Grupa 332AB

Quantization of AI Models on FPGA

Corha Maria
Bucevschi Vlad-Eusebiu
Popa Alexandru Stefan

1. Abstract

In ultimii ani, retelele neuronale profunde au revolutionat domenii precum viziunea artificiala, procesarea limbajului natural si sistemele autonome. Totusi, aceste modele sunt in mod uzual antrenate in aritmetica pe 32 de biti in virgula mobila (FP32), ceea ce implica un consum ridicat de memorie, timp de executie crescut si un necesar mare de energie. Pentru implementari pe dispozitive cu resurse limitate, precum FPGA-urile (Field Programmable Gate Arrays), devine esentiala reducerea preciziei numerice, proces numit cuantizare.

Acest raport analizeaza tehnicile moderne de cuantizare a retelelor neuronale (INT8, INT4, binar), precum si compromisurile acestora, cu accent pe implementarea eficientei pe FPGA folosind operatii pe punct fix si arhitecturi optimizate hardware. Sunt sintetizate peste 10 lucrari stiintifice relevante si sunt extrase principii solide pentru o implementare hardware performanta. In final, este propusa o arhitectura FPGA configurabila pentru cuantizarea modelelor AI, impreuna cu o strategie completa de proiectare, validare si comparare a performantelor.

2. Introducere

Odata cu cresterea complexitatii modelelor de invatare profund, cerintele hardware devin tot mai mari. Platformele GPU si TPU sunt optimizate pentru antrenare si inferenta rapida, insa nu sunt intotdeauna potrivite pentru dispozitive embedded sau edge computing, unde resursele sunt limitate. FPGA-urile ofera flexibilitate, paralelism masiv si consum redus, dar pentru a obtine performante competitive este necesara reducerea preciziei numerice a modelului.

Cuantizarea reduce dimensiunea parametrilor si a activarilor prin maparea valorilor FP32 in reprezentari pe mai putini biti: INT8, INT4 sau chiar 1 bit (retele binare). Astfel, costul operatiilor aritmetice scade, latenta inferentei se reduce, iar consumul energetic este mai mic.

3. Formularea problemei

Problema investigata este urmatoarea:

Cum poate fi implementata eficient cuantizarea modelelor AI pe FPGA, folosind tehnici moderne (INT8, INT4, binar), astfel incat sa se obtina un compromis optim intre acuratate si performanta hardware?

Aceasta implica:

- analizarea tehnicilor de cuantizare (post-training si quantization-aware)
- evaluarea preciziei pe diferite latimi de biti
- proiectarea arhitecturii hardware care foloseste aritmetica pe punct fix
- suportarea operatiilor uzuale: convolutie, inmultire matrice-vector, batch normalization
- implementarea unei solutii modulare, eficiente si scalabile

4. Background teoretic

4.1 Ce este cuantizarea?

Cuantizarea transforma valorile parametrilor si activarilor din reprezentari FP32 intr-o precizie redusa, de obicei integer cu punct fix.

Exemple:

- FP32 → INT8 (256 niveluri)
- FP32 → INT4 (16 niveluri)
- FP32 → INT1 (retele binare)

4.2 Avantajele cuantizarii

- Reducerea memoriei (de 4-32 ori)
- Viteza de inferenta crescuta
- Consum energetic redus
- Dimensiune hardware mai mica
- Potrivita pentru FPGA-uri

4.3 Tipuri de cuantizare

1. **Cuantizare post-antrenare (PTQ)**
    - nu necesita retrain
    - usor de aplicat
    - uneori scade acuratatea
2. **Cuantizare constienta de antrenare (QAT)**
    - modelul este antrenat avand precizie redusa
    - acuratate superioara
    - necesita resurse de antrenare
3. **Cuantizare uniforma vs. neuniforma**
4. **Cuantizare pe canal vs. pe strat**
5. **Mixed-precision**
    - fiecare strat poate avea propria precizie
    - optim pentru FPGA-uri

5. Analiza lucrarilor stiintifice (State of the Art)

Mai jos sunt analizate articolele relevante:

5.1 Trainable Fixed-Point Quantization for Deep Learning Acceleration on FPGAs (Dai et al., 2024)

- Propune o metoda de cuantizare antrenabila, unde limitele intervalului de cuantizare sunt optimizate in timpul antrenarii.
- Beneficii:
    - reduce erorile de aproximare
    - se potriveste perfect pentru hardware pe punct fix
- Relevanta:
    - permite obtinerea unei arhitecturi FPGA cu precizie reglabila

