UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Clasificarea si detecția bolii Alzheimer

propusă de

Andrei-Alexandru Baractaru

Sesiunea: *Februarie, 2025*

Coordonator științific

Conf. Dr. Anca Ignat

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Clasificarea si detecția bolii Alzheimer

Andrei-Alexandru Baractaru

Sesiunea: *Februarie, 2025*

Coordonator științific

Conf. Dr. Anca Ignat

Cuprins

Introducere

Creierul este unul dintre cele mai complexe și esențiale organe ale corpului uman, permițând toate funcțiile cognitive, emoționale și motorii. În contextul îmbătrânirii progresive a populației, bolile neurodegenerative, în special boala Alzheimer, au devenit o povară grea pentru sistemele de sănătate din întreaga lume. Boala Alzheimer este cel mai răspândit tip de demență, cu aproximativ 50 de milioane de persoane afectate la nivel mondial, și se estimează că acest număr se va tripla până în 2050. Natura sa neurodegenerativă afectează aproximativ 60-70% din toate cazurile de demență și este una dintre cauzele majore de handicap în rândul persoanelor în vârstă.

Motivul alegerii acestui subiect este că există o nevoie urgentă de a îmbunătăți detectarea precoce a bolii Alzheimer. Din punct de vedere microscopic, boala este marcată patologic de depunerea proteinei beta-amiloide în plăci și a proteinei Tau în încurcături neurofibrilare, ceea ce duce la moartea neuronală și atrofia cerebrală care se agravează progresiv. Aceste modificări ale structurii pot fi vizualizate cu ajutorul imagisticii prin rezonanță magnetică (RMN), însă evaluarea manuală a acestor imagini este o sarcină anevoioasă, consumatoare de timp și potențial subiectivă. În plus, accesul la experții neurologici este de obicei restricționat de perioadele lungi de așteptare pentru consultare, costurile exorbitante ale investigațiilor și distribuția inegală a experților.

Originalitatea subiectului este susținută de faptul că aceasta este o încercare nouă de a integra mai multe structuri de rețele neuronale profunde în analiza scanărilor cerebrale RMN. Deși există o serie de studii existente care utilizează metode de învățare automata(machine learning) pentru detectarea bolii Alzheimer, noutatea acestei lucrări este că aplicăm și comparăm simultan patru modele diferite de învățare profundă. Pe lângă adaptarea unor arhitecturi cunoscute precum VGG16, InceptionV3 și ResNet50, am propus, de asemenea, o arhitectură CNN personalizată(Custom CNN), special adaptată pentru analiza imaginilor cerebrale RMN. Implementarea lor în cadrul unei interfețe grafice ușor de utilizat și evaluarea lor comparativă detaliată oferă o perspectivă nouă asupra eficacității diferitelor metode în contextul specific al diagnosticării bolii Alzheimer.

Obiectivele generale ale acestei lucrări sunt îndreptate spre realizarea unui sistem inteligent de clasificare a stadiului bolii Alzheimer prin analiza imaginilor RMN cerebrale. Acest sistem este conceput pentru a oferi o evaluare automată și obiectivă a stadiului bolii prin clasificarea imaginilor în patru clase diferite: Non-Dement(Non-Demented), Foarte Ușor Dement(Very-Mild Demented), Ușor Dement(Mild-Demented) și Moderat Dement(Moderate-Demented). Prin implementarea și

evaluarea diferitelor arhitecturi de rețele neuronale, scopul acestei lucrări este de a determina cea mai adecvată metodă pentru problema de clasificare. Un alt obiectiv important este acela de a dezvolta o interfață grafică intuitivă într-un mod care să optimizeze utilizabilitatea sistemului într-un mediu clinic. Abordarea utilizată în dezvoltarea sistemului se bazează pe cele mai moderne metodologii de deep learning(învățare profundă) și de procesare a imaginilor. Fluxul de lucru începe cu preprocesarea imaginilor RMN, inclusiv redimensionarea la dimensiuni standard și normalizarea intensității pixelilor, împreună cu strategii de augmentare a datelor pentru îmbunătățirea antrenării. Implementarea și antrenarea modelelor de deep learning cuprinde configurarea atentă a arhitecturii rețelei neuronale, optimizarea hiperparametrilor și aplicarea strategiilor de învățare prin transfer(transfer learning) pentru modelele pre-antrenate. Performanța este evaluată pe baza metricilor standard, cum ar fi acuratețea, precizia și recall-ul.

Soluția este dezvoltată ca o aplicație desktop în Python, cu o interfață ușor de utilizat pentru încărcarea și analizarea imaginilor RMN ale creierului. Sistemul încorporează modele de deep learning în cadrul unei interfețe grafice, care permite utilizatorului să vizualizeze și să compare predicțiile diferitelor modele pe aceeași imagine. Acest lucru îmbunătățește măsurarea performanței comparative a diferitelor arhitecturi și oferă experților un instrument util pentru a ajuta procesul de diagnosticare.

Lucrarea este organizată în trei capitole principale, urmând un flux logic în prezentarea cercetării. Primul capitol, „Descrierea problemei”, este o examinare atentă a bolii Alzheimer, discutând efectul acesteia la nivel mondial și importanța diagnosticării timpurii. Acesta prezintă contextul social și medical al bolii, provocările actuale de diagnosticare și rolul important al imagisticii prin RMN în evaluarea bolii. Al doilea capitol, „Abordări anterioare”, trece în revistă tehnicile și metodele actuale în domeniul analizei imaginilor medicale, acoperind atât metodele clasice de procesare a imaginilor medicale, cât și aplicațiile recente ale învățării profunde în acest domeniu și subliniind limitările abordărilor actuale. Ultimul capitol, „Descrierea soluției”, prezintă în detaliu implementarea sistemului propus, începând cu aspectele teoretice și tehnologiile utilizate, urmată de descrierea setului de date și a etapei de preprocesare și încheind cu prezentarea rezultatelor experimentale și analiza comparativă a performanțelor diferitelor modele.

Contribuții

Pentru a realiza această lucrare de licență, a fost necesar, în primul rând, o documentare intensă în domeniul procesării imaginilor medicale și al algoritmilor de învățare profundă utilizați în diagnosticarea bolii Alzheimer. În timp ce principiile procesării imaginilor și rețelelor neuronale au fost învățate în facultate, implementarea lor în cazul particular al imagisticii medicale a necesitat o cercetare suplimentară.

Unul dintre elementele-cheie ale contribuției personale a fost crearea unei arhitecturi CNN personalizate, adaptată pentru clasificarea imaginilor RMN ale creierului. Acest lucru a inclus multe iterații și experimente pentru a găsi cea mai performantă structură de rețea. Concomitent, am implementat și modificat trei arhitecturi bine cunoscute (VGG16, InceptionV3 și ResNet50) pentru această problemă specială de clasificare, folosind strategii de învățare prin transfer pentru a valorifica cunoștințele anterioare ale modelelor pre-antrenate.

O contribuție semnificativă a fost crearea unei interfețe grafice ușor de utilizat în Python cu ajutorul bibliotecii Tkinter. Aceasta permite încărcarea și analiza în paralel a imaginilor RMN cu toate modelele implementate, permițând compararea directă a predicțiilor și o evaluare ușoară a performanței comparative a diferitelor arhitecturi. Interfața a fost dezvoltată cu accent pe ușurința în utilizare și pe claritatea prezentării rezultatelor, ceea ce face ca sistemul să fie adecvat pentru implementarea într-un cadru clinic.

În cele din urmă, am efectuat o analiză comparativă detaliată a performanței tuturor modelelor implementate cu ajutorul metricilor standard, de învățare automată. Analiza a oferit informații utile cu privire la cât de bine au funcționat diferitele abordări în cazul particular al clasificării stadiului bolii Alzheimer, contribuind astfel la înțelegerea mai bună a aplicabilității diferitelor arhitecturi de deep learning în analiza imagistică medicală.

Capitolul I - Descrierea problemei

Boala Alzheimer reprezintă o provocare semnificativă pentru sistemele de sănătate, nu numai din cauza volumului mare de pacienți afectați, ci și din cauza complexității procesului de diagnosticare. Spre deosebire de alte boli, diagnosticul bolii Alzheimer trebuie să fie pus printr-o sinteză de evaluare clinică, teste cognitive și imagistică, fiecare dintre acestea necesitând timp și experiență în diagnosticarea acestei afecțiuni. În special, analiza scanărilor RMN ale creierului este un proces elaborat care necesită ani de experiență, alături de pregătire specializată.

Un aspect specific care m-a determinat să aleg acest subiect este posibilitatea tehnologiei de a democratiza accesul la cunoștințele medicale. Accesul la specialiștii neurologici este restricționat în majoritatea locațiilor geografice, iar perioadele de așteptare pentru o consultație pot fi semnificative. Un sistem automat de analiză a imaginilor RMN ar putea oferi un prim nivel de screening, semnalând cazurile care necesită o atenție urgentă și ușurând astfel procesul de triaj.

Dezvoltarea acestui tip de sistem nu este doar o sarcină tehnică, ci și o posibilitate de a contribui la îmbunătățirea procesului de diagnosticare medicală. Integrând metodele contemporane de deep learning cu cunoștințele medicale convenționale, putem construi instrumente care să ajute experții în luarea deciziilor, făcând procesul mai eficient și mai accesibil. Această strategie nu este menită să înlocuiască examinarea clinică a medicului, ci mai degrabă să o îmbunătățească cu date obiective și măsurabile obținute din prelucrarea automată a imaginilor medicale.

Capitolul II – Abordari anterioare

Abordări teoretice

În ultimul deceniu, cercetarea în domeniul detectării automatizate a bolii Alzheimer utilizând imagini RMN a cunoscut o evoluție semnificativă. Un punct de referință important a fost stabilit de Suk și colaboratorii [1] în 2017, care au propus o metodă de învățare profundă multi-modală pentru diagnosticul bolii Alzheimer. Abordarea lor combină caracteristici extrase din multiple modalități de imagistică (RMN și PET), demonstrând o îmbunătățire semnificativă în acuratețea diagnosticului față de metodele anterioare.

Islam și Khan [2] au publicat în 2019 o cercetare extinsă care utilizează o arhitectură CNN adaptată pentru clasificarea imaginilor RMN ale creierului. Autorii și-au antrenat modelul pe setul de date OASIS și au înregistrat o acuratețe de 93,18% pentru a distinge între patru stadii de boală (non-dementat, foarte ușor dement, ușor dement și moderat dement).

O contribuție semnificativă a fost adusă de Valliani și Soni [3] care au utilizat tehnici de transfer learning cu arhitectura ResNet-50 pentru clasificarea stadiilor bolii Alzheimer. Cercetarea lor a demonstrat că utilizarea modelelor pre-antrenate poate reduce semnificativ timpul de antrenare menținând o acuratețe ridicată de 95.7% în clasificarea binară (Alzheimer vs. control).

Abordări practice

Figura 1: BrainScan [4]

BrainScan AI, dezvoltat de BrainCo Technologies, este unul dintre cele mai utilizate instrumente comerciale în analiza imagistică neurologică. Dezvoltată de o echipă de cercetători și clinicieni, aplicația utilizează algoritmi de învățare profundă pentru a scana imaginile cerebrale RMN și este capabilă să identifice mai multe tulburări neurologice, inclusiv boala Alzheimer. Sistemul oferă o interfață ușor de utilizat pentru încărcarea și analiza imaginilor și creează rapoarte complete privind modificările structurale identificate.

