Implementació d'un compressor basat en xarxes neuronals per a ús en satèl·lit

Informe de Progrés II: Optimització i Avaluació Experimental

Cristhian Omar Añez López
1635157

Desembre 2025

Índex

1 Seguiment de la Planificació i Ajustaments

Aquest segon informe cobreix l'execució de la Fase 3 (Integració i Optimització) i Fase 4 (Avaluació Experimental) de la planificació original.

1.1 Estat del Projecte

S'han assolit els objectius principals d'aquestes fases, aconseguint una implementació en C funcional, optimitzada i validada sobre la plataforma objectiu (Raspberry Pi 3B+). No obstant això, s'han realitzat ajustaments estratègics respecte a la proposta inicial.

1.2 Ajustaments i Justificació

L'ajustament més significatiu ha estat la **decisió de no integrar el codificador aritmètic (Montsec)** en l'executable final en C.

- **Planificació Inicial:** Es preveia fusionar la xarxa neuronal amb el codificador per entropia per generar un *bitstream* comprimit real.
- **Realitat (Ajustament):** El desenvolupament s'ha centrat a perfeccionar la inferència de la xarxa neuronal.
- **Justificació:**
  1. **Complexitat Tècnica i Prioritització:** Durant la Fase 3 es va detectar un *bug* crític en la quantització del paràmetre λ que degradava severament la qualitat (9.67 dB PSNR). Es va prioritzar la correcció d'aquest error matemàtic i l'optimització de la gestió de memòria RAM (crítica en la Raspberry Pi de 1GB) per davant de la integració del codificador entròpic, que és un mòdul extern.
  2. **Focus en la Comparativa:** L'objectiu del TFG és comparar l'eficiència de l'enfocament "après" (Deep Learning) vs. l'estàndard (CCSDS). Atès que el cost computacional dominant de SORTENY resideix en les convolucions i no en la codificació per entropia, la comparativa de temps i memòria continua sent vàlida analitzant l'etapa d'inferència.

2 Metodologia i Canvis (Fases 3 i 4)

2.1 Fase 3: Optimització per a Sistemes Encastats

La metodologia per a la Fase 3 s'ha basat en l'optimització agressiva de recursos per fer viable l'execució en una Raspberry Pi 3B+ (ARM Cortex-A53, 1GB RAM).

2.1.1 Gestió de Memòria: Estratègia "Ping-Pong"

L'implementació de referència en Python (TensorFlow) utilitzava fins a 800 MB de RAM, provocant fallades (*OOM Kills*) o ús intensiu de *swap* en la Raspberry Pi. Per resoldre-ho, s'ha implementat en C una estratègia de reutilització de buffers (Ping-Pong):

- En lloc de reservar memòria per a tots els tensors intermedis de la xarxa (Conv0, GDN0, Conv1...), es reserven només dos buffers de treball (`scratch_a` i `scratch_b`).
- Les capes s'executen seqüencialment alternant escriptura i lectura entre aquests dos buffers, reduint el consum de memòria en un factor de 9.5x.

2.1.2 Correcció de Bugs Crítics

Es va identificar que la implementació inicial produïa una qualitat de reconstrucció inacceptable (PSNR ≈ 9 dB). Mitjançant un anàlisi exhaustiu del codi comparada amb Python, es van descobrir tres errors:

- **Error 1 (GDN):** La fórmula de l'activació GDN era incorrecta, s'ha corregit.
- **Error 2 (Padding):** Les convolucions SignalConv2D utilitzaven padding variable en lloc de fix: pad = $\lfloor$kernel_size/2$\rfloor$.
- **Error 3 (Normalització):** Faltava dividir l'entrada per 65535 abans de la primera convolució.
- **Resultat:** Després de les correccions, s'ha assolit un PSNR de 76.73 dB, idèntic a Python.

2.2 Fase 4: Avaluació Experimental

La metodologia d'avaluació s'ha dissenyat per ser justa i reproduïble sobre maquinari real.

- **Plataforma:** Raspberry Pi 3B+ amb Raspberry Pi OS (64-bit).
- **Compilació:** Ús de *flags* específics per a Cortex-A53: `-O3 -mcpu=cortex-a53 -mtune=cortex-a53 -funsafe-math-optimizations`.
- **Comparativa a Tres Bandes:**
  1. **SORTENY C (Proposta):** Executable natiu optimitzat amb OpenMP.
  2. **SORTENY Python (Referència):** Script original utilitzant TensorFlow.
  3. **CCSDS-122 (Estàndard):** Implementació de referència MHDC en C++/Java.
- **Mètriques:** Temps d'execució (*wall-clock*), Pic de Memòria RAM (RSS) i Qualitat de reconstrucció (PSNR).

