UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE

Rezumat Teză de Doctorat

Sisteme de reprezentare a numerelor în ingineria calculatoarelor

Ștefan-Dan Ciocîrlan

Coordonator științific:

Prof. Dr. Ing. Răzvan-Victor Rughiniș

BUCUREȘTI
2023

CUPRINS

SINOPSIS

Calculele matematice reprezintă unul dintre elementele fundamentale ale umanității. Calculatoarele utilizează sisteme de reprezentare a numerelor pentru a emula seturile și operațiile matematice. Computerele digitale pot reprezenta doar subseturi finite ale seturilor infinite din matematică, Această limitare produce erori de calcul. Fiecare implementare software de calculator utilizează un anumit set de numere. Ar trebui să existe un sistem specific de reprezentare a numerelor ce poate produce cea mai bună precizie și eficiență pentru aplicația data. Soluția este utilizarea unui sistem care este suficient de bun. Recent au fost propuse noi sisteme de reprezentare a numerelor și s-au făcut cercetări semnificative pentru a vedea avantajele și dezavantajele în comparație cu soluția actuală. Discuția ar putea trece din nou printr-o "epocă întunecată" timp de câteva decenii, fără a avea o nouă propunere. Problema abordată în această teză este menținerea discuției despre sistemele de reprezentare a numerelor vie și interesantă. Soluția propusă reduce efortul depus pentru a intra în discuție prin crearea unei infrastructuri de cercetare mai bune. Infrastructura de cercetare conține o bibliotecă software și o bibliotecă generatoare de hardware. Biblioteca software propusă reduce timpul și efortul de a propune un nou sistem de reprezentare a numerelor la implementarea reprezentării binare și a constrângerilor numerice. Biblioteca generatoare de hardware îmbunătățește timpul până la unități hardware prototip. Infrastructura este utilizată pentru a propune trei noi sisteme de reprezentare a numerelor: MorrisHEB, MorrisBiasHEB și MorrisUnaryHEB. MorrisBiasHEB are primul sau al doilea cel mai bun rezultat în cadrul testelor de referință din literatura de specialitate. Unitățile sale hardware sunt printre cele mai rapide, cele mai eficiente din punct de vedere al resurselor și cele mai eficiente din punct de vedere energetic. Unitatea de acumulator Kulisch pentru MorrisBiasHEB pe 16 biți are o frecvență maximă de $1,81$ ori mai mare și utilizează cu $23,98\%$ mai puține LUT-uri decât al doilea cel mai bun din categoria respectivă. MorrisUnaryHEB se definește doar prin $dimensiunea$ sa și are cel mai mare procent de rezultate exacte pentru adunare ($37,6\%$), cea mai aglomerată "zonă de aur", o precizie zecimală mai bună la operațiile unare. Pe lângă aceste două biblioteci, teza oferă o bibliotecă de metode statistice (14 metode statistice), patru evaluări de calcul științific, două aplicații de învățare automată în precizie redusă și un sistem hardware-software complet specific unui sistem de reprezentare a numerelor. Pentru metodele statistice, Positul pe 16 biți oferă rezultate similare cu cele ale omologilor săi pe 32 de biți. Pe de altă parte, IEEE754 pe 16 biți nu oferă rezultate valide în cazul a 6 metode. Înlocuirea IEEE754 în calculul științific nu este necesară, chiar dacă Posit are cel puțin o zecimală corectă în plus la simularea cu N-Body. Aplicațtile de învățare automată de precizie redusă reduc dimensiunea unei rețele neurale profunde cu $66,78\%$, 75% și $76,78\%$ cu o degradare a preciziei mai mică de 2% prin utilizarea diferitelor sisteme de reprezentare a numerelor. Sistemul complet hardware-software Posit pe 16 biți ($es = 1$) oferă o creștere a vitezei de 18% pentru o rețea nurală convoluțională, o precizie similară și utilizează mai puține resurse hardware decât IEEE754 pe 32 de biți.

Cuvinte-cheie: Analiză numerică, Sisteme de reprezentare a numerelor, Arhitectura calculatoarelor, IEEE754, Posit, Calcul științific, Inteligență artificială, Metode statistice

1 INTRODUCERE

$1+1=2$ poate părea o ecuație simplă, amuzantă și inocentă, dar într-un fel este unul dintre elementele fundamentale ale umanității. Ca un prim argument, întreaga educație matematică umană se bazează pe ea. Pentru majoritatea oamenilor, este prima ipoteză/operație/echivalență pe care o învață. Apoi, toate conceptele matematice următoare sunt explicate pornind de la ea. De exemplu, următoarea adunare simplă $1+2=3$ este explicată ca $1+1+1=3$. Numărătoarea de unu dă răspunsul. Cu timpul, aceste operații au devenit inconștiente, iar valoarea numărului este separată de numărul de unu. Un alt exemplu este înmulțirea prezentată ca adunări multiple. Din tranzitivitate, toate conceptele matematice viitoare bazate pe aceste operații se bazează și ele pe $1+1=2$. Dacă conceptele matematice sunt adunate într-o structură arborescentă bazată pe dependența lor, atunci $1+1=2$ este un nod rădăcină sau cel puțin unul dintre cele mai apropiate noduri de rădăcină. Al doilea argument este că majoritatea tehnologiilor umane actuale se bazează pe sau sunt create cu ajutorul conceptelor matematice. În școli, este nevoie de cunoașterea matematicii pentru a învăța materii științifice mai complexe. Este posibil să se explice fizica fără concepte matematice. Nici mai puțin, Nu se poate susține că pentru unele aspecte complexe ale fizicii, calea matematică este cea mai ușoară. Se propune un joc al minții pentru cititor în explicarea electricității (AC/DC [1]) fără concepte matematice (fără $1+1=2$). La momentul redactării acestei teze, inteligența artificială începe să devină obișnuită și familiară oamenilor. Majoritatea modelelor de inteligență artificială sunt, de fapt, construcții matematice. Celălalt aspect important (pe care unii îl consideră chiar mai important decât modelul) al inteligenței artificiale este reprezentat de datele utilizate pentru antrenament. Aceste date sunt, de obicei, transformate și stocate într-o formă numerică. Similar cu inteligența artificială, există concepte precum prelucrarea digitală a semnalelor, calculul științific, statistica și modelele economice, care au fost și sunt parte a tehnologiilor umane actuale bazate pe concepte matematice. Acestea au avut, de asemenea, o dezvoltare similară cu cea a inteligenței artificiale. Chiar dacă inteligența artificială va fi înlocuită ca tehnologie emergentă/utilizată dominantă, istoria ne învață să prezicem cu o probabilitate mare că tehnologia care o va înlocui se va baza pe $2=1+1$.

Un sistem de reprezentare a numerelor (NRS) este un mod de reprezentare a conceptului de valoare numerică și a modului în care funcționează operațiile de bază asupra numerelor. Pur și simplu, acesta poate fi modul în care este scrisă și înțeleasă ecuația $1+1=2$. În prezent, sistemul uman de reprezentare a numerelor este cel zecimal. Sunt utilizate zece cifre (de la 0 la 9), iar rezultatul adunării a două cifre care are un rezultat mai mare de 9 adaugă o cifră în plus în stânga. Cifra cea mai din stânga este, de asemenea, considerată cifra cea mai semnificativă. Un alt sistem folosit în trecut este sistemul de reprezentare a

[1] nu trupa rock

numerelor romane, în care operația $1 + 1 = 2$ era $I + I = II$. Pentru ambele, operația din interiorul minții umane poate fi similară sau diferită. Cititorii se pot gândi la modul în care a evoluat pentru ei în mintea lor operația de înmulțire. La început, se foloseau adunări multiple, dar în timp, operația s-a îmbunătățit. De asemenea, aceasta poate avea căi diferite de îmbunătățire. Cititorii sunt invitați să analizeze diferența dintre metoda japoneză de înmulțire și metoda arabă de înmulțire. O altă diferență ar putea fi modul în care mintea umană proiectează valorile numerelor. Acestea sunt conceptele de bază ale unui sistem de reprezentare a numerelor:

- modul în care simbolurile reprezintă numerele are două procese: modul în care simbolurile sunt transformate în valori numerice (decodificare) și modul în care numerele sunt transformate în simboluri (codificare).
- modul în care se efectuează operațiile pentru numere.

Calculatoarele sunt creații umane care ajută la calculul matematic. În dezvoltarea calculatoarelor, au fost selectate diferite sisteme de reprezentare a numerelor. Aceste sisteme sunt cele binare și finite. Spre deosebire de mintea umană, limitele lor sunt cunoscute. Aceste limite creează erori. Cumva, s-a creat mitul conform căruia calculatoarele sunt perfecte și exacte în calcule. Acest mit este fals. Sistemele de reprezentare a numerelor sunt iluziile care mențin mitul în viață. Nici nu mai puțin, odată cu creșterea numărului de calcule și a încrederii oamenilor în calculele matematice ale calculatoarelor, iluzia începe să se destrame. Pentru a depăși acest lucru, au fost propuse noi sisteme de reprezentare a numerelor [10, 11]. Această teză a început ca o cercetare pentru a găsi domeniile în care sistemul de reprezentare a numerelor Posit [11] poate îmbunătăți iluzia. Au fost evaluate metode statistice, algoritmi de calcul științific, procesare digitală a semnalelor, inteligență artificială și implementare hardware. Pe această cale, autorul a propus trei noi sisteme de reprezentare a numerelor și le-a testat pe operațiile de bază față de sistemele de reprezentare a numerelor existente. În acel moment, problema era încă erorile care ar putea rupe iluzia, iar întrebările de cercetare au fost următoarele

- Care este cel mai bun sistem de reprezentare a numerelor pentru calculul general?
- Sunt sistemele de reprezentare a numerelor dependente de aplicația specifică?
- Care este cel mai eficient sistem de reprezentare a numerelor din punct de vedere energetic?
- Care este compromisul dintre viteză, precizie și consum de energie pentru un sistem de reprezentare a numerelor?

La aceste întrebări se răspunde prin intermediul capitolelor acestei teze, dar acesta nu a fost sfârșitul călătoriei sau al problemei abordate de această teză. Problema este că discuția despre sistemele de reprezentare a numerelor s-a oprit, iar omenirea a acceptat soluția "suficient de bună". Având în vedere implicațiile unui sistem de reprezentare a numerelor pentru calculele matematice pe calculator, acest lucru trebuie evitat în viitor. Lucrarea din această teză, pe

lângă evaluarea implicațiilor sistemelor de reprezentare a numerelor în mai multe domenii, cum ar fi statistica, inteligența artificială, calculul științific și arhitectura calculatoarelor, creează o infrastructură de cercetare care facilitează menținerea în viață a discuției despre sistemele de reprezentare a numerelor. Sunt implementate două biblioteci:

- O bibliotecă software al cărei scop este de a reduce efortul de a propune un sistem de reprezentare a numerelor pentru a defini codificarea/decodificarea și limitele numerelor pe care le poate reprezenta. Al doilea obiectiv al bibliotecii este de a face fără efort testele de referință pentru sistemele de reprezentare a numerelor propuse. În acest fel, cercetătorii și inginerii își pot testa ideea unui sistem de reprezentare a numerelor fără a trece prin implementarea unor criterii de referință plictisitoare. De asemenea, aceștia își obțin rezultatele în contrast cu sistemele de reprezentare a numerelor existente.
- O bibliotecă de generatoare hardware al cărei scop este de a genera unități funcționale multiple (FPU, KAU, unități binare/unitare, acceleratoare) cu eforturi similare bibliotecii software (codificare/decodificare și limite). Reperele și integrarea SoC nu necesită niciun efort din partea dezvoltatorului. Această bibliotecă este complementară bibliotecii software propuse. Ea oferă un pipeline rapid pentru un prototip hardware al unui nou sistem de reprezentare a numerelor propus.

Contribuțiile acestei teze sunt:

- Biblioteca software propusă poate reduce efortul de a propune un nou sistem de reprezentare a numerelor și de a compara mai multe sisteme de reprezentare a numerelor. Ea conține 14 sisteme generice de reprezentare a numerelor.
- Trei noi sisteme de reprezentare a numerelor prin introducerea conceptului de bit de exponent ascuns.
- Analiza a 14 metode statistice în cadrul sistemelor de reprezentare multiplă a numerelor în ceea ce privește precizia și dimensiunea de stocare.
- Patru criterii de referință în calculul științific pentru testarea diferitelor sisteme de reprezentare a numerelor.
- Două framework-uri de învățare automată de precizie redusă pentru reducerea dimensiunii rețelelor neuronale profunde, menținând însă acuratețea în limitele unui prag.
- Biblioteca de generatoare hardware propusă generează unități funcționale (FPU, KAU, module de operații binare/unitare) pentru sistemele de reprezentare a numerelor. De asemenea, reduce efortul de a avea un prototip hardware funcțional la implementarea doar a modulelor de codificare și decodificare.
- Unitatea generală de virgulă mobilă (GFPU) propusă funcționează simultan cu diferite sisteme de reprezentare a numerelor.
- Analiza unui întreg sistem hardware-software pentru un sistem de reprezentare a numerelor pe criterii de referință pe trei niveluri.
- Toate bibliotecile, cadrele și criteriile de referință de mai sus funcționează fără efort pentru orice nou sistem de reprezentare a numerelor adăugat la bibliotecile de generatoare software și hardware propuse.

2 DEZVOLTAREA BIBLIOTECII DE SOFTWARE NRSS (NRS-SL)

2.1 Bit ascuns de exponent

În această secțiune, sunt introduse cele trei noi reprezentări bazate pe virgulă mobilă conică Morris cu un bit de exponent ascuns.

2.1.1 MorrisHEB(size, g, r)

Punctul flotant conic introdus de Morris în [16] pare a fi un concept bun. Utilizarea sa a fost demonstrată în cadrul sistemului de pozit propus de Gustafson [11]. Problema majoră este reprezentată de multiplele modalități de a reprezenta același număr. O soluție pentru aceasta constă în împrumutarea conceptului de bit ascuns de la mantisă. Câmpul g nu reprezintă doar valoarea G, valoarea care dictează mărimea exponentului, ci și poziția celui mai semnificativ cel mai semnificativ bit setat în exponent. Dacă valoarea mărimii exponentului este menținută la $G+1$, atunci valoarea minimă absolută a exponentului este 2 atunci când $G = 0$. Exponentul este *exponent* = *semnul exponentului*. $\times$ $((1 \ll es)$ + *exponent binar*$)$. Este necesar ca valoarea exponentului să fie zero. A soluție pentru aceasta este schimbarea formulei pentru mărimea exponentului în $es = G - 1$. Adresa exponent este acum:

$$\text{exponent} = \begin{cases} (-1)^{\text{semnul exponentului}} \times (2^{es} + \text{ExponentBinar}), & es \neq -1 \\ 0, & es = -1. \end{cases} \tag{1}$$

Acest NRS se numește MorrisHEB(size, g, r).

Următoarea formulă este utilizată pentru a calcula valoarea tuturor celor trei noi NRS-uri reprezentări binare prezentate în această secțiune:

$$\text{valoare} = \begin{cases} 0, & \text{toți biții 0} \\ \text{NR}, & \text{primul bit 1 și restul 0s} \\ (-1)^{\text{sign}} \times 2^{\text{exponent}} \times (1 + \frac{\text{f}}{2^{\text{fs}}}), & \text{în caz contrar} \end{cases} \tag{2}$$

Diferențele constau în modul în care se calculează es și $exponentul$. MorrisHEB(size, g, r) subcontrolează la 0, supracontrolează la NR, și folosește TaperedFloatingPoint(size) pentru implementarea operațiilor.

Reprezentarea binară începe cu bitul de semn. Următorii g biți reprezintă valoarea G în format natural de bază 2. Bitul de semn al exponentului urmează după g. câmp. Următorii e biți ($es = G-1$) sau următorii biți rămași (oricare dintre aceștia este mai mică) reprezintă valoarea

exponentului binar în format natural de bază 2. Dacă es este mai mare decât numărul de biți rămași, biții rămași reprezintă cea mai semnificativi ai valorii exponentului binar. Cei mai puțin semnificativi biți rămași biți mai puțin semnificativi ai valorii exponentului binar vor fi considerați 0. După ce se ia biții exponentului, biții rămași sunt biți de fracție, iar numărul lor reprezintă mărimea fracției. În rezumat, formatul binar este:

$$s_f G_{\text{g-1}} G_{\text{g-2}} ... G_0 s_e e_{\text{es-1}} e_{\text{es-2}} ... e_0 f_{\text{fs-1}} f_{\text{fs-2}} ... f_0 ... f_0 \tag{3}$$

2.1.2 MorrisBiasHEB(size, g, r)

S-ar putea argumenta că problema reprezentărilor multiple nu este încă rezolvată deoarece chiar și exponentul poate avea valori multiple (pentru $es = -1$ exponentul nu contează). Problema provine din faptul că există un bit dedicat semnul exponentului. Această problemă este deja rezolvată în IEEE754 prin utilizarea unei valori de polarizare. Un bias valoare g este propusă. Semnul exponentului este semnul lui G și exponentul mărimea exponentului este $es = |G| - 1$. O altă problemă cu reprezentările Morris și MorrisHEB este că acestea nu au un ordin în formă binară. O soluție în acest sens este de a avea biții exponentului să fie negați atunci când G este negativ. Acest lucru facilitează implementarea unei unități de comparare hardware. NRS cu aceste caracteristici se numește MorrisBiasHEB(size, g, r), unde exponentul este:

$$\text{exponent} = \begin{cases} \text{signum(G)} \times (2^{es} + \text{exponent binar}), & es \neq -1 \\ 0, & es = -1. \end{cases} \tag{4}$$

MorrisBiasHEB(size, g, r) se subcontractualizează la 0, se supracontractualizează la NR, și folosește TaperedFloatingPoint(size) pentru implementarea operațiilor. Reprezentarea binară începe cu bitul de semn. Următorii g biți reprezintă valoarea G în bias cu *bias* $= 2^{g-1} - 1$. Aceasta înseamnă că $G =$ *g binar* $G -$ *bias*. Următorii es biți ($es = |G| - 1$) sau următorii biți rămași (oricare dintre ei este mai mici) reprezintă exponentul în format natural de bază 2, dacă signum(G) este 1. În caz contrar, aceștia trebuie negați, iar rezultatul este exponentul binar valoarea. În cazul în care es este mai mare decât numărul de biți rămași, restul biții rămași reprezintă cei mai semnificativi biți ai valorii exponentului binar. biții rămași mai puțin semnificativi ai valorii exponentului binar sunt considerați 0. După ce se iau biții exponentului, biții rămași sunt biți de fracție și numărul lor reprezintă mărimea fracției. Pe scurt, formatul binar este:

$$s G_{\text{g-1}} G_{\text{g-2}} ... G_0 e_{\text{es-1}} e_{\text{es-2}} ... e_0 f_{\text{fs-1}} f_{\text{fs-2}} ... f_0 \tag{5}$$

2.1.3 MorrisUnaryHEB(size, r)

Poate fi îmbunătățită în continuare MorrisBiasHEB(size, g, r)? De la ultimul standard de posit [11], s-a luat în considerare alegerea de a fixa mărimea exponentului pentru a face ca

acesta să depindă doar de mărime și pentru a ușura conversia între diferite mărimi. Acest lucru poate fi adaptat folosind o reprezentare unară pentru valoarea g (similar cu *regim* din posit). De asemenea, este nevoie de o valoare a mărimii exponentului de -1, astfel încât formula pentru mărimea exponentului este următoarea: -1:

$$\text{dimensiunea exponentului} = \begin{cases} -k-1, & k < 0 \\ \\ k-1, & k \geq 0. \end{cases} \tag{6}$$

unde k este regimul.