Figura 2: NeuroQuant [7]

NeuroQuant [5,6] de la CorTechs Labs este un alt exemplu de sistem automat de analiză a imaginilor RMN ale creierului. Software-ul este aprobat de FDA și CE și este instalat în numeroase clinici și spitale. Sistemul are capacitatea de a măsura volumele diferitelor structuri cerebrale și de a le compara cu bazele de date normative, ceea ce ajută la detectarea atrofiei cerebrale tipice pentru boala Alzheimer.

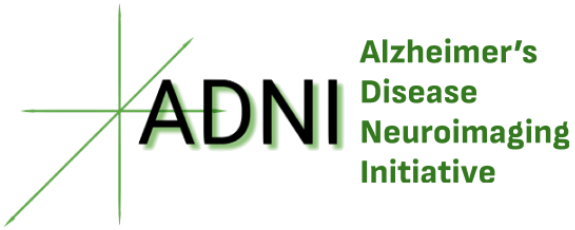

Figura 3: ADNI [8]

ADNI (Alzheimer's Disease Neuroimaging Initiative) a dezvoltat, de asemenea, propriile sale instrumente software pentru analiza imaginilor RMN, care sunt puse gratuit la dispoziția comunității de cercetători. Acestea includ algoritmi automatizați pentru segmentarea structurii creierului și sisteme de clasificare bazate pe învățarea automată.

Aceste metode anterioare au contribuit la avansarea cu pași mari a domeniului și au deschis calea pentru sistemele actuale de asistență pentru diagnosticarea bolii Alzheimer. Cu toate acestea, majoritatea acestor sisteme rămân limitate în ceea ce privește accesibilitatea și ușurința de utilizare, ceea ce se încearcă a fi îmbunătățit prin lucrarea prezentată aici.

Capitolul III - Descrierea soluției

3.1 Aspecte teoretice

3.1.1 Computer Vision

Computer Vision este o ramură a inteligenței artificiale care permite calculatoarelor să extragă și să interpreteze informații din imagini digitale și video, cu scopul final de a egala și de a depăși capacitățile vizuale umane. Datorită dezvoltării învățării profunde și a rețelelor neuronale, domeniul a crescut cu pași mari, în unele aplicații de detectare și etichetare depășind perfomanțele oamenilor.

Activitățile principale ale Computer Vision sunt clasificarea imaginilor (identificarea tipului de obiect), detectarea obiectelor (identificarea și localizarea obiectelor), segmentarea imaginilor (împărțirea în regiuni semantice), verificarea obiectelor (confirmarea prezenței) și detectarea reperelor (detectarea punctelor-cheie). Aceste funcționalități sunt confruntate cu provocări precum variabilitatea vizuală, condițiile de iluminare și ocluziunea obiectelor.

Pentru a depăși aceste dificultăți, sistemele recente utilizează metode sofisticate de învățare automată, în special rețele neuronale convoluționale, cu aplicații în domenii care variază de la diagnosticul medical la mașinile autonome(care se conduc singure) și realitatea augmentată.

3.1.2 Rețele neuronale convoluționale

Rețelele neuronale convoluționale sunt o clasă specializată de modele neuronale adaptate în mod specific procesării imaginilor. Acestea au capacitatea de a condensa imaginile într-o formă mai ușor de procesat, păstrând în același timp caracteristicile lor esențiale. Un avantaj cheie al acestor rețele este nivelul scăzut de complexitate în antrenare, cu mai puțini parametri în comparație cu rețelele convenționale complet conectate.

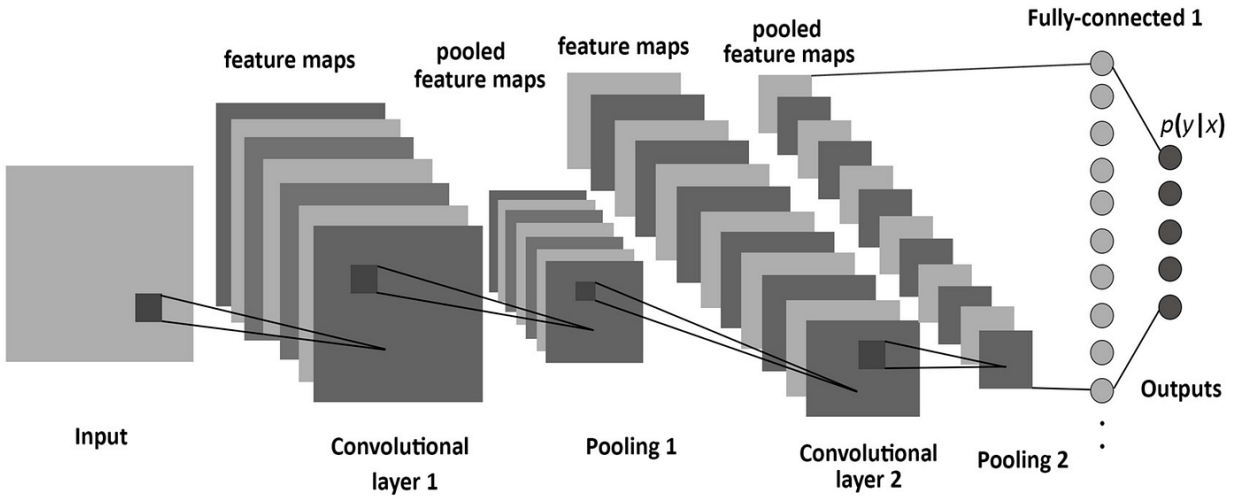

Figura 4: Rețea neuronală convoluțională [9]

Arhitectura unei rețele neuronale convoluționale include un număr de straturi specializate de mai multe tipuri, fiecare îndeplinind o anumită funcție în prelucrarea și analizarea imaginilor:

Stratul convoluțional, este nucleul acestor rețele, care funcționează prin utilizarea de filtre care învață să identifice diferite lucruri despre imaginea de intrare. Convoluția se realizează prin suprapunerea acestor filtre peste imagine, efectuând operații matematice care permit extragerea caracteristicilor. Straturile convoluționale inițiale au sarcina de a recunoaște caracteristici simple, de exemplu, margini, culori sau orientări, iar straturile ulterioare dezvoltă capacitatea de a identifica caracteristici mai complexe și abstracte.

Funcțiile de activare sunt importante pentru învățarea rețelei. Ele adaugă neliniaritate modelului, ceea ce permite rețelei să învețe tipare complexe. ReLU (Rectified Linear Unit) este o funcție de activare foarte comună, datorită eficienței sale de calcul și a eficacității în prevenirea problemei gradientului care dispare. Aceasta mapează valorile negative la zero și lasă valorile pozitive așa cum sunt, ceea ce permite rețelei să învețe mai eficient.

Stratul de pooling este conceput pentru a reduce dimensionalitatea reprezentărilor, condensând informațiile spațiale, dar păstrând caracteristicile cheie. Max Pooling, care reține valoarea maximă dintr-o regiune, și Average Pooling, care calculează media valorilor, sunt cele mai utilizate tipuri. Reducerea dimensionalității ajută la controlul timpului de antrenare și la prevenirea supraantrenării (overfitting).

Dropout este o metodă de regularizare care evită supraantrenarea prin dezactivarea aleatorie a unei proporții de neuroni în timpul antrenamentului. Aceasta obligă rețeaua să învețe caracteristici mai puternice și mai generalizate, deoarece rețeaua nu poate presupune că toți neuronii vor fi întotdeauna prezenți.

Straturile dense (Fully Connected) sunt poziționate aproape de sfârșitul rețelei și trebuie să integreze caracteristicile identificate de straturile anterioare pentru a finaliza clasificarea. Acestea convertesc reprezentările spațiale într-un vector unidimensional, astfel încât rețeaua să poată învăța relații complexe între caracteristicile extrase și clasele țintă.

În aplicația de detectare a bolii Alzheimer, această arhitectură elaborată permite sistemului să învețe caracteristicile specifice ale diferitelor stadii ale bolii într-un mod progresiv, de la detalii structurale de nivel scăzut la tipare complexe și unice pentru fiecare stadiu de evoluție a bolii.

3.1.3 Transfer Learning

Transferul de învățare (Transfer Learning) reprezintă o tehnică fundamentală în domeniul învățării automate, inspirată din capacitatea umană de a transfera cunoștințe între sarcini similare. Această abordare permite valorificarea cunoștințelor dobândite de un model în rezolvarea unei probleme pentru a facilita învățarea unor sarcini noi, dar înrudite.

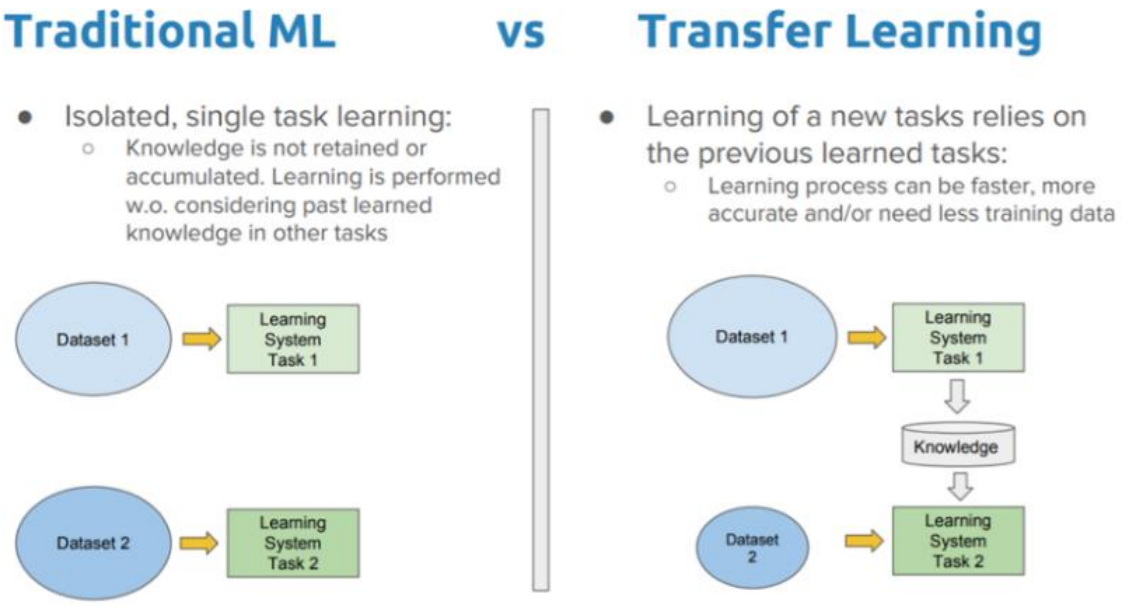

Figura 5: Transferul de învățare [10]

Această tehnică prezintă mai multe avantaje semnificative. În primul rând, reduce dramatic timpul și resursele computaționale necesare antrenării, întrucât majoritatea straturilor complexe ale rețelei sunt deja antrenate. În al doilea rând, permite obținerea unor rezultate bune chiar și cu seturi de date relativ mici, deoarece modelul pornește de la o înțelegere solidă a caracteristicilor vizuale generale. În plus, transferul de învățare ajută la îmbunătățirea generalizării modelului, întrucât acesta beneficiază de cunoștințele generale despre procesarea imaginilor acumulate în timpul antrenamentului inițial.

În cazul specific al lucrării prezentate, transferul de învățare ne permite să beneficiem de puterea unor arhitecturi complexe și bine optimizate, adaptându-le pentru sarcina specifică de analiză a imaginilor medicale. Această abordare este deosebit de valoroasă în contextul medical, unde adesea seturile de date sunt limitate ca dimensiune, dar necesită o precizie ridicată în clasificare.

