2. El ecosistema de modelos de IA

Buscar, evaluar licencias y usar modelos fácilmente.

Hugging Face se ha convertido en el **repositorio central de facto** para modelos de IA modernos, datasets, demos interactivas y herramientas de desarrollo. Su ecosistema integrado permite descubrir, probar y desplegar modelos.

El ecosistema Hugging Face

Hugging Face no es solo un repositorio, es un **ecosistema completo de cuatro pilares interconectados**.

Model Hub

Catálogo con más de 500.000 modelos preentrenados para todas las tareas imaginables: visión, NLP, audio, multimodal.

Datasets

Más de 100.000 datasets públicos listos para entrenar y evaluar, con APIs unificadas de carga y procesamiento.

Spaces

Demos web interactivas con Gradio o Streamlit, desplegadas sin servidor. Prueba modelos en segundos con interfaz gráfica.

Transformers

Librería Python que unifica la API de carga, inferencia y fine-tuning para miles de arquitecturas diferentes.

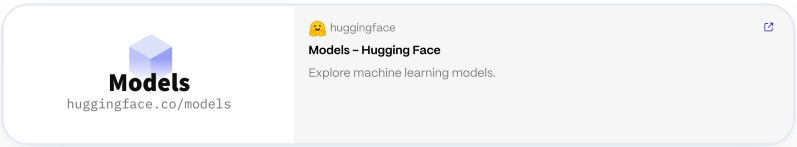
El flujo típico

01 **Buscar modelo**

02 **Probar en Space**

03 **Descargar con Transformers**

04 **Integrar en tu aplicación**



huggingface.co/models

Cómo buscar modelos efectivamente

El Model Hub permite **filtrar por múltiples criterios** para encontrar exactamente lo que necesitas:

- **Tarea:** object-detection, image-classification, text-generation, etc.
- **Licencia:** Apache-2.0, MIT, CreativeML, Research-only.
- **Popularidad:** número de descargas, likes, fecha de última actualización.

La Model Card es tu mejor amiga. Cada modelo debe incluir:

Descripción y uso

Para qué fue entrenado, qué tareas resuelve, cómo usarlo con código de ejemplo.

Limitaciones

Sesgos conocidos, casos donde falla, advertencias de seguridad y aspectos éticos.

Licencia

Términos legales de uso: ¿puedes usarlo comercialmente? ¿Debes atribuir autoría?

Archivos

Configuración JSON, pesos del modelo (.safetensors preferido), tokenizers, ejemplos.

Métricas

Resultados en benchmarks estándar, comparación con otros modelos.

Las descargas y los likes sirven para descubrir candidatos, no para decidir cuál funciona mejor en tu dominio. La fecha de última actualización avisa de modelos abandonados.

Antes de descargar, lee la ficha

La Model Card contesta si el modelo sirve para tu caso antes de gastar tiempo.

Buscas un clasificador para imágenes médicas. ¿Qué debes hacer primero?

- A Elegir el modelo con más descargas.
- B Filtrar por tarea y revisar la model card con el dominio y licencia en mente.
- C Descargar todos los modelos y comparar solo su tamaño.

Respuesta B. Exacto: la tarea reduce la búsqueda y la documentación permite juzgar la adecuación.

Un dominio sensible obliga a revisar para qué se entrenó el modelo, con qué datos y qué limitaciones declara. El número de descargas no responde a ninguna de esas preguntas.

Licencias: evita problemas legales

La licencia del modelo determina **cómo puedes usarlo legalmente**. Ignorar este aspecto puede causar problemas serios en producción.

Comercial OK

- **Apache-2.0:** muy permisiva, ideal para empresas.
- **MIT:** similar a Apache, aún más simple.
- **BSD:** variantes 2-clause y 3-clause, ambas permisivas.

Con restricciones

- **OpenRAIL/CreativeML:** uso comercial OK, pero con cláusulas de uso responsable.
- **CC BY-SA:** requiere compartir derivados bajo la misma licencia.

No comercial

- **Research Only:** solo investigación académica.
- **CC BY-NC:** prohibido uso comercial.

Caso especial AGPL: algunas familias YOLO populares usan AGPL, que **obliga a publicar tu código fuente completo** si integras el modelo en una aplicación web o servicio. Si no quieres publicar tu código, busca alternativas con Apache-2.0 como RT-DETR o RF-DETR.

