Tensorflow vs PyTorch

Tensorflow y PyTorch

- Entornos (frameworks) para aprendizaje automático/inteligencia computacional/deep learning.
- Proveen unidades básicas y complejas para trabajar con ANN: modelos, funciones de costo, optimizadores, métricas, etc.
- Análisis de prestaciones y diferencias de ambos frameworks, basado en diferentes indicadores (facilidad de Desarrollo, tiempos de ejecución, uso de memoria, etc.)
  - Características de PyTorch y TensorFlow
  - Rendimiento, precisión, entrenamiento y facilidad de uso
  - Principales diferencias entre PyTorch y TensorFlow

Tensorflow

- Framework open source para desarrollos de machine laerning y Deep learning de extremo a extremo, desarrollado por Google.
- Ejecuta en multiples plataformas (CPU, GPU, TPU, dispositivos móviles, etc.)
- Utilizado en la academia (universidades) y en empresas (Google, Uber, Microsoft, etc.)
- Tiene una version liviana (TensorFlow Lite) especialmente diseñada para machine learning en dispositivos edge computing. Ejecuta algoritmos livianos en dispositivos con recursos limitados (smartphones, microcontroladores y otros chips).

Tensorflow: ventajas

- Soporte y administración de bibliotecas: Respaldo de Google, actualizaciones frecuentes con nuevas funciones. Se utiliza en entornos de producción.
- Código abierto: disponible para una amplia gama de usuarios.
- Visualización de datos: proporciona la herramienta TensorBoard para visualizar datos y depuración sencilla de los nodos, reduce el esfuerzo de mirar todo el código y resuelve eficazmente la red neuronal.
- Compatibilidad con Keras: permite codificar funcionalidades en alto nivel y utilizar funciones específicas del sistema (pipelines, estimadores, etc.).
- Muy escalable: implementación flexible y distribuida.
- Compatibilidad: lenguajes C++, JavaScript, Python, C #, Ruby y Swift.
- Soporte de arquitectura: aceleración de hardware en paralelo en GPU, CPU y arquitectura propia TPU. Al trabajar con tensores, TPU es más eficiente que GPU y la CPU para el entrenamiento y ejecución de modelos.

Tensorflow: desventajas

- Eficiencia: en menos eficiente (en velocidad de cálculo) que otros frameworks. La eficiencia puede limitar su aplicabilidad.
- Dependencia: Los códigos debe ejecutarse utilizando cualquier plataforma de soporte, lo que aumenta la dependencia para la ejecución.
- Soporte de GPU: TensorFlow solo tiene soporte de NVIDIA para GPU y Python para la programación de GPU.

Pytorch

- Enfocado en eficiencia y usabilidad: se integra eficientemente y consistentemente con multiples bibliotecas y códigos desarrollados en Python. Simple de desarrollar y debuggear. Provee soporte para aceleradores de hardware y GPUs.
- Desarrollado en C++, tiene mucho menor overhear que otros frameworks.
- Diseñado para reducir los tiempos de diseño, entrenamiento y validación del ciclo de desarrollo de ANNs para la resolución de problemas concretos. Es muy popular en entornos académicos y científicos.
- Muchas bibliotecas y software de deep learning están diseñadas sobre PyTorch (e.g., Autopilot de Tesla y Pyro de Uber).

Pytorch: ventajas

- Basado en Python: directamente integrable con código y bibliotecas desarrolladas en Python.
- Simple de aprender: sintaxis similar a la de los lenguajes de programación. Curva de aprendizaje muy rápida.
- Debugging: permite debug con muchas herramientas de Python (pdb, ipdb, etc.)
- Grafos de cómputo dinámicos: permiten cambiar el comportamiento de la ANN en tiempo de ejecución. Mejora la optimización y los resultados (frente a los frameworks que consideran las ANN como objetos estáticos).
- Paralelismo de datos: distribución sencilla en mútiples cores de CPU o GPU.
- Comunidad muy active y documentación abundante y bien organizada.

Pytorch: desventajas

- Poco utilizado en entornos de producción: no es tan popular como en la academia.
- Interfaces de visualización y supervisión limitadas: es necesario utilizar herramientas de visualización de datos de Python o conectarse externamente a TensorBoard.
- No es una herramienta de desarrollo de extremo a extremo: el desarrollo de aplicaciones reales requiere convertir código PyTorch para implementar aplicaciones en servidores, estaciones de trabajo y dispositivos móviles.

