SISTEME DE PROCESARE LOCALĂ A ALGORITMILOR DE ÎNVĂȚARE AUTOMATĂ

Teză de doctorat – Rezumat

pentru obținerea titlului științific de doctor la

Universitatea Politehnica Timișoara

în domeniul de doctorat Inginerie Electronică, Telecomunicații și Tehnologii Informaționale

autor ing. Ioan LUCAN-ORĂȘAN

conducător științific Prof. Univ. Dr. Habil. Ing. Cătălin Daniel CĂLEANU

luna Septembrie anul 2024

Inteligența artificială este un domeniu de cercetare amplu cu o dezvoltare explozivă în ultimii ani, iar astăzi tot mai prezent pentru o multitudine de aplicații din lumea reală. Acest lucru înseamnă faptul că domeniul a ajuns într-o etapă suficient de avansată astfel încât a trecut de la nivel de cercetare la cazuri concrete de utilizare. Pe măsură ce algoritmii de inteligență artificială au ajuns la un nivel ridicat de maturitate, atenția cercetătorilor și a industriei a început să migreze către dispozitivele hardware locale („embedded") cu capabilități de execuție a acestora. Lucrarea de doctorat propune soluțiilor tehnice de bază pentru compresia rețelelor neuronale profunde cu scopul de a obține un model de rețea neuronală compatibil cu sistemele locale încorporate. Platforma hardware țintă este constituită de către microcontrolere cu CPU ARM, seria Cortex-M, datorita faptului că acestea arhitectură este înglobată în peste 90% din aplicațiile mobile curente.

Obiectivul general al tezei este reprezentat de către compresia rețelelor neuronale convoluționale și particularizarea cercetărilor pentru cazul unei aplicații din domeniul auto, și anume estimarea direcției privirii unui conducător auto. Acest lucru are o importanță deosebită din cauza dimensiunilor mari ale rețelelor neuronale de ultimă generație, care sunt antrenate și rulate pe stații de lucru sau servere la distanță (în nor) cu performanțe computaționale ridicate. O astfel de soluție nu este potrivită pentru aplicații care trebuie să ruleze în timp real, cu consum redus de energie, costuri reduse, și unde transmiterea datelor la distanță trebuie evitată din motive de confidențialitate și securitate. Prin urmare, compresia rețelelor neuronale are un rol important, cu scopul de a reduce dimensiunea unui model astfel încât să fie posibilă execuția inferenței pe dispozitive locale de procesare în timp real și cu consum redus de putere.

Contribuția fundamentală a acestei teze este studiul, descrierea și implementarea metodelor principale de compresie a rețelelor neuronale convoluționale împreună cu validarea acestora folosind diferite metrici de raportare pe microcontrolere din familia STM32 pe 32 de biți cu CPU ARM Cortex-M. În caz particular, metoda de distilare a cunoștințelor este pe larg explorată folosind compresia la nivel de straturi sau de filtre, precum și combinația acestora cu o prezentare detaliată a rezultatelor obținute.

Teza este organizată pe șase capitole și mai multe subcapitole, așa cum se prezintă în Figura 1. Primele trei capitole fac referire la partea introductivă și noțiuni teoretice generale, iar ultimele trei capitole prezintă experimentele și rezultatele esențiale din această lucrare.

Capitolul 1 prezintă motivația pentru elaborarea acestei lucrări, obiectivele stabilite inițial și o scurtă prezentare de nivel înalt cu privire la structura lucrării.

Capitolul 2 urmărește introducerea sub formă de noțiuni generale a unor informații teoretice care fac referire la cele mai comune arhitecturi de rețele neuronale profunde. Ulterior,

se pune accent pe complexitatea arhitecturilor descrise în contextul rulării acestora pe dispozitive hardware cu resurse limitate, fapt ce conduce la nevoia de a defini arhitecturi de rețele neuronale profunde pentru astfel de dispozitive. În consecință, urmează o scurtă descriere a acestor arhitecturi specifice împreună cu o serie de concluzii.

Capitolul 3 prezintă posibilitățile de execuție a procesului de inferență pentru modele de rețele neuronale profunde, cu scopul de a evidenția avantajele de rulare a inferenței pe dispozitive locale față de sistemele de calcul la distanță (în nor).

Capitolul 4 este primul capitol care aduce contribuții originale pentru această lucrare. Acesta prezintă o serie de dispozitive încorporate potrivite pentru rularea procesului de inferență a modelelor de rețele neuronale profunde împreună cu cele mai comune programe software și biblioteci care ajută la conversia modelelor într-un format compatibil cu dispozitivele încorporate. Ulterior, urmează o descriere a dispozitivelor încorporate cu CPU ARM Cortex-M utilizate în această lucrare împreună cu principalele avantaje ale acestora. Ultima parte din acest capitol cuprinde o analiză amănunțită a unui număr mare de lucrări publicate în ultimii ani care utilizează rețele neuronale profunde pentru diferite aplicații și care folosesc microcontrolere bazate pe CPU ARM Cortex-M. În final, sunt prezentate rezultate și concluzii importante cu privire la diferite aspecte, cum ar fi: arhitecturile utilizate, precizia obținută, caracteristici hardware utile, provocări și oportunități de cercetare.

Capitolul 5 prezintă în prima parte noțiunile teoretice cu privire la cele mai comune metode de compresie a rețelelor neuronale profunde: (1) cuantizarea parametrilor după procesul de antrenament sau în timpul acestuia, (2) eliminarea ponderilor sau a altor elemente structurale și (3) distilarea cunoștințelor. Adițional, se prezintă pe scurt și conceptele teoretice pentru (4) optimizarea modelelor în funcție de o anumită platformă hardware și (5) căutarea automată a unei arhitecturi optime. Capitolul se termină cu o prezentare detaliată a rezultatelor și concluziilor importante obținute în urma publicării a trei lucrări științifice despre cuantizare, eliminarea parametrilor și, respectiv, distilarea cunoștințelor.

Ultimul capitol prezintă concis principalele rezultate și contribuțiile personale ale autorului cu referire la diseminarea acestora prin intermediul publicațiilor științifice. Acesta se încheie cu prezentarea viitoarelor direcții de cercetare.

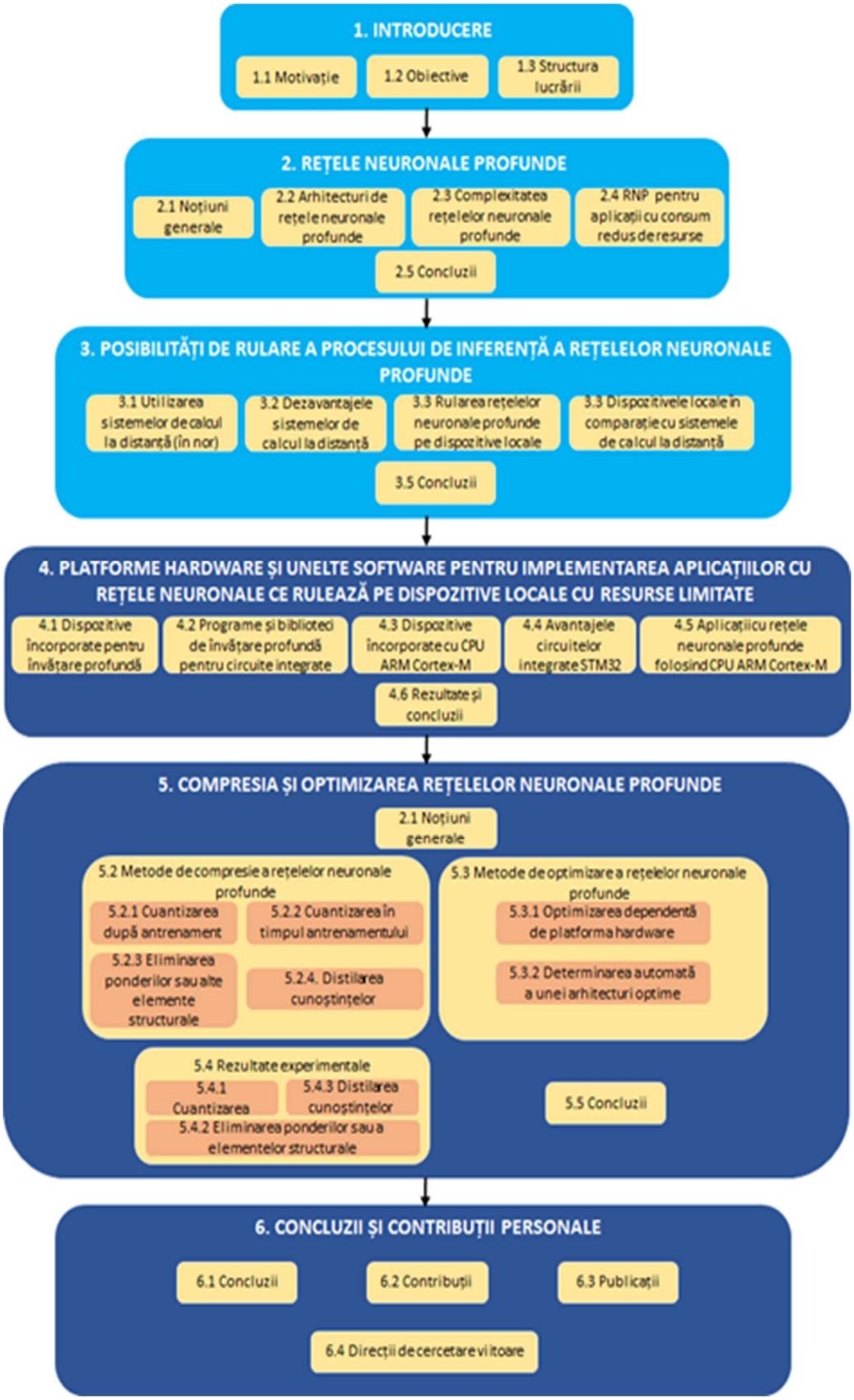

Figura 1. Structura lucrării pe capitole și subcapitole.

