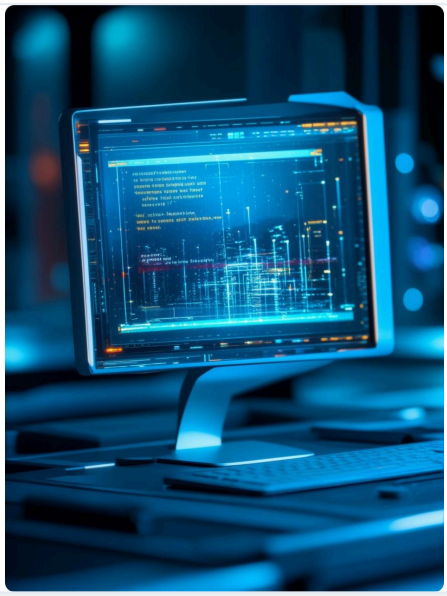
3. Multimodalidad: unir visión y lenguaje

CLIP · BLIP · Grounding DINO · VLM (Qwen2.5-VL)

Los modelos fundacionales multimodales representan un salto cualitativo: combinan visión y lenguaje para entender instrucciones en texto natural, describir imágenes, detectar objetos mediante descripciones verbales, y razonar sobre contenido visual.



Modelos fundacionales

Definición

Son modelos preentrenados a gran escala que aprenden representaciones generales y transferibles.

Importancia

Capturan patrones robustos gracias al preentrenamiento masivo y funcionan bien incluso en zero-shot.

Capacidades

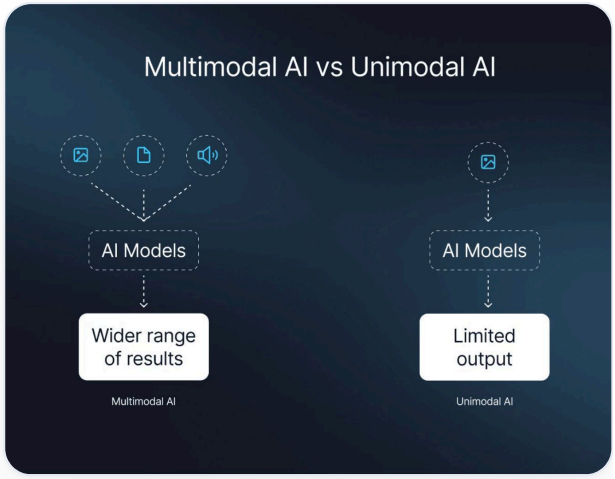
Permiten reutilizar el conocimiento en muchas tareas, reducen anotación y evitan entrenar desde cero.

Ejemplos

- **CLIP:** 400 millones de pares imagen-texto.
- **DINOv2:** 142 millones de imágenes (sin etiquetas).
- **SAM:** 11 millones de imágenes y 1.000 millones de máscaras (segmentaciones).

Multimodalidad

La multimodalidad añade una dimensión importante: estos modelos procesan **imagen + texto simultáneamente**, aprendiendo correspondencias entre ambos dominios. Esto les permite entender instrucciones verbales sobre contenido visual y generar descripciones textuales de imágenes.



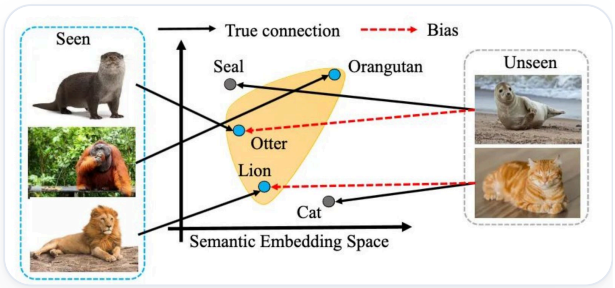
Ejemplos de modelos multimodales:

- **CLIP:** relaciona imágenes y texto en un espacio compartido.
- **BLIP:** genera descripciones y responde preguntas sobre imágenes.
- **Grounding DINO:** detecta objetos mediante descripciones textuales.
- **VLM (Qwen2.5-VL):** conversa y razona sobre contenido visual.

Zero-shot y composicionalidad

La capacidad **zero-shot** significa que el modelo puede resolver tareas nuevas sin entrenamiento específico para ellas. La clave está en la **composicionalidad**: el modelo aprende conceptos atómicos («perro», «rojo», «corriendo») y los combina libremente.

Ejemplo práctico: el modelo solo ha visto **nutria**, **orangután** y **león**. En una imagen nueva, **foca** y **gato** (nunca vistos) se proyectan en el **espacio semántico** cerca de los conceptos aprendidos (foca ≈ nutria, gato ≈ león). Así puede **reconocer clases nuevas sin reentrenar**, añadiendo categorías al instante y cubriendo con coste de anotación casi cero.



Impacto en producción

- Se ahorran miles de horas de anotación manual para nuevas variantes.
- Se puede experimentar en minutos en lugar de semanas.

Esta flexibilidad es revolucionaria para dominios industriales o médicos donde las categorías cambian constantemente.

Categorías nuevas sin reentrenar

En zero-shot, las clases llegan como texto y no como una cabeza reentrenada.

Quieres separar fotos de interior y exterior sin entrenar un clasificador nuevo.

- A Comparar la imagen con ambas descripciones de texto.
- B Cambiar la resolución de la foto.
- C Usar una caja fija.

Respuesta A. Exacto: CLIP ordena categorías propuestas como texto.

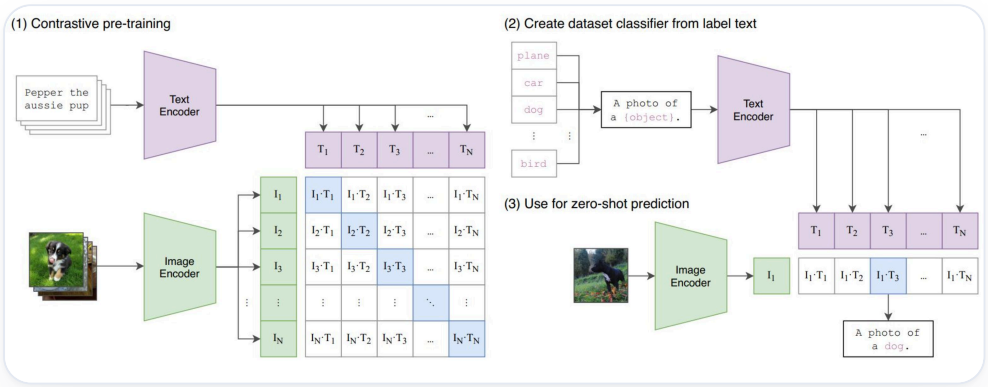
La composicionalidad permite combinar atributos como «rojo», «perro» y «corriendo». Esto acelera la experimentación, aunque las categorías propuestas y la formulación de los textos pueden cambiar el resultado.

CLIP: imagen - texto (Retrieval/Zero-Shot)

CLIP (Contrastive Language-Image Pre-training) aprende un **espacio de embeddings compartido** donde imágenes y textos similares están cerca y diferentes están lejos.

Arquitectura: un encoder de imagen y un encoder de texto procesan sus respectivas entradas en paralelo, proyectándolas al mismo espacio vectorial. La similitud se mide con producto escalar o coseno.

- **Clasificación zero-shot:** «¿Es esta imagen un gato o un perro?» → calculas similitud con ambos textos.
- **Búsqueda por texto:** «encuentra imágenes de tornillos oxidados» en un catálogo sin etiquetar.
- **Filtrado rápido:** CLIP es ligero y rápido, ideal como primera etapa antes de modelos más pesados.



Cómo funciona CLIP

CLIP relaciona imágenes y texto mediante un **entrenamiento contrastivo (contrastive learning)** y luego aplica ese conocimiento para tareas como la **clasificación zero-shot**.