3 Exposició dels Resultats

A continuació es presenten els resultats obtinguts processant una imatge Sentinel-2 (8 bandes, 512×512, uint16).

3.1 Validació de Qualitat i Paritat

Després de la correcció d'errors, la implementació en C ha assolit una paritat numèrica excel·lent amb la referència en Python.

Mètrica	SORTENY C	SORTENY Python
PSNR (dB)	**76.73**	76.73
Latents Idèntics	99.99%	
Correlació de Latents	0.99999998	

Taula 1: Comparativa de qualitat de reconstrucció. Ambdues implementacions produeixen resultats pràcticament idèntics.

3.2 Rendiment Computacional (Raspberry Pi 3B+)

La següent taula resumeix la mitjana de 3 execucions per a cada compressor.

Compressor	Temps (wall)	Throughput	Speedup	RAM (MB)	Ús CPU
SORTENY Python	6:55	5,053 px/s	1.0x	824	25%
SORTENY C	**5:06**	**6,853 px/s**	**1.36x**	**88**	**341%**
CCSDS-122	13 s	161,319 px/s	31.9x	6.5	98%

Taula 2: Resultats de rendiment en Raspberry Pi 3B+. Temps mesurat com a *wall-clock time*. Throughput calculat com $(8 \times 512 \times 512)/t$.

- **Temps:** La implementació en C és **1.36 vegades més ràpida** que la versió en Python/TensorFlow. Python només utilitza el 25% de CPU perquè TensorFlow no paral·lelitza eficientment en ARM, mentre que C aprofita els 4 nuclis (341% CPU).
- **Memòria:** Aquest és el resultat més crític. La versió en C redueix el consum de memòria en un factor de **9.4x**, baixant de 824 MB (perillosament a prop del límit d'1 GB de la placa) a uns confortables 88 MB. Això valida la viabilitat de la solució proposada per a sistemes embarcats.

4 Conclusions Provisionals

1. **Viabilitat Tècnica:** S'ha demostrat que és possible executar xarxes neuronals complexes de compressió (SORTENY) en maquinari modest (Raspberry Pi 3) mitjançant una implementació nativa en C, eliminant la dependència de llibreries pesades com TensorFlow.
2. **Eficiència de Memòria:** L'optimització de gestió de memòria ha estat un èxit rotund, convertint una aplicació que saturava el sistema (Python, 824 MB) en una de lleugera i viable (C, 88 MB).
3. **Cost de la Complexitat:** Tot i l'optimització, el cost computacional de les xarxes neuronals (5 minuts) continua sent molt superior al dels estàndards clàssics com CCSDS-122 (13 segons). Això planteja un compromís clar entre la flexibilitat/potencial de la IA i la velocitat dels algorismes fixos.
4. **Qualitat:** La implementació en C produeix resultats idèntics a Python (PSNR 76.73 dB, correlació de latents 0.99999998), validant matemàticament la correcció de la reimplementació.

5 Informació Addicional

5.1 Fonts d'Informació Consultades

Durant aquesta etapa s'han consultat principalment:

- Documentació tècnica del processador ARM Cortex-A53 (per a l'optimització de *flags* de compilació).
- Codi font de l'estàndard MHDC (CCSDS 122.0-B-1) per a la compilació i execució de la comparativa.
- Documentació de TensorFlow Lite i TensorFlow Compression per a l'anàlisi de discrepàncies numèriques.

5.2 Ús de la IA Generativa

D'acord amb la normativa establerta, es declara l'ús d'eines d'IA generativa com a suport en el desenvolupament:

- **Suport al Codi:** Assistència en la refactorització del codi C per a l'optimització de bucles i la implementació de la lògica de "ping-pong"de memòria.

- **Depuració:** Ajuda en l'anàlisi de logs d'errors de compilació (especialment amb el codi C++ antic del CCSDS) i en la identificació del *bug* de quantització de λ.
- **Redacció:** Suport en l'estructuració i redacció preliminar d'informes tècnics en format LaTeX (exemples, estructurar taules i corregir format de mots especials).