1. REȚELE NEURONALE PROFUNDE

Rețelele Neuronale Convoluționale *(„Convolutional Neural Networks", CNNs)* au fost folosite încă din anul 1980, când a fost proiectată o rețea neuronală profundă cu denumirea Neocognitron [1]. Folosind această arhitectură ierarhică de tip multistrat, a fost posibilă recunoașterea tiparelor vizuale.

CNN-urile sunt un tip specializat de rețele neuronale pentru procesarea datelor care au o topologie cunoscută, asemănătoare unei grile. Exemple de astfel de date pot fi cele din serii de timp, care pot fi reprezentate ca o grilă 1D care citește eșantioane la intervale regulate de timp sau date dintr-o imagine, care pot fi reprezentate ca o grilă 2D de pixeli. Rețelele convoluționale au avut un succes extraordinar pentru o mulțime de aplicații practice. Numele de "rețea neuronală convoluțională" indică faptul că rețeaua folosește o operație matematică numită convoluție. Convoluția este un tip specializat de operație liniară. Rețelele convoluționale sunt pur și simplu rețele neuronale care folosesc convoluția în locul multiplicării generale matriceale în cel puțin unul dintre straturile lor [2].

Rețelele de tip CNN sunt cel mai comun utilizate pentru probleme de clasificare, de exemplu imagini, și au o funcționare relativ similară creierul uman. CNN-urile „învață" conținutul unei imagini prin aplicarea unor filtre imaginii și prin procesarea metodelor diferitelor dimensiuni ale filtrului, cantității și operațiilor neliniare. Aceste filtre și operații sunt aplicate pe mai multe straturi, astfel încât dimensiunile spațiale ale fiecărui strat ulterior să scadă și adâncimile acestora să crească în timpul procesului de transformare a imaginii. Pentru fiecare filtrare aplicată, profunzimea conținutului învățat crește. Aceasta începe cu detectarea marginilor/muchiilor, urmată de recunoașterea formelor, apoi o colecție de forme numite entități și așa mai departe. Acest lucru este analog cu funcționarea cortexului vizual [3].

Figura 2 prezintă diagrama pentru un model tipic CNN și componentele acestuia. În primul pas, operația de convoluție folosind un număr de 64 de filtre cu dimensiunea 5 x 5 este aplicată unei imagini în tonuri de gri cu dimensiunea 48 x 48 pentru a produce o harta de caracteristici de număr și dimensiune 44 x 44 x 64. La pasul următor, operația de agregare („Max pooling") cu un filtru de dimensiunea 5 x 5 și pasul setat la 2 se aplică formei de intrare 44 x 44 x 64. Rezultatul acestei operații este o formă de ieșire redusă la dimensiunea 20 x 20 x 64. După prima operație de convoluție 2D, este aplicată funcția de activare neliniară ReLu și continuă după fiecare operație pană la ultimul strat complet conectat.

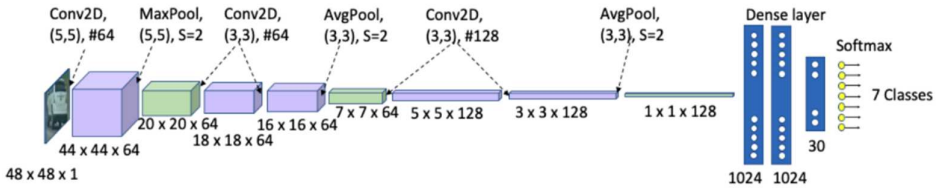

Figura 2. Arhitectura CNN tipică. Sursa [3].

Rețelele Neuronale Recurente *(„Recurrent Neural Networks", RNNs)* prezentate în lucrarea lui Rumelhart și colab. [4] din anul 1986 sunt o familie de rețele neuronale utilizate în principal pentru a detecta modele într-o secvență de date. La fel cum o rețea convoluțională este o rețea neuronală specializată pentru procesarea unei grile de valori *X*, cum ar fi o imagine, o rețea neuronală recurentă este o rețea neuronală specializată pentru procesarea unei secvențe de valori $x^{(1)}, \ldots, x^{n}$. În mod similar, așa cum rețelele convoluționale pot fi adaptate cu ușurință la imagini de dimensiuni mari, iar unele rețele convoluționale pot procesa imagini de dimensiuni variabile, rețelele recurente pot fi adaptate la secvențe mult mai lungi decât ar fi practic posibil

cu alte rețele nespecializate pentru date secvențiale. De asemenea, majoritatea rețelelor recurente pot procesa și secvențe de lungime variabilă [2].

O caracteristică principală a acestor rețele este posibilitatea de a suporta corelații de natură temporală între tiparele secvențiale de intrare. Acest lucru este realizabil datorită propagării înapoi a semnalului de ieșire *(„feedback")*, prin urmare semnalele de ieșire depind de valorile anterioare ale acestora. Figura 3 prezintă structura de bază a unei rețele neuronale recurente.

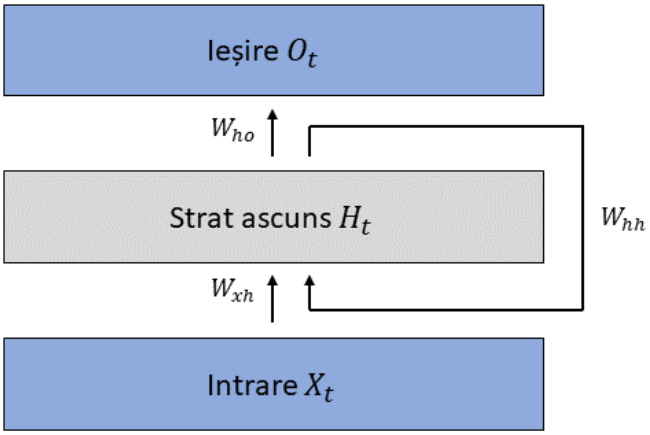

Figura 3. Rețea neuronală recurentă.

Cele mai frecvent utilizate tipuri de rețele neuronale recurente sunt: *(„Long short-term memory", LSTM)* și *(„Gated recurrent unit", GRU)*. Modelul LSTM a fost introdus în anul 1997 de către S. Hochreiter și J. Schmidhuber [5] și este caracterizat de o capacitate mai bună de memorare față de rețelele precedente, de patru ori mai multă memorie în comparație cu RNNs convenționale. Rețelele de tip GRU sunt modele care permit salvarea mai multor informații din trecut cu scopul de a obține o predicție mai bună.

Arhitectura ResNet a fost introdusă de către Kaiminh He și colab. în lucrarea [6] cu scopul principal de a rezolva problema de degradare a preciziei rețelelor neuronale profunde pe măsura ce adâncimea lor crește. Această degradare nu este o consecință de suprapotrivire *(„overfitting")*, motivul este faptul că după o anumită profunzime critică, ieșirea pierde informațiile de intrare, astfel încât corelația dintre intrare și ieșire începe să devieze, rezultând o scădere a preciziei.

Un ansamblu de șase modele cu diferite profunzimi au obținut o eroare de validare în top cinci de 3,57% și au câștigat primul loc la ILSVRC-2015 *(„ImageNet Large-Scale Visual Recognition Competition", ILSVRC)*. ILSVRC este o competiție pentru evaluarea algoritmilor de detecție a obiectelor și clasificarea imaginilor din 2010 până în 2017. Un exemplu de model ResNet este prezentat în Figura 4.

