



**ACADEMIA DE STUDII ECONOMICE DIN BUCUREŞTI**  
**Facultatea de Cibernetică, Statistică și Informatică Economică**

**Specializarea: Informatică Economică**

## **Lucrare de Licență**

### **-Tehnici de analiză avansată a datelor în domeniul medical**

**Cadrul didactic coordonator:**

**Prof. univ. doctor Șimonca (Botha) Iuliana**

**Absolvent:**

**Trăscău Teodor-Bogdan**

# Cuprins

Introducere.....	4
Capitolul 1. Fundamente ale rețelelor neuronale.....	5
1.1 Pionierii inteligenței artificiale. Lucrări ce au stat la baza apariției inteligenței artificiale.....	5
1.1.1 Norbert Wiener.....	5
1.1.2 Claude Shannon.....	7
1.1.3 Alan Turing.....	12
1.2 Perceptron.....	16
1.3 Backpropagation.....	18
1.3.1 Gradientul.....	20
1.3.2 Funcția de cost.....	25
1.3.3 Regula lanțului.....	27
1.3.4 Antrenarea rețelei.....	28
1.3.5 Forme ale algoritmului.....	35
1.4 Inteligența artificială după anul 2000. Apariția plăcilor video.....	38
1.5 AlexNet.....	39
1.6 Aplicații ale rețelelor neuronale în medicină.....	41
1.6.1 Diagnosticare.....	41
1.6.2 Analiză predictivă.....	41
1.6.3 Tratament personalizat.....	41
1.6.4 Monitorizarea pacientului de la distanță.....	42
Capitolul 2. Arhitecturi și tehnologii utilizate.....	43
2.1 CNN.....	43
2.1.1 Straturile convolutionale.....	43
2.1.2 Activare ReLU.....	46
2.1.3 MaxPooling.....	47
2.1.3 Straturi Perceptron.....	48
2.1.4 Controlul procesului de antrenare și combaterea ovefitting-ului.....	48
2.2 U-Net.....	51
2.2.1 Limitările segmentării manuale.....	52
2.2.2 Structura modelului.....	54
2.2.3 Implementarea arhitecturii.....	56
2.3 ResNet.....	58
2.3.1 Arhitectură ResNet.....	59
2.3.2 Implementarea arhitecturii.....	60
.....	63
2.4 VGG.....	63
Pentru că nu există prea multe particularități față de rețelele convoluționale obișnuite, definirea unui modul pentru implementarea rețelelor din clasa VGG nu va fi necesară.[VDCN].....	64
2.5. Clasificare.....	64
2.6. Tehnologii.....	65
Capitolul 3. Antrenarea rețelelor folosind imagini medicale.....	67
3.1 Leucemie.....	67
3.1.1. Set de date.....	69
3.1.2 Modelul pentru segmentare. Rezultate inițiale.....	70

3.1.3 Preprocesarea imaginilor. Rezultatele după preprocesare.....	74
3.1.4 Prima rețea pentru clasificarea imaginilor.....	78
3.1.5 Al doilea model pentru clasificare.....	81
.....	81
3.1.6 Compararea activărilor cu filtrele de segmentare.....	83
3.2 Tumori cerebrale.....	84
3.2.1. Setul de date folosit pentru segmentare.....	86
3.2.2 Preprocesarea imaginilor pentru setul de segmentare.....	88
3.2.3 Segmentarea imaginilor.....	88
3.2.4 Setul de date pentru clasificare.....	102
3.2.5 Clasificarea imaginilor.....	102
3.3 COVID-19.....	111
3.3.1 Setul de date.....	111
3.4 OCT Retinal.....	116
Concluzii.....	120

## **Introducere**

Ştiinţă datelor este un domeniu destul de nou ,însă care deja a devenit important pentru mai multe industrii şi domenii cum ar fi sănătate,economie,securitate,afaceri etc.

În ultimii inovaţiile aduse în tehnologie, cum ar fi apariţia de noi metode de stocare a datelor , apariţia reţelelor de obiecte şi aparate inteligente (Internet of Things) şi dezvoltarea de modele ce utilizează algoritmi de machine learning, dar şi creşterea numărului de utilizatori de internet a dus la generarea de volume mari de date. Aceste volume mari de date nestructurate pot fi utilizate pentru a obţine informaţii valoroase şi pentru a contribui la luarea decizilor in diverse domenii. Analiza acestor date poate identifica tipare, corelaţii şi tendinţe care ar fi dificil de observat prin intermediul unor metode tradiţionale.

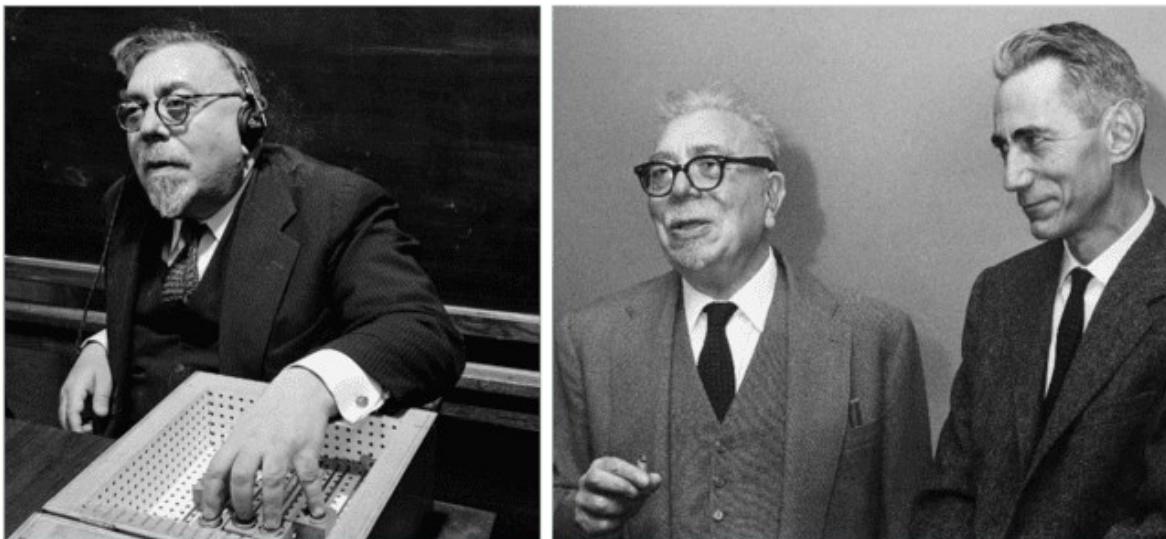
Pasiunea mea pentru matematică şi informatică m-a determinat să aleg acesta tema, deoarece ştiinţa datelor le combină pe amândouă, de asemenea am preferat aplicaţiile acesteia în domeniul medical ,deoarece aplicaţiile metodelor de analiza de date în domeniul medical reprezintă una dintre cele mai bune utilizări ale acestora, datorită faptului că medicii şi cercetătorii din domeniul medical au de a face zilnic cu un volum mare de date şi pentru că inovaţiile în domeniul medical pot contribui la îmbunătăţirea bunăstării generale a populaţiei.

# **Capitolul 1. Fundamente ale rețelelor neuronale**

## **1.1 Pionierii inteligenței artificiale. Lucrări ce au stat la baza apariției inteligenței artificiale**

Înainte de a vorbi despre inteligența artificială trebuie să aducem în discuție lucrările și persoanele care au pus bazele cunoașterii și tehnologiilor ce au permis dezvoltarea acestui domeniu. Norbert Wiener, Claude Shannon și Alan Turing sunt 3 dintre pionierii inteligenței artificiale. Contribuțiile lor au fost atât indirecte, prin elaborarea unor teorii fundamentale în matematică, logică, cibernetică și teoria informației, cât și directe, prin implicarea în proiecte și dezbateri ce vizau posibilitatea construirii unor „mașini capabile să gândească”.