01 **Imagen a vector**

La imagen se convierte en un vector (embedding) con el Image Encoder.

02 **Textos a vectores**

Los textos de las posibles clases se convierten en vectores con el Text Encoder.

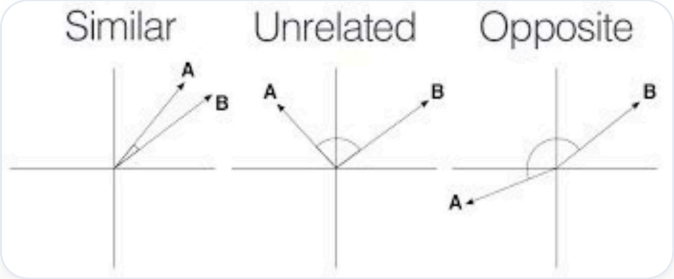
03 **Cálculo de similitud**

Se calcula la similitud entre el vector de la imagen y cada vector de texto.

04 **Predicción**

La clase con mayor similitud es la predicción final del modelo.

Ejemplo: para una foto de un gato y los textos «una astronauta…», «un gato…» y «un caballo…», las similitudes son [0.1, 0.9, 0.2] y la predicción es «un gato…».

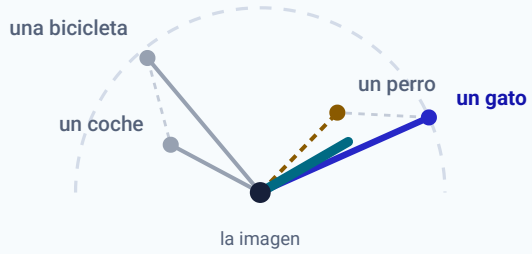


La similitud coseno mide el ángulo entre los dos vectores.

Cada texto obtiene su puntuación con cada imagen.

Gana el texto más cercano

La similitud coseno compara direcciones, no longitudes: mueve la imagen y mira qué texto se queda más cerca.



Los cuatro textos están a distancias distintas del origen y aun así solo cuenta el ángulo. Las líneas de puntos unen los dos que el modelo ha dejado cerca: gato con perro, bicicleta con coche.

un gato distancia 0,46

0,99

un perro distancia 0,17

0,96

una bicicleta distancia 1,18

−0,17

un coche distancia 0,96

−0,53

Gana «un gato» por ángulo (0,99), aunque «un perro» esté más cerca en línea recta: 0,17 frente a 0,46.

> Para qué sirve esto en la práctica: es el motor de la búsqueda por texto en un catálogo de imágenes, de la deduplicación («¿tengo ya esta foto?») y del primer filtro de moderación. En los tres casos se codifica una vez, se guardan los vectores y luego cada consulta es un producto escalar.

La similitud coseno solo mira el ángulo entre vectores: vale 1 cuando apuntan igual, 0 cuando son perpendiculares y −1 cuando se oponen. Por eso dos vectores de longitud muy distinta pueden ser casi idénticos para el modelo, como pasa aquí con la imagen y «un gato». La longitud de un embedding recoge sobre todo cuánta señal tiene el parche o el texto, no de qué habla, así que la práctica habitual es normalizar los vectores antes de guardarlos; con vectores normalizados, ordenar por coseno y ordenar por distancia euclídea dan el mismo resultado, y la distancia deja de estorbar. La agrupación tampoco es casual: los conceptos que aparecen juntos en los pares de entrenamiento acaban en direcciones próximas, y por eso gato y perro caen a un lado y bicicleta y coche al otro. De ahí se siguen dos consecuencias prácticas: la lista de candidatos condiciona la respuesta, porque el modelo elige siempre uno, y la formulación del texto desplaza su vector, así que reescribir el prompt puede cambiar el ganador sin tocar la imagen.

Entrenamiento de CLIP: aprendiendo por asociación

Entrenamiento contrastivo a gran escala (400M de pares imagen-texto).