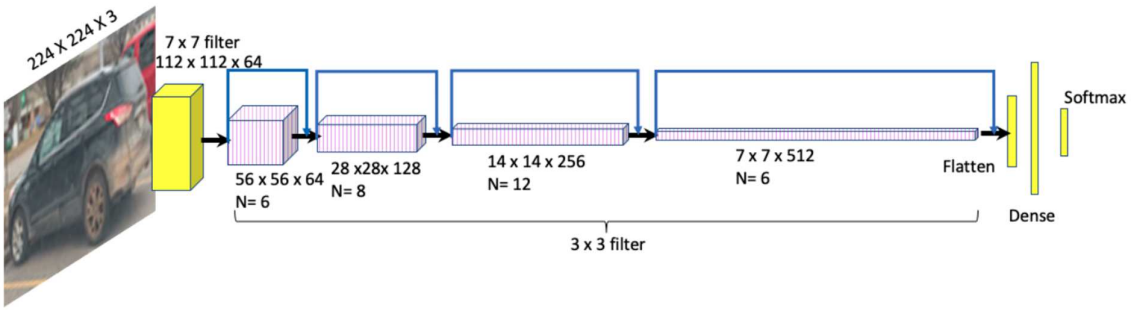

Figura 4. Arhitectura ResNet. Sursa [3].

2. EXPERIMENT DEMONSTRATIV

A. Prezentare

În lucrarea [7] am implementat și evaluat un model preantrenat CNN pe placa de evaluare STM32F779I-EVAL, fără ajutorul unui sistem de operare („bare metal”), de cost redus și eficientă din punct de vedere energetic, care funcționează cu un CPU ARM Cortex-M7. Acesta este un model de dimensiuni reduse (Figura 5) format din numai 3 straturi convoluționale și a fost antrenat folosind setul de date CIFAR-10 pentru problema de clasificare a imaginilor în 10 clase de ieșire (avion, automobil, pasăre, pisică, cerb, câine, broască, cal, navă și camion).

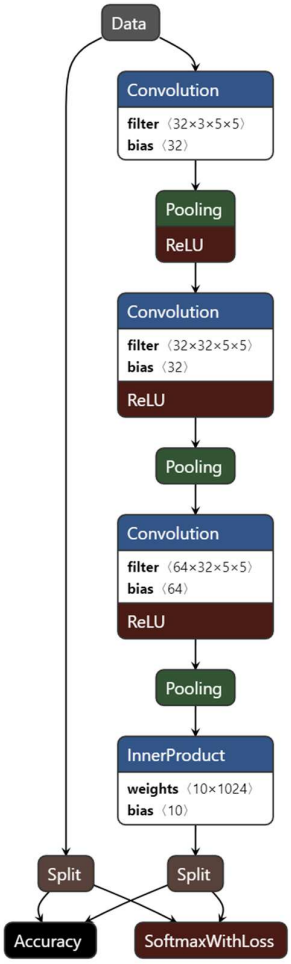

Figura 5. Diagrama bloc a modelului CNN *[7]*.

Pentru implementarea aplicației am utilizat o serie de dispozitive periferice:

- Dispozitive periferice disponibile pe placă: modulul cu cameră și modulul de afisaj LCD *(„Liquid-Crystal Display”, LCD);*
- Dispozitive periferice interne: DCMI *(„Digital Camera Module Interface”, DCMI),* acceleratorul "Chrom-ART" *(„Direct Memory Access 2D”, DMA2D).*

Modulul cu cameră este utilizat pentru achiziționarea imaginilor și modul de afișaj LCD pentru afișarea rezultatului după procesul de inferență. Dispozitivul intern DCMI este folosit pentru a asigura interfața cu modulul cu cameră, iar acceleratorul "Chrom-ART" DMA2D

pentru tranferul eficient al datelor.

Am implementat două variante de program: (1) un program care să achiziționeze imaginea de intrare de la modulul cu cameră și (2) un program care să testeze modelul CNN folosind subsetul de date pentru test CIFAR-10.

Folosind prima variantă de program am obținut rezultate pozitive așa cum se prezintă în Figura 6.

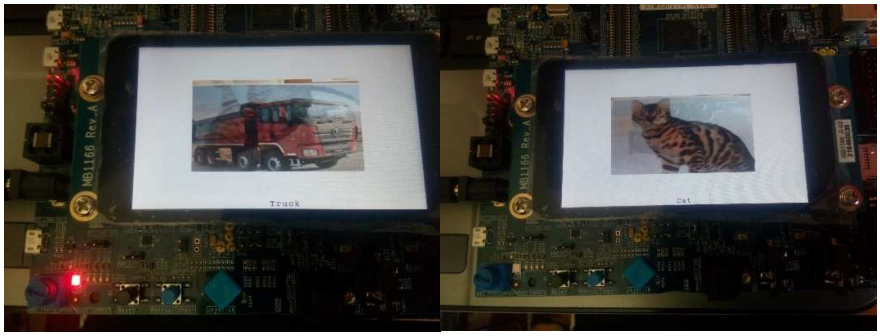

Figura 6. Rezultate obținute pentru imagini de testare achiziționate folosind modulul cu cameră *[7]*.

Pentru a doua variantă de program am folosit numărul total de imagini pentru test (10000) cu 1000 de imagini pentru fiecare clasă. Acestea le-am stocat pe un card microSD, organizate separat pe fișiere specifice fiecărei clase. Imaginile sunt citite de microcontroler prin intermediul dispozitivului periferic aferent pentru interfața cu cardul microSD („SD/SDIO/MMC"). Ulterior, se decodifică și testează pe rând câte o imagine din fiecare clasă. După fiecare test se verifică rezultatul obținut, iar rezultatele pozitive sunt stocate în memoria internă. După ce toate imaginile au fost testate, rezultatele stocate sunt salvate pe cardul microSD în format ".csv". Tabelul 1 prezintă acuratețea obținută pentru fiecare clasă.

Tabelul 1. Rezultatele de precizie obținute cu modelul CNN

Clasa de imagini	**Numărul de rezultate pozitive (%)**
Avion	657 (65,7)
Automobil	755 (75,5)
Pasăre	483 (48,3)
Pisică	494 (49,4)
Cerb	552 (55,2)
Câine	470 (47)
Broască	489 (48,9)
Cal	591 (59,1)
Navă	690 (69)
Camion	761 (76,1)

B. Concluzii

Datorită tehnologiei avansate care integrează funcțiile de procesare digitală a semnalului DSP cu suport hardware pentru o gamă largă de microcontrolere și GPU-uri, calculele generale și diverse funcții matematice pot fi implementate cu un număr mic de instrucțiuni și executate

într-un timp semnificativ mai scurt [8], [9].

Seriile de microcontrolere cu CPU ARM Cortex-M oferă utilizatorilor un set de funcții generale bazate pe DSP, cu intenția de a acoperi o gamă largă de cerințe pentru implementarea rețelelor neuronale. Biblioteca care cuprinde acest set de funcții se numește CMSIS-NN [10]. Unul dintre exemplele folosite pentru a demonstra acest set de funcții este o rețea neuronală convoluțională numită CIFAR 10, numele fiind direct legat de setul de date utilizat pentru antrenament. În lucrarea publicată [7], am folosit exemplul menționat cu scopul de a rula o rețeaua neuronală convoluțională pe platforma hardware de la ST, STM32F779I-EVAL.

Precizia generală obținută este în medie de 59,42% cu un timp de execuție de 77ms (aproximativ 13 cadre pe secundă). Deși precizia este încă departe de alte modele de ultimă generație optimizate pentru dispozitive încorporate, de exemplu o precizie de 92,4% așa cum este raportată în lucrarea [11], dimensiunea modelului folosit de mine este mult mai mică (140 kB în cazul față de 4,3 MB din [11]).

3. CUANTIZAREA POST-ANTRENAMENT

A. Prezentare

Cuantizarea este una dintre metodele utilizate pe scară largă pentru a reduce dimensiunea unui model. Acest lucru se realizează prin modificarea formatului numeric folosit pentru reprezentarea parametrilor. De exemplu, formatul de reprezentare poate fi modificat de la virgulă mobilă pe 32 de biți la un număr întreg pe doar 8 biți sau chiar mai puțini. Metoda de cuantizare este frecvent utilizată pentru compresia DNN-urilor, în special pentru a facilita rularea inferenței pe dispozitive de cost redus, cum sunt microcontrolerele pe 32 de biți. Chiar dacă sunt microcontrolere care dețin o unitate FPU de precizie, utilizarea acesteia este de obicei evitată pentru a reduce consumul de memorie și energie [12]. În același timp, există totuși multe microcontrolere fără o unitate FPU.

În lucrarea [13] am implementat și evaluat metodele de cuantizare post-antrenament disponibile folosind biblioteca TensorFlow Lite: (1) cuantizare pe interval dinamic (PTDRQ), (2) cuantizare completă pe numere întregi (PTIQ), (3) cuantizare în virgulă mobilă pe 16 biți (PTFQ) și (4) cuantizare pe numere întregi cu activări pe 16 biți (PTIQA).