Las condiciones se revisan por artefacto y por versión: el repositorio de código y los pesos publicados pueden llevar licencias distintas, y ambas pueden cambiar entre versiones. Esta clasificación orienta la búsqueda; la decisión final se toma leyendo la licencia vigente del artefacto concreto.

Comprueba licencia y caso de uso

La misma licencia cambia de respuesta según lo que vayas a hacer con el modelo.

AGPL

Servicio web o SaaS

Permitido con condiciones

La cláusula de red obliga a publicar tu código fuente completo aunque solo lo ofrezcas por web. Alternativas Apache-2.0: RT-DETR o RF-DETR.

Orientación para buscar, no asesoramiento legal. La licencia real se lee en el artefacto y la versión que vayas a usar.

La tabla recoge lo que explica la diapositiva anterior. Los tres casos que más sorprenden: AGPL no obliga a nada mientras no distribuyas ni sirvas el modelo, pero sí en cuanto lo expones por web; una licencia no comercial tampoco vale para un prototipo interno de empresa, porque el propósito ya es comercial; y CC BY-SA permite vender, pero obliga a publicar los derivados con la misma licencia.

Código y pesos pueden diferir

Un repositorio permisivo no garantiza que los pesos lo sean.

Un repositorio tiene código permisivo, pero los pesos incluyen otra licencia. ¿Qué debes revisar?

- A Solo la licencia del código.
- B Las condiciones de código, pesos, datos y uso de la versión concreta.
- C Nada si el modelo se puede descargar sin pagar.

Respuesta B. Exacto: cada recurso puede añadir términos distintos.

Conviene revisar la licencia del repositorio, la del checkpoint concreto y la de los datos de entrenamiento cuando se declara, además de la versión a la que se aplican.

Usar modelos

La librería Transformers hace que cargar y usar modelos sea **muy sencillo**. El modelo mental clave:

01 **Hub remoto**

El modelo vive en Hugging Face Hub identificado por su nombre (ej: «facebook/detr-resnet-50»).

02 **Descarga a caché**

Primera ejecución: descarga config.json, pesos .safetensors, y metadatos a ~/.cache/huggingface/

03 **Carga en memoria**

El modelo se instancia en RAM/VRAM según el dispositivo (CPU/GPU) que especifiques.

04 **Inferencia**

Procesas inputs y obtienes predicciones. Puedes usar GPU y mixed precision para acelerar.

Buenas prácticas

- **Seguridad:** prefiere .safetensors sobre .bin (formato más seguro).
- **Offline:** predescarga con snapshot_download, controla cache_dir.
- **Rendimiento:** usa float16 y device="cuda" cuando tengas GPU.
- **Pipelines:** para pruebas rápidas, usa pipeline("task"); en producción, clases específicas dan más control.

Inferencia en 3 líneas de código

```
from transformers import pipeline

# 1. Cargar modelo (automáticamente baja config y pesos)
detector = pipeline("object-detection", model="PekingU/rtdetr_r50vd")

# 2. Inferencia
results = detector("imagen.jpg")

# 3. Resultado
print(results)  # [{'label': 'cat', 'score': 0.99, 'box': {...}}]
```

> La librería **transformers** maneja la descarga, el pre-procesamiento y la inferencia automáticamente.

El identificador del modelo determina qué arquitectura y qué preprocesador se instancian. Para reproducibilidad conviene fijar también la revisión. En producción, las clases específicas permiten controlar batching, dispositivo, precisión y formato de salida.

Arma tu pipeline

Elige qué quieres hacer y dónde vas a ejecutarlo. El código de abajo es el que se pega en el cuaderno, con el modelo del Hub que corresponde a esa tarea.

Qué quieres hacer

- **Clasificar la imagen entera**
- Detectar los 80 objetos de COCO
- Decidir entre textos que yo escribo
- Localizar objetos descritos con palabras
- Describir la escena en una frase

Dónde lo ejecutas

CPU | GPU

```
from transformers import pipeline

pipe = pipeline(
    "image-classification",
    model="google/vit-base-patch16-224",
    device="cpu",
)
salida = pipe("foto.jpg")
print(salida[:3])
```

ViT base, 86 M de parámetros. Devuelve etiquetas de las 1.000 de ImageNet, así que solo sirve si tus clases están ahí. Arranca en cualquier portátil. Cuenta con segundos por imagen.

