Universitatea
Transilvania
din Brașov

ȘCOALA DOCTORALĂ INTERDISCIPLINARĂ

Facultatea de Inginerie Electrică și Știința Calculatoarelor

Ing. Corneliu ZAHARIA

Contribuții la Relația Inteligența Artificială-Hardware pentru Procesarea Imaginilor în Timp Real

Contributions to the Artificial Intelligence-Hardware Relationship in Real Time Image Processing

REZUMAT / ABSTRACT

Conducător științific

Prof.dr.ing. Florin SANDU

BRAȘOV, 2022

Universitatea Transilvania din Brașov

D-lui (D-nei) .....

COMPONENȚA

Comisiei de doctorat

Numită prin ordinul Rectorului Universității Transilvania din Braşov

Nr. ..... din .....

PREŞEDINTE:	Conf.dr.ing. Carmen GERIGAN - Universitatea "Transilvania" din Brașov
CONDUCĂTOR ŞTIINȚIFIC:	Prof.dr.ing. Florin SANDU – Universitatea "Transilvania" din Brașov
REFERENȚI:	Prof.dr.ing. Radu Adrian VASIU – Universitatea Politehnica Timișoara
	Prof.dr.ing. Romulus Mircea TEREBEȘ – Universitatea Tehnică din Cluj-Napoca
	Prof.dr.ing. Petre Lucian OGRUȚAN – Universitatea "Transilvania" din Brașov

Data, ora şi locul susținerii publice a tezei de doctorat:

18 Iulie 2022, ora 12:00, sala N.II.1.

Eventualele aprecieri sau observații asupra conținutului lucrării vor fi transmise electronic, în timp util, pe adresa corneliuzaharia@unitbv.ro

Totodată, vă invităm să luați parte la şedința publică de susținere a tezei de doctorat.

Vă mulțumim.

Cuprins

Table of Contents

Introducere

Sistemele de timp real (TR) – în contextul lucrării de față – sunt cele care asigură un răspuns comportamental într-o fereastră de timp determinată. Astfel, comportamentul nu trebuie să fie doar corect, ci și generat la timp. Latența acestor sisteme – întârzierile (legate de durata prelucrării ori a comunicațiilor de date – în rețea sau local, cu sub-ansamblele de stocare) – trebuie să fie cu cel puțin un ordin de mărime sub cele mai mici *constante de timp* ale proceselor deservite. Pentru a rezolva acest deziderat, resursele sistemului trebuie să fie folosite într-un mod strict controlat, determinist, iar sarcinile de rezolvat trebuie *planificate* astfel încât să nu suprasolicite nicio resursă critică.

În ultimii ani, de la sisteme relativ simple care rezolvă rapid sarcini de analiză elementară, s-a ajuns la sisteme complexe, cu *"decizie asistată"* – de algoritmi eficienți care interpretează volume mari de date. *Inteligența artificială (IA)* – definită de capacitatea de a rezolva probleme pentru care nu a fost explicit programată – este din ce în ce mai mult folosită în sistemele de TR, ceea ce aduce noi provocări în implementarea acestora. O nouă astfel de provocare este cerința ca o parte din aceste sistemele de TR să devină *mobile*, și în strânsă legătură cu aceasta, *reducerea consumului de energie* este o altă provocare din ce în ce mai greu de rezolvat, chiar în condițiile miniaturizării micro-electronice ce atinge deja limita de 3nm pentru dimensiunea unui tranzistor MOS integrat.

Oportunitatea tezei de doctorat

Una din cele mai importante oportunități ale acestei teze de doctorat constă în *maturizarea unor tehnologii* de implementare hardware (HW) care asigură *prețuri relativ scăzute* (mai ales pentru producția de masă, globalizată) unor soluții de miniaturizare, portabilitate-mobilitate și – chiar la nivel de terminal – *performanță computațională*, în special pentru calculele de IA, sub egida *"HW pentru IA".*

Dezvoltarea acumulatorilor și a tehnicilor de încărcare rapidă (cu sau fără fir – cu inducție) reprezintă o importantă oportunitate de *"funcționare pe bateri*i" a sistemelor moderne de TR. Sub aspect energetic, aceste dezvoltări vin în întâmpinarea unor oportunități de scădere a consumului pentru partea de procesare comparativ cu acela al părții de comunicații care impun regândirea alocării locale/centralizate a sarcinilor de calcul. Sub aspect computațional, performanța crescută a sistemelor mobile recente reprezintă și ea o importantă oportunitate pentru această re-alocare.

Pe de-o parte, avansul tehnologic permite concretizarea unor soluții algoritmice care își așteptau de multă vreme rândul pentru a fi puse în practică dar, pe de-altă parte, noile tehnologii reprezintă o oportunitate crucială pentru *dezvoltarea de noi algoritmi* (având în vedere că aceștia se pot implementa tot mai mult, parțial sau în întregime, direct în HW și nu numai în software – SW). Aceasta reprezintă o importantă provocare abordată în această lucrare – vizați fiind în special algoritmii pentru IA, fiind scontate rezultate importante: *noi arhitecturi computaționale, noi algoritmi,* sub egida *"IA pentru HW".*

Oportunitățile pentru care "este momentul acum" – asociate cu cerințe stringente ale pieței sau chiar ale comunității internaționale (în domenii societale-ecologice) – reprezintă "provocări" la care cercetarea doctorală prezentată în această lucrare a încercat să răspundă într-o măsură cât mai mare.

Obiectivele tezei de doctorat

Un prim obiectiv este *asigurarea resurselor sporite cerute de algoritmii de IA* – specifice *complexității computaționale* mărite, *volumului de date* considerabil ce trebuie prelucrat și, adesea, *vitezei mari de răspuns* impuse. De multe ori, un obiectiv suplimentar este *mobilitatea* și, implicit, funcționarea algoritmilor de IA într-un *buget de energie* dat, pentru a asigura *durata de funcționare* asociată cu *capacitatea bateriei* dar și cerințele de *disipare a căldurii*.

Așadar cercetările doctorale au ca obiectiv găsirea de soluții inovative pentru *implementarea de algoritmi de IA în sisteme de TR mobile*, astfel încât să fie respectate cerințele speciale de proiectare sus-menționate. Pentru atingerea acestui obiectiv, se urmărește detalierea cerințelor de proiectare, urmărind *evaluarea impactului* acestora asupra alternativelor

de implementare, cu scopul de *a selecta soluțiile particulare* dintre cele disponibile. Pentru această evaluare, obiectivul este *specificarea unor indicatori de performanță* (KPI,"Key-Performance Indicators") ce pot reprezenta *metrici* pentru eficiența soluțiilor practice (existente și potențiale), care trebuie analizate cu atenție pentru a trece la selectarea variantelor optime de implementare. Analiza indicatorilor de performanță are în vedere *obiectivul de a elabora un factor de merit* (compozit, cuprinzător) aferent unei soluții constructive. Pentru *validarea formulei propuse*, cercetarea doctorală urmărește evaluarea KPI pentru principalele arhitecturi computaționale existente. Ultima etapă a acestei analize are obiectivul de a identifica posibilitățile de *îmbunătățire a soluțiilor de implementare* ce respectă toate cerințele de proiectare – dezvoltare. Această ultimă fază de optimizare va urmări maximizarea factorului de merit. Pornind de la această optimizare a implementărilor specifice, un important obiectiv *este generalizarea soluțiilor propuse* către o gamă mai largă de aplicații.

În privința selectării soluției, un obiectiv este *alegerea între implementarea locală și cea centralizată* – "on the Edge (la marginea rețelei") sau "in the Cloud". Se urmărește *formularea criteriilor de selecție* – pentru a fundamenta alegerea făcută. Pentru majoritatea sistemelor de TR, constrângerile privind latența sunt problematice pentru implementările în Cloud, mărind importanța cercetărilor asupra soluțiilor "on the Edge" (în particular *Edge AI).*

În această teză de doctorat, abordarea algoritmilor are ca obiective *analiza structurilor de date* prelucrate, *specificarea rezultatelor așteptate*, *identificarea resurselor HW și SW alocate* pentru execuția algoritmilor. Obiectivul de *constituire a unui ecosistem algoritm – date – software – hardware* pentru soluțiile de TR, corespunde unei abordări integrative (quasi-holistice), fără a aborda separat componentele întrucât nu se poate face o simplă superpoziție (separare a cauzelor – modificări individuale asupra componentelor – și o recompunere a efectelor) deoarece aceste componente sunt într-o relație de strânsă interdependență. De aceea, soluțiile propuse nu sunt doar modificări la nivel HW sau SW, ci și soluții de modificare a algoritmilor de IA sau de modificare a reprezentării datelor (de intrare, ieșire sau intermediare). Astfel rezultă cele două categorii de soluții menționate anterior: soluții în care organizarea mai bună a datelor sau ameliorarea algoritmilor de IA ajută implementarea fizică – de exemplu la "decizia asistată" (de IA) pentru alocarea optimală a resurselor HW și pentru instanțierea-parametrizarea acestora (îmbunătățind performanțele care sunt în mod obișnuit specifice HW) – categoria *"IA pentru HW"* – precum și soluțiile în care măsurile luate la nivel fizic (HW) duc la utilizarea mai eficientă a datelor și la creșterea performanței algoritmilor de IA – categoria *"HW pentru IA".*

Un obiectiv foarte important al optimizării a fost gestiunea (în particular alocarea și instanțierea) *resurselor de calcul eterogene* prezente în marea majoritate a sistemelor de TR cercetate. În cadrul abordării eco-sistemice preconizate, această instanțiere – parametrizare corespunde unei "orientări pe obiecte", unei orientări "de sus în jos" ("top-down") ce transformă "clasa în obiect„ ; această fază a implementării reprezintă o concretizare sub-secventă fazei de "desfășurare – deployment" ("proiectare peste" resursele *disponibile* – contextual dar și *conjunctural, sporadic/dinamic* – a sarcinii computaționale aferente unui algoritm). Astfel, același algoritm, sau componentă a algoritmului, poate fi implementat(ă) pe diferite resurse de calcul existente în sistem: CPU (Central Processing Unit), DSP (Digital Signal Processor), GPU (Graphical Processing Unit), *Acceleratoare HW* generice sau specializate. *Partiționarea algoritmilor* și *selecția+alocarea resurselor de calcul potrivite* pentru fiecare componentă a algoritmului reprezintă un obiectiv central al cercetărilor efectuate. Majoritatea soluțiilor propuse sunt în acest sens *soluții hibride*, în care putem alege implementări atât HW cât și SW pentru diferite componente ale algoritmilor. Această abordare hibridă are anumite avantaje (ce vor fi abordate individual) care, în sistemele moderne, pot depăși dezavantajele complicării constructive, ale controlului relației dintre componentele sistemului și ale planificării temporale a activităților. Aceste avantaje vor fi prezentate detaliat pentru fiecare dintre soluțiile propuse.

Odată rezolvată problema partiționării HW-SW, un alt obiectiv este implementarea efectivă a unor *algoritmi complecși de IA concepuți pentru a beneficia de gestiunea resurselor de calcul paralel* (categoria "HW pentru IA"). Voi promova conceptul de *calcul masiv-paralel, eficient energetic* care permite rezolvarea a sute/mii/zeci de mii de operații pe "tact" CLK (Clock) folosind la minim bugetul energetic la dispoziție.

Astfel de soluții masiv-paralele pot fi implementate doar având în vedere și analizând în detaliu ecosistemul date – algoritm – SW – HW, pe de-o parte pentru a permite implementarea în HW a algoritmilor ceruți, pe de altă parte, însă, fără a supra-încărca în mod neesențial nucleele HW cu funcționalități care pot fi eliminate sau redimensionate. Obiectivul asumat e implementarea unor soluții de calcul masiv-paralel cu buget energetic neîntâlnite până acum în industrie.

Modul de abordare pentru găsirea și validarea soluțiilor propuse a impus și obiectivul de *a dezvolta o metodologie de proiectare specifică*. Această metodologie pornește de la analiza și documentarea soluțiilor precum și de la simularea matematică a acestora. Urmează implementarea soluțiilor, prin metode de *co-design HW-SW* , apoi validarea acestora prin *simulare comportamentală și implementarea de prototipuri*. Ultima etapă a metodologiei de proiectare constă în integrarea în sisteme comerciale și de producție pe scară largă

O atenție deosebită a fost acordată obiectivului de *diseminare a rezultatelor cercetării doctorale.* Pentru o valorizare corespunzătoare a ideilor a fost acordată prioritate obiectivului de *patentare a soluțiilor inovatoare și protecție a proprietății intelectuale.* Un alt obiectiv a fost *publicarea de articole și participarea la conferințe, precum și la discuții cu reprezentanți ai mediului economic.*

Ca urmare a acestor eforturi, majoritatea soluțiilor propuse vizează implementări în produse comerciale, prezente pe piață. Am urmărit să jalonez nivelele de maturitate tehnologică a soluțiilor - TRL (Technology Readiness Levels).

Organizarea tezei de doctorat

Teza este structurată pe 7 capitole precedate de o Introducere și succedate de o listă a referințelor.

- Introducerea prezintă oportunitatea și motivarea tematicii alese, precum și obiectivele propuse pentru activitatea de cercetare doctorală. De asemenea, abordează baza experimentală folosită în cadrul cercetării doctorale.
- Capitolul 1, intitulat "Sisteme de timp real cu inteligență artificială" detaliază modul de analiză, de folosire și de implementare a sistemelor care reprezintă subiectul cercetării doctorale. În acest capitol de contextualizare – problematizare, se pune accentul pe cerințele specifice sistemelor de TR și pe modul în care IA le poate influența. Se introduc conceptele de bază cercetate, precum ecosistemul date-algoritm-HW-SW. O secțiune importantă este reprezentată de analiza indicatorilor de performanță pentru implementarea algoritmilor IA în sisteme de TR. Acești indicatori sunt baza factorului de merit pentru fiecare soluție și vor fi folosiți în continuare pentru analiza soluțiilor existente sau propuse.
- Capitolul 2 analizează stadiul actual din domeniu, prin studiul arhitecturilor computaționale folosite în IA. Se au în vedere atât arhitecturile standard, de uz general (CPU, GPU, DSP, VPU, NPU) precum și arhitecturile specializate (implementate pe FPGA sau ASIC). Pentru fiecare arhitectură analizată se explică în detaliu indicatorii de performanță. Pentru aceste analize, cercetarea bibliografică a inclus o serie de articole cuprinzătoare de tip "survey".
- În capitolul 3 prezint soluții de implementare a acceleratoarelor HW pentru PI în sistemele de TR. Am pornit de barierele la care s-a ajuns în implementarea de arhitecturi complexe în sistemele de TR, arătând cum acestea pot fi depășite prin soluții arhitecturale specifice. În încheierea capitolului am arătat potențialul IA pentru îmbunătățirea acestor implementări, indicând direcțiile pentru următoarele două capitole.
- Capitolul 4 detaliază soluții de implementare din categoria "IA pentru HW". Prezint astfel soluții inovatoare de adaptare a algoritmilor IA pentru a folosi modulele HW în mod eficient. Am explicat modul de *integrare adaptivă a resurselor HW*, exemplificând cu modulul AHIP dezvoltat pentru PI. De asemenea, am prezentat *parametrizarea dinamică anticipativă a HW de către IA* și am ilustrat cum se poate realiza acest lucru pentru un algoritm de detecție de obiecte.

  Am arătat prin soluții concrete cum se poate *controla precizia implementării HW folosind IA pentru a spori toleranța la erori*. De asemenea am analizat *controlul prin IA al fluxului de date din HW* și modul cum IA poate beneficia de fuziunea senzorilor.

- Capitolul 5 se concentrează pe soluții din categoria "HW pentru IA". Am introdus și am concretizat conceptul de *procesare masiv-paralelă eficientă energetic,* prin intermediul căruia am explicat modul de proiectare a acceleratoarelor HW pentru rețele neurale complexe în TR. Prelucrarea multi-procesor este o extensie naturală a acceleratoarelor definite, prin intermediul căreia se pot implementa algoritmi mult mai complecși, folosiți în sisteme critice de TR. Am arătat cum sistemele multiprocesor pot funcționa în mod redundant. O altă secțiune a capitolului 5 detaliază un nou mod de proiectare a nucleelor HW, orientat nu numai pe puterea de calcul ci, în special, pe utilizarea eficientă a magistralei de date – indicatorul de performanță cel mai important.
- În capitolul 6 am prezentat modul în care tehnologiile și soluțiile propuse în capitolele anterioare sunt implementate în aplicații specifice, de nivel înalt TRL. Am dat exemple de câteva studii de caz pentru aplicații bazate pe accelerarea HW pentru IA. Am detaliat aplicațiile de detecție a obiectelor, stabilizare a imaginilor sau de monitorizare a exteriorului și interiorului autovehiculelor.
- Capitolul 7 este rezervat concluziilor generale, contribuțiilor originale și dezvoltărilor ulterioare. De asemenea se prezintă modul de validare și diseminare a rezultatelor științifice.
- Referințele cuprind lucrările publicate de autorul tezei (articole, grupuri de patente indexate Derwent, alte publicații în domeniu), apoi bibliografia și web-grafia.

  O mențiune deosebită merită brevetarea contribuțiilor originale, bine ilustrată în lista de referințe. Conceptele inovatoare au fost propuse pentru patentare în mai multe țări (cu indicativele US, CN, KR, JP, TW, etc), grupuri de țări (indicativ EP pentru European Patent Office) sau chiar pe plan mondial (indicativ WO pentru World Patent Office). Au fost evidențiate cele 25 de grupe de patente indexate în Clarivate Analytics Web of Science - Derwent Innovations Index.
- Anexa 1 cuprinde două articole reprezentative, indexate ISI.
- Anexa 2 ilustrează printr-un exemplu concret circuitul unei invenții ("AHIP - Advanced Hardware for Image Processing"): propunerea (cererea de brevetare – *A, "Application"),* negocierea revendicărilor *("claims"),* patentul inițial aprobat, iterații ulterioare (clarificări), cuplări de brevete, îmbunătățiri și completări, etc.

Figura 1 prezintă modul în care capitolele tezei abordează domeniile de interes ale cercetării doctorale. În lucrare am evitat o prezentare foarte detaliată a domeniilor de inteligență artificială sau de procesare a imaginilor, domenii vaste și greu de acoperit, dar ne-am concentrat pe relațiile dintre sistemele de TR, IA și PI. Contribuțiile principale, grupate în capitolele 4 și 5 se află la intersecția celor trei domenii de interes.

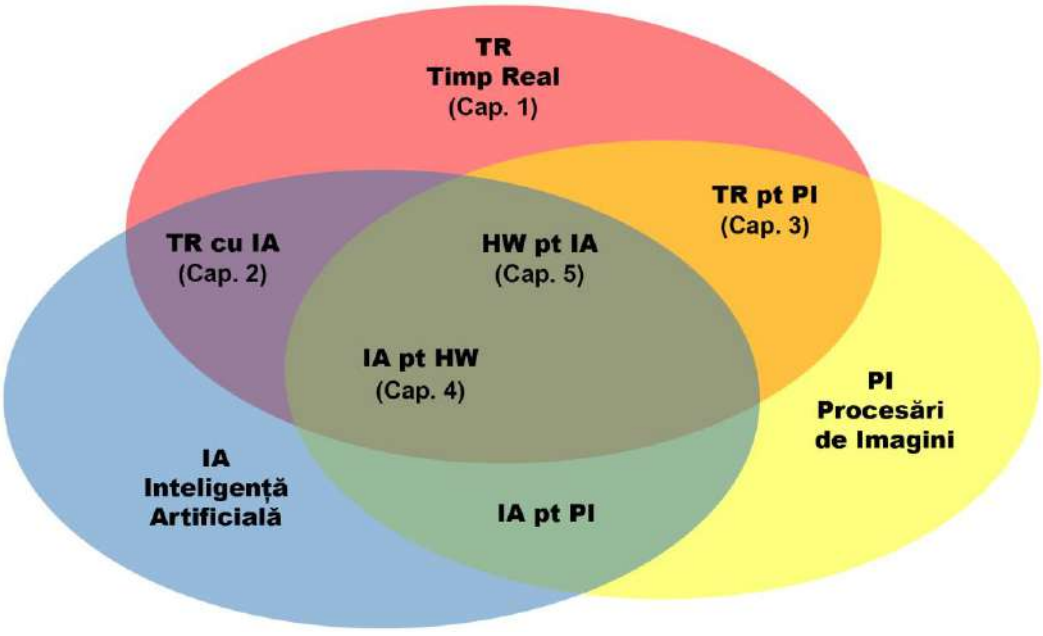


Figura 1: Aria de cercetare a lucrării

Baza experimentală

Pentru realizarea cercetării doctorale s-a folosit o bază experimentală modernă și diversificată, adaptată în funcție de nivelul TRL abordat.

Astfel, pentru TRL inferioare (1-3) s-a pus accentul pe simulări funcționale și modelări matematice. Pentru aceasta au fost folosite unelte SW corespunzătoare dintre care:

- Simularea folosind modele Matlab
- Modelarea comportamentală în C, C++
- Simulare HW de nivel înalt, HLS *(High Level Synthesis)*
- Simulare HW de nivel scăzut, RTL *(Register Transfer Level)* sau la nivel de poartă *(Netlist)*
- Simulare algoritmică folosind limbajul Python

Pentru TRL intermediare (4-6) , s-a folosit în principal prototiparea pe FPGA (Field Programmable Gate Arrays). Sistemele folosite au fost pe baza FPGA de la Xilinx, pornind de la modelele pe arhitectură Virtex și până la sisteme SoC de tip Ultrascale+. Sistemele de prototipare folosite sunt prezentate în Figura 2.

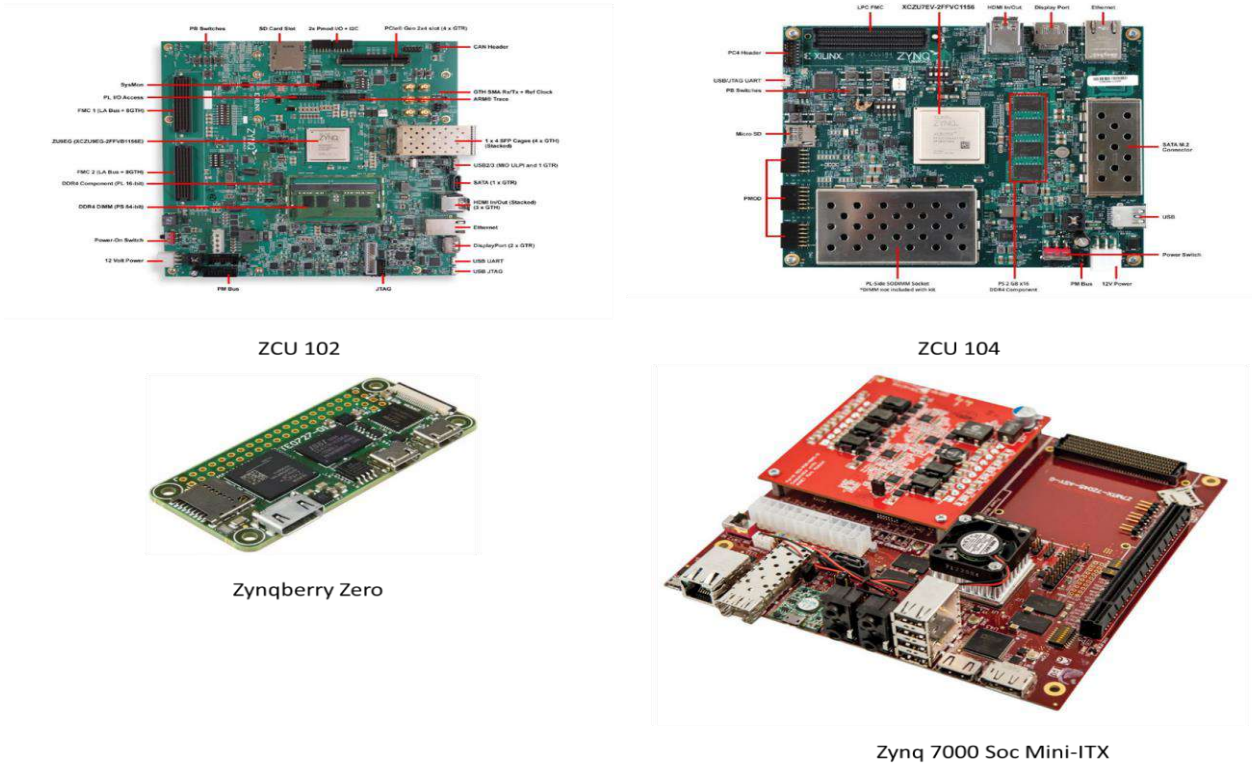


Figura 2: Sisteme cu FPGA folosite pentru prototipuri

Este nevoie de diferite adaptoare pentru ca sistemele cu FPGA să funcționeze în condițiile specificate, precum cele prezentate în Figura 2 precum și o serie de senzori, precum cei prezentați în Figura 3.

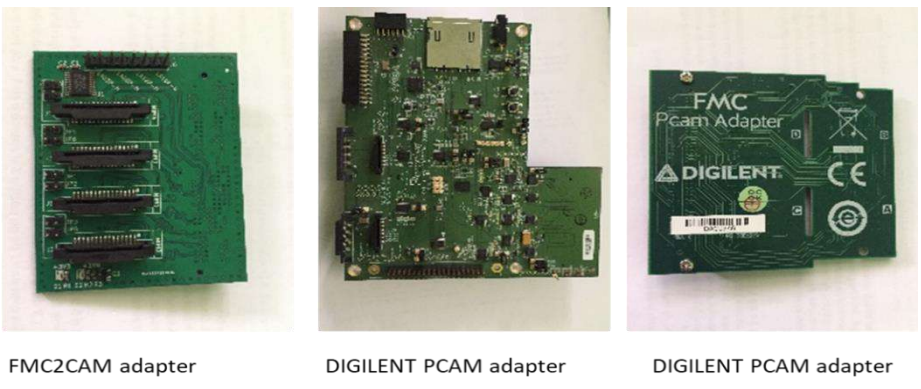


Figura 3: Plăci de adaptare pentru sistemele prototip

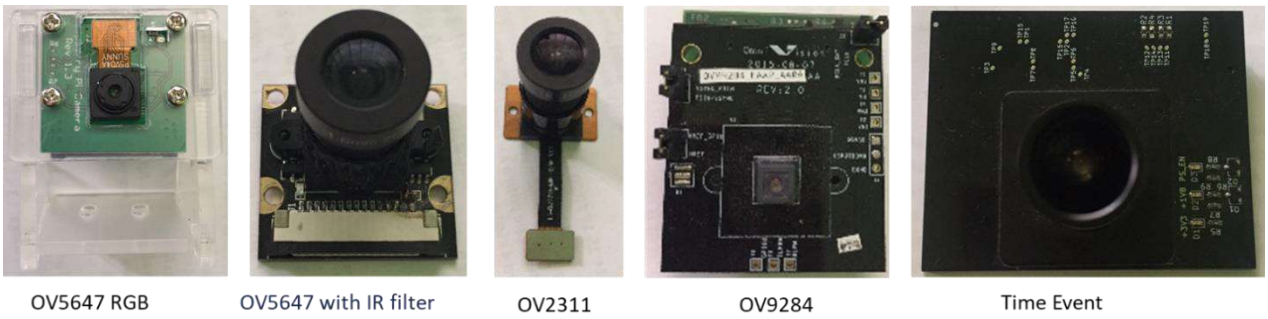


Figura 4: Senzori folosiți în sistemele de prototipare cu FPGA

Pentru sisteme mai complexe care au nevoie de mai multe circuite logice, s-au folosit sisteme cu FPGA multiple, așa cum sunt cel prezentate în Figura 5.

Figura 5: Sistem prototip cu FPGA multiple

În cazul în care rezultatele cercetării s-au concretizat în produse comerciale (TRL 7-9 și dincolo de acestea), baza experimentală a fost constituită din circuite integrate care implementează soluțiile propuse și prezentate pe larg în teza de doctorat – așa cum este exemplificat în Figura 6.

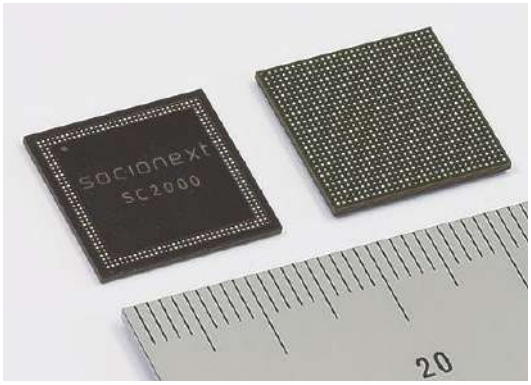


Figura 6: Chip SoC Socionext

Aceste circuite integrate sunt folosite în sisteme pentru aplicații specifice detaliate în capitolul 6.

Capitolul 1. Sisteme de Timp Real (TR) cu Inteligență Artificială (IA)

Sistemele de TR cu IA sunt analizate în literatura de specialitate din ultimii ani [34]. Cu toate aceste demersuri științifice, algoritmii de IA cu performanțe superioare, implementați în sisteme de TR, au căpătat o popularitate mai mare abia de curând, când sistemele HW au devenit relativ mai performante. Aceste sisteme au fost valorizate atât pentru arhitecturile generale de învățare asistată [35], [36], dar mai ales în direcția implementării HW a rețelelor neurale [37], [38], [39].

În majoritatea analizelor efectuate s-a urmărit modul în care algoritmi de IA existenți pot fi implementați în sistemele de TR cu menținerea criteriilor de performanță, preț sau energie consumată. Această analiză este limitată din cel puțin două puncte de vedere. În primul rând, o înțelegere detaliată a sistemelor de TR, cu modul de interacțiune dintre procesarea locală (Edge) și centralizată (Cloud) poate duce la o analiză mai intimă a relației dintre date – algoritmi – software și hardware, care pot să fie privite ca un *ecosistem.*

În al doilea rând, indicatorii de performanță analizați în contextul actual pentru sistemele HW trebuie să treacă de criteriile clasice de tip *Performance – Power – Area (PPA* – Performanță – Energie – Arie de siliciu sau cost) și trebuie să atingă aspecte critice precum flexibilitatea în utilizare, protecția și securitatea sau riscurile pentru utilizare.

În acest prim capitol introduc conceptele cu care voi opera pe parcursul tezei, și voi extrage direcțiile principale de analiză a sistemelor de timp real cu inteligență artificială.

1.1 Sisteme de timp real

Sistemele de calcul de TR sunt sisteme în care corectitudinea funcționării depinde nu doar de modul în care se execută sarcinile de lucru, ci și de momentul în care acestea sunt efectuate. Pentru ca sarcinile să se realizeze exact la momentul potrivit, sistemele de TR trebuie să permită controlul și chiar prezicerea apariției acestor sarcini. Așadar nu numai viteza ci și *promptitudinea* trebuie reglementate prin *planificare*.

Modelul de bază al unui sistem de TR

Sistemul de TR cuprinde diverse componente HW și SW încorporate astfel încât sarcinile specifice să poată fi efectuate cu constrângerile de timp permise. Precizia și corectitudinea sistemului de TR fac ca modelul să fie complex – Figura 7.

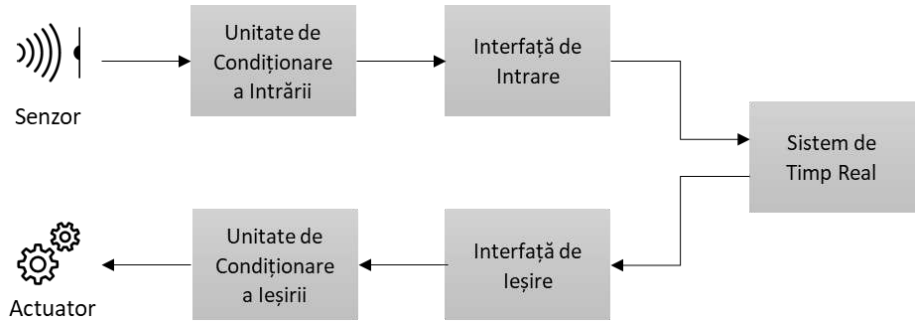


Figura 7: Arhitectura de bază a unui sistem de timp real

Senzorul: este utilizat pentru conversia unor evenimente sau caracteristici fizice în semnale electrice. Senzorii sunt dispozitive HW care preiau intrarea din mediu și o transmit sistemului prin conversia acesteia (mecanic → electric etc).

Actuatorul: este dispozitivul invers al senzorului. Acesta transformă semnalele electrice în "acționări" fizice, mai general, în "evenimente". Intrarea acestuia este de la unitatea de condiționare de ieșire.

Unitatea de condiționare a semnalului *(signal conditioning unit):* Sistemul nu poate folosi direct semnalul electric generat de senzor. Acesta trebuie prelucrat ("condiționat") pentru a putea fi utilizat.

- Unitatea de condiționare a intrărilor: Este utilizată pentru condiționarea semnalelor electrice provenite de la senzor.

- Unitatea de condiționare a ieșirii: Se utilizează pentru condiționarea semnalelor electrice provenite din sistem.

Interfețele: sunt utilizate în principal pentru conversia digital ↔ analog. Semnalele provenite de la unitatea de condiționare a intrărilor sunt analogice, iar sistemul funcționează digital, de aceea interfața de intrare asigură conversia analog-digitală a semnalului. În mod similar, în timp ce transmite semnalele la unitatea de condiționare a ieșirii, interfața de ieșire face conversia digital-analog a semnalului.

Sarcinile efectuate de sistemele de TR sunt de două feluri: sarcini periodice și sarcini dinamice.

Modelul de lucru pentru sistemele de TR cu IA

Pornind de la modelul de bază prezentat anterior, în contextul lucrării de față, un sistem de TR cu IA conține următoarele componente de procesare:

- Recepționarea semnalului
- Conversia semnalului de măsurat în semnal electric *("sensing")*
- Procesarea semnalului în domeniul analogic
- Conversia analog-digitală a semnalului
- Procesarea semnalului în domeniul digital

În funcție de domeniul de utilizare și de algoritmii de IA folosiți, procesarea în domeniul digital se împarte, tipic, în:

- Procesarea generică a semnalului digital (eliminarea zgomotului și a altor distorsiuni, normalizarea semnalului)
- Procesare pentru IA, care este împărțită în următoarele etape:
    - Pre-procesare
    - Inferență (rularea algoritmului de IA)
    - Post-procesare

Fluxul de date poate fi liniar (fiecare bloc folosește semnalul generat de blocul anterior "ca atare" – așa cum e generat de acesta) sau neliniar (în care semnalul este stocat și utilizat de blocul următor într-un mod specific, diferit de cel generat).

Memoria de stocare poate fi locală (între două blocuri consecutive ale fluxului de date), împărțită între mai multe blocuri (partajată) sau globală.

O schemă care să descrie acest mod de lucru este prezentată în Figura 8. În sistemele de procesare a semnalelor, fiecare bloc din fluxul de date este configurat independent, în funcție de modul de folosire.

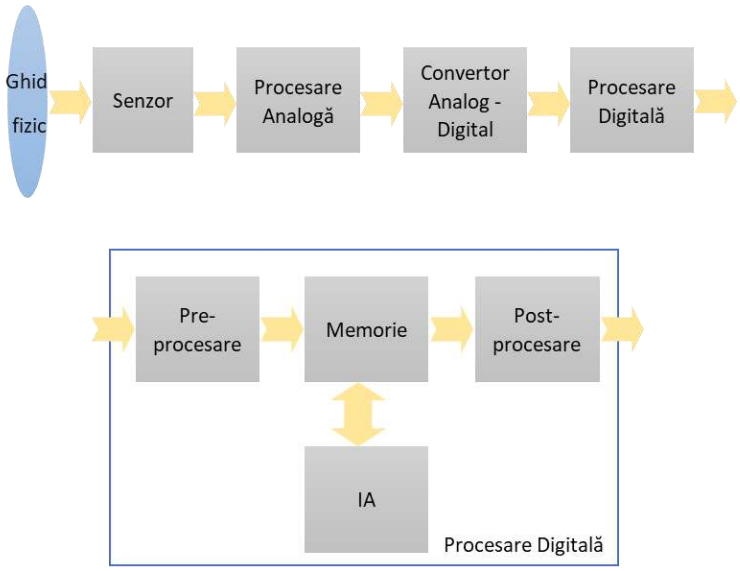


Figura 8: Modelul de lucru pentru sistemele de TR cu IA

1.2 Mediul distribuit (Edge-Cloud)

În practică se întâlnesc două tipuri de sisteme de TR, din punct de vedere a locului în care se rulează algoritmii IA – în special în contextul *mobilității.* Sistemele de tip Cloud funcționează pe principiul centralizării: datele recepționate de sistemele mobile sunt pre-procesate local, comprimate și apoi transmise în Cloud [2-CZ-A]. Procesarea se execută în Cloud iar apoi rezultatele sunt transmise sistemului mobil, pentru a fi folosite. Arhitectura generală a unui sistem de TR în Cloud este prezentată în Figura 9.

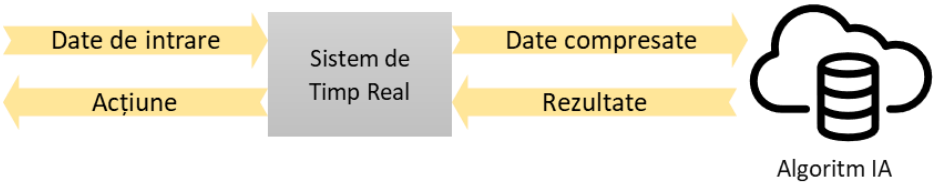


Figura 9: Arhitectura generală a unui sistem de timp real cu procesare în Cloud

Avantajele principale ale unei astfel de arhitecturi țin de capacitatea de procesare mare a sistemului Cloud. Astfel, algoritmi IA complecși pot fi rulați cu viteză mare în Cloud iar rezultatele sunt trimise sistemului mobil pentru a acționa în timp real. De asemenea, complexitatea sistemului mobil poate fi astfel mai scăzută, lucru care poate duce la costuri mai mici de implementare a unei astfel de soluții.

Principalul dezavantaj al unei soluții cu procesare IA în Cloud ține de *timpul de transmitere* necesar pentru trimiterea datelor și, respectiv, recepționarea rezultatului de către dispozitivul mobil (care se adaugă la timpul de calcul – chiar dacă acesta, în sine, poate fi mult redus). În sistemele de TR acest parametru este critic iar dacă timpul total de răspuns este depășit sistemul nu mai poate funcționa optim. În funcție de aplicație, acest factor poate descalifica arhitectura de procesare în Cloud.

Un alt dezavantaj ține de *volumul de date* care trebuie transmis(e) către Cloud. Deși datele sunt comprimate, sau/și pre-procesate, o cantitate mare de date trebuie transmisă permanent către sistemul Cloud, spre a fi procesate.

Aceste dezavantaje au o implicație semnificativă – pe de o parte asupra costurilor și, pe de altă parte, asupra energiei consumate.

O dată cu dezvoltarea tehnologică, *energia necesară calculelor pe dispozitivele mobile a devenit mult mai mică decât energia necesară pentru transferul datelor*. Așa cum este descris în [41] și este prezentat în Figura 10, energia folosită pentru transferul de date este mai mare cu câteva ordine de mărime față de energia necesară pentru calcule locale.

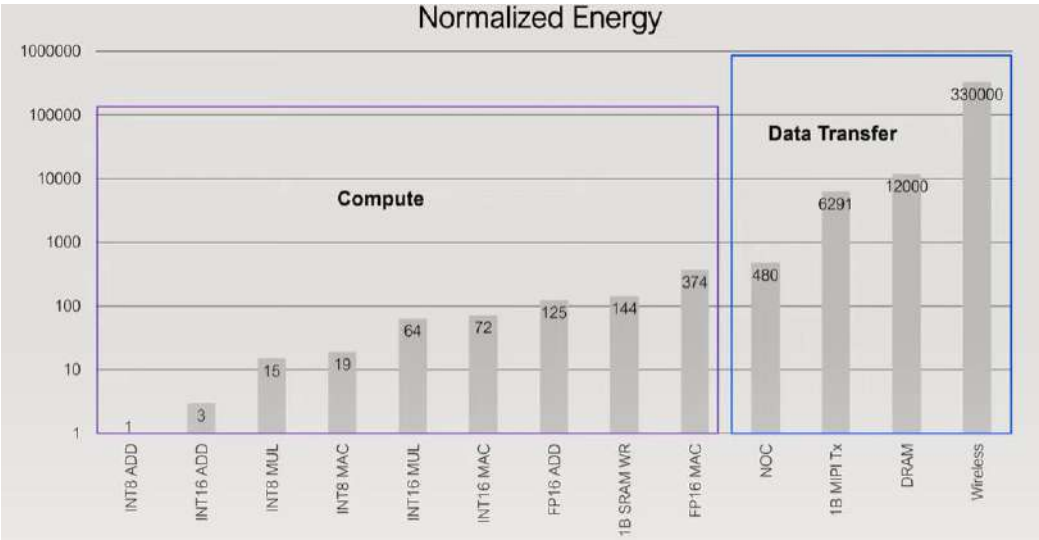


Figura 10: Energia normalizată consumată în SoC [41]

Acest decalaj relativ crește o dată cu noile tehnologii de implementare în siliciu și reprezintă o oportunitate pentru tranziția (totală sau parțială) a algoritmilor IA de la nivel central la nivel local (Cloud→Edge). În plus, dezvoltarea dispozitivelor mobile poate permite acum implementarea locală a unor algoritmi IA performanți, de calitate asemănătoare celor care rulează în Cloud.

Sistemele cu procesare locală *(Edge Computing* - "la marginea rețelei") modifică paradigma de funcționare a sistemelor de TR, astfel încât comunicarea în Cloud se poate face doar "pentru informare" / pentru sincronizare de nivel înalt. Arhitectura generală a unui sistem cu procesare locală (Edge) arată ca în Figura 11.

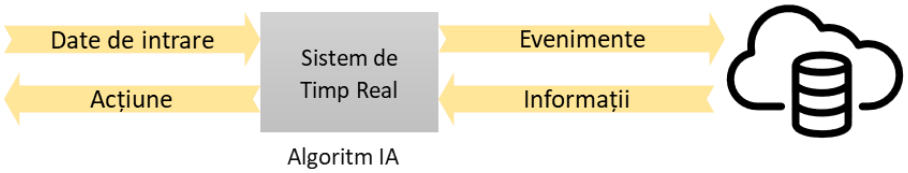


Figura 11: Arhitectura generală a unui sistem de timp real cu procesare locală (Edge)

Printre avantajele principale ale unui astfel de sistem se detașează viteza de răspuns sporită a sistemului la datele de intrare precum și minimizarea traficului în rețea. Dezavantajele țin de calitatea algoritmului IA (care este constrâns de capacitatea de procesare a sistemului mobil) și a puterii consumate pentru implementarea IA, precum și de costului mai mare al dispozitivului mobil (care trebuie să conțină resurse de calcul superioare) [42], [43].

De asemenea, actualizarea algoritmilor IA poate fi mai dificilă în cazul implementării *"on the Edge"* decât în cazul implementării în Cloud.

Un alt aspect important, mai ales în ultima perioadă, ține de *confidențialitatea* datelor și modul de *protecție la atacuri informatice.* Din acest punct de vedere, sistemele cu procesare locală oferă o soluție cu protecție superioară arhitecturală *(privacy by design)* față de modelul Cloud, deoarece datele brute, recepționate de sistemul mobil nu părăsesc fizic dispozitivul și sunt astfel mai puțin expuse.

Având in vedere cele prezentate anterior, se observă o tendință de relocare a proceselor de IA dinspre Cloud spre Edge. Aceasta este una dintre motivațiile lucrării de față. Astfel, *accentul este pe modalitățile de optimizare a procesării locale pentru sistemele de TR cu IA* în ideea creșterii performanțelor acestor sisteme.

Este interesant de amintit și o tendință de relocare a proceselor IA în sens invers, dinspre Edge înspre Cloud. Aceasta apare nu în ideea renunțării la Edge Computing, ci la suplimentarea acestuia cu capabilități superioare, rezolvate în Cloud. Astfel, se pot imagina sisteme în care IA locală (Edge AI) comunică cu IA globală (Cloud AI) pentru schimburi de informații și adaptarea algoritmilor locali cu informații culese și procesate global. Așadar repartizarea sarcinii computaționale (problema clasică a sistemelor distribuite) devine *o problemă de corelare în mediu distribuit* (Cloud / Edge) – centralizat / localizat ("la marginea rețelei").

1.3 Ecosistemul date–algoritmi–software (SW)–hardware (HW) pentru sisteme de IA de TR

Un ecosistem, conform definiției din Enciclopedia Britannica este un complex de organisme, mediul lor fizic și relațiile dintre acestea, într-un anumit spațiu. Conceptul a fost extins în alte domenii, în care există o legătură strânsă între diferite entități care se adaptează unele la comportamentul celorlalte. De exemplu, în industrie, conceptul de ecosistem este văzut ca o rețea de organizații, implicate în livrarea unui produs sau serviciu, atât prin colaborare cât și prin competiție.

Fiecare entitate a ecosistemului afectează și este afectată de celelalte, creând relații care *evoluează,* în care fiecare entitate trebuie să fie *adaptabilă* pentru a supraviețui în sistem.

Această teză de doctorat fiind orientată pe *relaționare,* vom considera în acest capitol un *ecosistem de TR cu IA,* ale cărui părți interacționează strâns – mergând până la modificări ale *comportamentului* acestora. O schemă a ecosistemului de procesare în TR cu IA este prezentată în Figura 12.

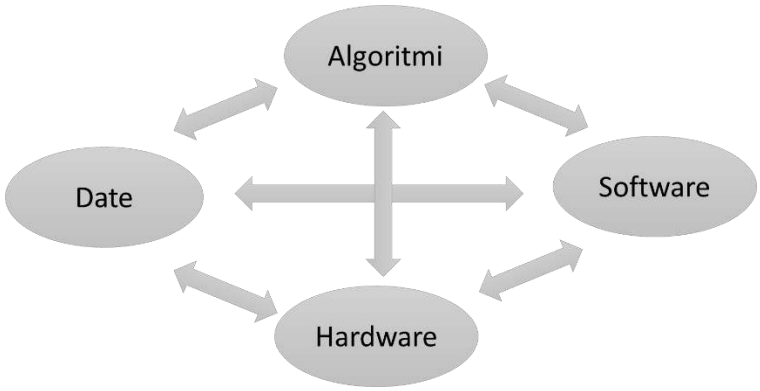


Figura 12: Ecosistemul de procesare în TR cu IA

În continuare sunt prezentați principalii contributori la ecosistemul de procesare în TR cu IA, cu accent pe parametrii lor care pot fi modificați pentru a îmbunătăți calitatea interacțiunilor.

Datele reprezintă obiectele cu care se operează în sistem. Datele pot fi recepționate, manipulate, stocate și transmise în funcție de nevoile sistemului. În ecosistemul de procesare cu IA, datele cu care se lucrează sunt transformate în mărimi digitale. Acestea pot fi reprezentate în diferite formate (în virgulă fixă sau mobilă) și pe rezoluții diferite (de la 1 bit până la 64 de biți per componentă). Alți parametri ai datelor cu care se lucrează sunt: raportul semnal/zgomot, eroarea relativă, gama dinamică, etc.

Algoritmii transformă datele de intrare în rezultate (date de ieșire). În general, pentru sistemele cu IA, algoritmii cu care se lucrează sunt hotărâtori pentru asigurarea răspunsului în TR. Deși rapiditatea este esențială, sunt totuși parametri ai algoritmilor care pot fi modificați (în marjele acceptabile de viteză) – începând chiar cu *tipul* acestor algoritmi. Pentru obținerea unor rezultate asemănătoare, se pot folosi o gama largă de tipuri de algoritmi cu IA, cum ar fi: rețelele neurale artificiale (ANN – "Artificial Neural Networks"), regresiile, arborii de decizie – ajungând până la "random forests" ("păduri aleatoare"), metoda nucleelor ("kernel approximation") și "mașinile cu suport vectorial" – SVM (Support Vector Machines), clasificatori bayesieni "naivi", metoda celor mai apropiați K vecini ("K nearest neighbors"), etc. *Calitatea* algoritmului este parametrul cel mai important de luat în considerare la alegerea acestuia. *Complexitatea* algoritmului poate fi și ea modificată în funcție de aplicație sau de rezultatele așteptate. *Toleranța la erori* este un alt parametru care poate fi relevant în acest context.

Modul în care algoritmii sunt implementați poate fi software (SW) și/sau hardware (HW).

Implementarea software presupune existența unui HW *programabil.* Implementarea SW poate fi parametrizată pentru viteză (optimizările pot fi făcute astfel încât rezultatele să fie generate cât mai rapid posibil) sau, pentru ușurință, în implementare și depanare (pentru acestea contează limbajele de programare folosite).

Modul de utilizare a resurselor HW poate fi de asemenea un parametru care se poate modifica, de exemplu prin manipularea memoriei într-un mod automat sau manual, ori prin folosirea de acceleratoare specifice (cum ar fi chip-ul NEON pentru grafica 3D pe 256 de biți [44]. Modul în care SW folosește procesarea în paralel sau modul de interacțiune cu alte componente SW sau HW reprezintă alți parametri ce contează, de asemenea, în ecosistemul de procesare în TR.

Implementarea SW poate fi locală – pe resursele HW existente pe platforma de TR – sau în Cloud. Se poate opta și pentru o implementare mixtă, în care o parte a SW rulează local iar o parte în Cloud, granița dintre Edge și Cloud putând fi de asemenea un parametru care se poate modifica.

Hardware-ul folosit pentru execuția algoritmilor reprezintă partea fizică în sistemele de TR. În aceste sisteme, HW este pe de o parte reprezentat de unitățile computaționale generice pe care rulează SW (CPU, GPU, DSP) precum și de blocurile specifice care pot implementa complet sau parțial algoritmii. Astfel, pot exista sisteme de TR în care algoritmii sunt complet implementați pe HW *dedicat* sau sisteme în care *o parte* a algoritmilor sunt implementați în SW.

Parametrii implementării HW sunt: tehnologia folosită pentru implementare, complexitatea implementării, timpul de răspuns, energia consumată, flexibilitatea în utilizare și protecția împotriva riscurilor (de eroare sau de securitate).

Implementarea HW poate fi locală sau în Cloud. Modul de folosire al acceleratoarelor HW în Cloud este un parametru care poate fi luat în considerare în sistemele de TR.

Un aspect esențial al ecosistemului este definit de relațiile dintre părțile implicate. Aceste relații sunt:

Relația dintre date și algoritmi: modul de reprezentare a datelor precum și aria posibilă de acoperire a acestora sunt componente esențiale în alegerea algoritmului potrivit pentru problema de rezolvat. De exemplu, pot fi aleși cu totul alți algoritmi pentru rezolvarea unei probleme în care datele de intrare sunt afectate de zgomot, decât în cazul contrar.

În celălalt sens, modul de proiectare a algoritmilor pot impune tipul de reprezentare a datelor sau numărul de biți folosiți pentru stocarea datelor digitale.

Relația dintre algoritmi și SW este una foarte strânsă. Deși în principiu, algoritmii pot fi dezvoltați și testați în diferite moduri, implementarea SW de nivel înalt este metoda cea mai utilizată acum. Translatarea algoritmului din reprezentarea SW de nivel înalt în implementare SW optimizată pentru sistemul de TR este o relație directă. Și implementarea SW de TR poate influența algoritmul, prin impunerea de constrângeri impuse de timpul de execuție sau de costul ori complexitatea de implementare în SW a construcției algoritmice.

Relația dintre HW și SW este de asemenea una directă. Resursele HW programabile definesc modul în care SW este implementat precum și performanțele la care acest SW rulează. Resursele HW dedicate au o implicație directă asupra părților de procesare SW care pot fi strămutate în HW precum și asupra modului de interacțiune a componentelor SW. Invers, implementarea SW poate impune constrângeri HW, prin dimensionarea resurselor HW astfel încât să poată rezolva cerințele impuse de SW.

Relația dintre date și HW: modul de reprezentare a datelor are un impact major și imediat asupra infrastructurii HW. Cel mai folosit parametru care impune un anumit tip de HW este tipul de reprezentare a datelor și numărul de biți.

Pentru implementări folosind date în virgulă fixă, pe 8 biți se va folosi un alt HW decât pentru implementări în virgulă mobilă pe 32 de biți.

