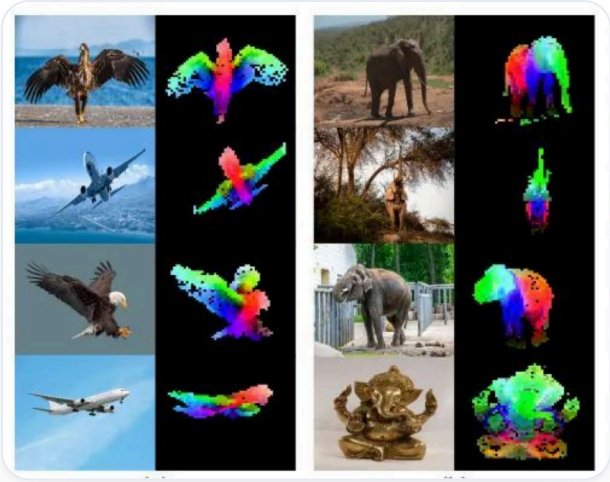
4. DINO (v1, v2, v3): ver sin etiquetas

Representaciones visuales generales para muchas tareas.

DINO (Self-Distillation with NO labels) y sus evoluciones representan un paradigma radicalmente diferente: aprender representaciones visuales potentes directamente de los píxeles, sin necesidad de etiquetas humanas.



¿Por qué auto-supervisado?

El aprendizaje auto-supervisado ofrece una vía para construir modelos de visión robustos y escalables, superando las barreras del etiquetado manual.

Aprendizaje supervisado

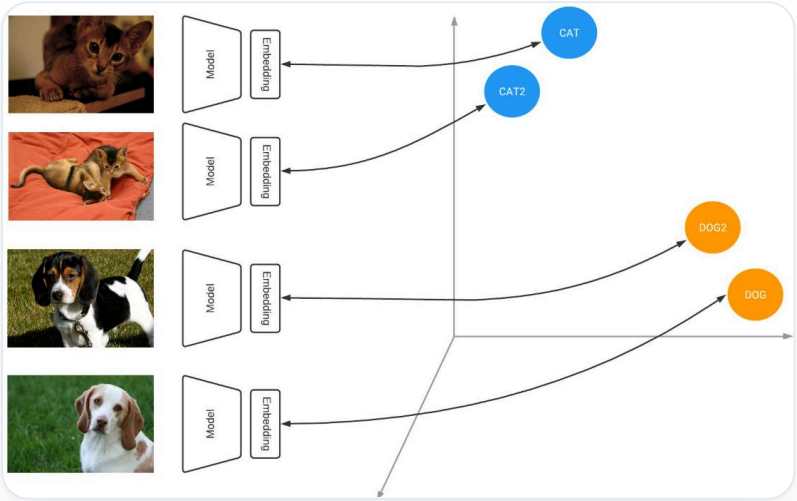
- Necesita etiquetas manuales de expertos.
- Costoso y lento de producir.
- Limitado a categorías predefinidas.
- No escala bien a grandes volúmenes de datos.

Ejemplo: ImageNet

- 14 millones de imágenes.
- 1000 categorías.
- Años de trabajo manual.
- Millones de dólares de inversión.

Aprendizaje auto-supervisado

- No requiere etiquetas manuales.
- Aprende de la estructura interna de los datos.
- Escala a billones de imágenes de manera eficiente.
- Genera representaciones más generales y transferibles.



Self-supervised learning explicado

El modelo genera sus propias señales de supervisión desde los datos.

Contrastive learning

- Imágenes similares → embeddings cercanos.
- Imágenes diferentes → embeddings lejanos.

Masked prediction

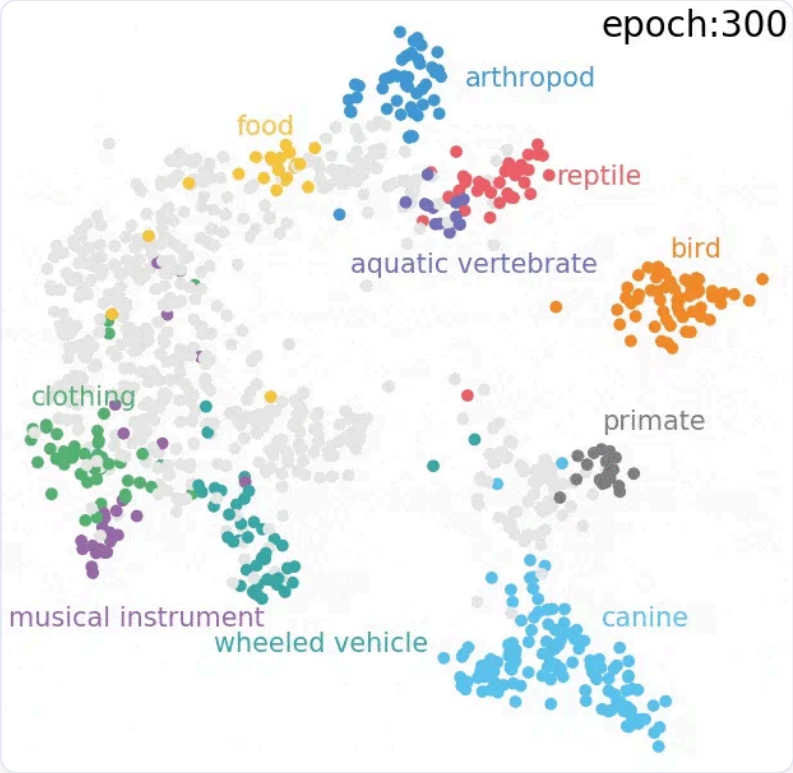
- Ocultar partes de imagen.
- Predecir partes ocultas.