### **1.1.1 Norbert Wiener**



*Figure 1: Norbert Wiener(1), Norbert Wiener și Claude Shannon(2)*

Norbert Wiener a fost profesor la MIT în perioada de după Primul Război Mondial. În acea perioadă, principalul său domeniu de activitate era matematica aplicată, Wiener ocupându-se de studiul mișcării browniene, al integralelor Fourier (utilizate în transmiterea și interceptarea mesajelor pe distanțe mari) și al ecuațiilor diferențiale, pentru care reușește să găsească o soluție parțială. Aceasta și-a

extins activitatea în anii următori, dorind să rezolve probleme din domeniul neurofiziologiei și al geneticii, pentru care primește premiul Boucher în 1933.

În cel de-Al Doilea Război Mondial, este implicat într-un proiect ce avea ca scop dirijarea automată a tirului artileriei. Wiener concepe modele matematice care pot prezice traectoria avioanelor și pot ajusta tirul în funcție de modificările acesteia. El a fost prima persoană care a scris despre ideea de „buclă de feedback” – un proces prin care modelele se ajustează în timp real odată cu apariția unor informații noi. Wiener și-a dat seama că „bucla de feedback” este specifică atât animalelor și oamenilor, cât și sistemelor mecanice. Realizarea că atât ființele vii, cât și sistemele mecanice se pot adapta la mediu prin expunerea la informații noi a dus la întemeierea ciberneticii în 1943. The Mathematician Forced to become a genius

Conform definiției lui Norbert Wiener, cibernetica este „studiu comunicării și al controlului în cadrul sistemelor biologice și mecanice”, unde feedbackul este componenta principală. În cazul corpului uman, care reprezintă un sistem biologic, putem observa cum creierul procesează stimulii primiți de la organele de simț și reglează comportamentul corpului pe baza acestora (de exemplu, reglarea bătăilor inimii și a respirației în cazul efortului sau adaptarea ochilor la întuneric). În cazul unui termostat, care reprezintă un sistem mecanic, putem observa cum acesta înregistrează constant temperatura camerei pentru a regla aerul emis și a menține temperatura constantă. Lucrările sale, ce puneau accentul pe asemănarea dintre om și mașină, au dus la apariția studiilor ce urmăreau crearea unei mașini capabile să imite comportamentul uman. The Mathematician Forced to become a genius

Ideeia de „buclă de feedback” a stat la baza formării procesului de antrenare a modelelor de învățare profundă, prin ajustarea continuă a parametrilor modelului pe baza diferențelor dintre informația nouă și cea prevăzută. De asemenea, modul în care Norbert Wiener a descris procesul prin care neuronii formează mai multe conexiuni pentru a se adapta la informații noi a influențat modul în care reprezentăm neuronii într-o rețea neuronală artificială. Totuși, s-a ales o abordare mai simplă pentru reprezentarea conexiunilor dintre neuroni, folosind greutăți (numere) care exprimă gradul de conexiune dintre doi neuroni. Acestea au valori pozitive mai mari decât zero, iar valoarea 0 semnifică absența unei conexiuni (activarea unui neuron nu influențează activarea celuilalt). Similar cu rețelele neuronale umane, nu există un grad maxim de conexiune, iar neuronii artificiali sunt organizați în straturi: neuronii aflați în același strat nu pot forma conexiuni între ei sau cu neuronii din straturi care nu sunt adiacente, dar aceste relații se stabilesc indirect prin conexiunile dintre straturi. Deși modelul

matematic simplifică structura creierului, rețelele artificiale sunt capabile să învețe concepte complexe, precum recunoașterea facială sau emoțională, și permit ajustarea conexiunilor într-un mod eficient. The Mathematician Forced to become a genius From Cybernetics to AI: the pioneering work of Norbert Wiener



Figure 2: Ştefan Odobleja

Doresc să menționez și contribuția lui Ştefan Odobleja, care, prin lucrarea sa din 1938 intitulată „Psihologia Consonantistă”, a pus bazele ciberneticii, fiind considerat un precursor al acesteia. El a propus ideea că „gândirea umană funcționează pe baza unui mecanism de feedback”, cu mult înaintea lui Norbert Wiener. Ulterior, Wiener a preluat aceste idei și le-a aplicat sistemelor mecanice. Lucrarea lui Odobleja nu a fost bine primită la momentul apariției, deoarece era greu de înțeles și părea imposibil de aplicat în realitate, fiind una cu caracter pur teoretic. Norbert Wiener, deținând cunoștințe din multiple domenii, a înțeles valoarea lucrării lui Odobleja și i-a găsit aplicații în sistemele de comunicare. Deși Wiener este recunoscut ca fondator al ciberneticii și a avut contribuții majore în domeniul inteligenței artificiale, realizările sale nu ar fi fost posibile fără descoperirea lucrării lui Ştefan Odobleja.

### 1.1.2 Claude Shannon

Ingineria electrică de dinaintea celui de-al Doilea Război Mondial era fundamental diferită de cea pe care o cunoaștem astăzi și nu îngloba principiile matematice și metodele de proiectare a circuitelor electrice folosite în prezent. Inginerii proiectau circuite specifice unei singure sarcini, care nu puteau fi reconfigurate pentru a fi utilizate în alte scopuri. ROThe Man Who Should Be As Famous As Einstein

Datorită lipsei unei metodologii universale de proiectare, circuitele electrice nu puteau fi modificate cu ușurință, fiind necesară conceperea unui nou circuit de fiecare dată când se dorea simplificarea sau optimizarea unui sistem existent.

Situatia s-a schimbat odată cu contribuția lui Claude Shannon, un absolvent al Universității din Michigan, care studiase în paralel ingineria electrică și matematica. Inspirat de lucrările lui George Boole, Shannon a realizat că algebra booleană poate fi folosită pentru a descrie funcționarea circuitelor electrice.

Claude ShannonROThe Man Who Should Be As Famous As Einstein

Algebra booleană se aplică acestor circuite deoarece ele au doar două stări posibile: închis sau deschis. Prin lucrarea sa de disertație, intitulată „*O analiză simbolică a rellelor și a circuitelor electrice*”(A Symbolic Analysis of Relay and Switching Circuits), Shannon a demonstrat că algebra booleană poate fi utilizată atât pentru proiectarea, cât și pentru analiza circuitelor electrice. El a conceput următoarele postulate pentru reprezentarea circuitelor folosind algebra booleană: „

0 – închis

1 – deschis

+ - serie

\* - paralel

1.

a.  $0 * 0 = 0$

Un circuit închis aflat în paralel cu alt circuit închis formează un circuit închis

b.  $1 + 1 = 1$

Un circuit deschis aflat în serie cu un alt circuit deschis formează un circuit deschis.

2.

a.  $1 + 0 = 1$

Un circuit închis aflat în paralel cu alt circuit închis formează un circuit închis

b.  $1 + 1 = 1$

Un circuit deschis aflat în serie cu un alt circuit deschis formează un circuit deschis.

4.

a.  $1 + 0 = 0 + 1 = 1$

Un circuit deschis aflat în serie cu unui închis formează un circuit deschis

b.  $0 * 1 = 1 * 0 = 0$

Un circuit închis aflat în paralel cu un circuit deschis formează un circuit închis

5.

a.  $0 + 0 = 0$

Un circuit închis aflat în serie cu alt circuit închis formează un circuit închis

b.  $1 * 1 = 1$

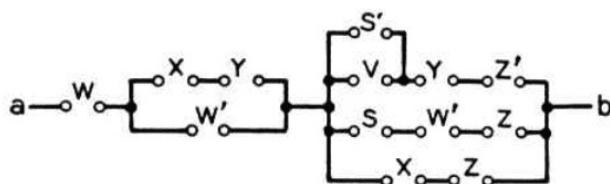
Un circuit deschis aflat în paralel cu un alt circuit deschis formează un circuit deschis.

6.

