**Universitatea Tehnică de Construcții din București**

**Facultatea de Hidrotehnică**

**Specializarea: Automatică și Informatică Aplicată**

**LUCRARE DE LICENȚĂ**

**Coordonator științific: Absolvent:**

**Ș.l. univ. dr. ing. Oana-Ramona FLANGEA Raluca-Ioana BRATU**

**București, 2024**

**CUPRINS**

[1. INTRODUCERE 3](#_Toc167966302)

[1.1. Sumar introductiv 3](#_Toc167966303)

[1.2. Motivarea alegerii temei 3](#_Toc167966304)

[1.3. Obiectivele propuse în cadrul proiectului 4](#_Toc167966305)

[1.4. Structura lucrării 5](#_Toc167966306)

[2. Aspecte teoretice referitoare la tema abordată 6](#_Toc167966307)

[2.1. Domenii de aplicare 6](#_Toc167966308)

[2.2. Sisteme de detectare și clasificare a imaginilor 7](#_Toc167966309)

[2.3. Machine Learning 9](#_Toc167966310)

[2.3.1. Învățarea supervizată 9](#_Toc167966311)

[2.3.2. Arbori de decizie 10](#_Toc167966312)

[2.3.3. Învățarea nesupervizată 10](#_Toc167966313)

[2.3.4. Învățarea semi-supervizată 11](#_Toc167966314)

[2.3.5. Antrenarea unui model 12](#_Toc167966315)

[2.4. Rețele Neuronale Artificiale 12](#_Toc167966316)

[2.4.1. Arhitecturile retelelor neuronale 12](#_Toc167966317)

[2.4.2. Învățare Hebbiană 13](#_Toc167966318)

[2.4.3. Regula de învățare a perceptronului 13](#_Toc167966319)

[2.4.4. Backpropagation learning 14](#_Toc167966320)

[2.4.5. Antrenarea și testarea rețelelor neuronale 14](#_Toc167966321)

1. INTRODUCERE
   1. Sumar introductiv

Această lucrare dorește realizarea unei aplicații dezvoltată pentru mediul Android, având ca scop identificarea obiectelor în timp real utilizând machine learning. Fluxul video al dispozitivului utilizat va fi analizat de către algoritmul aplicației și va oferi un raspuns imediat. Interfața intuitivă, dezvoltată cu Flutter și Dart, asigură o utilizare ușoară și accesibilă, în timp ce pentru partea de învățare automată am utilizat TensorFlow Lite.

* 1. Motivarea alegerii temei

Alegerea dezvoltării unei aplicații de identificare a obiectelor utilizând machine learning a fost motivată de mai mulți factori, atât pe plan personal, însă și de dorința de a oferi o soluție personalizabilă pentru mai multe tipuri de utilizatori. Pe plan personal, interesul meu pentru aplicațiile Android și machine learning a jucat un rol central în această decizie. Întotdeauna m-a intrigat modul în care tehnologia poate schimba și îmbunătăți viața de zi cu zi, iar crearea acestei aplicații mobile este o modalitate excelentă de a-mi extinde cunoștințele. Studiul învățării automate m-a captivat prin complexitatea sa și capacitatea sa de a rezolva o varietate de probleme, cum ar fi recunoașterea obiectelor, analiza datelor și predicțiile.

Mai mult, am dorit să construiesc o aplicație care să nu fie limitată la un singur tip de utilizator sau funcție. Prin urmare, am creat o soluție care poate fi personalizată rapid și simplu prin încărcarea mai multor modele de învățare automată în aplicație. Această flexibilitate permite aplicației să se adapteze la o varietate de domenii, cum ar fi învățământul și turismul.

În ceea ce privește educația, aplicația poate ajuta utilizatorii să învețe limbi străine prin identificarea și denumirea diferitelor obiecte în mai multe limbi, facilitând o metodă de învățare interactivă și vizuală. Această funcționalitate îmbunătățește experiența de învățare și motivează utilizatorii să folosească tehnologia modernă și captivantă.

În industria turismului, aplicația poate ajuta la recunoașterea și identificarea monumentelor istorice și a punctelor de interes, oferind utilizatorilor informații detaliate și relevante despre locurile pe care le vizitează. Prin urmare, turiștii pot avea la dispoziție un ghid digital personal, care le îmbunătățește experiența de călătorie și le oferă cunoștințe utile despre locațiile pe care le vizitează.

Această metodă reprezintă dorința mea de a dezvolta o aplicație care să fie utilă pentru o gamă largă de utilizatori. Scopul meu este de a oferi o soluție tehnologică integrată care să răspundă nevoilor diferitelor utilizatori și să demonstreze flexibilitatea și puterea tehnologiilor de dezvoltare mobilă și machine learning. În consecință, alegerea acestei teme a reprezentat nu numai un pas semnificativ în creșterea mea profesională, ci și un simbol al efortului meu de a oferi soluții utile și eficiente folosind tehnologia.

* 1. Obiectivele propuse în cadrul proiectului

Obiectivul principal al proiectului este crearea unei soluții inovatoare, accesibile și eficiente pentru identificarea obiectelor folosind învățarea automată. Această soluție satisface nevoile diverse ale utilizatorilor din diferite domenii. Identificarea ideii și crearea conceptului aplicației a fost primul pas către atingerea acestui obiectiv. Prin urmare, am recunoscut că este necesară o soluție rapidă și precisă pentru identificarea obiectelor în imagini, care ar putea fi utilizată în mai multe industrii, inclusiv retail, turism și educație. Scopul definit a fost crearea unei aplicații mobile care să folosească machine learning pentru a recunoaște și a oferi informații despre obiectele din imagini, adaptabilă pentru diverse utilizări.

Un alt obiectiv major a fost alegerea și utilizarea tehnologiilor potrivite. Deoarece framework-urile Flutter și limbajul Dart sunt capabile să producă aplicații atractive și responsive, care funcționează pe mai multe platforme cu o singură bază de cod, acestea au fost alese pentru front-end. Pentru partea de machine learning, am folosit TensorFlow Lite, selectat pentru optimizarea sa specifică pentru dispozitive mobile, asigurând performanțe ridicate și eficiență în utilizarea resurselor.

Pentru alegerea modelului de machine learning, am luat în considerare modul de dezvoltare și antrenare al diverselor opțiuni. După o analiză atentă, am decis asupra utilizării modelului SSD MobileNet. SSD (Single-Shot multibox Detector) MobileNet a fost ales pentru că oferă o combinație perfectă de viteză și precizie, care sunt vitale pentru aplicațiile mobile. Acest model este cunoscut pentru eficiența sa în detectarea obiectelor în timp real, având o arhitectură ușoară, care permite rularea eficientă pe dispozitive cu resurse limitate, cum sunt cele mobile. LIMBAJUL PYTHON PENTRU ANTRENAREA MODELULUI. GOOGLE COLAB pentru rularea codului python.

Un alt obiectiv major a fost dezvoltarea funcționalităților aplicației. Am pus accent pe crearea unei interfețe intuitive și prietenoase, care să permită utilizatorilor să primească informații despre obiectele identificate cu ușurință. Interfața care a fost construită cu ajutorul Flutter a permis crearea unei aplicații atractive și responsive, capabile să funcționeze pe mai multe platforme, având o bază de cod unică. În plus, am luat în considerare modul de dezvoltare al aplicației pentru a permite adăugarea ulterioară a funcționalităților suplimentare. Această metodă modulară și flexibilă îmi permite adăugarea unor noi caracteristici fără a afecta funcționalitățile existente.

* 1. Structura lucrării

În capitolul de \*\*Introducere\*\*, se prezintă necesitatea și relevanța unei astfel de aplicații în diverse domenii, se definesc obiectivele principale și specifice ale proiectului și se descrie modul în care este organizată lucrarea.

Capitolul de \*\*Motivare a alegerii temei\*\* discută motivațiile personale și profesionale pentru alegerea acestei teme, precum interesul pentru machine learning și dezvoltarea aplicațiilor Android. De asemenea, se prezintă beneficiile și impactul potențial al aplicației în domenii variate, cum ar fi educația și turismul.

În \*\*Studiul literaturii\*\*, se revizuiește literatura existentă privind utilizarea Flutter, Dart și TensorFlow Lite în dezvoltarea aplicațiilor mobile. Se analizează, de asemenea, algoritmii de machine learning pentru identificarea obiectelor, cu accent pe modelul SSD MobileNet și avantajele acestuia.

Capitolul de \*\*Conceperea și proiectarea aplicației\*\* include analiza cerințelor funcționale și non-funcționale ale aplicației. Arhitectura software este descrisă, incluzând modulele principale și interacțiunile dintre acestea. De asemenea, se prezintă principiile de design utilizate pentru a crea o interfață intuitivă și prietenoasă pentru utilizator.

În \*\*Implementarea aplicației\*\*, se detaliază dezvoltarea front-end-ului folosind Flutter și Dart pentru crearea interfeței și logica de interacțiune a utilizatorului. Se descrie integrarea și optimizarea modelului SSD MobileNet cu TensorFlow Lite, precum și fluxul de prelucrare a imaginii pentru identificarea obiectelor în timp real.

Capitolul de \*\*Concluzii și lucrări viitoare\*\* recapitulă principalele realizări și contribuții ale proiectului, discută provocările întâmpinate și limitările aplicației și sugerează posibile îmbunătățiri și extinderi ale aplicației, cum ar fi adăugarea de noi funcționalități sau îmbunătățirea performanței.

1. ASPECTE TEORETICE REFERITOARE LA TEMA ABORDATĂ
   1. Domenii de aplicare

Aplicația de identificare a obiectelor prin machine learning reprezintă o soluție inovatoare cu multiple domenii de aplicare. În domeniul educației, poate servi ca un instrument interactiv de învățare, facilitând recunoașterea și înțelegerea obiectelor din mediul înconjurător. De exemplu, învățarea limbilor străine poate fi îmbunătățită prin asocierea vizuală a cuvintelor cu obiectele reale, iar în lecțiile de istorie sau biologie, elevii pot identifica rapid monumente sau specii de plante și animale.

Un alt domeniu este industria turismului, unde aplicația poate schimba experiența turiștilor, ajutându-i să găsească rapid monumente istorice sau alte puncte de interes și să obțină informații relevante despre ele. Acest lucru contribuie la o explorare mai interactivă și educativă a destinațiilor turistice, îmbunătățind în același timp experiența turistică.

De asemenea, în industria medicală, poate fi folosită pentru identificarea și catalogarea instrumentelor medicale sau pentru asistarea personalului medical în diagnosticarea rapidă și precisă a unor afecțiuni. În ultimii ani, diverse tipuri de procesare și recunoaștere a imaginilor medicale au adoptat metode de învățare profundă (deep learning), inclusiv imagini endoscopice, imagini CT/RMN, imagini cu ultrasunete, imagini patologice etc. În prezent, tehnologia de învățare profundă este utilizată în principal în clasificare și segmentare în imaginile medicale (Cai, Gao, & Zhao, 2020).

În sectorul agricol, aplicația de identificare a obiectelor prin tehnici de învățare automată poate avea multiple aplicații și beneficii semnificative. Progresele continue în domeniul științei patologiei plantelor sunt esențiale pentru a îmbunătăți controlul bolilor și pentru a răspunde schimbărilor în presiunea bolilor, determinate de evoluția și deplasarea patogenilor plantelor, precum și de modificările în practicile agricole. Procesarea imaginilor poate fi o unealtă valoroasă în contextul aplicațiilor agricole, având multiple scopuri. Aceasta poate fi folosită pentru detectarea frunzelor, tulpinilor și fructelor afectate de boli, pentru a măsura zona afectată de boli, pentru a măsura forma și culoarea zonei afectate, precum și pentru a măsura dimensiunea și forma fructelor. (Prakash, Saravanamoorthi, Sathishkumar, & Parimala, 2017).