Rotation prediction

- Rotar imagen.
- Predecir ángulo de rotación.

Clustering

- Agrupar features similares sin saber qué representan.



Auto-destilación

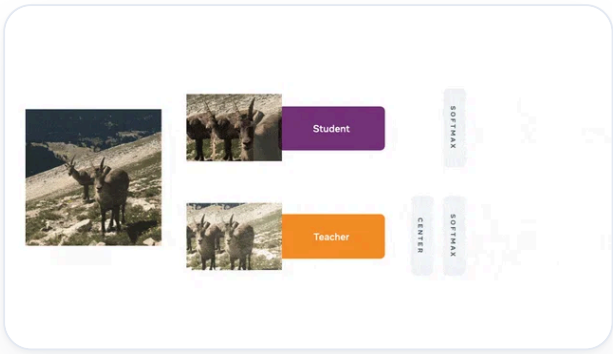
DINO entrena una red enseñándole a predecir la salida de su versión más estable, la red maestra (teacher), sobre diferentes vistas de la misma imagen. Este proceso fomenta la consistencia y la robustez de las representaciones.

Red estudiante (student network)

- Aprende activamente y ajusta sus pesos mediante backpropagation.
- Recibe diferentes aumentaciones (vistas locales o detalladas) de la imagen.

Red maestra (teacher network)

- Posee la misma arquitectura que la estudiante, pero con pesos estables.
- Sus pesos se actualizan lentamente a través de un promedio móvil (momentum update) de los pesos de la red estudiante.
- Recibe una vista global o contextual de la imagen.



El objetivo es que la red estudiante sea capaz de replicar la predicción de la red maestra, forzándola a aprender representaciones visuales de alta calidad sin etiquetas explícitas. Este mecanismo simula una auto-supervisión continua.

El papel de la red maestra

Las dos redes comparten arquitectura, pero no aprenden igual.

¿Qué función cumple la red profesora en DINO?

- A Ofrecer objetivos para que aprenda la estudiante.
- B Etiquetar manualmente cada imagen.
- C Generar texto sobre la imagen.

Respuesta A. Exacto: la profesora proporciona una referencia durante el entrenamiento.

La maestra no se entrena por backpropagation: sus pesos son un promedio móvil de los de la estudiante. Esa estabilidad es lo que da un objetivo consistente y evita el colapso de las representaciones.

Lo que DINO aprende solo

Al entrenar Vision Transformers (ViT) con el algoritmo DINO, el modelo aprende automáticamente a separar el objeto principal del fondo, logrando una representación visual interpretable y universal.

El descubrimiento: segmentación auto-aprendida. DINO permite que el modelo aprenda a segmentar objetos de manera inherente, sin necesidad de anotaciones humanas explícitas o funciones de pérdida de segmentación a nivel de píxel. Es un aprendizaje «viendo» el mundo de forma consistente.

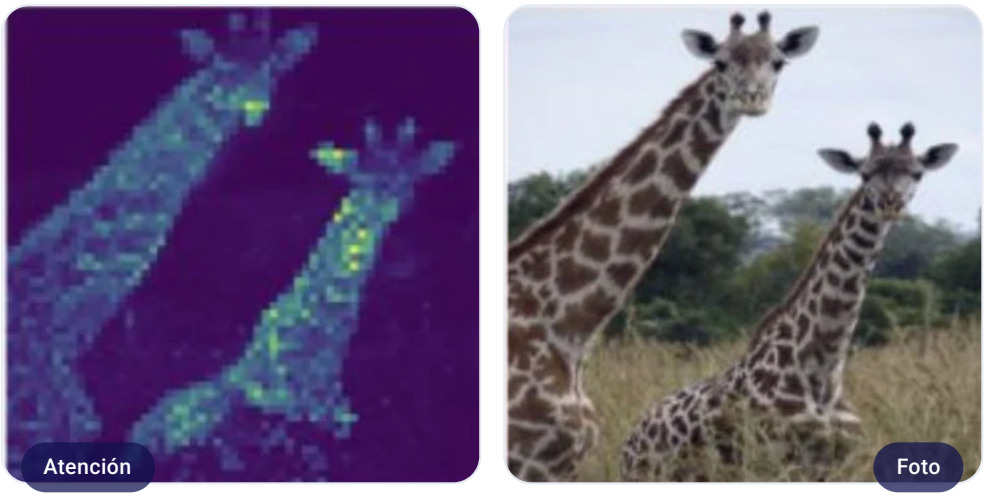
El poder de la auto-atención (self-attention):

