Universitatea Națională de Știință și Tehnologie POLITEHNICA
București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Analiza imaginilor cu fund de ochi pentru ajutor în diagnosticul retinopatiei diabetice

Lucrare de disertație

prezentat ca cerință parțială pentru obținerea titlului de
Master în domeniul *Inginerie Electronică, Telecomunicații și Tehnologia Informației*
programul de studii de master *Tehnici Avansate de Imagistică Digitală*

Conducător științific,
Conf. dr. ing. Laura Maria Florea

Absolvent,
Ing. Vintilescu Andreea-Alexandra

Anul 2025

Anexa 2

Universitatea Națională de Știință și Tehnologie POLITEHNICA din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Programul de masterat **TAID**

TEMA LUCRĂRII DE DISERTAȚIE
a masterandului VINTILESCU I. Andreea-Alexandra, 421-TAID

1. **Titlul temei:** Analiza imaginilor cu fund de ochi pentru ajutor în diagnosticul retinopatiei diabetice

2. Descrierea temei:

Retinopatia diabetică este o boală oculară cronică progresivă asociată unui grup de probleme oculare ca o complicație a diabetului. Această boală poate provoca pierdere severă a vederii sau chiar orbire. Specialiștii analizează imagini de fund de ochi pentru a diagnostica boala și a oferi tratamente specifice. Clasificarea corectă a imaginilor cu fund de ochi depinde de abilitatea și experiența specialiștilor, precum și de calitatea imaginilor. Scopul lucrării de față este de a folosi algoritmi de vedere computerizată bazați pe rețele convoluționale adânci pentru a crea o metodă de detectare si clasificare a retinopatiei diabetice. Metoda se va testa pe seturi de date publice. Implementarea se va face în Python
folosind biblioteci specifice.

3. Contribuția originală:

1. Se vor analiza metode deja propuse în literatura de specialitate pentru detecția și clasificarea automată a retinopatiei diabetice folosind imagini cu fund de ochi. 2. Se vor căuta seturi de date publice ce conțin imagini cu fund de ochi și adnotări pentru clasificarea retinopatiei diabetice. 3. Se vor analiza seturile de date găsite. 4. Se va propune o metodă bazată pe învățare profundă pentru detecția și clasificarea retinopatiei diabetice. 5. Se va analiza
performanța metodei propuse în diverse cazuri de antrenare/testare.

4. Resurse și bibliografie:

5. **Data înregistrării temei:** 2024-11-29 07:21:43

Conducător(i) lucrare,
Conf. dr. ing. Laura Maria FLOREA

Student,
VINTILESCU I. Andreea Alexandra

Responsabil program master,
Prof. dr. ing. Constantin VERTAN

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **256166cd0f**

Anexa 5

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul " *Analiza imaginilor cu fund de ochi pentru ajutor în diagnosticul retinopatiei diabetice*", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității Naționale de Știință și Tehnologie POLITEHNICA București ca cerință parțială pentru obținerea titlului de *Master* în domeniul Inginerie Electronică Telecomunicații și Tehnologii Informaționale, programul de studii de master Tehnici Avansate de Imagistică Digitală este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București*, 13.06.2025*

Absolvent,

Andreea-Alexandra Vintilescu

Cuprins

Lista figurilor

Lista Tabelelor

Lista Acronimelor

CCNOT	Controlled-Controlled NOT gate	Poartă logică „NU controlat-controlat”
CLS	Class token	Token de clasă
CNN	Convolutional Neural Network	Rețea neuronală convoluțională
CNOT	Controlled NOT gate	Poartă logică „NU controlat”
CSV	Comma-Separated Values	Valori separate prin virgulă
DME	Diabetic Macular Edema	Edem macular diabetic
DQN	Dressed Quantum Net	Rețea neuronală Q profundă
DR	Diabetic Retinopathy	Retinopatie diabetică
FC	Fully Connected	Strat complet conectat
IDRID	Indian Diabetic Retinopathy Image Dataset	Set de date cu imagini pentru retinopatie diabetică – India
MLP	Multi-Layer Perceptron	Perceptron cu mai multe straturi
MSE	Mean Squared Error	Eroare pătratică medie
NLP	Natural Language Processing	Procesarea limbajului natural
QCL	Quantum Circuit Learning	Învățarea circuitelor cuantice
QNN	Quantum Neural Network	Rețea neuronală cuantică
SVM	Support Vector Machine	Mașină de vectori de suport
VIT	Vision Transformer	Transformator vizual
VQC	Variational Quantum Circuit	Circuit cuantic variațional

1. Introducere

În lucrarea de față vom aborda dezvoltarea unei metode automate pentru analiza imaginilor de fund de ochi, cu scopul de a sprijini procesul de diagnosticare al retinopatiei diabetice ce reprezintă o afecțiune gravă și progresivă ce afectează retina și care, în lipsa unui diagnostic precoce și a unei intervenții corespunzătoare, poate conduce la orbire ireversibilă.

În practica medicală, identificarea și clasificarea retinopatiei diabetice se realizează, în mod tradițional, prin analiza imaginilor de fund de ochi (retinografii) de către medici specializați în oftalmologie. Totuși, acest proces presupune un nivel ridicat de expertiză, fiind influențat de experiența specialistului, de calitatea imaginilor și de timpul disponibil pentru examinare. Întrucât acest proces este unul anevoios, subiectiv și dependent de experiența medicului specialist, introducerea unui sistem automatizat și precis poate crește semnificativ eficiența screening-ului și poate reduce riscul diagnosticelor greșite.

Am ales această tema deoarece este una de actualitate și deosebit de relevantă atât din perspectiva tehnologică, cât și socială, dat fiind numărul tot mai mare de pacienți diagnosticați cu diabet, precum și necesitatea unor metode rapide și eficiente de monitorizare a complicațiilor asociate. În plus, tema acestei lucrări prezintă și o semnificație personală, întrucât tatăl meu suferă de diabet, ceea ce m-a făcut să fiu mai conștientă de riscurile implicate, în special cele legate de afectarea vederii. Acest context personal a reprezentat un factor motivațional important în alegerea acestei lucrări.

Progresele recente din domeniul învățării automate, în special în cadrul viziunii computerizate, au condus la dezvoltarea unor arhitecturi performante capabile să analizeze imagini medicale cu un grad ridicat de acuratețe. Rețelele neuronale convoluționale (eng. *Convolutional neural network*, CNN) s-au impus ca soluție de referință în numeroase aplicații de clasificare a imaginilor, printre care și în medicină. În paralel, introducerea transformatorilor vizuali (eng. *Vision Transformers*, ViT) a deschis noi perspective, propunând o abordare alternativă, bazată pe mecanisme de atenție, cu rezultate competitive în diverse sarcini de clasificare.

Lucrarea de față propune dezvoltarea unei metode automate pentru detecția retinopatiei diabetice, prin antrenarea și compararea a două arhitecturi de învățare profundă: un circuit cuantic integrat în arhitectura ResNet50, precum și Vision Transformer, o arhitectură modernă, bazată pe atenție.

Pentru implementarea modelelor s-a utilizat limbajul Python, împreună cu biblioteci precum PyTorch și PennyLane. În etapa experimentală au fost testate două seturi de date publice: „Diabetic Retinopathy 224x224 (APTOS 2019)” și Indian Diabetic Retinopathy. Acestea conțin imagini de fund de ochi adnotate în funcție de severitatea bolii. După evaluarea performanțelor obținute în urma

experimentelor, s-a optat pentru continuarea lucrării cu modelul care a oferit cele mai bune rezultate în ceea ce privește acuratețea și robustețea clasificării. Modelul a fost salvat sub forma unui fișier .pth, reprezentând starea finală a rețelei antrenate, utilizat ulterior pentru inferență și analiză în etapa de testare.

În ceea ce privește cuprinsul lucrarii, aceasta este structurată în cinci capitole. În Capitolul 2, sunt prezentate metode existente în literatura de specialitate, în capitolul 3 conceptele teoretice esențiale pentru înțelegerea metodelor utilizate, incluzând noțiuni despre învățarea automată, rețele neuronale convoluționale, transformatori vizuali și rețele neuronale cuantice. Capitolul 4 descrie procesul de implementare, incluzând selecția datelor, alegerea arhitecturilor și pașii concreți de antrenare. Ulterior, în capitolul 5 este prezentată analiza comparativă a rezultatelor obținute în urma experimentelor, iar în capitolul 6 sunt formulate concluziile și direcțiile viitoare de cercetare propuse.

2. Metode existente în literatura de specialitate

În literatura de specialitate, două direcții moderne și inovatoare atrag tot mai mult interes în analiza imaginilor medicale: utilizarea arhitecturilor de tip Vision Transformer, capabile să surprindă relații globale în cadrul imaginilor, și integrarea metodelor de învățare cuantică variațională, menite să îmbunătățească performanța modelelor în contexte cu date limitate. Cele din urmă fiind recent combinate cu rețele neuronale convoluționale clasice, ducând la apariția unor arhitecturi hibride.

În această secțiune sunt analizate trei lucrări relevante care exemplifică aceste tendințe: o arhitectură ViT combinată cu Capsule Network, și un model ResNet18 integrat cu un strat cuantic, ambele fiind testate pe setul de date APTOS 2019, precum și o metodă de filter pruning prin algoritmul Beta-Rank, aplicată pe un model ResNet56 antrenat pe setul de date IDRiD. Fiecare dintre aceste lucrări propune o perspectivă distinctă asupra modului în care pot fi îmbunătățite modelele existente, fie prin creșterea capacității lor de învățare, fie prin reducerea complexității arhitecturale fără sacrificii majore de performanță.

Prima lucrare analizată, intitulată *„Diabetic retinopathy prediction based on vision transformer and modified capsule network”* (Computers in Biology and Medicine, 2024), propune o arhitectură hibridă ce combină un model Vision Transformer (ViT) pre-antrenat cu un Capsule Network adaptat. ViT are rolul de a extrage caracteristici globale relevante prin mecanisme de atenție, în timp ce Capsule Network menține relațiile spațiale și ierarhice dintre componentele imaginii, oferind astfel un nivel mai profund de interpretabilitate morfologică.

Aplicată pe setul de date APTOS 2019, această combinație demonstrează robustețe în fața dezechilibrului între clase și performanță superioară în captarea detaliilor semnificative pentru diagnostic. Această arhitectură reflectă o direcție de dezvoltare orientată spre creșterea expresivității rețelelor, complementar cu celelalte abordări analizate.

În aceeași linie a extinderii capacităților arhitecturale, lucrarea "*Diabetic Retinopathy Detection Using Quantum Transfer Learning"* (Jain et al., 2024) propune o arhitectură ce combină un model ResNet18 cu un strat final cuantic, sub forma unui circuit variațional. Testat pe același set de date APTOS 2019, modelul a atins o acuratețe de aproximativ 97%, reflectând potențialul calculului cuantic în sarcini de clasificare medicală.

Circuitul VQC introduce fenomene specifice procesării cuantice, precum entanglementul și superpoziția, care oferă o reprezentare mai abstractă și mai generalizabilă a datelor extrase. Această abordare se dovedește utilă în special în situații în care datele sunt puține sau etichetate incomplet.

Spre deosebire de abordările anterioare, care urmăresc extinderea capacității de reprezentare și învățare, lucrarea *"Beta-Rank: A Robust Convolutional Filter Pruning Method for Imbalanced Medical*

Image Analysis" (2023) propune o tehnică de optimizare structurală. Metoda se aplică pe un model ResNet56 antrenat pe setul de date IDRiD și are ca scop reducerea complexității arhitecturale fără compromiterea performanței.

Prin evaluarea importanței filtrelor convoluționale și eliminarea celor redundante, algoritmul Beta-Rank a redus cu 46% numărul de operații și cu 42% parametrii modelului. Cu toate acestea, acuratețea a rămas competitivă, la 80.58%, demonstrând aplicabilitatea metodei în medii cu resurse limitate, precum aplicații embedded sau dispozitive portabile.

Cele trei metode prezentate în această secțiune oferă o imagine de ansamblu asupra varietății de soluții existente în literatura de specialitate pentru clasificarea retinopatiei diabetice. Fiecare dintre acestea evidențiază o direcție diferită de dezvoltare: ViT + Capsule Network aduce avantajul atenției globale și al relațiilor spațiale, VQC + ResNet18 explorează potențialul circuitelor cuantice pentru o generalizare superioară, iar Beta-Rank + ResNet56 se concentrează pe optimizarea arhitecturii pentru a reduce complexitatea fără pierderi semnificative de performanță.

Tabelul comparativ aferent acestor metode este inclus în capitolul 5.3, dedicat comparației cu rezultatele obținute în această lucrare. Prin această comparație, reiese că atât metodele axate pe maximizarea acurateței și a capacității de generalizare, cât și cele care urmăresc reducerea complexității arhitecturale și eficiența computațională, au un loc bine definit în contextul detecției retinopatiei diabetice, fiecare fiind potrivită în funcție de cerințele aplicației și de resursele disponibile. Cele trei metode analizate reflectă tendințele actuale în dezvoltarea de modele hibride pentru clasificarea imaginilor medicale. În timp ce ViT + Capsule Network și VQC + ResNet18 oferă performanțe ridicate prin extinderea capacităților arhitecturale, metoda Beta-Rank propune o soluție practică și eficientă pentru scenarii cu constrângeri hardware. Prin abordările lor complementare, aceste lucrări evidențiază diversitatea soluțiilor moderne pentru detecția automată a retinopatiei diabetice.

3. Noțiuni teoretice

3.1 Introducere în învățarea automată

Învățarea automată (eng. *Machine Learning*) reprezintă o ramură esențială a inteligenței artificiale, concentrată pe dezvoltarea de algoritmi și modele capabile să învețe automat din date primite, care mai apoi să-și îmbunătățească performanța în timp, fără a fi explicit programate pentru fiecare sarcină în parte. Aceste modele sunt instruite pe baza unor seturi de date care conțin exemple reprezentative, din care învață să identifice tipare și să formuleze predicții sau decizii. **[1]**

Dezvoltarea unui model de învățare automată implică un proces complex, compus din mai multe etape succesive care necesită o planificare atentă și o execuție riguroasă. Un flux de lucru bine definit contribuie la organizarea eficientă a sarcinilor și la creșterea performanței modelului final.

În figura 2.1 este ilustrată o schemă generală a fluxului de lucru pentru învățarea automată:

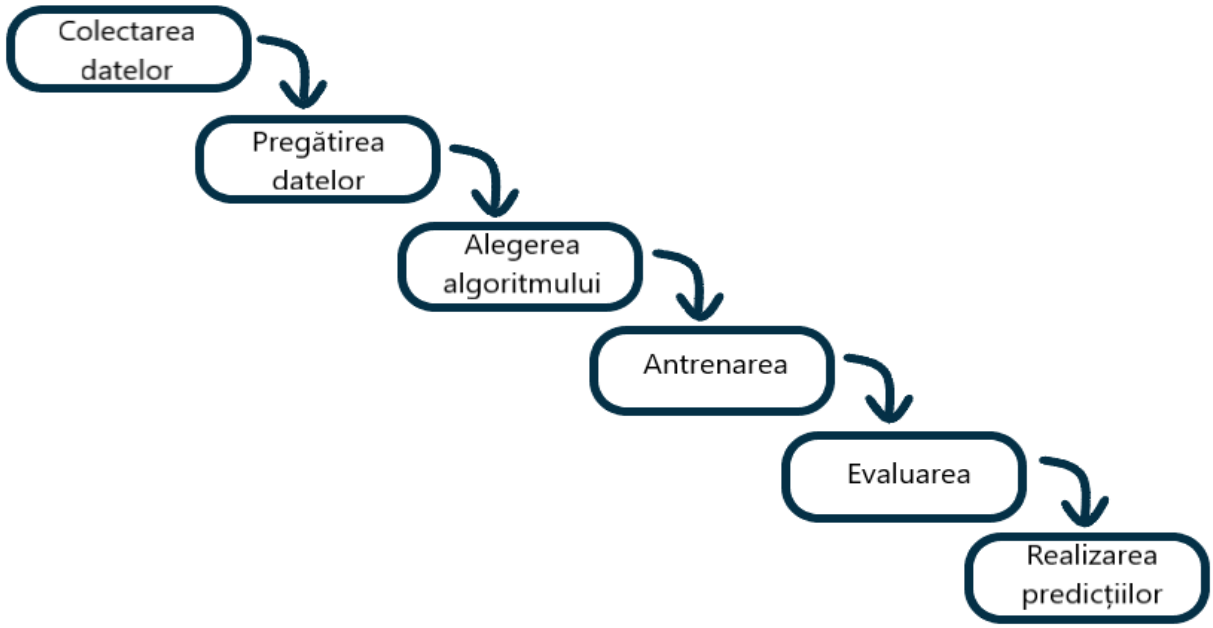

Figura 2.1. Diagrama fluxului de lucru pentru învățarea automată; *Sursa [5]*

Respectarea unui flux de lucru structurat în dezvoltarea modelelor de învățare automată este esențială pentru obținerea unor rezultate eficiente și generalizabile. Fiecare etapă, de la colectarea și pregătirea

datelor, până la alegerea algoritmului, antrenarea, evaluarea și realizarea predicțiilor, contribuie semnificativ la succesul final al sistemului dezvoltat. Gestionarea atentă a acestor pași permite evitarea erorilor comune și crește șansele implementării unor soluții robuste în aplicații reale.

3.1.1 Metode de învățare automată

În continuarea lucrării, pentru o mai bună înțelegere a metodelor utilizate în dezvoltarea sistemelor de învățare automată, va fi prezentată clasificarea general acceptată a tehnicilor de învățare automată, realizată în funcție de natura datelor și de modul de antrenare al modelelor, urmând ca ulterior să fie detaliată metoda aplicată în cadrul acestei lucrări:

- **Învățarea supervizată** (eng. *Supervised Learning*): implică utilizarea unor seturi de date etichetate, unde fiecărei intrări i se asociază o ieșire cunoscută. Modelul învață astfel să realizeze predicții precise pe baza acestor perechi de date. Este folosită frecvent în clasificare și regresie, fiind de altfel și metoda aplicată și în cadrul acestei lucrări. **[2]**

- **Învățarea nesupervizată** (eng. *Unsupervised Learning*): se bazează pe analiza unor date fără etichete, cu scopul de a descoperi structuri sau tipare ascunse, fără o ghidare explicită, deci nu presupune asocierea unei valori de ieșire fiecărei date de intrare. În schimb, modelul este orientat către descoperirea unor caracteristici sau structuri comune în întregul set de date. Deoarece adnotarea manuală a valorilor de ieșire este fie dificilă, fie imposibilă în anumite contexte, această abordare devine esențială în aplicații precum detectarea automată a obiectelor, recunoașterea facială sau analiza tiparelor complexe. **[1]**

- **Învățarea prin întărire** (eng. *Reinforcement Learning*): implică folosirea mai multor soluții la întâmplare, primind recompense sau penalizări în funcție de deciziile luate. Prin urmare, atunci când sistemul acesta execută comportamentul dorit ajunge să fie recompensat, iar în cazul unui comportament nedorit, să fie pedepsit. Astfel, sistemul este capabil să diferențieze acțiunile corecte de cele greșite. Acest tip de învățare a fost conceput inițial pentru inteligențe artificiale capabile să joace jocuri. **[1]**

Această clasificare este recunoscută în mod larg și este considerată standardul de bază în majoritatea lucrărilor de specialitate, însă odată cu evoluția tehnologiilor, au apărut și metode extinse sau hibride, care completează cele trei categorii principale:

- **Învățarea semi-supervizată** (eng. *Semi-Supervised Learning*): combină date etichetate și neetichetate, permițând obținerea de performanțe ridicate în contexte unde etichetarea completă a datelor este dificilă sau costisitoare. **[3]**

- **Învățarea auto-supervizată** (eng. *Self-Supervised Learning*): presupune generarea automată de sarcini artificiale din date neetichetate, modelul învățând astfel să construiască reprezentări utile fără adnotări manuale. **[3]**

- **Învățarea profundă** (eng. *Deep Learning*): reprezintă o abordare bazată pe rețele neuronale adânci, fiind aplicată atât în sarcini supervizate, cât și nesupervizate. Deep learning-ul a revoluționat domenii precum viziunea computerizată, recunoașterea vocală sau procesarea limbajului natural. **[4]**

- **Învățarea profundă prin întărire** (eng. *Deep Reinforcement Learning*): combină rețelele neuronale profunde cu tehnici de învățare prin întărire, permițând agenților să învețe comportamente complexe în medii dinamice. **[4]**

3.1.2 Învățarea supervizată

Având în vedere specificul problemei abordate în cadrul acestei lucrări, s-a optat pentru utilizarea învățării supervizate. În consecință, în cele ce urmează vom detalia această categorie și principalele sale caracteristici.

Învățarea supervizată reprezintă una dintre cele mai frecvent utilizate metode ale învățării automate, fiind esențială în construirea de sisteme capabile să efectueze sarcini precum clasificarea sau regresia. Principiul de bază al acestei metode constă în folosirea unor seturi de date etichetate, în care fiecărei instanțe de intrare i se asociază o ieșire corespunzătoare. Scopul modelului este de a învăța această relație între datele de intrare și etichetele aferente, astfel încât să poată generaliza pe baza celor învățate și dobândite și aplica asupra unor date noi, necunoscute. **[3]**

Procesul de antrenare presupune prezentarea unui număr suficient de exemple, fiecare fiind format dintr-un set de caracteristici și o etichetă asociată. În timpul antrenării, modelul își ajustează parametrii interni prin minimizarea unei funcții de eroare, care măsoară diferența dintre predicțiile generate și valorile reale. Pentru a exemplifica principiile prezentate anterior, în continuare este descris un scenariu concret care evidențiază aplicarea învățării supervizate într-o problemă de clasificare.

Să considerăm un scenariu simplu în care trebuie construit un clasificator capabil să distingă între imagini cu cămile, elefanți și vaci. În acest caz, algoritmul de învățare supervizată este antrenat folosind un set de imagini etichetate corespunzător – fiecare imagine este identificată ca fiind fie o cămilă, fie un elefant, fie o vacă. În timpul antrenării, modelul învață să recunoască trăsăturile caracteristice ale fiecărei clase. Ulterior, atunci când i se furnizează imagini noi, nevăzute anterior, algoritmul utilizează

cunoștințele dobândite pentru a prezice corect dacă imaginea aparține unei cămile, unui elefant sau unei vaci. Acest exemplu ilustrează principiul de bază al învățării supervizate, aplicat într-un context de clasificare a imaginilor.