5.2 Post-training quantization for efficient FPGA-based neural network acceleration (Salah et al., 2025)

- O abordare PTQ optimizata pentru FPGA, fara retrain.
- Foloseste calibrari de activare si minimizarea erorii de cuantizare.
- Contributie:
    - demonstreaza ca PTQ poate obtine acuratati bune pe hardware real

5.3 Quantized Convolutional Neural Networks: A Hardware Perspective (Zhang et al., 2025)

- Analizeaza impactul cuantizarii asupra implementarii hardware.
- Discutie profunda despre:
    - consum energetic
    - numar de resurse FPGA
    - latimea magistralelor
- Important pentru proiect:
    - arata clar compromisurile bit-width vs. cost hardware

5.4 Quantization-Aware NN Layers With High-throughput FPGA Implementation (Pistellato et al., 2023)

- Propune design-uri FPGA optimizate pentru straturi QAT.
- Include arhitecturi pipeline pentru conv2D cuantizate.
- Demonstreaza throughput foarte mare.

5.5 On-Chip Hardware-Aware Quantization for Mixed Precision Neural Networks (Huang et al., 2023)

- Introduce o metoda hardware-aware, unde alegerea preciziei se face automat in functie de limitari hardware.
- Relevanta:
    - permite design-uri FPGA foarte eficiente pentru fiecare strat in parte.

6. Tehnici de cuantizare detaliate

6.1 Cuantizare INT8

- Cel mai folosit standard
- Permite accelerare mare
- Acuratate aproape identica cu FP32 pe majoritatea modelelor

6.2 Cuantizare INT4

- Reduce memoria la jumatate fata de INT8
- Consum energetic si mai redus
- Acuratate scade, dar acceptabila pentru edge AI

6.3 Cuantizare binara (1-bit, XNOR-Networks)

- Cost minim
- Operatii XNOR + popcount
- Extrem de eficiente pentru FPGA
- Acuratate buna doar pentru anumite sarcini

7. Trade-offs intre precizie si performanta

Tehnica	Avantaje	Dezavantaje	Interpretare
INT8	Acuratate ridicata, compatibilitate buna	Resurse moderate	Standard industrial
INT4	Cost redus, throughput mare	Pierderi vizibile de acuratate	Optim pentru edge devices
Binar	Resurse minime	Acuratate mai slaba	Bun pentru clasificari simple

8. Consideratii hardware pentru FPGA

Implementarea retelelor neuronale cuantizate pe FPGA necesita o intelegere clara a caracteristicilor arhitecturale ale acestor dispozitive. Fiecare decizie de proiectare depinde

direct de modul in care FPGA-ul gestioneaza memoria, operatiile aritmetice, paralelizarea si latenta pipeline-urilor.

Mai jos sunt detaliate aspectele esentiale pentru implementarea eficienta a unui accelerator AI cuantizat.

8.1 Aritmetica pe punct fix si avantajele ei hardware

In FPGA-uri, operatiile pe punct fix sunt mult mai eficiente decat cele floating-point, deoarece:

- necesita mai putine resurse logice (LUT-uri)
- pot utiliza DSP-urile interne optimizate pentru operatii integer
- ofera un timp de propagare mai scurt
- reduc consumul energetic

Operatiile integer de forma:

- multiplicare int8 × int8
- acumulare pe 16 sau 32 biti
- saturare

pot fi mapate direct pe DSP48E1/DSP48E2 (in Xilinx), permitand o implementare paralela a sute sau mii de operatii MAC per ciclu.

Aritmetica pe punct fix reduce semnificativ costul:

- un MAC floating-point FP32 necesita ~700 LUT-uri
- un MAC integer 8-bit necesita sub 20 LUT-uri + 1 DSP

Acesta este motivul fundamental pentru care cuantizarea este esentiala in FPGA.

8.2 Organizarea memoriei si reducerea latimii magistralelor

Cuantizarea reduce latimea datelor:

- FP32 = 32 biti
- INT8 = 8 biti
- INT4 = 4 biti
- Binar = 1 bit

Reducerea latimii are efecte majore:

1. Scade latimea magistralelor interne
    - acces mai rapid la BRAM/URAM
    - mai putina congestie pe magistrale
2. Creste cantitatea de date ce poate fi stocata
    - memoria BRAM este fixa, deci putem depozita de 4–32 ori mai multi parametri
3. Permite paralelizarea pe scara mare
    - latimea fisica a BRAM permite citirea mai multor elemente per ciclu

8.3 Paralelizarea calculelor (Spatial Parallelism)

FPGAs sunt ideale pentru paralelizare spatiala, adica replicarea fizica a unitatilor de calcul.