- **Visión global:** cada parte de la imagen construye su comprensión procesando y relacionándose con otras ubicaciones, incluso las más distantes.
- **Comprensión de escena:** de esta manera, el modelo desarrolla una interpretación rica y de alto nivel de la escena completa.
- **Segmentación por atención:** al analizar los mapas de atención internos del modelo, se observa que estos se alinean directamente con regiones semánticas coherentes, es decir, el modelo «separa» los objetos automáticamente.

Los mapas de atención son una señal interna del modelo y se alinean a menudo con objetos, pero no son una máscara de segmentación garantizada ni validada píxel a píxel.

La atención, encima de la foto

Desliza para pasar de la imagen al mapa de atención del modelo. Nadie le dijo dónde estaba el objeto: estas máscaras salen de un entrenamiento sin una sola etiqueta.

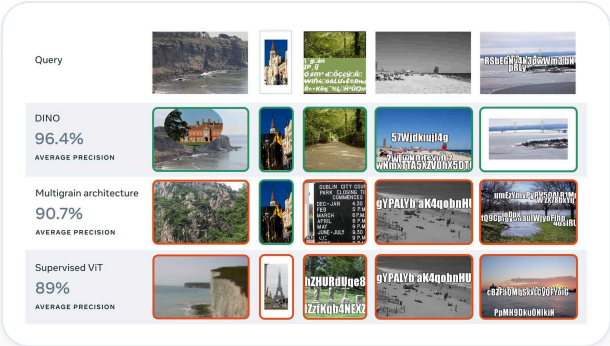


Dos animales del mismo tipo y el mapa los separa a los dos, con más intensidad en cuellos y cabezas.

Los recortes salen de la figura original del artículo de DINO, que muestra la atención del token de clase en la última capa de un ViT entrenado sin etiquetas. El mapa tiene la resolución de los parches, de ahí el aspecto cuadriculado: con parches de 8 píxeles se ve más fino que con parches de 16. Conviene recordar que es una señal interna del modelo y no una máscara validada píxel a píxel, así que sirve para entender qué mira y no como salida de segmentación.

DINO: identificación de copias de imágenes

Un hallazgo sorprendente es que DINO es excepcionalmente bueno para identificar copias de imágenes, incluso sin haber sido diseñado para ello. Este modelo podría convertirse en un estándar para sistemas de detección de copias, combatiendo la desinformación y protegiendo los derechos de autor.



Evolución de la familia DINO

DINO, acrónimo de «Self-DIstillation with NO labels», ha marcado un antes y un después en el aprendizaje auto-supervisado en visión artificial.

01 **2021: DINO v1**

Primer ViT auto-supervisado. Demostró que la atención del modelo podía realizar segmentación de objetos sin supervisión explícita.

02 **2022: DINO v2**

Escalado a 142M imágenes, con mejoras arquitectónicas. Generó features visuales mucho más robustas y generalizables.

03 **2024: DINO v3**

Entrenado con 1B+ imágenes, logrando una calidad de representación cercana a la supervisada. Base para nuevos modelos fundacionales.

> DINO es la base de modelos como SAM y Grounding DINO, proveyendo features universales para múltiples tareas de visión.

DINOv2 se publicó en 2023 y DINOv3 en 2025. Las fechas de la diapositiva señalan el momento en que cada generación se incorporó al curso.

Pipeline de procesamiento de datos DINOv2

DINOv2 se entrena con un vasto y diversificado conjunto de datos para lograr una representación visual robusta. El pipeline de procesamiento es la clave.

01 **Ingesta de datos**

Imágenes curadas y no curadas (de la web) se combinan para diversidad.

02 **Generación de embeddings**

Todas las imágenes se transforman en embeddings de alta dimensión.

03 **Eliminar duplicados**

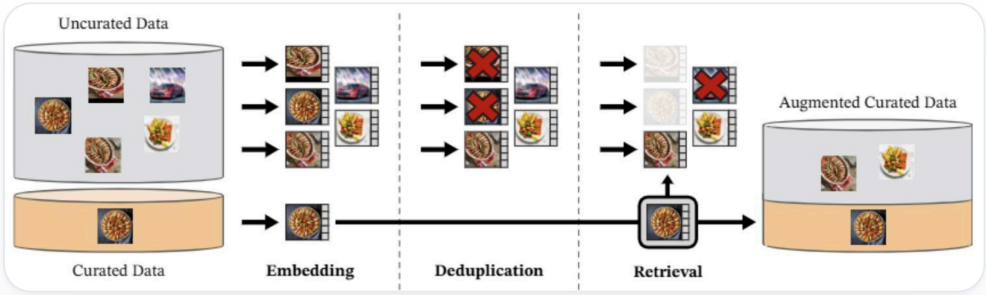
Se eliminan duplicados o imágenes muy similares de los datos no curados usando embeddings.

