1.3 Plataformas y herramientas

En los últimos años ha habido un desarrollo exponencial de tecnologías vinculadas a la IA en todos los niveles y ámbitos profesionales

Este curso pretende dar una visión completa de las mismas e iniciar al alumno en un uso más profundo y flexible de la IA para adaptarla a sus necesidades y reducir la dependencia.

> **Aunque se habla de *python* y librerías asociadas no es en absoluto la idea del curso aprender a programar pero sí ofrecer dicha posibilidad a los alumnos que quieran introducirse en el tema.**

A continuación mostramos las principales y más populares plataformas y tecnologías. Todas ellas se usan en todo el mundo como referencia y base de pruebas y aprendizaje de IA con modelos y conjuntos de datos propios y ajenos

Modelos de IA

Entre los **modelos de IA más populares** se pueden distinguir varios grupos según el tipo de información que procesan y su aplicación principal.

Modelos de lenguaje (LLM)

Están diseñados para **comprender y generar texto**. Se utilizan en asistentes conversacionales, generación de contenido, programación asistida, traducción o análisis de documentos.
Ejemplos representativos:

- **GPT** (OpenAI): base de muchos sistemas conversacionales.
- **BERT** (Google): muy utilizado en comprensión del lenguaje y buscadores.
- **LLaMA** (Meta): modelo abierto que ha impulsado el ecosistema *open source*.
- **Mistral, Qwen o Phi**: modelos más ligeros que permiten ejecutar IA en local.

Modelos de visión por computador

Están diseñados para **analizar imágenes o vídeo**. Permiten tareas como reconocimiento de objetos, clasificación de imágenes o detección de patrones visuales.
Ejemplos:

- **CNN (Convolutional Neural Networks)**: arquitectura clásica para reconocimiento de imágenes.
- **Vision Transformers (ViT)**: aplican la arquitectura transformer al análisis visual.
- **Modelos de difusión** (como **Stable Diffusion**): generan imágenes a partir de texto.

Modelos de audio y voz

Procesan **sonido o lenguaje hablado**. Se utilizan para reconocimiento de voz, síntesis de voz o generación de música.
Ejemplos destacados:

- **Whisper** (OpenAI): transcripción automática de voz a texto.
- **WaveNet** (DeepMind): generación de voz natural.
- **MusicGen** o **AudioLM**: generación de música o audio mediante IA.

Modelos multimodales

Estos modelos pueden **combinar varios tipos de datos** (texto, imagen, audio, etc.). Permiten tareas como describir imágenes, responder preguntas sobre vídeos o interactuar con distintos tipos de contenido al mismo tiempo.
Ejemplos:

- **CLIP** (OpenAI): relaciona texto e imágenes.
- **LLaVA**: integra modelos de lenguaje y visión.
- **Gemini** o **GPT multimodal**: capaces de trabajar con múltiples modalidades de información.

Sistemas avanzados basados en LLM: agentes de IA

Una evolución reciente son los **agentes de inteligencia artificial**, que utilizan modelos de lenguaje como núcleo de razonamiento pero además pueden **planificar tareas, usar herramientas externas y ejecutar acciones**.

Cuando varios agentes especializados colaboran dentro de un mismo sistema coordinado se habla de **orquestación de agentes**, una de las tendencias más recientes en el desarrollo de aplicaciones avanzadas de inteligencia artificial.

Plataformas de desarrollo y pruebas de modelos

Existen diferentes tipos de plataformas para trabajar con inteligencia artificial, que van desde **entornos colaborativos en la nube** hasta **herramientas que permiten ejecutar modelos de forma local**.