01 **La entrada: arquitectura de dos torres**

CLIP utiliza dos codificadores independientes que trabajan en paralelo, proyectando la información a un espacio compartido. El **Image Encoder** (Vision Transformer o ResNet) convierte la imagen en un vector numérico (embedding). El **Text Encoder** (un Transformer) convierte la descripción textual en otro vector numérico.

02 **El proceso: la matriz de N × N**

El modelo recibe un lote (batch) de N imágenes y sus N textos correspondientes, creando una matriz de N × N posibles parejas. En un lote de 32.000 pares, se generan 32.000 parejas correctas y millones de combinaciones incorrectas.

03 **El objetivo: maximizar la diagonal**

El modelo se entrena con una única regla simple (**Contrastive Loss**):

- Aumentar la similitud (producto escalar) de los pares correctos (en la diagonal).
- Disminuir la similitud de todas las combinaciones incorrectas (fuera de la diagonal).

Limitaciones de CLIP

Qué NO puede hacer

- Generar texto (solo embeddings).
- Detectar objetos (no proporciona bounding boxes).
- Segmentar imágenes (no produce máscaras).
- Responder preguntas sobre contenido visual.
- Contar objetos con precisión.

Problemas conocidos

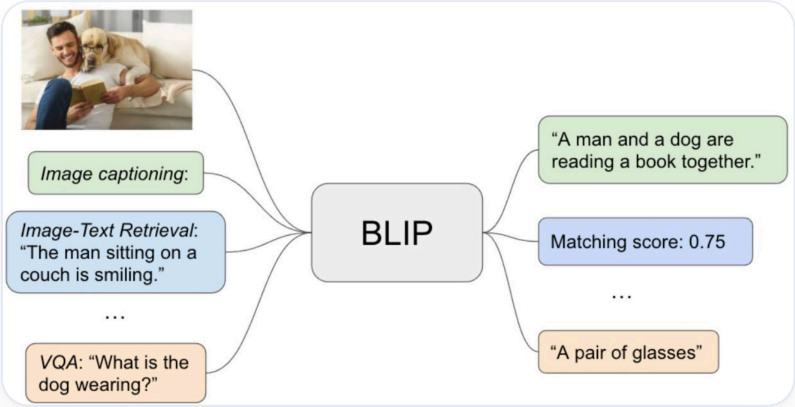
- Mal rendimiento con texto dentro de imágenes (OCR).
- Sesgos heredados del dataset de entrenamiento.
- Confunde atributos visuales muy finos.
- Poca comprensión de negaciones en el prompt.
- Alta sensibilidad a la formulación del prompt.

BLIP: describe y responde (generativo)

BLIP (Bootstrapping Language-Image Pre-training) va más allá de CLIP: no solo relaciona imágenes con texto, sino que **genera descripciones textuales** (captioning) y **responde preguntas** sobre el contenido visual (VQA).

Arquitectura: combina un encoder de imagen con un decoder de lenguaje basado en Transformers. La fusión multimodal ocurre mediante capas de cross-attention que permiten al texto «mirar» las features visuales.

- **Documentación automática:** generar descripciones de productos, escenas industriales o imágenes médicas.
- **Accesibilidad:** proporcionar descripciones textuales para personas con discapacidad visual.
- **VQA simple:** «¿Qué color tiene el casco?» → «Amarillo»; «¿Cuántas personas aparecen?» → «Dos».



Grounding DINO: texto → cajas (Open-Vocabulary)

Grounding DINO combina la potencia de DINO (detección) con la comprensión lingüística para realizar **detección guiada por lenguaje natural** sin clases fijas.

Cómo funciona: le das un prompt textual («encuentra todos los tornillos oxidados» o «localiza cascos amarillos») y el modelo devuelve bounding boxes para cada coincidencia, sin haber sido entrenado explícitamente en esas categorías.