În sens invers, constrângerile HW pot duce la reprezentări diferite ale datelor. Astfel se poate ajunge la rafinarea reprezentării diferitelor tipuri de date pentru a servi în mod optim HW existent. Un exemplu este implementarea rețelelor neurale în care acceleratoarele HW suportă ponderi pe un număr mic de biți (reprezentări binare, ternare).

Relația dintre date și SW este de asemenea una directă. Modul de reprezentare a datelor determină modul de implementare a SW. Invers, optimizări SW pot duce la cerințe diferite de reprezentare a datelor, de exemplu prin trecerea din virgulă mobilă în virgulă fixă sau prin transformarea spațiului de reprezentare a datelor (transformări Fourier, transformări ale spațiului de culoare, etc).

Relația dintre algoritmi și HW nu este exploatată suficient în proiectarea de sisteme de TR cu IA și, de aici, unul dintre obiectivele importante ale acestei teze de doctorat. Algoritmii de IA pot duce la alegerea unor arhitecturi HW specifice, în funcție de criteriile de performanță cerute. Mai mult, diferite resurse HW existente și disponibile în sistem pot fi folosite sau configurate în mod diferit la cererea algoritmilor IA. În sens invers, constrângerile HW pot duce la modificarea algoritmilor de IA, astfel încât aceștia să poată genera rezultatele așteptate chiar și în condiții ne-ideale (cum ar fi calcule în HW cu precizie scăzută).

Pot exista și relații mai complexe între componentele ecosistemului:

Algoritmii pot determina *dozajul* dintre HW și SW *(mutarea graniței* dintre HW și SW). În funcție de deciziile luate de IA, unele componente ale algoritmului pot fi rulate fie HW, fie SW. De exemplu, dacă IA decide că nu este necesară o prelucrare complexă a datelor, acestea se pot procesa rapid în HW. În momentul în care IA detectează evenimente care necesită o procesare mai detaliată sau specifică a datelor, această procesare se poate face în SW, beneficiind de resurse, timp și flexibilitate specifice. Acest concept este prezentat mai în detaliu în capitolul 4 ("IA pentru HW") în secțiunea de configurare dinamică anticipativă.

Datele pot influența HW folosit și algoritmii implementați. Spațiul de reprezentare a datelor recepționate poate fi divizat astfel ca, în funcție de acestea să fie folosit un HW specific. De exemplu, dacă datele sunt recepționate rar, sau acestea au valori care nu depășesc niște praguri, sistemul HW folosit este unul minimal, iar algoritmii folosiți sunt simpli și rapizi. Atunci când cantitatea de date crește sau valorile acestora ies dintr-o anumită gamă, atât comportamentul algoritmului, cât și implementarea HW folosită vor fi diferite. Acest concept este prezentat mai în detaliu în secțiunea de segmentare predictivă a datelor a capitolului 4.

Implementările HW permit folosirea unor date reprezentate diferit și utilizarea unor algoritmi adaptați. Constrângerile HW pot duce nu doar la reprezentarea datelor pe un număr diferit de biți sau la reprezentarea în virgulă mobilă sau fixă a acestora, ci pot duce la transformarea spațiului de reprezentare a datelor și la scăderea preciziei (creșterea erorilor) în cazul acestor transformări prin simplificarea calculelor. Aceste modificări de precizie pot fi *compensate prin reantrenarea algoritmilor de IA.* Aceasta relație este exemplificată în capitolul 4 – secțiunea privind adaptarea algoritmilor IA pentru implementări HW eficiente.

Toate aceste relații, care modifică înșiși participanții la ecosistem, sunt construite astfel încât să ducă la creșterea factorului de merit al implementării sistemului de TR. Ca urmare a acestor relații, se poate ajunge la rezultate arhitecturale care, la prima vedere, sunt contra-intuitive. De exemplu, poate fi obținută o creștere semnificativă a factorului de merit al implementării chiar cu prețul supradimensionării HW – pentru a permite adaptarea implementării SW în funcție de datele procesate și de algoritmul folosit.

1.4 Calculul eterogen

Sistemele moderne de PI în TR sunt sisteme de *calcul eterogen* ("heterogeneous computing"). Sistemele eterogene sunt sisteme care folosesc doua sau mai multe tipuri de nuclee de calcul diferite, cum ar fi: CPU, GPU, DSP, FPGA sau acceleratoare HW. Folosind nuclee de tipuri diferite, sistemele pot realiza sarcini pe care cele cu un singur tip de procesor nu le pot realiza. Sarcini specifice pot fi realizate mai rapid folosind resurse dedicate (cum ar fi acceleratoarele HW). Cantități mari de date pot fi procesate în paralel (folosind GPU) sau mai eficient (folosind DSP). Dispozitivele FPGA pot implementa funcții dedicate, care pot fi apoi *redefinite* de utilizator în HW.

Conceptual, sistemele eterogene nu sunt noi [45]. Cu toate acestea, e nevoie de soluții noi (înlesnite de avansul tehnologic din electronică) privind modul de interacțiune între diferitele componente de calcul. Această interacțiune trebuie adaptată în cazul sistemelor de TR care folosesc IA, sub aspectul optimizării capacității de procesare și a consumului energetic.

Provocări pentru noile sisteme de calcul eterogene

Așa cum se arată și în [48], [56], gestionarea resurselor la nivel de infrastructură și alocarea acestora către aplicații reprezintă o preocupare majoră pentru îmbunătățirea eficienței energetice, precum și a performanței și calității serviciilor oferite de sistemele eterogene. După cum am menționat anterior, sistemele eterogene utilizează un set bogat de tehnologii HW și SW. Diferite tehnologii HW moderne stau la baza creșterii eficienței energetice (mai multă specializare a elementelor de calcul pentru sarcini specifice) și ale performanței. Cu toate acestea, o astfel de eterogenitate creează provocări în gestionarea întregii infrastructuri, deoarece software-ul de *management* trebuie să înțeleagă diferențele dintre diferitele elemente de calcul – acest nivel superior în stiva SW este responsabil pentru gestionarea infrastructurii și alocarea resurselor pentru aplicații.

1.5 Indicatori de performanță pentru implementarea algoritmilor de IA în sisteme de TR

Arhitecturile computaționale folosite pentru implementarea algoritmilor de IA pot fi diverse și fiecare are avantajele sale. Pentru a putea compara în mod obiectiv diferite tipuri de abordări, voi analiza criteriile de decizie asupra soluțiilor optime pentru diferite tipuri de algoritmi si arhitecturi.

În această analiză voi lua în considerare implementarea finală a algoritmilor de IA, fără a insista asupra modului în care se face învățarea/antrenarea.

Criteriile luate în calcul pentru analizarea arhitecturilor computaționale folosite în implementarea algoritmilor de IA sunt exprimate (pe cât posibil măsurabil, cuantizabil, prin "metrici") sub forma următorilor indicatori de perfomanță "cheie" (critici) – KPI *("Key-Performance Indicators"):*

1. Capacitatea de procesare
2. Calitatea algoritmilor
3. Folosirea eficientă a memoriei
4. Energia consumată
5. Aria de siliciu necesară
6. Flexibilitatea în utilizare
7. Securitatea și protecția
8. Riscurile pentru utilizare

Capacitatea de procesare

Capacitatea de procesare (sau "puterea de calcul") disponibilă pentru fiecare tip de arhitectură analizată are o influență majoră asupra alegerii soluției potrivite pentru implementarea algoritmilor de IA.

- Sunt importante nuanțele termenului "putere" în această lucrare, dată fiind și importanța aspectelor energetice – deși "puterea de calcul" reprezintă un termen destul de răspândit, e mai puțin uzitat(ă) în această lucrare decât "puterea consumată" (în acest din urmă context am folosit mai mult "energia", unde a fost posibil).

Aceasta deoarece, pe de o parte, algoritmii de IA au nevoie în mod tradițional de putere de calcul mare, iar pe de altă parte în cazul în care puterea de calcul nu este suficientă, există puține soluții corective.

Capacitatea de procesare se măsoară în moduri diferite [46], însă pentru aplicațiile analizate, cea mai folosită *metrică* este MIPS *(Milions Instructions Per Second)* sau numărul de operații pe secundă.

O altă metrică pentru măsurarea puterii de calcul este FLOPS *(FLoating points Operations Per Second).* Această metrică este mai potrivită pentru algoritmii de IA care folosesc acest tip de operații.

Algoritmii mai noi de *procesare de imagini (PI)* precum și cei de IA folosesc adeseori calcule care implică înmulțiri de matrici. În acest caz, o altă metrică poate fi folosită, și anume **MAC/S** ***(Multiply & ACcumulate operations per Second)***. Aceasta este utilă pentru că de cele mai multe ori o înmulțire este urmată imediat de o adunare într-un acumulator, după formula următoare, care descrie utilizarea generică în *convoluția numerică*. În DSP, MAC se execută ca "operație atomică" – la DSP moderne se pot executa *mai multe MAC pe tact:* $(AB)_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$

În cazul implementării anumitor tipuri de algoritmi (precum cei de rețele neurale) această metrică este foarte folositoare, deoarece majoritatea operațiilor implementate sunt de tipul multiplicare-acumulare. Un număr de arhitecturi computaționale sunt atunci optimizate pentru a livra un număr mare de **MAC** pe secundă (nu mai este cel mai important numărul total de instrucțiuni executate).

Calitatea algoritmilor de IA pentru PI

Deoarece fiecare metodă de implementare a unui algoritm de IA pentru PI poate avea caracteristici diferite, sunt necesare criterii de evaluare a acestor implementări.

Calitatea algoritmilor are aspecte diferite, în funcție de tipul și destinația lor. În contextul de față, calitatea se consideră sub aspectul corectitudinii și relevanței (eficiența este condiționată mai mult de celelalte KPI cum ar fi capacitatea de procesare sau folosirea optimizată a memoriei).

O categorie importantă de algoritmi de PI este bazată pe *clasificare.* Pentru această categorie de algoritmi există două modalități uzuale de măsurare a calității: curba *P-R (Precision-Recall),* respectiv curba *ROC (Receiver Operating Characteristic)* [57], cele două curbe reprezentându-se variind un *parametru* – de exemplu pragul de detecție aferent respectivului algoritm – și rulând de multe ori algoritmul pentru fiecare valoare a parametrului (de exemplu, numărul de rulări poate fi cu cel puțin un ordin de mărime peste inversul toleranței acceptate a calculelor).

Folosirea eficientă a memoriei

Un alt factor important pentru implementarea algoritmilor de IA este modul în care este folosită memoria. Se impun a fi analizate mai multe aspecte referitoare la folosirea memoriei, cele mai importante fiind dimensiunea memoriei disponibile, viteza de acces la memorie și costul acesteia.

Cu cât algoritmii sunt mai simpli, cu atât nevoile de memorie scad și deci se pot folosi memorii mai apropiate de structurile obișnuite de calcul.

O problemă la fel de importantă precum *dimensiunea memoriei*, însă din ce în ce mai stringentă, este *viteza de acces la memorie.* Datorită faptului că algoritmii din domeniul IA au nevoie de cantități mari de date pentru a fi procesate, aceste date trebuie să fie rapid la dispoziția resurselor de calcul, atât la citire cât și la scriere.

Deși memoriile (mai ales cele dinamice) pot îndeplini azi de cele mai multe ori cerințele de dimensiune cerute de cei mai solicitanți algoritmi, viteza de rezolvare a algoritmilor este cel mai adesea limitată de viteza mică cu care datele pot fi aduse aproape de resursele de calcul [63].

Un grafic care arată decalajul între puterea de calcul și viteza de acces la memoria externă, în concordanță cu legea lui Moore, este cel prezentat în Figura 13.

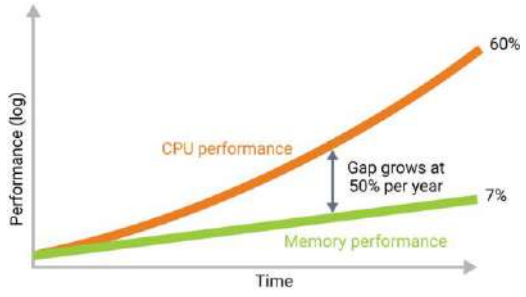


Figura 13: Diferența dintre performanța memoriei și CPU în timp [139]

Datorită faptului că performanțele procesoarelor (dar și ale altor tipuri de resurse computaționale) cresc cu o viteză semnificativ mai mare decât ale memoriilor, în arhitecturile computaționale pentru implementarea algoritmilor de IA se caută compromisuri pentru a compensa acest decalaj.

Memoriile statice si mai ales cele implementate pe registre au viteze de transfer semnificativ mai ridicate, însă în acest caz compromisurile trebuie făcute între viteza de transfer obținută și dimensiunea memoriei (cea care determină prețul).

Energia consumată

În mod tradițional, energia consumată de sistemele de calcul nu era un factor definitoriu în alegerea arhitecturilor potrivite pentru implementările de algoritmi specifici. În această paradigmă, sistemele erau folosite în "centre de calcul" unde puterea consumată (și implicit căldura disipată) puteau fi ținute sub control.

O dată cu proliferarea dispozitivelor mobile (laptops, telefoane, tablete, dispozitive purtabile – *wearables)* acest aspect devine din ce în ce mai important. Arhitecturile computaționale folosite nu trebuie doar să îndeplinească criteriile de

performanță, ci trebuie să permită și o durată de utilizare cât mai lungă a dispozitivelor (în care sunt implementate) în regim de alimentare de la baterii.

Din acest motiv, în momentul actual se caută un echilibru între puterea consumată și performanțe, nu arareori fiind preferată varianta mai puțin performantă în detrimentul celei care consumă mai multă putere.

Aria consumată pe siliciu

Un alt factor important în alegerea resurselor computaționale pentru implementarea algoritmilor de IA este costul implementării. Cel mai important factor în acest sens este dimensiunea fizică a circuitului implementat pe siliciu.

Creșterea ariei ocupate pe siliciu implică (pe lângă costurile aferente) și creșterea de putere consumată. Din acest motiv se evită o folosire excesivă a ariei ocupate pe siliciu, mai ales în dispozitivele mobile.

Pentru evaluarea arhitecturilor computaționale analizate, cea mai comună metrică este aria pe siliciu, măsurată in mm^2. Această metrică este foarte folositoare în estimarea costului de producție, însă variază semnificativ în funcție de un mare număr de factori precum: tehnologia folosită, frecvența de lucru, constrângerile arhitecturale.

Din acest motiv, o metrică ce poate fi utilă și care este mai puțin dependentă de factorii menționați, pentru circuitele electronice digitale, este numărul de porți echivalente folosite *(gatecount)* .

Deși fiecare implementare a unui circuit pe siliciu are constrângerile specifice , folosirea metricii referitoare la numărul de porți echivalente aduce o uniformizare în compararea diferitelor arhitecturi, implementate în tehnologii diferite.

Flexibilitatea în utilizare

Un criteriu important în alegerea arhitecturilor folosite pentru implementarea de algoritmi este dată de flexibilitatea în utilizare a arhitecturii alese. Aceasta se referă la faptul că după alegerea, implementarea și producerea în masă a dispozitivelor care folosesc arhitecturi computaționale pentru IA se mai pot aștepta, totuși, schimbări care să intervină în algoritmii folosiți.

Pentru aceasta, arhitecturile pot avea un grad diferit de flexibilitate, de la modificarea completă a arhitecturii computaționale (așa cum se întâmplă de exemplu cu dispozitivele de tip FPGA) până la imposibilitatea de a face schimbări, de oride fel, în implementarea aleasă (ca la circuitele digitale dedicate ASIC – *Application Specific Integrated Circuits).*

Între cele două extreme pot fi sisteme care nu pot schimba arhitecturile de calcul însă pot schimba atât descrierea (și parametrizarea) algoritmului precum și a datelor de antrenare (exemple sunt arhitecturile de tip CPU, GPU, DSP) sau sisteme *hibride,* în care descrierea algoritmului se poate schimba într-o mică măsură sau deloc, însă datele de antrenare pot fi modificate, pentru o *re-parametrizare / re-instanțiere* îmbunătățită (de exemplu, cu noi coeficienți, optimizați pe diferite criterii).

Alegerea fiecărui model descris are însă impact atât asupra costului soluției precum și (mai ales) asupra puterii consumate. Cu cât arhitecturile alese folosesc resurse mai flexibile, cu atât și costul și puterea consumată cresc.

Securitatea și protecția

În contextul lucrării de față, securitatea implementării algoritmilor de PI în TR se referă modul în care sistemul face față atacurilor din exterior. Se observă că în ultima perioadă aceste atacuri (în general atacuri cibernetice) [67], [68] sunt din ce în ce mai prezente, de aceea o securizare a sistemelor de PI este necesară. Atacurile pot fi folosite atât pentru distrugerea sau compromiterea sistemelor cât și pentru a produce rezultate incorecte (prin atacuri "adversariale") [69], [70]. Aceste aspecte trebuie luate în considerare atât la proiectarea arhitecturii sistemului, cât și la antrenarea algoritmilor de IA.

Pe de altă parte, o atenție sporită trebuie acordată protecției informațiilor manipulate de către sistemele de PI. Și în acest caz avem de a face cu două direcții de abordare. Pe de o parte este necesară protecția datelor personale ale utilizatorilor, așa cum este prevăzută de către legislația în vigoare, ca GDPR [71], cu accent pe metodele de protecție a datelor sensibile

(PII – *Personal Identifier Infromation)* iar pe de altă parte este necesară protecția împotriva copierii algoritmilor de PI dezvoltați.

Pentru protecția datelor personale, se pune din ce în ce mai mult accentul pe modul de proiectare a soluțiilor, prin încorporarea conceptului de *"privacy by design"* [72], [73]. Acest concept se bazează pe principii ca: pro-activitate (și nu doar reacție la probleme), definirea protecției în mod standard (implicit – *"default"),* încorporarea protecției în sistem, vizibilitate și transparență.

Pentru protecția algoritmilor, se pot lua măsuri precum criptarea codului sau "confuzionarea" intenționată *("obfuscation")* a acestuia. Aceste măsuri au sens mai ales pentru algoritmii care necesită un timp de dezvoltare mare, folosesc multe resurse (umane sau de calcul) pentru dezvoltare sau care au nevoie de date pentru antrenare, acestea fiind greu de procurat. În aceste situații algoritmii de PI devin costisitori și sunt ținte ale spionajului industrial.

O extindere a conceptului de "privacy by design" este acela de *"privacy by definition".* Aceasta înseamnă nu doar că modul de proiectare a soluțiilor are în vedere protecția și securitatea, dar și că problema este ridicată la nivelul definirii cerințelor (specificațiilor) soluției. Astfel, se poate avea în vedere abordarea unor soluții în care nu este nevoie de protecția datelor în moduri complicate deoarece datele folosite de sistem (încă de la intrare) sunt astfel achiziționate și manipulate încât să ofere protecția necesară. Exemple în această direcție pot fi legate de înlocuirea senzorilor de imagine pentru anumite sarcini specifice PI (cum ar fi detecția de obiecte sau recunoașterea gesturilor) cu senzori neuromorfici (cum sunt cei de capturare a evenimentelor video) sau de tip radar/lidar.

O discuție a modului în care se poate obține securitatea și protecția datelor poate fi privită și din perspectiva implementării soluțiilor în Cloud sau "on-the-Edge". Astfel, soluțiile în Cloud sunt în general mai expuse la furtul de informații personale, deoarece aceste informații părăsesc dispozitivul de achiziție și atunci există mai multe puncte de vulnerabilitate. Pe de altă parte, dispozitivele mobile sunt mai simple funcțional, de aceea sunt în general mai ușor susceptibile la atacuri [74], [75].

Riscuri pentru utilizare

Un alt factor important de luat în considerare în legătură cu flexibilitatea în utilizare este dat de riscul implicat de anumite tipuri de arhitecturi. Riscul poate fi asociat cu incertitudinea funcționării calitative a algoritmului implementat – în acest caz o soluție mai flexibilă poate avea avantajul de a putea corecta unele erori în algoritmul implementat.

Un alt risc important est dat de implementarea în siliciu a arhitecturilor computaționale. Din acest punct de vedere, cu cât o anumită arhitectură a fost mai des implementată, cu atât riscul este mai mic. Din acest motiv, structurile programabile standard (de tip CPU) prezintă un risc mai scăzut.

Riscul de copiere (legat de proprietatea intelectuală) este un alt factor luat în considerare de industria modernă. Folosirea de arhitecturi care nu sunt protejate suficient din punct de vedere al proprietății intelectuale poate aduce riscuri financiare semnificative. Pe de altă parte, dorința de a proteja proprietatea intelectuală poate determina folosirea de arhitecturi specifice mai puțin "transparente" – mai puțin programabile din exterior.

Dificultatea de implementare tehnologică

Există atâtea soluții foarte interesante care nu au fost implementate în practică datorită dificultăților tehnologice încât am considerat necesar să introducem sus-numitul indicator ca unul de performanță. Acesta este un indicator legat de prețul soluției, însă poate fi mai complex decât atât. Soluțiile care se implementează tehnologic dificil au fiabilitate mai scăzută și rata de defecte mai mare, de aceea costul este mai crescut.

Pe de altă parte, dificultățile tehnologice limitează numărul de companii care pot implementa o anumită soluție sau care doresc să își asume acest risc.

Posibilitățile de integrare între subsisteme care folosesc tehnologii diferite sau atipice sunt limitate, de aceea sunt de evitat soluțiile care folosesc această abordare.

Factorul de merit

Factorul de merit este folosit pentru a măsura calitatea diferitelor soluții care au indici de performanță (KPI) diferiți.

În cazul algoritmilor de TR, factorul de merit propus constă într-un produs de 3 valori,

$$MF = DMF \cdot CMF \cdot WMF$$

unde:

- *MF* – Merit Factor (Factor de merit)
- *DMF* – Disqualifying Merit Factor (factor de merit descalificant)
- *CMF* – Critical Merit Factor (factor de merit critic)
- *WMF* – Weighted Merit Factor (factor de merit ponderat).

Fiecare dintre cele 3 valori contribuie la factorul de merit total, în mod diferit.

Factorul de merit descalificant (DMF)

Factorul de merit descalificant ia în considerare criterii "blocante" pentru soluția de implementat. Există o sumă de criterii care pot descalifica o soluție. Dacă pentru unul din aceste criterii se depășește pragul limită, atunci soluția primește factorul de merit zero ceea ce determină descalificarea ei. Formula de calcul a factorului de merit descalificant este $DMF = \prod_{i=0}^{n} DK_i$, unde DKi – Disqualifying KPI (Indicator de performanță descalificant), care se calculează $DK_i = \begin{cases} 0, K_i > DTh_i \\ 1, K_i \leq DTh_i \end{cases}$, unde

- K_i – KPI (Indicator de performanță)
- DTh_i – Disqualifying Threshold (Prag de descalificare)

Exemple de DTh: Aria maximă de siliciu care poate fi fabricată, energia maximă consumată care poate fi disipată de sistem. Pentru indicatorii de performanță care nu sunt descalificanți se poate considera un "prag infinit".

Factorul de merit critic (CMF)

Factorul de merit critic ia în considerare pragurile critice pentru implementarea soluției de TR. Acesta este o sumă ponderată a indicatorilor de performanță critici $CMF = \frac{\sum_{i=0}^{n} cw_i \cdot CK_i}{\sum_{i=0}^{n} cw_i}$ unde:

- CK – Critical KPI (Indicator de performanță critic)
- Cw – critical weight (Pondere critică)

Iar pentru fiecare indicator de performanță $CK_i = \begin{cases} CC_0, K_i > CTh_{i0} \\ \vdots \\ CC_m, K_i > CTh_{im} \end{cases}$ unde:

- C_0 ... C_m – constante critice
- CTh – Praguri critice

Astfel, la depășirea unor praguri critice pentru fiecare criteriu de performanță critic, factorul de merit critic se ponderează cu o altă constantă.

De exemplu, dacă aria de siliciu depășește 10 mm^2 atunci costul chipului se dublează din cauza modului de decupare și de încapsulare diferit. Dacă puterea consumată trece de 2W, atunci este nevoie de radiator de putere activ, ceea ce crește costul de producție, sau dacă banda de transfer la DDR crește peste 70 Gbps, este nevoie de 2 interfețe DDR externe și de 2 chip-uri de memorie. Pentru indicatorii de performanță care nu sunt critici, ponderea acestora în CMDF este zero.

Ponderea fiecărui KPI este determinată de impactul acestuia în soluția finală.

Factorul de merit ponderat (WMF)

Factorul de merit ponderat reprezintă media ponderată a indicatorilor de performanță măsurați pentru soluția propusă. Spre deosebire de CMF și DMF, WMF variază liniar cu valoarea KPI și este ponderat corespunzător cu importanța acestuia:

$$WMF = \frac{\sum_{i=0}^{n} w_i \cdot K_i}{\sum_{i=0}^{n} w_i}$$ unde:

- K_j – Valoarea indicatorului de performanță
- W_j – ponderea indicatorului de performanță

Sunt indicatori de performanță pentru care nu există o metrică ușor de cuantificat (cum ar fi flexibilitatea în utilizare sau dificultatea de implementare tehnologică). În aceste cazuri, se urmărește totuși alocarea unei valori, cel mai adesea prin notarea KPI (cu note de la 1 la 10). Ponderea fiecărui indicator de performanță este diferită, importanța fiecăruia fiind dată de specificitatea soluției de implementat.

Există scenarii în care capacitatea de procesare reprezintă KPI determinant, pe când în alte scenarii calitatea algoritmilor trebuie ponderată cu costul implementării. Există soluții la care energia consumată reprezintă KPI predominant, urmat de modul de implementare tehnologică și aria de siliciu pentru minimizarea costurilor de producție.

În concluzie, toate criteriile considerate în alegerea arhitecturilor computaționale pentru implementarea algoritmilor de IA prezentate anterior trebuie luate în considerare. Ponderea fiecărui criteriu diferă însă de la o aplicație la alta. Există aplicații în care criteriul definitoriu e costul (cu implicații directe cel mai adesea în performanța și calitatea algoritmilor) iar altele în care viteza este esențială (chiar dacă energia consumată sau costul sunt afectate). Tot mai des se pune însă accentul pe energia consumată ca și criteriu definitoriu în alegerea arhitecturilor computaționale, mai ales pentru dispozitivele mobile. Sunt însă situații în care alte criterii (precum flexibilitatea în utilizare sau minimizarea riscurilor) sunt mai importante.

De aceea o analiză multicriterială se impune de fiecare dată când se aleg arhitecturi computaționale noi pentru aplicații specifice, analiză multicriterială în care ponderea criteriilor este dată de cerințele particulare aferente.

1.6 Sumarul capitolului

În capitolul 1 am prezentat conceptele fundamentale care vor fi folosite pe parcursul lucrării. Pentru analiza sistemelor de TR cu IA am pornit de la definițiile generale ale domeniului, apoi le-am particularizat pentru domeniul cercetării doctorale. Am analizat în detaliu sistemele de TR și am concluzionat care este modelul de lucru ce va fi folosit în continuare.

Am explicat care este impactul procesării în Cloud respectiv Edge pentru sistemele de TR, în particular în perspectiva folosirii aplicațiilor cu IA. În această perspectivă am redefinit modul în care se poate face partiționarea Edge-Cloud pentru sistemele de TR și care sunt avantajele și dezavantajele procesării locale.

O analiză atentă a fost făcută asupra interacțiunii între componentele de nivel înalt ale aplicațiilor de TR cu IA. Ca urmare, am considerat că datele, algoritmii (IA) sistemul SW și cel HW funcționează ca un ecosistem, în care relațiile dintre entități evoluează astfel încât fiecare din ele trebuie să fie adaptabilă pentru "a supraviețui" – pentru a atinge maximul de performanță în sistem. De aceea am detaliat relațiile dintre componentele sistemelor de TR, cu accent pe relația dintre IA și HW, precum și relațiile mai complexe între mai multe componente ale sistemului.

O implicație a relațiilor dintre componentele ecosistemului date-algoritmi-SW-HW este *calculul eterogen.* În această perspectivă am analizat modelele de organizare a sistemelor eterogene și provocările aduse de acestea în PI de TR.

Am concluzionat că pentru o analiză detaliată, atât a soluțiilor existente, cât și a celor propuse este nevoie de o definire corectă și completă a principalilor indicatori de performanță (KPI) specifici arhitecturilor care implementează soluții de TR pentru PI. Pornind de la indicatorii de performanță clasici (PPA) am adăugat și alții, unii propuși pentru prima dată în domeniu, precum flexibilitatea în utilizare sau securitatea și protecția. Toți acești indicatori de performanță sunt dificil de folosit în mod individual, de aceea am propus o metodă de a îi îngloba într-o valoare unică, denumită *factorul de merit* al soluției. Am propus o formulă de calcul care să țină cont atât de cerințele funcționale cât și de cele tehnologice ale soluției.

Capitolul 2. Arhitecturi computaționale folosite în sisteme cu IA

În mod tradițional, rularea algoritmilor se face folosind structuri standard de procesare, cele mai cunoscute fiind cele de tip CPU, DSP sau GPU detaliate în cele ce urmează. Pe lângă arhitecturile generice, voi prezenta în final și arhitecturile specializate, bazate pe FPGA și ASIC. În capitolul de față voi încerca analiza acestor tipuri de arhitecturi doar din punctul de vedere al implementării algoritmilor de tipul celor folosiți în inteligența artificială, fără a insista asupra altor aspecte specifice. Deși orizontul structurilor de calcul standard este extrem de vast, în analiza curentă se vor considera tipurile de arhitecturi care au fost folosite practic și pentru care am mobilizat atât resurse cât și experimente / scenarii de utilizare care să probeze criteriile de performanță analizate și să valideze soluțiile alese.

2.1 Unitatea Centrală de Procesare (CPU)

Aceste arhitecturi sunt cele mai folosite pentru implementarea întregii game de algoritmi și modul natural în care se încearcă diferite abordări. Datorită flexibilității în utilizare și a riscului mic de folosire a acestui tip de resurse, precum și datorită faptului că există resurse disponibile în majoritatea sistemelor în care se doresc implementări de algoritmi, CPU este cea mai utilizată pentru implementarea algoritmilor, inclusiv a celor de inteligență artificială.

Există o gamă largă de CPU disponibile la ora actuală, însă pentru implementări mobile, cele mai folosite arhitecturi sunt: Intel Atom, ARM Cortex, Beyond BA 22 și RISC-V.

Intel Atom

Deși Intel este producătorul cel mai de succes de CPU *(Central Processing Units)*, în lumea dispozitivelor mobile și mai ales în domeniul IoT *(Internet of Things)*, Intel are un succes limitat. Factorul cel mai important în utilizarea acestor CPU vine din familiarizarea întregii comunități academice și industriale cu ecosistemul Intel și, astfel, din ușurința în utilizare a acestor procesoare.

Procesoarele de tip Intel Atom sunt ușor de folosit și integrat, capacitatea de procesare precum și accesul la memorie sunt decente în gama CPU, însă puterea consumată și prețul sunt adeseori prea mari, astfel că raportul performanțe/cost poate limita folosirea în mod eficient în sistemele de calcul pentru implementarea algoritmilor din domeniul inteligenței artificiale.

ARM Cortex

Arhitecturile computaționale dezvoltate de ARM sunt cele mai folosite în dispozitivele *embedded* ("cu calculator încorporat"). Această gamă de procesoare a fost concepută pentru dispozitivele mobile atât din punct de vedere a performanțelor cât și din punct de vedere al puterii consumate.

Deși puterea de procesare este în general mai mică decât a procesoarelor din gama Intel, raportul performanță/putere consumată este cel mai adesea în avantajul procesoarelor din gama ARM. Există o gamă variată de procesoare cu arhitectură ARM, pornind de la cel mai simple sisteme (Cortex M0), care pot fi folosite fără plata drepturilor de autor, până la gama Cortex care e suficient de puternică pentru a rula programe dezvoltate pe baza bibliotecilor de învățare asistată.

Beyond BA22

O alternativă mai puțin costisitoare la procesoarele din gama ARM poate fi oferită de alte arhitecturi proprietare, precum BA22 [174]. Acest tip de procesor poate oferi performanțe comparative cu unele procesoare din seria ARM, însă modalitățile de integrare și de optimizare sunt mai limitate. Deși costul este mai mic, la performanțe asemănătoare, riscul pentru folosirea acestui tip de procesor este mai mare, datorită (pe de o parte) numărului mic de dispozitive existente cât și datorită premiselor adaptării algoritmilor SW pentru această arhitectură.

RISC-V

Un nou set de instrucțiuni, care a fost inițial dezvoltat pentru educație și cercetare este RISC V [175]. Acesta a devenit între timp un standard pentru *arhitectura deschisă* și există numeroase implementări industriale, pornind de la efortul făcut la University of California – Berkeley (inițiativa RISC - Reduced Instruction Set Computer).

Avantajul unei arhitecturi deschise este acela că poate fi folosită cu ușurință. Arhitectura setului de instrucțiuni este modulară, astfel că la bază se folosește un modul de instrucțiuni în virgulă fixă, pe 32 de biți, însă aceasta poate fi extinsă cu secțiunea "M" pentru multiplicări și divizări, "F" pentru virgulă mobilă, simplă precizie, "D" pentru precizie dublă, "Q" pentru precizie quadruplă, "P" pentru instrucțiuni SIMD *(Single Instruction, Multiple Data)* sau "V" pentru operații vectoriale. Toate aceste extensii pot fi folosite pentru optimizarea algoritmilor de IA și pot aduce creșteri de performanță semnificative pentru implementarea acestor algoritmi.

2.2 Procesoare Paralele (GPU, DSP, VPU, NPU)

O altă categorie importantă de arhitecturi computaționale standard sunt cele folosite în mod tradițional pentru algoritmi grafici și anume GPU *(Graphical Processing Units)*.

Deși utilizarea "procesoarelor grafice" GPU nu a fost inițial orientată pentru implementarea algoritmilor de IA, *similaritățile între această clasă de algoritmi și algoritmii de procesare grafică* au dus la utilizarea pe scară din ce în ce mai largă a GPU în IA.

Principalele caracteristici pe care le utilizează GPU și care le fac potrivite pentru algoritmii din domeniul IA sunt numărul mare – de zeci până la mii – de operații executate în paralel și banda mare de acces la memorie (zeci sau sute de Gbps).

O altă categorie de arhitecturi sunt cele de tip DSP *(Digital Signal Processors).* Deși funcționează folosind unități paralele de procesare (la fel ca GPU), DSP sunt mai puțin orientate spre algoritmi grafici și mai mult spre algoritmi de procesare de semnal. Unitățile DSP sunt mai adesea centrate pe implementarea eficientă a *convoluției numerice* bazată pe operațiile de tip Multiplicare & Acumulare în virgulă fixă (și mai puțin pe operații în virgulă mobilă).

Există o multitudine de soluții standard pentru implementări paralele, însă în analiza de față ne vom concentra pe soluțiile: nVidia Pascal, ARM Mali, Imagination PowerVR, AMD APU, Qualcomm Hexagon, Texas Instruments, Movidius VPU, Verisilicon VIP, CEVA DSP, Samsung NPU.

nVidia GPU

Unitățile grafice de la nVidia sunt standardul de facto pentru procesarea algoritmilor de IA la ora actuală. Numărul mare de unități paralele și banda mare de acces la memorie permit implementări în TR ale a algoritmilor de IA (rețele neurale, de exemplu), fără prea mari dificultăți.

Arhitectura nVidia Pascal, apărută în 2016, presupune împărțirea unui GPU într-un număr de GPC *(Graphics Processing Clusters)* la rândul lor compuse dintr-un număr de SM *(Streaming Multiprocessors)* care au un număr de nuclee *("cores")* de procesare dedicate.

Deși performanța acestor GPU este cea mai bună dintre arhitecturile analizate, atât din punct de vedere al puterii de calcul – mii de nuclee CUDA *(Compute Unified Device Architecture)* care merg până la frecvențe de GHz – sau al benzii de acces a memoriei (sute de GB/s), puterea consumată este pe măsură (sute de Wați).

Din acest motiv, aceste arhitecturi nu pot fi utilizate în dispozitive mobile, deși au un mare succes în etapa de cercetare-dezvoltare sau pentru aplicații în Cloud.

Pe ansamblu, bugetul de energie consumată per operație aritmetică pentru GPU poate fi mai bun decât în cazul CPU.

Flexibilitatea în utilizare este relativ mai mică decât pentru CPU, însă medii de programare tip OpenCL sau OpenGL pot utiliza în mod eficient resursele GPU.

Costul acestor arhitecturi este cel mai ridicat dintre opțiunile analizate, în primul rând datorită ariei pe siliciu extrem de mare folosită.

ARM Mali GPU

Varianta *embedded* dezvoltată de ARM pentru GPU se numește Mali, iar gama de GPU de la ARM diferă în funcție de utilizare, de la gama de performanță mare, la gama optimizată pentru aria pe siliciu minimă și până la varianta optimizată pentru putere consumată minimă.

Avantajul major al acestor tipuri de GPU este acela că se integrează ușor cu CPU din gama ARM și funcționează eficient cu NoC de la ARM.

Accesul la memorie se face prin interfața proprietară AXI, care este folosită pe scară largă în dispozitivele SoC actuale, mai ales cele *embedded.*

Puterea consumată de GPU de la ARM e de ordinul Waților, fiind semnificativ mai mică decât variantele performante de la nVidia și de aceea GPU ARM, asftel devenind o posibilă soluție pentru dispozitivele mobile.

Imagination PowerVR GPU

O altă variantă populară de GPU care pot fi folosite în dispozitive mobile pentru accelerarea algoritmilor de IA este dezvoltată de Imagination. GPU în gama PowerVR au performanțe asemănătoare cu ale celor de la ARM și pot fi folosite în aceleași tipuri de aplicații.

Suportul pentru programare OpenCL sau OpenGL reprezintă și el un avantaj pentru asigurarea programării eficiente a resurselor existente.

AMD APU

Spre deosebire de CPU, APU *(Accelerated Processing Unit)* constă dintr-un singur sub-sistem ce integrează în siliciu CPUs și unități grafice, eventual și *cache* mai mici. Această configurație consumă mai puțină energie decât sub-sistemele CPU și GPU discrete.

Există o schemă complicată de control al energiei consumate de GPU, prin reducerea tensiunii de alimentare, controlul frecvenței de ceas și alte mecanisme de control ale puterii consumate (prin oprirea ceasului sau a tensiunii de alimentare). Toate componentele sistemului comunică cu o memorie DDR4 sau LPDDR4 printr-o rețea de comunicare pe chip "Infinity fabric".

Qualcomm Hexagon DSP

Varianta de procesor VLIW *(Very Long Instruction Word)* de la Qualcom are denumirea de Hexagon. Aceasta arhitectură este una de tip DSP, cu suport nativ pentru operații în virgulă fixă însă cele mai noi versiuni permit și programarea în virgulă mobilă.

Pe acest tip de arhitectură computațională, se pot obține diferențe majore între implementările directe și unele atent optimizate ale aceluiași algoritm de IA.

În concluzie, deși performanțele pot fi similare (sau mai scăzute) decât a unui GPU din aceeași clasă pentru implementarea algoritmilor de IA, flexibilitatea în utilizare este mai mică (și productivitatea în programare de asemenea), în schimb avantajul este o putere consumată mai mică.

Texas Instruments DSP

Texas Instruments produc o gamă largă de DSP care pot fi folosite în aplicații care pornesc de la dispozitive purtabile *("wearables")* și până la avionică sau apărare.

La ora actuală, cele mai folosite DSP sunt cele din gama C66, care au performanțe de până la 179 GFLOPS sau 358 GMACS, la puteri consumate sub 10W.

Memoriile externe folosite sunt în general DDR3 pe 64b. Frecvențele de lucru pot trece de 1 GHz.

Pentru algoritmii de IA, mai ales pentru aplicații de vedere artificială în domeniul auto, soluția de la Texas Instrument este denumită EVE *(Embedded Vision Engine)* .

Movidius VPU

Arhitectura computațională de la Movidius este denumită Myriad și este descrisă de dezvoltatori ca un "VPU" *(Vectorial Processing Unit* – unitate de procesare pentru operații vectoriale). Arhitectura este optimizată pentru algoritmi de procesări de imagine.

Procesoarele vectoriale, variabile ca număr, interacționează cu *"Memory Fabric"* (o "țesătură" de celule de memorie) care asigură o bandă de acces foarte mare la memoria locală.

Procesoarele vectoriale, de tip VLIW sunt pe 128 de biți iar operațiile pot fi pe 8, 16 sau 32 de biți. Banda la memorie este de până la 400 GB/s.

Puterea consumată de această arhitectură este mai mică decât a GPU tradiționale și din această cauză este unul dintre puținele GPU care pot fi folosite în dispozitive *wearables* (precum în proiectul Google "Tango").

Verisilicon VIP

Un altă arhitectură de tip GPGPU (General Purpose GPU) este cea de la Verisilicon, denumit VIP – "Vision IP" *(Intellectual Property core)*. Deși nucleul de bază ("core") de la Verisilicon este foarte mic și eficient din punct de vedere al consumului de putere, prin împachetarea unui număr de astfel de componente se pot atinge performanțe de până la ordinul TFLOPS.

Performanța unei astfel de arhitecturi este foarte ridicată, mai ales pentru implementarea de algoritmi de tipul rețelelor neurale, iar puterea consumată este mai bună decât în cazul arhitecturilor mai generale.

CEVA DSP

Un alt producător de DSP utilizate pentru dispozitive *embedded,* cu aplicații în domeniul IA este CEVA. Ultima generație de CEVA DSP, XM4, poate conține, pe lângă unitățile vectoriale, și *unități definite de utilizator.* Acestea pot implementa instrucțiuni dedicate sau pot funcționa ca și procesoare.

Deși în mod nativ arhitectura CEVA nu permite o optimizare foarte eficientă a algoritmilor de IA, prin folosirea unor instrucțiuni dedicate sau a unor coprocesoare pentru operațiile specifice algoritmilor de acest tip (ca multiplicare de matrici sau reorganizare a datelor) se pot obține performanțe decente la bugete de putere competitive.

Samsung NPU

Samsung propune pentru sistemele de tip SoC dedicate telefoanelor mobile o arhitectură de tip NPU [77].

Fiecare unitate NPU are în componență o unitate de pregătire a datelor *(Data fetcher)* care pregătește datele de procesat din memoria de lucru (datele de intrare, ponderi și date intermediare), un modul de control al rarității (ponderilor nenule) "sparsity" (care salvează timp atunci când ponderile sunt nule), un modul de calcul pe *tensori* și un modul de calcul vectorial.

În concluzie, arhitecturile de tip GPU, DSP, VPU sunt cele mai performante arhitecturi dintre cele analizate. Aceasta se datorează, în principal, numărului mare de unități de procesare în paralel disponibile, precum și faptului că arhitecturile sunt optimizate pentru algoritmi de procesare de imagini, foarte asemănători, din punct de vedere al resurselor necesare, cu algoritmii de inteligență artificială.

Accesul la memorie este de asemenea superior arhitecturilor de tip CPU, nevoia de date a sistemelor de tip SIMD sau VLIW fiind acoperită de magistrale (bus-uri) "late" sau memorii locale foarte mari.

Puterea consumată este în general mare, mai mare decât a sistemelor cu CPU, însă raportul performanță/putere este în favoarea arhitecturilor de tip GPU.

Flexibilitatea în utilizare este păstrată, prin păstrarea paradigmelor de programare, însă rezultate optime se pot realiza prin folosirea unor extensii de programare de tip OpenCL, OpenGL sau mai nou OpenVX.

Există un număr de arhitecturi care pot fi programate direct din medii de dezvoltare a rețelelor neurale, precum Caffe sau TensorFlow. Aceasta simplifică "modul de livrare" a algoritmilor de acest fel.

Pentru obținerea celor mai bune rezultate, programarea arhitecturilor de tip GPU trebuie însă făcută cu atenție la detaliile de implementare HW, de aceea flexibilitatea în modificarea algoritmilor este mai mică decât în cazul CPU.

Cel mai mare dezavantaj al sistemelor de tip paralel este însă costul. GPU sau DSP ocupă o arie mare de siliciu, de aceea și costul de producție este inevitabil mare. La aceasta se adaugă și costurile necesare conectării optime a arhitecturilor de acest tip la sistem, cerințele de interconectare fiind ridicate, ceea ce duce la un cost total al soluției, în general, mare.

2.3 Arhitecturi computaționale specializate folosite în IA

Arhitecturile computaționale standard beneficiază de avantajul de a fi folosite într-o gamă largă de aplicații fără modificări HW, *programabilitatea* acestor arhitecturi fiind un avantaj. Pentru implementarea de sisteme în care algoritmii folosiți sunt pre-definiți, iar modificarea acestora nu se impune, se pot găsi arhitecturi computaționale specializate, care pot implementa în mod mai eficient anumiți algoritmi specifici sau anumite clase de algoritmi. În zona inteligenței artificiale aceste arhitecturi computaționale pot fi implementate pe FPGA sau ASIC.

Există de asemenea posibilitatea de dezvoltare a unor *IP cores* HW care pot implementa algoritmi sau părți de algoritmi în sisteme de tip SoC.

Implementări centrate pe FPGA pentru algoritmi de IA

Arhitecturile computaționale folosite pentru implementare pe FPGA (Field Programmable Gate Array) au avantajul flexibilității și a capacității de calcul mare (specifice dispozitivelor FPGA). Arhitectura FPGA diferă destul de mult de la dispozitiv la dispozitiv, însă cei mai cunoscuți producători, care aduc pe piață dispozitive capabile de a rula algoritmi de IA sunt Xilinx și Altera (acum Intel). De la Xilinx voi prezenta dispozitivele Zynq (și UltraScale+), iar de la Intel Arria 10 (cu extensia Intel® Deep Learning Inference Accelerator).

Xilinx Zynq

Structura internă a unui dispozitiv de tip Xilinx Zynq include o componentă programabilă *(PL – Programmable Logic)* și o componentă de sistem *(PS – Processing System).* Dacă componenta programabilă include logică programabilă (propriuzisă), DSP și memorie RAM, componenta de sistem include procesoare (unul sau mai multe), controlere pentru memoria externă RAM și alte periferice.

În conjuncție cu bibliotecile de IA dezvoltate de același producător (precum Xilinx Deep Neural Network library – xDNN), o aplicație poate fi implementată rapid și fără constrângeri majore de proiectare.

Intel Arria 10

Dispozitivele Arria de la Intel (Altera) sunt echivalentul celor de la Xilinx Zynq. De asemenea, există o componentă de sistem implementată în siliciu, care conține pe lângă un procesor *multi-core*, memorie *cache* și periferice pentru controlul și optimizarea algoritmilor implementați în partea programabilă.

Componentele programabile, LAB *(Logic Array Block)*, sunt compuse din blocuri de bază ALM *(Adaptive Logic Modules).* Acestea pot implementa funcții logice sau aritmetice precum și registre.

Un sfert din componentele LAB de pe dispozitiv pot fi folosite ca memorii (MLAB).

În concluzie, dacă din punct de vedere a capacității de calcul, dispozitivele FPGA sunt extrem de generoase – ca și din punct de vedere al accesului la memorie și al capacității acesteia – energia consumată este factorul care limitează folosirea dispozitivelor FPGA pe scală largă, alături de prețul prohibitiv.

Din punct de vedere al acestui consum de putere, adeseori puterea FPGA este mai mare decât a GPU cu o capacitate de calcul echivalentă. Prețul dispozitivelor de mare capacitate este în jurul a zecilor de mii de dolari.

Riscul pentru folosirea acestor dispozitive este minim. Atât algoritmii cât și resursele computaționale folosite se pot schimba rapid pentru a corecta probleme potențiale sau pentru a se adapta la noi cerințe.

ASIC pentru IA

O dată cu creșterea spectaculoasă a aplicațiilor din domeniul IA, s-a impus ca exploatarea capacității de calcul necesare să se facă cu circuite specifice, dedicate funcției de implementat. Din acest motiv au apărut mai multe circuite ASIC *(Application Specific Integrated Circuits)* care pot fi folosite în acest domeniu. Un promotor al acestor tipuri de arhitecturi computaționale este Google, care a construit primul procesor pentru rețele neurale folosit pe scară largă, TPU *(Tensor Processing Unit).* IBM a construit o serie de alte circuite, cel mai nou fiind TrueNorth. Intel, dupa achiziția Nervana, a creat un procesor dedicat pentru IA, numit Lake Crest.

Aceste circuite integrate sunt doar cele mai cunoscute în domeniul IA, o serie de alte companii făcând eforturi susținute pentru dezvoltarea de noi circuite integrate pentru aceste aplicații.

Google Tensor Processing Unit

Google a construit TPU după ce a folosit în mod extensiv atât arhitecturi standard, de tip GPU cât și arhitecturi programabile (FPGA) pentru a rezolva probleme dificile în domeniul IA. În urma acestor experimente, a fost conceput un circuit integrat dedicat, care are ca funcție principală accelerarea multiplicării de matrice

Din punct de vedere al performanțelor de calcul, TPU poate procesa 64K MAC pe ciclu. Memoria locală este mare, de 24 MB. Pentru a procesa și apoi stoca o astfel de cantitate de date, accesul la memoriile externe (DDR3) se face cu viteze de 30GB/s. Aritmetica în acest caz este pe 8 biți, ceea ce este suficient pentru cea mai mare parte a algoritmilor de IA folosiți. Logica de activare a neuronilor este însă pe 16 biți. La nevoie, se poate alege ca toată aritmetica să se execute pe 16 biți, cu scăderea corespunzătoare a performanțelor.

IBM TrueNorth

Un precursor al procesoarelor pentru IA de la Google a fost dezvoltat de IBM – True North, un circuit integrat specializat pentru rularea de rețele neurale.

Acest chip, dezvoltat in 2011, este inspirat foarte mult de biologie (”bio-inspirat”) și folosește principii asemănătoare cu ale acesteia. TrueNorth este o arhitectură non-von Neumann, modulară, paralelă, distribuită, scalabilă, inspirată de creierul uman. Circuitul are un număr de 256 de neuroni cu funcții de activare complexe și 1024x256 biți de SRAM ca memorie pentru sinapse.

Intel Lake Crest / Sprint Crest/ Spring Hill

Intel este o companie serios implicată în dezvoltarea de soluții pentru IA. După preluarea companiei Nervana, produsul acestora pentru rularea de algoritmi de IA a fost prezentat sub numele Lake Crest.

Informațiile despre acest chip sunt destul de lacunare, însă ambiția Intel este să dezvolte aceste chip-uri în conexiune cu CPU, astfel încât sa reducă timpul de rulare a algoritmilor IA de până la 100 de ori față de GPU.

Memoriile folosite sunt de tip HBM2 *(High Bandwidth Memory type 2),* de până la 12 ori mai rapide decât DDR4. Cei 32 GB de memorie HBM2, vor putea genera până la 1TB/s bandă de acces. Conexiunea între chip-uri va fi de tip nou, de 20 ori mai rapidă decât PCIe *(Peripheral Component Interconnect express).*

În concluzie, circuitele integrate specifice pentru rularea algoritmilor de IA au fost dezvoltate pentru a rula un subset de astfel de algoritmi. Datorită interesului crescut în domeniul rețelelor neurale, cele mai populare circuite integrate accelerează astfel de algoritmi.

Performanțele circuitelor dedicate sunt de neatins pentru alte tipuri de arhitecturi computaționale. Aceasta e valabil atât în cazul puterii de procesare cât și al accesului la memorie. Energia consumată este de asemenea mai mică per operație aritmetică decât la alte opțiuni (comparația între TPU si GPU arată o energie consumată de 40 de ori mai mică pentru aceleași operații).

Deși prețul pentru dezvoltarea unui circuit integrat dedicat este mare, de ordinul milioanelor de dolari, prețul final pe unitate este relativ mic pentru dispozitive produse pe scară largă. De aceea, dezvoltarea de astfel de circuite are sens doar dacă se folosesc în aplicații de masă.

Lipsa flexibilității în utilizare este un dezavantaj al acestor circuite. După turnarea în siliciu, modificările care se mai pot aduce arhitecturii computaționale sunt minime. De aceea o analiză atentă a cerințelor de proiectare este obligatorie, precum și o estimare a modificărilor de suportat pe timpul duratei de viață a produsului.