Un circuit poate fi doar închis ( $X=0$ ) sau deschis ( $X=1$ )

"

Mai jos au fost utilizate postulatele pentru a transforma circuitul din Figura 3 într-o ecuație algebrică (Figura 4).



$$\begin{aligned}
 X_{ab} &= W + W'(X + Y) + (X + Z)(S + W' + Z)(Z' + Y + S'V) \\
 &= W + X + Y + (X + Z)(S + 1 + Z)(Z' + Y + S'V) \\
 &= W + X + Y + Z(Z' + S'V).
 \end{aligned}$$

Figure 3: Circuit Inițial

Lucrarea lui Claude Shannon a stat la baza dezvoltării porților logice utilizate astăzi în toate dispozitivele moderne. Descoperirile lui Shannon din prima lucrare l-au ajutat să pună bazele tehnologiei informației în „O teorie matematică a comunicației”. Cea de-a doua lucrare a fost însă concepută mult mai târziu. Shannon credea că poate aduce contribuții și în alte domenii, aşa că, în următorii ani, nu a contribuit activ la dezvoltarea circuitelor electrice, însă lucrarea sa a dus la schimbarea modului în care acestea sunt concepute. Unul dintre domeniile în care Shannon a activat a fost genetica, unde a formulat o ecuație capabilă să prezică evoluția în timp a frecvenței unei alele într-o populație. ROThe Man Who Should Be As Famous As Einstein

În ciuda realizărilor sale, Shannon considera că nu ar putea aduce contribuții majore în domeniul geneticii. Drept urmare, a apelat la mentorul său de atunci, Vannevar Bush, pentru a-i solicita îndrumare privind direcția profesională. Bush l-a sfătuit să-și continue cercetările în domeniul matematicii aplicate pentru că matematica pură nu i s-ar potrivii. Din fericire, compania Bell Labs,

compania care avea să devină un pilon al inovația tehnologică prin crearea primului tranzistorul, iar mai apoi prin crearea sistemul de operare Unix și a limbajelor de programare B, C și C++, a remarcat potențialul lui Shannon și i-a oferit un post în cadrul departamentului de matematică aplicată.ROThe Man Who Should Be As Famous As Einstein Bell LabsA Mind At Play

În perioada în care Shannon a lucrat pentru Bell Labs, a avut contribuții importante în domeniul criptografiei. El a creat sistemul SIGSALY, un sistem de criptare ce folosea aritmetică modulară pentru a implementa o metodă de criptare end-to-end, care masca transmisia vocală, făcând sunetele să pară zgomot alb pentru oricine încerca să intercepteze transmisia fără a deține cheia de decriptare. Sistemul a fost utilizat pentru transmiterea de mesaje de către Aliați în cel de-al Doilea Război Mondial.Bell LabsClaude Shannon

Cea mai mare realizare a sa din acea perioadă rămâne, totuși, fundamentarea tehnologiei informației. Shannon a fost impresionat de ideea lui Alan Turing de a construi o „mașină universală” capabilă să rezolve orice problemă matematică ce poate fi descrisă printr-un algoritm. Shannon știa că proiectarea unei astfel de mașinării ar necesita o metodă universală de reprezentare a informației. „Ce este informația?”, „Cum poate fi reprezentată?” sunt două dintre întrebările la care Shannon trebuia să poată să răspundă, iar o parte din răspunsuri se aflau deja în lucrarea sa de disertație, „*O analiză simbolică a releeelor și a circuitelor electrice*”(A Symbolic Analysis of Relay and Switching Circuits) .

În 1948, Shannon publică lucrarea „O teorie matematică a informației”(A Mathematical Theory of Communication), în care propune o metodă universală de reprezentare a informației. Informația, indiferent de forma în care se află (text, audio, imagini etc.), poate fi reprezentată prin intermediul cifrelor binare, iar bitul (**binary digit**) devine principala unitate de măsură a informației. Înainte de Shannon, vocea era transmisă prin semnale electrice ce imitau undele sonore, ceea ce ducea la degradarea semnalului pe distanțe mari, distorsionarea sunetului și întreruperea transmisiei. Transmiterea prin biți devine astfel metoda principală de comunicare, bitul fiind mai ușor de convertit în semnale electrice decât sunetele întregi. De asemenea, bitul permite refacerea mesajelor în cazul apariției erorilor de transmisie .ROThe Man Who Should Be As Famous As Einstein

Shannon propune metode de corectare a datelor prin introducerea intenționată a redundanței în mesajele codificate, cum ar fi adăugarea unor biți de paritate pentru a verifica corectitudinea mesajului. Lucrarea sa a dus la crearea unei metode universale de reprezentare și transmitere a datelor și a

influențat decisiv modul în care a fost dezvoltată tehnologia în decenile următoare. ROThe Man Who Should Be As Famous As Einstein A Mathematical Theory of Communication

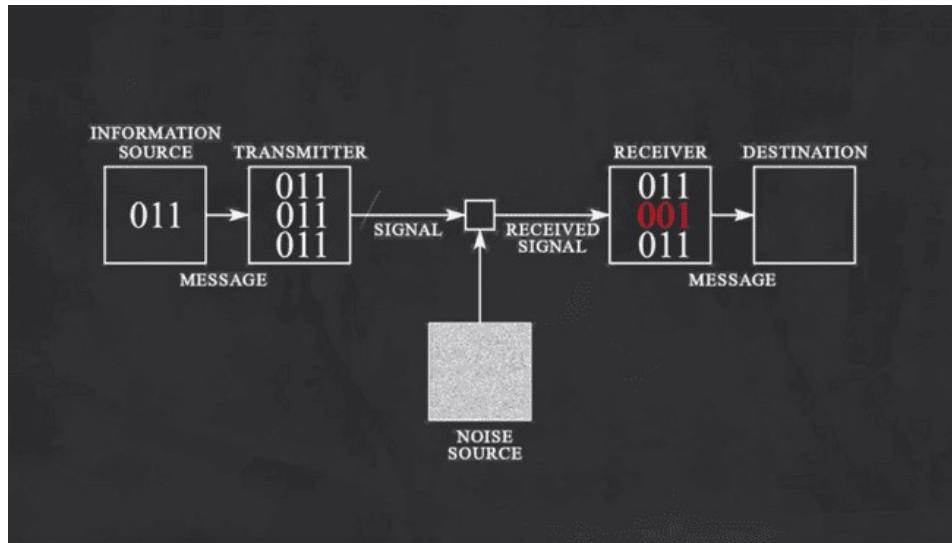


Figure 5: Detectarea și corectarea erorilor prin biți de paritate

Lucrările lui Shannon au contribuit indirect la apariția inteligenței artificiale prin crearea unei metode de reprezentare a informației din lumea reală la nivelul circuitelor electrice, dar Shannon a avut și contribuții directe. El a proiectat un șoarece electric capabil să găsească ieșirea din orice labirint. Adevaratul dispozitiv se afla, de fapt, sub labirint și era compus din mai multe părți care reprezentau direcția în care șoarecele trebuia să meargă. Împărțirea labirintului în sectoare de dimensiuni egale a fost o modalitate de a reprezenta labirintul prin intermediul unei matrice, în care valorile sunt reprezentate de direcții. Șoarecele era dirijat de aparat printr-un magnet aflat în interiorul său. Atunci când șoarecele se lovea de un zid, componenta aflată sub șoarece își schimba direcția. Prin această ajustare a componentelor, aparatul putea identifica și memora drumul corect. Aparatul poate fi observat în Figura 6. ROThe Man Who Should Be As Famous As Einstein

În 1949, Shannon a mai publicat o lucrare care detalia câteva metode prin care un calculator poate fi antrenat pentru a juca șah. Metodele de învățare utilizate de Shannon sunt, totuși, diferite de cele utilizate în rețelele neuronale, fiind doar abordări euristice, însă au fost printre primele experimente care aveau ca scop antrenarea unui calculator pentru rezolvarea unei probleme ce implică gândirea umană.ROThe Man Who Should Be As Famous As Einstein