- Elimina la necesidad de reentrenar para cada nueva categoría de objeto.
- Permite detección ad-hoc: cambias el texto y cambias lo que detectas.
- Auto-labeling.

Capacidades avanzadas

- **Detección compositiva:** «coches rojos y personas con cascos» → entiende atributos y combinaciones.
- **Referring expressions:** «la persona alta de la izquierda» → localización específica mediante descripciones.

Cómo funciona Grounding DINO

Grounding DINO integra procesamiento de lenguaje e imagen en un pipeline unificado, permitiendo una detección de objetos altamente flexible y adaptada a descripciones textuales.

01

Codificación de texto

El prompt textual («coche rojo y persona») se transforma en **features lingüísticas** ricas en significado a través de un modelo BERT.

02

Codificación de imagen

La imagen de entrada es procesada por un Swin Transformer para extraer **features visuales** de alta resolución.

03

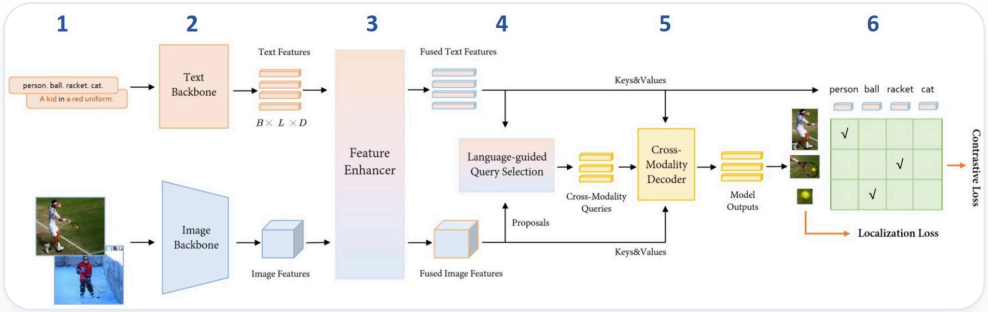
Fusión de características

Mediante **atención cruzada intermodal**, las features de texto guían la interpretación visual, alineando el lenguaje con las regiones espaciales relevantes de la imagen.

04

Detección head

Un decoder Transformer usa estas features fusionadas para predecir las **bounding boxes** y las **clases** de los objetos, todo en base al prompt original.



Innovaciones de Grounding DINO

Detección sin anchors

A diferencia de métodos clásicos, no requiere cajas predefinidas, lo que simplifica y flexibiliza el proceso.

Queries guiadas por lenguaje

El texto dirige directamente qué y dónde buscar en la imagen.

Vocabulario abierto

COCO: 80 clases fijas → Grounding DINO: infinitas clases, cualquier texto describible.

Sin NMS necesario

Evita el post-procesamiento de Non-Maximum Suppression, optimizando la predicción de cajas.

Atención cruzada

Permite una profunda interacción entre las representaciones visuales y lingüísticas.

Casos de uso y limitaciones de Grounding DINO

Casos de uso

- **Inspección industrial sencilla:** «defectos en soldadura», «tornillos faltantes», «grietas en superficie».
- **Retail y e-commerce:** «productos dañados», «etiquetas de precio», «artículos mal colocados».
- **Seguridad:** «personas sin casco», «vehículos en zona restringida», «objetos abandonados».
- **Agricultura:** «frutos maduros», «plagas en hojas», «equipos agrícolas».

Limitaciones

- **Ambigüedad lingüística:** prompts vagos («cosa roja») pueden generar detecciones imprecisas.
- **Objetos muy pequeños:** rendimiento reducido en objetos menores al 1 % del área de imagen.
- **Escenas complejas:** dificultades con múltiples objetos superpuestos o muy similares.
- **Dependencia del lenguaje:** requiere descripciones precisas; no funciona bien con jerga técnica muy específica.

Nombrar para localizar

Los modelos multimodales se distinguen por su salida: texto, puntuación o cajas.

Quieres localizar un defecto industrial nombrándolo en texto.