04 **Matching y enriquecimiento**

Imágenes no curadas visualmente similares a las curadas se añaden para enriquecer el dataset.

05 **Retrieval auto-supervisado**

El sistema expande el dataset añadiendo imágenes relevantes sin etiquetado manual.



Cómo funciona DINO

DINO opera mediante un innovador mecanismo de auto-destilación donde una red «estudiante» aprende de una red «maestra» en un ciclo iterativo y sin necesidad de etiquetas. El proceso se desarrolla en los siguientes pasos clave.

01 **Imagen y aumentaciones diferentes**

Una imagen de entrada se somete a diversas transformaciones (aumentaciones) para crear múltiples vistas, incluyendo crops locales y globales.

02 **Red estudiante: vistas locales**

La red estudiante recibe y procesa específicamente los crops locales, o las versiones más pequeñas y detalladas, de la imagen aumentada.

03 **Red maestra: vistas globales**

Simultáneamente, la red maestra procesa las vistas globales y más amplias de la misma imagen, capturando el contexto general.

04 **Estudiante predice salida de la red maestra**

El objetivo de la red estudiante es aprender a predecir la representación (salida) de la red maestra a partir de sus propias vistas locales.

05 **Maestra se actualiza por promedio**

Los pesos de la red maestra se actualizan de forma suave, utilizando un promedio móvil exponencial (momentum update) de los pesos de la red estudiante, asegurando estabilidad y progreso constante.

Lo que DINO aprende solo

Descubrimientos importantes

DINO es una forma eficiente de aprender representaciones y también revela una serie de capacidades emergentes que no fueron explícitamente entrenadas, demostrando cómo el modelo adquiere una comprensión profunda de la escena visual.

Segmentación semántica

Los mapas de atención de DINO actúan como máscaras de objetos, distinguiendo el primer plano del fondo. Identifica objetos coherentes en la escena sin ninguna anotación durante su entrenamiento.

Correspondencias

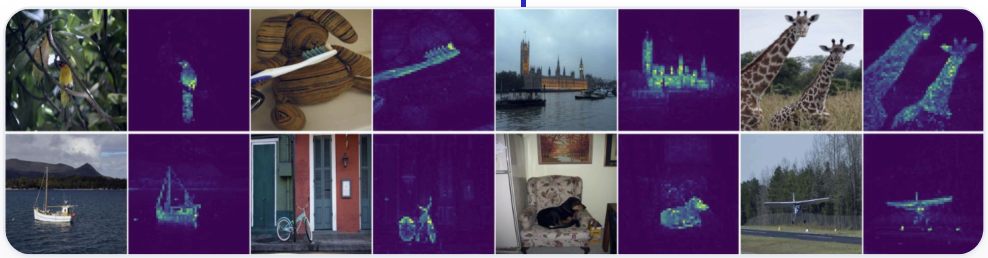
Es capaz de encontrar puntos equivalentes y coherentes entre diferentes vistas de una imagen o incluso entre imágenes distintas. Esta capacidad surge sin haber sido supervisado nunca para tareas de correspondencia.

Detección de objetos

Los heads de atención de DINO se focalizan naturalmente en objetos salientes dentro de la imagen. Esto ocurre sin que el modelo haya visto una sola bounding box durante su fase de preentrenamiento.

Clasificación

Utilizando las features extraídas por DINO, un clasificador k-NN simple puede lograr buenos resultados. DINO no utiliza etiquetas de clases para el entrenamiento, pero sus representaciones son altamente discriminativas.



DINO como extractor de características

DINO es una herramienta muy potente para la extracción de características visuales de alta calidad, esenciales para diversas aplicaciones de visión artificial, aprendiendo sin etiquetas.

Uso principal

Backbone para extraer características visuales universales de cualquier imagen, sin anotaciones.

Ventajas

- Generales y transferibles.
- Alto rendimiento en dominios nuevos.
- No sesgado a categorías específicas.

Calidad

- Densas (por patch).
- Semánticamente significativas.
- Altamente discriminativas.

Aplicaciones

- Clasificación (k-NN).
- Detección de anomalías.
- Clustering visual.

Cercanía según el modelo

La similitud coseno compara la dirección de dos embeddings normalizados.

Dos imágenes tienen embeddings cercanos. ¿Qué puedes concluir?