3.1.4 Early Stopping și optimizarea modelelor

Early Stopping este o tehnică de optimizare esențială utilizată în formarea rețelelor neuronale pentru a răspunde provocării fundamentale de a determina momentul optim pentru oprirea antrenamentului. Această tehnică previne supraantrenarea (overfitting) prin monitorizarea continuă a performanței modelului pe setul de validare, oprind antrenamentul atunci când nu se realizează îmbunătățiri semnificative.

În acest caz, Early Stopping este aplicat utilizând API-ul Keras cu parametri specifici pentru a urmări precizia pe setul de validare, precum și o perioadă de așteptare specificată (patience) înainte de oprirea antrenamentului. ModelCheckpoint a fost, de asemenea, utilizat pentru a salva cele mai bune ponderi învățate în timpul antrenamentului, iar optimizatorul Adam a fost utilizat pentru a permite reglarea eficientă a ponderilor rețelei.

Utilizarea acestor tehnici de optimizare asigură faptul că modelele evită supraantrenarea, mențin o capacitate puternică de generalizare și maximizează utilizarea resurselor computaționale disponibile, toate acestea fiind fundamentale pentru obținerea unor rezultate fiabile în clasificarea imaginilor RMN.

3.2 Tehnologii utilizate

3.2.1 TensorFlow și Keras

TensorFlow, dezvoltat de Google, reprezintă un framework de bază pentru învățare automată și deep learning, oferind instrumentele necesare pentru proiectarea, construirea și antrenarea modelelor de învățare profundă. Framework-ul excelează în operațiile cu tensori și calcul distribuit, fiind optimizat pentru performanță și scalabilitate.

Keras, pe de altă parte, este un API de nivel înalt scris în Python care rulează peste TensorFlow, simplificând semnificativ procesul de dezvoltare a rețelelor neuronale. Principalele sale avantaje includ o interfață intuitivă, modularitate și extensibilitate, precum și acces la o colecție vastă de modele pre-antrenate (precum VGG16, InceptionV3 și ResNet50) care au fost esențiale în implementarea soluției.

Combinația TensorFlow-Keras a permis implementarea eficientă a arhitecturilor complexe de rețele neuronale, beneficiind atât de puterea de calcul a TensorFlow, cât și de simplitatea și flexibilitatea oferită de Keras.

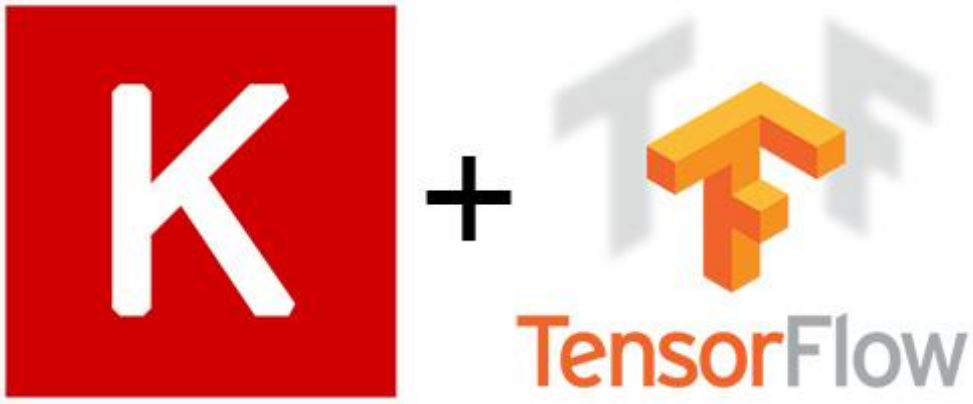

Figura 6: Keras & TensorFlow[11]

3.2.2 Tkinter pentru interfața grafică

Tkinter reprezintă biblioteca standard pentru dezvoltarea interfețelor grafice în Python, fiind aleasă datorită simplității și fiabilității sale. Este o soluție orientată spre aplicații desktop, oferind toate componentele necesare pentru crearea unei interfețe intuitive pentru clasificarea imaginilor RMN.

În cazul de față, Tkinter facilitează încărcarea și vizualizarea imaginilor medicale, precum și afișarea rezultatelor clasificării de la multiple modele într-o manieră clară și organizată.

3.2.3 Biblioteci suport (NumPy, Pandas, Matplotlib)

Implementarea se bazează pe trei biblioteci Python fundamentale pentru procesarea și analiza datelor:

NumPy este utilizat pentru procesarea eficientă a imaginilor RMN, oferind suport pentru operațiile matriceale necesare în manipularea și normalizarea datelor de intrare. Biblioteca permite operații rapide pe seturi mari de date și gestionarea eficientă a memoriei.

Pandas este folosit pentru organizarea și analiza rezultatelor experimentale, permițând manipularea. Cu ajutorul său gestionăm metricile de performanță ale modelelor, rezultatele clasificării și datele de antrenament într-un format ușor de analizat.

Matplotlib este responsabil pentru toate graficele de antrenament (acuratețe și loss) până la reprezentările comparative ale performanței diferitelor modele implementate. Biblioteca oferă funcționalități complete pentru crearea de grafice.

Figura 7: NumPy, Pandas, Matplotlib [12]

3.3 Setul de date

3.3.1 Prezentarea setului de date

Pentru realizarea acestei lucrări am utilizat un set de date de imagini RMN (Rezonanță Magnetică Nucleară) cerebrale pentru detectarea și clasificarea bolii Alzheimer. Setul de date conține un total de 6400 de imagini care sunt clasificate în patru clase diferite corespunzătoare diferitelor stadii ale bolii:

- Non_Demented (fără demență): 3200 imagini
- Very_Mild_Demented (demență foarte ușoară): 2240 imagini
- Mild_Demented (demență ușoară): 896 imagini
- Moderate_Demented (demență moderată): 64 imagini

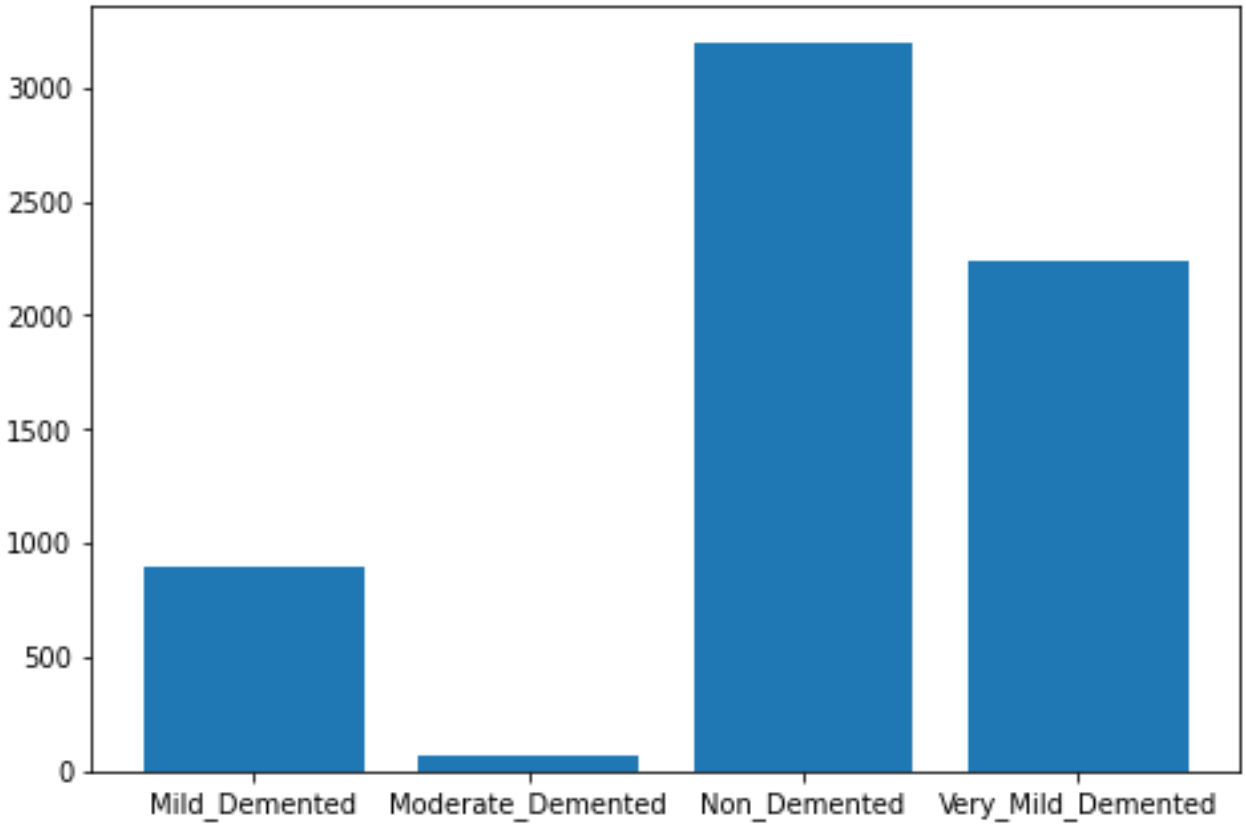

Figura 8: Distributia setului de date

Setul de date prezintă un dezechilibru semnificativ al claselor, cazurile Non_Demented fiind predominante, constituind 50% din totalul imaginilor, în timp ce cazurile Very_Mild_Demented reprezintă aproximativ 35%. Cazurile Mild_Demented constituie aproximativ 14% din setul de

date, în timp ce clasa Moderate_Demented are cea mai scăzută reprezentare, constituind doar 1% din totalul imaginilor.

Pentru a evalua configurația experimentală, setul de date a fost împărțit pe baza următoarelor raporturi:

- Set de antrenare (train): 80% din totalul datelor (5120 imagini)
- Set de validare (validation): 10% din totalul datelor (640 imagini)
- Set de testare (test): 10% din totalul datelor (640 imagini)

Fiecare imagine din setul de date reprezintă o scanare cerebrală RMN, oferind astfel atât clinicienilor, cât și sistemului o perspectivă vizuală asupra modificărilor structurale ale creierului asociate cu diferitele stadii ale bolii Alzheimer.

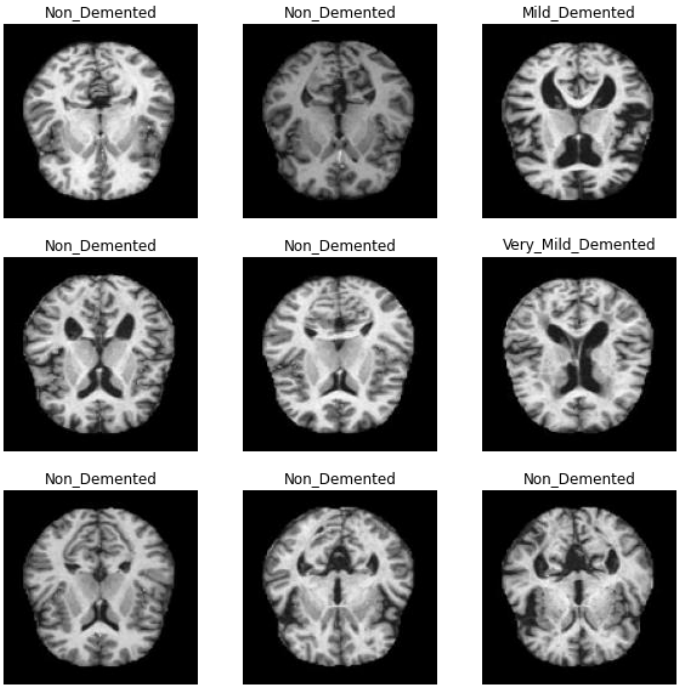

Figura 9: Exemplu cu seturi de imagini

3.3.2 Preprocesarea imaginilor

Pentru pregătirea setului de date de imagini RMN pentru detecția bolii Alzheimer, am implementat două etape principale de preprocesare folosind biblioteca TensorFlow:

1. Redimensionarea imaginilor la dimensiunea standard de 128x128 pixeli, pentru a asigura uniformitatea datelor de intrare și pentru optimizarea resurselor computaționale în procesul de antrenare a modelelor.