* 1. Sisteme de detectare și clasificare a imaginilor

Sistemele de detectare și clasificare a imaginilor reprezintă o componentă crucială în domeniul computer vision, oferind capacități semnificative de înțelegere a conținutului vizual al imaginilor. Aceste sisteme utilizează tehnici avansate de prelucrare a imaginilor și algoritmi de machine learning pentru a detecta, localiza și clasifica obiectele în imagini. În ultimele decenii, a existat un progres semnificativ în clasificarea imaginilor în următoarele domenii: dezvoltarea și utilizarea algoritmilor de clasificare avansați, cum ar fi algoritmii de clasificare subpixel, per-field și bazat pe cunoștințe, utilizarea diferitelor caracteristici de teledetecție, cum ar fi informații spectrale, spațiale, multitemporale și multisenzoriale, și încorporarea datelor complementare în procedurile de clasificare, date precum topografia, solul, rețelele rutiere și datele de recensământ (3).

Detectarea obiectelor în imagini înseamnă găsirea și localizarea obiectelor relevante într-o imagine. Rețelele neuronale convoluționale (CNN) sunt printre metodele cele mai utilizate în acest sens. Aceste rețele sunt capabile să extragă informații importante din imagini și să le folosească pentru a determina prezența și poziția obiectelor. Încă din anii 1980, rețelele neuronale convoluționale (CNN) au fost folosite în sarcini vizuale. Cu toate acestea, în ciuda unor aplicații separate, au rămas inactive până în anii 2000, când puterea de calcul a crescut și a apărut o cantitate mare de date etichetate, completată de algoritmi îmbunătățiți. Acest lucru le-au adus în prim planul unei renașteri rapide a rețelelor neuronale începând din 2012 (4). Există diverse metode de detectare a obiectelor în imagini, fiecare cu propriile sale caracteristici și avantaje. Printre acestea se numără:

1. R-CNN (Region-based Convolutional Neural Networks): Această metodă implică propunerea de regiuni candidat care pot conține obiecte. Mai întâi, se identifică regiunile propuse, apoi se extrag caracteristicile relevante din aceste regiuni și, în final, se clasifică obiectele utilizând o rețea neuronală.
2. Fast R-CNN și Faster R-CNN: Acestea sunt îmbunătățiri ale metodei R-CNN, concentrate pe optimizarea procesului de extragere a regiunilor și a clasificării. Prin accelerarea acestor etape, aceste abordări reduc timpul necesar pentru detectare.
3. YOLO (You Only Look Once): Această este o altă metodă populară care se concentrează pe eficiență. YOLO utilizează o singură rețea neuronală pentru a prezice direct clasele și localizările obiectelor într-o singură etapă. Această abordare simplifică semnificativ procesul de detectare, fiind rapidă și eficientă în timpul de execuție.

În studiile de caz (5) ce au ca temă principală aceste tehnici se observă performanțele și provocările acestor tehnici în ceea ce privește viteza, precizia sau simplitatea care sunt, de asemenea, comparate. De exemplu, performanța în ceea ce privește viteza a metodei YOLO este de aproximativ 21-155 cadre pe secundă, fiind cea mai rapidă, în timp ce precizia medie a Mask R-CNN este de aproximativ 47.3, depășind toate celelalte tehnici (5).

O altă tehnică eficientă este învățarea prin transfer (în engleză transfer learning). Aceasta implică pre-antrenarea unei rețele neuronale pe un set mare de date și fine-tuning-ul acesteia pentru a rezolva o problemă specifică de clasificare a imaginilor. Există mai multe tipuri de transfer learning (6), precum:

* Symmetric feature-based transfer learning;
* Parameter-based transfer learning;
* Relational-based transfer learning;
* Hybrid-based (instance and parameter) transfer learning;

Sisteme de învățare ansamblu (în engleză ensemble learning) au devenit din ce în ce mai populare în comunitatea de inteligență computatională și învățare automată în ultimele decenii. S-a demonstrat că sistemele de ansamblu sunt extrem de eficiente și extrem de versatile în ceea ce privește o gamă largă de domenii de problemă și aplicații practice. Sistemele de ansamblu au fost folosite mai târziu pentru a rezolva o varietate de probleme de învățare automată, inclusiv estimarea încrederii, caracteristicile lipsă, selecția de caracteristici, corectarea erorilor, datele neechilibrate pe clase, învățarea incrementală, datele neechilibrate, învățarea schimbării conceptului din distribuții nestationare și, în mod implicit, îmbunătățirea preciziei unui sistem de luare a deciziilor automatizat (7).

* 1. Machine Learning

Machine Learning (ML) este o ramură a inteligenței artificiale (IA) care se concentrează pe dezvoltarea sistemelor și algoritmilor care permit calculatoarelor să învețe și să îmbunătățească performanța unei anumite sarcini în mod autonom, fără a fi explicit programate pentru aceasta. În esență, învățarea automată oferă computerelor capacitatea de a învăța din date și experiență pentru a face predicții sau decizii viitoare. În scopul emulării inteligenței umane prin învățarea din mediul înconjurător, această subramură este în continuă evoluție. În noua eră a big data, algoritmii computaționali sunt considerați „cai de muncă”. De la recunoașterea modelelor, viziunea artificială, ingineria spațială, finanțele, divertisment și biologie computatională până la aplicații biomedicale și medicale, tehnologiile bazate pe învățare automată au fost folosite cu succes în multe domenii (8).

* + 1. Învățarea supervizată

Învățarea supervizată (în engleză supervised learning) este sarcina de învățare automată care constă în învățarea unei funcții care mapează o intrare la o ieșire pe baza perechilor de exemple intrare-ieșire. Ea deduce o funcție din date de antrenare etichetate, care constau într-un set de exemple de antrenare. Algoritmii de învățare supervizată sunt acei algoritmi care necesită asistență externă. Setul de date de intrare este împărțit în set de date de antrenare și de testare. Setul de date de antrenare are variabila de ieșire care trebuie prezisă sau clasificată (9).

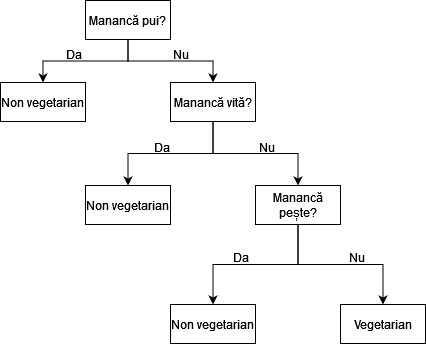
**A diagram of a model

Description automatically generated**

**Figura 2.3.1.** Fluxul de funcționare a învățării supervizate

* + 1. Arbori de decizie

Un arbore de decizie este un grafic care reprezintă alegerile și rezultatele lor sub forma unui arbore. Nodurile din grafic reprezintă un eveniment sau o alegere, iar muchiile graficului reprezintă regulile sau condițiile deciziei. Fiecare arbore constă din noduri și ramuri. Fiecare nod reprezintă atribute într-un grup care urmează să fie clasificate, iar fiecare ramură reprezintă o valoare pe care nodul o poate lua (9).



**Figura 2.3.2.** Exemplu de arbore de decizie

* + 1. Învățarea nesupervizată

În comparație cu învățarea supervizată menționată anterior, aceste tipuri de algoritmi sunt cunoscute sub numele de învățare nesupervizată (în engleză unsupervised learning), deoarece nu există un profesor sau răspunsuri adecvate. Astfel, aceștia sunt lăsați să găsească și să prezinte structuri în datele colectate. Algoritmii de învățare nesupervizată pot învăța un număr limitat de caracteristici din datele care sunt disponibile. Când se introduc date noi, aceștia folosesc caracteristicile pe care le-au învățat anterior pentru a recunoaște clasa datelor. Învățarea nesupervizată este folosită în principal pentru grupare (în engleză clustering) și reducerea caracteristicilor (în engleză feature reduction) (9).

Gruparea este procesul de împărțire a unui set de date în grupuri, sau clustere, astfel încât datele din fiecare grup să fie mai similare între ele decât cu datele din alte grupuri. Aceasta este utilă pentru a identifica segmentele de piață, pentru a examina comportamentul utilizatorilor sau pentru a găsi anomalii. K-means este un algoritm de grupare comun care împarte datele în grupuri K în funcție de similitudinile caracteristicilor.

Reducerea caracteristicilor reduce numărul de variabile (caracteristici) utilizate în analiză pentru a simplifica setul de date. Aceasta reduce zgomotul și îmbunătățește performanța algoritmilor de învățare automată. Pentru a reduce dimensalitatea datelor și pentru a menține variația maximă în cel mai mic set de caracteristici, tehnici precum Analiza Componentelor Principale (PCA) sunt frecvent utilizate.

* + 1. Învățarea semi-supervizată

Învățarea semi-supervizată folosește avantajele învățării supervizate și nesupervizate pentru a face față problemelor specifice ale muncii cu date incomplete. Datele etichetate care sunt necesare pentru antrenarea modelelor supervizate sunt de obicei costisitoare și necesită timp. Datorită faptului că nu sunt etichetate, datele neetichetate sunt adesea abundente și ușor de colectat. Învățarea semi-supervizată folosește aceste date neetichetate pentru a îmbunătăți performanța modelelor. Astfel, un număr mic de date etichetate ghidează procesul de învățare (9).

Învățarea semi-supervizată este utilă în multe domenii, în special acolo unde există o mulțime de date neetichetate și etichetarea manuală este dificilă sau costisitoare. Pentru a îmbunătăți precizia clasificării, un număr mic de imagini etichetate pot fi combinate cu un număr mare de imagini neetichetate în recunoașterea imaginilor și viziunea computerizată (9). Îmbunătățirea seturilor de date etichetate cu texte neetichetate poate îmbunătăți performanța modelelor în sarcini precum analiza sentimentelor sau traducerea automată în procesarea limbajului natural (NLP).

Învățarea semi-supervizată poate fi utilizată pentru a antrena modele de diagnosticare folosind un set mic de date medicale etichetate împreună cu o cantitate mare de date neetichetate în domeniul medical, unde obținerea datelor etichetate de înaltă calitate este adesea dificilă și costisitoare (9).

* + 1. Antrenarea unui model

Antrenarea unui model înseamnă determinarea valorilor bune pentru toate greutățile și decalajele din exemplele etichetate. Un algoritm de învățare automată folosește metoda de minimizare a riscului empiric în învățarea supervizată pentru a crea un model examinând mai multe exemple și încercând să descopere un model care minimizează pierderea (12).

Pierderea este o pedeapsă pentru presupunerea greșită. Cu alte cuvinte, pierderea este un număr care arată cât de proastă a fost predicția modelului pentru un singur exemplu. Pierderea este zero dacă predicția modelului este perfectă; în caz contrar, pierderea este mai mare (12). Scopul antrenării unui model este de a determina un set de greutăți și decalaje cu o pierdere medie scăzută pentru fiecare exemplu.

* 1. Rețele Neuronale Artificiale

O tehnică de învățare automată care a derivat din ideea de a imita creierul uman este rețeaua neuronală artificială (ANN), cunoscută și sub numele simplu de rețea neuronală. Modelele matematice ale sistemelor nervoase biologice au servit drept bază pentru dezvoltarea rețelelor neuronale. După introducerea neuronilor simplificați de către McCulloch și Pitts în 1943, rețelele neuronale (cunoscute și sub numele de modele conexioniste sau procesare distribuită paralelă) au suscitat interesul inițial (10). Elementele de bază ale rețelelor neuronale sunt neuronii artificiali sau, denumiți simplu, neuroni sau noduri. Într-un model matematic simplificat al neuronului, efectele sinapselor sunt reprezentate prin greutăți de conexiune (în engleză connection weights), care modulează influența semnalelor de intrare asociate, iar caracteristica neliniară a neuronilor este reprezentată printr-o funcție de transfer (10). Capacitatea de învățare a unui neuron artificial se obține prin ajustarea greutăților în conformitate cu algoritmul de învățare ales.