Pytorch vs Tensorflow: performance

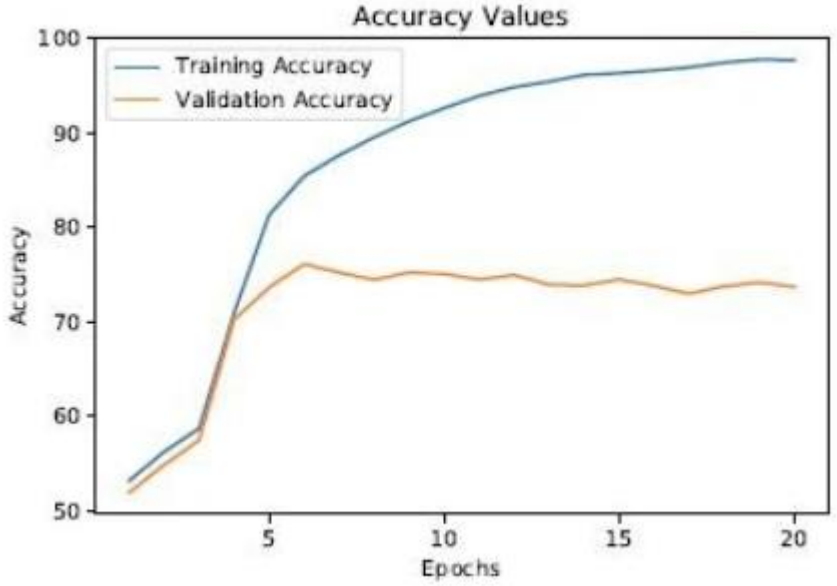
- Velocidad de procesamiento (muestras de datos procesadas por segundo) para diferentes modelos de ANN

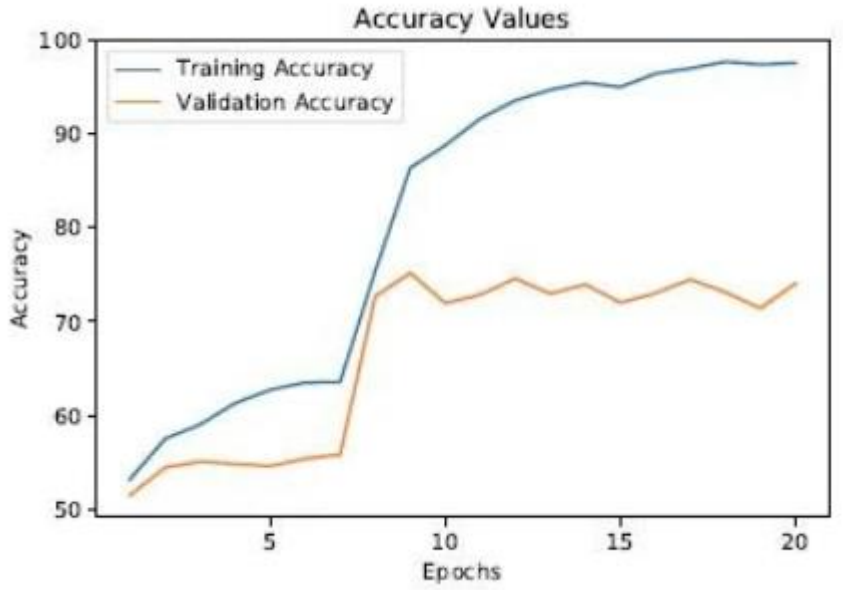
Framework	AlexNet	VGG-19	ResNet-50	MobileNet	GNMTv2	NCF
TensorFlow	1422 ± 27	66 ± 2	200 ± 1	216 ± 15	9631 ± 1.3%	4.8e6 ± 2.9%
PyTorch	1547 ± 316	119 ± 1	212 ± 2	463 ± 17	15512 ± 4.8%	5.4e6 ± 3.4%

- En general, PyTorch es más eficiente que TensorFlow

Pytorch vs Tensorflow: precisión

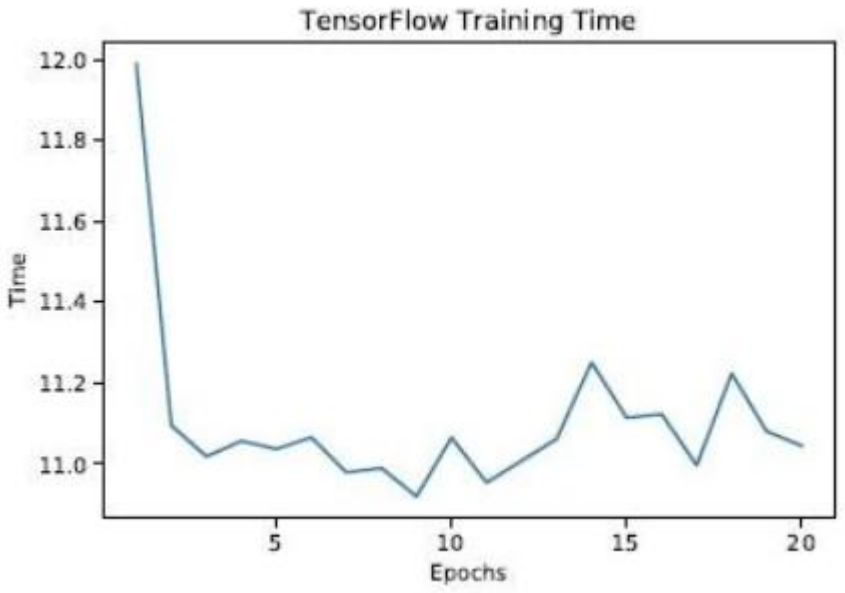
- Resultados similares para la precisión de entrenamiento y de validación.
- Patrones de aprendizaje muy precisos.
- Validación: resultados muy razoanbles, ambos frameworks tienen una media de precision del 78% en 20 épocas.

