therefore driven by **semantic reasoning**, rather than only class-level similarity

What is still missing? Precise pixel-level segmentation.
Qwen3-VL can reason about and localize the referred object, but coordinate outputs do not provide its exact shape or boundary.

Output: Semantically grounded object localization

Georgia Tech

Why Do We Need Pixel-Level Masks?

Bounding box vs. pixel mask

Box

Rough location only
Includes neighbors
No shape detail

Mask

Exact pixel shape
Isolates object
Sharp boundaries

box: "red backpack" → $[x_1, y_1, x_2, y_2]$

mask: "red backpack" → $M \in \{0,1\}^{H \times W}$
1 = target pixel, 0 = background

Where masks matter

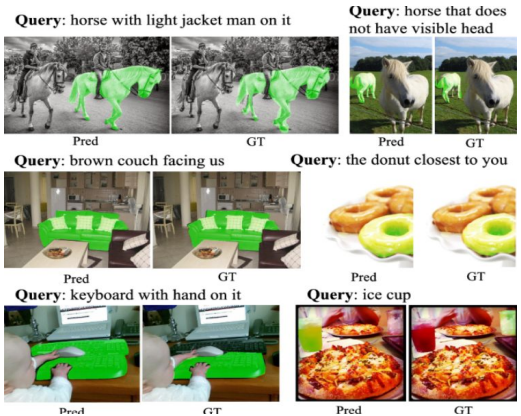
Robotics
Grasp target, not surroundings

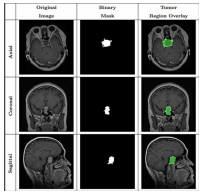
Autonomous driving
Pedestrian vs. road pixels

Image editing
Select object to edit or remove

Medical imaging
Measure tumor shape, not box

Bounding boxes answer "where is it?" — pixel masks answer "exactly which pixels are it?"





Georgia Tech

Why Not Simply Combine Qwen3-VL with an Existing Segmentation Model?

Option 1 — Qwen3-VL + SAM

query → Qwen3-VL → box → SAM → mask

SAM (Segment Anything Model): given a box or point, it cuts out a clean mask — but it can't read the sentence itself.

Why this isn't enough

- Qwen3-VL can only hand SAM a box — not "the mug behind the laptop", just a rectangle
- SAM segments whatever is in that box, blind to what was actually asked for
- Pick the wrong box (easy with tricky phrasing) and the mask is wrong too — no fixing it downstream

Also:
1) Complex Architecture
2) Inference Issues
3) Compute Intensive- costly

Option 2 — LLM-generated polygon

image+text → LLM → (x,y)... → mask

Idea: skip SAM entirely — have the LLM type out the mask's outline as coordinates, one point at a time.

Why this is difficult

- A simple outline needs a few points — a fuzzy or irregular one needs dozens
- Every extra point means more tokens generated one-by-one, and more chances to drift off-course
- Language models are built to predict words, not trace pixel-precise boundaries

Also:
1) Autoregressive - Slow inference

What we actually want

Don't force Qwen3-VL's understanding through a lossy box or a long list of coordinates — feed it straight into a decoder built to output a mask directly.

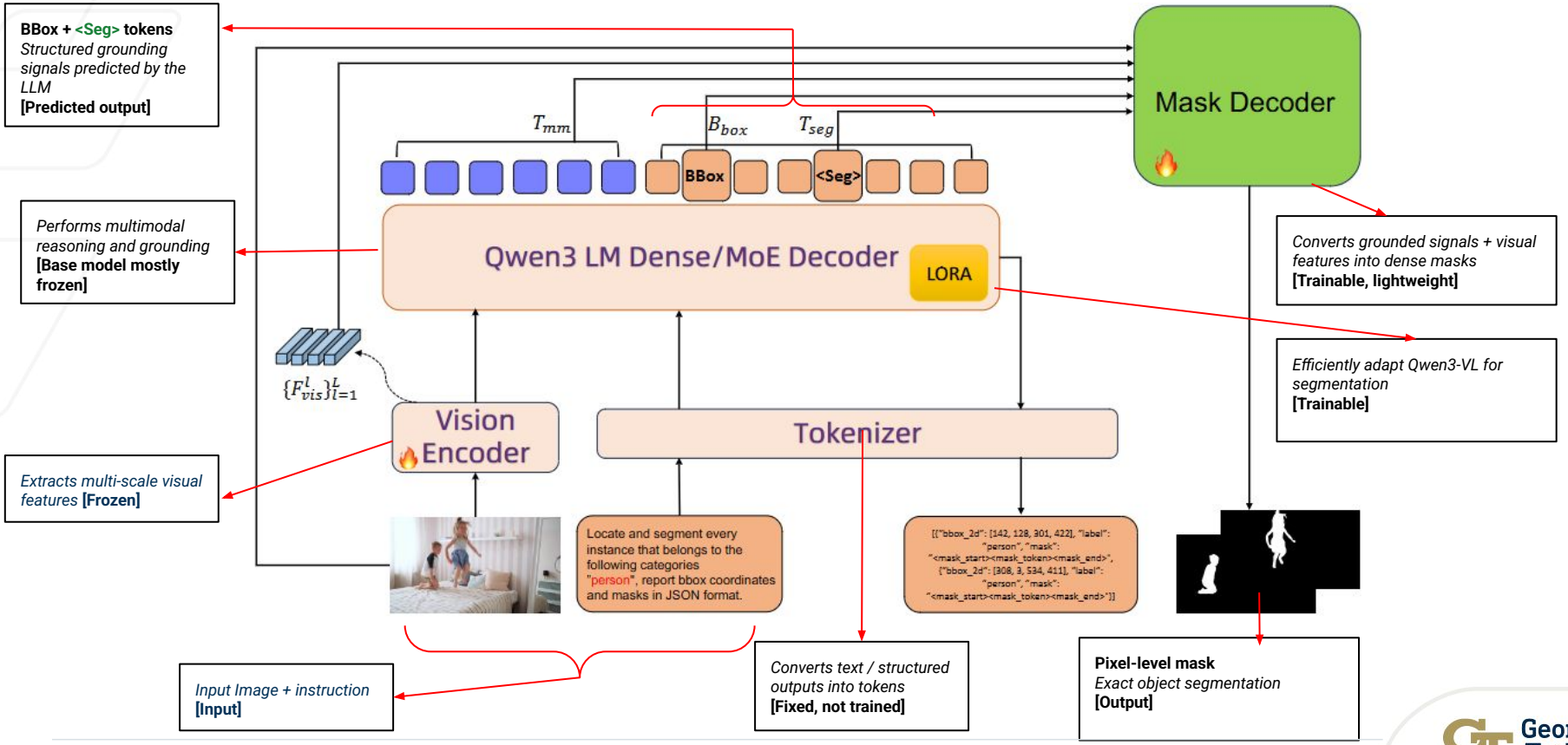
image+text → Qwen3-VL understanding → mask decoder → pixel mask

This is the motivation for Qwen3-VL-Seg.