* + 1. Arhitecturile retelelor neuronale

Arhitectura de bază constă din trei tipuri de straturi neuronale: stratul de intrare, stratul ascuns și stratul de ieșire. Rețelele de tip feed-forward au fluxul semnalului strict în direcția feed-forward de la unitățile de intrare la unitățile de ieșire. În ciuda faptului că datele sunt procesate în mai multe niveluri de unități, nu există conexiuni de feedback. În schimb, rețelele recurente au conexiuni de feedback. Spre deosebire de rețelele feed-forward, proprietățile dinamice ale rețelei sunt importante. În unele cazuri, valorile de activare ale unităților suferă un proces de relaxare astfel încât rețeaua evoluează către o stare stabilă în care aceste activări nu se mai schimbă (10). O rețea neuronală trebuie configurată astfel încât să producă setul de ieșiri dorit atunci când se aplică un set de intrări. Intensitatea conexiunilor poate fi determinată în moduri diferite. O metodă este de a stabili greutățile în mod clar, folosind informații prealabile. Rețeaua neuronală poate fi antrenată într-un alt mod prin furnizarea de modele de învățare și permiterea acesteia să-și schimbe greutățile în funcție de o regulă de învățare (10).

A diagram of a network

Description automatically generated

**Figura 2.4.1.** O rețea neuronala feed-forward cu cele 3 straturi (13).(refacut in romana)

Există trei categorii distincte de situații de învățare în rețelele neuronale: învățare supervizată, învățare nesupervizată și învățare prin armare (în engleză reinforcement learning). Învățarea prin armare constă în învățarea ce acțiuni să fie întreprinse - cum să se mapeze situațiile la acțiuni - pentru a maximiza un semnal numeric de recompensă. În comparație cu majoritatea formelor de învățare automată, algoritmul nu este instruit ce acțiuni să facă; mai degrabă, el trebuie să descopere ce acțiuni generează cea mai mare recompensă prin încercarea lor. În cele mai interesante cazuri, acțiunile pot afecta nu doar recompensa imediată, ci și situația următoare și, astfel, toate recompensele ulterioare (10). Aceste două caracteristici, căutarea prin încercare și eroare (în engleză trial-and-error) și recompensa amânată, sunt cele două trăsături distinctive cele mai importante ale învățării prin armare.

* + 1. Învățare Hebbiană

Paradigmele de învățare duc la ajustarea complicată a conexiunilor dintre unități, conform unei reguli de modificare. Regula lui Hebb (1949), care a oferit o teorie a comportamentului bazată pe cât posibil pe fiziologia sistemului nervos, a fost poate cea mai importantă lucrare din istoria conecționismului (10). Conecționismul susține că procesele cognitive trebuie explicate prin interacțiunile elementelor neuronale interconectate (11).

Această abordare subliniază importanța conexiunilor și a modului în care acestea se formează și se modifică în timpul învățării și a dezvoltării cognitive. Regula lui Hebb este adesea rezumată prin sintagma "neuronii care se declanșează împreună, se conectează împreună", evidențiind importanța activității simultane a neuronilor în procesul de consolidare a conexiunilor sinaptice (10).

* + 1. Regula de învățare a perceptronului

Perceptronul este o rețea neuronală cu un singur strat, așa că își poate antrena greutățile și decalajele pentru a crea un vector țintă corect atunci când este prezentat cu vectorul de intrare potrivit (10). Această regulă este tehnica de antrenare alesă, luând în calcul că perceptronul este ideal pentru problemele simple de clasificare a modelelor. Procedura aceasta este foarte similară cu regula lui Hebb, singura diferență constă în faptul că atunci când rețeaua răspunde corect, nu se modifică greutățile conexiunilor (10).

Regula de învățare a perceptronului este un exemplu simplu de algoritm de învățare supervizată în care modelul este instruit să obțină ieșirile dorite pe baza unui set de exemple de intrare și ieșire corecte. Acest lucru face perceptronul un instrument puternic pentru problemele de clasificare binară, cum ar fi identificarea obiectelor în imagini sau diagnosticarea medicală bazată pe date.

* + 1. Backpropagation learning

Perceptronul simplu poate gestiona doar probleme liniar separabile sau liniar independente. Prin calcularea derivatei parțiale a erorii rețelei în raport cu fiecare greutate, vom învăța puțin despre direcția în care se mișcă eroarea rețelei (10). Acest algoritm este cunoscut drept algoritmul de propagare inversă ( în engleză backpropagation algorithm) deoarece aplică aceste derivate parțiale la fiecare greutate, începând de la greutățile stratului de ieșire până la greutățile stratului ascuns și apoi până la greutățile stratului de intrare. Acest lucru demonstrează că este necesar să se cunoască derivațele parțiale calculate în stratul imediat următor pentru a modifica acest set de greutăți (10).

Rețelele neuronale sunt capabile să învețe și să se adapteze la modele complexe prin procesul de backpropagation. Acest proces le permite și să detecteze și să corecteze erorile în timpul antrenamentului (10). Rețeaua își îmbunătățește capacitatea de a face predicții mai precise și de a generaliza mai bine pe diferite exemple prin ajustarea iterativă a greutăților în funcție de erorile calculate.

Deoarece este eficient din punct de vedere computațional, algoritmul de backpropagation poate fi folosit cu succes pentru antrenarea rețelelor neuronale profunde, care au mai multe straturi ascunse. Acest lucru permite rețelelor să învețe despre reprezentări complexe ale datelor, capturând relațiile non-liniare și complexe dintre caracteristicile de intrare și ieșire (10).

* + 1. Antrenarea și testarea rețelelor neuronale

Pentru a obține cele mai bune rezultate în antrenarea rețelelor neuronale, este esențial să se utilizeze o gamă largă de exemple, cuprinzând fiecare aspect relevant al problemei. În cazul problemelor complexe, este necesar un număr mai mare de exemple pentru a acoperi întreaga gamă de situații posibile. Adăugarea unui nivel specific de zgomot sau variație aleatorie în datele de antrenament poate îmbunătăți robusteța și fiabilitatea rețelei, familiarizând-o cu variabilitatea naturală a datelor reale. Datele de antrenament slabe pot conduce la o rețea instabilă și imprevizibilă. De obicei, rețeaua este antrenată pentru un număr prestabilit de epoci sau până când eroarea de ieșire scade sub un anumit prag, garantând astfel că este obținut un nivel acceptabil de performanță (10).

Numărul de neuroni ascunși afectează cât de bine reușește rețeaua să separe datele. Un număr mare de neuroni ascunși va asigura învățarea corectă, iar rețeaua va putea prezice corect datele pe care a fost antrenată, dar performanța sa pe date noi, capacitatea sa de generalizare, va fi compromisă (10). Cu prea puțini neuroni ascunși, rețeaua poate fi incapabilă să învețe relațiile dintre date și eroarea nu va putea scădea sub un nivel acceptabil. Prin urmare, determinarea numărului de neuroni ascunși este o decizie esențială.

Rata de învățare reglează, în mod eficient, dimensiunea pasului luat în spațiul greutăților multidimensionale atunci când fiecare greutate este ajustată (10). În cazul în care rata de învățare aleasă este prea mare, există riscul de a depăși constant minimul local, ceea ce duce la oscilații și convergență lentă către o eroare mai mică. Pe de altă parte, dacă rata de învățare este prea mică, este posibil să fie necesar un număr mare de iterații pentru a ajunge la convergență, ceea ce afectează performanța rețelei (10).

* 1. Rețele neuronale convoluționale

Rețelele neuronale convoluționale (CNN-uri) sunt similare cu rețelele neuronale tradiționale (ANN-uri) datorită faptului că sunt formate din neuroni care se optimizează automat prin învățare. Întreaga rețea va transmite încă o funcție de scor perceptivă (greutatea), pornind de la vectorii imaginii de intrare până la rezultatul final al scorului funcției. Ultimul strat va conține funcții de pierdere asociate claselor, iar toate tehnicile obișnuite dezvoltate pentru ANN-urile tradiționale își păstrează valabilitatea (13).

Singura diferență semnificativă între CNN-uri și ANN-uri tradiționale este că CNN-urile sunt folosite în principal în domeniul recunoașterii de modele în imagini. Acest lucru permite codificarea caracteristicilor specifice imaginii în arhitectură, făcând rețeaua mai potrivită pentru sarcinile axate pe imagine - în timp ce se reduc și mai mult parametrii necesari pentru configurarea modelului (13).

CNN-urile sunt utilizate în principal pentru recunoașterea modelelor în imagini, lucru ce le diferențiază de ANN-urile convenționale. Acest lucru permite codificarea caracteristicilor imaginii în arhitectură specifice, ceea ce îmbunătățește rețeaua pentru sarcinile axate pe imagine. În același timp, se va reduce numărul de parametri necesari pentru configurarea modelului (13).

* + 1. Arhitectura CNN

CNN-urile sunt formate din trei tipuri de straturi diferite. Acestea sunt straturile complet conectate, straturile de convoluție și straturile de pooling. O arhitectură CNN se formează atunci când aceste straturi sunt suprapuse (13).

Funcționalitatea de bază a CNN poate fi împărțită în patru domenii cheie (13):

* 1. Stratul de intrare va stoca valorile pixelilor imaginii, așa cum este cazul în alte forme de ANN.
  2. Stratul de convoluție va determina ieșirea neuronilor care sunt conectați la regiuni locale ale intrării prin calculul produsului scalar între greutățile lor și regiunea conectată la volumul de intrare. Unitatea liniară rectificată (în engleză rectified linear unit, denumită în mod obișnuit ReLu ) își propune să aplice o funcție de activare „elementwise” la ieșirea activării produse de stratul anterior (13).
  3. Stratul de pooling va efectua simplu un eșantionare descrescătoare de-a lungul dimensiunii spațiale a intrării date, reducând în continuare numărul de parametri în acea activare.
  4. Straturile complet conectate vor face aceleași lucruri ca ANN-urile obișnuite și vor încerca să obțină scoruri de clasă din activități pentru a fi folosite pentru clasificare. În plus, se propune utilizarea ReLu între aceste straturi pentru a îmbunătăți performanța (13).

Această metodă simplă de transformare permite CNN-urilor să obțină scoruri de clasă pentru clasificare și regresie. Acest lucru se poate realiza treptat prin utilizarea tehnicilor de convoluție și eșantionare descrescătoare.

Înțelegerea doar a structurii generale a unei rețele CNN este, totuși, insuficientă. Optimizarea și crearea acestor modele pot fi destul de dificile și pot dura mult timp. Așadar, trebuie examinat fiecare strat în profunzime examinând în detaliu hiperparametrii și conectivitățile acestora (13). În plus față de aceste aspecte, este esențial să se aibă o înțelegere profundă a modului în care fiecare strat contribuie la procesul general de învățare al rețelei. Acest lucru permite ajustarea și optimizarea eficientă a parametrilor modelului pentru a obține performanțe maxime în diferite sarcini de învățare automată.