2. Rescalarea valorilor pixelilor din intervalul [0, 255] în intervalul [0, 1] folosind factorul de scalare 1/255, pentru a normaliza datele de intrare. Această transformare este esențială pentru stabilizarea procesului de învățare și accelerarea convergenței modelelor.

Procesul de preprocesare a fost implementat folosind clasa ImageDataGenerator din TensorFlow, care facilitează încărcarea și preprocesarea imaginilor. Setul de date a fost organizat în directoare separate pentru fiecare clasă (Non_Demented, Mild_Demented, Moderate_Demented și Very_Mild_Demented), respectând structura necesară pentru funcția flow_from_directory(). Imaginile RMN sunt tratate ca tensori 3D, păstrând cele trei canale de culoare RGB pentru a menține toate informațiile relevante din scanările cerebrale.

```
model.add(tf.keras.layers.Rescaling(1. / 255, input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)))
```

Figura 10: Preprocesarea datelor

3.4 Arhitecturile implementate

3.4.1 Arhitectura CNN proprie(Custom CNN)

Pentru detectarea bolii Alzheimer am propus o arhitectură CNN personalizată care constă în următoarele straturi:

1. Stratul de preprocesare:
    - Rescaling (1/255) pentru normalizarea imaginilor de intrare de dimensiune 128x128x3
2. Primul bloc convoluțional:
    - Strat convoluțional cu 16 filtre
    - Dimensiune kernel 3x3
    - Padding 'same' pentru păstrarea dimensionalității
    - Activare ReLU
    - Inițializare kernel "he_normal"
    - MaxPooling cu dimensiune 2x2 pentru reducerea dimensionalității
3. Al doilea bloc convoluțional:
    - Strat convoluțional cu 32 filtre
    - Dimensiune kernel 3x3
    - Padding 'same'
    - Activare ReLU

- Inițializare kernel "he_normal"
- MaxPooling cu dimensiune 2x2
- Dropout 0.20 pentru reducerea overfitting-ului

4. Al treilea bloc convoluțional:
    - Strat convoluțional cu 64 filtre
    - Dimensiune kernel 3x3
    - Padding 'same'
    - Activare ReLU
    - Inițializare kernel "he_normal"
    - MaxPooling cu dimensiune 2x2
    - Dropout 0.25
5. Straturi Dense pentru clasificare:
    - Flatten pentru transformarea feature map-urilor în vector unidimensional
    - Dense cu 128 neuroni și activare ReLU
    - Dense cu 64 neuroni și activare ReLU
    - Stratul final de clasificare cu 4 neuroni și activare softmax pentru cele 4 clase
6. Compilarea modelului:
    - Funcție loss(pierdere): sparse_categorical_crossentropy
    - Optimizer: Adam
    - Metrică urmărită: accuracy

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 128, 128, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 128, 128, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2097280
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 260
=================================================================
Total params: 2,129,380
Trainable params: 2,129,380
Non-trainable params: 0
```

Figura 11: Arhitectura CNN proprie

Această arhitectură folosește straturi convoluționale cu un număr crescător de filtre (16, 32, 64) pentru a învăța progresiv caracteristici din ce în ce mai complexe din imaginile RMN, în timp ce straturile de MaxPooling reduc dimensionalitatea și straturile de Dropout ajută la prevenirea overfitting-ului.

3.4.2 VGG16

VGG16 este o rețea neuronală convoluțională profundă dezvoltată de cercetătorii K. Simonyan și A. Zisserman de la Universitatea Oxford. Această arhitectură este cunoscută pentru structura sa uniformă și rezultatele excelente în domeniul clasificării imaginilor.

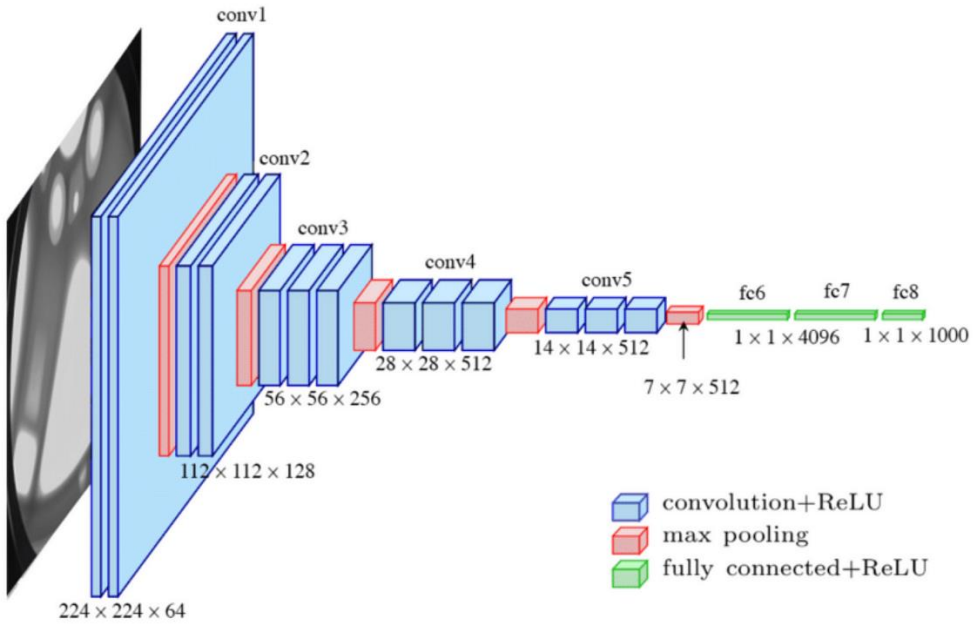

Figura 12: Arhitectura VGG16 [13]

Arhitectura VGG16 pentru lucrarea prezentată este structurată astfel:

1. Stratul de intrare acceptă imagini RMN cu dimensiunea de 128x128 pixeli și 3 canale de culoare (RGB)
2. Structura straturilor convoluționale:
    - 2 straturi convoluționale cu 64 filtre + MaxPooling
    - 2 straturi convoluționale cu 128 filtre + MaxPooling
    - 3 straturi convoluționale cu 256 filtre + MaxPooling
    - 3 straturi convoluționale cu 512 filtre + MaxPooling
    - 3 straturi convoluționale cu 512 filtre + MaxPooling
3. Straturi finale adaptate pentru clasificarea Alzheimer:
    - Un strat Dense cu 128 neuroni
    - Un strat Dense cu 64 neuroni
    - Stratul de ieșire cu 4 neuroni (corespunzător celor 4 clase) și funcție de activare softmax

Toate straturile convoluționale folosesc filtre de dimensiune 3x3 și funcția de activare ReLU. Straturile de MaxPooling utilizează o fereastră de 2x2 pixeli cu un pas (stride) de 2. În implementare, am folosit transfer learning, păstrând greutățile pre-antrenate ale VGG16 pe ImageNet pentru straturile convoluționale. Aceasta abordare ne permite să beneficiem de caracteristicile generale învățate pe un set mare de date și să le adaptăm pentru cazul specific al detectării Alzheimer. Pentru optimizarea rețelei am folosit algoritmul Adam cu o rată de învățare de 0.0001, care s-a dovedit a fi cea mai eficientă alegere pentru acest model.

```
base_model = VGG16(
    weights='imagenet',
    include_top=False,
    input_tensor=input_tensor
)

# Freeze the base model layers
base_model.trainable = False

# Add custom classification layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(4, activation='softmax')(x)

# Create the model
model = Model(inputs=input_tensor, outputs=predictions)

# Compile the model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Figura 13: Implementarea arhitecturii VGG16

3.4.3 InceptionV3

InceptionV3 este o arhitectură avansată de rețea neuronală convoluțională, care se remarcă prin eficiența sa computațională și numărul redus de parametri în comparație cu alte arhitecturi precum VGG16.

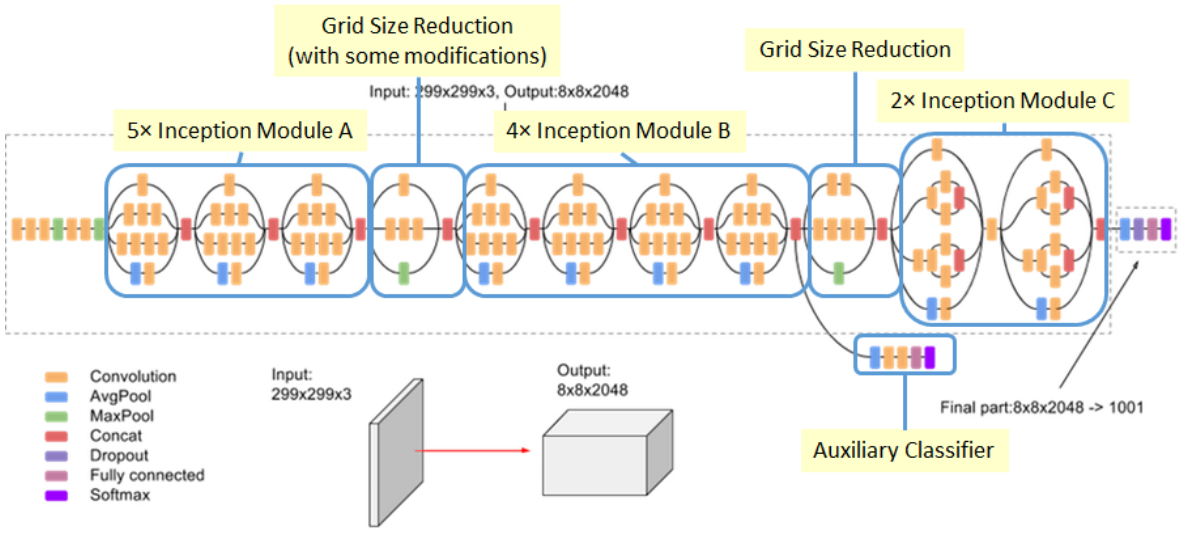

Figura 14: Arhitectura InceptionV3 [14]

În implementare, am utilizat modelul InceptionV3 cu următoarea configurație:

1. Stratul de intrare acceptă imagini RMN de dimensiune 128x128 pixeli cu 3 canale de culoare
2. Am folosit arhitectura de bază a InceptionV3 cu greutățile pre-antrenate pe ImageNet, beneficiind astfel de caracteristicile generale învățate pe un set extins de date.
3. Straturile finale adaptate pentru clasificarea Alzheimer:
    - Un strat Global Average Pooling pentru reducerea dimensionalității
    - Un strat Dense cu 128 neuroni și activare ReLU
    - Un strat Dense cu 64 neuroni și activare ReLU
    - Stratul de ieșire este compus din 4 neuroni și folosește funcția de activare softmax pentru clasificarea multi-class