Just 17M more parameters added to the 4B parameters from Qwen3-VL

Georgia Tech

Qwen3-VL-Seg: High-Level Architecture



Georgia Tech

Vision Encoder: Pixels to Visual Tokens

Before Qwen3-VL can reason about an image, it must convert raw pixels into a representation the language model can read. This happens in two stages: a vision transformer, then a merger.

Stage 1: the vision transformer

every layer is kept, not discarded

What each depth keeps

- Early layers: edges, textures, local shapes
- Middle layers: parts, spatial structure
- Deep layers: objects, relationships, meaning

Stage 2: the vision-language merger

The language model expects vectors in its own hidden dimension, so a merger reshapes visual features into compatible tokens, and compresses every 2x2 group of patches into one token.

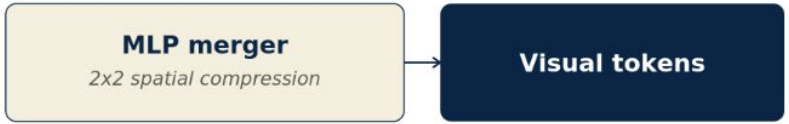


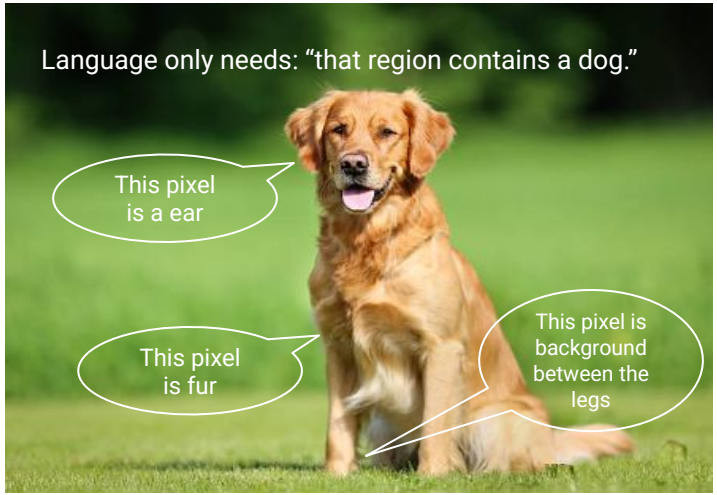
Image → patches → ViT → visual features at every depth
visual features → MLP merger (2x2 compression) → visual tokens

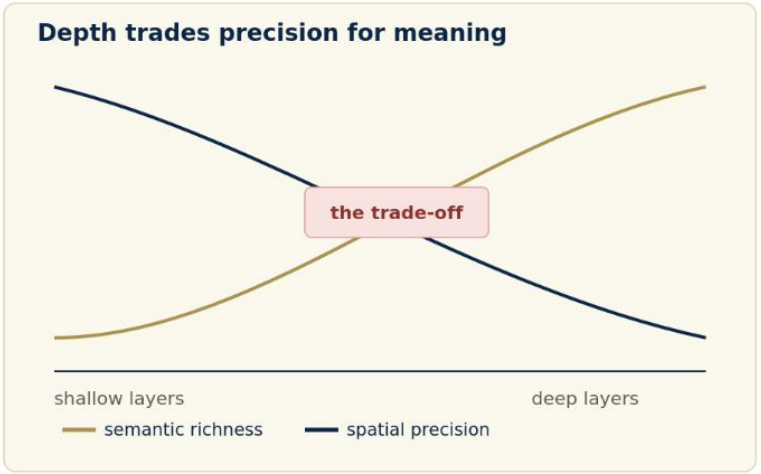
These tokens now carry both fine detail and semantic meaning, and are ready to sit alongside text.

Georgia Tech

Why Keep Every Depth?

Language understanding and pixel-level segmentation ask very different questions of the same image.
A single high-level feature map is excellent at the first question and weak at the second.





High-level features answer what an object is. Segmentation needs to know exactly where each pixel belongs, and that information is present earlier in the network, not only at the end.

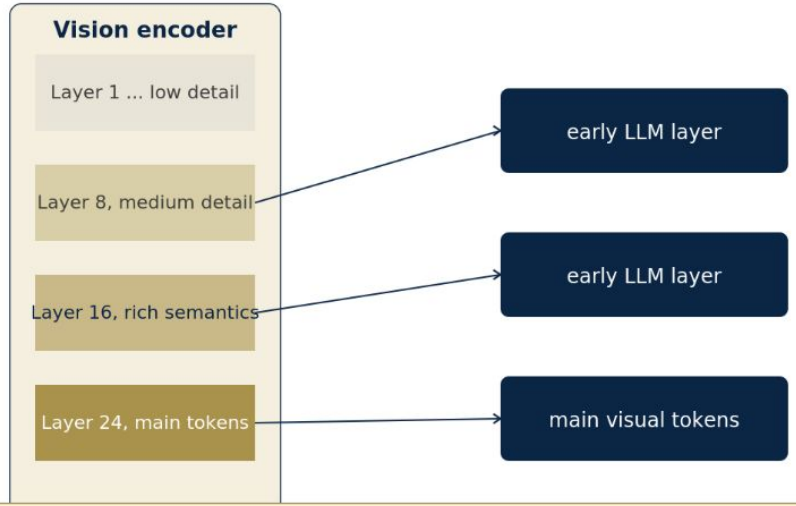
This is the motivation for going back into the vision transformer and collecting several depths at once.

Georgia Tech

DeepStack: Multiple Depths Into the LLM

Instead of sending only the final vision transformer layer to the language model, DeepStack extracts features from several depths and injects each one into a different early LLM layer.

Ordinary VLM: F^{L} → one LLM layer, one injection point



DeepStack, formally

$\{F_{vis}^{l}\}_{l=1}^{L}$ → multiple early LLM layers

Coarse detail, mid-level structure, and high-level semantics arrive side by side, rather than being compressed into one final vector before reasoning.

Each depth is routed to its own dedicated merger module and its own point of entry.