MorrisUnaryHEB(size, r) se subcoboară la 0, se supracoboară la NR, și folosește Tapered-FloatingPoint(size) pentru implementarea operațiilor sale. Reprezentarea binară începe cu bitul de semn. Următorul bit reprezintă primul bit de regim r_0. Următorii biți consecutivi cu aceeași valoare ca r_0 sunt considerați de regim biți de regim. Următorul bit după aceștia, dacă există, are valoarea negată a lui r_0 și este considerat, de asemenea, ca făcând parte din regim. Regimul k se calculează astfel:

$$k = \begin{cases} -\text{NoC0}, & r_0 = 0 \\ \\ \text{NoC1} - 1, & r_0 = 1. \end{cases} \tag{7}$$

Următorii es biți sau următorii biți rămași (oricare dintre aceștia este mai mic) reprezintă valoarea exponentului în format natural de bază 2, dacă semnul(k) este 1. În caz contrar, trebuie să fie negați, iar rezultatul este valoarea exponentului binar. În cazul în care es este mai mare decât biții rămași, biții rămași reprezintă valoarea cea mai semnificativi ai valorii exponentului binar. Cei mai puțin semnificativi biți rămași biți mai puțin semnificativi ai valorii exponentului binar sunt considerați 0. Se calculează exponentul ca:

$$\text{exponent} = \begin{cases} \text{signum(k)} \times (2^{es} + \text{exponent binar}), & es \neq -1 \\ 0, & es = -1. \end{cases} \tag{8}$$

După ce se iau biții de exponent, biții rămași sunt biți de fracție, iar biții lor numărul lor este mărimea fracției. În concluzie, formatul binar al MorrisUnaryHEB(size, r) este:

$$s r_0 r_1 \dots r_{\text{rs-2}} \overline{r_{\text{rs-1}}} e_{\text{es-1}} e_{\text{es-2}} \dots e_0 f_{\text{fs-1}} f_{\text{fs-2}} \dots f_0 \dots f_0 \tag{9}$$

2.2 Evaluare sistemelor de representare a numerelor

În această secțiune, cele trei noi SNR propuse, în plus față de NRS-urile bine cunoscute din literatura de specialitate au fost evaluate. În prima subsecțiune, prezentăm NRS-urile în curs de evaluare și caracteristicile acestora, cum ar fi valoarea absolută minimă, valoarea absolută maximă, intervalul dinamic și densitatea numerelor în logaritmii. logaritmică. În cea de-a doua

Tabela 1: NRS pe 16 biți - intervalul dinamic

NRS	*Min(abs(X))*	*Max(abs(X))/*1^{st}	Dynamic Range	2^{nd}	3^{rd}
FixedFloatingPoint(5, 10, RE)	3.054×10^{-4}	130944	9.6322	130880	130816
FixedPoint(8, 8, RE)	0.003	127.996	4.515	127.992	127.988
half-IEEE754/IEEE754(5, 10, RE)	5.960×10^{-8}	65504	12.040	65472	65440
Posit(16, 2, RE)	1.387×10^{-17}	72.057×10^{15}	33.715	45.035×10^{14}	11.258×10^{14}
Morris(16, 4, RZ)	9.207×10^{-19710}	1.086×10^{19709}	39418.071	5.887×10^{19689}	3.191×10^{19670}
MorrisHEB(16, 4, RZ)	4.630×10^{-9860}	2.159×10^{9859}	19718.668	3.295×10^{9854}	5.028×10^{9849}
MorrisBiasHEB(16, 4, RE)	6.061×10^{-39}	1.121×10^{77}	115.267	1.085×10^{77}	1.049×10^{77}
MorrisUnaryHEB(16, RE)	9.168×10^{-2467}	1.090×10^{2466}	4932.075	1.044×10^{1233}	5.809×10^{924}

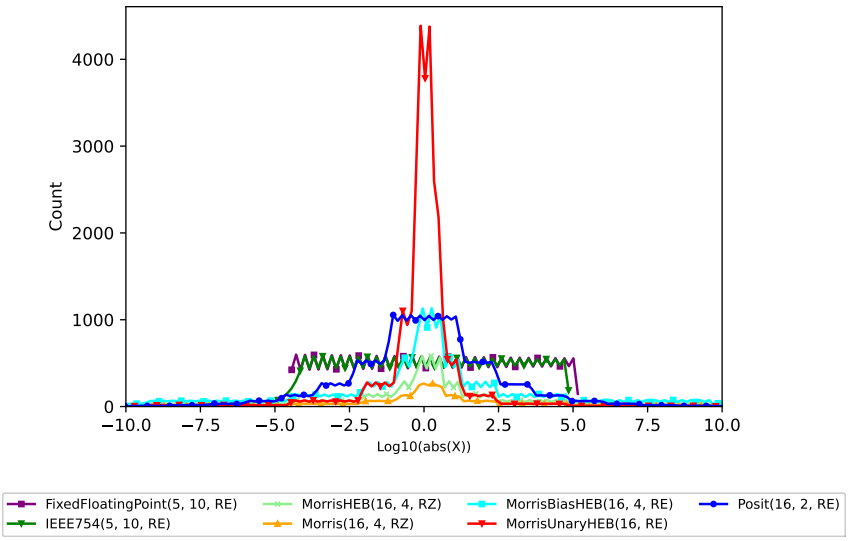


Figura 1: Distribuția valorilor absolute unice

subsecțiune se prezintă precizia zecimală a sistemelor unare operații pentru NRS-urile testate cu ajutorul graficelor CDF. În cea de-a treia subsecțiune, se prezintă sunt prezentate hărțile color ale operațiilor binare. În ultima subsecțiune se trece în revistă câteva repere celebre din literatura de specialitate. Următoarele notații sunt utilizate pentru rotunjire în această secțiune: *RZ* pentru rotunjirea spre zero și *RE* pentru rotunjire. către cea mai apropiată egalitate de par. Valorile prezentate în această secțiune sunt de obicei trunchiate la trei zecimale după virgulă.

2.2.1 NRS-uri în curs de evaluare și caracteristicile acestora

Tabelul 1 prezintă SRI în curs de evaluare cu valoarea lor minimă absolută, valoarea maximă absolută și intervalul dinamic atunci când dimensiunea totală este de 16 biți. Cele trei noi NRS-uri bazate pe Morris tapered cu un bit de exponent ascuns sunt comparate cu reprezentarea Morris implicită, fixă punct, virgulă mobilă fixă, virgulă mobilă fixă, IEEE754 și posit.

NRS-urile cu virgulă mobilă conică au o gamă dinamică mai mare și pot reprezenta valori mai mari de și valori absolute mai mici în comparație cu IEEE754 și cu virgulă fixă. Pe de altă parte parte, diferența dintre valorile consecutive poate fi de un ordin de mărime. Figura 1

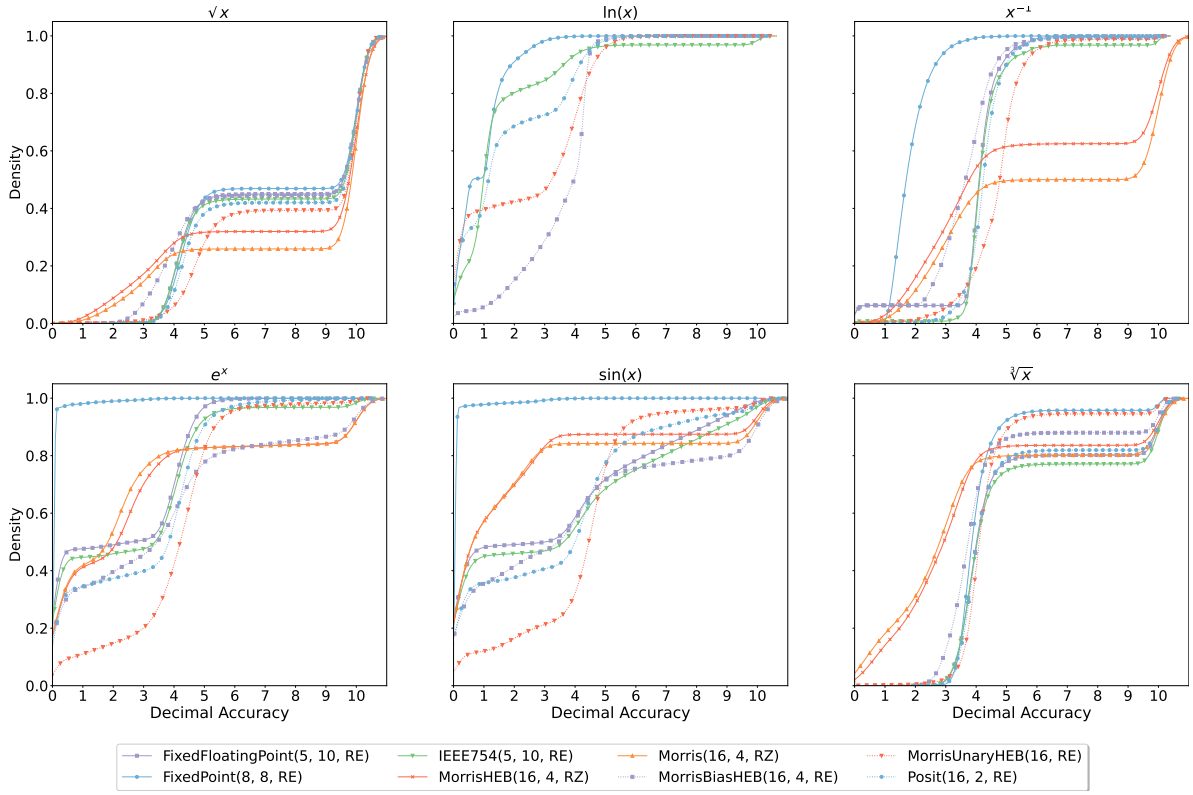


Figura 2: CDF de operații unitare

prezintă numărul de valori absolute unice pentru NRS pe 16 biți pe o scară logaritmică. Valoarea adăugată a bitului ascuns al exponentului poate fi observată în numărul crescut de numere pentru NRS-urile derivate din Morris. Un rezultat interesant este MorrisUnaryHEB(16, RE). "zona de aur": are 30.201 valori absolute unice în intervalul $(10^{-3}, 10^{3})$ față de 26.587 pentru Posit(16, 2, RE). Acest lucru, împreună cu intervalul dinamic mai mare, îl face un bun concurent pentru postul în rețelele neuronale profunde. Capitolul 5 evaluează acest lucru.

Diferența dintre regulile de underflow și overflow ale IEEE754(es, fs, r) și FixedFloatingPoint(es, fs, r) poate fi observată în cazul în care se observă scăderea treptată a debitului pentru IEEE754(es, fs, r) și valorile suplimentare mai mari pentru FixedFloatingPoint(es, fs, r). Utilizarea unei valori de regim pozitive pentru zero poate fi observată în inegalitatea distribuție inegală a valorilor MorrisUnaryHEB(16, RE) și Posit(16, 2, RE) în Figura 1.

2.2.2 Operații unitare

Figura 2 prezintă CDF a preciziei zecimale pentru operațiile cu rădăcină pătrată, logaritm natural, inversă, exponențială, sinusală și rădăcină cubică. Axa x reprezintă numărul de cifre exacte din rezultat. Pentru toate funcțiile seriei Taylor $(\ln(x), \sin(x), e^x)$, referința de precizie zecimală este rezultatul RationalNumber după 30 de iterații. Pentru o precizie zecimală de cel puțin trei cifre, MorrisUnaryHEB(16, RE) este cel mai bun NRS. Acest lucru se datorează valorilor sale absolute unice. Rețineți că exponențialul este singura funcție care crește

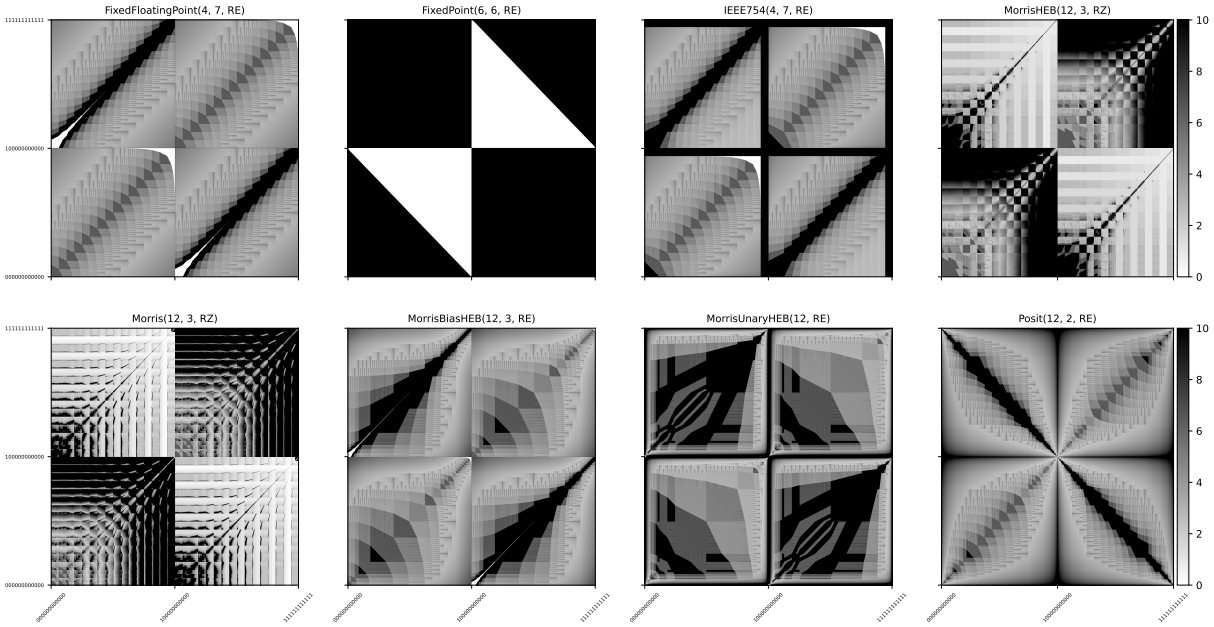


Figura 3: Hărți în culori pentru adiție

magnitudinea rezultatului.

2.2.3 Operații binare

Pentru operațiile binare, au fost alese NRS-uri pe 12 biți, deoarece 8 biți conțin prea puține informații, iar 16 biți ocupă prea mult spațiu de stocare pentru a păstra toate valorile. Rezultatele sunt prezentate sub formă de hărți colorate, unde negrul reprezintă o precizie de 10 sau mai multe cifre, în timp ce albul reprezintă zero sau mai puțin.

Harta de culori pentru adunare este prezentată în figura 3. scădere este similară cu adunarea. Acest grafic arată foarte bine de ce FixedPoint(is, es, r) este NRS-ul perfect pentru acumulatori dacă intervalul de rezultatelor este cunoscută. Spațiul de culoare albă reprezintă zona de depășire pentru valorile pozitive și negative. Asemănările dintre FixedFloatingPoint(es, fs, r) și IEEE754(es, fs, r) sunt evidente, dar se poate observa, de asemenea, efectul subcontinuare treptată în IEEE754(es, fs, r). Marginea neagră și liniile plus pentru IEEE754(es, fs, r) reprezintă NaN-uri (NaN plus orice altceva are ca rezultat un NaN).

Hărțile pentru Morris(size, g, r) și MorrisHEB(size, g, r) sunt diferite de celelalte hărți, deoarece reprezentările binare nu reprezintă reprezentări ordonate. valori ordonate. MorrisBiasHEB(size, g, r) arată ca un amestec între FixedFloatingPoint(es, fs, r) și Posit(size, es, r): prezintă caracteristici de virgulă mobilă conică prin având o relație invers proporțională între precizie și valoarea absolută valori. Adică, atunci când valorile absolute ale operanzilor cresc, valorile zecimale cresc. scade precizia zecimală. Posit(size, es, r) are o distribuție mai uniformă a preciziei. Rețineți că Posit(size, es, r) nu utilizează magnitudinea semnului, ci utilizează complementul lui 2 pentru numerele negative, astfel încât simetria hărții sale este diferită de celelalte hărți.

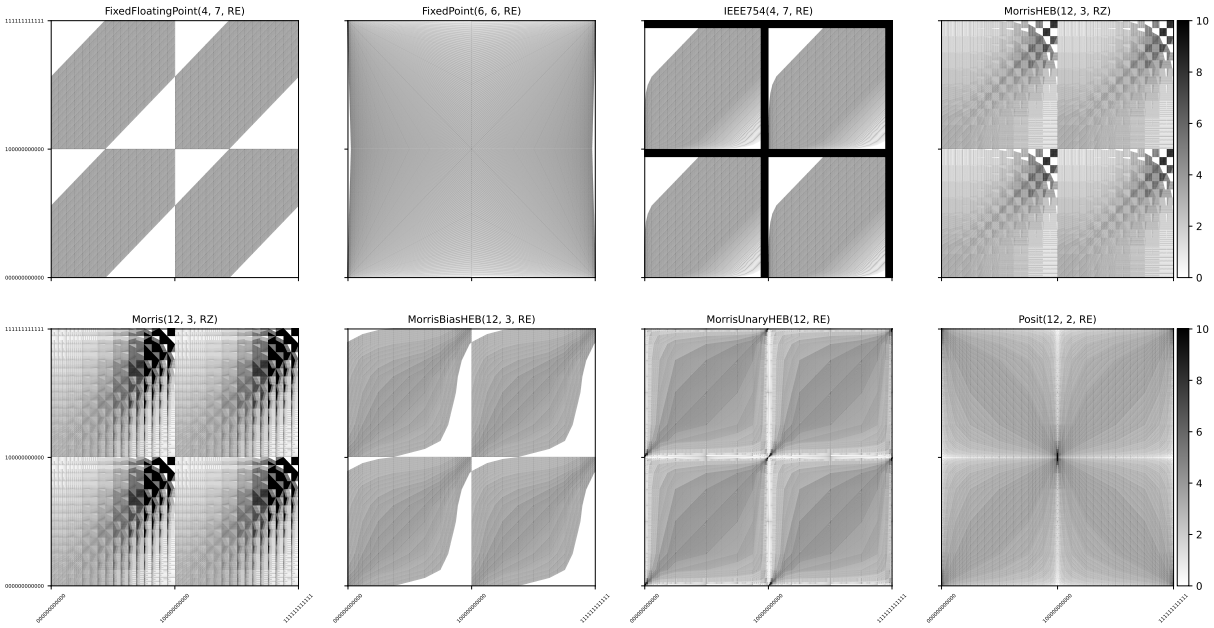


Figura 4: Hărți în culori pentru diviziune

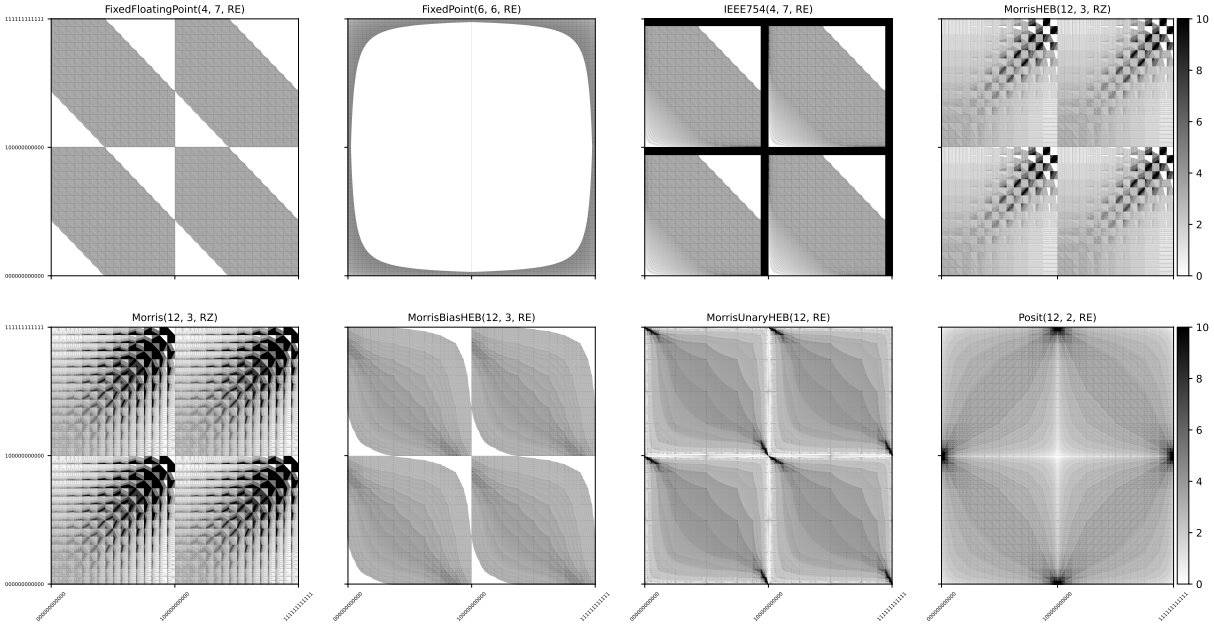


Figura 5: Hărți colorate pentru înmulțire

Tabela 2: Rezultatele operațiilor binare (ADD,DIV,MUL)

