

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

DermatoAI

Soluție pentru diagnosticarea afecțiunilor pielii

propusă de:

Apricopoi Andrei-Constantin

Sesiunea: iulie, 2024

Coordonator științific

Prof. Dr. Iftene Adrian

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ

DermatoAI
Soluție pentru diagnosticarea afecțiunilor pielii

Apricopoi Andrei-Constantin

Sesiunea: iulie, 2024

Coordonator științific
Prof. Dr. Iftene Adrian

Cuprins

Motivație	7
Introducere	8
Capitolul 1: Aplicații similare	9
1.1 SkinVision	9
1.2 Medic Scanner	10
1.3 Skin Scanner	11
1.3 Concluzie	12
Capitolul 2: Tehnologii utilizate	13
2.1 Flutter	13
2.2 Node.js	13
2.3 Express	14
2.4 MongoDB și Mongoose	14
2.5 Python	15
2.6 Azure Queue Storage	15
2.7 Azure Blob Storage	16
2.8 Google Maps API	16
2.9 OpenAI API	16
2.10 SendGrid	17
2.11 TensorFlow	17
2.12 Git și GitHub	18
2.13 Concluzie	18
Capitolul 3: Obținerea modelelor	19
3.1 ISIC Archive	19
3.2 Descrierea seturilor de date	19
3.2.1 HAM10000	19
3.2.2 BCN20000	20
3.2.3 Challenge 2019 și Challenge 2020	21
3.2.4 ArsenicSkinImageBD	22
3.2.5 Concluzie	23
3.3 Preprocesarea seturilor de date alese	24
3.3.1 Probleme ale seturilor de date alese inițial și soluțiile propuse	24
3.3.2 Tehnici de augmentare folosite	24
3.3.3 Analiza diferitelor configurații ale seturilor de date augmentate	26
3.3.4 Concluzie	28
3.4 Antrenarea modelelor	28
3.4.1 Noțiunile teoretice folosite la antrenare	28
3.4.1.1 Rețea Neuronală	28
3.4.1.2 Model Secvențial	29
3.4.1.3 Strat de Intrare	30
3.4.1.4 Strat Convoluțional	30

3.4.1.5 Strat de Pooling	30
3.4.1.6 Stratul Dens	31
3.4.1.7 Funcții de Activare	31
3.4.1.8 Rate de Învățare	32
3.4.1.9 Funcții de Optimizare	33
3.4.1.10 Batch și Epoci	33
3.4.1.11 Funcții de Pierdere	34
3.4.1.12 Entropie	34
3.4.1.13 Precizie, Recall si Scor F1	34
3.4.1.14 Matrice de Confuzie	35
3.4.2 Încercările inițiale și decizia folosirii unui model pre-antrenat	36
3.4.3 Model pre-antrenat VGG19	38
3.4.4 Model pre-antrenat MobileNetV3	38
3.4.5 Modelele pre-antrenate DenseNet169 și DenseNet201	39
3.4.6 Model pre-antrenat InceptionResNetV2	39
3.4.7 Antrenarea modelului binar	40
3.4.8 Antrenarea modelului multi-clasă	41
3.4.9 Concluzie	45
Capitolul 4: Arhitectura și implementarea aplicației	46
4.1 Structura întregului sistem	46
4.2 Structura serverului	48
4.3 Structura componentei Azure	49
4.4 Structura aplicației mobile	49
4.5 Compoziția bazei de date	50
4.6 Compoziția procesatorului de imagini	51
4.7 Concluzie	52
Capitolul 5: Utilizarea Aplicației	53
5.1 Autentificarea în aplicație	53
5.2 Resetarea și recuperarea parolei	53
5.3 Încărcarea de imagini și gestionarea predicțiilor	54
5.4 Vizualizarea paginii de informații pentru afecțiuni	55
5.5 Utilizarea chat-ului de inteligență artificială	56
5.6 Vizualizarea locațiilor dermatologice din zonă	56
5.7 Crearea de programări și primirea notificărilor	57
5.8 Vizualizarea profilului și trimiterea de feedback	58
5.9 Concluzie	59
Concluzii finale și posibile îmbunătățiri	59
Bibliografie	60

Motivație

S-a ales dezvoltarea și cercetarea unei aplicații mobile pentru diagnosticarea bolilor de piele pe baza imaginilor din mai multe motive importante.

Accesul la servicii medicale de înaltă calitate este limitat în multe țări nedezvoltate și în curs de dezvoltare din cauza lipsei de resurse financiare, a infrastructurii inadecvate și a lipsei de personal medical calificat. Dermatologia, fiind un domeniu specializat, suferă și mai mult din această perspectivă. O aplicație mobilă care să permită utilizatorilor să încarce imagini ale afecțiunilor lor cutanate pentru diagnostic preliminar ar putea reprezenta un instrument valoros pentru a umple acest gol. Prin accesul mai ușor la diagnosticare dermatologică, putem reduce inegalitățile în sănătate și îmbunătăți calitatea vieții pentru multe persoane din comunități dezavantajate.

În lumea aplicațiilor mobile și a noilor tehnologii, dermatologia este încă un domeniu care primește puțină atenție în ciuda progreselor semnificative în tehnologiile medicale. Majoritatea aplicațiilor actuale se concentrează pe fitness, sănătatea mentală sau afecțiunile cronice, lăsând dermatologia la o parte. Această lipsă de inovație poate fi cauzată de complexitatea diagnosticului dermatologic, care necesită adesea un ochi antrenat și experiență clinică. Totuși, progresele în învățarea automată și recunoașterea imaginilor oferă noi oportunități pentru a depăși aceste obstacole și a aduce dermatologia în era digitală.

Diagnosticarea bolilor de piele poate fi mult mai accesibilă cu ajutorul aplicațiilor mobile. Utilizatorii pot primi un diagnostic preliminar rapid, ceea ce le permite să înceapă tratamentul mai devreme și să reducă frica cauzată de incertitudinea medicală. Aceste aplicații pot, de asemenea, să sensibilizeze utilizatorii cu privire la importanța monitorizării sănătății pielii și la semnele timpurii ale afecțiunilor severe, cum ar fi cancerul de piele. Data colectată de utilizatori poate ajuta la îmbunătățirea continuă a algoritmilor de diagnosticare și poate sprijini cercetarea medicală în dermatologie.

O aplicație mobilă de diagnosticare a bolilor de piele poate îmbunătăți educația medicală preventivă. Utilizatorii pot învăța cum să recunoască simptomele incipiente ale diferitelor afecțiuni dermatologice, ceea ce îi poate motiva să urmeze un stil de viață sănătos și să caute asistență medicală atunci când afecțiunea este în stadiu incipient. Această metodă preventivă poate reduce probabilitatea apariției complicațiilor grave și poate contribui la o populație mai educată și mai sănătoasă. Învățarea continuă prin aplicație poate de asemenea să crească conștientizarea importanței protejării pielii și a sănătății generale.

Introducere

În era digitală, tehnologia devine tot mai importantă în domeniul sănătății, de la aplicații de telemedicină la monitorizarea stării de sănătate a pacienților. Aceste progrese tehnologice îmbunătățesc semnificativ accesul la serviciile medicale și creșterea calității îngrijirii.

Problemele de piele sunt comune, variate și afectează oameni de toate vârstele și din toate colțurile lumii. Diagnosticarea rapidă și corectă a acestor afecțiuni este vitală pentru a evita complicațiile și pentru a primi un tratament adecvat. În ciuda acestui fapt, accesul la dermatologi este limitat în multe zone din întreaga lume din cauza resurselor limitate, infrastructurii inadecvate și a lipsei de personal medical calificat.

Obiectivul principal al acestei lucrări este dezvoltarea și evaluarea unei aplicații mobile capabile să diagnosticheze diverse boli de piele pe baza imaginilor încărcate de utilizatori. Aplicația va utiliza algoritmi de învățare automată pentru a analiza imaginile și a furniza un diagnostic preliminar

Prin această lucrare, se dorește evidențierea potențialului tehnologiei mobile și a algoritmilor de învățare automată în domeniul dermatologiei, subliniind atât beneficiile aduse utilizatorilor, cât și provocările întâlnite în procesul de dezvoltare și implementare a unei astfel de aplicații.

În cele ce urmează, se vor prezenta în detaliu tehnologiile utilizate în dezvoltarea aplicației mobile pentru diagnosticarea bolilor de piele. Se va discuta despre alegerea și integrarea diferitelor framework-uri și biblioteci de învățare automată, care permit antrenarea și optimizarea modelelor de recunoaștere a imaginilor. Va fi descris procesul de obținere a modelelor de machine learning, incluzând descrierea setului de date și preprocesarea acestuia, selecția algoritmilor, antrenarea modelelor și evaluarea performanței acestora. De asemenea, se va explora arhitectura aplicației, evidențiind componentele cheie și modul în care acestea interacționează pentru a oferi un diagnostic. În plus, va fi explicat fluxul complet al aplicației, de la momentul în care utilizatorul încarcă o imagine, până la momentul în care primește un diagnostic preliminar. Acest flux va include detalii despre analiza imaginilor și interacțiunea cu utilizatorul printr-o interfață intuitivă și ușor de utilizat. Prin aceste explicații detaliate, se va oferi o înțelegere clară a întregului proces de dezvoltare și funcționare a aplicației, subliniind inovațiile tehnologice și beneficiile aduse utilizatorilor.

Capitolul 1: Aplicații similare

În cele ce urmează, vom examina câteva aplicații similare pentru diagnosticarea bolilor de piele care sunt disponibile în prezent. Se vor examina caracteristicile și funcționalitățile acestor aplicații, punând accent pe punctele lor slabe. În urma acestei analize comparative, DermatoAI va evidenția avantajele sale, subliniind diferența sa și îmbunătățirile față de soluțiile existente. Această prezentare va arăta modul în care DermatoAI ajută la progresele tehnologice și la îmbunătățirea accesului la diagnosticarea dermatologică.

1.1 SkinVision

SkinVision [1] este o aplicație mobilă care vă ajută să diagnosticați cancerul de piele și alte afecțiuni dermatologice în timp util (vezi Figura 1). Aplicația permite utilizatorilor să facă fotografii ale leziunilor suspecte de pe piele, iar algoritmul aplicației analizează imaginile pentru a identifica semnele posibile ale cancerului de piele.

Pentru a folosi SkinVision, utilizatorul trebuie să se înregistreze. Aceștia pot utiliza camera smartphone-ului pentru a face o fotografie a leziunii cutanate. Utilizatorii pot obține o imagine clară și bine focalizată, necesară pentru o analiză precisă, prin intermediul instrucțiunilor vizuale oferite de aplicație. Un algoritm de învățare automată folosește apoi o bază de date extinsă de imagini dermatologice pentru a analiza imaginea.

Un dezavantaj semnificativ al SkinVision este costul ridicat al utilizării. Pentru a beneficia de funcționalitățile complete, utilizatorii trebuie să plătească un abonament sau taxe pentru analize individuale, ceea ce poate fi inaccesibil pentru multe persoane. De asemenea, capturarea imaginilor de calitate poate fi dificilă, necesitând atenție pentru obținerea fotografiilor clare și bine iluminate, ceea ce poate fi o provocare pentru utilizatorii fără experiență tehnică.

În plus, SkinVision nu are un asistent AI de chat pentru suport în timp real, astfel utilizatorii trebuie să se bazeze exclusiv pe informațiile oferite de aplicație. Aplicația nu oferă nici funcționalitatea de a ghida utilizatorii către clinici dermatologice din apropiere, limitând sprijinul pentru continuarea îngrijirii medicale după primirea rezultatelor inițiale. Aceste limitări pot reduce eficacitatea și accesibilitatea aplicației.

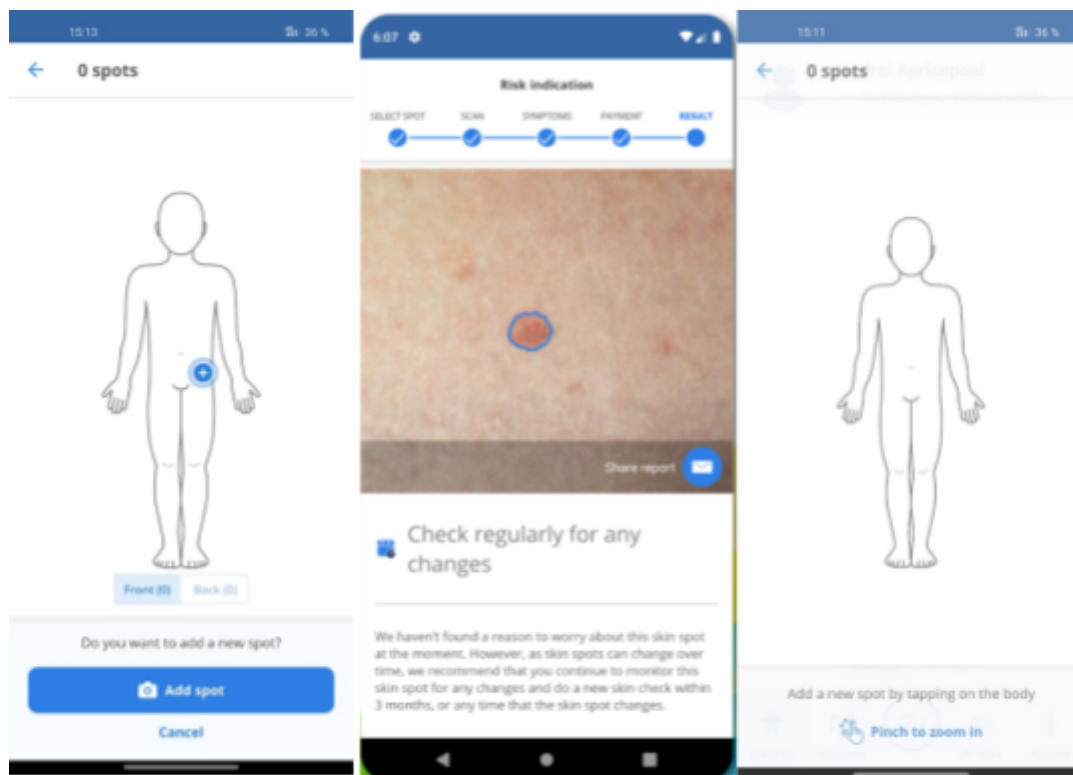


Figura 1. Exemplu din aplicația SkinVision

1.2 Medic Scanner

Medic Scanner [2] este o aplicație mobilă care analizează imaginile utilizatorului pentru a diagnostica o varietate de afecțiuni dermatologice (vezi Figura 2). Aplicația oferă un diagnostic preliminar după ce utilizatorii au fotografiat zonele de piele afectate.

Modul de utilizare al Medic Scanner este asemănător cu cel al SkinVision. Utilizatorii trebuie să creeze un cont și să facă o fotografie a zonei de piele afectate. Medic Scanner oferă, totuși, mai puține instrucțiuni și informații în comparație cu SkinVision, ceea ce face ca aplicația să pară mai simplă. Cu toate acestea, obținerea unei fotografii de calitate este mai dificilă, deoarece sunt necesare standarde stricte de iluminare și claritate a imaginii, ceea ce poate fi incomod și frustrant pentru utilizatori.

Aplicația oferă două scanări gratuite, după care utilizatorii trebuie să treacă la pachete plătite. Pentru ca imaginea să fie analizată de modele avansate, este necesară achiziționarea unui pachet special mai scump, ceea ce o face inaccesibilă pentru mulți oameni. De asemenea, nici aici, aplicația oferă suport de chat AI sau localizări ale dermatologilor din apropiere.

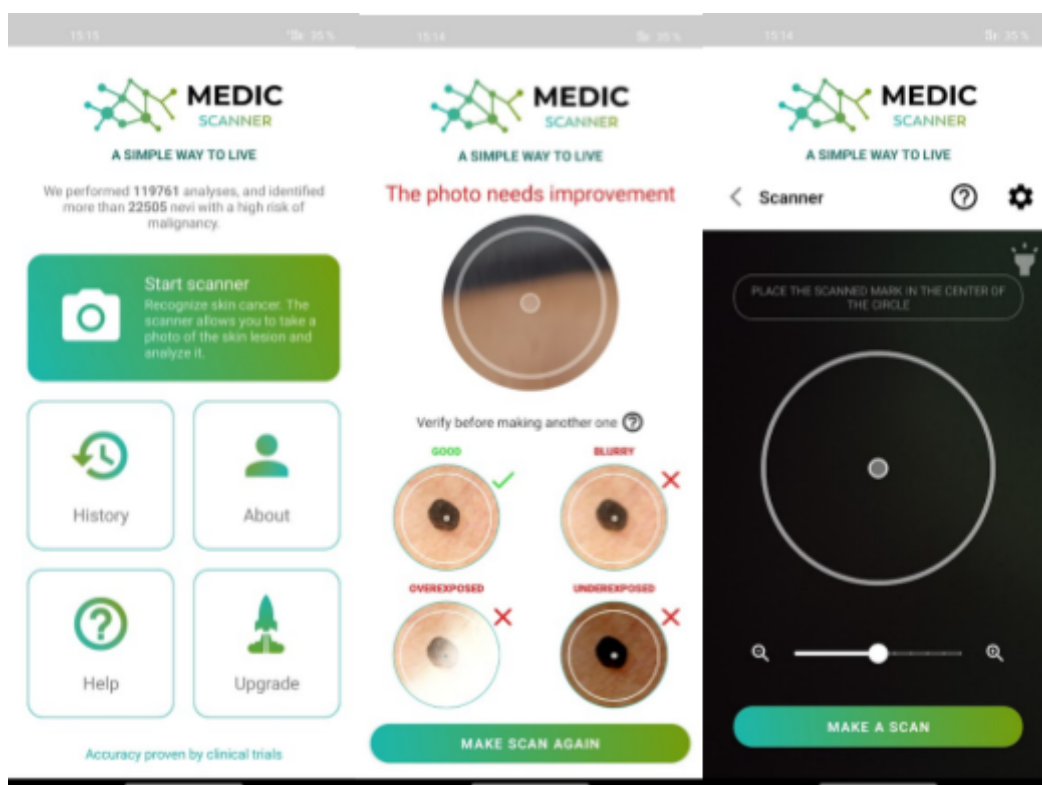


Figura 2. Exemplu din aplicația Medic Scanner

1.3 Skin Scanner

Skin Scanner [3] este o aplicație dermatologică ușor de folosit. Utilizatorii își creează un cont la început, după care pot încărca poze din telefon sau pot face poze cu camera telefonului la diferite afecțiuni ale pielii (vezi Figura 3). Totuși, aplicația oferă foarte puține încărcări de imagini gratuit, iar pentru a beneficia de mai multe încărcări, este necesară achiziționarea unui plan plătit.

Un aspect pozitiv al aplicației este accesul la articole legate de îngrijirea pielii și posibilitatea de a adăuga remindere pentru îngrijirea personală. Cu toate acestea, multe persoane au lăsat recenzii negative, evidențiind diverse probleme. De asemenea, ca și în cazurile celelalte, aplicația nu este echipată cu tool-uri AI, cum ar fi un chatbot pentru suport sau un instrument de găsire a dermatologilor din zonă, limitând astfel sprijinul oferit utilizatorilor.