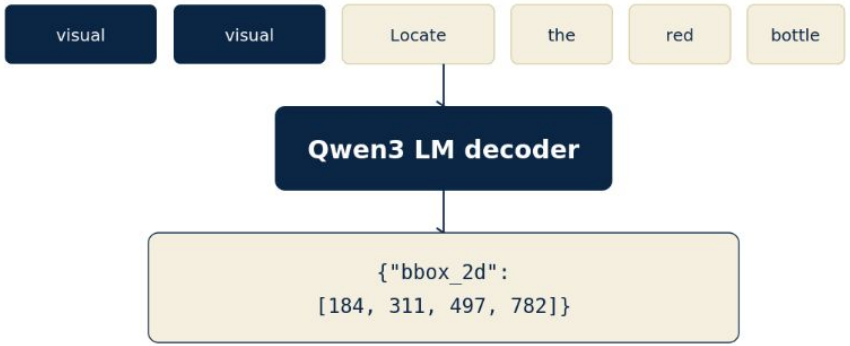
Interleaved-MRoPE, covered next, then keeps the position of every token consistent.

Qwen3-VL-Seg reuses exactly these intermediate features later, as the fine-grained visual input to the segmentation decoder, rather than collecting a separate set of features from scratch.

Georgia Tech

Joint Reasoning and Grounding in Qwen3

Visual tokens now sit inside the same sequence as the text instruction, and the Qwen3 decoder reasons jointly across both in a single forward pass, unlike OWL-ViT's separately encoded embeddings.



Only joint reasoning can resolve these

"*Which bottle is second from the bottom?*"

requires comparing position across several instances

"*Which person is interacting with the bicycle?*"

requires understanding a relationship, not a category

"*Find the object used to put out a fire.*"

requires reasoning about function, not appearance

The inherited ability that matters most: grounding

During multimodal training, Qwen3-VL learns to map a natural language description directly to coordinates. Given "locate the black cat sitting under the table," it already answers with a structured bounding box, without any additional segmentation-specific training.

Qwen3-VL already knows how to understand the instruction and roughly find the right object. Qwen3-VL-Seg is built to use that ability, not to relearn it from scratch.

Georgia Tech

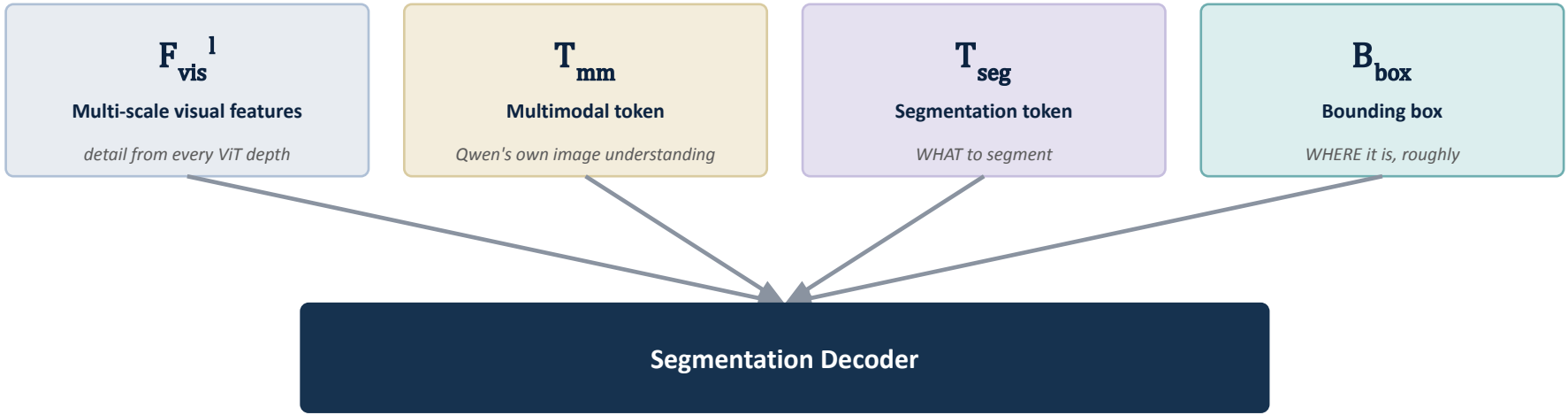
From Grounding to Segmentation

Qwen3-VL-Seg treats the bounding box predicted by Qwen3-VL not as the final answer, but as a semantic and spatial prior that guides a lightweight decoder to produce an exact segmentation mask.

Qwen3-VL already understands the instruction and roughly where the target is.
Qwen3-VL-Seg uses this knowledge while recovering precise pixel-level boundaries.

The Four Signals Feeding the Decoder

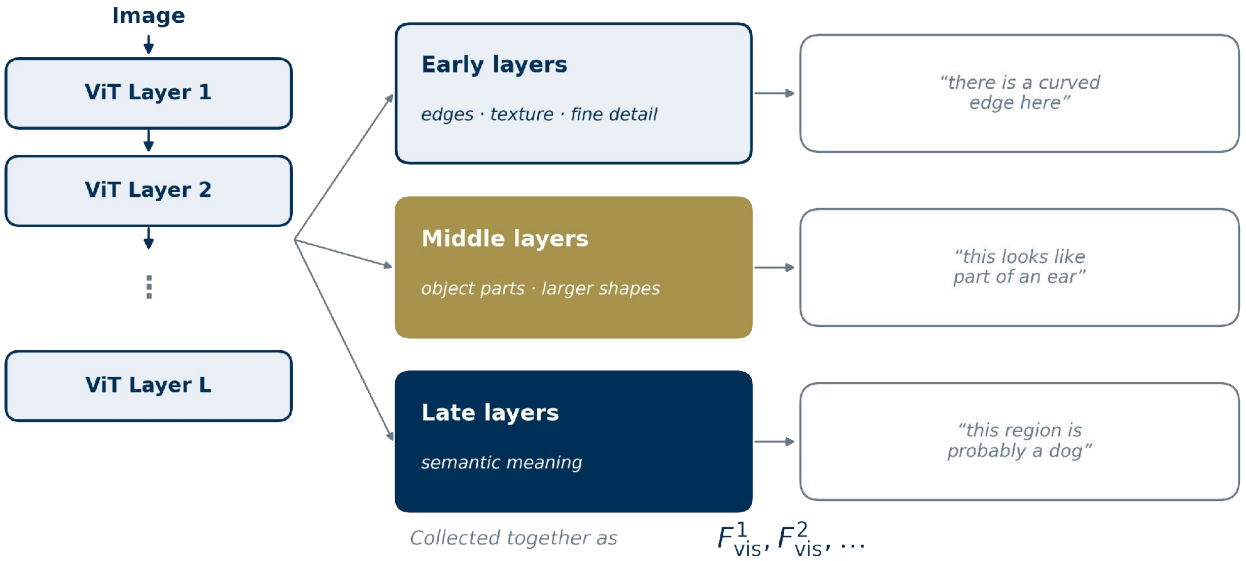
Qwen3-VL hands the segmentation decoder four pieces of information. Everything in the four components ahead is built from these.



