Actumlogos

La palabra reflexionada puesta en acción

Vision-Language Models:

Del primer token a la detección por instrucción

Taller • 12 horas • Zoom

Construye, a mano y desde cero, un modelo que ve y habla. Sin pesos preentrenados y sin bibliotecas de modelado: solo PyTorch, torchvision, numpy y matplotlib. Escribes las tres piezas que forman cualquier VLM moderno —un Vision Transformer que convierte una imagen en tokens, un decodificador generativo que produce texto y el conector multimodal que los une— y las entrenas desde inicialización aleatoria hasta que tu modelo de ~19 M de parámetros describe una escena y detecta objetos emitiendo sus coordenadas como texto. Honestos con el alcance: el corpus es sintético y controlado —escenas de figuras geométricas con color, posición y cajas exactas, generadas en el propio cuaderno—, y eso es justo lo que hace viable el "desde cero" dentro del tiempo de laboratorio y te deja ver cada mecanismo funcionando, no escondido detrás de una API. Al terminar no habrás llamado a un modelo ajeno: habrás escrito, línea por línea, cada componente y entenderás por qué funciona.

Lo que harás:

- Generar un corpus sintético y controlado (figuras geométricas con color, posición y cajas exactas) directamente en el cuaderno.
- Implementar un Vision Transformer desde cero: parcheo, proyección a tokens, codificación posicional 2D y bloque codificador.
- Entender y mitigar el problema de datos escasos: parches desplazados, atención con temperatura aprendible y aumento agresivo.
- Construir un tokenizador BPE sobre tu propio corpus y un decodificador causal moderno con RoPE, RMSNorm, SwiGLU y caché de claves y valores.
- Escribir el conector multimodal y comparar proyección lineal, perceptrón, agrupamiento espacial, remuestreo por atención y atención cruzada.
- Alinear modalidades y ajustar por instrucciones con un currículo de entrenamiento por etapas.
- Entrenar el modelo para describir imágenes y diagnosticar fallas típicas (colapso a la descripción frecuente, indiferencia a la imagen).
- Plantear la detección de objetos como modelado de lenguaje: cuantizar coordenadas en tokens y evaluar con IoU y precisión.

Herramientas a aprender a usar: PyTorch, torchvision, numpy y matplotlib. Nada más: sin pesos preentrenados y sin bibliotecas de modelado.

Contenido

Sesión 1 · Panorama y el Vision Transformer

Arrancas viendo el final: una demostración del modelo que vas a construir. Con ese objetivo claro, entiendes el mapa de las arquitecturas multimodales y por qué se impuso el enfoque del prefijo visual. Generas el corpus sintético que usarás todo el taller y pones los cimientos del codificador visual: cómo una imagen se convierte en tokens y qué decisión se esconde detrás del tamaño de parche.

- Panorama de los VLM: demostración del modelo terminado, arquitecturas multimodales y por qué se impuso el prefijo visual. Generación del corpus sintético.
- Vision Transformer, el fundamento: parcheo y proyección a tokens; el compromiso del tamaño de parche. Codificación posicional bidimensional. Por qué el codificador no lleva máscara causal.

Sesión 2 · El ViT en código y el problema de los datos escasos

Bajas la teoría a PyTorch: escribes la capa de parcheo, los embeddings posicionales, el bloque codificador y ensamblas el ViT completo, verificando formas y el costo de atención según el parche. Luego enfrentas el problema real de entrenar desde cero con pocos datos: sin el sesgo inductivo de localidad el modelo subajusta, y aprendes las técnicas para compensarlo.

- Laboratorio 1 — Implementar el ViT: capa de parcheo, embeddings posicionales, bloque codificador y ensamblado. Verificación de formas y del costo de atención según el parche.
- El problema de los datos escasos: ausencia de sesgo inductivo de localidad y subajuste. Parches desplazados, atención con temperatura aprendible y aumento agresivo. Preentrenamiento y mapas de atención.

Sesión 3 · Tokenizador y decodificador de texto

Ahora construyes el lado del lenguaje. Entrenas un tokenizador BPE sobre tu propio corpus y defines los tokens especiales del sistema. Escribes un decodificador causal con los ingredientes de los modelos actuales —RoPE, RMSNorm, SwiGLU y caché de claves y valores— y lo entrenas solo con texto, sin imágenes todavía, para estabilizar la alineación que viene después.