A grid of squares with a grid of circles

Description automatically generated with medium confidence

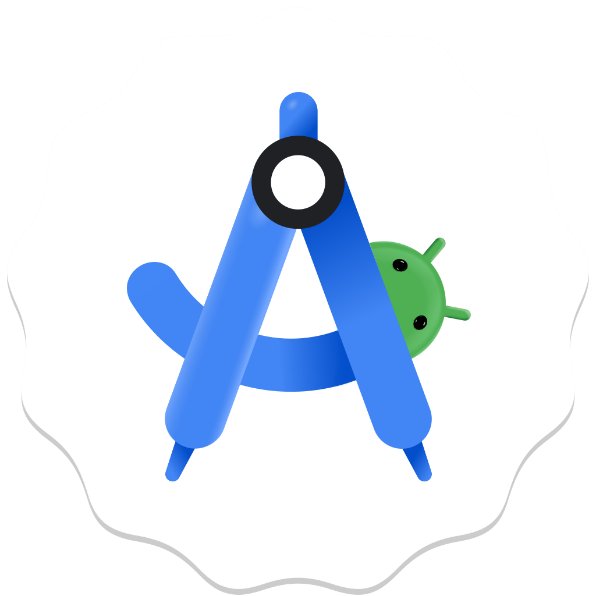
**Figură 2.5.1.** Activări luate de la primul strat convoluțional al unui CNN, după ce a fost antrenat pe setul de date din baza de date cu cifre scrise de mână a MNIST. Se poate observa cu rețeaua a reușit să recunoască caracteristicile unice ale unor cifre (de exemplu, cifrele sașe, șapte și opt) (13).

În jurul anilor 2010, cercetările privind analiza imaginilor cu rețele neuronale au încetinit. Acest lucru poate fi rezultatul unei percepții greșite a nivelului de complexitate și a cunoștințelor care sunt necesare pentru a începe modelarea acestor algoritmi de învățare automată extrem de puternici (13).

Chiar dacă cercetările au încetinit, interesul pentru analiza imaginilor cu rețele neuronale a persistat. Progresele ulterioare au evidențiat potențialul enorm al acestor tehnologii în domeniul recunoașterii și înțelegerii imaginilor. Interesul pentru analiza imaginilor a revenit în comunitatea științifică și industrială ca urmare a progreselor semnificative în arhitecturile rețelelor neuronale convoluționale (CNN), înțelegerea mai profundă a mecanismelor de învățare a adâncimilor și crearea de seturi de date mai complexe.

1. TEHNOLOGII ȘI INSTRUMENTE SOFTWARE UTILIZATE ÎN DEZVOLTAREA PROIECTULUI
   1. Android Studio

Android Studio, care a fost dezvoltat de Google, este un IDE (în engleză integrated development environment) oficial pentru dezvoltarea aplicațiilor Android, bazat pe platforma IntelliJ IDEA a JetBrains (14). Acesta înlocuiește mediul de dezvoltare Eclipse Android Development Tools (E-ADT) pentru aplicațiile native Android și este disponibil pentru instalare pe Windows, macOS și Linux (15).



**Figura 3.1.1.** Logo-ul Android Studio; acesta este același de la versiunea Android Studio Hedgehog | 2023.1.1 (14)

Android Studio a devenit o unealtă esențială pentru dezvoltatorii Android, facilitând procesul de creare a aplicațiilor pentru acest sistem de operare. Acesta permite dezvoltatorilor să scrie și să editeze codul într-un mod simplu și organizat, integrând un editor de cod eficient. În plus, suportul puternic pentru depanare și testare ajută la identificarea și rezolvarea problemelor în aplicații. Integrarea simplă cu platformele de gestionare a versiunilor este o caracteristică semnificativă, deoarece permite dezvoltatorilor să colaboreze în echipă și să monitorizeze modificările de cod.

În data de 7 mai 2019, limbajul de programare Java a fost înlocuit de către Kotlin, devenind astfel limbajul preferat de către cei de la Google pentru dezvoltarea de aplicații Android. În plus, Android Studio suportă limbajele Java și C++ (16).

Android Studio oferă o varietate de caracteristici esențiale pentru dezvoltarea aplicațiilor Android în versiunea stabilă actuală (17). Acestea includ integrare ProGuard și capabilități de semnare a aplicațiilor, suport pentru construirea bazată pe Gradle, refactorizare și remedieri rapide specifice pentru Android și instrumente Lint pentru a detecta probleme de utilizabilitate și performanță.

În plus, există un editor de layout bogat care permite utilizatorilor să tragă și să plaseze (în engleză drag-and-drop) componentele de interfață utilizator și asistenți bazati pe șabloane pentru a crea rapid design-uri și componente Android. O alta funcționalitate a Android Studio este crearea aplicațiilor Android Wear și integrarea nativă cu Google Cloud Platform, cum ar fi Firebase Cloud Messaging și Google App Engine. De asemenea, utilizatorii Android Studio pot folosi un dispozitiv virtual Android (emulator) pentru rularea și testarea aplicațiilor dezvoltate direct în cadrul IDE-ului.

Cu toate acestea, Android Studio are unele dezavantaje care trebuie luate în considerare. Pe computerele cu specificații mai modeste, performanța poate fi scăzută din cauza consumului excesiv de resurse de sistem. Android Studio poate părea la început greu și dificil pentru începători, ceea ce le va lua timp să înțeleagă toate capabilitățile și caracteristicile. În anumite situații, IDE-ul poate întârzia, în special în proiecte mari sau atunci când este folosit un emulator. Actualizările regulate pot aduce modificări care pot provoca probleme de compatibilitate cu proiectele existente sau cu anumite plugin-uri. Configurarea și personalizarea Gradle pot fi, de asemenea, dificile și pot necesita expertiză avansată pentru a optimiza complet procesul de construcție.

* + 1. Fluxul de lucru în Android Studio

Procesul de dezvoltare a aplicațiilor pentru Android este, în esență, identic cu cel pentru alte platforme de dezvoltare a aplicațiilor. Cu toate acestea, pentru a crea o aplicație Android bine concepută, este nevoie de o serie de instrumente specializate. Crearea unui proiect în Android Studio este un proces esențial pentru dezvoltatorii de aplicații Android.

Pentru a începe dezvoltarea unei aplicații Android, primul pas este descărcarea și instalarea Android Studio. Cele mai recente versiuni și actualizări ale programului pot fi descărcate de pe site-ul oficial al Android Studio. După finalizarea descărcării, urmează procesul de instalare. Este important să se urmeze toți pașii indicați de installer pentru a asigura o instalare corectă și completă. Odată instalat, Android Studio necesită configurarea SDK-ului Android. SDK-ul (Software Development Kit) conține toate fișierele și instrumentele necesare pentru dezvoltarea aplicațiilor Android. Fără o configurare corectă a acestuia, dezvoltarea aplicației poate întâmpina dificultăți majore, deoarece SDK-ul oferă acces la toate bibliotecile și resursele necesare pentru programare.

A diagram of a work flow

Description automatically generated

**Figură 3.1.2.** Fluxul de lucru al unui proiect în Android Studio (18).

Următorul pas este crearea unui nou proiect după instalarea și configurarea Android Studio. Acest lucru necesită deschiderea Android Studio și alegerea opțiunii „Începe un nou proiect Android Studio”. Programul va afișa un ghid care ajută la configurarea proiectului, unde se vor specifica detalii precum numele proiectului, locația pe hard disk unde va fi salvat proiectul și altele. În plus, se vor alege șabloanele pentru activități (activități) și layout-uri, cum ar fi „Activitate goală” sau „Activitate de bază”. Android Studio generează automat fișierele de cod care sunt necesare, cum ar fi layout-urile XML și activitățile Java/Kotlin, odată ce șablonul este selectat.

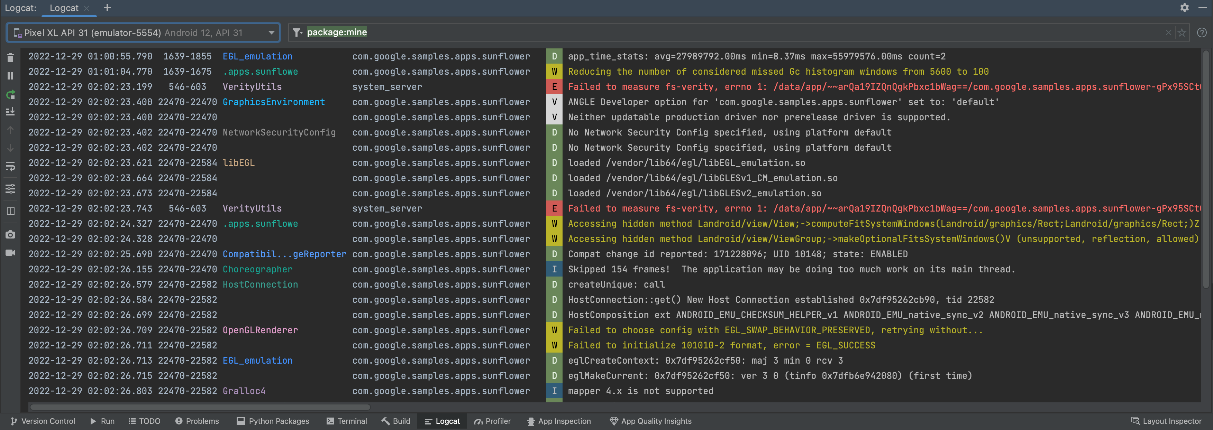
A screenshot of a computer

Description automatically generated

**Figură 3.1.3.** Documentele generate de către Android Studio după selectarea șablonului „Basic”

După configurarea proiectului, se poate începe scrierea codului aplicației. Codul de bază al aplicației va fi scris în fișierele Java sau Kotlin din directorul app/src/main/java. Fișierele XML care conțin interfeța utilizatorului vor fi create în directorul app/src/main/res/layout. Android Studio oferă un editor de cod avansat cu instrumente de depanare, completare automată și refactorizare. Pentru a facilita dezvoltarea și întreținerea pe termen lung, adoptarea practicilor bune de programare, cum ar fi comentariile clare și structura organizată a codului, este esențială. Acestea sunt practici pe care le-am urmat în vederea dezvoltării aplicației.

După scrierea unei părți din cod, este necesar să se construiască și să funcționeze aplicația pentru a evalua rezultatele. Conectarea unui dispozitiv Android real la computer sau utilizarea emulatorului Android Virtual Device (AVD) sunt necesare pentru a realiza acest lucru. Android Studio compilează proiectul și creează un fișier APK care poate fi instalat și testat pe dispozitivul ales. Dezvoltarea aplicației este un proces iterativ. După fiecare rundă de scriere și testare a codului, vor fi efectuate ajustări și optimizări în funcție de feedback și rezultate. Acest ciclu continuu de scriere, testare și îmbunătățire ajută la îmbunătățirea aplicației și la eliminarea greșelilor. Acest proces necesită utilizarea instrumentelor de depanare și optimizare ale Android Studio, cum ar fi Logcat pentru loguri, Android Profiler pentru performanță și Lint pentru detectarea problemelor de cod.

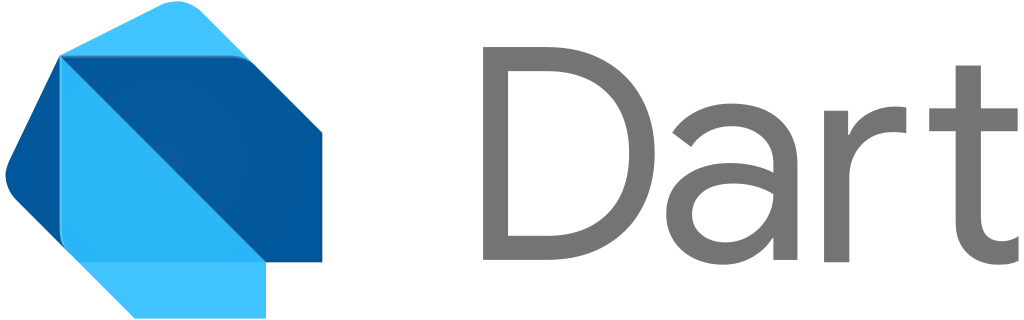


**Figura 3.1.4.** Interfața Logcat (19)

După finalizarea și testarea cu atenție a aplicației, următorul pas este publicarea. Generarea unei versiuni finale a fișierului APK și semnarea acestuia cu un certificat digital sunt toate implicate în acest proces. În continuare, fișierul APK semnat poate fi încărcat pe Google Play Store sau alte platforme de distribuție. Acest proces organizat ajută la gestionarea eficientă a proiectelor Android și garantează dezvoltarea de aplicații de înaltă calitate, optimizate pentru o gamă largă de dispozitive și versiuni de Android.

