



Mapa de Instrumentos de Principales Plataformas de Deep Learning

Estilo	 PyTorch	 TensorFlow	Claves Laborales
Prototipado (explorar ideas, probar arquitecturas, investigación aplicada)	- Sintaxis muy cercana a Python, más intuitiva. - Control total del ciclo de entrenamiento (forward, backward). - Fácil debugging (usa print, pdb, integración con notebooks). - Ecosistema para investigación: TorchVision, TorchAudio, TorchGeometric.	- Keras API muy concisa (model.fit, model.predict). - Rápido de levantar modelos estándar. - Buenas herramientas de visualización (TensorBoard). - Integración con Google Colab y TPU.	- PyTorch es preferido en investigación y prototipos exploratorios. - TensorFlow/Keras facilita armar “demos rápidas” para mostrar resultados.
Hackathon (tiempo limitado, equipos diversos, resultados rápidos y claros)	- Requiere más código “manual” → puede ser más lento. - Torch Lightning acelera, pero hay curva de aprendizaje extra. - Gran comunidad open-source para copiar ejemplos.	- Keras = rapidez: en 10–15 líneas tienes un modelo funcional. - Muchas plantillas listas (ej. transfer learning en 3–4 líneas). - Integración con Hugging Face y TF Hub simplifica el acceso a modelos.	- TF/Keras es más productivo en plazos cortos . - PyTorch es competitivo si el equipo ya maneja Lightning o repos listos.
Producción (despliegue, escalamiento, mantenimiento)	- TorchScript, ONNX → exportar a C++, móviles, edge. - PyTorch Serve para inferencia. - Mejor en escenarios edge/embebidos. - Adoptado cada vez más por Meta, Microsoft, OpenAI.	- TensorFlow Serving, TFLite, TensorFlow.js → ecosistema muy sólido en despliegue. - Excelente para móviles (Android/iOS) y nube (Google Cloud). - Estándar en corporaciones grandes.	- TF sigue fuerte en infraestructura empresarial . - PyTorch gana terreno en edge + startups tecnológicas .