Riscul este un alt factor care dezavantajează aceste arhitecturi computaționale. Pe lângă riscul dat de eventualele defecte în procesul de proiectare, un risc major ține de evoluția extrem de rapidă a tehnologiei în domeniul IA. Din acest motiv, se poate întâmpla ca un circuit care a fost proiectat și produs corect să nu mai fie folosit, atunci când noii algoritmi se îndepărtează de la direcția de proiectare inițială.

2.4 Sumarul capitolului

În acest capitol am analizat diferitele tipuri de arhitecturi computaționale folosite pentru implementarea algoritmilor de inteligență artificială. Această analiză s-a făcut în funcțiile de criteriile considerate a fi cele mai importante pentru implementarea algoritmilor vizați (aprecierea importanței s-a bazat și pe experiența acumulată în industrie).

Deși de cele mai multe ori atunci când se dezvoltă un algoritm nou atenția se focalizează doar asupra calității algoritmului, în cazul în care acest algoritm va fi implementat pentru uz industrial, trebuie făcut un *compromis* între calitate și un număr de alte criterii.

Aceste criterii au fost analizate în detaliu și anume: capacitatea de calcul, accesul la memorie, energia consumată, aria ocupată pe siliciu, flexibilitatea în utilizare și riscurile de securitate.

Pentru fiecare din aceste criterii s-a urmărit detalierea care să permită aplicarea pentru fiecare tip dintre arhitecturile computaționale analizate.

Dintre aceste criterii, flexibilitatea în utilizare și riscurile implicate de utilizarea anumitor arhitecturi apar arareori în alte analize din literatura de specialitate. S-a observat că ponderea criteriilor de alegere a soluțiilor de dezvoltare este diferită de la aplicație la aplicație, de aceea am propus folosirea unor analize multi-criteriale în alegerea arhitecturilor computaționale optime, unde ponderile criteriilor să corespundă cerințelor aplicației.

Arhitecturile computaționale standard analizate au fost grupate în două categorii: procesoare de uz general (CPU) și procesoare paralele (GPU, DSP, VPU).

Procesoarele analizate au fost în principal din gama care poate fi folosită în dispozitive mobile, care pot permite implementarea algoritmilor IA *on the Edge*. Dintre cele mai folosite CPU au fost analizate cele de la Intel (Atom), ARM (Cortex), Beyond (BA22) și arhitectura deschisă RISC-V. Prezentarea a fost orientată pe performanțele relativ la algoritmii de IA fiind identificate avantajele și dezavantajele fiecărei astfel de arhitecturi. A rezultat că CPU standard sunt soluții foarte flexibile pentru implementarea algoritmilor de IA și au riscul cel mai scăzut în utilizare dintre toate dispozitivele analizate. Dezavantajele acestor arhitecturi constau în energia consumată mare și capacitatea de calcul mai mică, aceasta din cauză că arhitecturile CPU standard nu sunt optimizate pentru algoritmi de IA.

Procesoarele paralele au tot arhitecturi standard și nu sunt destinate în mod explicit aplicațiilor de IA. Există o serie de arhitecturi paralele, grupate în aceeași secțiune, astfel încât am analizat împreună arhitecturi de tip GPU cu DSP sau VPU. Arhitecturile analizate au fost nVidia GPU, ARM Mali GPU, Imagination PowerVR GPU, Qualcomm Hexagon DSP, Texas

Instruments EVE DSP, Movidius VPU, Verisilicon VIP și CEVA DSP. Cele mai multe arhitecturi prezentate sunt de forma VLIW sau SIMD și de aceea sunt optime pentru implementarea algoritmilor de IA, care necesită un număr mare de repetări ale acelorași operații pentru date distincte.

Rezultatul analizei este că – pentru algoritmi de IA – arhitecturile paralele au performanțe mult mai bune comparativ cu procesoarelor clasice, flexibilitatea în utilizare scade puțin datorită cerințelor de programare optimală a resurselor, iar riscurile în utilizare sunt mici.

Cel mai mare dezavantaj al acestor arhitecturi computaționale constă în energia mare consumată. De asemenea, aria ocupată pe siliciu este un dezavantaj, ceea ce duce la costuri mari ale acestor arhitecturi.

O altă clasă de arhitecturi computaționale analizate au fost cele specializate. Aici au fost luate în considerare implementări de algoritmi de IA pe dispozitive de tip FPGA sau pe circuite integrate dedicate de tip ASIC.

Dintre dispozitivele FPGA analizate, pentru implementarea de algoritmi IA au fost considerate dispozitivele Zynq de la Xilinx și Arria 10 de la Intel (fost Altera).

Ambele arhitecturi cuprind sub-sisteme cu funcționalitate fixă (unități de control, CPU, periferice) precum și logică programabilă care poate fi ușor folosită pentru algoritmii IA. Specifică pentru implementarea acestor algoritmi este opțiunea de folosire a componentelor DSP din interiorul logicii programabile, care aduce un plus de performanță implementărilor.

Din analiza efectuată a rezultat că avantajul acestor arhitecturi este flexibilitatea în utilizare și riscul scăzut, datorită opțiunii de reprogramare completă a dispozitivului (la nivel de bit). Un dezavantaj major este energia consumată, mare în comparație cu toate celelalte arhitecturi analizate. Capacitatea de calcul poate fi mare iar eficiența utilizării memoriei este un avantaj al acestor arhitecturi doar dacă se folosește memoria locală, de pe chip.

Ultima clasă de arhitecturi analizate au fost cele implementate în circuite integrate specializate. Dintre acestea, au fost urmărite chip-urile TPU de la Google, TrueNorth de la IBM și LakeCrest de la Intel. Toate aceste arhitecturi s-au remarcat prin capacitate de calcul mare, foarte specializată pentru implementarea algoritmilor de IA. De asemenea, accesul la memorie este optimal, în raport cu celelalte arhitecturi analizate, atât la memoria locală cât și la cea externă. Consumul energetic este un alt avantaj major al acestor arhitecturi, datorită optimizărilor specifice pentru algoritmii implementați. Aria ocupată pe siliciu este însă mare și costul acestor dispozitive poate fi foarte mare, mai ales dacă volumele de producție nu sunt extrem de mari. Pe lângă cost, lipsa flexibilității în utilizare este un alt dezavantaj al circuitelor de tip ASIC, modificarea arhitecturii sau folosirea acesteia în modalități care nu au fost gândite de la început fiind improbabilă. Riscul este mare, atât din punctul de vedere al proiectării dispozitivelor cât și al algoritmilor folosiți.

Din analiza tuturor acestor arhitecturi computaționale, a rezultat că nu există câștigători clari din punct de vedere al criteriilor analizate iar fiecare model arhitectural are plusurile și minusurile sale.

De aceea, arhitecturile computaționale hibride sau eterogene merită o atenție deosebită – ele vor fi abordate în capitolele următoare. Considerăm că aceste arhitecturi pot echilibra balanța între avantajele și limitările arhitecturilor descrise anterior. Se impune și aprofundarea procesării de imagini în timp real, unde acest tip de arhitecturi au un mare potențial. Din acest motiv, cercetarea doctorală prezentată în continuare încearcă abordarea hibridă a arhitecturilor computaționale pentru implementarea de algoritmi specifici domeniului IA.

Capitolul 3. Accelerarea HW pentru PI de TR

La ora actuală există un număr extrem de mare de senzori de imagine pe piață, cu o rată de creștere de peste 10 miliarde de senzori noi în fiecare an. Din acest motiv, sistemele pentru procesarea imaginilor (PI) de timp real (TR) sunt foarte răspândite. Aplicațiile variază de la camere foto digitale, la domeniul auto sau militar, al securității sau medical și până la divertisment și IoT ("Internet of Things" – Internetul Obiectelor).

Achiziția și procesarea de imagini în TR a devenit deci o problemă comună, având rezolvări standard care și-au dovedit valabilitatea în timp. O dată cu sporirea cerințelor, în special prin creșterea dimensiunilor imaginilor procesate și a vitezei de achiziție a acestora (măsurată în numărul de cadre pe secundă) precum și prin introducerea de elemente noi în fluxul de PI (în special algoritmi de IA pentru PI), complexitatea sistemelor de PI în TR a crescut, iar metodele standard de procesare au devenit mai puțin eficiente.

În acest capitol voi prezenta sistemele tipice pentru PI de TR, pornind de la principalele provocări pe care le aduc cerințele de creștere a complexității.

Voi propune apoi o soluție specială de abordare, în care relația dintre componentele SW și HW este mult mai strânsă, astfel încât să fie depășite principalele dificultăți de implementare existente în sistemele tipice. Această abordare, hibridă SW-HW, pornește de la principiile prezentate în capitolul 1, în care sistemele de TR sunt privite ca eco-sisteme având intercorelări foarte strânse între componente. În cadrul acestei propuneri, voi analiza indicatorii de performanță specifici (prezentați de asemenea în capitolul 1). Soluțiile propuse vor fi apoi detaliate în capitolul 4 (în care IA ajută implementările HW) și în capitolul 5 (în care HW ajută la implementarea mai eficientă a IA).

3.1 Sisteme tipice pentru PI de TR

Un sistem tipic de PI arată ca și cel prezentat în Figura 14.

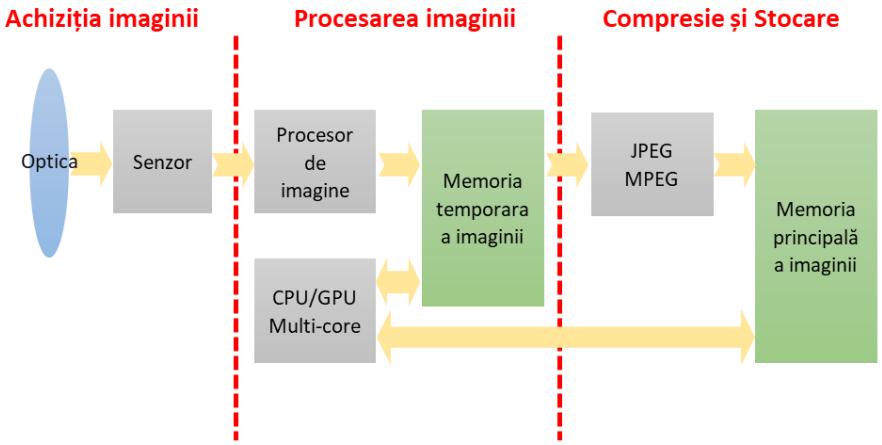


Figura 14: Sistem tipic pentru PI de TR

Indicatori de performanță critici (KPI)

Capacitatea de procesare

Capacitatea de procesare este dată, în sistemele tipice, de capacitatea CPU/GPU. Din acest motiv se încearcă utilizarea unor sisteme de calcul cât mai performante, de tip multi-core, care să rezolve nevoile computaționale curente.

Calitatea algoritmilor

Există două componente critice care afectează calitatea sistemelor de PI în TR tipice. În primul rând, este esențială calitatea ISP, care determină în cele din urmă calitatea imaginii de la ieșirea sistemului.

În al doilea rând, calitatea algoritmilor implementați în SW este esențială pentru rezolvarea problemelor conexe celei de calitate a imaginii (cum ar fi acelea de control în TR, comunicație sau procesare avansată a imaginilor).

Consumul de energie

Pentru un sistem mobil, care folosește un sistem de achiziție a imaginilor precum cel prezentat, consumul de energie este generat de diferite aplicații.Cei mai solicitanți pentru puterea consumată sunt algoritmii folosiți pentru achiziția de imagini (foto și video). În top 3 mai intră și jocurile care folosesc intensiv resurse de calcul, precum cele 3D.

Se observă deci nevoia de diminuare a consumului de energie pentru PI, iar folosirea pe scară largă a GPU nu rezolvă această problemă, așa cum s-a arătat anterior.

Folosirea eficientă a memoriei

În perspectiva în care dimensiunea imaginilor procesate crește continuu, în paralel cu numărul de cadre procesate pe secundă, banda la memorie pentru sistemul de PI devine foarte mare. În Figura 15 se explică problema provocărilor date de banda de transfer la memorie pentru sistemul de PI.

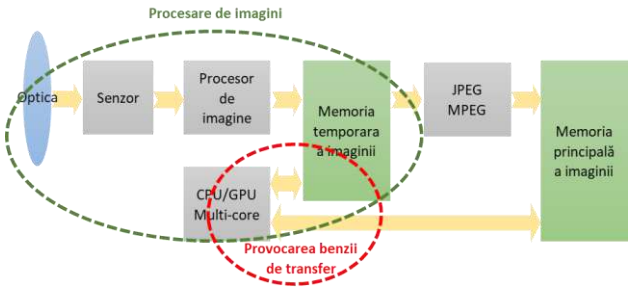


Figura 15: Provocarea benzii transfer la memorie

Sistemul de PI conține atât procesorul de imagine (ISP) cât și sistemul de uz general compus din CPU și/sau GPU. Datele sunt stocate și refolosite din memoria temporară (SDRAM).

Această abordare presupune că pentru fiecare cadru este necesară cel puțin o scriere și o citire din memorie, operație care uneori depășește banda disponibilă la memoria externă.

Chiar și doar citirea imaginii și scrierea rezultatelor în memorie devine o provocare, însă majoritatea traficului algoritmilor IA este dat de datele intermediare și respectiv de parametrii modelului IA.

Aria consumată pe siliciu

Așa cum s-a arătat anterior, costul circuitului integrat este direct proporțional cu aria de siliciu consumată. Pentru sistemele de PI în TR, majoritatea ariei de siliciu este ocupată de ISP urmat de sistemul de calcul (CPU/GPU).

Motivul pentru care în mod tradițional ISP consumă o arie semnificativă este dat de necesitatea acestuia de a implementa numeroase memorii tampon, pentru păstrarea datelor intermediare necesare PI.

Considerând memoriile interne ale ISP, dar și memoria cache a sistemului de calcul de uz general, se poate concluziona că circuitele integrate pentru PI sunt centrate pe memorie, peste 50% din aria de siliciu fiind consumată cu blocuri de memorie diferite.

3.2 Soluția hibridă propusă pentru PI de TR

Așa cum am arătat anterior, în domeniul PI, provocările sunt date pe de o parte de creșterea performanțelor sistemelor de achiziție de date (senzorii de imagine pot acum să capteze rezoluții de până la 41M pixeli sau video la rezoluție 8K la 120 de cadre pe secundă) iar, pe de altă parte, de complexitatea în creștere a algoritmilor de PI. În plus, din ce în ce mai des se întâlnesc sisteme multi-cameră sau multi-optică, ceea ce duce, o dată în plus, la redimensionarea cerințelor de procesare.

Pentru rezolvarea celor două probleme fundamentale în PI pentru dispozitivele mobile existente, am propus o abordare hibridă, prezentată în Figura 16.

Această abordare presupune existența unei componente HW care este integrată în interiorul ISP, pentru acces rapid și eficient la imagini. De asemenea, există o componentă SW, care folosește rezultatele generate de HW și definitivează algoritmul de implementat.

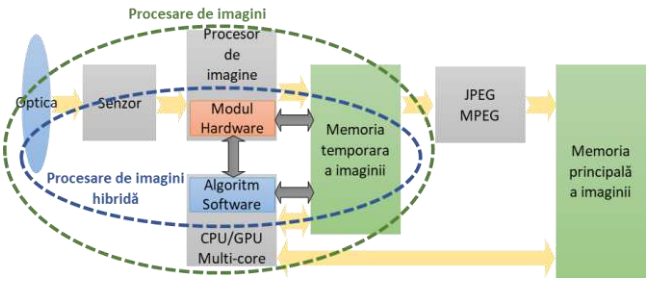


Figura 16: Soluție de procesare de imagini hibridă SW-HW

Astfel, algoritmii de PI complecși nu mai au o implementare pur SW, în care sistemul de calcul (CPU/GPU) este exclusiv responsabil pentru rularea acestora, ci o parte din necesarul de procesare este mutat în componente HW dedicate, care funcționează la *comanda* și în *configurația* definită de SW.

Modulul HW poate fi strâns cuplat cu celelalte componente ale fluxului de procesare de date, prin interfețe HW dedicate, care pot maximiza traficul de date necesare, asigurând o interacțiune foarte rapidă cu sistemul de TR. Memoria temporară este folosită în principal pentru a stoca rezultatele finale ale procesării HW, rezultatele intermediare fiind de cele mai multe ori păstrate în memorii interne ale blocurilor HW.

Algoritmul SW "vede" modulul HW ca un *accelerator pentru funcții specifice,* care se configurează corespunzător nevoilor algoritmului. Algoritmul SW folosește rezultatele modulului HW, în conformitate cu nevoile proprii, pentru *definitivarea* algoritmului de PI.

Avantajele acestei soluții sunt date de puterea de calcul scăzută necesară algoritmului SW, banda la memorie semnificativ scăzută, energia consumată mai mică, iar principalul dezavantaj este dat de flexibilitatea în utilizare.

Relația dintre componentele HW și SW

În Figura 17 se face o prezentare a conceptului de soluție hibridă HW-SW – cu detalierea interfețelor dintre modulele HW și SW.

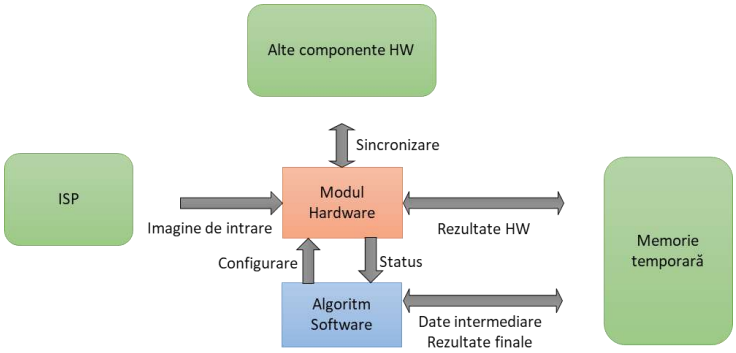


Figura 17: Interacțiunea dintre HW și SW în sisteme hibride

Modulul HW primește datele de intrare sub formă de imagine de intrare, pe interfețe HW specifice ISP. În general, imaginea este transmisă în format *raster,* cu viteza de 1 pixel pe fiecare tact de ceas (în sistemele cu viteze mari de achiziție se pot întâlni interfețe HW cu viteze mai mari de 1 pixel pe ceas). Această interfață este la nivel fizic, și nu necesită resurse computaționale sau trafic la memorie care să afecteze funcționarea sistemului de PI.

Opțional, se pot imagina soluții în care modulul HW se sincronizează cu alte module HW din fluxul de PI, pentru a asigura funcționarea corectă a sistemului. Asftel de module pot fi cele de interfață cu alți senzori (giroscop, accelerometru, senzori de mișcare, etc) sau alte module de procesare (pentru alți algoritmi de PI implementați în sistem).

Algoritmul SW își preia datele de la modulul HW, însă are nevoie să stocheze și propriile date intermediare sau finale. Aceste date sunt stocate în memoria temporară a sistemului (de tip SRAM sau DRAM). Așa cum s-a arătat anterior, accesul la memoria temporară est *arbitrat* între toate componentele HW care au acces la memorie și sub-sistemul SW și reprezintă principala provocare a proiectării sistemelor de PI în TR.

Partiționarea HW-SW a algoritmului

O etapă importantă a modului de implementare hibridă HW-SW a unui algoritm este *partiționarea optimală* a componentelor HW și SW. În contextul lucrării de față, modul optim de implementare presupune maximizarea *factorului de merit* al soluției propuse. De aceea trebuie luați în considerare toți indicatorii de performanță relevanți (KPI – "Key Perfromance Indicators") pentru implementarea soluției.

Mutarea graniței dintre SW și HW se poate face teoretic între două extreme:

- Întreg algoritmul este implementat în SW, datele de intrare și cele de ieșire sunt în memorie iar pentru procesare sunt folosite doar resursele generice de calcul (CPU/GPU)
- Întreg algoritmul este implementat în HW, datele de intrare și cele de ieșire se transmit prin interfețe specifice, iar pentru calcul se folosesc doar resurse HW dedicate.

Între cele două extreme, se poate imagina un *continuum* de soluții, în care o parte a funcționalității SW este mutată în HW. Criteriile după care se poate decide care funcționalitate este mutată din SW în HW, este, de asemenea, definită de indicatorii de performanță relevanți (KPI).

Datorită modului de interacțiune între componentele SW și HW prezentat anterior, se dorește o *minimizare a schimbului de date* între SW și HW. Pentru aceasta, se urmărește definirea cât mai *independentă* posibil a funcțiilor implementate în HW. Aceasta înseamnă că o dată ce modulul HW este configurat corect și pornit, acesta să poată genera ieșirile dorite cu *un minimum de întreruperi suplimentare* ale sub-sistemului SW.

De aceea, primul pas în partiționarea HW-SW îl reprezintă separarea (”spargerea”) algoritmului de implementat în funcțiuni bine definite (”atomizate”). Exemple de astfel de funcțiuni pot fi: manipularea imaginilor (scalare, rotire), transformate ale datelor – transformarea spațiului de culoare (DCT – ” Discrete Cosine Transform”, SIFT – ”Scale-Invariant Feature Transform”, transformare pentru a extrage caracteristicile invariante la scalare etc) sau specifice algoritmului de implementat (filtrări, comparări cu șabloane, operații cu matrici, convoluții, operații pe grafuri). Aceste funcțiuni sunt conectate între ele printr-o ”logică de control” (”control logic”).

O dată ce funcțiunile algoritmului sunt definite, se poate trece la analiza resurselor folosite de fiecare funcție în parte *(profilare).* Profilarea se realizează în scenariul în care toate funcțiile sunt implementate în SW și servește ca un reper (”baseline” – ”referință”) pentru deciziile de implementare următoare.

Cele mai bune candidate pentru implementare HW sunt funcțiile care consumă resurse de calcul semnificative atunci când sunt implementate în SW, au o interfață simplă cu celelalte funcții din sistem și eventual pot fi re-utilizate pentru o gamă mai largă de aplicații.

3.3 Potențialul IA în sistemele hibride HW-SW

Pentru optimizarea funcționării sistemelor hibride, HW-SW, folosirea algoritmilor de IA poate ajuta în diferite moduri.

Pe de o parte, algoritmii de IA pot optimiza controlul modului în care componentele HW sunt utilizate, configurate și sincronizate. Pe de altă parte, modul în care se implementează algoritmii de IA în soluțiile de TR prezintă provocări importante, de aceea și modul de implementare a acestor algoritmi poate fi optimizat (așa cum voi arăta în capitolul 5).

În prima categorie, am identificat următoarele metode de control al componentelor HW de către algoritmii de IA:

- Integrarea adaptivă a resurselor HW (decizie asistată de IA asupra configurației sistemului HW);
- Parametrizarea HW de către IA (programarea adaptivă prin instanțiere anticipativă la nivel de modul HW);
- Controlul preciziei calculelor implementate în HW (toleranța la erori sporită de IA);
- Controlul prin IA al fluxului de date din HW (fuziunea datelor secundare cu meta-date din fluxul principal);

Din a doua categorie, am analizat următoarele metode de optimizare a implementării algoritmilor de IA în HW:

- Optimizarea capacității de procesare;
- Utilizarea eficientă a magistralelor de date.

IA pentru HW - Integrarea adaptivă a resurselor HW

În cazul în care există o multitudine de resurse HW la dispoziție, acestea pot fi utilizate astfel încât să răspundă cerințelor specifice. Astfel, algoritmul IA poate decide folosirea nu în mod necesar a resurselor HW care generează rezultatele cele mai de calitate, ci uneori a resurselor HW care execută sarcinile cel mai rapid sau care consumă cea mai puțină energie.

Această libertate de alegere a resurselor HW folosite poate fi disponibilă în sisteme în care există componente HW diverse, care pot executa funcții controlate de SW în mod diferit. Mai mult, se poate astfel ajunge la implementarea în HW a unor resurse care nu sunt folosite decât în scenariile definite în mod expres de IA.

Figura 18 ilustrează o arhitectură matriceală sistolică omogenă *(homogeneous systolic array architecture).* Paradigma matricei sistolice, în care fluxul de date este procesat ritmic la comanda unui *data-counter* (contor de date) este echivalentul paradigmei von Neuman, în care instrucțiunile sunt executate la comanda unui *program-counter* (contor pentru program). Deoarece o matrice sistolică trimite și primește de obicei mai multe fluxuri de date (precum fluxurile sanguine în *sistolă)*, iar mai multe contoare de date sunt utilizate pentru a controla aceste fluxuri de date, paralelismul de date este implicit.

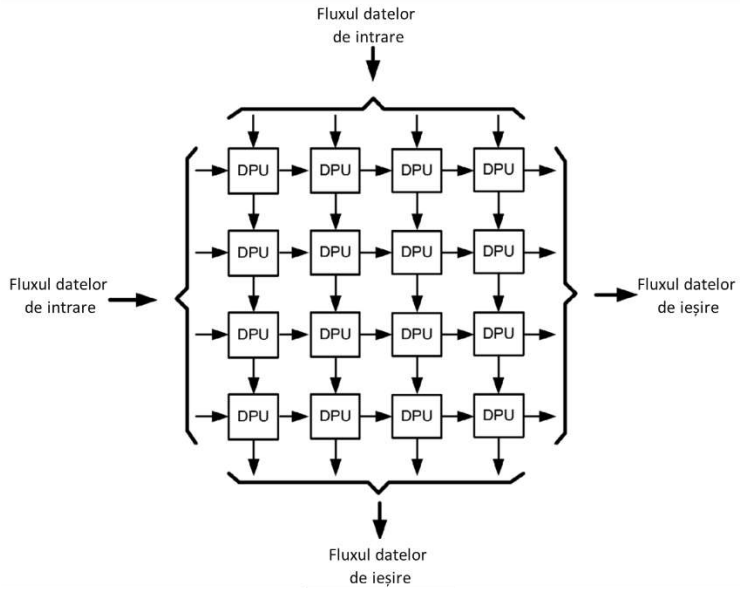


Figura 18: Arhitectura unei matrice sistolice omogene

O matrice sistolică este compusă din rânduri de tip matrice de unități de procesare a datelor numite celule. Unitățile de procesare a datelor (DPU - *Data Processing Unit)* sunt similare cu unitățile centrale de procesare (CPU), cu excepția lipsei unui contor de programe, deoarece operațiunea este declanșată de *transport,* adică prin sosirea unei unități de date. Fiecare celulă partajează informațiile cu vecinii săi imediat după procesare. Matricea sistolică este adesea dreptunghiulară sau are celulele aranjate în coloane și / sau rânduri în care datele circulă în matrice între DPU vecine, adesea cu date diferite care circulă în direcții diferite. Figura 18 ilustrează un astfel de exemplu de arhitectură matriceală sistolică omogenă.

Fluxurile de date care intră și ies din porturile matricei sunt generate de unități de memorie auto-secvențiale (ASM - *Auto Sequencing Memory).* Fiecare ASM include un contor de date. Fluxurile de date pot fi intrări sau ieșiri externe.

Matricele sistolice pot include șiruri de DPUs care sunt conectate la un număr mic de DPU vecine mai apropiate, într-o topologie asemănătoare unei rețele. DPUs efectuează o succesiune de operații pe date care se succed. Deoarece metodele tradiționale de sinteză a matricei sistolice au fost utilizate cu algoritmi algebrici, au rezultat doar matrice uniforme cu trasee de date liniare, astfel încât arhitecturile să fie aceleași în toate DPUs. O consecință este că numai aplicațiile cu dependențe regulate de date sunt în general implementate pe matrice sistolice clasice.

La fel ca la procesoarele SIMD *(single instruction/multiple data),* matricele sistolice sincrone operează în mod "*lock-step*" în care fiecare procesor alternează fazele de procesare / comunicare. De cealaltă parte, sistemele sistolice asincrone, folosesc un proces de "*hand-shaking*" între DPU. Acestea se numesc și "matrice în valuri" (de date) - *"wavefront arrays"*.

IA pentru HW - Parametrizarea dinamică anticipativă a HW de către IA

Pentru sistemele de PI în TR, etapele tipice ale procesării semnalelor sunt: Recepționarea semnalului, Conversia semnalului de măsurat în semnal electric (la nivelul senzor), Prelucrarea semnalului în domeniul analogic, Conversia analog-digitală (CAD) a semnalului, Procesarea semnalului în domeniul digital – simetric, se pot considera CDA, amplificări, acționări (la nivelul "actuator") ș.a.m.d.

În funcție de domeniul de utilizare și de algoritmii de IA folosiți, procesarea în domeniul digital (DSP – "Digital Signal Processing") se împarte, tipic, în:

- Procesarea generică a semnalului digital (eliminarea zgomotului și a altor distorsiuni, normalizarea semnalului)

- Algoritmii de IA, se împart, la rândul lor, în: Pre-procesare, calcule de IA (propriu-zise), Post-procesare

Fluxul de date poate fi liniar (fiecare bloc folosește semnalul generat de blocul anterior, așa cum e generat de acesta) sau neliniar (în care semnalul este stocat și utilizat de blocul următor într-un mod specific, diferit de cel generat de blocul anterior).

Memoria de stocare poate fi locală (între două blocuri consecutive ale fluxului de date), împărțită între mai multe blocuri (partajată) sau globală.

O schemă care ilustrează aceste sub-sisteme tipice implicate în procesarea semnalelor este prezentată în Figura 19. În sistemele de procesare a semnalelor, fiecare bloc din fluxul de date poate fi parametrizat independent, în funcție de modul de folosire.

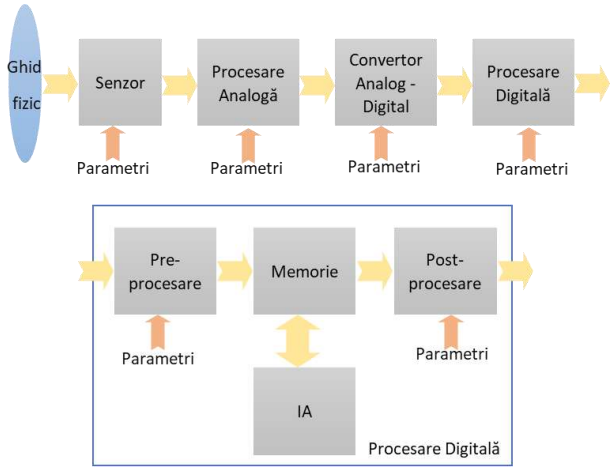


Figura 19: Modul de parametrizare a unui sistem de PI în TR

Astfel fiecare bloc are o funcție de transfer de tipul

$$Out = f(in, param)$$

În sistemele simple de procesare, parametrii sunt constanți în timp:

$$param = K$$

În sistemele mai complexe, parametrii pot fi modificați *dinamic* pe baza unor *statistici* asupra semnalelor recepționate *anterior*. Astfel, se poate obține o *adaptare* a comportamentului blocurilor unui sistem de TR, pentru un răspuns mai bun la semnalul de intrare.

$$param = f(statistics)$$
$$statistics = f(in)$$

Exemple:

- Detectarea nivelului maxim al semnalului și creșterea automată a factorului de amplificare
- Reglarea luminozității în imagini (în funcție de histograma luminozității pixelilor).

Principala problemă a parametrizării dinamice constă în modificarea bruscă a semnalului de intrare. În acest caz, o întârziere prea mare a buclei de reacție (semnal → statistici → parametrizare) afectează funcția de transfer a blocului de procesare.

Algoritmii de IA pot însă *anticipa* modificările semnalului de intrare, și astfel pot genera o parametrizare *predictivă* a blocurilor de procesare: $param = f(expected\ input)$

Această perspectivă presupune o înțelegere globală a sistemului de procesare de semnal, ca un *ecosistem,* în care componentele fizice, analogice, digitale și algoritmice sunt abordate global.

IA pentru HW - Controlul preciziei implementării HW

Un avantaj al implementării HW dedicate a algoritmilor matematici este acela de alegere detaliată a tipurilor de date folosite. Astfel, dacă în implementarea SW avem de a face cu tipuri de date standard (întregi sau în virgulă flotantă, pe 8, 16, 32 sau 64 de biți) în implementările HW se pot alege tipuri de date adaptate la tipul de problemă de rezolvat, granițele pentru reprezentarea datelor fiind mult mai ușor de ales.

Avantajele utilizării unor *tipuri de date adaptate* ("customized") reiese din numărul de operații care se pot implementa în paralel precum și din resursele fizice (de arie de siliciu sau energie) consumate.

Pentru algoritmii de PI, se pot folosi date reprezentate pe un număr variabil de biți, de la date reprezentate pe 1 bit (hărți binare) sau 2 biți (ponderi ternare) până la date de imagine pe 4, 8, 10, 12, sau 14 biți (recepționate de la senzorii de imagine actuali) și până la reprezentări pe 32, 40, 48 sau 64 de biți pentru date specifice diferiților algoritmi (pentru acumulare, adresare sau calcule matematice).

S-a observat că folosirea unui număr de biți mai mic decât necesarul teoretic poate duce la scăderea costului soluției prin scăderea ariei pe siliciu. Mai mult decât atât, în unele situații, scăderea preciziei calculelor matematice poate fi preferată, atâta timp cât rezultatele finale au o calitate apropiată de cea teoretică.

Acesta este cazul algoritmilor de PI cu IA, care se dovedesc robuști la erori mici de calcul. Prin natura algoritmilor de IA, prin chiar modul lor de antrenare, se acceptă o eroare a datelor de intrare.

Dacă această eroare este corelată cu cea generată de calculele efectuate în HW, atunci se poate imagina o soluție în care rezultatele finale sunt acceptabile, chiar dacă precizia calculelor intermediare este sub-optimală.

Un astfel de exemplu este prezentat în capitolul 4, în care se propune o implementare a unui algoritm de detecție a obiectelor folosind IA de tip SVM (Support Vector Machine) care folosește la intrare caracteristici ale imaginilor de tip histograme ale orientării gradienților (HOG – Histogram of Oriented Gradients) calculate cu o precizie mică.

Eroarea de calcul a orientării gradienților este dată de economisirea resurselor HW pentru implementarea funcției HOG, însă această eroare poate fi compensată într-o mare măsură de algoritmul IA implementat în SW.

Același principiu, în care implementarea HW folosește o *precizie definită de algoritmul IA,* ajustată urmărind rezultatele finale ale algoritmului, este folosit și în alte secțiuni ale lucrării, care tratează: precizia implementării procesării pixelilor în cadrul AHIP (capitolul 4) sau precizia implementării convoluțiilor în cadrul PCNN (capitolul 5).

IA pentru HW - Controlul prin IA al fluxului de date din HW

Algoritmii de IA pot folosi date diverse, provenite de la surse diferite. Pentru algoritmii de PI, se pot extrage date secundare ("meta-date") derivate din fluxul de date principal (de la senzorii de imagine).

Fluxul de date principal care este procesat în HW este controlat de IA și pe baza analizei acestui flux de date secundare: meta-datele (din fluxul principal) combinate cu rezultatul *fuziunii* datelor (secundare) de la *alți* senzori. Calitatea algoritmilor poate fi îmbunătățită atunci când sunt folosite și aceste surse alternative de informație.

În cazul corecției distorsiunilor pentru algoritmul de stabilizarea a imaginilor, datele de imagine nu sunt suficiente pentru înțelegerea traiectoriei de mișcare a sistemului de achiziție de imagini.

În această situație, algorimul IA are ca date de intrare și pe cele ale senzorilor inerțiali *(IMU – Inertial Measurement Units)* ce includ giroscopul (care măsoară unghiurile de rotire), accelerometrul (care măsoară accelerația deplasării) sau magnetometrul (care măsoară câmpului magnetic din jurul sistemului).

Provocările, rezultate din folosirea acestor tipuri de date diferite, apar în principal din modul de sincronizare a senzorilor. Detaliile implementărilor HW-SW sunt prezentate în capitolul 4 (corecția distorsiunilor din fluxul video) respectiv în capitolul 6 (stabilizarea imaginii pentru sisteme de TR).

Aceleași principii se aplică în cazul în care se folosesc alți senzori care trebuie sincronizați pentru a putea utiliza datele produse de aceștia în algoritmi de IA specifici.

HW pentru IA - Optimizarea capacității de procesare

Algoritmii de IA cei mai folosiți în prezent sunt cei neurali. Rețelele neurale convoluționale (CNN - Convolutional Neural Networks), cuprind de obicei mai multe straturi – straturi convoluționale, straturi complet conectate sau de eșantionare.

Pentru a rula cea mai simplă arhitectură de rețea tipică pentru detecția de obiecte, un sistem necesită, de obicei [83], [84], un sub-sistem de memorie LPDDR4 suplimentar (de **p**utere scăzută – LP, "Low Power" și cu **r**ată de **d**ate **d**ublă – DDR, "Double Data Rate", cu transfer de date atât pe frontul urcător cât și pe cel coborâtor al impulsurilor de ceas), un canal de 32 de biți și un procesor capabil să ruleze de la 100 de GMAC (Giga Multiply&Accumulate) – echivalentul mai multor nuclee ARM Cortex A75 cuadruple – până la mai multe TMAC (Tera MAC) – echivalentul câtorva zeci de nuclee ARM Cortex A75. În capitolul 5 se detaliază modul în care CNN pot fi implementate folosind resurse HW dedicate.

HW pentru IA - Utilizarea eficientă a magistralelor de date

Una dintre provocările cele mai mari ale implementării algoritmilor de IA în sistemele de PI în TR, este folosirea eficientă a benzii de transfer la memorie, așa cum a fost explicat anterior.

Din acest motiv, partiționarea HW-SW poate oferi opțiuni de implementare care să minimizeze traficul de date.

O direcție de cercetare este deci aceea în care se urmărește implementarea HW a IA în care KPI cel mai prioritar este modul de folosire a magistralelor de date, mai important decât alte criterii, precum viteza de procesare sau consumul de energie. Această abordare, prezentată în capitolul 5, schimbă deci paradigma de proiectare HW, astfel încât întregul proces este centrat pe modul de transfer eficient de date între sub-sisteme.

Soluțiile explorate țin de folosirea de memorii tampon pentru stocarea temporară a datelor, de atomizarea operațiilor implementate în HW pentru a evita mutarea datelor între blocurile HW sau de încapsularea funcționalităților, pentru minimizarea traficului între HW și SW.

O atenție specială este acordată proiectării blocurilor HW în funcție de modul optim de transfer al datelor (sub aspectul latenței, alinierii, volumului de date transferat).

3.4 Sumarul capitolului

În acest capitol am analizat modul în care se poate realiza accelerarea HW a algoritmilor de PI în TR. Pentru aceasta am descris în detaliu sub-sistemele tipice pentru PI de TR, pornind de la sub-sistemul optic al sistemului până la ieșirile acestuia (foto și video). Aceasta a permis explorarea indicatorilor de performanță critici, pornind de la capacitatea de procesare și până la aria consumată pe siliciu. O importanță specială a fost acordată folosirii eficiente a memoriei, aceasta fiind de cele mai multe ori factorul limitator în implementarea soluțiilor.

Pornind de la aceste criterii de performanță, am propus o soluție hibridă HW-SW pentru implementarea algoritmilor de PI în TR. Această soluție exploatează conceptul de ecosistem prezentat în capitolul 1, în direcția în care strămutarea HW ↔ SW a calculelor (relocarea lor totală sau parțială) permite optimizarea soluțiilor din punct de vedere al benzii de transfer la memorie, precum și al capacității de procesare sau al energiei consumate.

În cadrul acestor sisteme hibride HW-SW am analizat potențialul IA. Am identificat două mari direcții în care interacțiunea IA poate ajuta ecosistemul date-IA-HW-SW. Pe de o parte IA poate ajuta implementarea HW (”IA pentru HW”) iar pe de altă parte implementările HW pot crește performanțele IA (”HW pentru IA”).

Au fost stabilite direcțiile de cercetare din categoria ”IA pentru HW”. *Integrarea adaptivă a resurselor HW* presupune că algoritmul IA poate decide folosirea nu în mod necesar a resurselor HW care generează rezultatele cele mai de calitate, ci uneori a resurselor HW care execută sarcinile cel mai rapid sau care consumă cea mai puțină energie.

În cazul *parametrizării dinamice anticipative a HW de către IA,* controlul HW se face de către un algoritm IA care ține cont de datele care sunt recepționate și, totodată, anticipează modul în care acestea se schimbă. IA parametrizează (”instanțiază”) blocurile HW folosite pentru a procesa optim fluxul de date.

Controlul preciziei implementării HW folosind IA pentru a spori toleranța la erori se referă la faptul că, prin natura algoritmilor de IA, a însuși modului lor de antrenare, se presupune o eroare a datelor de intrare. Dacă această eroare este corelată cu cea generată de calculele efectuate în HW, atunci se poate imagina o soluție în care rezultatele finale sunt acceptabile, chiar dacă precizia calculelor intermediare este sub-optimală.

La *controlul prin IA al fluxului de date din HW,* fluxul de date principal care este procesat în HW este controlat de IA care analizează rezultatul fuziunii datelor secundare (de la alți senzori) cu meta-datele derivate din fluxul principal.

Din categoria ”HW pentru IA”, direcțiile de cercetare cuprind *optimizarea capacității de procesare* pentru algoritmii de IA cu rețele neurale convoluționale, precum și *utilizarea eficientă a magistralelor de date* prin schimbarea paradigmei de proiectare și concentrarea pe modul de optimizare benzii de transfer la memorie.

Capitolul 4. Inteligența Artificială pentru Hardware în PI

În categoria "IA pentru HW" au fost stabilite cele 4 direcții de cercetare (cărora le este dedicat acest capitol) în care IA poate controla componentele HW din cadrul sistemelor hibride. Acestea sunt: integrarea adaptivă a resurselor HW (decizie asistată de IA asupra configurației sistemului HW), parametrizarea HW de către IA (instanțierea adaptivă la nivel de modul HW), controlul preciziei calculelor implementate în HW (toleranța la erori sporită de IA) și controlul prin IA al fluxului de date din HW (bazat pe fuziunea datelor secundare cu meta-date din fluxul principal).

4.1 Integrarea adaptivă a resurselor HW

Așa cum am arătat în capitolul 3, se poate defini un sistem HW în care elementele de procesare, de tipul DPU (Data Processing Unit) sunt conectate între ele într-o matrice sistolică, astfel încât funcționarea DPU este *definită de fluxul de date.*

Modul de funcționare a sistemului poate fi controlat de IA, prin integrarea adaptivă a modulelor HW potrivite pentru a implementa funcțiile *necesare la un moment dat.* Astfel, IA poate decide care dintre componentele HW sunt folosite și care nu, precum și *modul de interconectare* a acestora.

Această *decizie asistată* de IA asupra *configurației* sistemului HW este concretizată în următoarele soluții pentru PI în sisteme de TR.

Sub-sistemul de accelerare HW pentru PI

Sub-sistemul propus [4-CZ-A], de accelerare HW pentru PI *(AHIP - Advanced Hardware for Image Processing),* procesează în HW datele de intrare recepționate de la sistemul de achiziție video, rezultatele fiind utilizate ulterior de către algoritmi PI de nivel superior. Sub-sistemul cuprinde un set de module HW care implementează primitive diferite pentru PI, gestionate de către un algoritm de control cu IA.

Separarea datelor de "interes" (pe diferite criterii, gestionate de algoritmii IA) constă în identificarea unor sub-seturi ("regiuni", "zone" – "colecții") și extragerea lor spre a fi tratate în mod special, mai amănunțit – de exemplu analizate în module HW dedicate.

Detalii de proiectare AHIP

Sistemul HW propus, implementat și brevetat, generează un set de *primitive* ale imaginii, în TR, pentru fiecare imagine recepționată. Recepția imaginilor se face serial, pixel cu pixel, cu o latență mult mai mică decât cea asociată achiziției întregii imagini. Aceste primitive sunt disponibile pentru procesare imediat după ce imaginea de intrare a fost achiziționată și pot fi folosite pe parcursul achiziției cadrului următor.

În plus, datele determinate de procesarea cadrului anterior dintr-o secvență de cadre pot fi folosite în conjuncție cu primitivele cadrului curent. Aceasta permite procesarea cadru cu cadru detaliată, fără a captura inițial un flux de cadre video de rezoluție scăzută.

Generarea de primitive de procesare diferite care pot oferi informații folositoare pentru cadrul curent implică operații care se efectuează cu viteza de 1 pixel pe ceas. Fiecare tip de primitivă este generată de un lanț de procesare care include mai multe blocuri de operații pe pixel. Aceste blocuri sunt conectate prin mai multe conexiuni interne, care *pot fi modificate în mod dinamic*. În cazuri mai puțin complexe, conexiunile pot fi fixe, dar intrarea primară a subsistemului HW poate fi comutată între senzor, IPP *(Image Processing Pipeline)* și memoria internă. Mai multe blocuri pot avea aceeași intrare de date. Mai mult, ieșirile mai multor blocuri pot fi combinate cu o funcție logică. Ieșirile individuale ale mai multor blocuri de procesare sunt combinate într-un singur flux de date înainte de a fi scrise în memoria externă (SDRAM) pentru *a optimiza folosirea memoriei și a magistralelor* externe la memorie. Datorită diferențelor de timp de procesare în diferitele blocuri, se introduc blocuri de sincronizare care aliniază corect fluxurile de date.

Primitivele de imagine generate pot fi folosite pentru a accelera performanța unui număr de algoritmi de PI. Pe lângă detecția de obiecte, mi-am adus contribuția la o serie de algoritmi de recunoaștere de persoane, de înfrumusețare a feței *(face beautification),* înregistrarea diferențelor între cadre, cuplarea imaginilor pentru panorame [3-CZ-A], [85], [86].

Disponibilitatea primitivelor de imagine simplifică semnificativ implementarea unui set de algoritmi de analiză a imaginilor. Aceasta poate reduce în mod particular tendința de a citi și a scrie întreaga imagine din/în memoria externă de către CPU sau GPU. De cele mai multe ori primitivele de imagine relevante sunt *citite o singură dată,* pentru a analiza sau îmbunătăți imaginea de intrare specifică unui anumit algoritm. Există posibilitatea de a cupla primitivele pentru mai mulți algoritmi împreună pentru a fi citite o singură dată, pentru a executa acești algoritmi pe o singură imagine de intrare. Acest mecanism *reduce semnificativ banda la memorie* pentru procesarea unui flux video. Dacă există posibilitatea folosirii bus-urilor dedicate pentru citire și scriere, se poate procesa o imagine de către sub-sistemul SW (la nivel CPU/GPU) în timp ce imaginea următoare este achiziționată și procesată de modulele AHIP.

În plus, această configurație a sistemului permite ca datele derivate din analiza unui cadru de imagine procesat de CPU / GPU să fie returnate ("alimentate înapoi" – *"fed back")* către IPP sau modulele AHIP pentru a adapta pre-procesarea unui cadru de imagine următor. Această adaptare detaliată atât a procesării globale a imaginii aplicată de IPP, cât și a procesării imaginii specifice scenei aplicată de AHIP, permite o performanță mai bună și o latență mai mică a unui dispozitiv de achiziție video. Acest lucru, la rândul său, permite adaptarea mai rapidă a achiziției video în situații în care condițiile de iluminare se schimbă, pe baza unei analize a regiunilor obiectelor de detectat și a hărților de culoare asociate obiectului. Astfel de tehnici se pot aplica cu succes sistemelor de achiziție video.

Pentru aceasta, se poate utiliza un contor de cadre (și logica asociată). La sfârșitul fiecărui ciclu de procesare a cadrelor, este posibilă reconfigurarea lanțurilor interne de procesare a pixelilor. Acest lucru poate implica încărcarea noilor tabele LUT, modificarea parametrilor de procesare a blocurilor individuale de procesare a pixelilor sau, în unele cazuri, reconfigurarea ordinii sau combinației logice a blocurilor într-un lanț de procesare. Blocurile sunt fie selectate, fie ocolite. În variante mai sofisticate, modulele de procesare a datelor partajează un port I/O pe unul sau mai multe bus-uri de date interne. Pot fi folosite *memorii cu porturi duale* pentru a asigura operații de citire/scriere aproape simultane.

Figura 20 prezintă un hardware IPP *(Image Processing Pipeline)* cu AHIP *(Advanced Hardware for Image Processing)* care include un flux de date direct.

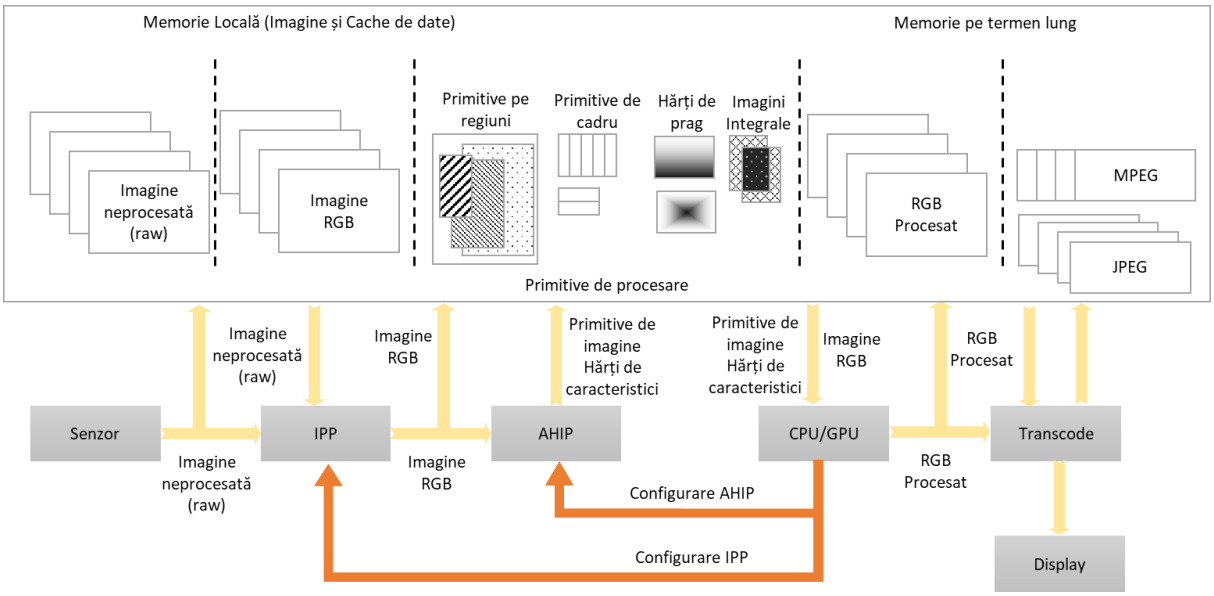


Figura 20: Arhitectura sistemului de detecție în TR cu configurare dinamică anticipativă

Fiecare pixel din imaginea de intrare este procesat în fiecare tact de ceas, și procesarea acestui pixel se poate face într-unul sau mai multe moduri. Mai multe tipuri diferite de ieșire pot fi generate în paralel din procesarea fiecărui pixel

individual. Mai multe instanțe ale fiecărui tip de procesare pot fi furnizate prin duplicarea elementelor HW. Deoarece acest subsistem HW poate procesa un pixel pe fiecare tact de ceas, el nu întârzie transferul pixelilor de imagine de la senzor și astfel poate fi implicat în orice etapă a IPP.

Un număr de tipuri generice de primitive de PI pot fi identificate și sunt generate de modulul AHIP. Pentru o nuanțare a termenilor, datele de imagine pot fi denumite „pixeli" (elemente de imagine), iar valorile datelor dintr-o primitivă de ieșire pot fi denumite „hărți-pixeli" (sau "meta-pixeli").

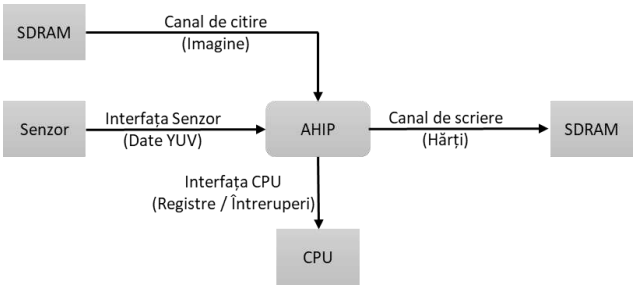


Figura 21: Interfețele fizice ale AHIP

Figura 21 ilustrează relațiile dintre un procesor principal, un modul HW AHIP, un senzor de imagine și canalele de citire/scriere a memoriei SDRAM. Interfața AHIP/CPU oferă acces la registre și la întreruperi. O interfață senzor pentru datele YUV conectează senzorul și blocul AHIP.

În cadrul AHIP sunt implementate mai multe tipuri de procesare a cadrelor de imagine. Câteva, mai importante, sunt descrise în cele ce urmează.

Procesarea cadru cu cadru în AHIP în aplicația de detecție a obiectelor

Figura 22 ilustrează utilizarea datelor din cadrul N–1 combinată cu un cadru curent N. Memoria de stocare – date imagine și cache – este segmentată în mod corespunzător celor două cadre (ale exemplului ilustrativ din Figura 22). În ceea ce privește urmărirea facială – ca exemplu reprezentativ – regiunile prezise sunt stocate în memorie atât pentru cadrul N–1, cât și pentru cadrul N. Din cadrul N–1, într-o porțiune a memoriei sunt stocate regiuni rafinate și date primitive pe regiuni. Acestea sunt utilizate pentru a genera date despre cadrele de imagine RGB și primitivele pentru cadrul N. Datele pe regiuni din cadrul N–1 sunt introduse în blocul AHIP împreună cu datele RGB ale cadrului N. AHIP scalează și re-mapează spațiul de culoare pe anumite date pe regiuni ale cadrului N–1, în blocul de scalare, apoi trece toate datele pe regiuni ale cadrului N–1 printr-un bloc de procesare a pixelilor pentru a fi stocate în memorie la porțiunea aferentă cadrului N. Datele RGB ale cadrului N pot fi sub-eșantionate și spațiul de culoare re-mapat înainte de prelucrarea pe pixeli. Procesarea pixelilor poate include procesarea cumulativă, directă, cu mască, pe regiune și/sau hibridă a pixelilor. Datele sunt apoi stocate în porțiunea aferentă cadrului N a memoriei.

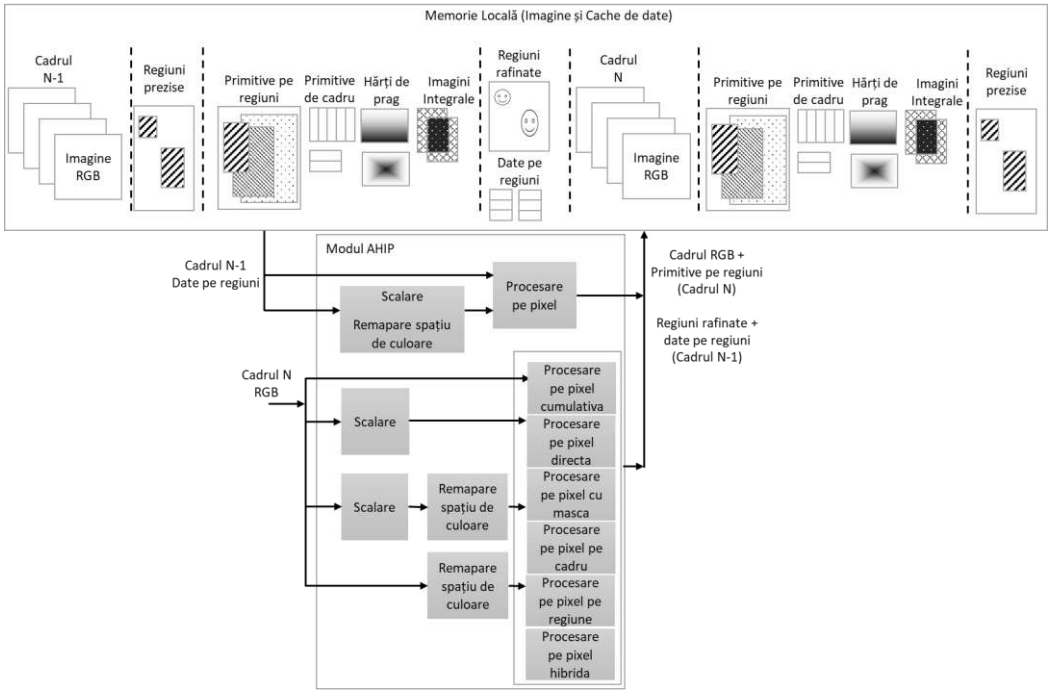


Figura 22: Modul de combinare a datelor de la cadre diferite în AHIP

Brevetarea contribuțiilor

Datorită importanței ideilor prezentate în acest capitol, s-a pus accent pe brevetarea contribuțiilor inovatoare. Astfel, conceptul inițial de accelerare HW folosind AHIP [15-CZ-P] a fost propus pentru patentare în mai multe țări, grupuri de țări sau chiar pe plan mondial. În Statele Unite, propunerea a fost înregistrată US20120008002A1 iar la nivel global WO2012004672A2 . Mai multe iterații au fost făcute pentru detalierea conceptului, așa ca la nivel global au fost înregistrate mai multe variante ale propunerii de patent (WO2012004672A3 și WO2012004672A3). De asemenea propunerea a fost trimisă pentru brevetare în China (CN103098077A), Japonia (JP2013531853A completată cu JP2013531853A5) precum și în Coreea de Sud (KR20130098298A).

Din cele 48 de revendicări (”claims”) depuse, au fost aprobate 37, prin patentele care au fost acordate în Statele Unite (US9053681B2), China (CN103098077B) Japonia (JP5819956B2) și Coreea (KR101646608B1).

Noi concretizări ale implementării AHIP în HW au fost propuse pentru patentare în 2015 în Statele Unite (US20150042670A1). Propunerea conține 20 de revendicări, care detaliază modul de interacțiune a modulelor HW pentru PI în TR. Toate cele 20 de solicitări au fost aprobate în 2017 prin patentul US9607585B2.

O nouă îmbunătățire a fost propusă în 2017, pentru a detalia modul de segmentare predictivă a datelor. Aceasta este prezentată în documentul US20170256239A1 printr-un număr de 20 de revendicări noi. Dinte acestea au rezultat 9 revendicări brevetate în 2019, prin patentul US10418001B2 .

În anul 2020 a fost propusă pentru patentare o nouă îmbunătățire a soluției, și anume aceea de formatare adaptivă a datelor și de transmisie a datelor comprimate adaptiv. Această propunere conține 20 de solicitări (US20200005739A1) care au fost aprobate în întregime în patentul indexat US10930253B2 în 2021.

Scalarea adaptivă a imaginilor

O altă soluție de integrare adaptivă a resurselor HW e dedicată scalării imaginilor [7-CZ-A].

Importanța scalării adaptive poate fi ilustrată de domeniul pazei și securității (de exemplu detecția intrării prin efracție) – scopul generic al sistemelor de achiziție de imagini este acela de a maximiza calitatea imaginii înregistrate și redate pentru a fi folosită de utilizatorii umani. Una dintre caracteristicile care au un impact major asupra calității imaginilor este rezoluția acestora. Cela mai multe sisteme moderne folosesc senzori de rezoluție ridicată, iar imaginile transmise și stocate în Cloud au crescut mult ca dimensiuni în ultima perioadă. S-a ajuns astfel la sisteme actuale care folosesc rezoluții HD, Full HD sau 4k pentru achiziția și transmisia video. Creșterea rezoluției imaginilor poate duce la o calitate superioară pentru afișare, dar cu costul creșterii semnificative a traficului de date de la Edge către Cloud [87].

Pentru procesarea datelor se implementează sisteme complexe de tip CPU/GPU multi-core [88]. Aceste sisteme controlează componenta HW .

Pe de-altă parte, în sistemele ”embedded” (cu capabilități de calcul încorporate), puterea consumată este unul dintre indicatorii de performanță care contează cel mai mult. Pentru acest motiv, toate componentele sistemului trebuie optimizate din punct de vedere al energiei consumate. Literatura de specialitate [41] prezintă consumul de energie normalizat în SoC (Systems on Chip) – cum am arătat în Figura 10.

Graficul demonstrează că transferul de date în general și transmisia fără fir (”wireless”) în particular, consumă o cantitate mare de energie, așa că îmbunătățirile în această zonă sunt critice.

Algoritmii de detecție de obiecte au evoluat în timp. În aplicații din domeniul securității, detectorul de obiecte este folosit pentru a declanșa alarmele potrivite atunci când diferite obiecte sunt prezente în scena de urmărit.

Un al treilea aspect (pe lângă vizualizarea cât mai bună pentru operatorul uman și consumul energetic cât mai redus) este următorul: rezoluția de lucru pentru cei mai buni algoritmi IA este mică în comparație cu rezoluția senzorilor, iar din acest motiv, imaginea recepționată de la senzor trebuie sa fie redimensionată înainte de a aplica algoritmul IA.

În mod uzual, rezoluția de lucru pentru algoritmii de detecție de obiecte care rulează în dispozitivele încorporate este VGA sau chiar mai mică [89], [90].

Posibilitățile de a utiliza IA în gestiunea optimală a modului de procesare a imaginilor (ISP – Image Signal Processing) au fost analizate în diferite lucrări. Fluxul de date brute, reducerea zgomotului, de-mozaicarea, amplificarea digitală, echilibrarea valorilor de alb, corecția de culoare, compresia valorilor gamma, maparea tonurilor au fost luate în considerare [90], [91], [92] ca factori de impact asupra alegerii și parametrizării "asistate" (de IA) a celor mai bune soluții de ISP.

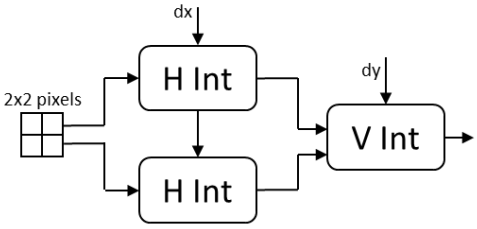
Cu toate acestea, cercetările intreprinse și prezentate în această lucrare au vizat noi soluții de gestiune a metodelor de scalare pentru a spori semnificativ calitatea algoritmilor de detecție de obiecte.

Soluția propusă

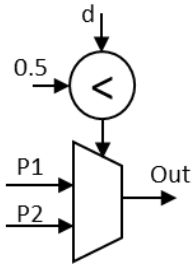
Au fost implementați fiecare din cei 3 algoritmi de scalare. Ieșirea a fost folosită de algoritmul de detecție de obiecte, așa cum a fost descris anterior în capitolul 4.

Senzorul a fost configurat să funcționeze la o rezoluție nativă (Full HD sau 4K) pe când algoritmul de scalare a fost configurat să genereze o imagine la rezoluția de funcționare a algoritmului de detecție de obiecte (QVGA sau 224x224 pixeli).

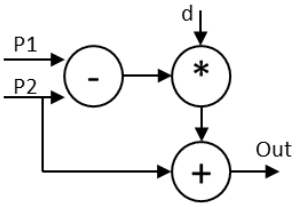
Implementarea HW pentru fiecare algoritm este prezentată în continuare. Algoritmii de interpolare "cel-mai-apropiat-vecin" și de interpolare biliniară folosesc o fereastră de pixeli 2x2, precum în Figura 23 a), unde *dx* și *dy* sunt $(x - x_1)$, respectiv $(y - y_1)$.



a) Implementarea generică pentru interpolările "cel-mai-apropiat-vecin" și biliniară



b) Implementarea HW a interpolării "cel-mai-apropiat-vecin"



c) Implementare HW a interpolării biliniare

Figura 23: Implementarea pentru interpolările "cel-mai-apropiat-vecin" și biliniară

Implementarea modulului de interpolare "cel-mai-apropiat-vecin" se regăsește în Figura 23 b) iar implementarea modulului de interpolare biliniară se regăsește în Figura 23 c).

Implementarea modulului de interpolare bi-cubică se regăsește în Figura 24:

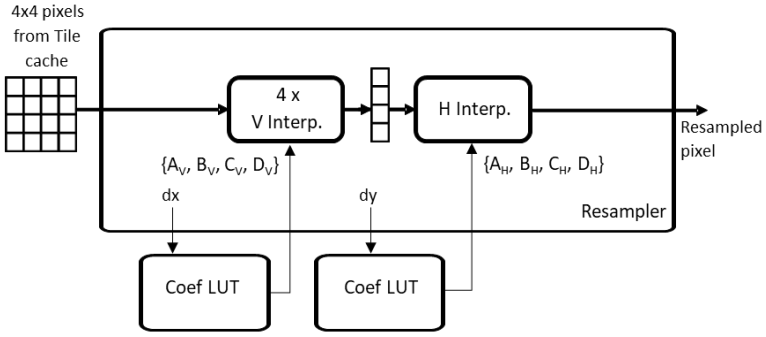


Figura 24: Implementare HW a interpolării bi-cubice

Energia consumată de fiecare modul în parte (valoare normalizată în raport cu interpolarea "cel-mai-apropiat-vecin") este prezentată în Tabelul 1.

Metoda de scalare	Energia (normalizată)	Resurse HW necesare
Cel-mai-apropiat-vecin	1	3 MUX
Biliniar	63	6 sumatoare, 3 multiplicatoare
Bicubic	345	15 sumatoare, 20 multiplicatoare

Tabelul 1: Valorile normalizate ale energiei consumate de către modulele de interpolare

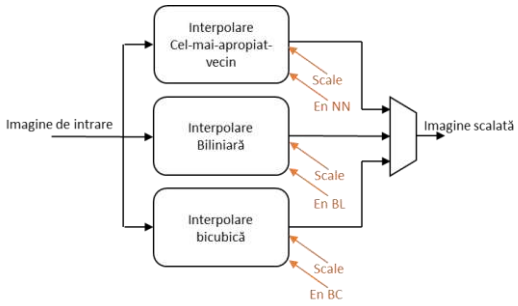


Figura 25: Implementarea modulelor de scalare cu semnale de configurare

Modulele de scalare implementate în HW sunt selectate și configurate de către algoritmul de IA (detecție de obiecte). Figura 25 prezintă implementarea modulului HW pentru scalare. Semnalele notate portocaliu reprezintă valori de configurare primite de la algoritmul de IA.

Cât timp nu sunt detectate alarme în dispozitiv *(Edge)* putem transmite datele în *Cloud* pentru a înregistra la o rezoluție scăzută. Rezoluția înaltă este necesară doar în cazul în care în câmpul vizual sunt detectate obiecte.

Rezultate: Am rulat algoritmii de scalare propuși și am analizat datele de ieșire obținute. Deși diferențele nu sunt majore în cazul imaginilor de intrare de calitate superioară, putem observa diferențe atunci când datele recepționate de la senzor sunt în lumină slabă sau cu zgomot.

Am testat algoritmul de detecție de obiecte folosind date de intrare scalate utilizând diverse tehnici, iar rezultatele sunt prezentate în Figura 26 (fiecare culoare reprezintă o altă clasă de obiecte). Am analizat curbele de precizie-recunoaștere (P-R) ca principală metrică a calității algoritmului. Am observat faptul că, dacă imaginile folosite pentru testare sunt augmentate cu zgomot (precum în cazul tipic de supraveghere), atunci calitatea algoritmului scade semnificativ.

În concluzie, pe baza soluției propuse, când în HW avem opțiunea de a implementa multiple module de scalare, *selecția adaptivă* între diferite metode generează pe de o parte un consum de energie foarte diferit și pe de altă parte, imagini de ieșire de calitate foarte diferită [97].

Algoritmul "cel-mai-apropiat-vecin" consumă semnificativ mai puțină energie decât celelalte metode, însă datele de ieșire sunt de calitate foarte scăzută, atât pentru uz uman cât și pentru procesare de către IA.

Consumul de energie pentru interpolarea biliniară este mai ridicat decât în cazul algoritmului "cel-mai-apropiat-vecin", dar calitatea imaginilor este îmbunătățită semnificativ, iar detecția de obiecte este efectuată mai bine.

Interpolarea bi-cubică este cel mai ineficient algoritm analizat din punct de vedere al energiei consumate, dar îmbunătățirile observate la calitatea imaginilor și la metricile detecției de obiecte nu sunt cu mult mai mari decât la interpolarea biliniară. Cu toate acestea, în cazuri foarte specifice, diferențele sunt observabile.

Am observat că, ținând cont de faptul că majoritatea algoritmilor de ultimă oră folosesc imagini de intrare de rezoluții scăzute, are sens ca imaginea recepționată de la senzor să fie scalată cât mai curând posibil (în cadrul ISP) pentru a procesa, stoca și transmite o imagine de rezoluție scăzută.

Acest fapt are un impact major în cantitatea de date trimise în Cloud, iar energia totală economisită prin acest mecanism este importantă. Totuși, în cazurile în care sunt necesare imagini de calitate si rezoluție ridicate, cum ar fi în domeniile de securitate, algoritmul de detecție de obiecte poate configura într-o manieră adaptivă hardware-ul de scalare.

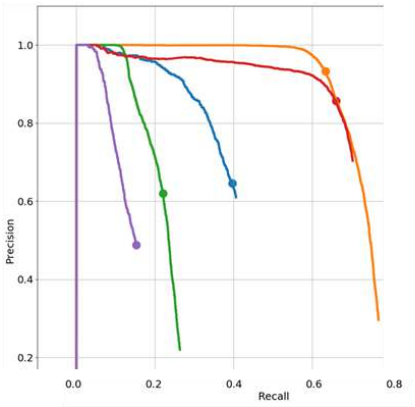
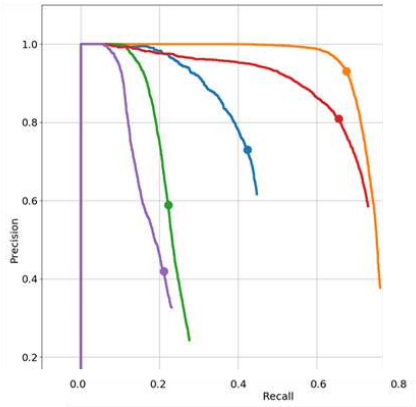
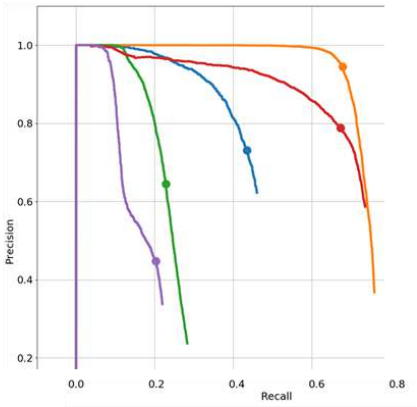
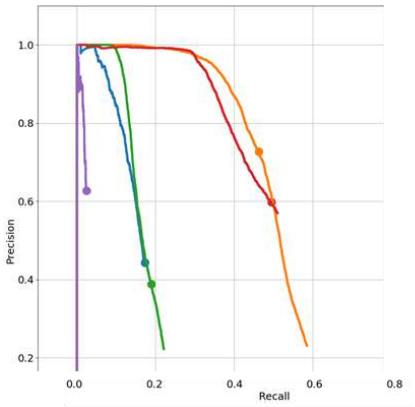
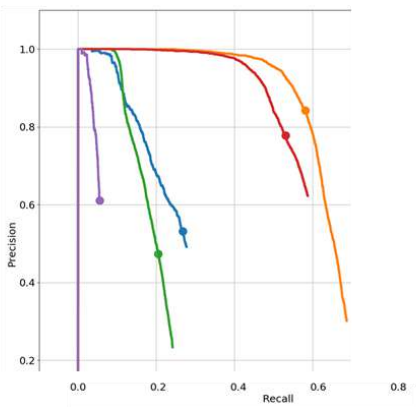
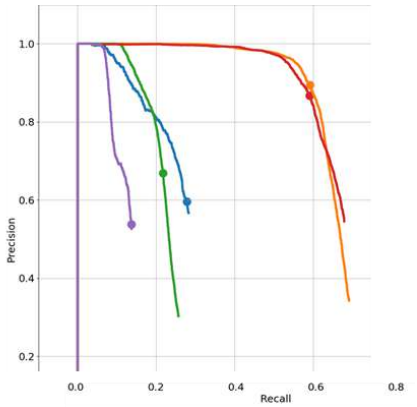
Cel mai apropiat vecin	Biliniar	Bicubic
Fără augmentare de zgomot		
		
Cu augmentare de zgomot		
		

Figura 26: Calitatea algoritmilor de detecție de obiecte (curbe P-R)

În acest caz, pentru o perioadă de timp, atât rezoluția imaginilor este mărită (pentru stocare și transmisie), cât și calitatea scalării este îmbunătățită (prin selecția unui modul de scalare de calitate mai ridicată).

Ca rezultat, în majoritatea cazurilor, când nicio alarmă nu este declanșată, soluția globală consumă foarte puțină putere, pe când în cazurile în care sunt declanșate alarme, soluția oferă o calitate ridicată.

4.2 Parametrizarea dinamică anticipativă a HW de către IA

Așa cum s-a arătat în capitolul 3, una din metodele prin care IA poate îmbunătăți implementarea HW a algoritmilor de PI în TR poate fi prin parametrizarea modulelor HW folosind un control dinamic, anticipativ.

Aceasta presupune controlul HW de către un algoritm IA, care configurează blocurile HW ținând cont de datele care sunt recepționate precum și prin anticiparea modului în care acestea se schimbă și, totodată, IA parametrizează blocurile HW folosite pentru a procesa optim fluxul de date.

În cadrul acestei secțiuni este exemplificată o implementare hibridă HW-SW pentru un algoritm de detecție de obiecte, în care care folosesc conceptele de parametrizare dinamică anticipativă.

Nuanțăm deosebirea între *configurare* (comutare/selecție la nivel constructiv, chiar arhitectural), ca urmare a unei *decizii asistate* prealabile, și *parametrizare* (instanțiere) consecutivă acesteia: este ca și cum am alege o formulă (din mai multe posibile) și apoi am da valori coeficienților din formulă.

Parametrizarea anticipativă pentru un algoritm de detecție a obiectelor

Sistemul propus constă într-un sistem optimizat pentru detecția de obiecte cu parametri diferiți. Potrivirea de șabloane *("template matching", TM)* constă în găsirea de similitudini între un model de referință și zone ale imaginii de interes [98], [99], [100], [101]. Complexitatea TM se pretează la implementarea IA într-un sub-sistem care efectuează în paralel analiza obiectelor, astfel încât să se detecteze parametrii acestora (tipul obiectului, orientarea, poziția sau condițiile de iluminare). Pe baza parametrilor detectați ai obiectelor, se modifică dinamic parametrii HW de către modulul de IA, astfel încât șablonul *("template")* folosit să fie optimizat pentru detecția obiectelor cu parametrii respectivi.

Arhitectura sistemului

Sistemul de detecție de obiecte cu accelerare HW (propus, implementat și brevetat) [10-CZ-P] este prezentat în Figura 27.

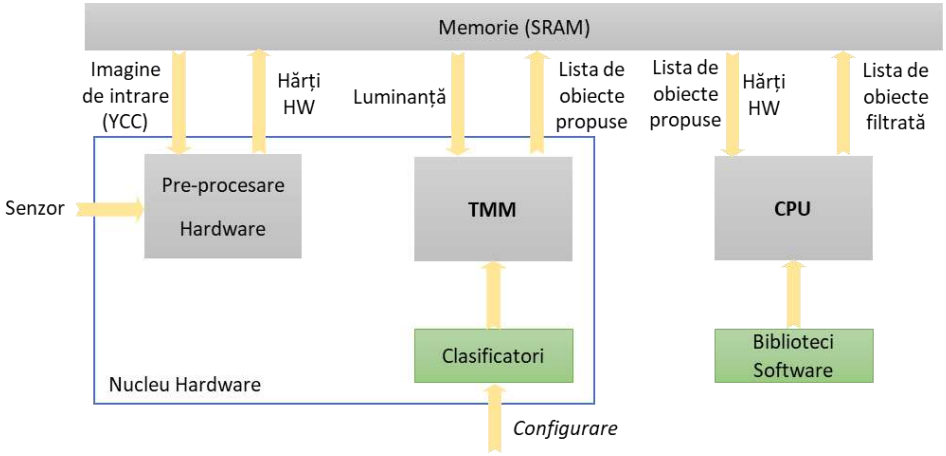


Figura 27: Arhitectura sistemului de detecție în TR cu accelerare HW

Imaginea este recepționată fie direct, de la un senzor de imagine (mod pre-vizualizare, *preview),* fie de la memoria sistemului (mod redare, *playback)* unde a fost stocată anterior. Blocul de accelerare HW pre-procesează datele primite, astfel încât să producă hărțile de caracteristici necesare algoritmului TM. Procesarea hărților de caracteristici se face în formatul YCC (luminanță – crominanță). Hărțile generate de modulul de accelerare HW sunt stocate în memorie.

Imaginea este procesată în TR, astfel încât toate componentele sistemului trebuie să poată rezolva rapid sarcinile specifice. De aceea, modulul de accelerare HW execută operațiile cu viteza senzorului, de 1 pixel pe tact.

Modulul TMM *(Template Matching Module)* este programat de către procesor să aplice clasificatori multipli pentru detecția de obiecte, în paralel asupra imaginii de intrare, pentru a determina diferite caracteristici ale obiectului detectat (tipul, orientarea, poziția, expunerea, condițiile de iluminare etc) și apoi poate comuta în mod dinamic modelele de detecție, alegând modelul cel mai potrivit cu caracteristicile determinate.

Clasificatorii de comparat în paralel pot fi un subset din clasificatorii complet definiți pentru a detecta obiecte cu caracteristici specifice (cum ar fi fețe cu o anumită orientare) sau pot fi clasificatori pentru a detecta doar anumite caracteristici (cum ar fi condiții de iluminare de noapte). Se pot face și combinații între diferite tipuri de clasificatori.

Detecția cu clasificatori (TM) implică redimensionarea imaginii la diferite scale locale. TMM poate fi programat să schimbe dinamic clasificatorii pe baza analizei de la imagine la imagine sau selectiv pe fiecare scală locală. Acest bloc aplică seturile de clasificatori în paralel, pentru a detecta dacă anumite obiecte apar în scenă. De asemenea, modulul poate fi programat pentru a determina dacă obiectul este poziționat - orientat într-un anumit mod (unghiul de expunere la planul imaginii etc), dacă obiectul apare complet sau parțial în scenă sau alte condiții – dacă obiectul a fost capturat sub o anumită direcție de iluminare, dacă imaginea este neclară sau expusă neoptimal (supra-expusă sau în condiții de iluminare scăzută) etc.

TMM poate de asemenea să cuprindă componente HW dedicate, care nu sunt programate, care pot funcționa ca și clasificatori ce rulează în paralel.

De exemplu, un set de clasificatori pot fi antrenați pentru a detecta fețele umane iluminate uniform și cu poziționare frontală la cameră, pe când alte seturi de clasificatori pot fi antrenați pentru a detecta fețele care sunt poziționate în direcții diferite, rotite sau iluminate neuniform. Alți clasificatori pot fi antrenați pentru a detecta anumite persoane din scenă; alți clasificatori pot fi antrenați pentru a detecta alte obiecte precum pisici, câini, mașini, case, munți, etc. Fiecare set de clasificatori poate fi aplicat în paralel pe imagine pentru a determina dacă e prezentă o față sau oricare dintre caracteristicile analizate.

Orientările sau condițiile de capturare ale imaginii pot include direcția de iluminare ale obiectului, rotația acestuia în plan, o variație a poziției 3D, o regiune a obiectului, dacă apare zâmbetul și "în ce măsură", dacă ochii clipesc și în ce măsură, dacă gura este deschisă și cât de mult, cât de neclară este fața, defecte ale modului de capturare a ochilor, umbrirea feței, ocluzii ale feței, culori ale feței, forme ale feței, sau combinații ale acestor caracteristici.

Unul sau mai multe module de urmărire pot fi folosite pentru a urmări în imagini consecutive oricare dintre aceste obiecte care au fost detectate.

Un exemplu de utilizare este acela în care este achiziționată o imagine în care apare o față iluminată neuniform. Mai mulți clasificatori sunt rulați în paralel pe datele achiziționate, inclusiv pentru a detecta diferite condiții de iluminare; Clasificatorul desemnat ca fiind cel mai potrivit face și o determinare proprie a condițiilor de iluminare.

Se poate aplica apoi o corecție specifică, pentru ca fața să capete o iluminare uniformă sau se poate opta pentru a lua în considerare, în continuare, fața iluminată neuniform (așa cum apare) spre a fi urmărită, editată, memorată sau procesată ca atare.

Un mod de iluminare, orientare, poziție sau alte condiții pot fi determinate pentru un obiect pe baza criteriilor de acceptare a datelor obiectului de către unul dintre clasificatori. Imaginea poate fi una dintr-un flux de imagini în serie care include obiectul. Doi sau mai mulți clasificatori pot fi aplicați dacă obiectul nu a putut fi detectat de primul (primii) clasificator(i).

Detecția de obiecte implică extragerea unei ferestre din imaginea recepționată. Doi sau mai mulți clasificatori pot fi aplicați în paralel și se pot antrena pentru a fi sensibili la caracteristicile regiunilor în care apar obiectele de detectat.

Schimbarea dinamică anticipativă a clasificatorilor

Prin schimbarea dinamică a clasificatorilor, rata de detecție pentru condiții limită – cum ar fi subexpunere, iluminare de fundal, sau supraexpunere – poate fi îmbunătățită. Se observă o îmbunătățire și în cazul în care direcția de iluminare a obiectului este diferită. Informațiile despre modul de iluminare pot fi furnizate pentru detecția de obiecte, de exemplu prin analiza imagine cu imagine a fluxului de date și înregistrarea istoricului de-a lungul mai multor imagini. Aceste informații pot fi utilizate pentru a selecta diferite tabele de îmbunătățire a contrastului, pentru a regla / îmbunătăți un cadru de intrare înainte de a aplica algoritmul TM. Prin generarea și apoi aplicarea diferiților clasificatori, care să furnizeze cea mai bună performanță în condiții de iluminare specifică, și schimbarea dinamică a acestora pe baza analizei obiectului și a cadrului de intrare, se obține soluția cea mai avantajoasă din punct de vedere al performanței.

TMM se poate integra într-un eco-sistem hibrid HW-SW. Așa cum se ilustrează în prezentarea AHIP *(Advanced Hardware for Image Processing),* un bloc HW pentru template matching (TMM) poate fi combinat cu un filtru SW avansat și cu componente de urmărire a obiectului. SW poate folosi ieșirea de la blocul HW TMM și propria analiză avansată a imaginii pentru a selecta cei mai buni clasificatori și tabelul pentru îmbunătățirea contrastului necesar unei anumite situații de iluminare. Acesta poate, de asemenea, schimba clasificatorii și tabelul pentru îmbunătățirea contrastului din interiorul modulului HW TMM, prin acest lucru aducând o performanță superioară pentru aceste condiții de iluminare specifice, în termen de viteză și/sau calitate a detecției. Acest lucru se poate face dinamic, pentru următoarele cadre din fluxul video live sau prin aplicarea re-detecției pe o imagine statică.

Există posibilitatea ca schimbarea să fie aplicată *doar* tabelului pentru îmbunătățirea contrastului sau *doar* pentru clasificatori, în funcție de condițiile de iluminare. Schimbarea dinamică a clasificatorilor poate fi aplicată atât în soluții complet HW, complet SW sau hibride.

Scalarea locală selectivă

Schimbarea clasificatorilor poate fi aplicată pe o anumită scală a imaginilor de intrare. La algoritmul de detecție de obiecte, când se antrenează un nou clasificator și apoi se baleiază imaginea de intrare pentru a găsi obiectul, fiecare grupă de antrenament sau de scanare este de obicei normalizată pentru metrica de varianță. Parametrii folosiți pentru normalizare sunt stocați odată cu definiția clasificatorilor. Pentru a reduce dimensiunea clasificatorilor (memoria în cazul AHIP sau *heap* în cazul SW), un nou tip de clasificatori poate fi introdus, care folosește *operația de divizare în locul celei de scădere*. Din acest motiv, informația despre varianță poate fi ignorată. Cu toate acestea, această abordare se pretează la cazul în care nu sunt condiții extreme de iluminare cum ar fi subexpunere, iluminare de fundal sau supraexpunere.

Schimbarea dinamică a clasificatorilor pe baza informațiilor de iluminare poate fi mai utilă în aceste situații. Folosirea unor clasificatori *dedicați* pentru cazurile de subexpunere, normale, sau supraexpunere generează rezultate mai bune. Fiecare din acești clasificatori poate fi o calibrare suplimentară a clasificatorului general de detecție de obiecte și în acest caz *memoria folosită este redusă.*

Această abordare funcționează bine pentru a compensa într-o anumită măsură lipsa normalizării cu varianța specifică clasificatorilor reduși ca dimensiune.

În situații mai complicate, cum ar fi atunci când mai multe obiecte cu caracteristici diferite apar în aceeași imagine, sau când sunt prezente obiecte iluminate atât normal cât și sub-iluminate, se poate face o altă abordare.

Dacă în abordarea precedentă compensarea se poate face global, pentru a normaliza întreaga imagine, am propus o soluție originală pentru cazul în care informațiile extrase pentru întreaga imagine diferă de informațiile locale (specifice unor zone ale imaginii): decizia de schimbare a clasificatorilor poate fi mutată în mod eficient către scalele locale, astfel că decizia de a aplica un anumit clasificator de iluminare (normal, sub-expus sau supra-expus) să fie diferită pentru anumite regiuni ale imaginii.

Fiecare dintre acești clasificatori specializați sunt antrenați folosind condiții de iluminare specifice, deci sunt adaptați pentru detecția obiectelor în aceste condiții particulare. Aceasta permite reducerea pierderilor datorate lipsei normalizării cu varianța, specifică clasificatorilor reduși ca dimensiune care pot fi implementați hardware. Se poate aplica schimbarea dinamică a clasificatorilor pentru fiecare regiune în parte.

Calculul valorii medii a fiecărei regiuni se poate implementa cu *numai patru accese la memorie*, folosind imaginea integrală [102], [103]. Comparând această valoare calculată cu anumite praguri, se poate lua *decizia* de iluminare potrivită.

Această abordare poate crește rata de detecție pentru cazurile dificile descrise anterior, fără a influența alte metrici de performanță. În plus, se obține un timp de detecție mai rapid, atât timp cât se poate face schimbarea dinamică a clasificatorilor în cadrul prezent, fără a mai aștepta ca SW să schimbe dinamic clasificatorii. În modul static, nu este nevoie de treceri succesive prin imagine.

Brevetarea contribuțiilor

Conceptul de accelerare folosind TMM pentru detecția de obiecte a fost propus pentru brevetare în anul 2012 [9-CZ-P] prin propunerea US20120106790A1 . Aceasta conține 25 de revendicări, care au fost aprobate în totalitate în anul 2015 (US8995715B2).

O continuare a patentului a fost propusă tot în anul 2015 [10-CZ-P]: propunerea US20150206030A1 introduce conceptul de parametrizare dinamică adaptivă, folosirea de DPU (Data Processing Units) multiple precum și procesarea HW pe regiuni. Toate cele 20 de noi revendicări au fost aprobate prin patentul US9639775B2 din 2017.

O îmbunătățire semnificativă a conceptului de TMM este adăugarea unui pre-filtru HW [11-CZ-P]. Această propunere a fost trimisă pentru brevetare în 2017 în Statele Unite (US20170185864A1), Europa (EP3213257A1), China (CN108431824A), dar și la nivel global (WO2017108222A1). Propunerea conține 26 de revendicări și acestea au fost aprobate în 2019 în Statele Unite (US10460198B2), precum și în Europa în 2018 (EP3213257B1).

Două noi propuneri de îmbunătățire au fost făcute în anul 2020, [12-CZ-P], [13-CZ-P], cu 20 de noi revendicări pentru generalizarea soluției (US20200050885A1 și US20200380291A1). Acestea sunt încă în așteptarea aprobăriiâ

4.3 Controlul preciziei implementării HW folosind IA pentru a spori toleranța la erori

Așa cum am arătat în capitolul 3, un alt mod prin care se poate îmbunătăți factorul de merit al unei soluții de PI în TR cu IA este prin controlul preciziei implementării HW. Prin natura algoritmilor de IA, a modului lor de antrenare, există o toleranță la erori în datele de intrare (o formă a robusteții acestor algoritmi SW). Așadar precizia calculelor HW nu are sens să fie prea mare și se poate reconsidera o simplificare a implementării HW.

Ilustrăm optimizarea prin IA a preciziei HW *(care nu e nevoie să fie mereu ridicată)* pentru PI într-o gamă de soluții de detecție a obiectelor bazate pe HOG, histograma orientării gradienților ("histogram of oriented gradients").

Obținerea HOG – pentru cel puțin o porțiune a unei imagini – necesită împărțirea respectivei porțiuni de imagine în celule, fiecare celulă cuprinzând o multitudine de pixeli de imagine. Pentru fiecare pixel din imagine care aparține unei celule, o componentă de gradient orizontal, gx, și o componentă de gradient vertical, gy, este obținută pe baza diferențelor de valori ale pixelilor de-a lungul a cel puțin unui rând al imaginii menționate și, respectiv, al unei coloane a imaginii menționate, incluzând pixelul. Un gradient este alocat unuia dintr-o multitudine de sectoare, fiecare sector extinzându-se printr-o serie de unghiuri de orientare. Sectoarele menționate sunt împărțite pe sectoarele adiacente de-a lungul liniilor, cu $gx=2^n \cdot gy$, unde n este orice valoare întreagă mai mare sau egală cu 1. Imaginea e împărțită în "celule".

Algoritmii de detecție a obiectelor sunt foarte populari, mai ales datorită numărului mare de aplicații în PI – cu accent pe detectarea persoanelor. Pentru sistemele încorporate ("embedded"), implementarea unor astfel de algoritmi este o provocare, mai ales când sunt luate în considerare (în raport cu resursele disponibile limitate) constrângerile de procesare în TR și cerințele de precizie ridicată.

Un algoritm calitativ pentru detecția obiectelor, potrivit pentru implementarea în TR pe dispozitivele inteligente "de la marginea rețelei" (nivel Edge) este descris în [104] și constă în producerea HOG urmată de aplicarea unui algoritm de învățare automată tip SVM *(Support Vector Machine)* pentru clasificarea setului de date.

Soluția originală propusă [1-CZ-A] constă într-o tehnică nouă de implementare a calculului HOG, optimizată pentru implementarea HW. Ipotezele pentru această tehnică de implementare sunt: 1) Calculul HOG nu trebuie să fie foarte precis, deoarece algoritmii de învățare automată sunt robuști și pot compensa mici erori și 2) contrar altor tehnici de optimizare din literatura de specialitate, accentul e pus pe interpolarea containerelor *(bins)* pentru HOG+SVM, foarte importante pentru calitatea și robustețea generală a algoritmului, în special pentru rotații.

Abordarea propusă constă în împărțirea orientării gradienților într-un număr mare de containere mici care să fie utilizate ca referință pentru interpolare și calcularea histogramelor. Pentru că detaliile de implementare sunt foarte importante pentru a se înțelege complexitatea implementării, detaliez resursele HW necesare și performanța unui astfel de sistem.

În locul căutării unghiurilor cunoscute (20^0 în [109] și [110]) sau doar a câtorva containere (în [107] și [108]), se adoptă o abordarea inversă: se caută unghiurile cele mai ușor de folosit pentru detecția în HW, folosind doar comparații simple, cu rezoluție suficientă pentru utilizarea în învățarea automată și păstrând opțiunea de interpolare între containere.

Soluția e ilustrată de Figura 28.

Unghiurile nu sunt egale, după cum se poate vedea în Tabelul 2.

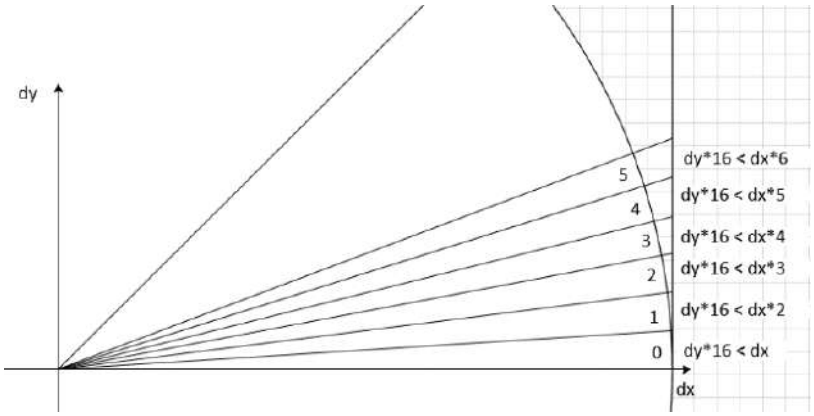


Figura 28: Containere mici, împărțite folosind comparații simple

Tabelul 2: Dimensiunile și marginile containerelor mici

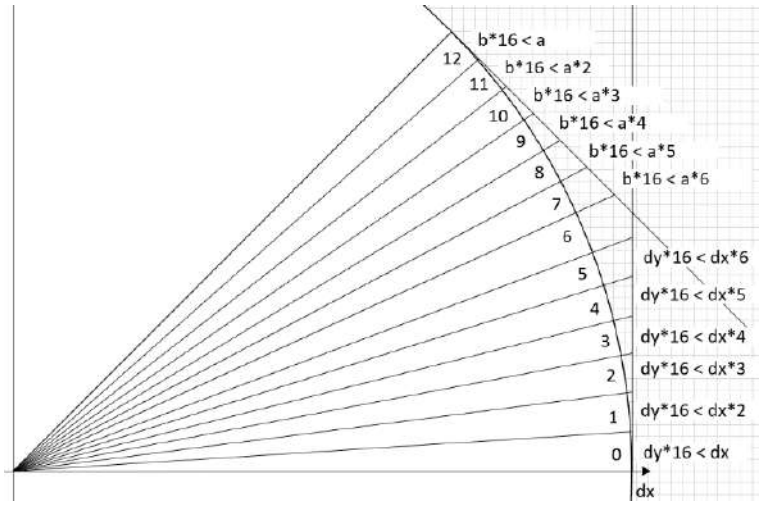
Numărul containerului	Marginea containerului (grade)	Mărimea containerului (grade)
0	3,58	3,58
1	7,13	3,55
2	10,62	3,49
3	14,04	3,42
4	17,35	3,32
5	20,56	3,20
6	23,63	3,07
7	26,57	2,94
8	29,36	2,79
9	32,01	2,65
10	34,51	2,50
11	36,87	2,36
12	39,09	2,22

Deoarece dimensiunile unghiurilor containerului sunt în scădere cu numărul alocat, am decis să iau în considerare doar primele 6 dintre ele cu dimensiuni între 3,58 și 3,20 grade. Prin schimbarea axei de referință la 45°, putem face același lucru începând de la 45 de grade în jos. Schimbarea axei cu 45° se poate face prin schimbarea de coordonate:

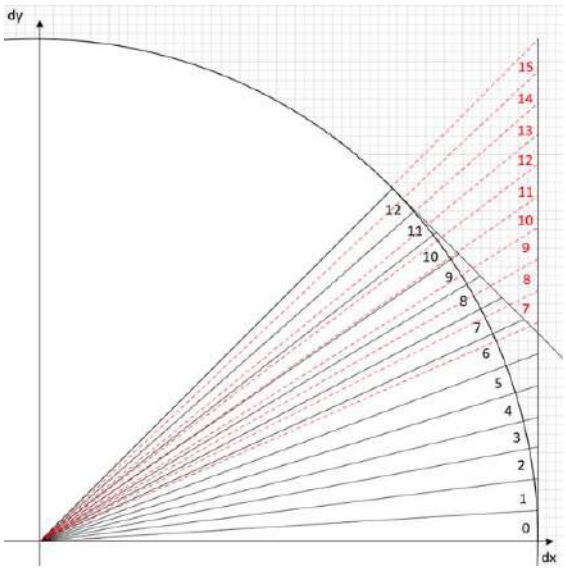
$$a = x + y;$$

$$b = x - y;$$

Containerele rezultate sunt prezentate în Figura 29 a)



a) Containerele după schimbarea axei la 45 °



b) Câștigul obținut prin schimbarea axei la 45 °

Figura 29: Distribuția containerelor la orientare 45 °

Containerele de la 12 la 7 sunt prezentate în Tabelul 3.

Tabelul 3: Marginile si mărimea containerelor după rotația axei cu 45°

Numărul containerului	Marginea containerului (grade)	Mărimea containerului (grade)
12	41,42	3,58
11	37,87	3,55
10	34,38	3,49
9	30,96	3,42
8	27,65	3,32
7	24,44	3,20

Tabelul 4: Reprezentarea containerelor mici în cadranul 0-45°

Numărul containerului	Marginea containerului (grade)	Mărimea containerului (grade)
0	3,58	3,58
1	7,13	3,55
2	10,62	3,49
3	14,04	3,42
4	17,35	3,32
5	20,56	3,20
6	24,44	3,89
7	27,65	3,20
8	30,96	3,32
9	34,38	3,42
10	37,87	3,49
11	41,42	3,55
12	45,00	3,58

Câștigul obținut prin schimbarea axei este prezentat în *Figura 29* b) (comparativ cu cele 15 containere care ar fi rezultat doar cu procedura inițială, în lipsa acestei schimbări de axe).

Containerul numerotat 6 este cel rezultat, după ce primele 6 (de la 0 la 5) și ultimele 6 containere (de la 7 la 12) sunt calculate, așa cum este prezentat în Tabelul 4.

Calcularea magnitudinii

Tipic, magnitudinea este calculată cu formula pitagoreică de mai sus, destul de dificil de implementat în HW.

Sunt propuse mai multe metode de optimizare, cum ar fi cele din lucrările [112], [113] și [114]. În cazul implementării propuse avantajul este că în interiorul unui container mic, magnitudinea are o variație foarte mică cu unghiul.

Acesta este motivul pentru care am considerat unghiul mediu (de orientare) aferent unui container (1,788° pentru intervalul 0 sau 5,3506° pentru intervalul 1, etc) și am calculat cosinusul acestui unghi, pentru a obține magnitudinea din componenta orizontală a gradientului, *Gx(x,y)*.

Pentru aceasta utilizăm formula:

$$m(x,y) = \frac{1}{\cos(\propto)} Gx(x,y)$$

Pentru unghiurile între 0° și 45°, constanta $\frac{1}{\cos(\propto)}$ este calculată per container si are valoarea între 1 și 1,372. Dacă multiplicăm valoarea fiecărei constante cu 256, putem memora valorile rezultatelor constantei K pentru fiecare container într-un LUT *(Look-Up Table)* doar numere întregi, între 256 și 351.

Rezultă că magnitudinea este:

$$m(x,y) = \frac{K * Gx(x,y)}{256}$$

Rădăcina pătrată si cele 2 multiplicări sunt astfel înlocuite de o singură înmulțire.

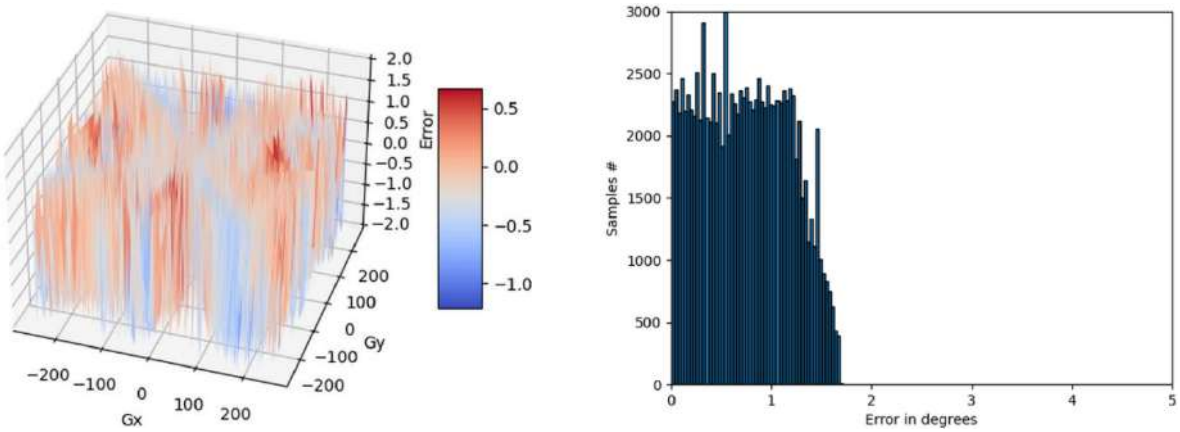
Evaluarea metodei propuse

Metoda de calcul magnitudinii și orientării este foarte avantajoasă în ceea ce privește resursele logice destul de reduse, ceea ce o face potrivită pentru implementarea hardware. Calculul containerelor mici necesită un total de 6 sumatoare și 12 comparatoare, în timp ce calculul magnitudinii necesită 1 multiplicator și un LUT cu 13 locații.

Având în vedere aceste aspecte, costul pentru calculul HOG este mai mic de 1000 porți logice în tehnologie ASIC. În FPGA, costul algoritmului este 143 de LUTs. Aceasta înseamnă că implementarea HW pentru algoritmul HOG se poate face cu costuri foarte mici.

Graficul de eroare pentru întregul spațiu de gradienți, luând în considerare valorile complete ale gradientului orizontal și vertical, este prezentat în Figura 30 a). Eroarea maximă este de 3 grade. Eroarea este calculată ca diferență între direcția gradientului standard, calculată cu formula tradițională, a arc-tangentei, și orientarea găsită prin aproximarea containerelor mici.

Reprezentarea erorii pentru orientarea gradienților este prezentată în Figura 30 b). Figura prezintă erorile în grade, pentru toate valorile gradienților orizontali și verticali.



a) Distribuția erorilor în funcție de gradienți b) Histograma erorilor de calcul a aproximării containerelor

Figura 30: Eroarea orientării pentru aproximarea containerelor mici

După cum este prezentată în [104], *calitatea totală* a algoritmului depinde de numărul de containere utilizate pentru algoritmul HOG+SVM, observând că un număr mai mare de containere generează rezultate mai bune.

În general, calitatea algoritmului HOG + SVM scade, așa cum este detaliat mai jos. *Figura 31* prezintă curba PR *(Precision-Recall)* - detaliată în capitolul 1 - pentru același algoritm de învățare care folosește atât metoda standard cât și metoda noastră de calcul al histogramei de gradienți orientați (HOG) (compromisul pentru eficiența implementării în hardware menționată mai sus este o mică scădere a performanței). Aceste observații sunt în concordanță cu alte tipuri de aproximări, așa cum sunt cele prezentate în lucrările [117], [118], [119] și [120].

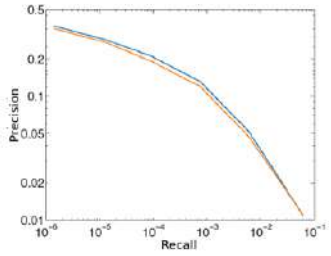


Figura 31: Curba PR pentru metoda originală (albastru) și metoda propusă (roșu)

Metoda prezentată pentru calculul HOG este foarte eficientă pentru o implementare HW. Am arătat cum se implementează în HW cu un cost foarte scăzut algoritmul de extracție a caracteristicilor HOG; metoda propusă generează erori minime fiind validată în contextul detectării obiectelor.

Eficiența algoritmului provine din optimizarea calculelor direcției gradientului, prin înlocuirea funcției costisitoare de arc-tangentă cu asocierea de containere mici asimetrice care aproximează direcțiile gradientului cu erori sub 2 grade.

Aceste erori au un impact foarte mic asupra calității algoritmilor de învățare automată care utilizează aceste caracteristici (SVM) și asupra calității generale a algoritmului de detectare a obiectelor (aceasta fiind afectată într-un mod foarte limitat).

Implementarea algoritmului HOG propus permite implementarea în TR a unui algoritm de detectare a obiectelor cu un consum energetic foarte mic.

Alte avantaje ale soluțiiei implementate sunt nevoile reduse de calcul pe dispozitivele inteligente de la nivel Edge, cu beneficii privind minimizarea transmisiei de date pentru casele inteligente sau rețelele de securitate.

Brevetarea contribuțiilor

Propunerile de implementare prezentate au fost depuse pentru brevetare în Statele Unite [14-CZ-P] Propunerea inițială (US2017098135A1) a fost depusă pentru brevetare în 2017 și prezintă modul de calcul a regiunilor asimetrice de calcul în HW. Au fost propuse 40 de revendicări, care se concentrează pe modul în care se pot defini 6 regiuni de orientare a gradienților pentru fiecare cadran. Propunerea de patent a fost emisă și în Europa (EP3058510A1) precum și la nivel global (WO2016083002A1). Toate cele 40 de revendicări au fost aprobate pentru brevetare în 2018 în Statele Unite (US9977985B2) dar și în Europa (EP3058510B1).

Am propus o îmbunătățire a soluției, care presupune definirea unor regiuni mult mai mici per cadran, precum și posibilitatea ponderării apartenenței fiecărei orientări la una dintre regiunile definite pentru HOG.

