TFM Co-diseño IA+FPGA: Tensor-Train (TT) y kernel de contracción TT en FPGA

ONE_PAGER — Co-diseño IA+FPGA para capas lineales en Tensor-Train (TT)

Mariano Millañanco Fernández
Ciudad Real, España
Universidad de Sevilla — Máster en Microelectrónica
UTAMED — Máster en Inteligencia Artificial
github.com/tangodelta217/TFM_FPGA_TT_LLM_INDRA
mariano.millananco@gmail.com

2026

Resumen

Se desarrolla un TFM de co-dise?o IA+FPGA para acelerar capas lineales mediante compresi?n Tensor-Train (TT) y un kernel de contracci?n TT en FPGA. El estado actual incluye demo CPU reproducible con TT-SVD, m?tricas de compresi?n y error, y un skeleton HLS/RTL con interfaces AXI y mapa de registros. La evaluaci?n usa golden model en Python y define KPIs HW como objetivos (TBD).

Abstract

This TFM targets IA+FPGA co-design to accelerate linear layers through Tensor-Train (TT) compression and an FPGA TT contraction kernel. Current status includes a reproducible CPU demo with TT-SVD, compression and error metrics, and an HLS/RTL skeleton with AXI interfaces and a register map. Evaluation uses a Python golden model and defines hardware KPIs as targets (TBD) for the FPGA phase.

Keywords: Tensor-Train; TT-SVD; FPGA; AXI; capas lineales

Resumen ejecutivo (entregables medibles)

- Demo CPU reproducible (TT-SVD y comparaci?n denso vs TT) con m?tricas en `docs/assets/kpi_table.md` y `docs/assets/demo_output.txt`.
- Skeleton HW con AXI, mapa de registros y diagrama de bloques; V&V con golden model y tests; KPIs HW como objetivos (TBD).

1. Introducción

Las capas lineales en inferencia en borde son bandwidth-bound y el tráfico a DDR domina la latencia, afectando SWaP / determinismo / soberanía. El TFM aborda la reducción de tráfico mediante compresión Tensor-Train (TT) y un kernel de contracción TT en FPGA con streaming y control AXI.

2. Metodología y arquitectura

La operación base es $y = Wx$, donde W se aproxima en Tensor-Train (TT) con cores $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k^{out} \times n_k^{in} \times r_k}$. La descomposición se obtiene con TT-SVD y se valida en CPU con la referencia NumPy.

El kernel HW se diseña con AXI4-Lite, AXI-Stream/DMA y buffers locales. El layout de cores y el mapa de registros están en `docs/assets/register_map.md`.

3. Evidencia, plan y limitaciones

Evidencia actual (CPU): compresi?n 7.53x–341.33x y error relativo L2 8.85e-01 a ~ 3e-15; tiempos en la tabla KPI (mediana de 40 repeticiones). Ver `docs/assets/kpi_table.md`, `docs/assets/bench_tradeoff.png` y `docs/assets/demo_output.txt`.

Plan de evaluaci?n: comparaci?n TT vs denso, tests autom?ticos y co-simulaci?n HW/SW con contadores de cycles/stalls. Resultados esperados: reducci?n del tr?fico a DDR y mayor estabilidad temporal; KPIs HW de latencia, throughput, recursos y determinismo como objetivos (TBD). Limitaciones actuales: sin bitstream ni implementaci?n HLS/RTL funcional y m?tricas solo en host CPU.

[1] I. V. Oseledets, "Tensor-Train Decomposition", *SIAM Journal on Scientific Computing*, vol. 33, n.º 5, págs. 2295-2317, DOI: 10.1137/090752286 dirección: https://doi.org/10.1137/090752286

[2] I. V. Oseledets y E. E. Tyrtyshnikov, "Breaking the Curse of Dimensionality, or How to Use SVD in Many Dimensions", *SIAM Journal on Scientific Computing*, vol. 31, n.º 5, págs. 3744-3759, DOI: 10.1137/090748330 dirección: https://doi.org/10.1137/090748330

[3] T. G. Kolda y B. W. Bader, "Tensor Decompositions and Applications", *SIAM Review*, vol. 51, n.º 3, págs. 455-500, DOI: 10.1137/07070111X dirección: https://doi.org/10.1137/07070111X

[4] A. Novikov, D. Podoprikhin, A. Osokin y D. Vetrov, "Tensorizing Neural Networks", *arXiv preprint*, dirección: https://arxiv.org/abs/1509.06569

[5] T. Garipov, D. Podoprikhin, A. Novikov y D. Vetrov, "Ultimate Tensorization: Compressing Convolutional and FC Layers Alike", *arXiv preprint*, dirección: https://arxiv.org/abs/1611.03214