- Tokenizador y decodificador: tokenizador BPE sobre el corpus propio y tokens especiales del sistema. Decodificador causal: RoPE, RMSNorm, SwiGLU y caché de claves y valores.
- Laboratorio 2 — Decodificador solo texto: entrenamiento sobre las descripciones, sin imágenes. Pruebas de causalidad y de equivalencia de la caché. Estabiliza la alineación posterior.

Sesión 4 · El conector multimodal

Aquí unes los dos mundos. Estudias las formas de conectar visión y lenguaje —proyección lineal, perceptrón, agrupamiento espacial, remuestreo por atención y atención cruzada— con sus ventajas y costos, y decides qué congelar y con qué tasas entrenar. En el laboratorio haces la unión fina: sustituyes el token marcador, alineas posiciones y máscaras, enmascaras etiquetas y resuelves el relleno por lotes.

- El conector multimodal: proyección lineal, perceptrón, agrupamiento espacial, remuestreo por atención y atención cruzada: ventajas y costos. Congelado y tasas diferenciadas.
- Laboratorio 3 — La unión: sustitución del token marcador, máscara de atención, alineación de posiciones, enmascaramiento de etiquetas y relleno por lotes. Pruebas.

Sesión 5 · Currículo de entrenamiento y descripción de imágenes

Con las tres piezas listas, el orden en que las entrenas lo decide todo. Aprendes el currículo por etapas —alineación de modalidades primero, ajuste por instrucciones después— y qué se congela en cada una y por qué. Luego lo ejecutas de punta a punta para la tarea de descripción, generando muestras periódicas y aprendiendo a leer los diagnósticos: cuándo el modelo colapsa a la descripción más frecuente o ignora la imagen.

- Currículo de entrenamiento: alineación de modalidades y ajuste por instrucciones; qué se congela en cada etapa y por qué el orden importa. Plantilla única para ambas tareas.
- Laboratorio 4 — Descripción de imágenes: ejecución de ambas etapas con generación periódica sobre un conjunto fijo. Diagnóstico: colapso a la descripción frecuente e indiferencia a la imagen.

Sesión 6 · Detección como modelado de lenguaje

Cierras con la idea más elegante del taller: tratar la detección de objetos como generación de texto. Cuantizas las coordenadas en tokens discretos —cinco por objeto— y decides el número de intervalos y el orden de los objetos en la secuencia. En el último laboratorio serializas anotaciones, entrenas de forma conjunta, dibujas las cajas y evalúas con IoU y precisión.

- Detección como modelado de lenguaje: cuantización de coordenadas en tokens discretos; cinco tokens por objeto. Número de intervalos y ordenamiento de los objetos en la secuencia.
- Laboratorio 5 — Detección y evaluación: serialización de anotaciones, entrenamiento conjunto, dibujo de cajas, IoU y precisión.

Al terminar tendrás:

- Un VLM completo escrito por ti en PyTorch —codificador visual, decodificador y conector— entrenado desde inicialización aleatoria.
- Un modelo de ~19 M de parámetros que describe escenas y detecta objetos emitiendo sus coordenadas como texto.
- La comprensión real de cada mecanismo (ViT, codificación posicional 2D, RoPE, RMSNorm, SwiGLU, caché KV, conectores multimodales), no una API que llamas sin saber qué hace.
- El criterio para saber cuándo el enfoque "desde cero" es adecuado y cuándo conviene partir de troncales preentrenados.
- Un cuaderno reproducible con corpus, entrenamiento y evaluación que puedes extender a tus propios experimentos.
- Constancia de participación verificable en línea.

¿Para quién es este taller?

- Personas que ya usan PyTorch y modelos de deep learning pero los tratan como caja negra y quieren entender, por dentro, cómo funciona un VLM.
- Estudiantes e ingenieros de ML y visión por computadora que quieren pasar de "usar un modelo multimodal" a "poder construir uno".
- Quienes preparan investigación o proyectos multimodales y necesitan dominar cada componente para modificarlo con criterio.

Este taller NO es para ti si:

- Buscas un curso introductorio de Python o de deep learning. Aquí se asume que ya programas en PyTorch y entiendes redes neuronales; el foco está en la arquitectura multimodal.
- Quieres ajustar (fine-tunear) modelos preentrenados como LLaVA o Qwen-VL. Aquí construyes todo desde cero, sin pesos preentrenados ni bibliotecas de modelado.
- Esperas resultados de producción sobre fotografías reales. El dominio es sintético y controlado a propósito; sobre fotos reales, entrenar a esta escala desde cero da resultados pobres y la ruta correcta sería otra.

Requisitos

Python y PyTorch a nivel intermedio, y fundamentos de redes neuronales y entrenamiento (retropropagación, optimizadores, sobre/subajuste). Se recomienda contar con una GPU, propia o en la nube, para seguir los laboratorios en vivo.

Clases

2 semanas, 6 sesiones de 2 horas (12 hrs en total): Ma22, Mi23, J24, Ma29, Mi30 de septiembre y J1 de octubre de 2026, de 20:00 a 22:00 (hora de la Ciudad de México). Todo se imparte por Zoom y queda grabado; tendrás acceso a las grabaciones durante 1 año para repasar a tu ritmo.

Instructor

M. en C. Israel Prado — LinkedIn

Testimonios

Hemos capacitado a más de 2,000 personas en IA — visita sitio anterior

Medios de pago

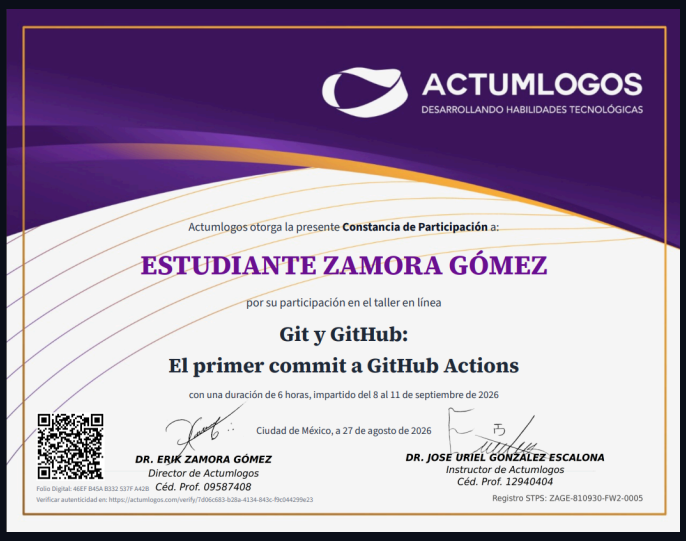
PayPal, transferencia SPEI, Tarjetas de crédito/débito, Link, Apple Pay, Google Pay.
Ver checkout

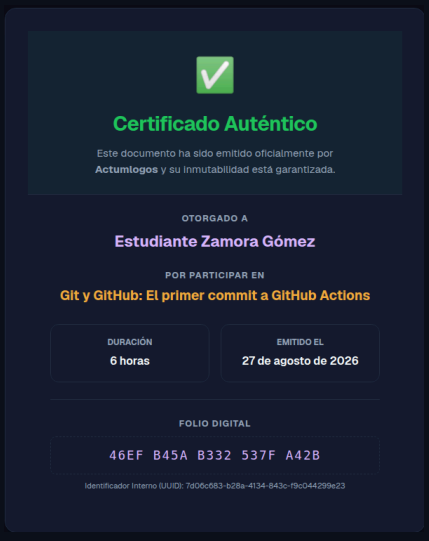
Constancia de participación con folio único, verificable en línea

Al terminar recibirás una constancia de participación firmada por el instructor y el director de Actumlogos — agente capacitador registrado ante la Secretaría del Trabajo y Previsión Social. Cada constancia incluye un folio único con el que cualquier persona puede verificar su autenticidad en línea, y podrás compartirla directamente en LinkedIn.

Registro STPS: ZAGE-810930-FW2-0005

Este es un ejemplo para ilustrar que esperar:





Conócenos
www.actumlogos.com