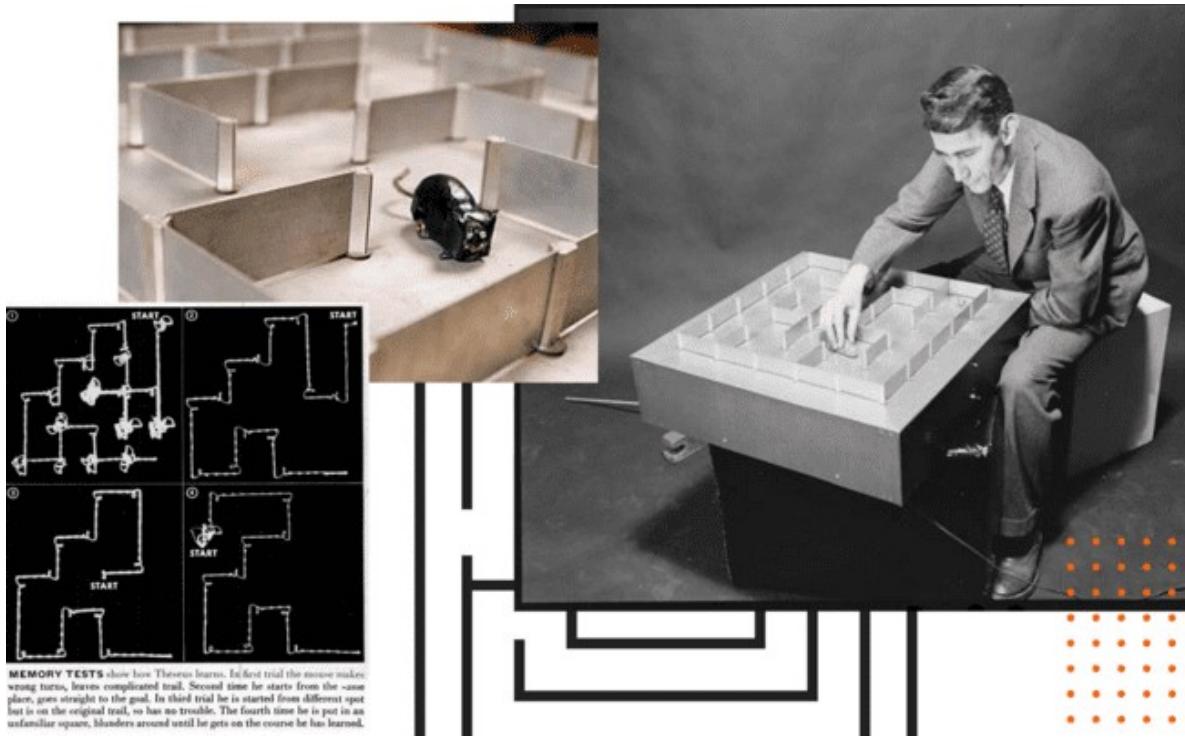


Figure 6: Theseus, şoarecele electric

### 1.1.3 Alan Turing

Problema deciziei a apărut odată cu inventarea primului aparat de calcul, în anul 1624. Aparatul, creat de Gottfried Wilhelm Leibniz, era capabil să efectueze operații de adunare, scădere, înmulțire și împărțire folosind numere cu până la șase zecimale. În ciuda perioadei în care a fost realizat, dispozitivul conceput de Leibniz oferea funcționalități regăsite și în calculatoarele moderne. În interiorul său se afla un registru care memora calculele anterioare. Pe lângă unitatea de stocare, aparatul dispunea și de un mecanism de detectare a overflow-ului. Capacitatea sa de reprezentare a numerelor era limitată la șase cifre, iar atunci când rezultatul unei operații depășea această limită, un clopoțel amplasat pe marginea dispozitivului suna pentru a semnala eroarea. Mechanical Calculator

După construirea acestui aparat, Leibniz și-a propus să creeze un dispozitiv capabil să determine valoarea de adevăr a oricărei propoziții matematice. Totuși, nu a reușit să proiecteze un astfel de aparat. Experimentul său i-a determinat pe intelectualii vremii să se întrebe dacă o astfel de mașină ar putea fi, într-adevăr, realizată și, ulterior, să încerce să demonstreze posibilitatea existenței sale. Această problemă a fost denumită ulterior „Problema deciziei”. The Decision Problem

Trei secole mai târziu, în anul 1928, problema revine în atenția comunității științifice. David Hilbert și Wilhelm Ackermann publică propria lor formulare a problemei deciziei (în germană: *Entscheidungsproblem*). Cei doi matematicieni căutau o demonstrație formală care să stabilească dacă este posibilă construirea unui algoritm sau a unei mașini capabile să decidă valoarea de adevăr a oricărei propoziții matematice.

Încercând să rezolve această problemă, Alan Turing pune bazele funcționării calculatoarelor și a programelor care rulează pe acestea. Pentru rezolvarea problemei propuse de David Hilbert și Wilhelm Ackermann, Turing concepe un dispozitiv teoretic capabil să determine valoarea de adevăr a oricărei propoziții matematice. Aparatul lui Turing, denumit „mașină universală”, era compus dintr-o bandă și un

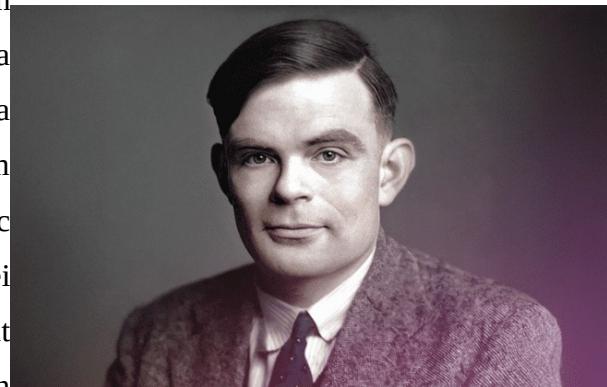


Figure 7: Alan Turing

scanner, acesta din urmă având capacitatea de a verifica conținutul unei căsuțe și de a-l modifica înainte de a trece la următoarea. ROAlan Turing - Celebrating the life of a genius ROTuring Machines - How Computer Science Was Created By Accident The Curch-Turing ThesisON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

Pe bandă se aflau informațiile despre problemă, codificate, de regulă sub formă de cifre binare, iar scannerul conținea algoritmul și stările necesare evaluării. Scanner-ul parcurgea banda în ambele direcții și o modifica până la apariția unui simbol care marca finalizarea procesului și indica dacă axioma putea fi sau nu verificată. ROTuring Machines - How Computer Science Was Created By Accident ROTuring Machines Explained - Computerphile

Folosind acest dispozitiv teoretic, Turing demonstrează că nu este posibilă crearea unui aparat sau algoritm capabil să determine valoarea de adevăr a oricărei axiome. Mai mult decât atât, el evidențiază o problemă fundamentală: în cazul anumitor axiome, nu putem să știm dacă acestea pot fi dovedite sau nu, ceea ce determină aparatul să ruleze la nesfârșit („Problemă a opririi” — *The Halting Problem*). ROTuring Machines - How Computer Science Was Created By Accident ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

Dorind să testeze limitele aparatului său, Turing propune o situație ipotetică în care urmărește să determine dacă este posibilă conceperea unui program capabil să stabilească dacă un alt program va intra într-o buclă infinită. Programul H este un program ipotetic care poate face această analiză, iar datele de ieșire (output-ul) ale lui H devine datele de intrare (input-ul) pentru un alt doilea program, Programul A. Dacă H spune că programul analizat va rula la infinit, Programul A se oprește. Dacă H spune că programul se va opri, atunci Programul A intră într-o buclă infinită.

**ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM**

Însă atunci când programul analizat de H este Programul A, se produce un paradox: dacă H spune că A se oprește, atunci A va rula la infinit, iar dacă H spune că A va rula la infinit, atunci A se va opri. Această contradicție arată că nu putem crea un program care să determine cu certitudine dacă un alt program (oricare ar fi acesta) se va opri sau nu la un moment dat, figura 8 reprezintă schema procesului.

**ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM**

Lucrările lui Turing nu au fost apreciate imediat, din cauza naturii lor abstracte, dar au oferit răspunsuri la întrebări fundamentale legate de limitele calculatoarelor și au crea şabloane pentru proiectarea lor și a limbajelor de programare.

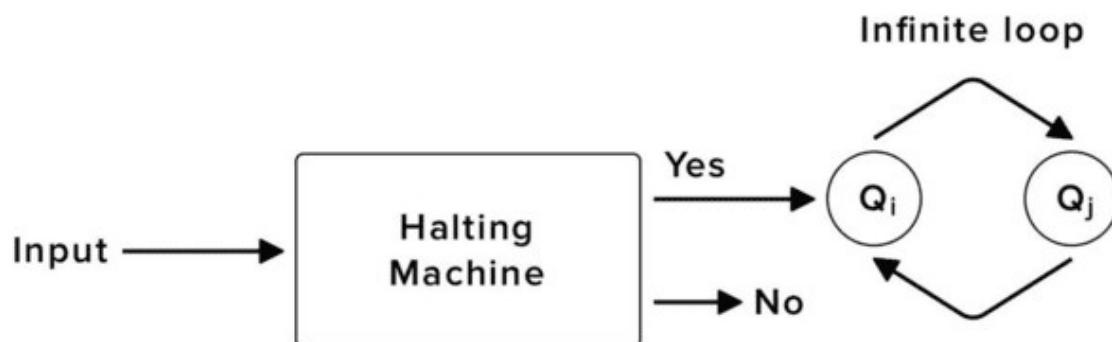


Figure 8: Paradoxul lui Turing

O altă contribuție a lui Alan Turing a fost descifrarea mesajelor generate de aparatul Enigma, folosit de Germania în cel de-al Doilea Război Mondial pentru a cripta mesajele transmise între

diviziile armatei. Aparatul era compus dintr-o tastatură și un panou cu led-uri: atunci când o literă era tastată, se aprindea un led aflat pe panou ce corespunde unei alte litere.

În interiorul aparatului se aflau trei rotoare care schimbau permutarea literelor la fiecare apăsare de tastă, generând un cod de criptare diferit pentru fiecare literă introdusă. În plus, aparatul avea și porturi ce puteau fi interconectate pentru a amesteca manual mesajul înainte și după trecerea semnalului prin rotoare, făcând decriptarea mesajelor mai dificilă.

Cum funcționa Mașina Enigma?

Pentru decriptare era folosit același aparat, configurat identic cu cel folosit pentru criptare. În fiecare zi, era stabilit un set diferit de rotoare și o configurație diferită pentru conexiunile dintre porturi. După tastarea unei litere, operatorul nota pe o foaie litera care se aprindea pe panou, iar apoi mesajul era transmis mai departe. Persoana care primea mesajul introducea literele criptate în aparat, iar ledurile care se aprinseau indicau literele mesajului decriptat.

Cum funcționa Mașina Enigma?

În ciuda complexității metodei de criptare, designul aparatului Enigma prezenta un defect major: în urma amestecării, nicio literă nu putea avea ca și corespondent aceeași literă, fiind posibilă decriptarea mesajelor prin excludere.

ROFlaw in the Enigma Code - Numberphile

Cum funcționa Mașina Enigma?

După ce au descoperit acest defect, echipa care se ocupa de decriptarea mesajelor, din care făcea parte și Alan Turing, a construit un dispozitiv capabil să decripteze mesajele transmise folosind aparatul Enigma în doar 15 minute. Dispozitivul, denumit Bombe, a schimbat soarta celui de-al Doilea Război Mondial, oferindu-le Aliaților o metodă de a intercepta mesajele Axei.

Bombe



Figure 9: Aparatul Enigma

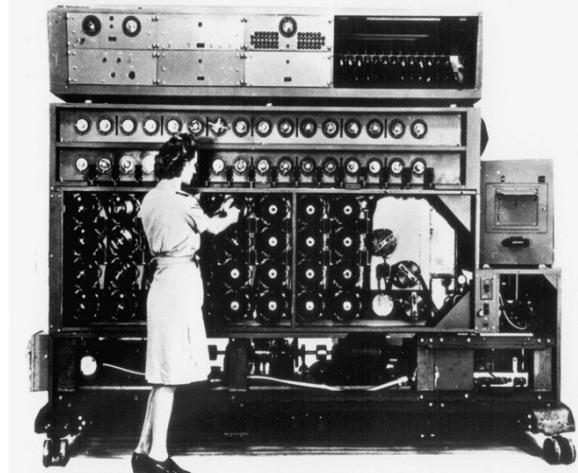


Figure 10: Bombe

Alan Turing a contribuit direct la apariția inteligenței artificiale prin lucrarea sa din 1950, „Mașini de calcul și inteligență”(COMPUTING MACHINERY AND INTELLIGENCE), în care abordează ideea unei mașini capabile să gândească. În stilul său clasic, Turing inventează o situație ipotetică în care, în fața unui interogator, se află o persoană reală și o mașină. Scopul interogatorului este de a pune întrebări pentru a afla care este persoana reală. Conform definiției lui Alan Turing, o mașină cu adevărat capabilă să gândească ar fi una care ar putea induce în eroare interogatorul, făcându-l să credă că ea este persoana reală. Situația prezentată anterior poartă denumirea de test Turing și a dus la apariția programelor ce detectează utilizatorii falși. Într-un mod similar cu Claude Shannon, Turing a dezvoltat un algoritm de șah denumit Turochamp.Turochamp

## 1.2 Perceptron

Conceptul de rețea neuronală artificială își are originile în anul 1943, odată cu publicarea lucrării „*Un calcul logic al ideilor imanente în activitatea nervoasă*”(A LOGICAL CALCULUS OF THE IDEAS IMMANENT INNERVOUS ACTIVITY), de către neurofiziologul Warren McCulloch și logicianul Walter Pitts. Inspirați de scările lui Alan Turing, Walter Pitts și Warren McCulloch au conceput primul model matematic de reprezentare al neuronilor umani ce putea fi aplicat circuitelor electrice. Modelul propus de Walter Pitts era inspirat de modul în care funcționează neuronii umani și are la bază principiul „totul sau nimic”. Acest principiu afirmă că neuronul uman se comportă asemenea unui intrerupător: atunci când stimulul este suficient de puternic, impulsul electric generat este maxim;

În caz contrar, neuronul nu produce niciun impuls electric. Walter Pitts a modelat neuronul ca pe o funcție liniară care primește mai multe input-uri ca parametri, iar stimulul final dat de suma ponderată a input-urilor (figura 11). Dacă stimulul rezultat depășea un prag stabilit anterior, se obținea starea 1, indicând că stimulul a fost suficient de puternic pentru a produce activarea neuronului; în caz contrar, se obținea starea 0, ceea ce indică faptul că neuronul nu a fost activat.