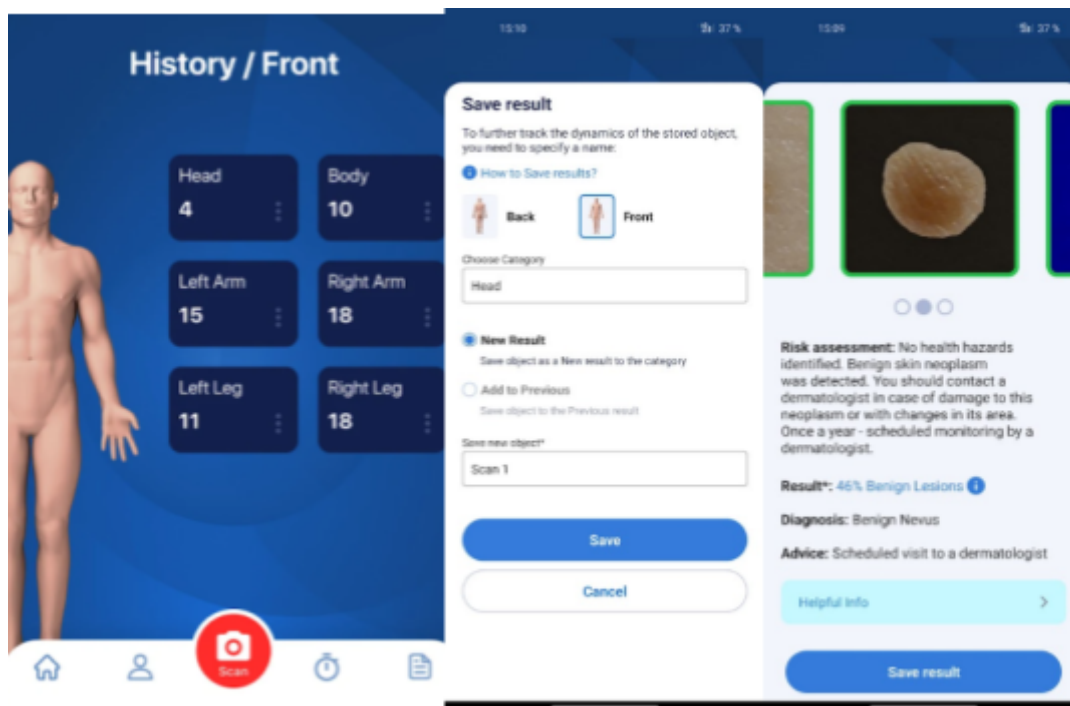


Figura 3. Exemplu din aplicația Skin Scanner

1.3 Concluzie

În realitate, aplicațiile descrise mai sus, precum SkinVision, Medic Scanner și AI Dermatologist: Skin Scanner, sunt foarte probabil să aibă performanțe mai bune, o experiență a utilizatorului mai plăcută și, în general, o calitate superioară a produsului. Aceste aplicații beneficiază de echipe și resurse considerabile în spatele lor, având acces la volume mari de date și algoritmi avansați. Cu toate acestea, pentru a le utiliza la maxim, utilizatorii trebuie să plătească pentru abonamente sau pachete suplimentare, ceea ce le face inaccesibile pentru persoanele cu resurse financiare limitate.

Deși aceste aplicații sunt bine concepute și utile, ele nu oferă un ajutor real celor care au cea mai mare nevoie de ele - oamenii fără posibilități financiare care nu își permit să le utilizeze. De aceea, s-a dorit crearea unei aplicații de costuri minime, care să integreze caracteristici relativ noi pe care nu le-am mai observat la alte aplicații până acum. Printre aceste caracteristici se numără adăugarea unui chatbot care să asiste utilizatorul în timpul folosirii aplicației și să răspundă la diferite întrebări dermatologice, precum și un instrument de găsire a dermatologilor din zona utilizatorului, cu acces direct la Google Maps în aplicație.

Capitolul 2: Tehnologii utilizate

În capitolul următor, vor fi prezentate tehnologiile utilizate în realizarea aplicației și rolul fiecăreia în cadrul proiectului. Se va detalia ce funcționalitate îndeplinește fiecare tehnologie și de ce au fost alese aceste soluții specifice. Argumentele pentru selecția tehnologiilor vor fi discutate, evidențiind avantajele și beneficiile aduse de acestea în dezvoltarea și funcționarea aplicației.

2.1 Flutter

Flutter [4] este un framework de dezvoltare open-source creat de Google, utilizat pentru a construi aplicații mobile, web și desktop dintr-o singură bază de cod. Folosind limbajul de programare Dart, Flutter permite crearea de interfețe de utilizator rapide și atractive, compatibile atât cu Android, cât și cu iOS.

Pentru realizarea aplicației mobile, s-a ales Flutter datorită robusteii și ușurinței sale de utilizare. Framework-ul oferă widget-uri predefinite și o colecție vastă de pachete, simplificând procesul de dezvoltare. Aplicațiile construite cu Flutter au performanțe ridicate, fiind compilate nativ, ceea ce asigură timpi de răspuns rapizi și interacțiuni fluide.

Un alt avantaj al Flutter este posibilitatea de a extinde aplicația către alte platforme, inclusiv web, utilizând aceeași bază de cod. Această flexibilitate economisește timp și resurse, făcând Flutter o soluție ideală pentru proiecte multi-platformă. Aceste caracteristici au făcut ca Flutter să fie alegerea optimă pentru dezvoltarea aplicației, asigurând o experiență de utilizator de înaltă calitate și eficiență în dezvoltare.

2.2 Node.js

Node.js [5] este un mediu de execuție JavaScript construit pe motorul V8 al Chrome, utilizat pentru a crea aplicații scalabile de rețea și servere web. Pentru dezvoltarea API-ului aplicației, s-a ales Node.js datorită eficienței sale și a capacității de a gestiona un număr mare de conexiuni simultane cu latență redusă.

Node.js a fost ales pentru baza API-ului datorită mai multor avantaje. Primul avantaj este modelul său de operare asincron și bazat pe evenimente, care permite serverului să proceseze multiple solicitări fără a bloca execuția. Acest lucru asigură o performanță ridicată și un timp de răspuns rapid, esențial pentru aplicațiile moderne.

Al doilea avantaj este ecosistemul său vast de pachete și module disponibile prin npm (Node Package Manager). Acest ecosistem facilitează integrarea rapidă a diferitelor

funcționalități și servicii, reducând timpul și efortul necesar pentru dezvoltarea și întreținerea API-ului. Aceste caracteristici fac din Node.js o alegere ideală pentru crearea de API-uri performante și scalabile.

2.3 Express

Express [6] este un framework web minimal și flexibil pentru Node.js, conceput pentru a simplifica dezvoltarea de aplicații web și API-uri. Oferind un set de instrumente robuste pentru crearea de servere și gestionarea rutelor, Express se numără printre cele mai populare și ușor de utilizat framework-uri pentru dezvoltarea de API-uri. Setup-ul inițial este simplu și rapid, permițând dezvoltatorilor să creeze rapid rute și să gestioneze cererile HTTP. Pentru aplicațiile care nu necesită un backend foarte complex, Express este soluția ideală datorită simplității sale.

Framework-ul permite integrarea ușoară cu alte middleware-uri și module Node.js, facilitând astfel extinderea funcționalităților aplicației. De asemenea, Express beneficiază de o comunitate mare și activă, care oferă suport și resurse abundente, cum ar fi tutoriale și pachete open-source. Aceste resurse ajută la rezolvarea rapidă a problemelor și la implementarea rapidă a noilor funcționalități. Toate aceste caracteristici fac din Express o alegere excelentă pentru dezvoltarea de API-uri rapide și eficiente

2.4 MongoDB și Mongoose

MongoDB [7] este o bază de date NoSQL orientată pe documente, care stochează datele în structuri BSON (Binary JSON). A fost aleasă ca soluție pentru implementarea bazei de date datorită ușurinței de configurare și eficienței sale pentru aplicații de dimensiuni mici sau medii. MongoDB se integrează foarte bine cu Express, oferind flexibilitate și performanță în gestionarea datelor. Un plus major față de alte soluții de baze de date este MongoDB Atlas, un serviciu de hosting care permite stocarea bazei de date în cloud. Astfel, baza de date nu este păstrată local și poate fi accesată simplu printr-un connection string în aplicație.

Mongoose [8] este o bibliotecă Object Data Modeling (ODM) pentru MongoDB și Node.js, care oferă un mod elegant de a gestiona datele în MongoDB. Mongoose permite definirea schemelor pentru documentele din colecții, asigurându-se că datele respectă structurile definite. Aceasta adaugă o suprafață suplimentară de validare și control asupra datelor. Mongoose se integrează perfect cu MongoDB, oferind un API ușor de utilizat, care face gestionarea datelor mai intuitivă și eficientă. Alegerea Mongoose alături de MongoDB asigură o soluție robustă și flexibilă pentru dezvoltarea proiectului.

2.5 Python

Python [9] este un limbaj de programare de nivel înalt, cunoscut pentru sintaxa sa clară și concisă, care face programarea accesibilă și eficientă. Este un limbaj interpretat, orientat pe obiecte, utilizat pe scară largă pentru dezvoltarea de aplicații web, automatizarea sarcinilor, analiza datelor, inteligență artificială și învățare automată. Python oferă o mulțime de avantaje, printre care ușurința de învățare și utilizare, o comunitate vastă și activă, și o multitudine de librării și framework-uri care simplifică și accelerează dezvoltarea aplicațiilor.

Utilizarea Python în acest proiect a fost esențială pentru mai multe aspecte ale dezvoltării și implementării aplicației. Python a fost folosit pentru a preprocesa seturile de date, realizând augmentarea și balansarea acestora pentru a asigura o calitate și diversitate adecvată a datelor de antrenament. De asemenea, Python a fost utilizat pentru rularea scripturilor care antrenează modelele de machine learning, datorită suportului excelent oferit de librării precum TensorFlow și Keras. În plus, Python este utilizat în aplicație prin intermediul unui worker, care procesează imaginile primite de la utilizatori, le trece prin modele de învățare automată și actualizează rezultatele obținute.

2.6 Azure Queue Storage

Azure Queue Storage [10] este un serviciu oferit de Microsoft Azure care permite stocarea și gestionarea mesajelor într-o coadă, permițând comunicarea asincronă între componentele aplicațiilor. Este un serviciu ușor de utilizat, fiind accesibil prin API-uri simple, și se numără printre serviciile Microsoft Azure care pot fi folosite gratuit pentru o perioadă limitată, ceea ce îl face atractiv pentru dezvoltatorii care doresc să testeze și să implementeze soluții în cloud fără costuri inițiale mari.

În aplicația noastră, Azure Queue Storage este utilizat pentru a crea o coadă în cloud unde sunt plasate toate cererile de procesare a imaginilor venite de la utilizatori. Aceasta asigură gestionarea eficientă a sarcinilor, permițând ca imaginile să fie procesate în ordine, fără a pune timpi de așteptare inutili pe server. Workerul din aplicație preia cererile din coadă și procesează imaginile una câte una, garantând astfel o utilizare optimă a resurselor și o experiență mai bună pentru utilizatori. Utilizarea Azure Queue Storage contribuie astfel la scalabilitatea și fiabilitatea aplicației, asigurând un flux de lucru eficient și continuu.

2.7 Azure Blob Storage

Azure Blob Storage [11] este un serviciu de stocare în cloud de la Microsoft Azure, destinat stocării a mari volume de date, cum ar fi documente, imagini și videoclipuri. Acesta face parte din pachetul de servicii Azure care pot fi utilizate gratuit pentru o perioadă de probă, oferind dezvoltatorilor oportunitatea de a testa și implementa soluții de stocare în cloud.

La noi în aplicație, Azure Blob Storage este utilizat pentru a stoca imagini eficient în cloud, evitând astfel suprasolicitarea bazei de date MongoDB. Acest lucru permite o gestionare mai eficientă a spațiului de stocare și a resurselor aplicației. Pentru a accesa o imagine stocată, este necesar doar linkul imaginii respective, simplificând astfel gestionarea datelor.

Configurarea și utilizarea Azure Blob Storage este simplă și se integrează fără probleme cu tehnologiile menționate anterior, cum ar fi Node.js, Express și MongoDB. Aceasta oferă o soluție robustă și scalabilă pentru stocarea datelor, contribuind astfel la performanța și eficiența generală a aplicației.

2.8 Google Maps API

Google Maps API [12] este un set de servicii oferit de Google care permite dezvoltatorilor să integreze funcționalități de hărți și localizare în aplicațiile lor. Acesta oferă o gamă largă de funcții, inclusiv afișarea hărților, căutarea locațiilor și calcularea rutelor.

În aplicația noastră, Google Maps API este folosit pentru a căuta toate clinicile dermatologice disponibile într-o anumită rază specificată pornind de la locația utilizatorului. Am ales să folosim Google Maps API deoarece se integrează foarte bine cu Node.js prin intermediul pachetelor disponibile, este ușor de utilizat și, în general, este soluția de top pentru gestionarea hărților și locațiilor. Aceasta permite utilizatorilor noștri să găsească rapid și eficient doctori dermatologi în apropierea lor.

2.9 OpenAI API

OpenAI API [13] este un serviciu care permite accesul la modelele de inteligență artificială dezvoltate de OpenAI. În termeni simpli, acest API permite aplicațiilor să genereze texte și răspunsuri inteligente pe baza unor întrebări sau instrucțiuni date de utilizatori.

La noi în aplicație, OpenAI API este utilizat pentru a genera răspunsuri la întrebările utilizatorilor despre condițiile dermatologice. Acest API este simplu de implementat și

permite adăugarea rapidă a unei funcționalități de chat în aplicație, oferind utilizatorilor răspunsuri relevante și precise. De asemenea, OpenAI API dispune de o librărie Node.js foarte intuitivă, care facilitează integrarea sa în sistemul nostru, contribuind astfel la îmbunătățirea experienței utilizatorilor prin răspunsuri interactive.

2.10 SendGrid

SendGrid [14] este un serviciu de e-mail bazat pe cloud care permite trimiterea de e-mailuri la scară mare, inclusiv e-mailuri tranzacționale și de marketing. Este utilizat de dezvoltatori și companii pentru a gestiona și optimiza comunicarea prin e-mail cu utilizatorii lor. SendGrid este folosit de către noi pentru a trimite două tipuri de e-mailuri esențiale. Primul tip de e-mail este cel de verificare a utilizatorului, trimis pentru a confirma adresa de e-mail a acestuia și pentru a activa contul. Al doilea tip de e-mail este destinat resetării parolei, oferind utilizatorilor posibilitatea de a-și reseta parola în cazul în care o uită.

Am ales SendGrid datorită fiabilității și ușurinței de utilizare. SendGrid oferă o integrare simplă cu Node.js și dispune de o infrastructură robustă, asigurând livrarea promptă și sigură a e-mailurilor.

2.11 TensorFlow

TensorFlow [15] este un framework open-source dezvoltat de Google pentru crearea și antrenarea modelelor de învățare automată și rețele neuronale. Acesta oferă un set de instrumente și biblioteci care permit dezvoltatorilor să construiască și să implementeze modele complexe de deep learning.

În aplicația noastră, TensorFlow este folosit pentru a antrena modelele de rețea neuronală necesare pentru analizarea și diagnosticarea imaginilor dermatologice. Alegerea TensorFlow a fost motivată de integrarea sa cu Keras, o bibliotecă de nivel înalt care rulează peste TensorFlow și simplifică procesul de creare și antrenare a modelelor. Keras oferă o interfață ușor de utilizat și permite dezvoltarea rapidă a prototipurilor, făcându-l ideal pentru proiectele care necesită o iterație rapidă și eficientă.

Keras este folosit și pentru preprocesarea datelor, oferind instrumente care simplifică foarte mult procesul de augmentare și transformare a datelor, asigurând astfel calitatea și diversitatea necesară pentru antrenarea eficientă a modelelor.

2.12 Git și GitHub

Git [16] este un sistem de control al versiunilor distribuit, care permite gestionarea eficientă a modificărilor aduse codului sursă într-un proiect software. A fost folosit în aplicație pentru a urmări modificările și a păstra un istoric complet al tuturor schimbărilor efectuate în cod.

GitHub [17] este o platformă de hosting pentru proiecte Git, care oferă un set de instrumente suplimentare pentru managementul de proiect. Prin utilizarea GitHub, codul sursă este stocat în cloud, ceea ce facilitează accesul de oriunde și oferă un mediu sigur pentru backup și versionare.

2.13 Concluzie

În dezvoltarea aplicației noastre, a fost utilizat un stack de tehnologii diversificat și bine integrat, care se completează reciproc în mod eficient. Aceste tehnologii sunt accesibile și ușor de configurat, fiind alese cu gândul la dimensiunea proiectului, volumul de date utilizate și eficiența generală dorită. Alegerea acestor tehnologii a permis nu doar un proces de dezvoltare lin, ci și crearea unei aplicații robuste, capabile să ofere o experiență de utilizator de calitate.

Capitolul 3: Obținerea modelelor

În acest capitol se va pune accent pe procesul de obținere a modelelor, detaliind sursele seturilor de date utilizate, procesul de augmentare a acestora, precum și antrenarea modelelor folosind setul de date cel mai potrivit. Se va descrie cum au fost selectate și prelucrate datele pentru a asigura diversitatea și calitatea necesară, urmând apoi să se explice metodele folosite pentru antrenarea modelelor, astfel încât să se obțină performanțe cât mai bune în diagnosticarea bolilor de piele.

3.1 ISIC Archive

ISIC Archive (International Skin Imaging Collaboration) [18] este o platformă dedicată cercetării în domeniul dermatologiei, care colectează și distribuie imagini și date despre diverse afecțiuni ale pielii. Acesta oferă un vast set de date compus din imagini dermatologice de înaltă calitate, adnotate și validate de experți, fiind o resursă esențială pentru dezvoltarea și antrenarea modelelor de învățare automată destinate diagnosticării bolilor de piele.

Pentru acest proiect, seturile de date au fost obținute din ISIC Archive. A fost necesară crearea unui cont pe platformă pentru a avea acces la datele disponibile. După înregistrare, seturile de date relevante au fost descărcate, aceste fiind HAM10000, BCN20000 și datele din provocările ISIC din anii 2019 și 2020 pe care le voi detalia în continuare.

3.2 Descrierea seturilor de date

În continuare, vom analiza patru seturi de date obținute din ISIC Archive, și anume HAM10000, BCN20000, precum și datele din provocările ISIC din anii 2019 și 2020. De asemenea, vom examina un set de date binar de imagini dermatologice obținut de la National Library of Medicine folosit pentru crearea unui model care să decidă dacă pielea dintr-o poza este sănătoasă sau nu.

3.2.1 HAM10000

Setul de date **HAM10000** [19] a fost obținut din ISIC Archive și conține 11.720 de imagini dermatologice (vezi Figura 4), fiecare având o rezoluție de 600 pixeli în lățime și 450 pixeli în înălțime. Acest set de date este dezechilibrat (vezi Figura 5), deoarece majoritatea imaginilor sunt de tip nevus. De asemenea, setul de date nu oferă o diversitate mare de afecțiuni (labels), incluzând un total de opt etichete diferite: vascular_lesion,

squamous_cell_carcinoma, pigmented_benign_keratosis, nevus, melanoma, dermatofibroma, basal_cell_carcinoma și actinic_keratosis.