NRS	Exact			Average Accuracy			Kops		
	ADD	DIV	MUL	ADD	DIV	MUL	ADD	DIV	MUL
FixedFloatingPoint(4, 7, RE)	16.4%	2.4%	2.2%	3.3	2.4	2.4	191	374	278
FixedPoint(6, 6, RE)	75.0%	0.9%	0.9%	0.0	2.8	0.5	6535	2551	4117
IEEE754(4, 7, RE) (12.1% NaNs)	28.6%	14.3%	14.4%	3.2	2.7	2.7	362	370	326
Posit(12, 2, RE)	12.4%	4.2%	4.2%	2.8	4.0	2.8	150	206	253
Morris(12, 3, RZ)	20.9%	22.1%	26.4%	4.9	1.5	1.5	145	256	347
MorrisHEB(12, 3, RZ)	14.2%	8.9%	8.8%	5.4	1.9	1.8	185	301	385
MorrisBiasHEB(12, 3, RE)	20.2%	2.2%	2.2%	3.4	2.7	2.9	148	221	261
MorrisUnaryHEB(12, RE)	37.6%	1.9%	1.9%	4.2	3.0	3.0	142	219	263

Rezultatele privind precizia zecimală pentru înmulțire sunt prezentate în Figura 5. Problema de depășire a FixedPoint(is, es, r) este următoarea evidentă, în timp ce scăderea treptată a debitului ajută IEEE754(es, fs, r). Marginile negre ale IEEE754(es, fs, r) provin de la valorile NaN. MorrisUnaryHEB(size, r), Posit(size, es, r) și MorrisBiasHEB(size, g, r). prezintă proprietățile lor în virgulă mobilă conică în valuri (sau benzi) de precizie. Comparând MorrisUnaryHEB(size, r) și Posit(size, es, r), regula pentru underflow poate fi observată ca o bandă albă în harta colorată a MorrisUnaryHEB(size, r). Aceste rezultate sugerează că regula Posit(size, es, r) pentru underflow ar putea fi să fie cea mai bună pentru a fi implementată într-un NRS.

Precizia pentru diviziune este prezentată în Figura 4. efect al subcontracției treptate în IEEE754(es, fs, r) poate fi observat în dreapta-jos a cadranelor, spre deosebire de FixedFloatingPoint(es, fs, r). Negrul negre din IEEE754(es, fs, r) provin de la valorile NaN. Posit(size, es, r) și FixedPoint(is, es, r) prezintă cele mai uniforme modele. MorrisUnaryHEB(size, r),Posit(size, es, r), și MorrisBiasHEB(size, g, r) își prezintă proprietățile lor în virgulă mobilă conică în valuri (sau benzi) de precizie. Comparând MorrisUnaryHEB(size, r) și Posit(size, es, r), regula pentru underflow poate fi observată ca o bandă albă în harta de culori a MorrisUnaryHEB(size, r). Aceste rezultate sugerează că regula Posit(size, es, r) pentru underflow ar putea fi să fie cea mai bună pentru a fi implementată într-un NRS. Rezultatele privind precizia zecimală pentru înmulțire sunt similare cu cele pentru împărțire și sunt omise din motive de spațiu. de spațiu.

Tabelul 2 prezintă procentul de rezultate exacte, media precizia zecimală medie a rezultatelor inexacte, precum și numărul de operații (mii) pe secundă (Kops). Din IEEE754(4, 7, RE), ar trebui să se elimine $12,1\%$ din rezultate, deoarece acestea reprezintă NaN-uri. Rezultatele interesante din tabelul 2 sunt: (i) număr mare de rezultate exacte pentru MorrisUnaryHEB(12, RE), (ii) numărul relativ bun de rezultate precizie zecimală medie a rezultatelor inexacte pentru MorrisUnaryHEB(12, RE) și MorrisBiasHEB(12, 3, RE) pentru toate operațiile, și (iii) valoarea relativ scăzută a procent de rezultate exacte pentru Posit(12, 2, RE). (acest lucru se datorează faptului că mărirea mărimii exponentului).

Tabela 3: Benchmarks in [10]

NRS	John Wallis	Kahan u_{30}	Jean Micheal Muller	Siegfried Rump	r_1 DA	David Bailey
32-bit NRSs						
FixedFloatingPoint(8, 23, RE)	3.091	100	$(0, 0, 0, 0)$	-63.382×10^{28}	5.612	(NR, NR)
FixedPoint(16, 16, RE)	3.091	100	$(1, 1, 1, NR)$	NR	3.787	(NR, NR)
IEEE754(8, 23, RE)	3.091	100	$(0, 0, 0, 0)$	-1.901×10^{30}	5.612	(NR, NR)
Posit(32, 2, RE)	3.091	100	$(0, 0, 0, 0)$	1.172	5.996	$(-4, 2)$
Morris(32, 4, RZ)	3.091	99.999	$(0, 0, 0, 0.995)$	20.282×10^{30}	4.599	$(0, 1)$
MorrisHEB(32, 4, RZ)	3.091	99.999	$(0, 0, 0, 0.989)$	15.211×10^{30}	4.945	$(2, 1)$
MorrisBiasHEB(32, 4, RE)	3.091	100	$(0, 0, 0, 0)$	-25.353×10^{29}	5.612	$(1, 0.5)$
MorrisUnaryHEB(32, RE)	3.091	100	$(0, 0, 0, 0.999)$	1.172	5.612	$(2, 0)$
64-bit NRSs						
FixedFloatingPoint(11, 52, RE)	3.091	99.999	$(0, 0, 0, 0)$	11.805×10^{20}	14.258	$(0, 1.333)$
FixedPoint(32, 32, RE)	3.091	100	$(1, 1, 1, 1)$	NR	8.570	(NR, NR)
IEEE754(11, 52, RE)	3.091	99.999	$(0, 0, 0, 0)$	11.805×10^{20}	14.258	$(0, 1.333)$
Morris(64, 5, RZ)	3.091	99.999	$(0, 0, 0, 0)$	47.223×10^{20}	14.133	$(-1.142, 2.071)$
MorrisHEB(64, 5, RZ)	3.091	99.999	$(0, 0, 0, 0)$	1.172	15.032	$(-1, 2)$
MorrisBiasHEB(64, 5, RE)	3.091	99.999	$(0, 0, 0, 0)$	11.802×10^{20}	15.030	$(-1.017, 2.035)$
MorrisUnaryHEB(64, RE)	3.091	99.999	$(0, 0, 0, 0)$	37.778×10^{21}	15.031	$(-1.004, 1.997)$
Posit(64, 2, RE)	3.091	100	$(0, 0, 51.669 \times 10^{14}, 84.750 \times 10^{8})$	1.172	15.891	$(-1.013, 2)$
RationalNumber	3.091	6.004	$(1, 1, 1, 1)$	-0.827	1	$(-1, 2)$

2.2.4 Repere literare

În tabelul 3, rezultatele evaluărilor propuse de Gustafson în [10] sunt rezumate. Evaluările propuse sunt următoarele:

- Produsul John Wallis: $2 \times \prod_{i=1}^{n} \frac{(2\times i)^2}{(2\times i-1)\times(2\times i+1)}$ pentru $n = 30$,
- Seria Kahan: $u_{i+2} = 111 - \frac{1130}{u_{i+1}} + \frac{3000}{u_i \times u_{i+1}}$ pentru u_{30},
- Jean Micheal Muller: $E(0) = 1, E(z) = \frac{e^z-1}{z}, Q(x) = |x-\sqrt{x^2+1}| - \frac{1}{x+\sqrt{x^2+1}}, H(x) = E((Q(x))^2)$ pentru $H(15), H(16), H(17), H(9999)$,
- Siegfried Rump: 333,74 $\times y^6 + x^2 \times (11 \times x^2 \times y^2 - y^6 - 121 \times y^4 - 2) + 5,5 \times y^8 + \frac{x}{2\times y}$ pentru $x = 77517$ și $y = 33096$,
- Precizia zecimală pentru r_1 din formula pătratică pentru $a = 3, b = 100, c = 2$,
- Sistemul de ecuații al lui David Bailey: $0.25510582 \times x + 0.52746197 \times y = 0.79981812, 0.80143857 \times x + 1.65707065 \times y = 2.51270273$ rezolvat cu regula lui Cramer.

Observați că niciunul dintre SNR nu trece toate evaluările. Problema este legată de limitările reprezentărilor finite.

În tabelul 4, rezultatele mai multor teste de referință de la literatura de specialitate [6, 9, 11] sunt prezentate. Aceste criterii de referință sunt:

- aria triunghiului subțire pentru $a = 7, c = b = \frac{7+2^{-25}}{2}$,
- formula $x = (\frac{27/10-e}{\pi-(\sqrt{2}+\sqrt{3})})^{67/16}$,
- fracția $\frac{x^n}{n!}$ pentru $x = 7, n = 20$ și $x = 25, n = 30$,
- constanta lui Planck $h = 6,626070150 \times 10^{-34}$,
- numărul Avogadro $L = 6.02214076 \times 10^{23}$,
- viteza luminii $c = 299792458$,

Tabela 4: Alte repere literare [6, 9, 11]

NRS	Thin Triangle	x	$\frac{x^n}{n!}$	Planck	L	c	$\overline{e}$	k
32-bit NRSs								
FixedPoint(16, 16, RE)	0	2.289	(0, 0)	0	0	0	0	0
IEEE754(8, 23, RE)	0	4.370	(7.135, 0)	8.727	8.075	7.839	8.004	7.782
Posit(32, 2, RE)	1.204	5.684	(4.339, 0)	0.627	4.091	6.969	4.213	4.037
Morris(32, 4, RZ)	0	5.101	(6.016, 5.604)	6.347	6.429	6.969	7.347	6.480
MorrisHEB(32, 4, RZ)	0	5.098	(6.245, 6.188)	6.680	6.784	7.839	7.347	6.480
MorrisBiasHEB(32, 4, RE)	0	5.682	(6.911, 8.619)	7.053	7.219	7.839	7.347	6.878
MorrisUnaryHEB(32, RE)	0	6.875	(6.017, 5.289)	6.031	5.919	6.969	7.347	5.566
64-bit NRSs								
FixedPoint(32, 32, RE)	8.343	7.976	(0, 0)	0	0	∞	0	0
IEEE754(11, 52, RE)	16.710	13.030	(16.644, 15.905)	16.952	17.028	∞	16.540	16.537
Morris(64, 5, RZ)	16.710	14.221	(16.644, 14.875)	16.172	15.810	∞	16.026	15.467
MorrisHEB(64, 5, RZ)	16.710	14.710	(16.950, 15.794)	16.172	16.238	∞	16.540	15.807
MorrisBiasHEB(64, 5, RE)	17.341	14.831	(17.263, 16.012)	16.952	17.028	∞	16.540	16.197
MorrisUnaryHEB(64, RE)	17.341	15.323	(15.743, 14.942)	16.172	15.458	∞	16.026	15.467
Posit(64, 2, RE)	17.835	15.672	(14.018, 8.372)	11.094	13.200	∞	14.263	12.974

- sarcina de $\overline{e} 1.602176634 \times 10^{-19}$,
- constanta Boltzmann $k = 1.380649 \times 10^{-23}$.

Valorile din tabelul 4 reprezintă precizia zecimală a rezultatelor în comparație cu rezultatul corect. Primele două criterii de referință sunt favorabile pentru Posit(size, es, r), în timp ce ultimele șase sunt favorabile pentru IEEE754(es, fs, r). Morris și NRS-urile sale derivate prezintă rezultate apropiate de cea mai bună NRS pentru fiecare referință. MorrisBiasHEB(size, g, r) are rezultate bune pentru întregul spectru de repere.

2.3 Perspective asupra sistemelor de reprezentare a numerelor

În acest capitol, (i) este prezentată o bibliotecă Scala care simplifică adăugarea, testarea și reglarea fină a sistemelor de reprezentare a numerelor (NRS), (ii) sunt introduse trei noi NRS-uri, bazate pe virgulă mobilă conică Morris, și (iii) aceste trei NRS-uri propuse sunt analizate împreună cu NRS-uri bine cunoscute, cum ar fi virgulă mobilă IEEE754 și pozit. Prin adăugarea bitului de exponent ascuns la virgulă mobilă conică Morris în trei forme diferite, NRS-urile rezultate au devenit concurenți pentru IEEE754 și posit. MorrisBiasHEB(size, g, r) prezintă cele mai bune rezultate în cazul reperelor din literatură pe 32 și 64 de biți, în comparație cu celelalte NRS-uri. Pe de altă parte, MorrisUnaryHEB(size, r) este un candidat excelent pentru calculele de învățare automată datorită datorită populației sale din "zona de aur", a domeniului dinamic, a percentilei de rezultate exacte pe adunare și precizia zecimală medie a rezultatelor inexacte la înmulțire. Biblioteca prezintă o performanță de 200 Kops, care este suficient de bună pentru testarea și evaluarea SNR-urilor, dar nu suficient pentru aplicațiile din lumea reală. În lucrările viitoare, biblioteca va fi integrată cu sistemul Aparapi bibliotecă[1] și testată pe GPU și utilizată pentru mașini modele de învățare cu Spark.

[1] https://aparapi.com/

3 NRS ÎN DOMENIUL CALCULULUI ȘTIINȚIFIC

3.1 Rezultatele Benchmark-urilor

Pentru criteriile de referință de mai jos, au fost utilizate următoarele unsprezece SNR: IEEE754 pe 32 de biți (8 biți exponent și 23 de biți mantissa) numit IEEE754, Posit pe 32 de biți (2 biți exponent) numit Posit, IEEE754 pe 16 biți (5 biți exponent și 10 biți mantissa) numit half-IEEEEE754, IEEE754 pe 16 biți (8 biți exponent și 7 biți mantissa) numit bfloat16, IEEE754 pe 19 biți (8 biți exponent și 10 biți mantissa) numit TF32, IEEE754 pe 24 de biți (7 biți exponent și 16 biți mantissa) numit FP24, IEEE754 pe 24 de biți (8 biți exponent și 15 biți mantissa) numit PXR24, virgulă mobilă pe 32 de biți (8 biți exponent și 23 de biți mantissa) numit FloatP, virgulă fixă pe 32 de biți (16 biți întregi și 16 biți fracționari) numit FixedP, virgulă fixă de precizie infinită numit IP_NR_FixedP (FixedPoint(∞)), virgulă mobilă de precizie infinită numit IP_NR_FloatP (FloatingPoint(∞)). Metoda de rotunjire utilizată a fost rotunjirea la cea mai apropiată egalitate de par.

3.1.1 Reper de referință pentru metoda gradientului conjugat

Pentru benchmark-ul metodei gradientului conjugat, NRS de referință este FractionalNumber(512, 512, RE) pe 1024 de biți. (numitor întreg de 512 biți și numitor întreg fără semn de 512 biți), $N = 5$, MAX POWER $= 3$. Deoarece la fiecare iterație a algoritmului se execută mai multe înmulțiri de matrice, ordinea de mărime a valorilor utilizate crește rapid în timpul execuției. Acesta este motivul pentru care nu au fost generate numere de ordin de

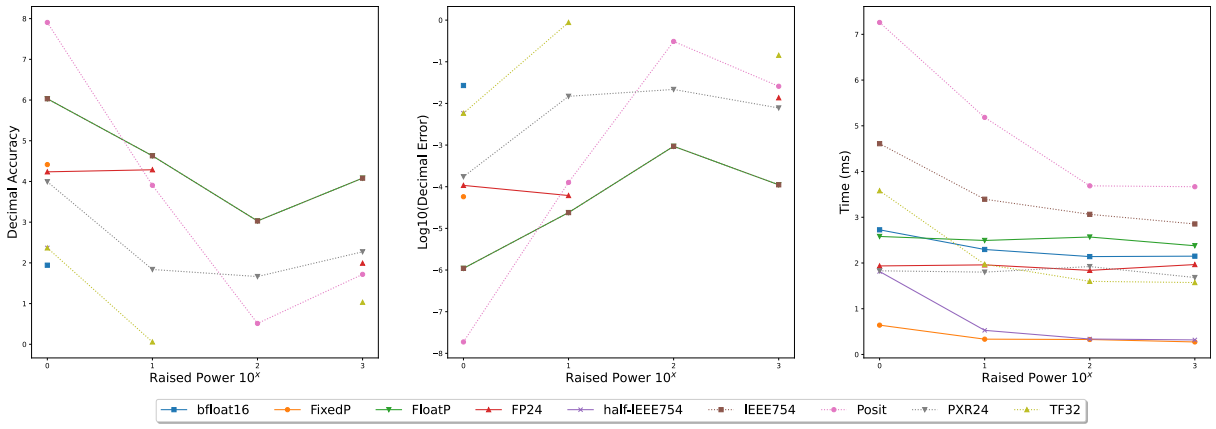


Figura 6: Precizia zecimală, eroarea zecimală și timpul de calcul pentru metoda de referință a gradientului conjugat

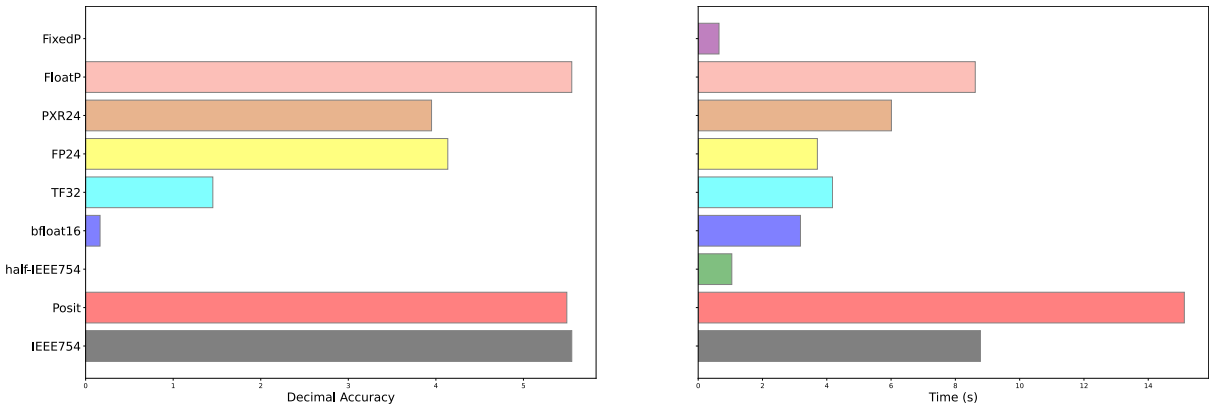


Figura 7: Precizia zecimală și timpul de calcul pentru testul de referință al integrării lui Simpson

mărime mare ca date de intrare. Din nou, avantajul Posit pe numere mici poate fi observat, dar dezavantajul său cu numere mari iese în evidență și mai mult în timpul acestui test. Se poate observa că își pierde jumătate din precizie în timpul testului, ceea ce îl face chiar mai puțin precis decât PXR24, care utilizează cu 8 biți mai puțin decât Posit. TF32 prezintă un comportament care este mai greu de predeterminat. După mai multe iterații ale algoritmilor, după cum se poate observa în figura 6, precizia sa nu pare să țină cont de ordinea valorilor de intrare. Ea urmează un model neregulat care crește și scade cu multe zecimale exacte în general, deși pentru câteva rulări a păstrat o diferență constantă de o zecimală. În ceea ce privește timpul, rezultatele au fost destul de previzibile. IEEE754 și Posit necesită cel mai mult timp, dar produc și cele mai exacte rezultate. Indiferent de numărul de biți utilizați și de precizia obținută, celelalte NRS-uri au avut aproximativ același interval de timp de funcționare.