- A El modelo las representa como similares.
- B Son idénticas píxel a píxel.
- C Pertenecen a una clase legalmente autorizada.

Respuesta A. Exacto: la cercanía expresa similitud según esa representación.

Una similitud alta significa que el modelo representa las imágenes de forma próxima. No demuestra identidad píxel a píxel ni una categoría concreta.

Entendiendo qué aprende DINO

Para comprender cómo DINO procesa y entiende el mundo visual, utilizamos diversas técnicas de visualización que revelan la riqueza de sus representaciones auto-supervisadas.

Mapas de atención

Visualizan las ponderaciones de auto-atención del modelo, mostrando explícitamente dónde se focaliza la red. Revelan segmentación implícita de objetos y la relación entre elementos.

PCA de features

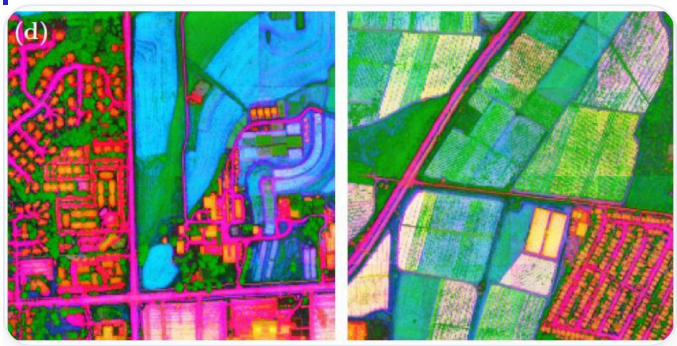
Aplica Análisis de Componentes Principales a las características densas extraídas por DINO. Se mapean los primeros 3 componentes a canales RGB para visualizar agrupaciones semánticas.

t-SNE / UMAP

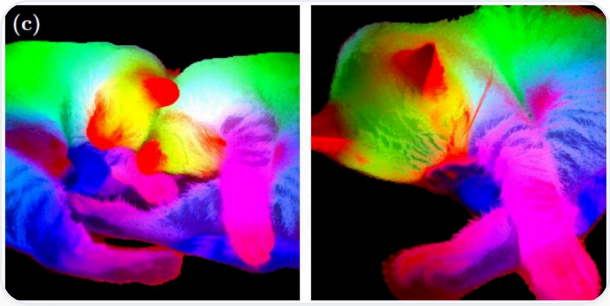
Estas técnicas de reducción de dimensionalidad no lineal proyectan los embeddings de alta dimensión en un espacio 2D. Muestran cómo DINO agrupa conceptos visuales similares sin etiquetas.

Vecinos cercanos

Identifica parches de imagen o imágenes completas con las representaciones DINO más similares. Útil para validar la coherencia de las características y encontrar correspondencias visuales.



PCA de features densas mapeada a canales RGB.



Vecinos cercanos entre parches de imágenes distintas.

PCA es una proyección lineal útil para colorear características densas. t-SNE y UMAP son técnicas no lineales que proyectan embeddings a 2D para explorar vecindarios. Ninguna crea etiquetas ni demuestra por sí sola que un cluster sea una clase real.

De 768 números a un color

Las imágenes de colores chillones que ilustran DINO no son un filtro ni un mapa de calor. Son las características del modelo pintadas, y el procedimiento cabe en cuatro pasos.

01 **Un vector por parche**

La imagen se parte en parches de 16 × 16 píxeles y el modelo devuelve un vector por parche, de 768 números en un ViT base. Una imagen de 224 × 224 da 196 vectores.

02 **PCA sobre todos los parches**

Se juntan esos vectores y se buscan las tres direcciones en las que más varían. Es álgebra pura, sin entrenamiento ni etiquetas.

03 **Tres números por parche**

Cada parche se proyecta sobre esas tres direcciones. Los 768 números se quedan en tres, que es justo lo que cabe en un color.

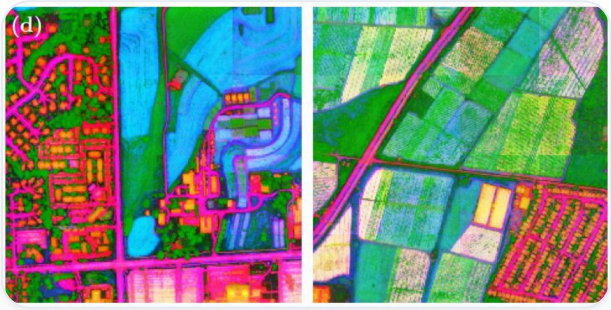
04 **Los tres números se pintan**

Se reescalan a 0-255 y se usan como rojo, verde y azul. El parche vuelve a su sitio en la rejilla y aparece la imagen de colores.

Por qué el ala de un avión sale del mismo color que el ala de un pájaro. El color no lo decide el píxel, lo decide el vector. Dos parches con vectores parecidos reciben proyecciones parecidas y, por tanto, el mismo color, aunque uno sea metal gris y el otro plumas marrones.