Figura 4. Exemple de imagini din setul inițial HAM10000

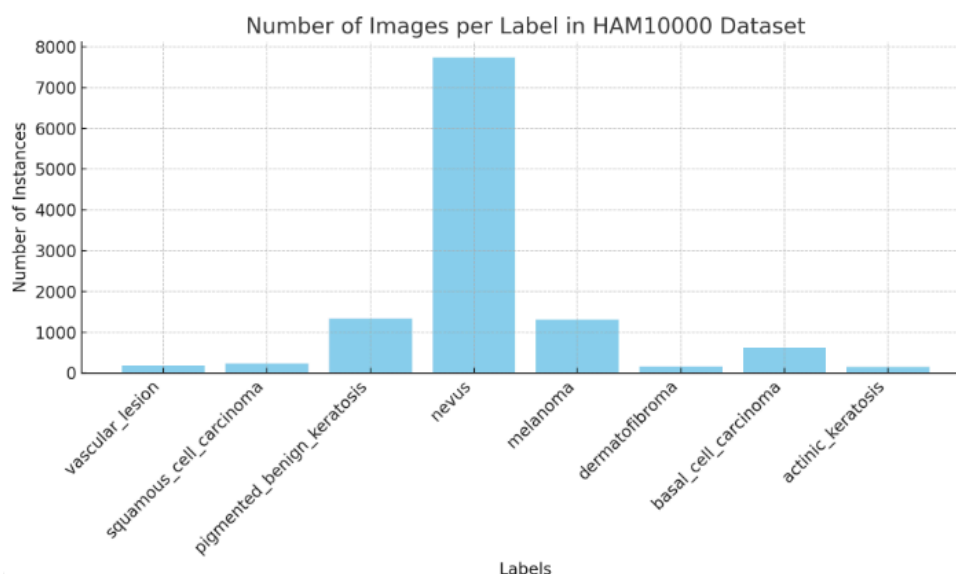


Figura 5. Numărul de imagini per label din setul HAM10000

3.2.2 BCN20000

Setul de date **BCN20000** [20] a fost descărcat și el din ISIC Archive și conține 12.400 de imagini dermatologice (vezi Figura 6). Imaginile din acest set de date au, în general, o rezoluție de 1024x1024 pixeli, oferind o calitate ridicată pentru antrenarea modelelor de învățare automată. Setul de date BCN20000 acoperă următoarele afecțiuni: actinic keratosis, basal cell carcinoma, dermatofibroma, melanoma, nevus, seborrheic keratosis, solar lentigo, squamous cell carcinoma și vascular lesion.

Un dezavantaj major al setului de date BCN20000 este că multe dintre imagini sunt marcate cu o zonă neagră încercuită, ceea ce poate afecta negativ procesul de antrenare a modelelor. Această particularitate vizuală poate introduce zgomot și artefacte în datele de antrenament, reducând astfel eficiența și acuratețea modelului. În plus, am identificat și multe duplicate în acest set de date, ceea ce poate duce la un antrenament redundant și la

suprapotrivirea modelului (overfitting), afectând performanța generală a sistemului de diagnosticare.

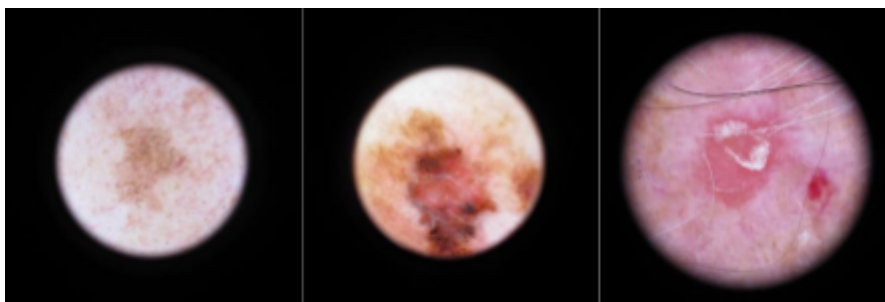


Figura 6. Exemple de imagini din setul BCN20000

3.2.3 Challenge 2019 și Challenge 2020

Seturile de date **ISIC Challenge 2019** [21] și **ISIC Challenge 2020** [22] au fost folosite în cadrul unor competiții internaționale destinate dezvoltării și evaluării modelelor de diagnosticare a afecțiunilor dermatologice. Setul de date din 2019 conține aproximativ 25.000 de imagini, iar cel din 2020 conține aproximativ 33.000 de imagini, ambele fiind împărțite pe etichete similare celor din alte seturi de date, precum actinic keratosis, basal cell carcinoma, dermatofibroma, melanoma, nevus, seborrheic keratosis, squamous cell carcinoma și vascular lesion (vezi Figura 7).

Un dezavantaj major al acestor seturi de date este nebalansarea acestora, cu o discrepanță semnificativă în numărul de imagini per etichetă. De asemenea, rezoluțiile imaginilor variază enorm, de la 640x480 pixeli până la 6000x4000 pixeli, ceea ce complică procesul de preprocesare și antrenare a modelelor. În plus, există multe duplicate vizibile cu ochiul liber, ceea ce poate duce la probleme de suprapotrivire (overfitting).

Un alt obstacol major este reprezentat de fișierul de metadata, care ar trebui să conțină etichetele asociate fiecărei imagini. Cu toate acestea, doar aproximativ 20% din imagini au etichete asociate, ceea ce face lucrul cu aceste seturi de date extrem de dificil și inefficient. Aceste provocări pot explica de ce aceste seturi de date sunt denumite "challenge", reflectând dificultățile pe care le prezintă.

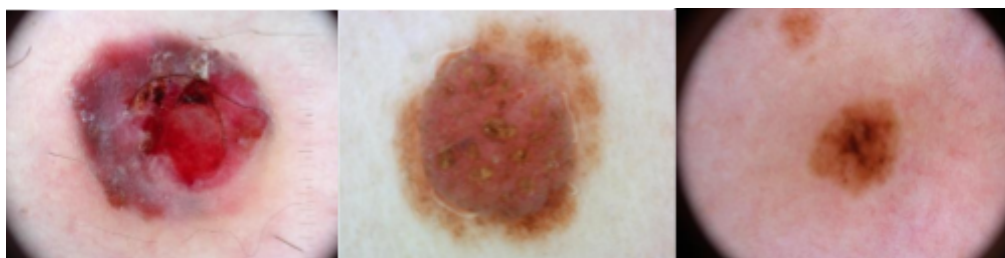


Figura 7. Exemple din seturile Challenge 2019 și 2020

3.2.4 ArsenicSkinImageBD

ArsenicSkinImageBD [23] este un set de date binar obținut de la National Library of Medicine (vezi Figura 8). Acesta conține imagini dermatologice împărțite în două categorii: piele sănătoasă și piele afectată de arsenicoză. Setul de date este utilizat pentru a crea un model binar care să poată decide dacă o imagine a pielii este sănătoasă sau afectată.



Figura 8. Exemple de imagini cu piele sanatoasa din setul ArsenicSkinImageBD

Setul de date ArsenicSkinImageBD include un număr total de 1.096 de imagini, dintre care 741 sunt imagini cu piele infectată și 355 sunt imagini cu piele sănătoasă (vezi Figura 9). Imaginile sunt de rezoluție foarte mare, asigurând astfel detalii suficiente pentru antrenarea precisă a unui model. Acest set de date este esențial pentru dezvoltarea unui model capabil să detecteze dacă pielea este afectată de vreo boală sau nu.



Figura 9. Numărul de imagini din setul de date ArsenicSkinImageBD în funcție de label

3.2.5 Concluzie

După analizarea seturilor de date, s-au luat decizii importante privind setul de date cu care se va merge mai departe pentru implementarea aplicației. Pentru modelul multi-clasă, care decide tipul de afecțiune prezentă în imaginea pielii, a fost ales setul de date HAM10000. Acest set a fost preferat deoarece este robust și pare să nu conțină foarte multe imagini realizate din unghiuri diferite. Fișierul de metadata este complet și corect, iar imaginile au o dimensiune fixă, ceea ce ajută la consistența preprocesării. De asemenea, setul nu include multe imagini cu zgomot sau artefacte nedorite, cum ar fi cercurile negre pe marginea imaginii. În plus, HAM10000 este un set de date relativ mare, oferind un volum considerabil de date pentru antrenarea modelelor. Singura problemă majoră a acestui set este nebalansarea etichetelor, ceea ce va necesita tehnici suplimentare de preprocesare pentru a echilibra datele.

Pentru modelul binar, care verifică dacă pielea este sănătoasă sau nu, s-a ales să se folosească setul de date ArsenicSkinImageBD, fiind singurul set disponibil cu aceste specificații. Cu toate acestea, setul va necesita ajustări pentru a îndeplini cerințele acestui tip de model, asigurând astfel o performanță optimă în detectarea stării de sănătate.

3.3 Preprocesarea seturilor de date alese

3.3.1 Probleme ale seturilor de date alese inițial și soluțiile propuse

Setul de date HAM10000 prezintă câteva probleme notabile. Principala problemă este dezechilibrul major al datelor, majoritatea imaginilor fiind de tipul nevus. Acest dezechilibru poate afecta performanța modelului, ducând la predicții în favoarea etichetei nevus. În plus, printre setul de date au fost identificate imagini care induceau artefacte nedorite, ceea ce poate influența negativ acuratețea modelului.

Pentru a rezolva aceste probleme, s-au întreprins următoarele acțiuni: Imaginile care păreau defecte sau care introduceau artefacte nedorite au fost eliminate manual din setul de date. S-au folosit diferite tehnici de augmentare a datelor pentru a crea mai multe configurații ale setului. Configurațiile includeau fie reducerea numărului de imagini cu eticheta nevus la jumătate, fie creșterea numărului de instanțe pentru celelalte etichete, asigurând un set de date mai echilibrat.

Pentru setul de date binar ArsenicSkinImageBD, problemele majore erau legate de calitatea și relevanța imaginilor. Imaginile cu piele infectată erau specific pentru arsenicoză, ceea ce nu reflecta neapărat toate tipurile de infecții posibile. În plus, setul de date healthy conținea multe duplicate, ceea ce putea duce la o suprapotrivire (overfitting) a modelului.

Pentru a aborda aceste probleme, s-au implementat următoarele soluții: Imaginile care semănau foarte mult între ele în eticheta healthy au fost eliminate manual pentru a reduce redundanța datelor. Imaginile din eticheta not healthy au fost înlocuite cu imagini din setul de date HAM10000. Acest lucru a fost făcut pentru a asigura că modelul are exemple clare de piele infectată cu diverse afecțiuni prezente în setul de date ce conține afecțiunile, asigurând astfel o predictibilitate mai bună.

Aceste soluții au fost implementate pentru a îmbunătăți calitatea și echilibrul seturilor de date, asigurând astfel performanțe mai bune ale modelelor de învățare automată pe parcursul antrenării.

3.3.2 Tehnici de augmentare folosite

Augmentarea datelor [24] este o tehnică utilizată în învățarea automată și procesarea imaginilor pentru a mări diversitatea unui set de date fără a adăuga noi imagini reale. Aceasta se realizează prin aplicarea unor transformări aleatorii asupra imaginilor existente, cum ar fi rotația, translația, ajustarea luminozității, zoom-ul și răsturnările, generând astfel variații ale

acestor imagini. Rolul augmentării este de a îmbunătăți capacitatea modelului de a generaliza pe date noi, reducând riscul de overfitting și făcând modelul mai robust la variațiile din imagini. Aceasta este o listă cu toate tipurile de augmentări folosite:

- Revizuirea și eliminarea manuală a imaginilor care conțin artefacte nedorite sau care sunt de calitate inferioară. Aceasta ajută la menținerea unui set de date curat și fără zgomot care ar putea afecta performanța modelului.
- Rotirea imaginilor cu un anumit unghi, de exemplu, până la 20 de grade în orice direcție. Aceasta introduce variații ale orientării imaginii, ajutând modelul să fie robust la modificările de orientare ale obiectelor din imagini.
- Modificarea luminozității imaginilor într-un anumit interval, cum ar fi între 70% și 120% din luminozitatea originală. Aceasta permite modelului să se adapteze la imagini capturate în condiții de iluminare diferite.
- Deplasarea orizontală a imaginilor cu un anumit procent din lățimea lor. Aceasta introduce variabilitate în poziționarea orizontală a obiectelor în imagini.
- Deplasarea verticală a imaginilor cu un anumit procent din înălțimea lor. Aceasta introduce variabilitate în poziționarea verticală a obiectelor în imagini.
- Aplicarea unui zoom aleatoriu pe imagini, mărin­d sau micșorând dimensiunea obiectelor din imagini. Aceasta ajută modelul să recunoască obiectele la diferite scale și distanțe.
- Inversarea imaginilor pe orizontală. Aceasta ajută la crearea unui set de date mai variat și la reducerea riscului de suprapotrivire pe imagini care au o orientare specifică.
- Inversarea imaginilor pe verticală. Deși mai puțin comună decât răsturnarea orizontală, aceasta poate introduce variabilitate suplimentară în setul de date.
- Umplerea părților goale ale imaginii rezultate în urma augmentărilor (de exemplu, după rotație sau translație) prin reflexia marginilor imaginii. Aceasta păstrează continuitatea vizuală și previne introducerea de artefacte vizuale.
- Reducerea numărului de instanțe [25] din clasele majoritare pentru a echilibra setul de date. Aceasta se realizează prin eliminarea aleatorie a unor instanțe din clasele cu prea multe exemple.
- Creșterea numărului de instanțe din clasele minoritare pentru a echilibra setul de date. Aceasta se realizează prin replicarea sau generarea de noi exemple din clasele subreprezentate, pentru a avea un număr similar de exemple din toate clasele.

3.3.3 Analiza diferitelor configurații ale seturilor de date augmentate

Pentru setul HAM10000, am conceput mai multe configurații augmentate pentru a îmbunătăți echilibrul și diversitatea datelor:

1. **Configurația 1:** Augmentare Medie și Undersampling pe labelul Nevus

Această configurație a implicat reducerea numărului de imagini cu eticheta nevus la jumătate și creșterea numărului de imagini pentru celelalte etichete între 1.000 și 2.000 de imagini (vezi Figura 10), într-un final obținând un număr de 13.104 imagini.

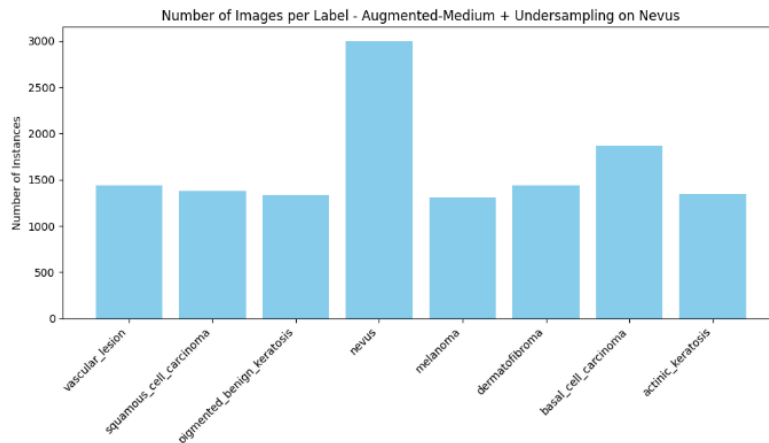


Figura 10. Numărul de imagini din configurația medie per label

2. **Configurația 2:** Augmentare Balanced

În această configurație, toate etichetele au același număr de imagini. Numărul de imagini cu eticheta nevus a fost redus de la peste 7.000 la 1.500, iar celelalte etichete au fost aduse la aproximativ 1.500 de imagini prin augmentare (vezi Figura 11). Setul final balansat are aproximativ 12.000 de imagini.

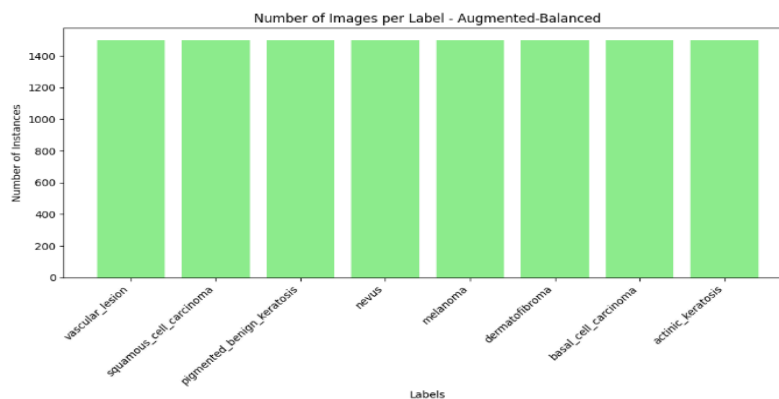


Figura 11. Numărul de imagini din configurația balanced per label

3. Configurația 3: Augmentare Full

Această configurație a păstrat numărul original de 7.000 de imagini pentru nevus și a crescut numărul de imagini pentru celelalte etichete la peste 3.000 de imagini având într-un final un număr de 31.211 de imagini (vezi Figura 12).

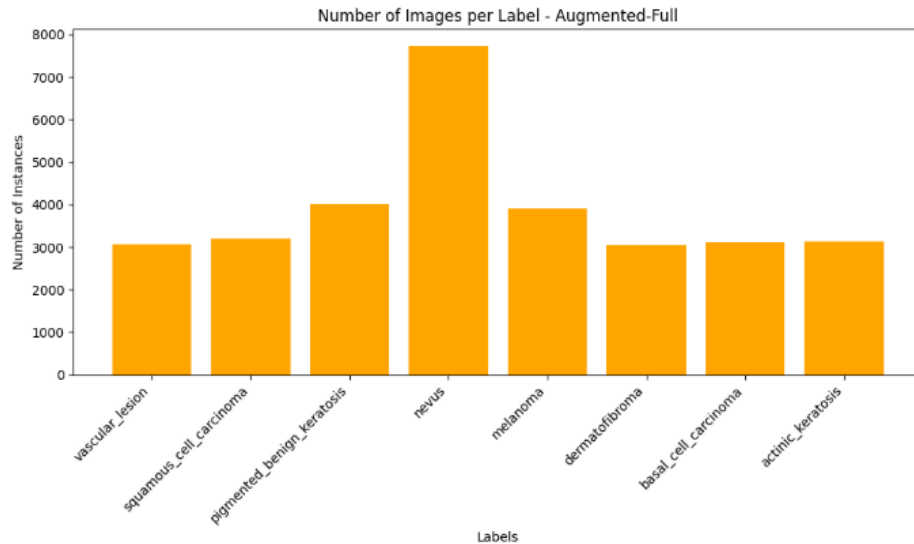


Figura 12. Numărul de imagini din configurația full per label

Pentru modelul binar ArsenicSkinImageBD, am realizat o singură configurație utilizată mai departe. Am eliminat subsetul inițial pentru eticheta not healthy și am adăugat imagini din fiecare categorie din setul HAM10000 pentru a ajunge la un număr decent de imagini de aproximativ 2.960 pentru subsetul not healthy. De la 473 de imagini pe subsetul healthy, am ajuns la 2.958 augmentând fiecare imagine de 4 ori după eliminarea manuală a imaginilor defecte iar în final avem un număr de aproape 6.000 de imagini (vezi Figura 13).