The next four slides walk through exactly how these four signals get turned into a pixel mask: Component 1 builds a searchable memory, Component 2 builds a query, Component 3 recovers fine detail, and Component 4 predicts and refines the mask.

Signal 1: Multi-Scale Visual Features

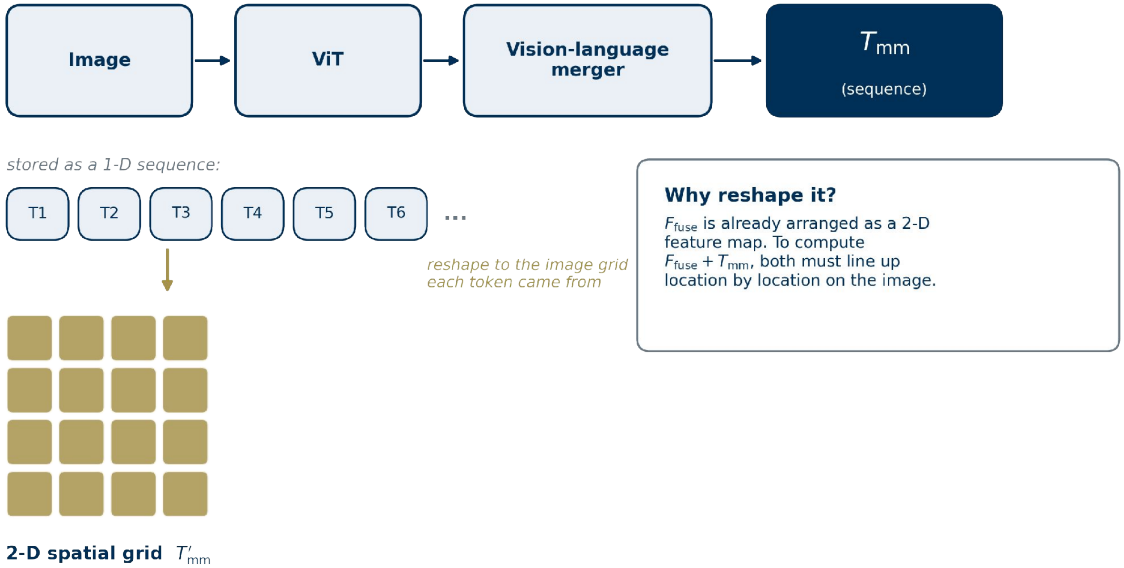
Different ViT depths see different things. Segmentation needs all of them, not just the last layer's semantics.



Georgia Tech

Signal 2: The Multimodal Token

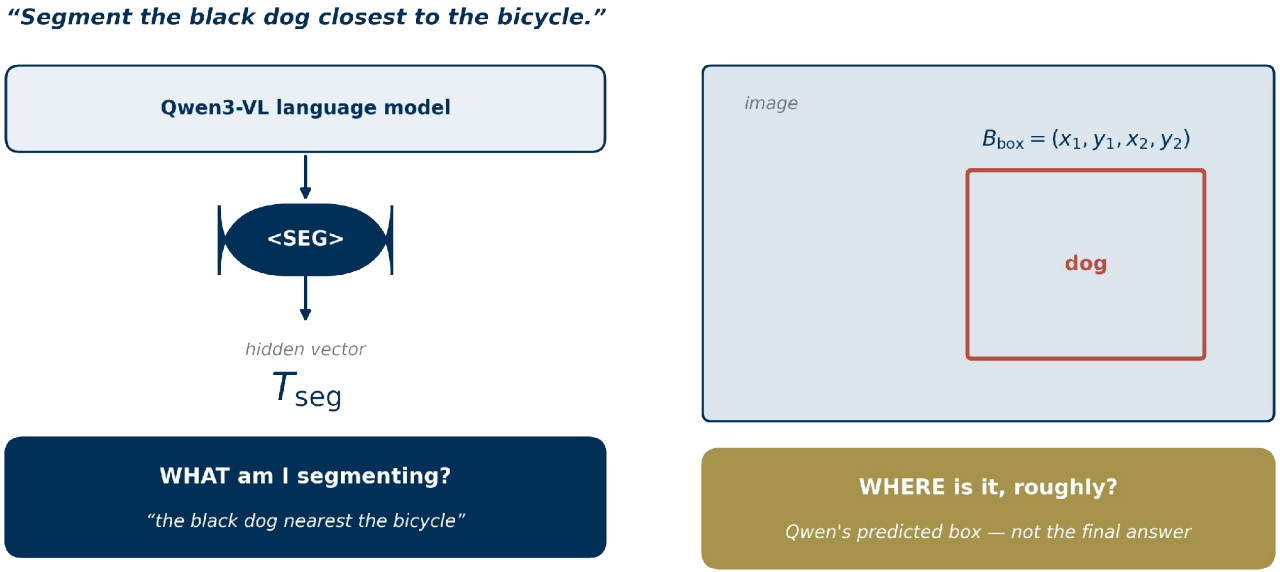
Qwen's high-level image understanding, reshaped back into the 2-D grid it originally came from.