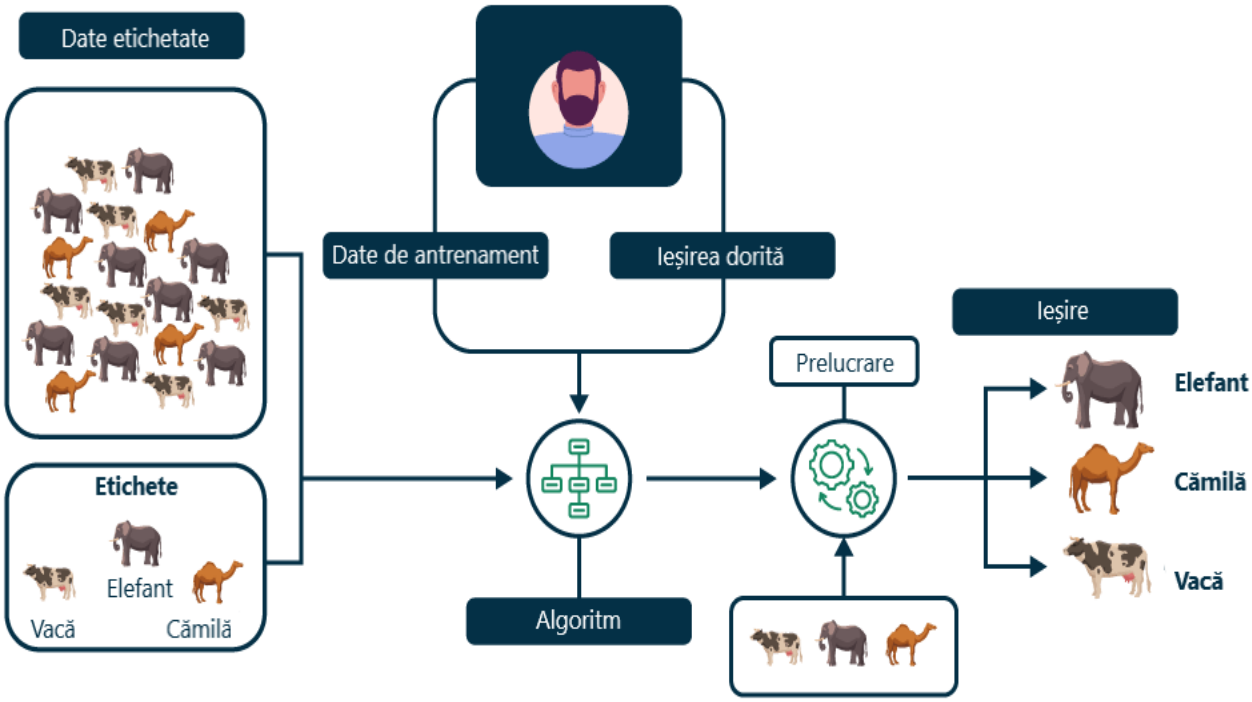

Figura 2.2. Principiul de bază al învățării supervizate; *Sursa [3]*

Astfel, problemele abordate se pot împărți în două categorii principale: **clasificarea** și **regresia**.

Clasificarea presupune atribuirea unei etichete discrete unui exemplu de date pe baza caracteristicilor sale. Modelul învățat trebuie să aleagă una dintre clasele posibile, unde fiecare observație de intrare este încadrată într-o categorie specifică. Dacă luam spre exemplu figura anterioară, un exemplu tipic de clasificare ar fi reprezentat de recunoașterea imaginilor, mai exact diferențierea între imagini cu elefanți și cămile. [3]

Pe de altă parte, **regresia** vizează prezicerea unei valori continue asociate unui exemplu de date. Spre deosebire de clasificare, aici ieșirea modelului nu aparține unei clase discrete, ci reprezintă o valoare numerică. Un exemplu ar putea fi estimarea greutății unui elefant adult pe baza unor caracteristici precum vârsta și dimensiunile corporale. În acest caz, modelul învățat nu atribuie o clasă discretă, ci prevede o valoare continuă reprezentând greutatea exprimată în kilograme. [3]

Deci, în contextul retinopatiei diabetice, problemele pot fi abordate atât ca sarcini de clasificare, cât și de regresie. Un exemplu de clasificare constă în încadrarea imaginilor cu fund de ochi în funcție de

stadiul bolii. În schimb, o problemă de regresie ar putea presupune estimarea procentului de suprafață oculară afectată sau a unui scor continuu de severitate, pe baza caracteristicilor extrase din imagini.

În această lucrare, problema abordată este formulată ca o sarcină de clasificare, unde fiecare imagine oculară este etichetată corespunzător unuia dintre cele cinci stadii ale retinopatiei diabetice, și anume: fără retinopatie diabetică (No DR), retinopatie diabetică ușoară (Mild), retinopatie diabetică moderată (Moderate), retinopatie diabetică severă (Severe) sau retinopatie diabetică proliferativă (Proliferative DR). Scopul principal este dezvoltarea unui model de învățare supervizată capabil să extragă și să înțeleagă caracteristicile relevante din imagini și să generalizeze corect pentru date noi, necunoscute anterior, contribuind astfel la automatizarea procesului de diagnostic.

3.1.3 Provocări și limitări în învățarea automată

Deși învățarea automată a cunoscut o dezvoltare accelerată în ultima vreme, cu impact major asupra numeroaselor domenii de activitate, implementarea practică a modelelor de machine learning rămâne asociată unor provocări semnificative. Gestionarea corectă a acestor dificultăți este esențială pentru obținerea unor soluții robuste și generalizabile.

Așadar, printre principalele provocări identificate se numără:

- Lipsa sau insuficiența datelor de antrenare
- Overfitting-ul
- Underfitting-ul
- Dezechilibrul claselor
- Alegerea modelului și a hiperparametrilor

3.2 Rețele Neuronale Convoluționale (CNN)

Rețelele neuronale convoluționale (eng. *Convolutional Neural Networks,* CNN) reprezintă algoritmi de învățare profundă specializați în aplicații privind procesarea și recunoașterea imaginilor. CNN-ul preia la intrare o imagine și analizează diferitele componente vizuale, acordând importanță distinctă fiecărui obiect sau caracteristică relevantă. Scopul principal al acestor rețele este de a transforma imaginile într-o formă mai simplă și mai ușor de interpretat, fără a pierde informațiile esențiale necesare pentru realizarea unor predicții corecte. **[15]**

Structura unui CNN este alcătuită din mai multe straturi succesive: straturi convoluționale pentru extragerea caracteristicilor, straturi de pooling pentru reducerea dimensiunii datelor și straturi complet

conectate pentru realizarea predicției finale. Învățarea în cadrul CNN-urilor are loc prin propagarea inversă a erorilor, metodă care ajustează treptat parametrii rețelei pe baza performanței obținute.

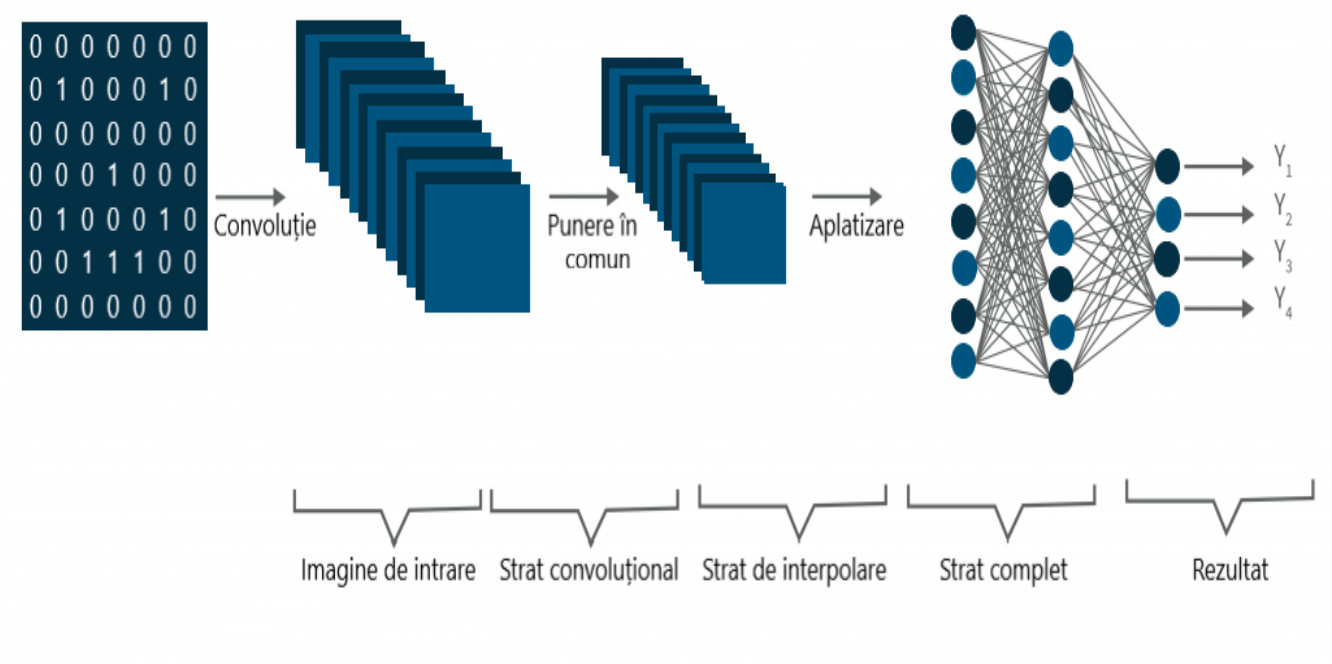

Figura 2.3: Arhitectura tipică a unui CNN; *Sursa [15]*

Spre deosebire de arhitecturile clasice, care procesează doar secțiuni restrânse dintr-o imagine la un moment dat, CNN-urile pot analiza imaginea în întregime, ceea ce le permite să păstreze contextul vizual și să îmbunătățească semnificativ calitatea predicțiilor realizate.

După cum am menționat anterior, o rețea CNN este alcătuită dintr-o succesiune de mai multe straturi, însă principalele straturi sunt următoarele:

- Straturi convoluționale
- Straturi de pooling (interpolare)
- Straturi complet conectate.

Arhitectura începe de regulă cu un strat convoluțional, urmat de alte straturi convoluționale și/sau de pooling, iar în final este utilizat un strat complet conectat pentru realizarea predicției. Straturile inițiale sunt responsabile cu extragerea caracteristicilor de bază, cum ar fi culorile, marginile sau unghiurile, în timp ce straturile superioare pot detecta structuri complexe și obiecte complete. Astfel, prin trecerea succesivă prin aceste straturi, modelul reușește să construiască o reprezentare din ce în ce mai abstractă a datelor, conducând în final la identificarea corectă a obiectelor din imagine. **[16]**

3.2.1 Straturi convoluționale

Stratul convoluțional reprezintă componenta fundamentală a rețelelor neuronale convoluționale, locul în care au loc majoritatea operațiilor de procesare a datelor de intrare. Acest strat utilizează un filtru, denumit și kernel sau nucleu, pentru a efectua operații de convoluție asupra imaginii.

Procesul începe prin deplasarea kernel-ului de-a lungul înălțimii și lățimii imaginii, acoperind întreaga zonă vizuală prin multiple iterații. La fiecare poziție, se calculează un produs punctiform între valorile kernel-ului și valorile pixelilor corespunzători din imagine. Rezultatul acestor calcule este o hartă de caracteristici care reflectă prezența și intensitatea anumitor trăsături vizuale în diferite regiuni ale imaginii. **[15]**

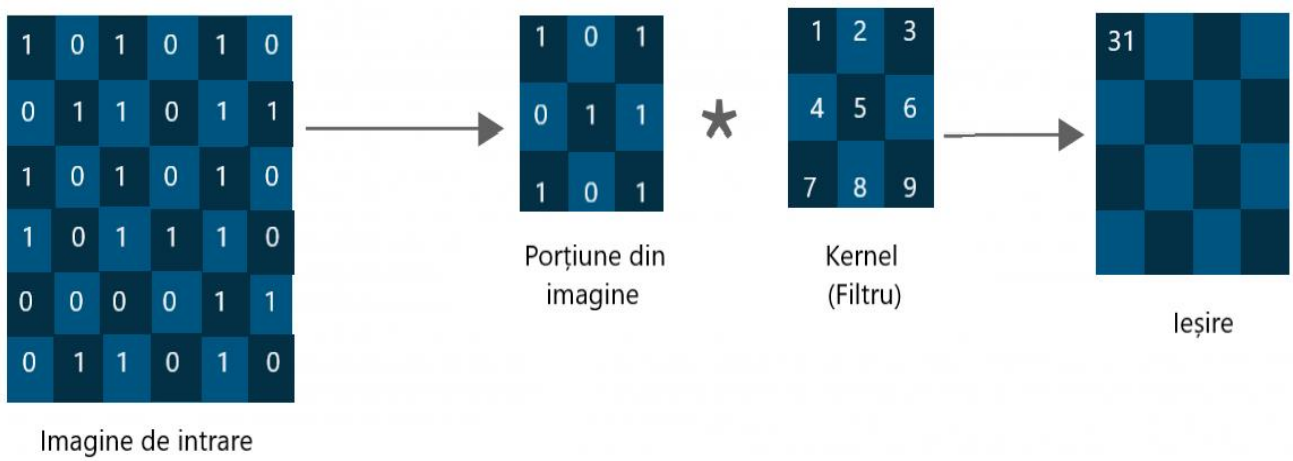

Figura 2.4 : Modul de funcționare al stratului convoluțional; *Sursă: [15]*

Straturile convoluționale dintr-o rețea CNN pot fi suprapuse pentru a construi o ierarhie de caracteristici, în care fiecare strat superior prelucrează și sintetizează informațiile extrase de straturile inferioare. Această organizare permite rețelei să învețe trăsături vizuale din ce în ce mai abstracte, începând de la detalii simple și ajungând la structuri complexe.

3.2.2 Straturi de interpolare

Stratul de pooling are rolul de a reduce dimensiunea spațială a reprezentărilor intermediare din rețea, diminuând astfel numărul de parametri, precum și resursele computaționale necesare. În cazul analizei imaginilor oculare, acest strat contribuie la simplificarea informației extrase în etapele anterioare, permițând modelului să păstreze trăsăturile esențiale pentru clasificarea stadiilor retinopatiei, eliminând în același timp detaliile redundante. **[15]**

La fel ca straturile convoluționale, stratul de pooling parcurge imaginea folosind un filtru, însă în loc să aplice o operație de convoluție, acesta execută o funcție de agregare. Cele mai frecvent utilizate funcții sunt:

- Interpolare maximă, unde selectează cea mai mare valoare dintr-o zonă definită a imaginii;
- Interpolare medie, unde se calculează media valorilor din aceeași zonă.

Deși acest proces conduce la o scădere semnificativă a dimensiunii datelor procesate, acesta aduce și multe beneficii importante: reduce riscul de overfitting, îmbunătățește viteza de antrenare și face rețeaua mai robustă la variații locale. Deși presupune o pierdere parțială de informație, pooling-ul contribuie la generalizarea mai eficientă a modelului, ceea ce este esențial în aplicațiile medicale, unde precizia și stabilitatea rezultatelor sunt prioritare.

3.2.3 Straturi conectate complet

Stratul complet conectat (eng. *Fully Connected*, FC) este, de regulă, plasat în partea finală a arhitecturii unui CNN, imediat înainte de stratul de ieșire responsabil cu clasificarea. Acest strat are rolul de a transforma datele extrase anterior, ce se află sub formă de hărți de caracteristici bidimensionale, într-un vector unidimensional, pregătit pentru luarea deciziei finale. **[16]**

În cazul clasificării imaginilor de fund de ochi pentru detecția retinopatiei diabetice, stratul FC are rolul de a agrega și interpreta semnificația caracteristicilor detectate în etapele anterioare, conducând la atribuirea unei clasele de severitate (fără retinopatie, ușoară, moderată, severă sau proliferativă) imaginii primite la intrarea rețelei. În funcție de complexitatea arhitecturii, pot fi introduse unul sau mai multe astfel de straturi complet conectate, fiecare contribuind la procesarea finală a datelor înainte de predicția propriu-zisă. **[16]**

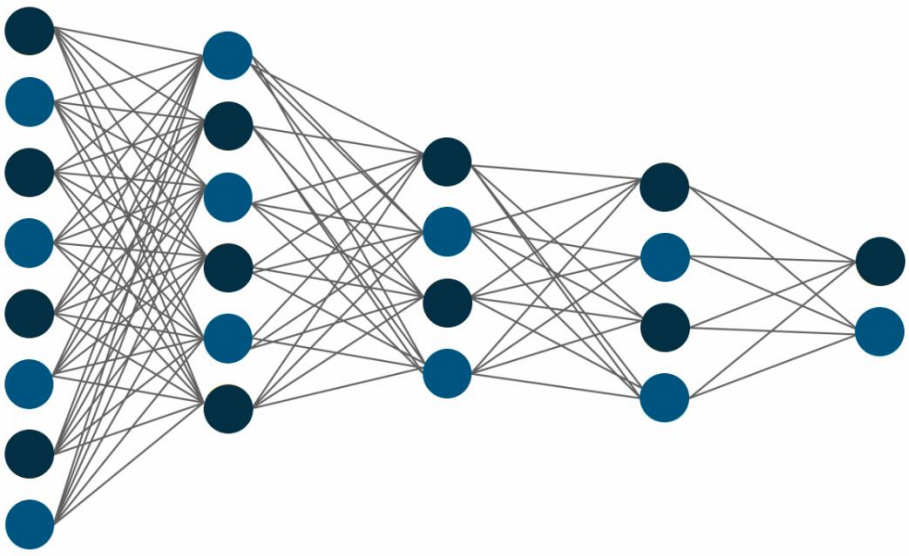

Figura 2.5 : Reprezentare schematică a conexiunilor în stratul FC; *Sursă: [15]*

Deși o arhitectură tipică a rețelelor CNN este compusă, în general, din cele trei straturi prezentate anterior, este important de menționat faptul că, în rețelele moderne, stratului complet conectat nu îi revine în mod obligatoriu rolul final în rețea, iar arhitectura poate fi extinsă și adaptată în funcție de complexitatea aplicației și de nevoile specifice ale modelului.

3.3 Rețele neuronale cuantice (QNN)

Rețelele neuronale cuantice, cunoscute sub denumirea de *Quantum Neural Networks*, reprezintă o tehnologie aflată în dezvoltare care combină idei din două domenii importante: învățarea automată și calculul cuantic. Ele pornesc de la structura generală a rețelelor neuronale clasice, dar în loc să folosească biți și operații matematice obișnuite, se bazează pe concepte specifice mecanicii cuantice.

Aceste rețele pot fi complet cuantice, operând direct pe stări cuantice, sau hibride, în care datele clasice sunt codificate în stări cuantice. QNN-urile moderne sunt în general implementate ca circuite cuantice parametrizate (algoritmi variaționali), adaptate pentru a învăța modele complexe din date vizuale. Aceste rețele oferă avantaje semnificative în contextul detecției automate, cum ar fi capacitatea de a învăța operații unitare necunoscute și de a se adapta la variațiile subtile din imagini. De asemenea, ele pot fi folosite pentru reducerea zgomotului în date, reconstrucția sau augmentarea imaginilor și chiar generarea de noi exemple sintetice utile pentru antrenare. **[23]**

3.3.1 Qubiții și caracteristicile acestora

Dacă in rețelele neuronale clasice unitățile binare folosite pentru procesarea informației sunt reprezentate de biți ce pot prezenta doar una dintre cele două valori, echivalentul cuantic este numit qubit. Acesta se poate afla într-o superpoziție de stări, care se păstrează până la momentul în care qubitul este măsurat, sistemul reducându-se la una dintre cele două stări posibile, în funcție de distribuția de probabilitate asociată amplitudinilor sale. **[22]**

Așadar, un qubit poate reprezenta simultan ambele stări, fiind descris ca un vector normalizat într-un spațiu Hilbert complex bidimensional $\mathbb{H}=\mathbb{C}^2$:

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle \qquad (1)$$

unde α și β sunt numere complexe, $|\alpha|^2 + |\beta|^2 = 1$ și $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Prin urmare, un grup de qubiți poate stoca și prelucra mult mai multă informație decât un număr echivalent de biți. Această caracteristică oferă un potențial ridicat de paralelism în procesarea datelor.

Totuși, qubitul se distinge și prin alte proprietăți unice, imposibil de reprodus în sistemele clasice, precum fenomenul de entanglement. Această proprietate implică faptul că doi sau mai mulți qubiți devin interdependenți într-un mod profund, adică starea unuia dintre ei nu mai poate fi descrisă independent de starea celorlalți, indiferent de distanța spațială care îi separă. **[22]**

De asemenea, și interferența joacă un rol central în dinamica qubitului. Amplitudinile de probabilitate care caracterizează starea sa pot interacționa constructiv sau distructiv în cadrul unui algoritm cuantic. Acest fenomen permite consolidarea șanselor de obținere a răspunsurilor corecte și suprimarea celor asociate rezultatelor eronate, reprezentând astfel cheia eficienței anumitor algoritmi cuantici.