Pentru retele cuantizate, putem implementa:

- paralelizare pe canale (channel-wise parallelism)
- paralelizare pe pixeli (pixel-wise parallelism)
- paralelizare pe kernel-uri (filter-level parallelism)

De exemplu, pentru o convolutie 3×3 cu 64 de canale, putem procesa 8, 16 sau chiar 32 canale in paralel, in functie de resurse.

8.4 Pipeline si reducerea latentei

Pipeline-ul permite executarea operatiilor in flux continuu:

- un strat conv2D poate fi impartit pe etape: citire, multiplicare, acumulare, activare
- fiecare etapa ruleaza in paralel pe date diferite

Cuantizarea usureaza pipeline-ul deoarece:

- operatiile integer sunt mai rapide
- consumul de resurse este mai mic
- sunt necesare mai putine cicluri pentru normalizare

8.5 Per-channel quantization si normalizare fuzionata

Cuantizarea per-canal permite alocarea unui factor de scalare diferit pentru fiecare canal. Avantaje:

- reduce eroarea de cuantizare
- imbunatateste semnificativ acuratatea
- se potriveste bine cu arhitectura FPGA, unde fiecare canal poate avea propriile unitati MAC

Batch Normalization poate fi fuzionat cu convolutia:

- multiplicarea scale + bias devine o operatie integer simpla
- reduce numarul de straturi hardware

9. Arhitectura propusa (pentru implementare)

Arhitectura propusa este modulara, configurabila si complet optimizata pentru operatii integer pe 8, 4 si 1 bit.

9.1 Prezentare de ansamblu

Arhitectura include urmatoarele module:

1. Unitatea de cuantizare (Quantizer Unit)
2. Blocurile de aritmetica integer (MAC Arrays)
3. Engine dedicat pentru Conv2D
4. Engine pentru straturi Dense (Fully Connected)
5. Normalizare si activare cuantizata
6. Unitati de memorie (BRAM Buffers si Weight Banks)
7. Scheduler si Control Unit
8. Unitate pentru precizie configurabila (Mixed Precision Controller)

9.2 Modulul de cuantizare

Realizeaza:

- generarea factorilor de scalare
- rotunjire si clipare

- transformare FP32 → INTx
- mapare la reprezentare simetrica cu zero-point

Pentru INT8:

INT8_value = round(FP32_value / scale)

Pentru INT4:

INT4_value = clip(round(value / scale), -8 .. 7)

Pentru Binar:

binary_value = sign(value)

9.3 Blocurile MAC integer (Multiply–Accumulate Arrays)

Sunt aranjate pe matrice 2D, cu suport pentru:

- int8 × int8
- int4 × int4
- XNOR-popcount pentru retele binare

Aceste blocuri reprezinta "motorul" acceleratorului.

9.4 Engine Conv2D

Implementat cu:

- linii de intarziere (line buffers) pentru streaming de pixeli
- ferestre glisante (sliding windows)
- paralelizare pe canale

Design-ul permite:

- o noua convolutie per ciclu (daca exista resurse)
- latenta pipeline minimizata
- reutilizarea intensiva a datelor, reducand accesul la memorie

9.5 Engine Dense (Fully Connected Layer)

Organizat ca matrice de MAC-uri cu acces paralel la greutati:

- citeste vectorul de intrare in stream

- inmulteste cu toate greutatile in paralel
- acumuleaza rezultatele pe 16/32 biti

9.6 Blocul de activare

Pentru cuantizare, activare ReLU devine triviala:

relu_q(x) = max(0, x)

Acest lucru ocupa aproape zero resurse hardware.

9.7 Memory Banks

Arhitectura foloseste:

- **BRAM** pentru activari
- **URAM** pentru greutati mari
- **Distributed RAM** pentru buffere mici

Memoria este optimizata pentru latime mica (4–8 biti pe element).

9.8 Scheduler si Control Unit

Coordoneaza:

- ordinea straturilor
- setarea preciziei
- sincronizarea bufferelor
- controlul pipeline-ului

Este esential pentru functionarea corecta si minimizarea latentei.

10. Metodologie de implementare

Implementarea cuantizarii pe FPGA necesita o abordare structurata. Mai jos este prezentat procesul complet, pas cu pas.