Pentru optimizarea modelului am folosit algoritmul Adam cu o rată de învățare de 0.0001. Această arhitectură oferă un bun compromis între performanță și eficiență computațională pentru sarcina de clasificare a stadiilor bolii Alzheimer.

```
base_model = InceptionV3(
    weights='imagenet',
    include_top=False,
    input_tensor=input_tensor
)

# Freeze the base model layers
base_model.trainable = False

# Add custom classification layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(4, activation='softmax')(x)

# Create the model
model = Model(inputs=input_tensor, outputs=predictions)

# Compile the model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Figura 15: Implementarea arhitecturii InceptionV3

3.4.4 ResNet50

ResNet50 este o arhitectură de rețea neuronală convoluțională profundă care a câștigat competiția ILSVRC 2015. În implementare, am configurat ResNet50 astfel:

1. Stratul de intrare acceptă imagini RMN de dimensiune 128x128 pixeli cu 3 canale de culoare
2. Arhitectura de bază a ResNet50 constă din:
    - Convoluție inițială cu kernel 7x7 și MaxPooling 3x3
    - 4 etape de blocuri reziduale:
        - Etapa 1: 3 blocuri (64, 64, 256)
        - Etapa 2: 4 blocuri (128, 128, 512)
        - Etapa 3: 6 blocuri (256, 256, 1024)
        - Etapa 4: 3 blocuri (512, 512, 2048)

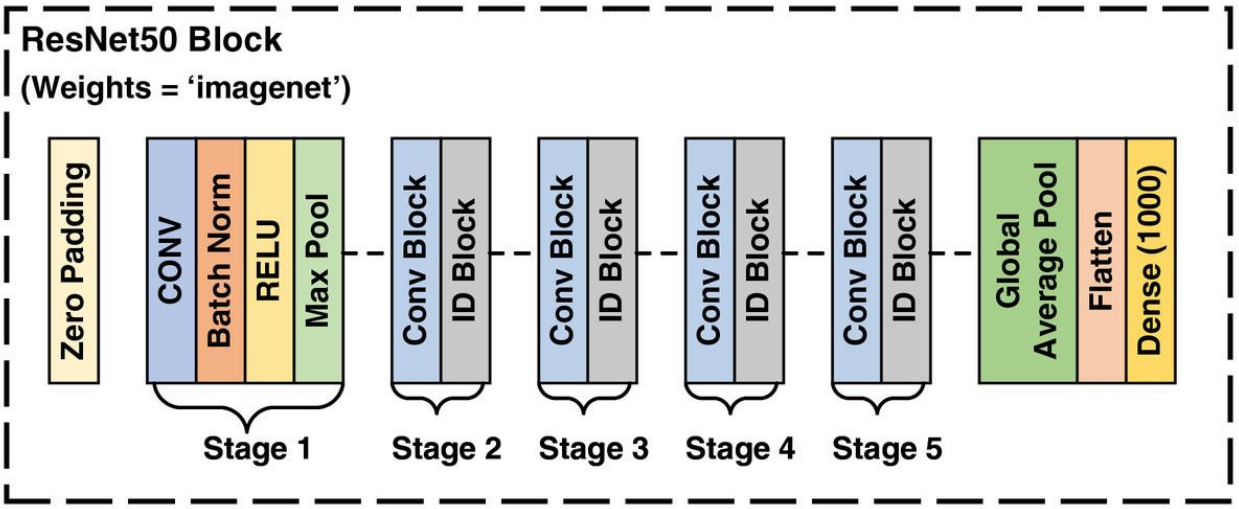

Figura 16: Arhitectura ResNet50 [15]

3. Straturile finale adaptate pentru clasificarea Alzheimer:
    - Un strat Global Average Pooling
    - Un strat Dense cu 128 neuroni și activare ReLU
    - Un strat Dense cu 64 neuroni și activare ReLU
    - Stratul de ieșire cu 4 neuroni și funcție de activare softmax

Pentru optimizarea modelului am folosit algoritmul Adam cu o rată de învățare de 0.0001.

```
base_model = ResNet50(
    weights='imagenet',
    include_top=False,
    input_tensor=input_tensor
)

# Freeze the base model layers
base_model.trainable = False

# Add custom classification layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(4, activation='softmax')(x)

# Create the model
model = Model(inputs=input_tensor, outputs=predictions)

# Compile the model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Figura 17: Implementarea arhitecturii ResNet50

3.5 Implementarea soluției

3.5.1 Pipeline-ul de antrenare

Pipeline-ul de antrenare implementat pentru detecția Alzheimer-ului poate fi împărțit în mai multe etape esențiale:

1. **Încărcarea și organizarea setului de date**: Primul pas este încărcarea setului de date utilizând API-ul *tf.keras.preprocessing.image_dataset_from_directory*, care optimizează folosirea memoriei prin prelucrarea datelor în batch-uri:

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "./output/train",
    seed=123,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=64
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "./output/val",
    seed=123,
    image_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=64
)
```

Figura 18: Încărcarea și organizarea setului de date

2. **Definirea și inițializarea modelelor**: Pentru fiecare arhitectură (Custom CNN, VGG16, InceptionV3, ResNet50), pipeline-ul creează și configurează modelul corespunzător. În cazul modelelor pre-antrenate, se realizează și transferul de învățare:

```
# Creare modele
custom_model = create_custom_model()
inception_model = create_inception_model()
vgg16_model = create_vgg16_model()
resnet_model = create_resnet_model()

# Compilare cu parametri optimizati
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

Figura 19: Definirea și inițializarea modelelor

3. **Configurarea mecanismelor de monitorizare și optimizare**: Pentru fiecare model, se configurează funcții callback specifice pentru monitorizarea și optimizarea procesului de antrenare:

```
# Define callbacks
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True,
    mode='min',
    verbose=1
)

# Add ModelCheckpoint to save the best model during training
checkpoint = ModelCheckpoint(
    f'{model_name.lower().replace(" ", "_")}.keras',
    monitor='val_accuracy',
    mode='max',
    save_best_only=True,
    verbose=1
)
```

Figura 20: Configurarea mecanismelor de monitorizare și optimizare

4. **Procesul de antrenare**: Antrenarea efectivă a modelelor se realizează într-o manieră iterativă pentru fiecare arhitectură:

```
for model, name in zip(models, model_names):
    history = train_model(model, train_ds, val_ds, name)
    histories.append(history)

    # Save the trained model
    model.save(f'{name.lower().replace(" ", "_")}_model.keras')
```

Figura 21: Procesul de antrenare

5. **Salvarea și evaluarea modelelor:** După antrenare, fiecare model este salvat și apoi evaluat pe setul de date de testare:

```
# Evaluate models on test set
results_df = evaluate_models(models, test_ds, model_names)
```

Figura 22: Salvarea și evaluarea modelelor

Pentru toate cele patru modele implementate (Custom CNN, VGG16, InceptionV3 și ResNet50), au fost utilizați aceiași parametri de antrenare pentru a permite o evaluare constantă a performanței. Antrenarea a fost configurată cu o dimensiune a batchului de 64, o limită de 100 de epoci și utilizarea optimizatorului Adam.. Early stopping a fost configurat cu o răbdare (patience) de 10 epoci pentru toate modelele, monitorizând loss-ul pe setul de validare.

3.5.2 Strategii de optimizare

În timpul implementării, au fost utilizate o varietate de tehnici de optimizare pentru a îmbunătăți stabilitatea și performanța modelelor. Aceste tehnici au fost folosite în faza de antrenare, precum și în proiectarea structurală a modelelor

1. **Dropout pentru regularizare**: Pentru prevenirea supraantrenării(overfitting), au fost utilizate straturi Dropout cu rate variabile în funcție de adâncimea rețelei. În algoritmul dezvoltat, am folosit:

```
model.add(tf.keras.layers.Dropout(0.20))#Dupa al doilea bloc convoluțional
model.add(tf.keras.layers.Dropout(0.25))#Dupa al treilea bloc convoluțional
```

2. **Inițializarea Kernel-urilor**: Pentru toate straturile convoluționale, a fost utilizată strategia de inițializare „he_normal”, care este optimizată pentru funcția de activare ReLU:

```
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu',
                                 kernel_initializer="he_normal"))
```

Figura 23: Inițializarea Kernel-urilor

3. **Transfer Learning și Fine-Tuning**: Pentru modelele pre-antrenate (VGG16, InceptionV3, ResNet50), a fost utilizată metoda de învățare prin transfer prin înghețarea straturilor convoluționale și prin antrenarea numai a straturilor dense superioare:

```
# Freeze the base model layers
base_model.trainable = False

# Add custom classification layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
predictions = Dense(4, activation='softmax')(x)
```

Figura 24: Transfer Learning și Fine-Tuning

4. **Optimizarea cu Adam:** Optimizatorul Adam a fost utilizat pentru toate modelele, deoarece poate ajusta rata de învățare în funcție de fiecare parametru:

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="adam", metrics=["accuracy"])
```

Figura 25: Optimizarea cu Adam

5. **Early Stopping & Model Checkpoint**: După cum se descrie în secțiunea 3.5.1, au fost utilizate două mecanisme complementare în scopul optimizării și salvării modelului. Early Stopping monitorizează „val_loss" și oprește antrenamentul atunci când nu se observă nicio îmbunătățire pe parcursul a 10 epoci, restabilind cele mai bune ponderi găsite simultan. În plus, ModelCheckpoint salvează versiunile modelului care au cea mai mare precizie pe setul de validare. Această strategie garantează salvarea celei mai performante versiuni a modelului, chiar și în timpul antrenării continue.

Rezultatele experimentale au demonstrat eficiența acestor tehnici în reducerea overfitting-ului și îmbunătățirea acurateței generale a clasificării.

3.5.3 Mecanisme de validare

Am utilizat mai multe tehnici de validare pentru a asigura acuratețea și fiabilitatea modelelor. Aceste tehnici au fost incluse atât în faza de antrenare, cât și în evaluarea ulterioară a modelelor:

1. **Validare în timpul antrenamentului:** În faza de antrenare, fiecare model este evaluat continuu pe setul de validare pentru a monitoriza performanța și a preveni supraantrenarea:

```
hist = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=100,
    batch_size=64,
    verbose=1,
    callbacks=[early_stopping, checkpoint]
)
```

Figura 26: Validare în timpul antrenamentului

2. **Evaluarea pe setul de test**: La sfârșitul antrenării, fiecare model este testat pe setul de testare independent pentru a obține o estimare a performanței:

```
def evaluate_models(models, test_ds, model_names):
    results = []
    for model, name in zip(models, model_names):
        print(f"\nEvaluating {name}...")
        test_loss, test_accuracy = model.evaluate(test_ds)
        results.append({
            'Model': name,
            'Test Loss': test_loss,
            'Test Accuracy': test_accuracy
        })

    return pd.DataFrame(results)
```

Figura 27: Evaluarea pe setul de test

3. **Monitorizarea progresului antrenării:** Pentru a demonstra și evalua progresul modelelor în timpul antrenării, am creat funcții de vizualizare pentru metrici cheie:

```
def plot_comparison_results(histories, model_names):
    metrics = ['accuracy', 'loss']
    fig, axes = plt.subplots(1, 2, figsize=(15, 5))

    for metric, ax in zip(metrics, axes):
        for hist, name in zip(histories, model_names):
            ax.plot(hist.history[metric], label=f'{name} (train)')
            ax.plot(hist.history[f'val_{metric}'], label=f'{name} (val)')

        ax.set_title(f'Model {metric.capitalize()}')
        ax.set_xlabel('Epoch')
        ax.set_ylabel(metric.capitalize())
        ax.legend()

    plt.tight_layout()
    plt.show()
```

Figura 28: Monitorizarea progresului antrenării

3.5.4 Aplicația de clasificare

Aplicația dezvoltată prezintă o interfață intuitivă pentru analiza imaginilor RMN și determinarea stadiilor bolii Alzheimer. Implementarea sa cuprinde trei componente principale: structura aplicației, interfața cu utilizatorul și procesul de clasificare.

În ceea ce privește structura aplicației, aceasta este construită folosind o arhitectură modulară în Python care garantează o delimitare clară a responsabilităților. La nivelul de bază, sunt definite constante esențiale precum dimensiunea imaginii (128x128 pixeli) și categoriile pentru stadiile bolii. La lansare, aplicația încarcă cele patru modele alocate pentru clasificare din locațiile lor respective salvate, gata pentru utilizare imediată.

Interfața cu utilizatorul este implementată folosind Tkinter, favorizând designul modern și ușurința în utilizare. Fereastra principală este plasată central pe ecran, cu dimensiuni optimizate de 600x700 pixeli și utilizează o schemă de culori atractivă din punct de vedere vizual, dominată de nuanțe de albastru deschis (#B8DBD9). Principalele elemente ale interfeței includ o secțiune centrală pentru afișarea imaginii RMN încărcate, un buton de încărcare a imaginii și o zonă desemnată pentru afișarea rezultatelor multiple ale clasificării.

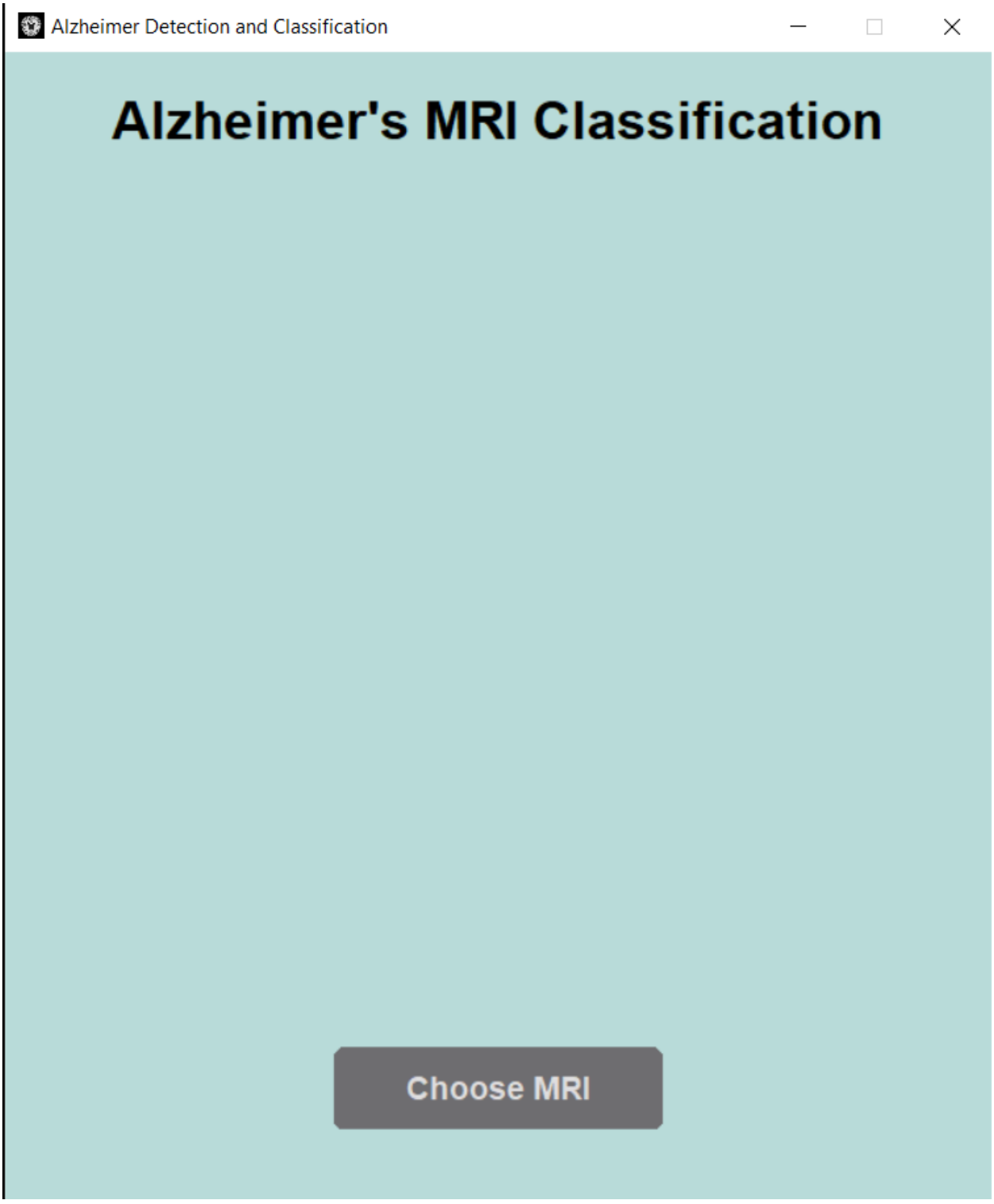

Figura 29: Aplicația de clasificare

Procesul de clasificare începe cu preprocesarea imaginii încărcate, ce implică citirea fișierului, decodarea acestuia în format JPEG utilizând trei canale de culoare și redimensionarea acestuia la dimensiunea standard de 128x128 pixeli. După preprocesare, imaginea este afișată în interfață ca o previzualizare de 200x200 pixeli. Clasificarea efectivă este efectuată în paralel pe

toate modelele încărcate, fiecare predicție fiind reprezentată în interfață printr-un cod de culori: verde pentru predicțiile corecte și roșu pentru predicțiile incorecte, în funcție de cunoașterea clasei actuale.

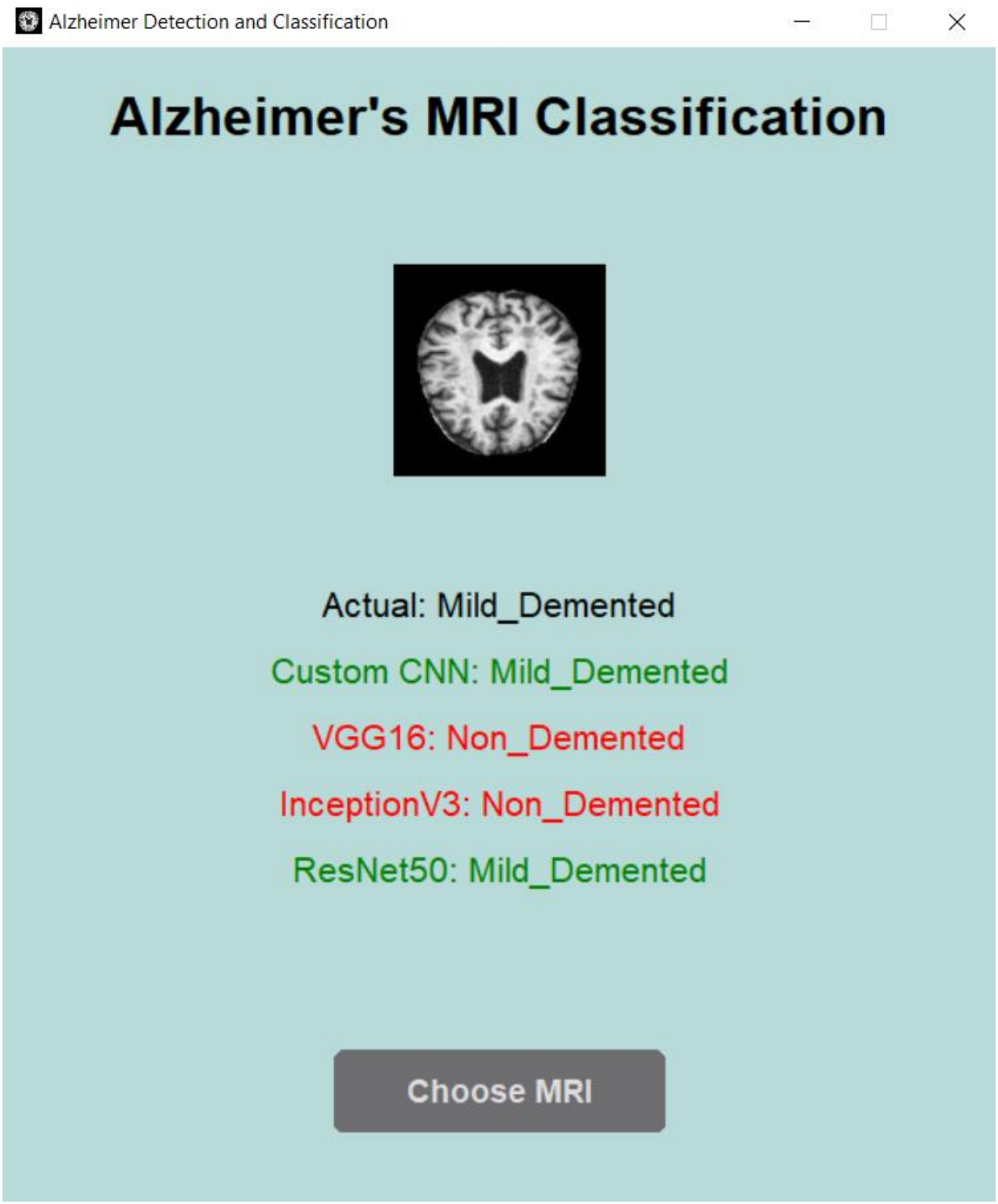

Figura 30: Aplicația de clasificare

3.6 Evaluarea modelelor

3.6.1 Metrici de performanță

Pentru evaluarea riguroasă a modelelor de clasificare a bolii Alzheimer, am utilizat un set complet de indicatori de performanță, care au fost calculați pe setul de testare independent. Implementarea acestor măsuri a fost realizată folosind funcționalitatea oferită de bibliotecile TensorFlow și scikit-learn.

Principalul criteriu de evaluare a performanței generale a fost acuratețea (accuracy), calculată pentru fiecare model individual utilizând metoda evaluate_models():

```
def evaluate_models(models, test_ds, model_names):
    results = []
    for model, name in zip(models, model_names):
        print(f"\nEvaluating {name}...")
        test_loss, test_accuracy = model.evaluate(test_ds)
        results.append({
            'Model': name,
            'Test Loss': test_loss,
            'Test Accuracy': test_accuracy
        })
```

Figura 31: Funcția pentru măsurarea acurateții

Pentru fiecare model și clasă, am calculat trei măsurători cheie: Acuratețe, Recall și F1-Score. Acești parametri au fost determinați utilizând implementarea scikit-learn, oferind o imagine detaliată a performanței fiecărui model pe setul de testare. Pentru a calcula acești parametri, am implementat următoarea funcționalitate:

```python
def calculate_performance_metrics(model_path, test_ds, class_names):
    model = tf.keras.models.load_model(model_path)
    predictions = []
    true_labels = []

    for images, labels in test_ds:
        pred = model.predict(images, verbose=0)
        predictions.extend(np.argmax(pred, axis=1))
        true_labels.extend(labels.numpy())

    precision = precision_score(true_labels, predictions, average=None)
    recall = recall_score(true_labels, predictions, average=None)
    f1 = f1_score(true_labels, predictions, average=None)

    metrics_df = pd.DataFrame({
        'Class': class_names,
        'Precision': [f'{x:.3f}' for x in precision],
        'Recall': [f'{x:.3f}' for x in recall],
        'F1-Score': [f'{x:.3f}' for x in f1]
    })

    return metrics_df
```

Figura 32: Funcția pentru măsurarea celorlalte metrici

Pentru a monitoriza procesul de antrenare, am urmărit evoluția pierderilor și a preciziei atât pe setul de antrenare, cât și pe setul de validare:

```python
def plot_comparison_results(histories, model_names):
    metrics = ['accuracy', 'loss']
    fig, axes = plt.subplots(1, 2, figsize=(15, 5))

    for metric, ax in zip(metrics, axes):
        for hist, name in zip(histories, model_names):
            ax.plot(hist.history[metric], label=f'{name} (train)')
            ax.plot(hist.history[f'val_{metric}'], label=f'{name} (val)')

        ax.set_title(f'Model {metric.capitalize()}')
        ax.set_xlabel('Epoch')
        ax.set_ylabel(metric.capitalize())
        ax.legend()

    plt.tight_layout()
    plt.show()
```

Figura 33: Funcția pentru măsurarea pierderilor

Acești parametri oferă o imagine completă a performanței modelelor, permițând atât o evaluare generală a acurateței, cât și o analiză detaliată a comportamentului pentru fiecare clasă. Subliniem faptul că, pentru problema clasificării bolii Alzheimer, este deosebit de important să se mențină un echilibru între acuratețe și recall, deoarece atât fals-pozitivele, cât și fals-negativele pot avea implicații semnificative în context medical.

3.6.2 Comparația modelelor

Analizând rezultatele experimentale prezentate în Figura 34, putem observa că există diferențe semnificative în performanța celor patru modele implementate. Modelul Custom CNN a obținut cea mai bună acuratețe globală de 99,53%, urmat de VGG16 cu 92,02%, în timp ce ResNet50 și InceptionV3 au performanțe mai modeste, cu o acuratețe de 84,04% și, respectiv, 75,27%. De asemenea, loss-ul modelelor urmează un model similar, Custom CNN având cea mai mică pierdere (0,0160), în timp ce InceptionV3 prezintă cea mai mare pierdere (0,6023).

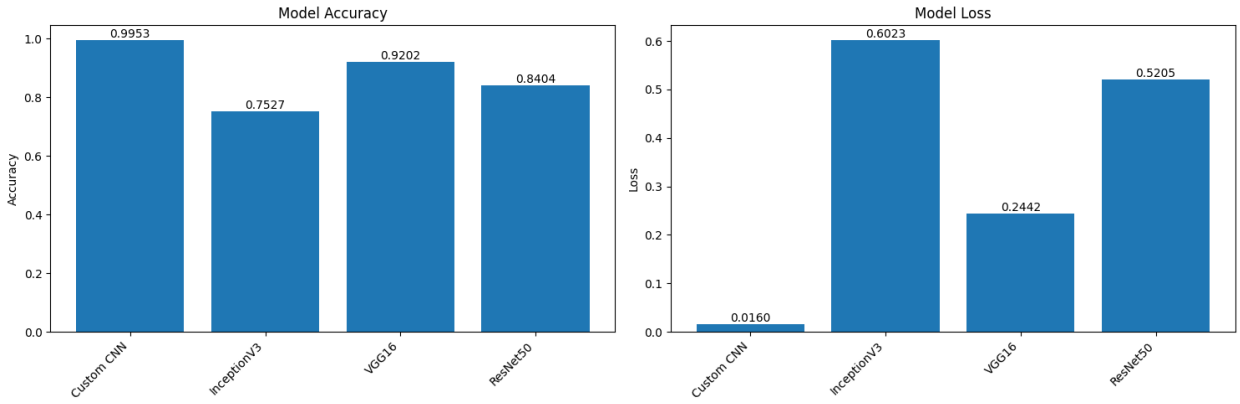

Figura 34: Comparație a acurateții și pierderilor ale modelelor

Analizând matricele de confuzie și metricile detaliate pentru fiecare model:

Custom CNN (Figura 35):

- Prezintă cea mai bună performanță în clasificarea tuturor claselor
- Foarte puține erori de clasificare între clase (doar 2-3 cazuri pe categorie)
- Menține o acuratețe ridicată pentru toate clasele de demență

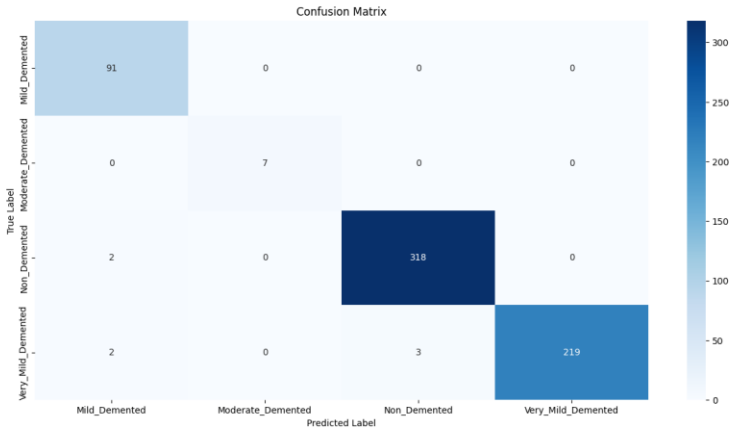

Figura 35: Matricea de confuzie pentru modelul propriu

VGG16 (Figura 36):

- Performanță bună, dar cu mai multe confuzii între clase
- Prezintă confuzii mai frecvente între Very_Mild_Demented și Non_Demented
- Menține o acuratețe bună pentru Moderate_Demented.

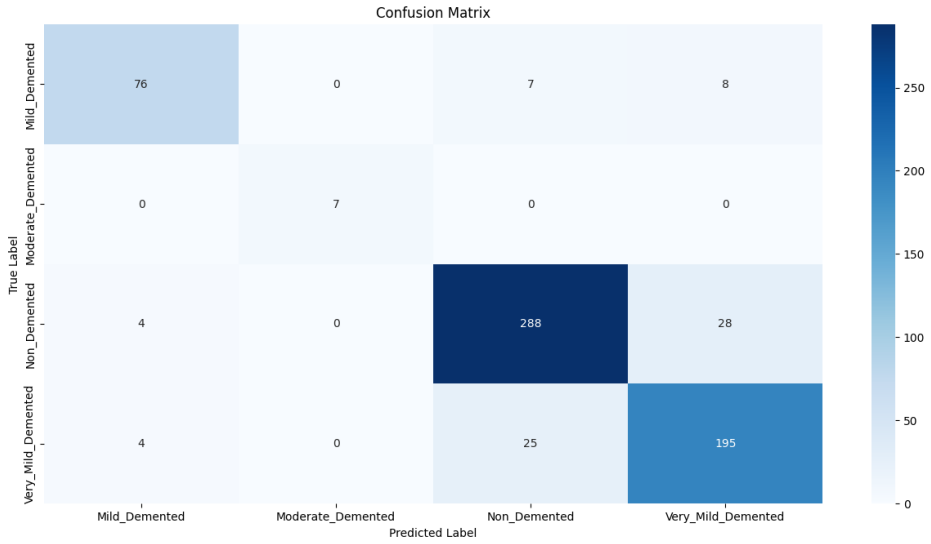

Figura 36: Matricea de confuzie pentru modelul VGG16

InceptionV3 (Figura 37):

- Performanță moderată cu confuzie semnificativă între clase
- Dificultate în diferențierea între Very_Mild_Demented și Non_Demented
- Cele mai multe clasificări greșite pentru Mild_Demented.

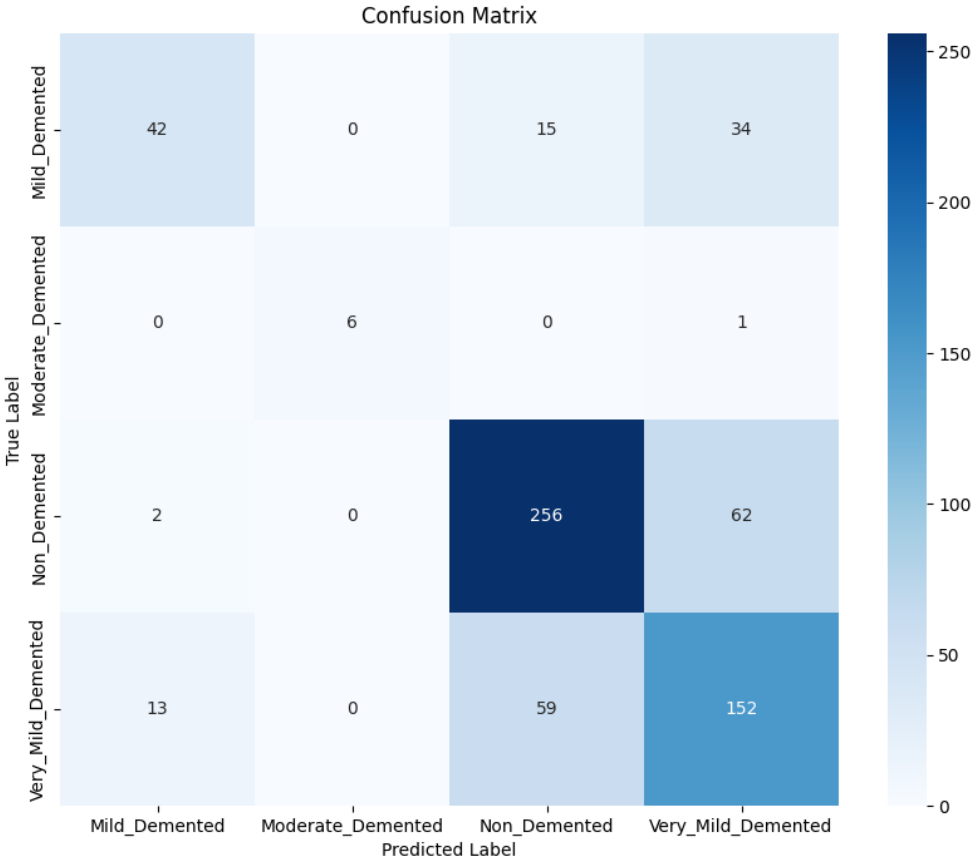

Figura 37: Matricea de confuzie pentru modelul InceptionV3

ResNet50 (Figura 38):

- Performanță similară cu InceptionV3
- Confuzii notabile între Very_Mild_Demented și Non_Demented
- Mai puține erori în clasificarea Mild_Demented în comparație cu InceptionV3.

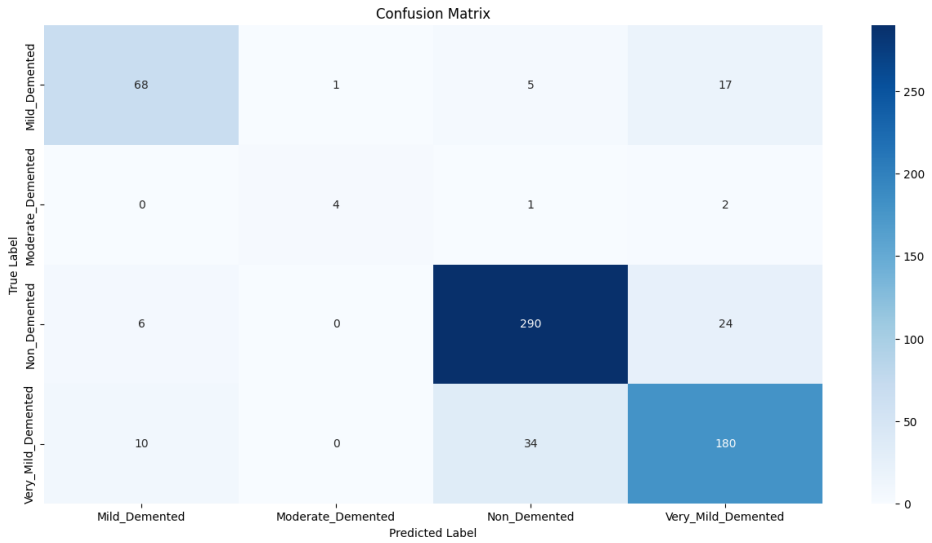

Figura 38: Matricea de confuzie pentru modelul ResNet50

Analizând metricile detaliate din Figura 39, este clar că modelul Custom CNN are coerență în toate măsurile de performanță (precizie, recall, scor F1) pentru fiecare categorie, cu valori de peste 0,95. VGG16, pe de altă parte, prezintă rezultate bune de performanță, dar inegale intre clase, în timp ce atât InceptionV3, cât și ResNet50 se confruntă cu mai multe dificultăți în diferențierea anumitor categorii, după cum reiese din scorurile mai mici.

Class	Precision	Recall	F1-Score	Model
Mild_Demented	0.958	1.000	0.978	Custom CNN
Moderate_Demented	1.000	1.000	1.000	Custom CNN
Non_Demented	0.991	0.994	0.992	Custom CNN
Very_Mild_Demented	1.000	0.978	0.989	Custom CNN
Mild_Demented	0.905	0.835	0.869	VGG16
Moderate_Demented	1.000	1.000	1.000	VGG16
Non_Demented	0.900	0.900	0.900	VGG16
Very_Mild_Demented	0.844	0.871	0.857	VGG16
Mild_Demented	0.737	0.462	0.568	InceptionV3
Moderate_Demented	1.000	0.857	0.923	InceptionV3
Non_Demented	0.776	0.800	0.788	InceptionV3
Very_Mild_Demented	0.610	0.679	0.643	InceptionV3
Mild_Demented	0.810	0.747	0.777	ResNet50
Moderate_Demented	0.800	0.571	0.667	ResNet50
Non_Demented	0.879	0.906	0.892	ResNet50
Very_Mild_Demented	0.807	0.804	0.805	ResNet50

Figura 39: Metrici de perfomanță

O observație importantă este că toate modelele au cea mai bună performanță în clasificarea instanțelor Non_Demented, ceea ce se datorează cel mai probabil numărului mai mare de date de antrenare pentru această clasă. Clasificarea între Very_Mild_Demented și Non_Demented este o provocare comună pentru toate modelele, cu excepția Custom CNN, sugerând că acele caracteristici subtile care disting cele două clase sunt mai bine reprezentate de arhitectura customizată.

3.6.3 Analiza rezultatelor

Conform rezultatelor prezentate în secțiunile anterioare, se pot face unele concluzii interesante cu privire la eficacitatea și comportamentul modelelor utilizate pentru detectarea bolii Alzheimer.

Examinarea preciziei globale, astfel cum se arată în figura 34, oferă o ierarhie clară a modelelor: Custom CNN (99,53%) > VGG16 (92,02%) > ResNet50 (84,04%) > InceptionV3 (75,27%). Această ierarhie este susținută și de valorile pierderilor, unde Custom CNN are cea mai mică pierdere (0,0160) și InceptionV3 are cea mai mare (0,6023).

O observație interesantă este superioritatea modelului Custom CNN asupra arhitecturilor pre-antrenate. Acest rezultat neașteptat ar putea fi explicat printr-o serie de ipoteze: arhitectura relativ mai simplă a modelului Custom CNN s-ar putea potrivi mai bine proprietăților particulare ale imaginilor RMN; în timp ce modelele pre-antrenate, chiar și cu complexitatea lor, ar putea fi susceptibile la o posibilă supraspecializare pentru imaginile naturale; iar procesul de învățare prin transfer ar putea necesita o reglare mai precisă pentru acest tip specific de date medicale.

Atunci când se iau în considerare matricele de confuzie (figurile 35, 36, 37 și 38), se pot observa unele caracteristici comune: toate modelele prezintă performanțe optime în clasificarea cazurilor Non_Demented, cel mai probabil datorită unei prezențe mai reprezentative în setul de date; există o tendință generală de confuzie între clasele Non_Demented și Very_Mild_Demented, ceea ce arată că detectarea diferențelor subtile dintre aceste faze este o provocare; iar clasa Moderate_Demented prezintă cele mai puține clasificări corecte, ceea ce poate fi atribuit numărului limitat de exemple disponibile în setul de date.