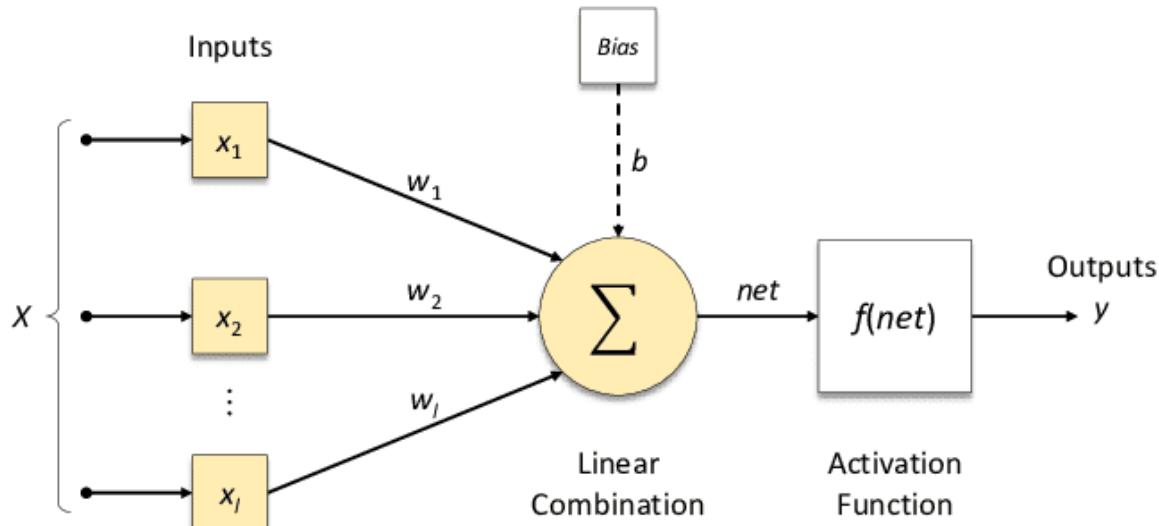


Figure 11: Neuron Pitts-McCulloch

În ciuda potențialului pe care îl avea proiectul lui Walter Pitts și Warren McCulloch, acesta nu a putut fi continuat. Norbert Wiener, care contribuise la realizarea proiectului, și-a retras sprijinul în urma unei dispute cu McCulloch, iar pentru mult timp ideea de neuron artificial a rămas una pur teoretică.

Primul Perceptron a fost creat de Frank Rosenblatt în 1957 pe baza modelului Pitts-McCulloch. La vremea aceea limbajele de programare nu deveniseră încă populare, prima versiune de Fortran fiind lansată cu doar 3 ani înainte. Rețeaua lui Frank Rosenblatt nu era program, ci un aparat ce implementa algoritmul lui Pitts-McCulloch la nivel de circuit. Rosenblatt a folosit aparatul pentru clasificarea imaginilor alb-negru, de exemplu diferențierea patrulaterelor convexe de cele concave. Datorită naturii rețelei, erau acceptate doar input-uri binare. The First Neural Networks Principles of Neurodynamics

ROPerceptrons: The First Trainable Neural Networks | Teaching Computers to Learn, Part 3 explică modul de antrenare al rețelei astfel:

1. Se alege o valoare pentru rata de antrenare a modelului

2. Se apelează funcția neuronului pentru vectorul de stimului
3. Se verifică acuratețea rezultatului
4. Dacă predicția e mai mare, se scad greutățile cu rata de antrenare pentru input-urile cu valoarea 1, altfel, acestea vor crește.
5. Se reiterează setul de antrenament până când toate instanțele au fost clasificate corect sau până când s-a atins la numărul maxim de iterații.

Ex:

$$th = 0 - \text{prag} (<=0 \Rightarrow 0, >0 \Rightarrow 1)$$

$$lr = 1 \text{ (rată de antrenare)}$$

Pentru vectorul  $[0,1,0,1,1]$  și valoarea reală 0, greutățile curente  $[-1,2,3,1,-2]$

$$\Rightarrow s = 0 * -1 + 1 * 2 + 3 * 0 + 1 * 1 + (-2) * 1 = 0 + 2 + 0 + 1 - 2 = 1 > 0 \Rightarrow 1 \neq 0$$

$1 > 0 \Rightarrow$  se vor scădea greutățile pt input-urile cu valoarea 1

$0 -1$ , nu se ajutează

$$1 2 \Rightarrow 2 - 1 = 1$$

$0 3$ , nu se ajutează

$$1 1 \Rightarrow 1 - 1 = 0$$

$$1 -2 \Rightarrow -2 - 1 = -3$$

greutăți ajustate  $[-1,1,3,0,-3]$

Capacitățile perceptronului au fost exagerate de atât de presă, cât Frank Rosenblatt însuși, care îl prezenta ca fiind „mașina care poate gândi ca un om” și susținea că, în timp, ar putea rezolva orice problemă ce necesită gândirea umană. Perceptronul, dezvoltat în anii 1950 și finanțat de Marina americană, a fost unul dintre primele modele de rețea neuronală artificială și a trezit un val de entuziasm atât în rândul cercetătorilor, cât și al publicului larg. Cu toate acestea, în 1969, Marvin Minsky și Seymour Papert au publicat lucrarea „Perceptrons”(Perceptrons. An Introduction to Computational Geometry ), în care au evidențiat limitările perceptronului. Ei au arătat că perceptronul

nu poate rezolva probleme care nu sunt linear separabile, oferind ca exemplu funcția logică XOR (sau exclusiv), care nu putea fi învățată de un perceptron simplu.

Această critică, combinată cu limitările hardware ale vremii și lipsa unor metode eficiente de antrenare a rețelelor cu mai multe straturi, a dus la un declin al interesului față de rețelele neuronale și inteligența artificială în general. Această perioadă, cunoscută drept prima „iarnă a inteligenței artificiale” (*AI Winter*), a marcat o stagnare a cercetărilor și investițiilor în domeniu timp de mai bine de un deceniu. Odată cu dezvoltarea calculatoarelor moderne și descoperirea algoritmului Backpropagation în anii 1980, interesul pentru inteligența artificială a crescut din nou.

### 1.3 Backpropagation

Algoritmul Backpropagation, în forma cunoscută astăzi, a fost popularizat abia în 1986 de David E. Rumelhart, Geoffrey Hinton și Ronald J. Williams, printr-o analiză experimentală a modului prin care Backpropagation poate fi utilizat pentru a antrena rețele neuronale cu mai multe straturi. Adoptarea algoritmului Backpropagation a fost facilitată de schimbarea modului de reprezentare a datelor. Pentru că datele din lumea reală sunt valori cantitative și continue (valori de pixeli, temperaturi, amplitudini de sunete), s-a renunțat la principul de „totul sau nimic” inspirat de neuronii umani. Utilizarea datelor reale a extins aplicabilitatea rețelelor neuronale, dar erau încă limitate de natura lor lineară. David E. Rumelhart, Geoffrey Hinton și Ronald J. Williams au rezolvat problema linearității prin utilizarea funcțiilor de activare peste rezultatele inițiale, și au folosit ca exemplu problema XOR. Funcțiile de activare au rolul de a face neuronii artificiali să se comporte asemenea neuronilor umani. În acest document vor fi utilizate doar activările Sigmoid și ReLU (Rectified Linear Unit). O perioadă îndelungată de timp a fost folosită tangenta hiperbolică pentru a simula activările neuronilor umani, dar s-a renunțat la folosirea tangentei în favoarea metodelor amintite anterior. BackpropagationLearning representations by back-propagation errors