Această propunere a fost trimisă pentru patentare în 2019 (US20190205691A1) și conține 7 revendicări noi. Propunerea a fost trimisă de asemenea la nivel European (EP3459010A1) și global (WO2017198861A1) dar și pentru patentare în China (CN109478242A).

Patentul conține 20 de revendicări prin care se explică în detaliu conceptul de regiuni asimetrice mici pentru implementarea HW a HOG și a fost aprobat în 2020 în Statele Unite (US10839247B2).

4.4 Controlul prin IA al fluxului de date din HW

O altă modalitate prin care IA poate controla în mod eficient modul de lucru al componentei HW este prin controlul fluxului de date.

Din datele de prelucrat, algoritmii de IA pot extrage informații suplimentare, care să definească la un nivel superior modul în care fluxul de date este procesat pentru algoritmii de PI în TR.

Datele de intrare pot proveni din surse diferite, de aceea, în acest caz putem avea de a face cu o fuziune a senzorilor. Nu doar senzorii de imagine contribuie cu date la luarea deciziilor, ci și alți senzori cum ar fi cei inerțiali (giroscop, accelerometru) sau în alte game de frecvențe (radar, lidar, senzori în infraroșu).

- Fluxul principal de date este procesat de sistemul HW de TR (în cazul stabilizării de imagine este vorba de fluxul video). Fluxul secundar de date (meta-date din fluxul video și ieșiri de la alți senzori secundari) este analizat de IA pentru a controla fluxul de date principal.

În cazul particular al distorsiunilor imaginilor din fluxul video, decizia pentru corecția distorsiunilor nu se poate lua strict pe baza datelor video, deoarece acestea pot induce erori mari de estimare a distorsiunilor. Din acest motiv, urmărim construirea unui sistem HW, în care decizia de corecție este definită de IA, pe baza analizei întregului set de senzori care există la dispoziția sistemului de PI în TR.

Corecția distorsiunilor din fluxul video folosind datele generate de IA

Inteligența artificială furnizează, în această gamă de soluții – prezentată în cele ce urmează – o decizie asistată (asupra tipului de corecție) și, totodată, o *parametrizare* (a corecției astfel alese).

Figura 32 prezintă un exemplu de sistem digital de achiziție a imaginilor, inclusiv grupul de lentile WFOV.

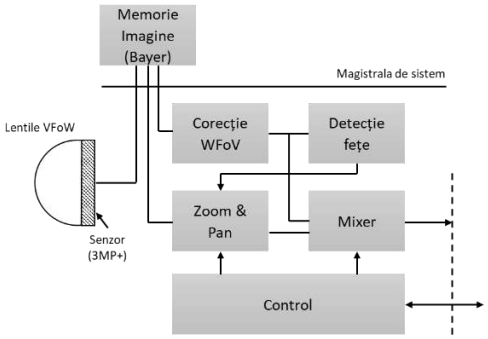


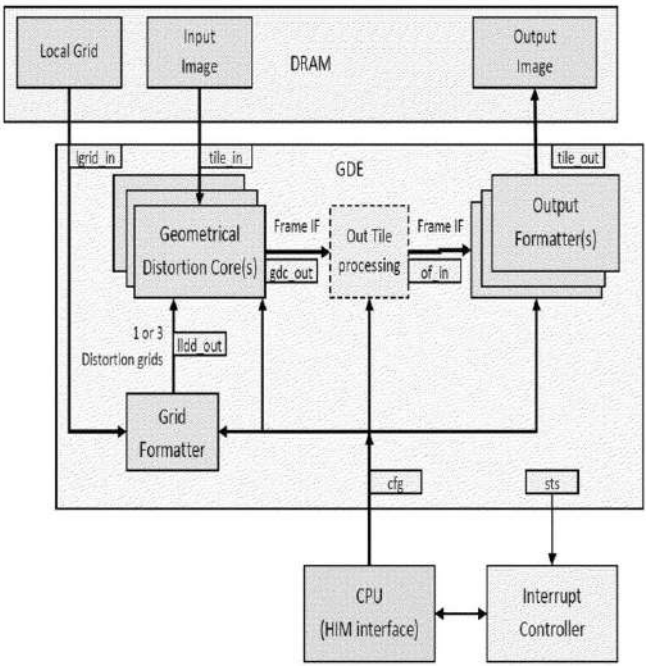
Figura 32: Sistem de achiziție de imagini folosind lentile WFOV

Aici, imaginile WFOV distorsionate sunt citite de la un senzor printr-un flux de prelucrare care poate efectua o pre-procesare simplă a unei imagini, înainte de a fi citită, pe o magistrală, în memoria sistemului. Astfel de sisteme pot utiliza module HW de corecție sau sub-module conectate direct sau indirect la magistrala de sistem pentru citirea imaginilor succesive stocate în memorie și pentru procesarea imaginii înainte fie de a returna imaginea procesată în memoria sistemului, fie de a redirecționa imaginea procesată pentru prelucrare ulterioară.

În Figura 32, un modul de corecție WFOV citește succesiv imagini (sau porțiuni de imagine) distorsionate și furnizează imagini (sau porțiuni de imagine) corectate într-un modul de detectare a feței (FD) și de urmărire. Un controler de sistem controlează diferitele module HW, controlerul de sistem fiind receptiv la comenzi primite printr-o interfață de control de la aplicații SW care rulează pe sistemul cu care interacționează un utilizator. În Figura 32, la controler este conectat un modul "Zoom & Pan" (mărire și scalare) și acest modul, la rândul său, comunică cu modulul de corecție WFOV pentru a determina ce parte a unei imagini achiziționate trebuie citită din memoria sistemului în vederea corecției și, de exemplu, pentru afișarea pe vizorul camerei foto-video și redirecționarea către modulul de detectare a feței. În acest exemplu, un modul mixer va adăuga zonele limită ce înconjoară fețele (care au fost detectate/urmărite) pentru afișare pe vizor.

Implementarea soluției de corecție

În Figura 33 este prezentată structura pentru manipularea distorsiunilor geometrice într-un sistem digital de achiziție a imaginilor care implementează soluția propusă. După voi explica în detaliu mai jos, corectorul de distorsiune geometrică (GDE – "Geometrical Distorsion Engine") este capabil să elimine în mod eficient distorsiunile introduse de un sistem de lentile WFOV, dar și să compenseze distorsiunile cauzate de tremurul camerei video și să corecteze distorsiunile introduse de un utilizator prin interacțiunea cu o aplicație care rulează pe sistemul de achiziție. O astfel de distorsiune provocată de utilizator poate necesita o transformare afină, o transformare a culorilor sau o transformare a imaginii pentru a aplica efecte speciale imaginii și secvenței de imagini achiziționate de camera digitală.



Procesarea distorsiunilor pe fiecare plan de culoare al unei imagini, de exemplu RGB, YUV sau LAB este independentă de celelalte, și astfel, în cadrul GDE, un singur nucleu de corecție a distorsiunii geometrice (GDC – Geometric Distorsion Correction) procesează un singur plan de culoare, oferind astfel o mai mare flexibilitate la nivel de sistem. Un singur GDC poate procesa fiecare plan de culoare secvențial sau mai multe instanțe de GDC (cum sunt cele din Figura 33) pot procesa toate planurile unei imagini în același timp.

În această prezentare, termenul de "grid" (grilă, raster) este utilizat pentru a se referi la o matrice de blocuri rectangulare ale imaginii ("tiles" – țigle, dale, plăcuțe). Fiecare dală este definită de cele patru colțuri ale sale și acestea sunt denumite noduri.

Figura 33: Arhitectura sistemului hardware de corecție a distorsiunilor [17-CZ-P]

O transformare mapează coordonatele nodurilor dintr-un grid în funcție de tipul distorsiunii care trebuie corectată. Această transformare este exemplificată în Figura 34.

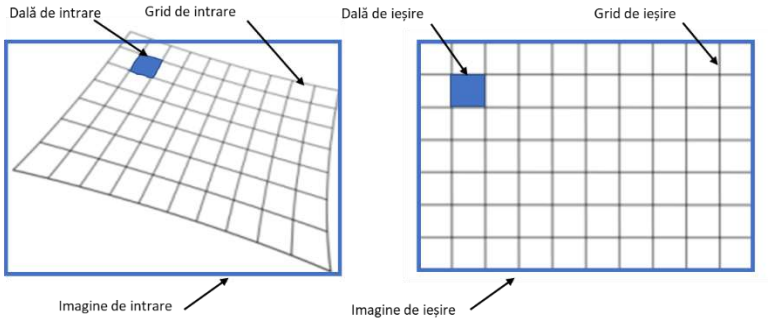


Figura 34: Transformarea imaginii folosind un grid de corecție

GDC procesează "dală" după "dală" imaginea de intrare sub controlul unei unități de formatare a grilei (GFU – Grid Formatter Unit). GDC preia dalele ("tiles") de intrare (tile_in) din DRAM în conformitate cu adresele furnizate de GFU și le procesează, producând pixelii corectați pentru dalele ("tiles") de ieșire respective (gdc_out) în ordine raster normală.

În exemplul din Figura 35, grila locală care cuprinde nodurile $N_1 \ldots . N_n$ suferă o transformare *flip* (oglindire) și de perspectivă și astfel ar putea compensa rotația variabilă a senzorului de imagine în raport cu ansamblul lentilelor, precum și pentru simularea sau compensarea unei vizualizări în oglindă.

Deci, se poate vedea cum nodul N_1 se deplasează de la stânga sus la dreapta jos și invers cu nodul N_n, și că partea de sus transformată a rețelei locale devine mai comprimată decât partea de jos. Efectul transformării grilei locale asupra geometriei și locației unei anumite dale este, de asemenea, ilustrat. De asemenea, transformarea grilei locale poate compensa distorsiunile variabile cauzate de schimbările de perspectivă pentru diferite regiuni locale din cadrul unei imagini de intrare, în special în cazul sistemelor WFOV. Astfel, grila locală poate ajuta la compensarea gradului mai mare de distorsiune găsit în fețele de la marginea unui câmp vizual larg față de cele situate (sau detectate) în centrul câmpului vizual.

Valorile din antetul grilei locale sunt utilizate de *L Grid Calc* pentru a configura registrele responsabile pentru accesarea informațiilor despre grila locală din DRAM. După aceasta, GFU începe să citească coeficienții nodului de rețea locală din DRAM, unul câte unul. Coordonatele transformate pentru nodurile grilei sunt apoi transmise unui bloc *Affine* (dacă este activat). În această implementare, blocul *Affine* înmulțește coordonatele nodului de intrare u,v cu o matrice 2×3 care cuprinde coeficienții $a_1 \ldots a_6$ din transformata afină *(AT - Affine Transformation)* pentru a produce valori de coordonate de ieșire u', v':

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix}$$

Într-o implementare alternativă, transformarea afină de mai sus poate fi modificată pentru a oferi o transformare extinsă care poate corecta perspectiva. Aici, blocul de transformare înmulțește coordonatele nodului de intrare u,v cu o matrice care cuprinde coeficienții $a_1 \ldots . a_9$ pentru a produce valori de coordonate de ieșire u', v':

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} a_5 \\ a_6 \end{bmatrix} * D$$

unde

$$D = \frac{1}{u * a_7 + v * a_8 + a_9}$$

Astfel, dacă a_7=a_8=0 și a_9=1 transformarea este o transformare afină ca mai sus. În alte cazuri, a_7 și a_8 pot fi variate, cu a_9 fixe și egale cu 1. Cu toate acestea, activarea variată a_7, a_8 și a_9 oferă cea mai flexibilă soluție.

În implementările ulterioare, transformarea afină poate fi adaptată pentru a corecta două transformări de perspectivă, folosind o matrice de omografie (de mapare) cu coeficienți definiți corespunzător.

Valorile acestor coeficienți matriceali $a_1 . . . . a_6$ și, eventual, a_7, a_8 și/sau a_9 sunt stocate în regiștrii interni ai GFU. Acești regiștri stochează valori ale coeficientului ce pot fi programate de două ori: într-un prim mod, Global Affine Transform, menționat mai sus, regiștrii pot fi programați de CPU înainte de începerea procesării cadrului și valorile lor sunt menținute constante pentru toate grilele locale ale unui cadru întreg; în al doilea mod, Local Affine Transform, valorile regiștrilor sunt citite din DRAM împreună cu un index de nod care indică când trebuie încărcat un nou set de coeficienți de transformare afină.

După cum s-a menționat mai sus, al doilea mod permite actualizări de transformare dinamică și corecție a distorsiunilor tip "rolling shutter" (efectul de scanare "obturator derulant") împreună cu compensarea mișcării camerei. Astfel, se va vedea că în acest exemplu, transformarea afină cuprinde o transformare a locațiilor coordonatelor nodului din transformarea locală (Lt).

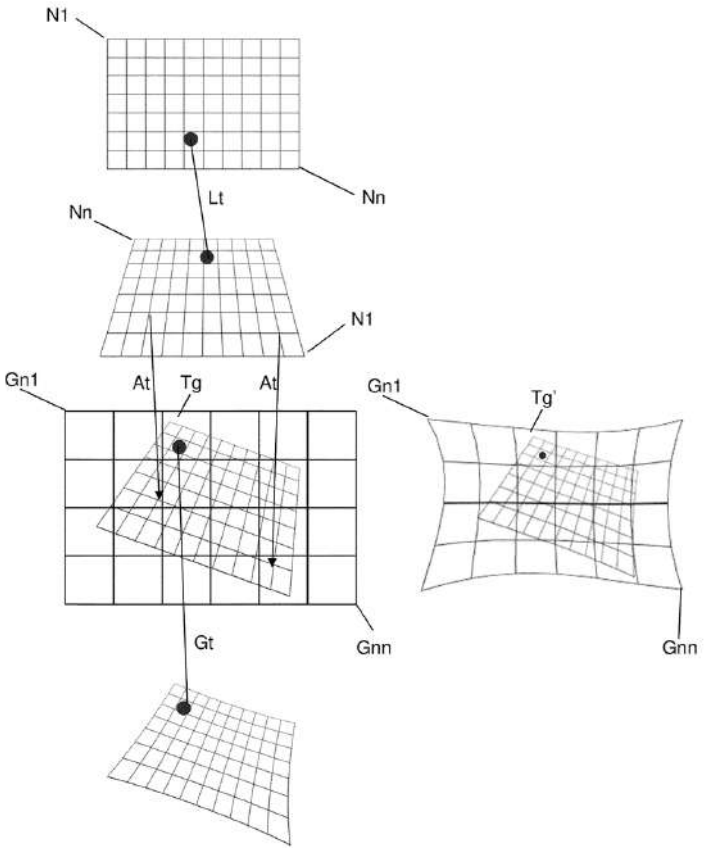


Figura 35: Modul de transformare a grilei de corecție

În exemplul de față prezentat în Figura 35, transformarea afină (At) cuprinde o transformare globală afină care rotește întreaga grilă și astfel ar putea compensa o nealiniere de rotație (într-un plan paralel cu planul senzorului de imagine) dintre obiectiv și senzorul de imagine.

Transformarea globală (Gt) cuprinde o cartografiere a coordonatelor nodului Gn_1 la Gn_n al unei grile globale pentru a lua în considerare distorsiunea lentilelor. Pentru o anumită coordonată nod după transformare afină (At) și sau transformare grilă locală (Lt), *G Grid Calc* găsește nodurile grilei globale din jurul acelei locații și interpolează maparea acelor noduri ale dalei grilă globală la nodul local și afin transformat al grilei locale pentru a determina transformarea globală (Gt) a acelei locații de nod.

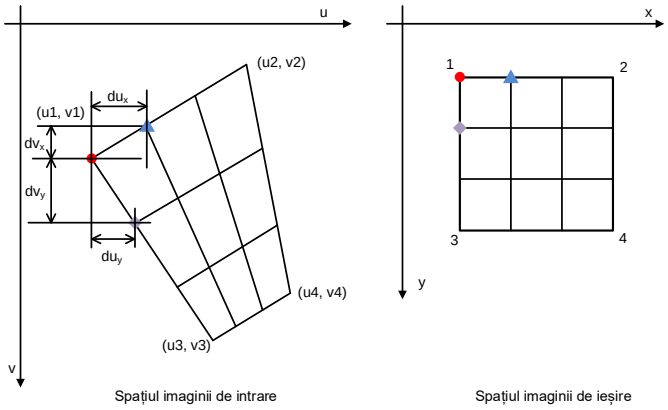


Figura 36: Exemplu de transformare a unei dale a imaginii

Figura 36 prezintă o dală în spațiul de ieșire (dreapta) și de intrare (stânga) al imaginii. În acest exemplu, dala conține 4×4 pixeli. *LLDD Calculator* primește coordonatele celor patru colțuri (u_1,v_1) la (u_4,v_4) și calculează "diferențele finite" parțiale (du_x , dv_x etc.) necesare unui "adresator" din cadrul GDC pentru interpolarea liniară a fiecărei coordonate de pixeli (u,v). După cum am arătat mai sus, cunoscând diferitele transformări necesare pentru a compensa distorsiunea determinată de cameră, mișcare și utilizator, *LLDD Calculator* poate determina zona necesară a spațiului imaginii de intrare definit de (u_1,v_1) ... (u_4,v_4) la spațiul de imagine de ieșire delimitat definit de nodurile 1, 2, 3, 4.

Brevetarea contribuțiilor

Brevetul [17-CZ-P] pentru stabilizarea de imagine folosind date de la senzori diferiți pentru construirea grilei de corecție a fost propus în anul 2014 în Statele Unite (US20140009568A1) la nivel European (EP2870585A1) și global (WO2014005783A1). În Statele Unite, patentul a fost aprobat în 2015 (US8928730B2) și conține detalierea conceptelor din 57 de revendicări. În Europa, patentul aprobat cuprinde doar 15 revendicări și a fost publicat în 2016 (EP2870585B1).

Soluțiile detaliate privind modul de implementare HW, modul de folosire a memoriei pentru o procesare eficientă precum și modul de programare a nucleelor HW [18-CZ-P] au fost propuse pentru brevetare în anul 2015 atât în Statele Unite (US20150178897A1) cât și în Europa (EP3101622A2 și EP3101622A3). Toate cele 21 de revendicări au fost aprobate în 2016 atât în Statele Unite (US9262807B2) dar și în Europa (EP3101622B1).

O nouă îmbunătățire a fost propusă în 2015 (US20150262344A1) referitor la adăugarea corecției de perspectivă [19-CZ-P]. În urma procesului de patentare, brevetul, obținut în 2016 (US9280810B2) conține 21 de revendicări aprobate.

4.5 Sumarul capitolului

Capitolul 4 detaliază soluțiile de tip *"IA pentru HW".* În cadrul conceptului de integrare adaptivă a resurselor HW am prezentat în detaliu sub-sistemul HW *AHIP,* care este folosit în sistemele hibride HW-SW pentru procesarea imaginilor și implementat în circuitele integrate pentru telefoane mobile sau camere foto.

Parametrizarea HW de către IA prin configurare dinamică anticipativă a fost exemplificată prin modulul HW TMM, care permite detecția de obiecte folosind algoritmul de potrivire a șabloanelor *("template matching").*

Controlul preciziei implementării HW folosind IA pentru a spori toleranța la erori a fost concretizat printr-o metodă de calcul – relativ imprecis dar eficient din punct de vedere al implementării HW – bazat pe Histograma Orientării Gradienților (HOG) folosită apoi de un algoritm de IA de tip SVM.

Controlul prin IA al fluxului de date din HW a fost concretizat prin implementarea HW a unui bloc de corecție a distorsiunilor (GDE) – controlat de un algoritm IA care folosește atât meta-date rezultate din fluxul video principal cât și date ale fluxului secundar preluat de senzorii inerțiali.

Am concluzionat că implementările HW prezentate beneficiază într-o mare măsură de ajutorul IA – proiectarea acestora făcându-se în tandem cu a algoritmilor care le asigură controlul inteligent.

Capitolul 5. Hardware pentru Inteligența Artificială în PI

Așa cum s-a arătat în capitolul 3, cealaltă modalitate prin care IA poate conlucra cu sub-sistemul HW în cadrul sistemelor de TR este *optimizarea implementărilor HW* astfel încât algoritmii de IA specifici să poată rula eficient pe aceste resurse HW.

Această categorie de soluții a fost denumită generic, în lucrare, *"HW pentru IA",* în ideea în care nu mai avem de a face cu modalități în care algoritmul IA gestionează modulele HW (așa cum a fost detaliat în capitolul 4). Considerăm aici modulele HW care funcționează independent, cu scopul de a efectua operațiile dedicate implementării algoritmilor de IA. În acest fel e completată o *relație "în buclă",* prin care se asigură o legătură strânsă dintre componentele HW, SW și algoritmice în cadrul implementării eficiente de sisteme PI cu IA.

În acest capitol se pune accentul pe două direcții de optimizare HW pentru a implementa algoritmii IA. Prima direcție este de implementare HW a rețelelor neurale convoluționale prin optimizarea capacității de procesare. Aceasta se poate face prin implementare masiv-paralelă, cu un număr mare de unități de calcul (de ordinul sutelor sau miilor). În cazul sistemelor încorporate *("embedded systems"),* un număr mare de unități de calcul duce la un consum de energie ridicat, așa cum am arătat în capitolul 1 – de aceea soluțiile propuse trebuie în mod necesar să fie dezvoltate pentru o eficiență energetică crescută.

A doua direcție de optimizare este aceea prin care se pune accentul pe utilizarea cât mai eficientă a magistralei de date. Așa cum am arătat în capitolul 3, magistrala de date poate constitui un "loc îngust" pentru sistemele de TR, mai ales pentru cele cu IA, care folosesc cantități mari de date pentru procesare. De aceea am propus o modalitate de proiectare HW orientată pe folosirea eficientă a magistralei de date pentru algoritmii de PI folosind IA.

5.1 Optimizarea capacității de procesare pentru algoritmii de IA prin implementare HW masiv-paralelă eficientă energetic

Optimizarea capacității de procesare pentru algoritmii de IA a fost realizată în etape. Pentru implementarea în TR a rețelelor neurale convoluționale am dezvoltat un modul dedicat de procesare ("Motor" CNN sau PCNN – *Programmable Convolutional Neural Network)* care implementează un număr mare de blocuri de calcul de tip MAC (Multiplicare – ACumulare) cu consum mic de energie.

Pentru extinderea funcționăriii PCNN am abordat opțiunea de *periferic cu IA.* Aceasta a permis explorarea soluției multi-procesor, în care mai multe module funcționează în paralel pentru execuția unor sarcini IA mai complexe.

În cele din urmă am abordat modul în care aceste soluții muti-procesor pot asigura o redundanță în funcționare, cerință esențială mai ales pentru domeniile cu exigențe speciale – cum ar fi domeniul auto căruia îi sunt dedicate o serie de soluții prezentate în capitolul aplicativ (cap.6).

Procesarea HW pentru rețele neurale convoluționale

Ideea soluției propuse este următoarea: O rețea neurală convoluțională *(CNN – Convolutional Neural Network)* pentru un sistem de procesare a imaginilor cuprinde o memorie cache care corespunde la o cerere de a citi un bloc de pixeli N×M dintr-o hartă de intrare pentru a furniza un bloc de pixeli N×M la ieșire. Un motor de calcul al convoluțiilor citește blocuri de pixeli de la ieșirea memorie cache, combină aceste blocuri de pixeli cu un set corespunzător de ponderi pentru a furniza produsul matricial corespunzător și supune produsul unei funcții de activare pentru a oferi o valoare a pixelilor rezultanți.

Memoria cache cuprinde o multitudine de memorii intercalate capabile să furnizeze simultan pixelii N×M la portul de ieșire într-un singur tact de ceas. Un controler oferă un set de ponderi motorului de convoluție înainte de a procesa o hartă de intrare, determină motorul de calcul al convoluțiilor să parcurgă harta de intrare prin incrementarea unei locații specificate

pentru blocuri succesive de pixeli și generează o hartă de ieșire în memoria cache prin scrierea valorilor pixelilor de ieșire în locații succesive

Odată determinată structura CNN, adică hărțile de intrare, numărul de straturi convoluționale, numărul de hărți de ieșire, dimensiunea nucleului de convoluție, gradul de sub-eșantionare, numărul de straturi complet conectate (și valoarea maximă a vectorilor lor, pentru normalizare) precum și ponderile care urmează să fie utilizate în nucleele stratului de convoluție și straturile complet conectate utilizate pentru clasificarea entităților sunt determinate prin antrenarea pe baza unui set de date eșantion care conține instanțe pozitive și negative etichetate ale unei anumite clase (de exemplu, fețe etichetate ca zâmbitoare și regiuni de interes care conțin fețe care nu zâmbesc). Platforme adecvate pentru facilitarea antrenării unei CNN sunt disponibile de la: Theano [123], Caffe [124], PyThorch [125] sau TensorFlow [126]. *Structura aleasă este cea care se pretează la ajustarea iterativă prin IA, pentru a optimiza performanța CNN.*

Este util să încorporăm un "Motor" CNN într-un sistem de procesare a imaginilor, astfel încât clasificarea caracteristicilor să poată fi efectuată în timp real pe măsură ce imaginile sunt achiziționate sau cel puțin la scurt timp după aceea. De exemplu, o CNN ar putea fi încorporată într-un sistem de achiziție de imagini, cum este cel prezentat în Capitolul 4 [17-CZ-P], [18-CZ-P], [19-CZ-P].

Pentru a face toate acestea, viteza de răspuns și memoria folosită pentru CNN sunt criterii de performanță critice.

Motorul CNN (PCNN – *Programmable Convolutional Neural Network)* poate fi ușor încorporat într-un sistem de procesare a imaginilor; motorul CNN poate fi configurat programabil pentru a funcționa cu diferite caracteristici care extrag straturi convoluționale, precum și straturi de clasificare a caracteristicilor. Motorul CNN include o memorie cache de imagini cu o arhitectură "pipeline" capabilă să furnizeze rapid informații despre harta de intrare unui motor de convoluție, astfel încât o convoluție 3D care implică un anumit număr de hărți de intrare să poată fi efectuată într-un număr minim de cicluri de ceas.

PCNN *(Programmable CNN)* este un sistem *configurabil* în care blocuri de pixeli de diferite dimensiuni pot fi citite simultan dintr-o memorie cache a imaginii pentru prelucrare spre a produce date de ieșire.

Detalii de implementare

Figura 37 prezintă o diagramă bloc a PCNN implementat într-un sistem de achiziție a imaginilor. PCNN se conectează la o magistrală de sistem și poate accesa memoria principală (DRAM) în care sunt scrise imaginile achiziționate. Fluxul de date pentru achiziția de imagini poate furniza și imagini preprocesate care sunt scrise în DRAM. Aceste metode au fost descrise în detaliu în Capitolul 4.

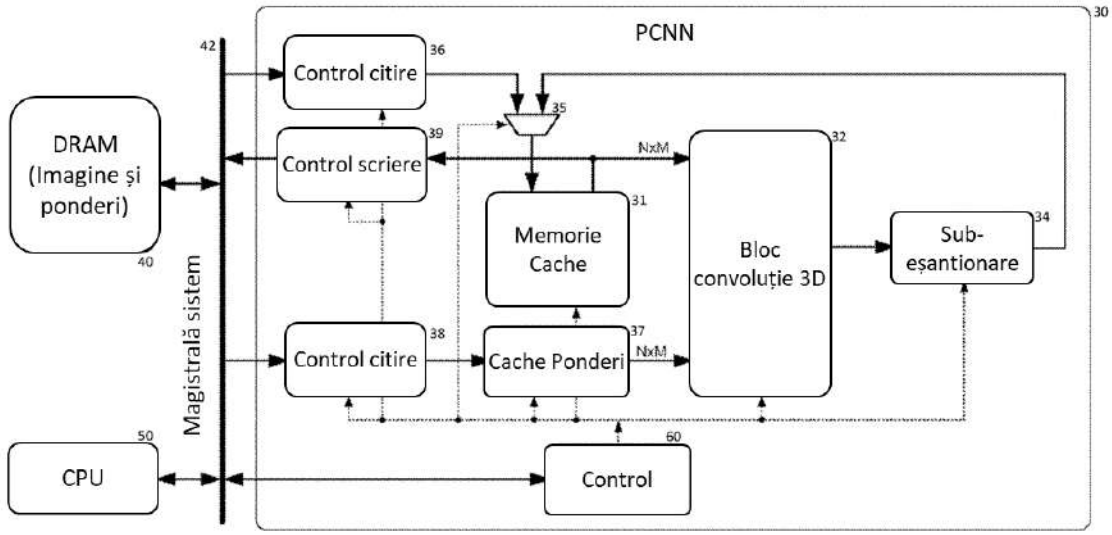


Figura 37: Diagrama bloc a PCNN

O aplicație de TR executată de un CPU (50) semnalează unui bloc de control din cadrul PCNN (60), pe magistrala de sistem, că trebuie analizată și clasificată o regiune de interes (ROI) a unei imagini stocate în DRAM și descrisă de CPU. Blocul de control poate fi implementat ca o mașină de stare fixă sau poate fi programabil (un procesor). În ambele cazuri, succesiunea configurabilă a operațiunilor de extragere și clasificare a caracteristicilor care trebuie efectuate de PCNN este determinată de CPU prin setarea registrelor în cadrul blocului de control, prin magistrala de sistem.

Odată configurat, blocul de control comunică apoi cu diferite module din cadrul PCNN pentru a citi informațiile necesare despre imagine (sau despre harta de imagine) și informațiile despre ponderile CNN din DRAM și pentru a procesa aceste informații înainte de a furniza o clasificare a informațiilor despre imagine către DRAM sau către aplicația apelantă pentru a fi utilizate ulterior.

PCNN (30) cuprinde următoarele module:

O **memorie cache** expune un port de intrare de date (din) și un port de ieșire de date *(dout)* pentru restul motorului CNN. Datele sunt citite prin portul de intrare a datelor *(din)* fie de la DRAM prin intermediul controlerului de citire, fie de la ieșirea unui motor de convoluție prin sub-eșantionare.

Multiplexorul este configurat pentru a permite furnizarea inițială a datelor de imagine (hartă) prin intermediul controlerului de citire din DRAM, dar ulterior și pentru ca informațiile generate de diferitele straturi ale motorului de convoluție (și sub-eșantionare) să fie citite înapoi în memoria cache.

În Figura 38, o imagine (sau hartă ROI) inițială este încărcată mai întâi în memoria cache și se scrie de obicei de la adresa de start (0x00, offset 0) din memoria cache. Este de notat că memoria cache este abordată într-un mod 2D cu o adresă – care cuprinde un rând de adrese și un offset la adresă. După primul strat de convoluție și sub-eșantionare, un număr de hărți, 5 în acest exemplu, Stratul 1 Harta 0 . . . . Stratul 1 Harta 4, sunt generate. În acest caz, ele sunt scrise pentru a adresa liniile din memoria cache a imaginii (31) după ultima locație de adresă a imaginii/hărții ROI inițiale.

Deoarece aceste hărți sunt de obicei sub-eșantionate în raport cu imaginea (harta ROI) inițială la o scară de 2, patru dintre aceste hărți pot fi scrise în același interval de adrese ca imaginea (harta ROI) inițială. De asemenea, deoarece lățimea acestor hărți este o fracțiune din cea a imaginii/hărții ROI, mai multe hărți de nivel 1 pot fi scrise în același spațiu de adrese (scrise prin deplasare una față de cealaltă cu o lățime a hărții – în vederea compensării, "*image stride*").

După cum se va vedea, pe măsură ce procesarea progresează, pentru fiecare strat nou, numărul de hărți poate fi mai mare, dar dimensiunea hărților scade de obicei din cauza sub-eșantionării.

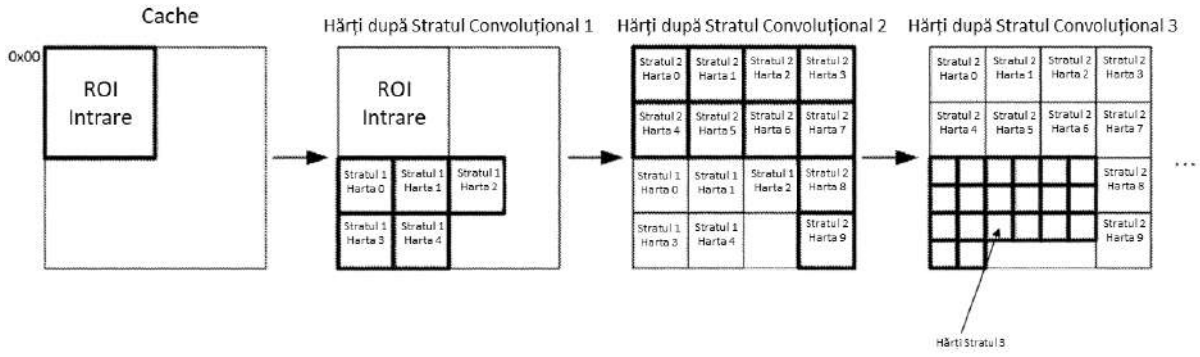


Figura 38: Exemplu pentru organizarea memoriei cache pentru imagini de intrare și hărți intermediare

De notat că în scopul accesului la memoria cache, vectorii produși de straturile de clasificare a obiectelor spațiale sunt tratați ca și cum ar fi hărți 1D care conțin celule de 1×W cu valori ale pixelilor.

În fiecare caz, o locație de pornire, care cuprinde adresa de bază, deplasarea (offset-ul) unei hărți sau al unui vector din memoria cache este determinată de modulul de control, în funcție de configurația primită de la CPU.

Memoria cache este încărcată inițial cu o imagine / hartă de intrare din DRAM. Apoi, toată procesarea poate fi efectuată numai folosind această memorie cache fără a fi nevoie de a accesa DRAM extern pentru informații despre imagine, până

când clasificarea este completă. După fiecare etapă de convoluție/sub-eșantionare, hărțile/vectorii de imagini intermediari sunt scrise înapoi într-o altă zonă a memoriei cache determinată de blocul de control. Aceste hărți/vectori se pot citi apoi, pentru procesare, de către următorul strat de procesare, înainte de a scrie alte hărți/vectori intermediari sau finali înapoi într-o altă zonă a memoriei cache. Hărțile/vectorii scriși în memoria cache pot suprascrie hărți/vectori din mai mult de o iterație anterioară, de exemplu, hărțile de ieșire de la stratul 2 pot suprascrie imaginea/harta de intrare originală, deoarece aceste date nu mai sunt necesare pentru straturile ulterioare.

Brevetarea soluției

Soluția de accelerare HW a rețelelor neurale convoluționale (PCNN), [22-CZ-P], a fost propusă pentru patentare în China (CN108701236A) și la nivel global (WO2017129325A1) în anul 2016. Patentul a fost aprobat în S.U.A. în anul 2017 și conține 26 de revendicări acceptate (US9665799B1).

Tot în anul 2017, s-a propus o îmbunătățire a soluției, [23-CZ-P], prin propunerea US20170221176A1 în Statele Unite și EP3408798A1 în Europa. Această îmbunătățire se referă la folosirea unui nou format de date, în virgulă fixă pe 8 biți, și la calculul diferit al offset-ului. Propunerea conține 28 de revendicări pentru a detalia modul de funcționare a soluției. Propunerile au fost aprobate în 2019 atât în Statele Unite (US10497089B2) cât și în Europa (EP3408798B1).

În anul 2020 s-a continuat patentul [24-CZ-P] cu o propunere (US20200126178A1) de optimizare a managementului memoriei. Patentul aprobat în 2021 are 20 de revendicări aprobate (US11087433B2).

Periferice multi-procesor pentru IA

Soluția propusă cuprinde un sub-sistem periferic pentru procesarea datelor, care este conectat printr-o interfață fizică la un dispozitiv de calcul gazdă (”host”). Perifericul conține un controler local, conectat la memoria locală printr-o magistrală internă și oferă acces dispozitivului gazdă la datele stocate pe perifericul de procesare printr-un API (”Application Programming Interface”) dedicat sistemului de fișiere.

Procesorul neural cuprinde un număr de sub-sisteme (”motoare”) de procesare a algoritmilor neurali, care pot fi programate. Memoria stochează informații despre configurația rețelei, despre imaginea de intrare, informații intermediare și informații de ieșire produse de fiecare ”motor” de procesare a rețelei neurale. Controlerul local este configurat să primească informații de configurare a rețelei și despre imaginea de intrare printr-o comandă de scriere către API al sistemului de fișiere, iar pentru informațiile de ieșire din memoria locală pentru regăsire de către gazdă, tot printr-o comandă de citire via API din sistemul de fișiere.

Motorul programabil CNN (PCNN), prezentat anterior în Figura 37, propune un ***procesor neural*** de uz general, capabil să încarce rețelele (de exemplu, echivalentul programului într-un procesor tradițional) printr-un canal de memorie separat de datele imaginii de intrare și de rezultate.

PCNN (Programmable CNN) cuprinde un procesor iterativ care rulează rețelele neurale strat cu strat. Configurația straturilor, fie în mod specific convoluțională, complet conectată, pentru grupare sau pentru de-grupare, tipurile și valorile ponderilor și funcțiile de activare ale neuronului sunt toate programabile prin definiția rețelei.

Utilizarea unui astfel de procesor neural împreună cu un procesor gazdă separat poate reduce cerințele de putere pentru un sistem, permițând în același timp rețelei neurale să funcționeze în TR pe fluxuri de imagini. Cu toate acestea, PCNN trebuie să fie integrat în sistem încă din procesul de fabricație.

În literatura de specialitate s-a propus mutarea tuturor circuitelor logice și a întregii memorii temporare necesare unui CNN pe un periferic de mare viteză și construirea de co-procesoare cu acceleratoare CNN dedicate și lățime de bandă la memoria locală foarte mare, așa cum a fost prezentat în capitolul 2. Cu toate acestea, astfel de soluții necesită drivere de nivel scăzut în sistemul gazdă, astfel că nu sunt ușor de implementat alături de sistemele existente.

Un periferic de procesare poate fi implementat astfel încât să apară pentru sistemul gazdă ca un periferic de stocare care este susținut de aproape toate sistemele de operare printr-o interfață fizică standard de mare viteză, de exemplu, USB, SDIO, SATA, UFS, NVMe, eMMC. Aplicațiile pot fi distribuite pe spațiul de stocare furnizat pe perifericul propriu-zis, încărcat după cum este necesar și apoi executat la cerere de către un dispozitiv gazdă.

Implementarea soluției propuse

În Figura 39 este prezentat un periferic de procesare care oferă o sistemului gazdă acces la un grup de CNNs programabile, folosind o metodă RPC *(Remote Procedure Call)* printr-un API standard al sistemului de fișiere (susținut de aproape toate sistemele gazdă la care este atașat un dispozitiv de stocare). Gazda în acest caz este un CPU – pentru un sistem de calcul la care este conectat perifericul de procesare. Ca atare, dispozitivul de calcul care poate fi un smartphone, tabletă, laptop, computer desktop sau dispozitiv de calcul personalizat (cum ar fi o cameră de securitate), include un slot pentru card SD sau un port USB ori un conector periferic echivalent. Gazda poate cuprinde unul sau mai multe procesoare separate care comunică cu dispozitivul periferic de procesare printr-o interfață exterioară.

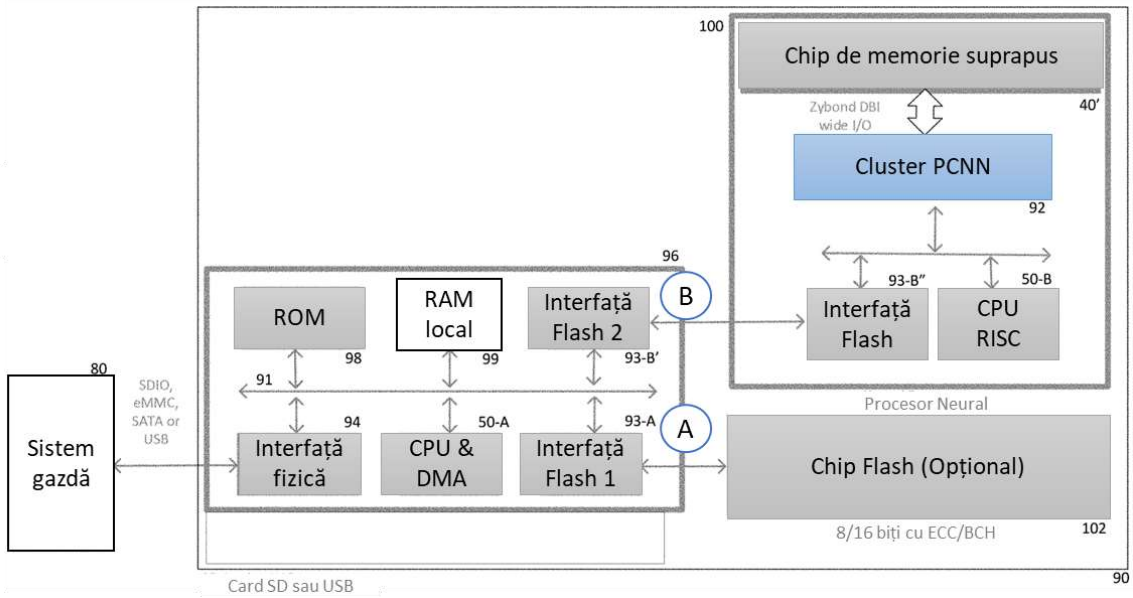


Figura 39: Arhitectura unui periferic de procesare în care PCNN e mapat în spațiul de adresare a memoriei Flash

Sistemul de fișiere acționează ca un suport pentru datele de intrare în clusterul PCNN, precum și pentru rezultatele CNN.

Sistemul de fișiere este, de asemenea, utilizat de gazdă pentru a transfera datele de rețea (configurația stratului, tipul, ponderile, instrucțiunile) către clusterul PCNN și pentru a controla funcționarea (pornirea, oprirea etc.) acestuia.

În implementarea ilustrată în Figura 39, perifericul de procesare este implementat într-un card SD sau un "stick" USB cu o interfață fizică convențională. Astfel de dispozitive de stocare periferică cuprind de obicei un controler, inclusiv un procesor cu acces la memoria programului și memoria volatilă printr-o magistrală de sistem paralelă. Interfața fizică convertește semnalizarea serială între dispozitivul periferic și gazdă, folosind oricare dintre protocoalele convenționale (inclusiv SDIO, eMMC, SATA sau USB), în date paralele – pentru comunicarea pe magistrala paralelă.

În cardurile SD convenționale sau USB "sticks", CPU gazdă citește / scrie informații din / în memoria Flash printr-o interfață specifică Flash în timp ce mapează comunicațiile de la gazdă la informațiile stocate. Este comun ca pentru astfel de dispozitive să includă o interfață Flash dual, astfel încât datele pot fi scrise / citite în paralel în / de la memorie Flash printr-o pereche de interfețe, pentru a accelera accesul gazdei la dispozitiv. Un exemplu de astfel de control "dual channel" este familia F4 de controlere de memorie Flash de la Hyperstone GmbH.

O primă propunere de implementare presupune ca în loc de interfațarea memoriei Flash cu a doua interfață Flash, un procesor neural care încorporează clusterul PCNN este accesat prin a doua interfață Flash și propria interfață Flash omoloagă.

CPU poate fi acum re-programat (prin actualizarea conținutului memoriei ROM), astfel încât accesul la memoria Flash să nu se mai facă pe ambele interfețe Flash (A si B) ci doar prin interfața A – în unele scenarii de utilizare, perifericul nu trebuie neapărat să funcționeze ca unul de stocare, iar atunci această funcție poate fi eliminată prin desemnarea memoriei Flash ca un chip opțional.

Separat, CPU (A) este programat să răspundă la accesarea dedicată a sistemului de fișiere pentru a scrie / citi informații prin a doua interfață Flash (B) la / de la un al doilea CPU (B) în cadrul procesorului neural.

Figura 40 ilustrează un Cluster PCNN în detaliu. Clusterul PCNN include propriul CPU (care comunică cu procesorul gazdă) rezident pe dispozitiv. Clusterul include un număr, în acest caz 4, de CNN programabile independent (30-A ... 30-D) de tipul celor prezentate în Figura 37. În clusterul PCNN, CNNs individuale nu trebuie să fie aceleași și, de exemplu, unul sau mai multe CNNs ar putea avea caracteristici diferite față de celelalte. De exemplu, un CNN poate permite ca un număr mai mare de canale să fie combinate într-o convoluție decât celelalte, iar aceste informații ar fi utilizate atunci când se configurează PCNN. Fiecare CNN (30-A ... 30-D), mai degrabă decât să funcționeze în principal folosind fie memoria de sistem (99), (102) utilizează în schimb o memorie partajată (40'). Astfel, procesorul gazdă, în legătură cu CPU al clusterului și cu un controler de memorie (210) pregătesc pentru transfer imaginile inițiale, precum și informații de configurare a rețelei, din memoria (99), (102) în memoria partajată (40'). Pentru a facilita un astfel de transfer, procesorul gazdă (50) / (50-A, 50B) poate folosi o memorie cache (52).

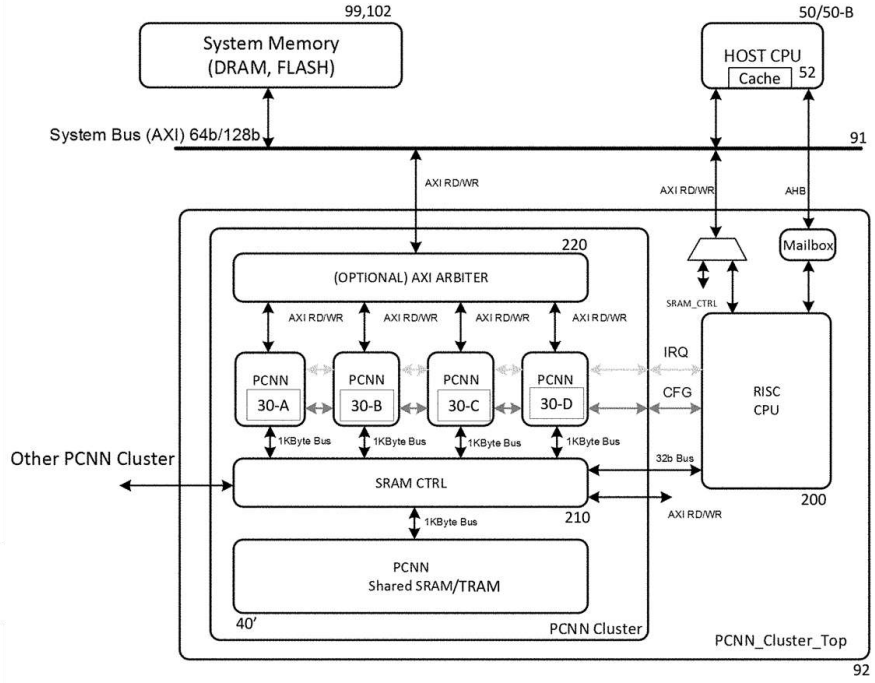


Figura 40: Arhitectura unui cluster PCNN

Figura 41: Detalii de implementare a memoriei unui cluster PCNN

Din punctul de vedere al clusterului PCNN (92), sistemul funcționează după cum urmează:

1. Controlerul gazdă (în legătură cu CPU gazdă) și CPU din PCNN încarcă configurația rețelei neurale și ponderile din memoria sistemului gazdă în memoria partajată, fie prin intermediul memoriei sistemului periferic (în cazul în care informațiile au fost generate dinamic de gazdă), fie de la memoria Flash (în cazul în care informațiile sunt pre-stocate). Informațiile de configurare a rețelei trebuie scrise numai din partea controlerului gazdă de fiecare dată când o rețea / un

algoritm trebuie să fie schimbat(ă) / inițializat(ă) și, prin urmare, acest lucru ar trebui să fie mai puțin frecvent față de rata la care sunt procesate imaginile.

2. Controlerul gazdă (în legătură cu CPU gazdă) și CPU din PCNN încarcă imaginile de intrare prin intermediul memoriei sistemului periferic în memoria partajată. Aceste informații sunt cel mai probabil generate dinamic de gazdă și, prin urmare, nu sunt stocate ca fișier în memoria Flash, dar este totuși posibil ca unele aplicații care rulează pe gazdă să poată extrage informații inițiale despre imagine din fișierele pre-stocate în memoria Flash.

3. Registrele de control în cadrul fiecărui PCNN (30-A ... 30-D) sunt configurate în funcție de programul de rețea neurală și adresele de ponderi din memoria partajată și în conformitate cu imaginea sau harta care urmează să fie prelucrate de către PCNN.

4. PCNN (30-A ... 30-D) poate fi acum activat și, odată activat, controlerul din fiecare PCNN activat citește hărțile de intrare din memoria partajată a sistemului în conformitate cu setările din registrele de control și procesează hărțile așa cum este specificat de programul de rețea neurală. Hărțile intermediare pot fi stocate local în fiecare PCNN (30-A) într-o memorie cache de imagini, așa cum s-a arătat în capitolul anterior sau temporar în memoria sistemului, la adrese configurabile.

5. După finalizarea procesării, rezultatele sunt scrise în memoria sistemului.

6. PCNN comută în modul inactiv și emite o întrerupere a procesorului PCNN.

7. CPU al PCNN poate scrie acum informații înapoi la CPU gazdă, astfel încât acesta să poată citi fișierele sau sub-directoarele corespunzătoare \CNNxx\dataout. CPU poate notifica acum gazda că rezultatele sunt disponibile pentru prelucrare ulterioară, după cum este necesar.

În timp ce transferul de informații cu memoria sistemului sau memoria Flash poate fi canalizat prin procesorul gazdă, este de asemenea posibil să se furnizeze fiecare PCNN (30-A ... 30-D) într-un cluster cu acces la magistrala de sistem, printr-un arbitru comun.

Implementările obținute, cum ar fi cele ilustrate în Figura 40, sunt capabile să proceseze între 30 și 60 de cadre/secundă, atât timp cât transferul de date in/out nu depășește lățimea de bandă disponibilă prin interfața fizică.

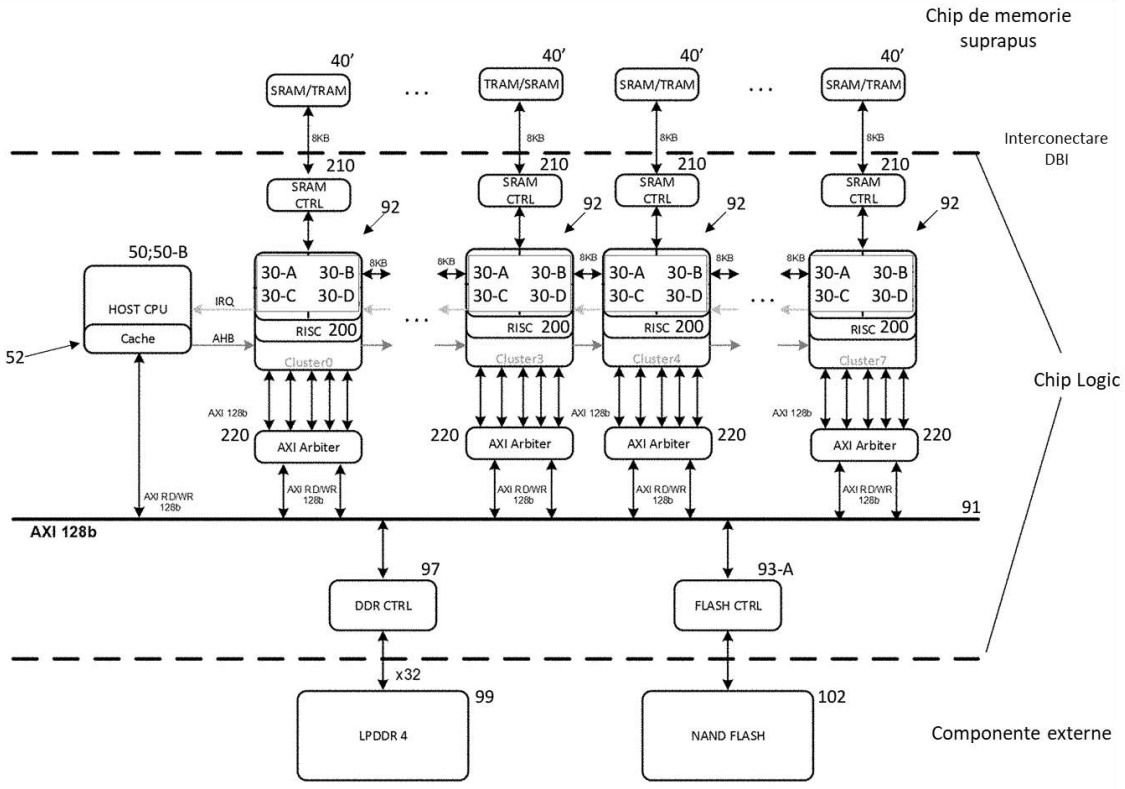


Figura 42: Arhitectura unui cluster PCNN pe chip-uri suprapuse

În Figura 42 este arătat cum este posibil să se realizeze clustere PCNN în lanț sub controlul unui anumit procesor gazdă în cooperare cu procesorul local PCNN (RISC). Furnizarea unui număr atât de mare de CNNs programabile individual (de exemplu, în acest caz 4×8 = 32) poate fi deosebit de utilă pentru a permite rularea rețelelor care cuprind mai multe straturi diferite. Compilatoarele convenționale din mediile Caffe, TensorFlow, Torch, Theano, MatConvNet etc. pot fi utilizate pentru a proiecta rețele care pot fi distribuite pe mai multe nuclee și clustere PCNN, prezentate în Figura 42.