3.1.2 Criteriul de referință al integrării Simpson

Pentru benchmark-ul de integrare Simpson, NRS de referință este NRS pe 1024 de biți FractionalNumber(512, 512, RE). (numitor întreg de 512 biți și numitor întreg fără semn de 512 biți) , $N = 1000$, $[a, b]$ is $[0, 10]$, and the functions are: x^2, x^3, $\sqrt{x}$, $\sqrt[3]{x}$, $\sqrt[3]{\frac{x^2+\sqrt{x}}{x+7}}$, $(\sqrt[5]{x^2+34} \times x^3)^{\frac{3}{4}}$. Fiind un algoritm care nu efectuează atât de multe calcule complexe, acuratețea nu diferă prea mult între reprezentările cu același număr de biți (figura 7, de exemplu Posit, IEEE754, FloatP). În acest test, timpul de calcul devine relevant, deoarece se poate observa că, deși cele 3 SNR care utilizează 32 de biți au obținut aceeași precizie, diferența de timp dezavantajează puternic Posit, care rulează de două ori mai mult decât IEEE754.

3.1.3 Reper de referință pentru simularea N-Body

Pentru testul de referință pentru simularea cu N-Body, NRS de referință este FractionalNumber(512, 512, RE) pe 1024 de biți. (numitor întreg de 512 biți și numitor întreg fără semn de

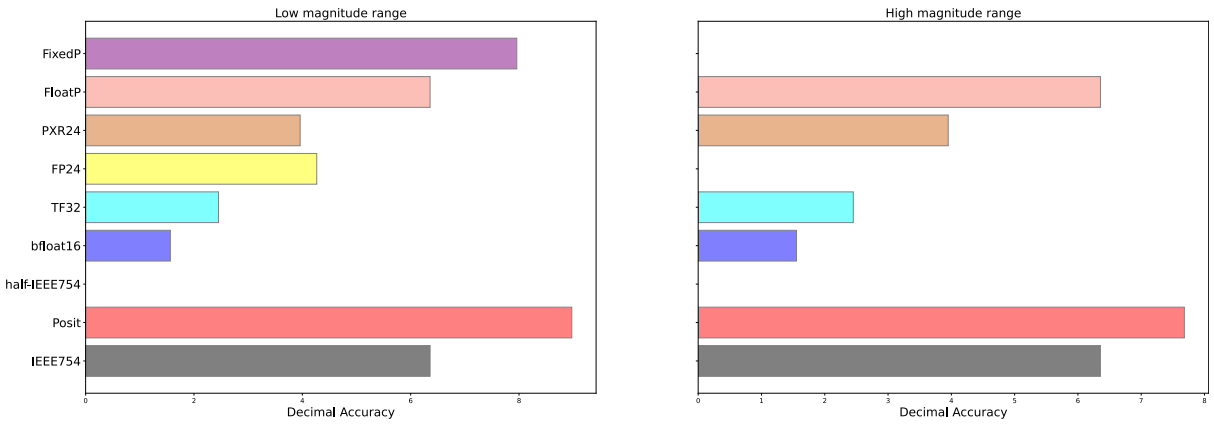


Figura 8: Precizia zecimală pentru benchmark-ul de simulare cu N-Body

512 biți), $N = 5$, $M = 10$, pasul de timp de iterație $= 1$(secundă), $G = 6,67498 \times 10^{-11}$. Pozițiile, vitezele și masa au fost separate în două intervale de ordin de mărime (mică și mare). Pentru intervalul de ordin de mărime mare, pozițiile x și y ale fiecărei particule sunt generate în intervalul $[-10^{11}, 10^{11}]$, vitezele x și y ale fiecărei particule sunt generate în intervalul $[30, 5 \times 10^6]$, iar masa fiecărei particule este generată în intervalul $[1, 6 \times 10^5]$. Pentru intervalul de ordine de mărime mică, pozițiile x și y ale fiecărei particule sunt generate în intervalul $[0, 10^4]$, vitezele x și y ale fiecărei particule sunt generate în intervalul $[30, 50]$, iar masa fiecărei particule este generată în intervalul $[10, 60]$. Figura 8 sugerează superioritatea lui Posit prin obținerea a aproape trei zecimale exacte în plus față de standardul actual IEEE754 și a unei zecimale exacte în plus față de al doilea cel mai precis NRS FixedP. Cu toate acestea, performanța remarcabilă a Posit și FixedP poate fi ușor explicată prin intervalele din care au fost alese datele de intrare pentru fiecare particulă. Au fost alese numere mici pentru coordonatele particulelor pentru ca acestea să fie suficient de apropiate una de cealaltă astfel încât forța de atracție dintre ele să fie vizibilă. Utilizarea valorilor de ordin de mărime mare (chiar și în afara zonei de aur) a făcut ca FixedP să returneze NR.

3.2 Perspective privind rezultatele calculului științific

Acest capitol a propus un punct de referință pentru algoritmi de calcul științific de bază în cadrul diferitelor NRS. Criteriul de referință a utilizat patru algoritmi: înmulțirea matricelor, rezolvarea unui sistem liniar de ecuații prin metoda gradientului conjugat, integrarea prin formula lui Simpson și simularea cu N-corp. Datele de intrare ale algoritmului au fost variate, astfel încât performanța tuturor celor unsprezece SRI să poată fi analizată în circumstanțe diferite. Rezultatele simulărilor oferă o perspectivă asupra compromisului dintre diferitele NRS-uri. Zona de aur a Posit [6] a fost validată prin toate experimentele. Posibilitatea de a utiliza semiprecizia IEEE754 în anumite cazuri specifice, în care intervalul de numere nu depășește domeniul său dinamic, este un rezultat interesant. Acesta poate fi considerat o soluție NRS mai rapidă și mai eficientă din punct de vedere energetic pentru o aplicație de calcul științific. În cazul în care dezvoltatorii doresc o soluție mai rapidă, NRS în virgulă fixă

poate fi opțiunea lor, având în vedere cerințele oferite în acest capitol. Înlocuirea standardului actual cu un nou NRS poate fi necesară, având în vedere rezultatele prezentate. Cu toate acestea, lipsa implementării hardware și a adoptării de către industrie va întârzia acest proces.

4 UTILIZAREA NRS PENTRU METODE STATISTICE

4.1 Evaluarea metodelor statistice

Conform figurii 9, se poate observa că acuratețea obținută de Posit este mai bună decât cea a IEEE754 pe același număr de biți. De asemenea, obținerea de rezultate valide pe mai puțini biți, așa cum s-a observat pentru Posit în unele cazuri, reprezintă un câștig atât în ceea ce privește memoria, cât și viteza de execuție. Conform rezultatelor obținute la punctul 4.1, unde a fost analizată performanța IEEE754 versus Posit pe operații matematice elementare, se poate considera că diferența de precizie între Posit, IEEE754 și rezultatul real era de așteptat. Acest lucru se datorează faptului că Posit oferă o precizie mult mai bună decât IEEE754 pe un număr mai mic de biți atunci când se discută despre adunare și scădere. Totuși, acuratețea sa scade atunci când sunt implicate și calcule de variantă sau de radical, în special pe seturi de date mai mari. Se poate observa că precizia obținută pe diferite sisteme de reprezentare

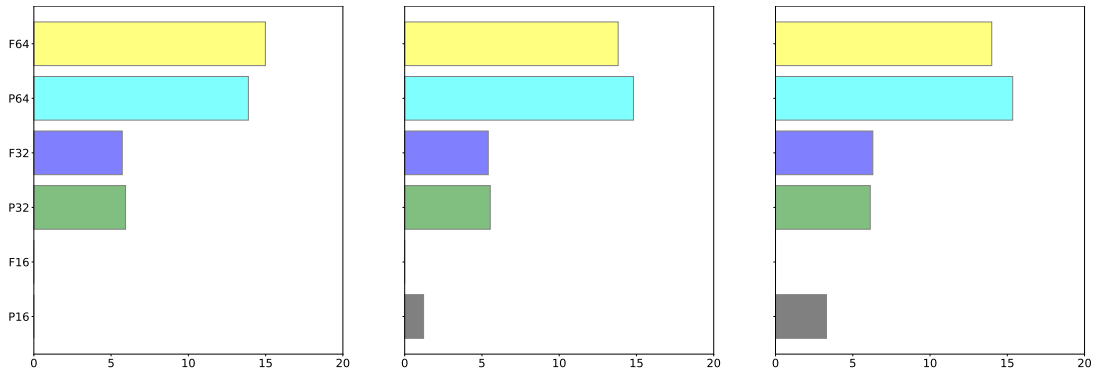


Figura 9: Precizia zecimală pentru coeficientul Pearson

a numerelor este apropiată atât pentru un set de date, cât și pentru două seturi de date. Deși Posit este mai aproape de valoarea reală a rezultatului la orice număr de biți, diferențele nu sunt extrem de mari. Cu toate acestea, el furnizează totuși mai multe cifre corecte decât omologul său. Testul Kolmogorov-Smirnov conține doar operații matematice de bază (adunare, scădere, înmulțire și împărțire). Astfel, în medie, pentru seturile de date cu valori mai mari decât 1, acuratețea celor două tipuri de date este adesea similară, iar acest lucru este valabil și în acest exemplu. Trebuie remarcat faptul că pe 16 biți, biții utilizați de IEEE754 nu erau suficienți pentru a reprezenta rezultatul. Conform figurii 11, posit are mai multe cifre zecimale identice cu rezultatul real în comparație cu IEEE754. Astfel, precizia sistemului de reprezentare a numerelor Posit oferă în continuare câteva zecimale semnificative și, în general, un rezultat mai apropiat de valoarea reală. Testul Shapiro-Wilk conține, pe lângă operațiile matematice de bază (adunare, scădere, înmulțire și împărțire), operații de variație și ridicare

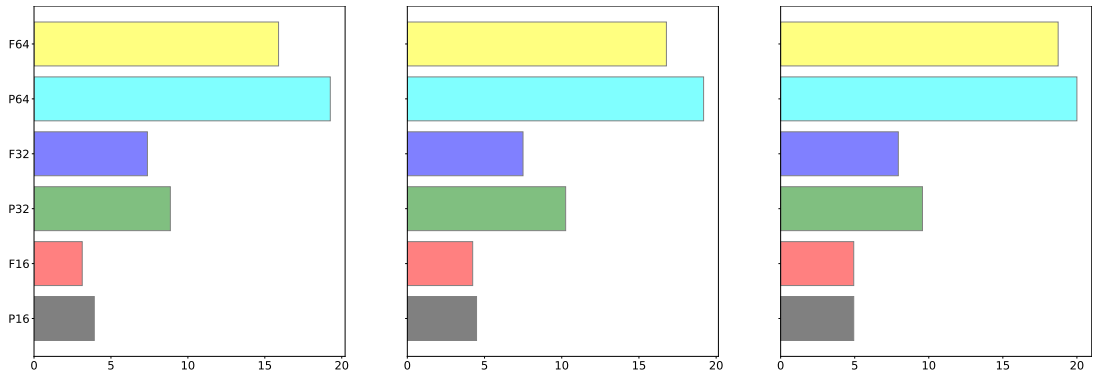


Figura 10: Precizia zecimală pentru testul Kolmogorov-Smirnov pentru un set de date

la puteri. Deși Posit prezintă o problemă de acuratețe în calculul variației în comparație cu IEEE754, multitudinea de operații din formulă, plus numărul de biți pe care este reprezentat numărul, a făcut ca sistemul de reprezentare a numerelor Posit să aibă o acuratețe mai bună decât concurentul său încă. Pe măsură ce setul de date crește, rezultatele operațiilor devin mai mari în valoare absolută, iar cei 16 biți (5 biți de exponent și 10 biți de fracție) nu mai sunt suficienți pentru a produce un rezultat valid pentru IEEE754. În același timp, Posit pe 16 biți continuă să furnizeze rezultate relevante cu o precizie rezonabilă. Operațiile matematice

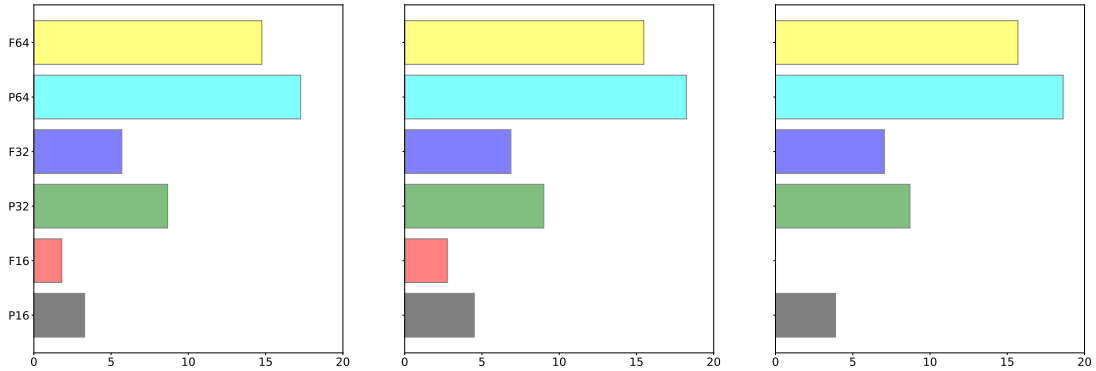


Figura 11: Precizia zecimală pentru testul Shapiro-Wilk

pentru determinarea statisticii ANOVA implică, pe lângă aritmetica de bază, operații pentru variață. Conform rezultatelor de la punctul 4.3, Posit oferă rezultate mai puțin precise în ceea ce privește calculul variației. Cu toate acestea, toată această acumulare de operații a dus la o precizie similară între Posit și IEEE754, diferența fiind numărul de zecimale afișate, unde Posit câștigă. Trebuie remarcat faptul că acuratețea lui Posit este în continuare mai bună decât cea a lui IEEE754 atât pentru ANOVA unidirecțională, cât și pentru ANOVA bidirecțională. Conform figurii 13, precizia Posit este mai bună decât cea a IEEE754, are un număr mai mare de zecimale, iar numărul de zecimale exacte este mai mare decât IEEE754. Testul Kruskall-Wallis este un test de rang, ceea ce înseamnă că are valori de la 1 la N - numărul de valori din setul de date. Prin urmare, pentru seturi de date mici și medii, problema reprezentării nu apare nici măcar pe 16 biți. În plus, faptul că formula este compusă din operații aritmetice

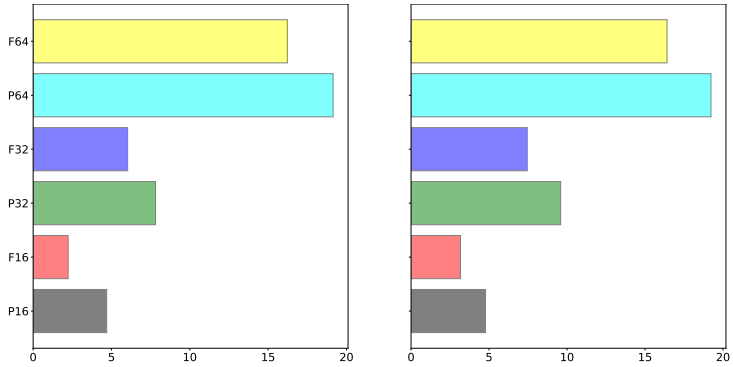


Figura 12: Precizia zecimală pentru ANOVA în două direcții

de bază contribuie la ideea anterioară. În cadrul Figura 14), se poate observa că Posit are o

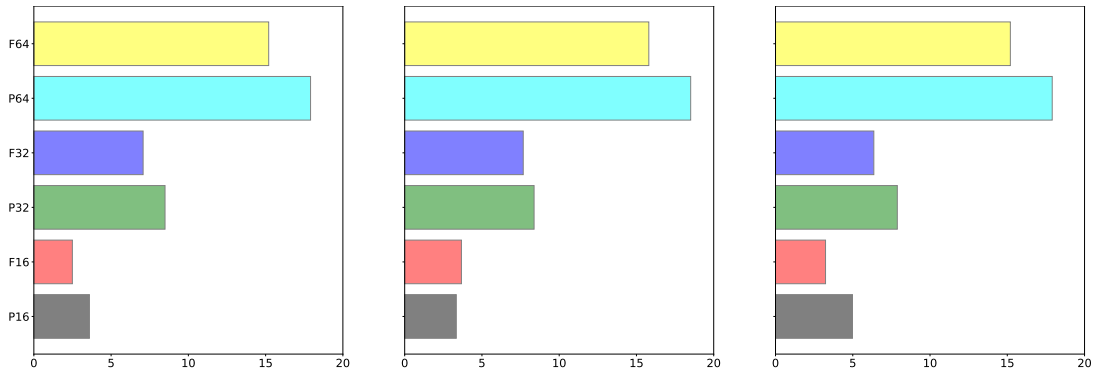


Figura 13: Precizia zecimală pentru testul Kruskall-Wallis

acuratețe ușor mai bună la toate testele și prezintă mai multe cifre corecte, dar diferențele nu sunt extrem de mari între cele două. Desigur, pe măsură ce numărul de intrări (inclusiv calculele) crește, rezultatele obținute în F32 au o acuratețe mai mică. Precizia mai mare a P32 se poate datora numărului maxim de biți declarați pentru exponent, comparativ cu F32, care are un număr fix de biți.