În final, măsurarea unui qubit este procesul prin care informația cuantică este convertită într-o valoare clasică, permițând obținerea unui rezultat interpretabil. Acest proces este însă ireversibil și determinist doar la nivel statistic: fiecare execuție a aceluiași algoritm poate produce un rezultat diferit, dar distribuția generală a ieșirilor reflectă starea cuantică inițială și evoluția sistemului. **[20]**

În figura 2.6 se pot observa stărilor cuantice ale unui qubit, în care superpoziția este vizualizată ca un punct aflat oriunde pe suprafața sferei, nu doar la polii corespunzători stărilor clasice $|0\rangle$ și $|1\rangle$. Această interpretare geometrică ilustrează nu doar natura stărilor cuantice, ci și modul în care acestea pot fi manipulate prin porți cuantice. **[20]**

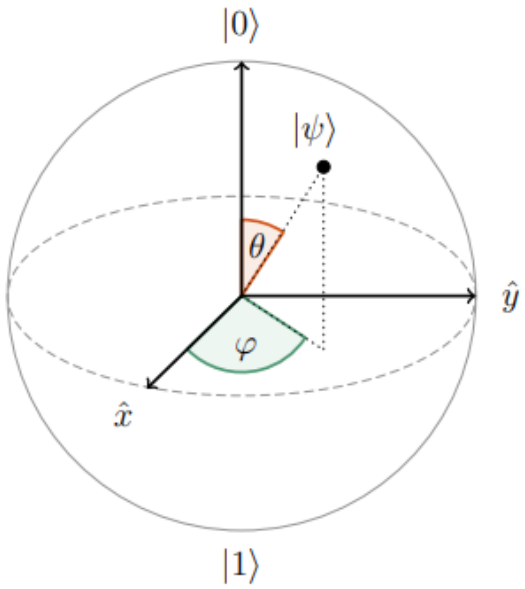

Figura 2.6: Sfera Bloch - reprezentarea stărilor cuantice ale unui qubit; *Sursă [23]*

3.3.2 Porți cuantice

Pentru explorarea rețelelor cuantice, sunt necesare circuitele cuantice. Acestea descriu pe lângă procesele de inițializare a stărilor qubiților, aplicarea de operații asupra acestora și citirea rezultatelor prin măsurători.

Prin urmare, operațiile asupra qubiților se realizează prin porți cuantice care corespund unor transformări unitare. În mod fundamental, ele constituie blocurile de construcție ale oricărui algoritm cuantic, fiind analogul logicii booleene din circuitele clasice. Acestea controlează modul în care datele sunt procesate în spațiul Hilbert al stărilor cuantice.

Porțile cuantice pot fi clasificate în două mari categorii: porți cu un singur qubit și porti cu 2 sau mai mulți qubiti.

Porțile cu un singur qubit afectează starea unui singur qubit și sunt utilizate pentru a genera superpoziție, a efectua rotații și a schimba faza amplitudinilor, astfel:

- Hadamard (H)

Aceste porți creează o superpoziție egală între stările $|0\rangle$ și $|1\rangle$

$$H|0\rangle=\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$$

- Porți de rotație în jurul axelor: $R_X(\theta)$, $R_Y(\theta)$, $R_Z(\theta)$

Aceste porți permit ajustarea fină a stării cuantice și sunt frecvent utilizate în rețelele neuronale cuantice, unde echivalează cu funcțiile de activare controlabile din rețelele neuronale clasice.

- Porți Pauli (X, Y, Z)

Aceste porți oferă operații logice de bază: poarta X inversează starea qubitului (similar cu o poartă NOT), poarta Y introduce o combinație între inversare și rotație, iar poarta Z aplică o modificare de fază asupra componentei $|1\rangle$.

Pe de altă parte, porțile care acționează asupra a doi sau mai mulți qubiți sunt esențiale pentru generarea entanglementului, o proprietate exclusiv cuantică ce permite corelarea puternică între qubiți.

- Porți CNOT (Controlled-NOT)

Aceste porți modifică starea unui qubit țintă doar dacă qubitul de control se află în starea |1⟩. Această poartă este un element de bază în circuitele variaționale și este adesea utilizată în construcția rețelelor de tip QNN pentru a lega parametric stările intermediare ale mai multor qubiți.

- Porți CZ (Controlled-Z)

Aceste porți aplică o inversare de fază asupra stării |11⟩|, lăsând celelalte stări neschimbate.

- Toffoli (CCNOT)

Aceste porți condiționează o inversare logică de starea simultană a două qubiți de control. Prin combinarea acestor porți în secvențe bine definite, se construiesc circuite cuantice cu funcții de decizie parametrizabile.

3.3.3 Arhitecturi cuantice

În ceea ce privește tipurile de arhitecturi QNN, de-a lungul timpului au fost dezvoltate și adaptate diferitelor tipuri de date, scopuri și constrângeri hardware. Dintre acestea, trei modele se remarcă prin aplicabilitate, eficiență și popularitate în literatura de specialitate: *Variational Quantum Circuit* (VQC), *Dressed Quantum Network* (DQN) și *Quantum Circuit Learning* (QCL). Aceste arhitecturi sunt caracterizate printr-o integrare hibridă între componente clasice (pentru extragerea trăsăturilor) și componente cuantice (pentru procesare și luare a deciziei), oferind un echilibru între complexitatea modelului și fezabilitatea implementării. **[23]**

Arhitectura VQC reprezintă una dintre cele mai utilizate forme de rețele neuronale cuantice în aplicații de clasificare, inclusiv în analiza imaginilor medicale. Aceasta constă într-un circuit cuantic variațional, format dintr-un strat de codificare (în care datele clasice sunt translatate în rotații aplicate asupra qubiților), urmat de o secvență de porți cuantice antrenabile, organizate într-un ansamblu. VQC permite modelarea relațiilor neliniare între trăsături și este potrivit pentru date de dimensiune redusă, preprocesate anterior. **[22]**

Arhitectura DQN reprezintă o soluție hibridă modernă, concepută pentru a combina avantajele rețelelor neuronale clasice cu puterea procesării cuantice. Într-un astfel de model este utilizat un backbone clasic, precum un ResNet preantrenat, pentru extragerea trăsăturilor relevante din imaginile de intrare. În cadrul acestei lucrări, DQN este implementată prin încadrarea unui circuit cuantic variațional între două straturi complet conectate clasice: primul, de preprocesare care are rolul de a reduce dimensionalitatea trăsăturilor extrase și de a le transforma într-un format compatibil cu codificarea cuantică, iar al doilea, de postprocesare care preia valorile rezultate în urma măsurării cuantice și care generează predicția finală a modelului. **[22]**

În figura 2.7 se regăsește modelul de rețea hibrid, folosit și în cadrul acestei lucrări:

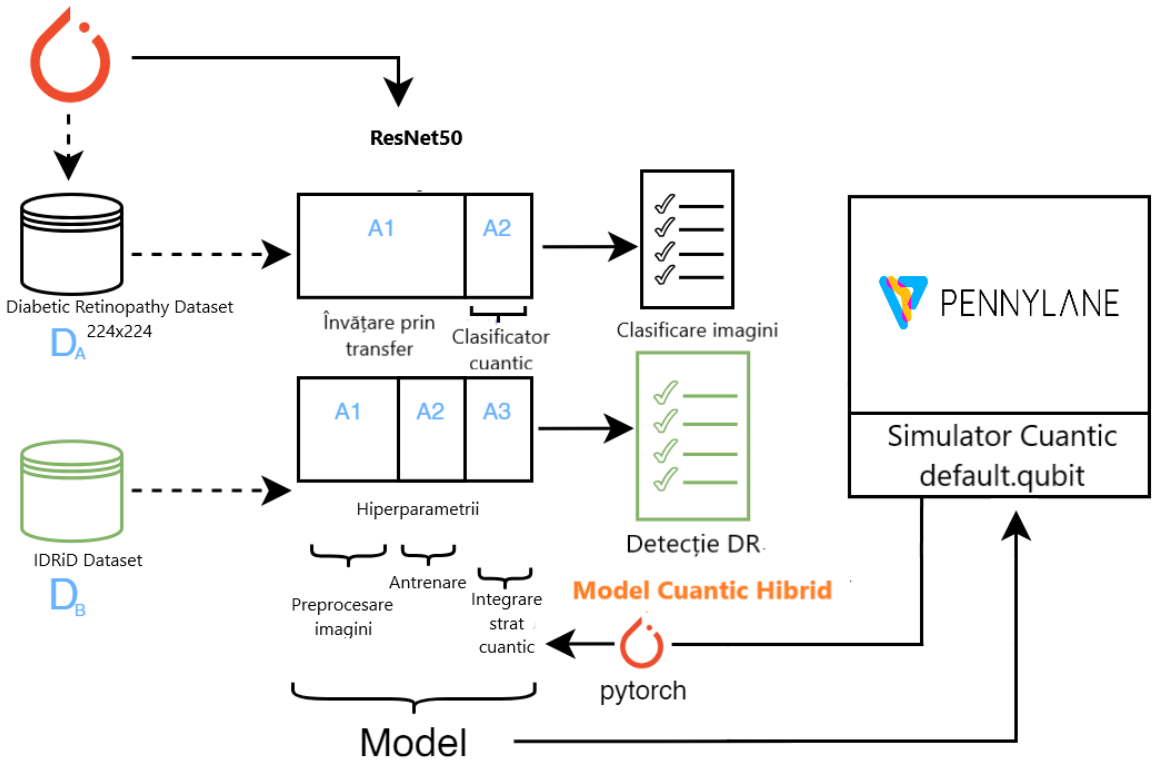

Figura 2.7: Modelul de rețea hibrid Dressed Quantum Net

În cele din urmă, Quantum Circuit Learning este o arhitectură simplificată de rețea neuronală cuantică, în care circuitul cuantic este utilizat pentru a aproxima o funcție de decizie sau o distribuție probabilistică asupra datelor clasice. De obicei, modelul constă într-o secvență fixă de porți parametrizate și nu implică neapărat un ansamblu clasic pentru preprocesare. QCL este folosit în probleme de regresie sau clasificare binară și poate fi antrenat cu un set redus de date, având o complexitate de implementare mai scăzută comparativ cu VQC sau DQN. [23]

Prin urmare, alegerea tipului de rețea neuronală cuantică depinde în mod direct de natura problemei abordate, complexitatea datelor și capabilitățile hardware disponibile.

3.4 Transformatori Vizuali (ViT)

Deși rețelele neuronale convoluționale rămân o soluție dominantă în sarcinile de clasificare a imaginilor, inclusiv în detecția automată a retinopatiei diabetice, evoluțiile recente în domeniul învățării profunde au adus în prim plan o nouă arhitectură promițătoare, și anume Vision Transformer. Aceasta renunță complet la straturile convoluționale tradiționale și adoptă o abordare inspirată din procesarea

limbajului natural (NLP), tratând imaginile ca secvențe de patch-uri. Astfel, fiecare imagine este împărțită în regiuni pătrate de dimensiuni fixe, numite patch-uri, care sunt aplatizate și transformate în vectori de dimensiune fixă, echivalenți cu "tokens" NLP. [17]

La această secvență de tokens se adaugă un token special de clasificare [CLS] și embedding-uri poziționale care păstrează informația despre poziția fiecărui patch în imagine. Întreaga secvență este apoi procesată de encoderul Transformer, care învață relațiile globale dintre toate patch-urile, prin mecanismul de autoatenție. [17]

La final, predicția este realizată pe baza tokenului [CLS], care agregă informația contextuală necesară clasificării. Această abordare permite ViT să învețe relații globale între regiuni ale imaginii, în contrast cu abordarea locală a CNN-urilor. Deși necesită seturi de date mari pentru a atinge performanțe competitive, ViT a demonstrat rezultate superioare pe benchmark-uri de clasificare a imaginilor atunci când este pre-antrenat la scară largă. [17]

În figura 2.10 este ilustrat fluxul principal de procesare, începând cu împărțirea imaginii în patch-uri și terminând cu generarea predicției finale pe baza tokenului de clasificare [CLS]:

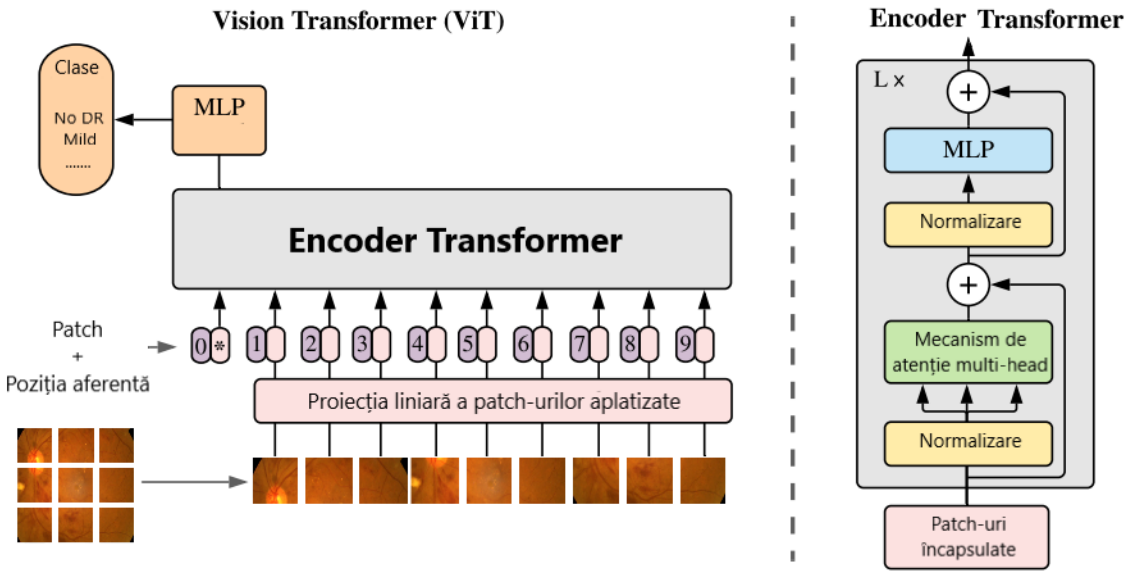

Figura 2.10: Arhitecturii ViT și a fluxului său de procesare; *Sursa [17]*

Pentru o și mai bună înțelegere a modului de funcționare, în continuare sunt detaliați pașii arhitecturali esențiali care definesc structura internă a modelului Vision Transformer: [17]

- Împărțirea imaginii în patch-uri

Imaginea de intrare, de dimensiune H × W × C, este divizată în patch-uri pătrate, fiecare de dimensiune fixă P × P. Astfel, rezultă un număr total de patch-uri N = H × $\frac{W}{P2}$, fiecare reprezentând o regiune locală din imagine. Fiecare patch este apoi aplatizat într-un vector unidimensional de dimensiune P^2*C.

- Proiecția patch-urilor în spațiul de embedding

Fiecare patch aplatizat este transformat într-un vector de dimensiune fixă, utilizând un strat de proiecție liniară. Această proiecție are rolul de a uniformiza reprezentarea patch-urilor, permițând astfel ca ele să fie tratate ca o secvență coerentă de intrări în encoderul Transformer.

- Adăugarea tokenului [CLS] și a embeddingurilor poziționale

Pentru a realiza clasificarea, se adaugă un token special [CLS] în fața secvenței de patch-uri. Acesta este proiectat să acumuleze informația globală pe parcursul procesării, fiind ulterior utilizat pentru generarea predicției. În plus, fiecărui patch i se adaugă un vector de poziție antrenabil, care oferă modelului informații despre locația fiecărui element în imagine.

- Procesarea secvenței de patch-uri prin encoderul Transformer

Secvența completă, alcătuită din patch-urile proiectate, embeddingurile poziționale și tokenul [CLS], este transmisă unui encoder Transformer format din multiple blocuri repetate. Fiecare bloc conține două componente principale:

  - un strat de autoatenție multi-head, care permite învățarea relațiilor globale între toate patch-urile;
  - o rețea neuronală complet conectată (MLP), care adaugă capacitate de modelare neliniară.

Ambele componente sunt precedate de normalizarea efectivă și conectate prin conexiuni reziduale pentru a facilita antrenarea profundă.

- Clasificarea

După ce întreaga secvență este procesată, ieșirea asociată tokenului [CLS] este extrasă și transmisă către un cap de clasificare (de obicei un MLP). Acesta generează predicția finală, de exemplu stadiul retinopatiei diabetice în cazul unei imagini de fund de ochi.

Deși ViT aduce o serie de avantaje față de arhitecturile convoluționale clasice, aceasta implică și anumite provocări, în special în ceea ce privește necesarul de date și resurse computaționale. Pentru a contura o imagine completă asupra potențialului acestei arhitecturi în domeniul viziunii computerizate, vor fi prezentate pe rând avantajele și limitările față de CNN, capacitatea de scalare și flexibilitatea structurală, relevanța în aplicațiile medicale. Aceste perspective vor evidenția de ce ViT a devenit o

alternativă viabilă și competitivă în sarcini vizuale complexe, inclusiv în contextul detectării automate a retinopatiei diabetice.

3.4.1. Avantaje și limitări ale Vision Transformer față de CNN

Arhitectura Vision Transformer se diferențiază fundamental de rețelele neuronale convoluționale prin modul de abordare a procesării imaginilor. În timp ce CNN-urile învață caracteristici locale prin aplicarea de filtre pe regiuni restrânse ale imaginii, ViT folosește mecanismul de autoatenție globală pentru a analiza întregul conținut vizual simultan. Astfel, modelul poate corela informații provenite din zone îndepărtate ale unei imagini, un lucru dificil de realizat în mod direct cu rețelele convoluționale clasice. **[17]**

În plus, ViT elimină complet utilizarea straturilor convoluționale, ceea ce duce la o arhitectură mai simplificată și mai modulară. Absența unor componente specifice CNN-urilor, precum straturile de pooling, facilitează integrarea cu alte modele Transformer din domeniul procesării limbajului natural, favorizând astfel dezvoltarea unor arhitecturi multi-modale. Totodată, datorită mecanismului de autoatenție multi-head, ViT este capabil să surprindă relații spațiale complexe și subtile, ceea ce îl face deosebit de util în scenarii în care elementele vizuale sunt dispersate sau slab definite – cum este cazul anumitor leziuni greu de identificat în cadrul imaginilor cu fund de ochi.

Cu toate acestea, ViT nu beneficiază de inductive bias-urile încorporate în CNN-uri, cum ar fi echivarianța la translație. Din acest motiv, performanța ViT depinde în mod semnificativ de antrenarea pe seturi de date mari și variate. În absența acestora, capacitatea de generalizare poate fi afectată, iar complexitatea atenției globale implică un consum computațional ridicat, mai ales în cazul imaginilor de rezoluție înaltă. **[17]**

3.4.2 Scalabilitatea și flexibilitatea arhitecturii ViT

Un avantaj major al ViT este scalabilitatea sa arhitecturală. Datorită designului inspirat din procesarea limbajului natural, ViT poate fi extins cu ușurință prin adăugarea de straturi suplimentare sau creșterea dimensiunii vectorilor latenti, fără a fi necesare modificări structurale profunde. **[18]**

Această flexibilitate permite antrenarea unor modele foarte mari, cu sute de milioane de parametri, beneficiind de optimizările existente în ecosistemul NLP. În plus, îl face ideal pentru pre-antrenare la scară largă, urmată de fine-tuning pe seturi mai mici, o strategie ce a dus la obținerea unor rezultate de referință pe benchmark-uri vizuale precum ImageNet și CIFAR-100. **[19]**

Mai mult, arhitectura sa permite adaptarea facilă la diverse constrângeri computaționale sau de date, precum și în cazul lucrării acesteia în care această adaptabilitate a fost esențială. Prin urmare, ViT poate fi calibrat în funcție de dimensiunea setului de date și resursele disponibile, menținând un echilibru între complexitate și performanță.

4. Implementare

4.1 Librării folosite

În cadrul acestei implementări, au fost utilizate mai multe librării, grupate în funcție de rolul lor specific în arhitectura aplicației. Acestea au contribuit la gestionarea datelor, prelucrarea imaginilor, antrenarea modelelor de învățare automată și vizualizarea rezultatelor.

Librăriile de bază:

- *os* a fost utilizată pentru operații de manipulare a fișierelor și directoarelor, precum navigarea prin structura de foldere, verificarea existenței fișierelor, construirea dinamică a căilor si crearea de directoare.
- *csv* a fost folosită pentru citirea fișierelor de adnotări în format .csv, conținând informații despre etichetele imaginilor. De asemenea, a fost utilă în extragerea rapidă și simplă a datelor fără a necesita procesarea manuală a fișierelor text.
- *pandas* a fost folosită pentru manipularea eficientă a fișierului CSV cu denumirile imaginilor și etichetele asociate. Aceasta a reprezentat o abordare mai eficientă și mai intuitivă.

Librăriile de procesare de imagini:

- *PIL* a fost utilizată pentru deschiderea, conversia și salvarea imaginilor. De asemenea, a permis transformări de bază, fiind esențială în etapa de preprocesare a imaginilor.
- *torchvision.transforms* – a fost utilizattă pentru augmentarea și normalizarea imaginilor.

Librăriile de învățare automată:

- *torch* a reprezentat biblioteca principală pentru rețele neuronale și manipularea tensorilor.
- *torchvision* a fost folosită pentru funcționalitățile legate de imagini și modele pre-antrenate.
- *transformers* utilizată în cadrul acestei lucrări pentru încărcarea modelului ViT.
- *torch.utils.data* folosită pentru lucrul cu seturi de date personalizate și DataLoader.

Librăriile de vizualizare a datelor:

- *matplotlib* a fost utilizată pentru reprezentarea grafică a evoluției performanței modelului pe parcursul antrenării.
- *seaborn* a fost folosită pentru afișarea matricelor de confuzie sub formă de heatmap, facilitând evidențierea distribuției clasificărilor corecte și incorecte între clasele retinopatiei diabetice

4.2 Abordarea problemei

În această secțiune vom vorbi despre modul în care a fost abordată problema clasificării imaginilor în funcție de severitatea retinopatiei diabetice și etapele implementării soluției, cu scopul final de a construi un model robust care poate oferi o predicție precisă. Procesul de implementare a fost structurat în mai multe etape esențiale, fiecare cu importanța sa în obținerea unui rezultat final funcțional și valid. Etapele parcurse sunt prezentate în continuare:

4.2.1 Înțelegerea problemei

Clasificarea imaginilor din setul de date a presupus încadrarea fiecărei imagini într-una dintre clasele corespunzătoare severității retinopatiei diabetice (de la prima clasă ce ilustrează absența afecțiunii, până la ultima care reprezintă o formă severă a acestei boli). Fiecare imagine conține numeroase detalii fine, precum microanevrisme, hemoragii sau exsudate, care pot fi greu de observat, chiar și de către specialiști. În acest context, alegerea unui model capabil să surprindă aceste subtilități vizuale a fost esențială.