Am folosit trei modele de rețele neuronale convoluționale cu diferite dimensiuni: DenseNet121 (7.0 MB) [14], ResNet50 (23.5 MB) [6] și SE-PreAct-ResNet101 (42.5 MB) [15]. Am ales aceste modele pentru a urmării impactul asupra preciziei și raportul de compresie în funcție de dimensiune. Pentru antrenament, am folosit setul de date CIFAR-10 cu sarcina de clasificare a imaginilor.

Toate cele patru metode de cuantizare le-am implementat folosind un flux de execuție similar, așa cum este descris în diagrama bloc prezentată în Figura 7. După etapa de antrenament, modelele au fost salvate în format SavedModel, fiind un format compatibil pentru conversia TensorFlow Lite. După antrenarea modelului (Pasul 1), în Pasul 2 modelul cu format SavedModel este convertit la format TensorFlow Lite pentru a utiliza dimensiunea acestuia ca punct de referință pentru determinarea raportului de compresie. În Pasul 3, modelul este convertit în format TensorFlow Lite cu opțiunea de cuantizare activată. În Pasul 4, raportul de compresie este calculat ca raport dintre dimensiunea inițială a modelului în virgulă mobilă (calculată la Pasul 2) și dimensiunea modelului după cuantizare. În ultima etapă (Pasul 5) modelul cuantizat este testat folosind imaginile de test CIFAR-10, ulterior fiind calculată precizia.

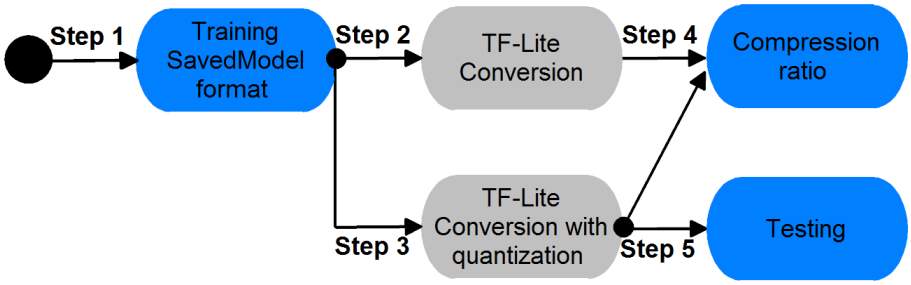

Figura 7. Diagrama bloc pentru cuantizarea postantrenament TensorFlow Lite *[13]*.

B. Concluzii

Figura 8 prezintă raportul de compresie obținut pentru fiecare model DNN după aplicarea metodelor de cuantizare. Deoarece modelele originale în virgulă mobilă folosesc o reprezentare pe 32 de biți și cuantizarea este efectuată pe o reprezentare de 8 biți, este de așteptat să se obțină un raport de compresie de 4×. Pe baza rezultatului obținut, raportul de compresie este puțin mai mic decât 4×, iar folosind un model DNN cu o dimensiune mică (de exemplu, DenseNet121) raportul are tendința să fie mai mic față de celelalte, în special pentru cuantizarea pe interval dinamic și cuantizarea pe numere întregi cu activări pe 16 biți. Folosind metoda de cuantizare în virgulă mobilă pe 16 biți, raportul de compresie este de 2×. Pentru această metodă, diferența dintre modelele DNN este nesemnificativă. În concluzie, raportul de compresie folosind metodele de cuantizare TensorFlow Lite variază puțin în funcție de dimensiunea modelului.

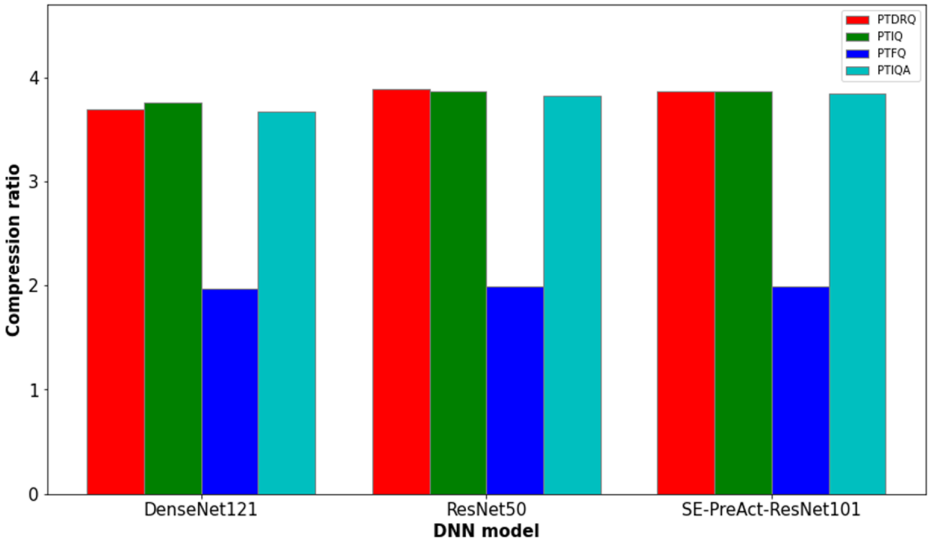

Figura 8. Raportul de compresie *[13]*.

Tabelul 2 prezintă precizia obținută după fiecare metodă de cuantizare, împreună cu precizia înainte de cuantizare și numărul total de parametri. La modul general, pierderea de precizie este mai mică de 1%. Prin urmare, metodele de cuantizare post-antrenament nu afectează într-un mod semnificativ precizia.

Mai precis, folosind cuantizarea în virgulă mobilă pe 16 biți pierderea de precizie este nesemnificativă, în timp ce dimensiunea modelului este redusă cu 50%. În unele cazuri, de exemplu pentru DenseNet121 și ResNet50 precizia este chiar îmbunătățită. Folosind tehnica PTIQA se obține o creștere a preciziei de 0,01% pentru DenseNet121, iar cu tehnica PTDRQ se înregistrează o creștere a preciziei de 0,02% pentru ResNet50. Pentru SE-PreAct-ResNet101, cea mai mică pierdere de precizie este obținută folosind tehnica PTFQ urmată de PTIQA.

Folosind cuantizarea completă pe numere întregi, se înregistrează cea mai mare scădere a preciziei. În același timp, totuși o scădere mică. În cel mai rău caz, aceasta este de doar 0,43% pentru modelul DenseNet121. Un rezultat diferit este obținut pentru SE-PreAct-ResNet101, unde cea mai mică precizie a fost înregistrată folosind cuantizarea pe interval dinamic.

Tabelul 2. Rezultatele de precizie înainte și după cuantizare.

Caracteristici/Model	**Modelul DNN**		
	DenseNet121	***ResNet50***	***SE-PreAct-ResNet101***
Dimensiunea modelului	7.0 MB	23.5 MB	42.5 MB
Precizia inițială	95.51%	95.13%	94.76%
PTDRQ	95.47%	95.15%	**94.63%**
PTIQ	**95.08%**	**95.03%**	94.69%
PTFQ	95.51%	95.13%	94.75%
PTIQA	95.52%	95.12%	94.72%

În lucrarea [13] am comparat diferite metodele de cuantizare post-antrenament folosind trei modele diferite de rețele neuronale convoluționale. Acestea sunt clasificate în funcție de dimensiunea lor ca fiind mici, medii și mari. Raportul de compresie folosind cuantizarea pe 8 biți este puțin sub 4×, iar folosind cuantizarea în virgulă mobilă pe 16 biți este de 2×. Am observat faptul că pentru categoria de dimensiuni mici, raportul de compresie tinde să fie mai mic. Degradarea preciziei este aproape nesemnificativă; în cel mai rău caz, impactul este de doar 0,43% pentru categoria de dimensiuni mici atunci când se folosește metoda PTIQ.

4. ELIMINAREA PONDERILOR ÎN FUNCȚIE DE MAGNITUDINE

A. Prezentare

S-a observat faptul că anumite elemente structurale dintr-o rețea sunt redundante și contribuția lor la predicția finală este mică. Pe baza acestei observații, s-a ajuns la concluzia că eliminarea acestor elemente este posibilă pentru a reduce dimensiunea unui model, fără un impact semnificativ asupra preciziei. Astfel de exemple pot fi: eliminarea conexiunilor, straturilor convoluționale sau neuronilor din straturile complet conectate. O provocare care intervine cu această tehnică este găsirea unei strategii potrivite pentru identificarea elementelor ce pot fi eliminate cu o degradare minimă asupra performanței rețelei.

Spre deosebire de cuantizare, eliminarea este o tehnică care vizează problema supra-parametrizării rețelelor neuronale. Se cunoaște faptul că dimensiunea unei rețele neuronale nu este într-o corespondență completă cu performanța sa, prin urmare, eliminarea acestor parametri sau elemente de structură va rezulta într-o dimensiune mai mică a rețelei, fără a afecta performanța. Pe lângă alte metode de compresie, aceasta vine ca o strategie esențială pentru reducerea dimensiunii modelului față de nevoia tot mai mare de implementare a modelelor de învățare profundă pe dispozitive cu resurse limitate.