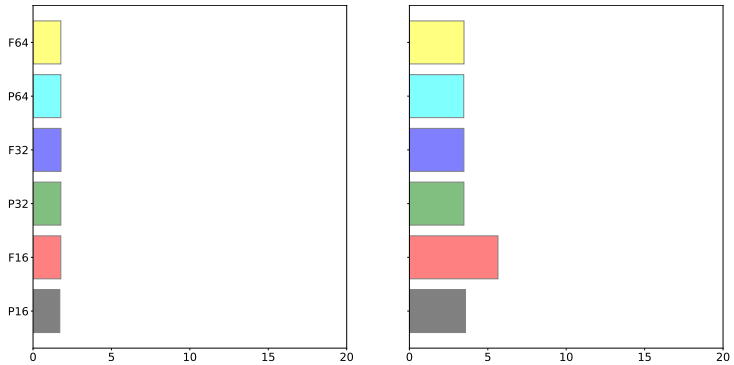


Figura 14: Precizia zecimală pentru testul T pentru perechi

4.2 Perspective asupra rezultatelor metodelor statistice

Acest capitol a examinat efectul modificării sistemului de reprezentare a numerelor în cadrul metodelor statistice. Deși, la prima vedere, o diferență de câteva sutimi poate părea să nu aibă un impact prea mare asupra rezultatului final, există domenii, cum ar fi biostatistica sau astrostatica, în care orice zecime în plus este considerată o informație nouă semnificativă. Funcțiile statistice implementate au fost alese pentru că sunt funcții relevante în cadrul domeniului statistic. Metodele statistice elementare au fost implementate în Python, iar corelația, detectarea distribuției și diferențele statistice au fost scrise în Scala. Limbajul Scala a fost utilizat pentru funcțiile complexe deoarece sistemul Posit din acesta este mai puternic și oferă suport pentru mai multe operații. În lupta dintre sistemul de reprezentare a numerelor Posit și sistemul de reprezentare a numerelor IEEE754, deja consacrat, au fost observate următoarele rezultate. Rezultate observate:

- Deși setul de date pe care se efectuează testul contează, de asemenea, Posit oferă întotdeauna o precizie mai bună decât IEEE754 pe același număr de biți, indiferent de metoda statistică utilizată.
- În funcție de setul de date și de metoda statistică, IEEE754 poate să nu ofere un rezultat valid (în acest caz pe 16 biți), în timp ce Posit oferă un rezultat bun pe același număr de biți; acest lucru se datorează flexibilității lui Posit în ceea ce privește exponentul și fracția, în comparație cu IEEE754, care are un număr fix (5 biți pentru exponent și 10 biți pentru fracție).
- În cazul testelor care se bazează aproape în întregime pe calcularea variației în formulă, cum ar fi testul t, IEEE754 are o precizie mai bună decât Posit.
- În cazul testelor care conțin și calculul variației plus alte operații aritmetice, cum ar fi ANOVA, Posit oferă în continuare o precizie mai bună, un număr mai semnificativ de cifre zecimale și rezultate mai bune pe mai puțini biți.
- În cazul Mann-Whitney și Chi-Square, Posit și IEEE754 au o acuratețe apropiată, formula fiind destul de simplă din punct de vedere matematic și, prin urmare, nu există diferențe semnificative de calcul; cu toate acestea, se poate considera că Posit este mai bun decât IEEE754 deoarece poate oferi rezultate semnificative pe mai puțini biți.

În urma acestui capitol, au fost studiate punctele forte și punctele slabe ale sistemului de reprezentare a numerelor Posit în cadrul metodelor statistice, în comparație cu sistemul actual de reprezentare a numerelor IEEE754. Operațiile implementate au fost: aritmetică de bază, corelații și diferențe statistice. Principalul motiv pentru care Posit este un concurent puternic în lupta pentru înlocuirea sistemului IEEE754 este caracterul fluctuant al exponentului și al mantissei. Acest lucru ajută la obținerea unor rezultate semnificative pe un număr mai mic de biți, oferind astfel o variantă care consumă mai puțină memorie și este mai rapidă în calculele statistice. Posit poate fi considerat un înlocuitor pentru IEEE754 în metodele statistice, având adesea performanțe mai bune. Sunt necesare încă cercetări mai aprofundate pentru a vedea întreaga putere a acestui nou sistem de reprezentare a numerelor.

5 IMPACTUL NRS ASUPRA INTELIGENȚEI ARTIFICIALE

5.1 Rezultatele Framework-urilor

Pentru prima evaluare LPML, setul de date MNIST este utilizat pentru antrenarea unei rețele neuronale formate din două straturi complet conectate, urmate de un strat logSoftmax. Rata de învățare de $0,01$, dimensiunea lotului de 64 și numărul de epoci de 3 au fost utilizate pentru instruire. Rezultatele temporale pentru propagarea înainte și înapoi sunt prezentate în tabelul 5. Timpul total pentru Posit și Fixed-Point a fost estimat. Cheltuielile software sunt prea mari pentru a putea efectua o instruire fezabilă folosind alte sisteme de reprezentare a numerelor decât IEEE754. Din acest moment, Posit și Fixed-Point au fost utilizate numai pentru inferență și optimizarea preciziei. O îmbunătățire consistentă a versiunii software a Posit sau Fixed-Point sau o implementare hardware poate redeschide subiectul instruirii. Cadrul LPML este pregătit pentru acest lucru și poate fi utilizat.

Rețeaua IEEE754 rezultată are o acuratețe de 93,47%. Pentru straturile complet conectate a fost utilizată conversia de precizie uniformă, iar rezultatele sunt prezentate în tabelul 6. Sistemul de reprezentare a numerelor cu virgulă fixă poate arăta limitele rețelelor neuronale. Scăderea la 4 biți pentru partea de fracții produce o degradare a preciziei de 26,26%, incomparabilă cu scăderea părții de numere întregi la 4 biți (0,19% degradare a preciziei). În cadrul testelor, trecerea la 2 biți pentru părțile întregi duce la o degradare a preciziei sub 50%. Acest lucru validează faptul că Fixed-Point este mai sensibil la dimensiunea fracțiilor decât la cea a părților întregi în rețelele neuronale. Posit are rezultate mai bune decât Fixed-Point chiar și în cazul acestei rețele mici cu aceeași precizie. Rezultatul interesant este că Posit(32,2) crește precizia rețelei cu 0,15%. Posit(16,1) reduce memoria utilizată pentru straturile complet conectate cu 50%, cu o degradare a preciziei de doar 0,02%. Posit(8,0) oferă o soluție pentru o reducere mare a memoriei (75%) cu un cost asupra preciziei de 1,29%. Având în vedere

NRS	Precizie	Forward Propagation (one)	Forward Propagation (total)	Backward Propagation (one)	Backward Propagation (total)
IEEE754	32	0.0006s	13.1s	0.0006s	12.6s
Posit	8	43.2s	33.75h	74.5s	58.20h
Posit	16	42.6s	29.25h	73.9s	57.73h
Fixed-Point	4-4	455s	355h	878s	685h
Fixed-Point	8-8	460s	359h	899s	702h

Tabela 5: Timp de propagare înainte și înapoi

NRS	Precizie	Acuratețe	Δ Acc	Memorie redusă
IEEE754(8,23)	32	93.47%	+0.00%	0%
Posit(8,0)	8	92.18%	-1.29%	75%
Posit(16,1)	16	93.45%	-0.02%	50%
Posit(32,2)	32	93.62%	+0.15%	0%
Fixed-Point(4,4)	8	67.21%	-26.26%	75%
Fixed-Point(4,8)	12	93.28%	-0.19%	62.5%
Fixed-Point(8,8)	16	93.42%	-0.05%	50%

Tabela 6: Precizia și reducerea memoriei pentru diferite sisteme de reprezentare a numerelor pe o rețea simplă

Metoda	Stratul Precizie	Acuratețe	Reducerea memoriei
Classic	32-32-32-32-32-32-32-32-32-32	94.11%	0%
KD	32-32-32-32-32-32-32-32-32-32	94.98%	0%
KD+PO	32-32-32-16-16-16-16-8-8-32	93.85%	43.47%

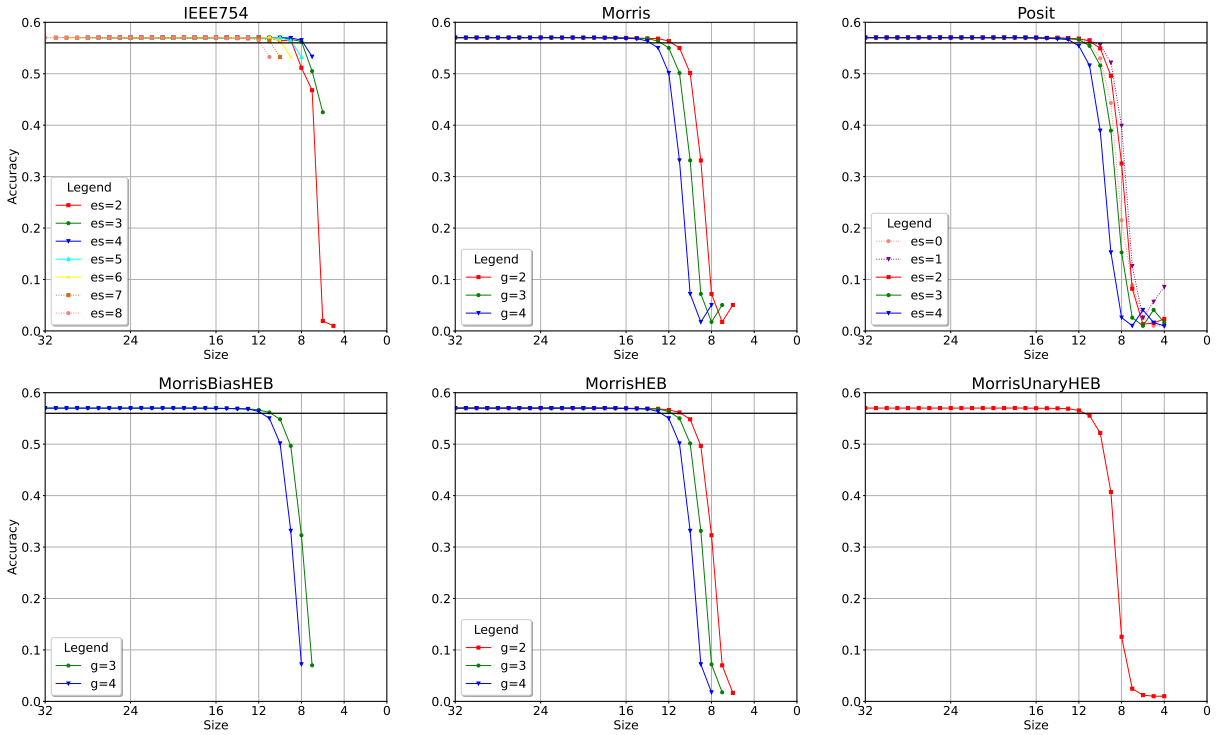
Tabela 7: Acuratețea pentru ResNet18 pe diferite metode de instruire

performanța slabă a Fixed-Point pentru LPML chiar și pe o rețea mică, dar și în următoarele experimente, rezultatul lor a fost omis. Următoarele experimente LPML prezintă utilizarea Posit și IEEE754.

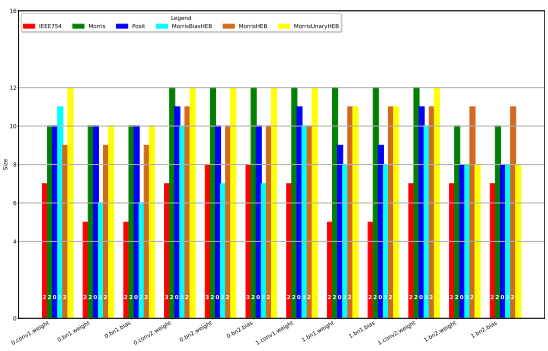
Pentru a testa formarea prin distilare a cunoștințelor LPML și optimizarea preciziei straturilor, au fost utilizate setul de date CIFAR-10, optimizatorul ADAM, rata de învățare de $0,005$ și pragul de precizie de 2%. Familia de rețele neuronale profunde este Resnet, Resnet34 este rețeaua profesorului, iar Resnet18 este rețeaua elevului. Creșterea acurateței pentru antrenamentul prin distilare a cunoștințelor față de antrenamentul clasic este validată în tabelul 7 cu o valoare de 0,87%. Utilizarea optimizării preciziei straturilor pentru modelul antrenat prin distilare a cunoștințelor are o degradare a acurateței de 0,26% în comparație cu antrenamentul clasic, o reducere a memoriei pentru straturile complet conectate și straturile convoluționale de 43,47%. Resnet34 are o acuratețe de 95,23% în cazul modelului clasic. Resnet18 instruit cu distilarea cunoștințelor și optimizarea preciziei straturilor reduce dimensiunea memoriei pentru straturile complet conectate și straturile convoluționale cu 66,75% pentru o degradare a preciziei de 1,38%. Resnet18 instruit cu distilarea cunoștințelor fără optimizarea preciziei straturilor reduce spațiul de stocare pentru aceleași straturi cu $41,19\%$, cu o degradare a preciziei de doar $0,25\%$. Utilizarea distilării cunoștințelor și a optimizării preciziei straturilor poate reduce la jumătate spațiul de stocare al unei rețele neuronale profunde, cu o degradare de aproape 1%.

Rezultatele

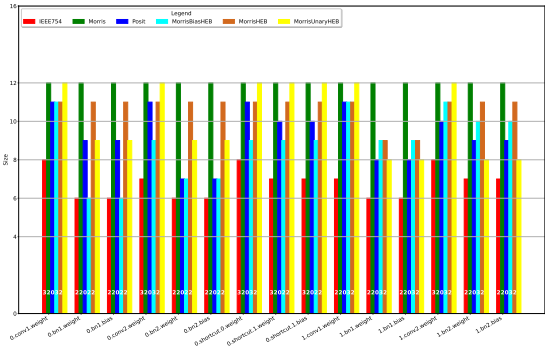
Figura 15: Precizia uniformă a PCML



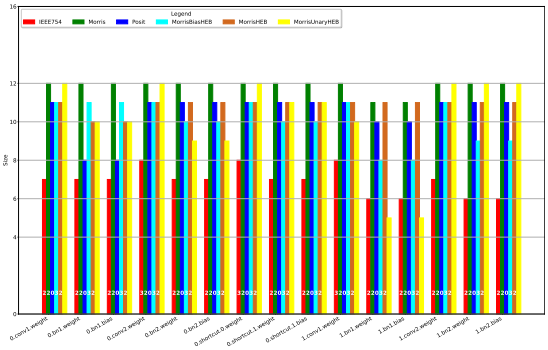
Pentru prima evaluare CPML, setul de date CIFAR-100 este utilizat pentru antrenarea unei rețele neuronale ResNet18. Optimizatorul utilizat este ADAM cuplat cu planificatorul ratei de învățare StepLR. Rata de învățare de $0,001$, dimensiunea lotului de 128 și numărul de epoci de 10 au fost utilizate pentru instruire. În primul rând, a fost utilizată optimizarea cu precizie uniformă, iar rezultatele sunt prezentate în figura 15. Pragul de precizie ales este de 1% și se vede sub forma liniei orizontale îngroșate. IEEE754 ne oferă informații despre valorile minime. Este nevoie de cel puțin 3 biți pentru mantisă. Toate sistemele IEEE754 cu o mantisă mai mică de 3 se află sub linie. Pentru biții de exponent, cea mai bună valoare este 4. Valorile mai mari sunt inutile, iar valorile mai mici se degradează mai repede în ceea ce privește precizia. Valorile exponentului sunt cuprinse între $[-7, 8]$. IEEE754 are cea mai mică dimensiune, cu o valoare de 8 pentru IEEE754(4,3). Urmează MorrisBiasHEB, MorrisHEB și Posit, cu o dimensiune de 11. Ceea ce este interesant pentru Posit este faptul că $es = 1$ are cea mai bună acuratețe pe toate dimensiunile, dar standardul $es = 2$ este încă aproape. Morris și MorrisUnaryHEB au nevoie de cel puțin 12 biți pentru a rămâne în pragul de precizie. MorrisHEB îmbunătățește Morris cu 1 bit. MorrisUnaryHEB este benefic deoarece nu are nevoie de elemente suplimentare pentru a cunoaște forma sa personalizată, la fel ca celelalte sisteme de reprezentare a numerelor. Pentru fiecare sistem de reprezentare a numerelor se alege cea mai mică dimensiune care se încadrează în pragul de precizie pentru fiecare sistem de reprezentare a numerelor pentru a intra în procesul de optimizare a preciziei straturilor. Figura 16 conține dimensiunile fiecărui sistem de reprezentare numerică utilizat pentru straturile din Resnet18. Cu alb pe fiecare bară este scrisă dimensiunea exponentului



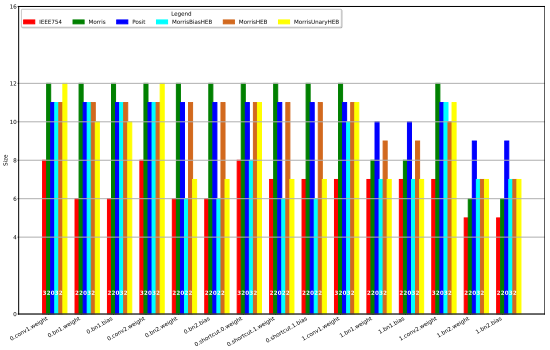
(a) Bloc 1



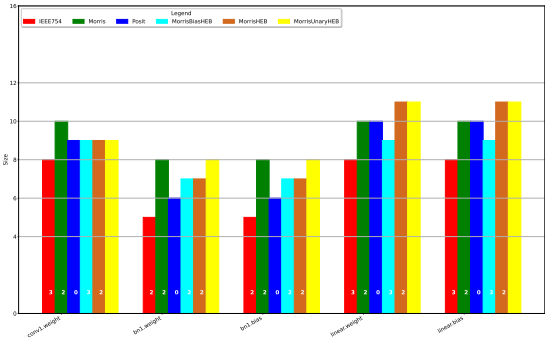
(b) Bloc 2



(c) Bloc 3



(d) Bloc 4



(e) Primul și ultimul bloc

Figura 16: Optimizarea preciziei stratului Resnet18

	IEEE754	Morris	Posit	MorrisBiasHEB	MorrisHEB	MorrisUnaryHEB
Uniform Accuracy	56.27%	56.35%	56.48%	56.17%	56.17%	56.55%
Uniform Size (KB)	10957	16435	15066	15066	15066	16435
Uniform MR	75%	62.5%	65.62%	65.62%	65.62%	62.5%
LPO Accuracy	55.38%	55.49%	55.49%	55.24%	55.23%	55.62%
LPO Size (KB)	10217	16412	15035	14662	14763	15688
LPO MR	6.75%	0.14%	0.20%	2.68%	2.01%	4.54%
LPO+Uniform MR	76.68%	62.55%	65.69%	66.54%	66.31%	64.20%

Tabela 8: Rezultatele CPML

pentru IEEE754 și Posit, iar valoarea g pentru Morris, MorrisBiasHEB și MorrisUnaryHEB. Singurul sistem care are toate mărimile sub 8 este IEEE754. Următorul cel mai bun sistem este MorrisBiasHEB, iar cel mai prost sistem este Morris. IEEE754 prezintă intervalul de valori ale exponentului pentru fiecare strat. Singurele sisteme care au rezultate mai bune decât IEEE754 pe unele dintre straturi sunt MrrosiBiasHEB și MorrisUnaryHEB. Cititorului trebuie să i se reamintească faptul că MorrisUnaryHEB are nevoie de informații suplimentare reprezentate de cifrele albe. În tabelul 8 sunt prezentate rezultatele optimizării uniformității și a preciziei pe straturi. Precizia inițială este de $57,01\%$. Pentru IEEE754, cu o degradare a preciziei mai mică de 2%, spațiul de stocare este redus cu $76,68\%$. Reducerea de memorie (MR) între rețeaua optimizată cu precizie uniformă și rețeaua optimizată cu precizie pe straturi (LPO) este mai mică de 10% pe toate sistemele (MR LPO). Costul suplimentar al păstrării metadatelor și al procesării optimizării preciziei pe straturi ar putea depăși beneficiile. Să presupunem că cele mai bune dimensiuni din toate sistemele de reprezentare numerică pentru fiecare strat sunt luate în considerare pentru reprezentarea modelului final. În acest caz, dimensiunea rețelei va fi cu numai 1 KB mai mică decât cea a omologului IEEE754. Rezultatele prezentate arată că optimizarea cu precizie uniformă IEEE754 este cea mai bună alegere. Aceasta reduce spațiul de stocare cu 75%, pierzând mai puțin de 1% în precizie. Acest lucru se explică prin faptul că formarea inițială și testarea inferenței se realizează pe IEEE754 pe 32 de biți. Lucrările viitoare trebuie să analizeze utilizarea altor sisteme de reprezentare numerică pentru instruire. Acest lucru este improbabil, având în vedere lipsa de hardware specializat.