Las cuatro tareas usan la misma llamada y cambian el identificador del modelo, que es lo que hace el Hub cómodo para probar. Las dos que aceptan `candidate_labels` (CLIP y Grounding DINO) son las que permiten cambiar de clases sin reentrenar. El argumento `device` admite «cpu» o «cuda»; con varias tarjetas se indica el índice.

Qué te ahorra un pipeline

Conviene saber qué pasos quedan ocultos antes de llevarlo a producción.

¿Qué simplifica un pipeline de Transformers?

- A La preparación de entrada y la ejecución de una tarea común.
- B La necesidad de conocer la licencia y la model card.
- C El entrenamiento de cualquier modelo sin datos.

Respuesta A. Exacto: el pipeline agrupa pasos frecuentes para una primera prueba.

El pipeline reúne preprocesado, inferencia y posprocesado para una tarea común. No elige el modelo por ti ni garantiza el mejor rendimiento en tu dominio.

Del Hub a tu código

Dos formas de cargar el mismo modelo: rápida para probar, específica para producción.

```
# Para probar: el pipeline resuelve pre y posprocesado
from transformers import pipeline

detector = pipeline("object-detection", model="PekingU/rtdetr_r50vd")
print(detector("mi_imagen.jpg"))

# Para producción: control sobre dispositivo, precisión y salida
import torch
from transformers import AutoImageProcessor, AutoModelForObjectDetection

model_id = "PekingU/rtdetr_r50vd"
processor = AutoImageProcessor.from_pretrained(model_id, revision="main")
model = AutoModelForObjectDetection.from_pretrained(model_id, revision="main")
model = model.to("cuda").eval()
```

- Fija **revision** si quieres poder reproducir el resultado dentro de seis meses.
- Descarga a caché la primera vez, a `~/.cache/huggingface/`. Para entornos sin red, `snapshot_download` antes.
- Prefiere los pesos en **.safetensors** cuando el repositorio los ofrezca.

¿Me cabe en la GPU?

Primera pregunta al elegir modelo, y se responde con una multiplicación.

Parámetros del modelo **300 M**

Precisión

float16 o bfloat16

T4 · 16 GB | L4 · 24 GB | A100 · 40 GB | Colab gratis · 15 GB | Portátil · 8 GB

Memoria estimada para inferencia

0,8 GB

Solo los pesos

0,6 GB

Cabe en 5 de las 5 tarjetas de la lista, contando solo inferencia.

Pesos por precisión más un 35 % de margen para activaciones y fragmentación, con lotes pequeños. Entrenar necesita bastante más, porque además guarda gradientes y estados del optimizador.

La cuenta es directa: parámetros por bytes de cada precisión. En float32 cada parámetro ocupa 4 bytes, en float16 dos y cuantizado a int8 uno. Por eso pasar de float32 a float16 es lo primero que se prueba cuando un modelo no entra, y suele costar muy poca calidad en inferencia.

Candidatos reales para visión

Filtra por tarea y por licencia, que son los dos criterios que descartan antes.

`PekingU/rtdetr_r50vd`	Detección COCO, tiempo real	**Apache-2.0**	~42 M
`facebook/detr-resnet-50`	El DETR original, de referencia	**Apache-2.0**	~41 M
`hustvl/yolos-small`	Detección con ViT puro	**Apache-2.0**	~31 M
`IDEA-Research/grounding-dino-base`	Cajas a partir de una frase	**Apache-2.0**	~233 M
`google/owlv2-base-patch16`	Vocabulario abierto por texto	**Apache-2.0**	~154 M
`facebook/sam2.1-hiera-large`	Segmentación con prompts, imagen y vídeo	**Apache-2.0**	~224 M
`facebook/dinov2-base`	Features densas sin etiquetas	**Apache-2.0**	~86 M
`openai/clip-vit-base-patch32`	Imagen y texto en un espacio común	**MIT**	~151 M
`briaai/RMBG-1.4`	Recorte de fondo con canal alfa	**No comercial**	~44 M

Lista consultada el 17 de septiembre de 2026 y pensada como punto de partida, no como ranking. Los tamaños son aproximados y la licencia debe confirmarse en la ficha del modelo y de sus pesos antes de usarlo, porque puede cambiar entre versiones. Fíjate en el caso de RMBG-1.4: excelente para recortar fondo, pero su licencia no permite uso comercial sin acuerdo.