Plataformas de modelos y repositorios de IA

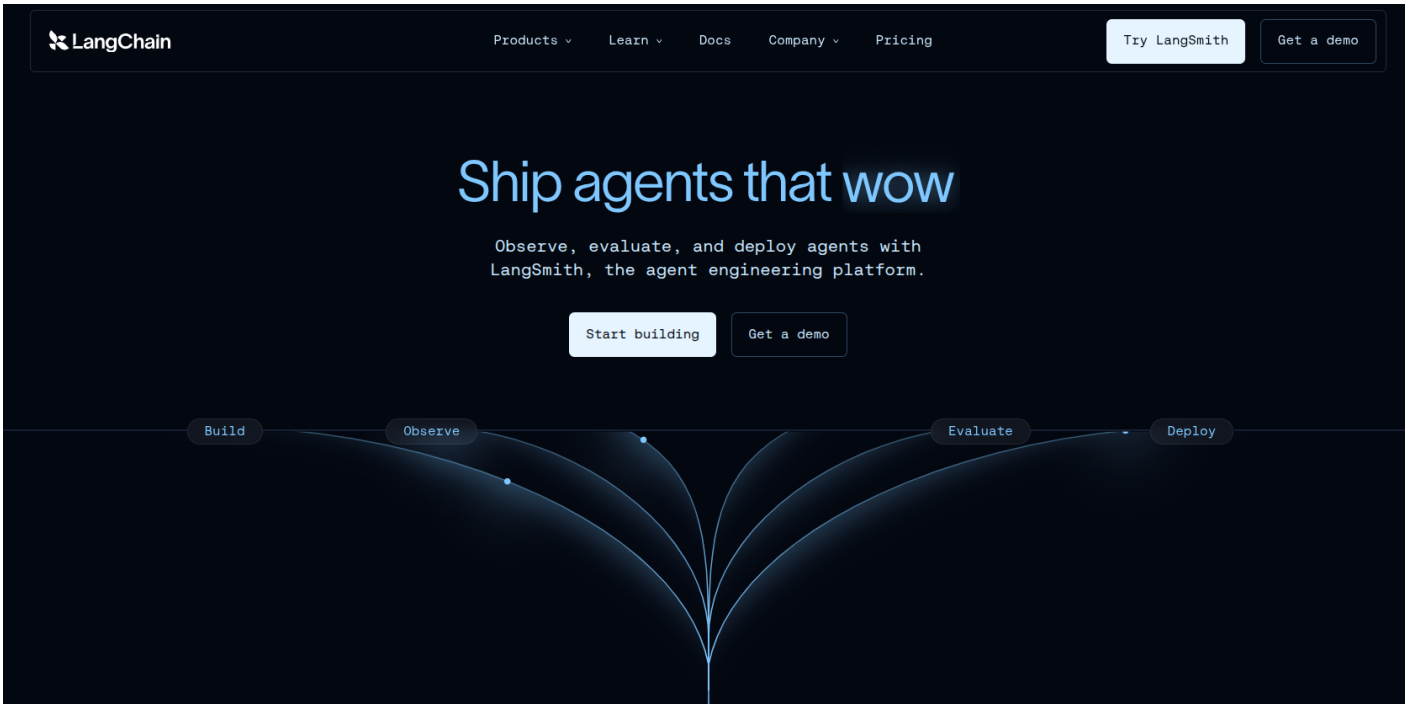
Son espacios donde investigadores y desarrolladores publican modelos, datasets y demos listas para usar.
Ejemplos:

- **Hugging Face**: uno de los mayores repositorios de IA del mundo, con millones de modelos, datasets y demos interactivos (*Spaces*).
- **ModelScope** o repositorios similares: plataformas que permiten descargar y probar modelos abiertos.

Frameworks para construir aplicaciones con LLMs

Facilitan la creación de aplicaciones complejas combinando modelos, bases de datos y herramientas externas.
Ejemplos:

- **LlamaIndex***: especializado en integrar datos propios con modelos de lenguaje.*
- **LangChain**: framework para conectar LLMs con fuentes de datos, APIs o bases de conocimiento.



Vista de la web de lancghain para programar, explorar y probar modelos

Entornos de desarrollo en la nube

Permiten experimentar con IA sin instalar nada en el ordenador, usando notebooks con acceso a GPU o TPU.
Ejemplos:

- ***Google Colab**: notebooks de Python en la nube con acceso sencillo a GPUs.*
- ***Kaggle Notebooks**: entorno similar, muy utilizado en ciencia de datos y competiciones de IA.*

Plataformas empresariales de Machine Learning

Ofrecen infraestructuras completas para entrenar, desplegar y gestionar modelos de IA en producción.
Ejemplos:

- ***AWS SageMaker**: entorno de desarrollo y despliegue de modelos en la nube de Amazon.*
- ***Azure Machine Learning**: plataforma de Microsoft para crear y operar sistemas de IA.*