5.2 Perspective privind Framework-urile de învățare automată

Cadrul LPML oferă o modalitate de instruire a rețelelor neuronale profunde cu Posit și Fixed-Point. Având în vedere implementările software și hardware actuale, acest tip de instruire este ineficient în ceea ce privește timpul de calcul. Posit este de cinci ori mai lent, iar Fixed-Point este de șase ori mai lent cu șase ordine de mărime. Fixed-Point ar putea fi utilizat pentru inferență și stocare pe dispozitive cu constrângeri de energie. Rezultatele obținute pe rețelele mici arată o degradare a preciziei de $0,19\%$ pentru o reducere a dimensiunii de stocare de $62,5\%$ pentru straturile complet conectate. Posit oferă un compromis mai bun pentru precizie.

Pe o rețea mică cu două straturi complet conectate urmate de un strat LogSoftmax, Posit pe 16 biți ($es = 1$) reduce dimensiunea cu 50% cu o pierdere de precizie de $0,02$.

În cazul rețelelor neuronale profunde, antrenamentul de distilare a cunoștințelor LPML combinat cu optimizarea preciziei straturilor reduce dimensiunea unei rețele Resnet34 cu $66,75\%$ cu o pierdere de $1,38\%$ în precizie sau, fără optimizarea preciziei straturilor, cu o reducere a dimensiunii de $41,19\%$ cu o pierdere de numai $0,25\%$. Creșterea preciziei pentru antrenarea prin distilare a cunoștințelor a fost validată cu $0,87\%$ pe o Resnet18 antrenată cu o Resnet34 față de o Resnet18 antrenată clasic. Utilizarea optimizării preciziei straturilor reduce dimensiunea straturilor complet conectate și convoluționale cu $43,47\%$, cu o pierdere de $1,13\%$. CPML oferă o optimizare uniformă sau de precizie a straturilor în șase sisteme diferite de reprezentare a numerelor. Degradarea acurateței pentru optimizarea preciziei straturilor este mai mică de 2%, cu o reducere a memoriei de $76,68\%$ în cel mai bun caz (IEEE754). Toate sistemele de reprezentare a numerelor au o reducere a memoriei de peste 60% în cazul optimizării cu precizie uniformă sau pe straturi. Optimizarea cu precizie uniformă utilizând IEEE754 are cele mai bune rezultate, reducând dimensiunea de stocare cu 75% și pierzând doar $0,74\%$ în precizie. Optimizarea preciziei pe straturi nu prezintă beneficii semnificative în comparație cu optimizarea cu precizie uniformă, iar costul păstrării metadatelor pentru fiecare strat ar putea face ca diferența să fie și mai mică. Aspectul interesant al rezultatelor este că se utilizează un IEEE754 pe 8 biți ($es = 4$ și $fs = 3$) în precizie uniformă. Acesta este singurul sistem de reprezentare a numerelor cu o precizie uniformă egală sau mai mică de 8. Aceste rezultate sunt influențate de faptul că antrenamentul și testarea inferenței se face pe IEEE754 pe 32 de biți.

Lucrările viitoare trebuie să se concentreze asupra implementării unor acceleratoare hardware pentru a face fezabilă în timp instruirea în cadrul altor sisteme de reprezentare a numerelor. Lucrările actuale se concentrează asupra sistemelor încorporate, dar având în vedere rezultatele bune în ceea ce privește precizia, în viitor pot fi analizate rețele neuronale profunde și seturi de date mai complexe.

6 DEZVOLTAREA BIBLIOTECII GENERATOARE DE HARDWARE (NRS-HGL)

6.1 Evaluare FPGA

Evaluarea FPGA a vizat placa Virtex-7 VC709 Evaluation Platform. Condițiile sunt: temperatură de 25C, radiator ridicat, flux de aer de 500 LFM și tip de proces - maxim. Xilinx Vivado 2022.1 rulează pe Windows 11 Education 21H2 22000.1219.

Se evaluează un număr de 25 de sisteme de reprezentare a numerelor (8 biți 5, 16 biți 8, 19 biți 1, 24 biți 2, 32 biți 7, 64 biți 2). Dimensiunile interne ale exponenților și fracțiilor acestora sunt prezentate în tabelul 9. Sistemele de reprezentare a numerelor derivate din IEEE754 utilizează aceeași cantitate de biți pentru dimensiunile interne ca și în reprezentarea lor binară. În comparație cu MorrisHEB, MorrisHEB schimbă un bit de exponent pentru un bit de fracție. MorrisBiasHEB este apropiat de omologii IEEE754. MorrisUnaryHEB utilizează cei mai mulți biți dintre toate sistemele de reprezentare a numerelor de aceeași dimensiune. Morris și MorrisHEB îl urmează. Operațiile unare/binare, modulele TFPU, KAU și GFPU sunt evaluate în ceea ce privește consumul de energie, utilizarea resurselor hardware, întârzierea, perioada și frecvența maximă. Toate unitățile utilizează un singur ciclu. Performanța are loc de îmbunătățire. Rezultatele prezentate trebuie considerate ca fiind de referință.

Pornind de la modulele de adunare/substracție pe 8 biți, cel care are cel mai mic consum de energie este Morris. Morris are mai multe reprezentări, iar acest lucru poate fi semnificativ în cazul dimensiunilor de biți mai mici. MorrisBiasHEB este cel mai eficient în ceea ce privește frecvența, întârzierea, perioada și resursele hardware. Dimensiunea internă a exponentului este de 2, astfel încât poate reprezenta doar valori $-3, -2, -1, 0, 1, 2, 3$ pentru exponent. Este posibil ca intervalul dinamic să trebuiască să fie mai mare pentru ca unele aplicații să fie considerate NRS utilizabile. Compromisul de a lua un bit de exponent pentru a-l transforma într-un bit de mantisă se observă în diferențele dintre Morris și MorrisHEB. Numărul de LUT-uri scade, iar frecvența maximă crește pentru dimensiuni de bit mai mici. Posit și MorrisUnaryHEB au un cost aproape dublu al LUT-urilor și o scădere a frecvenței maxime. Posit are cel mai mare consum de energie dintre acestea. Dacă se trece la 16 biți, este evidentă dominația lui MorrisBiasHEB în ceea ce privește viteza (frecvență maximă, întârziere, perioadă). Posit și MorrisUnaryHEB au o cantitate aproape dublă de LUT-uri în comparație cu celelalte sisteme de reprezentare a numerelor. Diferența dintre Morris și MorrisHEB în ceea ce privește consumul de energie și LUT-urile este minimă. Brainfloat de la Google (bfloat16) este mai bun decât IEEE754 de semiprecizie în toate departamentele (consum mai mic de energie, mai puține resurse hardware, viteză mai bună). În ceea ce privește viteza, IEEE754

Number representation system	Internal Exponent	Internal Fraction	Accumulator Size	Accumulator Fraction
Morris(2, 8, RoundZero)	4	3	62	30
Morris(3, 16, RoundZero)	8	10	1028	516
Morris(4, 32, RoundZero)	16	25	262162	131090
MorrisHEB(2, 8, RoundZero)	3	4	34	18
MorrisHEB(3, 16, RoundZero)	7	11	520	264
MorrisHEB(4, 32, RoundZero)	15	26	131094	6558
MorrisUnaryHEB(8, RoundEven)	6	5	130	64
MorrisUnaryHEB(16, RoundEven)	14	13	32770	16384
MorrisUnaryHEB(32, RoundEven)	30	29	2×10^9	1×10^9
MorrisBiasHEB(2, 8, RoundEven)	2	5	20	12
MorrisBiasHEB(3, 16, RoundEven)	4	12	66	34
MorrisBiasHEB(4, 32, RoundEven)	8	27	808	296
Half Float(Round Even)	5	10	80	48
Google Brainfloat(Round Even)	8	7	522	266
Nvidia Tensorfloat(Round Even)	8	10	528	272
AMD FP24(Round Even)	7	16	286	156
Pixar PXR24(Round Even)	8	15	538	282
Float(Round Even)	8	23	554	298
Double(Round Even)	11	52	4196	2148
Posit(8, 0, RoundEven)	3	5	26	12
Posit(16, 1, RoundEven)	5	12	114	56
Posit(16, 2, RoundEven)	6	11	226	112
Posit(32, 2, RoundEven)	7	27	482	240
Posit(32, 3, RoundEven)	8	26	960	480
Posit(64, 3, RoundEven)	9	58	1986	992

Tabela 9: Sistemele de reprezentare a numerelor Dimensiunea internă

Number representation system	Power (W)	Hardware Resources	Delay (ns)	Period (ns)	Frequency (MHz)
Morris(2, 8, RoundZero)	1.001	207 LUT	16.213	16.240	61.57
MorrisHEB(2, 8, RoundZero)	1.007	197 LUT	14.731	14.758	67.75
MorrisUnaryHEB(8, RoundEven)	1.01	434 LUT	18.772	18.799	53.19
MorrisBiasHEB(2, 8, RoundEven)	1.006	186 LUT	14.406	14.433	69.28
Posit(8, 0, RoundEven)	1.012	313 LUT	18.210	18.247	54.80
Morris(3, 16, RoundZero)	1.018	674 LUT	17.728	17.755	56.32
MorrisHEB(3, 16, RoundZero)	1.019	678 LUT	17.441	17.468	57.24
MorrisUnaryHEB(16, RoundEven)	1.02	1258 LUT	20.890	20.918	47.80
MorrisBiasHEB(3, 16, RoundEven)	1.021	605 LUT	16.057	16.085	62.16
Google Brainfloat(Round Even)	1.008	541 LUT	22.830	26.855	37.23
Half Float(Round Even)	1.012	678 LUT	26.168	30.193	33.12
Posit(16, 1, RoundEven)	1.024	932 LUT	22.897	22.921	43.62
Posit(16, 2, RoundEven)	1.015	836 LUT	27.740	27.367	36.54
Nvidia Tensorfloat(Round Even)	1.011	641 LUT	25.957	25.984	38.48
AMD FP24(Round Even)	1.01	988 LUT	28.211	31.246	32.00
Pixar PXR24(Round Even)	1.011	874 LUT	28.763	28.790	34.73
Morris(4, 32, RoundZero)	1.034	1680 LUT	19.828	19.855	50.36
MorrisHEB(4, 32, RoundZero)	1.039	1647 LUT	20.524	20.551	48.65
MorrisUnaryHEB(32, RoundEven)	1.039	3302 LUT	23.914	23.941	41.76
MorrisBiasHEB(4, 32, RoundEven)	1.039	1394 LUT	19.226	19.279	51.86
Float(Round Even)	1.017	1611 LUT	39.017	39.044	25.62
Posit(32, 2, RoundEven)	1.036	2098 LUT	29.740	29.807	33.54
Posit(32, 3, RoundEven)	1.031	1943 LUT	31.802	31.829	31.41
Double(Round Even)	1.027	3537 LUT	49.796	53.917	18.54
Posit(64, 3, RoundEven)	1.068	4053 LUT	33.165	33.257	30.06

Tabela 10: Generatoare de adiție + scădere rezultate FPGA

Number representation system	Power (W)	Hardware Resources	Delay (ns)	Period (ns)	Frequency (MHz)
Morris(2, 8, RoundZero)	1.002	439 LUT	17.296	17.323	57.72
MorrisHEB(2, 8, RoundZero)	1.003	484 LUT	18.161	18.188	54.98
MorrisUnaryHEB(8, RoundEven)	1.003	916 LUT	26.037	26.064	38.36
MorrisBiasHEB(2, 8, RoundEven)	1.002	454 LUT	20.062	20.089	49.77
Posit(8, 0, RoundEven)	1.003	894 LUT	26.007	26.034	38.41
Morris(3, 16, RoundZero)	1.004	1374 LUT, 1 DSP	36.048	36.075	27.72
MorrisHEB(3, 16, RoundZero)	1.006	1557 LUT, 1 DSP	41.585	41.622	24.02
MorrisUnaryHEB(16, RoundEven)	1.005	2262 LUT, 1 DSP	44.282	44.309	22.56
MorrisBiasHEB(3, 16, RoundEven)	1.007	1456 LUT, 1 DSP	39.705	39.732	25.16
Google Brainfloat(Round Even)	1.005	1527 LUT	33.281	33.308	30.02
Half Float(Round Even)	1.004	1892 LUT, 1 DSP	47.395	47.422	21.08
Posit(16, 1, RoundEven)	1.008	2399 LUT, 1 DSP	58.177	58.204	17.18
Posit(16, 2, RoundEven)	1.01	2178 LUT, 1 DSP	52.312	52.339	19.10
Nvidia Tensorfloat(Round Even)	1.005	1911 LUT, 1 DSP	48.206	48.233	20.73
AMD FP24(Round Even)	1.008	2208 LUT, 1 DSP	54.446	54.473	18.35
Pixar PXR24(Round Even)	1.008	2196 LUT, 1 DSP	53.402	53.429	18.71
Morris(4, 32, RoundZero)	1.014	4124 LUT, 4 DSP	95.787	95.814	10.43
MorrisHEB(4, 32, RoundZero)	1.015	4384 LUT, 4 DSP	115.435	115.462	8.66
MorrisUnaryHEB(32, RoundEven)	1.012	6733 LUT, 4 DSP	113.903	113.930	8.77
MorrisBiasHEB(4, 32, RoundEven)	1.016	3945 LUT, 4 DSP	89.558	89.585	11.16
Float(Round Even)	1.009	5129 LUT, 2 DSP	103.478	103.505	9.66
Posit(32, 2, RoundEven)	1.022	4652 LUT, 4 DSP	87.890	87.917	11.37
Posit(32, 3, RoundEven)	1.022	4784 LUT, 4 DSP	88.823	88.850	11.25
Double(Round Even)	1.037	17989 LUT, 9 DSP	214.545	214.572	4.66
Posit(64, 3, RoundEven)	1.065	13304 LUT, 15 DSP	250.921	250.948	3.98

Tabela 11: Rezultatele FPGA ale generatoarelor TFPU

sunt cele mai lente în această situație. Un rezultat interesant este faptul că regula consumului de energie și numărul de LUT-uri pentru schimbul de biți de exponent cu biți de mantisă se schimbă pentru dimensiuni mai mari. O transformare a unui bit de exponent într-un bit de mantisă pentru Posit crește consumul de energie și numărul de LUT-uri. Acest lucru este validat de sistemele de reprezentare a numerelor pe 24 de biți (FP24, PXR24). La 32 de biți, MorrisUnaryHEB are un număr aproape triplu de LUT-uri în comparație cu celelalte sisteme. Mărimile interne sunt similare cu Posit pe 64 de biți și IEEE754. Un MorrisUnaryHEB pe 64 de biți este considerat o suprautilizare a resurselor hardware, chiar și cu avantajele rezultatelor exacte ale adunării/substracției, ceea ce îl face un excelent acumulator inexact. IEEE754 are cele mai proaste rezultate, chiar dacă celelalte au exponenți și fracții interne de dimensiuni mai mari. Cheltuielile suplimentare nu pot proveni decât de la modulele de decodare și codificare. Posit este al doilea după MorrisUnaryHEB în ceea ce privește numărul de LUT-uri și mai lent decât toate sistemele omologe de reprezentare a numerelor divizate Morris. MorrisBiasHEB își păstrează viteza mai bună și o mai bună utilizare a resurselor pe 32 de biți. Toate sistemele de reprezentare a numerelor divizate cu exponent ascuns de Morris au un consum de energie mai mare. Pe de altă parte, IEEE754 are cel mai mic consum de energie. Pe 64 de biți, Posit are un consum de energie mai mare și utilizează mai multe LUT-uri (14,58 Următoarele module sunt TFPU-urile din tabelul 11. La sistemele de reprezentare a numerelor pe 8 biți, ținând cont de numărul de valori diferite pe care le poate reprezenta (Morris) și de domeniul dinamic (MorrisBiasHEB), cea mai bună alegere este MorrisHEB, urmat de Posit, care are un număr dublu de LUT-uri și aproape jumătate din frecvența maximă. Pe 16 biți, nu este surprinzător faptul că bfloat16 câștigă ușor, fiind singurul care nu necesită un DSP. Nici mai puțin, bfloat16 este mai degrabă un sistem de reprezentare inexactă a numerelor, potrivit pentru calcul decât să se poată corecta singur, precum învățarea automată (domeniul său

Number representation system	Power (W)	Hardware Resources	Delay (ns)	Period (ns)	Frequency (MHz)
Morris(2, 8, RoundZero)	0.993	931 LUT	16.762	16.789	59.56
MorrisHEB(2, 8, RoundZero)	0.993	542 LUT	14.301	14.328	69.79
MorrisUnaryHEB(8, RoundEven)	0.996	2035 LUT	24.456	24.483	40.84
MorrisBiasHEB(2, 8, RoundEven)	0.992	353 LUT	12.896	12.913	77.44
Posit(8, 0, RoundEven)	0.993	609 LUT	20.338	20.365	49.10
Morris(3, 16, RoundZero)	1.054	23215 LUT, 3 DSP	56.353	56.380	17.73
MorrisHEB(3, 16, RoundZero)	1.021	11706 LUT, 3 DSP	52.346	52.373	19.09
MorrisUnaryHEB(16, RoundEven)	1.111	50985 LUT, 3 DSP	68.988	69.015	14.48
MorrisBiasHEB(3, 16, RoundEven)	0.995	1474 LUT, 3 DSP	18.997	19.024	52.56
Google Brainfloat(Round Even)	1.013	13656 LUT	74.764	74.791	13.37
Half Float(Round Even)	1.008	1939 LUT, 3 DSP	39.543	39.570	25.27
Posit(16, 1, RoundEven)	0.999	2595 LUT, 3 DSP	35.008	35.035	28.54
Posit(16, 2, RoundEven)	1.048	5639 LUT, 3 DSP	41.725	41.752	23.95
Nvidia Tensorfloat(Round Even)	1.036	14707 LUT, 3 DSP	56.037	56.064	17.83
AMD FP24(Round Even)	1.012	7493 LUT, 3 DSP	43.264	43.291	23.09
Pixar PXR24(Round Even)	1.025	13055 LUT, 3 DSP	50.964	50.991	19.61
Morris(4, 32, RoundZero)	1.101	50047 LUT, 12 DSP	72.134	72.161	13.85
MorrisBiasHEB(4, 32, RoundEven)	1.049	20149 LUT, 12 DSP	53.383	53.410	18.72
Float(Round Even)	1.043	16313 LUT, 6 DSP	56.072	56.099	17.82
Posit(32, 2, RoundEven)	1.04	14639 LUT, 12 DSP	60.295	60.322	16.57
Posit(32, 3, RoundEven)	1.091	27952 LUT, 12 DSP	67.068	67.095	14.90
Double(Round Even)	1.23	62370 LUT, 27 DSP	82.051	82.078	12.18
Posit(64, 3, RoundEven)	1.267	61636 LUT, 45 DSP	84.637	84.664	11.81