Que el modelo los haya dejado cerca significa que ha aprendido algo sobre ellos sin que nadie se lo dijera: los dos son superficies alargadas que salen de un cuerpo central y sostienen la figura. Esa es la afirmación fuerte de DINO, y el color es solo la forma de verla.

Con el mismo razonamiento, la rueda de un coche y la rueda de una bicicleta caen juntas, y el cielo cae con el mar. Lo que agrupa es el papel que juega la región, no su textura.



Aquí las carreteras salen magenta en las dos imágenes, y la vegetación verde. Ningún color estaba definido de antemano.

> Para qué sirve más allá de la ilustración: es la forma rápida de comprobar si un modelo entiende tus imágenes antes de montar nada encima. Si al pintar el PCA de tus piezas la zona buena y la zona defectuosa salen de colores distintos, ese backbone te vale; si sale todo del mismo color, cambia de modelo o de resolución antes de perder una semana.

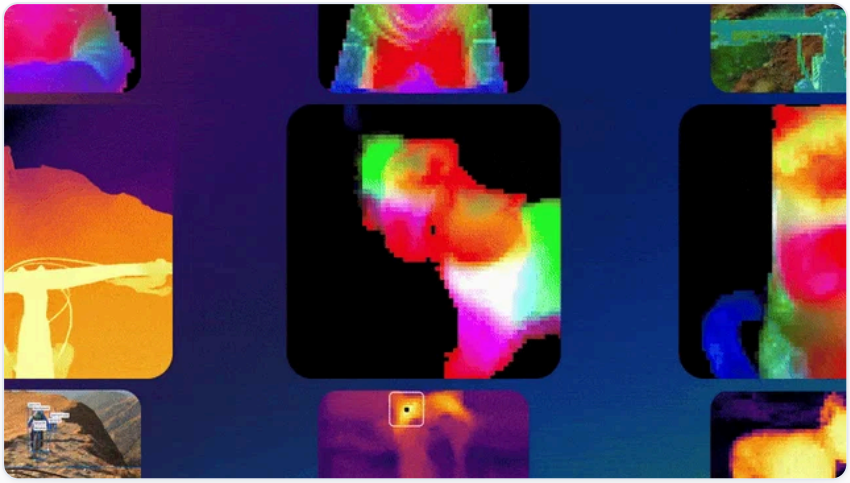
Dos avisos prácticos. El primero: los colores no son comparables entre imágenes salvo que el PCA se ajuste con todas a la vez, porque cada ajuste elige sus propias direcciones y el signo de cada componente es arbitrario. Por eso las figuras publicadas suelen calcular el PCA sobre el conjunto entero de imágenes que enseñan. El segundo: la primera componente suele separar objeto y fondo

más que partes concretas, así que es habitual usarla como máscara, quedarse solo con los parches del objeto y repetir el PCA sobre ellos para que las tres componentes se gasten en distinguir partes. El procedimiento no depende de DINO: se puede aplicar a las características densas de cualquier modelo, y lo que cambia de uno a otro es cuánta estructura aparece.

DINO v3

Fíjate en que el color de cada región se mantiene mientras la escena se mueve: el modelo no ha visto una sola etiqueta.

DINO v3: características densas



Cada parche conserva su representación entre fotogramas, que es lo que permite usarlas para correspondencias y seguimiento.

Cuándo usar (y cuándo no) DINO

DINO no es una solución universal. Úsalo cuando tenga sentido estratégico.

Úsalo si…

- No tienes etiquetas o son muy costosas de obtener.
- Las clases o categorías cambian frecuentemente y reentrenar es inviable.
- Necesitas features reutilizables para múltiples tareas (segmentación, anomalías, búsqueda…).
- Quieres descubrir patrones sin asumir taxonomía previa.
- Buscas robustez a cambios de dominio (iluminación, fondo, escala).

Evítalo si…

- Tienes muchas etiquetas fiables y un objetivo de clasificación bien definido (supervisado será mejor).
- Requieres tiempo real en edge con hardware muy limitado (DINO es más pesado que MobileNet).
- Necesitas explicabilidad detallada y reglas interpretables (modelos más simples o clásicos).
- El problema es extremadamente específico y las features generales no aportan (raro, pero posible).

Regla empírica: si tienes etiquetas abundantes y un objetivo fijo, entrena supervisado. Si no, empieza con DINO features y ahorra semanas de trabajo.

Anomalías con DINOv2/v3 (estilo PatchCore)

La detección de anomalías es de vital importancia en contextos donde los fallos son raros o impredecibles. DINO, combinado con un enfoque tipo PatchCore, ofrece un método robusto y eficiente para identificar desviaciones sin necesidad de etiquetas de anomalía.

01 **Imágenes de referencia «normales»**

Se recopilan entre 10 y 50 imágenes que representan el estado normal o sin defectos del objeto o proceso a monitorear.