Plataformas para ejecutar IA en local

Permiten trabajar con modelos abiertos sin depender de servicios en la nube, lo que mejora la privacidad y el control de datos.
Ejemplos:

- ***Ollama**: herramienta sencilla para ejecutar modelos open source en el ordenador.*
- ***LM Studio**: interfaz gráfica para descargar y usar LLMs locales.*

Motores de inferencia optimizados

Son herramientas especializadas para ejecutar modelos grandes de forma eficiente.
Ejemplos:

- ***vLLM**: motor optimizado para servir LLMs con alto rendimiento y bajo consumo de memoria, muy utilizado en servidores de IA.*

En conjunto, estas plataformas permiten **experimentar, desarrollar y desplegar aplicaciones de inteligencia artificial**, tanto en entornos educativos y de investigación como en sistemas profesionales a gran escala.

Técnicas clave

El uso práctico de los modelos de inteligencia artificial no depende solo del modelo en sí, sino también de una serie de técnicas que permiten adaptarlos, controlarlos o integrarlos con datos propios.

Prompting (ingeniería de instrucciones)

Consiste en formular correctamente las instrucciones que se dan al modelo para obtener mejores resultados. Incluye técnicas como zero-shot prompting (pedir una tarea directamente sin

ejemplos), few-shot prompting (mostrar algunos ejemplos para guiar la respuesta) y chain-of-thought prompting, donde se pide al modelo que razone paso a paso para mejorar la calidad de la respuesta.

RAG (Retrieval-Augmented Generation)

Es una técnica que permite a un modelo consultar información externa antes de generar la respuesta. En lugar de depender solo de lo aprendido durante el entrenamiento, el modelo puede buscar información en bases de datos, documentos o repositorios y utilizar esos datos como contexto. Se utiliza con frecuencia en sistemas de consulta sobre documentos, asistentes empresariales o aplicaciones de “hablar con tus datos”.

Fine-tuning (ajuste fino)

Consiste en volver a entrenar un modelo fundacional utilizando datos específicos para especializarlo en una tarea concreta. Por ejemplo, un modelo general puede ajustarse con documentos médicos, legales o educativos para mejorar su precisión en ese ámbito.

Embeddings y búsqueda semántica

Los modelos pueden transformar textos en representaciones numéricas llamadas embeddings que capturan el significado del contenido. Esto permite realizar búsquedas semánticas, encontrar documentos relacionados o agrupar información similar aunque no contengan exactamente las mismas palabras.

Uso de herramientas externas

Los modelos pueden conectarse con APIs, bases de datos, buscadores o programas externos para ampliar sus capacidades. De esta forma pueden consultar información actualizada, realizar cálculos o ejecutar acciones fuera del propio modelo.

Agentes de inteligencia artificial y orquestación de agentes

Una evolución reciente consiste en utilizar los modelos de lenguaje como agentes capaces de planificar tareas, tomar decisiones y utilizar herramientas de forma autónoma. En sistemas más avanzados, varios agentes especializados pueden colaborar entre sí bajo un sistema de coordinación, lo que se conoce como orquestación de agentes. Este enfoque permite construir aplicaciones de IA más complejas capaces de resolver tareas largas o procesos completos de trabajo.

Infraestructura y hardware

El desarrollo y uso de sistemas de inteligencia artificial requiere una infraestructura informática capaz de procesar grandes volúmenes de datos y ejecutar modelos complejos. Dependiendo del

tipo de aplicación, esta infraestructura puede ir desde ordenadores personales hasta centros de datos especializados.

Procesadores especializados

Los modelos de IA suelen ejecutarse en hardware optimizado para cálculos paralelos. Las unidades de procesamiento gráfico (GPU) se han convertido en el estándar para entrenar y ejecutar redes neuronales, ya que pueden realizar miles de operaciones simultáneamente. También existen otros aceleradores como las TPU (Tensor Processing Units) desarrolladas por Google o los chips especializados para IA integrados en dispositivos móviles.