* 1. Dart

Dart este un limbaj optimizat pentru client care permite crearea de aplicații rapide pe orice platformă. Scopul său este de a oferi cel mai productiv limbaj de programare pentru dezvoltarea multi-platformă, precum și o platformă de execuție flexibilă pentru framework-uri de aplicații (20). Acesta a fost lansat în anul 2011 fiind dezvoltat de către Google și proiectat de către Lars Bark și Kasper Lund. Dart este un limbaj orientat pe obiecte, bazat pe clase, care include un set bogat de caracteristici ce facilitează dezvoltarea rapidă și eficientă a aplicațiilor.



**Figura 3.2.1.** Logo-ul limbajului de programare Dart (20)

Datorită multor caracteristici remarcabile, Dart este o alegere atractivă pentru dezvoltatori. Sintaxa sa familiară este una dintre aceste caracteristici. Datorită sintaxei sale comparabile cu alte limbaje de programare populare precum Java, JavaScript și C#, dezvoltatorii care cunosc deja aceste limbaje pot trece mai ușor la Dart (20). Această familiaritate scurtează curba de învățare și permite dezvoltatorilor să producă mai mult în mai puțin timp.

Flexibilitatea limbajului în compilarea este o altă caracteristică semnificativă. Compilarea Just-in-Time (JIT) și Ahead-of-Time (AOT) sunt ambele acceptate de Dart (20). În timpul dezvoltării, compilarea JIT este utilă pentru că permite depanarea eficientă a codului și încărcarea rapidă a modificărilor. În schimb, compilarea AOT ajută la generarea codului optimizat înainte de rulare, îmbunătățind performanța aplicațiilor în producție.

În plus, Dart suportă tipuri de date statice și dinamice. Aceasta permite dezvoltatorilor să definească tipuri de date statice pentru o mai mare siguranță și performanță, dar oferă și flexibilitate pentru tipurile de date dinamice dacă este necesar. Scrierea unui cod mai robust și mai adaptabil este posibilă datorită acestei flexibilități.

Limbajul Dart oferă siguranță de tip pentru că folosește verificarea statică a tipurilor pentru a se asigura că valoarea unei variabile corespunde întotdeauna tipului static al variabilei. Acest lucru este cunoscut sub numele de „sound type”. Adnotările de tip sunt opționale datorită inferenței de tip, deși tipurile sunt necesare. În plus, flexibilitatea sistemului de tipare Dart permite utilizarea unui tip dinamic împreună cu verificări la rulare. Acest lucru poate fi util în timpul experimentării sau pentru codul care trebuie să fie deosebit de dinamic (20).

Conceptul de „null safety” este incorporat în Dart. Acest lucru indică faptul că valorile nu pot fi nule decât în cazul în care este specificat acest lucru. Prin utilizarea analizei statice a codului, Dart poate asigura că nu există excepții de nulitate la rulare. În cazul în care Dart constată că o variabilă nu poate fi nulă, acea variabilă nu va fi niciodată nulă, spre deosebire de o mulțime de alte limbi care sunt *null-safe*. Se poate observa că non-nulitatea este păstrată la rulare dacă este examinat codul în execuție în debugger (20).

Dart oferă o gamă largă de biblioteci de bază care oferă componente esențiale pentru numeroase sarcini de programare de zi cu zi (20):

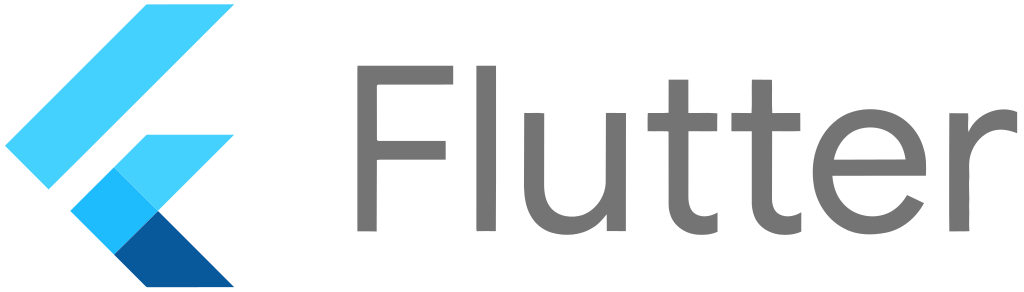
* Fiecare program Dart are tipuri integrate, colecții și alte caracteristici fundamentale.
* Cozi, liste înlănțuite, hashmaps și arbori binari sunt exemple de tipuri de colecții mai bogate.
* Utilizează encoding și decoding pentru a converti diferite reprezentări de date, cum ar fi JSON și UTF-8.
* Generarea numerelor aleatorii, precum și constantele și funcțiile matematice (dart: math)
* Include clase precum Future și Stream și permite programarea asincronă (dart:async)
* Lista care gestionează bine tipurile numerice SIMD și datele de dimensiuni fixe, cum ar fi întregi nesemnate pe 8 biți (dart:typed\_data).
* Suport pentru aplicații non-web, cum ar fi fișiere, socket-uri, HTTP și alte I/O (dart:io).
* Interfețe pentru funcții externe, care pot fi utilizate pentru a interacționa cu alte coduri care prezintă o interfață de tip C (dart:ffi).
* Programare concurentă folosind izolate: lucrători independenți care comunică doar prin mesaje și nu partajează memoria (dart:isolate).
* Elemente HTML și alte componente pentru aplicațiile web care interacționează cu browserul și Document Object Model (DOM) (dart:html).

Multe API-uri sunt furnizate într-un set larg de pachete, în afară de bibliotecile de bază. Echipa Dart distribuie numeroase pachete suplimentare utile, cum ar fi „crypto”, „markdown”, „intl” și multe altele (20).

Dart beneficiază de un ecosistem robust și de o gamă variată de instrumente de dezvoltare.

* Dart SDK: SDK-ul Dart include un compilator, un runtime, și un set de instrumente de dezvoltare și debugging, care facilitează întregul proces de dezvoltare.
* Pub: Pub este managerul de pachete pentru Dart, similar cu npm pentru JavaScript. Pub permite gestionarea dependențelor, publicarea și distribuirea pachetelor.
* IDE-uri și Editoruri: Dart este suportat de numeroase IDE-uri și editori de text, cum ar fi IntelliJ IDEA, Visual Studio Code și Android Studio, care oferă suport pentru completare automată, debugging și refactorizare.
  1. Flutter

Flutter, o platformă (în engleză framework) open-source creat de Google, permite crearea de aplicații mobile, web și desktop folosind o singură bază de cod. Flutter, care a fost lansat pentru prima dată în 2017, a câștigat rapid popularitate printre dezvoltatori datorită capacității sale de a crea interfețe de utilizator atractive și eficiente, care funcționează nativ pe mai multe platforme. Flutter oferă o soluție completă pentru dezvoltarea de aplicații multi-platformă, datorită limbajului de programare Dart (21).



**Figura 3.3.1.** Logo-ul Flutter

O aplicație Flutter compilată include atât codul de interfață cu utilizatorul, cât și motorul de redare, care măsoară aproximativ patru MB comprimați (21). Acest lucru contrastează cu multe alte structuri de interfață a utilizatorului care includ doar codul interfaței și se bazează pe un motor de redare separat. De exemplu, aplicațiile Android native care se bazează pe SDK-ul Android la nivel de dispozitiv sau aplicațiile web HTML și JavaScript care se bazează pe motorul HTML și JavaScript al utilizatorului. Deoarece pipeline-ul său de redare este controlat în întregime de Flutter, suportul pentru mai multe platforme este simplificat deoarece necesită doar ca platforma să suporte rularea codului nativ, de exemplu, prin interfața nativă Java a Android.

Flutter este un instrument puternic și flexibil pentru dezvoltare datorită numeroaselor sale caracteristici. În primul rând, Flutter permite crearea de interfețe de utilizator nativ puternice care se comportă ca aplicațiile native, asigurând performanțe superioare pe toate platformele.

În plus, Flutter permite dezvoltatorilor să economisească timp și resurse scriind un singur set de cod care funcționează pe Android, iOS, web și desktop.

De asemenea, funcționalitatea de repornire rapidă (în engleză hot reload) a Flutter permite dezvoltatorilor să monitorizeze modificările aduse codului fără a pierde starea aplicației, ceea ce accelerează ciclul de dezvoltare.

Un alt aspect al limbajului este că Flutter oferă o gamă largă de widget-uri personalizabile care se conformează standardelor de design Material pentru Android și pentru iOS, ceea ce facilitează crearea de interfețe complexe și atractive (21). Un „widget” este componenta de bază a unui program Flutter, care la rândul său poate include alte „widget-uri”. Un widget gestionează logica, interacțiunea și designul elementului de interfață cu utilizatorul, folosind o abordare asemănătoare cu React. Flutter creează widget-uri în sine pe baza fiecărui pixel, spre deosebire de alte kituri cross-platform precum Xamarin și React Native, care folosesc componentele native ale platformei pentru a le crea.

Există două tipuri de widget-uri: fără stare și cu stare (în engleză stateless și stateful). Widget-urile fără stare nu se actualizează decât dacă intrările lor se schimbă, ceea ce înseamnă că nu vor trebui reconstruite atunci când alte elemente ale ecranului se schimbă. Pe de altă parte, widget-urile cu stare au capacitatea de a utiliza metoda *setState()* pentru a actualiza și redesena starea internă (21).

Cu toate că widget-urile sunt principala metodă de construire a aplicațiilor Flutter, desenarea directă pe un canvas poate fi, de asemenea, preferată. Această funcție a fost folosită pentru implementarea motoarelor de joc în Flutter.

* + 1. Dezvoltarea programelor cu ajutorul Flutter

Pentru a începe dezvoltarea cu Flutter, este necesar să se descarce și să se instaleze SDK-ul Flutter de pe site-ul oficial. Această instalație include și configurarea mediului de dezvoltare, precum și instalarea pluginurilor necesare pentru editorul preferat (Visual Studio Code, IntelliJ sau Android Studio, procedeu descris în capitolul anterior).

Metoda de a începe un nou proiect în Flutter este simplă și rapidă. Se poate crea un proiect nou cu comanda „flutter create” din terminal sau cu opțiunile IDE-ului preferat. Fișierele de configurare de bază și structura de directoare necesare vor fi create automat prin această acțiune.

A screenshot of a computer

Description automatically generated

**Figura 3.3.2.** Pentru crearea unui nou proiect de tip Flutter în Android Studio, primul pas este localizarea fișierului ce găzduiește SDK-ul Flutter

A screenshot of a computer

Description automatically generated

**Figura 3.3.3.** Câmpurile ce trebuie completate pentru finalizarea procesului de crearea a proiectului

În timpul procesului de creare a unui nou proiect în Flutter, există o serie de câmpuri disponibile pentru a configura setările proiectului în funcție de cerințele specifice ale proiectului. În primul rând, se solicită furnizarea unui descriptor unic pentru numele proiectului. Acest nume este esențial pentru gestionarea și organizarea proiectelor în dezvoltare și trebuie să indeplinească anumite criterii. Numele nu poate conține majuscule și nici spații, astfel trebuie sa fie de tipul „nume\_proiect”.