Din perspectiva metricilor detaliate (figura 39): Custom CNN prezintă un echilibru remarcabil între precizie și recall pentru toate clasele; în schimb, VGG16 prezintă performanțe acceptabile, dar inegale pentru diferitele clase. InceptionV3 și ResNet50 au mai multe dificultăți în menținerea unui echilibru între precizie și recall, în special pentru clasele cu mai puține exemple

Aceste rezultate sugerează că, în contextul specific al detecției Alzheimer-ului din imagini RMN, o arhitectură mai simplă dar specifică poate depăși modelele mai complexe pre-antrenate. Acest lucru are implicații importante pentru dezvoltarea viitoarelor sisteme de asistență în diagnosticarea Alzheimer-ului, unde simplitatea și acuratețea pot fi preferate complexității excesive.

Concluzii

Această lucrare a prezentat dezvoltarea și implementarea unui sistem de detecție și clasificare a bolii Alzheimer utilizând tehnici de deep learning aplicate pe imagini RMN cerebrale. Sistemul are capacitatea de a clasifica imaginile în patru categorii distincte: Non_Demented, Very_Mild_Demented, Mild_Demented și Moderate_Demented, oferind astfel un instrument potențial valoros pentru asistarea procesului de diagnostic medical.

Rezultatele experimentale au demonstrat eficacitatea deosebită a modelului Custom CNN, care a atins o acuratețe remarcabilă de 99.53%, depășind arhitecturi complexe pre-antrenate precum VGG16 (92.02%), ResNet50 (84.04%) și InceptionV3 (75.27%). Acest rezultat este cu atât mai semnificativ cu cât arhitectura customizată este mai simplă decât modelele pre-antrenate, sugerând că în cazul specific al imaginilor RMN cerebrale, o abordare specializată și mai directă poate fi mai eficientă decât utilizarea unor arhitecturi generale complexe.

Consider că aplicația dezvoltată are un potențial semnificativ în contextul medical actual, unde diagnosticarea timpurie a bolii Alzheimer reprezintă o provocare majoră. Sistemul poate servi ca instrument de screening preliminar, oferind medicilor specialiști un punct de plecare în procesul de diagnostic. Este important de menționat că acest sistem nu își propune să înlocuiască expertiza medicală, ci să o complementeze, oferind o analiză obiectivă și rapidă a imaginilor RMN.

Pentru îmbunătățirea ulterioară a sistemului, am identificat mai multe direcții potențiale de cercetare și dezvoltare. În primul rând, colectarea și includerea unui număr mai mare de imagini RMN, în special pentru clasele subreprezentate precum Moderate_Demented, ar putea îmbunătăți semnificativ robustețea modelului. O altă direcție importantă de dezvoltare ar fi adăugarea unor mecanisme care să evidențieze zonele din imaginile RMN care influențează cel mai mult decizia modelului, crescând astfel transparența și utilitatea sistemului în context medical. Transformarea aplicației într-o platformă web accesibilă ar putea facilita utilizarea sistemului în diverse contexte medicale, iar colaborarea cu instituții medicale pentru testarea extensivă a sistemului în condiții reale și validarea rezultatelor pe seturi de date independente ar aduce plus-valoare sistemului.

În concluzie, rezultatele obținute sunt promițătoare și demonstrează potențialul tehnologiilor de inteligență artificială în domeniul medical. Această lucrare demonstrează că inteligența artificială poate oferi suport valoros în domeniul medical, contribuind la îmbunătățirea procesului de diagnostic și, în ultimă instanță, la îmbunătățirea îngrijirii pacienților.

Bibliografie

1. Suk, H. I., Lee, S. W., & Shen, D. (2017). Deep ensemble learning of sparse regression models for brain disease diagnosis. Medical image analysis, 37, 101-113.
2. Islam, J., & Zhang, Y. (2019). Deep Convolutional Neural Networks for Automated Diagnosis of Alzheimer's Disease and Mild Cognitive Impairment Using 3D Brain MRI. In International Conference on Brain Informatics (pp. 359-369).
3. Valliani, A., & Soni, A. (2017). Deep residual learning for image recognition of Alzheimer's disease. Rev. Biomed. Eng, 10, 1-7.
4. https://media.licdn.com/dms/image/v2/D4D0BAQENHionw7dt3g/company-logo_200_200/company-logo_200_200/0/1704982814550/brainscan_ai_logo?e=2147483647&v=beta&t=oV4TgD10BDlSVYWnGV92KpJcljH0O8N84DchNsCPQIg
5. Ross, D. E., Ochs, A. L., DeSmit, M. E., Seabaugh, J. M., & Havranek, M. D. (2015). Man versus machine: comparison of radiologists' interpretations and NeuroQuant® volumetric analyses of brain MRIs in patients with traumatic brain injury. The Journal of neuropsychiatry and clinical neurosciences, 27(4), 322-329.
6. Brewer, J. B. (2009). Fully-automated volumetric MRI with normative ranges: translation to clinical practice. Behavioural neurology, 21(1-2), 21-28.
7. https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSAkw8J_NyhoRTX3T5uzufd3_6bbR_PH8bOKQ&s
8. https://adni.loni.usc.edu/wp-content/themes/adni_2023/images/adni-logo.png
9. https://miro.medium.com/v2/resize:fit:1400/0*RtZCRwjVNV8cJMoc.png
10. https://cdn.prod.website-files.com/5fb24a974499e90dae242d98/5fb24a974499e9ce892431d4_Untitled.png
11. https://blog.keras.io/img/keras-tensorflow-logo.jpg
12. https://img-c.udemycdn.com/course/750x422/4729170_1eb1.jpg
13. https://miro.medium.com/v2/resize:fit:850/1*B_ZaaaBg2njhp8SThjCufA.png
14. https://www.researchgate.net/publication/368398785/figure/fig3/AS:11431281179378845@1691187838826/InceptionV3-model-architecture-17.png
15. https://dfzljdn9uc3pi.cloudfront.net/2024/cs-2219/1/fig-3-2x.jpg
16. https://www.tensorflow.org/api_docs/python/tf/keras

17. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/
18. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
19. https://neurohive.io/en/popular-networks/vgg16/
20. https://keras.io/api/applications/inceptionv3/
21. https://iq.opengenus.org/resnet50-architecture/
22. https://docs.python.org/3/library/tkinter.html
23. https://matplotlib.org/stable/index.html
24. https://numpy.org/doc/stable/
25. https://pandas.pydata.org/docs/
26. https://www.who.int/news-room/fact-sheets/detail/dementia
27. https://keras.io/api/preprocessing/image/
28. https://www.tensorflow.org/tutorials/images/transfer_learning
29. https://scikit-learn.org/stable/modules/model_evaluation.html
30. https://www.alzheimers.gov/alzheimers-dementias/alzheimers-disease
31. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
32. https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be
33. https://www.kaggle.com/code/tutenstein/99-8-acc-alzheimer-detection-and-classification/input
34. https://machinelearningmastery.com/image-augmentation-deep-learning-keras/