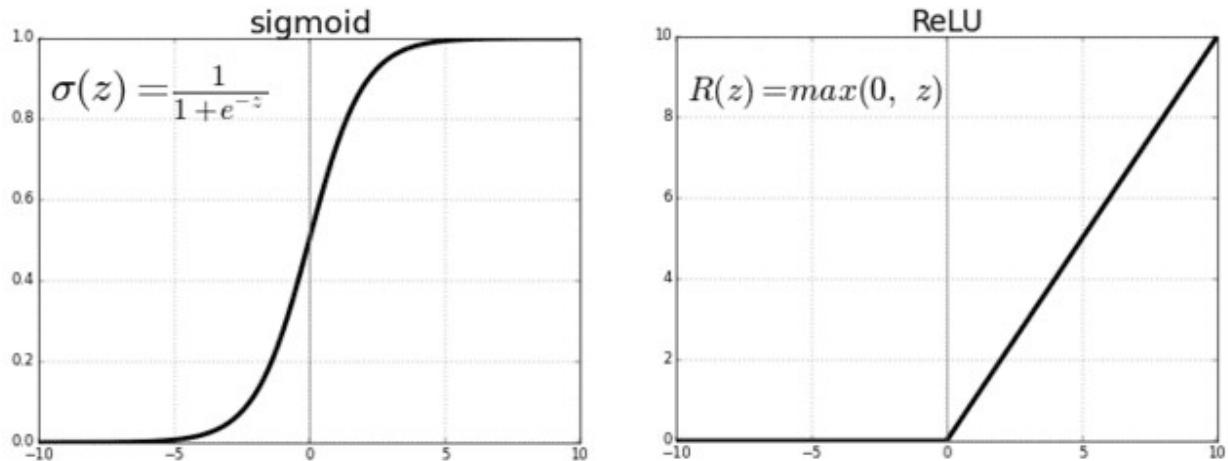


Figure 12: Funcții de activare

Table 1: Comparație ReLu-Sigmoid

Sigmoid	Relu
Restrânge rezultatul funcției liniare la intervalul [0,1]	Transformă valorile negative în 0.
Este o modalitate de a reprezenta modul în care neuronul răspunde la stimul	Neuronii nu pot răspunde negativ la stimul, pot doar să producă sau nu impulsuri electrice de diferite intensități. De aceea funcția ReLU este utilizată pentru a impune această regulă.
0 – neuronul nu răspunde la stimul	A devenit cea mai utilizată funcție de activare după
1 – neuronul răspunde la stimul și produce impulsul maximul	ce fost folosită în 2012 în rețeaua AlexNet.
De obicei este folosit pentru clasificare binară în stratul de output, dar poate fi folosit și pentru activările straturilor anterioare.	

Backpropagation este un algoritm de ajustare al parametrilor rețelelor neuronale baza calculului diferențial. Metoda presupune pasarea erorilor generate la nivelul straturilor superioare către cele inferioare. A fost prima metodă de antrenare a rețelelor MLP (Multi-Layer Perceptron) și RNN (Recurrent Neural Network). Ambele arhitecturi sunt mult mai simple decât cele utilizate în zilele

noastre, dar au fost revoluționare în perioada aceea și puteau fi folosite pentru recunoașterea cifrelor și a literelor, recunoașterea vocală și crearea mașinilor ce se pot conduce singure.

Cele două concepte matematice ce stau la baza algoritmului sunt gradientul unei funcții și derivarea în lanț. Backpropagation

### 1.3.1 Gradientul

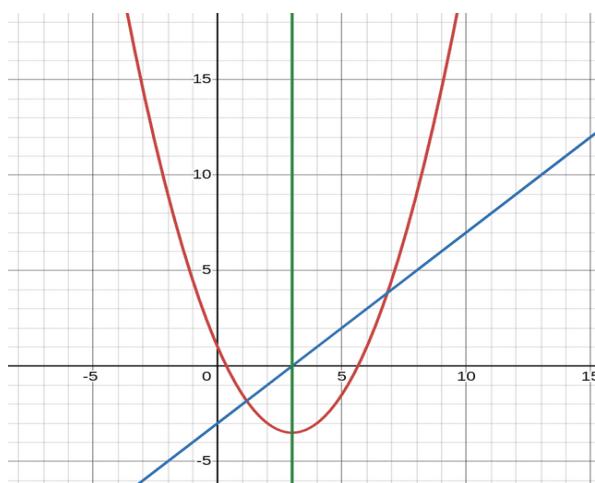
Pentru a înțelege ideea de gradient și pentru a putea implementa algoritmul Gradient Descent, trebuie să pornim de la definiția unei derivate.

*Fie  $f : (a, b) \rightarrow R$  și  $x_0$  un punct ce aparține intervalului  $(a, b)$ , atunci derivata lui  $f$  în  $x_0$  este:*

$$\text{Derivata} \\ f'(x_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{(x_0 + h) - x_0}$$

Derivata unei funcții reprezintă rata cu care funcției își modifică valoarea atunci când se modifică argumentul acesteia. Cu alte cuvinte, derivata descrie modul în care funcția variază de-a lungul axei Ox. Putem spune că derivata în punctul  $x_0$  reprezintă panta/tangenta la graficul funcției în acel punct și descrie direcția și înclinația acesteia. Astfel, derivata funcției poate fi comparată cu o „busolă”: polii reprezintă sensurile axei Ox, iar acul busolei este dat de semnul derivatei, indicând direcția în care funcția crește sau scade. Valoarea absolută a derivatei arată în schimb cât de abruptă este panta. Conform teoremei lui Fermat, un punct de extrem local (minim sau maxim) apare acolo unde derivata este zero (cu condiția ca derivata să existe în acel punct).

Pentru funcția:  $f : R \rightarrow R$ ,  $f(x) = \frac{x^2}{2} - 3x + 1$ , cu derivata  $f'(x) = x - 3$ , graficul funcției este:



Figure

13: Grafic Derivată

Pe grafic se pot observa proprietățile menționate anterior: derivata are valori negative pe intervalul în care funcția este strict descrescătoare (se îndepărtează de maxim), iar pe intervalul în care funcția este strict crescătoare, derivata are valori pozitive (se apropiște de maxim). Punctul în care derivata este 0 corespunde minimului global al funcției inițiale.

În cazul de față putem determina ușor un minim sau un maxim local/global, egalând derivata cu 0 ( $x_0 - 3 = 0 \Rightarrow x_0 = 3$ ). Dar ce facem atunci când nu putem determina valorile pentru un minim sau un maxim local?

Să luăm ca exemplu funcția  $g(x) = e^{\frac{-x^2}{10}} \sin(2x) + \frac{x^3}{20} - \frac{x}{2}$ , cu derivata

$$g'(x) = \frac{-4xe^{\frac{-x^2}{10}}\sin(2x) + 3x^2 - 10}{20} + 2e^{\frac{-x^2}{10}}\cos(2x).$$

Cu siguranță nu am putea determina un minim local prin egalarea derivatei cu 0. Dar aşa cum am explicat mai devreme, derivata se asemănă cu o „busolă”. Ne putem folosi de proprietățile derivatei pentru a găsi un minim local astfel:

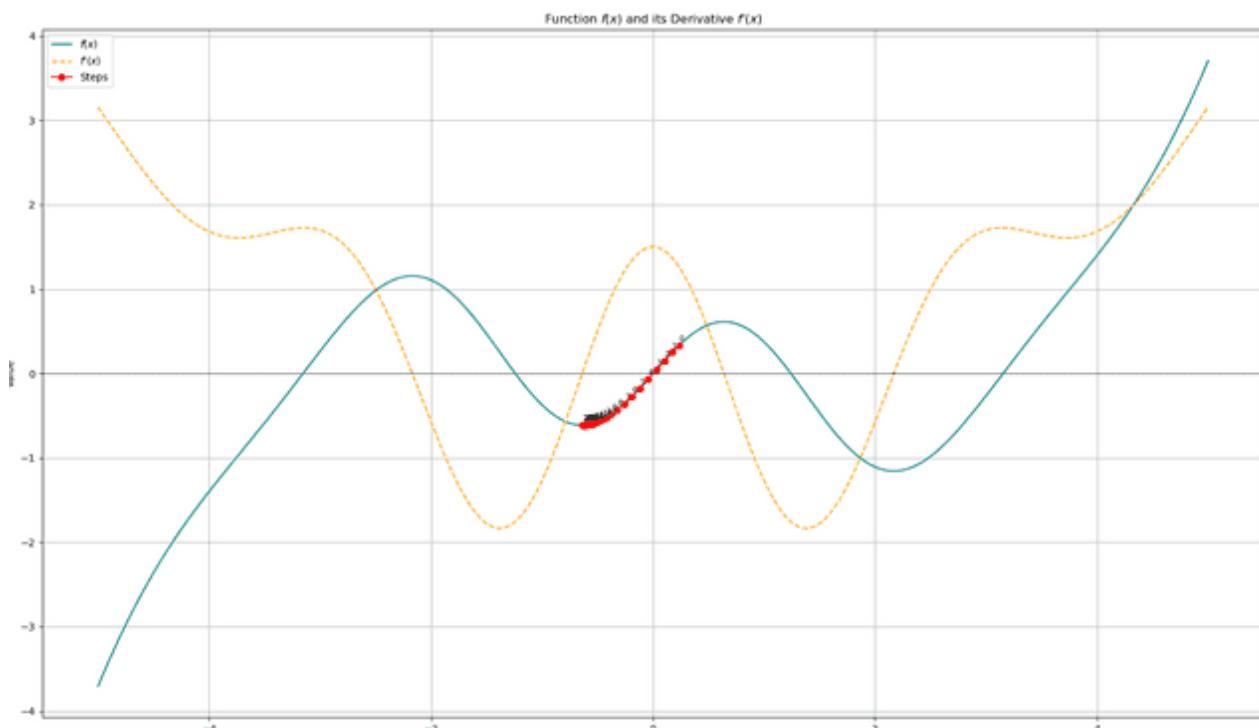
1. Alegem un  $x_0$  oarecare și calculăm derivata în punctul ales. Semnul derivatei ne va arata cum ne plasăm față de cel mai apropiat minim ( negativ – ne aflăm în stânga unui minim, pozitiv – ne aflăm în dreapta unui minim ).
2. Alegem o valoare pentru rata de antrenare lr. Folosind analogia „busolei”, rata de antrenare reprezintă „numărul de pași pe care îi parcurgem până când verificăm iar busola”.
3. Ajustăm  $x_0$  cu formula  $x_0 = x_0 - lr * g'(x)$  ( ne deplasăm spre minim-ul local ).