Clusterele înlănțuite secvențial pot comunica sub controlul procesorului gazdă și al procesoarelor locale PCNN, astfel încât rezultatele procesării de către un CNN în cadrul unui anumit cluster să poată fi comunicate prin intermediul controlerului SRAM pentru prelucrare ulterioară de către un alt CNN în același grup sau într-un alt cluster, fără a fi nevoie să fie reprogramată configurația rețelei pentru orice CNN dat. Alternativ, arhitectura poate fi utilizată pentru a implementa rețele de procesare complet separate, permițând procesorului neural să funcționeze în mai multe moduri pentru a găzdui fie aplicații de filtrare a obiectelor, fie aplicații de detectare și recunoaștere a obiectelor, de exemplu, care operează:

(i) rețele independente (mici) pe aceleași date imagine;

(ii) rețele independente (mici) pe diferite date imagine; sau

(iii) o rețea (mare) pe aceleași date imagine.

Indiferent dacă este disponibil un singur nucleu sau mai multe, așa cum se arată în Figura 42, valorile straturilor intermediare (hărți de obiecte spațiale) pot fi scrise fie înapoi la memoria principală (în special în cazul în care e disponibil accesul AXI – Advanced eXtensible Interface – printr-un arbitru de bus), fie la memoria partajată locală.

După cum se indică în Figura 42, arhitectura poate fi implementată cu *chip-uri suprapuse*. Metodele de suprapunere pentru astfel de chip-uri includ ZiBond și DBI (Direct Bond Interconnect – de la Invensas) și acestea oferă conexiuni 3D între chip-uri care să permită interconectare cu un număr mare de semnale. Astfel, modulele logice care cuprind gazda, interfețele de memorie și clustere PCNN (într-o măsură) pot fi puse în aplicare pe un chip cu memorie partajată pe unul sau mai multe chip-uri.

Deci memoria partajată poate fi implementată fie:

(i) Numai SRAM – stocarea atât a configurației rețelei, cât și a datelor stratului intermediar;

(ii) Ca o combinație de SRAM (pentru a menține configurația rețelei) și TRAM (RAM tranzitoriu) pentru a stoca datele stratului intermediar);

- În TRAM, memoria e implementată cu celule tranzitorii unice (sarcina pentru fiecare celulă programată este stocată în capacitatea substratului MOS). Astfel de celule permit aplicarea unei încărcări pentru fiecare bit de memorie. Datorită faptului că utilizarea informațiilor este extrem de rapidă (de ordinul milisecundelor) nu e nevoie de circuite pentru reîmprospătarea memoriei, ca în cazul DRAM convențional.

(iii) numai TRAM - numai datele stratului intermediar vor fi stocate aici, informațiile de configurare a rețelei fiind preluate de CPU PCNN din memoria Flash opțională; aceasta ar putea oferi un sistem mai compact atâta timp cât densitatea Flash depășește pe cea a SRAM.

Memoria internă pentru CNN individuale ale clusterelor ar putea fi, de asemenea, implementată pe chip-uri separate. În orice caz, matricea de memorie separată poate folosi un "nod tehnologic" eterogen (tehnologii diferite de la un chip la celălalt, sau față de chip-ul logic). Deci, în timp ce chip-ul logic poate fi proiectat folosind tehnologia cu cea mai mare densitate (de exemplu, 10 nm), matricea de memorie poate fi proiectată folosind o tehnologie mai ieftină (de exemplu, de 28 nm).

Un aspect important al implementărilor descrise mai sus e că sistemul de fișiere nu garantează citirea / scrierea *imediată,* astfel că sistemul de fișiere va încerca să memoreze în cache / să întârzie citirea / scrierea cât mai mult posibil. Sistemul de fișiere nu garantează nici *ordinea* scrierilor în diferite fișiere (din nou din cauza memorării în cache).

În finalul acestui paragraf trebuie menționat că informațiile despre ponderi și configurația de rețea utilizate în cadrul clusterelor PCNN (92) pot fi rezultatul unei instruiri a-priori extinse și, eventual, bazate pe seturi mari de date care pot implica resurse semnificative pentru asamblare și etichetare. Din acest motiv, poate fi extrem de util să se protejeze astfel de informații prin tehnici criptografice – înainte de scrierea lor în memorie. De exemplu, datele despre ponderi și configurare pot fi criptate folosind informații cheie specifice unui dispozitiv, astfel încât, dacă astfel de date au fost copiate din memoria unui anumit dispozitiv, acestea să nu poată fi apoi implementate într-un alt dispozitiv. Aceste tehnici, împreună cu alte metode și sisteme de securizare, pot fi utilizate pentru a proteja aceste informații împotriva accesului neautorizat.

Brevetarea soluției

Soluția de implementare a perifericelor multi-procesor cu IA a fost propusă pentru patentare în 2019 [15-CZ-P], în Statele Unite (US2019065410-A1), Europa (EP3580691-A1 și EP3699816-A1), China (CN111052134-A) precum și la nivel global (WO2019042703-A1). Până în prezent, au fost acordate brevete în Statele Unite (US10713186-B2) și Europa (EP3580691-B1), în celelalte jurisdicții patentele fiind în așteptare.

Redundanța multi-procesor în rețele neurale convoluționale

Un sistem bazat pe rețele neurale multi-procesor cuprinde o multitudine de motoare de procesare a rețelei, fiecare pentru procesarea unuia sau mai multor straturi ale unei rețele neuronale în funcție de o configurație de rețea. O memorie stochează cel puțin temporar informații de configurare a rețelei, informații despre imaginea de intrare, informații intermediare despre imagine și informații de ieșire pentru motoarele de procesare a rețelei.

Cel puțin unul dintre "motoarele" de procesare a rețelei este configurat (în rest fiind inactiv) pentru a identifica informațiile de configurare și informațiile de imagine de intrare care urmează să fie procesate de un alt motor de procesare a rețelei (țintă) și pentru a utiliza informațiile de configurare și informațiile de imagine de intrare pentru a reproduce procesarea motorului de procesare a rețelei. Acest motor este configurat pentru a compara cel puțin o parte din informațiile obținute de motorul de procesare a rețelei (țintă) cu informațiile corespunzătoare generate de motorul de procesare a rețelei pentru a determina dacă fie motorul de procesare a rețelei țintă, fie motorul de procesare a rețelei, funcționează corect.

Soluția propusă și brevetată a fost implementată într-un sistem de monitorizare a conducătorului auto (DMS – Driving Monitoring System) având arhitectura din Figura 43.

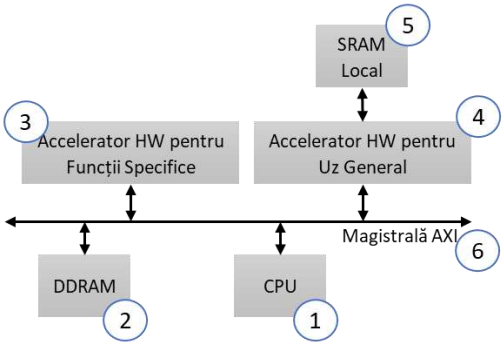


Figura 43: Arhitectura tipică a unui sistem de monitorizare a conducătorului auto (DMS)

Astfel de sisteme pot conține un procesor gazdă (1), eventual *multi-core,* și memorie de sistem (2), de exemplu un modul de memorie LPDDR4 (cu un singur sau cu mai multe canale). Pot include de asemenea modulele de co-procesare (3), (4) ce pot cuprinde acceleratoare HW de uz general (4), (de exemplu pentru rețele neurale convoluționale *programabile* – PCNN) sau diverse blocuri DSP (Digital Signal Processing – de procesare a semnalului digital), așa cum am prezentat în secțiunea anterioară sau [127], sau acceleratoare HW dedicate unor funcții specifice (3), (de exemplu pentru detectarea feței, cum am prezentat în capitolul 4 – TMM, Template Matching Module – sau corecția distorsiunii imaginii așa cum am prezentat în capitolul 4 – GDE, Geometrical Distortion Engine).

Atât procesorul de uz general (1), cât și acceleratorul HW de uz general (4) și acceleratorul HW pentru funcții specifice (3) primesc informații fie direct, fie din memoria (2) prin magistrala de sistem (6) de la diferiți senzori plasați în jurul unui vehicul, pentru a furniza informații necesare sub-sistemelor de control, ECU (Electronic Control Units).

Înainte de a fi încorporate într-un vehicul, sistemele auto trebuie, în general, să se conformeze unor standarde de siguranță – cum ar fi ISO 26262 [128], [173] care specifică nivelul de integritate a siguranței autovehiculului (ASIL – Automotive Safety Integrity Levels). Începând cu cel mai scăzut, ASIL-A, continuând cu B și C, se ajunge la ASIL-D care este cel mai înalt nivel de siguranță utilizat în industria auto.

Primul mecanism utilizat pentru a asigura că acceleratoarele de procesare oferă nivelul de siguranță ASIL-D este *redundanța.* În acest caz, mai multe acceleratoare de procesare execută fiecare aceeași funcție, apoi rezultatele de la fiecare accelerator de procesare sunt comparate și orice diferență semnalată unui procesor gazdă. Aceasta reprezintă, desigur, măsuri acoperitoare de siguranță, dar necesită o arie de siliciu și un consum de energie crescute față de o implementare non-redundantă.

În continuare, prezint modul în care se pot folosi un sub-sistem de procesare al unui cluster PCNN, pentru a rula temporar o configurație (aferentă unei programări PCNN), aceeași configurație care e dedicată de fapt unui alt sub-sistem de procesare. Rezultatele de la fiecare PCNN sunt comparate și, dacă nu sunt egale, se poate identifica o defecțiune la unul din ele sau la celălalt.

Dacă rezultatele sunt identice, după rularea în acest mod redundant, primul din cele două sub-sisteme poate reveni la propria sarcină desemnată.

Fiecare cluster PCNN poate funcționa fie în mod independent (ca în Figura 44) sau în mod redundant (ca în Figura 45).

Implementările se bazează pe un cluster (92) sau un CNN (30) individual care este prevăzut atât cu imagini de intrare sau hărți, cât și cu configurația rețelei necesară pentru a fi executată de CNN, adică definițiile pentru fiecare strat al unei rețele și ponderile care urmează să fie utilizate în diferitele straturi ale rețelei, de fiecare dată când este vorba de a procesa una sau mai multe imagini / hărți de intrare.

Poate fi de dorit să se ruleze diferite rețele la momente de timp diferite și frecvențe diferite. De exemplu, într-un vehicul cu una sau mai multe camere frontale, un cluster PCNN dedicat identificării pietonilor în câmpul vizual al camerei poate fi executat la peste 30 de cadre pe secundă, în timp ce într-un vehicul cu o cameră orientată spre șofer, un cluster PCNN dedicat identificării expresiilor faciale ale șoferului se poate executa la mult sub 30 de cadre pe secundă. În mod similar, unele rețele pot fi mai complexe decât altele și, prin urmare, pot implica timpi de procesare diferiți, chiar dacă sunt executate la aceeași frecvență.

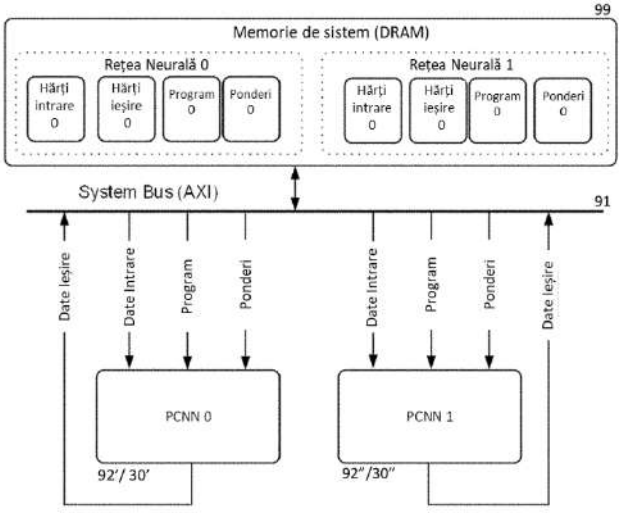


Figura 44: Funcționarea PCNN în mod independent

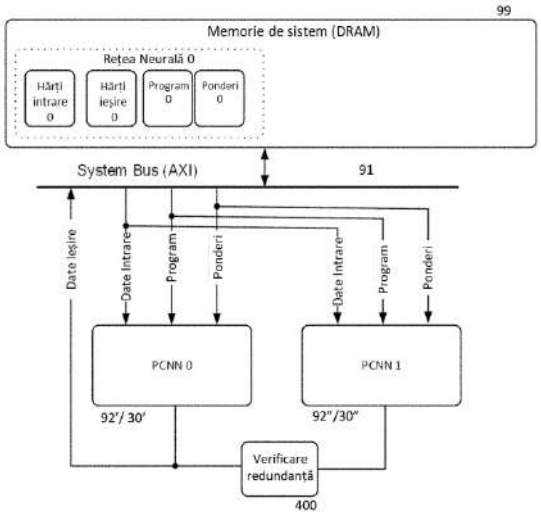


Figura 45: Funcționarea PCNN în mod redundant

În felul acesta, vor exista perioade de timp în care unul sau mai multe dintre clusterele PCNN, într-un sistem multi-procesor pentru rețele neurale, va fi inactiv (așa cum se arată în Figura 44).

Unele dintre aceste clustere PCNN pot fi puse sub controlul procesorului gazdă, direct sau prin intermediul procesoarelor clusterului respectiv, fiind capabile să preia comenzile și datele programului destinat altor CNN țintă, fie din memorie fie prin magistrala de sistem.

În aceste cazuri, așa cum este ilustrat în Figura 45, un cluster PCNN poate fi comutat să funcționeze într-un mod redundant, caz în care, folosind hărțile de intrare, informații de configurare / definiția programului și ponderile pentru celălalt cluster PCNN ("țintă"), va reproduce procesarea din clusterul PCNN țintă.

Fiecare astfel de cluster PCNN, atunci când operează în modul redundant, poate continua să execute programul specific PCNN țintă, fie până când procesarea este finalizată, fie până când clusterul PCNN primește o comandă de la procesorul gazdă care îi solicită să treacă la execuția propriului program de rețea în mod independent.

Rezultatele procesării, chiar rezultatele parțiale, pot fi apoi comparate prin folosirea unui CPU la dispoziție din cadrul clusterului PCNN, după cum indică blocul de decizie (400).

Utilizarea unui CPU comun unui număr de CNN individuale, pentru a efectua o verificare a redundanței unui cluster PCNN individual, elimină sarcina de la CPU gazdă – de a identifica oportunitățile de efectuare a testelor – și, de asemenea, reduce circuitele logice care trebuie implementate pentru a furniza o astfel de funcționalitate fiecărui CNN individual. În mod similar, deoarece fiecare CPU oferă acces pentru fiecare CNN individual la magistrala de sistem (91), acesta poate acționa cu ușurință *în numele lor* pentru a identifica oportunitățile de testare a altor clustere PCNN.

Brevetarea contribuțiilor

Conceptul de redundanță pentru rețele multi-procesor descris anterior a fost propus pentru patentare în anul 2020, [16-CZ-P]. Cele 11 revendicări au fost prezentate în Statele Unite (US20200184321A1), Europa (EP3668015A2 și EP3668015A2) precum și în China (CN111311476A). Toate revendicările au fost aprobate în Europa în 2021 (EP3668015B1).

5.2 Proiectarea HW orientată pe utilizarea eficientă a magistralei de date

Un sistem de PI cuprinde un modul de normalizare conectat printr-o magistrală la memoria care stochează o imagine în care a fost identificată o regiune de interes *(ROI – Region Of Interest)*.

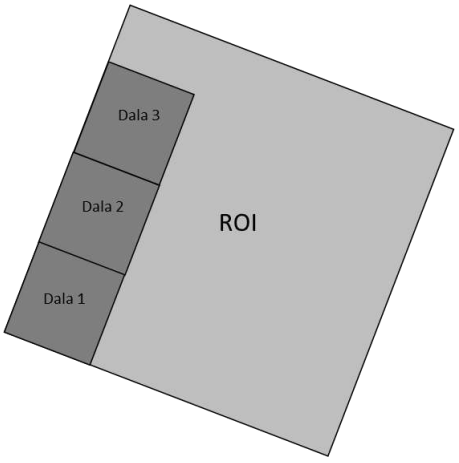


Figura 46: Regiunea de Interes divizată în "dale"

ROI e încadrată de un dreptunghi care are o orientare non-ortogonală în cadrul imaginii (Figura 46). Modulul de normalizare este organizat pentru a împărți regiunea de interes într-una sau mai multe felii, fiecare felie cuprinzând o multitudine de "dale" ("plăcuțe" – *"tiles")* dreptunghiulare adiacente. Pentru fiecare felie, sistemul citește succesiv informații ale ROI pentru fiecare "dală" din memorie (citirea unei porțiuni a imaginii care se întinde cel puțin pe lățimea feliei). Pentru fiecare dală, sistemul sub-eșantionează ROI într-o memorie tampon, la o scală (S_D <2) necesară pentru versiunea normalizată a ROI. Sistemul sub-eșantionează apoi (cu o scală fracțională) și rotește respectiva dală din memoria tampon pentru a produce porțiunea normalizată respectivă din ROI (la scara necesară pentru ROI normalizată).

Datele sub-eșantionate și rotite sunt acumulate pentru fiecare dală din memoria tampon în ROI normalizat, pentru prelucrarea ulterioară de către sistemul de PI.

Soluția propusă în continuare cuprinde o tehnică îmbunătățită pentru normalizarea unei ROI din interiorul unei imagini.

În această soluție, termenul normalizat este utilizat pentru o versiune a unei ROI care are o orientare ortogonală, astfel încât să poată fi memorată ca o zonă dreptunghiulară regulată de o anumită dimensiune (sau una dintr-un număr limitat de dimensiuni date). De asemenea, poate fi util ca și conținutul din ROI să aibă o anumită orientare, astfel încât, dacă, de exemplu, un obiect dintr-un ROI a fost detectat cu o orientare semnificativ diferită de verticala obișnuită – cum ar fi răsturnat cu susul în jos sau pe-o parte –versiunea normalizată să stocheze întotdeauna obiectul detectat într-o orientare verticală.

În Figura 47, este prezentat un bloc de normalizare (NORM) pentru un sistem de PI (10). Ca și în cazul corecției distorsiunilor (caz prezentat în capitolul 4), un modul de pre-procesare/filtrare (20) primește o imagine de la un senzor și scrie o imagine filtrată (40-1) printr-o magistrală de sistem (30), la memorie (40).

Un modul de corecție a distorsiunii (50) poate corecta acum imaginea filtrată (40-1) – versiunea corectată (40-2) a imaginii filtrate (40-1) se scrie apoi în memorie (40).

Obiectele de interes pot fi fețe și atunci un modul (60) de detectare și urmărire a feței (tot în cazul corecției distorsiunilor, introdus în capitolul 4 și detaliat aici) este utilizat pentru a identifica orice ROI (o față, în acest exemplu) în cadrul imaginii corectate de distorsiuni (40-2). Meta-datele care indică locația oricărei astfel de ROI în imaginea (40-2) pot fi stocate fie împreună cu imaginea, fie în orice altă porțiune adecvată a memoriei (40), unde datele pot fi preluate de alte module de PI.

În funcție de varietatea de clasificatori de detectare a feței sau de forma de urmărire utilizată de modulul de detectare și urmărire a feței, modulul (60) poate identifica fețele într-un număr de orientări diferite în cadrul imaginii corectate, fără distorsiuni, (40-2). De asemenea, în funcție de varietatea de dimensiuni ale clasificatorului utilizat pentru identificarea fețelor, și dimensiunea ROI ce reprezintă fețele detectate poate varia semnificativ.

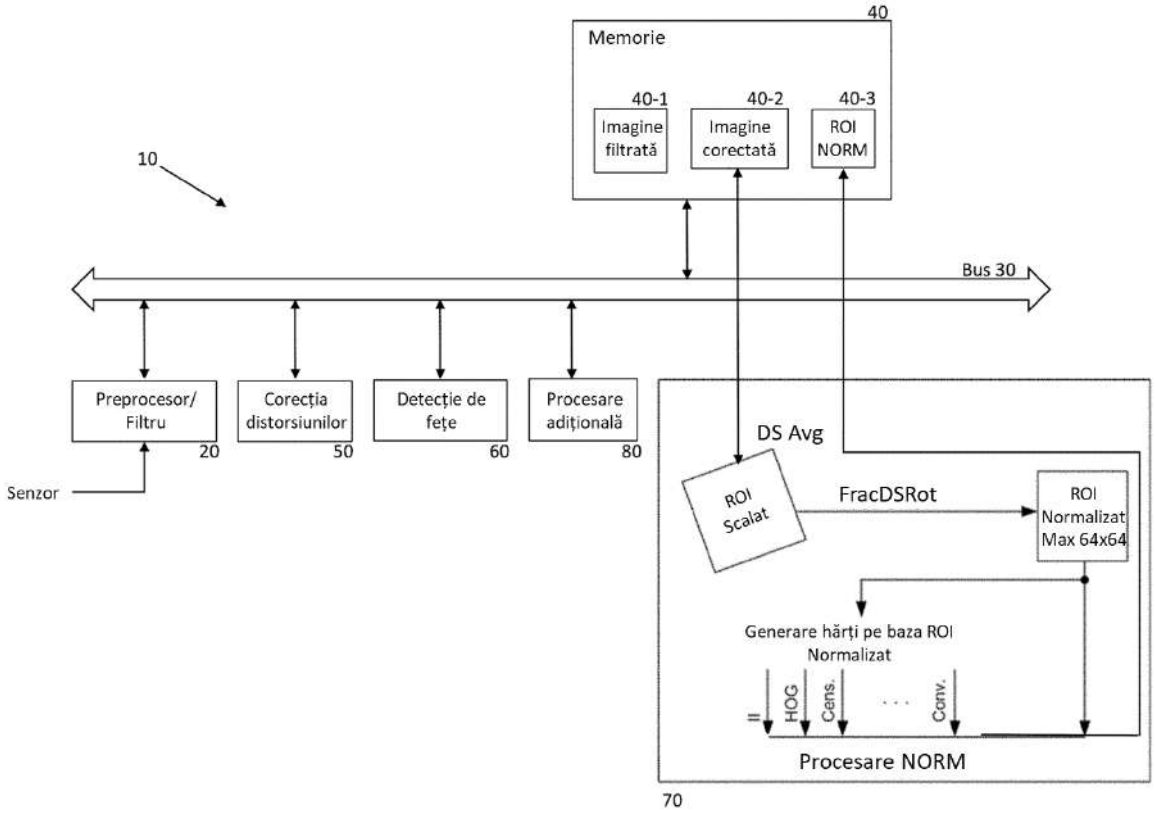


Figura 47: Blocul de Normalizare (NORM) într-un sistem de PI

Un modul de normalizare (70) identifică orice ROI într-o imagine corectată (40-2) și re-eșantionează această ROI pentru a oferi o versiune normalizată (40-3), NORM-ROI, scrisă în memoria (40). Această versiune normalizată poate fi apoi utilizată de modulele ulterioare de procesare (80), cum ar fi recunoașterea facială. Această prelucrare ulterioară poate fi efectuată fie prin module de procesare dedicate, fie prin CPU (ne-reprezentat) – pentru dispozitive de PI: camere digitale statice, camere video digitale, camere pentru smartphones, tablete sau orice calculator cu sub-sistem video.

De asemenea, odată ce versiunea normalizată (40-3) a unei ROI a fost calculată, modulul de normalizare (70) poate efectua o PI ulterioară pe versiunea normalizată (40-3), pentru a furniza informații utilizate în mod obișnuit de modulele de prelucrare ulterioară (80), după cum se va descrie mai detaliat în continuare.

În Figura 48, este prezentată ordinea de citire a "dalelor" din jumătățile de cadrane 0 și 1. Pentru ROI situate la astfel de unghiuri non-cardinale, modulul de normalizare pornește citirea de la o adresă situată în afara ROI - *Rd Ctrl starting point.*

Pentru jumătățile de cadrane numerotate par – 0, 2, 4 și 6, feliile sunt citite rând cu rând de sus în jos, așa cum se arată în Figura 48a; iar pentru jumătățile de cadrane impare numerotate 1, 3, 5 și 7, acestea sunt citite de jos în sus, așa cum se arată în Figura 48b.

Liniile din fiecare felie sunt întotdeauna citite din memorie începând de la marginea din stânga a primei dale. Ulterior, pixelii pentru fiecare dală sunt citiți rând cu rând, până când se ating datele pentru ultima linie a ultimei dale. Folosind punctul cel mai din stânga al unei dale și fie punctul de sus (pentru cadranele pare) sau de jos (pentru cadranele impare) al unei dale ca adresă de pornire, se evită necesitatea de a calcula trigonometric o adresă de pornire prin extrapolare.

Procesul se repetă apoi în același mod pentru fiecare felie dintr-o ROI, adresele fiind citite deplasat cu lățimea feliilor.

După cum este ilustrat în Figura 48, pornind de la această margine din stânga a plăcii 1 și de la o linie corespunzătoare fie părții de sus, fie celei de jos a plăcii în funcție de jumătatea cadranului, modulul (70) citește porțiunile de linie ale ROI din memoria (40). În Figura 48, *Rd Ctrl starting point* este afișat ca și coincizând cu marginea cea mai din stânga a plăcii, dar, în practică, această adresă va corespunde cu adresa unui cuvânt de memorie ce începe imediat în partea stângă a marginii din stânga a dalei.

Lățimea porțiunii de linie citite din memorie trebuie să fie suficient de mare pentru a prelua toate datele dalelor dintr-o ROI din memoria (40) și astfel se extinde cu numărul de cuvinte de memorie (plus 1) necesare pentru a stoca o lățime a dalei de pixeli (WTILE = *Tile width)* rotită la un unghi (Θ = 22,5 °). Numărul de cuvinte necesar pentru a stoca pixeli este WTILE / cos (Θ).

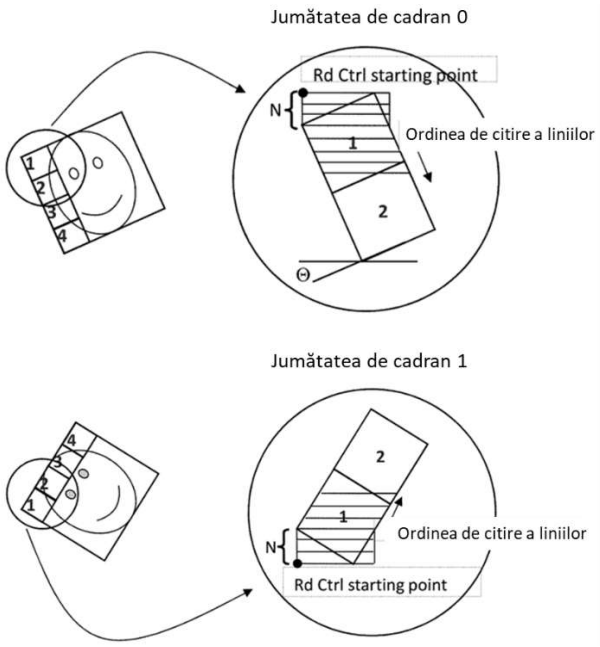


Figura 48: Modul de citire a regiunilor unui obiect de interes

După cum se observă din Figura 48, lățimea fiecărei porțiuni de linie citite pentru o felie este constantă de-a lungul acesteia și astfel adresa de pornire a citirii pentru fiecare linie avansează de-a lungul unei felii pentru a ne asigura că porțiunea de linie citită din memorie preia datele necesare dalei. După cum voi arăta în continuare, pentru prima dală dintr-o felie (în acest caz dala 1), acest avans este neliniar prin faptul că adresa de pornire a citirii este aceeași până când procesul ajunge la linia care conține punctul cel mai din stânga al dalei și, ulterior, adresa de pornire a citirii avansează după cum e necesar pentru a ne asigura că toate datele dalei se încadrează în porțiunea de linie citită. Pentru dalele ulterioare, avansul este liniar de-a lungul feliei.

Odată ce modulul de normalizare (70) a citit porțiunile de linie de care are nevoie pentru o "dală", acestea sunt prelucrate, așa cum e ilustrat mai detaliat în Figura 49.

Fiecare dală de intrare este sub-eșantionată și rotită în 2 pași: sub-eșantionare medie, sau la cel mai apropiat vecin *(DSAvg),* urmată de sub-eșantionare bi-liniară (cu scală fracțională) sau la cel mai apropiat vecin, cu o scală maximă de 1,99 cu rotație *(FracDSRot).*

Astfel, *DSAvg* sub-eșantionează o porțiune de dală, deoarece este citit din memorie la într-o scară de două ori dimensiunea necesară pentru dală în ROI normalizat. *DSAvg* scrie informațiile sale sub-eșantionate la o memorie tampon *DSAvgBuf* care, în exemplul ilustrat, are o dimensiune totală de 32×32 pixeli. Dalele sub-eșantionate produse de pasul *DSAvg* pot fi stocate în 4 memorii intercalate în *DSAvgBuf,* astfel încât în timpul procesării de *fracDSRot,* 4 pixeli pot fi citiți în fiecare ciclu de ceas și interpolați pentru a produce un singur pixel al ROI normalizată.

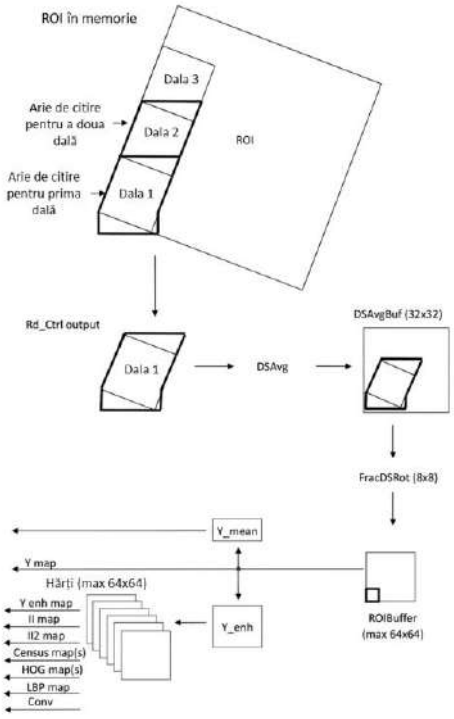


Figura 49: Modul de procesare a dalelor

Desigur, *DSAvgBuf* ar putea fi organizat diferit dacă *FracDSRot* ar folosi alte scheme de interpolare, cum ar fi interpolarea bi-cubică.

FracDSRot mapează apoi date sub-eșantionate pentru fiecare dală la o ieșire de 8×8 pixeli care este scrisă într-o memorie tampon *ROIBuffer* având o dimensiune totală de 64×64 pixeli (astfel, în acest exemplu, doar un sfert din această memorie va fi umplut pentru o anumită ROI).

Locația în *ROIBuffer* a informațiilor normalizate pentru o dală depinde de cadranul citit și de răsturnarea sau nu a ROI în timpul normalizării.

Odată ce procesarea primei dale dintr-o felie este finalizată, porțiunile de linie pentru dalele ulterioare sunt citite din memoria (40) începând de la linia ce urmează ultimei linii a primei dale, și sub-eșantionate în *DSAvgBuf,* astfel încât nu este necesară o citire duplicată a memoriei pentru citirea dalelor dintr-o anumită felie; singurele informații suplimentare citite din memorie sunt zona "non-dală" (vizibilă de exemplu sub dala 1 în partea de sus din Figura 49) din primele linii ale primei dale și ultimele linii ale ultimei dale a unei felii. Această citire suplimentară ar putea fi redusă în continuare, dar cu o complexitate crescută ce rezultă din faptul că nu se citește același număr de cuvinte pe linie pentru o felie completă.

Procesarea continuă până când o felie este completă – atunci când linia care conține partea de sus (sau de jos) a unei felii este citită din memorie. *DSAvgBuf* poate fi apoi golit, resetat sau suprascris și procesul repetat pentru fiecare felie, până când ROI este complet și toate dalele pentru un ROI au fost scrise în memoria tampon ROI.

Informațiile din aceste dale pot fi răsturnate orizontal și/sau vertical în timpul etapelor *DSAvg* și/sau *FracDSRot,* astfel încât conținutul memoriei tampon ROI să corespundă unui obiect într-o anumită orientare, indiferent de orientarea obiectului în cadrul imaginii capturate inițial. Alternativ, acest lucru ar putea fi făcut odată ce *ROIBuffer* a fost parcurs.

Chiar și luând în considerare aranjamentul neregulat al informațiilor despre dala sub-eșantionată din *DSAvgBuf,* ar fi posibil să se reducă dimensiunea *DSAvgBuf* la mai puțin de 32×32 pixeli, dar această reducere nu este critică.

Brevetarea contribuțiilor

Propunerea de analiză în HW a ROI cu eficientizarea modului de acces la memoria principală s-a făcut în anul 2017 [20-CZ-P] în S.U.A. (US20170061639A1), Europa (/EP3341912A1), China (CN108352054A) precum și la nivel global (WO2017032468A1). Toate cele 22 de revendicări au fost aprobate în anul 2018 în S.U.A. (US10115003B2) și în Europa (EP3341912B1).

O idee nouă, care a adăugat valoare modului de funcționare a nucleului HW a fost aceea de a realiza rotirea și scalarea imaginii într-un singur pas [21-CZ-P]. Propunerea aceasta a fost trimisă pentru patentare în 2019 (US20190130164A1) și a fost aprobată în 2021 (US11106894B2). Toate cele 20 de revendicări au fost aprobate.

5.3 Sumarul capitolului

În capitolul 5 am prezentat soluțiile de tip HW pentru IA. Prima categorie de soluții a fost în direcția de optimizare a capacității de procesare pentru algoritmii de IA prin implementare HW masiv-paralelă eficientă energetic. Pentru aceasta am introdus blocul HW de tip PCNN, care este un procesor neural cu utilitate extinsă datorită programabilității. Programarea acestuia nu se face cu instrucțiuni (ca la CPU) ci cu *parametrii* straturilor rețelelor neurale care sunt rulate.

Am explorat modul în care PCNN poate fi folosit în periferice conectate la un sistem gazdă prin interfețe de tip USB sau SDIO și care pot transforma un simplu periferic de stocare într-un dispozitiv inteligent care să ruleze algoritmi neurali. În această direcție am extins folosirea PCNN la sisteme multi-procesor prin construirea de clustere PCNN care împart resurse comune (de memorie și control).

În cadrul sistemelor multiprocesor, am arătat modul în care mai multe PCNN pot fi folosite redundant, pentru detecția erorilor de funcționare sau a defectelor de fabricație, foarte utile în sistemele din domeniul auto (o parte din acestea vor fi prezentate în capitolul 6).

Proiectarea HW orientată pe utilizarea eficientă a magistralei de date a fost exemplificată prin implementarea unui modul HW de normalizare (NORM) care folosește magistrala de date într-un mod original, astfel încât operațiile de rotire și scalare să folosească cât mai puțină bandă de transfer la memorie.

Am concluzionat că modulele HW propuse și implementate pot ajuta la creșterea performanțelor algoritmilor de IA, fie prin creșterea capacității de procesare fie prin folosirea eficientă a memoriei – ambele abordări esențiale în dezvoltarea de soluții de PI în TR cu IA.

Capitolul 6. Aplicații industriale bazate pe accelerarea HW pentru IA

6.1 Etapele implementării - Codesign pentru IA

Aplicațiile implementate pe baza soluțiilor prezentate în capitolele 4 și 5 au fost dezvoltate după un proces care a avut ca element de bază conceptul de *co-design.* Așa cum am arătat încă din capitolul 1, procesarea de TR cu IA trebuie abordată ca eco-sistem, în care datele, algoritmii, SW dar și HW trebuie analizate, concepute și construite împreună, orice decizie de modificare a uneia din componente putând afecta modul de funcționare a celorlalte.

Mai mult, în timpul procesului de implementare pentru aplicații industriale, există influențe care vin din afara sferei strict tehnice, cu implicații *legale* (în principal în domeniul protecției *proprietății intelectuale* sau al securității datelor) sau *comerciale* (care țin de aspectele financiare și de colaborare între entitățile *business).*

Aceste influențe pot avea impact asupra deciziilor tehnice și trebuie analizate atent în cadrul procesului de cercetare-dezvoltare.

De aceea, procesul prezentat în continuare pune accent pe ideea de *co-design,* în care elementul principal este algoritmul de implementat, în jurul căruia se iau măsurile potrivite de construire a soluției, pentru a maximiza factorul de merit.

Pentru a implementa aplicații bazate pe accelerarea HW pentru IA, am parcurs etapele de proiectare prezentatea în continuare. Am marcat (*) etapele specifice codesign-ului HW-SW cu adaptarea specificațiilor funcționale ale unui sub-sistem luând în considerare cerințele celuilalt sub-sistem pentru domeniul IA. Principial, codesign-ul HW-SW pornește de la un *algoritm de nivel înalt* – o arhitectură de tip *mașină algoritmică de stare,* "mai presus de HW sau SW". Pentru a obține *partiționarea HW-SW* – folosind o anumită orientare (în cazul de față către aplicații IA de TR), "mapată" ("desfășurată" – *"deployed")* pe soluțiile propuse în capitolele anterioare, și *disponibilitatea* acestor resurse speciale – se va hotărî asupra "concretizării" unor "blocuri" ale algoritmului de nivel înalt: fie ca blocuri SW (ale unei "organigrame" – "diagrame logice") fie ca blocuri HW (module ale schemei constructive). Acest co-design urcă, așadar, până la nivelul superior, algoritmic (în această lucrare am evidențiat algoritmii care *rulează direct în HW,* spre deosebire de acea cotă, acea parte de algoritmi care vor rula, în mod tradițional, în SW). *În co-design pentru IA am urmărit păstrarea la fiecare nivel și în orice etapă de proiectare a posibilității de alegere între HW sau SW și ridicarea până la nivelul algoritmului generic a acestei dualități.*

Aceste etape de proiectare sunt: *Planificarea aplicației; Definirea cerințelor aplicației; Descrierea algoritmului IA la nivel înalt; Testarea rezultatelor algoritmului; Analiza complexității; Alocarea de resurse (*); Partiționarea HW-SW (*); Definirea interfețelor dintre HW și SW; Implementarea algoritmului în SW; Modelarea matematică a algoritmului implementat în HW (*); Verificarea funcționării algoritmului (*); Analiza detaliată a KPI (profilare) (*); Implementarea algoritmului în HW (nucleul IP); Prototiparea aplicației pe FPGA (demonstrator); Măsurarea KPI pentru fiecare scenariu de utilizare; Validarea rezultatelor aplicației; Extragerea ideilor pentru proprietatea intelectuală; Definirea și implementarea documentației colaterale; Integrarea nucleului IP în circuitul de producție (*); Integrarea algoritmului SW în sistem; Validarea funcționării sistemului; Măsurarea calității aplicației de producție (*); Măsurarea resurselor folosite de aplicație; Livrarea pentru producție; Suport pentru utilizare.*

Stadiile de maturitate tehnologică – TRL (Technology Readiness Levels)

Pentru aplicațiile industriale detaliate în continuare, am indicat stadiile de maturitate tehnologică (TRL – *Technology Readiness Levels)* așa cum au fost definite pentru proiectele publice în Statele Unite [134] și apoi adaptate pentru programele europene [135], [136]. Cele 9 stadii de maturitate tehnologică sunt prezentate în Tabelul 5.

Primele 3 nivele (TRL1, 2 3) corespund activităților de cercetare. La finalul acestor activități, e disponibil un demonstrator (PoC – Proof of Concept) experimental, iar ideile tehologiei pot fi brevetate sau publicate.

Nivelele TRL 4,5,6 corespund activităților de dezvoltare. Tehnologia este validată apoi se poate demonstra prin intermediul prototipurilor – inițial în laborator, apoi într-un mediu relevant pentru aplicație. La finalul acestor activități, avem un sistem robust, care funcționează în mediul relevant pentru aplicații.

Stadiul de maturitate	Descriere
TRL 1	Formularea principiilor de bază
TRL 2	Formularea conceptelor tehnologice
TRL 3	Construirea demonstratorului experimental
TRL 4	Validarea tehnologiei în laborator
TRL 5	Validarea tehnologiei în mediu relevant
TRL 6	Demonstrarea tehnologiei în mediul relevant
TRL 7	Demostrarea prototipului sistemului în mediul operațional
TRL 8	Omologarea sistemul complet
TRL 9	Funcționarea sistemului în mediul operațional

Tabelul 5: Stadiile de maturitate tehnologică (TRL) [135]

Ultimele 3 nivele de maturitate tehnologică (TRL 7,8,9) corespund etapei de implementare. În această etapă, prototipul este demonstrat în mediul operațional, e testat și omologat pentru producție, ultima etapă fiind cea în care sistemul funcționează corect în mediul operațional.

După atingerea TRL9, se poate trece la producția de masă.

6.2 Detecția de obiecte în TR folosind algoritmi de IA implementați în HW

Detecția de obiecte este una dintre cele mai folosite aplicații în PI de TR. Se poate folosi în sisteme de achiziție a imaginilor pentru a configura în mod corect fluxul de achiziție foto/video în cazul apariției obiectelor de interes în imaginile recepționate, caz în care se pot aplica algoritmi de corecție a imaginilor de tip AAA (Auto-focus, Auto-expunere sau Auto-balans de alb, „white balance") analizând zona din imagine a obiectului detectat.

Un alt caz de utilizare este acela în care detecția de obiecte poate declanșa fie alarme (ca în cazul sistemelor de supraveghere video) fie execuția altor algoritmi de PI (ca în cazul sistemelor de recunoaștere).

Partiționarea HW-SW

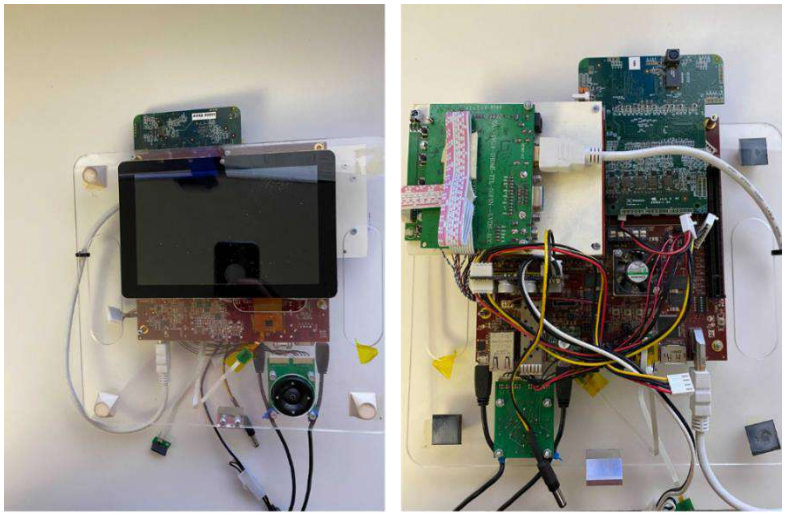
Pentru a satisface cerințele expuse mai sus, am realizat partiționarea HW-SW astfel încât componentele algoritmului care necesită resurse mari de calcul (rezultate în urma analizei profilului algoritmului) să fie implementate în HW, iar componentele care necesită o logică de control complicată să fie implementate în SW.

Pentru aceasta, modulul care face pre-procesarea imaginilor (PRE), modulul care rezolvă problema " potrivirii de șabloane" ("template matching" – TMM) precum și modulul de procesare a regiunilor (AHIP – Advanced Hardware for Image Processing) au fost implementate în HW, iar celelalte componente ale algoritmului au fost implementate în SW.

Implementarea fizică a aplicației

După proiectarea nucleelor HW, și implementarea componentelor SW, sistemul a fost prototipat pe FPGA. În Figura 50 este prezentată platforma FPGA folosită.

Figura 50: Platforma FPGA Xilinx mini-ITX



Partea posterioară a demonstratorului Partea frontală a demonstratorului

Figura 51: Demonstrator FPGA pentru detecția de obiecte

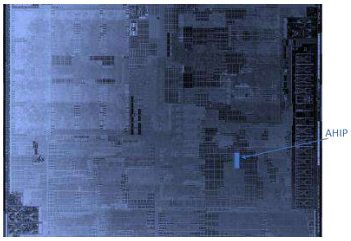


Figura 52: Implementare comercială a unui chip care conține HW pentru detecția de obiecte

Figura 53: Placă de dezvoltare pentru validarea funcționării ASIC

Platforma mini-ITX de la Xilinx a fost preferată pentru că dispozitivul FPGA din familia Zynq 7000 este destul de puternic pentru implementarea nucleelor IP proiectate și în plus are un sub-sistem SW destul de capabil, bazat pe microprocesorul ARM Cortex A9, ce poate rula la 800 MHz. Memoria DRAM disponibilă pe FPGA este generoasă (1 GB PS DDR3 SDRAM) iar interfețele externe disponibile sunt numeroase și relativ ușor de utilizat – QSPI, SATA-III, microSD, Ethernet, HDMI, Audio input and output, USB, PCIe, FMC (FPGA Mezzanine Card) HPC (High Pin Count), JTAG, GPIO).

Pentru a demonstra funcționarea aplicației, a fost nevoie de construirea unor periferice, pentru achiziția și afișarea de imagini. Acestea au fost încorporate într-un demonstrator, așa cum este prezentat în Figura 51. Pe partea frontală a platformei FPGA au fost montate placa pentru senzorul de imagine și driverul HDMI pentru display. Pe partea posterioară a fost montat un ecran pentru afișarea imaginii și a informațiilor suplimentare precum și un PCB (Printed Circuit Board) de control cu butoane. Toate aceste componente au fost montate pe un suport de plexiglass cu mânere, pentru o manevrare ușoară.

Integrarea pe chip s-a făcut conform metodologiei VLSI prezentate anterior. O implementare comercială a fost realizată în tehnologia de 22 nm TSMC (de la Taiwan Semiconductor Manufacturing Company), HPC. SoC realizat conține componentele HW pentru detecția de obiecte, evidențiate în Figura 52.

După implementarea ASIC, funcționalitatea algoritmului a fost validată pe placa de dezvoltare construită în mod special pentru acest scop și prezentată în Figura 53.

În figură se pot observa, pe lângă ASIC, și sistemul de lentile, cu motoarele pentru auto-focalizare, interfețele exterioare pentru analiză și "debugging" (ethernet, USB) precum și pinii de depanare necesari pentru observarea funcționării chip-ului.

Produsul final este cel prezentat în Figura 54. În acest exemplu de implementare e prezentă o cameră foto de calitate înaltă care poate detecta obiecte în mișcare, cu corecția parametrilor de imagine în TR și folosită în special pentru evenimente sportive.

Figura 54: Exemplu de implementare a aplicației de detecție de obiecte într-un produs comercial

6.3 Stabilizarea imaginii pentru sisteme de TR

Stabilizarea imaginii în fluxurile video este o aplicație foarte utilă pentru camerele "de acțiune", care filmează adeseori imagini în mișcare.

Pentru ca aplicația să funcționeze corect, sunt necesare mai mult surse de informații. Acestea ajută la estimarea cu precizie a mișcării camerei și reprezintă sursa corecțiilor care se fac pe imagine.

Trebuie asigurată o sincronizare perfectă între toate aceste surse de informații (achiziția de imagini, giroscop, accelerometru) și modulul de corecție a imaginilor.

Cel mai simplu mod de stabilizare a imaginilor folosește un sistem mecanic numit *gimbal* (o montură inventată în antichitate ca suport ce previne vărsarea unei călimări, ulterior cunoscut și în suspensia cardanică sau în inelele primelor giroscoape).

Acesta funcționează pe principiul inerțial. Avantajele acestui sistem sunt că poate susține un echipament de achiziție a imaginilor voluminos și poate suporta unghiuri mari de rotire a imaginii. Dintre dezavantajele sistemului putem aminti faptul că el însuși este voluminos și greu, consumă o cantitate relativ mare de energie iar timpul de reacție este relativ mare.

Un sistem digital de stabilizare a imaginii presupune un senzor cu un câmp vizual mare, iar stabilizarea se face în domeniul digital, prin corecția imaginilor recepționate folosind date preluate de la mai mulți senzori.

Un sistem de stabilizare digital are ca principal avantaj faptul că nu introduce greutate suplimentară sistemului de achiziție de imagine, consumă puțină energie, are un răspuns extrem de rapid și poate face o corecție avansată a imaginii, inclusiv pentru efectul de "rolling shutter" (efectul de scanare "obturator derulant" – prezentat în continuare). Principalul dezavantaj al soluției constă în limitarea unghiurilor de rotire a imaginii (suportate).

Pentru detecția mișcării senzorului de imagine, se folosesc două tipuri de intrări: cele furnizate de un senzor inerțial (de tip giroscop-accelerometru) și cele furnizate de senzorul de imagine. Este foarte important ca cele două intrări să fie perfect sincronizate, printr-o procedură de sincronizare care marchează pe de o parte timpul exact al fiecărui eșantion recepționat

de la senzorul inerțial, iar pe de altă parte momentele în care senzorul de imagine începe achiziția imaginii (SOF – Start of Frame) și când se termină achiziția imaginii (EOF – End of Frame).

Toate aceste informații, precum și imaginea efectivă de intrare și modelul de proiecție al camerei, intră într-un modul de IA, care le analizează și generează în TR un grid de corecție care va fi folosit pentru corecția fiecărui cadru din imaginea de intrare.

Corecția imaginii se face în HW cu un bloc de tip GDE (Geometrical Distortion Engine), așa cum a fost prezentat în capitolul 4. Rezultatul este o imagine de ieșire, care este corectată de distorsiuni și care în secvență video are mișcarea camerei stabilizată. O altă opțiune poate fi înlocuirea blocului HW de tip GDE cu o procesare pe GPU de uz general, însă această soluție presupune o energie consumată semnificativ mai mare.

În situația în care întreaga procesare prezentată este efectuată în HW, se asigură o latență minimă și, în consecință, o acuratețe maximă.

Implementarea soluției

Pentru implementarea soluției de stabilizare a imaginii, am folosit resursele HW prezentate în capitolul 4, pentru controlul fluxului prin IA al datelor provenite din fuziunea senzorilor.

Blocul HW de corecție a distorsiunilor (GDE) a fost implementat în tandem cu un sistem IA pentru analiza datelor de la senzori care să producă grid-ul de corecție necesar stabilizării de imagini.

Întregul sistem a fost prototipat pe FPGA și testat pe un dispozitiv care induce mișcare controlată (”shaker”).

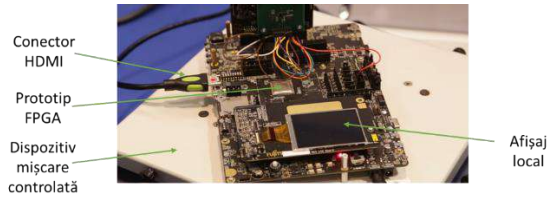


Figura 55: Sistem de prototipare FPGA pentru stabilizarea imaginilor

Figura 56: Circut integrat care implementează sistemul de stabilizare a imaginilor

Figura 57: Cameră de acțiune cu stabilizare de imagine cu fuziunea senzorilor

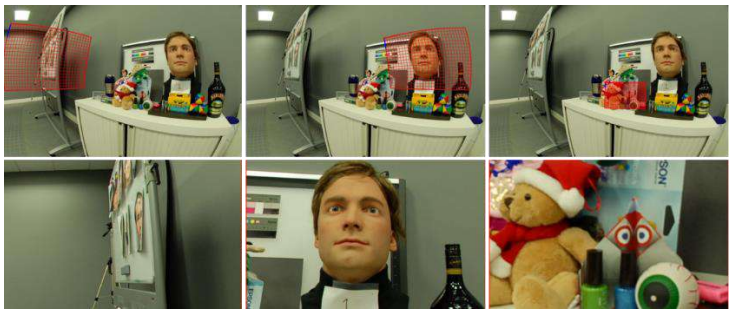


Figura 58: Corecția distorsiunilor pentru Zoom

După validarea funcționării pe FPGA, s-a trecut la integrarea într-un SoC implementat pe tehnologia de 22nm.