02 **Extracción de features DINO**

Se utiliza DINOv2 (ViT-B/14 para un buen balance) o DINOv3 (para mayor robustez) como extractor de características.

03 **Embeddings por parche**

Se extraen los tokens de atención (no solo el CLS) y se normalizan con L2, capturando representaciones detalladas para cada parche de la imagen.

04 **Generación del banco de normalidad (CoreSet)**

A partir de los embeddings de las imágenes normales, se construye un CoreSet (un subconjunto representativo del 1-10 %) para almacenar la normalidad.

05 **Imagen de test**

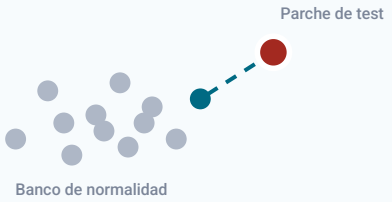
Una nueva imagen, que podría contener anomalías, se procesa de la misma manera para extraer sus embeddings por parche.

06 **Predicción de anomalías (kNN)**

Se calcula la distancia kNN (L2 o coseno) de cada parche de la imagen de test a su vecino más cercano en el CoreSet. Esto genera un heatmap de anomalías y un score global.

Lejos de todo lo normal

El banco solo guarda parches sin defectos: la puntuación es la distancia al más parecido que encuentre.



Esquema con dos dimensiones. DINO compara embeddings de varios cientos por parche.

Anomalía

Distancia al vecino

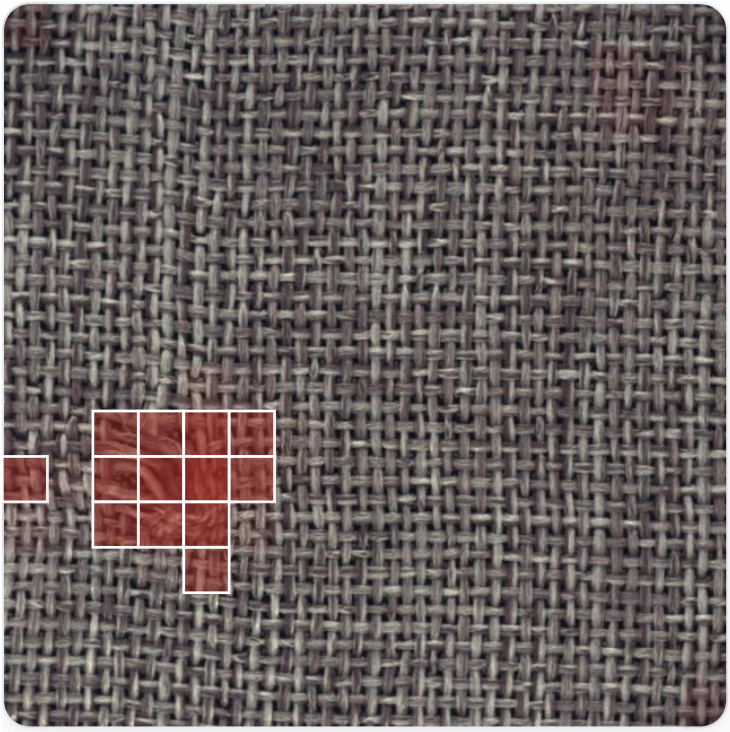
21,58

El parche queda a 21,58 de lo más parecido que hay en el banco, por encima del umbral 14,00.

Esta página recoge el estado con el parche al 55 % del recorrido. El banco no contiene ni un solo ejemplo de defecto: por eso el método sirve cuando los fallos son raros o desconocidos. Un parche se considera anómalo cuando ningún vector del banco se le parece lo suficiente, y el umbral se fija observando parches normales de validación, no adivinando. Con cada parche puntuado así, la imagen completa produce un heatmap y un score global.

Aceptar o rechazar la pieza

La misma alfombra, buena y con un defecto. Mueve el umbral y mira qué celdas se encienden en cada una.



Cada celda es un parche. El color de fondo es su puntuación; el borde marca las que superan el umbral.

Se rechaza

Celdas marcadas

13

Puntuación más alta

1,89

Qué pieza pasa por la cámara

Pieza con defecto

Pieza buena

El defecto queda marcado con 13 celdas, y la más alta llega a 1,89. La pieza sale de la línea.

> Así se monta una inspección real: el banco se llena con piezas buenas de la propia línea, el umbral se fija con un lote de validación mirando cuántas falsas alarmas aguanta el proceso, y el mapa sirve para enseñar al operario *dónde* está el problema, no solo que existe.