În acest sens, mi-am propus să folosesc modelul profesor din lucrarea [16] pentru implementarea tehnicilor de eliminare a ponderilor bazate pe magnitudine. Un model cu matrice de ponderi care conține mai multe valori de 0 poate fi mai eficient în ceea ce privește amprenta de memorie și timpul de inferență. Cu toate acestea, dispozitivul hardware utilizat poate impune

anumite limitări și operațiile care implică ponderile de valori 0 să nu fie optimizate. În această perspectivă, acceleratoarele hardware dedicate au făcut progrese în ultimii ani. Studiul recent al lui V Isaac–Chassande și colab. [17] prezintă o imagine de ansamblu și detalii de proiectare a acceleratoarelor hardware dedicate de ultimă generație pentru calculul cu matrice de valori 0 în mod aleator. Modelul profesor este conceput pentru estimarea direcției privirii ochilor în aplicații din domeniul auto. În contextul interacțiunii om-calculator, domeniul de estimare a privirii este foarte important. În prezent, neatenția șoferului este încă un element major care contribuie la coliziunile auto. Caracteristicile sistemelor avansate de asistență pentru șofer *(„ADAS")* au fost special concepute pentru a atenua incidența accidentelor auto cauzate de neatenția șoferului. Estimarea direcției privirii șoferului este considerată o intrare principală pentru algoritmii care pot recunoaște lipsa de atenție în timpul conducerii.

Setul de instrumente pentru optimizarea unui model TensorFlow [18] oferă soluții pentru eliminarea ponderilor în funcție de magnitudine. Folosind această metodă de eliminare, ponderile care au valori mici sunt setate la zero cu scopul de a elimina conexiunile inutile. Aceasta se aplică în timpul procesului de antrenament pentru a permite rețelei neuronale să se adapteze la schimbările provocate. Procesul de antrenare poate fi doar un reglaj fin și nu este obligatoriu să fie efectuat de la zero. Utilizarea unui model preantrenat și aplicarea unui pas de reglaj fin este de regulă o opțiune mai bună.

Programarea eliminării poate fi configurată ca eliminare constantă sau folosind o funcție de descreștere polinomială. Cu prima configurație, o valoare țintă este definită ca procent din ponderi care vor fi zero. Folosind funcția de descreștere polinomială, programarea de eliminare se efectuează în conformitate cu funcția de descreștere polinomială prin specificarea valorii inițiale și finale.

Scopul acestei lucrări este de a evalua și compara diferite scheme de eliminare, cum ar fi programările de eliminare constantă și funcția de descreștere polinomială pentru eliminarea ponderilor în funcție de magnitudine, utilizând setul de instrumente de optimizare a unui model TensorFlow și o arhitectură CNN personalizată pentru problema de estimare privirii ochilor.

B. Concluzii

Programarea de eliminare constantă se aplică folosind o valoare de final în intervalul [0,10 ... 0,89] cu pasul de incrementare 0,01. Acest lucru înseamnă faptul că modelul este supus la 80 de configurații diferite și bucle de antrenament. O buclă de antrenament este efectuată timp de 35 de epoci. Programarea folosind funcția de descreștere polinomială are o configurație similară cu eliminarea constantă, cu o valoare de final în același interval [0,10 ... 0,89] și pasul de 0,01, cu același număr de bucle și epoci de antrenament. Valoarea de început este setată la 0,00 pentru fiecare buclă de antrenament.

Tabelul 3 prezintă rezultatele obținute pentru programarea de eliminare constantă. Acesta cuprinde precizia maximă de validare, precizia maximă de testare, valoarea finală de eliminare și raportul de compresie la care am obținut precizia maximă de testare. Experimentele au fost efectuate de trei ori pentru a calcula o medie a rezultatelor și pentru a evidenția comportamentul general. Rezultatul mediu pentru precizia de validare este de 75,06%, în timp ce pentru precizia de testare este de 73%. Considerând precizia de testare a modelului profesor de 74,48% [16], am obținut o valoare de eliminare finală medie de 0,73 și un raport de compresie de 7,74 cu o mică scădere a preciziei de numai 1,48%.

Cel mai bun rezultat a fost obținut pentru al doilea experiment. În acest caz, degradarea preciziei este de numai 1,08%, valoarea de eliminare finală este de 0,75, în timp ce raportul de compresie este de 8,06.

Tabelul 3. Rezultatele folosind programarea de eliminare constantă.

Rezultat	Eliminarea constantă		
	Experiment1	***Experiment2***	***Experiment3***
Precizia de validare	75.06%	75.23%	74.89%
Precizia de testare	72.47%	73.40%	73.15%
Valoarea de eliminare finală	0.76	0.75	0.68
Raportul de compresie pentru precizia de test	8.38	8.06	6.79

Tabelul 4 prezintă rezultatele obținute pentru programarea de eliminare folosind funcția de descreștere polinomială. Rezultatele sunt prezentate utilizând indicatorii de performanță anterior menționați. Cu această configurație, rezultatul mediu pentru precizia de validare este de 73,8%, iar media pentru precizia de testare este de 72,9%. Aceste rezultate au fost obținute cu o valoare finală de eliminare medie de 0,57 și raportul de compresie 5,52.

În acest caz, cel mai bun rezultat a fost obținut pentru primul experiment, cu o degradare a preciziei de 0,74%, o valoare finală de eliminare de 0,59 și un raport de compresie de 5,69.

Folosind programarea de eliminare constantă, ponderile modelului sunt reduse semnificativ până la 0,76 valoare finală de eliminare, obținând un raport maxim de compresie de 8,38 cu o degradare a preciziei de testare de numai 1-2%. Cu programarea de eliminare folosind funcția de descreștere polinomială, valoarea finală de eliminare obținută este maxim de 0,62 cu un raport de compresie de 6, mai mică decât în cazul programării de eliminare constantă, în timp ce degradarea preciziei este similară cu eliminarea constantă. Pe baza rezultatelor de mai sus, se poate deduce faptul că pentru a obține valoare finală de eliminare mare, programarea de eliminare constantă este o alegere mai potrivită în comparație cu funcția de descreștere polinomială.

Tabelul 4. Rezultatele folosind programarea cu funcția de descreștere polinomială.

Rezultat	Programarea cu funcția de descreștere polinomială		
	Experiment1	***Experiment2***	***Experiment3***
Precizia de validare	73.79%	73.53%	74.21%
Precizia de testare	73.74%	72.47%	72.47%
Valoarea de eliminare finală	0.59	0.5	0.62
Raportul de compresie pentru precizia de test	5.69	4.88	6

În această lucrare am implementat și evaluat programările de eliminare a ponderilor în funcție de magnitudine: programarea de eliminare constantă și programarea folosind funcția de descreștere polinomială. Folosind programarea de eliminare constantă, se poate obține un raport

de compresie de până la 8,06 cu o valoare finală de eliminare 0,75, în timp ce degradarea preciziei este de numai 1,08%. Folosind programarea cu funcția de dezintegrare polinomială, rezultatul este mai puțin satisfăcător în ceea ce privește raportul de compresie, acesta fiind de până la 5,69 cu o valoare finală de 0,59, în timp ce degradarea preciziei este de 0,74%. Prin urmare, programarea de eliminare constantă este o alegere mai bună în comparație cu funcția de dezintegrare polinomială.

5. DISTILAREA CUNOȘTINȚELOR

A. Prezentare

În lucrarea [16] mi-am propus să abordez problema estimării direcției privirii ochilor cu aplicabilitate în domeniul auto. Propunerea mea folosește un concept de distilare a cunoștințelor aplicat unei arhitecturi de model CNN personalizat, cu rol de model profesor. Domeniul estimării privirii are o importanță semnificativă în contextul interacțiunii om-calculator și pentru multiple aplicații din diverse domenii, cum ar fi industria auto, cercetarea de piață și domeniul medical. În prezent, neatenția șoferilor continuă să fie un factor esențial care în cele mai multe situații conduce la coliziunile auto. Implementarea sistemelor avansate de asistență a șoferului *(„Advanced Driver-Assistance Systems", ADAS)* s-a propus ca fiind o potențială soluție pentru atenuarea incidenței accidentelor auto cauzate de neatenția șoferului. Privirea șoferului poate fi un indicator pentru detecția stării de oboseală sau lipsa de atenție în timpul conducerii. În astfel de situații, este posibil să se transmită avertismente conducătorului auto și dacă este necesar, să fie decise măsuri adecvate pentru a evita o coliziune [19].

Compresia unui model DNN este o practică comună pentru obținerea unei rețele de mici dimensiuni pentru dispozitive hardware de cost redus și resurse limitate. Distilarea cunoștințelor este o metodă particulară care implică antrenament pentru a transfera cunoștințe de la o rețea mai performantă de dimensiuni mari către o altă rețea care are o dimensiune semnificativ mai mică. Bucilua și colab. [20] au demonstrat cu succes pentru prima dată faptul că cunoștințele dobândite de un ansamblu mare de modele pot fi transferate către un singur model mai mic. Scopul utilizării acestei metode este de a antrena rețeaua student astfel încât aceasta să reproducă performanța rețelei profesor, dar cu o dimensiune mai mică, respectiv mai puține resurse de memorie și de calcul.

Procesul de implementare a algoritmului de distilare a cunoștințelor presupune următoarele etape principale: (1) definirea rețelelor profesor și student, (2) antrenarea rețelei profesor și (3) antrenarea rețelei student cu transfer de cunoștințe de la rețeaua profesor. Acești trei pași sunt prezentați pe scurt în continuare.

1) Definirea rețelelor profesor și student

Rețeaua profesor poate fi un model standard de înaltă performanță cu un număr mare de parametri (de exemplu, ResNet, DenseNet sau EfficientNet), în timp ce rețeaua student poate fi de asemenea un model standard, dar cu un număr mai mic de parametrii. În funcție de sarcină, se poate considera și o arhitectură personalizată a rețelelor profesor sau student.

2) Antrenarea rețelei profesor

Rețeaua profesor este antrenată folosind o procedură standard de antrenament, cu scopul de a obține cea mai bună acuratețe.

3) Antrenarea rețelei student cu transfer de cunoștințe de la rețeaua profesor

În această etapă, este folosit algoritmul de distilare a cunoștințelor. Propagarea înainte este efectuată pentru rețelele profesor și student, în timp ce propagarea înapoi se aplică numai rețelei student. Funcția principală de pierdere este definită folosind două funcții distincte de pierdere: funcția de pierdere student și funcția de pierdere de distilare.

Cunoștințele unui model sunt clasificate în trei tipuri diferite: cunoștințe bazate pe răspuns, cunoștințe bazate pe caracteristici și cunoștințe bazate pe relații [21]. Cunoștințele bazate pe răspuns se concentrează pe stratul final de ieșire, unde modelul student va învăța

predicțiile modelului profesor. Cunoștințele bazate pe caracteristici utilizează cunoștințele datelor din straturile intermediare ale modelului profesor pentru antrenarea modelului student. Cunoștințele bazate pe relații se concentrează pe corelația dintre hărțile caracteristicilor, grafice, matricea de similitudine, încorporările caracteristicilor sau distribuțiile probabilistice. În această teză, am folosit cunoștințele bazate pe răspuns deoarece acestea au arătat cele mai bune rezultate pentru diferite sarcini și aplicații.

Diagrama bloc pentru algoritmul de distilare a cunoștințelor este prezentată în Figura 9.

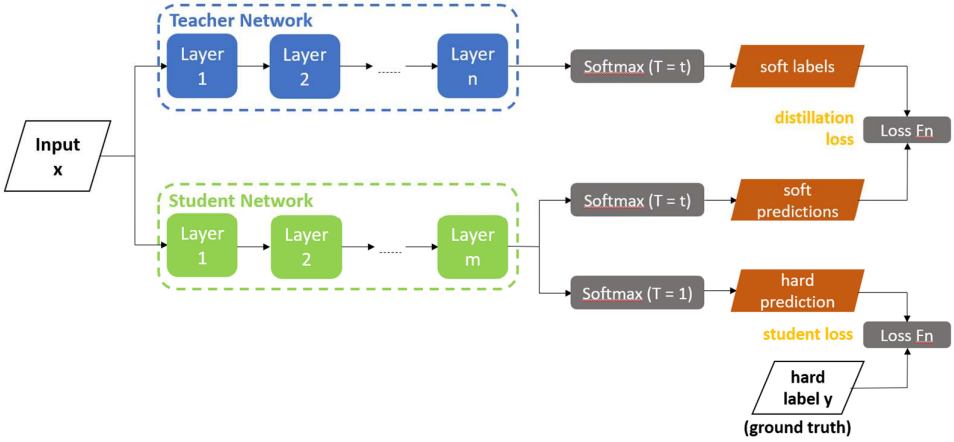

Figura 9. Diagrama bloc pentru distilarea cunoștințelor *[16]*.

Transferul de cunoștințe de la modelul profesor la modelul student se realizează prin minimizarea funcției principale de pierdere cu scopul de a produce aceleași probabilități ca și cele produse de modelul profesor. Mai precis, acest lucru face referire la ieșirea funcției *„Softmax”* aplicată predicțiilor înainte de a fi normalizate. Pentru aceste predicții ale modelului profesor se folosește de regulă denumirea de *„logits”*. Cel mai frecvent, clasa corectă a distribuției probabilității are un nivel mai ridicat în comparație cu celelalte probabilități de clasă care sunt aproape de zero. Prin urmare, rezultatul oferit de această distribuție de probabilitate este foarte similar cu etichetele de adevăr ale setului de date. În ceea ce privește acest comportament, Hinton și colab. [22] au introdus parametrul softmax *„temperature”*. Prin notarea acestui parametru cu T, probabilitatea p_i a clasei i din *„logit”* z_i se calculează cu ecuația (5.1).

$$p_i = \frac{exp\ (z_i/T)}{\sum_j exp\ (z_j/T)} \tag{5.1}$$

Când parametrul T este setat pe valoarea 1, ecuația devine o funcție *„Softmax”* standard. Atunci când T are o valoare mai mare de 1, distribuția probabilității va oferi mai multe informații despre clasele pentru care modelul profesor a raportat o predicție apropiată de clasa corectă. Acestea sunt cunoștințele modelului profesor care sunt transferate modelului student folosind algoritmul de distilare. Atunci când funcția de pierdere de distilare este calculată folosind etichetele *„soft”*, aceeași valoare a lui T este utilizată pentru a calcula funcția *„softmax”* pe *„logits”* modelului student.

S-a observat antrenarea modelului cu distilarea cunoștințelor ca fiind un beneficiu folosind și etichetele de adevăr ale setului de date. Prin urmare, a doua funcție de pierdere este calculată cu parametrul T setat pe valoarea 1. Aceasta este o funcție standard de pierdere numită funcția de pierdere a modelului student. Prin notarea funcției de pierdere de distilare cu H_d (0.2) și a funcției de pierdere a modelului student cu H_s (0.3), funcția principală de pierdere se calculează folosind ecuația (0.4):

$$H_d = H\big(\sigma(z_t; T = \tau), \sigma(z_s; T = \tau)\big) \quad (0.2)$$

$$H_s = H(y, \sigma(z_s; T = 1)) \quad (0.3)$$

$$L(x; W) = \alpha * H_s + \beta * H_d \quad (0.4)$$

unde H este funcția de pierdere entropie încrucișată, σ este funcția *„Softmax”*, z_t și z_s sunt *„logits”* modelelor profesor și student, y este eticheta de adevăr, x este intrarea, W sunt parametrii modelului student, iar α și β sunt coeficienți.

Modelul profesor este un model CNN personalizat de dimensiuni reduse, având numai 5 straturi convoluționale. Acesta este folosit ca model de bază pentru modelele student. Pentru a obține un model de student de dimensiune mai mică, am aplicat două metode diferite: compresia pe straturi și cea pe lățime. Compresia pe straturi presupune reducerea numărului de straturi convoluționale, în timp ce compresia pe lățime înseamnă reducerea numărului de filtre. Modelul student optim este considerat cel care are dimensiunea mai mică, dar în același timp nu a suferit o pierdere semnificativă de precizie. Pentru a crește spațiul de căutare în găsirea modelului student cel mai optim, sunt considerate și diverse combinații de compresie pe straturi și pe lățime.

Pentru compresia pe straturi, am definit următoarele modele student:

- *Student_cut5*, unde am renunțat la ultimul strat convoluțional;
- *Student_cut4*, unde am renunțat la ultimele două straturi convoluționale.

Pentru compresia pe lățime, am definit următoarele modele student:

- *Student_width10*, un model cu 10% mai puține filtre;
- *Student_width30*, un model cu 30% mai puține filtre;
- *Student_width50*, un model cu 50% mai puține filtre.

Combinațiile de compresie pe straturi și pe lățime sunt definite prin aplicarea compresiei pe lățime modelelor rezultate din compresia pe straturi. Am folosit denumirea *Student_cutX&widthY*, unde *X* este modelul cu compresie pe straturi și *Y* este modelul cu compresie pe lățime. În total, au fost obținute 11 modele student cu scopul de a găsi configurația optimă.

B. Concluzii

- **Rezultatele folosind algoritmul de distilare a cunoștințelor pe stația PC**

În funcție de cerințele utilizatorului, compresia unui model poate să fie pentru mai multe tipuri de rezultate: (1) reducerea dimensiunii modelului, fără un impact asupra performanței sau chiar îmbunătățirea preciziei dacă acest lucru este posibil, (2) reducerea dimensiunii modelului și a timpului de inferență, în timp ce se acceptă o mică degradare a preciziei, iar (3) reducerea semnificativă a dimensiunii modelului și a timpului de inferență, dar păstrând precizia în limite acceptabile.

Pe baza rezultatelor obținute, pentru primul caz pot fi utilizate următoarele modele: *Student_cut5*, *Student_width10* și *Student_cut5&width10*. În cazul acestor modele, precizia este mai mare față de cea a modelului profesor. Cel mai mare raport de compresie și reducere a

timpului de inferență sunt obținute cu modelul *Student_cut5&width10*. Totuși, dacă cerința utilizatorului nu este de a obține precizia maximă, atunci modelul *Student_cut5&width10* este cea mai potrivită soluție.

Pentru al doilea caz, cu următoarele modele am obținut o scădere a preciziei cu mai puțin de 1% în comparație cu modelul profesor: *Student_cut4*, *Student_width30* și *Student_cut5&width30*. Cel mai mare raport de compresie și reducere a timpului de inferență sunt obținute pentru modelul *Student_cut5&width30*, care este cel mai potrivit pentru această categorie.

Atunci când dimensiunea modelului și reducerea timpului de inferență sunt considerate cele mai importante aspecte, modelele următoare sunt cele mai potrivite: *Student_width50*, *Student_cut5&width50*, *Student_cut4&width10*, *Student_cut4&width30* și *Student_cut4&width50*. Cel mai mare raport de compresie și reducere a timpului de inferență s-au obținut cu modelul *Student_cut4&width50*, dar în cazul acesta precizia este de numai 68,29%. Modelul din această categorie care are cea mai mare precizie este *Student_cut4&width10*. Precizia acestuia este apropiată de cea a modelului profesor, iar raportul de compresie este de 3,24.

Pe baza rezultatelor prezentate mai sus, pot fi sumarizate următoarele concluzii:

- Folosind algoritmul de distilare a cunoștințelor, precizia poate fi îmbunătățită cu până la 9,5% față de utilizarea unei proceduri convenționale de antrenament. Acest lucru se poate obține folosind modelul obținut în urma metodei de compresie pe straturi și pe lățime: *Student_cut4&width30*;
- Algoritmul de distilare a cunoștințelor este mai eficient atunci când coeficientul α este setat la 0 sau 0,5, iar T este setat la o valoare mai mare, cum ar fi 14. Prin urmare, eficiența este mai mare atunci când funcția principală de pierdere (0.4) se bazează numai pe funcția de pierdere de distilare H_d sau atunci când funcția de pierdere student H_s și funcția de pierdere de distilare H_d au aceeași pondere;
- Folosirea în combinație a compresiei pe straturi și pe lățime este mai eficientă față de utilizarea lor în mod independent;
- Timpul de inferență este redus mai mult folosind compresia pe lățime față de compresia pe straturi. Combinația acestora conduce la o îmbunătățire, dar nu semnificativ de mare.

- **Rezultatele pe platforma hardware țintă STM32H747I-DISCO**

Scopul validării pe platforma STM32H747I-DISCO este de a demonstra faptul că precizia rezultată este similară cu cea obținută pe stația PC, dat fiind faptul ca nu se aplică nici o compresie care ar putea introduce o reducere a preciziei. Diferența mică care este vizibilă poate fi o consecință a faptului că precizia este calculată folosind instrumente diferite.

În ceea ce privește utilizarea memoriei RAM și ROM, următoarele considerații pot fi prezentate pe scurt în funcție de compresia pe straturi sau pe lățime: (1) utilizarea memoriei RAM este mai mică folosind compresia pe lățime (în cazul compresiei pe straturi, diferența nu este semnificativă), (2) utilizarea memoriei ROM este treptat mai mică pe măsură ce dimensiunea modelului scade (în cazul acesta, nu este vizibilă o diferență între compresia pe straturi și pe lățime, iar (3) combinarea compresiei pe straturi și a compresiei pe lățime este mai eficientă, deoarece memoria RAM necesară este de asemenea redusă.

Complexitatea MACC este mai mare atunci când se utilizează compresia pe straturi. Aceasta este redusă semnificativ atunci când se utilizează compresia pe lățime. Cea mai

eficientă alegere este de asemenea și din acest punct de vedere folosirea unei combinații de compresie pe straturi și pe lățime.

Timpul de inferență urmărește o distribuție similară celei prezentate în cadrul rezultatelor obținute pe stația PC. În concluzie, timpul de inferență este redus mai mult folosind compresia pe lățime, iar combinația cu compresia pe straturi conduce la îmbunătățiri. În cazul modelului *Student_cut4&width30*, timpul de inferență este de 870.5 ms, iar precizia este de 73.66%. Acest lucru denotă faptul că o viteză de un cadru pe secundă poate fi obținută cu o precizie apropiată de valoarea maximă. Acest timp de inferență poate fi acceptat pentru aplicațiile practice, fapt ce permite implementarea unui sistem de detecție a privirii în timp real pe o platformă hardware de cost și consum de energie redus.

În zilele noastre rețelele neuronale profunde și paradigma lor asociată de învățare profundă sunt omniprezente în domeniul auto. Lucrarea publicată [16] se referă la o procedură de optimizare care vizează implementarea unui model de rețea neuronală pentru problema de estimare a direcției privirii ochilor pe un hardware de cost redus. Propunerea mea folosește un concept de distilare a cunoștințelor aplicat unei arhitecturi CNN personalizate, numită modelul profesor. Pe baza acestui fapt, mai multe modele student CNN sunt derivate folosind tehnici de compresie pe strat și pe lățime. Ulterior, acestea sunt evaluate în ceea ce privește diferite metrici de performanță cum ar fi, dimensiunea rețelei neuronale și timpul de inferență. Am propus metodele de compresie analizate care sunt mai potrivite și pot răspunde cerințelor specifice ale utilizatorilor, cum ar fi dimensiunea modelului, precizia și timpul de inferență. În cele din urmă, am evaluat modelele student pe dispozitivul încorporat STM32H747IDISCO în termeni de precizie, utilizarea memoriei, complexitatea MACC și timpul de inferență. Folosind algoritmul de distilare a cunoștințelor, precizia poate fi îmbunătățită cu până la 9,5% față de procedura convențională de antrenament. Se obține un raport de compresie de până la 8,86 cu o scădere de mai puțin de 10% a preciziei. Timpul de inferență folosind compresia pe lățime este redus mai mult. Combinarea compresiei pe straturi și pe lățime este mai eficientă și un compromis bun între dimensiunea modelului, precizia și timpul de inferență. Rezultatele validării pe hardware-ul STM32 au arătat faptul că pentru a reduce utilizarea memoriei RAM și ROM, combinația de compresie pe straturi și pe lățime este soluția optimă. Tot această soluție reprezintă alegerea potrivită pentru optimizarea simultană a complexității MACC și a timpului de inferență. Cu toate acestea, am reușit să identific unele limitări ale metodologiei utilizate, deoarece aceasta transferă doar cunoștințe legate de rezultatele modelului inițial și nu captează reprezentările interne învățate de modelul profesor. De asemenea, modelul student poate învăța mai puțin din variantele de model profesor unde există un decalaj important de arhitectură față de modelul student.

6. CONCLUZII ȘI CONTRIBUȚII PERSONALE

Rețelele neuronale profunde au ajuns în ultimii ani într-o etapă foarte avansată, fiind utilizate tot mai frecvent pentru diferite aplicații și dispozitive din viața oamenilor. În mod evident, inteligența artificială va fi un domeniu care marchează profund progresul tehnologic din secolul curent. Utilizarea modelelor de rețele neuronale de la aplicații foarte izolate, cum ar fi ceasurile inteligente până la aplicațiile care rezolvă cât mai multe cerințe cu un singur model, a condus la nevoia unei diversități foarte mari de modele. În același timp, se cunoaște faptul că modelele de ultimă generație cu ajutorul cărora s-au obținut cele mai bune performanțe sunt foarte costisitoare din punct vedere al memoriei ocupate, cerințelor de calcul și consumului de energie. Din acest motiv, în perioada relativ recentă și în prezent nivelul de cercetare a atins un punct unde concentrarea se află pe proiectarea unor modele de dimensiuni mai reduse cu performanțe satisfăcătoare și care să fie potrivite pentru clasa de aplicații izolate (menționată

mai sus). Între timp, pentru a folosi progresul deja existent cu rețele neuronale de mari dimensiuni, compresia rețelelor neuronale preantrenate este un alt punct de cercetare paralel. Aceste direcții sunt justificate de numărul mare de ordinul miliardelor de dispozitive locale care necesită algoritmi de inteligență artificială [23]. Prin urmare, am remarcat relevanța unei cercetări la nivel de studii doctorale în această direcție, având ca scop principal compresia rețelelor neuronale profunde.

Principala contribuție a acestei teze este studiul, descrierea și implementarea algoritmilor de compresie a rețelelor neuronale profunde precum cuantizarea, distilarea cunoștințelor și eliminarea ponderilor folosind rețele neuronale convoluționale de diferite dimensiuni și un model specific pentru aplicația de detecție a privirii conducătorului auto. Implementarea s-a realizat folosind diferite configurații cu scopul de a prezenta într-un mod detaliat rezultatele cu privire la precizie, raportul de compresie obținut și timpul de inferență cu accent pe configurația potrivită în funcție de cerințele sistemului.

Pe parcursul acestei teze de doctorat am studiat mai mult de 160 de titluri bibliografice, am elaborat două articole publicate în reviste ISI Q2 („IEEE Access”), Q3 („MDPI Electronics”) și indexate „Web of Science”, o lucrare de conferință indexată „ISI Proceedings” și „Web of Science”, o lucrare de conferință indexată BDI și două lucrări de conferință care se află la ora actuală în curs de review. Pentru toate aceste lucrări am contribuit în calitate de prim autor, mai puțin pentru ultima lucrare de conferință care este în curs de redactare și unde voi fi în calitate de al doilea autor. Doresc să punctez faptul că în prezent lucrările au în total un număr de 31 de citări conform raportului Google Academic. În secțiunea următoarea am prezentat contribuțiile personale la nivel de detaliu pentru fiecare lucrare științifică.

Contribuțiile personale prezente în articolul de revistă ISI Q3 („MDPI Electronics”) [24] sunt:

- Am prezentat câteva dispozitive hardware de cost redus (mai puțin de 10 USD pentru un microcontroler) și reprezentative împreună cu bibliotecile sau instrumentele suport care ajută la implementarea algoritmilor de învățare automată pe acestea;
- Am rezumat 13 lucrări din ultimii ani unde s-au utilizat microcontrolere cu CPU ARM Cortex-M pentru rularea algoritmilor de învățare automată. Am urmărit să punctez următoarele aspecte principale: arhitectura modelului, caracteristicile hardware (cum ar fi: amprenta de memorie disponibilă, arhitectura CPU și frecvența de operare), instrumentele și bibliotecile suport utilizate împreună cu rezultatele obținute în termeni de: precizie, timp de inferență și consum de energie;
- Am descris într-un mod amănunțit rezultatele obținute cu accent pe cazurile unde s-au obținut cele mai bune rezultate. De asemenea, am discutat principalele motive care reprezintă în multe situații un impediment pentru obținerea rezultatelor dorite;
- Am evidențiat provocările și oportunitățile de cercetare care subliniază aspecte importante și reprezintă o provocare pentru progresul viitor din acest domeniu.

Contribuțiile personale prezente în articolul de revistă ISI Q2 („IEEE Access”) [16] sunt:

- Am definit și antrenat o arhitectură CNN personalizată cu scopul de a clasifica estimarea privirii ochilor, având o dimensiune de 6.14 MB și o precizie obținută de 77.48%;
- Am aplicat metodologia de distilare a cunoștințelor cu diferite configurații pentru a antrena mai multe modele student definite folosind metode de compresie pe straturi și pe lățime. Pentru modelul profesor am utilizat arhitectura CNN personalizată;
- Am prezentat o evaluare a rezultatelor experimentale cu scopul de a identifica cea mai

potrivită metodă de compresie în funcție de cerințele utilizatorului, cum ar fi: dimensiunea modelului, precizia dorită și timpul așteptat pentru o inferență;
- Am validat modelele student folosind interfața de validare AI a pachetului de extensie X-CUBE-AI pe dispozitivul hardware STM32H747I-DISCO. În acest scop, am prezentat detalii cu privire la acuratețea obținută, utilizarea memoriei, complexitatea MACC și timpul de inferență.

Contribuțiile personale prezente în lucrarea de conferință indexată „ISI Proceedings" și „Web of Science" [7] sunt:

- Am convertit un model de rețea CNN din formatul generic Caffe la un format compatibil cu microcontrolerul STM32;
- Am implementat interfețele dintre modulele periferice (cum ar fi, modulul de afisaj LCD și modulul cu cameră) și modelul CNN, astfel încât împreună să formeze un sistem funcțional;
- Am validat modelul CNN direct pe microcontroler la modul real folosind modulul cu cameră și am testat comportamentul general al modelului folosind imaginile de test ale setului de date CIFAR 10.

Contribuțiile personale prezente în lucrarea de conferință indexată BDI [13] sunt:

- Am evaluat comparativ raportul de compresie care se obține utilizând cuantizarea pe numere întregi pe 8 biți și cuantizarea în virgulă mobilă pe 16 biți folosind patru tehnici PTQ diferite;
- Am analizat raportul de compresie în funcție de dimensiunea modelului DNN, folsind trei modele diferite;
- Am evidențiat influența tehnicilor PTQ asupra preciziei modelelor.

Contribuțiile personale prezente în lucrarea care vizează eliminarea ponderilor în funcție de magnitudine în curs de redactare sunt:

- Am aplicat tehnica de eliminare a ponderilor în funcție de magnitudine unui model personalizat CNN antrenat cu scopul de a clasifica estimarea privirii ochilor;
- Am evaluat comparativ programările de eliminare constantă și funcția de dezintegrare polinomială cu privire la raportul de compresie, gradul de eliminare, precizia de test și validare;
- Am prezentat o evaluare a rezultatelor experimentale cu scopul de a identifica care este programarea potrivită pentru eliminarea ponderilor în vederea obținerii unui grad ridicat de eliminare și a unui raport mare de compresie, fără o pierdere semnificativă de precizie.

Bibliografie

[1] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks,* vol. 1, p. 119–130, 1988.

[2] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, Cambridge, MA: MIT Press, 2016.

[3] K. Kar, Mastering Computer Vision with TensorFlow 2.x, May, 2020.

[4] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, p. 533–536, 1986.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.,* vol. 9, p. 1735–1780, 1997.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016.

[7] I. L. Orășan and C. D. Căleanu, "ARM Embedded Low Cost Solution for Implementing Deep Learning Paradigms," Timisoara, 2020.

[8] A.-A. Erofei, C.-F. Druţa and C. D. Căleanu, "Embedded Solutions for Deep Neural Networks Implementation," Timisoara, 2018.

[9] R. MIRSU, S. MICUT, C. CALEANU and D. B. MIRSU, "Optimized Simulation Framework for Spiking Neural Networks using GPU's," vol. 12, pp. 61-68, 2012.

[10] L. Lai, N. Suda and V. Chandra, "CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs," January 2018.

[11] M. Ayi and M. El-Sharkawy, "RMNv2: Reduced Mobilenet V2 for CIFAR10," Las Vegas, NV, USA, 2020.

[12] J. Lee, S. Kang, J. Lee, D. Shin, D. Han and H.-J. Yoo, "The Hardware and Algorithm Co-Design for Energy-Efficient DNN Processor on Edge/Mobile Devices," *IEEE Trans. Circuits Syst.,* Vols. 67-I, p. 3458–3470, 2020.

[13] I. L. Orășan, C. Seiculescu and C. D. Caleanu, "Benchmarking TensorFlow Lite Quantization Algorithms for Deep Neural Networks," Timisoara, 2022.

[14] G. Huang, Z. Liu, L. V. D. Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," Honolulu, 2017.

[15] J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 42, p. 2011–2023, 2020.

[16] I. L. Orasan, A.-I. Bublea and C.-D. Caleanu, "Deep Learning-Based Eye Gaze Estimation for Automotive Applications Using Knowledge Distillation," *IEEE Access,* vol. 11, p. 120741–120753, 2023.

[17] V. Isaac–Chassande, A. Evans, Y. Durand and F. Rousseau, "Dedicated Hardware Accelerators for Processing of Sparse Matrices and Vectors: A Survey," vol. 21, pp. 1-26, 2024.

[18] TensorFlow, "Model optimization," [Online]. Available: https://www.tensorflow.org/model_optimization. [Accessed 6 July 2024].

[19] H. U. Draz, M. I. Ali, M. U. G. Khan, M. Ahmad, S. Mahmood and M. A. Javaid, *An Embedded Solution of Gaze Estimation for Driver Assistance using Computer Vision,* 2021.

[20] C. Buciluă, R. Caruana and A. Niculescu-Mizil, "Model compression," in *Proceedings*

of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, 2006.

[21] J. Gou, B. Yu, S. J. Maybank and D. Tao, "Knowledge Distillation: A Survey," *Int. J. Comput. Vis.,* vol. 129, p. 1789–1819, 2021.

[22] G. Hinton, O. Vinyals and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv e-prints,* p. arXiv:1503.02531, March 2015.

[23] L. Ding, "Artificial intelligence solutions running on STM32," [Online]. Available: https://www.st.com/content/dam/specialevents-assets/electronica-china-2023/st-edge-ai-english.pdf. [Accessed 7 July 2024].

[24] I. L. Orășan, C. Seiculescu and C. D. Căleanu, "A Brief Review of Deep Neural Network Implementations for ARM Cortex-M Processor," *Electronics,* vol. 11, p. 2545, August 2022.