10.1 Faza 1 – Selectarea si pre-procesarea modelului

- se alege un model simplu pentru inceput (ex: CNN pentru clasificare)
- se converteste in format compatibil (ONNX, TensorFlow Lite)
- se elimina straturile floating-point inutile (ex. dropout)

10.2 Faza 2 – Aplicarea cuantizarii

PTQ (Post-Training Quantization)

1. colectarea statisticilor de activare
2. determinarea intervalelor min–max
3. maparea parametrilor la INT8 si INT4
4. calibrare

QAT (Quantization-Aware Training)

1. antrenare cu "fake quantization nodes"
2. simularea cuantizarii in timpul forward-pass
3. ajustarea parametrilor pentru a compensa erorile

10.3 Faza 3 – Exportul parametrilor cuantizati

Se genereaza tabele cu:

- greutati cuantizate (INT8, INT4, INT1)
- scale si zeropoint
- biasuri recalibrate

10.4 Faza 4 – Implementarea hardware

1. scrierea modulelor Verilog/VHDL pentru MAC integer
2. generarea sistemului de line buffers pentru convolutie
3. implementarea activarii cuantizate
4. implementarea controller-ului de precizie
5. mapare memorie pentru greutati si activari
6. integrarea modulelor intr-o arhitectura completa

10.5 Faza 5 – Simulare

- se foloseste ModelSim, QuestaSim sau XSIM
- se verifica functionalitatea fiecarui modul
- se testeaza corelatia cu modelul software

10.6 Faza 6 – Sintetizare si implementare pe FPGA

- se foloseste Vivado / Quartus
- se analizeaza:
    - utilizare resurse (LUT, FF, DSP, BRAM)
    - frecventa maxima
    - timpii de propagare
    - consum energetic

10.7 Faza 7 – Validare finala si comparatii

Se compara:

- acuratatea FP32 vs INT8 vs INT4
- throughput hardware
- consumul energetic
- resursele utilizate

11. Rezultate asteptate

Chiar inainte de implementare, pe baza rezultatelor raportate in literatura, se pot estima urmatoarele:

11.1 Acuratate

- **INT8**: pierdere medie de 0.5–1%
- **INT4**: pierdere intre 1–4%
- **Binar**: pierdere 10–20%, dar depinde de model

Pentru modele mici (edge AI):

- QAT poate recupera aproape toata acuratatea pierduta

11.2 Performanta hardware

Estimari:

- INT8 poate fi de **3–6 ori mai rapid** decat FP32
- INT4 poate fi de **6–12 ori mai rapid**

- Binar poate atinge **50–100× accelerare**

11.3 Consum de resurse FPGA

Cuantizarea reduce:

- consumul de BRAM de 4–32 ori
- consumul de DSP-uri cu 2–8 ori
- consumul energetic total cu 30–70%

11.4 Throughput

Pentru un accelerator cu paralelizare medie:

- INT8: 200–400 GOPS
- INT4: 400–800 GOPS
- Binar: >2000 GOPS

11.5 Scalabilitate si flexibilitate

Arhitectura propusa permite:

- schimbarea preciziei pe strat
- adaugarea de noi straturi cu configuratie modificabila
- extinderea retelelor pentru sarcini mai complexe

12. Concluzii

Cuantizarea este esentiala pentru deploy-ul retelelor neuronale pe FPGA. Prin reducerea preciziei, obtinem performante foarte bune si costuri hardware scazute, pastrand acuratatea la nivel acceptabil. Analiza literaturii confirma ca tehnicile moderne precum trainable quantization, hardware-aware quantization si mixed precision sunt cele mai eficiente pentru implementari FPGA. Arhitectura propusa in acest raport ofera o baza solida pentru dezvoltarea unui accelerator AI optimizat pentru edge AI pe FPGA.

13. Bibliografie

1. Dai, Dingyi, et al. "Trainable fixed-point quantization for deep learning acceleration on FPGAs." arXiv:2401.17544 (2024).
2. Salah, Oumayma Bel Haj, et al. "Post-training quantization for efficient FPGA-based neural network acceleration." Integration, 2025.

3. Zhang, Li, et al. "Quantized convolutional neural networks: a hardware perspective." Frontiers in Electronics, 2025.
4. Pistellato, Mara, et al. "Quantization-aware NN layers with high-throughput FPGA implementation." Sensors, 2023.
5. Huang, Wei, et al. "On-chip hardware-aware quantization for mixed precision neural networks." arXiv:2309.01945 (2023).
6. Courbariaux, M., et al. "BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1." NIPS, 2016.
7. Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." CVPR, 2018.
8. Rastegari, Mohammad, et al. "XNOR-Net: ImageNet classification using binary convolutional neural networks." ECCV, 2016.
9. Hubara, Itay, et al. "Quantized neural networks: Training neural networks with low precision weights and activations." arXiv, 2016.
10. Umuroglu, Yaman, et al. "FINN: A framework for fast, scalable quantized neural network inference on FPGAs." FPGA Conference, 2017.