Georgia Tech

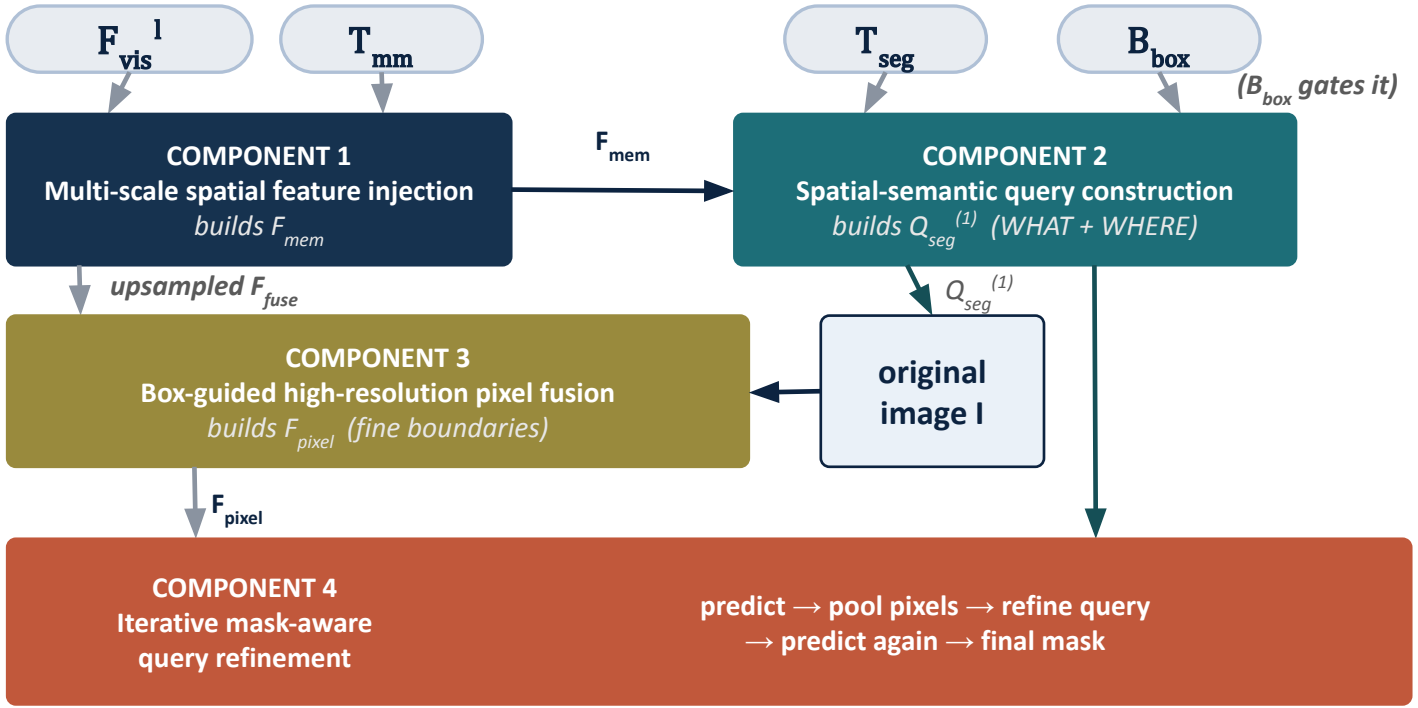
Signals 3 & 4: What and Where

The <SEG> token's hidden state says WHAT to segment; Qwen's predicted box says roughly WHERE.



The Decoder Architecture at a Glance

Four components turn those signals into a pixel-accurate mask: first build a memory, then search it.



Qwen3-VL-Seg Decoder: See->Target->Zoom->Refine

1. SEE: Build a rich visual map

Component 1: Multi-scale Spatial Feature Injection

For this image:
"Understand the whole scene: dog, trees, ground, edges, body parts."

Analogy:
Look at the scene at multiple levels of detail.

2. TARGET: Decide exactly what to segment

Component 2: Spatial-Semantic Query Construction

For this image:
"Segment **this dog**, approximately **inside this region**."

Analogy:
Point to the object you actually care about.

3. ZOOM: Recover precise pixel details

Component 3: Box-Guided High-Resolution Fusion

For this image:
"Zoom in around the dog to recover its ears, legs, fur edges, and body boundary."

Analogy:
Use a magnifying glass around the target.

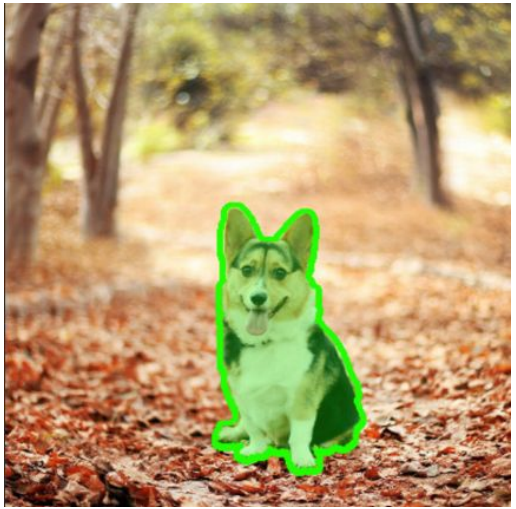
4. REFINE: Predict, inspect, and correct

Component 4: Iterative Mask-Aware Refinement

Predict a first mask → inspect the pixels selected → improve the target representation → predict again.

Analogy:
Trace the dog once, inspect your outline, then redraw it more accurately.

User Prompt: *"Segment the dog sitting on the leaf-covered path."*

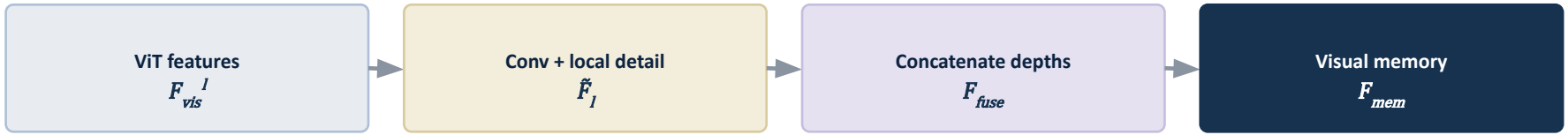


Component 1: The Visual Memory

1 Memory — 2 Query — 3 Pixels — 4 Refine

Step 1 of 4. Before the decoder can decide what to segment, it needs something to search.

Starting point: Qwen3-VL already handed off a box and a rough idea of what to look for. Component 1 turns the ViT's raw features into a map worth searching.



reshape every depth to a common format

$$X_0^{(l)} = \mathrm{Conv}_{1\times 1}(F_{vis}^{l})$$

- Different ViT depths see different things: edges early, parts in the middle, meaning late. All of them get kept, not just the last layer.

add cheap local edge detail, gently

$$\tilde{F}_l = X_0^{(l)} + s \cdot \mathrm{GELU}(\mathrm{DWConv}(\mathrm{GroupNorm}(X_0^{(l)})))$$

- The local detail branch is a residual that starts almost off ($s \approx 0.001$), so it barely touches Qwen's pretrained features until training shows it is worth it.

- Depths are concatenated, not summed, so a learned layer decides how much of each depth actually helps, instead of blending them by default.

combine into the searchable memory

$$F_{mem} = T'_{mm} + F_{fuse} + P_{mem}$$

- The 256-channel target is deliberately small. Keeping each depth's adapter output compact is what makes concatenating four of them affordable.

WHAT EACH SYMBOL MEANS

F_{vis}^{l} the raw ViT feature at depth l, before any adaptation

$X_0^{(l)}$ F_vis at that depth, reshaped to a shared 256-channel format

$\tilde{F}_l$ X0 plus a small, cheap local-detail correction

F_{fuse} every depth's corrected feature, concatenated and mixed together

T_{mm}' Qwen's own image understanding, reshaped back into the 2D grid

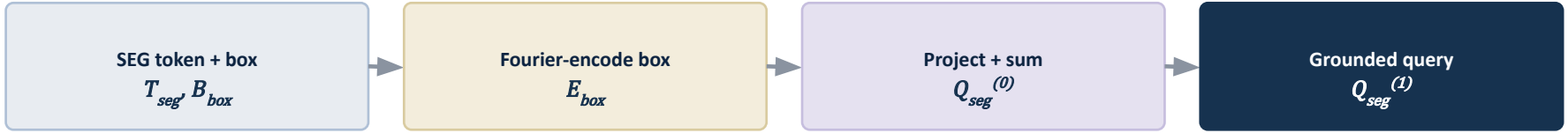
P_{mem} learned position embeddings; this decoder runs its own cross-attention, so it needs its own sense of location

F_{mem} the finished visual memory, ready for Component 2 to search

Component 2: Building the Query

Step 2 of 4. With a memory to search, the decoder now has to decide exactly what it is looking for.

