Clase 166 — TF Serving + gRPC (+ ONNX, TensorRT, vLLM/TGI)

Parte: 2 — Deep Learning · Fuente: Géron, cap. 19 § Deploying a Model to a Service + docs ONNX, TensorRT, vLLM, TGI. Duración estimada: 100 min.

Clase 166 — TF Serving + gRPC (+ ONNX, TensorRT, vLLM/TGI)

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 19 § Deploying a Model to a Service + docs ONNX, TensorRT, vLLM, TGI. Duración estimada: 100 min.*

Objetivo

Servir un modelo entrenado a producción. Aprender TF Serving (oficial de TensorFlow, gRPC/REST, batching) y las alternativas modernas multi-framework: ONNX + ONNX Runtime (portable a cualquier framework / runtime), TensorRT (NVIDIA, máxima velocidad en GPU NVIDIA), y para LLMs: vLLM y TGI (Text Generation Inference) — específicos del caso autoregresivo con continuous batching.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Exportar un modelo Keras a SavedModel: model.save('servable/1/', save_format='tf').
- Levantar TF Serving con Docker: docker run -p 8501:8501 -v $PWD/servable:/models/m tensorflow/serving --model_name=m.
- Hacer requests REST/gRPC.
- Exportar a ONNX con tf2onnx / torch.onnx.export, servir con ONNX Runtime.
- Conocer cuándo usar TensorRT (latencia mínima en GPU NVIDIA) o vLLM/TGI (LLMs).

Temas

- SavedModel format (TF nativo).
- TF Serving: configuración, versioning, model warm-up, batching.
- gRPC vs REST: gRPC más rápido (binary protobuf); REST más simple.
- Complemento moderno: ONNX/ONNX Runtime, TensorRT, vLLM/TGI.

Versión profundizada — 2026

El tema moderno que antes vivía como complemento dentro de esta clase ahora tiene su(s) clase(s) propia(s) con patrón completo, ejercicios y homework:

- Clase 139a — ONNX y ONNX Runtime: portabilidad e inference optimizada

Definiciones y características

- SavedModel: formato TF para modelos servibles (variables + graph + signature).
- Signature: contrato del modelo (predict con inputs/outputs nombrados).
- TF Serving: server escrito en C++, batching automático, versioning.
- gRPC: protocolo binario sobre HTTP/2. Más rápido que REST.
- ONNX: formato intermedio standard.
- TensorRT: optimizador NVIDIA específico.
- Triton: servidor multi-framework.

Dataset / recursos

- Un modelo ya entrenado de las clases previas (Fashion-MNIST MLP).
- Librerías: tensorflow, tensorflow-serving-api, tf2onnx, onnxruntime, opcional tensorrt.

Ejercicios

1. Export SavedModel: model.export('servable/1/') (Keras 3). Inspeccionar la estructura assets, variables, saved_model.pb.
2. TF Serving con Docker: levantar el container con el modelo montado, hacer una request REST a localhost:8501/v1/models/m:predict.
3. ONNX export: convertir el TF model a ONNX con tf2onnx. Cargar con ONNX Runtime y verificar misma predicción.
4. vLLM: levantar vllm con mistralai/Mistral-7B-Instruct. Cliente OpenAI-style. Medir tokens/sec.
5. Latencia comparada: TF Serving REST vs gRPC vs ONNX Runtime CPU vs TensorRT GPU sobre el mismo modelo. Reportar latencia P50 y P99.

Homework verificable

Servir un modelo de Fashion-MNIST con TF Serving y comparar con ONNX:

1. model.export('servable/1/').
2. Docker run TF Serving.
3. Convert TF → ONNX.
4. Inference con ambos sobre los mismos 100 inputs; verificar outputs idénticos (±1e-5).
5. Comparar latencia.

Criterio de aceptación: predicciones de TF Serving y ONNX Runtime coinciden; latencia reportada para ambos.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
TF Serving no encuentra modelo	Estructura de carpetas mal. Fix: model_nam
Outputs cambian entre TF y ONNX	Conversión incompleta (capas custom). Fix:
TensorRT engine no portable entre GPUs	Es específico al hardware. Fix: rebuild en
vLLM no carga modelo cuantizado	Algunas quantizations no soportadas. Fix:
Latencia P99 alta	Sin warm-up. Fix: warmup_count=10 en TF Se

Preguntas frecuentes

¿REST o gRPC?

gRPC para producción serio (más rápido). REST para debugging y compatibilidad universal.

¿ONNX es siempre más rápido que TF Serving?

No siempre. ONNX brilla en CPU + casos específicos. TF Serving está más optimizado para batching en GPU.

¿TensorRT necesario?

Solo si latencia es crítica. Para 99 % de los casos, ONNX Runtime + GPU es suficiente.

¿Cómo manejo versioning de modelos?

TF Serving por default sirve el "latest" entre las carpetas numeradas. Para A/B: serve varias versiones, configurar weights en model_config_list.

¿LLM en TF Serving o vLLM?

vLLM siempre para LLMs. TF Serving es ineficiente para autoregresivo.

Referencias

- Géron, cap. 19 — Deploying a Model to a Service.
- TF Serving docs.
- ONNX docs, ONNX Runtime.
- Triton Inference Server.
- vLLM docs, TGI docs.

Siguiente clase

Clase 167 — ONNX y ONNX Runtime: portabilidad e inference optimizada

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb