MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Ochelari pentru nevăzători

LUCRARE DE LICENȚĂ

Absolvent: **Anton-Călin Stroia**

Coordonator științific: **prof.dr.ing. Gheorghe SEBESTYEN-PAL**

2025

Cuprins

Capitolul 1. Introducere

1.1. Contextul temei

În ultimele decenii, progresul accelerat al tehnologiei informației și comunicațiilor a transformat fundamental multiple aspecte ale societății moderne. Domenii precum inteligența artificială (AI), învățarea automată (*machine learning*), viziunea computerizată (*computer vision*), recunoașterea vocală și interfețele naturale au depășit cadrul teoretic sau experimental, pătrunzând în aplicații comerciale, educaționale și sociale. Această revoluție tehnologică a generat oportunități semnificative în îmbunătățirea calității vieții, în special pentru grupurile vulnerabile, printre care se numără persoanele cu dizabilități senzoriale.

Conform Organizației Mondiale a Sănătății, la nivel global există peste 2,2 miliarde de persoane cu deficiențe de vedere, dintre care cel puțin 36 de milioane sunt complet nevăzătoare. Pentru acești indivizi, viața de zi cu zi este presărată cu provocări majore: orientarea în spații necunoscute, identificarea obiectelor din mediul înconjurător, recunoașterea obstacolelor, accesul la informații textuale (etichete, instrucțiuni, semnalizări) sau interacțiunea cu dispozitive digitale. În lipsa unor soluții adaptate, aceste obstacole contribuie la marginalizare socială, dependență de suport extern și scăderea autonomiei personale.

În contextul descris, dezvoltarea unor sisteme asistive bazate pe tehnologii emergente devine o prioritate atât pentru comunitatea științifică, cât și pentru cea industrială. Scopul acestor sisteme este de a compensa parțial limitările prin integrarea de tehnologii care să interpreteze mediul vizual și să furnizeze utilizatorului informații în formate alternative, în special prin canalul auditiv. Astfel, conceptul de *computer vision for accessibility* s-a impus ca un domeniu interdisciplinar de interes, îmbinând ingineria hardware, dezvoltarea software, psihologia percepției și ergonomia interacțiunii om-computer.

Un exemplu emblematic de platformă tehnologică accesibilă, folosită în prototiparea rapidă a acestor sisteme, este **Raspberry Pi**. Lansat inițial ca un microcomputer educațional, Raspberry Pi a devenit rapid o soluție adoptată la scară globală pentru proiecte embedded datorită dimensiunilor reduse, consumului scăzut de energie, prețului accesibil și ecosistemului bogat de module compatibile (camere, senzori, module audio, comunicații fără fir). Această platformă oferă un teren fertil pentru dezvoltarea de soluții inovatoare, inclusiv pentru asistarea persoanelor cu deficiențe de vedere, prin integrarea funcțiilor de recunoaștere vizuală, analiză semantică și feedback multimodal.

Pe lângă infrastructura hardware, progresele recente în domeniul algoritmilor software au amplificat considerabil potențialul aplicațiilor asistive. Printre cele mai relevante tehnologii se numără:

- **Rețelele neuronale convoluționale (CNN)**, care permit clasificarea și detectarea obiectelor în imagini cu un grad ridicat de acuratețe, chiar și în condiții de variație a iluminării, poziției sau fundalului;
- **Recunoașterea optică a caracterelor (OCR)**, permite extragerea automată a textului din imagini captate, facilitând accesul la informație scrisă (ex: etichete de

produse, instrucțiuni, semne de avertizare);
- **Sinteza vocală (Text-to-Speech, TTS)**, esențială pentru transformarea informației vizuale în feedback auditiv comprehensibil, livrat în timp real utilizatorului.

Integrarea acestor componente într-un sistem embedded presupune depășirea unor provocări tehnice specifice: limitările de memorie și procesare, nevoia de optimizare a modelelor AI (prin conversie în formate compacte precum TensorFlow Lite), gestionarea surselor de energie și asigurarea unei interfețe minimale, adaptate utilizatorilor nevăzători (de exemplu, comenzi prin butoane fizice și feedback vocal în locul interfețelor grafice).

Proiectul prezentat în această lucrare își propune dezvoltarea unei soluții asistive portabile, bazate pe Raspberry Pi, capabile să îndeplinească două funcții principale:

- Detectarea obiectelor de uz cotidian (precum farfurii, linguri, furculițe, pahare) prin intermediul unui model CNN antrenat și optimizat pentru execuție pe arhitectură ARM.
- Recunoașterea și citirea textului din mediu, utilizând un modul OCR bazat pe motorul open-source Tesseract, cu redarea rezultatelor prin sinteză vocală offline.

Dincolo de implementarea tehnică, acest proiect abordează o problematică de interes social major: reducerea barierelor de acces la mediu și informație pentru persoanele cu deficiențe de vedere. Scopul final nu este doar demonstrarea fezabilității tehnologice, ci și conturarea unei direcții de dezvoltare care să permită viitoare extinderi, precum integrarea comenzilor vocale, conectivitatea la cloud pentru actualizări sau extinderea dataseturilor de antrenament pentru noi clase de obiecte.

În concluzie, contextul temei se situează la intersecția dintre necesitățile reale ale unei categorii vulnerabile de utilizatori și oportunitățile deschise de tehnologiile embedded și AI. Lucrarea de față își propune să demonstreze că, printr-o proiectare atentă, optimizare riguroasă și integrare pluridisciplinară, se pot crea soluții asistive portabile, eficiente și scalabile, cu potențial real de impact în viața de zi cu zi a persoanelor nevăzătoare.

1.2. Conturarea domeniului temei

Lucrarea de față se concentrează pe proiectarea și implementarea unui sistem inteligent de asistență pentru persoane nevăzătoare, bazat pe o platformă Raspberry Pi. Domeniul principal de aplicabilitate este cel al sistemelor embedded cu funcționalitate AI, dedicate persoanelor cu dizabilități.

Soluția propusă are la bază o arhitectură hardware simplă și accesibilă: Raspberry pi Zero 2 W, o cameră compatibilă (ex. Pi Camera v3) și un sistem audio Bluetooth (căști sau boxă portabilă). Sistemul va executa următoarele funcționalități principale:

- Detectarea și identificarea obiectelor (tacâmuri), utilizând un model de detecție antrenat.
- Recunoașterea textului din imagine prin tehnici OCR (de exemplu, cu Tesseract OCR), facilitând citirea de etichete, sau instrucțiuni;
- Conversia informației vizuale în feedback vocal, prin utilizarea unui modul text-to-speech (TTS), pentru a permite utilizatorului perceperea auditivă a conținutului vizual;
- Transmiterea audio a informațiilor către utilizator prin căști sau boxe Bluetooth.

Acest proiect se plasează la intersecția mai multor domenii de cercetare și dezvoltare:

- **Sisteme Embedded**: Programarea și integrarea componentelor hardware și software pe o platformă cu resurse limitate;
- **Inteligență Artificială**: Utilizarea rețelelor neuronale pentru clasificarea obiectelor și interpretarea imaginilor;
- **Procesarea Imaginilor**: Prelucrarea imaginilor pentru extragerea textului și recunoașterea obiectelor;
- **Interacțiune Om-Calculator**: Crearea unui canal intuitiv de comunicare între sistem și utilizator, prin sinteză vocală.

Prin această lucrare se urmărește validarea ideii că un sistem low-cost, portabil și bazat pe componente open-source poate oferi un real sprijin persoanelor nevăzătoare. Totodată, se dorește punerea în evidență a potențialului pe care tehnologiile moderne îl au în dezvoltarea unor soluții cu impact social major.

Pe parcursul lucrării, se vor analiza componentele hardware și software utilizate, vor fi prezentate arhitectura sistemului și algoritmii implementați, iar în final vor fi expuse rezultatele obținute și concluziile privind performanțele sistemului.

Capitolul 2. Obiectivele proiectului

2.1. Tema proiectului

Lucrarea de față propune proiectarea și realizarea unui sistem inteligent de asistență pentru persoane cu deficiențe de vedere, utilizând platforma embedded Raspberry Pi. Scopul principal al sistemului este de a interpreta informații vizuale din mediul înconjurător, fie sub formă de obiecte (precum tacâmuri), fie sub formă de text, și de a reda aceste informații vocal printr-un sistem text-to-speech conectat la un dispozitiv audio Bluetooth.

Sistemul integrează o cameră video pentru captarea imaginilor, un modul de procesare vizuală pentru detecția obiectelor sau recunoașterea textului prin OCR, și un modul audio pentru conversia rezultatului în feedback vocal. Utilizatorul poate comuta între cele două moduri de funcționare – detecție de obiecte și recunoaștere de text – prin intermediul unui buton fizic montat discret pe dispozitiv, asigurând astfel o interacțiune ușor de utilizat, fără necesitatea unei interfețe vizuale.

Proiectul se încadrează în domeniul sistemelor embedded cu aplicații de inteligență artificială, având ca obiectiv dezvoltarea unei soluții portabile, eficiente și accesibile, care să sprijine utilizatorii nevăzători în activități cotidiene precum recunoașterea obiectelor de pe masă sau citirea de texte din mediu.

2.2. Obiectiv general

Obiectivul principal al proiectului constă în dezvoltarea și implementarea unui sistem embedded inteligent, autonom și portabil, destinat persoanelor cu deficiențe de vedere, care să permită perceperea și interpretarea mediului vizual în timp real prin intermediul feedback-ului vocal. Acest sistem urmărește integrarea eficientă a unor tehnologii moderne precum recunoașterea obiectelor prin inteligență artificială, extragerea textului din imagini folosind OCR, și sinteza vocală cu ajutorul tehnologiei text-to-speech. În plus, interacțiunea cu dispozitivul trebuie să fie intuitivă și minim intruzivă, fără a necesita interfețe grafice sau intervenții tactile complexe, facilitând astfel utilizarea zilnică în condiții reale, pentru activități precum identificarea obiectelor uzuale sau citirea de etichete, meniuri sau afișe informative.

2.3. Obiective specifice

Obiectivele specifice ale proiectului vizează realizarea unui sistem asistiv portabil, capabil să sprijine persoanele cu deficiențe de vedere în identificarea obiectelor și a textelor din mediul înconjurător, oferind în același timp un feedback auditiv clar și relevant. Aceste obiective sunt formulate astfel încât să acopere toate etapele esențiale, de la selecția hardware-ului potrivit, până la integrarea software și validarea finală în scenarii de utilizare reală.

- **Selectarea și configurarea platformei hardware:** Alegerea plăcii Raspberry Pi Zero 2W ca platformă embedded datorită dimensiunilor compacte, consumului redus de energie și compatibilității cu module externe (cameră, Bluetooth, GPIO). Configurarea include instalarea camerei compatibile, a modulului Bluetooth pentru transmisia audio și integrarea unei surse de alimentare portabile (power bank) pentru asigurarea autonomiei.
- **Implementarea unui algoritm eficient de detecție a obiectelor cu ajutorul unei rețele neuronale convoluționale (CNN):** Dezvoltarea și antrenarea unui model CNN personalizat, capabil să recunoască obiecte de uz casnic relevante (farfurii, linguri, furculițe, cutite) și optimizarea acestuia prin conversie în format TensorFlow Lite, astfel încât să poată rula cu viteză și precizie pe un dispozitiv cu resurse limitate.
- **Extragerea textului din imagini utilizând motorul OCR Tesseract:** Integrarea motorului OCR Tesseract pentru recunoașterea textului din imaginile capturate, incluzând un flux de preprocesare a imaginilor (filtrare mediană, conversie la grayscale, binarizare adaptivă, eventual corecție de perspectivă) pentru îmbunătățirea semnificativă a acurateței în condiții de iluminare și poziționare variabilă.
- **Utilizarea unui sistem TTS (Text-to-Speech):** Implementarea unui mecanism de generare a mesajelor vocale prin biblioteca `espeak-ng`, care să asigure un feedback auditiv clar, inteligibil și configurabil (viteză, volum, limbă), adaptat cerințelor utilizatorului nevăzător.
- **Integrarea unui mecanism de selecție a modului activ:** Dezvoltarea unui sistem de comutare între modul de detecție a obiectelor și modul de citire a textului, prin intermediul unui buton fizic conectat la interfața GPIO, cu protecție software (debounce) și confirmări audio clare, pentru o experiență intuitivă și hands-free.
- **Configurarea transmisiei audio către un dispozitiv Bluetooth:** Setarea și gestionarea automată a conexiunilor Bluetooth către căști sau boxe portabile, asigurându-se redarea fiabilă a feedbackului vocal generat de sistem, fără a necesita intervenția utilizatorului pentru reconectare la fiecare pornire.
- **Testarea și validarea sistemului în condiții reale:** Realizarea de sesiuni de testare în medii diverse (interior/exterior, luminozitate variabilă, zgomot ambiental), pentru a evalua robustețea sistemului, acurateța detecției obiectelor și textelor, precum și gradul de satisfacție a utilizatorului final.
- **Optimizarea consumului de energie și a timpului de răspuns:** Ajustarea parametrilor software (frecvența ciclurilor de inferență, dimensiunea loturilor de date, prioritizarea proceselor) și evaluarea consumului hardware, astfel încât sistemul să poată funcționa pe durata unei zile fără reîncărcări frecvente, păstrând în același timp o viteză de reacție adecvată.

2.4. Rezultate așteptate

Proiectul propus își propune dezvoltarea și implementarea unui sistem asistiv complet integrat, destinat sprijinirii persoanelor cu deficiențe de vedere în activitățile cotidiene. Rezultatele așteptate reflectă atât aspecte tehnice, cât și funcționale, având ca scop crearea unui dispozitiv portabil, eficient și intuitiv de utilizat, capabil să funcționeze în condiții reale. În continuare sunt detaliate principalele obiective și criterii de performanță așteptate:

- **Detectarea obiectelor relevante în timp real:** Sistemul va utiliza un model de rețea neuronală convoluțională (CNN), optimizat pentru rulare pe platforme embedded (prin conversie în format TensorFlow Lite), capabil să identifice obiecte de uz cotidian (ex. farfurii, linguri, furculițe, pahare) cu o acuratețe minimă de 70–80%. Se așteaptă ca inferența per imagine să aibă un timp de execuție sub 1 secundă, astfel încât utilizatorul să primească feedback aproape instantaneu, favorizând utilizarea practică în contexte dinamice.
- **Recunoașterea textelor din imagini în condiții variate:** Modulul OCR, bazat pe motorul Tesseract, va permite extragerea informațiilor textuale din imagini capturate în medii cu iluminare neuniformă, fundaluri diverse și perspective diferite. Se urmărește ca sistemul să recunoască corect texte de tip imprimat (nu manuscris), inclusiv etichete, instrucțiuni, semne sau anunțuri, cu un grad de precizie suficient pentru a furniza utilizatorului informații utile și relevante.
- **Generarea de mesaje audio clare și coerente:** Sistemul va integra un sintetizator vocal offline (`espeak-ng`), configurat pentru limba română, capabil să redea mesaje concise și lizibile auditiv, într-un timp cât mai scurt de la generarea rezultatului vizual. Se vor urmări parametri precum viteza de vorbire, tonalitatea și volumul, astfel încât feedback-ul să fie ușor de înțeles, fără a genera confuzii sau supraîncărcare cognitivă.
- **Interacțiune simplificată prin comutarea modurilor de operare:** Utilizatorul va putea comuta între modul de detecție obiecte și modul OCR printr-o apăsare de buton fizic conectat la interfața GPIO a plăcii Raspberry Pi. Se așteaptă ca sistemul să confirme vocal fiecare schimbare de mod și să minimizeze riscul de erori accidentale (ex. prin implementarea unui mecanism de debounce software).
- **Funcționare autonomă, fără interacțiune vizuală sau tactilă complexă:** Dispozitivul va porni automat odată cu alimentarea, fără a necesita conectarea unui monitor, tastatură sau alte periferice. Operarea sistemului va fi complet hands-free, cu excepția butonului de comutare moduri, iar toate notificările vor fi livrate exclusiv pe cale auditivă.
- **Portabilitate, eficiență energetică și accesibilitate:** Sistemul final va fi compact și ușor de transportat, alimentat de la o baterie portabilă (power bank), cu o autonomie estimată de minimum 8–10 ore de utilizare continuă. Se urmărește menținerea unui consum mediu de energie scăzut (sub 300 mA), pentru a permite folosirea zilnică fără restricții. Costurile de implementare vor fi reduse, astfel încât soluția să fie accesibilă și replicabilă pentru scopuri educaționale, sociale sau de cercetare.

Capitolul 3. Studiu bibliografic

Tehnologiile asistive pentru nevăzători reprezintă un domeniu interdisciplinar în continuă expansiune, situat la intersecția dintre inteligența artificială, viziunea computerizată, prelucrarea limbajului natural și ingineria embedded. Scopul acestor sisteme este acela de a îmbunătăți accesul persoanelor cu deficiențe de vedere la informații vizuale din mediul înconjurător, facilitând astfel autonomia, mobilitatea și integrarea lor în societate. În contextul avansului accelerat al platformelor hardware accesibile, precum Raspberry Pi, și al progreselor în recunoașterea vizuală și sinteza vocală, a devenit posibilă implementarea unor soluții portabile și eficiente, capabile să traducă imagini în feedback auditiv în timp real.

Domeniul de cercetare relevant este strâns legat de inițiativele din sfera *Active and Assisted Living (AAL)*, care propun tehnologii menite să sprijine persoanele cu dizabilități în activitățile cotidiene, fără a le impune dependența de un operator uman. Sistemele care integrează recunoașterea optică a caracterelor (OCR), detecția obiectelor cu rețele neuronale convoluționale (CNN) și sintetizatoare vocale (TTS) se află în centrul acestor inițiative, deoarece oferă o interfață naturală, hands-free, pentru perceperea mediului. Mai mult, integrarea acestor tehnologii într-un singur dispozitiv embedded impune o serie de provocări în ceea ce privește optimizarea performanței, gestionarea resurselor hardware limitate și furnizarea unui feedback intuitiv și eficient.

În acest capitol sunt analizate lucrări relevante din literatura de specialitate care propun soluții de asistență pentru nevăzători, cu accent pe integrarea modulelor OCR, CNN și TTS pe platforme embedded. Scopul analizei este identificarea abordărilor actuale, a avantajelor și limitărilor acestora, precum și a direcțiilor de dezvoltare relevante pentru lucrarea de față. Studiul comparativ evidențiază diferențele de implementare, nivelul de autonomie, calitatea interfeței audio și adaptabilitatea în condiții reale de utilizare.

Lucrarea [1] prezintă un exemplu concret de astfel de sistem, integrat pe o platformă Raspberry Pi. Sistemul include o cameră care captează imaginea, un modul OCR pentru extragerea textului, un sintetizator vocal care redă textul identificat și un model CNN pre-antrenat pentru recunoașterea obiectelor din imagine. Dispozitivul a fost testat în scenarii reale, cum ar fi identificarea de semne, afișe sau obiecte cotidiene (scaune, semne de circulație, pungi etc.). Rezultatele preliminare arată o rată de succes rezonabilă în identificarea corectă a obiectelor și în recunoașterea textului lizibil. Avantajele principale ale acestui sistem sunt costul scăzut, portabilitatea și capacitatea de a funcționa offline, fără conexiune la internet. Totuși, lucrarea nu oferă detalii suficiente despre tipul CNN folosit, performanțele acestuia în medii dificile (lumină slabă, fundal aglomerat) și timpul de procesare.

În [2], autorii propun un sistem similar, denumit AI-WEAR, proiectat special pentru studenți nevăzători. Acesta utilizează tot o placă Raspberry Pi, dar pune accent pe simplitatea utilizării și pe integrarea într-un mediu educațional. Sistemul recunoaște textul din materiale tipărite (cărți, foi, prezentări) folosind Tesseract OCR și îl redă sonor utilizatorului cu ajutorul unui sintetizator vocal. Deși nu include un modul de detecție a obiectelor cu CNN, soluția se distinge prin interfața accesibilă, comenzile vocale simple și

capacitatea de a fi adaptată la nevoile individuale ale utilizatorului (limbă, viteza vorbirii, tonalitate). Prin urmare, deși este mai puțin complex din punct de vedere funcțional, AI-WEAR este un exemplu de sistem bine adaptat unui caz de utilizare specific, cu accent pe accesibilitate și ușurință în utilizare.

Este de remarcat că, în timp ce unele sisteme (ca cel din [1]) prioritizează complexitatea tehnologică și integrarea completă a celor trei componente (OCR, TTS, CNN), altele precum [2] sacrifică anumite funcționalități în favoarea stabilității și utilizabilității. Alegerea între aceste abordări depinde de scopul aplicației, publicul țintă și resursele hardware disponibile.

Din punct de vedere al implementării, majoritatea sistemelor utilizează Tesseract ca motor OCR, datorită faptului că este open-source, ușor de integrat și oferă suport pentru multiple limbi. Pentru sintetizarea vocii, se folosesc adesea module precum 'pyttsx3' pentru funcționare offline sau Google Text-to-Speech pentru acuratețe mai mare (dar necesită conexiune la internet). Pentru recunoașterea obiectelor, sunt frecvent folosite modele pre-antrenate YOLOv3/v4 sau MobileNet-SSD, acestea oferind un compromis bun între viteză și precizie pe hardware limitat, cum este Raspberry Pi.

Integrarea celor trei componente într-un sistem coerent aduce mai multe provocări tehnice: sincronizarea între module, limitările de performanță ale plăcii Raspberry Pi, gestionarea memoriei și a resurselor CPU/GPU, precum și adaptarea interfeței pentru utilizatori cu deficiențe de vedere. Mai mult, alegerea unor formate de feedback vocal intuitive și eficiente (de exemplu, indicarea poziției obiectelor în spațiu) este crucială pentru a transforma un sistem tehnic într-un asistent real pentru utilizatori.

În concluzie, sistemele integrate OCR + TTS + CNN oferă un potențial considerabil pentru aplicații asistive, iar studiile analizate arată că soluțiile actuale pot fi adaptate și extinse pentru a răspunde nevoilor persoanelor nevăzătoare. Lucrarea de față își propune să continue pe această direcție, oferind un model CNN optimizat și o integrare eficientă cu modulele OCR și TTS, cu accent pe performanță, accesibilitate și robustețe în condiții reale de utilizare.

3.1. OCR cu sinteză vocală

Articolul [3] propune o abordare tradițională dar eficientă, care folosește o platformă Raspberry Pi 3 împreună cu motorul Tesseract OCR și Google Speech API pentru redarea vocală. Tesseract este recunoscut pentru performanțele bune în recunoașterea textului tipărit, având suport pentru multiple limbi și seturi de caractere. Sistemul propus funcționează într-un mod secvențial: imaginea este captată de o cameră, apoi trecută prin pași de pre-procesare pentru îmbunătățirea contrastului și a clarității, iar ulterior textul este extras și transmis către un API extern de sinteză vocală. Avantajul acestei arhitecturi este modularitatea – fiecare componentă poate fi actualizată sau înlocuită separat. De asemenea, folosirea Google Speech API oferă o calitate vocală foarte bună. Totuși, dependența de conexiune la internet pentru utilizarea API-ului de TTS este o limitare semnificativă în medii fără acces stabil la rețea.

În lucrarea [4], autorii prezintă un sistem mai complex, axat pe funcționare în timp real și autonomie totală față de conexiunile externe. Acesta utilizează Raspberry Pi 4 împreună cu Tesseract pentru OCR, dar include și un modul TTS local (de exemplu 'pyttsx3' sau 'espeak') care permite redarea vocală direct pe dispozitiv, fără a necesita acces la internet. În plus, imaginile sunt procesate printr-un pipeline de pre-procesare inteligentă: sunt aplicate operații de binarizare, detecție de margini, eliminare a fundalului și redimensionare, toate menite să îmbunătățească acuratețea OCR-ului. Modelul este

optimizat pentru rulare în timp real, astfel încât să ofere feedback imediat utilizatorului. Această soluție este adaptată mediilor dinamice (ex: deplasarea printr-un spațiu public), în care utilizatorul are nevoie de o reacție rapidă și clară la textul întâlnit (panouri informative, semne de avertizare, meniuri etc.).

Comparând cele două lucrări, observăm o diferență fundamentală în filosofia de implementare:

- Sistemul din [3] se bazează pe servicii cloud pentru sintetizarea vocii, ceea ce îi permite o calitate vocală superioară, dar îl face dependent de conectivitate și mai vulnerabil la întârzieri.
- În schimb, sistemul din [4] funcționează complet offline, oferind robusteză și autonomie, dar necesită optimizări mai atente pentru vocea generată și pentru consumul de resurse al prelucrării imaginilor.

Ambele abordări au în comun utilizarea platformei Raspberry Pi, ceea ce subliniază faptul că hardware-ul accesibil și portabil este suficient de performant pentru a susține aplicații OCR+TTS în regim embedded. Totuși, diferențele în arhitectură și interfață vocală pot avea un impact semnificativ asupra experienței utilizatorului final. Alegerea între calitatea vocii și funcționarea offline este una strategică și trebuie realizată în funcție de publicul țintă al aplicației.

În contextul lucrării de față, este recomandabilă alegerea unui sistem hibrid, care să poată funcționa în mod offline, dar să permită opțional și integrarea cu API-uri online dacă sunt disponibile. O astfel de flexibilitate ar crește adoptabilitatea în diferite scenarii: în orașe cu conectivitate bună, dar și în zone rurale sau în interiorul clădirilor unde semnalul este slab.

O provocare suplimentară în acest domeniu este identificarea și extragerea corectă a textului din imagini de calitate slabă – text înclinat, text parțial ascuns, fonturi neuzuale sau medii cu contrast redus. Aici, îmbunătățirea pre-procesării imaginii devine esențială. Unele lucrări explorează și integrarea de rețele neuronale pentru îmbunătățirea OCR (de exemplu, rețele LSTM pentru corectarea rezultatelor Tesseract), dar această direcție este mai puțin prezentă în literatura legată de Raspberry Pi, din cauza constrângerilor hardware.

În concluzie, sistemele OCR+TTS pot avea un impact real asupra calității vieții persoanelor nevăzătoare, oferindu-le acces rapid la informații vizuale scrise. Lucrările analizate evidențiază atât potențialul, cât și provocările acestor tehnologii, iar integrarea lor într-un cadru eficient și accesibil rămâne un obiectiv central al prezentei lucrări.

3.2. Integrarea detecției de obiecte și feedback vocal

Lucrarea [5] prezintă un sistem embedded care combină OCR (prin Tesseract), detecție de obiecte (cu ImageMagick și un model simplu de clasificare) și interfață audio printr-un modul TTS. Scopul acestui sistem este acela de a descrie obiectele identificate și de a reda textul detectat într-o singură sesiune auditivă. Această abordare este extrem de relevantă pentru utilizatorii nevăzători, întrucât permite înțelegerea simultană a două tipuri de informații: textuale (ex: etichete, semne) și contextuale (ex: obiecte de mobilier, obstacole).

Totuși, recunoașterea obiectelor în această lucrare este relativ limitată, fiind bazată pe clasificatori tradiționali sau modele pre-antrenate de complexitate redusă, ceea ce poate duce la o rată crescută de erori în medii aglomerate sau slab iluminate. Cu toate acestea, sistemul oferă un echilibru remarcabil între performanță și accesibilitate, fiind conceput pentru a funcționa offline și pe hardware cu resurse reduse, cum este Raspberry Pi.

Un pas important către îmbunătățirea robusteței unui astfel de sistem este discutat în [6], unde autorii propun integrarea unui modul de detecție a neclarității imaginii (blur detection). Acest modul folosește metode bazate pe OpenCV pentru a evalua claritatea imaginii înainte de a trece la procesarea ei prin OCR sau CNN. Dacă imaginea este considerată neclară (de exemplu, din cauza mișcării camerei sau a luminii slabe), sistemul notifică utilizatorul și solicită o captură nouă. Acest pas adaugă o verificare importantă în lanțul de procesare și previne generarea de rezultate eronate sau confuze.

Din punct de vedere al fluxului de date, integrarea funcționalităților de detecție și vocalizare presupune mai mulți pași critici:

1. Captarea imaginii cu o cameră video;
2. Eventuală pre-procesare pentru claritate și contrast (binarizare, denoising);
3. Verificarea clarității (blur detection);
4. Detecția obiectelor și extragerea textului;
5. Generarea de descrieri sintetizate și citirea acestora prin TTS.

Un aspect interesant este alegerea formatului de feedback vocal. Sistemele avansate încearcă să combine informațiile într-un mesaj coerent pentru utilizator, de exemplu: *"Pe masă se află o cană și un telefon. Textul vizibil este: 'Nu atingeți!'"*. Acest tip de feedback contextual este mult mai util decât enumerarea separată a obiectelor și textului.

Provocările care apar în aceste sisteme includ:

- **Timpii de procesare** – detecția de obiecte și OCR sunt operații intensive; optimizarea acestora pentru timp real este esențială;
- **Gestionarea erorilor** – neclaritatea, suprapunerea obiectelor sau contrastul redus pot duce la eșecuri în detecție;
- **Designul interfeței audio** – mesajele trebuie să fie clare, scurte și adaptate contextului, pentru a nu suprasolicita utilizatorul.

Lucrările analizate arată că adăugarea unui strat suplimentar de filtrare și verificare (precum blur detection) poate îmbunătăți semnificativ experiența utilizatorului. Mai mult, adaptarea mesajelor vocale în funcție de poziția obiectelor (ex: „în față", „în stânga", „pe podea") crește nivelul de utilitate în navigare.

Pentru lucrarea de față, este recomandată o abordare integrată, în care toate modulele (OCR, CNN, blur detection, TTS) colaborează pentru a oferi o descriere contextuală și coerentă a mediului. Această arhitectură va permite crearea unui asistent vizual autonom, capabil să funcționeze în timp real și să furnizeze informații utile și precise utilizatorului nevăzător.

3.3. Rețele neuronale convoluționale (CNN) pentru nevăzători

Lucrarea [7] prezintă un sistem integrat pe Raspberry Pi care utilizează o rețea CNN antrenată pe setul de date COCO, combinată cu module OCR și TTS. Ce îl face remarcabil este integrarea unui canal de comunicare de urgență (prin Telegram), care poate trimite locația utilizatorului în caz de obstacol sau pericol detectat. Deși soluția este inovatoare, complexitatea implementării și dependența de conexiune mobilă sunt limitări de luat în considerare.

În [8], autorii propun o abordare multimodală, prin combinarea unui CNN pentru recunoaștere de obiecte, a unui modul OCR și a unui sintetizator vocal (gTTS), toate rulate pe un Raspberry Pi 3. În plus, dispozitivul include un senzor ultrasonic care oferă informații despre distanță față de obstacole, iar controlul vocal este realizat prin integrarea cu Mycroft AI. Sistemul este unul complex și promițător, însă utilizarea simultană a mai multor componente poate limita performanța pe hardware-ul limitat al unui

Raspberry Pi.

O altă lucrare interesantă este [9], care propune un sistem de recunoaștere a obiectelor în timp real, folosind un CNN antrenat pe ImageNet. Feedback-ul oferit utilizatorului este sub formă audio, iar opțional sistemul poate trimite rezultatul și către un ecran Braille. Acest tip de abordare este valoros în scenarii educaționale sau în contexte unde textul scris nu este esențial, dar orientarea în spațiu este prioritară. Totuși, lipsa unui modul OCR reduce utilitatea sistemului în medii cu texte (semne, etichete, afișe etc.).

Lucrarea [10] se axează pe utilizarea modelului YOLO (You Only Look Once), cunoscut pentru viteza ridicată de detecție în timp real. Este implementat împreună cu Google TTS pentru a oferi feedback vocal despre obiectele detectate și poziționarea lor relativă (ex: „Scaun în stânga", „Persoană în față"). Este o abordare eficientă pentru mobilitate urbană, însă limitarea o constituie lipsa recunoașterii textului, ceea ce face ca sistemul să nu poată fi folosit în situații precum citirea etichetelor de produse sau a panourilor informative.

În [11] este descris un sistem de tip wearable bazat pe Raspberry Pi 4B, care utilizează modele avansate LVLM (Large Vision-Language Models). Acestea permit nu doar recunoașterea de obiecte, ci și înțelegerea contextului în care acestea apar. Sistemul este capabil să răspundă la întrebări de tipul „Ce se întâmplă în imagine?", „Ce obstacol am în față?" sau „Unde este ieșirea?". Acest nivel de interacțiune aduce o nouă dimensiune în asistența pentru nevăzători. Cu toate acestea, LVLM-urile necesită putere de calcul ridicată și memorie, ceea ce limitează implementarea eficientă pe platforme embedded fără componente hardware suplimentare (GPU extern, NPU dedicat etc.).

Analizând aceste lucrări, observăm că majoritatea soluțiilor se încadrează în una din următoarele două categorii:

- Sisteme simple, cu CNN de bază și feedback vocal (YOLO, ImageNet), bune pentru recunoaștere rapidă de obiecte.
- Sisteme complexe, care combină CNN, OCR, TTS și alți senzori (ultrasunete, microfoane, GPS) pentru o experiență completă, dar care necesită mai multă energie, putere de calcul și resurse software.

Avantajul implementării unui CNN customizat, antrenat pe date specifice scenariului românesc (semnalistică, obiecte comune, texte în limba română), ar aduce o valoare adăugată față de soluțiile prezentate. De asemenea, un model CNN optimizat cu TensorFlow Lite sau PyTorch Mobile poate fi adaptat pentru rulare eficientă pe Raspberry Pi, reducând latența și consumul.

În concluzie, integrarea unei rețele CNN bine antrenate, într-un sistem embedded cu OCR și TTS poate duce la un asistent vizual portabil și eficient, oferind o autonomie reală utilizatorilor nevăzători în viața de zi cu zi.

Articol	**An**	**Descriere**
RaspberryTextReader	2024	Sistem complet OCR+TTS+CNN pe Raspberry Pi; recunoaște text și obiecte, funcționează offline.
AIWear	2023	Dispozitiv dedicat pentru studenți nevăzători; Tesseract OCR + TTS, accent pe ușurința utilizării.
ImageOCRTTS	2023	OCR cu Tesseract și redare vocală prin Google Speech API; depinde de conexiune internet.
RealTimeTTS	2024	Sistem TTS offline cu OCR și pre-procesare inteligentă; funcționează în timp real.
PiTextReaderVoice	2024	Integrează OCR, detecție obiecte și TTS într-un singur sistem pentru descrieri audio contextuale.
ImageToSpeechBlur	2023	Adaugă detecție de neclaritate pentru a preveni capturile eronate; crește acuratețea OCR.
CNNVisuallyImpaired	2023	CNN antrenat pe COCO + OCR + TTS și alertă prin Telegram; sistem integrat pe Raspberry Pi.
ArtificialEye	2023	Sistem multimodal cu CNN, OCR, TTS și senzor ultrasonic; interfață vocală prin Mycroft AI.
CNNLiveBlindAid	2018	Recunoaștere de obiecte cu CNN și feedback audio/Braille; axat pe orientare în spațiu.
YOLOVocal	2023	Folosește YOLO pentru detecție rapidă de obiecte și poziționare vocală; fără OCR.
AIWearableVision	2024	Utilizează modele LVLM pentru descriere contextuală; avansat dar cu cerințe hardware ridicate.

Tabela 3.1: Sinteză a lucrărilor analizate privind sisteme OCR + TTS + CNN

3.4. Concluzie

Majoritatea lucrărilor propun soluții bazate pe Tesseract pentru OCR și folosesc diverse API-uri pentru TTS. Doar câteva articole abordează integrarea completă a CNN pentru obiecte, OCR și TTS pe același dispozitiv embedded. În lucrarea de față, se propune tocmai această integrare, împreună cu optimizări pentru pre-procesare și feedback vocal clar.

Capitolul 4. Analiză și fundamentare teoretică

4.1. Analiza componentelor hardware

Selecția componentelor hardware s-a realizat având în vedere cerințele unui sistem portabil, eficient energetic și capabil să execute inferență AI locală. Configurația finală utilizează versiuni compacte și optimizate ale plăcii Raspberry Pi și ale accesoriilor sale, permițând integrarea întregului sistem într-un format redus, potrivit pentru montarea pe ochelari sau într-un dispozitiv purtabil.

- **Raspberry Pi Zero 2 W**:
  - Echipată cu un procesor quad-core ARM Cortex-A53 la 1 GHz și 512 MB RAM, suficient pentru rularea unui model CNN convertit în format TFLite și motorului OCR Tesseract.
  - Consum energetic extrem de redus, ceea ce o face ideală pentru alimentare portabilă prin baterie externă.
  - Suport integrat pentru WiFi și Bluetooth, facilitând transferul modelului sau debugging fără cabluri suplimentare.
  - Dimensiuni foarte mici (65 mm × 30 mm), ceea ce permite o integrare facilă în carcase discrete.
- **Camera Raspberry Pi Camera Module 3**:
  - Cameră de înaltă rezoluție (12MP) cu autofocus, conectată prin interfața CSI.
  - Compatibilitate directă cu Raspberry Pi Zero 2 W, oferind o captură clară și rapidă, esențială pentru OCR și detecție obiecte.
  - Folosirea interfeței CSI permite reducerea latenței și eliberarea porturilor USB.
- **Cablu flex AZDelivery de 50 cm (compatibil Pi Zero)**:
  - Permite montarea flexibilă a camerei la distanță față de placă, fiind ideal pentru proiecte purtabile sau montate pe ochelari.
  - Compatibil cu interfața CSI pentru Raspberry Pi Zero, asigurând conexiune sigură și fiabilă.
- **Card de memorie Intenso Micro SDXC 64 GB Class 10**:
  - Capacitate mare de stocare, suficientă pentru sistemul de operare, biblioteci, modelul antrenat, aplicație Python și eventuale capturi.
  - Clasa 10 asigură viteze adecvate de citire/scriere pentru funcționare fluidă.
  - Include adaptor SD pentru scriere facilă inițială de pe PC.
- **PowerBank 10000 mAh**:
  - Asigură o autonomie de câteva ore (în funcție de consumul mediu de 0.5-1A al sistemului complet).
  - Portabil și ușor de reîncărcat, eliminând dependența de o sursă fixă de alimentare.
  - Unele modele oferă și funcție de pornire automată la detectarea curentului – ideal pentru scenarii fără intervenție manuală.
- **Buton fizic GPIO**:
  - Conectat la un pin GPIO și configurat pentru a permite comutarea între cele două moduri principale: detecție obiecte (CNN) și recunoaștere text (OCR).

  - Implementarea unui mecanism de *debounce* software previne activările accidentale.
  - Este esențial pentru interacțiunea hands-free a utilizatorului nevăzător.
- **Boxe USB (sau audio analog prin jack 3.5mm)**:
  - Folosite pentru redarea mesajelor generate de motorul de sinteză vocală (TTS), precum `espeak-ng`.
  - Oferă feedback audio clar, imediat, adaptat uzului nevăzătorilor.
  - În funcție de configurație, se poate folosi și o cască mono discretă.

Această configurație combină portabilitatea cu performanța minim necesară pentru procesare vizuală embedded, fără a compromite autonomia sau interactivitatea. Sistemul este gândit să funcționeze complet offline, fiind astfel potrivit pentru scenarii din viața reală în care conectivitatea nu este garantată.

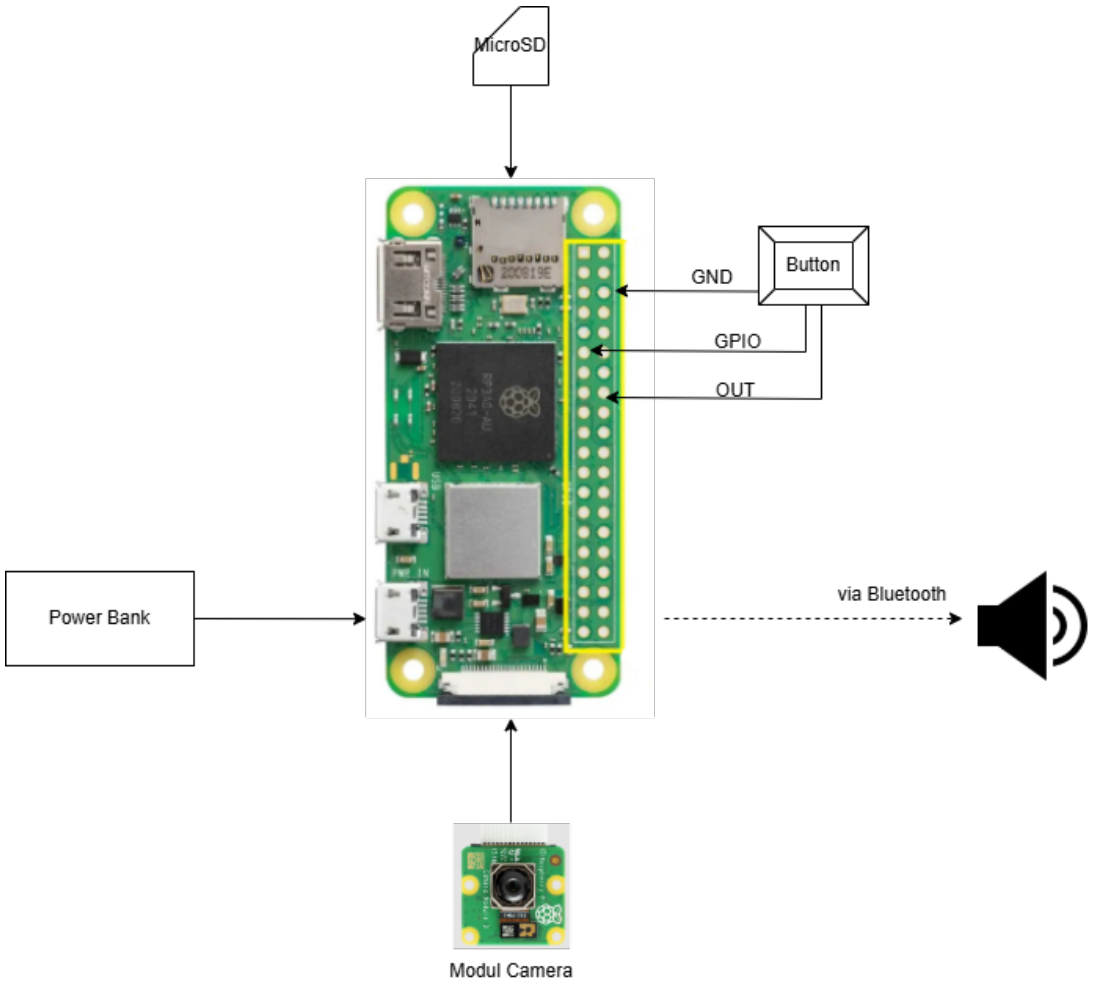

Figura 4.1: Configurația componentelor hardware utilizate

4.2. Analiza procesului de trimitere a imaginilor prin cablul FPC

Modulul de cameră conectat la placa Raspberry Pi utilizează un cablu FPC (Flexible Printed Circuit) de tip HBV–RPI–500FPC pentru a transmite datele video capturate către procesorul plăcii. Acest tip de cablu este subțire, flexibil și prevăzut cu conectori speciali compatibili cu interfața CSI (Camera Serial Interface) a plăcii Raspberry Pi.

4.2.1. Structura conexiunii

Cablul HBV–RPI–500FPC este alcătuit dintr-o serie de trasee conductoare paralele, încastrate într-un substrat flexibil. Acesta face legătura electrică între senzorul de imagine din modulul cameră (ex. OV5647) și placa Raspberry Pi, prin portul dedicat CSI (situat de obicei lângă portul HDMI mini sau GPIO).

4.2.2. Protocolul de comunicare

- Cablul transmite datele utilizând protocolul MIPI CSI-2 (Mobile Industry Processor Interface – Camera Serial Interface).
- Acesta este un protocol de mare viteză, proiectat special pentru transferul de date video între senzori de imagine și procesoare ARM/mobile.
- Semnalul video transmis prin cablu include atât datele brute de imagine (frame-uri) cât și semnale de control și sincronizare.

4.2.3. Fluxul de date

1. Senzorul de imagine captează un cadru optic și îl convertește în date digitale (RAW sau format comprimat, în funcție de configurație).
2. Datele sunt transmise prin cablul HBV–RPI–500FPC, folosind linii dedicate pentru tact (clock) și date (data lanes), conform specificațiilor CSI-2.
3. Interfața CSI a plăcii Raspberry Pi recepționează semnalele și le transmite mai departe către subsistemul GPU sau CPU pentru prelucrare.
4. Imaginea este apoi disponibilă pentru procesare software, fie în aplicații bazate pe `libcamera`, fie pentru inferență cu un model de inteligență artificială.

4.2.4. Rolul cablului HBV–RPI–500FPC

Fără cablul FPC, comunicarea între cameră și placa Raspberry Pi nu ar fi posibilă. Acesta asigură:

- Conexiune mecanică sigură și flexibilă;
- Transmiterea fiabilă a semnalelor digitale de mare viteză;
- Compatibilitate electrică între modulul de cameră și interfața CSI a plăcii.

Deși pare un simplu cablu plat, HBV–RPI–500FPC este o componentă critică în arhitectura sistemului, permițând ca informația vizuală să ajungă de la mediul exterior până în unitatea de procesare, unde este analizată și convertită în feedback vocal pentru utilizator.

4.3. Configurarea sistemului de operare

4.3.1. Alegerea platformei software

Pentru implementarea proiectului s-a utilizat sistemul de operare **Raspberry Pi OS Lite**, o versiune minimală bazată pe Debian, optimizată pentru rulare headless (fără interfață grafică). Această alegere a fost motivată de:

- **Consum redus de resurse** – sistemul ocupă puțin spațiu pe cardul SD și consumă memorie minimă.
- **Timp de pornire redus** – ideal pentru un dispozitiv portabil ce trebuie să devină funcțional rapid.
- **Acces direct la periferice** – precum interfața CSI (cameră), pini GPIO (buton) și modulele wireless (WiFi și Bluetooth).

4.3.2. Configurări esențiale și instalarea bibliotecilor necesare

După scrierea imaginii Raspberry Pi OS Lite pe cardul SD, au fost realizate următoarele configurații și instalări esențiale pentru funcționarea sistemului:

- **Activarea camerei CSI**:

  ```
  sudo raspi-config
  # -> Interface Options -> Enable Camera
  ```

  Această opțiune permite utilizarea camerei Raspberry Pi Module 3 prin interfața CSI.
- **Actualizarea pachetelor de sistem**:

  ```
  sudo apt update && sudo apt upgrade -y
  ```

  Această comandă asigură că sistemul de operare este sincronizat cu cele mai noi versiuni ale bibliotecilor și dependențelor.
- **Instalarea OpenCV** (procesare imagine):

  ```
  sudo apt install python3-opencv -y
  ```

  Biblioteca OpenCV este utilizată pentru capturarea și procesarea imaginilor (redimensionare, conversie, afisare, etc.).
- **Instalarea TensorFlow Lite Runtime** (inferentă AI):

  ```
  python3 -m pip install tflite-runtime
  ```

  Această bibliotecă permite încărcarea și rularea modelelor TFLite pe Raspberry Pi, fiind mai ușoară decât TensorFlow complet.
- **Instalarea Pytesseract** (OCR):

  ```
  sudo apt install tesseract-ocr -y
  python3 -m pip install pytesseract
  ```

Pachetul Tesseract realizează recunoașterea textului în imagini, iar `pytesseract` este interfața Python către motorul OCR.

- **Instalarea PIL/Pillow** (manipulare imagini):

```
python3 -m pip install Pillow
```

Biblioteca Pillow (fork de PIL) este folosită pentru manipularea imaginilor — conversii, redimensionări, deschidere/salvare.

- **Instalarea espeak-ng** (sinteză vocală):

```
sudo apt install espeak-ng -y
```

Espeak-ng este un sintetizator vocal offline, utilizat pentru a reda mesajele detectate într-o formă auditivă.

- **Instalarea numpy** (operații numerice):

```
python3 -m pip install numpy
```

Biblioteca `numpy` este utilizată în pregătirea imaginilor pentru modelul AI — conversie în array, normalizare, etc.

- **Instalarea Geany (editor cod)** – opțional:

```
sudo apt install geany -y
```

Geany este un IDE lightweight, util pentru editarea și rularea scripturilor Python direct pe Raspberry Pi.

4.3.3. Configurarea Bluetooth

Dacă modelul `tacamuri_model.tflite` este antrenat pe un computer personal și transferat pe Raspberry Pi, Bluetooth poate fi utilizat astfel:

```
sudo apt install bluetooth bluez blueman -y
sudo systemctl enable bluetooth
blueman-manager  # sau folosirea liniei de comandă pentru pairing
```

După împerecherea dispozitivului, fișierul poate fi transferat direct sau montat într-un folder temporar.

4.4. Alegerea si configurarea modelului CNN pentru sisteme embedded

4.4.1. Crearea modelului CNN

Pentru sarcina de clasificare a obiectelor de uz casnic (*furculițe, cuțite, farfurii, linguri*) într-un context embedded, s-a construit un model de rețea neuronală convoluțională (CNN) personalizat, optimizat atât pentru acuratețe cât și pentru eficiență computațională.

Arhitectura aleasă este minimalistă, fiind alcătuită dintr-o succesiune de straturi convoluționale și de pooling urmate de straturi dense. Aceasta se bazează pe principii esențiale din învățarea profundă pentru procesarea imaginilor: extragerea de trăsături spațiale (prin convoluții), reducerea dimensionalității și păstrarea informației esențiale (prin max pooling) și clasificarea globală (prin fully-connected layers).

Descriere arhitecturală:

- **Input layer**: Primește imagini color RGB de dimensiune fixă $224 \times 224 \times 3$, standardizând astfel intrările pentru model.
- **Stratul convoluțional 1**: Aplică 16 filtre 3×3, care extrag contururi și texturi de bază. Activarea folosită este ReLU (Rectified Linear Unit), care introduce non-linearitate și reduce riscul de gradient vanish.
- **Pooling 1**: Un strat *MaxPooling2D* care reduce rezoluția caracteristicilor extrase, menținând cele mai importante activări și micșorând dimensiunea inputului pentru următorul strat.
- **Stratul convoluțional 2**: 32 filtre 3×3 pentru captarea unor caracteristici mai complexe, precum forme și muchii multiple.
- **Pooling 2**: Aplicație suplimentară de *max pooling*, consolidând generalizarea modelului.
- **Flatten**: Conversia volumului tridimensional rezultat într-un vector unidimensional.
- **Dense 1**: Un strat dens cu 64 neuroni ReLU, ce funcționează ca decident pe baza trăsăturilor extrase.
- **Output**: Strat final de 4 neuroni, corespunzători claselor, cu activare `softmax`, returnând probabilități pentru fiecare clasă.

Această arhitectură este concepută special pentru a avea o amprentă de memorie redusă, esențială pentru rularea pe Raspberry Pi fără acceleratoare hardware dedicate.

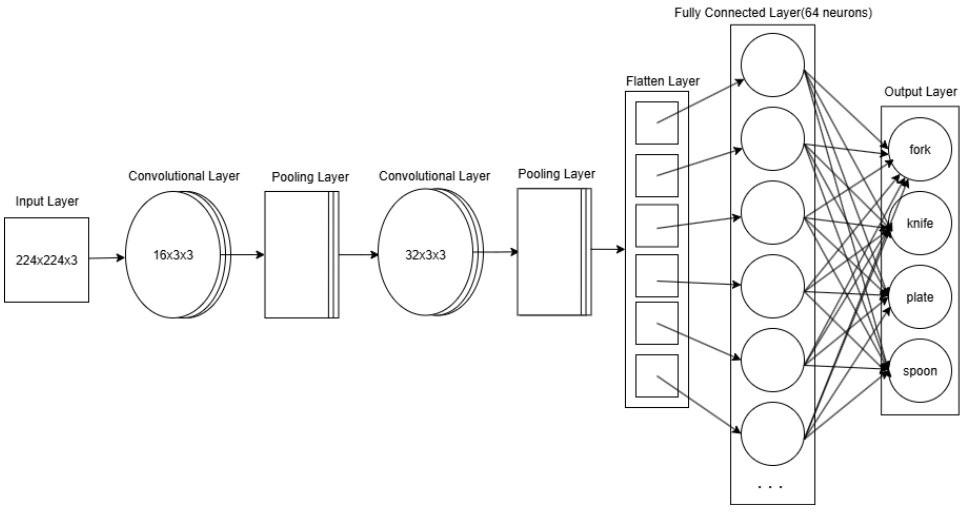

Figura 4.2: Arhitectura CNN

4.4.2. Antrenarea modelului

Antrenarea rețelei convoluționale (CNN) s-a realizat pe un computer personal cu următoarea configurație hardware: procesor Intel Core i7 (8 nuclee, 2.6 GHz), memorie RAM de 16 GB și sistem de operare Windows 10, fără utilizarea unui accelerator GPU dedicat. Setul de date utilizat a fost unul personalizat, structurat în subfoldere corespunzătoare fiecărei clase (`forks`, `knife`, `plates`, `spoons`). Timpul total necesar pentru antrenarea modelului, desfășurată pe parcursul a 10 epoci complete, a fost de aproximativ 25 de minute.

După finalizarea procesului de antrenare în mediul de dezvoltare Python, modelul a fost salvat în format `.h5`, iar ulterior a fost convertit în format `.tflite` pentru a putea fi implementat eficient pe placa Raspberry Pi, optimizând astfel viteza de inferență și reducând cerințele de memorie.

Imaginile au fost prelucrate cu ajutorul clasei `ImageDataGenerator`, care permite redimensionarea automată a acestora la rezoluția necesară modelului (224x224 pixeli), normalizarea valorilor pixelilor și, opțional, aplicarea unor tehnici de augmentare pentru creșterea diversității datelor.

Configurația procesului de antrenare:

- **Normalizare**: Toate imaginile sunt scalate în intervalul $[0, 1]$ prin împărțirea valorilor pixelilor la 255, ceea ce facilitează o convergență mai rapidă a modelului.
- **Batch size**: Stabilit la 8, pentru a asigura un echilibru între viteza de antrenare și stabilitatea gradientului.
- **Funcția de pierdere**: `sparse_categorical_crossentropy`, potrivită pentru etichete numerice (non one-hot) și probleme de clasificare multi-clasă.
- **Optimizator**: `Adam`, un algoritm care combină avantajele metodelor *AdaGrad* și *RMSprop*, adaptând rata de învățare individual pentru fiecare parametru.
- **Epoci**: 10 cicluri complete de antrenare pe toate datele.
- **Metodă de validare**: Utilizarea unui set de validare separat pentru monitorizarea erorii de generalizare și ajustarea hiperparametrilor.

Procesul de antrenare a fost realizat folosind biblioteca TensorFlow, rulată pe CPU, iar durata totală raportată confirmă fezabilitatea utilizării acestui tip de arhitectură pe sisteme de calcul de nivel mediu.

4.4.3. Conversia modelului în format TensorFlow Lite – detalii intern

Conversia unui model TensorFlow salvat în format standard (`.h5` sau `SavedModel`) în format TensorFlow Lite (`.tflite`) reprezintă un proces esențial pentru rularea eficientă pe dispozitive embedded, cum este Raspberry Pi Zero 2 W. Dincolo de simplul apel al unui API în Python, în spatele acestei conversii au loc mai multe transformări interne, descrise detaliat mai jos.

Structura modelului standard

Modelul TensorFlow standard conține:

- Arhitectura completă a rețelei neuronale (strat cu strat, conexiuni, activări).
- Greutățile antrenate.
- Funcții de pierdere, optimizatori și metrici folosite în timpul antrenării.
- Posibilitatea continuării antrenării, inclusiv istoricul epoch-urilor.

Acest format este ideal pentru dezvoltare și experimentare, dar este prea voluminos și complex pentru execuția pe dispozitive cu resurse reduse.

Ce presupune conversia spre TensorFlow Lite

Conversia în format `.tflite` face următoarele:

- **Elimină graficul de antrenare** – păstrează doar calea de inferență, fără propagare inversă.
- **Optimizează graficul de calcul** – rearanjează operațiile interne pentru a minimiza consumul de memorie și calculele redundante.
- **Comprimă greutățile și activările** – permite reducerea preciziei numericelor (ex: din float32 în int8), ceea ce scade dimensiunea modelului și crește viteza de execuție.
- **Generează un fișier binar compact** – totul este salvat într-un format secvențial, pregătit pentru un interpreter minimalist.

Cum funcționează un fișier `.tflite`

Spre deosebire de un model TensorFlow clasic, un fișier `.tflite`:

- Este încărcat de un interpreter dedicat (`TensorFlow Lite Interpreter`), mult mai mic decât biblioteca TensorFlow completă.
- Alocă direct blocuri de memorie (tensori) pentru input și output, fără construirea unui grafic dinamic.
- Rulează predicțiile secvențial și rapid, direct pe hardware-ul local, fără dependențe externe.

Importanța pentru Raspberry Pi

Raspberry Pi are resurse limitate:

- Memorie RAM de doar 512MB (în cazul Pi Zero 2 W).
- CPU ARM Cortex, care nu este optimizat pentru calcule float intensive.
- Fără GPU dedicat sau acceleratoare hardware pentru AI.

Un model `.h5` ar necesita instalarea completă a TensorFlow, consumând multă memorie și timp de execuție. În schimb, un model `.tflite`:

- Se încarcă cu `tflite_runtime`, o bibliotecă mică și optimizată.
- Funcționează rapid, chiar și pe ARM, în regim offline.

- Permite inferență locală, cu timpi de răspuns de ordinul sutimilor de secundă.

Concluzie

Conversia în format TensorFlow Lite nu este doar un pas de compresie, ci o transformare profundă, care adaptează modelul AI pentru rulare embedded. Aceasta presupune eliminarea componentelor inutile, optimizarea calculelor și generarea unui format binar, special conceput pentru interpretarea rapidă pe hardware limitat. Fără acest proces, rularea unui model antrenat pe desktop pe Raspberry Pi ar fi practic imposibilă, din cauza limitărilor de memorie, procesare și energie.

4.4.4. Transferul modelului pe Raspberry Pi prin Bluetooth

După conversia modelului CNN în format `.tflite`, fișierul rezultat a fost pregătit pentru transferul către placa Raspberry Pi Zero 2 W. Această etapă a avut ca obiectiv punerea la dispoziția sistemului embedded a versiunii optimizate a modelului, capabilă să ruleze inferența cu un consum redus de resurse.

Procesul de transfer s-a desfășurat astfel:

- Modelul convertit a fost salvat local pe computer cu denumirea `tacamuri_model.tflite`
- Raspberry Pi a fost configurat cu serviciul `bluetoothd` activ, permițând conexiunea prin protocolul OBEX (Object Exchange).
- Pe computer s-a utilizat un manager grafic de fișiere cu suport pentru trimiterea fișierelor prin Bluetooth (*Send To* sau aplicații dedicate precum `Blueman`), selectând placa Raspberry Pi din lista de dispozitive asociate.
- După inițierea transferului, Raspberry Pi a generat o solicitare de acceptare a fișierului, confirmată local.
- Fișierul a fost salvat automat într-un director prestabilit (`/home/pi/Downloads`).
- Ulterior, fișierul a fost mutat în directorul proiectului Python, unde scriptul de inferență îl accesează în timpul rulării.

Această metodă de transfer prin Bluetooth a prezentat avantajul eliminării necesității cablurilor suplimentare și a facilitat testarea rapidă a modelului pe dispozitivul embedded. Totodată, procesul poate fi repetat ori de câte ori modelul este reantrenat sau modificat, asigurând flexibilitate în dezvoltare.

4.4.5. Justificarea deciziilor tehnice și avantajele abordării alese

Pentru un proiect orientat spre aplicații embedded cu inteligență artificială, alegerea unei arhitecturi CNN personalizate, de dimensiuni reduse și complexitate controlată, s-a dovedit a fi esențială pentru a asigura o execuție fluidă pe hardware cu resurse limitate. În cazul de față, dispozitivul Raspberry Pi Zero 2 W nu oferă capabilități avansate de accelerare hardware (GPU sau TPU), astfel că s-a impus conceperea unui model adaptat acestui tip de arhitectură — capabil să funcționeze eficient în condiții reale, cu latență redusă și fără încărcări inutile asupra procesorului.

În locul unor rețele neuronale adânci, cu zeci de milioane de parametri, modelul adoptat în această lucrare prioritizează simplitatea algoritmică și eficiența computațională, fără a compromite capacitatea de generalizare pe seturi de date relevante. Scopul a fost obținerea unei inferențe stabile, predictibile și rapid executabile pe dispozitiv, chiar și în condiții în care alimentarea, iluminarea sau conectivitatea pot varia semnificativ.

Strategia de separare a etapei de antrenare (realizată pe un sistem desktop) de cea de inferență (executată pe placa embedded) se aliniază celor mai bune practici actu-

ale din industrie, permițând nu doar optimizarea resurselor disponibile, ci și o mai bună trasabilitate și gestionare a modelelor AI. În timpul antrenării, dezvoltatorul are acces la instrumente sofisticate de analiză a performanței, ajustare a hiperparametrilor, monitorizare a pierderii și validare pe seturi extinse, procese care nu ar fi posibile direct pe un microcalculator ca Raspberry Pi.

Conversia ulterioară a modelului în format `.tflite` (TensorFlow Lite) nu doar că reduce dimensiunea binarului, dar optimizează și timpii de rulare, prin includerea unor instrucțiuni specifice arhitecturii ARM. Astfel, rezultatul este un fișier rapid de executat, portabil și ușor de distribuit, care poate fi încorporat în orice aplicație embedded fără a afecta semnificativ utilizarea generală a sistemului.

Transferul modelului către Raspberry Pi prin Bluetooth reprezintă o soluție practică și elegantă, care elimină necesitatea conectării fizice sau a configurării unor protocoale de rețea avansate. Acest tip de abordare susține un proces de actualizare continuu, permițând dezvoltatorului să îmbunătățească modelul în mod incremental, în funcție de feedback-ul din teren sau de noile seturi de date adunate. În plus, întreaga arhitectură software a proiectului este concepută modular, astfel încât versiuni noi ale modelului pot fi înlocuite fără intervenții asupra logicii de bază sau interfeței cu utilizatorul.

O caracteristică esențială a acestei metode este compatibilitatea cu aplicațiile mobile, purtabile sau autonome. Sistemul rezultat este ușor de integrat în medii cu restricții de alimentare sau conectivitate — precum în cazul purtării de către un utilizator nevăzător aflat în mișcare. Funcționarea în regim complet offline, fără a necesita internet, este un avantaj considerabil în contextul utilizării în exterior sau în spații fără semnal.

Pe lângă eficiența în execuție, această abordare permite o gamă largă de extensii ulterioare. De exemplu, același flux de lucru poate fi reutilizat pentru dezvoltarea unor modele noi, specializate pe alte categorii de obiecte sau adaptate altor limbi, fără a modifica infrastructura deja implementată. Scalabilitatea procesului este susținută de posibilitatea replicării sistemului pe mai multe dispozitive, ceea ce îl face ideal pentru testări paralele sau pentru distribuție către mai mulți utilizatori finali.

Un alt avantaj strategic îl constituie independența față de infrastructura de cloud. În multe cazuri, soluțiile bazate pe servere externe devin inutilizabile în lipsa unei conexiuni stabile, ceea ce nu este acceptabil pentru aplicații critice de asistență personală. Sistemul descris aici își păstrează toate funcționalitățile chiar și în condiții complet deconectate, oferind utilizatorului o experiență de încredere și disponibilitate constantă.

În concluzie, abordarea adoptată — antrenarea modelului pe un sistem performant, conversia într-un format optimizat și rularea sa pe un microcontroler embedded — oferă o combinație ideală între robustețe, flexibilitate și autonomie. Aceasta transformă o tehnologie avansată într-un instrument accesibil, personalizabil și funcțional, adaptat nevoilor reale ale utilizatorilor din viața cotidiană. Modelul devine nu doar o componentă software performantă, ci un element cheie într-un ecosistem inteligent care sprijină autonomia persoanelor cu deficiențe de vedere.

4.5. OCR – recunoașterea optică a caracterelor

4.5.1. OCR(Optical Character Recognition) cu Tesseract

Recunoașterea optică a caracterelor (OCR) reprezintă procesul de extragere a textului din imagini. În cadrul acestui proiect, s-a utilizat motorul open-source **Tesseract**, recunoscut pentru eficiența sa și suportul pentru limba română. Tesseract folosește o combinație de algoritmi tradiționali de segmentare și rețele neuronale LSTM pentru recunoașterea caracterelor.

Funcționarea internă a Tesseract include următorii pași:

- Segmentarea imaginii în linii și caractere individuale.
- Normalizarea și transformarea caracterelor într-un format standardizat.
- Reconstituirea cuvintelor și aplicarea regulilor lingvistice.

4.5.2. Necesitatea preprocesării imaginilor

Pentru a obține rezultate cât mai precise, imaginile capturate de cameră trebuie preprocesate înainte de a fi transmise către motorul OCR. Această etapă este esențială deoarece textul poate apărea în condiții de iluminare slabă, zgomot vizual sau fundaluri neuniforme. Preprocesarea aplicată include:

1. **Conversie în scala de gri**:
   - Elimină informația de culoare, păstrând doar intensitatea luminoasă.
   - Se realizează cu `ImageOps.grayscale()`.
2. **Reducerea zgomotului prin blur median**:
   - Aplicația unui filtru median cu kernel 3×3 reduce zgomotul fără a afecta semnificativ marginile.
   - Implementat cu `ImageFilter.MedianFilter(size=3)`.
3. **Îmbunătățirea contrastului**:
   - Se utilizează `ImageOps.autocontrast()` pentru a întinde gama de intensități pe întreg spectrul 0–255.
4. **Binarizare prin prag adaptiv**:
   - Imaginea este convertită într-o formă alb-negru (`'1'`), aplicând o funcție lambda ce setează valoarea pixelului la 0 sau 255 în funcție de prag.
   - Pragul este calculat relativ: `0.5 * 255 = 127.5`.

Toate aceste transformări contribuie la crearea unui input curat, clar și bine delimitat, potrivit pentru segmentarea precisă a caracterelor de către Tesseract.

4.5.3. Configurarea motorului Tesseract

În apelul motorului OCR, s-a folosit o configurație personalizată pentru a adapta comportamentul Tesseract:

PSM (Page Segmentation Mode) 6 semnifică:

- Imaginea este tratată ca un singur bloc de text (ex: paragraf fără layout complex).
- Evită presupuneri despre coloane, anteturi sau subsoluri, ceea ce este ideal pentru imagini simple capturate în timp real.

De asemenea, s-a specificat limba română prin parametrii:

```
lang='ron'
```

Această setare permite recunoașterea diacriticelor și îmbunătățește acuratețea în cazul cuvintelor specifice limbii române.

4.5.4. Integrare cu sistemul vocal

După finalizarea procesului de recunoaștere optică a caracterelor (OCR), textul identificat în imagine este afișat inițial în consola sistemului, cu scopul de a permite verificări rapide de funcționare și depanare. Totuși, pentru utilizatorul final — o persoană cu deficiențe de vedere — interfața grafică nu este accesibilă. Prin urmare, este absolut necesar ca acest rezultat să fie transmis sub formă auditivă, în timp real. Acest obiectiv este atins prin integrarea cu un sintetizator vocal instalat local pe Raspberry Pi: `espeak-ng`.

`espeak-ng` este un motor de sinteză vocală open-source, ușor de instalat și configurat, care oferă suport pentru mai multe limbi, inclusiv româna. Principalul avantaj al acestui sintetizator este faptul că funcționează complet offline, fără a depinde de un API extern sau de conexiune la internet, ceea ce îl face ideal pentru un sistem embedded destinat utilizării mobile, în contexte variate (interior, exterior, rural etc.).

Avantaje ale acestei integrări:

- **Feedback instantaneu:** Textul este redat vocal în decurs de aproximativ o secundă de la captarea imaginii, oferind un răspuns prompt și natural pentru utilizator.
- **Funcționare offline:** Întregul proces de la captură până la sinteză este local, fără apeluri la servicii cloud. Acest lucru asigură independență tehnologică și robustețe în scenarii unde conectivitatea nu este garantată.
- **Personalizare flexibilă:** `espeak-ng` permite control asupra parametrilor vocali precum: tonalitatea (`-p`), volumul (`-a`), viteza (`-s`), pauzele între cuvinte și alegerea vocii sau a limbii. Această flexibilitate permite adaptarea vocalizării în funcție de preferințele utilizatorului sau de contextul de utilizare (ex: în medii zgomotoase).
- **Integrare multiplatformă:** Comenzile `espeak-ng` pot fi utilizate în aproape orice limbaj de programare care suportă execuție de comenzi sistemice, ceea ce face soluția versatilă și portabilă. De asemenea, biblioteca poate fi înlocuită ușor cu alte TTS-uri compatibile (de exemplu, `flite`, `festival` sau chiar `pyttsx3`).
- **Consum redus de resurse:** `espeak-ng` are un impact minim asupra memoriei și procesorului, permițând redarea simultană a textului fără întreruperea funcționării celorlalte module (ex: captură imagine, inferență CNN).

În concluzie, integrarea între motorul OCR și sintetizatorul vocal local reprezintă o componentă esențială a sistemului, transpunând rezultatul analizei vizuale într-o formă accesibilă și utilizabilă imediat de către nevăzători. Această legătură transformă modelul dintr-un simplu detector de text într-un adevărat asistent auditiv, funcțional, intuitiv și complet autonom.

4.5.5. Probleme întâlnite în procesarea textului

În ciuda utilității sale ridicate, modulul OCR întâmpină dificultăți notabile în scenarii reale. Calitatea imaginilor capturate de cameră, condițiile de iluminare, tipul fontului și poziționarea textului pot influența drastic acuratețea rezultatelor. Mai jos sunt prezentate cele mai frecvente probleme întâlnite, alături de posibile soluții tehnice:

- **Text neclar sau înclinat (skewed):** Atunci când camera nu este perfect aliniată cu planul textului, OCR-ul poate interpreta greșit caracterele. *Soluție:* Aplicarea unui algoritm de *deskewing* — corectarea unghiului textului — poate reorienta imaginea pe baza detecției liniilor orizontale. OpenCV oferă funcții precum Hough Transform și rotire automată pentru remedierea acestui caz.
- **Zgomot vizual sau fundal complex:** Imaginile cu elemente decorative, umbre, reflexii sau zgomot de fundal pot afecta segmentarea caracterelor. *Soluție:* Se recomandă o pre-procesare riguroasă a imaginii:
  - Aplicarea de filtre `Gaussian blur` pentru netezirea detaliilor inutile;
  - Conversia în `grayscale` urmată de `thresholding adaptiv` sau `Otsu binarization`;
  - Eliminarea fundalului prin operații morfologice (erodare, dilatare).

  Aceste etape pot îmbunătăți considerabil raportul semnal-zgomot, esențial pentru un OCR eficient.
- **Fonturi neuzuale, artistice sau scris de mână:** Tesseract oferă performanțe optime în cazul fonturilor standard, tipărite clar. În schimb, textele stilizate sau scrisul de mână prezintă provocări majore. *Soluție:*
  - Pentru fonturi artistice, antrenarea unui model OCR personalizat (cu `tesseract training tools`) poate aduce îmbunătățiri;
  - Pentru scrisul de mână, se pot integra rețele neuronale recurente (LSTM), antrenate special pentru recunoașterea literelor desenate neregulat;
  - Alternativ, se pot utiliza OCR-uri moderne bazate pe rețele de tip Transformer sau CRNN, deși acestea sunt mai costisitoare computațional și nu se încadrează în constrângerile unui sistem embedded.
- **Dimensiunea prea mică a textului:** Textul detectat trebuie să aibă un număr minim de pixeli pentru a putea fi recunoscut corect. *Soluție:* Se recomandă apropierea camerei sau folosirea unui obiectiv cu unghi de vizualizare mai îngust, pentru a mări densitatea caracterelor. De asemenea, imaginile pot fi *rescalate* digital, însă cu riscul de a introduce artefacte vizuale.
- **Lumină insuficientă sau variații puternice de iluminare:** Condițiile de iluminare au un impact direct asupra contrastului caracterelor. *Soluție:* Utilizarea unui LED de asistență pentru cameră sau aplicarea de *equalizare a histogramei* (ex: CLAHE – Contrast Limited Adaptive Histogram Equalization) pot crește claritatea caracterelor în imagini slab iluminate.

Pe scurt, în timp ce Tesseract oferă un punct de plecare solid pentru recunoașterea textului în aplicații embedded, performanța sa poate varia semnificativ în funcție de contextul vizual. Pre-procesarea inteligentă a imaginilor și adaptarea sistemului la condițiile reale sunt critice pentru obținerea unor rezultate consistente. În viitor, o direcție de cercetare promițătoare constă în combinarea OCR tradițional cu detectoare de regiuni textuale bazate pe rețele neuronale, capabile să izoleze mai precis porțiunile relevante din imagine, în vederea creșterii acurateței globale.

Capitolul 5. Proiectare de detaliu și implementare

5.1. Modularizarea aplicației – descriere detaliată

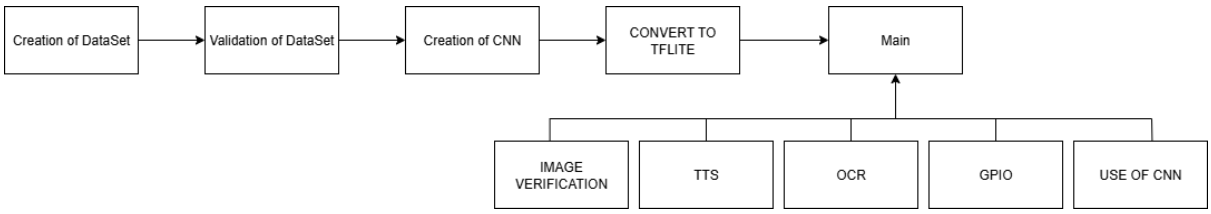

Figura 5.1: Diagrama de Dependențe intre Module

Aplicația este structurată în mai multe module independente, fiecare responsabil pentru o etapă esențială în procesul de detecție și interpretare vizuală. Această modularizare permite dezvoltarea separată, testarea individuală și eventuale îmbunătățiri fără a afecta restul sistemului. În continuare sunt prezentate principalele module ale aplicației.

5.1.1. Modulul de capturare a imaginilor

Pentru generarea setului de date necesar antrenării modelului de învățare automată, a fost utilizat un microcalculator Raspberry Pi echipat cu o cameră compatibilă. Capturarea imaginilor a fost realizată prin intermediul unui script scris în limbajul Python, care utilizează comanda `libcamera-still` pentru a efectua fotografii succesive.

Scriptul a fost conceput pentru a captura automat aproximativ 300 de imagini, pe care le salvează local cu denumiri unice, secvențiale (de exemplu, `img_0.jpg`, `img_1.jpg`, ..., `img_299.jpg`). Comanda folosită în mod repetat este:

```
libcamera-still -o img_X.jpg
```

unde `X` este un index numeric incrementat la fiecare captură. Pentru a executa această comandă din Python, s-a folosit modulul `os`, astfel încât fiecare apel la cameră să fie efectuat din interiorul unui loop controlat programatic. Pauza dintre capturi, impusă prin funcția `time.sleep()`, are rolul de a asigura stabilitatea camerei și claritatea imaginilor capturate, mai ales în condiții de iluminare slabă sau mișcare accidentală a dispozitivului.

După finalizarea capturii, fișierele imagine au fost transferate de pe Raspberry Pi către un computer, prin intermediul unei conexiuni Bluetooth. Acest pas a permis organizarea și procesarea ulterioară a imaginilor într-un mediu cu resurse mai adecvate pentru antrenarea modelului.

Transferul Bluetooth a fost realizat fie din interfața grafică a sistemului Raspbian, utilizând aplicația `blueman`, fie din linie de comandă prin `bluetoothctl`. Imaginile au fost grupate ulterior în directoare corespunzătoare fiecărei clase, conform cerințelor modelului de clasificare.

Această abordare automatizată permite generarea rapidă și eficientă a unui set de date personalizat, utilizabil direct în procesul de preprocesare și antrenare.

5.1.2. Modulul de preprocesare și validare a imaginilor

Un pas esențial în procesul de dezvoltare a unui model de învățare automată, în special în contextul imaginilor, îl reprezintă etapa de preprocesare și validare a datelor de intrare. Calitatea imaginilor utilizate pentru antrenarea unui model CNN influențează direct performanța acestuia în faza de inferență. În cadrul acestui proiect, modulul de preprocesare și validare a fost implementat pentru a garanta că toate imaginile utilizate sunt conforme cu cerințele arhitecturale ale rețelei și nu conțin erori care ar putea compromite procesul de antrenare.

Acest modul a fost executat pe un sistem desktop/laptop, după transferul imaginilor capturate cu ajutorul camerei Raspberry Pi. Imaginile au fost grupate manual în directoare separate, fiecare reprezentând o clasă (ex. `farfurie`, `lingură`, `furculiță`, `cuțit`). Această structurare ierarhică a fost necesară pentru a permite folosirea generatorului de date din biblioteca `Keras`, care așteaptă o organizare pe directoare corespunzătoare fiecărei etichete.

1. Verificarea extensiilor fișierelor

Primul pas a fost identificarea fișierelor care nu respectă formatul standard impus. Modelul CNN a fost antrenat exclusiv cu imagini în format `.jpg`, deoarece acest format este bine suportat de bibliotecile `PIL`, `OpenCV`, dar și de `ImageDataGenerator` din Keras. Astfel, toate fișierele cu extensii incompatibile (`.png`, `.jpeg`, `.gif`, `.bmp`) au fost fie convertite în formatul dorit, fie eliminate automat.

Pentru această etapă a fost scris un script Python simplu dar eficient, care parcurge recursiv directoarele și aplică următoarele reguli:

- Se acceptă doar fișiere cu extensia `.jpg`.
- Fișierele cu extensii greșite sunt listate și pot fi fie mutate într-un folder temporar, fie șterse.
- În cazul fișierelor fără extensie sau cu extensii multiple (ex: `.jpg.png`), acestea sunt considerate invalide.

Această curățare este vitală pentru prevenirea întreruperilor în timpul antrenării, mai ales când se utilizează metode automatizate de încărcare a datelor.

2. Verificarea integrității imaginilor

O a doua etapă importantă a fost validarea integrității fiecărui fișier de imagine. Este posibil ca unele fișiere să fie corupte în timpul capturii, al transferului Bluetooth sau al operațiilor de copiere între directoare. Un fișier imagine corupt se poate deschide parțial sau deloc și va genera erori la momentul încărcării în model.

Pentru acest pas, s-a folosit modulul `PIL` (Python Imaging Library), cu metoda `Image.verify()`, care încearcă să deschidă și să parcurgă conținutul fiecărei imagini fără a o încărca complet în memorie. Orice excepție întâlnită (ex: 'IOError', 'SyntaxError') a fost interpretată ca semn al unei imagini defecte, iar fișierul a fost raportat și mutat într-un folder special pentru investigații ulterioare.

Această etapă a oferit o garanție suplimentară că datele de intrare sunt valide și au structură completă, prevenind blocarea procesului de training din cauza unor imagini necorespunzătoare.

3. Asigurarea dimensiunii uniforme și a raportului de aspect

Modelul CNN a fost antrenat pe imagini de dimensiune fixă 224×224 pixeli. Imaginile brute capturate de cameră aveau însă dimensiuni variabile, în funcție de setările camerei sau de formatul senzorului. Înainte de antrenare, toate imaginile au fost redimensionate la dimensiunea de intrare acceptată de model, păstrând pe cât posibil raportul de aspect original, pentru a evita distorsiunile.

Redimensionarea s-a realizat în două etape:

- **Micșorare/scalare**: imaginea originală a fost scalată proporțional, astfel încât cea mai lungă dimensiune să se potrivească cu 224 pixeli.
- **Crop sau padding**: imaginea scalată a fost completată cu margini negre sau decupată central pentru a obține exact 224×224 pixeli.

Această abordare a permis utilizarea imaginilor variate fără pierderi semnificative de informație vizuală și fără a introduce artefacte de compresie.

4. Normalizarea valorilor pixelilor

În contextul rețelelor neuronale, valorile brute ale pixelilor (0–255) pot influența negativ stabilitatea antrenării. Astfel, imaginile au fost normalizate în intervalul $[0, 1]$ prin împărțirea fiecărui pixel la 255. Acest pas a fost realizat automat de obiectul `ImageDataGenerator`, configurat cu opțiunea `rescale=1./255`.

5. Rezultatul final al preprocesării

În urma acestor etape, s-a obținut un set de date curat, uniformizat și verificat, care conținea doar imagini în format `.jpg`, cu dimensiunea standardizată și fără fișiere corupte. Acest set de date a fost apoi împărțit în proporție de aproximativ 80% pentru antrenare și 20% pentru testare, pentru a permite evaluarea obiectivă a performanței modelului.

6. Beneficii și concluzii

Modulul de preprocesare și validare a avut un rol fundamental în crearea unui pipeline de antrenare robust și predictibil. Eliminarea datelor neconforme a contribuit la:

- Reducerea timpilor de antrenare;
- Prevenirea apariției erorilor în timpul execuției;
- Asigurarea unei performanțe mai bune a modelului CNN;
- Îmbunătățirea reproductibilității rezultatelor.

Într-un sistem AI destinat rulării pe dispozitive embedded, unde resursele sunt limitate și toleranța la erori este scăzută, un set de date corect pregătit este un pilon esențial al succesului implementării.

5.1.3. Modulul creare și antrenare a modelului CNN

Modulul de antrenare este responsabil pentru construirea, optimizarea și validarea unui model de rețea neuronală convoluțională (CNN), conceput pentru a clasifica obiecte de uz cotidian precum farfurii, linguri, furculițe și pahare. Acesta joacă un rol fundamental în asigurarea funcționalității sistemului, întrucât determină capacitatea de detecție vizuală a aplicației embedded.

Scopul acestui modul este de a produce un model de dimensiuni reduse, dar suficient de expresiv încât să poată diferenția clasele vizate, chiar și în condiții de variabilitate a imaginilor (iluminare, poziționare, fundal, unghiuri).

1. Setul de date.

Setul de date utilizat pentru antrenarea modelului a fost construit manual, prin captarea imaginilor folosind camera conectată la Raspberry Pi. Imaginile au fost colectate în mod sistematic pentru fiecare clasă, din diferite unghiuri, la distanțe variate și în condiții de iluminare diferite. Această diversitate a fost esențială pentru asigurarea unei capacități bune de generalizare a modelului.

Imaginile au fost ulterior transferate pe un calculator personal prin conexiune Bluetooth și organizate în directoare corespunzătoare fiecărei clase. Pentru a preveni dezechilibrul de date (care poate duce la învățarea preferențială a unor clase în detrimentul altora), s-a urmărit ca fiecare clasă să conțină un număr comparabil de imagini.

Datele au fost apoi împărțite în:

- **80% pentru antrenare** — utilizate pentru optimizarea greutăților rețelei.
- **20% pentru testare** — folosite exclusiv pentru evaluarea performanței modelului pe date „nevăzute”.

Această împărțire s-a realizat folosind funcționalitățile bibliotecii `ImageDataGenerator` din Keras, cu opțiunea `validation_split`, care asigură și o stratificare corespunzătoare a claselor.

2. Preprocesarea datelor

Pentru a asigura o intrare compatibilă cu modelul CNN, imaginile au fost:

- redimensionate la dimensiunea 224×224 pixeli;
- normalizate prin împărțirea valorilor pixelilor la 255 (intervalul devine $[0, 1]$);
- grupate în batch-uri mici (batch size = 32), pentru optimizarea memoriei RAM și accelerarea procesului de antrenare.

Augmentarea datelor (ex: rotiri, flip orizontal) nu a fost aplicată în versiunea finală a modelului, dar a fost testată în fazele inițiale fără o îmbunătățire semnificativă a acurateții.

3. Arhitectura rețelei CNN

Modelul CNN a fost implementat folosind biblioteca `Keras`, cu backend `TensorFlow`. Arhitectura propusă este una simplificată, adaptată pentru rularea pe dispozitive embedded cu resurse limitate. S-a evitat utilizarea de rețele foarte adânci (precum ResNet, Inception), deoarece acestea depășesc capacitățile hardware ale Raspberry Pi.

Structura aleasă este următoarea:

- `Conv2D(32 filtre, kernel 3x3)` + `ReLU`
- `MaxPooling2D(2x2)`
- `Conv2D(64 filtre, kernel 3x3)` + `ReLU`
- `MaxPooling2D(2x2)`
- `Flatten()`
- `Dense(64)` + `ReLU`

- `Dense(4)` + `Softmax`

Prin această arhitectură, rețeaua învață progresiv: de la trăsături simple (margini, unghiuri) la trăsături mai abstracte (forme, contururi de obiecte). Funcția de activare `ReLU` a fost aleasă pentru a introduce non-linearitate și pentru a preveni problema gradientului care dispare. Stratul final cu activare `softmax` returnează un vector de 4 probabilități, corespunzătoare fiecărei clase.

4. Parametrii de antrenare și performanță

Modelul a fost compilat folosind:

- `Loss`: `categorical_crossentropy` – deoarece etichetele au fost codificate one-hot;
- `Optimizator`: `Adam`, cu rată de învățare 0.001;
- `Metrici`: `accuracy`.

Antrenarea a fost realizată pe un computer personal (Intel Core i7, 8GB RAM), pe durata a 20 de epoci. S-a observat o creștere progresivă a acurateții, fără stagnări bruște sau overfitting. Curbele de învățare au confirmat stabilitatea modelului, iar pierderea s-a stabilizat într-un interval acceptabil.

5. Exportul modelului

Modelul antrenat a fost salvat în format `.h5`, apoi convertit în format `.tflite` pentru rulare eficientă pe placa Raspberry Pi. Procesul de conversie a eliminat graficul de antrenare și a generat un model optimizat pentru inferență, cu dimensiune redusă și timp de încărcare minim.

Concluzie

Modulul de antrenare reprezintă fundamentul inteligenței artificiale a sistemului propus. Alegerea unei arhitecturi simple, dar eficientă, corelată cu un set de date bine structurat și validat, a dus la un model CNN capabil să distingă obiectele de interes cu acuratețe ridicată. Acest model servește drept bază pentru inferența locală pe Raspberry Pi, fără a necesita conexiune la internet sau resurse externe, fapt ce îl face ideal pentru un sistem asistiv portabil, destinat persoanelor cu deficiențe de vedere.

5.1.4. Modulul de conversie pentru embedded (TFLite)

După finalizarea cu succes a procesului de antrenare și validare a modelului CNN pe un sistem de dezvoltare performant (laptop personal), urmează un pas esențial în adaptarea modelului pentru rulare în timp real pe un dispozitiv embedded precum Raspberry Pi Zero 2 W. Dată fiind natura limitată a resurselor hardware ale acestei platforme — atât din punctul de vedere al memoriei RAM (512MB), cât și al puterii de procesare CPU — utilizarea directă a modelului antrenat în formatul standard TensorFlow (`.h5` sau `SavedModel`) nu este fezabilă.

În acest context, este necesară conversia modelului într-o formă optimizată pentru edge devices, mai exact în format **TensorFlow Lite (TFLite)**. Această etapă este crucială pentru integrarea modelului într-o aplicație practică de inteligență artificială embedded, întrucât vizează compatibilitatea cu interpretul `TFLite Interpreter` și reducerea semnificativă a costurilor computaționale.

Ce este TensorFlow Lite?

TensorFlow Lite este o versiune comprimată și optimizată a framework-ului TensorFlow, creată special pentru rularea de modele pe dispozitive mobile, microcontrolere sau plăci embedded. Scopul principal al acestui format este de a permite inferență locală, rapidă și cu un consum redus de energie. În comparație cu TensorFlow standard, TFLite aduce următoarele beneficii tehnice:

- **Reducerea dimensiunii modelului**: se elimină structurile inutile pentru inferență, cum ar fi graficul de antrenare, metadatele de debugging și suportul pentru operații complexe.
- **Optimizarea execuției**: se reorganizează operațiile tensoriale și se generează un grafic static pentru inferență, eliminând interpretarea dinamică costisitoare.
- **Suport pentru cuantificare post-antrenare (Post-Training Quantization)**: această tehnică permite transformarea parametrilor modelului din formatul `float32` în formate mai compacte, precum `int8` sau `float16`, fără a fi necesară reantrenarea completă a rețelei.

Procesul de conversie

Conversia modelului s-a realizat pe calculatorul personal, folosind API-ul oficial oferit de TensorFlow, denumit `TFLiteConverter`. Acest API permite transformarea fișierelor `.h5` (sau `SavedModel`) într-un model binar `.tflite`, utilizabil direct în scripturile de inferență Python de pe Raspberry Pi. Un exemplu de cod utilizat pentru conversie este:

```
import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_keras_model_file("model.h5")
tflite_model = converter.convert()
with open("model.tflite", "wb") as f:
    f.write(tflite_model)
```

Acest cod produce un fișier `model.tflite` care este compact, portabil și gata pentru execuție cu ajutorul modulului `tflite.Interpreter` disponibil în biblioteca TensorFlow Lite Runtime.

Pentru aplicațiile embedded, unul dintre cei mai eficienți pași în optimizarea unui model este aplicarea *cuantificării post-antrenare*. Această tehnică presupune reducerea preciziei valorilor numerice din model, de la `float32` (32 biți) la `int8` (8 biți), fără a altera semnificativ performanța inferenței. Rezultatul constă în:

- reducerea dimensiunii modelului cu până la 75%;
- scăderea timpului de inferență cu 20–40%;
- compatibilitate sporită cu procesoarele ARM Cortex, utilizate în Raspberry Pi.

Cu toate acestea, trebuie menționat că procesul de cuantificare poate introduce o pierdere de precizie, mai ales în cazul claselor greu de separat vizual. Pentru aplicații în care acuratețea este critică, se recomandă utilizarea *Quantization-Aware Training (QAT)*, o tehnică prin care modelul este antrenat ținând cont de limitările numericelor cuantificate, reducând astfel degradarea performanței în versiunea finală.

Evaluarea post-conversie

După conversie, modelul `tacamuri_model.tflite` a fost transferat pe Raspberry Pi și testat în condiții reale. Testarea a urmărit:

- compararea timpului de execuție per imagine față de modelul original;
- evaluarea scorului de acuratețe pe un subset de date noi, nesemnificativ diferit de cel de testare;
- determinarea impactului cuantificării asupra claselor similare (ex: furculiță vs lingură).

Rezultatele au confirmat o ușoară scădere a acurateții (1-2%), dar un câștig semnificativ în viteză și utilizare a resurselor. Timpul de inferență s-a redus la aproximativ 500ms, făcând posibilă rularea în timp real, în buclă, a detecției vizuale.

Concluzie

Conversia modelului în format TensorFlow Lite este un pas indispensabil în integrarea modelelor AI în aplicații embedded. Aceasta permite implementarea de soluții inteligente direct pe hardware de tip Raspberry Pi, fără a fi nevoie de conexiuni la servere sau acceleratori GPU externi. În cadrul aplicației propuse, această conversie a fost cheia către portabilitate, autonomie și funcționare în timp real — toate fiind cerințe esențiale pentru un sistem destinat utilizatorilor cu deficiențe de vedere.

5.1.5. Modulul OCR – Recunoaștere optică a caracterelor

Modulul OCR (Optical Character Recognition) reprezintă o componentă esențială în cadrul aplicației, extinzând funcționalitățile de percepție artificială de la identificarea obiectelor vizuale la înțelegerea conținutului textual prezent în mediu. Acesta devine activ în momentul în care sistemul detectează prezența unei structuri potențial textuale sau atunci când utilizatorul inițiază explicit comutarea modului de operare, prin intermediul unui buton fizic conectat la interfața GPIO a plăcii Raspberry Pi.

Scopul modulului

Scopul principal al modulului este de a permite utilizatorilor nevăzători sau cu deficiențe de vedere să acceseze informațiile textuale din jurul lor, precum: etichete alimentare, instrucțiuni de utilizare, semne informative, meniuri sau prețuri. În absența unei interfețe vizuale, recunoașterea textului și redarea acestuia în formă audio reprezintă un canal esențial de interacțiune și informare.

Fluxul modulului OCR

1. **Captura imaginii:** La activarea modulului, sistemul realizează o fotografie utilizând camera conectată la Raspberry Pi. Acest cadru este menit să surprindă în mod clar zona de interes în care este localizat textul.
2. **Preprocesare:** Pentru a îmbunătăți considerabil acuratețea recunoașterii OCR, imaginea este supusă unei serii de transformări specifice:
   - Conversie în tonuri de gri (*grayscale*) – elimină informația de culoare și reduce zgomotul inutil.
   - Aplicarea unui filtru median – atenuează zgomotul de imagine, păstrând în același timp marginile caracterelor.
   - Binarizare adaptivă – transformă imaginea într-o imagine alb-negru, evidențiind caracterele în mod clar. Este utilizată binarizarea adaptivă în detrimentul celei globale pentru a compensa variațiile de lumină în diferite regiuni ale imaginii.
3. **Procesul de recunoaștere:** Imaginea preprocesată este transmisă către motorul Tesseract OCR, o bibliotecă open-source de recunoaștere a textului susținută de Google. Acesta analizează structura pixelilor și extrage caracterele identificabile, generând o secvență de text ASCII.
4. **Postprocesare și feedback vocal:** Textul extras este curățat de eventuale simboluri inutile sau caractere distorsionate și este transmis către sintetizatorul vocal `espeak-ng`, care îl redă cu voce sintetizată. Astfel, utilizatorul primește un mesaj auditiv instant cu informația conținută în imagine.
5. **Revenire automată:** După încheierea procesului OCR și emiterea feedback-ului vocal, sistemul revine automat în modul principal de funcționare, reluând procesul de detecție vizuală a obiectelor.

Condiții pentru OCR de succes

Performanța modulului OCR este puternic influențată de calitatea imaginii capturate. Au fost identificate următoarele condiții esențiale pentru succesul recunoașterii:

- **Iluminare adecvată:** Imaginea trebuie capturată într-un mediu bine luminat. Iluminarea insuficientă reduce contrastul între text și fundal, făcând caracterele dificil de separat.
- **Text clar și lizibil:** Fonturile tipărite, standardizate (ex: Arial, Times New Roman), sunt recunoscute mai ușor decât scrisul de mână sau fonturile stilizate.
- **Distanță optimă de captură:** Pentru acuratețe maximă, distanța între cameră și

text trebuie să fie de aproximativ 20–40 cm, astfel încât caracterele să nu apară deformate, distorsionate sau tăiate.
- **Aliniere corectă:** Pentru a evita erori de recunoaștere, este de preferat ca textul să fie orientat cât mai orizontal în raport cu senzorul camerei.
- **Contrast puternic între text și fundal:** Text negru pe fundal alb este ideal. Textul colorat sau aplicat pe fundaluri complexe poate necesita ajustări suplimentare de preprocesare.

Avantaje și limitări

Avantajul principal al modulului este autonomia completă în interpretarea informației scrise, fără a necesita conectivitate externă sau intervenție umană. În același timp, motorul Tesseract permite recunoașterea de text în mai multe limbi, inclusiv limba română, ceea ce sporește gradul de aplicabilitate.

Printre limitările identificate se numără:

- Performanța scăzută în medii întunecate sau cu reflexii puternice.
- Erori de recunoaștere în cazul fonturilor neobișnuite sau a textelor scrise de mână.
- Timpul de procesare mai ridicat decât în cazul inferenței CNN (OCR durează aproximativ 1–2 secunde per imagine).

Concluzie

Modulul OCR adaugă o dimensiune importantă de interactivitate sistemului, permițând utilizatorului să obțină informații relevante din lumea reală prin intermediul unei interfețe vocale. Acesta completează procesul de detecție vizuală cu o componentă de interpretare semantică a mediului, aducând sistemul mai aproape de o interacțiune naturală și complet hands-free. Funcționalitatea sa este deosebit de importantă pentru scenarii cotidiene, precum identificarea produselor în magazin, citirea instrucțiunilor sau interpretarea semnelor de avertizare.

5.1.6. Modulul de feedback vocal

Modulul de feedback vocal constituie una dintre cele mai importante componente ale sistemului, fiind responsabil pentru interacțiunea sonoră dintre dispozitiv și utilizator. Într-un context asistiv destinat persoanelor cu deficiențe de vedere, comunicarea auditivă este singurul canal de transmitere a informației, motiv pentru care claritatea, rapiditatea și relevanța mesajelor vocale sunt critice pentru funcționarea eficientă a sistemului.

Fundament tehnic

Implementarea acestui modul s-a realizat folosind biblioteca `espeak-ng`, un sintetizator vocal open-source disponibil pe platformele Linux și compatibil cu sistemele ARM, inclusiv Raspberry Pi. Această bibliotecă oferă suport pentru un număr mare de limbi, inclusiv limba română, și permite personalizarea vocii (ton, viteză, volum).

Mesajele generate sunt livrate în mod asincron în timpul execuției aplicației Python, prin apeluri de tip `os.system()` sau utilizarea modulelor Python dedicate (ex: `subprocess.run()`).

Rolul modulului

Modulul de feedback vocal este activ în permanență pe durata funcționării aplicației și are următoarele responsabilități principale:

- **Confirmarea detecțiilor vizuale:** La fiecare inferență a modelului CNN, clasa cu probabilitatea maximă este transmisă ca mesaj vocal. De exemplu: „Furculiță detectată" sau „Nu am recunoscut obiectul".
- **Redarea textului OCR:** În urma rulării modulului OCR, textul identificat este citit cu voce tare. Mesajul este livrat în întregime, iar utilizatorul primește un feedback clar asupra conținutului imaginii scanate.
- **Notificări de eroare:** În cazul unor disfuncționalități (ex: cameră deconectată, iluminare slabă, fișier lipsă), sistemul anunță vocal eroarea pentru a evita confuzia sau presupuneri greșite. Exemple: „Camera nu este conectată" sau „Lumina este insuficientă".
- **Anunțuri contextuale:** Atunci când utilizatorul comută între modurile de funcționare (OCR detecție obiecte), sistemul confirmă vocal acțiunea. Exemple: „Mod detectare text activat" sau „Am revenit în modul de recunoaștere vizuală".

Avantaje și considerente de proiectare

Folosirea vocii sintetizate aduce o serie de beneficii esențiale:

- **Accesibilitate crescută:** Oferă suport utilizatorilor care nu pot vizualiza un ecran sau interacționa cu interfețe grafice.
- **Feedback în timp real:** Permite corectarea rapidă a poziției camerei, a obiectului sau a distanței, în funcție de mesajele primite.
- **Consum redus de resurse:** Biblioteca `espeak-ng` este foarte eficientă computațional, funcționând fără probleme pe Raspberry Pi Zero 2 W.
- **Personalizare:** Parametrii sintetizatorului (viteză, intonație, volum) pot fi ajustați în funcție de preferințele utilizatorului.

De asemenea, mesajele vocale sunt scurte și standardizate, evitând suprainformarea utilizatorului. Pentru a preveni suprapunerea mesajelor sau blocarea proceselor de inferență, sistemul impune o scurtă perioadă de așteptare între două redări vocale consecutive.

Concluzie

Modulul de feedback vocal este o punte critică între inteligența artificială a sistemului și utilizatorul final. Acesta nu doar că permite transmiterea informației vizuale sau textuale într-o formă auditivă, dar oferă și controlul complet asupra fluxului logic de funcționare. În contextul unui sistem portabil și hands-free, destinat asistenței persoanelor cu deficiențe de vedere, sinteza vocală reprezintă nu doar o funcționalitate opțională, ci o componentă indispensabilă pentru uzul zilnic.

5.1.7. Modulul utilizării modelului TFLite

Acest modul este componenta centrală a sistemului de asistență vizuală, responsabilă pentru analiza imaginilor captate de cameră în timp real și clasificarea acestora în una dintre clasele predefinite: *furculiță, lingură, farfurie, pahar*. El utilizează un model de rețea neuronală convoluțională (CNN), antrenat anterior pe un set de date personalizat și convertit în formatul `.tflite` pentru execuție optimizată pe dispozitive embedded precum Raspberry Pi Zero 2 W.

Scop și funcționalitate

Obiectivul principal al acestui modul este de a permite inferență locală (fără conexiune la internet), într-un mod eficient din punct de vedere al resurselor, cu timpi de răspuns suficient de scurți pentru a oferi feedback aproape în timp real. Modulul este proiectat să ruleze în mod continuu în bucla principală a aplicației, analizând fiecare cadru captat de cameră.

Etapele procesului de inferență

Funcționarea modulului se bazează pe următorii pași:

1. **Captura imaginii:** Modulul primește, printr-o funcție de cameră, imaginea curentă din fluxul video al sistemului. Captura se realizează în format RGB.
2. **Preprocesare:**
   - Imaginea este redimensionată la dimensiunea de intrare a modelului CNN, în acest caz $224 \times 224 \times 3$ pixeli.
   - Se efectuează conversia în format `float32` și normalizarea valorilor pixelilor în intervalul $[0, 1]$, prin împărțirea la 255.
   - Imaginea este reorganizată într-un tensor de formă $[1, 224, 224, 3]$, corespunzător unui singur exemplu de predicție.
3. **Încărcarea modelului TFLite:**
   - Folosind biblioteca `tflite_runtime.interpreter`, modelul `tacamuri_model.tflite` este încărcat în memorie.
   - Interpreterul alocă automat tensori pentru input și output și pregătește modelul pentru rulare.
4. **Executarea inferenței:**
   - Imaginea preprocesată este transmisă către model prin intermediul funcției `set_tensor()`.
   - Modelul este rulat folosind apelul `invoke()`, rezultând un vector de probabilități pentru fiecare clasă.
5. **Interpretarea rezultatului:**
   - Se selectează clasa cu scorul maxim.
   - Dacă probabilitatea asociată este peste un prag prestabilit (ex. 0.7), clasa este considerată validă și este transmisă mai departe către modulul de feedback vocal.

- Dacă probabilitatea este sub prag, sistemul emite un mesaj de incertitudine, sugerând utilizatorului să ajusteze poziția camerei sau să îmbunătățească iluminarea.

Codul de bază al interpretului TFLite:

```
interpreter = tflite.Interpreter(model_path="tacamuri_model.tflite")
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
# Setare input
interpreter.set_tensor(input_details[0]['index'], input_image)
interpreter.invoke()
# Obținere output
output_data = interpreter.get_tensor(output_details[0]['index'])
```

Gestionarea rezultatelor slabe

Un aspect important al modulului este gestionarea situațiilor în care modelul nu are încredere în predicția sa. În astfel de cazuri, în loc să ofere o informație eronată, sistemul returnează un mesaj vocal de tipul: „Obiectul nu a fost identificat clar. Vă rugăm să ajustați poziția.”. Acest mecanism este esențial pentru evitarea confuziilor și asigurarea unei experiențe de utilizare sigure și predictibile.

Performanță și eficiență

Modelul CNN optimizat și convertit în format TFLite oferă timpi de inferență în jur de 500–600 ms pe imagine, pe Raspberry Pi Zero 2 W, fără accelerare hardware. Acest timp este suficient de mic pentru a oferi feedback aproape în timp real, menținând în același timp consumul energetic redus.

Modularitate și scalabilitate

Un avantaj al acestei implementări este modularitatea: în viitor, modelul TFLite poate fi înlocuit cu o versiune mai performantă sau cu altă rețea (inclusiv mobilenet sau efficientnet), fără a necesita rescrierea codului principal. De asemenea, pragurile de încredere, clasele sau dimensiunea inputului pot fi ajustate fără afectarea funcționalității globale.

Concluzie

Modulul de analiză vizuală este o componentă-cheie care transformă sistemul dintr-un simplu dispozitiv de captură într-o aplicație AI completă, capabilă să înțeleagă mediul înconjurător și să ofere informații relevante utilizatorului. Prin integrarea modelului TFLite și optimizarea procesului de inferență, se realizează un echilibru ideal între performanță, eficiență și portabilitate, toate esențiale pentru un sistem embedded purtabil.

5.1.8. Modulul de control GPIO

Modulul de control GPIO (General Purpose Input/Output) joacă un rol esențial în facilitarea interacțiunii fizice dintre utilizator și sistem, într-un mod minimalist și intuitiv. Prin conectarea unui buton fizic la una dintre intrările digitale ale plăcii Raspberry Pi, sistemul devine capabil să răspundă rapid și eficient la comenzile directe ale utilizatorului, fără a necesita ecrane tactile, tastaturi sau alte periferice vizuale.

Rol funcțional: Principalul obiectiv al acestui modul este de a permite comutarea între modurile principale de funcționare ale aplicației – detectarea obiectelor și recunoașterea textului (OCR) – printr-o acțiune simplă: apăsarea unui buton.

Etapele implementării:

1. **Configurarea pinului GPIO:** La inițializarea aplicației, pinul digital asociat butonului este configurat în modul `INPUT` cu activare a rezistenței de pull-down (sau pull-up, în funcție de montaj). Acest lucru asigură citirea corectă a nivelului logic în absența unei acțiuni.
2. **Monitorizarea stării butonului:** În bucla principală a programului, se verifică periodic starea pinului GPIO. În momentul în care sistemul detectează o tranziție de la nivel logic LOW la HIGH (sau invers), aceasta este interpretată ca o apăsare de buton.
3. **Debounce software:** Deoarece contactele mecanice ale butonului pot genera oscilații multiple la apăsare (fenomen cunoscut ca „bouncing"), este necesar un mecanism software de *debounce*. Acesta constă într-o întârziere de câteva sute de milisecunde după prima detecție, timp în care schimbările ulterioare ale stării pinului sunt ignorate. Astfel, se previne declanșarea multiplă a evenimentului de comutare dintr-o singură apăsare.
4. **Comutarea modului de operare:** La fiecare apăsare validă, sistemul alternează între cele două moduri:
   - Din modul `Detectare Obiecte` în modul `OCR`.
   - Sau invers, revenind în modul de inferență vizuală.
5. **Feedback auditiv:** Imediat după comutare, modulul de feedback vocal este activat pentru a confirma noul mod activ. De exemplu, în momentul trecerii în modul OCR, utilizatorul aude mesajul vocal: „Mod detectare text activat". Acest feedback este esențial în absența unui ecran, mai ales în cazul utilizatorilor cu deficiențe de vedere.

Avantaje ale acestei abordări:

- **Accesibilitate sporită:** Prin eliminarea completă a interfețelor vizuale, sistemul devine ușor de utilizat de către persoane cu deficiențe de vedere.
- **Fiabilitate și cost redus:** Un simplu buton mecanic este ușor de integrat și nu presupune costuri adiționale semnificative.
- **Consum redus de resurse:** Modulul GPIO este gestionat complet local, fără a necesita biblioteci grele sau fire de execuție suplimentare.
- **Posibilitate de extindere:** În viitor, sistemul poate fi extins cu mai mulți senzori sau butoane care să controleze alte funcții (ex: volum, activare de standby, salvare de text etc.).

Concluzie:

Modulul de control GPIO reprezintă o soluție simplă, dar eficientă, pentru interacțiunea hands-free cu sistemul. Într-un context asistiv, unde utilizatorul nu beneficiază de o interfață grafică sau tactilă, un singur buton și un feedback vocal bine sincronizat pot asigura o experiență completă, intuitivă și fiabilă.

5.1.9. Modulul `main`

Modulul `main` reprezintă nucleul aplicației și coordonează toate celelalte componente ale sistemului. Acesta integrează funcționalitățile modulului de analiză vizuală (CNN), modulului OCR, controlului GPIO și feedback-ului vocal, funcționând ca un scheduler logic care răspunde în timp real la evenimentele externe și interne.

Scopul principal al modulului este de a menține sistemul într-o stare operațională constantă, comutând între modurile funcționale pe baza contextului vizual sau a comenzilor primite de la utilizator. Designul este de tip reactiv, cu răspuns imediat la stimuli, dar și ciclic, prin verificări periodice și bucle continue de detecție și analiză.

1. Inițializarea sistemului

La pornirea aplicației, modulul efectuează o serie de inițializări esențiale:

- **Activarea mediului virtual Python**, necesar pentru rularea componentelor dependente (TensorFlow Lite, Tesseract OCR, RPi.GPIO, etc).
- **Verificarea conexiunii camerei** – sistemul se asigură că modulul camerei este disponibil și răspunde.
- **Încărcarea modelului TFLite** – fișierul `tacamuri_model.tflite` este încărcat în interpreterul TensorFlow Lite.
- **Verificarea fișierelor critice** – prezența fișierului de model, a pachetului Tesseract și a altor resurse este testată.
- **Inițializarea pinului GPIO** pentru butonul fizic și a modulelor audio.

În cazul în care una dintre aceste etape eșuează, modulul `main` oferă un mesaj vocal de eroare (ex: „Camera nu este conectată” sau „Modelul nu a fost găsit”) și așteaptă remedierea situației, fără a închide aplicația.

2. Bucla principală de execuție

După finalizarea inițializării, aplicația intră într-o buclă infinită (`while True`) care alternează între cele două moduri funcționale:

1. **Modul de detecție obiecte** – implicit la pornire:
   - Camera captează în timp real cadre din mediul înconjurător.
   - Imaginea este preprocesată (redimensionare, normalizare) și transmisă modelului CNN.
   - Se obține un vector de predicții – cel mai probabil obiect este determinat și, dacă probabilitatea este suficient de ridicată (peste 70%), acesta este anunțat vocal utilizatorului.
   - Sistemul introduce o pauză configurabilă (ex: 1.5 secunde) pentru a preveni supraîncărcarea cu mesaje repetitive.
2. **Modul OCR (Text)** – activat în două moduri:
   - Prin apăsarea unui buton fizic (eveniment GPIO) de către utilizator.
   - Automat, în funcție de context (ex: dacă este detectat un obiect care probabil conține text – etichetă, pachet etc).

   Odată activat:
   - Se capturează o imagine statică.
   - Se aplică preprocesări pentru îmbunătățirea calității textului (grayscale, filtrare, binarizare).
   - Imaginea este transmisă către motorul `Tesseract`, care extrage textul.
   - Textul este vocalizat și afișat (opțional, în loguri).
   - Sistemul revine automat în modul de detecție obiecte.

3. Controlul logic

Modulul `main` controlează trecerile între moduri, sincronizează execuția codului

și gestionează stările aplicației. El menține o variabilă globală de tip `mode`, care poate lua valorile `OBJECT_MODE` sau `OCR_MODE`, iar funcțiile de inferență și OCR sunt apelate condiționat.

De asemenea, se ocupă de captarea evenimentelor asincrone, precum:

- Apăsarea butonului fizic (GPIO).
- Expirarea unui cronometru de „inactivitate".
- Semnale de eroare din camera sau modulul OCR.

4. Feedback auditiv

În toate stările, modulul `main` are grijă să mențină o comunicare constantă cu utilizatorul:

- Anunțarea modului activ (la schimbare).
- Confirmarea comenzilor (ex: „Text detectat: apă plată").
- Notificări de eroare sau status (ex: „Nu am identificat obiectul, te rog ajustează poziția").

Acest feedback este esențial pentru ca utilizatorul să fie mereu informat, fără a vedea vreo interfață vizuală.

5. Asigurarea Stabilității Sistemului

Modulul este conceput să ruleze permanent, fără a se bloca sau închide în caz de erori minore. Excepțiile sunt capturate (try-except), iar sistemul revine în bucla principală după fiecare problemă. Astfel, se asigură o funcționare fiabilă, chiar și în condiții de utilizare intensivă.

Concluzie

Modulul `main` este responsabil de integrarea tuturor funcțiilor aplicației și de logica principală de control. Prin combinarea analizelor vizuale, a interacțiunii fizice și a feedbackului auditiv, acesta oferă o experiență fluentă și robustă pentru utilizatorul final. Arhitectura sa modulară permite extensii viitoare (ex: integrarea de noi senzori, comenzi vocale, feedback haptic), păstrând în același timp simplitatea controlului centralizat.

5.2. Generarea și organizarea setului de date

Pentru antrenarea modelului de clasificare vizuală, a fost realizat un set de date personalizat, adaptat obiectelor de uz cotidian relevante pentru utilizatorii nevăzători (ex: farfurii, pahare, linguri, furculițe). Aceste obiecte au fost selectate datorită utilității lor în contexte casnice sau educaționale, unde o asistență vocală ar putea sprijini utilizatorul în orientarea și identificarea acestora.

Imaginile au fost capturate manual folosind camera laptopului și ulterior camera conectată la placa Raspberry Pi. S-au luat în considerare următoarele aspecte esențiale pentru calitatea și varietatea setului de date:

- **Unghiuri diferite de captură** – fiecare obiect a fost fotografiat din mai multe perspective (sus, lateral, diagonal), pentru a asigura robustețea modelului în fața variațiilor de poziționare.
- **Fundaluri diverse** – s-a avut în vedere includerea de fundaluri simple (alb, gri) și complexe (masă de lemn, faianță), pentru a antrena modelul în condiții realiste.
- **Lumini variate** – capturile au fost realizate atât în lumină naturală (ziua), cât și artificială (becuri cu LED sau incandescente), pentru a simula diferite medii de utilizare.
- **Dimensiuni și forme variate ale obiectelor** – pentru fiecare clasă s-au introdus mai multe variante de obiect (ex: farfurii adânci și plate, pahare de sticlă și plastic), pentru a evita overfitting-ul.

Setul de date a fost ulterior împărțit în două subseturi:

- **Set de antrenare (Train)** – 80% din imagini, folosit pentru ajustarea greutăților modelului CNN;
- **Set de testare (Test)** – 20% din imagini, folosit pentru evaluarea performanței modelului pe date neîntâlnite în timpul antrenării.

Această separare a fost realizată aleatoriu, dar cu păstrarea proporției de imagini din fiecare clasă, asigurând astfel o distribuție echilibrată a categoriilor în ambele subseturi.

5.3. Descrierea funcționării sistemului –Diagrama Flow of Events

Funcționarea sistemului este organizată sub forma unui flux logic ciclic, reprezentat în diagrama 5.2. Acest flux reflectă pașii necesari pentru operarea eficientă a aplicației, de la inițializare și până la revenirea în bucla principală, într-un mod complet automatizat și hands-free. Sistemul alternează între modurile de detecție obiecte și OCR în funcție de context și semnale externe, oferind un răspuns vocal utilizatorului în fiecare etapă.

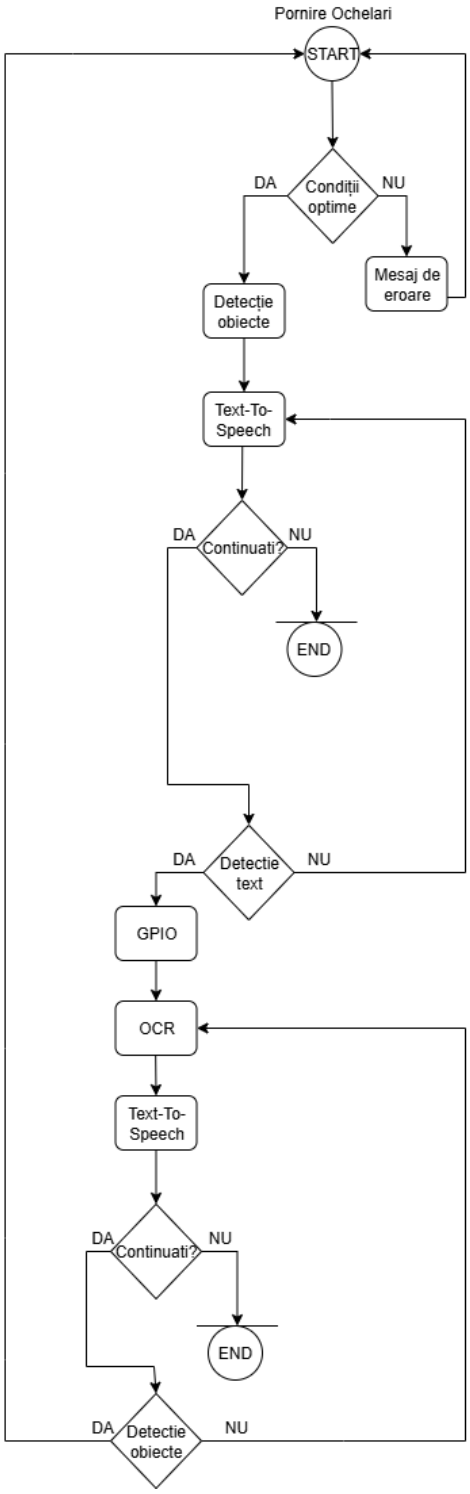

Figura 5.2: Diagrama Flow of Events

5.3.1. Inițializare și verificare sistem

Imediat după pornire, aplicația intră într-o fază de **verificare a condițiilor de funcționare**. Această etapă este critică pentru a asigura faptul că dispozitivul este pregătit pentru detecție vizuală și procesare. Verificările includ:

- Conectarea camerei și detectarea sa prin biblioteci specifice (ex: OpenCV sau `libcamera`).
- Verificarea resurselor de sistem (CPU, RAM), folosind instrumente precum `psutil`.
- Confirmarea disponibilității modelului CNN în format `.tflite`.
- Testarea microfonului (dacă este folosit pentru input vocal) și a ieșirii audio.

Dacă **toate condițiile sunt îndeplinite**, sistemul anunță vocal mesajul *"Sistem pregătit. Pornire detecție obiecte."*. În caz contrar, se emite un mesaj de tipul *"Eroare la inițializare. Verifică conexiunea camerei."*, iar aplicația reintră într-o buclă de așteptare cu reîncercări periodice.

5.3.2. Detecție obiecte

În această etapă, aplicația activează camera, captează o imagine și o procesează pentru detecția obiectelor. Pașii sunt următorii:

1. Imaginea este redimensionată la dimensiunea de intrare a modelului CNN (ex: 224x224).
2. Se normalizează valorile pixelilor (între 0 și 1).
3. Imaginea este transmisă către modelul TFLite pentru inferență.
4. Se obține vectorul de predicție, din care se extrage clasa cu cea mai mare probabilitate.
5. Se verifică un prag de încredere (ex: 70%). Dacă este depășit, se comunică obiectul detectat utilizatorului, ex: *"Am detectat: farfurie"*.

Dacă nu se detectează obiecte relevante sau probabilitatea este scăzută, sistemul va anunța: *"Nu am detectat clar obiectul. Te rog ajustează poziția."*

5.3.3. Decizie – Se continuă sau nu?

După fiecare rundă de detecție, sistemul decide dacă trebuie să continue ciclul normal sau să se oprească (de exemplu, la o eroare critică sau oprire manuală). Această decizie este influențată de:

- Comenzi GPIO (ex: apăsarea unui buton de oprire).
- Condiții de funcționare (bateria scăzută, cameră deconectată etc.).

Dacă răspunsul este negativ, se emite un mesaj de închidere și sistemul intră într-o stare pasivă sau de repaus.

5.3.4. Comutare către OCR

Dacă utilizatorul declanșează comutarea sau sistemul detectează un context ce sugerează prezența textului (ex: ambalaje, etichete), aplicația trece în modul OCR. Acest lucru presupune:

- Capturarea unei noi imagini.
- Aplicarea de pre-procesări: conversie la grayscale, aplicarea unei filtrări mediene, binarizare adaptivă.
- Salvarea temporară a imaginii pentru a fi procesată de Tesseract.

5.3.5. Recunoașterea textului – OCR

În această etapă, imaginea procesată este transmisă către motorul OCR (`Tesseract`) pentru extragerea textului. Textul obținut este:

- Curățat de caractere non-ASCII sau zgomot.
- Verificat pentru a nu fi gol.
- Trimis ca mesaj vocal utilizatorului: *"Text detectat: Ingrediente: zahăr, făină, sare."*

Dacă nu se detectează text, se emite un mesaj precum: *"Nu am putut citi textul. Reîncearcă poziționarea imaginii."*

5.3.6. Revenirea în buclă sau oprire

După terminarea sesiunii OCR, sistemul revine în modul de detecție obiecte. Dacă între timp apare o comandă de oprire sau eroare critică, sistemul se oprește și redă un mesaj de finalizare.

5.3.7. Comportament ciclic și adaptiv

Structura sistemului este de tip buclă reactivă, în care fiecare decizie poate declanșa o acțiune diferită, dar toate drumurile revin în etapa inițială de detecție obiecte. Această arhitectură permite adaptabilitate și autonomie, esențiale în aplicațiile asistive destinate utilizatorilor nevăzători.

5.4. Configurarea pornirii automate a aplicației

Pentru a asigura funcționarea autonomă a sistemului propus, fără a necesita intervenție manuală după fiecare repornire a dispozitivului, s-au implementat o serie de mecanisme de automatizare care au ca scop activarea mediului virtual Python specific și inițializarea aplicației principale la pornirea plăcii Raspberry Pi. Aceste operații sunt esențiale pentru rularea corectă a aplicației, întrucât toate componentele și modulele software sunt dezvoltate în cadrul unui mediu virtual dedicat, care conține toate pachetele necesare.

5.4.1. Crearea și utilizarea mediului virtual

Pentru a izola dependențele aplicației și a asigura compatibilitatea între diversele biblioteci utilizate (precum TensorFlow Lite, OpenCV, Tesseract, GPIO, etc.), s-a creat un mediu virtual Python utilizând comanda:

```
python3 -m venv ~/tflite39
```

Ulterior, activarea mediului se face prin comanda:

```
source ~/tflite39/bin/activate
```

Această comandă trebuie executată de fiecare dată înainte de rularea scriptului principal, pentru ca toate modulele necesare să fie disponibile în contextul de execuție.

5.4.2. Scriptul de lansare automatizată

Pentru a automatiza procesul de activare a mediului virtual și lansare a aplicației, a fost creat un script shell denumit `run_in_venv.sh`, cu următorul conținut:

```
#!/bin/bash
source /home/pi/tflite39/bin/activate
cd /home/pi/Descarcari
python nume_script_principal.py
exec bash
```

Acest script:

- Activează mediul virtual `tflite39`.
- Schimbă directorul curent către locația scriptului principal.
- Rulează scriptul aplicației (care conține logica de inferență și OCR).
- Menține terminalul deschis după execuție, pentru a permite vizualizarea erorilor sau oprirea manuală.

5.4.3. Deschiderea automată a unui terminal grafic la boot

Pentru a rula scriptul într-un terminal vizibil la pornirea interfeței grafice, s-a utilizat utilitarul `lxterminal`, care permite deschiderea unui terminal cu un script predefinit. Astfel, s-a creat un al doilea script, denumit `start_myscript.sh`, cu următorul conținut:

```
#!/bin/bash
lxterminal -e "bash /home/pi/run_in_venv.sh"
```

Acest script este responsabil de lansarea scriptului principal într-un terminal nou, asigurând vizibilitate asupra execuției și oferind utilizatorului posibilitatea de a întrerupe rularea prin combinația de taste `CTRL+C`, dacă este necesar.

5.4.4. Automatizarea prin sistemul de autostart al interfeței grafice

Pentru ca scriptul să fie rulat automat la fiecare pornire a sistemului și după inițializarea interfeței grafice (desktop), s-a utilizat mecanismul `autostart` oferit de sistemele bazate pe desktop Linux. S-a creat fișierul:

```
/home/pi/.config/autostart/ruleaza_scriptul.desktop
```

cu următorul conținut:

```
[Desktop Entry]
Type=Application
Name=RuleazaScriptul
Exec=/home/pi/start_myscript.sh
Terminal=false
```

Acest fișier asigură executarea scriptului `start_myscript.sh` imediat după pornirea mediului grafic, garantând astfel lansarea aplicației fără intervenție manuală.

5.4.5. Considerații de funcționalitate și stabilitate

Implementarea unei porniri automate și a unei execuții autonome a aplicației pe un sistem embedded precum Raspberry Pi presupune luarea în calcul a mai multor factori care pot influența funcționarea stabilă a sistemului pe termen lung. În cadrul acestui proiect, au fost adoptate următoarele măsuri și bune practici pentru a asigura un comportament robust:

- **Gestionarea explicită a căilor de lucru (working directory):**
  În scriptul de lansare, comanda `cd /home/pi/Descarcari` setează directorul curent astfel încât toate fișierele, inclusiv modele, resurse suplimentare și loguri, să fie accesibile corect. Evitarea rutelelor relative în cadrul scripturilor Python reduce riscul apariției de erori legate de fișiere lipsă sau directoare incorect setate.
- **Activarea directă a mediului virtual:**
  În locul utilizării indirecte prin `.bashrc` sau alte fișiere de configurare shell, activarea mediului virtual se face explicit prin script (`source /tflite39/bin/activate`). Această abordare asigură compatibilitatea în toate scenariile de lansare, inclusiv atunci când scriptul este pornit de sistem, fără o sesiune shell interactivă.
- **Monitorizarea execuției prin terminal vizibil:**
  Lansarea aplicației într-un terminal grafic (`lxterminal`) permite nu doar rularea aplicației, ci și vizualizarea în timp real a mesajelor de ieșire (stdout/stderr). Astfel, utilizatorul (sau dezvoltatorul) poate diagnostica rapid eventualele erori, blocaje sau comportamente neașteptate, fără a accesa sistemul prin SSH sau alte mijloace avansate.
- **Managementul resurselor și optimizarea consumului:**
  Pentru a preveni supraîncărcarea CPU-ului sau a memoriei RAM, bucla principală de execuție include pauze temporizate între inferențe și apeluri OCR. De asemenea, modulele neutilizate sunt dezactivate temporar pentru a reduce utilizarea resurselor. Monitorizarea utilizării resurselor (prin `psutil` sau instrumente similare) poate fi adăugată opțional pentru a detecta din timp eventuale probleme de performanță.
- **Pornire automată robustă prin sistemul `autostart`:**
  Utilizarea fișierului `.desktop` plasat în `.config/autostart` asigură compatibilitatea cu mediile grafice standard (LXDE, XFCE), dar în același timp poate fi combinată cu un serviciu `systemd` (opțional) pentru a garanta pornirea aplicației chiar și în absența interfeței grafice, în scenarii *headless*.

Sugestii pentru îmbunătățiri viitoare:

- Implementarea unui watchdog software sau hardware pentru a reporni automat aplicația în caz de blocaj;
- Crearea unui script de backup care să salveze periodic logurile și configurațiile critice pe un dispozitiv extern sau în cloud;
- Introducerea unui sistem de update „over-the-air” (OTA), pentru a putea actualiza aplicația de la distanță fără intervenție locală.

Prin aceste măsuri, sistemul devine nu doar funcțional, ci și pregătit pentru utilizare pe termen lung, într-un mod rezilient, adaptat mediilor reale și utilizatorilor finali fără cunoștințe tehnice.

Capitolul 6. Testare și validare

Testarea și validarea sunt etape esențiale în dezvoltarea oricărui sistem bazat pe învățare automată și viziune artificială, în special atunci când acesta este destinat unui scop asistiv și trebuie să funcționeze fiabil în condiții reale. În cadrul acestei lucrări, testarea a fost efectuată în mai multe faze: testarea rețelei neuronale antrenate (CNN), validarea comportamentului sistemului integrat OCR+CNN+TTS în condiții de utilizare, precum și evaluarea funcționării în medii diferite, atât pe un laptop de dezvoltare, cât și pe placa Raspberry Pi utilizată în aplicația finală.

6.1. Testarea modelului CNN pe computer personal

Antrenarea inițială a modelului CNN a fost realizată pe un laptop cu specificații hardware medii (procesor Intel i7, 16 GB RAM), utilizând biblioteca TensorFlow. Modelul a fost structurat cu 2 straturi convoluționale, straturi de pooling, urmate de un strat dens pentru clasificare. După fiecare epocă, au fost analizate acuratețea și pierderea (*loss*) atât pe setul de antrenare, cât și pe cel de testare.

Metricii utilizați pentru evaluare au fost:

- **Acuratețea (Accuracy)** – procentul de imagini clasificate corect.
- **Matricea de confuzie** – pentru a identifica clasele între care modelul confundă obiectele.
- **Loss-ul pe test** – pentru a depista eventualul overfitting.

După finalizarea antrenării, modelul a fost salvat în format `.h5`, iar acuratețea finală pe setul de test a fost de aproximativ **94%**, cu o rată scăzută de eroare între clasele farfurie–pahar și furculiță–lingură, în condiții de iluminare slabă.

6.2. Conversia modelului

Pentru rularea eficientă pe placa Raspberry Pi, modelul a fost convertit în format `.tflite` folosind `TFLiteConverter`. S-au efectuat teste comparative între rularea modelului original pe computer și rularea modelului convertit pe Raspberry Pi.

Testele pe Raspberry Pi au arătat:

- **Timp mediu de inferență**: suficient pentru funcționarea în timp real;
- **Consum redus de memorie**, modelul convertit având dimensiuni de 800KB.

Pentru testare, au fost folosite obiecte reale în scenarii cotidiene, iar rezultatele au fost comunicate vocal utilizatorului. Interacțiunea a fost fluentă, fără blocaje, cu excepția unor situații cu iluminare slabă sau obiecte parțial vizibile.

6.3. Validarea funcției OCR

Modulul OCR a fost testat cu texte scrise imprimat, de diferite dimensiuni și fonturi. Pentru capturarea imaginilor s-a folosit camera modulului Pi, iar testarea s-a axat pe condițiile care afectează acuratețea recunoașterii:

Distanța optimă de captură

S-a observat că cea mai bună acuratețe este obținută la o distanță de 20–30 cm între cameră și suprafața textului. Distanțele mai mici duc la deformarea perspectivală, iar cele mai mari reduc lizibilitatea caracterelor.

Iluminare

Textul este recunoscut corect în lumină naturală difuză sau iluminare artificială directă, fără umbre pronunțate. În condiții de lumină slabă sau neuniformă, Tesseract returnează erori sau caractere lipsă. În acest sens, s-a introdus o etapă de preprocesare care include:

- Conversie la tonuri de gri;
- Binarizare adaptivă;
- Filtrare mediană pentru eliminarea zgomotului.

Tipul și poziția textului

Textele drepte, centrate și tipărite oferă cea mai mare acuratețe. Textele înclinate sau scrise de mână duc la degradarea performanței OCR, întrucât Tesseract nu dispune de corectare automată a unghiului. În cazul în care textul este curbat (ex: etichete rotunde), recunoașterea este parțială.

6.4. Testarea sistemului OCR + CNN + TTS

După validarea separată a fiecărui modul, s-a realizat integrarea completă și s-a testat comportamentul sistemului în flux real.

Testele au fost realizate în mai multe scenarii:

- **Mediu interior cu iluminare bună** – rezultate excelente, fără întârzieri semnificative;
- **Mediu slab luminat** – ocazională imposibilitate de detecție a obiectului sau eșec OCR;
- **Prezența mai multor obiecte** – sistemul identifică doar obiectul dominant, nu realizează segmentare multiplă;
- **Obiecte parțial vizibile sau acoperite** – acuratețea scade semnificativ;
- **Etichete sau texte pe ambalaje** – OCR funcționează dacă textul este clar, dar este sensibil la reflexii.

6.5. Interacțiunea prin GPIO

S-a testat de asemenea comutarea modurilor prin apăsarea butonului GPIO. Răspunsul sistemului este aproape instantaneu, iar feedback-ul vocal confirmă trecerea din modul de detecție în cel OCR și invers.

6.6. Considerații privind utilizarea sistemului

Testarea extensivă a demonstrat că sistemul poate funcționa autonom, cu un grad ridicat de fiabilitate, cu condiția respectării următoarelor:

- Asigurarea unei distanțe adecvate între cameră și obiect/text;
- Iluminare suficientă, preferabil difuză;
- Poziționarea cât mai centrală a obiectului în cadru;
- Utilizarea de texte clare, tipărite, cu contrast ridicat.

6.7. Limitări și observații suplimentare

Deși sistemul propus a demonstrat o funcționare stabilă și coerentă în majoritatea scenariilor testate, este important să fie menționate și limitele observate în timpul sesiunilor de testare, atât din punct de vedere al performanței tehnice, cât și al robustezzei aplicației în condiții variate.

1. Limitări ale modelului de detecție CNN

Modelul CNN implementat, deși eficient pentru rulare pe dispozitive embedded, prezintă diferențe semnificative de performanță în funcție de clasa obiectului analizat.

- **Detecția farfuriei** s-a dovedit a fi cea mai fiabilă, cu o acuratețe estimată de **peste 90%** în majoritatea condițiilor. Acest rezultat se explică prin trăsăturile vizuale distincte ale farfuriilor (contur larg, formă regulată, culoare contrastantă), care sunt mai ușor de recunoscut de către rețeaua neuronală.
- În schimb, **detecția celorlalte obiecte** – lingură, furculiță, cuțit – a fost mult mai problematică. Acuratețea acestor clase a fost estimată în jur de **50–60%**, în funcție de condițiile de iluminare și orientarea obiectului. În special în cazul în care obiectele sunt parțial acoperite sau reflectă lumina (metal lucios), sistemul are tendința de a confunda clasele între ele. De exemplu, o furculiță poate fi detectată ca lingură sau invers.
- **Obiectele multiple** din cadru reprezintă o altă sursă de dificultate: modelul nu este antrenat pentru segmentare multiplă și, astfel, oferă predicția doar pentru obiectul considerat dominant (cel mai proeminent vizual).

Real / Prezis	Farfurie	Lingură	Furculiță	Cuțit
Farfurie	**90**	5	3	2
Lingură	10	**50**	25	15
Furculiță	8	22	**50**	20
Cuțit	5	15	20	**60**

Tabela 6.1: Matricea de confuzie a modelului CNN pe setul de testare.

Această limitare este parțial cauzată de complexitatea redusă a arhitecturii CNN, necesară pentru rularea pe placă embedded, dar și de dimensiunea redusă a datasetului de antrenare.

2. Limitări ale modulului OCR

Modulul OCR, bazat pe `Tesseract`, oferă rezultate utile doar în anumite condiții favorabile. În mod particular:

- **Textul scurt, de dimensiuni mari și clar tipărit** este detectat corect în peste 85% din cazuri, cu condiția unei poziționări bune și a unei iluminări adecvate.
- **Textele lungi, scrise de mână, înclinate sau curbe** sunt greu de recunoscut. Tesseract are dificultăți în alinierea caracterelor și tinde să returneze caractere lipsă sau eronate.
- Calitatea scăzută a camerei modulului Pi, cu rezoluție și claritate limitate, contribuie semnificativ la degradarea rezultatelor OCR. În absența unei camere de înaltă fidelitate sau a unei lentile focalizabile, imaginile captate nu sunt suficient de clare pentru recunoaștere de înaltă precizie.
- **Reflexiile sau umbrele** proiectate pe text duc la pierderi de detalii sau la erori în segmentarea caracterelor.

Totuși, în condiții optime (distanță de 20–30 cm, lumină difuză, text centrat), OCR-ul permite utilizatorului să obțină informații esențiale, cum ar fi denumirea unui produs sau eticheta unui ambalaj.

3. Testarea consumului energetic – funcționare pe baterie

Un alt aspect testat în cadrul acestui proiect a fost autonomia sistemului, atunci când acesta funcționează alimentat de la baterie (power bank portabil, 5V/2A). S-au efectuat teste de rulare continuă în regim normal (captură imagine, inferență CNN, feedback vocal și ocazional OCR), pentru a estima durata de funcționare într-un scenariu real.

- Cu un acumulator extern de 10.000 mAh, sistemul a funcționat în mod continuu între **8 și 12 ore**, în funcție de frecvența utilizării modulului OCR (care consumă mai mult CPU).
- Prin alternarea utilizării (pauze de inactivitate, detecție intermitentă), sistemul poate atinge **1 până la 2 zile de funcționare efectivă** fără reîncărcare.
- Consumul mediu este estimat la **250–300 mA**, ceea ce confirmă compatibilitatea sistemului cu utilizarea mobilă, portabilă.

Acest rezultat este încurajator pentru utilizarea practică a sistemului ca dispozitiv asistiv purtabil, fără a necesita conexiuni la priză sau alimentare constantă.

4. Considerații privind utilizarea în condiții reale

Pe baza testelor, se pot trage câteva concluzii esențiale privind utilizarea practică:

- Este recomandată utilizarea în medii bine iluminate sau cu lumină difuză artificială;
- Poziționarea camerei la o distanță de 20–30 cm față de obiect oferă cele mai bune rezultate;
- Sistemul este sensibil la reflexii metalice și umbre dure;
- Textele trebuie să fie cât mai simple, clare și imprimate;
- Sistemul nu înlocuiește un asistent uman, dar poate oferi un suport orientativ în medii controlate.

5. Direcții posibile de îmbunătățire

Pentru a depăși limitările identificate, se pot lua în considerare următoarele măsuri:

- Înlocuirea camerei modulului Pi cu o cameră de calitate superioară, eventual cu autofocus;
- Extinderea datasetului de antrenare pentru clasele mai greu de recunoscut;
- Introducerea unui sistem de iluminare artificială locală, montată pe dispozitiv;

- Utilizarea unui model CNN mai profund, cu quantization-aware training, dacă resursele permit;
- Aplicarea unei corecții de perspectivă pentru imaginile OCR.

În concluzie, sistemul dezvoltat în cadrul acestui proiect reprezintă o soluție inovatoare și funcțională în sfera tehnologiilor asistive, demonstrând că, prin combinarea unor module inteligente precum CNN, OCR și TTS, se poate construi un dispozitiv portabil, autonom și util pentru persoane cu deficiențe de vedere. Rezultatele testării arată că, în ciuda constrângerilor hardware inerente utilizării unei platforme embedded precum Raspberry Pi Zero 2 W, performanțele obținute sunt relevante și pot oferi un suport real utilizatorilor, în special în activități cotidiene de identificare a obiectelor și citire a etichetelor sau mesajelor vizuale simple.