- A Usar Grounding DINO y evaluar con imágenes etiquetadas.
- B Suponer que el prompt elimina falsos positivos.
- C Medir solo el tamaño del checkpoint.

Respuesta A. Exacto: el prompt guía la búsqueda, pero la evaluación valida el uso.

CLIP puntúa textos frente a una imagen, BLIP genera lenguaje y Grounding DINO devuelve cajas. Para localizar un defecto nombrándolo, la salida que necesitas es una caja.

Anatomía de un VLM: ¿cómo ve el LLM?

Vision encoder (ojos)

Extrae **features visuales** de la imagen (por ejemplo ViT o SigLIP), transformando píxeles en una representación numérica de alto nivel.

Projector / Adapter (traductor)

Una capa (MLP o Linear) que convierte las **features visuales** en «tokens visuales», adaptándolos para que tengan el mismo tamaño y formato que los tokens de texto.

LLM Backbone (cerebro)

Recibe [tokens de sistema] + [tokens visuales] + [tokens de usuario] como una única secuencia. La imagen se convierte en una secuencia de tokens pseudo-palabras que se concatenan al prompt de texto.

VLM (Qwen2.5/3-VL): conversa y razona sobre imágenes

Los **Vision-Language Models** (VLM) como Qwen2.5-VL llevan la multimodalidad al extremo: además de describir o detectar, también **razonan, comparan, extraen datos estructurados** y mantienen conversaciones sobre contenido visual.

Capacidades avanzadas:

- **OCR con comprensión:** lee texto y entiende su significado y estructura (facturas, formularios, diagramas).
- **Análisis de gráficos:** interpreta charts, tablas y visualizaciones, extrayendo insights.
- **Instrucciones paso a paso:** «explica cómo funciona este mecanismo» o «identifica errores en este montaje».
- **Salida estructurada:** puede devolver JSON con campos específicos que solicites.
- **Resolución dinámica (Naive Dynamic Resolution):** a diferencia de modelos anteriores, procesa imágenes con cualquier relación de aspecto y resolución nativa, clave para leer documentos y textos pequeños sin deformarlos.

> **Trade-off importante:** los VLM son mucho más pesados y lentos que CLIP o BLIP. Úsalos solo donde aporten valor real, no como solución universal. Para tareas simples, CLIP o Grounding DINO son suficientes.

Mejores prácticas de prompting

Para maximizar la eficacia de los modelos multimodales, hay que formular los prompts de manera estratégica y específica para cada tipo de modelo.

Para CLIP

- Usar templates como «una foto de {}».
- Múltiples formulaciones del mismo concepto.
- Evitar negaciones directas en el prompt.

Para Grounding DINO

- Separar conceptos con « . » (punto).
- Ser descriptivo pero conciso.
- Incluir atributos visuales clave.

Para Qwen y otros VLM

- Instrucciones claras y específicas.
- Especificar formato de salida (por ejemplo, JSON).
- Dar ejemplos concretos si es posible.

Malos prompts

- «¿Está bien?»
- «no hay personas sin casco»

Buenos prompts

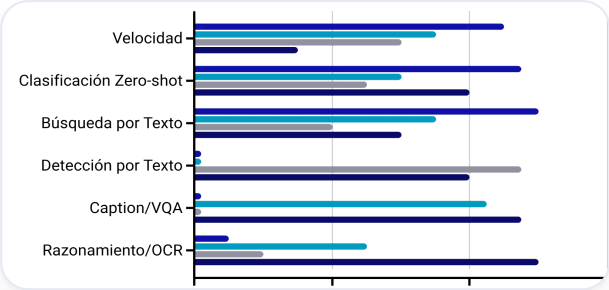
- «Lista todos los defectos visibles en esta pieza metálica»
- «personas con casco . personas sin casco» (para comparar)

¿Qué modelo uso para cada tarea?

La elección del modelo correcto depende de la **complejidad de la tarea** y las **restricciones de latencia**.