Circuitul integrat rezultat este cel prezentat în Figura 56. Acesta conține, pe lângă blocul de corecție a distorsiunilor - GDE, aplicația de stabilizare a imaginilor și un ISP (Image Signal Processor) de calitate superioară.

Interfețele exterioare ale circuitului sunt PCIe Gen2 (cu 2 perechi de căi – ”lanes”), USB2.0 Host/Device, USB3.0 Device, SD, UHS-I/UHS-II, RGMII pentru Gigabit Ethernet. Codarea fluxului video se face cu un codec de tip H.264.

Principalul domeniu de utilizare comercială a circuitului integrat a fost pentru ”camerele de acțiune” (sus-menționate).

Un exemplu de folosire comercială a soluției implementate este în dispozitivul din Figura 57

Pentru testare, s-a analizat modul de corecție a distorsiunilor în scenarii diferite. Câteva exemple cu rezultatele corecției de distorsiuni sunt prezentate în Figura 58.

Pentru toate scenariile analizate s-au efectuat măsurători a calității imaginilor, pe de o parte, precum și a indicatorilor de performanță ai soluției (energie, putere computațională, memorie folosită, lărgimea de bandă precum și latența), pe de-altă parte.

Rezultatele stabilizării de imagine se pot urmări în mod eficient în secvențe video demonstrative. Un exemplu este cel publicat la conferința Embedded Vision Alliance, prezentat în [171]. Un alt exemplu este prezentat online la [172].

6.4 Monitorizarea exteriorului și interiorului autovehiculelor

Soluțiile prezentate în capitolele 4 și 5 pot fi aplicate în domeniul auto prin soluții de monitorizare a interiorului și respectiv al exteriorului autovehiculelor.

Pentru aceste aplicații este nevoie de implementarea algoritmilor de PI folosind IA, în TR, constrângerile principale fiind legate de dimensiunile fizice (de gabarit) ale soluțiilor precum și de modul de disipare a căldurii generate ca urmare a consumului de energie. Aplicațiile prezentate în continuare au în vedere implementarea HW a algoritmilor IA, pentru un răspuns rapid în contextul specific al supravegherii și conducerii autovehiculelor.

Pentru început voi prezenta modul în care IA poate ajuta la supravegherea periferică a unui autovehicul, folosind senzorii de imagine deja existenți pe mașină și controlul câmpului vizual al acestora în funcție de scenariul de utilizare dorit.

În continuare voi prezenta o aplicație de monitorizare a șoferului în timpul conducerii autovehiculului, în acest caz principala problemă rezolvată fiind aceea a implementării unui număr de algoritmi IA care să funcționeze în TR pe un sistem HW cât mai ieftin.

Modificarea asistată de IA a câmpului vizual exterior al vehiculelor

Un vehicul poate include una sau mai multe camere cu montură mobilă – care se poate deplasa pentru a regla unghiul de vizualizare al camerei video în raport cu corpul vehiculului. Camera poate fi mobilă în virtutea faptului că este cuplată la o oglindă laterală mobilă, astfel încât mișcarea oglinzii laterale să schimbe un câmp vizual al camerei cuplată la oglinda laterală. Oglinda laterală poate fi rotită în jurul unei axe de rotație între o primă poziție în care un câmp vizual al camerei este îndreptat într-o primă direcție (de exemplu, spre sol) și o a doua poziție în care câmpul vizual al camerei este îndreptat într-o a doua direcție, diferită de prima direcție (de exemplu, spre exteriorul vehiculului sau spre o ușă a acestuia).

Oglinda laterală poate fi atașată pe corpul vehiculului și poate fi dispusă într-o primă poziție (poziție de desfășurare). În prima poziție, oglinda laterală se poate extinde lateral spre exterior față de vehicul, astfel încât o lungime a oglinzii laterale să fie paralelă cu suprafața pe care se află vehiculul. Oglinda laterală poate fi dispusă în prima poziție atunci când vehiculul se mișcă, vehiculul este ocupat, ușile vehiculului sunt deblocate etc. În prima poziție, camera poate fi dispusă pe o suprafață inferioară sau descendentă a oglinzii laterale, astfel încât o axă optică a camerei să fie îndreptată spre un prim unghi (de exemplu, un unghi sub un orizont). În prima poziție, un câmp vizual al camerei poate fi dispus în jos, astfel încât să ofere unui conducător al vehiculului o vedere asupra solului (sau a altor zone greu de văzut) din jurul vehiculului în timp ce conducătorul auto operează vehiculul. Vizualizarea camerei poate fi afișată pe un afișaj din cabina vehiculului.

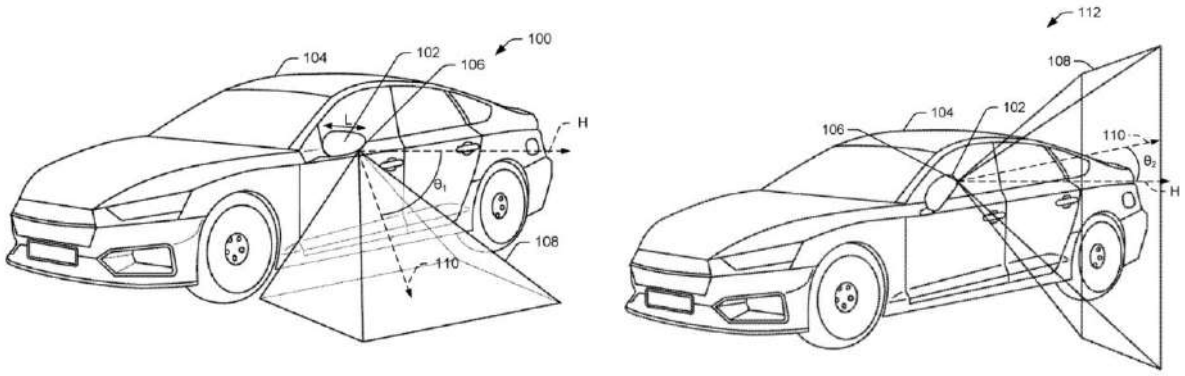
Oglinda laterală poate fi mutată într-o a doua poziție (de exemplu, o poziție înclinată sau pliată). În a doua poziție, oglinda laterală poate fi poziționată astfel încât lungimea oglinzii laterale să se extindă paralel cu fereastra laterală a ușii vehiculului. Oglinda laterală poate fi dispusă în a doua poziție atunci când vehiculul este parcat, vehiculul este staționar, vehiculul este neocupat, ușile sunt blocate, vehiculul "simte" (detectează) un obiect (obiecte) în apropiere. În a doua poziție, camera poate fi orientată astfel încât axa ei optică să fie îndreptată la un al doilea unghi (de exemplu, la orizont sau deasupra orizontului). Prin mutarea oglinzii laterale din prima poziție în a doua poziție, câmpul vizual al camerei poate fi ajustat pentru a captura

o zonă diferită. Câmpul vizual poate fi schimbat de la capturarea unei suprafețe de drum în jurul unei margini a vehiculului (în prima poziție) la captarea unei porțiuni dintr-un mediu care înconjoară vehiculul din care persoanele sau obiectele se pot apropia de vehicul (în a doua poziție). În a doua poziție, camera poate fi utilizată în scopuri de securitate sau în alte scopuri (descrise în continuare). De exemplu, camera poate fi utilizată pentru a identifica un utilizator autorizat al vehiculului, pentru a detecta un potențial furt sau vandalism. În primul caz, un utilizator autorizat al vehiculului poate fi identificat prin prelucrarea datelor imagine capturate de cameră, iar ușile vehiculului pot fi deblocate pentru a permite acestui utilizator autorizat să acceseze vehiculul.

Într-o variantă de implementare, unul sau mai mulți algoritmi de vedere computerizată pot fi executați (pe datele de imagine capturate de la cameră), fie local de către un dispozitiv de calcul al vehiculului, fie de la distanță de către un dispozitiv de calcul conectat la rețea. Algoritmii de vedere computerizată pot fi utilizați pentru a identifica utilizatorii autorizați ai vehiculului, pentru a determina când obiectele se află la o distanță de prag față de vehiculul, pentru a determina când un utilizator neautorizat încearcă să obțină acces la vehicul etc. Vehiculul inteligent poate lua măsuri în funcție de determinarea algoritmului de viziune computerizată. De exemplu, vehiculul poate bloca ușile, poate debloca ușile, poate deschide una sau mai multe uși, poate activa un sistem de securitate al vehiculului, poate trimite o notificare unui utilizator autorizat al vehiculului etc.

Figura 59 a) descrie o vedere de perspectivă a unei prime poziții (100) a unei oglinzi laterale (102) a unui vehicul (104) aflat în mers, oglinda laterală incluzând o cameră video (106). Într-o astfel de poziție, oglinda laterală (102) este orientată pentru a ajuta șoferul să vadă zonele din spatele și din lateralul vehiculului (104), care pot fi în afara vederii periferice a șoferului. Camera (106) poate fi dispusă pe o suprafață inferioară a oglinzii laterale (102), astfel încât câmpul vizual (108) al camerei să fie îndreptat în jos. Liniile câmpului vizual (108) descriu un posibil câmp de vizualizare al camerei (106); trebuie remarcat faptul că liniile câmpului vizual sunt doar prezentate ca exemplu. Câmpul vizual real poate depinde de tipul, designul și/sau orientarea unei anumite camere. De exemplu, în prima poziție (100), o axă optică (110) a camerei (106) poate fi îndreptată la un unghi sub un orizont (așa cum indică linia H). Camera poate fi fixată rigid pe oglinda laterală (102) sau această oglindă poate include unul sau mai multe elemente de acționare pentru a panorama sau înclina camera (106) în raport cu oglinda laterală (102).

Așa cum am menționat anterior, oglinda laterală (102) poate fi dispusă în prima poziție (100), în timp ce șoferul poate fi nevoit să se folosească de ea (șoferul se află în vehicul, vehiculul se află în mișcare, etc). Datele de imagine capturate de camera (106) a oglinzii laterale pot fi afișate pe unul sau mai multe afișaje din interiorul vehiculului (104), permițând astfel șoferului să aibă o vizibilitate sporită în mediul din jurul vehiculului.



a) Poziția camerei din oglindă în mers b) Poziția camerei din oglindă în repaus

Figura 59: Poziția camerei din oglindă în mers

Figura 59 b) prezintă o perspectivă asupra unei a doua poziții (112) a oglinzii laterale (102) a vehiculului (104). Așa cum am menționat anterior, în a doua poziție oglinda laterală poate fi pliată. A doua poziție (112) a oglinzii laterale (102) direcționează o axă optică (110) a camerei (106) la un unghi deasupra orizontului (H). Câmpul vizual (108) al camerei (106)

poate fi direcționat spre exteriorul vehiculului (104), astfel încât camera (106) să capteze date de imagine ale mediului din jurul vehiculului, ale persoanelor sau obiectelor care se apropie de vehicul etc.

În a doua poziție (112), camera poate capta în continuare date de imagine care pot include o porțiune a (sau porțiuni ale) vehiculului. De exemplu, în a doua poziție, câmpul vizual (108) poate captura o zonă din apropierea unui mâner de ușă al vehiculului (104), astfel încât să capteze imagini ale unei zone de care o persoană se poate apropia atunci când încearcă să intre în vehicul. Oglinda laterală (102) poate fi mutată în poziția a doua (112) atunci când oglinda laterală nu este necesară pentru conducătorul vehiculului. De exemplu, oglinda laterală poate fi mutată în a doua poziție atunci când vehiculul este parcat, vehiculul este neocupat, vehiculul este staționar etc. O astfel de mișcare poate include rotație, pliere, translație. Oglinda laterală se poate deplasa de la prima poziție (100) la a doua poziție (112) ca răspuns la oricare dintre evenimentele menționate anterior.

În timp ce vehiculul este ocupat, oglinda laterală (102) poate fi în prima poziție (100), apoi, la trecerea vehiculului în modul de parcare, oglinda laterală se poate deplasa automat în a doua poziție (110).

Brevetarea soluției

Soluția [26-CZ-P] a fost propusă pentru brevetare în anul 2020 (US2020156592-A1). Această propunere conține 20 de revendicări care detaliază modul de folosire a oglinzii mobile pentru monitorizarea exteriorului vehiculului. Această propunere a fost aprobată în 2020 (US10836353-B2) – conține 16 revendicări aprobate.

În anul 2021, au fost propuse îmbunătățiri ale soluției, prin adăugarea de revendicări referitoare la identificarea accesului ne-autorizat în proximitatea autovehiculului [27-CZ-P]. Propunerea (US2021129797-A1) este încă în așteptarea aprobării.

Monitorizarea șoferului în interiorul autovehiculului

Pentru monitorizarea interiorului automobilelor în TR este necesară o suită de algoritmi de IA care să funcționeze în paralel. Acești algoritmi sunt grupați într-o aplicație de tipul DMS *(Driving Monitoring System)* pentru monitorizarea șoferului sau într-o aplicație de tip OMS *(Occupancy Monitoring System)* pentru monitorizarea tuturor ocupanților autovehiculului.

Un exemplu de set de algoritmi care compun aplicația DMS este prezentat în Figura 60.

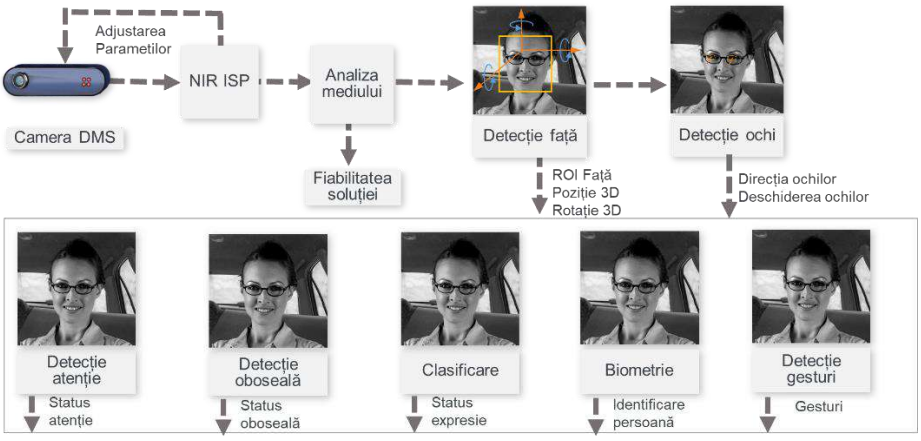


Figura 60: Componentele aplicației de monitorizare a șoferului

Camera DMS funcționează în infraroșu *(NIR – Near InfraRed)* deoarece este necesară iluminarea activă a șoferului pe timp de noapte. Pentru aceasta se folosesc LED-uri NIR, care nu disturbă șoferul sau pasagerii.

Îmbunătățirea calității imaginii se face cu un ISP dedicat, care funcționează pe imagini NIR. Acesta poate ajusta parametrii camerei, de la timpul de expunere sau câștigul acesteia până la intensitatea LED de iluminare.

O dată ce calitatea imaginii este asigurată, se face o verificare a contextului, pentru a asigura fiabilitatea soluției. Astfel, se detectează dacă senzorul este murdar sau obturat sau dacă fluxul de achiziție a imaginii este blocat.

Urmează algoritmii de bază DMS, și anume acela de detecție a feței și de detecție a ochilor. Acești algoritmi de detecție de obiecte pot folosi principiile detaliate în capitolul 4, implementarea HW fiind făcută cu blocuri de tip TMM *(Template Matching Module)* sau AHIP *(Advanced Hardware for Image Processing).* Ca urmare a detecției de obiecte se pot determina ROI (regiunea feței respectiv regiunile ochilor) care apoi sunt analizate suplimentar.

Pentru fețe se determină poziția 3D și orientarea 3D, iar pentru ochi se determină direcția acestora și gradul de închidere/deschidere.

Implementarea demonstratorului

Pentru implementarea algoritmilor IA, am ales un dispozitiv FPGA de cost redus, Xilinx Zynq 7010. Acesta are și avantajul că poate fi produs pentru industria auto (nivelul calitativ *"Automotive Grade").*

Arhitectura soluției implementată pe FPGA este prezentată în Figura 61.

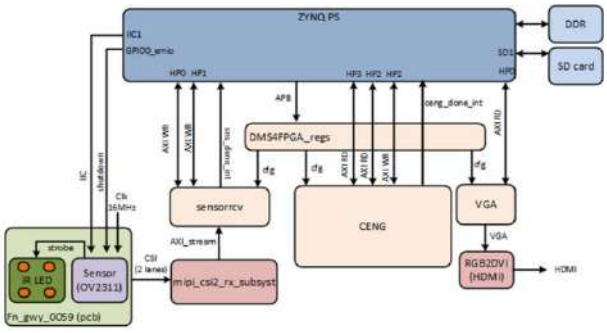


Figura 61: Arhitectura FPGA pentru implementarea DMS

Figura 62: Demonstrator FPGA pentru DMS - varianta 1

Figura 63: Demonstrator FPGA pentru DMS - varianta 2

Figura 64: Testarea demonstratorului DMS

Placa cu senzorul OV2311, precum și componentele FPGA sunt controlate de Zynq PS ("Processing System") care conține CPU și comunică și cu memoria externă (DDR) și memoria nevolatilă (SD Card). Pe FPGA a fost implementat un receptor de imagine, pe interfața MIPI CSI2 (mipi_csi2_rx_subsyst), iar ieșirea este realizată cu un driver VGA, pe interfața HDMI (RGB2DVI-HDMI).

Implementarea algoritmilor neurali s-a făcut în blocul denumit generic CENG *(Convolutional ENGine).* Acest bloc, de tip PCNN (așa cum a fost prezentat în capitolul 5) comunică cu sistemul de procesare și cu memoria prin interfețe AXI. Configurarea și statusul CENG se citesc din/în un set de registre, controlate de CPU (DMS4FPGA_regs).

Au fost implementate 2 sisteme de prototipare. Primul demonstrator a fost construit pentru a valida funcționarea aplicației și a permite alte modificări sau depanarea mai facilă. Pentru aceasta, s-a ales un PCB (cablaj imprimat – Printed Circuit Board) mai mare, de 30x30 cm, iar dispozitivul FPGA folosit a fost unul mai mare, de tip Zynq 7020. Figura 62 prezintă această variantă construită.

Caracteristicile de viteză de calcul sunt:

- Detecția de fețe, implementată în SW, se realizează în 73ms;
- Detecția parametrilor DMS, implementată în HW neural se realizează în 27ms;
- Post-procesarea neurală, implementată în SW, se realizează în 15ms;
- Demonstratorul funcționează cu 15 cadre pe secundă.

Cel de-al doilea demonstrator (nivel TRL6) a fost construit pentru a reduce dimensiunile fizice (pentru o mai bună integrare în autovehicul). Implementarea fizică este prezentată în Figura 63.

Testarea aplicației

Pentru testare, s-a folosit un sistem de test pentru orientarea capului și direcția ochilor, așa cum e prezentat în Figura 64.

Poziția demonstratorului este deasupra volanului, iar câmpul vizual al acestuia este orientat către șofer.

Ieșirea demonstratorului este pe interfața HDMI, iar rezultatele sunt afișate pe un ecran TV de mari dimensiuni. Pe acesta se pot urmări atât modul de detecție a feței șoferului, cât și parametrii DMS.

Metricile de bază DMS includ poziția și unghiurile de rotire 3D ale feței, precum și direcția ochilor.

Metricile suplimentare ale DMS se referă la ocluzii (ochelari, mască, cască telefon) precum și la gradul de închidere a ochilor. Sistemul a fost testat în laborator pe un număr de 23 de subiecți cu caracteristici diferite. Demonstratorul este în curs de testare în autovehicul.

6.5 Sumarul capitolului

În capitolul 6 am prezentat aplicațiile industriale bazate pe soluțiile hibride HW-SW cu IA propuse în capitolele 4 și 5. Pentru aceasta am redefinit etapele implementării aplicațiilor, cu evidențierea modului în care abordarea de tip ecosistem *date-IA-SW-HW* influențează fiecare etapă de proiectare. Am detaliat, de asemenea, nivele de maturitate tehnologică (TRL) și am plasat fiecare dintre soluțiile dezvoltate la nivelul corespunzător.

Aplicația industrială la care am adus principalele contribuții a fost cea de detecție a obiectelor. Am explicat etapele de proiectare a aplicației precum și rezultatele obținute, atât din punct de vedere al calității acestora cât și al sistemelor comerciale existente acum pe piață.

O altă aplicație la care am adus contribuții din punct de vedere a implementării HW-SW este aceea de stabilizare a imaginilor. Am explicat cum blocurile propuse în capitolul 4 au ajutat la optimizarea soluției și la implementarea acesteia în sisteme comerciale de tip camere foto sau camere de acțiune.

O altă categorie de aplicații abordată a fost în domeniul auto – monitorizarea autovehiculelor.

Monitorizarea exterioară a autovehiculelor a fost concretizată printr-un sistem de tip HW-SW cu schimbarea câmpului vizual al camerelor video din exteriorul mașinii sub controlul algoritmilor de IA.

Monitorizarea interioară a autovehiculelor s-a concentrat pe monitorizarea șoferului prin integrarea mai multor algoritmi, pornind de la cel de detecție a obiectelor și până la monitorizarea direcției privirii. Am arătat modul în care PCNN poate fi folosit în astfel de sisteme.

Am arătat deci, prin intermediul celor trei clase de aplicații, cum modulele HW dezvoltate pentru accelerarea algoritmilor de PI pentru sistemele în TR, ajutate de IA sau care ajută IA, pot duce la optimizarea implementărilor și la soluții care sunt validate atât din punct de vedere al inovației (prin patente) dar și comercial (prin suita de produse industriale care le încorporează).

Capitolul 7. Concluzii generale, contribuții originale și dezvoltări viitoare

În cadrul acestui capitol sunt prezintate concluziile asupra rezultatelor cercetării doctorale, soluțiile propuse precum și validarea acestora prin intermediul articolelor publicate, al patentelor și propunerilor oficiale de patente, precum și al prezentărilor susținute în comunitățile de specialitate.

La nivel teoretic-conceptual, teza abordează aspecte legate modul în care se poate optimiza relația inteligența artificială-hardware în sistemele de procesare a imaginilor în timp real, abordând analitic toate componentele care compun o soluție constructivă: arhitectura HW și SW, algoritmii și datele. Aceste componente sunt văzute ca un *eco-sistem*, în care fiecare modificare a unei componente implică/necesită schimbări la celelalte componente.

Pentru aceasta, am analizat calitativ și cantitativ *criteriile de performanță* ale algoritmilor de IA care funcționează în TR. Am abordat aceste criterii de performanță în corelație, și am propus *un factor de merit* și o *procedură de calcul* acoperitoare pentru diverse soluții de implementare. Pentru a verifica criteriile de performanță ce contribuie la acest factor de merit, am considerat mai multe soluții funcționale, de la sistemele clasice de implementare (CPU/GPU) până la cele dedicate, foarte specifice unor tipuri de aplicații din domeniul PI.

- În cadrul acestei analize, am observat că, deși unele sisteme existente pot fi destul de complexe, cu putere de calcul superioară și resurse eterogene de implementare a algoritmilor, este complicat ca în sistemele existente să fie atinse toate cerințele de performanță. De aceea, pentru probleme specifice, am găsit soluții noi de implementare, cu factor de merit superior. Aceste soluții au fost apoi *generalizate pentru o gamă mai largă de probleme* cu cerințe asemănătoare. Deși majoritatea aplicațiilor au vizat probleme de analiză și implementare a algoritmilor de PI, soluțiile găsite se pot aplica unei *game largi de sisteme pentru procesarea de semnale*.

În capitolul al 3-lea, am abordat modul în care se poate realiza accelerarea HW pentru PI de TR. Am analizat relația dintre componentele soluțiilor de IA (HW, SW, date) și modul în care se poate maximiza factorul de merit al acestor soluții prin accelerare HW. Pe aceste principii, am decelat rezultatele cercetărilor doctorale pentru accelerarea algoritmilor de IA pentru sisteme de TR, grupând soluțiile propuse în două categorii: *"IA pentru HW"* și *"HW pentru IA"*.

În capitolul 4 am propus patru metodologii prin care IA poate contribui la proiectarea și implementarea unor sisteme HW mai eficiente, *exploatând specificul informației procesate de algoritmii IA*.

Am arătat deosebirea între configurare (la nivel constructiv) ca urmare a unei *decizii asistate* prealabile și *parametrizare* (instanțiere) consecutivă acesteia.

- Am introdus conceptul de *Integrare adaptivă a resurselor hardware*, prin care algoritmul IA poate decide folosirea nu în mod necesar a resurselor HW care generează rezultatele cele mai de calitate, ci uneori a resurselor HW care execută sarcinile cel mai rapid sau care consumă cea mai puțină energie.
- Un alt concept explicat și demonstrat a fost acela *de parametrizare dinamică anticipativă a HW de către IA*, în care controlul HW se face de către un algoritm IA, care parametrizează blocurile HW ținând cont de datele care sunt recepționate precum și prin anticiparea modului în care acestea se schimbă. IA parametrizează (instanțiază) blocurile HW folosite pentru a procesa optim fluxul de date.
- Am arătat cum se face *controlul preciziei implementării HW folosind IA pentru a spori toleranța la erori* – prin natura algoritmilor de IA, a modului lor de antrenare, există o toleranță la erori în datele de intrare (o formă a *robusteții* acestor algoritmi SW). Așadar precizia calculelor HW nu are sens să fie prea mare și se poate reconsidera o simplificare a implementării HW.
- Am propus un mecanism de *control prin IA al fluxului de date din HW*, în care fluxul de date principal care este procesat în HW este controlat de IA care analizează rezultatul fuziunii datelor secundare (de la alți senzori) cu meta-datele din fluxul principal.

Capitolul 5 cuprinde categoria "HW pentru IA" a soluțiilor prezentate în lucrare, axată pe tehnologii de accelerare HW a algoritmilor de IA. În această categorie am grupat soluțiile prin care module HW dedicate ajută algoritmii sub aspectul optimizării capacității de procesare a în calculele de IA, prin implementarea *masiv paralelă, eficientă energetic*. Am evidențiat avantajele obținute atunci când se execută în HW mii de operații în paralel pentru a crește viteza de execuție a algoritmilor IA, prin soluțiile propuse care permit un buget energetic extrem de redus.

O altă direcție de cercetare din categoria "HW pentru IA" a vizat *optimizarea folosirii magistralei de date*, banda de transfer fiind unul dintre criteriile de performanță critice pentru multe sisteme de TR. Am prezentat modul de implementare a unui sistem HW orientat pe utilizarea eficientă a magistralei de date.

Pentru a crește viteza de execuție a algoritmilor IA am prezentat o soluție inovatoare, prin care se pot realiza *dispozitive periferice cu IA*. Aceste dispozitive periferice pot recepționa datele de la sistemul *host* ("gazdă") și pot realiza, în același timp cu transferul I/O de date, și procesarea acestora cu algoritmi IA – *"pipeline IA".*

Toate aceste soluții originale au fost implementate în diferite aplicații și produse finite – de nivel ridicat TRL (Technology Readiness Level). În capitolul 6, dedicat nivelului aplicativ, am prezentat 3 studii de caz – exemple de validare a soluțiilor prezentate în domeniul de mare tehnicitate *"automotive"*.

- Am exemplificat modul în care detecția de obiecte poate fi implementată folosind tehnologiile de configurare dinamică anticipativă sau de segmentare predictivă a datelor.
- Am prezentat studiul de caz al stabilizării de imagini, în care am realizat fuziunea senzorilor precum și implementarea HW pentru optimizarea utilizării magistralei de date.
- Un alt exemplu de validare practică a soluțiilor propuse a fost acela al monitorizării interiorului unui autovehicul, folosind configurarea dinamică anticipativă, segmentare predictivă a datelor, fuziunea senzorilor și implementarea HW pentru optimizarea utilizării magistralei de date.

Pentru toate aceste aplicații a fost prezentat detaliat modul de punere în practică – fiecare implementare a fost evaluată, și cu scopul de a extrage idei de îmbunătățire a soluțiilor.

7.1 Contribuții originale

Această secțiune prezintă principalele rezultate ale cercetării doctorale, în ordinea capitolelor tezei.

În ceea ce privește soluțiile propuse și demonstratoarele dezvoltate pentru validarea lor, la o bună parte dintre acestea a fost marcat nivelul de disponibilitate tehnologică atins (TRL – *Technology Readiness Level).*

- Demonstratoarele au grupat mai multe soluții tehnologice în aplicații care să ilustreze practic avantajele oferite de noile capabilități. Majoritatea au fost la nivel de demonstrator tehnologic ("proof of concept") în mediul operațional – TRL6.
- *Aplicațiile moderne de prelucrare a imaginilor* s-au constituit într-un domeniu vast și divers de validare a ideilor studiate și a soluțiilor propuse în cadrul tezei de doctorat. O mare parte din soluțiile elaborate în acest domeniu au ajuns să depășească TRL9, fiind aplicate în produse comerciale de scară largă.

Contribuțiile au fost evidențiate cu simbolul **©**.

© Am particularizat echilibrarea Edge – Cloud a procesării pentru cazul sistemelor de TR care folosesc IA.

- Pentru soluțiile de PI în TR cu IA care funcționează în mediu distribuit, am analizat avantajele și dezavantajele procesării în Cloud comparativ cu cea "de la marginea rețelei" – Edge.
- Am introdus conceptul de corelare a prelucrărilor în mediu distribuit (Cloud / Edge) pentru *repartizarea dinamică* centralizat / localizat a a sarcinii computaționale

© Am introdus conceptul de ecosistem date-algoritmi-SW-HW pentru sistemele de TR.

- Am analizat relațiile dintre aceste patru categorii în cazul aplicațiilor de TR pentru PI cu IA. Am analizat interdependențele – între fiecare pereche de categorii precum și interacțiunile lor complexe, multiple, directe și indirecte – și am arătat cum fiecare este influențat pornind de la o schimbare inițială.
- Analiza acestor relații a fost apoi utilizată pe tot parcursul tezei, pentru a orienta proiectarea soluțiilor propuse (în special metodologia de co-design pentru IA al sistemelor prezentate în capitolul 6).

© Am identificat și am extins indicatorii de performanță utilizați pentru implementarea algoritmilor de PI cu IA în sisteme de TR.

În urma analizei soluțiilor bazate pe IA pentru domeniul PI în TR, precum și a interacțiunii cu producătorii de circuite integrate pe plan global și cu dezvoltatorii de aplicații cu IA pentru PI, am evidențiat cei mai importanți indicatori de performanță pentru arhitecturile computaționale folosite în acest domeniu.

- Deși, în literatura de specialitate, un număr de lucrări prezintă diferite analize ale acceleratoarelor HW pentru IA, majoritatea se limitează la analiza indicatorilor de performanță tipici pentru dezvoltarea de circuite integrate – categoria PPA *(performance-power-area)*.
- Abordarea originală a constat în includerea unor indicatori de performanță specifici domeniului PI în TR, evidențiind relevanța lor majoră în practica industrială și fundamentarea lor academic. Datorită limitărilor tehnologice sau a constrângerilor (cerințelor) de TR, anumiți indicatori de performanță (precum folosirea eficientă a memoriei) pot să fie critici – "metrici descalificante" – pentru validarea unora din soluțiile propuse.

Pe această bază, am urmărit să pun în evidență indicatorii de performanță pentru fiecare dintre soluțiile prezentate în lucrare, precum și balansarea/compromisul între diferiți indicatori de performanță atunci când a fost posibil.

© Am introdus "securitatea și protecția" precum și "riscurile pentru utilizare" ca indicatori de performanță determinanți pentru optimizarea soluțiilor de procesare cu IA în TR.

Aspectele de securitate în fața atacurilor sau de protecție a datelor sunt extrem de importante în contextul actual și trebuie luate în considerare în implementarea de soluții noi. Intensificarea atacurilor nu doar asupra sistemelor critice ci și asupra dispozitivelor mobile (fie pentru a beneficia de date confidențiale, pentru a distruge sau doar pentru a discredita) este o realitate pe care lucrarea de față o adresează în contextul în care sistemele de PI au acces la date cu caracter confidențial.

Mecanismele de protecție a datelor sunt importante atât datorită cerințelor legale (GDPR sau echivalent) dar și datorită sensibilizării utilizatorilor la aceste aspecte, în ultimii ani.

- În dezvoltarea soluțiilor am luat în considerare conceptul de *proiectare în vederea protecției ("privacy by design")* dar și de extensia acestuia prin *specificații în vederea protecției - ("privacy by definition")*.
- Am arătat cum securizarea datelor poate fi extinsă mai mult decât prin proiectare în vederea protecției, urmărind obținerea de soluții în care datele procesate nu mai prezintă riscuri deoarece nu mai conțin informații private.

O noutate constă și în concentrarea pe *protecția algoritmilor* și introducerea indicatorilor de performanță specifici în proiectarea soluțiilor de TR pentru PI.

- Algoritmii de IA au devenit tot mai complecși, iar efortul pentru dezvoltarea acestora, precum și pentru a produce datele necesare antrenării lor este semnificativ și în creștere. De aceea este necesar ca acești algoritmi să poată fi *protejați la expunerea lor exterioară.*

Riscurile pentru utilizare, care acoperă aspectele referitoare, pe de o parte la riscurile de *apariție a defectelor* iar pe de altă parte la riscurile privind *proprietatea intelectuală* sunt un alt indicator de performanță pe care l-am tratat în lucrare. Deși din punct de vedere academic, aceste aspecte ar putea să pară mai puțin interesante, din punct de vedere economic ele au un impact major și trebuie să fie luate în considerare.

© Am propus un factor de merit (și o metodă de calcul a acestuia) pentru aplicațiile de PI cu IA de TR.

Pe lângă ponderarea metricilor de performanță a aplicațiilor de TR, metoda de calcul al factorului de merit include 2 componente noi. Pe de o parte se definește factorul de merit descalificant, care are ca scop descalificarea (cu reducerea la zero a factorului de merit) în cazul în care unele criterii de performanță depășesc limitele admisibile pentru soluția avută în vedere (cum ar fi depășirea consumului de energie avut la dispoziție de un dispozitiv mobil).

Pe de altă parte, am introdus *factorul de merit critic,* care definește pragurile la care meritul soluției este semnificativ schimbat. Aceste praguri sunt definite de cerințele soluției precum și de limitările tehnologice (cum ar fi numărul de circuite de memorie externă necesare). Am demonstrat că se pot lua decizii justificate cantitativ pentru alegerea diferitelor soluții / arhitecturi de implementare.

© Am dezvoltat o arhitectură pentru soluțiile hibride HW-SW de PI în TR.

Această arhitectură permite rezolvarea unor probleme critice ale sistemelor de TR: "locul îngust" ("bottleneck") de la accesarea memoriei – problema benzii de transfer – precum și problema vitezei de procesare. Deși arhitectura a fost dezvoltată inițial pentru PI în TR, am prezentat modul în care aceasta poate fi aplicată pentru alte domenii ale procesării de semnale în TR. Aceste principii au fost apoi folosite pentru noile soluții tehnologice prezentate în cadrul tezei, bazate pe (re-)alocarea dinamică a resurselor HW-SW ale sistemelor hibride funcție de schimbarea la nivelul datelor și al cerințelor de prelucrare a acestora, cu scopul de a maximiza raportul dintre performanța computațională și consumul energetic.

© Am identificat direcțiile prin care IA are potențialul de a crește performanța sistemelor hibride HW-SW pentru PI de TR.

În **categoria IA pentru HW** (capitolul 4), au fost identificate 4 direcții de cercetare în care IA poate controla componentele HW din cadrul sistemelor hibride. Acestea sunt: integrarea adaptivă a resurselor HW (decizie asistată de IA asupra configurației sistemului HW), parametrizarea HW de către IA (configurarea adaptivă la nivel de modul HW), controlul preciziei calculelor implementate în HW (toleranța la erori sporită de IA), controlul prin IA al fluxului de date din HW (bazat pe fuziunea datelor secundare cu meta-date din fluxul principal).

© Am introdus conceptul de integrare adaptivă a resurselor HW.

În relația dintre IA și HW, am arătat cum se iau deciziile asistate de IA asupra configurației sistemului HW. Pentru aceasta am definit și implementat un sub-sistem de accelerare HW pentru PI, AHIP (Advanced Hardware for Image Processing) care ilustrează conceptul de integrare adaptivă a resurselor

- Acest sub-sistem funcționează ca o colecție de module HW care implementează primitive pentru procesarea de imagini. Controlul sub-sistemului se face de către algoritmii de IA care decid ce blocuri HW sunt folosite și în ce mod. IA decide ca anumite blocuri să fie selectate (efectiv utilizate) sau nu (trecute în starea de *rezervă* activă și *ocolite* de fluxul de date). Așadar, în arhitectura HW se pot include blocuri de procesare paralele suplimentare, care sunt puse în funcțiune doar când algoritmul de IA decide acest lucru.
- Blocurile de calcul de bază se interconectează pentru a crea *hărți de caracteristici* – primivite de procesare de nivel mai înalt.
- Pentru optimizarea accesului la memorie, se pot cupla mai multe blocuri de calcul al primitivelor care *să citească datele din memorie o singură dată.*
- Funcționarea într-un sistem de matrice *sistolică* – tipică în PI – presupune faptul că fiecare modul HW are o latență foarte mică, iar *modul de lucru înlănțuit permite calculul foarte rapid* al primitivelor. Ca rezultat, primitivele de imagine sunt disponibile la scurt timp (câteva tacte de ceas după recepționarea imaginii). Ca urmare *se poate lucra în pipeline* – în timpul unui cadru se calculează primitivele de imagine ale acestuia iar concomitent se rulează algoritmul de PI de nivel înalt pe cadrul anterior. În plus, AHIP permite *combinarea de primitive din cadre succesive* pentru a crea primitive de nivel superior, mai eficiente din punct de vedere al dimensiunilor și modului de utilizare.

O contribuție inovatoare a fost și aceea prin care procesarea de TR este *anticipativă:* algoritmul IA poate anticipa detalii ale conținutului cadrului video actual pe baza informațiilor analizate din cadrele anterioare, și în acest fel se pot identifica blocurile HW necesare pentru implementarea optimă a funcțiilor de PI.

- Lanțul de procesare este astfel *re-configurat dinamic, anticipativ.*
- De asemenea, fiecare bloc HW poate fi *parametrizat* de către algoritmul de control.
- Deoarece deciziile anticipative ale IA pot fi imprecise, AHIP permite definirea de dimensiuni în HW ale regiunilor de interes (ROI) aproximative, care apoi sunt rafinate în SW. ROI pot fi definite de un modul de analiză a imaginilor.

Datele rezultate după calculul HW se pot trimite direct către memoria sistemului sau pot fi *compresate adaptiv* – întrucât compresia poate avea ca intrare informațiile furnizate de IA, cum ar fi poziția ROI.

Integrarea adaptivă a resurselor HW folosind AHIP a fost demonstrată în cadrul aplicației de detecție a obiectelor.

- Am participat, cu aceste soluții originale, la implementarea pe FPGA a unor serii succesive de demonstratoare pentru detectoare de obiecte. Aceste sisteme funcționează în TR, chiar *la frecvențele mai mici* suportate de dispozitivele FPGA, ceea ce ilustrează avantajele tehnologice deosebite ale soluțiilor.
- Am contribuit la promovarea și integrarea AHIP pentru implementarea algoritmilor de detecție în SoC (Systems on Chip) comerciale pentru telefoane mobile, camere foto digitale sau camere "de acțiune" (sport, intervenții de salvare etc). Dintre telefoanele mobile care implementează tehnologia de detecție și corecție ale fețelor se pot aminti dispozitivele Huawei P8 si Honor P7 [162]. Dintre camerele foto digitale, această tehnologie este prezentă în Nikon Coolpix [163] și Nikon Z Series [164].

  Alte dezvoltări comerciale ale acestor soluții de detecție a obiectelor pot fi găsite în dispozitive LG, Oppo, Socionext sau ZTE [165] iar sursele publice de prezentare a acestor soluții includ canale YouTube [166] și similare [167].

Din punct de vedere al disponibilității tehnologice, am depășit TRL9, prin aducerea sistemului AHIP la nivel comercial.

Diseminarea acestor contribuții a fost făcută prin grupul de patente indexate Derwent [15-CZ-P], care cuprinde 19 documente publicate în Statele Unite, China, Corea de Sud, Japonia și la nivel global.

© Am conceput o metodă nouă de scalare adaptivă pentru sistemele de achiziție video cu stocare în Cloud.

Pe baza principiului de integrare adaptivă a resurselor HW pentru PI controlată de IA, am conceput un sistem în care scalarea imaginilor achiziționate de un sistem video se face folosind module HW *selectate* de către IA. Algoritmul *detectează evenimentele* importante din fluxul video (apariția de obiecte de interes) și decide ce metodă de scalare este selectată dintre cele disponibile în implementarea HW.

Prin această metodă se poate alege o metodă de scalare de *calitate mai scăzută* atunci când nu sunt detectate obiecte de interes sau *mai ridicată* atunci când obiectele de interes sunt prezente în fluxul video.

Am analizat modul de implementare în HW a fiecărei metode de scalare și am evidențiat costul (dat de aria de siliciu folosită) pentru fiecare caz. Analiza a luat în considerare consumul de energie în fiecare scenariu, transferul de date, precum și calitatea imaginilor rezultate și a algoritmului de detecție de obiecte.

Diseminarea acestor contribuții s-a făcut prin articolul [7-CZ-A].

© Am propus un concept nou de parametrizare HW de către IA – care completează configurarea dinamică anticipativă

Acest concept presupune configurarea modulelor HW prin IA, ca urmare a analizei datelor de la intrarea sistemului și anticipării modului de evoluție a acestora. Astfel, IA poate anticipa care vor fi caracteristicile datelor de intrare și ca urmare poate parametriza (instanția) modulele HW de procesare în mod dinamic.

Conceptul a fost pus în practică prin dezvoltarea unei soluții de detecție a obiectelor de tip TMM (Template Matching Module) - segmentare pe porțiuni asemănătoare cu un model ("șablon" – "template").

Într-o fază preliminară (de "acumulare a experienței") s-a creat un set de șabloane exprimate ca modele matematice (o pre-clasificare comprehensivă).

În funcție de datele care sosesc, în particular caracteristicile obiectului detectat (tipul, orientarea, poziția, expunerea, condițiile de iluminare etc), selectăm șabloanele adecvate ("matching") folosite ca bază anticipativă și în final coeficienții acestor formule ale modelului matematic sunt rafinați (de către IA) în timpul fazei de parametrizare.

TMM poate funcționa cu o serie de componente HW dedicate (cu funcție fixă) sau programate (de către algoritmul de IA) care lucrează în paralel pentru a găsi potrivirea cu un număr mare de șabloane pre-definite.

Pe baza analizei anticipative prin IA, blocurile HW pot fi programate *inițial cu clasificatori simpli*, care apoi se pot extinde cu clasificatori mai complecși. Aceștia se folosesc când se detectează date care au potențialul de a se încadra în șabloane și atunci o analiză mai detaliată a acestor date este necesară.

- Configurarea dinamică anticipativă este aplicată și în cazul în care se modifică anticipativ clasificatorii folosiți pentru detecție prin modificarea parametrilor blocurilor HW – analiza *cadrelor* anterioare din fluxul video contribuind la selecția clasificatorilor ce vor fi folosiți în cadrele următoare.
- De asemenea, se folosește configurarea dinamică anticipativă prin configurarea blocurilor HW în funcție de caracteristicile locale ale *regiunii* de analizat. IA prelucrează local datele din regiunea de analizat și schimbă dinamic configurarea blocurilor HW pe baza acestei analize.

Am propus implementarea unui sistem de pre-filtrare a datelor, prin folosirea unui bloc HW care *elimină* rapid regiunile care nu prezintă interes pentru detecția de obiecte datorită faptului că nu îndeplinesc cerințele de bază pentru potrivirea cu șabloanele pre-definite.

Conceptul TMM de configurare dinamică anticipativă a fost de asemenea demonstrat în cadrul aplicației de detecție a obiectelor prezentată în capitolul 6 (dedicat aplicațiilor avansate). Pentru aceasta au fost dezvoltate o serie de demonstratoare FPGA iar aplicația comercială a fost implementată pentru telefoane mobile și camere foto digitale.

Diseminarea contribuțiilor a fost făcută prin grupul de patente indexate Derwent [9-CZ-P] și [10-CZ-P].

Extinderea conceptului de TMM cu pre-filtre HW a fost publicată prin grupul de patente indexate Derwent [11-CZ-P], [12-CZ-P] și [13-CZ-P].

© Am propus o soluție de control al preciziei implementării HW folosind IA pentru a spori toleranța la erori.

Aceast concept, prin care *se scade în mod intenționat precizia de calcul a implementării HW* (reducând spectaculos costurile), a fost exemplificat pentru un algoritm de detecție a obiectelor de tip HOG+SVM (Histogram of Oriented Gradients + Support Vector Machines). Am minimizat implementarea HW a Histogramelor Orientării Gradienților arătând că algoritmul IA de clasificare tip SVM *poate compensa pierderea de precizie a calculelor HW,* făcând posibilă astfel reducerea complexității constructive.

Partea inovativă a fost dată de modalitatea simplă de implementare în HW a unui număr mare de containere mici de orientări pentru gradienți care sunt asimetrice (ca urmare a simplificării calculelor geometrice și, implicit, a reducerii complexității circuitelor de calcul).

Calculul magnitudinii gradienților a fost de asemenea optimizat, prin aproximarea acesteia în funcție de containerul din care gradientul face parte (pe baza unui LUT – Look Up Table).

Implementarea efectivă în HW a fost realizată în două moduri diferite, cu *optimizarea căilor critice* sau cu *reducerea ariei de siliciu.*

Am prezentat rezultatele cercetării în articolul [1-CZ-A].

De asemenea, am brevetat aceste contribuții în cadrul grupului de patente indexate Derwent [14-CZ-P].

Soluția a fost validată prin integrarea în camere comerciale DSLR (Digital Single Lens Reflex) unde au fost folosite pentru detecția de persoane (mai ales în publicul evenimentelor sportive).

© Am propus și am demonstrat o soluție (tip "IA pentru HW") de control prin IA al fluxului de date din HW.

Fluxul principal de date este procesat de sistemul HW de TR (în cazul stabilizării de imagine este vorba de fluxul video). Fluxul secundar de date – meta-date din fluxul video și ieșiri de la alți senzori secundari – obținut prin "fuziunea senzorilor", este analizat de IA pentru a gestiona fluxul de date principal.

Aceste principii au fost ilustrate în cadrul unui modul HW de corecție a distorsiunilor din fluxul video.

- Algoritmul de IA poate detecta unde în imagine apar obiecte de interes și poate defini parametri de corecție specifici pentru aceste regiuni. Modulul HW poate corecta distorsiuni date de ansamblul lentilelor, de mișcările relative ale senzorului precum și cele generate de efectul de obturator derulant ("rolling shutter"). Toate aceste tipuri de distorsiuni sunt cuantificate de IA ce programează modulul HW.
- Am definit o arhitectură HW pentru procesarea fluxului video de înaltă rezoluție în TR în vederea corecției distorsiunilor geometrice - GDE (Geometrical Distorsion Engine). Modul de funcționare a GDE a fost explicat, pe baza grilelor de corecție și a procesării "dalelor" ("tiles") corespunzătoare acestei grile.

Implementarea HW a avut în vedere modul optim de folosire a magistralelor de date, prin folosirea eficientă a *citirii și scrierii în rafală* ("burst").

Pentru prima dată a fost definit un modul HW care combină toate distorsiunile detectate de IA (distorsiuni globale, locale, de mișcare și cauzate de obturatorul derulant) într-una singură care este apoi folosită pentru corecția fluxului video. Structura internă a acestui modul (GFU – Grid Formatter Unit) a fost prezentată în detaliu.

Blocul de corecție a distorsiunilor pentru fiecare dală (GDC – Geometrical Distorsion Core) a fost de asemenea descris. Am pus accentul pe modul eficient de folosire a memoriilor locale pentru optimizarea benzii de transfer la memorie și pe maximizarea calității algoritmului de re-eșantionare.

Pentru validarea acestor contribuții, modulul GDE a fost implementat inițial pe FPGA și demonstrat în cadrul aplicației de stabilizare a imaginilor (așa cum am arătat în capitolul 6). Pentru aceasta am construit un sistem de testare care face ca întreg ansamblul FPGA – senzor – sistem optic să vibreze pentru a simula deplasările camerei în mișcare.

Aplicațiile EIS (Electronic-Image-Stabilization) implementate pe baza soluțiilor propuse sunt cuprinse în dispozitive Socionext [170]. Am propus un *Chipset fără GPU* care poate procesa video HD și 4K până la 60 de cadre pe secundă, permițând obținerea de videoclipuri care sunt practic neafectate de mișcarea și vibrațiile platformei, corectând în același timp distorsiunile cauzate de lentilele cu unghi larg întâlnite în mod obișnuit în drone și camere de acțiune [171], [172].

Diseminarea rezultatelor cercetării s-a făcut prin brevetare – în grupurile de patente indexate Derwent [17-CZ-P], [18-CZ-P] și [19-CZ-P]; în același sens, pentru a evidenția validarea contribuțiilor, am făcut și trimiterile anterioare [170], [171], [172].

În **categoria HW pentru IA** (capitolul 5), au fost identificate 2 direcții în care se poate optimiza implementarea HW pentru execuția eficientă a algoritmilor de IA: optimizarea capacității de procesare și respectiv utilizarea eficientă a magistralelor de date.

© Am cercetat modul (tip "HW pentru IA") în care se poate optimiza capacitatea de procesare a algoritmilor IA ce beneficiază de accelerare HW.

Am studiat implementarea în HW a algoritmilor de IA pentru a identifica primitivele de calcul cele mai folosite pentru rețelele neurale, și am analizat diferențele de implementare a acestora pe resurse HW generice (precum CPU, GPU) sau pe HW dedicat.

Am propus un sistem HW *programabil* pentru implementarea rețelelor neurale convoluționale (PCNN – Programmable Convolutional Neural Network). Acesta este un sistem configurabil în care blocuri de pixeli de diferite dimensiuni pot fi citite simultan dintr-o memorie cache a imaginii, în vederea prelucrării pentru a produce date de ieșire.

- PCNN poate fi ușor încorporat într-un sistem de PI; acesta poate fi configurat – poate fi programat pentru a funcționa cu diferite tipuri de straturi convoluționale și straturi de clasificare. PCNN include o memorie cache de imagini cu o arhitectură "pipeline" capabilă să furnizeze rapid informații (harta 2D a datelor de intrare) unui "motor" de convoluție, astfel încât o convoluție 3D care implică un anumit număr de hărți de intrare să poată fi efectuată într-un număr minim de cicluri de ceas.

Am prezentat în detaliu arhitectura internă a PCNN și modul de funcționare a acestuia. Procesarea eficientă în TR a rețelelor neurale presupune un număr mare de operații executate în paralel. Este importantă metoda de folosire a memoriilor interne ale PCNN.

- Am prezentat o metodă inovatoare pentru modul de alocare a hărților convoluționale în memoria cache.
- De asemenea, am prezentat modul în care se poate face implementarea HW eficientă a memoriei cache a sistemului PCNN.

Diseminarea contribuțiilor a fost făcută prin grupul de patente indexate Derwent [22-CZ-P], [23-CZ-P] și [24-CZ-P].

Blocul PCNN a fost validat printr-un demonstrator pe FPGA pentru aplicații de tip detecție de persoane. Unul dintre algoritmii de IA care a rulat pe PCNN a fost prezentat în articolul [3-CZ-A]. Implementarea pe ASIC a fost făcută pentru accelerarea rețelelor neurale care funcționează în TR pe camere foto digitale.

© Am extins procesarea rețelelor neurale de la un singur modul PCNN la arhitecturi multi-procesor.