Următorul pas este specificarea locației unde fișierele proiectului vor fi salvate. Acest lucru oferă controlul asupra structurii de stocare a datelor proiectului. Există posibilitatea de a adăuga o descriere a proiectului, care oferă detalii suplimentare care să ofere o înțelegere mai clară a obiectivelor sau caracteristicilor proiectului. Această opțiune este disponibilă ca opțiune opțională.

În secțiunea destinată tipului proiectului sunt disponibile diferite opțiuni, cum ar fi aplicație, modul sau pachet, în funcție de natura proiectului ce urmează să fie creat. Această selecție are un impact asupra structurii și organizării proiectului.

De asemenea, trebuie specificate limbajele de programare preferate pentru platformele Android și iOS, pentru a se asigura că aplicația este compatibilă și funcțională pe diferite platforme. Opțiunile disponibile pentru Android sunt Kotlin și Java, iar pentru iOS sunt Swift și Objective-C.

În cele din urmă, există opțiunea de a selecta platformele disponibile pentru proiect, cum ar fi Android, iOS și Web. În cadrul proiectului meu am ales toate opțiunile disponibile pentru a asigura că un număr cât mai mare de utilizatori au acces la aplicație.

După finalizarea procesului, orice proiect nou de tip Flutter conține următoarea structură de fișiere, împreuna cu un fișier *README.md*. Acesta oferă câteva surse de inspirație pentru proiectul dorit.

A screenshot of a computer program

Description automatically generated

**Figura 3.3.4.** Fișierul README generat

A screenshot of a computer

Description automatically generated

**Figura 3.3.5.** Documentele generate în cadrul proiectului nou

După crearea unui proiect Flutter, în directorul proiectului sunt generate mai multe fișiere și directoare esențiale pentru dezvoltarea aplicației. Unul dintre aceste directoare este cunoscut sub numele de „lib”, care este responsabil pentru stocarea fișierelor sursă ale aplicației. Aici se află fișierele Dart care conțin codul aplicației, împreună cu logica aplicației, definirea widget-urilor și alte funcționalități.

În afara directorului „lib” există un fișier „pubspec.yaml”. Gestionarea dependențelor proiectului și specificarea detaliilor aplicației, cum ar fi numele, versiunea și descrierea, se întâmplă în cadrul acestui fișier. În plus, se pot adăuga aici pachete externe care sunt necesare proiectului, cum ar fi pachete de interacțiune cu servicii web sau de gestionare a stării aplicației. În cadrul aplicației mele, am adăugat în interiorul acestui document pachetele pentru accesarea camerei dispozitivului și a modelului de învățare automată.

După crearea proiectului, va fi generat un fișier numit „main.dart”, care servește drept punct de intrare în aplicație. Codul principal al aplicației este inclus în acest fișier și este responsabil pentru inițierea și funcționarea aplicației Flutter.

În plus, alte directoare și fișiere sunt aflate în directorul proiectului, cum ar fi „android” și „ios”, care includ fișiere de configurare și resurse specifice platformei pentru a construi și rula o aplicație pe dispozitivele Android și iOS.

De asemenea, sunt create directoare suplimentare esențiale, cum ar fi „test”, care include teste unitare și de integrare pentru a se asigura că codul este de calitate, și „assets”, care pot fi utilizate pentru a stoca resurse statice, cum ar fi imagini, fonturi sau fișiere de date. Pot fi create și alte fișiere sau directoare în funcție de configurația proiectului. Pentru a lucra eficient la dezvoltarea aplicațiilor Flutter, este esențială înțelegerea structurii și conținutului acestora.

**A screenshot of a computer program

Description automatically generated**

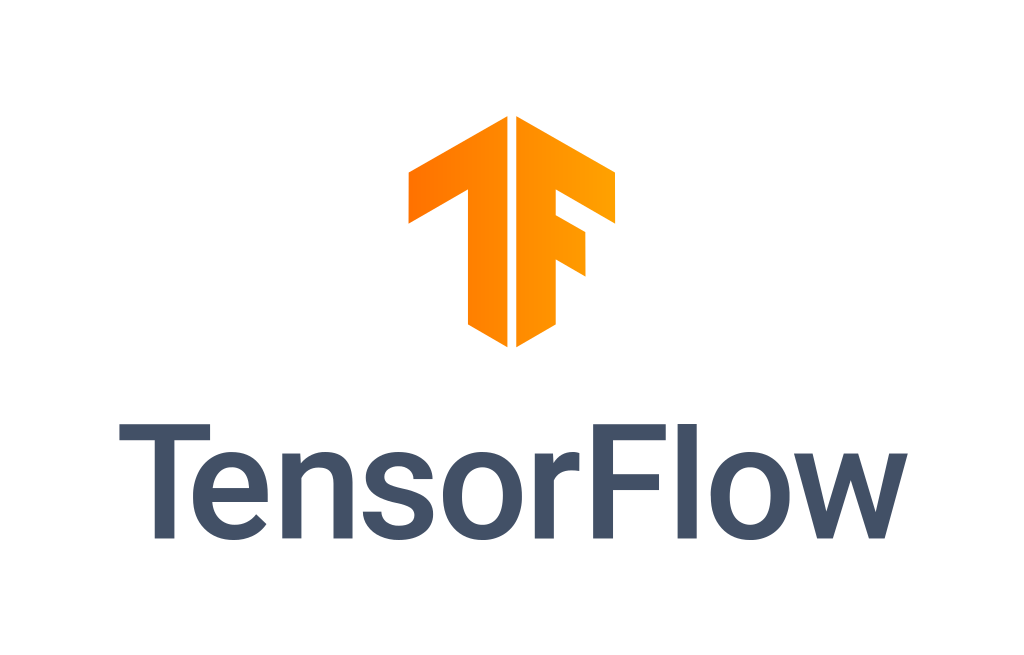
**Figura 3.3.6**. Clasa de bază generată de către Flutter pentru a da un punct de start dezvoltatorilor.

* 1. TensorFlow

TensorFlow este o bibliotecă de software pentru învățarea automată și inteligența artificială gratuită și open-source. Este capabil să îndeplinească o gamă largă de sarcini, dar se concentrează în special pe antrenarea și inferența rețelelor neuronale adânci (23).

A fost creat de echipa Google Brain pentru utilizarea în cercetare și producție internă a companiei. Prima versiune a fost publicată în 2015 cu licența Apache License 2.0 (23). În septembrie 2019, Google a lansat TensorFlow 2.0, o actualizare.

TensorFlow funcționează cu o gamă largă de limbaje de programare, cum ar fi Python, JavaScript, C++ și Java, ceea ce îl face ușor de utilizat într-un număr mare de sectoare (22).



**Figura 3.4.1.** Logo-ul TensorFlow (22).

TensorFlow este un instrument esențial pentru dezvoltarea, antrenarea și implementarea modelelor de învățare automată și inteligență artificială. Pentru lucrul cu date, definirea arhitecturilor modelelor și antrenarea modelelor pe seturi de date mari și complexe, aceasta oferă instrumente puternice.

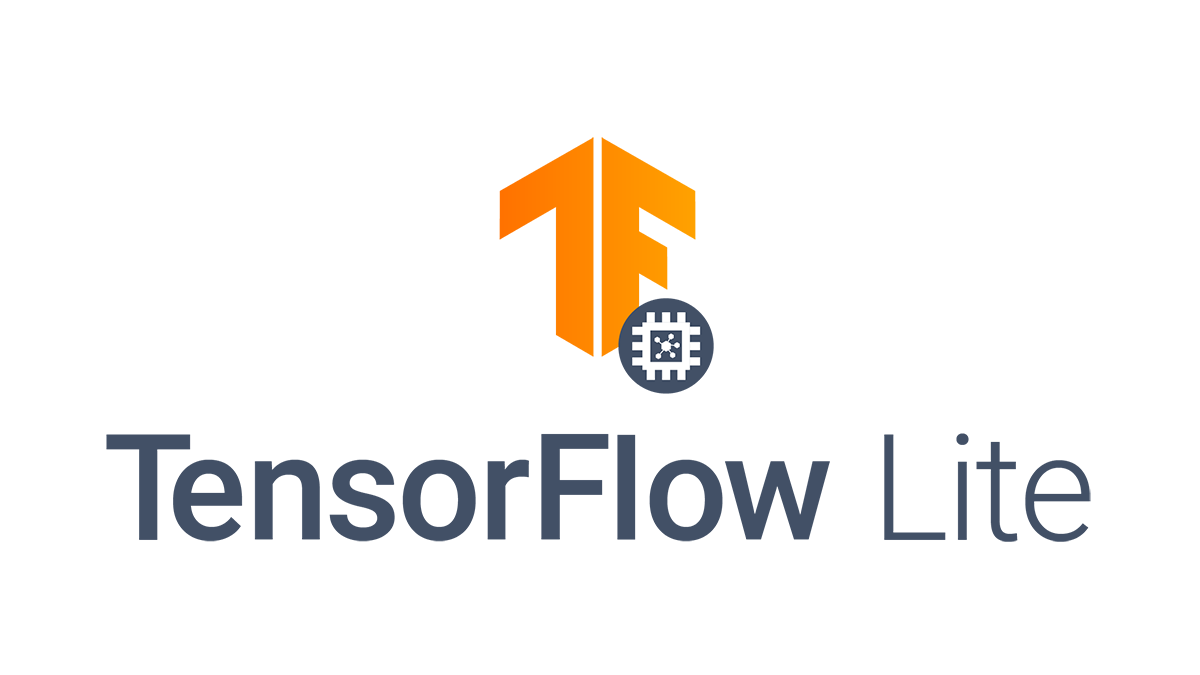
TensorFlow poate fi folosit pentru o mulțime de domenii, cum ar fi procesarea limbajului natural, traducerea automată, clasificarea și recunoașterea imaginilor și analiza datelor. Este folosit în industrie, cercetare și aplicații practice pentru a rezolva probleme complexe care necesită inteligență artificială.

TensorFlow este o alegere populară pentru dezvoltatorii de aplicații de învățare automată datorită multitudinii de caracteristici pe care le oferă. Prima caracteristică ar fi flexibilitatea, unde se poate observa cum TensorFlow oferă dezvoltatorilor flexibilitate prin utilizarea sa într-o gamă largă de medii de dezvoltare și limbaje de programare.

O altă caracteristică este scalabilitatea, deoarece această platformă este optimizată pentru a lucra cu seturi de date mari. Iar, nu în ultimul rând, dezvoltatorii iau în considerare eficiența pe care TensorFlow o oferă. Acesta accelerează antrenarea și evaluarea modelelor prin optimizarea automată a utilizării resurselor hardware, cum ar fi procesorul grafic (GPU) și unitatea de procesare tensorială (TPU).

* + 1. TensorFlow Lite

TensorFlow Lite este o versiune mai simplă a TensorFlow care este concepută să funcționeze pe dispozitive mobile și să fie încorporate. Acesta permite implementarea modelelor de învățare automată pe dispozitive care au resurse limitate, cum ar fi telefoanele mobile, tabletele și dispozitivele IoT (Internet of Things).



**Figura 3.4.2.** Logo-ul TensorFlow Lite

În mai 2017, Google a anunțat o suită de software specială pentru dezvoltarea mobilă, TensorFlow Lite (24). În ianuarie 2019, echipa TensorFlow a lansat o previzualizare pentru dezvoltatori a motorului de inferență GPU mobil cu OpenGL ES 3.1 Compute Shaders pe dispozitive Android și Metal Compute Shaders pe dispozitivele iOS.

Unul dintre primele framework-uri open-source care au implementat învățarea automată pe microcontrolere a fost uTensor. Pentru a face integrarea cu orice proiect încorporat, uTensor transforma modelele de învățare automată în fișiere sursă C++ lizibile și autoconținute. Era optimizat pentru dispozitivele încorporate cu putere redusă și restricționate, având o bază solidă în TensorFlow și MbedOS. În mai 2019, Google a anunțat că TensorFlow Lite Micro (cunoscut și sub numele de TensorFlow Lite pentru microcontrolere) și uTensor de la ARM (Arm Holdings plc, o firmă din Regatul Unit la Marii Britanii și al Irlandei de Nord care se ocupă de fabricarea de unități centrale de procesare) vor fuziona (25).