Esta página recoge el estado de la pieza defectuosa con umbral 1,30. Las puntuaciones están precalculadas con un descriptor clásico, no con DINO: cada celda de 35 × 35 píxeles se resume en siete estadísticas de intensidad y de gradiente, se normaliza con las celdas buenas y se puntúa por la distancia a la celda buena más parecida, que es exactamente la regla de PatchCore. Una celda de la pieza buena se compara con las demás celdas buenas y nunca consigo misma, así que su puntuación es honesta. El mapa se suaviza con una media de 3 × 3 para que un parche raro suelto no dispare la alarma. Con este descriptor la pieza buena llega como máximo a 1,28 y la defectuosa a 1,89: el margen existe pero es estrecho, y por debajo de 1,20 aparecen falsas alarmas. Con características de DINOv2 el margen se abre, que es justamente el motivo de usarlas, y el cuaderno del tema lo comprueba con el mismo par de imágenes.

Del embedding a la decisión

Un vector por sí solo no dice si algo está bien o mal.

Quieres detectar una anomalía industrial con DINO. ¿Qué falta además del embedding?

- A Ejemplos normales y evaluación del umbral.
- B Una etiqueta textual para CLIP.
- C Solo más memoria GPU.

Respuesta A. Exacto: la distancia requiere referencias y una decisión validada.

Hace falta una referencia de normalidad y un criterio de distancia. Sin banco y sin umbral, un embedding es solo un punto en el espacio.

Un vector por imagen, sin etiquetas

El extractor que alimenta búsqueda por similitud, clustering y detección de anomalías.

```
import torch
from PIL import Image
from transformers import AutoImageProcessor, AutoModel

model_id = "facebook/dinov2-base"
processor = AutoImageProcessor.from_pretrained(model_id)
model = AutoModel.from_pretrained(model_id).eval()

def embedding(ruta):
    image = Image.open(ruta).convert("RGB")
    inputs = processor(images=image, return_tensors="pt")
    with torch.no_grad():
        salida = model(**inputs).last_hidden_state
    cls = salida[:, 0]                      # resumen global de la imagen
    return torch.nn.functional.normalize(cls, dim=-1)

a, b = embedding("pieza_1.jpg"), embedding("pieza_2.jpg")
print("similitud:", float(a @ b.T))
```

- El **token CLS** resume la imagen entera. Los demás, `salida[:, 1:]`, son un vector por parche y sirven para lo espacial.
- **Normaliza siempre** antes de comparar: la similitud coseno mide dirección, no longitud.
- El modelo va congelado. No hay entrenamiento aquí, solo extracción.

Montar un detector de anomalías en una tarde

Sin un solo ejemplo de defecto, con las cifras de la diapositiva anterior.

01 **Reúne entre 10 y 50 imágenes normales**

De la misma cámara, la misma distancia y la misma luz que vas a tener en producción. La variabilidad que no esté aquí se marcará como anomalía.

02 **Extrae embeddings por parche y normaliza**

Usa los tokens de parche, no solo el CLS, y normaliza con L2. Así cada región tiene su propia representación.

03 **Construye el banco con un CoreSet del 1 al 10 %**

No guardes todos los vectores: quédate con un subconjunto que cubra el espacio. Reduce memoria y acelera la búsqueda sin perder cobertura.

04 **Fija el umbral con imágenes normales de validación**

Puntúa unas cuantas imágenes buenas que no estén en el banco y coloca el umbral por encima de su distancia máxima, con margen. No lo adivines.

05 **Puntúa y revisa el heatmap, no solo el número**

El máximo por parche da el score de la imagen; el mapa completo te dice si el modelo mira donde tú mirarías.

> **Señal de alarma:** si el heatmap se enciende en el fondo o en los bordes en vez de en la pieza, el problema es el encuadre o la iluminación, no el modelo.

Antes de fiarte de un banco de normalidad

Seis comprobaciones para un método que no ha visto ni un defecto.

- [ ] **Las imágenes normales cubren la variación real**
  Turnos, lotes, luz de mañana y de tarde. Lo que falte se marcará como anomalía.
- [ ] **Encuadre estable**
  Si la pieza cambia de posición entre fotos, la distancia sube sin que haya defecto.
- [ ] **Umbral fijado con validación**
  Con imágenes buenas apartadas, no con las del propio banco.
- [ ] **Heatmap revisado a ojo**
  Que se encienda donde está el defecto y no en el fondo.
- [ ] **Falsos positivos contados**
  Cuántas piezas buenas estás parando. Ese es el coste que notará producción.
- [ ] **Plan de actualización**
  Qué haces cuando cambie el proveedor, el lote o la iluminación.

Explora los embeddings

Extrae representaciones globales y densas con DINOv2.

1. Guarda una copia en Drive.
2. Compara dos imágenes.
3. Interpreta patches y PCA.

Abrir en Colab ↗

Compara el token global para similitud entre imágenes y los tokens de patch para estructura espacial. Observa si PCA, vecinos cercanos y distancia coseno cuentan una historia coherente; una visualización atractiva no sustituye la validación de la tarea final.