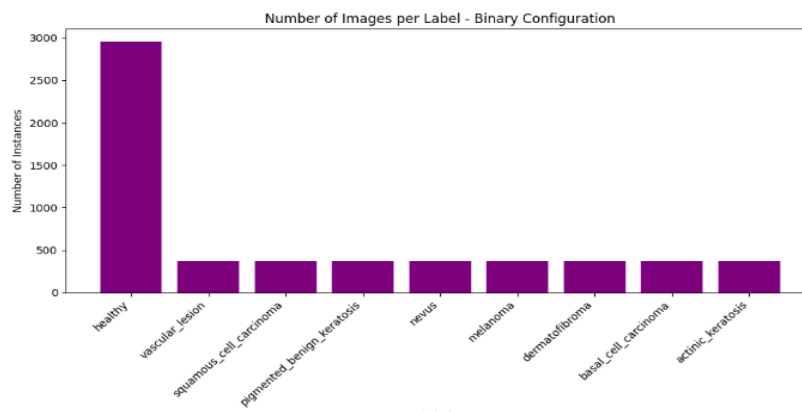


Figura 13. Numărul de imagini din configurația binară pe subcategorii

3.3.4 Concluzie

Aceste configurări nu au fost realizate după tehnici specifice, ci mai degrabă după intuiție, încercând să evităm un număr prea mare de imagini dintr-un singur label pentru a preveni crearea unui model biased către un anumit rezultat. Am încercat să nu generăm un număr exagerat de imagini prin augmentare, deoarece, deși imaginile sunt puțin diferite una de cealaltă, ele se aseamănă foarte mult, iar acest lucru ar putea duce la "duplicatele" nedorite, care pot diminua capacitatea modelului de a generaliza pe date noi. De asemenea, nu s-a dorit crearea unui set prea complex și greu de procesat.

În cele din urmă, s-a decis ca setul de date augmentat și balansat să fie alegerea pentru antrenarea modelului de diagnosticare a condițiilor de piele. Pentru setul binar, am mers pe singura variantă propusă deoarece acest model nu necesită un set prea complex și precis din aceste puncte de vedere. Singurul lucru important pentru acest set binar a fost eliminarea inițială a imaginilor "not healthy" și înlocuirea lor cu imagini din setul HAM10000, pentru a avea un model mai robust și precis.

3.4 Antrenarea modelelor

În acest subcapitol, vom explora noțiunile teoretice folosite în procesul de antrenare a modelelor de învățare automată, problemele apărute inițial și soluțiile implementate. Vom discuta despre tranziția de la utilizarea unui set de straturi custom pentru antrenare la folosirea diferitelor modele pre-antrenate, peste care am adăugat straturi personalizate. De asemenea, vom prezenta rezultatele obținute în urma acestor modificări și optimizări ale procesului de antrenare. Este de menționat ca antrenarea tuturor modelelor a fost făcută având 70% din date pentru antrenare iar 30% pentru validare.

3.4.1 Noțiunile teoretice folosite la antrenare

3.4.1.1 Rețea Neuronală

O rețea neuronală [26] este un model computațional inspirat de structura și funcționarea creierului uman, format dintr-o rețea de unități de calcul simple numite neuroni (vezi Figura 14). Acești neuroni sunt organizați în straturi: stratul de intrare, unul sau mai multe straturi ascunse și stratul de ieșire. Fiecare neuron dintr-un strat este conectat la neuronii din stratul următor prin conexiuni ponderate, care determină influența fiecărui neuron asupra activității neuronilor din straturile ulterioare. Rețelele neurale sunt utilizate pentru a învăța relații complexe și modele în date, prin ajustarea ponderilor acestor conexiuni pe baza erorilor de predicție, proces cunoscut sub numele de antrenament.

Rețelele neuronale sunt extrem de versatile și pot fi aplicate la o gamă largă de probleme, inclusiv clasificare, regresie, recunoaștere de imagini, procesare de limbaj natural și multe altele. Procesul de antrenament implică prezentarea unui set de date de antrenament rețelei și utilizarea unui algoritm de optimizare pentru a minimiza funcția de pierdere, care măsoară diferența dintre predicțiile rețelei și valorile reale. Pe măsură ce rețeaua este antrenată, ajustările ponderilor permit modelului să învețe reprezentări și caracteristici din date, îmbunătățind astfel acuratețea și performanța în sarcinile de predicție.

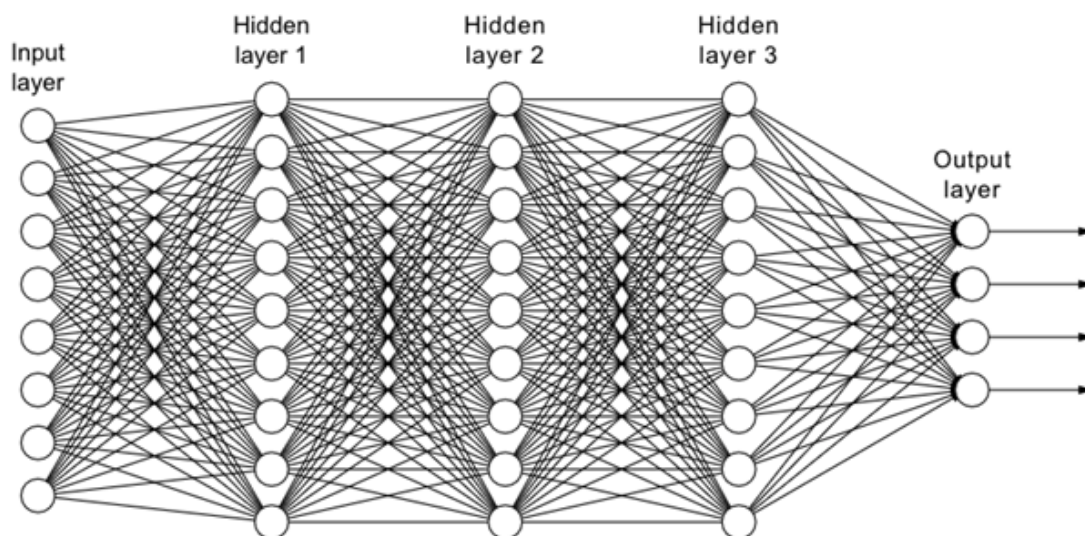


Figura 14. Reprezentare vizuală a unei rețele neuronale

3.4.1.2 Model Secvențial

Un model secvențial este o modalitate simplă de a construi o rețea neuronală prin adăugarea straturilor unul câte unul într-o ordine liniară. Fiecare strat are exact un strat precedent și un strat următor, cu excepția primului și ultimului strat (vezi Figura 15). Această abordare este intuitivă și ușor de implementat, fiind ideală pentru rețelele feed-forward unde fiecare strat contribuie direct la procesarea și transformarea datelor. Modelul secvențial este folosit pentru a construi rețele neurale în care straturile sunt legate în linie, fiind potrivit pentru probleme de clasificare.

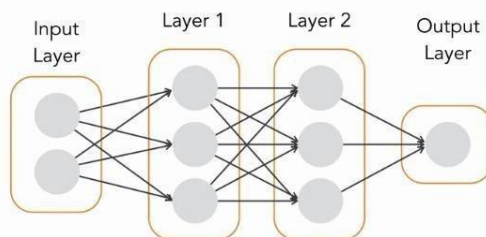


Figura 15. O succesiune de straturi care formează un model secvențial

3.4.1.3 Strat de Intrare

Stratul de intrare este primul strat al rețelei neuronale, care primește datele brute de intrare (în cazul nostru, imaginea propriu-zisă). Acesta definește forma și dimensiunea datelor care vor fi procesate de rețea. Stratul de intrare este esențial pentru a prelua datele în formatul corect și pentru a le transmite către straturile ulterioare pentru prelucrare și extragerea caracteristicilor. Acesta asigură că datele sunt adaptate pentru a putea fi utilizate de rețea și este esențial pentru funcționarea corectă a întregului model.

3.4.1.4 Strat Convoluțional

Un strat convoluțional aplică filtre (kernels) pe datele de intrare pentru a extrage caracteristici locale, cum ar fi muchii, texturi și forme (vezi Figura 16). Filtrele parcurg imaginea printr-o operațiune de convoluție și generează hărți de caracteristici (feature maps) care evidențiază diverse aspecte ale datelor. Straturile convoluționale sunt esențiale pentru rețelele neurale convoluționale (CNN) și sunt utilizate în principal pentru recunoașterea și analiza imaginilor. Acestea permit rețelei să învețe caracteristici vizuale esențiale care sunt cruciale pentru sarcini de clasificare și detectare în imagini.

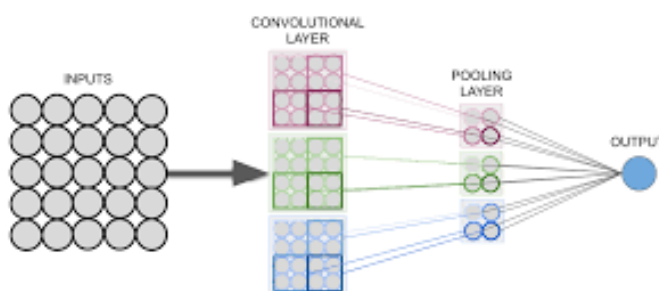


Figura 16. Reprezentarea unui strat convoluțional

3.4.1.5 Strat de Pooling

Un strat de pooling este un strat dintr-o rețea neuronală convoluțională (CNN) care reduce dimensiunile spațiale ale hărților de caracteristici generate de straturile convoluționale anterioare. Aceasta se face prin agregarea valorilor din regiuni vecine ale hărților de caracteristici într-un mod specific, reducând astfel numărul de parametri și calculul necesar pentru straturile ulterioare. Straturile de pooling ajută la controlul overfitting-ului și contribuie la obținerea unei invarianțe la translații și mici deformări în datele de intrare.

Max pooling (vezi Figura 17) este un tip specific de strat de pooling care selectează valoarea maximă din fiecare regiune vecină a hărții de caracteristici. De exemplu, într-o

operațiune de max pooling cu o fereastră de 2x2, fiecare grup de patru valori adiacente din harta de caracteristici este înlocuit cu valoarea cea mai mare dintre acestea. Aceasta permite captarea celor mai puternice caracteristici și reduce dimensiunea hărții de caracteristici, păstrând în același timp informațiile esențiale. Max pooling este frecvent utilizat în rețelele neuronale convoluționale datorită eficienței sale în reducerea dimensiunii și controlului overfitting-ului.

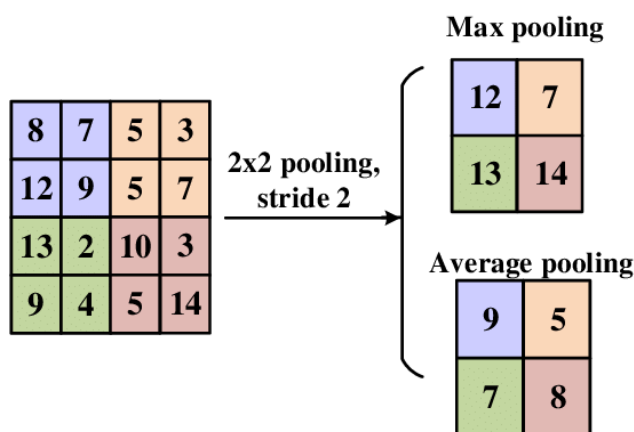


Figura 17. Straturile Max și Average Pooling

3.4.1.6 Stratul Dens

Straturile dense sunt straturi complet conectate în care fiecare neuron este conectat la toți neuronii din stratul precedent. Aceste straturi sunt responsabile pentru combinarea caracteristicilor extrase de straturile anterioare și pentru realizarea clasificării finale. Straturile dense folosesc funcții de activare precum ReLU (Rectified Linear Unit) pentru a introduce non-liniaritate în rețea și funcții de activare softmax sau sigmoid pentru a produce probabilitățile finale de clasificare. Aceste straturi sunt esențiale pentru obținerea unei decizii finale bazate pe caracteristicile.

3.4.1.7 Funcții de Activare

Funcțiile de activare [27] sunt componente esențiale ale rețelelor neuronale, care sunt aplicate la ieșirile neuronilor pentru a introduce non-liniaritate în model. Aceste funcții permit rețelei să învețe și să modeleze relații complexe între datele de intrare și etichetele de ieșire. Fără funcțiile de activare, rețeaua ar fi limitată la modele liniare simple, care nu ar putea captura corect relațiile non-liniare prezente în datele reale. Funcțiile de activare transformă valorile brute ale neuronilor într-un interval specific, facilitând astfel procesul de învățare și optimizare a modelului.

Printre cele mai comune funcții de activare se numără ReLU (Rectified Linear Unit), care activează neuronii doar pentru valori pozitive, oferind astfel o simplă și eficientă metodă de introducere a non-liniarității. Sigmoid este o altă funcție populară, care mapează valorile între 0 și 1, fiind utilă în special pentru probleme de clasificare binară (vezi Figura 18). Funcția softmax transformă ieșirile unui strat într-un set de probabilități, fiecare corespunzând unei clase diferite, fiind utilizată în clasificarea multi-clasă.

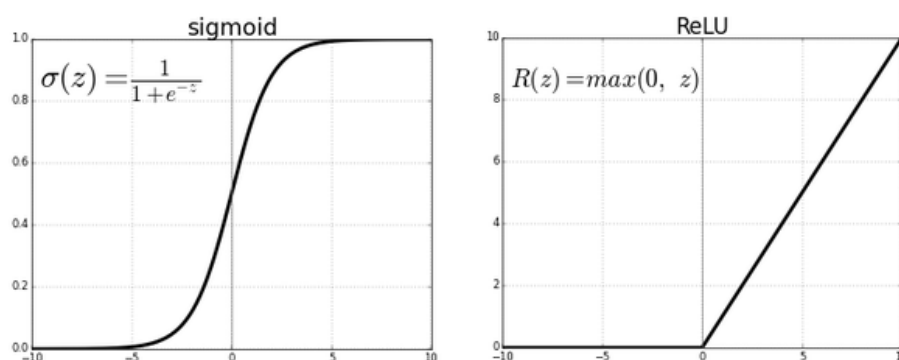


Figura 18. Reprezentarea pe grafic a funcțiilor Sigmoid și ReLU

3.4.1.8 Rate de Învățare

Rata de învățare este un hiperparametru esențial în antrenarea rețelelor neuronale, care controlează cât de mult sunt ajustate greutatele modelului în funcție de eroarea observată la fiecare pas de antrenament. Aceasta determină dimensiunea pașilor pe care algoritmul de optimizare îi face în direcția minimizării funcției de pierdere. O rată de învățare adecvată poate accelera procesul de antrenament și poate asigura convergența la o soluție optimă (vezi Figura 19). Alegerea unei rate de învățare potrivite este crucială pentru succesul antrenamentului modelului. Dacă rata de învățare este prea mare, modelul poate să sară peste soluția optimă, ducând la oscilații și instabilitate în procesul de antrenament.

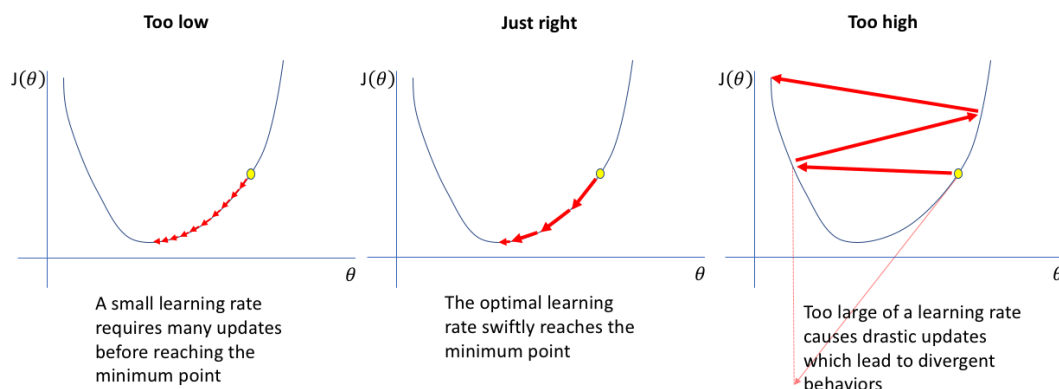


Figura 19. Reprezentarea diferitelor mărimi ale ratei de învățare

Pe de altă parte, dacă rata de învățare este prea mică, antrenamentul poate deveni foarte lent și modelul poate rămâne blocat într-un minim local, fără să ajungă la performanța maximă posibilă. Adesea, se utilizează strategii adaptative de ajustare a ratei de învățare, cum ar fi `ReduceLROnPlateau`, care micșorează rata de învățare atunci când îmbunătățirile în pierdere devin nesemnificative, pentru a optimiza procesul de antrenament.

3.4.1.9 Funcții de Optimizare

Funcțiile de optimizare sunt algoritmi esențiali pentru ajustarea greutateilor rețelei neuronale astfel încât să se minimizeze funcția de pierdere, care măsoară diferența dintre predicțiile modelului și valorile reale. Acești algoritmi determină modul în care greutateile sunt actualizate în timpul antrenamentului, influențând astfel eficiența și viteza de convergență a modelului. Una dintre cele mai utilizate funcții de optimizare este Adam (Adaptive Moment Estimation), care combină avantajele algoritmilor AdaGrad și RMSProp, adaptând ratele de învățare pentru fiecare parametru și accelerând convergența.

3.4.1.10 Batch și Epoci

În procesul de antrenare a rețelelor neuronale, batch-ul și epocile sunt două concepte esențiale. Un batch reprezintă numărul de exemple de antrenament utilizate pentru a actualiza o singură dată parametrii modelului. În loc să treacă prin întregul set de date pentru fiecare actualizare, algoritmul de optimizare ajustează greutateile după ce procesează fiecare batch. O epocă reprezintă o trecere completă prin întregul set de date de antrenament, ceea ce înseamnă că toate exemplele din setul de date sunt utilizate o dată pentru a actualiza modelul.

Alegerea unui număr prea mic de batch-uri (batch size mic) poate duce la fluctuații mari ale funcției de pierdere, ceea ce face ca antrenamentul să fie mai instabil. Pe de altă parte, un batch size prea mare poate necesita mai multă memorie și poate reduce capacitatea modelului de a generaliza, deoarece ajustările greutateilor sunt mai puțin frecvente. Alegerea optimă a dimensiunii batch-ului depinde de specificul datelor și de resursele hardware disponibile.

În ceea ce privește epocile, alegerea unui număr prea mic de epoci poate face ca modelul să fie subantrenat, adică să nu învețe suficient din datele de antrenament. În schimb, un număr prea mare de epoci poate duce la supraantrenare (overfitting), unde modelul învață prea bine detaliile și zgomotul din setul de date de antrenament, ceea ce îi reduce performanța

pe datele noi. Găsirea echilibrului corect prin experimentare și validare este esențială pentru a obține un model bine antrenat și capabil să generalizeze corect.

3.4.1.11 Funcții de Pierdere

Funcțiile de pierdere, sau funcțiile cost, sunt esențiale în antrenarea rețelelor neuronale, deoarece măsoară diferența dintre predicțiile modelului și valorile reale ale datelor de antrenament. Aceasta diferență este folosită pentru a ajusta greutatea modelului, permițându-i să învețe din erorile sale și să îmbunătățească performanța.