Tabela 12: Rezultatele FPGA ale generatoarelor KAU

de aplicare, de asemenea). Pentru o abordare aritmetică mai generală, MorrisBiasHEB este cel mai bun compromis, fiind aproape de bfloat16 în ceea ce privește performanța hardware. Posit și MorrisUnaryHEB necesită cele mai multe LUT-uri, dar cel puțin MorrisUnaryHEB are o frecvență maximă mai bună decât IEEE754 de semiprecizie. Posit cu $es = 1$ are rezultate mai proaste decât sistemele IEEE754 pe 24 de biți. În departamentul de 32 de biți, este o surpriză faptul că Posit are rezultate mai bune în ceea ce privește viteza. De asemenea, are cel mai mare consum de energie dintre toate. MorrisBiasHEB are performanțe comparabile (11,16 față de 11,37 și 11,25) folosind cu 15,2% mai puține LUT-uri și un consum de energie mai neglijabil (1,016 față de 1,022). Cel mai eficient consum de energie este IEEE754, care utilizează cu două DSP-uri mai puțin. Departamentul de 64 de biți are un IEEE754 mai eficient din punct de vedere energetic și mai rapid decât Posit. Rezultatele KAU-urilor sunt în tabelul 12. Dimensiunile acumulatorului și ale fracției pentru fiecare sistem de reprezentare a numerelor sunt în tabelul 9. Pentru toate sistemele de reprezentare a numerelor, dimensiunea fracției acumulatorului este limitată la 1024, iar dimensiunea acumulatorului este limitată la 2048. Numărul de biți necesari pentru sistemele de reprezentare a numerelor divizate Morris nu este fezabil, cu excepția MorrisBiasHEB. Deoarece dimensiunile acumulatorilor lor sunt plafonate, acumulatorii sunt inexacți. Pe 8 biți, domeniul dinamic ridicat al MorrisUnaryHEB, Morris și Posit oferă un număr mare de LUT-uri pentru implementarea unui acumulator exact. Ținând cont de domeniul dinamic, Posit este cea mai bună alegere. MorrisHEB are o valoare maximă a exponentului de 7 (24 Posit), iar MorrisUnaryHEB are nevoie de mai multe LUT-uri (2053 vs. 609). În departamentul de 16 biți, din nou, alegerea este între MorrisBiasHEB și bfloat16. Primul are un consum de energie mai mic și o frecvență maximă mai bună. Cel din urmă nu are nevoie de DSP-uri. Posit rămâne o soluție alternativă. Celelalte variante Morris devin nefezabile din cauza domeniului dinamic extrem de ridicat și sunt excluse din evaluarea pe 32 de biți. Pe 32 de biți, Posit cu $es = 2$ este cea mai bună alegere pentru consumul de energie

Size (bits)	Power (W)	Hardware Resources	Delay (ns)	Period (ns)	Frequency (MHz)
32	1.021	12008 LUT, 4 DSP	155.646	155.673	6.42
64	1.394	34613 LUT, 16 DSP	449.650	449.677	2.22

Tabela 13: GFPU Generatoare FPGA Rezultate FPGA

și utilizarea resurselor, iar viteza este apropiată de cea de-a doua alegere MorrisBiasHEB. Rezultatele pe 64 de biți confirmă costul ridicat al resurselor hardware pentru acumulatori în acest departament. Chiar și cu acest cost ridicat, frecvența maximă are rezultate acceptabile (10MHz). Ultimul modul testat este GFPU. Acesta are nevoie de aproape de trei ori mai multe resurse hardware decât FPU-ul specific omologului FPU. Aceste valori provin de la modulele de codificare și decodificare multiple. Consumul de energie pentru 32 de biți este similar cu omologii TFPU. GFPU pe 64 de biți are un consum de energie mai mare, de 30,9%. Numărul de DSP-uri este similar cu cel al omologilor lor. Frecvența maximă este aproape la jumătate (6,4 față de 8,6 și 2,2 față de 3,9).

6.2 Perspective privind rezultatele unităților propuse

Capitolul de față propune o infrastructură de cercetare hardware pentru sistemele de reprezentare a numerelor. Aceasta este validată prin adăugarea a trei noi sisteme de reprezentare a numerelor: MorrisHEB, MorrisBiasHEB și MorrisUnaryHEB, dar și prin implementarea unor sisteme clasice precum IEEE754, Morris și Posit. Au fost implementate proceduri de generare pentru FPU, KAU și GFPU. Implementarea pentru adăugarea unui nou sistem de reprezentare a numerelor se reduce la module de codificare și decodificare. Punctele de referință și unitățile vor fi generate fără efort suplimentar.

Rezultatele evaluării FPGA prezentate reprezintă o bază de referință de la care se pot face îmbunătățiri viitoare. Sistemul intern în virgulă mobilă poate fi îmbunătățit cu optimizările realizate pentru IEEE754 în ultimele decenii. O astfel de îmbunătățire va îmbunătăți fiecare sistem de reprezentare a numerelor implementat în bibliotecă.

A fost validată utilizarea lui Posit pentru implementarea acumulatorilor Kulisch. Cu toate acestea, MorrisUnaryHEB pe 8 biți poate fi o alternativă în cazul în care este nevoie de o gamă dinamică mai mare, cu prețul resurselor hardware și al consumului de energie. Pe 16 biți, MorrisBiasHEB este un sistem de reprezentare a numerelor mai eficient și mai rapid pentru un acumulator Kulisch. bfloat16 de la Google este o alternativă pentru a avea un hardware mai puțin complex. Pentru 32 de biți, alegerea este, de asemenea, între MorrisBiasHEB și Posit (viteză versus resurse și energie). Costul acumulatorilor Posit și IEEE754 pe 64 de biți este ridicat (peste 60000LUTS cu 27 DSP sau 45 DSP).

O GFPU este o unitate propusă care poate avea operanzi și rezultate în diferite sisteme de reprezentare a numerelor de aceeași dimensiune a lățimii de bit. Numărul de LUT-uri este

triplat, iar frecvența maximă este înjumătățită în comparație cu o FPU specifică. Beneficiile utilizării unor sisteme diferite de reprezentare a numerelor în funcție de aplicație sau de părți ale aplicației pot depăși costurile.

În cazul FPU specifice (denumite TFPU), Posit prezintă rezultate promițătoare pentru implementarea pe 32 de biți. MorrisBiasHEB pe 32 de biți este apropiat de Posit pe 32 de biți în ceea ce privește performanța și are o utilizare mai bună a resurselor hardware și un consum de energie mai mic. Pe 16 biți, MorrisBiasHEB este soluția cea mai bună pentru aritmetica generală, iar bfloat16 este mai bună pentru utilizarea învățării automate. În departamentul pe 64 de biți, IEEE754 prezintă rezultate mai bune decât omologul Posit.

Optimizarea virgulă mobilă internă în lucrările viitoare este o cale bună pentru îmbunătățirea performanțelor bibliotecii. Unităților propuse li se pot adăuga acceleratoare de inteligență artificială și de procesare a semnalelor digitale.

7 PROCESOARE CU NRS

7.1 Evaluarea procesoarelor

Această secțiune evaluează și analizează abordarea propusă pe baza acurateței, eficienței, ariei estimate și a puterii. Se face o comparație între FPU original pe 32 de biți al Rocket Chip, care pretinde că implementează standardul IEEE 754 (**FP32**), cu *E-PAU* propus, care funcționează cu sisteme Posit de trei lățimi de bit, și anume 8 biți cu exponent pe 1 bit, indicat prin **Posit(8,1)** sau **P8**, 16 biți cu exponent pe 2 biți, indicat prin **Posit(16,2)** sau **P16**, și 32 biți cu exponent pe 3 biți, indicat prin **Posit(32,3)** sau **P32**. *E-PAU* propus este scris în Chisel și integrat cu Rocket Chip [1] și utilizează platforma de dezvoltare Freedom E310[1] de la SiFive pentru a implementa și sintetiza codul pentru a rula pe un FPGA Arty A7-100T.

7.1.1 Repere pentru procesoare

Pentru a evalua abordarea, sunt selectate următoarele repere care utilizează operații în virgulă mobilă. Aceste criterii de referință sunt organizate în trei niveluri, după cum urmează. Reperele de nivel unu sunt utilizate pentru a evalua atât acuratețea, cât și eficiența, în termeni de cicluri, a *E-PAU* propus în comparație cu FPU IEEE 754 original al Rocket Chip. Aceste criterii de referință reprezintă calculul unor constante matematice bine cunoscute care utilizează serii și secvențe. În special, sunt calculate următoarele constante: π și e (numărul lui Euler), folosind serii numerice, așa cum se arată în tabelul 14. Pentru π, se utilizează seriile Leibniz și Nilakantha [5]. Deoarece seria Leibniz converge lent, se execută timp de două milioane de iterații. În schimb, seria Nilakantha converge mai rapid și rulează timp de 200 de iterații. Pentru e, se utilizează seria lui Euler, care converge rapid. Astfel, se execută timp de 20 de iterații. În plus față de π și e, se calculează $sin(1)$ pentru 100 de iterații. Nivelul doi constă în nuclee utilizate de obicei în aplicațiile ML [15], așa cum sunt rezumate în tabelul 16. Pentru aceste nuclee, se evaluează eficiența *E-PAU* față de FPU în termeni de cicluri. Corectitudinea rezultatelor este verificată în raport cu ieșirile de referință. În continuare, este prezentată o scurtă descriere a fiecărui nucleu. Înmulțirea matricelor (**MM**) implementează înmulțirea a două matrici pătrate, care este adesea utilizată în sarcini de lucru ML și HPC. În cadrul testelor, matricile acomodează dimensiuni de până la $n = 182$. K-means (**KM**) grupează un set de puncte multidimensionale în k grupuri, sau clustere, pe baza distanței lor euclidiene. KM este adesea utilizat în aplicații de ML și de analiză a datelor. Algoritmul K-nearest neighbours

[1] https://github.com/sifive/freedom

(**KNN**) clasifică un punct multidimensional pe baza distanței euclidiene față de cei k vecini cei mai apropiați. Regresia liniară (**LR**) este un nucleu utilizat în ML și în analiza datelor. Se utilizează, de asemenea, Regresia liniară multivariată, care constă în operații matriceale și vectoriale. Naive Bayes (**NB**) implementează un model bayesian simplu. Nucleul de clasificare (sau de decizie) (Classification (or Decision) Tree (**CT**) este utilizat în ML și în analiza datelor pentru a reprezenta o variabilă țintă pe baza unor atribute de intrare. Se evaluează atât crearea (antrenarea), cât și utilizarea (inferența) de CT. În acest scop, setul de date Iris este utilizat ca intrare pentru criteriile de referință de nivel doi, cu excepția MM. Acest set de date este format din $n = 150$ puncte de date cu $m = 4$ dimensiuni reprezentând flori. Aceste puncte aparțin la $k = 3$ clase. Nivelul trei al suitei de criterii de referință reprezintă modele ML complete. În lucrarea actuală, rețeaua neuronală convoluțională (CNN) implementată în Caffe și antrenată pe setul de date CIFAR-10[2]. Deși CNN-ul original are 14 straturi, iar fișierul de parametri are o dimensiune de 351 kB, acesta nu poate fi adaptat la un testbed cu o dimensiune limitată a memoriei. Prin urmare, se iau doar ultimele patru straturi ale acestui CNN, începând de la *relu3*, și se generează un cod C standard cu alocări statice de memorie. Prin instrumentarea cadrului Caffe, au fost colectați toți parametrii și datele de intrare ale stratului *relu3* sub formă de fișiere binare cu valorile *FP32*. Aceste fișiere binare sunt convertite în toate cele trei dimensiuni de pozit, și anume P8, P16 și P32, sunt transformate în obiecte și sunt legate cu codul C generat pentru a obține executabilul RISC-V final. Validarea se execută pe toate cele 10 000 de imagini din setul de date de testare CIFAR-10 prin rularea executabilelor pe FPGA. Rezultatele predicției sunt comparate cu execuția de referință pe o gazdă x86/64.

7.1.2 Precizia și eficiența

Nivelul unu. Se evaluează acuratețea și eficiența lui Posit în comparație cu virgulă mobilă IEEE 754 pe 32 de biți, cu o singură precizie (*FP32*), folosind reperele de nivel unu rezumate în Tabelul 14 și Tabelul 15. Acuratețea se măsoară în cifre fracționare exacte în comparație cu valoarea de referință. Eficiența reprezintă numărul de cicluri de care are nevoie Rocket Chip care rulează pe FPGA pentru a executa secțiunea semnificativă a programului. Pentru poziția , accelerarea este calculată în raport cu execuția *FP32*. Rezultatele prezentate în tabelul 14 arată că *Posit(32,3)* obține o precizie similară sau mai bună în comparație cu *FP32*, economisind până la 23% din ciclurile utilizate de FPU *FP32*, atunci când se calculează π cu serii Leibniz. Pe de altă parte, [6] arată că orice Posit poate fi reprezentat cu acurateță de un float IEEE754 de o dimensiune mai mare. Acesta este motivul pentru care în scripturile de evaluare se utilizează virgulă mobilă IEEE 754 de 64 de biți, cu dublă precizie. Operațiile de poziționare necesită mai puține cicluri pentru a fi finalizate. Astfel, aplicațiile cu un număr mai mare de iterații prezintă o eficiență mai bună. De exemplu, *Posit(32,3)* este cu 30%, 9% și 3% mai rapid decât *FP32* pentru π Leibniz cu două milioane de iterații, π Nilakantha cu 200 de iterații și, respectiv, e cu 20 de iterații. Analiza a arătat că această creștere a

[2] `https://www.cs.toronto.edu/~kriz/cifar.html`

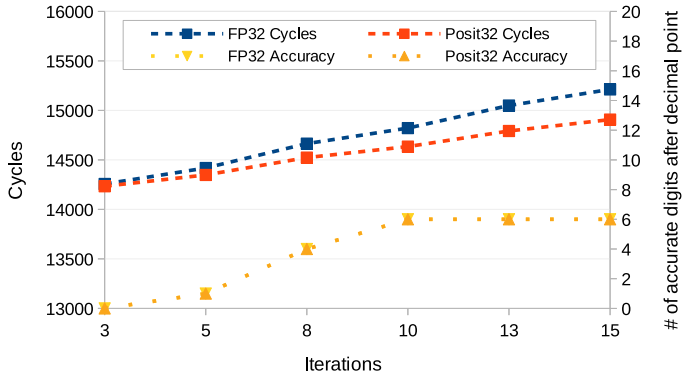


Figura 17: Precizia și eficiența calculului numărului lui Euler folosind *FP32* și *Posit(32,3)*

Tabela 14: Acuratețe (Nivel unu de referință)

Application	**Iterations**	**Accuracy** [actual value | number of exact fraction digits]							
		FP32		**Posit(8,1)**		**Posit(16,2)**		**Posit(32,3)**	
π (Leibniz)	2,000,000	3.14159	5	3.5	0	3.14	2	3.14159	5
π (Nilakantha)	200	3.1415929	6	3.125	1	3.141	3	3.1415922	6
e (Euler)	20	2.7182819	6	2.625	0	2.718	3	2.7182817	6
$sin(1)$	100	0.8414709	7	0.78	0	0.8413	3	0.84147098	8

Tabela 15: Eficiență (Repere de nivel 1)

Application	**Iterations**	**Efficiency** [cycles | speedup]						
		FP32	**Posit(8,1)**		**Posit(16,2)**		**Posit(32,3)**	
π (Leibniz)	2,000,000	216,022,827	166,022,835	1.30	166,022,829	1.30	166,022,830	1.30
π (Nilakantha)	200	57,940	52,937	1.09	52,952	1.09	52,937	1.09
e (Euler)	20	15,598	15,177	1.03	15,177	1.03	15,177	1.03
$sin(1)$	100	16,663	16,270	1.02	16,273	1.02	16,298	1.02

Tabela 16: Eficiență (Repere de nivel doi). Un fundal gri înseamnă că rezultatul este diferit de cel de referință.

Benchmark	**Input Size**	**Efficiency** [cycles | speedup]						
		FP32	**Posit(8,1)**		**Posit(16,2)**		**Posit(32,3)**	
Matrix Multiplication (MM)	n = 182	418,177,415	418,063,614	1.0	418,063,629	1.0	418,177,423	1.0
k-means (KM)	Iris dataset n = 150 m = 4 k = 3	19,150,075	18,879,618	1.01	18,971,747	1.01	19,011,507	1.01
k Nearest Neighbours (KNN)		151,402	138,140	1.10	143,313	1.06	144,136	1.05
Linear Regression (LR)		1,419,794	-	-	-	-	1,398,643	1.02
Naive Bayes (NB)		398,254	407,330	0.98	397,869	1.0	399,893	1.0
Classification Tree (CT)		633,560	101,940	6.2	615,792	1.03	629,936	1.01

vitezei rezultă din operațiile mai rapide de înmulțire și împărțire pe Posit. Aceasta, la rândul său, este rezultatul unei gestionări mai simple a excepțiilor și a cazurilor de colț în Posit. În mod intuitiv, diferența de eficiență crește odată cu numărul de iterații. Figura 17 arată că *Posit(32,3)* atinge aceeași precizie ca *FP32* cu mai puține cicluri pe măsură ce numărul de iterații crește. **Nivelul doi.** *Posit(32,3)* și *Posit(16,2)* conduc la aceleași rezultate finale ca și *FP32* atunci când se execută benchmark-uri de nivel doi, economisind până la 6% din cicluri, așa cum se arată în Tabelul 16. Cu toate acestea, LR atât cu *Posit(8,1)* cât și cu *Posit(16,2)* prezintă o excepție în afara intervalului. Acest lucru se datorează faptului că unul dintre determinanții calculați de program depășește intervalul acestor două mărimi Posit. În plus, programele care operează cu *Posit(8,1)* produc rezultate greșite pe toate criteriile de referință, cu excepția CT. Acest lucru arată că Posit de dimensiuni mici nu este potrivit pentru anumite nuclee ML care necesită o precizie numerică ridicată. Această observație este în contrast cu unele dintre lucrările conexe [2, 3, 13]. Cu toate acestea, evaluarea se face pe un set de date diferit, și anume setul de date Iris. În testele următoare, *Posit(8,1)* se comportă mai bine pe un CNN parțial. Pe de altă parte, *Posit(16,2)* oferă o alternativă bună la reprezentările în virgulă mobilă pe 32 de biți. **Nivelul 3.** În comparație cu execuția de referință pe o gazdă x86/64, CNN-ul CIFAR-10 cu *FP32*, *Posit(32,3)* și *Posit(16,2)* care rulează pe FPGA cu un nucleu Rocket Chip prezintă aceeași precizie Top-1 ca și modelul de referință, și anume 68,15%. Chiar și *Posit(8,1)* atinge o precizie rezonabilă de 62,68%. În ceea ce privește viteza, toate cele trei reprezentări Posit sunt cu aproximativ 18% mai rapide decât execuția cu *FP32*. Rezultatele obținute cu *Posit(16,2)* și *Posit(8,1)* sunt promițătoare și deschid o serie de optimizări viitoare. De exemplu, aceste formate economisesc jumătate și trei sferturi din memoria necesară pentru reprezentarea intrărilor și a parametrilor în comparație cu *FP32* sau *Posit(32,3)* pe 32 de biți. Apoi, prin împachetarea a doi operanzi *Posit(16,2)* și patru operanzi *Posit(8,1)* per instrucțiune, timpul de execuție poate fi redus de două și, respectiv, de patru ori. O altă sursă de pierdere a preciziei este cauzată de sub sau supraîncărcarea în timpul execuției. De exemplu, stratul *prob* include, printre alte operații, exponențierea. Pe *Posit(8,1)*, exponențierea poate duce cu ușurință la o subîncărcare sau supraîncărcare. Pentru a testa această ipoteză, parametrii sunt păstrați în format posit pe 8 biți în memorie, dar *E-PAU* folosi *Posit(16,2)* și convertesc între aceste două formate în timpul execuției. Rezultatul este mai bun decât se aștepta, deoarece acuratețea Top-1 a acestei abordări este de 68,47%, mai mare decât acuratețea execuției de referință pe *FP32*. Acest rezultat confirmă ipoteza conform căreia sursa principală a inexactității din *Posit(8,1)* se află în timpul execuției și arată că utilizarea unei abordări hibride cu posturi de dimensiuni diferite poate economisi memorie fără a pierde din acuratețe.