Regla de decisión rápida:

- ¿**Velocidad y filtrar**? → CLIP
- ¿**Descripciones o Q&A simple**? → BLIP
- ¿**Localizar objetos por texto**? → Grounding DINO
- ¿**Razonamiento complejo o datos estructurados**? → VLM



El gráfico ordena capacidades relativas, no mide latencia en un hardware concreto. Para decidir hay que medir en el dispositivo de destino con el tamaño de entrada real.

Aplica la regla de decisión

Cada familia resuelve bien una cosa y mal las demás.

Necesitas una frase que describa una imagen.

- A BLIP.
- B CLIP.
- C Grounding DINO.

Respuesta A. Exacto: BLIP genera texto condicionado por una imagen.

Generar una frase es tarea de un modelo generativo. CLIP solo puntúa textos que tú escribes y Grounding DINO devuelve cajas, no lenguaje.

Limitaciones y buenas prácticas

Los modelos multimodales son muy potentes pero tienen **limitaciones importantes** que debes tener en cuenta.

Alucinaciones: los modelos generativos pueden inventar detalles que no existen en la imagen. Especialmente común en VLM cuando preguntas sobre aspectos ambiguos.

Sesgos: los datos de entrenamiento contienen sesgos sociales y culturales que se reflejan en las predicciones. Valida siempre con datos representativos de tu población objetivo.

Prompts ambiguos: «encuentra el problema» puede dar resultados inconsistentes.

Estrategias de mitigación

- **Ajusta thresholds:** empieza con confianza 0.5, sube si hay muchos FP, baja si hay muchos FN.
- **Valida con tus datos:** los benchmarks públicos no garantizan rendimiento en tu dominio.
- **Human-in-the-loop:** en casos críticos (medicina, seguridad), siempre revisa las predicciones.
- **Prompting específico:** evita negaciones («sin defectos»), prefiere afirmaciones («tornillos en buen estado»). Pide formato de salida explícito (JSON con campos concretos).

Clasificar sin entrenar nada

Escribes las clases como texto y CLIP te ordena cuál encaja mejor.

```
import torch
from PIL import Image
from transformers import CLIPProcessor, CLIPModel

clip_id = "openai/clip-vit-base-patch32"
processor = CLIPProcessor.from_pretrained(clip_id)
model = CLIPModel.from_pretrained(clip_id)

image = Image.open("mi_imagen.jpg").convert("RGB")
clases = ["una foto de una pieza en buen estado",
          "una foto de una pieza con óxido",
          "una foto de una pieza rota"]

inputs = processor(text=clases, images=image, return_tensors="pt", padding=True)
with torch.no_grad():
    probs = model(**inputs).logits_per_image.softmax(dim=1)[0]

for clase, p in zip(clases, probs):
    print(f"{p:.2f}  {clase}")
```

- La plantilla **«una foto de …»** suele funcionar mejor que la palabra suelta.
- El modelo **siempre elige una** de tus clases. Si ninguna encaja, te dará la menos mala: añade una clase «ninguna de las anteriores».
- Evita las negaciones: «pieza en buen estado» funciona mucho mejor que «pieza sin óxido».

Taller de prompts

Dos prompts que fallan en clase y cómo se arreglan.

Quieres que Grounding DINO encuentre a la vez cascos puestos y cabezas sin casco. ¿Qué prompt escribes?

- A «no hay personas sin casco»
- B «seguridad en la obra»
- C «persona con casco . persona sin casco»

Respuesta C. Correcto. Dos conceptos en positivo, separados por el punto que Grounding DINO usa para distinguirlos, y cada uno nombra algo visible.

Le pides a un VLM que revise fotos de piezas y te devuelva los defectos. ¿Qué instrucción funciona mejor?

- A «¿Está bien esta pieza?»
- B «Lista los defectos visibles en esta pieza metálica. Devuelve JSON con los campos tipo, ubicación y confianza. Si no ves ninguno, devuelve una lista vacía.»
- C «Describe la imagen con el máximo detalle posible»

Respuesta B. Correcto. Pide en positivo, fija el formato de salida y contempla explícitamente el caso sin defectos, que es donde más alucinan los generativos.