Există diverse tipuri de funcții de pierdere, fiecare fiind potrivită pentru anumite tipuri de probleme. Categorical Crossentropy este utilizată în principal pentru probleme de clasificare multi-clasă, unde obiectivul este de a atribui fiecare exemplu unei singure clase dintre mai multe posibile. Binary Crossentropy este folosită pentru probleme de clasificare binară, unde fiecare exemplu este atribuit uneia dintre cele două clase.

3.4.1.12 Entropie

Din punct de vedere teoretic, entropia este o măsură a incertitudinii sau a aleatorietății asociate cu o variabilă aleatoare sau cu un set de date. În contextul teoriei informației, entropia cuantifică cantitatea de informație necesară pentru a descrie starea unui sistem. În învățarea automată și în rețelele neuronale, entropia este adesea utilizată în formularea funcțiilor de pierdere, cum ar fi entropia încrucișată (cross-entropy), care măsoară diferența între distribuțiile probabilistice prezise de model și distribuțiile reale (vezi Figura 20). Entropia ajută la evaluarea a cât de bine un model capturează incertitudinea din datele de intrare și la îmbunătățirea performanței acestuia prin optimizare.

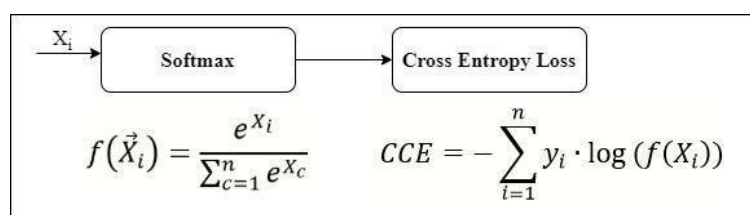


Figura 20. Calculul Cross Entropy Loss folosind funcția Softmax

3.4.1.13 Precizie, Recall si Scor F1

Precizia este o măsură a acurateței unui model de clasificare, reprezentând proporția de predicții corecte pentru o anumită clasă față de toate predicțiile realizate pentru acea clasă

(vezi Figura 21). O precizie mare indică faptul că un model face puține predicții fals pozitive, fiind importantă în aplicațiile unde costul unei alarme false este ridicat.

Recall-ul, sau sensibilitatea, măsoară proporția de exemple pozitive corect identificate de model din toate exemplele pozitive reale. Un recall mare indică faptul că modelul detectează majoritatea cazurilor pozitive, fiind esențial în situațiile în care ratările (false negative) trebuie minimizate, cum ar fi în diagnosticarea medicală.

Scorul F1 este media armonică dintre precizie și recall, oferind o măsură echilibrată a performanței modelului, mai ales atunci când există un dezechilibru între clase. Scorul F1 este util pentru a evalua cât de bine se comportă un model în clasificarea exemplelor pozitive, luând în considerare atât precizia, cât și recall-ul, oferind o imagine completă a performanței modelului.

		POSITIVE	NEGATIVE
ACTUAL VALUES	POSITIVE	TP	FN
	NEGATIVE	FP	TN

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Figura 21. Formulele de calcul pentru Precizie, Recall și Scor F1

3.4.1.14 Matrice de Confuzie

Matricea de confuzie este un instrument de evaluare utilizat în învățarea automată pentru a vizualiza performanța unui model de clasificare. Aceasta este o matrice de dimensiuni $n \times n$, unde n reprezintă numărul de clase, și în care fiecare celulă (i, j) indică numărul de exemple din clasa i care au fost clasificate ca fiind din clasa j (vezi Figura 22). Elementele diagonale ale matricei reprezintă exemplele corect clasificate, în timp ce elementele non-diagonale indică erorile de clasificare. Matricea de confuzie oferă o vedere detaliată asupra modului în care un model se descurcă pe fiecare clasă, evidențiind atât predicțiile corecte, cât și erorile de clasificare, permițând astfel o analiză detaliată a performanței modelului și identificarea ariilor unde acesta poate fi îmbunătățit.

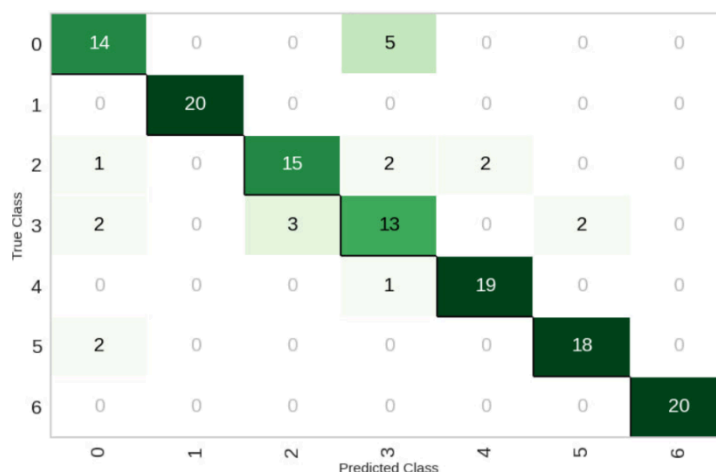


Figura 22. Exemplu de matrice de confuzie

3.4.2 Încercările inițiale și decizia folosirii unui model pre-antrenat

Pentru antrenarea modelului multi-clasă destinat diagnosticării afecțiunilor dermatologice, am început cu o arhitectură de model custom și diverse ajustări ale hiperparametrilor. Scriptul inițial a utilizat un model secvențial cu mai multe straturi convoluționale și dense (vezi Figura 23), împreună cu tehnici de regularizare și augmentare a datelor. Cu toate acestea, rezultatele obținute au fost mediocre, cu un scor F1 maxim de aproximativ 60% în medie. Acest rezultat suboptimal a ridicat întrebarea dacă performanțele modelului ar putea fi îmbunătățite semnificativ prin utilizarea modelelor pre-antrenate, dat fiind că modelele custom nu puteau atinge performanțele dorite.

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(target_size[0], target_size[1], 3), padding='same'),
    BatchNormalization(),
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(64, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Conv2D(128, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(len(train_generator.class_indices), activation='softmax')
```

Figura 23. Arhitectura modelului inițial pe setul HAM10000

În următoarele două figuri sunt prezentate valorile obținute în urma încercărilor inițiale. Matricea de confuzie (vezi Figura 24) și raportul de clasificare (vezi Figura 25) arată limitările modelului custom, cum ar fi acuratețea scăzută și variabilitatea ridicată între diferitele clase. De exemplu, precizia pentru anumite clase era foarte scăzută, indicând dificultăți în clasificarea corectă a acestor afecțiuni. Scriptul de antrenare utilizat a inclus tehnici standard de preprocesare a imaginilor și ajustare a hiperparametrilor, dar nu a reușit să producă un model suficient de performant pentru utilizare practică.

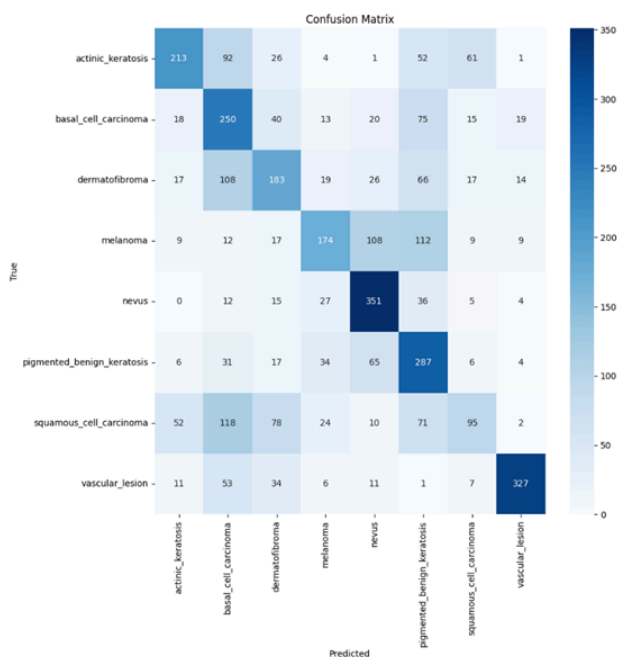


Figura 24. Matricea de confuzie pentru modelul inițial

	precision	recall	f1-score	support
actinic_keratosi	0.65	0.47	0.55	450
basal_cell_carcinoma	0.37	0.56	0.44	450
dermatofibroma	0.45	0.41	0.43	450
melanoma	0.58	0.39	0.46	450
nevus	0.59	0.78	0.67	450
pigmented_benign_keratosi	0.41	0.64	0.50	450
squamous_cell_carcinoma	0.44	0.21	0.29	450
vascular_lesion	0.86	0.73	0.79	450
accuracy			0.52	3600
macro avg	0.54	0.52	0.52	3600
weighted avg	0.54	0.52	0.52	3600

Figura 25. Valorile pentru acuratețe, recall și scor F1 în funcție de label

Într-un final, s-au încercat diferite configurații de arhitectură de model și ajustarea parametrilor, dar după multe încercări tot nu au fost obținute rezultate satisfăcătoare. Deși am experimentat cu diverse straturi convoluționale, normalizări, dropout, ajustând hiperparametri precum rata de învățare, numărul de epoci și dimensiunea batch-urilor, performanța modelului a rămas sub așteptări. În ciuda eforturilor considerabile și a optimizărilor aplicate, modelele custom au continuat să ofere un F1 score maxim de aproximativ 60%, ceea ce nu era suficient pentru a asigura o acuratețe și o fiabilitate adecvată pentru diagnosticarea afecțiunilor dermatologice.

Astfel, a fost luată decizia folosirii unor modele pre-antrenate, deoarece acestea sunt antrenate pe seturi foarte mari de date și sunt construite pentru a fi capabile de detectarea eficientă a caracteristicilor în imagini. Peste aceste modele pre-antrenate, se adaugă straturi custom la final, adaptate specific pentru sarcina noastră de clasificare a afecțiunilor dermatologice, pentru a îmbunătăți performanța.

3.4.3 Model pre-antrenat VGG19

VGG19 este o arhitectură de rețea neuronală convoluțională (CNN) dezvoltată de echipa de cercetare de la Visual Geometry Group (VGG) de la Universitatea Oxford. Este cunoscută pentru performanța sa excelentă în recunoașterea și clasificarea imaginilor, demonstrată în competiția ImageNet Large Scale Visual Recognition Challenge (ILSVRC) din 2014.

VGG19 constă din 19 straturi ponderate, care includ 16 straturi convoluționale și 3 straturi complet conectate (dense). Această arhitectură folosește filtre convoluționale mici de 3x3 pixeli, cu pas de 1 pixel și padding de 1 pixel, pentru a păstra dimensiunile spațiale ale imaginilor de intrare. După fiecare două sau trei straturi convoluționale, urmează un strat de pooling, de obicei max pooling, care reduce dimensiunile spațiale ale hărților de caracteristici. Ultimele trei straturi sunt dense și sunt urmate de un strat softmax pentru clasificarea finală.

Unul dintre avantajele majore ale utilizării VGG19 ca model pre-antrenat este faptul că acesta a fost antrenat pe un set foarte mare de date (ImageNet), care conține milioane de imagini și mii de clase. Acest lucru înseamnă că VGG19 a învățat deja caracteristici vizuale complexe care sunt relevante pentru o gamă largă de imagini.

3.4.4 Model pre-antrenat MobileNetV3

MobileNetV3 [28] este o arhitectură de rețea neuronală convoluțională dezvoltată de Google, concepută pentru a oferi performanțe ridicate cu eficiență computațională optimizată,

fiind ideală pentru aplicații pe dispozitive mobile și integrate. Arhitectura sa utilizează o combinație de straturi convoluționale și straturi de inversare a rezoluției (inverted residual blocks), alături de tehnici avansate precum squeeze-and-excitation (SE) blocks și activări non-liniare hard-swish.

Este specializat în realizarea de clasificări eficiente din punct de vedere al resurselor, menținând în același timp o performanță comparabilă cu modelele mai mari și mai complexe. A fost antrenat pe setul de date ImageNet, ceea ce înseamnă că și el a învățat să recunoască și să extragă caracteristici vizuale relevante dintr-o gamă largă de imagini.

3.4.5 Modelele pre-antrenate DenseNet169 și DenseNet201

DenseNet169 și DenseNet201 sunt ambele arhitecturi de rețea neuronală convoluțională din familia DenseNet, recunoscute pentru conectivitatea lor densă între straturi. În aceste modele, fiecare strat primește hărțile de caracteristici de la toate straturile anterioare, ceea ce îmbunătățește propagarea gradientului și reutilizarea caracteristicilor.

DenseNet169 are 169 de straturi ponderate, în timp ce DenseNet201 are 201 straturi ponderate. Structura mai adâncă a DenseNet201 îi permite să învețe caracteristici și mai complexe și detaliate din datele de intrare, ceea ce poate duce la o performanță superioară în clasificarea și recunoașterea imaginilor. Cu toate acestea, adâncimea suplimentară a DenseNet201 vine cu un cost computațional mai mare, necesitând mai multe resurse pentru antrenare și inferență comparativ cu DenseNet169.

În practică, alegerea între DenseNet169 și DenseNet201 depinde de cerințele specifice ale aplicației și de resursele disponibile. DenseNet201 este ideal pentru scenarii care necesită o precizie extrem de ridicată și capacitatea de a învăța din seturi de date foarte complexe, în timp ce DenseNet169 oferă un echilibru mai bun între performanță și eficiență, fiind potrivită pentru situații în care resursele sunt mai limitate.

3.4.6 Model pre-antrenat InceptionResNetV2

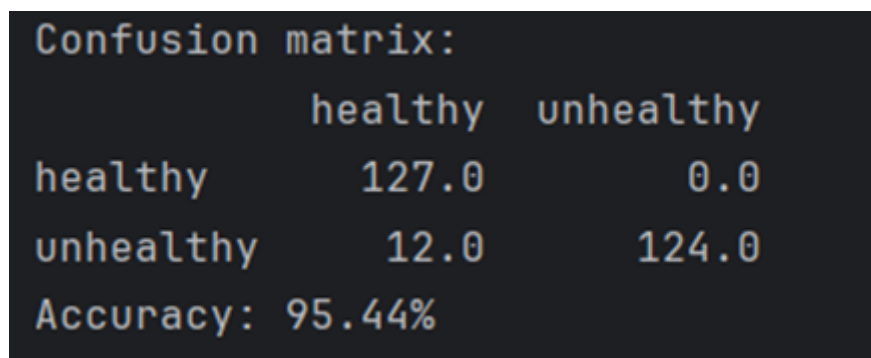
InceptionResNetV2 este o arhitectură de rețea neuronală dezvoltată de Google, care combină modulele Inception și conexiunile reziduale din ResNet. Acest model utilizează module Inception pentru a extrage caracteristici la diverse scale și conexiuni reziduale pentru a facilita antrenarea rețelelor adânci, prevenind problemele de gradient disipat sau exploziv. Structura InceptionResNetV2 include 164 de straturi ponderate și folosește activarea ReLU.

Modelul începe cu un strat convoluțional mare, urmat de module Inception și straturi reziduale, și se încheie cu straturi dense și un strat softmax pentru clasificare. Această

arhitectură permite extragerea de caracteristici complexe și detaliate din imagini, menținând în același timp eficiența antrenamentului. InceptionResNetV2 este ideal pentru aplicații de recunoaștere vizuală avansată, cum ar fi clasificarea imaginilor și detectarea obiectelor, datorită combinației sale de putere de reprezentare și stabilitate.

3.4.7 Antrenarea modelului binar

Pentru antrenarea modelului binar [29] destinat clasificării imaginilor ca fiind healthy sau not healthy, am utilizat configurația augmentată a setului de date ArsenicSkinImageBD discutată anterior. În acest caz, nu am folosit modele pre-antrenate, deoarece diferențele dintre imaginile "healthy" și "not healthy" erau foarte ușor de recunoscut. Acest context specific ne-a permis să dezvoltăm o soluție custom de arhitectură, fără a necesita complexitatea și resursele suplimentare asociate cu modelele pre-antrenate.



The image shows a terminal-style output of a confusion matrix and accuracy. The text is as follows:

```
Confusion matrix:
              healthy  unhealthy
healthy      127.0     0.0
unhealthy    12.0     124.0
Accuracy: 95.44%
```

	healthy	unhealthy
healthy	127.0	0.0
unhealthy	12.0	124.0

Accuracy: 95.44%

Figura 26. Acuratețea obținută la antrenarea modelului binar

Am reușit să obținem o precizie de aproape 96% (vezi Figura 26) prin ajustarea hiperparametrilor și folosind tehnici de oprire prematură a antrenării pentru a preveni supra-antrenarea modelului. Procesul de antrenare a implicat experimentarea cu diferite structuri de rețea, folosind straturi convoluționale, dense, normalizări și optimizarea parametrilor de antrenament, cum ar fi rata de învățare și dimensiunea batch-urilor, până când am obținut cel mai bun rezultat posibil. Abordarea personalizată din figura de mai jos a fost suficientă pentru a satisface cerințele de performanță ale aplicației noastre.

```

model = Sequential([
    Conv2D(16, (3, 3), activation='relu', input_shape=(target_size[0], target_size[1], 3), padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    Conv2D(32, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(64, activation='relu'),
    Dropout(0.5),
    BatchNormalization(),

    Dense(1, activation='sigmoid')
])

```

Figura 27. Arhitectura finală a modelului binar

3.4.8 Antrenarea modelului multi-clasă

Pentru a antrena modelul multi-clasă [30] destinat diagnosticării condițiilor pielii, s-au folosit diverse combinații de structuri, îmbinând antrenarea unui număr arbitrar de straturi din modelul pre-antrenat și adăugând peste acestea straturi custom. Antrenarea acestui model s-a realizat utilizând mai multe modele pre-antrenate: VGG19, MobileNetV3, DenseNet169, DenseNet201 și InceptionResNetV2.

Din păcate, pentru modelele VGG19 și MobileNetV3, am obținut rezultate neașteptat de slabe, mai slabe chiar decât cele ale arhitecturii custom folosite inițial. MobileNetV3 a avut o performanță foarte slabă, cu o acuratețe de doar 12% la validare sau overfitting agresiv. Similar, VGG19 a avut o performanță slabă, cu o acuratețe de 30% la validare și overfitting. De menționat faptul că la antrenarea modelelor multi-clasă s-au folosit un număr de 20 epoci în batch-uri de câte 16.

În schimb, pentru celelalte trei modele pre-antrenate am obținut rezultate bune:

- DenseNet169: a obținut un scor F1 de 77%, demonstrând o capacitate solidă de a clasifica corect imaginile dermatologice.