7.1.3 Utilizarea resurselor

Ca o aproximare a suprafeței cipului ocupate de implementare, utilizarea resurselor FPGA a *E-PAU* este evaluată în comparație cu FPU original al Rocket Chip. Este evaluată utilizarea resurselor FPGA ale întregului sistem, și anume SiFive Freedom E310 cu un nucleu

Tabela 17: Utilizarea resurselorFPGA a întregului cip Rocket pe SiFive Freedom E310

Resource	FP32	Posit(8,1)	Posit(16,2)	Posit(32,3)
Logic LUT	29,335	19,367 (-34%)	25,598 (-13%)	38,155 (+30%)
FF	14,756	11,596 (-21%)	12,031 (-19%)	12,951 (-12%)
DSP	15	5 (-67%)	8 (-47%)	19 (+27%)
SRL	58	60	60	60
LUTRAM	924	924	924	924
BRAM	14	14	14	14

Rocket Chip cu un FPU/*E-PAU*, care rulează pe FPGA Arty A7-100T. În timp ce rezultatele de aici denotă economii în termeni de resurse din perspectiva FPGA, economii similare sau chiar mai mari în ceea ce privește suprafața vor fi obținute atunci când proiectul este implementat pe un ASIC [7, 14]. Economiile în ceea ce privește aria cipului sunt direct legate de economiile de energie statică și dinamică și, prin urmare, sunt esențiale pentru aplicațiile cu restricții de putere redusă, cum ar fi dispozitivele de margine bazate pe IoT. Tabelul 17 prezintă utilizarea diferitelor resurse FPGA în ceea ce privește atât implementările Posit, cât și *FP32*. Toate implementările utilizează aceeași cantitate de resurse de memorie (Shift-register Look up table – SRL, LUTRAM și BRAM), ceea ce indică faptul că comparația implică doar FPU modificat, restul sistemului fiind același în toate implementările. Pentru economii semnificative de suprafață și de energie fără pierderi mari de precizie, *Posit(16,2)* este o opțiune viabilă care economisește aproape 50% din DSP-uri, ceea ce se traduce prin unități de multiplicare-acumulare (MAC) într-un flux ASIC. Aceste economii de suprafață ar trebui să se traducă printr-o scădere de 50% a puterii, deoarece MAC-urile reprezintă o putere mai mare în comparație cu flop-urile sau alte elemente logice [8]. În schimb, *Posit(32,3)* utilizează cu 30% mai multe LUT-uri și 27% mai multă utilizare în comparație cu *FP32*. Aceste rezultate sunt mai proaste decât cele raportate în [4], care are nevoie de doar 4% mai multe LUT-uri în comparație cu FPU, dar similare cu cele raportate în [12]. Pe de altă parte, FPU-ul original al Rocket Chip este o lucrare în curs de desfășurare. Este posibil ca aceasta să nu implementeze toate cazurile limită ale standardului IEEE754. Cu toate acestea, utilizarea mai mare a resurselor de către *Posit(32,3)* poate fi contrabalansată de creșterea vitezei sale, ceea ce duce la o eficiență energetică și de timp mai mare în comparație cu *FP32*.

7.2 Perspective asupra rezultatelor procesoarelor

Acest capitol explorează oportunitatea de a înlocui standardul tradițional IEEE754 cu sistemul Posit nou propus [11] în contextul învățării automate la limită. Este prezentată implementarea propusă a unui *E-PAU* pentru a înlocui FPU-ul original într-un nucleu RISC-V. Aceasta este prima lucrare care face o evaluare amănunțită a sistemului Posit față de *FP32* pentru a determina (i) dacă este mai bună conversia hardware sau software între Posit și *FP32*, (ii) performanța timp-energie atât pentru nucleele matematice, cât și pentru cele de ML, și (iii) perspective privind alegerea lățimii de biți a sistemului Posit pentru diferite tipuri de aplicații. Implementarea este evaluată pe un FPGA utilizând platforma SiFive Freedom E310.

Acuratețea, eficiența în termeni de cicluri, utilizarea resurselor FPGA și puterea celor trei dimensiuni Posit sunt comparate cu cele ale unui IEEE754 pe 32 de biți, cu o singură precizie.

Spre deosebire de lucrările anterioare [2,3,13], Posit pe 8 biți nu produce acuratețea necesară pentru a înlocui *FP32* în aplicațiile ML obișnuite. Pe de altă parte, Posit pe 32 de biți nu prezintă îmbunătățiri spectaculoase în ceea ce privește eficiența față de *FP32*. Deși atinge aceeași precizie sau o precizie mai mare și poate accelera execuția atunci când programul are înmulțiri și diviziuni, are nevoie de aproximativ 30% mai multe resurse FPGA și utilizează cu 6% mai multă energie în comparație cu *FP32*. Cu toate acestea, Posit pe 16 biți prezintă cele mai bune rezultate. Chiar dacă prezintă o acuratețe mai mică în calculele științifice, acestea produc rezultate corecte pentru majoritatea nucleelor și aplicațiilor ML, necesitând în același timp mai puțină suprafață și energie în comparație cu *FP32*. De exemplu, *Posit(16,2)* atinge aceeași precizie Top-1 ca *FP32* pe un CNN CIFAR-10, prezentând în același timp o creștere de viteză de 18%.

8 CONCLUZIE

Teza de față abordează o infrastructură de cercetare pentru discuțiile privind sistemele de reprezentare a numerelor. Biblioteca software propusă reduce timpul și efortul de a propune un nou sistem de reprezentare a numerelor la implementarea reprezentării binare și a limitelor numerelor pe care le conține. Reperele și bibliotecile implementate prin interfața NRS-SL beneficiază de lucrul cu orice nou sistem de reprezentare a numerelor propus. Teza oferă o bibliotecă de metode statistice, repere de calcul științific, o funcție Fast-Fourier-Transform și două cadre de învățare automată de mică precizie. Toate acestea au fost utilizate pentru a evalua diferite sisteme de reprezentare a numerelor. Biblioteca de generatoare hardware îmbunătățește drumul de la o idee la mai multe unități de prototip hardware. Implementarea unui sistem de reprezentare a numerelor diminuează costul modulelor de codificare și decodificare. Împreună cu bibliotecile complementare din literatura de specialitate [17], cele două biblioteci îmbunătățesc infrastructura de cercetare. Noua infrastructură de cercetare ajută la discutarea sistemelor de reprezentare a numerelor prin reducerea semnificativă a nivelului de intrare pentru a propune și testa un nou sistem de reprezentare a numerelor.

Biblioteca software propusă a fost utilizată pentru adăugarea a trei noi sisteme de reprezentare a numerelor. MorrisBiasHEB combină avantajele virgulei flotante clasice și ale virgulei flotante conice și a obținut primul sau al doilea cel mai bun rezultat în cadrul analizelor de referință din literatura de specialitate. Este un concurent semnificativ în calculul general pentru IEEE754. MorrisUnaryHEB este un extraordinar sistem de virgulă mobilă conică definit doar prin mărimea $size$. Are cele mai exacte rezultate la adunare ($37,6\%$), numeroasele "zone de aur", o precizie zecimală mai bună la operațiile unare și o gamă dinamică mai mare decât IEEE754, Posit și MorrisBiasHEB. Aceste caracteristici îl fac un adversar formidabil în domeniul inteligenței artificiale. Adăugarea unui bit de exponent ascuns a deschis o nouă cale în dezvoltarea sistemelor de reprezentare a numerelor.

Nouă sisteme de reprezentare a numerelor cu precizie finită au fost evaluate folosind patru criterii de referință pentru calculul științific: înmulțirea matricelor, metoda gradientului conjugat, integrarea Simpson și simularea N-Body. Rezultatele obținute pentru numere mici validează "zona de aur" a sistemelor cu virgulă mobilă conică. Diferența dintre rezultatele IEEE754 și cele ale altor sisteme de reprezentare a numerelor nu este semnificativă, cu excepția simulării N-Body. În cazul simulării N-Body, sistemul Posit pe 32 de biți ($es = 2$) oferă cu două zecimale corecte în plus pentru numerele de mărime mică și cu una în plus pentru numerele de mărime mare față de IEEE754 pe 32 de biți. Nici mai puțin pe aplicațiile de calcul științific, IEEE754 ar putea rămâne standardul fără beneficii semnificative din partea implementării hardware a altor sisteme de reprezentare a numerelor.

Biblioteca de metode statistice oferă metode statistice de 14. Positul pe 16 biți ($es = 1$) prezintă rezultate similare cu omologii pe 32 de biți. Pe de altă parte, IEEE754 pe 16 biți (semiprecizie) nu poate produce un rezultat valid pe 6 dintre ele. Chiar și atunci când produce unul, are o precizie zecimală mai mică decât Posit pe 16 biți. Precizia zecimală se degradează odată cu numărul de intrări și cu valorile mai mari ale setului de date. Cititorul trebuie să înțeleagă că, contrar convingerilor, un rezultat al unei metode statistice pe un set de date mai extins poate fi mai departe de răspunsul corect decât se așteaptă. Sistemul de reprezentare a numerelor utilizat are un impact asupra acestuia. Posit oferă o acuratețe zecimală mai bună decât IEEE754 pentru toate metodele, cu excepția celor bazate ferm pe variață, cum ar fi testul T și varianța însăși.

Cele două cadre de învățare automată de precizie redusă ajută sistemele inteligente încorporate prin reducerea dimensiunii de stocare a unei rețele neuronale profunde. Acestea reduc dimensiunea cu $66,75\%$, respectiv $76,68\%$, cu o degradare a preciziei mai mică de 2%. Primul cadru (LPML) utilizează distilarea cunoștințelor și optimizarea preciziei straturilor cu un set limitat de sisteme de reprezentare a numerelor: generic Fixed-Point, Posit pe 8 biți ($es = 0$), Posit pe 16 biți ($es = 1$), Posit pe 32 de biți ($es = 2$), IEEE754 pe 32 de biți). Creșterea acurateței prin utilizarea distilării cunoștințelor în locul instruirii clasice este validată de o valoare de $0,87\%$. Rețeaua Resnet34 (acuratețe: $95,23\%$) este utilizată ca rețea profesor. Rețeaua studenților Resnet18 obține o acuratețe de $94,98\%$ după instruire (acuratețe clasică: $94,11\%$). Optimizarea preciziei straturilor reduce dimensiunea de stocare a rețelei Resnet18 cu $43,41\%$. Precizia finală este de $93,25\%$. Reducerea memoriei în comparație cu Resent34 este de $66,75\%$. Al doilea cadru (CPML) utilizează optimizarea uniformă și a preciziei pe straturi cu sisteme de reprezentare a numerelor implementate în NRS-SL. Optimizarea uniformă a preciziei reduce dimensiunea Resent18 (precizie: $57,01\%$) cu cel puțin $62,25\%$ (Morris) și cu un maxim de 75% (IEEE754) fără a pierde mai mult de 1% în precizie. Următoarea etapă de optimizare a preciziei straturilor nu este semnificativ de eficientă, iar costurile suplimentare generate de precizia diferită a straturilor ar putea depăși beneficiile.

Librăria generatoare de hardware generează unități funcționale ca FPU, KAU și module unare/-binare. Pentru sistemele de reprezentare a numerelor cu un interval dinamic ridicat, unitățile de acumulatori inexacte reprezintă o soluție dacă se acceptă eroarea nesemnificativă. Unitatea generală de virgulă mobilă (GFPU) poate funcționa simultan cu mai multe sisteme de reprezentare a numerelor ca operanzi sau rezultat. Costul hardware este triplu, iar viteza este la jumătate în comparație cu o FPU cu un sistem specific de reprezentare a numerelor. Beneficiile utilizării unui anumit sistem de reprezentare a numerelor pentru o anumită parte a aplicației ar putea depăși costul. În departamentul de 16 biți, bfloat16 de la Google prezintă cea mai bună performanță hardware într-o unitate de virgulă mobilă (mai puține LUTS, mai puține DPS-uri, viteză mai bună). Domeniul său de aplicare este reprezentat de aplicațiile de învățare automată. Pentru calculul general, MorrisBiasHEB și Posit sunt cele mai bune alternative. Cea mai bună alegere în departamentul de 32 de biți este între MorrisBiasHEB și Posit. Pentru acceleratoarele care utilizează în principal adunarea și înmulțirea, MorrisBiasHEB este cea mai bună decizie posibilă. Acesta are o precizie bună, cel mai bun timp de

calcul și cele mai mici resurse hardware necesare. Unitatea de acumulatori Kulisch pentru MorrisBiasHEB pe 16 biți are o frecvență maximă aproape dublă ($52,56$ MHz față de $28,54$ MHz) și cu aproape un sfert mai puține LUTS (1474 față de 1939) decât al doilea cel mai bun din această categorie.

Un sistem hardware-software complet pentru testarea unui sistem de reprezentare a numerelor poate oferi o perspectivă diferită. Posit a testat performanțele sale față de IEEE754 pe un CPU RochetChip [1]. Posit a validat așteptarea de a fi mai rapid decât IEEE754, dar cu costul resurselor hardware și al consumului de energie. Posit pe 16 biți prezintă o creștere de viteză de 18% pe cel de-al treilea nivel de referință, și necesită mai puține resurse hardware și consumă mai puțină energie.

Operații noi, alternative sau optimizate vor fi adăugate la biblioteca software în cadrul lucrărilor viitoare. Categoria sistemelor de reprezentare a numerelor de interval este o altă cale care merită parcursă, care poate oferi soluții pentru problemele pe care sistemul clasic de reprezentare a numerelor nu le poate rezolva. Paleta de unități pentru biblioteca de generatoare hardware va crește prin adăugarea de acceleratoare de inteligență artificială, acceleratoare de procesare digitală a semnalelor sau GPU-uri. Biblioteca de metode statistice trebuie să adauge mai multe metode care să fie testate. Reperele de calcul științific își pot crește complexitatea pentru a oferi perspective mai bune. Cadrele de precizie redusă permit testarea pe rețele mai complexe care nu sunt dedicate sistemelor integrate. Cea mai importantă activitate viitoare va fi crearea și evaluarea mai multor sisteme hardware-software pentru a dezvolta un adversar adecvat pentru IEEE754. MorrisBiasHEB are performanțe hardware și de calcul general excelente, care trebuie să fie investigate într-un SoC (System on the Chip) în cadrul lucrărilor viitoare. Pe o altă cale, MorrusUnaryHEB ar putea schimba domeniul calculului de inteligență artificială.

BIBLIOGRAFIE

[1] Krste Asanovic, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, et al. The Rocket Chip Generator. *University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, 2016.

[2] Zachariah Carmichael, Hamed F Langroudi, Char Khazanov, Jeffrey Lillie, John L Gustafson, and Dhireesha Kudithipudi. Deep positron: A deep neural network using the posit number system. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1421–1426, 2019.

[3] Zachariah Carmichael, Hamed F Langroudi, Char Khazanov, Jeffrey Lillie, John L Gustafson, and Dhireesha Kudithipudi. Performance-efficiency trade-off of low-precision numerical formats in deep neural networks. In *Proceedings of the Conference for Next Generation Arithmetic*, pages 1–9, 2019.

[4] Rohit Chaurasiya, John L. Gustafson, Rahul Shrestha, Jonathan Neudorfer, Sangeeth Nambiar, Kaustav Niyogi, Farhad Merchant, and Rainer Leupers. Parameterized Posit Arithmetic Hardware Generator. In *Proc. of 36th IEEE International Conference on Computer Design*, pages 334–341, 2018.

[5] Jose Cintra. Calculating the Number PI Through Infinite Sequences. http://archive.today/2Nf1G, 2014.

[6] Florent De Dinechin, Luc Forget, Jean-Michel Muller, and Yohann Uguen. Posits: the good, the bad and the ugly. In *Proceedings of the Conference for Next Generation Arithmetic*, pages 1–10, 2019.

[7] Andreas Ehliar and Dake Liu. An ASIC Perspective on FPGA Optimizations. In *Proc. of International Conference on Field Programmable Logic and Applications*, pages 218–223, 2009.

[8] James Garland and David Gregg. Low Complexity Multiply-accumulate Units for Convolutional Neural Networks with Weight-sharing. *ACM Transactions on Architecture and Code Optimization*, 15(3):1–24, 2018.

[9] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5–48, 1991.

[10] John L. Gustafson. *The End of Error: Unum Computing*. Chapman & Hall/CRC Computational Science. Taylor & Francis, 2015.

[11] John L Gustafson and Isaac T Yonemoto. Beating floating point at its own game: Posit arithmetic. *Supercomputing Frontiers and Innovations*, 4(2):71–86, 2017.

[12] M. K. Jaiswal and H. K. . So. Universal Number Posit Arithmetic Generator on FPGA. In *Proc. of Design, Automation Test in Europe Conference Exhibition*, pages 1159–1162, 2018.

[13] Jeff Johnson. Rethinking floating point for deep learning. *arXiv preprint arXiv:1811.01721*, 2018.

[14] I. Kuon and J. Rose. Measuring the Gap Between FPGAs and ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):203–215, 2007.

[15] Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Teman, Xiaobing Feng, Xuehai Zhou, and Yunji Chen. PuDianNao: A Polyvalent Machine Learning Accelerator. In *Proc. of 20th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 369–381, 2015.

[16] Robert Morris. Tapered floating point: A new floating-point representation. *IEEE Transactions on Computers*, 100(12):1578–1579, 1971.

[17] E. Theodore L. Omtzigt, Peter Gottschling, Mark Seligman, and William Zorn. Universal Numbers Library: design and implementation of a high-performance reproducible number systems library. *arXiv:2012.11011*, 2020.