TensorFlow Lite include o serie de instrumente care ajută la optimizarea și conversia modelelor TensorFlow pentru utilizarea pe dispozitive mobile. Acest lucru permite dezvoltatorilor să integreze funcționalitățile de învățare automată în aplicațiile lor mobile fără a avea nevoie de o conexiune la internet și fără a depinde de servere externe pentru inferența modelului.

Printre caracteristicile TensorFlow Lite se numără eficiența, performanța și integrarea ușoară. Astfel, o performanță înaltă poate fi atinsă pentru inferența modelelor, cu timpi de răspuns rapizi și consum redus de energie, luând în calcul că TensorFlow Lite optimizează modelele de învățare automată pentru a rula eficient pe dispozitivele mobile, folosind resursele hardware disponibile.

De asemenea, acesta facilitează integrarea modelelor de învățare automată în aplicațiile mobile existente, permițând dezvoltatorilor să creeze aplicații mai inteligente și mai interactive. TensorFlow Lite include o serie de biblioteci și instrumente care ajută la conversia și optimizarea modelelor TensorFlow pentru implementarea pe dispozitive mobile. Aceste instrumente sunt utile pentru dezvoltatori pentru a transforma modelele existente într-un format care este potrivit pentru dispozitivele mobile.

Astfel, procesul de a antrena și a implementa modelul de învățare automata folosit în aplicație nu a fost unul complicat, însă a necesitat multă atenție din partea mea.

* + 1. Detectarea obiectelor cu TensorFlow

Detectarea obiectelor reprezintă o sarcină complexă în domeniul vizual al învățării automate, iar TensorFlow oferă instrumente pentru a aborda această provocare. Printr-o imagine sau flux video, un model de detectare a obiectelor poate identifica care dintre un set cunoscut de obiecte ar putea fi prezente și poate oferi informații despre pozițiile lor în cadrul imaginii.

Un model de detectare a obiectelor poate fi antrenat pentru a identifica prezența și poziția mai multor clase de obiecte. De exemplu, un model ar putea fi antrenat cu imagini cu diferite fructe, fiecare fiind etichetată cu clasa fructului (de exemplu o banană sau un măr) și date care arată unde apare fiecare obiect în imagine .

Atunci când este furnizată o imagine modelului, acesta va genera o listă a obiectelor pe care le-a detectat, locația unei cutii încadrate care conține fiecare obiect și un scor care indică cât de sigur este că detectarea a fost corectă (26).

O imagine este furnizată modelului de detectare a obiectelor, iar formatarea acesteia trebuie să fie conformă cu cerințele modelului. De exemplu, există un scenariu în care o imagine de intrare are 300 x 300 pixeli și are trei canale de culoare: roșu, verde și albastru. Imaginea trebuie transformată într-un buffer aplatizat, adică o serie de valori liniare, pentru a alimenta modelul. Deoarece fiecare pixel are trei canale de culoare, avem 270.000 de valori de pixel pentru imaginea noastră de 300 x 300 pixeli (26). În cazul în care modelul este cuantificat, fiecare valoare din buffer trebuie reprezentată printr-un singur byte, cu o valoare care variază de la 0 până la 255.

Astfel, modelul produce patru vectori, indexați de la 0 la 3. Vectorul cu indexul 3 conține variabila întreagă care indică numarul total de detectări (N). Vectorul cu index 0 conține locații, acesta fiind un tablou multidimensional care conține [N][4] valori zecimale care variază de la 0 până la 1. Tablourile interne conțin cutii încadrate în forma [sus, stânga, jos și dreapta] (26).

Vectorul 1 conține clasele obiectelor detectate. Fiecare indică indexul unei etichete de clasă din fișierul de etichete și este reprezentat printr-un șir de N numere întregi, care au rezultat ca valori zecimale. Iar vectorul cu index 2 corespunde scorurilor, acesta un vector de N de valori zecimale care variază de la 0 până la 1 indică probabilitatea că a fost găsită o clasă.

Numărul de rezultate este un parametru setat în timpul exportului modelului de detectare către TensorFlow Lite.

A table with numbers and a number of numbers

Description automatically generated with medium confidence

**Tabelul 3.4.1.** Exemplu de rezultate al unui model antrenat pe câteva fructe

Pentru a interpreta aceste rezultate, se pot examina scorurile și locațiile fiecărui obiect care a fost găsit. Scorul constă într-un număr între 0 și 1 și indică încrederea că obiectul a fost detectat corect. Cu cât numărul este mai aproape de 1, cu atât este mai probabil ca modelul să fi detectat obiectul corect.

Se poate alege un prag de tăiere sub care se vor elimina rezultatele de detectare, în funcție de aplicație. În cadrul aplicației mele, am decis să aplic un prag de 45%, ceea ce ar indica o probabilitate de 45% că detectarea este validă. În exemplul prezent în tabelul 3.4.1., ultimele două fructe vor fi ignorate deoarece au scorurile sub 0,45. Pragul trebuie ales în funcție de nivelul de confort în ceea ce privește rezultatele fals pozitive (elemente identificate greșit sau zone ale imaginii marcate greșit ca obiecte) sau rezultatele fals negative (elemente reale care nu pot fi identificate din cauza lipsei de încredere).

În figura 3.4.3, se observă cum creionul, fiind un obiect pe care nu l-am inclus în antrenamentul modelului de învățare automată, a fost identificat greșit drept un „cuțit”. Acesta este un exemplu clar de rezultat fals pozitiv, caz care ar fi putut evitat dacă aș fi folosit un prag de 60%.

Această situație subliniază importanța alegerii unui prag adecvat în procesul de identificare a obiectelor. Pragul de 60% menționat anterior ar fi putut reduce semnificativ numărul de rezultate fals pozitive, îmbunătățind astfel precizia modelului. Cu toate acestea, este important de menționat că alegerea unui prag mai mare ar putea duce la creșterea numărului de rezultate fals negative, ceea ce ar însemna că obiecte reale ar putea fi omise.

Prin urmare, pentru a obține rezultate optime, este esențial să se echilibreze aceste două aspecte. Acest exemplu subliniază, de asemenea, importanța unei varietăți mai mari de obiecte în setul de date de antrenament pentru a îmbunătăți capacitatea modelului de a identifica obiectele în mod corespunzător.

A pencil with a green border

Description automatically generated

**Figura 3.4.3.** Algoritmul recunoaște greșit un cuțit în loc de creion

Există o multitudine de factori și indicatori care trebuie luați în considerare atunci când se evaluează eficiența unui prag specific într-un model de învățare automată. Unul dintre acești indicatori este rata de eroare, care arată procentul de predicții incorecte ale modelului, care includ rezultate fals pozitive și fals negative, din totalul predicțiilor. Un prag eficient ar trebui să aibă capacitatea de a reduce această rată de eroare.

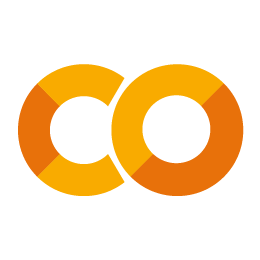
Pentru a determina locația obiectului, modelul va returna un vector format din patru numere reprezentând un dreptunghi delimitator care înconjoară poziția fiecărui obiect identificat. Numerele pentru modelul de pornire sunt ordonate astfel: [sus, stânga, jos, dreapta]. Valoarea „sus” indică distanța în pixeli dintre marginea superioară a dreptunghiului și partea de sus a imaginii. Valoarea „stânga” indică distanța dintre marginea stângă a dreptunghiului și partea stângă a imaginii de intrare. De asemenea, marginile de jos și dreapta sunt reprezentate într-un mod similar.

Modelele de detectare a obiectelor acceptă imagini de dimensiuni specifice. Astfel, am scris cod pentru a decupa și redimensiona imaginea brută pentru a se potrivi cu dimensiunea de intrare a modelului, deoarece dimensiunea acesteia este probabil diferită de dimensiunea imaginii brute capturate de camera dispozitivului.

Valorile pixelilor emise de model indică poziția în imaginea decupată și redimensionată, așa că pentru a le interpreta corect, acestea trebuie scalate pentru a se potrivi cu imaginea brută.

* 1. Google Colab

Google Colab, cunoscut și sub numele de „Colaboratory”, este un serviciu de calcul în cloud dezvoltat de Google în 2017. Scopul său a fost de a facilita cercetarea în calculul de date și inteligența artificială, oferind un mediu de lucru asemănător cu Jupyter Notebook, dar cu acces la resurse de calcul mai puternice, cum ar fi GPU-uri (27) (28).



**Figura 3.5.1.** Logo-ul Google Colab (29)

Programul este foarte asemănător cu Google Docs, instrumentul său de colaborare la nivel de documente, dar care nu poate rula cod și nu poate afișa rezultatele codului în documente. Colaboratory este gratuit și derivat din Jupyter, un proiect open-source popular în știința datelor. Google Colab se diferențiază semnificativ de Jupyter Notebook prin faptul că este bazat pe cloud, în timp ce Jupyter nu este. Acest lucru înseamnă că utilizatorul nu trebuie să se preocupe de descărcarea și instalarea produselor pe hardware-ul propriu dacă folosește Google Colab (27).

Utilizatorii pot crea notebook-uri sau documente care pot fi editate în același timp, precum Google Docs. Este compatibil cu Python 2.7 și trebuie utilizat cu Google Chrome. Software-ul este, de asemenea, integrat cu Google Drive, astfel încât oamenii pot partaja proiecte sau copia proiectele pe care alții le-au partajat pe propriile conturi (28).

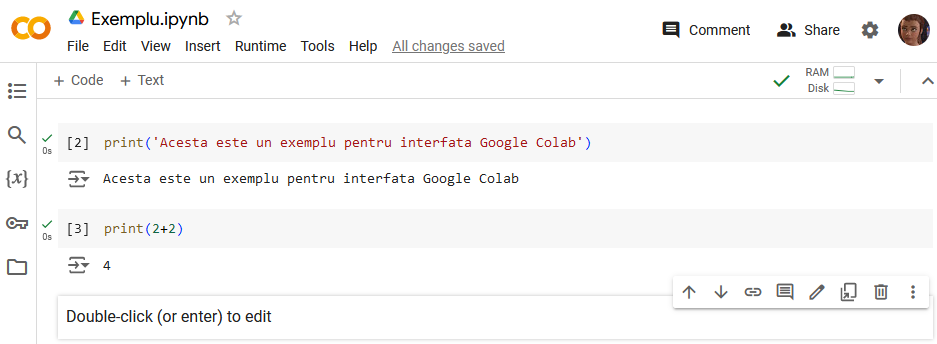
Interfața simplă și accesibilă face navigarea extrem de ușoară pentru utilizatori, indiferent de nivelul de experiență. Aceasta garantează o curbă de învățare rapidă, considerând cum etichetele și organizarea logică a elementelor de meniu și a funcțiilor permit accesul rapid la diferite funcționalități.

Pentru prelucrearea proiectelor complexe de învățare automată și știința datelor, accesul gratuit la GPU și TPU este un avantaj semnificativ. Utilizarea mai rapidă a algoritmilor și scurtarea timpului necesar pentru antrenarea modelelor sunt posibile datorită acestor resurse hardware puternice. Abilitatea de a importa proiecte care sunt compatibile cu învățarea automată și știința datelor din alte surse este foarte benefică pentru colaborarea și reutilizarea codului existent. Acest lucru face mai ușor integrarea cu alte platforme.