Din acest motiv este nevoie să fie luate în considerare alternative moderne față de rețelele neuronale convoluționale tradiționale. Un exemplu relevant l-ar purea reprezenta arhitectura Vision Transformer, care utilizează mecanismul de atenție globală pentru a modela relațiile dintre toate regiunile unei imagini, indiferent de distanța lor spațială. Această abordare s-ar putea dovedi promițătoare în acestă lucrare, unde contextul global și distribuția leziunilor pot influența semnificativ interpretarea și diagnosticul.

4.2.2 Alegerea setului de date

În etapa de început a proiectului, am analizat mai multe seturi de date relevante pentru tema lucrării, cu scopul de a identifica cele mai potrivite surse pentru antrenarea și testarea unui model de clasificare a retinopatiei diabetice. Criteriile avute în vedere au inclus dimensiunea setului de date, calitatea imaginilor, diversitatea cazurilor reprezentate, precum și structura fișierelor.

După această etapă de documentare și evaluare, au fost selectate pentru testare două seturi de date:

- IDRiD (Indian Diabetic Retinopathy Image Dataset)
- Diabetic Retinopathy - 224x224, varianta resized pentru APTOS 2019

Setul de date **IDRiD** (Indian Diabetic Retinopathy Image Dataset) este un set de date complex cu imagini la o rezoluție înaltă de aproximativ 4288 x 2848 pixeli, creat pentru a sprijini cercetarea în diagnosticarea automată a retinopatiei diabetice (DR) și a edemului macular diabetic (DME). Este unul dintre cele mai bine cunoscute și utilizate seturi de date pentru dezvoltarea și evaluarea algoritmilor de segmentare și clasificare în acest domeniu. Setul conține atât adnotări pentru clasificarea severității retinopatiei diabetice și a edemului macular, cât și măști de segmentare pentru diverse tipuri de leziuni (microanevrisme, hemoragii, exsudate tari și moi).

Prin comparație, setul de date **Diabetic Retinopathy 224x224 (2019)**, oferă imagini deja redimensionate și structurate într-un format mai prietenos pentru antrenare, cu un fișier CSV clar ce asociază fiecare imagine cu eticheta sa corespunzătoare, deci fiecare imagine este încadrată într-una dintre cele cinci clase corespunzătoare severității retinopatiei diabetice, de la absența afecțiunii (clasa 0) până la forme avansate (clasa 4). Această simplitate a permis o integrare rapidă cu modelul și o concentrare mai mare asupra etapelor de experimentare și analiză a rezultatelor.

Inițial, pentru această etapă a cercetării, s-a decis continuarea lucrării cu al doilea set de date, păstrând însă setul IDRiD ca alternativă pentru experimente viitoare sau pentru validări suplimentare ale modelului. Această alegere a fost motivată de faptul că imaginile sunt deja redimensionate uniform la 224x224 pixeli, iar fișierul CSV asociat este simplu de interpretat și utilizat. Prin comparație, setul IDRiD necesită o preprocesare suplimentară și o mapare mai complexă între directoare și adnotări, ceea ce ar fi crescut timpul de implementare fără beneficii semnificative în această etapă a cercetării.

Mai mult decât atât, această decizie a fost luată și în urma unei analize statistice pe ambele seturi, comparând distribuția instanțelor pe clase. În urma acestora, s-a observat că, deși ambele seturi conțin etichete de severitate între 0 și 4, setul IDRiD prezintă o problemă serioasă de dezechilibru și un volum semnificativ mai mic de date. Se poate observa un dezechilibru pronunțat, cu o predominanță a claselor 0 și 2, în timp ce clasele 1 și 4 sunt semnificativ subreprezentate, aspect care poate afecta negativ capacitatea modelului de a învăța reprezentări corecte pentru toate categoriile.

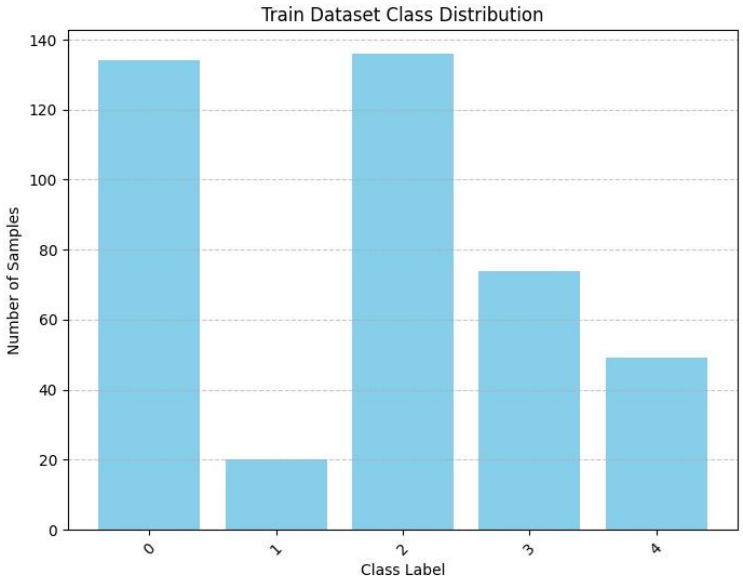

Figura 3.1. Distribuția claselor în setul de antrenament – dataset IDRiD

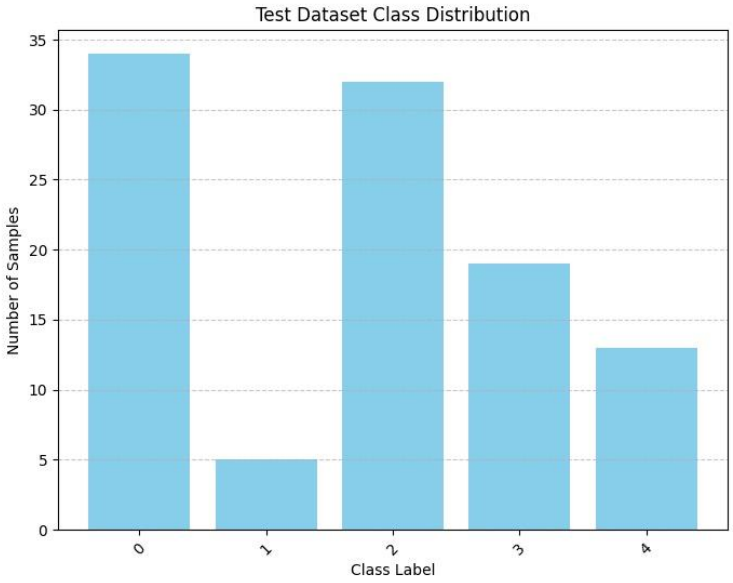

Figura 3.2. Distribuția claselor în setul de testare – dataset IDRiD

Prin contrast, setul de pe Diabetic Retinopathy 224x224 are o acoperire mai extinsă și o distribuție relativ mai echilibrată. Graficul ilustrează o distribuție mai echilibrată a imaginilor în cele două seturi. Deși clasa 0 rămâne predominantă, restul claselor sunt reprezentate în număr mai mare față de datasetul IDRiD. Acest aspect îl face mai potrivit pentru o sarcină de clasificare multi-clasă, reducând riscul de bias și crescând șansele ca modelul să generalizeze corect. Mai mult, acest set este deja preprocesat (224x224 px), ceea ce facilitează integrarea în modele pre-antrenate precum ViT, fără costuri suplimentare de resurse computaționale.

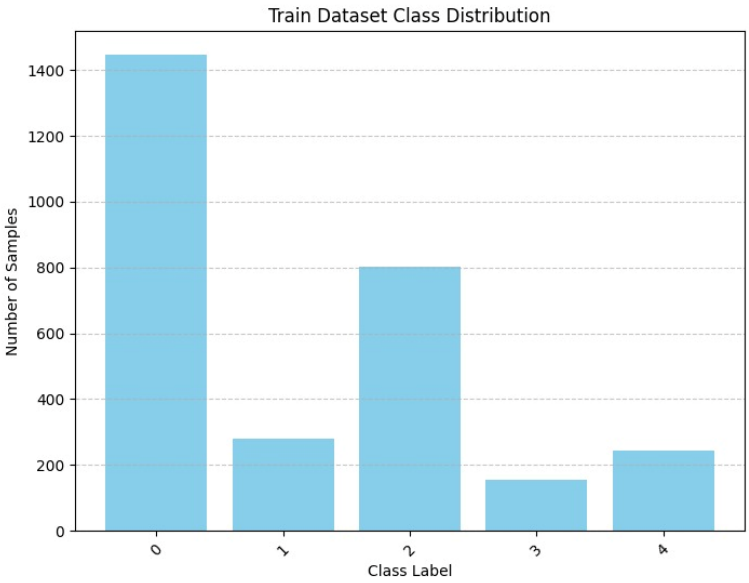

Figura 3.3. Distribuția claselor în setul de antrenament – dataset Diabetic Retinopathy 224x224

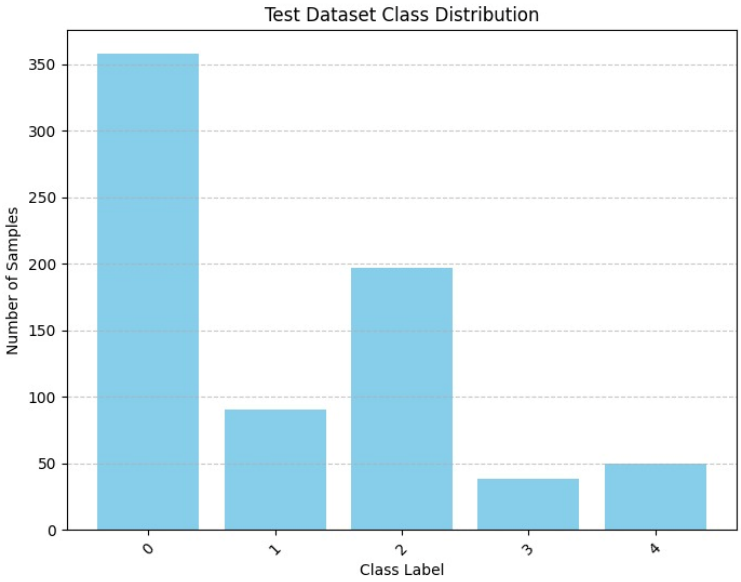

Figura 3.4. Distribuția claselor în setul de testare – dataset Diabetic Retinopathy 224x224

Însă, ulterior evaluărilor preliminare a celor două seturi de date selectate și din curiozitatea de a urmării rezultatele obținute în urma antrenării pe primul set de date, am decis continuarea antrenării cu ambele. Această alegere are la bază nevoia de a testa robustețea și capacitatea de generalizare a modelelor propuse. Prin utilizarea ambelor seturi de date în paralel, se asigură o comparație obiectivă și relevantă a performanțelor arhitecturilor de rețea într-un cadru experimental echilibrat.

4.2.3 Alegerea unei arhitecturi

Clasificarea imaginilor din setul de date a presupus ca fiecare imagine să fie încadrată într-una din cele cinci clase. Fiecare imagine conținând detalii fine, cum ar fi microanevrisme, hemoragii sau exsudate, detalii ce pot fi subtile și greu de detectat chiar și de către specialiști. Prin urmare, alegerea unui model de clasificare care să poată surprinde aceste subtilități a reprezentat o etapă foarte importantă în realizarea acestei lucrări.

Prin urmare, au fost explorate două arhitecturi distincte de clasificare a retinopatiei diabetice: o rețea neuronală hibridă construită pe baza unei arhitecturi clasice CNN, integrată cu un circuit cuantic de tip *DressedQuantumNet* și arhitectura Vision Transformer, ambele fiind implementate și antrenate pe aceleași seturi de date pentru a putea compara performanțele și comportamentul acestora în mod obiectiv.

Modelul hibrid se bazează pe o arhitectură de tip *Dressed Quantum Network* (DQN), o formă hibridă de rețea neuronală care combină procesarea clasică cu cea cuantică. Structura generală a modelului este compusă din trei blocuri funcționale:

- Stratul de preprocesare clasic constă într-un set de straturi dense care transformă vectorul de trăsături extras anterior într-un vector de dimensiune redusă, compatibil cu codificarea într-un sistem cu 4 qubiți. Valorile rezultate sunt normalizate în intervalul $[-\frac{\pi}{2}, \frac{\pi}{2}]$ și sunt utilizate mai departe în rotațiile cuantice în jurul axei alese.

- Circuitul cuantic variațional care reprezintă componenta centrală a rețelei, este implementat pe un simulator de qubiți (PennyLane) și constă într-un circuit cu 4 qubiți și 10 straturi variaționale, fiecare strat fiind compus dintr-o rețea de porți CNOT pentru generarea entanglementului și porți parametrizate R_Y care realizează rotații în jurul axei Y. Parametrii acestor porți sunt antrenabili și reprezintă un analog al ponderilor din rețelele neuronale clasice.

- Stratul de postprocesare clasic care primește vectorul de ieșire cuantic și îl transformă printr-un strat complet conectat într-un vector de dimensiune egală cu numărul de clase pentru sarcina de clasificare. Acest strat este responsabil de generarea predicției finale a modelului.

Acest model a fost antrenat folosind funcția de pierdere *CrossEntropyLoss*, iar ca optimizator s-a utilizat Adam. Performanțele au fost evaluate periodic, însă în ciuda stabilității rețelei, modelul a prezentat o sensibilitate mai mare la dezechilibrele din setul de date și o capacitate redusă de generalizare față de ViT, mai ales pentru clasele subreprezentate.

În contrast, modelul ViT, bazat pe arhitectura ViTForImageClassification, s-a dovedit a fi mai performant atât în ceea ce privește acuratețea pe setul de validare, cât și în capacitatea sa de a capta relații globale din imagine – un aspect crucial în detecția trăsăturilor subtile din imagini.

Implementarea arhitecturii Vision Transformer în cadrul proiectului a presupus integrarea modelului ViT într-un flux de lucru complet, care a inclus:

- Preluarea și transformarea imaginilor;
- Inițializarea și ajustarea modelului;
- Antrenarea;
- Validarea rezultatelor.

Modelul a fost importat din biblioteca *transformers* folosind clasa *ViTForImageClassification.* Acesta a fost inițializat cu un număr de ieșiri egal cu 5 (corespunzător celor cinci clase posibile). Stratul de clasificare a fost înlocuit pentru a corespunde acestor cerințe, iar restul modelului a fost păstrat pre-antrenat.

4.3 Baza de date

După cum am menționat anterior, a fost folosit atât setul de date ales inițial, intitulat „Diabetic Retinopathy 2019 - resized 224x224”, cât și cel de-al doilea set de date, IDRiD (Indian Diabetic Retinopathy Image Dataset).

În ceea ce privește primul set de date, acesta conține un total de aproximativ 3662 imagini pre-procesate de dimensiune 224 x 224 pixeli. Adnotările sunt furnizate sub forma unui fișier CSV, unde fiecare linie conține numele imaginii și eticheta corespunzătoare, aferentă clasei în care imaginea de fund de ochi este încadrată.

Citirea fișierului și încărcarea imaginilor a fost realizată printr-o clasă personalizată ***RetinopathyDataset***, care extinde clasa *torch.utils.data.Dataset* și permite iterarea prin întregul set de imagini și returnarea perechii (imagine, etichetă). Imaginile sunt stocate într-un singur director (all_images) iar fișierul train.csv conține numele fișierelor și clasa asociată. Acest format simplificat a permis o implementare rapidă și eficientă.

Totuși, pentru o mai bună performanță și generalizare, setul a fost împărțit în 80% date de antrenare și 20% date de validare, folosind funcția *random_split* din PyTorch. Transformările aplicate imaginilor includ augmentări precum rotație aleatorie și flip orizontal, convertirea la tensor și normalizarea. Aceste transformări sunt diferite pentru datele de antrenare și cele de validare pentru a asigura diversitate în timpul învățării și stabilitate în timpul testării.

În vederea automatizării procesului de diagnosticare, clasificarea corectă a imaginilor a reprezentat un pas esențial. Setul de date utilizat etichetează fiecare imagine pe baza unui sistem standardizat care delimitează cele cinci stadii distincte ale bolii:

- **Clasa 0 – Fără retinopatie diabetică (No DR)**

În această categorie sunt incluse imaginile care nu prezintă semne evidente ale bolii. Retina apare intactă, fără microanevrisme, hemoragii sau alte leziuni. Această clasă este importantă pentru stabilirea unui punct de referință în antrenarea modelului.

- **Clasa 1 – Retinopatie diabetică ușoară (Mild DR)**

Se observă primele modificări patologice, în special prezența câtorva microanevrisme (dilatații ale capilarelor retiniene). Este o etapă incipientă, adesea asimptomatică, dar critică pentru depistarea precoce a bolii.

- **Clasa 2 – Retinopatie diabetică moderată (Moderate DR)**

  În acest stadiu apar multiple microanevrisme și hemoragii punctiforme. Pot fi prezente și exsudate moi sau dure. Retina începe să prezinte afecțiuni semnificative, iar riscul de dezvoltare a altor forme severe este crescut.

- **Clasa 3 – Retinopatie diabetică severă (Severe DR)**

  Se caracterizează printr-un număr mare de leziuni retiniene, cu microanevrisme, hemoragii difuze și anomalii vasculare extinse. De regulă, este afectată o mare parte din retina, iar pacientul necesită monitorizare medicală strictă.

- **Clasa 4 – Retinopatie diabetică proliferativă (Proliferative DR)**

  Este stadiul cel mai avansat și periculos, caracterizat prin neovascularizație (formarea de vase sanguine anormale), hemoragii vitreene și risc crescut de dezlipire de retină. Această etapă poate duce la pierderea permanentă a vederii dacă nu este tratată rapid.