```

base_model = DenseNet169(include_top=False, weights='imagenet', input_shape=(target_size[0], target_size[1], 3))
for layer in base_model.layers[:-40]:
    layer.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Dropout(0.8)(x)
x = Dense(256)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Dropout(0.7)(x)

predictions = Dense(len(train_generator.class_indices), activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

model.compile(
    loss=CategoricalCrossentropy(),
    optimizer=Adam(learning_rate=0.00007),
    metrics=['accuracy', tf.keras.metrics.Precision(), tf.keras.metrics.Recall(), F1Score()]
)

```

Figura 28. Arhitectura finală a modelului multi-clasă bazat pe DenseNet169

	precision	recall	f1-score	support
actinic_keratosi	0.89	0.82	0.85	450
basal_cell_carcinoma	0.75	0.86	0.80	450
dermatofibroma	0.89	0.82	0.85	450
melanoma	0.72	0.47	0.57	450
nevus	0.66	0.90	0.76	450
pigmented_benign_keratosi	0.64	0.78	0.70	450
squamous_cell_carcinoma	0.83	0.72	0.77	450
vascular_lesion	0.92	0.84	0.88	450
accuracy			0.78	3600
macro avg	0.79	0.78	0.77	3600
weighted avg	0.79	0.78	0.77	3600

Figura 29. Valorile pentru acuratețe, recall și scor F1 în funcție de label (DenseNet169)

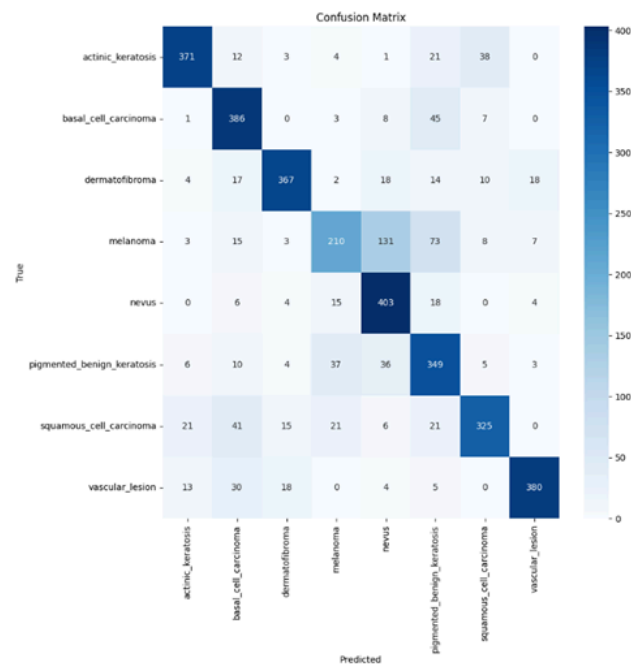


Figura 30. Matricea de confuzie (DenseNet169)

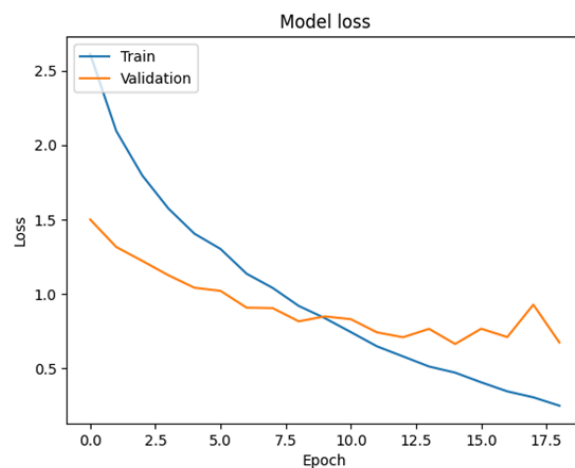


Figura 31. Graficul funcției de pierdere la antrenare și validare (DenseNet169)

- DenseNet201: a avut o performanță și mai bună, cu un scor F1 de 78%, indicând o capacitate ușor superioară de generalizare.

```
base_model = DenseNet201(include_top=False, weights='imagenet', input_shape=(target_size[0], target_size[1], 3))

for layer in base_model.layers[:-20]:
    layer.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)

x = Dense(256)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Dropout(0.7)(x)

predictions = Dense(len(train_generator.class_indices), activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
model.compile(
    loss=CategoricalCrossentropy(),
    optimizer=Adam(learning_rate=0.00007),
    metrics=['accuracy', tf.keras.metrics.Precision(), tf.keras.metrics.Recall(), F1Score()]
)
```

Figura 32. Arhitectura finală a modelului multi-clasă bazat pe DenseNet201

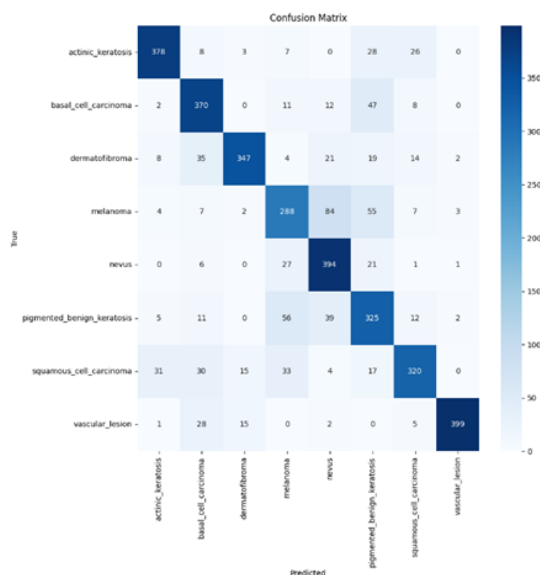


Figura 33. Graficul funcției de pierdere la antrenare și validare (DenseNet201)

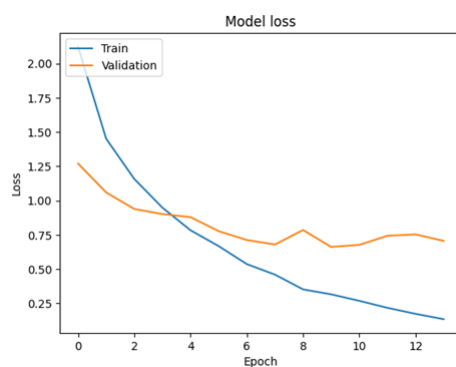


Figura 34. Graficul funcției de pierdere la antrenare și validare (DenseNet201)

- InceptionResNetV2: a fost cel mai performant model, obținând un scor F1 de 80%, arătând cea mai bună capacitate de clasificare a afecțiunilor pielii.

```
base_model = InceptionResNetV2(include_top=False, weights='imagenet', input_shape=(target_size[0], target_size[1], 3))
for layer in base_model.layers[:-100]:
    layer.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)