Am explicat cum PCNN este utilizabil ca un *procesor neural de uz general,* capabil "să încarce rețelele" (de exemplu, echivalentul încărcării programului într-un procesor tradițional) printr-un canal de memorie separat de datele imaginii de intrare și de rezultate. Procesul este iterativ, rețelele neurale sunt rulate strat cu strat. Configurația straturilor – în mod specific convoluțională, complet conectată, pentru grupare sau pentru de-grupare a datelor – precum și tipurile și valorile ponderilor și funcțiile de activare ale neuronului sunt toate programabile în PCNN prin definiția rețelei.

Am explicat modul de funcționare a PCNN într-un *periferic de procesare.* Un astfel de periferic de procesare poate fi implementat astfel încât să apară ("să se vadă") pentru un sistem gazdă obișnuit (susținut de aproape toate sistemele de operare) ca un periferic de stocare accesibil printr-o interfață fizică standard de mare viteză. Aplicațiile pot fi distribuite pe spațiul de stocare furnizat de perifericul propriu-zis, încărcat după cum este necesar, și apoi executat la cerere de către dispozitivul gazdă.

- Am detaliat arhitectura perifericului de procesare care conține PCNN, *mapat în spațiul de adresare a memoriei.*
- Am explicat modul în care sistemul de fișiere al perifericului poate fi folosit pentru rularea rețelelor neurale.
- Am extins modul de folosire a PCNN prin construirea unui *cluster PCNN.* Acest sistem multi-procesor neural conține un număr de PCNN controlate de un procesor de uz general (RISC) și folosește o memorie comună RAM.
- Am definit modul în care memoria comună poate fi folosită în modul multi-procesor.
- Am explorat modul în care implementarea unui sistem multi-procesor de tip cluster PCNN poate fi făcută cu *chip-uri suprapuse,* folosind metode de suprapunere 3D clasice sau cu un număr mare de conexiuni între chip-uri (de tip ZiBond sau DBI).
- Am explicat modul în care clusterele PCNN pot fi înlănțuite pentru rularea rețelelor neurale complexe.
- Am detaliat modul în care memoria comună poate fi implementată cu memorii de tip DRAM (Dynamic Random Access Memory) însă cu *eliminarea circuitului de reîmprospătare* ("refresh"). Această opțiune este folositoare în cazul PCNN deoarece durata de stocare a informațiilor temporare în memoria comună este pre-definită și foarte mică (sub durata pe care ar fi impus-o reîmprospătarea).

Am exemplificat cum PCNN poate fi folosit în aplicații complexe, cum sunt cele din domeniul auto, precum și în sisteme cu mai multe surse de date (”sensor fusion”).

Pentru aplicațiile de PI în TR, am exemplificat modul de folosire a PCNN cu sisteme specifice de PI (precum cele descrise în capitolul 4 – pentru calculul de primitive de imagine, detecție de obiecte sau corecție de distorsiuni).

Am arătat modul în care clusterele PCNN pot comunica între ele printr-o rețea de comunicație.

De asemenea, am explicat modul în care procesoarele RISC din clusterele PCNN pot fi folosite ca CPU de uz general sau cum memoria locală a clusterelor PCNN poate fi folosită de CPU gazdă ca memorie de lucru.

Am analizat modul în care se poate face re-configurarea sistemelor cu clustere PCNN în cazul defectării (în procesul de fabricație) a unora dintre procesoarele neurale (”self healing”, auto-vindecare).

Un aspect nou îl prezintă și modul în care se poate face criptarea informațiilor pentru definiția rețelelor. Acest mecanism permite *protecția împotriva copierii structurii* rețelelor neurale, factor important pentru păstrarea proprietății intelectuale asupra implementării IA.

Diseminarea contribuțiilor a fost făcută prin grupul de patente indexate Derwent [25-CZ-P].

Sistemul multi-procesor a fost demonstrat parțial pe FPGA. Datorită dimensiunii mari a componentelor HW, am optat pentru a implementa doar sub-sisteme multi-procesor PCNN. În capitolul 6 am explicat în detaliu modul de demonstrare practică a PCNN pentru monitorizarea interiorului autovehiculelor. Au fost demonstrați algoritmi de IA cu rețele neurale convoluționale – printre care cei de analiză facială sau de urmărire a capului și a privirii șoferului..

© Am introdus redundanța în sistemele multi-procesor pentru rețele neurale.

Conceptul presupune că un prim ”motor” de procesare a rețelei (PCNN) este configurat (în rest, celelalte fiind inactive la început), fiind extrase informațiile de configurare și informațiile de imagine de intrare care urmează să fie furnizate apoi unui alt ”motor” de procesare (”țintă”). PCNN este programat pentru a compara o parte din informațiile obținute de motorul ”țintă” cu informațiile corespunzătoare generate de primul motor de procesare pentru a verifica dacă ambele funcționează corect.

Am explicat în detaliu modul de funcționare a PCNN multiple independent sau redundant, cu compararea rezultatelor de ieșire.

Am prezentat scenariile de decizie (detecție/corecție de eroare sau oprire a funcționării) atunci când analiza rezultatelor în cazul funcționării redundante conduce la ipoteza unui PCNN defect.

Am detaliat modul de *testare oportunistă* a PCNN, folosind capacitatea de calcul *de rezervă* – în timpul în care un PCNN nu este folosit pentru funcții critice.

Diseminarea contribuțiilor a fost făcută prin grupul de patente indexate Derwent [16-CZ-P].

© Am conceput o metodă de proiectare HW orientată pe utilizarea eficientă a magistralei de date

Am propus și dezvoltat un mecanism de procesare HW trecând de la criteriul vitezei de procesare la criteriul optimizării benzii de transfer.

- Astfel, am construit un sistem care, deși este mai complex computațional, rezolvă cu eficiență maximă problema transferului de date dintre modulul HW și memoria centrală. Exemplificarea acestei soluții de utilizare eficientă a benzii de transfer s-a făcut în contextul algoritmilor de rotire și scalare a imaginilor.
- Am structurat intern un bloc de normalizare a imaginilor (NORM) – am arătat modul de împărțire a regiunilor de interes pe ”dale” (”tiles”), și ordinea de prelucrare a acestor dale în funcție de unghiul de rotire a regiunii, pentru *minimizarea traficului de date pe magistrală.*
- Am reglementat modul de citire a datelor din memoria principală, pentru utilizarea eficientă a magistralei de date. Am analizat impactul acestui nou mod de citire asupra fluxului de date din cadrul blocului NORM.

- Am prezentat modul de organizare a memoriei interne a NORM, organizare ce contribuie de asemenea la optimizarea traficului de date pe magistrală.

O idee nouă a fost aceea de a efectua două operații geometrice diferite (rotația și scalarea) într-un singur pas HW. Astfel, parametrii de re-eșantionare se calculează pentru ambele operații, se combină și se aplică *o singură dată* sub-sistemului HW aferent. În acest fel am redus și mai mult traficul pe magistrala de date.

Optimizări suplimentare (pe care le-am ilustrat cu o serie de transformări utile algoritmilor de analiză facială) sunt aplicabile dacă se efectuează operații suplimentare de PI după ce imaginea este scalată și rotită.

Validarea și diseminarea contribuțiilor a fost făcută prin grupul de patente indexate Derwent [20-CZ-P] și [21-CZ-P].

© În vederea unui co-design pentru IA, am redefinit etapele implementării aplicațiilor pentru PI de TR.

Am urmărit păstrarea la fiecare nivel și în orice etapă de proiectare a posibilității de alegere între HW sau SW și ridicarea până la nivelul algoritmului generic a acestei dualități.

Pe baza experienței în dezvoltarea de soluții comerciale, am punctat modul în care procesul de implementare a aplicațiilor se modifică atunci când se urmărește optimizarea la nivel de *eco-sistem date-IA-HW-SW.*

Am explicat toate etapele implementării aplicațiilor de PI în TR, cu accent pe implementarea HW și pe relația dintre componentele algoritmice și HW.

- În acest context am marcat nivelul de maturitate tehnologică (TRL) ale aplicațiilor implementate.

© La nivel aplicativ, am dezvoltat o nouă metodă de monitorizare a exteriorului autovehiculelor prin folosirea IA pentru controlul HW.

Am definit un nou mecanism de reorientare a oglinzilor retrovizoare exterioare, care să permită schimbarea câmpului vizual al camerelor video montate pe acestea.

Am explicat cum poate funcționa sistemul HW în corelație cu algoritmii de IA (precum cei de detecție de obiecte sau cei de recunoaștere a persoanelor). În felul acesta am evidențiat modul în care aplicațiile de monitorizare a exteriorului autovehiculului pot funcționa optimal.
Am detaliat etapele necesare pentru a implementa sistemul hibrid de monitorizare, prin relaționarea îmbunătățită a IA cu HW.

- În acest context, am propus folosirea camerelor exterioare montate pe oglinzile laterale și pentru protecția împotriva accesului neautorizat în mașină.

Diseminarea contribuțiilor a fost făcută prin grupul de patente indexate Derwent [26-CZ-P] și [27-CZ-P].

7.2 Dezvoltări viitoare

În urma cercetărilor făcute în contextul acestei teze de doctorat am identificat mai multe direcții viitoare de dezvoltare:

- Extinderea conceptului de *ecosistem date-IA-SW-HW* dincolo de PI. Am observat că algoritmii audio devin deja suficient de complicați pentru a justifica și ei această abordare – implementările de algoritmi cu rețele neurale pentru procesarea audio (cum ar fi separarea surselor audio sau detecția de evenimente audio) fiind din ce în ce mai prezente în literatură și în industrie. Accelerarea HW a acestor algoritmi va devini deci un subiect important în viitorul apropiat.
- Utilizarea extinsă a fuziunii senzorilor poate aduce avantaje suplimentare algoritmilor clasici de PI. Pentru aceasta, o integrare mai detaliată a diferitelor surse de semnal pentru a rezolva probleme specifice poate fi foarte folositoare. Direcțiile principale de cercetare pentru fuziunea senzorilor pot duce la integrarea cu senzorii de imagine a unor senzori de tipul lidar, radar, termici, infraroșu sau senzori de detecție a evenimentelor. Prin fuziunea acestor senzori se pot concepe arhitecturi HW inovative, cu performanțe net superioare celor actuale.
- Folosirea mai eficientă a arhitecturilor de circuite integrate *suprapuse* este o direcție de cercetare foarte promițătoare care poate aduce îmbunătățiri semnificative indicatorilor de performanță critici pentru sistemele de TR. În mod

specific, modalitățile de interconectare 3D între circuitele integrate pot crește și mai mult viteza de comunicare între "logică" și memorie și pot ridica la un nivel superior viteza de calcul în HW pentru IA.

- Arhitecturile de rețele neurale evoluează rapid la ora actuală. De aceea, accelerarea HW doar a straturilor convoluționale ar putea să nu mai fie de ajuns într-un viitor apropiat. Pentru aceasta, concentrarea pe noile structuri de rețele neurale, precum rețelele neurale recurente (RNN – Recurrent Neural Networks), rețele cu straturi "de atenționare" ("attention layers") sau "transformaționale" ("transformers") trebuie să-și găsească modalitatea optimă de accelerare HW.
- Analiza rarității (ponderilor nenule) – "sparsity" – în rețelele neurale, precum și cuantizarea ponderilor și a datelor din straturilor intermediare este abordată în prezent pentru reducerea puterii computaționale și a energiei consumate. Deși soluțiile (de tip PCNN) prezentate în lucrare iau în considerare aceste aspecte, o analiză mai detaliată a arhitecturilor HW care să valorifice avantajele acestor metode poate fi încununată de noi succese.
- Arhitecturile deschise sunt din ce în ce mai populare. Printe acestea, arhitectura RISC-V reprezintă una dintre cele mai promițătoare direcții de cercetare. Prin extinderea setului de instrucțiuni cu instrucțiuni specifice accelerării algoritmilor de IA, s-ar putea ajunge la implementări HW atât eficiente din punct de vedere funcțional cât și ușor de integrat HW-SW
- Folosirea acceleratoarelor HW este foarte eficientă din punct de vedere al indicatorilor de performanță specifici aplicațiilor de implementat, însă există în continuare o reținere a celor mai mulți ingineri de a dezvolta soluții hibride HW-SW cauzată de mediile de programare / dezvoltare / testare existente. Pentru a rezolva această problemă, o analiză detaliată a mediilor de dezvoltare care să permită codesign HW-SW dar și codesign HW-SW-IA este utilă, iar găsirea unor medii de dezvoltare mai prietenoase cu utilizatorii este binevenită. În aceastp direcție, este necesar să fie mai bine investigate soluții de tip HLS *(High Level Synthesis* – Sinteză de Nivel Înalt) sau medii de programare bazate pe limbaje de scripting (de exemplu Python).
- Migrarea dinspre Cloud spre Edge și invers a fost discutată sumar în cadrul tezei pentru sistemele de PI în TR. Pe măsură ce capabilitățile Cloud cresc (pe de o parte) iar sistemele Edge sunt și ele mai capabile (pe de-altă parte), balansul dintre Cloud și Edge este în continuă schimbare. În unele aplicații specifice, studiul mai amănunțit al limitei dintre Edge și Cloud poate aduce noi beneficii sistemelor hibride prezentate în teză.
- În domeniul *automotive,* implementările HW ale blocurilor detaliate pe parcursul tezei trebuie să îndeplinească cerințe speciale, atât din punct de vedere al procesului de cercetare-dezvoltare cât și din punct de vedere structural și al securității. Adaptarea modulelor HW prezentate pentru cerințele standardelor din industria auto reprezintă o oportunitate pentru viitorul apropiat, mai ales în perspectiva în care apar tot mai des aplicații ce folosesc intensiv IA în autovehicule.
- Deși majoritatea propunerilor de arhitecturi și soluții inovatoare au fost patentate, acest proces trebuie să continue pentru acoperirea întregului spectru de contribuții noi, prin interacțiunea cu birourile de patentare (atât locale cât și globale). Pentru aceasta, voi continua procesul care este deja pornit pentru aprobarea revendicărilor propuse spre patentare precum și formalizarea noilor contribuții în propuneri de patentare.

Referințe

Publicații proprii

Open Researcher and Contributor IDentifier (ORCID) 0000-0002-8170-7176
https://orcid.org/0000-0002-8170-7176

Articole indexate ISI

– International Science Index – Web of Science - Core Collection, Clarivate Analytics
(aceste articole sunt indexate și Scopus și IEEE Xplore)

[1-CZ-A] **Zaharia, C.**, Sandu, F., & Balan, A. (2020, December). Usage of Asymetric Small Binning to Compute Histogram of Oriented Gradients for Edge Computing Image Sensors. In *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)* (pp. 1-6). IEEE.

[2-CZ-A] Machidon, O., Sandu, F**., Zaharia, C.**, Cotfas, P., & Cotfas, D. (2014, May). Remote SoC/FPGA platform configuration for cloud applications. In *2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)* (pp. 827-832). IEEE. [17 citări, din care 7 în publicații ISI]

[3-CZ-A] **Zaharia, C.**, Dinu, D., & Caliman, A. (2017, May). PVANet optimization for person detection. In *2017 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM) & 2017 Intl Aegean Conference on Electrical Machines and Power Electronics (ACEMP)* (pp. 959-964). IEEE. [1 citare]

[4-CZ-A] **Zaharia, C.**, Bigioi, P., & Corcoran, P. M. (2011, January). Hybrid video-frame pre-processing architecture for HD-video. In *2011 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 89-90). IEEE. [9 citări, din care 4 în publicații ISI]

[5-CZ-A] Bigioi, P., **Zaharia, C.**, & Corcoran, P. (2012, January). Advanced hardware real time face detector. In *2012 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 528-529). IEEE. [indexată și IEEE Xplore ; 15 citări, din care 7 în publicații ISI]

[6-CZ-A] Nica, A., Balan, A., **Zaharia, C.**, & Balan, T. (2021, February). Automated Testing of GUI Based Communication Elements. In *International Conference on Remote Engineering and Virtual Instrumentation* (pp. 380-390). Springer, Cham.

Articol indexat SCOPUS – în curs de indexare ISI

[7-CZ-A] **Zaharia, C.**, Sandu, F., & Danciu, G. (2021, November). Adaptive Scaling for Image Sensors in Embedded Security Applications. In *2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet)* -. IEEE.

Alte publicații în domeniu

[8-CZ-A] Nicula, D., Ionita, M., **Zaharia, C.**, Tulbure, T., Jipa, R., Denes, N. - Limbaje de descriere hardware - Aplicații - Universitatea" Transilvania" din Brașov, (2002)

Patente indexate în Clarivate Analytics Web of Science - Derwent Innovations Index

Conceptele inovatoare au fost propuse pentru patentare în mai multe țări (cu indicativele US, CN, KR, JP, TW, etc), grupuri de țări (indicativ EP pentru European Patent Office) sau pe plan mondial (indicativ WO pentru World Patent Office). Au fost evidențiate 25 de grupe de patente – pentru fiecare indexare, pe lângă prefixele ce corespund cu indicativele menționate anterior, sufixele A și B corespund, respectiv, cererii de brevetare și patentului acordat. Citările sunt extrase din baza de date Google Academic. Cele 6 cazuri de continuări de patente sunt marcate în paranteze drepte și sunt ordonate cronologic.

[9-CZ-P] Sultana, B., Petrescu, S., Nicolau, R., Ursachi, V. I., Bigioi, P., **Zaharia, C.**, Corcoran P., Fulop, S. & Gangea, M. – *„Face Or Other Object Detection Including Template Matching" [Hardware]* - 2012 - DIIDW:2012F08407 [52 citări]
Brevete: US2012106790-A1; US8995715-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2012F08407

[10-CZ-P] Sultana, B., Petrescu, S., Nicolau, R., Ursachi, V. I., Bigioi, P., **Zaharia, C.**, Corcoran P., Fulop, S. & Gangea, M. – „*Face Or Other Object Detection Including Template Matching*" *[Multiple data processing]* – 2015 - DIIDW:201541872D [7 citări]
Brevete: US2015206030-A1; US9639775-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201541872D

[11-CZ-P] Nicoara, N., Raceala, C., **Zaharia, C.**, Fulop, S., & Iovita, O. - *"Image processing system" [Prefiltering]* - 2017 - DIIDW:2017433617 [4 citări]
Brevete: US2017185864-A1; WO2017108222-A1; EP3213257-A1; EP3213257-B1; CN108431824-A; US10460198-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2017433617

[12-CZ-P] Nicoara, N., Raceala, C., **Zaharia, C.**, Fulop, S., & Iovita, O. - *"Image processing system" [Specific - Windowing]* - 2020 - DIIDW: 202012929
Brevet: US2020050885-A1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:202012929Q

[13-CZ-P] Nicoara, N., Raceala, C., **Zaharia, C.**, Fulop, S., & Iovita, O. - *"Image processing system" [Generic]* - 2020 - DIIDW: 2020C0641
Brevet: US2020380291-A1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2020C0641V

[14-CZ-P] Munteanu, M. C., Georgescu, V., **Zaharia, C.**, & Suciu, I. - *"Method for producing a histogram of oriented gradients"* - 2018 - DIIDW:201722756Q [4 citări]
Brevete: US2017098135-A1; WO2017198861-A1; US9977985-B2; CN109478242-A; EP3459010-A1; US2019205691-A1; US10839247-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201722756Q

[15-CZ-P] **Zaharia, C.**, Bigioi, P., & Corcoran, P. - *"Real-time video frame pre-processing hardware"* - 2019 - DIIDW:2012A62692 [26 citări]
Brevete: WO2012004672-A2; US2012008002-A1; WO2012004672-A3; WO2012004672-A9; JP2013531853-W; KR2013098298-A; CN103098077-A; US2015042670-A1; US9053681-B2; JP5819956-B2; KR1646608-B1; CN103098077-B; CN106408502-A; US9607585-B2; US2017256239-A1; US10418001-B2; US2020005739-A1; CN106408502-B; US10930253-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2012A62692

[16-CZ-P] Fulop S., **Zaharia, C.**, & Bigioi, P. - *"Multi-processor neural network processing apparatus"* - 2020 - DIIDW: 202050355Q
Brevete: US2020184321-A1; EP3668015-A2; CN111311476-A; EP3668015-A3; EP3668015-B1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:202050355Q

[17-CZ-P] Munteanu, M., Pososin, A., Stec, P., & **Zaharia, C.** - "*Method and system for correcting a distorted input image*" *[Image Stabilization]* - 2014 - DIIDW:2014A78350 [20 citări]
Brevete: US2014009568-A1; WO2014005783-A1; US8928730-B2; EP2870585-A1; EP2870585-B1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2014A78350

[18-CZ-P] Stec, P., Pososin, A., Munteanu, M., & **Zaharia, C.** - "*Method and system for correcting a distorted input image*" *[Hardware Implementation]*- 2015 - DIIDW: 2015364145 [12 citări]
Brevete: US2015178897-A1; US9262807-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2015364145

[19-CZ-P] Stec, P., Pososin, A., Munteanu, M. C., **Zaharia, C.**, & Georgescu, V. - *"Method And System For Correcting A Distorted Input Image" [Perspective Correction]*- 2016 - DIIDW:2015551294 [25 citări]
Brevete: US2015262344-A1, US9280810-B2, EP3101622-A2, EP3101622-A3, EP3101622-B1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2015551294

[20-CZ-P] Georgescu, V., Munteanu, M. C., Bigioi, P., **Zaharia, C.**, Fulop, S., & Simon, G. - *"Image processing apparatus" [Normalization]* - 2018 - DIIDW:201714663M [5 citări]

Brevete: US2017061639-A1; WO2017032468-A1; EP3341912-A1; CN108352054-A; US10115003-B2; EP3341912-B1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201714663M

[21-CZ-P] Georgescu, V., Munteanu, M. C., Bigioi, P., **Zaharia, C.**, Fulop, S., & Simon, G. - *"Image processing apparatus" [One step rotation and scaling]* – 2019 - DIIDW: 201937402C
Brevete: US2019130164-A1; US11106894-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201937402C

[22-CZ-P] Munteanu, M. C., Caliman, A., & **Zaharia, C.** - *"Convolutional neural network (CNN)" [HW Acceleration - PCNN]* - U.S. Patent No. 9,665,799. Washington, DC: U.S. Patent and Trademark Office - 30 May 2017 - DIIDW:2017344715 [31 de citări]
Brevet: US9665799-B1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2017344715

[23-CZ-P] Munteanu, M. C., Caliman, A., **Zaharia, C.,** & Dinu, D. - *"Convolutional neural network (CNN)" [Floating point 8 optimizations]* - 2017 - DIIDW: 201752317J [37 de citări]
Brevete: US2017221176-A1; US10497089-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201752317J

[24-CZ-P] Bigioi, P., Munteanu, M. C., Caliman, A., **Zaharia, C.,** & Dinu, D., & Kahriman, A. - *"Convolutional neural network (CNN)" [Advanced Memory Management]* - 2019 - DIIDW: 201752615T Brevete: WO2017129325-A1; CN108701236-A; EP3408798-A1; US2020126178-A1; EP3408798-B1; US11087433-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201752615T

[25-CZ-P] **Zaharia, C.,** Bigioi, P., (WO, EP, CN) / Bigioi, P., & **Zaharia, C.** (US) - *"Peripheral processing device"* – 2019 - DIIDW:201919589P [4 citări]
Brevete: US2019065410-A1; WO2019042703-A1; EP3580691-A1; CN111052134-A; EP3580691-B1; US10713186-B2; EP3699816-A1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201919589P

[26-CZ-P] **Zaharia, C.** - *"Vehicle side view camera system with adjustable field of view"* - 2020 - DIIDW: 2020439412 [2 citări]
Brevete: US2020156592-A1; US10836353-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2020439412

[27-CZ-P] **Zaharia, C.** - *"Vehicle side view camera system with adjustable field of view" [un-authorized access]* - U.S. Patent No. 10,836,353. Washington, DC: U.S. Patent and Trademark Office - 17 Nov. 2020 - DIIDW:2021451205
Brevet: US2021129797-A1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2021451205

Alte brevete in domeniu

[28-CZ-P] Liu, X., Gutierrez, R. C., Leang, P. K., Mendez, J. A., **Zaharia, C.**, Drimbarean, A. F., & Bigioi, P. G. - *"MEMS-based optical image stabilization"* - U.S. Patent No. 8,855,476. Washington, DC: U.S. Patent and Trademark Office - 7 Oct. 2014 - DIIDW:2013E28560 [55 de citări]
Brevete: US2013077945-A1; US8855476-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2013E28560

[29-CZ-P] Gutierrez, R. C., Calvet, R. J., Liu, X., Leang, P. K., Mendez, J. A., **Zaharia, C.**, Drimbarean, A. F., & Bigioi, P. G. - *"MEMS-based optical image stabilization"* – 2013 - DIIDW: 2013E30185 [12 citări]
Brevete: US2013076919-A1; WO2013049688-A2; WO2013049688-A3; TW201319628-A; CN103842875-A; US9019390-B2; CN103842875-B; CN106569310-A; TW584001-B1; CN106569310-B
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2013E30185

[30-CZ-P] Liu, X., Gutierrez, R. C., Leang, P. K., Mendez, J. A., **Zaharia, C.**, Drimbarean, A. F., & Bigioi, P. G. - *"MEMS-based optical image stabilization"* - 2015 - DIIDW: 2015209813
Brevete: US2015085363-A1; US9664922-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2015209813

[31-CZ-P] Liu, X., Gutierrez, R. C., Leang, P. K., Mendez, J. A., **Zaharia, C.**, Drimbarean, A. F., & Bigioi, P. G. - *"MEMS-based optical image stabilization"* - 2017 - DIIDW: 201778062L
Brevete: US2017329152-A1; US10203515-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201778062L

[32-CZ-P] Liu, X., Gutierrez, R. C., Leang, P. K., Mendez, J. A., **Zaharia, C.**, Drimbarean, A. F., & Bigioi, P. G. - *"MEMS-based optical image stabilization"* - 2019 - DIIDW: 201949635X
Brevete: US2019171032-A1; US10558057-B2
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:201949635X

[33-CZ-P] Gigushinski, O., Kovalsky, S., Cohen, N., Oz, Y., Stec, P., Drimbarean, A., **Zaharia, C.** & Munteanu, M. C. – *"Pixel mapping using a lookup table and linear approximation"* - U.S. Patent No. 8,340,462. Washington, DC: U.S. Patent and Trademark Office - 25 Dec. 2012. - DIIDW:2012R76133 [15 citări]
Brevete: US8340462-B1
https://www-webofscience-com.am.e-nformation.ro/wos/diidw/full-record/DIIDW:2012R76133

Bibliografie

[34] A. Garvey and V. Lesser, "A survey of research in deliberative real-time artificial intelligence," *Real-Time Systems,* vol. 6, p. 317–347, 1994.

[35] V. Gadepally, J. Goodwin, J. Kepner, A. Reuther, H. Reynolds, S. Samsi, J. Su and D. Martinez, "Ai enabling technologies: A survey," *arXiv preprint arXiv:1905.03592,* 2019.

[36] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi and J. Kepner, "Survey of machine learning accelerators," in *2020 IEEE high performance extreme computing conference (HPEC)*, 2020.

[37] Y. Chen, Y. Xie, L. Song, F. Chen and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering,* vol. 6, p. 264–274, 2020.

[38] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masera and M. Martina, "An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks," *Future Internet,* vol. 12, p. 113, 2020.

[39] S. Mittal and S. Umesh, "A survey on hardware accelerators and optimization techniques for RNNs," *Journal of Systems Architecture,* vol. 112, p. 101839, 2021.

[40] Z. Akkus, Y. H. Aly, I. Z. Attia, F. Lopez-Jimenez, A. M. Arruda-Olson, P. A. Pellikka, S. V. Pislaru, G. C. Kane, P. A. Friedman and J. K. Oh, "Artificial intelligence (AI)-empowered echocardiography interpretation: a state-of-the-art review," *Journal of Clinical Medicine,* vol. 10, p. 1391, 2021.

[41] S. Rabii, "AR/VR Applications at Facebook Reality Labs: Silicon Challenges and Research Directions (Keynote)," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2021.

[42] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal,* vol. 7, p. 7457–7469, 2020.

[43] Y.-L. Lee, P.-K. Tsung and M. Wu, "Techology trend of edge AI," in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2018.

[44] J. McCormack, R. McNamara, C. Gianos, L. Seiler, N. P. Jouppi and K. Correll, "Neon: a single-chip 3d workstation graphics accelerator," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, 1998.

[45] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik and O. O. Storaasli, "State-of-the-art in heterogeneous computing," *Scientific Programming,* vol. 18, p. 1–33, 2010.

[46] J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, Elsevier, 2011.

[47] M. J. Flynn, "Some computer organizations and their effectiveness," *IEEE transactions on computers,* vol. 100, p. 948–960, 1972.

[48] O. Terzo, K. Djemame, A. Scionti and C. Pezuela, Eds., "Heterogeneous Computing Architectures," in *Heterogeneous Computing Architectures*, CRC Press, 2019.

[49] O. Terzo, K. Djemame, A. Scionti and C. Pezuela, Heterogeneous Computing Architectures: Challenges and Vision, CRC Press, 2019.

[50] A. Munshi, "The opencl specification," in *2009 IEEE Hot Chips 21 Symposium (HCS)*, 2009.

[51] A. Munshi, B. Gaster, T. G. Mattson and D. Ginsburg, OpenCL programming guide, Pearson Education, 2011.

[52] R. Kobayashi, N. Fujita, Y. Yamaguchi, A. Nakamichi and T. Boku, "GPU-FPGA heterogeneous computing with opencl-enabled direct memory access," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2019.

[53] B. Gaster, L. Howes, D. R. Kaeli, P. Mistry and D. Schaa, Heterogeneous computing with openCL: revised openCL 1., Newnes, 2012.

[54] J. Sanders and E. Kandrot, CUDA by example: an introduction to general-purpose GPU programming, Addison-Wesley Professional, 2010.

[55] R. Andonie, A. Cataron, H. Galmeanu, M. Ivanovici and L. Sasu, Algoritmi și Structuri de Date pentru Imagistică și Bioinformatică. Note de curs, Brasov: Editura Universitatii Transilvania din Brasov, 2013.

[56] P.-K. Tsung, T.-C. Chen, C.-H. Lin, C.-Y. Chang and J.-M. Hsu, "Heterogeneous computing for edge ai," in *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2019.

[57] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006.

[58] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern recognition,* vol. 30, p. 1145–1159, 1997.

[59] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014.

[60] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[61] P. Kellman and E. R. McVeigh, "Image reconstruction in SNR units: a general method for SNR measurement," *Magnetic resonance in medicine,* vol. 54, p. 1439–1447, 2005.

[62] B. Aiazzi, L. Alparone, S. Baronti, A. Garzelli and M. Selva, "MTF-tailored multiscale fusion of high-resolution MS and Pan imagery," *Photogrammetric Engineering & Remote Sensing,* vol. 72, p. 591–596, 2006.

[63] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas and K. Yelick, "A case for intelligent RAM," *IEEE micro,* vol. 17, p. 34–44, 1997.

[64] C. Hao, Y. Chen, X. Zhang, Y. Li, J. Xiong, W.-m. Hwu and D. Chen, "Effective Algorithm-Accelerator Co-design for AI Solutions on Edge Devices," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020.

[65] S. Bhat, I. T. Basten, I. M. Geilen and I. S. Stuijk, "Energy models for network-on-chip components," *On Dec,* 2005.

[66] S. Rikhi, "Leading at the edge of Moore's Law with Intel Custom Foundry," in *Intel Developer Forum*, 2014.

[67] V. Barannik, S. Podlesny, A. Krasnorutskyi, A. Musienko and V. Himenko, "The ensuring the integrity of information streams under the cyberattacks action," in *2016 IEEE East-West Design & Test Symposium (EWDTS)*, 2016.

[68] A. Khadka, P. Karypidis, A. Lytos and G. Efstathopoulos, "A benchmarking framework for cyber-attacks on autonomous vehicles," *Transportation research procedia,* vol. 52, p. 323–330, 2021.

[69] Y. Dong, H. Su, B. Wu, Z. Li, W. Liu, T. Zhang and J. Zhu, "Efficient decision-based black-box adversarial attacks on face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[70] F. V. Massoli, F. Carrara, G. Amato and F. Falchi, "Detection of face recognition adversarial attacks," *Computer Vision and Image Understanding,* vol. 202, p. 103103, 2021.

[71] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing,* vol. 10, p. 10–5555, 2017.

[72] P. Schaar, "Privacy by design," *Identity in the Information Society,* vol. 3, p. 267–274, 2010.

[73] A. Cavoukian, "Privacy by design," 2009.

[74] L. Tawalbeh, F. Muheidat, M. Tawalbeh, M. Quwaider and others, "IoT Privacy and security: Challenges and solutions," *Applied Sciences,* vol. 10, p. 4102, 2020.

[75] D. W. Chadwick, W. Fan, G. Costantino, R. De Lemos, F. Di Cerbo, I. Herwono, M. Manea, P. Mori, A. Sajjad and X.-S. Wang, "A cloud-edge based data security architecture for sharing and analysing cyber threat information," *Future Generation Computer Systems,* vol. 102, p. 710–722, 2020.

[76] J. Turley, "Introduction to Intel® Architecture: The Basics," *Intel Corporation,* 2014.

[77] J.-W. Jang, S. Lee, D. Kim, H. Park, A. S. Ardestani, Y. Choi, C. Kim, Y. Kim, H. Yu, H. Abdel-Aziz and others, "Sparsity-aware and re-configurable NPU architecture for samsung flagship mobile SoC," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.

[78] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers and others, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017.

[79] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," in *2011 IEEE custom integrated circuits conference (CICC)*, 2011.

[80] J. Nakamura, Image Sensors and Signal Processing for Digital Still Cameras (Optical Science and Engineering), CRC Press, 2005.

[81] B. E. Bayer, *Color imaging array,* Google Patents, 1976.

[82] S. Dave, R. Baghdadi, T. Nowatzki, S. Avancha, A. Shrivastava and B. Li, "Hardware acceleration of sparse and irregular tensor computations of ML models: A survey and insights," *Proceedings of the IEEE,* vol. 109, p. 1706–1752, 2021.

[83] V. Sze, Y.-H. Chen, T.-J. Yang and J. S. Emer, "Efficient processing of deep neural networks," *Synthesis Lectures on Computer Architecture,* vol. 15, p. 1–341, 2020.

[84] V. Sze, Y.-H. Chen, T.-J. Yang and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE,* vol. 105, p. 2295–2329, 2017.

[85] T. Leyvand, D. Cohen-Or, G. Dror and D. Lischinski, "Digital face beautification," in *ACM Siggraph 2006 Sketches*, 2006, p. 169–es.

[86] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision,* vol. 74, p. 59–73, 2007.

[87] D. T. Nguyen, M. Quach, G. Valenzise and P. Duhamel, "Lossless coding of point cloud geometry using a deep generative model," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 31, p. 4617–4629, 2021.

[88] M. Silberstein, "GPUs: High-performance accelerators for parallel applications: the multicore transformation (ubiquity symposium)," *Ubiquity,* vol. 2014, p. 1–13, 2014.

[89] C.-Y. Wang, I.-H. Yeh and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," *arXiv preprint arXiv:2105.04206,* 2021.

[90] M. Buckler, S. Jayasuriya and A. Sampson, "Reconfiguring the imaging pipeline for computer vision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[91] P. Hansen, A. Vilkin, Y. Krustalev, J. Imber, D. Talagala, D. Hanwell, M. Mattina and P. N. Whatmough, "ISP4ML: The role of image signal processing in efficient deep learning vision systems," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021.

[92] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet and J. T. Barron, "Unprocessing images for learned raw denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[93] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing,* vol. 29, p. 1153–1160, 1981.

[94] L. He, H. Wu and X. Zhao, "The application of Lanczos interpolation in video scaling system based on FPGA," in *Twelfth International Conference on Signal Processing Systems*, 2021.

[95] K. Papadopoulos and K. Vlachos, "Efficient Projective Transformation and Lanczos Interpolation on ARM Platform using SIMD Instructions.," in *VISIGRAPP (4: VISAPP)*, 2018.

[96] S. Safinaz and A. R. Kumar, "VLSI realization of Lanczos interpolation for a generic video scaling algorithm," in *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*, 2017.

[97] S. Boukhtache, B. Blaysat, M. Grédiac and F. Berry, "FPGA-based architecture for bi-cubic interpolation: the best trade-off between precision and hardware resource consumption," *Journal of Real-Time Image Processing,* vol. 18, p. 901–911, 2021.

[98] M. K. Dabhi and B. K. Pancholi, "Face detection system based on Viola-Jones algorithm," *International Journal of Science and Research (IJSR),* vol. 5, p. 62–64, 2016.

[99] Y.-Q. Wang, "An analysis of the Viola-Jones face detection algorithm," *Image Processing On Line,* vol. 4, p. 128–148, 2014.

[100] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2009.

[101] S. Theodoridis and K. Koutroumbas, Pattern Recognition, Fourth Edition, Elsevier, 2009.

[102] P. Jones, P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *University of Rochester. Charles Rich*, 2001.

[103] K. G. Derpanis, "Integral image-based representations," *Department of Computer Science and Engineering York University Paper,* vol. 1, p. 1–6, 2007.

[104] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[105] P.-Y. Chen, C.-C. Huang, C.-Y. Lien and Y.-H. Tsai, "An efficient hardware implementation of HOG feature extraction for human detection," *IEEE Transactions on Intelligent Transportation Systems,* vol. 15, p. 656–662, 2013.

[106] S. Ghaffari, P. Soleimani, K. F. Li and D. Capson, "FPGA-based implementation of HOG algorithm: Techniques and challenges," in *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2019.

[107] T. P. Cao and G. Deng, "Real-time vision-based stop sign detection system on FPGA," in *2008 Digital Image Computing: Techniques and Applications*, 2008.

[108] N. Pettersson, L. Petersson and L. Andersson, "The histogram feature-a resource-efficient weak classifier," in *2008 IEEE intelligent vehicles symposium*, 2008.

[109] K. Negi, K. Dohi, Y. Shibata and K. Oguri, "Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm," in *2011 International Conference on Field-Programmable Technology*, 2011.

[110] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann and K. Doll, "FPGA-based real-time pedestrian detection on high-resolution images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013.

[111] C. Bourrasset, L. Maggiani, C. Salvadori, J. Sérot, P. Pagano and F. Berry, "FPGA implementations of Histograms of Oriented Gradients in FPGA," in *Workshop on architecture of smart cameras (WASC)*, 2013.

[112] M. Bilal, A. Khan, M. U. K. Khan and C.-M. Kyung, "A low-complexity pedestrian detection framework for smart video surveillance systems," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 27, p. 2260–2273, 2016.

[113] T. Adiono, K. S. Prakoso, C. D. Putratama, B. Yuwono and S. Fuada, "Practical implementation of a real-time human detection with HOG-AdaBoost in FPGA," in *TENCON 2018-2018 IEEE Region 10 Conference*, 2018.

[114] A. Sasongko and B. Sahbani, "VLSI Architecture for Fine Grained Pipelined Feature Extraction using Histogram of Oriented Gradient," in *2019 IEEE 7th Conference on Systems, Process and Control (ICSPC)*, 2019.

[115] G. Toacse and D. Nicula, Electronica Digitala, Dispozitive, Circuite, Proiectare, Bucuresti: Editura Tehnica, 2005.

[116] M. D. Ercegovac and T. Lang, Digital Arithmetic, 1st Edition, Elsevier, 2003.

[117] K. Karthikeyan, A. V. Karpagam and M. Manikandan, "Influence of binning in Histograms of Oriented Gradients method representation," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2017.

[118] M.-S. Wang and Z.-R. Zhang, "FPGA implementation of HOG based multi-scale pedestrian detection," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, 2018.

[119] R. Benenson, M. Mathias, R. Timofte and L. Van Gool, "Pedestrian detection at 100 frames per second," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[120] J. Kristensen and J.-O. Nilsson, "Fast triangular binning kernel approximations for weighted gradient histogram creation," in *2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2014.

[121] P. M. Corcoran, P. Bigioi and P. Stec, "Wide field of view (WFoV) imaging for consumer devices," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*, 2014.

[122] M. L. V. Pitteway and D. J. Watkinson, "Bresenham's algorithm with grey scale," *Communications of the ACM,* vol. 23, p. 625–626, 1980.

[123] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky and others, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints,* p. arXiv–1605, 2016.

[124] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

[125] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga and others, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems,* vol. 32, 2019.

[126] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard and others, "{TensorFlow}: A System for {Large-Scale} Machine Learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016.

[127] S. Shibahara, C. Takahashi, K. Fukuoka, Y. Kitaji, T. Irita, H. Hara, Y. Shimazaki and J. Matsushima, "A 16 nm FinFET heterogeneous nona-core SoC supporting ISO26262 ASIL B standard," *IEEE Journal of Solid-State Circuits,* vol. 52, p. 77–88, 2016.

[128] R. Salay, R. Queiroz and K. Czarnecki, "An analysis of ISO 26262: Using machine learning safely in automotive software," *arXiv preprint arXiv:1709.02435,* 2017.

[129] R. Szeliski, S. Winder and M. Uyttendaele, "High-quality multi-pass image resampling," *Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2010-10,* 2010.

[130] B. Froba and A. Ernst, "Face detection with the modified census transform," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*., 2004.

[131] C.-H. Chan, J. Kittler and K. Messer, "Multi-scale local binary pattern histograms for face recognition," in *International conference on biometrics*, 2007.

[132] M. Weber and J. Weisbrod, "Requirements engineering in automotive development-experiences and challenges," in *Proceedings IEEE Joint International Conference on Requirements Engineering*, 2002.

[133] M. Mutz, M. Huhn, U. Goltz and C. Kromke, "Model based system development in automotive," *SAE SP,* p. 1–10, 2003.

[134] J. C. Mankins and others, "Technology readiness levels," *White Paper, April,* vol. 6, p. 1995, 1995.

[135] M. Héder, "From NASA to EU: the evolution of the TRL scale in Public Sector Innovation," *The Innovation Journal,* vol. 22, p. 1–23, 2017.

[136] I. Bruno, G. Lobo, B. V. Covino, A. Donarelli, V. Marchetti, A. S. Panni and F. Molinari, "Technology readiness revisited: a proposal for extending the scope of impact assessment of European public services.," in *ICEGOV*, 2020.

[137] S. Yang, P. Luo, C.-C. Loy and X. Tang, "Wider face: A face detection benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[138] V. Jain and E. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," UMass Amherst technical report. (Vol. 2, No. 6), 2010.

Web-grafie

- accesată în anul universitar 2021-2022

[139] Brock K., Sunopsys, "Building Efficient Deep Learning Accelerators from the Foundation Up", https://www.synopsys.com/designware-ip/technical-bulletin/building-efficient-deep-learning-dwtb_q318.html

[140] ARM, "ARM7TDMI Tecnical Reference Manual", https://developer.arm.com/documentation/ddi0210/c/Introduction/Block--core--and-functional-diagrams

[141] CAST, "BA22-CE 32-bit Cache-Enabled Embedded Processor", https://www.cast-inc.com/processors/32-bit/ba22-ce

[142] Harris M., nVidia, "Inside Pascal: NVIDIA's Newest Computing Platform", https://developer.nvidia.com/blog/inside-pascal/

[143] ARM, "Mali-G72", https://developer.arm.com/Processors/Mali-G72

[144] ARM, "Mali-470", https://developer.arm.com/Processors/Mali-470

[145] Beets, K., "Imagination, Introducing Furian: the architectural changes", https://blog.imaginationtech.com/introducing-furian-the-architectural-changes/

[146] Schiesser T., "AMD launches Ryzen 6000 series for laptops: What's new with the Zen 3+ architecture?", https://www.techspot.com/news/93424-amd-launches-ryzen-6000-mobile-what-new-zen.html

[147] Qualcomm, "DSP Processor", https://developer.qualcomm.com/software/hexagon-dsp-sdk/dsp-processor

[148] "Vision-Based Advanced Driver Assistance: TI Hopes You'll Give Its Latest SoCs a Chance", https://www.bdti.com/InsideDSP/2013/10/23/TI

[149] Movidius, "VPU Product brief", https://static6.arrow.com/aropdfconversion/5a53549959ba1304f155469049d98b3ade903558/1463156689-2016-04-29_vpu_productbrief.pdf

[150] Verisilicon, "Neural Network Processor IP Series for AI Vision and AI Voice", https://verisilicon.com/en/IPPortfolio/VivanteVIP9000

[151] CEVA, "CEVA-XM4 - Intelligent imaging and vision processor for low-power embedded systems" https://www.ceva-dsp.com/product/ceva-xm4/

[152] "Xilinx Zynq®-7000 SoC First Generation Architecture", https://www.mouser.co.uk/new/xilinx/xilinx-zynq-7000-socs/

[153] Xilinx, "Flexible DSP Solutions" ,https://www.xilinx.com/products/technology/dsp.html

[154] Intel, "Intel® Arria® 10 FPGAs & SoCs", https://www.intel.com/content/www/us/en/products/details/fpga/arria/10/article.html

[155] DesignNews, "Programmable Logic- How do they do that?", https://slideplayer.com/slide/15148092/

[156] Lake Crest - Microarchitectures - Intel Nervana, https://en.wikichip.org/wiki/nervana/microarchitectures/lake_crest

[157] Spring Crest - Microarchitectures - Intel Nervana, https://en.wikichip.org/wiki/nervana/microarchitectures/spring_crest

[158] Spring Hill - Microarchitectures – Intel, https://en.wikichip.org/wiki/intel/microarchitectures/spring_hill

[159] [ISP] Image Processing Pipeline, An Overview, http://iamard.blogspot.com/2014/01/isp-image-processing-pipeline-overview.html

[160] Bigioi P., "Ultra-Efficient VPU: Low-power Deep Learning, Computer Vision and Computational Photography", https://www.edge-ai-vision.com/2017/08/ultra-efficient-vpu-low-power-deep-learning-computer-vision-and-computational-photography-a-presentation-from-fotonation/

[161] *Elk O. (2018). A Peek at Semantic Segmentation 2018.* - https://medium.com/@kangyan.cn/a-peek-at-semantic-segmentation-2018-a7ed168ff73f

[162] *Huawei and FotoNation Partnership* – https://investor.xperi.com/news/news-details/2015/Huawei-and-FotoNation-Strengthen-Partnership-with-Collaboration-on-Latest-Smartphones/default.aspx
[163] *News on Nikon Coolpix* – https://www.nikon.com/news/2009/0203_coolpixs230_04.htm
[164] *Nikon imaging, the Z Series* – https://imaging.nikon.com/lineup/mirrorless/z_7_2/spec.htm
[165] *Xperi NASDAQ Annual Report* – https://www.annualreports.com/HostedData/AnnualReports/PDF/NASDAQ_XPER_2020.pdf
[166] *FotoNation Demonstration of Its Image Processing Unit (IPU) 2.0* – www.youtube.com/watch?v=zVE02XgMU2A
[167] *Xperi – Embedding programmable Deep learning NNs (DNNs) in low power SoCs* – www.edge-ai-vision.com/2018/07/embedding-programmable-dnns-in-low-power-socs-a-presentation-from-xperi/
[168] *Xperi – Driver monitoring systems* – www.edge-ai-vision.com/2020/12/driver-monitoring-systems-present-and-future-a-presentation-from-xperi/
[169] *Xperi – World-first in-cabin monitoring* – www.prophesee.ai/2021/04/21/xperi-develops-world-first-in-cabin-monitoring-technologies
[170] *FotoNation and Socionext deliver EIS Technology for On-the-Move Video* – www.socionext.com/en/pr/sn_pr20160426_01e.pdf
[171] *FotoNation – Demonstration of EIS* – www.edge-ai-vision.com/2016/08/fotonation-demonstration-of-electronic-image-stabilization/
[172] *FotoNation – Rock-steady image stabilization on a smartphone* – https://vimeo.com/160214764
[173] *International Standards Organisation - Road vehicles - Functional safety* - https://www.iso.org/standard/68383.html
[174] *Beyond Semiconductor – The BA22 processors family* – https://www.beyondsemi.com/processors/
[175] RISC-V International – The instruction set architecture (ISA) – A.Waterman, Y.Lee, D.Patterson, K.Asanovic – https://riscv.org/wp-content/uploads/2015/05/riscv-compressed-spec-v1.7.pdf

Rezumat (RO)

Teza de doctorat "Contribuții la Relația Inteligența Artificială-Hardware pentru Procesarea Imaginilor în Timp Real" se axează pe identificarea, analiza, definirea conceptuală/arhitecturală și implementarea de soluții pentru sistemele de procesare de imagini în timp real care folosesc algoritmi cu inteligență artificială (IA). Lucrarea definește cerințele specifice ale sistemelor analizate și propune *metrici de analiză* a soluțiilor implementate. Abordarea algoritmilor presupune analiza structurilor de date prelucrate, specificarea rezultatelor așteptate și identificarea resurselor hardware (HW) și software (SW) alocate pentru execuția algoritmilor. Se constituie astfel un *ecosistem algoritm – date – software – hardware* pentru soluțiile de timp real, care corespunde unei abordări integrative (quasi-holistice) deoarece aceste componente sunt într-o relație de strânsă interdependență. De aceea, soluțiile propuse nu sunt doar modificări la nivel HW sau SW, ci și soluții de modificare a algoritmilor de IA sau de modificare a reprezentării datelor (de intrare, ieșire sau intermediare). Astfel rezultă două categorii de soluții. Prima categorie este aceea în care organizarea mai bună a datelor sau ameliorarea algoritmilor de IA ajută implementarea fizică – categoria *"IA pentru HW"*. Astfel, se propun soluții pentru integrarea adaptivă a resurselor HW, parametrizarea HW de către IA, controlul preciziei calculelor implementate în HW precum și controlul prin IA al fluxului de date din HW. Cea de a doua categorie conține soluțiile în care măsurile luate la nivel fizic (HW) duc la utilizarea mai eficientă a datelor și la creșterea performanței algoritmilor de IA – categoria *"HW pentru IA"*. Se propun soluții pentru optimizarea capacității de procesare pentru rețelele neurale precum și pentru utilizarea eficientă a magistralelor de date. Modul de abordare pentru găsirea și validarea soluțiilor propuse duce la dezvoltarea unei metodologii de proiectare specifică, cu accent pe *co-design,* în care elementul principal este algoritmul de implementat, în jurul căruia se iau măsurile potrivite de construire a soluției. Finalitatea constă într-un număr mare de patente și în integrarea în *sisteme comerciale* produse pe scară largă, care este prezentată pe studii de caz relevante (detecția de obiecte, stabilizarea imaginilor și monitorizarea exteriorului și interiorului autovehiculelor).

Abstract (EN)

The "Contributions to the Artificial Intelligence-Hardware Relationship in Real Time Image Processing" PhD thesis focuses on the identification, analysis, conceptual/architectural definition, and implementation of solutions for real-time image processing systems that use artificial intelligence (AI) algorithms. The thesis defines the specific requirements of the analyzed systems and proposes *analysis metrics* for the implemented solutions. Algorithms approach involves the analysis of processed data structures, specifying the expected results, and identifying the hardware (HW) and software (SW) resources allocated for the algorithms execution. This constitutes an *algorithm – data – software – hardware ecosystem* for real-time solutions, corresponding to an integrative approach (quasi-holistic) because these components are in a close interdependence relationship. Therefore, the proposed solutions are not only changes at the HW or SW level, but also solutions to modify the AI algorithms or to modify the representation of the data (input, output or intermediate). This results in two categories of solutions. First category is the one where better data management or improved AI algorithms help physical implementation – the *"AI for HW".* Thus, solutions are proposed for the adaptive integration of HW resources, the parameterization of HW by AI, accuracy control of the calculations implemented in HW as well as AI control of the HW data flow. The second category contains solutions where measures taken at the physical level (HW) lead to more efficient use of data and increased performance of AI algorithms – the *category "HW for AI"*. Solutions are proposed to optimize the processing capacity for neural networks as well as to efficiently use data buses. The approach for finding and validating the proposed solutions leads to the development of a specific design methodology, with an emphasis on *co-design,* in which the main element is the algorithm to be implemented, around which the right measures to build the solution are taken. The aim is a large number of patents and the integration in large-scale produced *commercial systems*, which is presented on relevant case studies (object detection, image stabilization and vehicles exterior and interior monitoring).