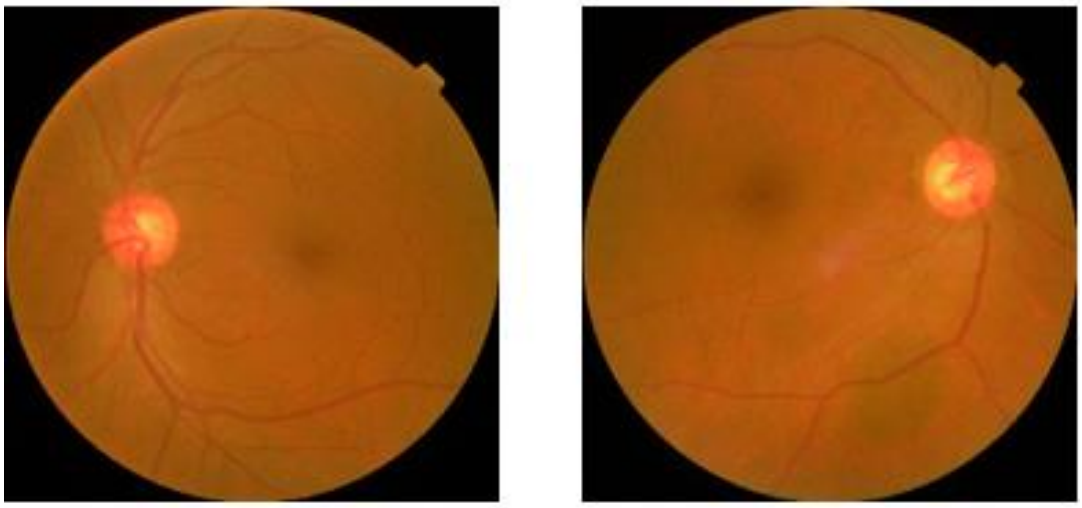

Figura 3.5: Imagine oculară reprezentativă clasei 0 - Diabetic Retinopathy

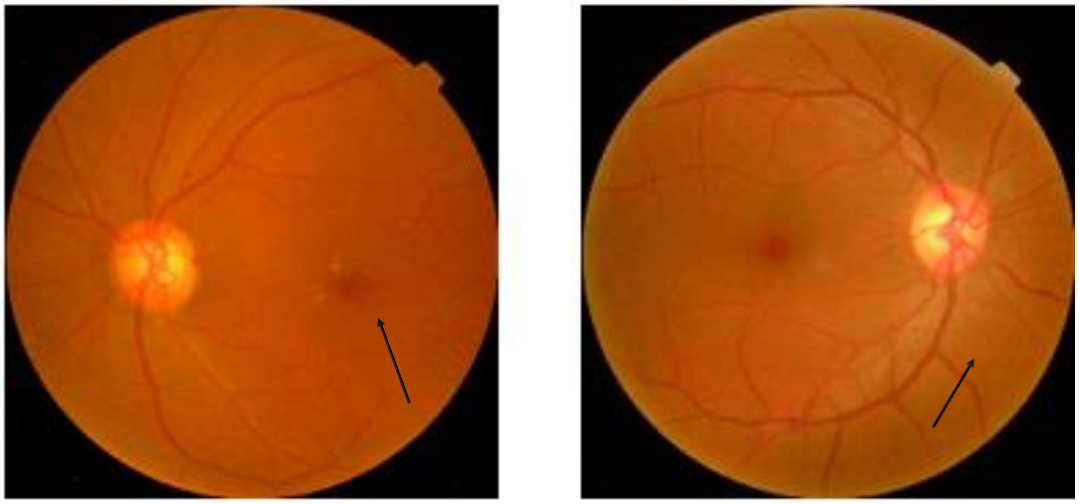

Figura 3.6: Imagine oculară reprezentativă clasei 1 - Diabetic Retinopathy

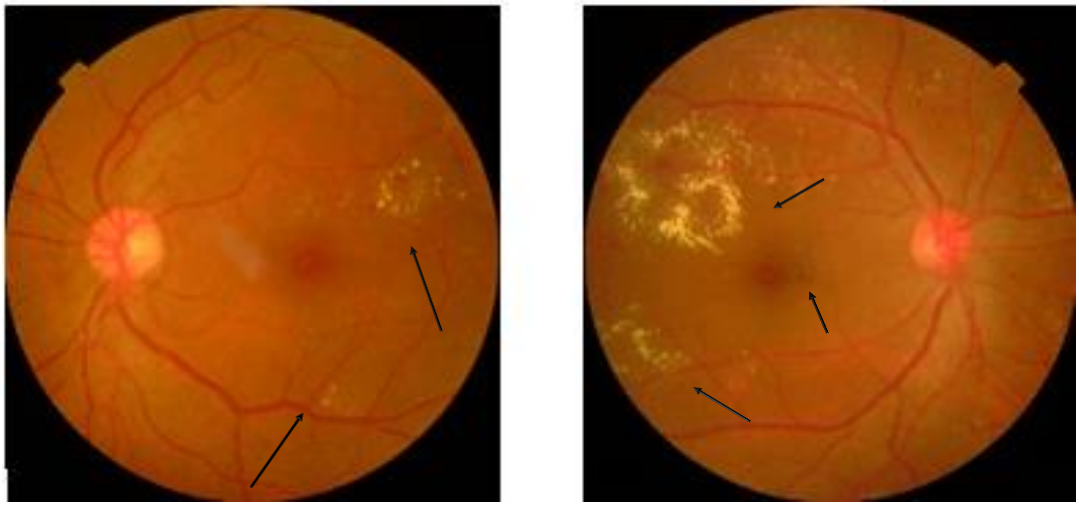

Figura 3.7: Imagine oculară reprezentativă clasei 2 - Diabetic Retinopathy

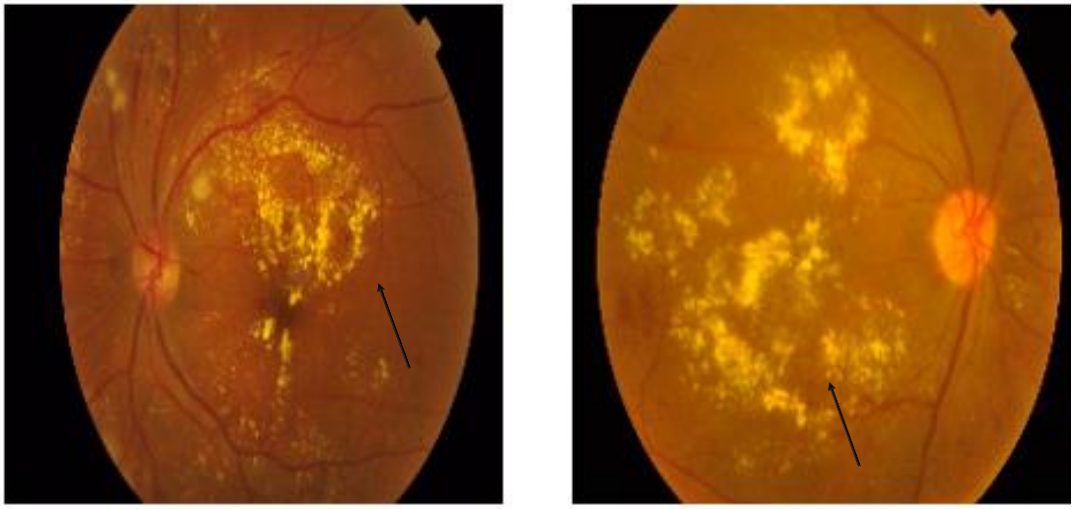

Figura 3.8: Imagine oculară reprezentativă clasei 3 - Diabetic Retinopathy

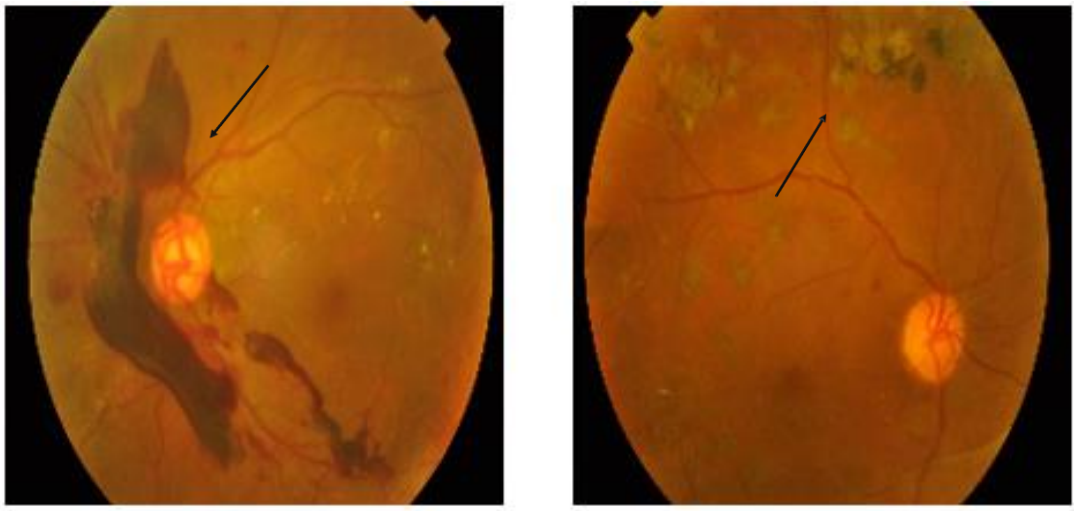

Figura 3.9: Imagine oculară reprezentativă clasei 4 - Diabetic Retinopathy

Pe de altă parte, setul de date IDRiD (Indian Diabetic Retinopathy Image Dataset) conține un total de 516 imagini color, provenite dintr-un context clinic real cu o rezoluție înaltă de aproximativ 4288 × 2848 pixeli. Aceste imagini sunt distribuite în trei subseturi distincte, fiecare corespondent unui task specific:

- Clasificarea severității retinopatiei diabetice;
- Segmentarea leziunilor patologice;
- Segmentarea structurilor anatomice ale fundului de ochi.

Întrucât lucrarea de față viza doar clasificarea severității retinopatiei diabetice, au fost luate în consideroare numai documentele în acest scop. Astfel, imaginile au fost preluate din directoarele „*Training Set”* și *„Testing Set”,* aflate în structura folderului „*B. Disease Grading*”, iar etichetele

corespunzătoare fiecărei imagini au fost extrase din fișierele CSV „*IDRiD_Disease Grading_Training Labels.csv”* și „ *IDRiD_Disease Grading_Testing Labels.csv”,* situate în directorul „*2. Groundtruths*”.

Aceste fișiere conțin, pentru fiecare imagine, numele fișierului și nivelul de severitate al bolii, codificat de la 0 (absența bolii) la 4 (retinopatie proliferativă). S-a definit o clasă personalizată care permite încărcarea perechilor imagine-eticheta, aplicând totodată transformări precum: redimensionarea uniformă la 224×224 pixeli, augmentări precum rotiri, scalări și simetrii orizontale, urmate de o normalizare standard.

În continuare sunt prezentate câteva exemple extrase din subseturile setului de date IDRiD. Fiecare imagine reprezintă o retinografie color, cu rezoluție ridicată, provenită dintr-un context clinic real, iar alăturarea etichetei corespunzătoare indică nivelul de severitate al retinopatiei, conform sistemului de clasificare pe cele cinci stadii ale bolii.

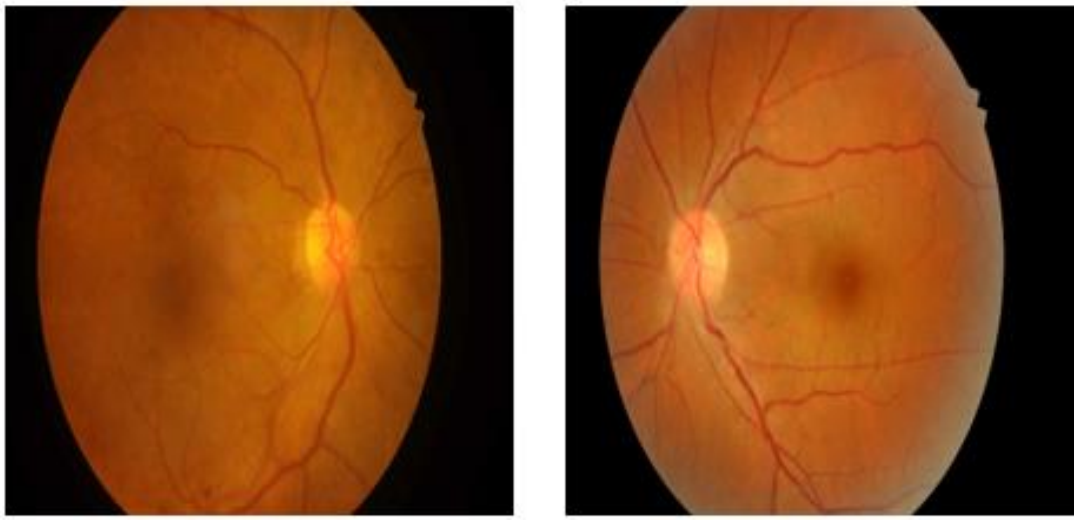

Figura 3.10: Imagine oculară reprezentativă clasei 0 – IDRiD

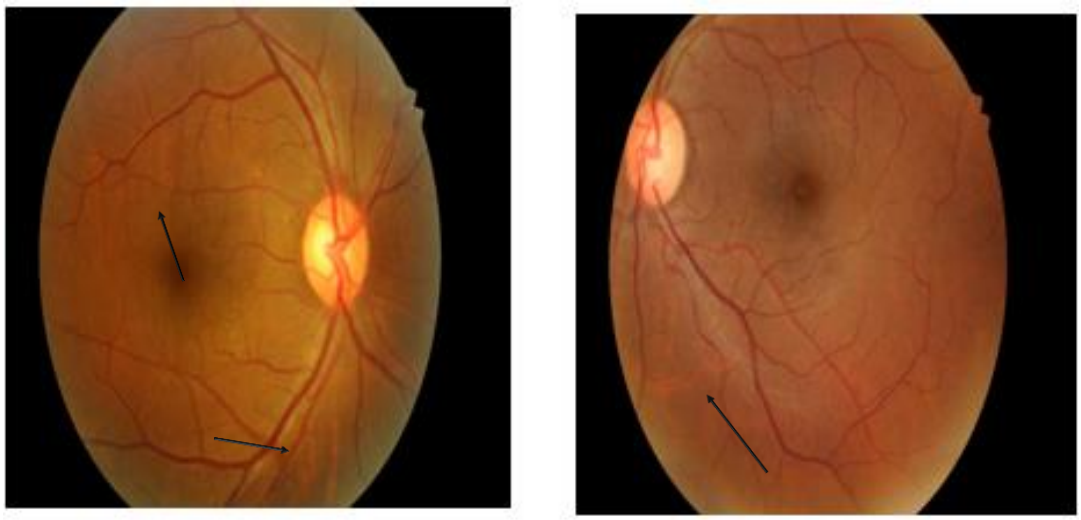

Figura 3.11: Imagine oculară reprezentativă clasei 1 – IDRiD

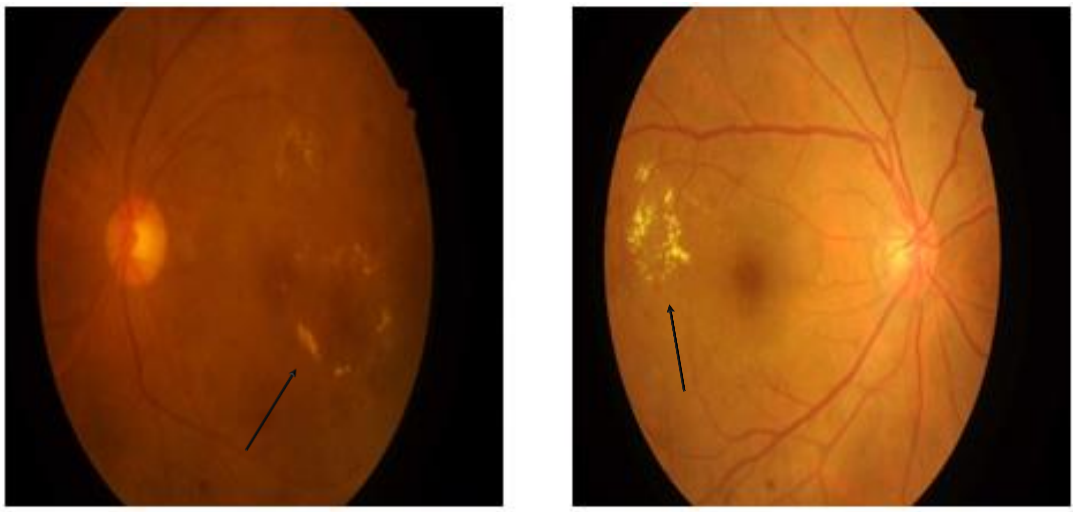

Figura 3.12: Imagine oculară reprezentativă clasei 2 – IDRiD

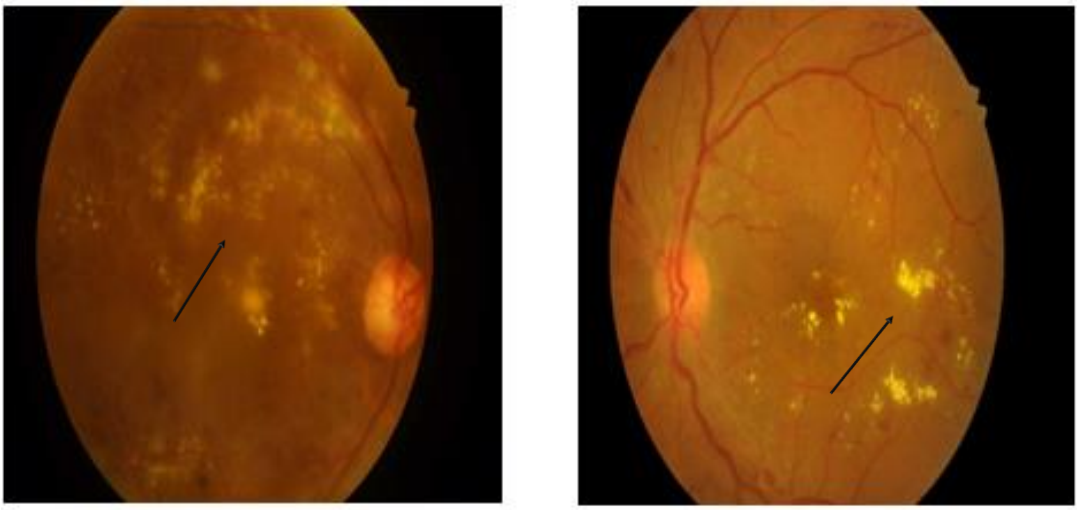

Figura 3.13: Imagine oculară reprezentativă clasei 3 – IDRiD

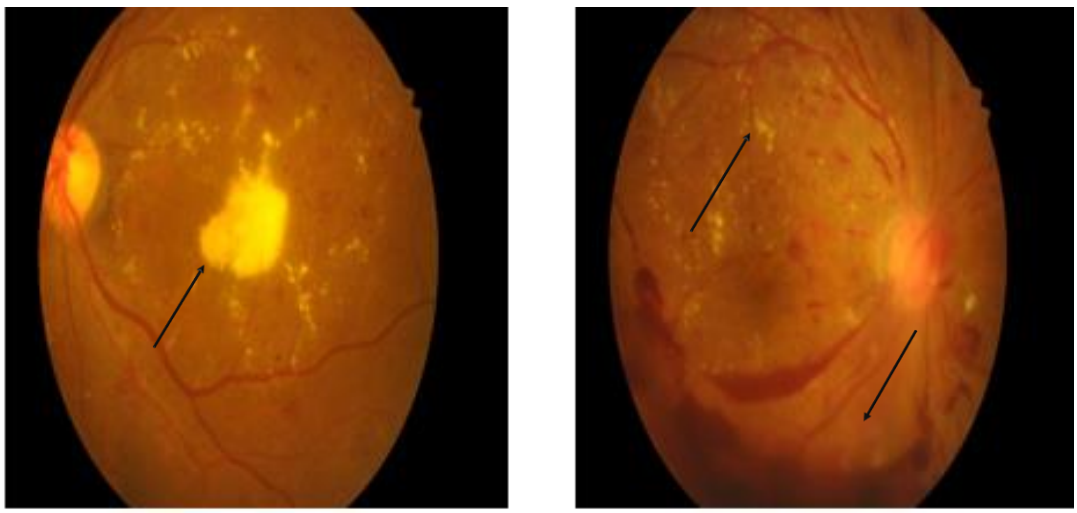

Figura 3.14: Imagine oculară reprezentativă clasei 4 – IDRiD

4.4 Antrenarea rețelei

În vederea evaluării performanței arhitecturii propuse pentru clasificarea imaginilor, a fost realizat procesul de antrenare, ce are ca scop ajustarea parametrilor rețelei, astfel încât aceasta să poată învăța un set de reprezentări semnificative pentru distingerea între cele cinci clase corespunzătoare gradelor de retinopatie. Alegerea arhitecturii și a strategiilor de antrenare a fost fundamentală atât pentru particularitățile setului de date, cât și pe complexitatea sarcinii, urmărindu-se maximizarea acurateței și a capacității de generalizare a modelului. Procesul de antrenare a constat în mai multe componente esențiale, pe care le voi detalia în continuare:

4.4.1 Configurarea setului de date

Procesul de antrenare a fost realizat pe un set de date ce conține imagini ce reflectă diferite grade ale retinopatiei. Setul de date a fost împărțit în două subseturi: unul de antrenament și unul de validare. Împărțirea a fost realizată într-un raport de 80% pentru antrenament și 20% pentru validare, asigurând astfel o utilizare eficientă a datelor disponibile și reducând riscul de suprasolicitare a modelului. Fiecare subset a fost procesat folosind tehnici de augmentare a datelor pentru a mări diversitatea setului de antrenament și a îmbunătăți capacitatea modelului de a generaliza. De asemenea, au fost aplicate transformări precum normalizarea pixelilor pentru a îmbunătăți convergența în timpul antrenării.

4.4.2 Arhitectura DressedQuantumNet

Arhitectura DressedQuantumNet reprezintă o rețea hibridă ce combină prelucrarea clasică a caracteristicilor cu un circuit cuantic variațional. Această abordare a permis integrarea unui număr redus de qubiți, care procesează informația sub forma rotațiilor și entanglementului. În cadrul lucrării, DressedQuantumNet a fost folosită ca modul de decizie final, înlocuind complet stratul de clasificare clasic, iar prin interacțiunea dintre componentele clasice și cuantice, rețeaua a reușit să exploateze trăsături relevante din imagini într-un mod diferit față de metodele pur clasice. Această arhitectură este promițătoare mai ales în contexte în care datele sunt limitate, iar integrarea unui spațiu de reprezentare cuantic poate aduce o mai bună separabilitate între clase.

4.4.3 Arhitectura ViT

Arhitectura ViT se bazează pe un mecanism de atenție globală, care permite modelului să capteze relațiile dintre toate regiunile unei imagini, indiferent de distanța lor spațială. Această caracteristică s-a dovedit esențială în cadrul lucrării, întrucât distribuția leziunilor și a contextului general al imaginilor sunt factori cheie în stabilirea unui diagnostic corect. Prin urmare, ViT a fost capabil să abordeze eficient

complexitatea imaginilor ce ilustrau diferite stadii ale retinopatiei diabetice. De asemenea, arhitectura sa scalabilă a permis antrenarea pe seturi mari de date, ceea ce a adus un avantaj semnificativ în procesarea unui volum mare de imagini, esențial pentru aplicarea în domeniul medical, unde seturile de date pot fi mari și variate.

4.4.4 Implementarea modelului

Modelul hibrid a fost implementat pornind de la o arhitectură clasică ResNet50, ale cărei straturi finale au fost înlocuite cu un lanț de straturi complet conectate și o rețea cuantică variațională definită în PennyLane, care reduc treptat dimensionalitatea trăsăturilor extrase (de la 2048 la 8). Aceste trăsături compacte sunt ulterior prelucrate de modulul *DressedQuantumNet*, ce integrează un circuit cuantic variațional format din patru qubiți și zece straturi de rotații și entanglement.

Datele au fost mapate în spațiul cuantic prin rotații Ry, urmate de straturi de entanglement implementate cu porți CNOT aplicate în mod intercalat. Ieșirea circuitului constă în valorile de expectație ale operatorului Pauli-Z pentru fiecare qubit, care sunt apoi utilizate pentru clasificare printr-un strat final complet conectat. Parametrii circuitului cuantic (rotațiile variaționale) sunt învățați împreună cu restul rețelei, folosind un mecanism de autodiferențiere compatibil cu PyTorch.
Această abordare hibridă a permis integrarea avantajelor rețelelor pre-antrenate clasice cu flexibilitatea și expresivitatea circuitelor cuantice, favorizând în același timp un design modular și scalabil.

Modelul ViT a fost implementat folosind biblioteca *transformers*, pe baza modelului pre-antrenat ViTForImageClassification. Pentru a-l adapta la cerințele sarcinii noastre, stratul de iesire a fost modificat pentru a corespunde celor 5 clase ale setului de date ales. În prima fază, parametrii rețelei pre-antrenate au fost înghețați pentru a preveni ajustarea acestora pe parcursul antrenării, doar stratul de ieșire fiind lăsat să învețe din datele dispuse. Această abordare a favorizat o convergență mai rapidă, reducând timpul necesar antrenării și prevenind overfitting-ul pe un set de date relativ mic.

4.4.5 Configurarea funcției de pierdere și a optimizatorului

În scopul optimizării procesului de antrenament și pentru a compensa dezechilibrul dintre clase, am utilizat funcția de pierdere *CrossEntropyLoss*, la care am aplicat ponderi pentru fiecare clasă, calculate pe baza frecvenței lor în setul de date prin funcția *compute_class_weights*. Această funcție calculează ponderile inverse pentru fiecare clasă pe baza frecvenței lor în setul de date. Mai precis, funcția calculează câte instanțe există pentru fiecare clasă și le inversează astfel încât modelul să acorde mai multă atenție claselor mai puțin frecvente și să nu fie afectat de clasele dominante, prevenind astfel supraadaptarea modelului.

În cadrul antrenamentului modelului, am optat pentru utilizarea optimizatorului Adam, datorită caracteristicilor sale avansate care combină avantajele algoritmilor de tip momentum și rata de învățare adaptivă. Algoritmul calculează și utilizează atât media ponderată a gradientului, cât și media pătratelor gradientelor, ajustând automat ratele de învățare pentru fiecare parametru al modelului. Acest lucru asigură o convergență rapidă și o stabilitate crescută, mai ales în cadrul rețelelor complexe, cum este cazul arhitecturii ViT. Mai exact, optimizatorul Adam a fost aplicat asupra parametrilor stratului de ieșire al rețelei, iar actualizarea acestora s-a realizat pe baza gradienților. Această alegere a fost motivată de avantajelor pe care optimizatorul Adam le prezintă în gestionarea seturilor mari de date și a arhitecturilor complexe, fiind mai puțin sensibil la inițializările parametrilor și capabil să depășească minime locale în procesul de optimizare.