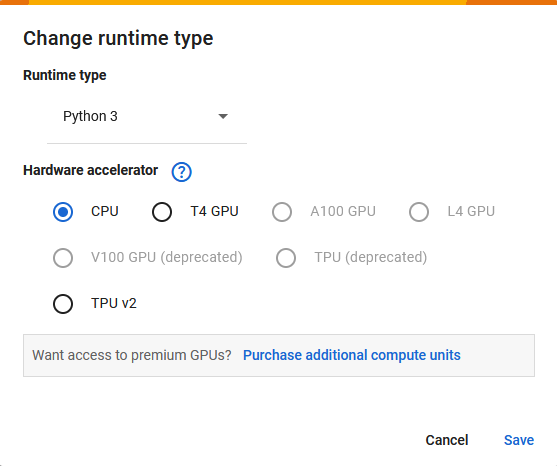
Controlul versiunilor automat, asemănător cu cel din Google Docs, garantează că toate modificările sunt salvate și monitorizate în timp real. Aceasta garantează că nicio informație nu este pierdută și, dacă este necesar, este posibil să se întoarcă la versiunea anterioară a proiectului. Este un instrument esențial pentru gestionarea proiectelor complexe în care participanții lucrează împreună. Compatibilitatea platformei este îmbunătățită prin integrarea cu diferite alte instrumente, cum ar fi GitHub, Jupyter Notebook, BLACKBOX AI, Codeium, CodeSquire, Google Workspace, Neptune.ai, StrongDM și Google Drive, printre altele (27). Aceasta asigură un ecosistem organizat în care utilizatorii pot accesa și utiliza o varietate de resurse și servicii. Acest lucru îmbunătățește fluxul de lucru și eficiența operațională.

În cadrul acestei lucrări, Google Colab a fost folosit pentru a antrena modelul de învățare automată folosit în cadrul aplicației de identificat obiecte. Am ales Google Colab din cauza nevoii de un GPU mai performant decât cel disponibil pe sistemul local.

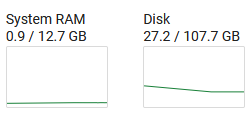
Codul Python pentru antrenarea modelului a fost încărcat într-un notebook Google Colab și a fost executat într-un mediu de execuție cu procesoare grafice pregătite pentru învățare automată. Acest lucru a permis modelului să se antreneze mult mai rapid decât ar fi fost posibil pe sistemul local.



**Figura 3.5.2.** Interfața Google Colab, unde se poate vedea cum fiecare linie de cod este rulată separat



**Figura 3.5.3.** Diferitele tipuri de acceleratoare de hardware disponibile

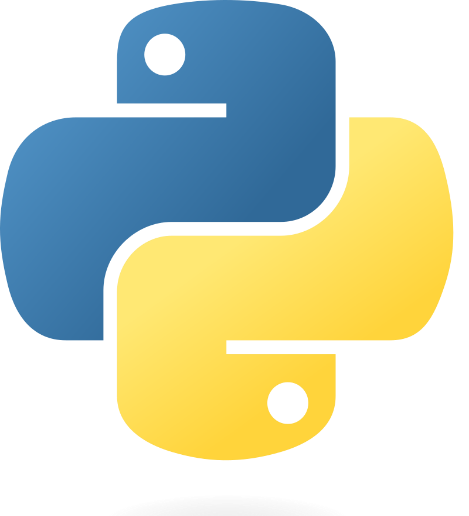


**Figura 3.5.4.** Monitorizarea resurselor în timp real oferită de către Google Colab

* 1. Python

Python este un limbaj de programare de nivel înalt, interpretat și dinamic, renumit pentru sintaxa sa simplă și ușor de învățat. Python a fost dezvoltat de Guido van Rossum în 1991 și a fost conceput pentru a se concentra pe dezvoltarea rapidă a aplicațiilor și pe lizibilitatea codului. Python este compatibil cu mai multe paradigme de programare, inclusiv programarea funcțională, orientată pe obiecte și procedurală (30).

La sfârșitul anilor 1980, Guido van Rossum a început să lucreze la Python ca succesor al limbajului de programare ABC. În 1991, a lansat prima versiune, Python 0.9.0. Python 2.0 a fost lansat în 2000, iar Python 3.0 a fost lansat în 2008, atunci când a fost făcută o revizuire semnificativă pentru a asigura compatibilitatea completă cu versiunile anterioare. Python 2.7.18, care a fost lansat în 2020, a fost versiunea finală a seriei Python 2 (30). Python este unul dintre cele mai populare limbaje de programare și este foarte popular în comunitatea de învățare automată.



**Figura 3.6.1.** Logo-ul Python (Python, fără an)

Sintaxa simplă și accesibilă a Python este binecunoscută de către utilizator, deoarece aceasta face citirea și scrierea codului mai ușoară. Acest lucru asigură că dezvoltatorii cu orice nivel de experiență pot înțelege și modifica cu ușurință codul scris în Python. În schimb, dezvoltatorii experimentați apreciază Python pentru capacitatea sa de a permite scrierea de cod curat și eficient, care este ușor de întreținut și de extins.

Python are o varietate de biblioteci și framework-uri care acoperă aproape orice domeniu de aplicare. Framework-uri populare pentru dezvoltarea web precum Django și Flask oferă instrumente flexibile pentru construirea aplicațiilor web. Biblioteci precum Pandas și NumPy sunt instrumente utile în domeniul științei datelor care permit analiza și manipularea eficientă a datelor. În plus, Python excelează în învățarea automată, împreună cu biblioteci precum TensorFlow și scikit-learn, care facilitează crearea și antrenarea modelelor de învățare automată. Python este un limbaj extrem de versatil care poate fi folosit pentru o varietate de aplicații datorită acestor biblioteci și framework-uri.

Tipizarea dinamică și o combinație de numărare a referințelor și un colector de detectare a ciclurilor sunt utilizate în Python pentru gestionarea memoriei. Rezolvarea dinamică a numelor, cunoscută și sub numele de legare întârziată (în engleză late binding), leagă numele metodelor și variabilelor în timpul execuției programului.

Designul său permite programarea funcțională în corespondență cu tradiția Lisp. Python are funcții de filtrare, mapare și reducere (în engleză filter, map și reduce), precum și liste, dicționare, seturi și expresii generatoare. Biblioteca standard are două module, "itertools" și "functools", care funcționează pentru a implementa instrumente funcționale împrumutate din programarea automată standard și Haskell (30).

Structura simplă și ușor de înțeles a limbajului Python mi-a permis să mă concentrez pe elementele esențiale ale antrenării modelului meu de învățare automată fără a mă pierde în detaliile tehnice complexe. Astfel, aceste caracteristici m-au ajutat să mă concentrez pe optimizarea performanței și perfecționarea algoritmilor, obținând rezultate cu o precizie ridicată în aplicația mea de învățare automată.

1. Studiu de caz: „Aplicație Android pentru identificarea obiectelor în timp real folosind machine learning”
   1. Proiectarea sistemului

Analiza sistemului informatic este o etapă crucială în crearea și optimizarea aplicației mele Android pentru identificarea obiectelor în timp real folosind învățarea automată. Am folosit diverse metode de analiză pentru a asigura o înțelegere completă a sistemului și a cerințelor asociate.

În primul rând, procesul a început cu colectarea informațiilor despre sistem. Aceste informații includ specificațiile și cerințele necesare, precum și cerințele funcționale și non-funcționale. Astfel, faza aceasta de început mi-a oferit o înțelegere generală a sistemului și a funcțiilor sale. Scopul principal al aplicației este de a permite utilizatorilor să identifice obiecte în timp real cu camera dispozitivului Android, acest lucru fiind funcționalitatea centrală a aplicației.

Performanța și ușurința în utilizare au fost exemple de cerințe non-funcționale pe care a trebuit să le implementez. Am decis că aplicația trebuie să proceseze imagini în timp real, iar în același timp să ofere utilizatorului o viziune clară asupra obiectului identificat împreună cu detaliile aferente.

Am creat o structură modulară cu responsabilități clare pentru fiecare parte a aplicației. De exemplu, am separat modulul de procesare a imaginii și de afișare a rezultatelor de modulul de capturare a imaginii. Această metodă a simplificat dezvoltarea și testarea, oferind, de asemenea, mai multă flexibilitate atunci când aplicația va fi îmbunătățită sau actualizată în viitor.

* 1. Cerințe funcționale

O aplicație trebuie să aibă anumite funcționalități și comportamente pentru a satisface cerințele utilizatorului, acestea fiind cunoscute drept cerințe funcționale. Astfel, mi-am propus ca aplicația să îndeplinească un număr de caracteristici cheie, precum:

* Detectarea obiectelor în timp real: Aceasta funcționalitate permite utilizatorului să se foloseasca de aplicatie pentru a detecta obiectele din mediul inconjurator în timp real. Astfel, aplicația trebuie să aiba acces la camera dispozitivului folosit și să poată prelucra fluxul video în timp real. După detectare, prin intermediul interfeței, utilizatorul va ști cu exactitate unde se află obiectul, cu ajutorul unui chenar.
* Identificarea obiectelor în timp real: Această funcționalitate oferă cu o anumită precizie numele obiectului detectat și permite utilizatorului să afle mai multe detalii despre ce se afla în fluxul video al dispozitivului. Prin identificare se înțelege capacitatea aplicației de a eticheta fiecare obiect detectat, împreună cu acuratețea detecției.
* Selecția limbii pentru etichetare: Această cerință exprimă abilitatea utilizatorului de a alege limba în care rezultatele vor fi afișate. Pentru început, am considerat că patru opțiuni sunt suficiente, anume română, engleză, germană și franceză. Pe viitor voi putea adăuga mai multe limbi străine în această listă. Astfel, decizia utilizatorului asupra limbii se va face în meniul principal, unde vor fi evidențiate cele 4 opțiuni într-o secțiune bine definită. Limba selectată trebuie să fie imediat aplicată asupra etichetelor obiectelor detectate.
  1. Cerințe non-funcționale

Pe lângă cerințele funcționale, anumite aspecte care nu fac parte din funcționalitățile aplicației trebuie specificate. Aceste cerințe non-funcționale se referă la calitatea aplicației și atenția la detalii oferită în dezvoltarea acesteia. Astfel, printr-o analiză riguroasă, am ajuns la concluzia că aplicația trebuie să îndeplinească următoarele cerințe:

* Timp de răspuns scăzut: Aplicația trebuie să ofere un timp de răspuns rapid și o întârziere maximă de 1 secundă pentru procesarea și afișarea rezultatelor recunoașterii obiectelor. Când utilizează aplicația pentru a identifica obiecte din mediul lor înconjurător, utilizatorii așteaptă rezultate rapide și imediate.
* Utilizarea eficientă a resurselor: Pentru ca aplicația să funcționeze bine pe diferite dispozitive mobile, trebuie să folosească bine resursele hardware, cum ar fi CPU-ul și memoria RAM. Trebuie luat în considerare faptul că utilizatorii au dispozitive cu diferite specificații hardware și software. Prin urmare, este esențial ca aplicația să fie optimizată pentru a funcționa bine pe o varietate de configurații.
* Stabilitate: Aplicația trebuie să fie stabilă și să ruleze fără erori pe toate dispozitivele disponibile.
* Portabilitate: Luând în considerare diferitele tipuri de platforme și dispozitive pe care utilizatorii le pot folosi, am folosit tehnologii pentru a mă asigura că aplicația mea este disponibilă pe mai multe platforme, inclusiv Android, iOS și desktop.
* Capacitate de extindere: Prin utilizarea unui model de machine learning offline, aplicația trebuie să fie flexibilă la anumite schimbări aduse modelului sau extinderi asupra lui. În cazul în care modelul este dezvoltat în plus, aplicația trebuie să ruleze la fel de bine precum versiunile anterioare.