predictions = Dense(len(train_generator.class_indices), activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
model.compile(optimizer=Adam(learning_rate=0.0006),
              loss=CategoricalCrossentropy(),
              metrics=['accuracy', tf.keras.metrics.Precision(), tf.keras.metrics.Recall(), F1Score()])
```

Figura 35. Arhitectura finală a modelului multi-clasă bazat pe InceptionResNetV2

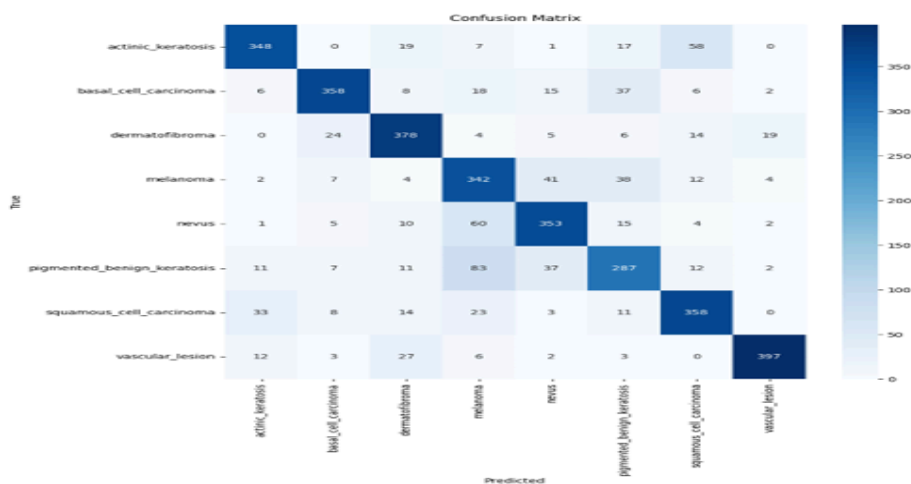


Figura 36. Matricea de confuzie (InceptionResNetV2)

	precision	recall	f1-score	support
actinic_kerato	0.84	0.77	0.81	450
basal_cell_carcinoma	0.87	0.80	0.83	450
dermatofibroma	0.80	0.84	0.82	450
melanoma	0.63	0.76	0.69	450
nevus	0.77	0.78	0.78	450
pigmented_benign_kerato	0.69	0.64	0.66	450
squamous_cell_carcinoma	0.77	0.80	0.78	450
vascular_lesion	0.93	0.88	0.91	450
accuracy			0.78	3600
macro avg	0.79	0.78	0.78	3600
weighted avg	0.79	0.78	0.78	3600

Figura 38. Valorile pentru acuratețe, recall și scor F1 în funcție de label (InceptionResNetV2)

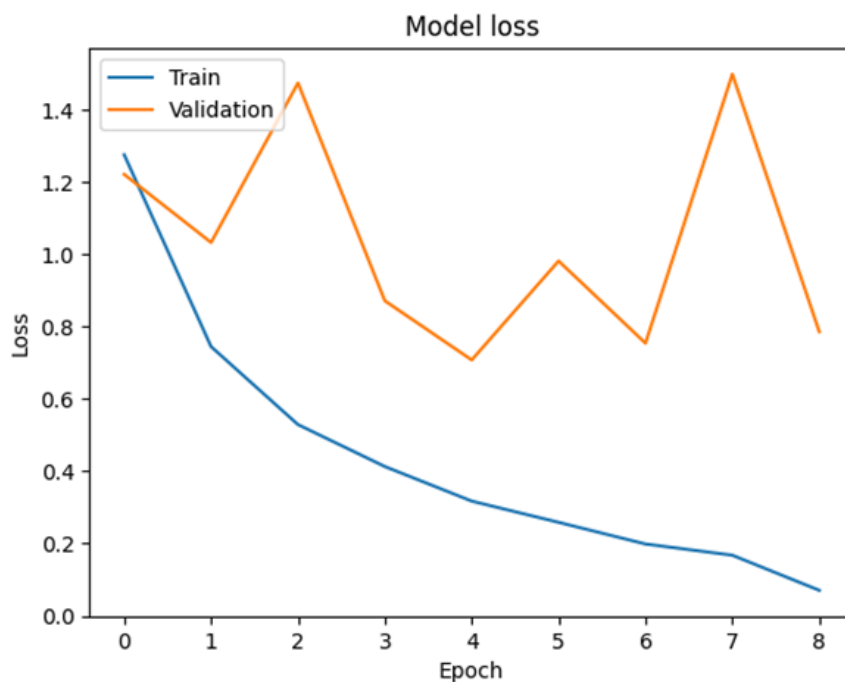


Figura 39. Graficul funcției de pierdere la antrenare și validare (InceptionResNetV2)

3.4.9 Concluzie

În această secțiune sunt prezentate doar câteva rezultate ale antrenării, dar în realitate au fost efectuate zeci de astfel de antrenări până să obținem un rezultat satisfăcător. Am observat că modelele pre-antrenate au fost soluția creșterii preciziei modelului nostru. În cele din urmă, am obținut un model binar cu o precizie mare pentru a verifica dacă pielea este sănătoasă sau nu, și trei modele de precizie relativ mare pentru clasificarea condițiilor pielii, fiecare bazându-se pe diferite modele pre-antrenate (DenseNet169, DenseNet201 și InceptionResNetV2). Aceste modele vor fi folosite mai departe în aplicația noastră pentru diagnosticarea afecțiunilor dermatologice.

Capitolul 4: Arhitectura și implementarea aplicației

În acest capitol, vom discuta despre structura principalelor componente ale aplicației și modul în care acestea se îmbină pentru a crea o soluție funcțională. Vom analiza fiecare componentă a arhitecturii, explicând rolul și funcționalitatea fiecăreia, precum și modul în care acestea colaborează pentru a forma un sistem complet.

4.1 Structura întregului sistem

Sistemul este alcătuit din mai multe componente interconectate între ele, fiecare având un rol specific în funcționarea aplicației. Componenta de Flutter reprezintă aplicația mobilă, prin intermediul căreia utilizatorii pot încărca imagini și primi diagnostice. Serverul se ocupă cu procesarea requesturilor, asigurând comunicarea între aplicație și restul sistemului. În Azure, avem o coadă pentru gestionarea cererilor de procesare și un container pentru stocarea imaginilor încărcate. Baza de date MongoDB este utilizată pentru stocarea informațiilor, în timp ce procesatorul de imagini se ocupă de partea de diagnosticare a condițiilor pielii.

Următoarele diagrame C1, C2 și C3 [31] ar trebui să ofere o perspectivă mai bună asupra arhitecturii aplicației și a modului în care componentele sale interacționează (vezi Figurile 40, 41 și 42). Aceste diagrame vor ilustra structura și conexiunile principale dintre diferitele părți ale sistemului.

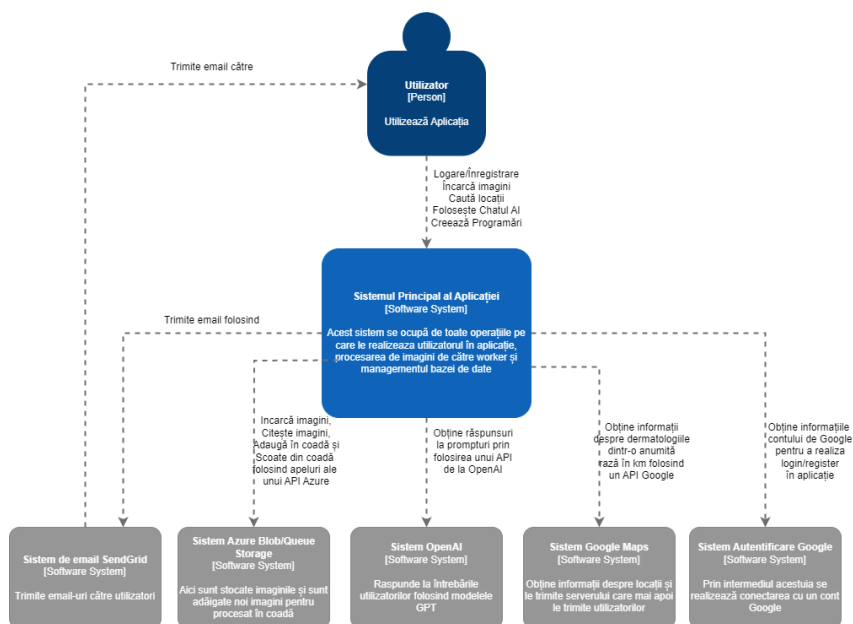


Figura 40. Diagrama C1 (context) a aplicației

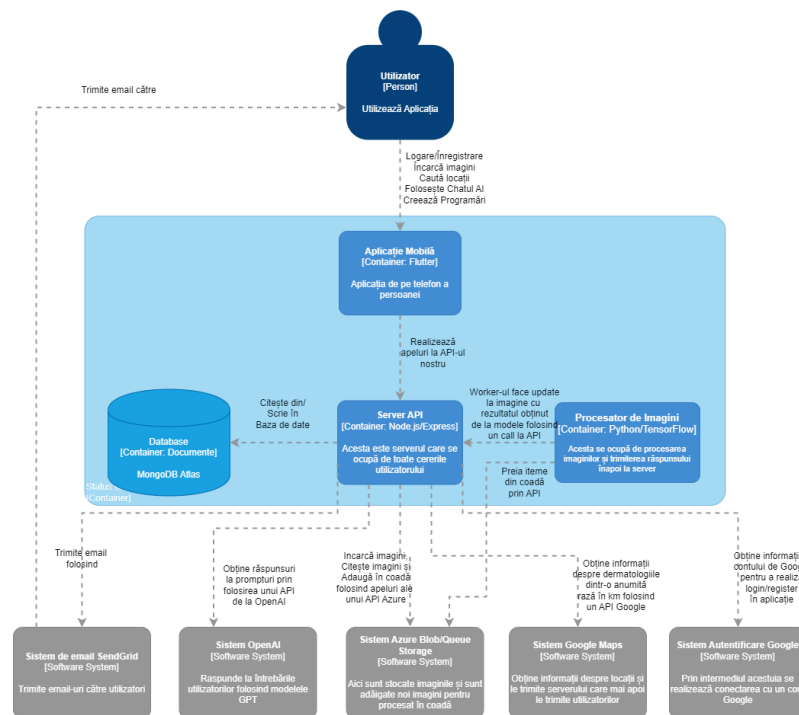


Figura 41. Diagrama C2 (container) a aplicației

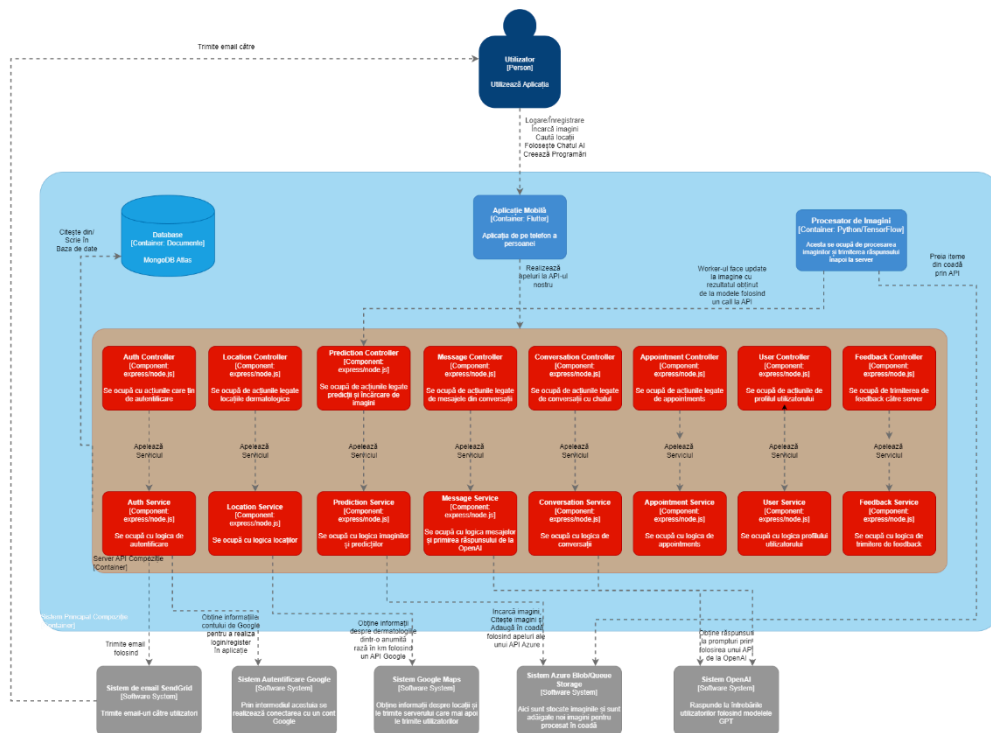


Figura 42. Diagrama C3 (component) a aplicației

Următoarea diagramă scoate în evidență modul în care diferiți actori folosesc sistemul, demonstrând fluxurile de interacțiuni și rolurile fiecăruia în cadrul aplicației.

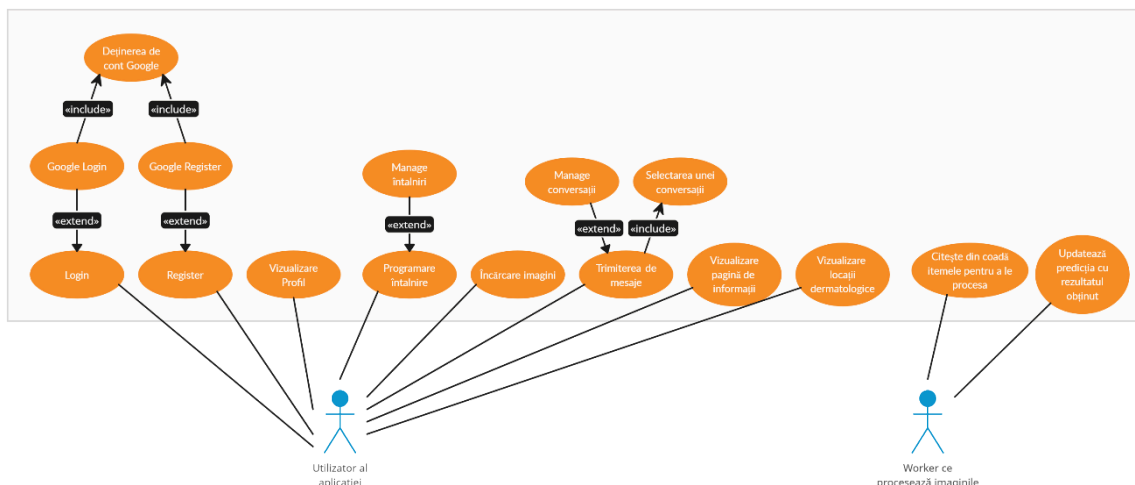


Figura 43. Diagrama cazurilor de utilizare

4.2 Structura serverului

Partea de Server/API este scrisă în Node.js/JavaScript folosind framework-ul Express și oferă mai multe servicii care interacționează atât cu utilizatorii de aplicații, cât și cu procesatorul de imagini. Serverul este format din următoarele servicii:

- Serviciul de autentificare care se ocupă cu logarea și înregistrarea utilizatorilor, atât prin aplicație cât și prin Google. De asemenea, include funcționalitatea de recuperare sau resetare a parolei. Sesiunea se realizează prin token-uri JWT și include refresh tokens care sunt stocate într-o tabelă separată.
- Serviciul de utilizatori oferă informațiile legate de profilul utilizatorului, permițând accesul și vizualizarea datelor personale.
- Serviciul de feedback se ocupă cu stocarea feedback-ului primit de la utilizatori, ajutând la îmbunătățirea continuă a aplicației.
- Serviciul de locații dermatologice folosește un serviciu extern care implementează API-ul Google Maps pentru a returna toate dermatologiile dintr-un anumit range din zona utilizatorului.
- Serviciul de programări se ocupă cu managementul crearea și managementul programărilor. Prin acest serviciu, utilizatorul își poate face programări în aplicație și primește notificări legate de acestea

- Serviciile de conversații și mesaje permit crearea de conversații și trimiterea de mesaje în cadrul acelor conversații. Printr-un serviciu extern care implementează API-ul OpenAI, utilizatorii primesc răspunsuri la mesajele trimise.
- Cel mai important serviciu este serviciul de predicții. Acesta se ocupă cu primirea imaginilor de la utilizator, stocarea lor în Azure Blob Storage și punerea în Azure Queue a ID-ului imaginii pentru a fi procesată de worker. Acest serviciu se ocupă și de actualizarea predicției cu rezultatul primit de la worker și de managementul predicțiilor, folosind diferite funcții pentru a returna către utilizator predicțiile dorite.

4.3 Structura componentei Azure

Componenta Azure a sistemului nostru este alcătuită din două părți principale: un container Blob Storage și o coadă Azure:

- **Container Blob Storage:** Acest container este utilizat pentru stocarea tuturor imaginilor încărcate de utilizatori. Imaginile sunt stocate în Blob Storage pentru a fi procesate și pentru a primi predicții. Acesta este singurul caz de utilizare pe care îl avem pentru Blob Storage, asigurând stocarea eficientă și accesibilă a imaginilor în cloud.
- **Coadă Azure:** Coadă Azure este folosită pentru a gestiona elementele încărcate de utilizatori. În coadă sunt adăugate obiecte care conțin mai multe informații, cum ar fi ID-ul utilizatorului, ID-ul imaginii, ID-ul predicției care trebuie actualizată și un token care trebuie să corespundă cu cel din baza de date pentru predicția respectivă, astfel încât rezultatul să fie luat în considerare. Această operațiune de adăugare este realizată de server, iar apoi worker-ul nostru preia itemele din coadă în ordine pentru a le procesa.

4.4 Structura aplicației mobile

Aplicația mobilă este scrisă în Flutter, având Dart ca limbaj de programare, și este alcătuită din mai multe componente și ecrane esențiale. La început, aplicația afișează un splash screen, urmat de pagini de logare și înregistrare. După autentificare, utilizatorul este redirecționat către pagina de home, care include un navbar poziționat jos, având un buton central pentru a încărca sau face poze și șase butoane de navigare. Fiecare dintre aceste componente oferă funcționalități pentru afișarea rezultatelor serviciilor serverului. Ecranele includ secțiuni pentru predicții, chat, programări, vizualizare profil, afișare locații

dermatologice, precum și widgeturi și componente vizuale pentru gestionarea resurselor (creare, actualizare, ștergere).

Aplicația conține și un modul de API, care, folosind Flutter/Dart și funcții asincrone, creează clase pentru fiecare tip de serviciu și standardizează rezultatele folosind modele obiect pentru a reține informațiile din payloadul răspuns de server. Aceste informații sunt afișate pe ecran prin intermediul ecranelor și componentelor vizuale create, folosind data providers pentru actualizarea în timp real a informațiilor de pe ecran.

De asemenea, aplicația include funcționalități de verificare a inputului pentru a asigura un sistem solid care informează utilizatorul dacă datele trimise către server sunt greșite. Există un sistem de avertizare a utilizatorului în cazul apariției unor erori prin intermediul unor snackbars widgets. Utilizatorii pot trimite feedback și evalua aplicația, permițând dezvoltatorilor să îmbunătățească continuu aplicația pe baza feedback-ului primit.

Un alt aspect important al aplicației este funcționalitatea de preprocesare a imaginilor înainte de a fi trimise către server. Imaginile sunt redimensionate pentru a avea dimensiunea de 600x450 pixeli, păstrând raportul de 4:3, și se aplică rotații dacă este necesar, pentru a se asigura că imaginea este în format landscape. Această preprocesare este esențială deoarece serverul procesează doar imagini în acest format.

Deoarece modul prin care se face procesarea imaginilor este unul asincron și pentru a nu face serverul să aștepte, după încărcarea cu succes a imaginii, serverul trimite ca răspuns o notificare pentru a informa aplicația mobilă că imaginea a fost încărcată cu succes. Ulterior, timp de 1 minut, aplicația va face un request la fiecare 10 secunde cu ID-ul primit al predicției pentru a verifica dacă imaginea a fost procesată. O altă variantă ar fi fost crearea de socket-uri, dar aceasta ar fi adus complicații suplimentare la design și la gestionarea logicii în aplicație, astfel că am optat pentru prima variantă.

4.5 Compoziția bazei de date

Avem o bază de date NoSQL în MongoDB (vezi Figura 44), folosind Mongoose pentru validarea datelor înainte de introducerea acestora în baza de date. Baza de date este stocată în Atlas, pe cloud, și avem acces la ea utilizând un simplu connection string. Baza de date este formată din 7 colecții, care vor fi reprezentate vizual în următoarea diagramă:

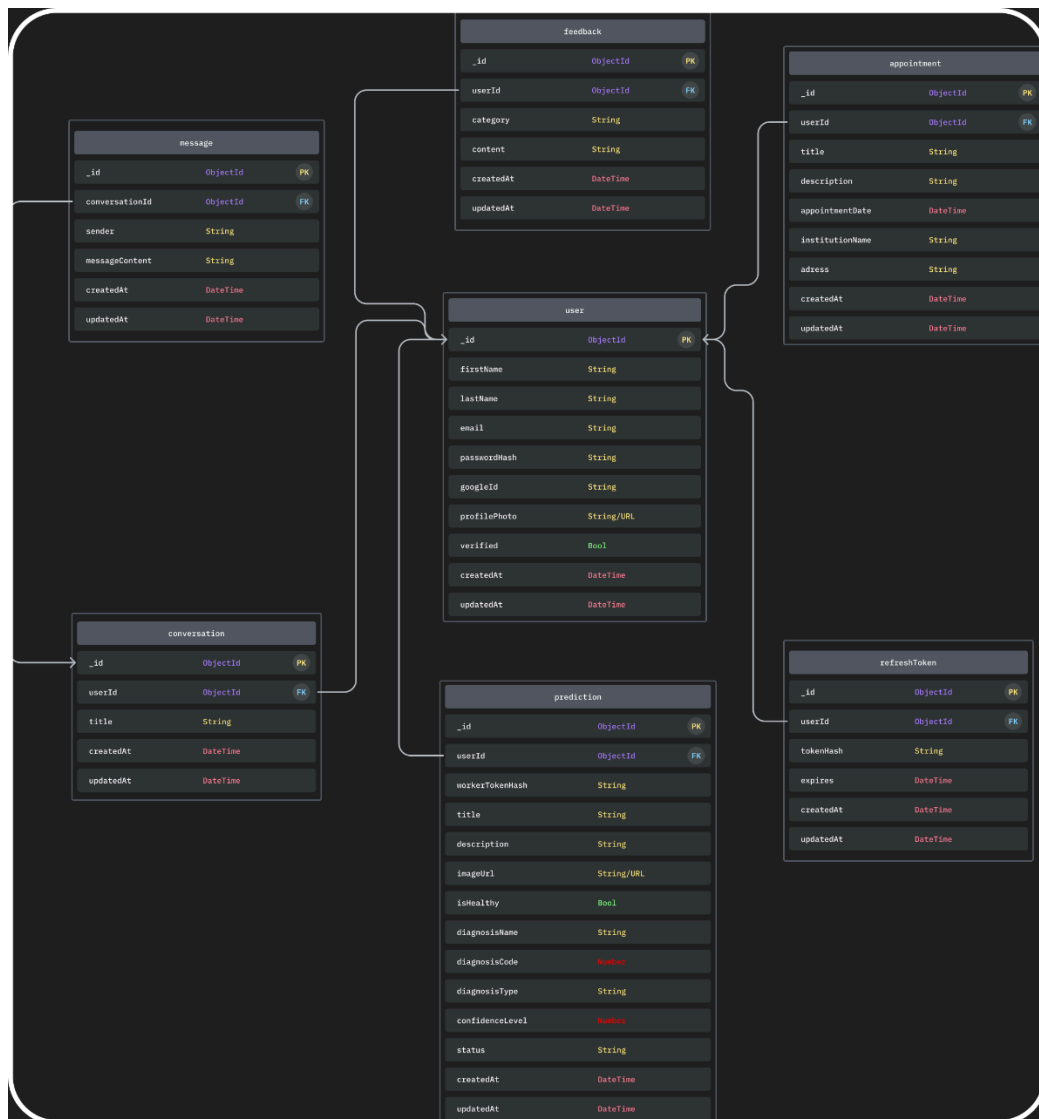


Figura 44. Schema bazei de date

4.6 Compoziția procesatorului de imagini

Procesatorul de imagini, denumit și worker, este scris în Python și utilizează biblioteca TensorFlow pentru încărcarea modelelor CNN și procesarea imaginilor. Worker-ul este format din două componente principale: un fișier start.py și un modul prediction_utils.

În fișierul start.py se încarcă toate modelele, configurațiile și obiectele necesare pentru a comunica cu coada și Blob-ul Azure. Aici se inițializează și procesele care vor prelucra imaginile în mod asincron.

Modulul prediction_utils conține mai multe clase esențiale. Există o clasă pentru trecerea unei imagini prin modelele multi-clasă și returnarea unui răspuns și o altă clasă pentru procesarea imaginilor prin modelul binar și returnarea unui răspuns. De asemenea, include o clasă pentru preprocesarea imaginilor înainte de a fi introduse în model, o clasă pentru conversia datelor numerice produse de model în date care pot fi parsate ca proprietăți

JSON și un fișier care conține valori constante reprezentând diferite afecțiuni, codurile lor și numărul de tipuri. În plus, există o clasă Worker care se ocupă de procesarea efectivă a imaginilor și trimiterea răspunsului înapoi către server.

Flow-ul worker-ului începe cu încărcarea modelelor CNN în memorie la începutul programului, urmată de crearea unui număr de X procese separate care vor procesa împreună itemele din coadă în mod asincron pentru a accelera procesul. Numărul de workere este specificat la început.

Din coada Azure se iau câte 5 iteme per request și sunt adăugate într-o coadă de multiprocessing, concepută pentru a lucra cu mai multe procese simultan fără probleme de concurență. Fiecare worker ia imagini din coada locală și le procesează trecând imaginea prin 4 modele în total. Mai întâi, folosind modelul binar, se verifică dacă imaginea este healthy sau not healthy. Dacă imaginea este healthy, procesul se oprește și se trimite răspunsul la server. Dacă imaginea nu este healthy, aceasta este trecută prin 3 modele multi-clasă. La final, se face un vot majoritar între cele 3 modele, iar acel rezultat este considerat final. Dacă modelele nu ajung la un consens, se presupune că imaginea nu a putut fi procesată sau nu s-a putut găsi un rezultat pentru ea.

Rezultatele obținute sunt trimise către server. Întregul proces este bine tratat, având blocuri try-catch în toate punctele sensibile ale programului pentru a preveni comportamente neașteptate sau trimiterea de date inconsistente.

4.7 Concluzie

Aplicația are o arhitectură ușor complexă, folosind o multitudine de tehnologii în spate, dar designul este unul profesional și separat în mai multe componente, fiecare cu rolul său specific. Am avut grijă ca toate componentele să trateze posibilele erori apărute pe parcursul rulării aplicației, asigurând astfel stabilitatea și fiabilitatea sistemului.

Capitolul 5: Utilizarea Aplicației

În acest capitol, va fi prezentată aplicația și modul de utilizare a acesteia din perspectiva unui utilizator obișnuit. Vom explora paginile aplicației, detaliind acțiunile care pot fi realizate pe fiecare pagină, pentru a oferi o înțelegere clară a funcționalităților disponibile. De asemenea, vor fi atașate screenshot-uri ilustrative care să ofere o imagine de ansamblu mai bună asupra interfeței și a modului de utilizare al aplicației.

5.1 Autentificarea în aplicație