4.4.6 Antrenarea modelului

Antrenarea modelului a avut loc pe o perioadă de 50 de epoci. În cadrul fiecărei epoci, modelul a fost evaluat atât pe setul de antrenament, cât și pe cel de validare, pentru a se asigura că nu are loc fenomentul de overfitting. În timpul antrenării, s-a salvat periodic cel mai bun model, aceasta fiind oprită atunci când nu s-au observat îmbunătățiri semnificative în performanța modelului.

4.5 Evaluarea performanței

După finalizarea procesului de antrenare, a urmat evaluarea performanței modelului pentru a înțelege cât de bine a învățat modelul folosit și cât de bine generalizează pe date noi. Pentru a facilita analiza, s-au generat tabele și grafice comparative care evidențiază evoluția pe parcursul celor 50 de epoci. Aceastea oferă o perspectivă clară asupra modului în care modelul învață, semnalând potențialele probleme precum overfitting-ul sau lipsa de convergență.

5. Rezultate

În această lucrare au fost dezvoltate și comparate mai multe configurații de model, obținute prin combinarea a două arhitecturi: ResNet50 integrată cu circuitul cuantic DressedQuantumNet și modelul pre-antrenat ViT, ambele fiind aplicate pe două seturi de date diferite: IDRiD și Diabetic Retinopathy (APTOS resized). Scopul acestei comparații a fost acela de a observa atât influența arhitecturii asupra performanței de clasificare, cât și comportamentul modelelor în funcție de volumul și distribuția datelor de antrenare.

Toate modelele au fost configurate spre antrenare pe un număr de 50 de epoci, folosind aceeași configurație de optimizare și același set de transformări pentru preprocesarea imaginilor: redimensionare la 224x224, normalizare și augmentarea datelor de antrenare. Evaluarea a fost realizată pe subsetul de test, fără augmentări suplimentare, fiind într-un final măsurată acuratețea.

5.1 Performanța obținută

În urma experimentelor efectuate, s-au analizat performanțele obținute pentru cele două modele, antrenate pe cele două seturi de date distincte, unde am variat hiperparametrii de antrenare, în special rata de învățare și dimensiunea batch-ului și s-au testat astfel două valori pentru rata de învățare (0.001 și 0.0001) și două dimensiuni ale batch-ului (16 și 32).

Rezultatele au evidențiat faptul că arhitectura ViT a oferit în mod constant performanțe superioare, obținând o acuratețe maximă de 80% pe setul IDRiD, spre comparație de modelul cuantic ResNet50 care a înregistrat o acuratețe semnificativ mai mică, ceea ce indică o capacitate mai redusă de învățare în acest context. Pe setul Diabetic Retinopathy, ViT a obținut o acuratețe de 78.9%, iar modelul cuantic ResNet a atins 63.1%, menținându-se astfel o diferență clară în favoarea arhitecturii ViT. Toate imaginile au fost redimensionate la 224 × 224 pixeli înainte de a fi procesate de rețea, eliminând astfel impactul diferențelor de rezoluție inițială. Astfel, performanțele obținute reflectă în principal calitatea datelor, coerența etichetării și caracteristicile arhitecturii, și nu diferențe de dimensiune sau detalii tehnice ale imaginilor.

În ceea ce privește influența setului de date, se observă că setul Diabetic Retinopathy a permis obținerea unor rezultate competitive pentru ambele arhitecturi, datorită volumului mai mare de date disponibile. Totuși, cea mai mare performanță a fost obținută pe IDRiD, sugerând că un set de date bine etichetat și omogen poate compensa lipsa de volum și poate conduce la o învățare mai eficientă, în special în cazul modelelor avansate precum ViT.

În concluzie, Vision Transformer s-a dovedit a fi cea mai robustă și performantă arhitectură din toate configurațiile testate. Performanțele sale ridicate pentru ambele seturi de date, sub diferite combinații de hiperparametri, validează potențialul acestor rețele în aplicațiile din domeniul medical. Totodată, rezultatele evidențiază importanța selecției adecvate a datelor și a alegerii corecte a parametrilor de antrenare, în funcție de complexitatea arhitecturii și natura setului de date utilizat.

Rezultatele obținute în urma antrenărilor se regăsesc în tabelul de mai jos:

Model	**Set de date**	**Learning Rate**	**Batch size**	**Acuratete**	**Precizie**	**Recall**	**F1**
Resnet	IDRiD	0.001	16	36.58%	0.20	0.08	0.11
ViT	IDRiD	0.001	16	63.65%	0.40	0.30	0.34
Resnet	Diabetic Retinopathy	0.001	16	62.63%	0.46	0.21	0.28
ViT	Diabetic Retinopathy	0.001	16	63.15%	0.77	0.60	0.67
Resnet	IDRiD	0.001	32	20.00%	0.20	0.08	0.11
ViT	IDRiD	0.001	32	80.00%	0.80	0.89	0.84
Resnet	Diabetic Retinopathy	0.001	32	47.36%	0.31	0.18	0.23
ViT	Diabetic Retinopathy	0.001	32	57.89%	0.69	0.47	0.56
Resnet	IDRiD	0.0001	16	30.00%	0.31	0.29	0.30
ViT	IDRiD	0.0001	16	40.00%	0.40	0.38	0.39
Resnet	Diabetic Retinopathy	0.0001	16	63.75%	0.77	0.60	0.67
ViT	Diabetic Retinopathy	0.0001	16	78.94%	0.46	0.44	0.45
Resnet	IDRiD	0.0001	32	20.00%	0	0	0

Tabel 4.1. Rezultate obținute în urma antrenării pe 50 de epoci

În cele ce urmează se pot observa câteva grafice care evidențiază evoluția pe parcursul antrenării a celor mai bune rezultate obținute. Pentru toate experimentele, antrenarea modelelor a fost configurată să se desfășoare pe parcursul a 50 de epoci. Cu toate acestea, în majoritatea cazurilor, procesul de antrenare a fost întrerupt automat înainte de atingerea numărului maxim de epoci, ca urmare a utilizării unui mecanism de Early Stopping. Această tehnică are scopul de a preveni supraînvățarea și presupune oprirea procesului de antrenare atunci când nu mai este observată o îmbunătățire semnificativă a performanței pe setul de validare după un anumit număr de epoci consecutive.

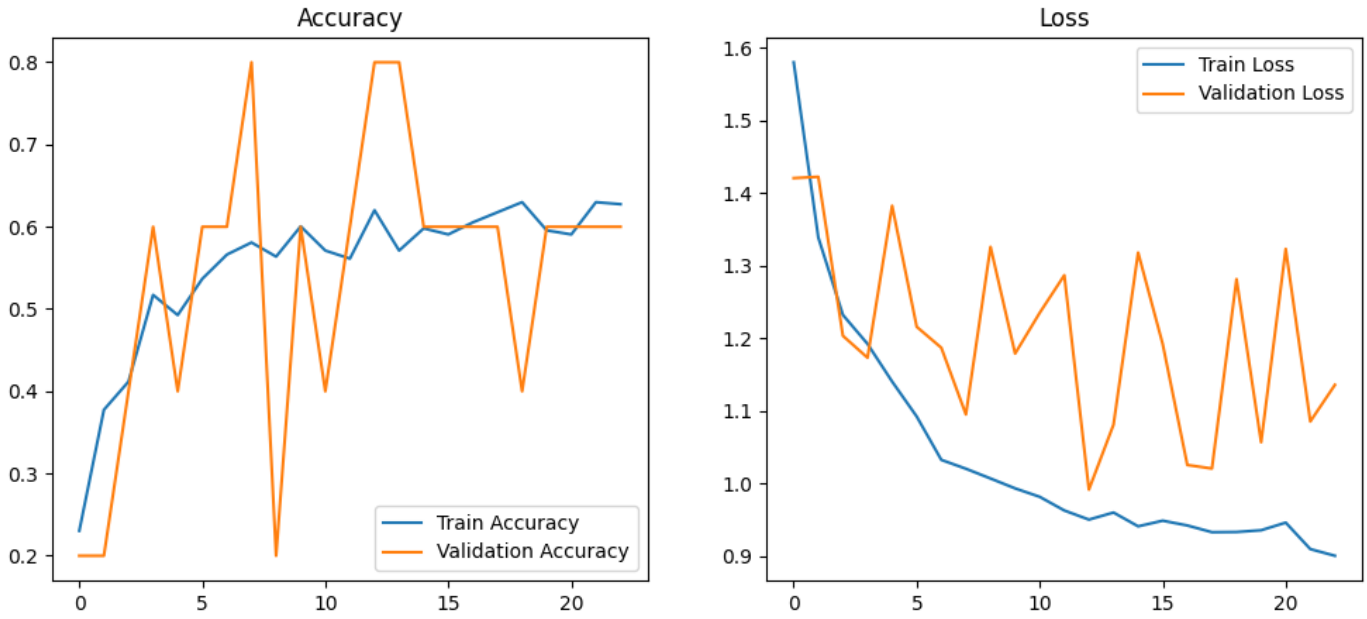

Figura 4.1: Rezultat obținut pentru modelul ViT antrenat pe setul de date IDRiD – 80%

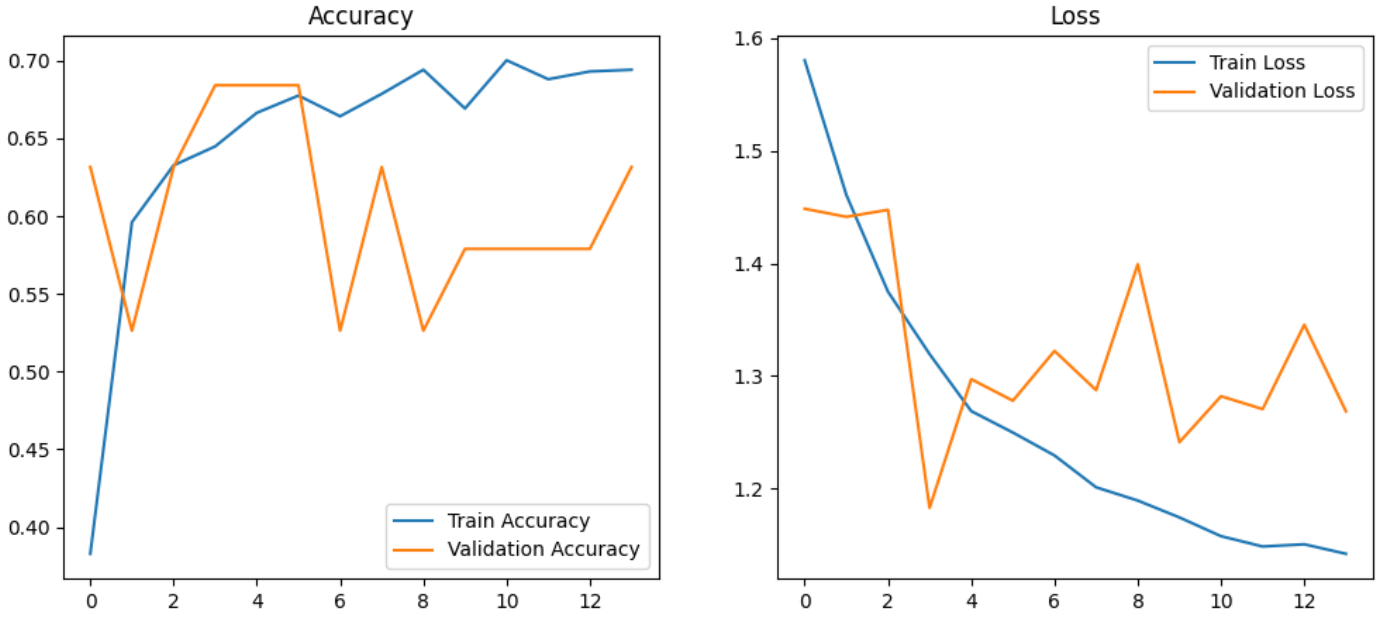

Figura 4.2: Rezultat obținut pentru modelul cuantic ResNet50 antrenat pe setul de date Diabetic Retinopathy 224x224 – 63%

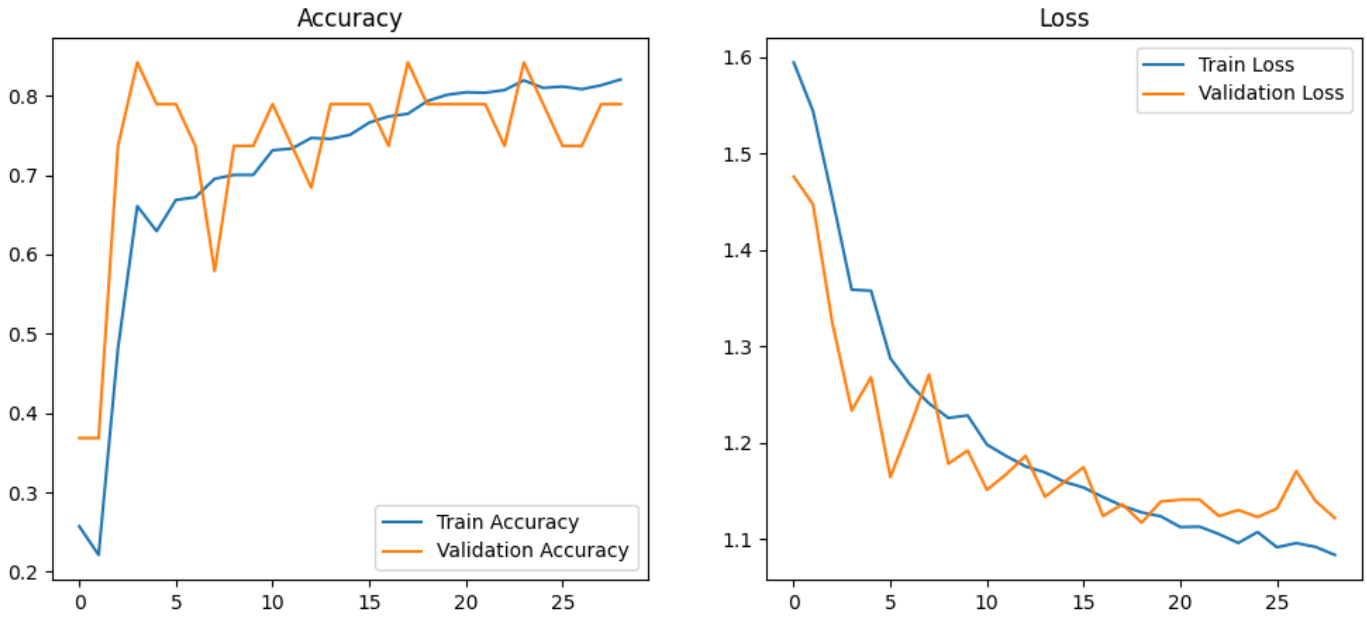

Figura 4.3: Rezultat obținut pentru modelul ViT antrenat pe setul de date Diabetic Retinopathy 224x224 – 78%

Analiza comparativă a celor trei configurații de modele evidențiază diferențe importante în comportamentul arhitecturilor testate. Modelul ViT antrenat pe IDRiD a demonstrat cea mai bună performanță generală, cu un echilibru remarcabil între acuratețe, recall și F1-score, fiind eficient în detectarea corectă a cazurilor de retinopatie severă. În schimb, aplicarea aceluiași model pe setul Diabetic Retinopathy resized a condus la scoruri mult mai scăzute pentru recall și F1, sugerând dificultăți în identificarea claselor minoritare, cel mai probabil datorită dezechilibrului de date. Deși a avut o acuratețe mai modestă, modelul ResNet antrenat pe același set a obținut o precizie mai ridicată, indicând o tendință spre predicții mai sigure, dar mai restrictive. Astfel, fiecare arhitectură prezintă avantaje diferite, iar alegerea finală depinde de obiectivul aplicării: sensibilitate crescută sau minimizarea alarmelor false.

De asemenea, pentru a evalua capacitatea modelului de a distinge corect între diferitele stadii ale retinopatiei diabetice, a fost generată și matricea de confuzie. Aceasta oferă o imagine detaliată asupra distribuției predicțiilor față de etichetele reale, evidențiind atât clasificările corecte, cât și erorile de predicție între clasele apropiate.

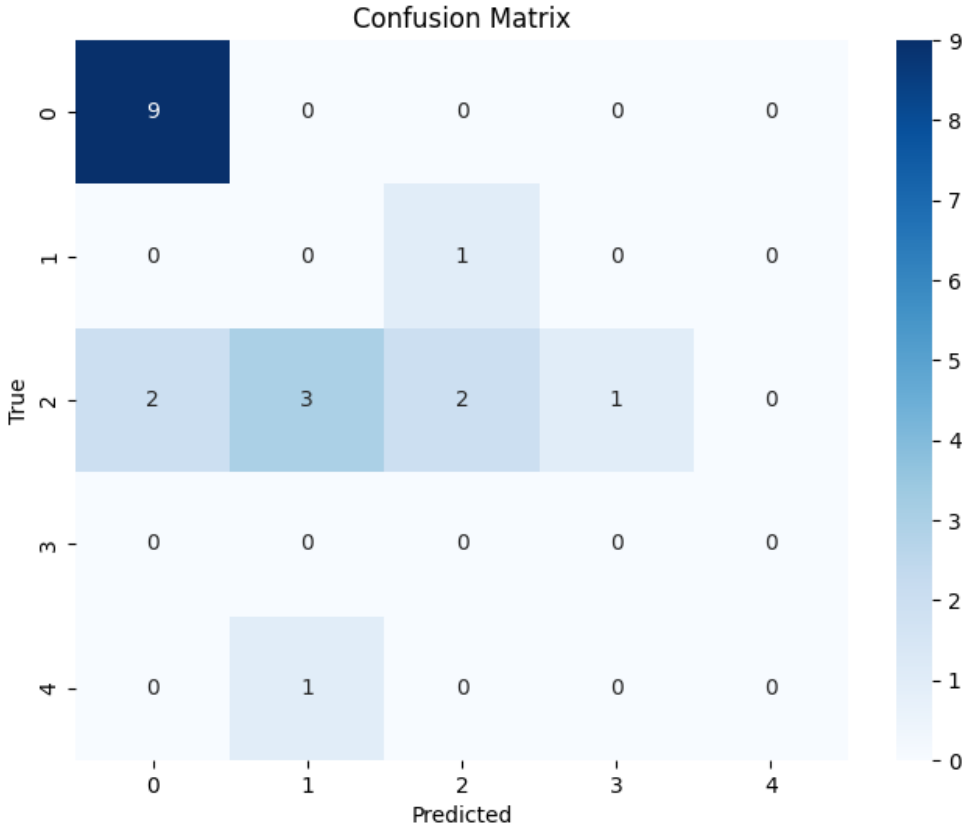

Figura 4.4: Matricea de confuzie pentru modelul ViT antrenat pe setul de date Kaggle Diabetic Retinopathy

Modelul a reușit să clasifice corect toate cele 9 imagini din clasa 0, corespunzătoare cazurilor fără semne de retinopatie, ceea ce sugerează o bună capacitate de recunoaștere imaginilor ce nu prezinta anomalii. În schimb, pentru clasa 2 (retinopatie moderată), performanța a fost semnificativ mai slabă: din 8 exemple, doar 2 au fost clasificate corect, în timp ce restul au fost confundate cu clasele 0, 1 și 3. Acest comportament reflectă dificultatea modelului de a diferenția între stadiile intermediare ale bolii, probabil din cauza similitudinii vizuale dintre imagini sau a distribuției dezechilibrate a datelor.

Clasele 1 și 4 au fost de asemenea slab reprezentate în setul de validare (câte un singur exemplu fiecare), iar modelul nu a reușit să le clasifice corect: imaginea din clasa 1 a fost etichetată ca aparținând clasei 2, iar cea din clasa 4 a fost confundată cu clasa 1. De remarcat este și absența completă a clasei 3 în setul de validare, ceea ce a împiedicat evaluarea performanței modelului pentru această categorie.

În ceea ce privește modelul cuantic ResNet50, matricea de confuzie evidențiază o capacitate bună de clasificare, în special pentru cazurile fără retinopatie, dar și pentru cele de severitate medie.

Modelul nu a reușit să clasifice corect niciun exemplu din clasele 1 și 4, fiecare dintre acestea având o singură imagine în setul de validare, ambele fiind confundate cu clasa 2. Aceste erori pot fi puse pe seama dezechilibrului între clase și a numărului redus de exemple din setul de testare. Clasa 3 nu a fost prezentă în setul de validare, astfel că performanța modelului pentru aceasta nu a putut fi evaluată.