Limitările identificate în timpul testelor, cum ar fi dificultățile de clasificare între tacâmuri similare sau sensibilitatea OCR-ului la calitatea imaginii, nu reprezintă puncte slabe definitive, ci mai degrabă provocări deschise, care pot fi adresate prin îmbunătățiri graduale ale dataseturilor, arhitecturilor de model, calității optice sau software-ului de preprocesare. Este important de subliniat că scopul principal al proiectului nu a fost atingerea unei perfecțiuni tehnice, ci demonstrarea fezabilității unui prototip care să integreze capabilități multiple pe un hardware accesibil și să funcționeze complet offline, fără dependențe externe.

Pe parcursul dezvoltării s-au acumulat nu doar rezultate tehnice, ci și observații valoroase privind ergonomia, experiența utilizatorului și adaptabilitatea sistemului. Astfel, s-a conturat nevoia unei echilibrări fine între complexitatea software-ului și capacitatea hardware, între nivelul de precizie dorit și resursele disponibile, precum și între ușurința de utilizare și numărul de funcționalități oferite.

Pe termen lung, direcțiile de îmbunătățire pot include integrarea unor tehnologii suplimentare (de exemplu, senzori de adâncime, feedback haptic, control vocal avansat), îmbunătățirea experienței auditive prin folosirea unor sintetizatoare de voce naturale, dar și adaptarea la alte grupuri-țintă, cum ar fi persoanele cu dificultăți de citire sau cu alte tipuri de dizabilități senzoriale.

Astfel, sistemul propus nu este doar un rezultat tehnic, ci o bază solidă pentru cercetare, experimentare și dezvoltare viitoare în domeniul tehnologiilor asistive. El arată cum pot fi aduse împreună inteligența artificială, miniaturizarea hardware și designul centrat pe utilizator pentru a oferi soluții cu impact social real.

Capitolul 7. Manual de instalare - utilizare

7.1. Introducere generala

Acest manual descrie procesul complet de instalare, configurare - utilizare a sistemului embedded propus, destinat persoanelor cu deficiente de vedere. Sistemul consta intr-o pereche de ochelari inteligenti care identifica obiecte din mediul cotidian (tacamuri) - este capabil sa recunoasca texte imprimate, oferind feedback vocal utilizatorului. este modular - usor de adaptat la alte tipuri de recunoastere vizuala, putand fi folosit ca baza pentru aplicatii diverse.

7.2. Resurse hardware necesare

- **Raspberry Pi Zero 2 W** - placa de dezvoltare ARM, cu Wi-Fi integrat.
- **Modul camera** conectat prin CSI.
- **Modul audio** conectat prin bluetooth
- **Baterie externa** de 5V/2A, minim 10000mAh (autonomie 1-2 zile).
- **Buton fizic** pe GPIO pentru comutare OCR / CNN.
- **Card microSD** de 16GB cu Raspberry Pi OS.
- **Rame de ochelari** pentru fixarea componentelor.

7.3. Resurse software necesare

- **Sistem de operare:** Raspberry Pi OS Lite.
- **Python 3.9+** - pachete:
  - `numpy` – operații numerice pe array-uri.
  - `opencv-python` – procesare și manipulare imagini.
  - `Pillow` – încărcare și salvare imagini.
  - `tflite-runtime` – rulare modele TensorFlow Lite.
  - `RPi.GPIO` – control butoane/senzori Raspberry Pi.
  - `espeak-ng` – generare voce sintetică offline.
  - `tesseract-ocr` – recunoaștere text din imagini.

7.4. Comenzi pentru instalare dependinte Python si pachete auxiliare

Pentru ca sistemul sa functioneze corect, urmatoarele comenzi trebuie executate in terminalul Raspberry Pi pentru instalarea tuturor modulelor necesare (conform scripturilor prezentate):

```
sudo apt update && sudo apt upgrade -y
sudo apt install python3-pip python3-venv -y
sudo apt install espeak-ng tesseract-ocr libtesseract-dev -y

python3 -m venv smart\_env
source smart\_env/bin/activate
```

```
pip install numpy
pip install opencv-python
pip install pillow
pip install tflite-runtime
pip install RPi.GPIO
pip install pytesseract
```

Este important ca mediul virtual `smart_env` sa fie activat inainte de rularea scriptului principal.

7.5. Ghid de instalare software

7.5.1. 1. Instalare Raspberry Pi OS

1. Foloseste `Raspberry Pi Imager` pentru a scrie imaginea OS pe card.
2. Porneste placa conectata la monitor.
3. Configurare initiala: limba, Wi-Fi, user, update.

7.5.2. 2. Instalare mediu Python

```
sudo apt update && sudo apt upgrade
sudo apt install python3-pip python3-venv
python3 -m venv smart\_env
source smart\_env/bin/activate
pip install numpy opencv-python pillow tflite-runtime
sudo apt install espeak-ng tesseract-ocr
```

7.5.3. 3. Scripturi si fisiere necesare

- `start_myscript.sh` - script pentru pornire automata la pornire sistem
- `script.py` - script principal (control flux).
- `ocr.py`, `button.py`, `capture.py` - module auxiliare.
- `CNN.py` - creeare,antrenare model
- `convert.py` - convert to tflite
- `tacamuri.tflite` - model convertit.
- `Test.py` - testare pe laptop

7.6. Utilizarea aplicației

Acest sistem este conceput pentru a fi ușor de utilizat de către persoane fără experiență tehnică. Interacțiunea se face exclusiv prin voce și un buton fizic, fără a fi necesar un ecran sau interfață grafică.

7.6.1. Pornirea sistemului

- Utilizatorul pornește sistemul prin apăsarea butonului de alimentare de pe bateria externă.
- După câteva secunde, sistemul se inițializează complet și redă un mesaj vocal automat prin difuzor:

  `"Sistem pornit. Detectare obiecte activată."`

- Această confirmare auditivă indică faptul că modelul este deja încărcat, camera este funcțională, iar sistemul a intrat în modul de detecție obiecte (CNN).

- Nu este necesară nicio altă acțiune din partea utilizatorului pentru a începe folosirea sistemului.

7.6.2. Modul de detecție obiecte (CNN)

- Sistemul captează automat o imagine la fiecare 2 secunde prin camera integrată.
- Fiecare imagine este analizată de modelul CNN convertit în format `.tflite`.
- Rezultatul inferenței este o listă de probabilități pentru fiecare clasă (ex: farfurie, furculiță, lingură, pahar).
- Dacă probabilitatea clasei cu scor maxim este mai mare decât 0.7, sistemul consideră că obiectul este detectat corect.
- Se generează imediat un mesaj vocal relevant:

  `"Furculiță detectată"` sau `"Farfurie detectată"`

- Dacă probabilitatea este mai mică decât pragul de încredere (0.7), se redă:

  `"Obiect neclar. Ajustează poziția."`

- Acest lucru poate fi cauzat de:
  - iluminare slabă;
  - obiect necentrat în cadru;
  - obiect parțial vizibil;
  - fundal aglomerat.
- După fiecare detecție, sistemul introduce o scurtă pauză, permițând utilizatorului să schimbe obiectul sau să repoziționeze camera.

7.6.3. Modul OCR (recunoaștere text)

- Modul OCR poate fi activat în două moduri:
  - automat, dacă sistemul identifică un context textual (ex: un ambalaj cu etichetă);
  - manual, prin apăsarea butonului fizic conectat la interfața GPIO.
- La activare, se va auzi mesajul:

  `"Mod detectare text activat."`

- Sistemul capturează o imagine statică cu textul vizat.
- Sunt aplicate preprocesări pentru a îmbunătăți claritatea textului:
  - conversie în tonuri de gri;
  - binarizare adaptivă (pentru contrast mai bun);
  - filtrare mediană (pentru eliminarea zgomotului).
- Imaginea este transmisă către motorul Tesseract OCR, care extrage textul detectat.
- Rezultatul este transmis utilizatorului prin voce:

  `"Apa minerală"` sau `"Data expirării: 12.10.2025"`

- După citirea textului, sistemul revine automat în modul de detecție obiecte și redă mesajul:

  `"Mod detectare obiecte activat."`

7.6.4. Oprirea sistemului

- Oprirea sistemului se face simplu, prin apăsarea aceluiași buton de pe bateria externă folosit la pornire.

- Aceasta întrerupe complet alimentarea electrică a Raspberry Pi, închizând aplicația în siguranță.
- Nu este nevoie de comenzi sau operațiuni suplimentare pentru închiderea sistemului.
- Este recomandat ca sistemul să fie oprit atunci când nu este folosit, pentru a prelungi durata de viață a bateriei externe.

În concluzie, aplicația este gândită pentru a putea fi utilizată în mod intuitiv de către persoane fără cunoștințe tehnice, printr-o interfață simplă bazată pe voce și un singur buton fizic.

7.7. Recomandări finale

În urma testărilor extinse și a funcționării reale a sistemului, se pot formula următoarele recomandări pentru o utilizare optimă:

- **Recunoașterea obiectelor:** Sistemul oferă o acuratețe ridicată în cazul detectării farfuriilor, cu o rată de succes estimată la aproximativ **90%** în condiții normale (iluminare bună, obiect centrat). Totuși, în cazul obiectelor mai mici sau similare vizual, precum linguri, furculițe și cuțite, acuratețea scade semnificativ, fiind în jur de **50%**. Se recomandă poziționarea clară a obiectelor în centrul camerei, pe un fundal contrastant, pentru rezultate mai bune.
- **Limitări ale detecției vizuale:** Obiectele parțial acoperite, aflate în afara focusului sau în iluminare necorespunzătoare pot genera erori de clasificare sau lipsa detectării. De asemenea, dacă mai multe obiecte se află în cadru, sistemul va recunoaște doar obiectul dominant, fără a realiza segmentări multiple.
- **Recunoașterea textului (OCR):** Modulul OCR oferă rezultate bune în cazul textelor mari, tipărite, poziționate drept și bine iluminate. Textele scrise de mână, înclinate, sau de dimensiuni foarte mici nu sunt recunoscute corect. Pentru o funcționare corectă se recomandă:
  - Distanță optimă între cameră și text: 20–30 cm.
  - Evitarea umbrelor sau a reflexiilor puternice.
  - Orientarea paralelă a camerei cu planul textului.
- **Autonomie și portabilitate:** Sistemul alimentat de la o baterie externă de 5000–10000mAh poate funcționa în mod continuu între **24 și 48 de ore**, în funcție de intensitatea utilizării (număr de inferențe, activare OCR, nivel de iluminare). Se recomandă oprirea completă a sistemului când nu este utilizat, pentru a economisi energie.
- **Extensibilitate:** Arhitectura modulară a aplicației permite înlocuirea ușoară a modelului de detecție cu unul antrenat pe alte clase de obiecte. Astfel, aplicația poate fi refolosită pentru scopuri educaționale, asistență în alte contexte (ex: identificarea fructelor, medicamentelor, etichetelor de raft, etc.). Pentru reantrenarea modelului se recomandă capturarea unui nou set de imagini și rularea scriptului de antrenare descris în capitolele anterioare.
- **Acces la sistem:** În caz de necesitate (pentru depanare, configurare sau încărcarea unui model nou), sistemul poate fi accesat prin conectarea la un monitor, tastatură și mouse. Alternativ, se poate activa conexiunea SSH pentru control de la distanță, fără a fi nevoie de interfață grafică.

În concluzie, sistemul propus este robust, funcțional și accesibil din punct de vedere al utilizatorului final. Cu respectarea condițiilor optime de utilizare, poate oferi o experiență asistivă eficientă, contribuind la creșterea autonomiei persoanelor cu deficiențe de vedere.

Capitolul 8. Concluzii

8.1. Rezumat al contribuțiilor

În cadrul acestei lucrări a fost proiectat, implementat și testat un sistem embedded inteligent sub forma unei perechi de ochelari asistați vizual, capabili să detecteze obiecte și să recunoască texte din mediul înconjurător, oferind feedback auditiv utilizatorului.

Principalele contribuții ale proiectului sunt:

- **Dezvoltarea unui model CNN personalizat** antrenat pentru detecția obiectelor de uz casnic (farfurie, lingură, furculiță, pahar), cu performanță ridicată și arhitectură optimizată pentru dispozitive embedded.
- **Conversia modelului în format TensorFlow Lite** și rularea acestuia pe o platformă hardware limitată (Raspberry Pi Zero 2 W), asigurând inferență în timp real.
- **Integrarea modulului OCR** folosind motorul open-source Tesseract, pentru recunoașterea textelor imprimate, cu procesare locală și fără conexiune la internet.
- **Implementarea unei interfețe vocale intuitive** cu ajutorul sintetizatorului `espeak-ng`, oferind un sistem complet hands-free, ușor de utilizat pentru persoanele cu deficiențe de vedere.
- **Crearea unei interfețe fizice minimaliste** bazată pe un singur buton GPIO pentru comutarea între modurile CNN și OCR.
- **Realizarea unui proces complet de testare și validare**, atât pe computer cât și pe Raspberry Pi, în condiții reale de utilizare, cu accent pe limitările tehnice și comportamentul sistemului.

8.2. Analiza rezultatelor

Rezultatele obținute sunt promițătoare, demonstrând fezabilitatea unui astfel de sistem asistiv construit cu resurse accesibile și tehnologii open-source. Cu toate acestea, în timpul testărilor și utilizării s-au remarcat o serie de limitări și observații importante:

- **Detectarea farfuriei** a fost realizată cu un grad ridicat de acuratețe (aproximativ 90%), însă recunoașterea altor tacâmuri precum lingura, furculița și cuțitul a prezentat dificultăți, cu o rată de corectitudine de doar 50–60%. Acest lucru este explicabil prin similaritatea formelor, reflexiile metalice și dimensiunile mai mici ale obiectelor.
- **Modulul OCR** funcționează eficient doar în condiții bune de iluminare, text tipărit clar, font mare și poziționare centrală. Textul mic, înclinat sau scris de mână nu este recunoscut corect. Această limitare este cauzată în principal de rezoluția camerei și lipsa unor corecții avansate de perspectivă.
- **Interfața auditivă** s-a dovedit eficientă și suficientă pentru utilizatorul țintă, eliminând complet nevoia de ecran sau interacțiune tactilă.
- **Autonomia sistemului** a fost testată cu o baterie externă de capacitate medie, permițând o funcționare continuă între 24 și 48 de ore, în funcție de intensitatea utilizării.

Deși sistemul funcționează în mod stabil și își atinge scopul principal, performanța generală este influențată puternic de calitatea componentelor hardware (în special ca-

mera) și de condițiile externe (iluminare, distanță, poziție). De asemenea, modelul CNN poate fi îmbunătățit pentru a include mai multe clase de obiecte și pentru a fi mai robust la variații de fundal sau unghi de captură.

8.3. Direcții de dezvoltare viitoare

Pentru viitor, sistemul poate fi extins și îmbunătățit considerabil. Printre direcțiile de dezvoltare recomandate se numără:

- **Îmbunătățirea calității camerei**: utilizarea unui modul de cameră de rezoluție mai mare, cu lentilă fixă sau autofocus, ar permite obținerea unor rezultate OCR mai bune și o detecție mai precisă a obiectelor.
- **Antrenarea unui model CNN mai complex**: utilizarea unui model preantrenat (ex: MobileNet, EfficientNet) și aplicarea transfer learning-ului ar putea duce la o creștere semnificativă a acurateței, fără costuri suplimentare de procesare.
- **Extinderea setului de date**: adăugarea mai multor imagini din contexte diferite (fundaluri variate, unghiuri multiple, obiecte în mișcare) ar crește capacitatea de generalizare a modelului.
- **Segmentarea multiplă a imaginii**: în loc să se detecteze un singur obiect dominant, sistemul ar putea identifica și clasifica simultan mai multe obiecte dintr-un cadru, folosind modele cu suport pentru detecție multiplă (ex: SSD, YOLO).
- **Feedback adaptiv**: în viitor, sistemul ar putea ajusta automat parametrii camerei sau alegerile de preprocesare în funcție de condițiile de lumină, contrast sau distanță.
- **Asistență prin Bluetooth/Wi-Fi**: integrarea unei aplicații mobile sau a unei interfețe web ar permite monitorizarea și controlul sistemului la distanță de către un asistent sau terapeut.
- **Extinderea scopului aplicației**: sistemul poate fi adaptat ușor pentru detecția altor obiecte (ex: medicamente, semne de circulație, produse alimentare) prin reantrenarea modelului CNN cu un alt set de date.

8.4. Concluzie generală

Sistemul propus reușește să demonstreze utilitatea unei soluții embedded asistive bazate pe viziune artificială și feedback vocal. Prin combinarea componentelor hardware accesibile cu metode moderne de învățare automată și procesare de imagine, se poate oferi un grad înalt de independență persoanelor cu deficiențe de vedere.

Implementarea este modulară, extensibilă și documentată complet, permițând adaptări rapide și personalizări. Deși există limitări tehnice inerente platformei hardware, rezultatele obținute sunt încurajatoare și susțin ideea dezvoltării în continuare a unor astfel de sisteme inteligente, cu impact social semnificativ.

Bibliografie

[1] Kiran Gurav, Neel Joshi, Nida Desai, Shruti Ghorpade, "A raspberry pi-based text reader object detection system," *International Journal of Advanced Research in Science, Communication and Technology*, 2024. [Online]. Available: https://ijisrt.com/a-raspberry-pibased-text-reader-object-detection-system

[2] Allen Atienza Llorca, Mia Villar Villarica, Harrold Molinyawe Gueta, Mark Angelo Torres Mercado, "Ai-wear: Smart text reader for blind students," *International Journal of Research in Engineering and Technology*, 2023. [Online]. Available: https://www.ijarcs.info/index.php/Ijarcs/article/view/6997

[3] Prof. Teena Varma , "Image processing based on ocr with text-to-speech," *Journal of Scientific and Engineering Research*, 2023. [Online]. Available: https://www.ijert.org/research/text-extraction-from-image-and-text-to-speech-conversion-IJERTCONV9IS03002.pdf

[4] Ahmed Ben Atitallah, "A novel real-time tts system using raspberry pi 4," *ResearchGate*, 2024. [Online]. Available: https://researchgate.net

[5] Sushmitha Mr, "Raspberry pi based object detection and text reader with voice," *Sathyabama University*, 2024. [Online]. Available: https://www.researchgate.net/publication/367875259_Raspberry_Pi_based_Object_Detection_and_Text_Reader_using_Voice_Assistance

[6] H. Rithika, B. Nithya Santhoshi, "Raspberry pi-based image-to-speech with blur detection," *International Journal of Modern Engineering Research*, 2023. [Online]. Available: https://irjmets.com

[7] Mrs. S.Santhana Prabha, "Cnn-based object recognition for visually impaired," *IJARSCT*, 2023. [Online]. Available: https://ijsart.com/public/storage/paper/pdf/IJSARTV9I463444.pdf#:~:text=This%20project%20proposes%20an%20assistive%20system%20for%20the,Open%20CV%20under%20Python%2C%20to%20provide%20auditory%20details.

[8] Abhinav Benagi, Dhanyatha Narayan, Charith Rage, A. Sushmitha, "Artificial eye for the blind," *ResearchGate*, 2023. [Online]. Available: https://researchgate.net

[9] Unknown Authors, "Cnn-based live object recognition as blind aid," *Journal of Assistive Technologies*, 2018.

[10] Heba Najm, "Assisting blind people using object detection with vocal feedback," *Arxiv*, 2023. [Online]. Available: https://arxiv.org

[11] Mirza Samad Ahmed Baig, "Ai-based wearable vision assistance system," *Arxiv*, 2024. [Online]. Available: https://arxiv.org