Servidores y centros de datos

Las empresas que desarrollan modelos fundacionales suelen utilizar grandes centros de datos con miles de GPUs conectadas entre sí. Estas infraestructuras permiten entrenar modelos con billones de parámetros utilizando enormes cantidades de datos. Los centros de datos modernos también incluyen sistemas de almacenamiento de alto rendimiento y redes de alta velocidad para mover grandes volúmenes de información entre máquinas.

Computación en la nube

Muchos proyectos de inteligencia artificial utilizan servicios de computación en la nube que permiten acceder a hardware potente sin necesidad de comprarlo. Plataformas como AWS, Google Cloud o Microsoft Azure ofrecen instancias con GPU o TPU que se pueden alquilar por horas para entrenar o ejecutar modelos de IA.

Infraestructura local

Además de la nube, cada vez es más común ejecutar modelos de inteligencia artificial en equipos locales. Ordenadores personales con GPUs modernas pueden ejecutar modelos abiertos de tamaño medio, especialmente si están optimizados para uso local. Esto permite mayor control sobre los datos y reduce la dependencia de servicios externos.

Optimización y eficiencia

Debido al gran consumo de recursos de los modelos actuales, han surgido técnicas para mejorar su eficiencia, como la cuantización de modelos, la reducción de precisión o el uso de motores de inferencia optimizados. Estas técnicas permiten ejecutar modelos grandes en hardware más limitado, facilitando su uso en entornos educativos, dispositivos personales o aplicaciones empresariales.

El entrenamiento y la inferencia de modelos IA requieren hardware acelerado. A continuación se compara los principales:

Tipo de hardware	Características/uso	Rendimiento relativo	Costo/ejecución	Accesibilidad educativa
GP **U (NVIDIA)**	Procesador paralelo general. Optimizado para matrices (CUDAM, Tensor Cores). Soporta PyTorch, TF.	H100/A100: ~1–3 PFLOPS (FP16) por unidad. Gran VRAM (80–141GB). Soporta batch grande y redes de atención extensas.	Alto: $4–10/h (GPU en nube). Tarjetas PC ~$800–$3000 según modelo.	Muy accesibles: Colab/Kaggle ofrecen GPUs gratis; muchas universidades usan GPUs gaming.
TPU (Google Cloud)	ASIC tensor específico. Integración fuerte con TensorFlow/JAX. Diseñado para inferencia y entrenamiento de ML en la nube. No disponible fuera de Google Cloud.	v6e: ~2 PFLOPS FP16 por chip. Masivo paralelismo (bajo costo por token).	Pago por uso: ~$2.70/h por TPU v6e (nube Google). No hay versión local; uso sólo en servicios Google (Cloud TPU o Colab TPU gratuita).	Limitado: Colab da pequeñas TPUs gratis; uso educativo real en nube (p.ej. Google Cloud for Education créditos).
NPU / Neural Engine	Unidades IA en chips de móviles/PCs (ex. Apple, Huawei). Muy eficientes energéticamente. Se usan en visión, NLP en dispositivo.	Ej.: Apple ANE v5 (A15): 15.8 TFLOPS (FP16). La primera ANE (A11) fue 0.6 TFLOPS; cada gen crece mucho.	Integrado en dispositivos (smartphone/tablet). No se compra separado. Costo = el dispositivo (iPhone/AirPods/Mac con M-series).	Alta: Los estudiantes llevan móviles con NPU. Google Coral (Edge TPU) ~$75 es asequible para demos de edge.
FPGA	Hardware reconfigurable (p.ej. Xilinx). Puede diseñarse el circuito específico para IA.	Rendimiento moderado. Menos paralelo que GPU en FP, pero baja latencia.	Alto de entrada: tarjetas FPGA avanzadas ~miles USD.	Bajo: Difícil de programar (Verilog) en cursos básicos; se usa más en investigación/industria. Existen kits educativos (Digilent) pero limitados.
ASIC (EdgeTPU)	Chips específicos para IA (ej. Google Edge TPU, USB accelerator). Ultraeficientes para inferencia puntual.	Edge TPU (Google): ~4 TOPS/W. Rendimiento limitado a modelos pequeños (p.ej. MobileNet, BERT pequeño).	Moderado: Edge TPU USB ~$75. Otros ASIC (Graphcore IPU, Habana) solo en servidores costosos.	Bueno: Edge TPUs para IoT / educación (Raspberry Pi + Coral). TPU/ASIC empresariales no disponibles en escuela.
Neuromórficos	Chips de investigación (Intel Loihi, IBM TrueNorth). Imitan redes neuronales físicas spiking.	Aún experimentales. Muy bajo consumo (ej. mil millones de OPS por segundo gastando milivatios).	*Experimental.* No comercial generalizada.	Muy bajo: solo en laboratorios especializados.