Component 1 handed off F_{mem}, a searchable map of the whole image. Component 2 turns Qwen's box and its own SEG token into a single query that can search it.



turn 4 raw numbers into a richer signal

$$E_{box} = \gamma(x_1) \oplus \gamma(y_1) \oplus \gamma(0.2\log w + 0.5) \oplus \gamma(0.2\log h + 0.5)$$

- Four raw box coordinates are a weak signal on their own, so a Fourier encoding, plus a log on width and height, gives the network something richer to reason about.

combine WHAT and WHERE into one query

$$Q_{seg}^{(0)} = \text{LayerNorm}(\text{MLP}_{box}(E_{box}) + W_{seg}\, T_{seg})$$

- The box mostly exists to disambiguate. “Black dog” is ambiguous with three dogs in frame; “black dog, roughly here” usually is not.

ground the query in the visual memory

$$Q_{seg}^{(1)} = \text{Decoder}(Q_{seg}^{(0)},\ F_{mem})$$

- The query only becomes useful once it cross-attends over Component 1's memory. Before that step it is still just a description, not a grounded target.

- Notice what is missing: there is no text-to-box cross-attention here. The box stays purely additive context until Component 3 uses it to gate pixels.

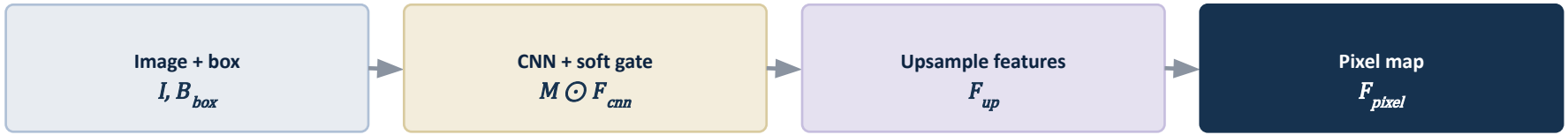
WHAT EACH SYMBOL MEANS

- T_{seg} hidden state of the SEG token: the WHAT
- B_{box} Qwen's predicted box: the WHERE, roughly
- $\gamma(\cdot)$ Fourier encoding; turns one number into sine and cosine at several frequencies
- $\oplus$ concatenate the encoded pieces together
- E_{box} the Fourier-encoded box: x1, y1, log-width, log-height
- W_{seg}, MLP_{box} learned projections into the decoder's shared space
- $Q_{seg}^{(0)}$ the query before it has looked at the image at all
- $Q_{seg}^{(1)}$ the query after grounding itself in F_mem

Component 3: High-Resolution Pixel Fusion

Step 3 of 4. The query knows what it wants. It still cannot trace an exact edge.

Component 2 grounded the query in F_{mem}, but ViT features are coarse by construction. Component 3 goes back to the raw pixels to recover the boundary detail that got lost along the way.



fade in the box instead of cutting hard

$$M(x,y) = \sigma(\alpha(x-x'_1))\,\sigma(\alpha(x'_2-x))\,\sigma(\alpha(y-y'_1))\,\sigma(\alpha(y'_2-y))$$

combine fine detail with upsampled semantics

$$F_{pixel} = F_{up} \oplus (M \odot F_{cnn})$$

- Transformer features are good at "what" and bad at "exactly where." A shallow CNN on the raw pixels is what recovers the fine boundary detail.
- The gate is deliberately soft ($\alpha \approx 20$), not a hard crop, so a slightly wrong box does not silently delete real pixels sitting near the edge.
- The result gets concatenated, not blended, with the upsampled semantic features from Component 1, so neither signal has to compromise.
- F_{cnn} is computed once for the whole image and reused. The point was never that a CNN is expensive, it is that ViT features alone were never going to have this resolution.

WHAT EACH SYMBOL MEANS

I the raw input image

F_{cnn} fine edge and texture detail from a small CNN, computed once for the whole image

M(x,y) the soft gate: high inside the enlarged box, fading smoothly at the edges

α controls how sharply the gate fades (≈ 20 here)

⊙ multiply elementwise

⊕ concatenate, not add

F_{up} Component 1's fused features, upsampled to match resolution

F_{pixel} the finished high-resolution map the mask gets drawn on

Component 4: Predict, Pool, Refine

1 Memory — 2 Query — 3 Pixels — 4 Refine

Step 4 of 4. Everything so far becomes a mask, and the model gets a second look at its own answer.

Component 2 built the query. Component 3 built the pixel map. Component 4 is where those two finally meet, produce a mask, and then get a chance to correct it.

Query + pixel map
$Q_{seg}^{(1)}, F_{pixel}$

→ **Compare to every pixel**
$\Psi \rightarrow \sigma(\cdot)$

→ **Pool + refine query**
$F_{tar}, Q_{seg}^{(2)}$

→ **Final mask + IoU**
M, s_{IoU}

pool the confident pixels into a description

$$F_{tar} = \frac{\sum_{h,w} \sigma(M_{logit}^{(1)}) \odot F_{pixel}}{\sum_{h,w} \sigma(M_{logit}^{(1)}) + \varepsilon}$$

use that description to sharpen the query

$$Q_{seg}^{(2)} = \text{LayerNorm}(Q_{seg}^{(1)} + \phi_{ref}(F_{tar}))$$

the model's own guess at mask quality

$$\hat{s}_{IoU} = W_{IoU} Q_{seg}^{(2)}$$

- The first mask is treated as a rough draft. Its confident pixels get pooled into a description of what the target actually looks like.
- That description updates the query for a second pass, so the model is not starting over. It is correcting itself with everything it already computed.
- The IoU score has no access to ground truth. It is the model's own confidence, useful as a filter, not as proof that the mask is right.
- There is no third pass. Two looks is a deliberate stopping point, presumably a compute and accuracy trade-off, though the paper never explains why two and not three.