Pruébalo ahora, sin instalar nada

Un Space es el modelo ya montado y corriendo en el navegador. Arrastras tu foto y ves la salida, sin entorno, sin pesos y sin GPU.

Quitar el fondo

Sube una foto de producto y descarga el recorte con transparencia. Es el modelo del tema 6 puesto a trabajar.

Space de BRIA RMBG 1.4

Profundidad de una sola foto

Sube cualquier foto y mira el mapa de distancias. Prueba con una imagen con espejos o cristales y verás dónde falla.

Space de Depth Anything V2

Detectar escribiendo la clase

Página de la tarea, con los modelos, los datasets y los Spaces que están vivos hoy. Escribe «una caja de cartón» y mira qué encuentra.

Tarea: detección zero-shot

Segmentar

La misma idea para máscaras: modelos de segmentación semántica, de instancias y promptable, con demos que puedes abrir al momento.

Tarea: segmentación de imagen

Puntos clave

Pose de personas y puntos de interés para matching, con los 413 modelos que hay publicados y sus demos.

Tarea: detección de puntos clave

Preguntar sobre una imagen

Los modelos que reciben imagen y texto y responden. Sube una foto de una etiqueta y pídele los campos en JSON.

Tarea: imagen y texto a texto

El reto de diez minutos

- [ ] **Busca por tarea, no por nombre**
  Entra en la página de la tarea que te interesa y ordena por descargas del último mes.
- [ ] **Abre un Space y sube una foto tuya**
  Una de tu trabajo, no una de internet. Lo que falle aquí va a fallar en producción.
- [ ] **Mira la licencia antes de emocionarte**
  Está en la ficha del modelo, no en la del Space. Si pone AGPL o «research only», anótalo.
- [ ] **Copia el fragmento de la ficha**
  Casi todas traen tres líneas listas para pegar en el cuaderno.
- [ ] **Apunta el tamaño y la fecha**
  Parámetros y última actualización. Un modelo abandonado hace tres años es una deuda, no un atajo.

> **Dos avisos antes de subir nada.** Un Space público procesa tu imagen en el servidor de quien lo publicó, así que ahí no van fotos de cliente, de personas identificables ni de piezas bajo acuerdo de confidencialidad. Y los Spaces se duermen, se rompen y desaparecen: por eso los enlaces de arriba apuntan a la página de la tarea, que sigue viva aunque una demo concreta caiga.

Un Space es un repositorio con una aplicación Gradio o Streamlit y un fichero de dependencias, que Hugging Face construye y ejecuta. En el nivel gratuito corre en CPU y se duerme tras un rato sin visitas, de ahí que a veces tarde en arrancar o falle al construirse. Cualquiera puede duplicar un Space en su cuenta con el botón de duplicar, cambiarle el modelo y tener su propia demo en minutos, que es la forma más rápida de enseñar un resultado a alguien que no programa. Para un piloto interno conviene duplicarlo en privado en lugar de usar el público.

Antes de quedarte con un modelo del Hub

Siete comprobaciones que evitan rehacer el trabajo.

- [ ] **La tarea coincide**
  La etiqueta del Hub dice lo que necesitas, no algo parecido.
- [ ] **La licencia permite tu uso**
  Comprobada en el repositorio y en los pesos, para la versión que vas a usar.
- [ ] **La ficha declara limitaciones**
  Sesgos, casos donde falla, datos de entrenamiento. Si no dice nada, desconfía.
- [ ] **Cabe en tu hardware**
  Con la precisión que vas a usar y con margen para el lote.
- [ ] **Formato de pesos seguro**
  .safetensors antes que .bin.
- [ ] **Revisión anotada**
  Identificador y revisión guardados para poder reproducir.
- [ ] **Probado con tus datos**
  Una decena de imágenes tuyas dicen más que cualquier métrica publicada.

Busca y prueba un modelo

Compara la documentación con lo que observas al ejecutar el cuaderno.

1. Guarda una copia en Drive.
2. Ejecuta el modelo propuesto.
3. Elige otro y justifica la decisión.

Abrir en Colab ↗

Para que la comparación sea reproducible, anota el identificador y la revisión del modelo, la tarea, la licencia, el preprocesador, el dispositivo y el consumo aproximado de memoria. Las descargas o likes sirven para descubrir candidatos, no para decidir cuál funciona mejor en tu dominio.