La intrarea în aplicație, utilizatorul este întâmpinat de un splash screen, după care este redirecționat către pagina de login. Aici, dacă utilizatorul are un cont creat din aplicație, se poate autentifica folosind emailul și parola. De asemenea, dacă contul a fost creat folosind Google, există opțiunea de a se autentifica prin Google. Pagina de login include un checkbox care permite utilizatorului să rămână logat în aplicație. Există, de asemenea, o opțiune de a naviga către pagina de înregistrare, unde utilizatorul poate crea un cont fie completând toate câmpurile necesare din aplicație, fie prin autentificare cu Google. După finalizarea înregistrării, vom fi rugați să intrăm pe adresa de email pentru a ne verifica contul.

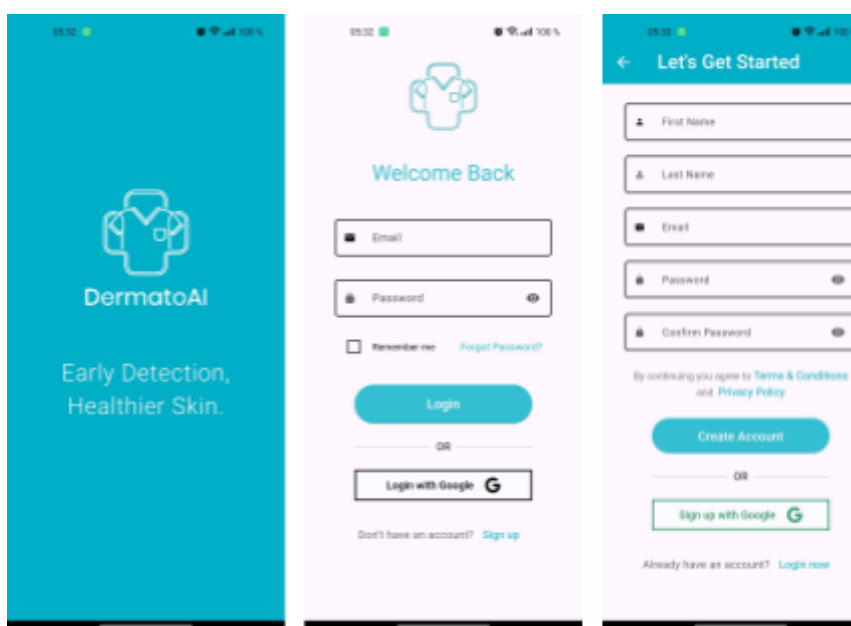


Figura 45. Ecranele de înregistrare, logare și ecranul splash inițial

5.2 Resetarea și recuperarea parolei

Recuperarea parolei se face pe pagina de login, accesând textul care întreabă dacă utilizatorul și-a uitat parola. Utilizatorul va fi redirecționat către o pagină separată unde va

trebui să introducă adresa de email. După ce emailul este introdus, un token va fi trimis pe această adresă. Utilizatorul va lua acel token și îl va introduce în aplicație împreună cu noua parolă și confirmarea noii parole.

Partea de resetare sau schimbare a parolei se realizează din pagina de profil din aplicație. Utilizatorul va trebui să introducă parola veche, parola nouă și o confirmare a noii parole. Aceasta asigură că doar utilizatorii autentificați și care își cunosc parola curentă pot schimba parola, oferind un nivel suplimentar de securitate.

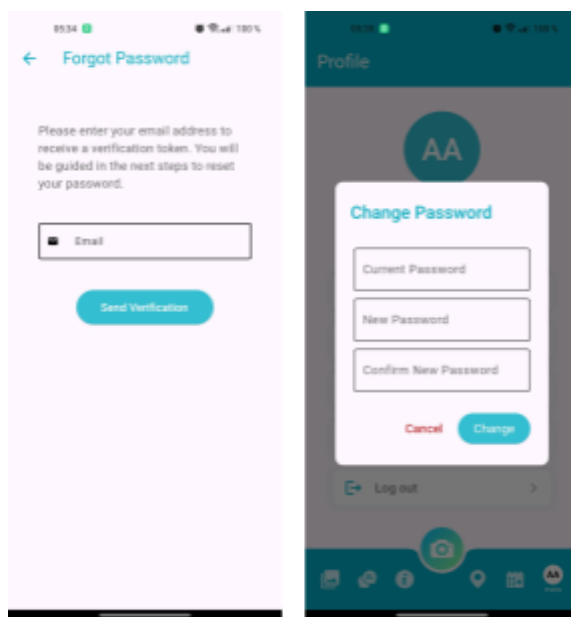


Figura 46. Ecranele pentru recuperarea parolei

5.3 Încărcarea de imagini și gestionarea predicțiilor

După ce utilizatorul se autentifică cu succes, acesta este redirecționat către pagina de home. Pe această pagină, există un floating button cu o iconiță de cameră, care permite utilizatorului să încarce imagini. Utilizatorul are două opțiuni: fie să încarce imagini din telefon, fie să facă o poză pe loc. După ce imaginea este încărcată, aceasta va fi afișată împreună cu celelalte imagini pe tabul de predictions/results.

În acest tab, statusul fiecărei predicții va fi afișat cu galben până când procesarea este completă. Utilizatorul poate face click pe fiecare predicție pentru a vedea mai multe detalii despre ea. Detaliile sunt prezentate într-un format frumos și intuitiv, oferind toate informațiile relevante despre acea predicție. De asemenea, utilizatorul are posibilitatea de a șterge sau de a face update la predicție (titlu și descriere) prin intermediul butoanelor intuitive disponibile pe această pagină. Aici mai există și un buton care să ne creeze automat o conversație cu chat-ul

pentru a putea cu ușurință să obținem informațiile necesare despre diagnostic în cel mai rapid mod.

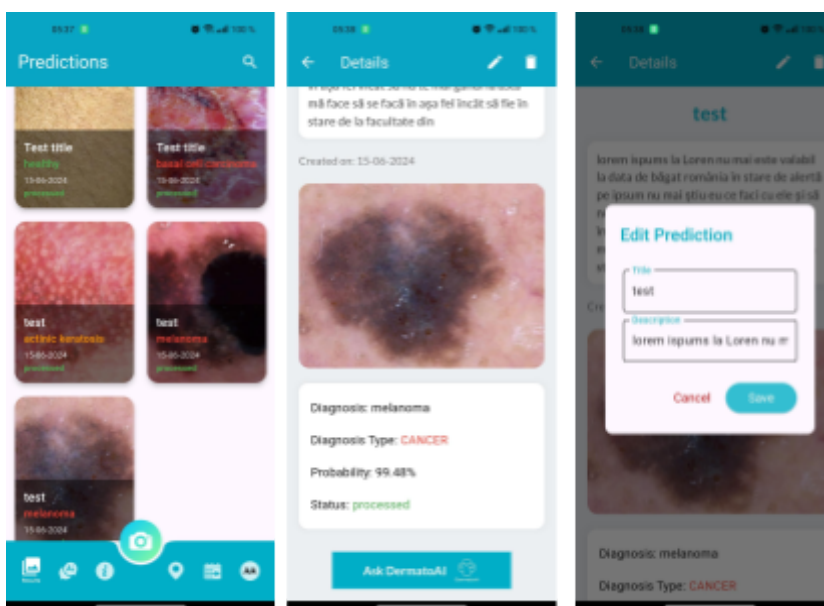


Figura 47. Ecranele pentru vizualizarea și managementul predicțiilor

5.4 Vizualizarea paginii de informații pentru afecțiuni

Pagina sau tabul de informații din aplicație este dedicată prezentării tuturor afecțiunilor pe care le poate diagnostica modelul nostru. Aici sunt afișate informații detaliate despre fiecare afecțiune, inclusiv ce este, cum apare, tratamente disponibile și gradul de gravitate. Aceasta este o pagină statică, având un scop pur informativ, pentru a oferi utilizatorilor cunoștințe suplimentare și pentru a-i ajuta să înțeleagă mai bine diagnosticul primit.

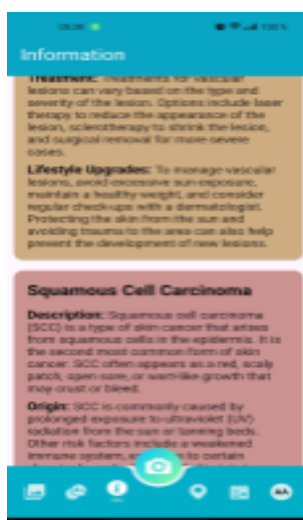


Figura 48. Ecranul pentru vizualizarea informațiilor despre afecțiuni

5.5 Utilizarea chat-ului de inteligență artificială

Pe pagina de home, există un tab dedicat conversației cu chat-ul nostru de inteligență artificială. După ce utilizatorul face click pe acest tab, poate crea o conversație nouă sau poate intra într-o conversație deja creată anterior din lista afișată. Într-o pagină nouă, utilizatorul va vedea toate mesajele vechi din conversație și va putea adăuga altele folosind input bar-ul intuitiv din partea de jos a ecranului.

De asemenea, există opțiuni pentru a șterge conversația și pentru a actualiza titlul conversației (update), oferind utilizatorilor un control complet asupra interacțiunilor lor cu chat-ul de inteligență artificială. Această funcționalitate permite utilizatorilor să primească răspunsuri și asistență în timp real pentru diverse întrebări și preocupări legate de afecțiunile pielii.

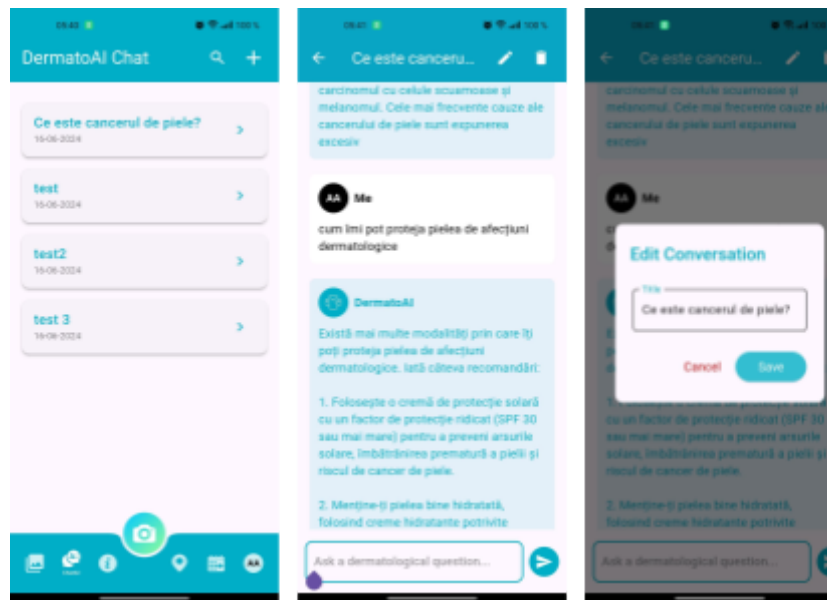


Figura 49. Ecranele de vizualizare a conversațiilor și trimitere de mesaje

5.6 Vizualizarea locațiilor dermatologice din zonă

Pe tabul de locations/clinics din bara de navigare de pe pagina de home, utilizatorii pot accesa pagina de locații. Aici, se va afișa un slide bar de la 1 până la 50, reprezentând numărul de kilometri ca rază pentru căutarea dermatologilor. Utilizatorul poate ajusta slide bar-ul pentru a selecta un range specific, iar apoi, în partea de jos a paginii, se va afișa o listă cu toate locațiile găsite.

Fiecare locație este prezentată cu imagini și informații relevante, iar utilizatorul poate face click pe oricare dintre ele pentru a deschide aplicația Google Maps de pe telefon. Aceasta

va permite utilizatorului să vadă exact unde se află locația și să obțină mai multe detalii, facilitând astfel găsirea și accesarea serviciilor dermatologice din apropiere.

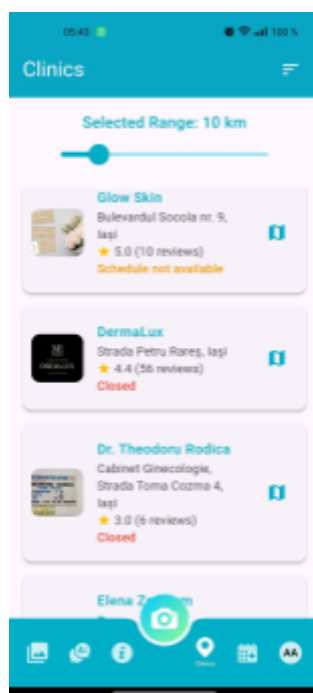


Figura 50. Ecranul de vizualizare a clinicilor dermatologice

5.7 Crearea de programări și primirea notificărilor

În aplicație, utilizatorii pot accesa tabul de appointments/meets de pe pagina de home, care îi va duce la o pagină nouă unde va fi afișată o listă cu toate programările făcute. Pe această pagină există și un buton care permite crearea unei noi programări. Utilizatorii trebuie să introducă titlul și data pentru programare, utilizând un date picker pentru a selecta data corespunzătoare. Acest lucru asigură că aplicația va notifica utilizatorul în background cu privire la programarea respectivă.

Când utilizatorul face click pe o programare din listă, este redirecționat către o pagină unde poate vedea detaliile programării, o poate șterge sau îi poate da update cu noi informații. Aceasta funcționalitate permite utilizatorilor să-și gestioneze eficient programările și să primească notificări pentru a nu uita de ele.

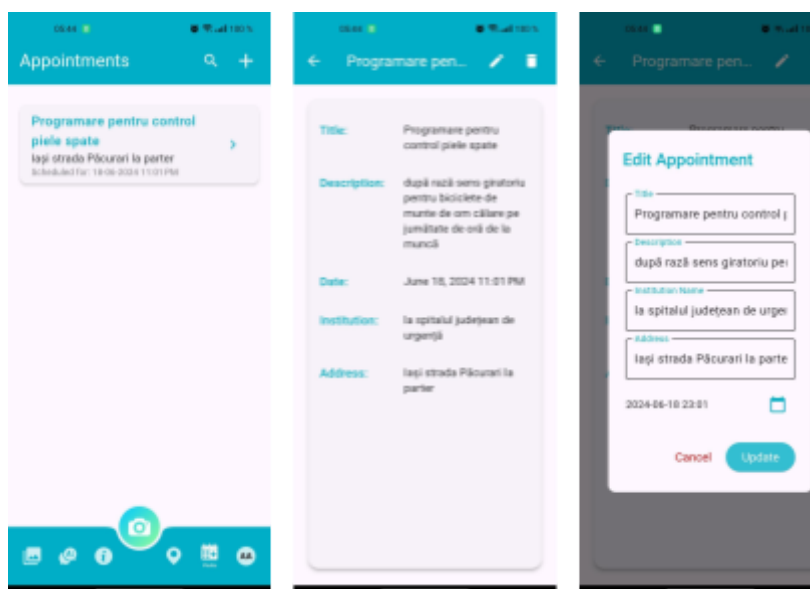


Figura 51. Ecranele de vizualizare și management a programărilor

5.8 Vizualizarea profilului și trimiterea de feedback

Pe pagina de profil, accesată din tabul de pe bara de navigare din home, utilizatorii vor avea afișate informațiile contului lor, cum ar fi emailul, numele, prenumele și dacă adresa de email este verificată. De asemenea, pe această pagină există un buton pentru feedback.

Când utilizatorul apasă pe butonul de feedback, se va deschide un popup în care poate selecta categoria feedback-ului și introduce feedback-ul într-un input dedicat. După trimiterea feedback-ului, utilizatorului i se va afișa un mesaj de mulțumire pentru feedback-ul oferit.

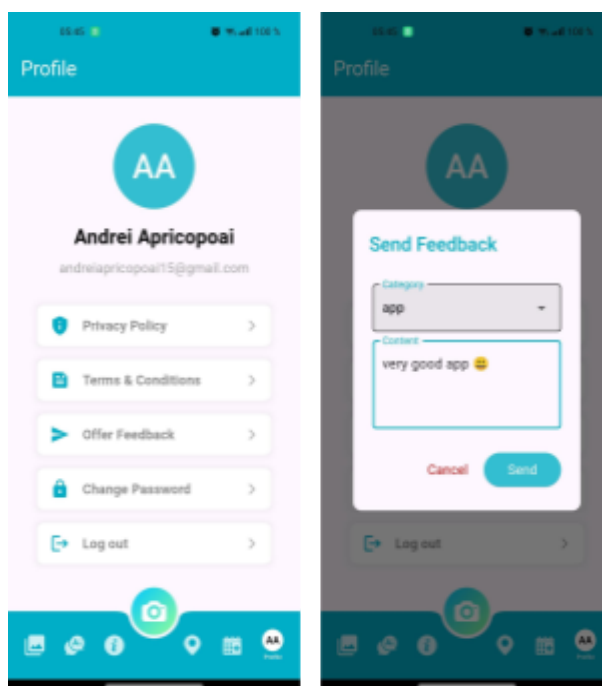


Figura 52. Ecranele de vizualizare a profilului și trimiterea de feedback

5.9 Concluzie

Aplicația este intuitivă și ușor de utilizat, având un design profesional și bine structurat. Fiecare funcționalitate, de la autentificare până la încărcarea imaginilor și gestionarea predicțiilor, este accesibilă și ușor de navigat. Utilizatorii beneficiază de un control complet asupra interacțiunilor lor cu aplicația, asigurându-se astfel o experiență plăcută.

Concluzii finale și posibile îmbunătățiri

DermatoAI este o aplicație care are la bază patru modele de rețele neuronale convoluționale (CNN) și vine ca soluție pentru diagnosticarea afecțiunilor pielii prin intermediul imaginilor încărcate de pe telefon. Această aplicație este destinată zonelor în care nu există doctori dermatologi sau pentru oamenii care nu își permit să meargă la un dermatolog. DermatoAI funcționează ca un preexaminator și nu ar trebui să înlocuiască vizita la dermatolog, deoarece aceasta rămâne cea mai relevantă și sigură opțiune din orice punct de vedere. Aplicația oferă diverse funcționalități, de la încărcarea de imagini pentru a obține predicții, până la utilizarea chatului AI, crearea de programări în aplicație și căutarea dermatologilor din zonă.

Cu siguranță, DermatoAI nu este o aplicație perfectă și poate beneficia de anumite îmbunătățiri. Printre acestea se numără crearea de noi modele, mai robuste și mai solide, care să funcționeze pe o multitudine de afecțiuni dermatologice, nu doar pe cele opt afecțiuni. De asemenea, ar fi utilă integrarea funcționalității de programare directă la clinici din aplicație, folosind diverse API-uri sau colaborări cu clinicile dermatologice. În prezent, programările se pot face doar local în aplicație și nu sunt trimise către clinici. Astfel, dezvoltarea unor funcționalități care să permită trimiterea programărilor către clinici ar aduce un beneficiu semnificativ utilizatorilor.

Bibliografie

- [1] “SkinVision - Find Skin Cancer – Apps on Google Play.” Accessed: Jun. 09, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.rubytribe.skinvision.ac&hl=en_AU
- [2] “Medic Scanner - skin analyze – Apps on Google Play.” Accessed: Jun. 09, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=health.medicalscanner.app&hl=en_AU
- [3] “AI Dermatologist: Skin Scanner – Apps on Google Play.” Accessed: Jun. 09, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.aidermatologist&hl=en_AU
- [4] “Docs | Flutter.” Accessed: Jun. 11, 2024. [Online]. Available: <https://docs.flutter.dev/>
- [5] “Index | Node.js v22.3.0 Documentation.” Accessed: Jun. 17, 2024. [Online]. Available: <https://nodejs.org/docs/latest/api/>
- [6] “Installing Express.” Accessed: Jun. 17, 2024. [Online]. Available: <https://expressjs.com/en/starter/installing.html>
- [7] “MongoDB Documentation.” Accessed: Jun. 17, 2024. [Online]. Available: <https://www.mongodb.com/docs/>
- [8] “Mongoose v8.4.1: Getting Started.” Accessed: Jun. 17, 2024. [Online]. Available: <https://mongoosejs.com/docs/>
- [9] “Welcome to Python.org,” Python.org. Accessed: Jun. 17, 2024. [Online]. Available: <https://www.python.org/doc/>
- [10] normesta, “Introduction to Azure Queue Storage - Azure Storage.” Accessed: Jun. 17, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/storage/queues/storage-queues-introduction>
- [11] “Azure Blob Storage | Microsoft Azure.” Accessed: Jun. 17, 2024. [Online]. Available: <https://azure.microsoft.com/en-us/products/storage/blobs>
- [12] “Google Maps Platform Documentation,” Google for Developers. Accessed: Jun. 17, 2024. [Online]. Available: <https://developers.google.com/maps/documentation>
- [13] “OpenAI Platform.” Accessed: Jun. 17, 2024. [Online]. Available: <https://platform.openai.com>
- [14] “Email API - Start for Free,” SendGrid. Accessed: Jun. 17, 2024. [Online]. Available: <https://sendgrid.com/en-us/solutions/email-api>
- [15] “Tutorials | TensorFlow Core.” Accessed: Jun. 11, 2024. [Online]. Available: <https://www.tensorflow.org/tutorials>
- [16] “Git - Documentation.” Accessed: Jun. 17, 2024. [Online]. Available: <https://www.git-scm.com/doc>
- [17] “GitHub.com Help Documentation,” GitHub Docs. Accessed: Jun. 17, 2024. [Online]. Available: <https://docs.github.com/en>
- [18] “ISIC | International Skin Imaging Collaboration,” ISIC. Accessed: Jun. 10, 2024. [Online]. Available: <https://www.isic-archive.com>
- [19] P. Tschandl, C. Rosendahl, and H. Kittler, “The HAM10000 dataset, a large collection of multi-source dermoscopic images of common pigmented skin lesions,” *Sci. Data*, vol. 5, no. 1, p. 180161, Aug. 2018, doi: 10.1038/sdata.2018.161.

- [20] "ISIC Archive." Accessed: Jun. 11, 2024. [Online]. Available: <https://api.isic-archive.com/collections/249/>
- [21] "ISIC Archive." Accessed: Jun. 11, 2024. [Online]. Available: <https://api.isic-archive.com/collections/70/>
- [22] "ISIC Archive." Accessed: Jun. 11, 2024. [Online]. Available: <https://api.isic-archive.com/collections/65/>
- [23] I. A. Emu *et al.*, "ArsenicSkinImageBD: A comprehensive image dataset to classify affected and healthy skin of arsenic-affected people," *Data Brief*, vol. 52, p. 110016, Dec. 2023, doi: 10.1016/j.dib.2023.110016.
- [24] N. Aslam, "Training a CNN from Scratch using Data Augmentation," Analytics Vidhya. Accessed: Jun. 11, 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2022/09/training-a-cnn-from-scratch-using-data-augmentation/>
- [25] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.
- [26] "What are Convolutional Neural Networks? | IBM." Accessed: Jun. 11, 2024. [Online]. Available: <https://www.ibm.com/topics/convolutional-neural-networks>
- [27] S. SHARMA, "Activation Functions in Neural Networks," Medium. Accessed: Jun. 11, 2024. [Online]. Available: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>
- [28] "Papers with Code - MobileNetV3 Explained." Accessed: Jun. 12, 2024. [Online]. Available: <https://paperswithcode.com/method/mobilenetv3>
- [29] "Papers with Code - Binary Input Layer: Training of CNN models with binary input data." Accessed: Jun. 11, 2024. [Online]. Available: <https://paperswithcode.com/paper/binary-input-layer-training-of-cnn-models>
- [30] W. Ezat, M. Dessouky, and N. Ismail, "Multi-class Image Classification Using Deep Learning Algorithm," *J. Phys. Conf. Ser.*, vol. 1447, p. 012021, Jan. 2020, doi: 10.1088/1742-6596/1447/1/012021.
- [31] "The C4 model for visualising software architecture." Accessed: Jun. 11, 2024. [Online]. Available: <https://c4model.com/>