En resumen: las **GPU** son el estándar ampliamente usado (fáciles de acceder en colabs, PCs propias o nubes académicas). Las **TPU** ofrecen mayor eficiencia por coste en cargas de inferencia, pero sólo están en Google Cloud (aunque Colab da acceso limitado). Los **NPUs** son útiles para IA

en móviles y dispositivos embebidos, mejorando privacidad y energía. FPGAs y ASICs sirven para casos muy particulares, no tan comunes en entornos educativos. Los aceleradores neuromórficos son aún investigación.

Además, como muestra la comparación de [49], GPUs (p.ej. NVIDIA H100/H200) tienen más VRAM y mejor soporte software (CUDA/PyTorch), mientras que TPUs se especializan en cargas TensorFlow con alta eficiencia. Por ejemplo, la H100 entrega ~150 tokens/s para LLaMA-70B con vLLM en AWS (mayor throughput), mientras que un TPU v6e puede dar ~120 tokens/s con TensorFlow pero con sólo 32 GB de memoria, necesitando 8 chips para LLaMA-70B.

Dónde ejecutar modelos: nube vs local vs edge

- **Nube**: Plataformas como Colab, AWS, Azure facilitan la puesta en marcha sin instalar nada. Se escala según demanda pero requiere conexión y genera costos (por cómputo/tiempo). Útil para demos en clase o proyectos que necesitan GPUs fuertes puntualmente.
- **Local (on-premises)**: Ejecutar modelos en PCs, laptops o servidores propios. Ventaja en control de datos (privacidad) y sin latencia de red. Limitación en recursos de hardware: típicamente sólo CPUs o GPU de escritorio (RTX/Pascal/Turing) y menor RAM que un servidor. A menudo viable para inferencia en modelos medianos o entrenamiento ligero.
- **Edge (dispositivos)**: Modelos corriendo en smartphones, IoT o dispositivos embebidos. Ventajas: latencia ultrabaja, privacidad total (datos no salen del aparato), operación offline. Desventajas: recursos muy limitados (se usan NPUs con poca memoria). Como indica el caso de Multiverse, comprimir modelos para edge puede democratizar IA («Edge Computing: enable AI on resource-limited devices, reducing cloud reliance»). En educación, esto se ve en proyectos que usan móviles o Arduino/NPU (p.ej. reconocimiento de imágenes on-device).

Bibliotecas python

Para los que se animen a programar estas son las librerías más populares de amplio uso en el mundo del desarrollo de la IA

PyTorch

Es una biblioteca orientada al aprendizaje profundo que permite trabajar con tensores y entrenar redes neuronales utilizando CPU o GPU. Se caracteriza por su modo de ejecución flexible, muy utilizado en investigación, y por su ecosistema de herramientas especializadas como TorchVision para visión por computador o TorchAudio para procesamiento de audio. También ofrece utilidades para producción como TorchScript o TorchServe.

TensorFlow

Es una plataforma desarrollada por Google para crear y desplegar modelos de aprendizaje automático de forma completa. Incluye APIs sencillas como Keras para diseñar redes neuronales y herramientas adicionales para distintos entornos, como TensorFlow Lite para dispositivos móviles o TFX para sistemas de producción. Los modelos pueden exportarse en diferentes formatos para ejecutarse en servidores o dispositivos.

Scikit-learn

Es una de las bibliotecas más utilizadas en Python para aprendizaje automático clásico. Proporciona implementaciones de algoritmos de regresión, clasificación, clustering y reducción de dimensionalidad. Aunque no está diseñada para redes neuronales profundas, es muy utilizada en ciencia de datos para tareas de preprocesamiento, evaluación de modelos y experimentación educativa.

Revision #8
Created 2026-03-04 17:44:36 CET by Luis Hueso
Updated 2026-03-16 20:14:05 CET by Luis Hueso