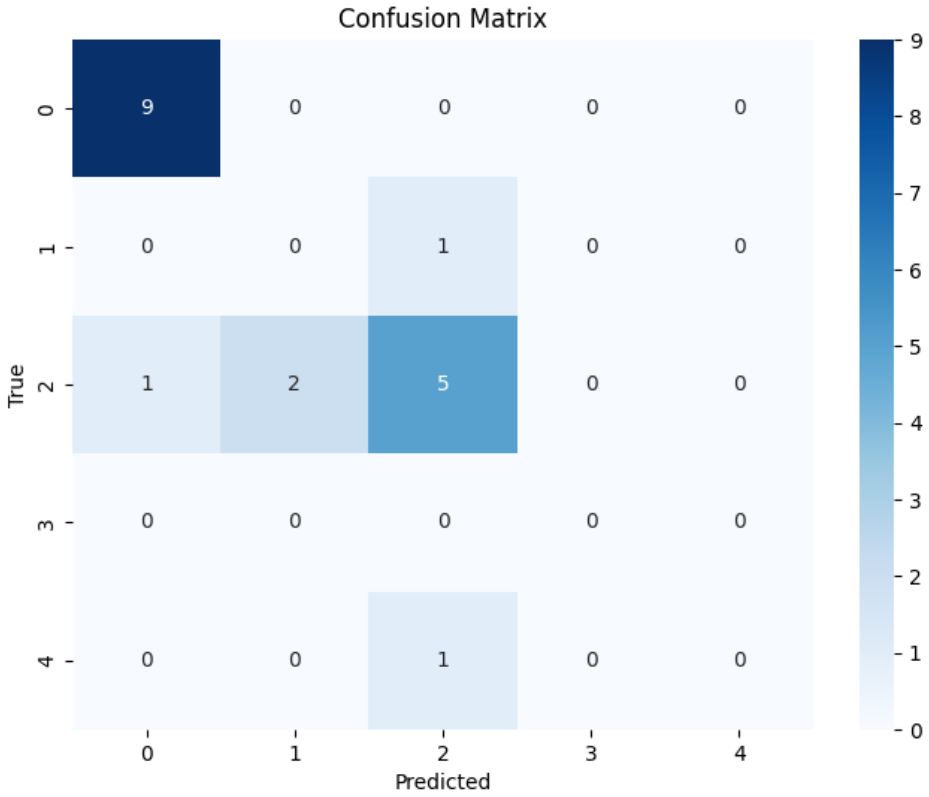

Figura 4.5:. Matricea de confuzie pentru modelul cuantic ResNet antrenat pe setul de date Kaggle Diabetic Retinopathy

Prin urmare, rezultatele evidențiază faptul că modelul ViT prezintă o performanță solidă în identificarea cazurilor fără retinopatie, dar are dificultăți în clasificarea precisă a stadiilor intermediare ale bolii. Totuși, datorită performanței echilibrate și scorului F1 ridicat, modelul ViT antrenat pe IDRiD a fost selectat pentru analiza finală. Acesta oferă o capacitate superioară de generalizare, reușind să identifice corect majoritatea cazurilor pozitive de retinopatie diabetică, fapt esențial într-un context medical, unde sensibilitatea clasificatorului este prioritară.

5.2 Evaluarea vizuală

În cadrul acestui subcapitol sunt prezentate rezultatele vizuale obținute pe imagini din setul de testare, cu scopul de a evidenția modul în care modelul distinge între diferitele stadii ale retinopatiei diabetice. Fiecare imagine este însoțită de eticheta reală și de predicția generată de model, oferind o perspectivă calitativă asupra corectitudinii clasificării.

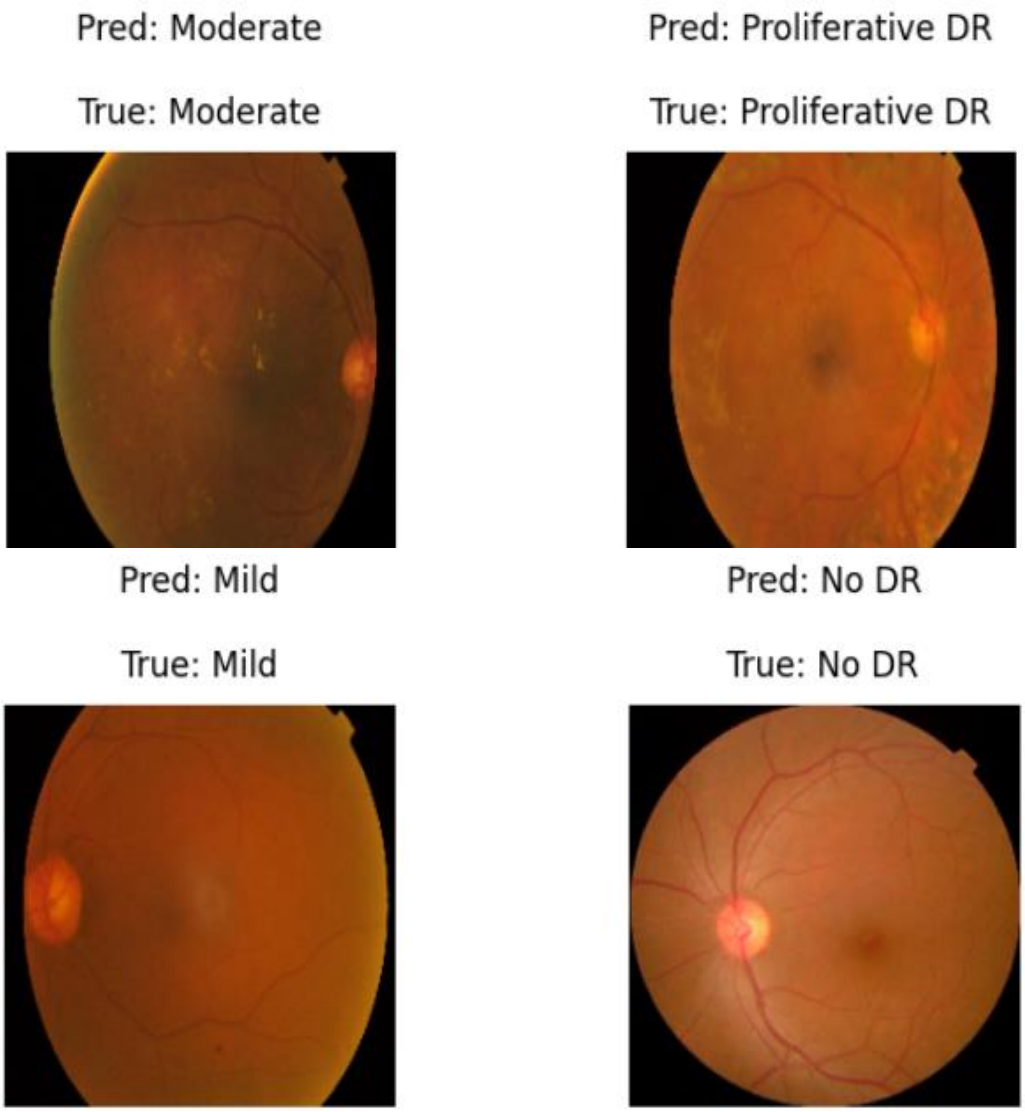

Figura 4.6: Clasificarea corectă în toate cele 4 cazuri prezentate

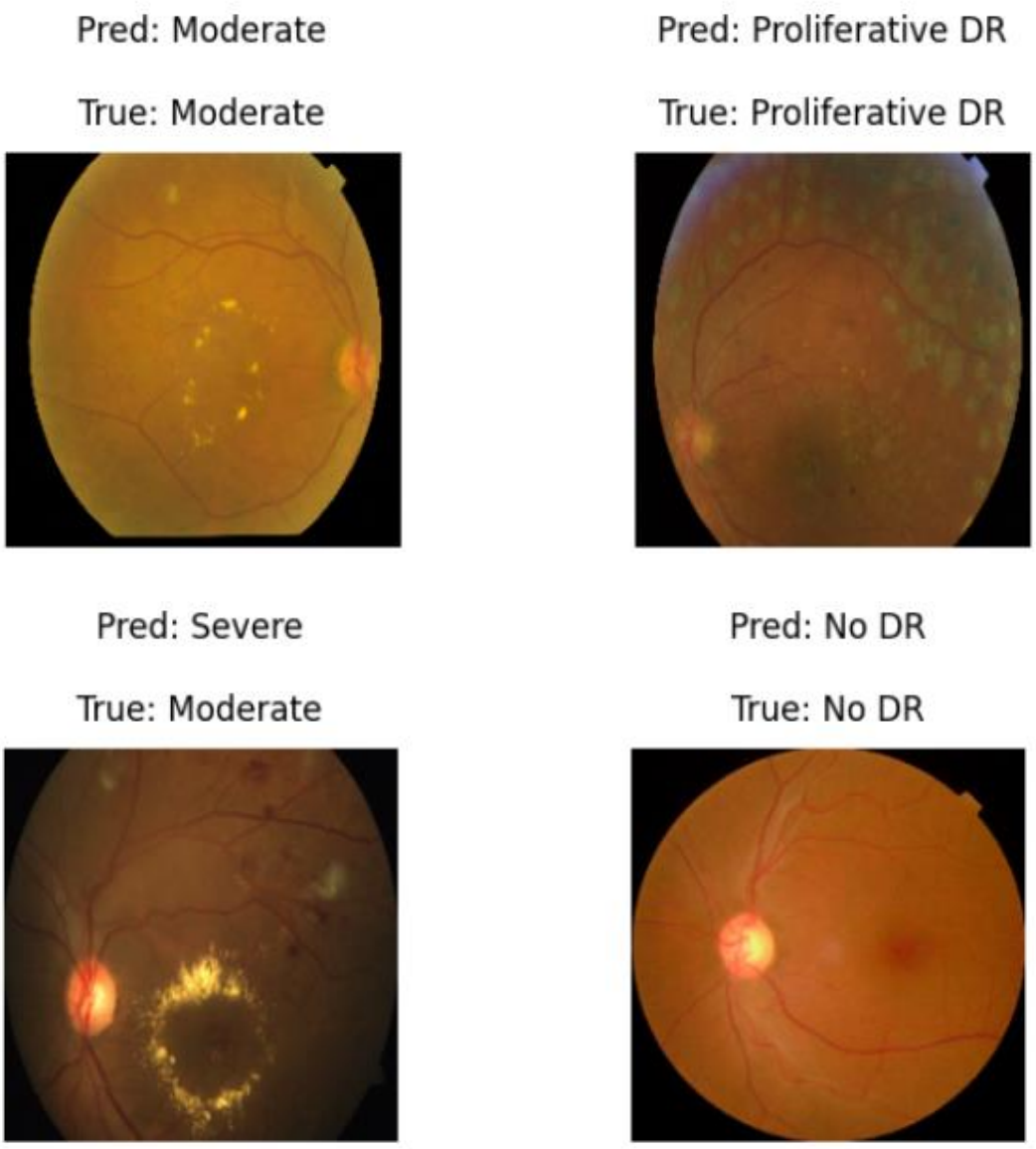

Figura 4.7: Clasificare variată: 3 clasificări corecte și 1 eronată

În figura 4.7 sunt prezentate patru imagini de fund de ochi extrase aleatoriu, fiecare fiind însoțită de eticheta reală și de predicția realizată de model. Trei dintre aceste imagini au fost clasificate corect, în timp ce o singură imagine (colțul stânga-jos) a fost etichetată eronat, modelul indicând un stadiu mai avansat al bolii decât cel reală.

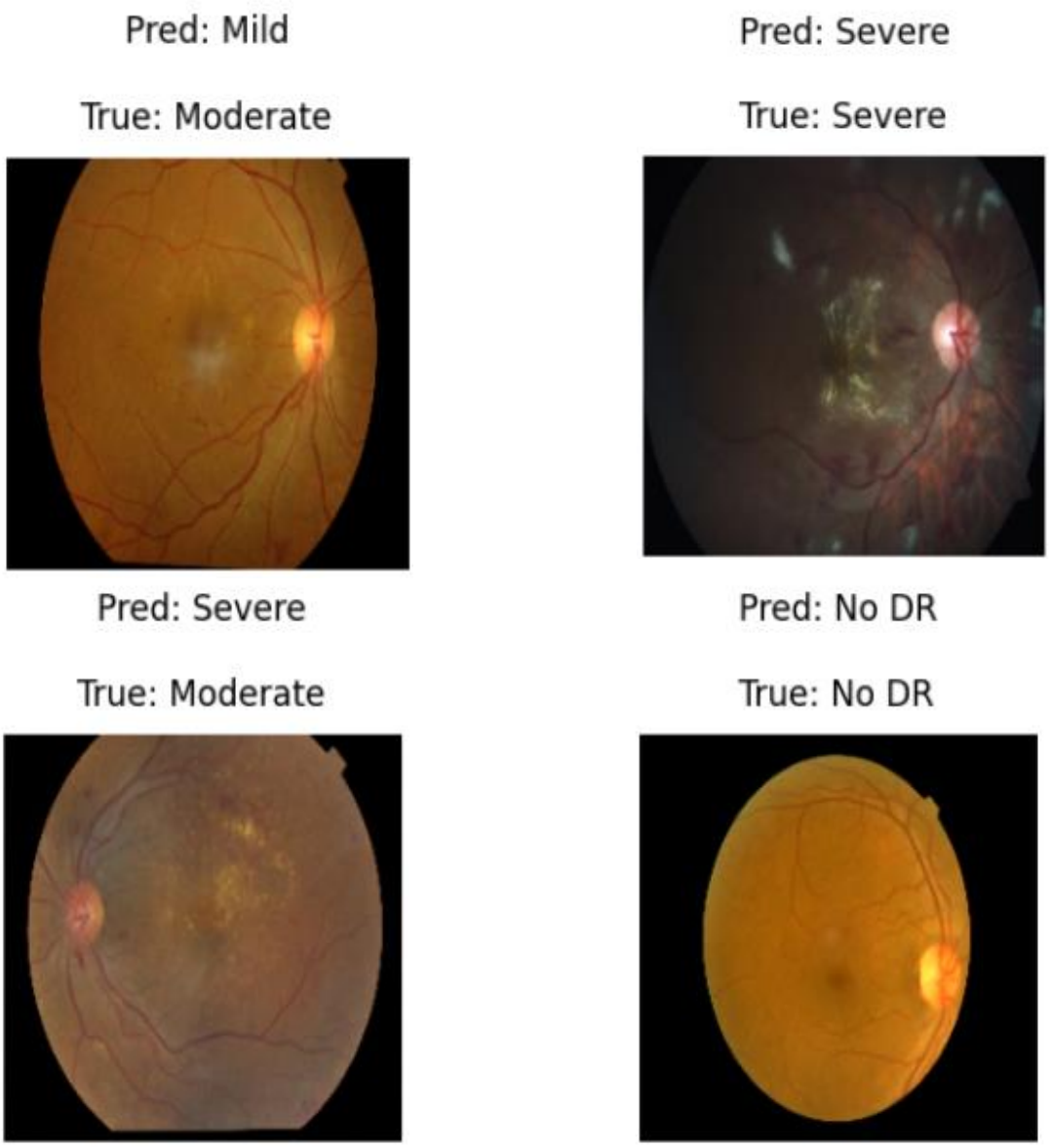

Figura 4.8: Clasificare variată: 2 clasificări corecte și 2 eronate

Rezultatele din cadrul figurii 4.8 sugerează faptul că modelul întâmpină dificultăți în diferențierea stadiilor „Mild” și „Moderate”, respectiv „Moderate” și „Severe”. Aceste confuzii apar deoarece granițele vizuale dintre clase sunt subtile, iar diferențele de severitate nu sunt marcate prin trăsături anatomice evidente. Fiind clase învecinate din punct de vedere clinic, ele presupun adesea prezența unor leziuni similare, dar cu intensități sau extinderi diferite, ceea ce poate duce la erori de clasificare chiar și în cazul unui model bine antrenat.

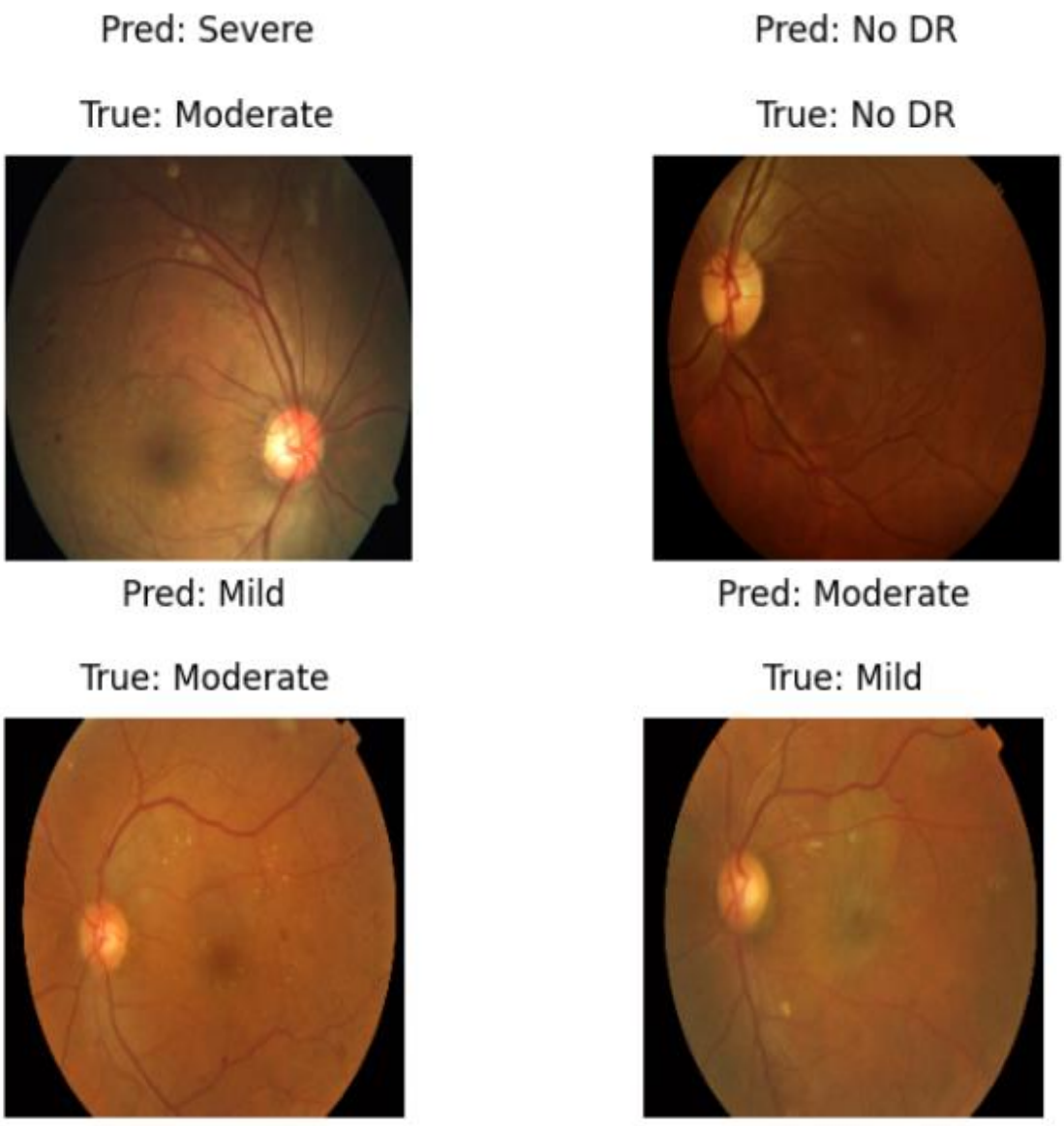

Figura 4.9: Clasificare variată: 1 clasificăre corectă și 3 eronate

În cadrul figurii 4.9, doar unul dintre cele 4 stadii a fost clasificat corect. Celelalte trei cazuri prezendând erori, în special confuzii între stadiile „Mild”, „Moderate” și „Severe”. Astfel de confuzii reflectă dificultatea modelului în a distinge cu precizie între formele intermediare ale retinopatiei diabetice, acolo unde granițele vizuale pot fi subtile, iar caracteristicile patologice se suprapun parțial.

Cu toate acestea, este important de remarcat faptul că, în toate cazurile testate până în acest punct pentru care eticheta reală a fost „No DR” (absența retinopatiei), modelul a oferit o predicție corectă. Acest aspect indică o capacitate bună de identificare a ochiului sănătos și este esențial din perspectivă clinică, întrucât reduce riscul de fals pozitiv pentru pacienții care nu necesită tratament sau urmărire.

Este de menționat faptul că acuratețea ridicată pentru clasa 0 ar putea fi influențată și de distribuția dezechilibrată a datelor din setul de antrenament, în care imaginile etichetate cu „No DR” sunt semnificativ mai numeroase comparativ cu celelalte clase. Acest dezechilibru poate determina modelul să fie mai bine ajustat pe această categorie, crescând astfel probabilitatea de clasificare corectă în cazul ochilor sănătoși.

Acest rezultat se menține chiar și în contextul aplicării unei funcții de pierdere ponderată, menită să penalizeze mai sever erorile asociate claselor sub-reprezentate. Prezența frecventă a imaginilor din clasa „No DR” în timpul antrenării a favorizat, totuși, o generalizare mai stabilă a modelului pentru această categorie. Acest aspect sugerează că expunerea repetată la o clasă dominantă poate avea un impact semnificativ asupra performanței, chiar și în prezența unor mecanisme de compensare a dezechilibrului.

5.3 Comparația cu metedele din literatura de specialitate

Cele mai bune rezultate obținute în cadrul acestei lucrări aparțin modelului ViT antrenat pe setul IDRiD, cu o acuratețe de 80%, urmat de varianta antrenată pe setul de date Diabetic Retinopathy 224x224, o versiune preprocesată (resized) a setului original APTOS 2019 Blindness Detection, cu un rezultat de 78% și ulterior modelul ResNet50 combinat cu rețea cuantică DressedQuantumNet, care a atins 57,89% pe același set. Deși modelul cuantic nu a depășit arhitectura ViT din punct de vedere al acurateței, el rămâne relevant din perspectivă exploratorie.

Mai jos se regăsesc două tabele comparative între modelele propuse în lucrarea de față și alte modele state-of-the-art pentru cele două seturi de date:

Model	**Dataset**	**Acuratețe obținută**
ViT (model propus)	Diabetic Retinopathy 224x224	78.94 %
DQN + ResNet50 (model propus)	Diabetic Retinopathy 224x224	63 %
ViT +Capsule Network	Diabetic Retinopathy 224x224	78.64 %
VQC + ResNet18	APTOS 2019	97 %

Tabel 4.2: Performanța modelelor pe setul de date APTOS 2019 și APTOS 2019 redimensionat

Model	Dataset	Acuratețe obținută
ViT (model propus)	IDRiD	80 %
DQN + ResNet50 (model propus)	IDRiD	62 %
Beta-Rank + ResNet56	IDRiD	80.58 %
RSG-Net (EfficientNet)	IDRiD	86 %

Tabel 4.3: Performanța modelelor pe setul de date IDRiD

Comparația realizată în această secțiune evidențiază performanțele modelelor propuse în raport cu metode consacrate din literatura de specialitate, pe seturile APTOS 2019 (în ambele variante, redimensionat și original) și IDRiD.