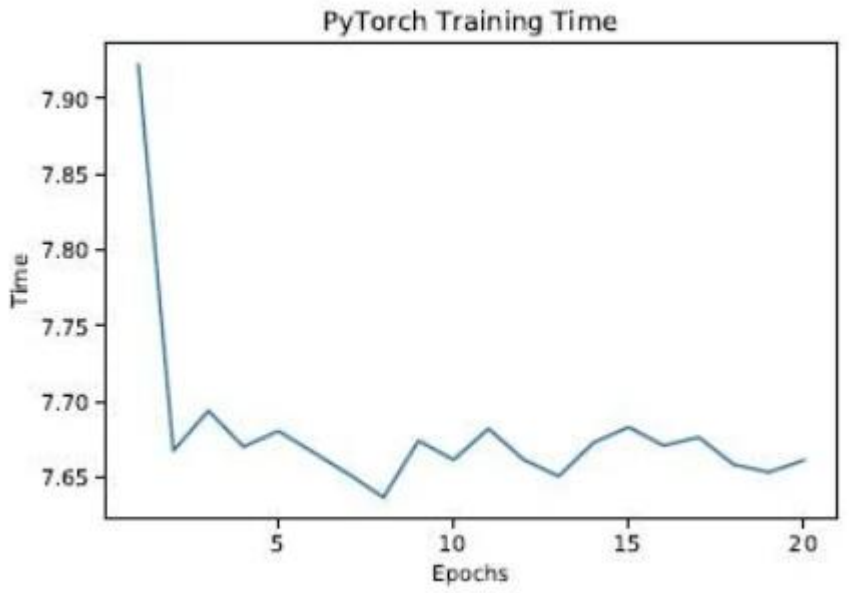




Pytorch vs Tensorflow: tiempo de entrenamiento

- En general, los tiempos de entrenamiento de TensorFlow son significativamente mayores a los que require el mismo modelo desarrollado en PyTorch.
- Ejemplo: promedio de 11.19 segundos para TensorFlow vs. 7.67 para PyTorch (deep ANN en Google Colab).





Pytorch vs Tensorflow: uso de memoria

- Uso de memoria RAM durante el entrenamiento: Tensorflow require significativamente menos memoria que PyTorch.
  - En el ejemplo considerado: Tensorflow utiliza 1.7 GB RAM vs PyTorch 3.5 GB RAM, en promedio.
- Sin embargo, ambos modelos tienen mayores demandas de RAM durante la etapa de carga de los datos.
  - En el ejemplo considerado: 4.8 GB para TensorFlow vs. 5 GB para PyTorch.

- Los requisitos de memoria pueden limitar la utilización de ambos frameworks, especialmente al utilizar GPU

Pytorch vs Tensorflow: facilidad de uso

- El modelo de programación orientada a objetos de PyTorch en general permite reducer los tiempos de implementación de los modelos.
- La especificación de los mecanismos de manejo de datos es más intuitive en PyTorch que en TensorFlow.
- TensorFlow tiene una curva de aprendizaje ligeramente steeper learning curve due to the low-level implementations of the neural network structure.
- El modelo de bajo nivel de Tensorflow permite desarrollos de ANN más personalizados con características y prestaciones más específicas.
- TensorFlow prove la biblioteca de alto nivel Keras, que simplifica su uso y permite una presentación más intuitive de los conceptos básicos de ANN [por ejemplo, para enseñanza].

Pytorch vs Tensorflow: resumen

- Bibliotecas de amplio uso, con comunidades amplias.
- Pytorch con mayor diffusion en la acedemia, Tensorflow con mayor aplicabilidad industrial y comercial
- Ambos frameworks tienen niveles similares de precision.
- Los tiempos de entrenamiento de TensorFlow son mayores que los de PyTorch, pero Tensorflow tiene menos requisites de memoria RAM.
- PyTorch permite diseños y prototipos rápidos, pero TensorFlow permite incorporar características específicas en los modelos de ANN.

Pytorch vs Tensorflow: resumen

- TensorFlow trata las ANN como objetos estáticos, para cambiar el comportamiento del modelo es necesario reiniciar. PyTorch permite modificar las ANN ‘on the fly’ en tiempo de ejecución [puede ser útil para la optimización del modelo].
- Tensorflow require una herramienta externa para debug, que permita examiner los cálculos de cada capa/neurona en cada paso. PyTorch permite debuggear con cualquiera de las herramientas disponbiles de Python.
- PyTorch and TensorFlow proven mecanismos para acelerar el desarrollo de modelos y reducir códigos redundantes/repetitivos. PyTorch en más ‘pythonico’ (basado en orientación a objetos), mientras que TensorFlow prove más opciones predefinidas y proporciona una mayor flexibilidad para incorporar características.