WHAT EACH SYMBOL MEANS

- Ψ compares the query to every pixel location in F_pixel
- $M_{logit}^{(1)}$ raw per-pixel compatibility scores from the first pass
- $\sigma(\cdot)$ sigmoid; turns logits into a 0 to 1 confidence map, the first mask
- $\odot$ multiply elementwise
- F_{tar} a confidence-weighted average of F_pixel
- ε a small constant that only prevents dividing by zero
- ϕ_{ref} reshapes F_tar so it can combine with the query

Putting It All Together

Four components, four questions, one mask.

Component	Input	Output	Question it answers
1. Spatial feature injection	ViT layers + T_{mm}	F_{mem}	What visual information exists across the image?
2. Spatial-semantic query construction	T_{seg} + B_{box} + F_{mem}	$Q_{seg}^{(1)}$	Which specific object do I want?
3. High-res pixel fusion	image + box + F_{fuse}	F_{pixel}	Where are the precise boundaries?
4. Mask-aware refinement	$Q_{seg}^{(1)}$ + F_{pixel}	final mask $\hat{M}$	Which exact pixels belong to the target?

SA1B-ORS: A Two-Part Open-World Referring Dataset

Built from 2M raw SA-1B images; two complementary subsets trade off category breadth against instance-level precision.

Subset	Samples	What it supervises	Example expression
SA1B-CoRS	1.05M	Category name → one or more entities of that class	“person”, “car”
SA1B-DeRS	1.94M	Attributes / relations / context → a single unique target	“the person with long red hair holding the man in the brown jacket”

Scale and composition (vs. RefCOCO / LVIS and across instruction types):

2M → 3M samples
Sampled from SA-1B; orders of magnitude more categories and training samples than RefCOCO or LVIS

64.8% descriptive
vs. 29.6% single-instance category and 5.5% multi-instance category instructions

Long-tailed categories
person, building, sky dominate — mirrors real open-world scene statistics

Curation Pipelines: SA1B-CoRS vs. SA1B-DeRS

Both pipelines start from raw SA-1B images and end in MLLM-verified referring supervision — CoRS scales category coverage, DeRS resolves instance-level ambiguity.

Stage	SA1B-CoRS (5 stages → 1.05M)	SA1B-DeRS (3 stages → 1.94M)
1	Instance Distillation — RAM+ tagging, semantic filtering, grounding verification, hierarchical NMS	Instruction Curation — MLLM compares original vs. highlighted image, writes a discriminative referring expression
2	Coarse Mask Acquisition — VL grounding + SAM2 mask	Cognitive Verification — MLLM re-grounds the instruction; discard if IoU < 0.8 vs. ground truth
3	Fine Mask Merging — fragment absorption, pixel-level refinement	Saliency Selection — discard instances with a small mask-to-image area ratio
4	MLLM Verification — alignment check, integrity check	
5	Caption Generation — convert verified mask into a referring expression	

ORS-Bench: In-Distribution and Out-of-Distribution Evaluation

Pairs an in-distribution benchmark with six out-of-distribution stress tests to probe the operational limits of referring segmentation.

ORS-ID-Bench	Samples
Single-instance category	2,465
Multiple-instance category	1,823
Phrase (RefCOCO/+/g)	2,946
Descriptive	1,821
Total	**9,055**

ORS-OOD-Bench	Distribution shift
Category OOD	Categories absent or $<10^{-4}$ frequency in training
Area OOD	Extreme area ratio: <0.002 or >0.7 of image
Instruction OOD	Indirect / negative / interrogative / relational / long-context phrasing
Occlusion OOD	High-occlusion targets
Lighting OOD	Low-light / nighttime captures
Risk-Sensitive Scene	Autonomous driving, medical diagnosis — rare in training

200 curated samples per OOD dimension (1,200 total) — every sample manually verified for mask/instruction alignment.

Two Training Stages, Each Freezing the Opposite Half

Stage 1 changes the vision system. Stage 2 changes the language system. Never both at the same time.

Starting point: Qwen3-VL-4B already reasons and talks about images. It just cannot draw a pixel mask, so a new 17M-parameter decoder gets added to learn that one skill.

	STAGE 1 Learn to segment (10,000 steps)	**STAGE 2** Recover reasoning (5,000 steps, tiny learning rate)
Original LLM Weights	**FROZEN** A small LoRA patch adapts it instead of changing it directly	**TRAINABLE** The patch gets folded in, then the full model can update
Vision Encoder	**TRAINABLE** Learns pixel detail, but 100 times slower than everything else	**FROZEN** Keeps the pixel skills Stage 1 just built
Mask Decoder (new, 17M params)	**TRAINABLE** Starts from nothing and learns to draw masks	**TRAINABLE** Keeps training so those mask skills do not fade

Why split it into two stages: doing it all at once makes the model worse at general reasoning (MMMU drops from 67.4 to 63.4). Two stages let Qwen learn to segment first, then recover its reasoning without losing that new skill.

Training Recipe and Implementation

Backbone: Qwen3-VL-4B. The mask decoder adds only 17M parameters (~0.4% of the base MLLM).

Stage	Trainable	Data mix	Iters	LR schedule
1 — Segmentation-Centric Adaptation	LoRA (rank 32) + vision encoder + mask decoder; LLM backbone frozen	Public RES datasets + SA1B-ORS	10,000	1e-4 cosine (vision encoder 0.01×)
2 — Synergistic Enhancement	LLM backbone (LoRA merged) + mask decoder; vision encoder frozen	3:1:2 mix — referring seg. : general multimodal : reasoning	5,000	7e-7 cosine decay

- Objective: text-generation loss + segmentation loss (weighted BCE + DICE on predicted masks).
- Instruction format: ChatML template; the model emits structured JSON per instance — bbox_2d, label, and a mask placeholder token (<mask_start><mask_token><mask_end>) that bridges to the mask decoder.
- Stage 1 builds raw segmentation capability but can regress general reasoning/OCR skills; Stage 2 restores that balance while keeping segmentation quality.

Referring Expression Segmentation & Comprehension

Closed-set evaluation on RefCOCO / RefCOCO+ / RefCOCOg — segmentation (cIoU) and comprehension (Prec@0.5).

RES — cIoU (Val)

Method	RefCOCO	RefCOCO+	RefCOCOg
SAM3	75.5	67.3	73.4
MLLMSeg	79.5	75.4	78.1
Youtu-VL	80.7	76.2	76.5
Unipixel	80.5	74.3	76.3
Qwen3-VL-Seg (Ours)	**82.3**	**76.2**	**78.2**

REC — Prec@0.5 (Val)

Method	RefCOCO	RefCOCO+	RefCOCOg
Florence-2	93.4	88.3	91.2
Grounding DINO	90.6	82.8	86.1
Youtu-VL	93.6	90.1	92.2
MLLMSeg	91.6	86.0	88.3
Qwen3-VL-Seg (Ours)	**92.7**	**87.8**	**89.1**

Best on 6 of 8 RES splits without an external segmenter (e.g. +7.4 cIoU over LISA on RefCOCO Val); SOTA among MLLM-based REC methods, closing the gap to specialist Vision Generalist models.