Pe setul APTOS 2019 redimensionat la 224×224, modelul ViT propus în această lucrare a obținut o acuratețe de 78.94%, comparabilă cu alte abordări bazate precum ViT + Capsule Network cu o acuratețe de 78.64%.

Pe setul IDRiD, modelul propus a reușit să atingă o acuratețe maximă de 80%, o performanță remarcabilă având în vedere dificultatea ridicată a setului, acesta fiind caracterizat de un volum redus de date și o distribuție dezechilibrată a claselor. Acest rezultat plasează modelul în proximitatea performanțelor obținute de restul metodelor precum Beta-Rank + ResNet110 unde performanțele au atins 81% sau RSG-Net (EfficientNet) cu 86%.

În concluzie, modelele dezvoltate în această lucrare oferă o bază solidă pentru aplicații reale de screening automatizat în oftalmologie, deschizând direcții de cercetare viitoare în optimizarea rețelelor hibride cu componente cuantice și aplicarea acestora pe seturi clinice mai diverse și mai complexe.

6. Concluzii

Așa cum a fost menționat și în introducere, lucrarea de față a avut ca scop dezvoltarea unei metode automate pentru detectarea retinopatiei diabetice, utilizând imagini de fund de ochi și aplicând cele mai noi direcții din domeniul învățării profunde, în special rețele de tip Vision Transformer și arhitecturi hibride cu componente cuantice (DressedQuantumNet).

Pașii principali realizați în cadrul proiectului au fost:

- Studierea fundamentelor învățării automate, a rețelelor neuronale convoluționale clasice și cuantice și a arhitecturilor Transformer aplicate pe imagini medicale;
- Documentarea și selecția seturilor de date publice: Diabetic Retinopathy 224x224, o varianta resized a setului de date APTOS 2019 și IDRiD;
- Implementarea a două modele: unul bazat pe ResNet50 integrat cu un circuit cuantic și unul bazat pe Vision Transformer;
- Preprocesarea imaginilor și configurarea etichetelor pentru clasificarea în cele 5 clase;
- Antrenarea și validarea modelelor pe fiecare set de date, incluzând folosirea de early stopping și testare multiplă pentru diferite batch size-uri și learning rate-uri;
- Realizarea unor evaluări atât cantitative , cât și calitative;
- Compararea performanței modelului propus cu alte metode din literatura de specialitate.

Rezultatele obținute sunt destul de satisfăcătoare, în ideea în care modelul ViT a obținut o acurateță de 78.94% pe setul de date Diabetic Retinopathy 224x224, iar pe IDRiD s-a atins un vârf de 80%, valoare comparabilă cu metode avansate precum Beta-Rank sau RSG-Net, în timp ce modelul ResNet50+DQN a atins valori cuprinse între 62 % și 63%. Astfel, modelul final ales a demonstrat robustețe în detectarea cazurilor de retinopatie diabetică.

Consider că implementarea acestui proiect a fost complexă și valoroasă din punct de vedere tehnic (prin integrarea componentei cuantice într-o arhitectură vizuală modernă), fiind de asemenea și aplicativă, într-un context medical cu impact direct în screeningul bolilor oftalmologice. Proiectul poate sta la baza unor sisteme de diagnosticare asistată, contribuind la eficientizarea procesului de triere în cazul pacienților diabetici.

Totuși, deși performanțele obținute sunt satisfăcătoare, există direcții clare de îmbunătățire care pot crește acuratețea și robustețea modelului propus:

- Pre-antrenarea modelelor pe seturi mai mari de imagini medicale;
- Antrenarea pentru un număr mai mare de epoci și cu o strategie de scheduler adaptiv pentru rata de învățare;

- Extinderea seturilor de date utilizate, prin combinarea mai multor surse sau generarea de imagini sintetice;
- Testarea și integrarea altor arhitecturi hibride;
- Interpretabilitate sporită prin adăugarea de hărți de atenție și analiza deciziilor modelului asupra cazurilor greșit clasificate;
- Evaluarea pe date clinice reale, obținute în colaborare cu instituții medicale, pentru o validare în condiții practice.

În concluzie, scopul lucrării a fost atins: s-a realizat o arhitectură modernă, scalabilă și eficientă pentru detecția retinopatiei diabetice, testată pe două seturi de date publice, cu rezultate competitive. Lucrarea oferă o bază solidă pentru cercetări ulterioare în direcția utilizării rețelelor neurale și a metodelor cuantice în analiza imaginilor oftalmologice.

7. Bibliografie

[1] *Machine Learning Categories of Machine Learning*, Tutorialspoint. [Online]. Disponibil la: https://www.tutorialspoint.com/machine_learning. Accesat la: 23 apr. 2025.

[2] *What Is Machine Learning? Definition, Types, and Examples*, Coursera. [Online]. Disponibil la: https://www.coursera.org/articles/what-is-machine-learning?. Accesat la: 23 apr. 2025.

[3] *Types of Machine Learning*, GeeksforGeeks. [Online]. Disponibil la: https://www.geeksforgeeks.org/types-of-machine-learning/. Accesat la: 24 apr. 2025.

[4] *Five Machine Learning Types to Know*, IBM. [Online]. Disponibil la: https://www.ibm.com/think/topics/machine-learning-types. Accesat la: 24 apr. 2025.

[5] A. AlDelemy și R. Abd-Alhameed, „Binary Classification of Customer's Online Purchasing Behavior Using Machine Learning," *Journal of Techniques*, vol. 5, pp. 163–186, 2023, doi: 10.51173/jt.v5i2.1226. Accesat la: 26 apr. 2025.

[6] G. Vidhya, D. Nirmala și T. Manju, „Quality Challenges in Deep Learning Data Collection in Perspective of Artificial Intelligence," *Journal of Information Technology and Computing*, vol. 4, nr. 1, pp. 46–58, iun. 2023, doi: 10.48185/jitc.v4i1.725. Accesat la: 26 apr. 2025.

[7] H. Nugroho, „A Review: Data Quality Problem in Predictive Analytics," *IJAIT (International Journal of Applied Information Technology)*, vol. 7, nr. 02, p. 79, aug. 2023, doi: 10.25124/ijait.v7i02.5980. Accesat la: 27 apr. 2025.

[8] Q. Ling, „Machine Learning Algorithms Review," *Applied and Computational Engineering*, vol. 4, nr. 1, pp. 91–98, mai 2023, doi: 10.54254/2755-2721/4/20230355. Accesat la: 26 apr. 2025.

[9] C. Chai, J. Wang, Y. Luo, Z. Niu și G. Li, „Data Management for Machine Learning: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2022, doi: 10.1109/TKDE.2022.3148237. Accesat la: 27 apr. 2025.

[10] A. Bianchi *et al.*, „Trustworthy Machine Learning Predictions to Support Clinical Research and Decisions," în *Proc. IEEE 36th Int. Symp. on Computer-Based Medical Systems (CBMS)*, iun. 2023, doi: 10.1109/CBMS58004.2023.00222. Accesat la: 27 apr. 2025.

[11] H. Nugroho, „A Review: Data Quality Problem in Predictive Analytics," *IJAIT*, vol. 7, nr. 02, p. 79, aug. 2023, doi: 10.25124/ijait.v7i02.5980. Accesat la: 28 apr. 2025.

[12] T. Talaei Khoei, H. Ould Slimane și N. Kaabouch, „Deep Learning: Systematic Review, Models, Challenges, and Research Directions," *Neural Computing and Applications*, vol. 35, nr. 31, pp. 23103–23124, sept. 2023, doi: 10.1007/s00521-023-08957-4. Accesat la: 30 aprilie 2025.

[13] G. Villa, C. Tipantuña, D.S. Guamán, G.V. Arévalo și B. Arguero, „Machine Learning Techniques in Optical Networks: A Systematic Mapping Study," *IEEE Access*, vol. 11, pp. 98714–98750, 2023, doi: 10.1109/ACCESS.2023.3312387. Accesat la: 30 aprilie 2025.

[14] M. Tuays Almuqati *et al.*, „Challenges in Supervised and Unsupervised Learning: A Comprehensive Overview," *IJAIT*, vol. 14, nr. 4, 2024. Accesat la: 30 aprilie 2025.

[15] *CNN Architecture – Detailed Explanation*, InterviewBit. [Online]. Disponibil la: https://www.interviewbit.com/blog/cnn-architecture/. Accesat la: 30 aprilie 2025.

[16] *What are Convolutional Neural Networks?*, IBM. [Online]. Disponibil la: https://www.ibm.com/think/topics/convolutional-neural-networks. Accesat la: 30 apr. 2025.

[17] A. Dosovitskiy *et al.*, „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," în *Proc. ICLR 2021*. Accesat la: 01 mai 2025.

[18] H. Touvron *et al.*, „Training Data-Efficient Image Transformers & Distillation Through Attention," [Online]. Disponibil la: https://arxiv.org/abs/2012.12877, doi: 10.48550/arXiv.2012.12877. Accesat la: 01 mai 2025.

[19] Z. Liu *et al.*, „Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," *Proc. ICCV 2021*, pp. 9992–10002, doi: 10.1109/ICCV48922.2021.00986. Accesat la: 01 mai 2025.

[20] *Exploring the Power of Quantum Transfer Learning*, DataToBiz. [Online]. Disponibil la: https://www.datatobiz.com/blog/quantum-transfer-learning/. Accesat la: 17 mai 2025.

[21] *Quantum Transfer Learning via Pennylane and ResNet*, Kaggle. [Online]. Disponibil la: https://www.kaggle.com/code/prakharsinghchouhan/quantum-transfer-learning-via-pennylane-and-resnet. Accesat la: 17 mai 2025.

[22] G. Subbiah *et al.*, „Quantum Transfer Learning for Image Classification," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 2023. Accesat la: 18 mai 2025.

[23] L. Wang, Y. Sun și X. Zhang, „Quantum Deep Transfer Learning," *New Journal of Physics*, vol. 23, nr. 10, p. 103010, oct. 2021, doi: 10.1088/1367-2630/ac2a5e. Accesat la: 22 mai 2025.

8. Anexe

Fișierul **Dataset.py** conține clasele și funcțiile necesare pentru încărcarea imaginilor și a etichetelor asociate din cele două seturi de date utilizate (IDRiD și Diabetic Retinopathy). Include transformări standard de preprocesare, precum redimensionarea imaginilor la 224×224 pixeli, augmentări pentru setul de antrenament și normalizarea valorilor pixelilor. De asemenea, codul asigură împărțirea datelor în seturi de antrenament și validare, pregătind astfel datele pentru a fi utilizate de rețelele de clasificare.

```
# https://ieee-dataport.org/open-access/indian-diabetic-retinopathy-image-dataset-idrid

import os
import csv

from torchvision import transforms
from torch.utils.data import Dataset
from PIL import Image
import pandas as pd

classification_root = 'B. Disease Grading'
segmentation_root = 'A. Segmentation'

images_dir = '1. Original Images'
csv_path = '2. Groundtruths'
segmentaion_path = '2. All Segmentation Groundtruths'



def read_csv_to_lists(file_path, image_col, grade_col):
    image_names = []
    retinopathy_grades = []

    with open(file_path, mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            image_names.append(row[image_col]+'.jpg')
            retinopathy_grades.append(int(row[grade_col]))

    return image_names, retinopathy_grades

normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])

data_transforms = {
    'train':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.RandomAffine(0, shear=10, scale=(0.8,1.2)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        normalize
    ]),
    'validation':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.ToTensor(),
        normalize
    ]),
}

class RetinopathyDataset(Dataset):
    def __init__(self, image_names, labels, image_dir, transform=None):

        self.image_names = image_names
        self.labels = labels
        self.image_dir = image_dir
```

```
        self.transform = transform

    def __len__(self):
        return len(self.image_names)

    def __getitem__(self, idx):
        img_path = os.path.join(self.image_dir, self.image_names[idx])
        image = Image.open(img_path).convert("RGB")
        label = self.labels[idx]

        if self.transform:
            image = self.transform(image)

        return image, label


# https://www.kaggle.com/datasets/sovitrath/diabetic-retinopathy-224x224-2019-data

root_dir_2 = 'all_images'
csv_file_2 = 'train.csv'

class RetinopathyDataset_2(Dataset):
    def __init__(self, csv_file, root_dir, transform=None):

        self.labels_df = pd.read_csv(csv_file)
        self.root_dir = root_dir
        self.transform = transform

        self.valid_indices = []
        self.idx_mapping = {}  # Maps dataset index to original dataframe index

        for idx in range(len(self.labels_df)):
            img_name = self.labels_df.iloc[idx, 0] + '.png'
            img_path = os.path.join(self.root_dir, img_name)
            if os.path.exists(img_path):
                self.valid_indices.append(idx)

           self.idx_mapping = {i: original_idx for i, original_idx in enumerate(self.valid_indices)}

    def __len__(self):
        return len(self.valid_indices)

    def __getitem__(self, idx):

        original_idx = self.idx_mapping[idx]

        img_name = self.labels_df.iloc[original_idx, 0] + '.png'
        label = self.labels_df.iloc[original_idx, 1]

        img_path = os.path.join(self.root_dir, img_name)
        try:
            image = Image.open(img_path).convert('RGB')
        except Exception as e:
            print(f"Error loading image {img_path}: {e}")
            image = Image.new('RGB', (224, 224), color='black')
            label = 0  # or some default label

        if self.transform:
            image = self.transform(image)

        return image, label


def generate_grading_dataset():
    train_csv_path = os.path.join(classification_root, csv_path, 'a. IDRiD_Disease Grading_Training
Labels.csv')
    test_csv_path = os.path.join(classification_root, csv_path, 'b. IDRiD_Disease Grading_Testing
Labels.csv')

    train_image_names, train_retinopathy_grades = read_csv_to_lists(train_csv_path, 'Image name',
'Retinopathy grade')
    test_image_names, test_retinopathy_grades = read_csv_to_lists(test_csv_path, 'Image name',
'Retinopathy grade')
```

```
    #print(set(train_retinopathy_grades), len(set(train_retinopathy_grades)))


    return train_image_names, train_retinopathy_grades, test_image_names, test_retinopathy_grades

if __name__ == "__main__":
    from torch.utils.data import random_split
    import os
    import torch
    import matplotlib.pyplot as plt
    from collections import Counter


    train_image_names, train_retinopathy_grades, test_image_names, test_retinopathy_grades =
generate_grading_dataset()


    train_dataset = RetinopathyDataset(train_image_names,
                                       train_retinopathy_grades,
                                       image_dir=os.path.join(classification_root, images_dir, 'a. Training
Set'),
                                       transform=data_transforms['train'])

    test_dataset = RetinopathyDataset(test_image_names,
                                       test_retinopathy_grades,
                                       image_dir=os.path.join(classification_root, images_dir, 'b. Testing
Set'),
                                       transform=data_transforms['validation'])
    image_datasets = {
        'train': train_dataset,
        'validation': test_dataset
    }


    seed = 42
    generator = torch.Generator().manual_seed(seed)

    # Create datasets
    # full_dataset  = RetinopathyDataset_2(csv_file=csv_file_2,
    #                                      root_dir=root_dir_2,
    #                                      transform=data_transforms['train'])

    # # Split dataset into train and test
    # train_size = int(0.8 * len(full_dataset))
    # test_size = len(full_dataset) - train_size
    # train_dataset, test_dataset = random_split(full_dataset, [train_size, test_size],
generator=generator)

    # Function to count class occurrences
    def get_class_distribution(dataset):
        labels = [dataset[i][1] for i in range(len(dataset))]
        return Counter(labels)

    # Get class distributions
    train_class_counts = get_class_distribution(train_dataset)
    test_class_counts = get_class_distribution(test_dataset)

    # Plot function
    def plot_class_distribution(class_counts, title):
        classes = list(class_counts.keys())
        counts = list(class_counts.values())

        plt.figure(figsize=(8, 6))
        plt.bar(classes, counts, color='skyblue')
        plt.xlabel("Class Label")
        plt.ylabel("Number of Samples")
        plt.title(title)
        plt.xticks(rotation=45)
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.show()

    # Plot train and test class distributions
    plot_class_distribution(train_class_counts, "Train Dataset Class Distribution")
    plot_class_distribution(test_class_counts, "Test Dataset Class Distribution")
```

Fișierul **Train_all_cases.py** definește funcția principală de antrenare, care permite rularea automată a tuturor combinațiilor dintre cele două arhitecturi implementate și cele două seturi de date. Codul gestionează inițializarea modelului, calculul ponderilor pentru clasele dezechilibrate, salvarea ponderilor rețelei și realizarea metricilor de performanță.

```
import os
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import transforms
from collections import Counter
import matplotlib.pyplot as plt
import numpy as np
from dataset import RetinopathyDataset, RetinopathyDataset_2, generate_grading_dataset, data_transforms
from models import create_model, create_model_vit

def compute_class_weights(dataset, device):
    labels = [dataset[i][1] for i in range(len(dataset))]
    class_counts = Counter(labels)
    total_samples = sum(class_counts.values())
    num_classes = len(class_counts)
    weights = {cls: total_samples / (num_classes * count) for cls, count in class_counts.items()}
    weight_tensor = torch.tensor([weights[i] for i in range(num_classes)],
dtype=torch.float32).to(device)
    return weight_tensor


def train_model(model, dataloaders, criterion, optimizer, scheduler, device, num_epochs,
save_path='model.pth', patience=5):
    best_loss = float('inf')
    best_model_wts = None
    train_acc, val_acc, train_loss, val_loss = [], [], [], []
    patience_counter = 0

    for epoch in range(num_epochs):
        print(f"Epoch {epoch+1}/{num_epochs}")
        for phase in ['train', 'validation']:
            model.train() if phase == 'train' else model.eval()

            running_loss = 0.0
            running_corrects = 0
            total_samples = 0

            for inputs, labels in dataloaders[phase]:
                try:
                    inputs, labels = inputs.to(device), labels.to(device)
                    optimizer.zero_grad()

                    with torch.set_grad_enabled(phase == 'train'):
                        outputs = model(inputs)
                        # Extract logits from the output object
                        if hasattr(outputs, 'logits'):
                            logits = outputs.logits
                        else:
                            logits = outputs

                        loss = criterion(logits, labels)
                        _, preds = torch.max(logits, 1)

                        if phase == 'train':
                            loss.backward()
                            optimizer.step()

                    running_loss += loss.item() * inputs.size(0)
```

```
                    running_corrects += torch.sum(preds == labels.data)
                    total_samples += inputs.size(0)
                except:
                    print("batch skipped")
                    continue

            epoch_loss = running_loss / total_samples
            epoch_acc = running_corrects.double() / total_samples

            print(f"{phase.capitalize()} Loss: {epoch_loss:.4f}, Acc: {epoch_acc:.4f}")

            if phase == 'train':
                train_loss.append(epoch_loss)
                train_acc.append(epoch_acc.item())
                scheduler.step()
            else:
                val_loss.append(epoch_loss)
                val_acc.append(epoch_acc.item())
                if epoch_loss < best_loss:
                    best_loss = epoch_loss
                    best_model_wts = model.state_dict()
                    patience_counter = 0
                    torch.save(model.state_dict(), save_path)
                else:
                    patience_counter += 1
                    if patience_counter >= patience:
                        print("Early stopping")
                        model.load_state_dict(best_model_wts)
                        plot_training(train_acc, val_acc, train_loss, val_loss,
save_path.replace('.pth', '.png'))
                        return model

    model.load_state_dict(best_model_wts)
    plot_training(train_acc, val_acc, train_loss, val_loss, save_path.replace('.pth', '.png'))
    return model


def plot_training(train_acc, val_acc, train_loss, val_loss, save_path):
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(train_acc, label='Train Accuracy')
    plt.plot(val_acc, label='Validation Accuracy')
    plt.title('Accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(train_loss, label='Train Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.title('Loss')
    plt.legend()

    plt.savefig(save_path)
    plt.close()


def run_training(dataset_class, model_fn, csv_file=None, root_dir=None, save_path='model.pth'):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    if dataset_class == RetinopathyDataset:
        train_image_names, train_grades, test_image_names, test_grades = generate_grading_dataset()
        train_dataset = RetinopathyDataset(train_image_names, train_grades, os.path.join('B. Disease
Grading', '1. Original Images', 'a. Training Set'), data_transforms['train'])
        test_dataset = RetinopathyDataset(test_image_names, test_grades, os.path.join('B. Disease
Grading', '1. Original Images', 'b. Testing Set'), data_transforms['validation'])
    else:
        full_dataset = RetinopathyDataset_2(csv_file, root_dir, transform=data_transforms['train'])
        train_size = int(0.8 * len(full_dataset))
        test_size = len(full_dataset) - train_size
        torch.manual_seed(42)
        train_dataset, test_dataset = torch.utils.data.random_split(full_dataset, [train_size,
test_size])

    dataloaders = {
        'train': DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True),
```

```
        'validation': DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False)
    }

    model = model_fn(num_classes=5).to(device)
    if hasattr(model, 'classifier'):
        params_to_optimize = model.classifier.parameters()
    elif hasattr(model, 'fc'):
        params_to_optimize = model.fc.parameters()
    else:
        params_to_optimize = model.parameters()

    optimizer = optim.AdamW(params_to_optimize, lr=LEARNING_RATE)
    scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=STEP_SIZE, gamma=GAMMA)
    class_weights = compute_class_weights(train_dataset, device)
    criterion = nn.CrossEntropyLoss(weight=class_weights)

    train_model(model, dataloaders, criterion, optimizer, scheduler, device, num_epochs=EPOCHS,
save_path=save_path)

EPOCHS = 50
LEARNING_RATE = 1e-4
BATCH_SIZE = 16
GAMMA = 0.5
STEP_SIZE = 5

if __name__ == "__main__":
    # 1. ResNet on Dataset 1
    run_training(RetinopathyDataset, create_model, save_path='resnet_dataset1.pth')
    # 2. ViT on Dataset 1
    run_training(RetinopathyDataset, create_model_vit, save_path='vit_dataset1.pth')
    # 3. ResNet on Dataset 2
    run_training(RetinopathyDataset_2, create_model, csv_file='train.csv', root_dir='all_images',
save_path='resnet_dataset2.pth')
    # 4. ViT on Dataset 2
    run_training(RetinopathyDataset_2, create_model_vit, csv_file='train.csv', root_dir='all_images',
save_path='vit_dataset2.pth')
```