Las tres reglas que resuelven la mayoría de los casos: pedir en positivo en lugar de en negativo, nombrar lo que se busca en vez de preguntar si algo está bien, y separar conceptos con un punto cuando el modelo es Grounding DINO. Para un VLM, además, conviene decir el formato de salida que esperas.

¿Cuál de los cuatro te sirve?

Responde a dos preguntas y sale el modelo, con la línea para empezar.

¿Qué necesitas que devuelva?

- (●) Una puntuación por cada opción que yo escriba
- ( ) Una frase escrita sobre la imagen
- ( ) Cajas alrededor de lo que nombro
- ( ) Datos estructurados o un razonamiento

¿Cuánta latencia aguantas?

- (●) Poca: proceso muchas imágenes
- ( ) Bastante: son pocas y me importa el resultado

CLIP

Ordena tus textos candidatos frente a la imagen. Es el más rápido y ligero, ideal como primer filtro de un catálogo.

```
CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
```

BLIP

Genera descripciones y responde preguntas sencillas sobre la imagen, con un coste muy inferior al de un VLM.

```
BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
```

BLIP, y un VLM si te quedas corto

Empieza por BLIP. Si necesitas que la frase razone o siga instrucciones, salta a Qwen-VL asumiendo el coste.

`Salesforce/blip-image-captioning-base` y, si hace falta, `Qwen/Qwen2.5-VL-7B-Instruct`

Grounding DINO

Le das una frase y devuelve cajas, sin haber sido entrenado en esa categoría. Separa conceptos con un punto.

```
AutoModelForZeroShotObjectDetection.from_pretrained("IDEA-Research/grounding-dino-base")
```

Un VLM como Qwen2.5-VL

Razona, compara y devuelve JSON con los campos que le pidas. Es el más pesado, así que resérvalo para donde aporte.

`Qwen/Qwen2.5-VL-7B-Instruct`

Replantea el problema

Razonar y extraer estructura es caro. Si necesitas volumen, resuelve primero con CLIP o Grounding DINO y manda al VLM solo los casos dudosos.

Ese filtrado en dos etapas suele bajar el coste un orden de magnitud sin perder los casos que importan.

Antes de fiarte de una salida multimodal

Siete comprobaciones para modelos que hablan con mucha seguridad.

- [ ] **La lista de candidatos es completa**
  Con CLIP el modelo elige siempre uno. Si falta la opción «ninguna», te devolverá la menos mala.
- [ ] **Prompts en positivo**
  «Tornillos en buen estado» en vez de «sin óxido». Las negaciones se entienden mal.
- [ ] **Formato de salida pedido explícitamente**
  Si esperas JSON, dilo y da un ejemplo de los campos.
- [ ] **Alucinaciones buscadas a propósito**
  Prueba con imágenes donde la respuesta correcta sea «no se ve». Un generativo tiende a inventar.
- [ ] **Sesgos revisados con tus datos**
  Valida con población y escenarios representativos de tu uso real.
- [ ] **Umbral ajustado, no heredado**
  Empieza en 0,5 y muévelo según veas falsos positivos o negativos.
- [ ] **Humano en el bucle donde importa**
  Medicina, seguridad y decisiones sobre personas siempre con revisión.

Cuatro formas de mirar

Compara CLIP, BLIP, Grounding DINO y Qwen con la misma imagen.

1. Guarda una copia en Drive.
2. Cambia imagen y prompts.
3. Contrasta salidas y errores.

Abrir en Colab ↗

Usa la misma imagen para separar capacidades: CLIP ordena textos, BLIP genera una descripción, Grounding DINO localiza conceptos y Qwen responde instrucciones. Registra el prompt, el modelo y el error observado; una salida más larga o fluida no implica una respuesta más correcta.