Open-World Referring Segmentation (ORS-ID-Bench)

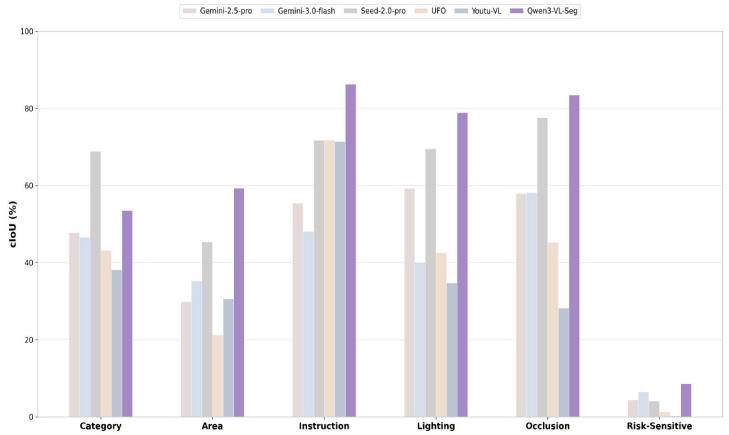
cIoU / Prec@0.5 across four open-world instruction formats (best result in each column in gold).

Method	Single-Inst.	Multi-Inst.	Phrasal	Descriptive
Gemini-2.5-pro	77.7 / 85.1	75.9 / 83.8	58.7 / 73.1	44.6 / 45.4
Youtu-VL	77.6 / 86.4	—	53.2 / 69.0	73.7 / 70.9
SAM3	94.2 / 97.3	94.6 / 91.4	66.6 / 79.1	75.5 / 78.2
Qwen3-VL-Seg (Ours)	**96.0 / 97.4**	**93.2 / 91.6**	**82.8 / 94.3**	**91.0 / 91.3**

The gain widens as instructions get more language-intensive:
+19.0 cIoU / +13.6 P@0.5 over the best baseline on phrasal instructions; +15.5 cIoU / +13.1 P@0.5 on descriptive instructions.

General-purpose MLLMs and segmentation-centric models both lose accuracy once a target can't be named by a simple category — Qwen3-VL-Seg's box-guided decoder keeps semantic grounding and pixel-level boundaries aligned even then.

OOD Generalization & General Multimodal Competence



Benchmark	Instruct	Stage 1	Stage 2 (Ours)
MMBench-EN	83.9	86.2	84.2
MMMU (val)	67.4	63.4	66.2
DocVQA (val)	95.3	93.8	94.1
RefCOCO (val)	**91.6**	**91.8**	**92.3**

Stage 2 restores general multimodal competence lost during Stage-1's segmentation-centric adaptation, while keeping RefCOCO grounding strong.

cIoU of 53–86% across five OOD dimensions — category, area, instruction, lighting, occlusion — but only ~8.6% on risk-sensitive scenes (autonomous driving, medical imaging).

Consistently outperforms general MLLMs and unified perception models on 5 of 6 OOD axes — risk-sensitive scenes remain an open challenge for every method compared.

Georgia Tech

Ablation: What Makes the Mask Decoder Work

RefCOCO (Val); the same trend holds on RefCOCO+ and RefCOCOg (see paper Table 5).

Variant	mIoU	cIoU	P@0.5	P@0.7	P@0.9
Qwen box + SAM	74.3	70.3	83.2	76.9	39.6
Ours (freeze ViT)	81.8	80.9	92.8	87.0	44.1
Ours (w/o multi-scale ViT)	82.7	81.9	92.7	88.7	49.0
Ours (w/o high-res image branch)	82.7	81.9	92.8	88.7	48.0
Ours (full)	**82.8**	**82.3**	**92.8**	**88.7**	**50.2**

Every component earns its parameters.
Adapting the vision encoder, injecting multi-scale features, and fusing the high-resolution image branch each contribute — gaps are largest on P@0.9 (39.6 → 50.2), the metric most sensitive to boundary quality.

Limitations

1

Risk-sensitive scenes remain unsolved

cIoU falls to just ~8.6% on autonomous-driving and medical-imaging queries — every method tested fails here, not only ours.

2

Mask quality inherits box quality

The decoder treats the MLLM-predicted box as ground truth; a wrong or imprecise box propagates directly into the final mask.

3

Two-stage recipe adds complexity

Stage 1's segmentation-centric LoRA adaptation regresses general reasoning/OCR; a full second fine-tuning stage is needed to recover it.

4

Training data is machine-verified, not human-labeled

SA1B-ORS's 3M samples are curated and verified by an MLLM (Qwen3-VL-Plus); only the much smaller ORS-Bench eval set is manually checked.

The gains are real, but bounded by the model's own grounding accuracy and the domains it was trained to see.

Conclusion

- Qwen3-VL-Seg: a lightweight framework (17M params, ~0.4% overhead) extending a pretrained MLLM from box-level grounding to pixel-level referring segmentation — no external segmenter (e.g. SAM) required.
- Key idea: treat the MLLM-predicted box as a structural prior, and let a box-guided decoder progressively refine it into a mask.
- SA1B-ORS + ORS-Bench: scalable open-world training data (3M samples) and a rigorous in-distribution / out-of-distribution evaluation suite.
- Strong across closed-set and open-world settings, with the clearest gains on language-intensive instructions and OOD generalization.
- Broadly preserves general-purpose multimodal competence after segmentation-oriented adaptation.

Takeaway: reusing an MLLM's own grounding priors as structural priors is an effective path to unified, efficient referring segmentation.

Our Take: Qwen3-VL-Seg

What We Liked

Reused box-guided scaffolding.
Grounding does the localization and a lightweight decoder turns it into a mask, instead of a segmentation head built from scratch.

Forgetting-aware two-stage training.
Freezing the opposite half each stage protects both the model's original VQA ability and its new mask skill.

Honest evaluation.
ORS-Bench separates in-distribution from OOD cases, and the ablation isolates what the decoder actually contributes.

What Felt Lacking

Machine-verified referring expressions.
SA1B-ORS labels come from automated pipelines, not human review, which can leave noisy or unnatural phrasing in training data.

Query complexity gap persists.
Compound and out-of-distribution queries still drag performance down; the paper names the gap without closing it.

Mask quality inherits box quality.
Errors in the upstream grounding box carry straight into the mask, and how often that happens is not closely measured.

Net take: a well-tested extension of grounding into pixel space, still only as good as the box it starts from.