Pentru valori mici ale lui lr, ne vom putea apropiă mai mult de minim-ul local, dar vor fi necesari mai mulți pași.

Pentru valori mari lr ne vom apropiă mai repede de minim, dar vom fi mai departe de minim-ul real decât în cazul anterior. În cazul valorilor foarte mari, algoritmul poate să sără peste minim.

4. Repetăm până când ne-am apropiat suficient de minim sau până când am ajuns la numărul maxim de iterații.

Implementarea completă a algoritmului în Python se poate găsi în Implementare Gradient Descent pe caz bidimensional.



Valoarea reală a minim-ului local este ( -0.63846, -0.61268 ), iar punctul găsit de algoritm este ( 0.6384, -0.6127 ). Algoritmul nu va putea găsi mereu minim-ul global și se va opri la primul minim găsit. Funcția  $g(x)$  nici măcar nu are un minim global, în partea stângă, graficul coboară spre -infinit. Situația în care algoritmul rămâne blocat într-un platou și nu poate găsii minim-ul global este specifică și rețelelor neuronale, unde fenomenul descris duce la blocarea procesului de antrenare.

Algoritmul descris mai devreme reprezintă un caz particular al algoritmului gradient descent în două dimensiuni. Să vedem acum cum arată algoritmul în trei dimensiuni, pentru ca mai apoi să ajungem la o formă generală a acestuia.

În cazul funcțiilor cu mai multe variabile vom deriva parțial în funcție de fiecare dintre acestea, considerând celelalte variabile constante.

Ex:

$$f: R^2 \rightarrow R f(x, y) = \frac{4x^2 + xy + 2y^2 - 1}{10} , \text{ cu derivatele parțiale } f_x'(x, y) = \frac{8x+y}{10}, \text{ respectiv}$$

$$f_y'(x, y) = \frac{x+4y}{10}.$$

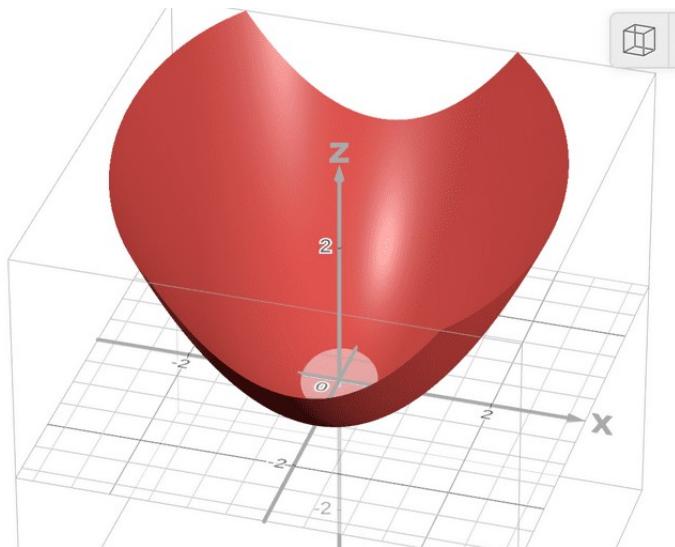


Figure 16: Graficul lui  $f(x,y)$

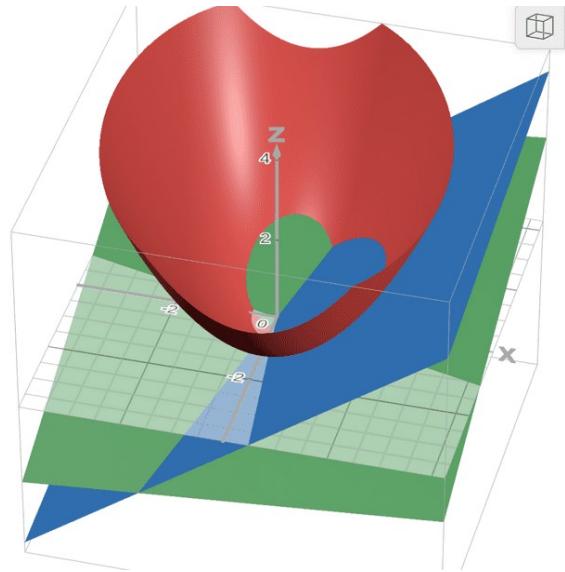


Figure 15: Graficul lui  $f(x,y)$  + graficele derivatele parțiale

Din figurile 15 și 16 putem observa că derivatele își păstrează proprietățile chiar și pentru funcțile cu mai multe dimensiuni. Ambele derive au valoarea 0 în punctul de minim local. Semnul se interpretează puțin diferit față de cazul bidimensional. Semnul derivatelor parțiale indică direcția funcției atunci când modificăm valoarea unei variabile, fără a modifica valoarea celeilalte variabile, adică indică direcția pe o singură axă. Pentru a obține direcția finală trebuie să combinăm cele două direcții. Combinarea direcțiilor se face prin scrierea vectorului ce conține derivatele parțiale pentru fiecare ramură, acest vector poartă denumirea de gradient.

$$\nabla f(x, y) = [f_x'(x, y), f_y'(x, y)]$$

Gradient Descent, cunoscut și sub denumirea de „algoritmul celei mai abrupte pante descendente”, reprezintă o metodă iterativă de optimizare utilizată pe scară largă în domeniul învățării automate. Scopul principal al algoritmului este de a determina un minim local pentru funcția de cost

asociată unui modelului. Minimizarea costului va duce la minimizarea diferențelor dintre predicțiile modelului și valorile reale observate. Gradient Descent stă la baza antrenării majorității modelelor de învățare supravegheată.

Funcția  $f(x, y) = \sin(x)\cos(y) + \frac{(x^2 + y^2)}{10}$  a fost aleasă pentru scrierea pașilor algoritmului gradient descent. Derivatele parțiale sunt  $f'_x = \cos(x)\cos(y) + \frac{x}{20}$  și  $f'_y = -\sin(x)\sin(y) + \frac{y}{20}$

Gradient Descent pentru funcția din exemplu:

1. Scriem gradientul funcției.

$$\nabla = [f'_x(x, y), f'_y(x, y)] = \left[ \cos(x)\cos(y) + \frac{x}{20}, -\sin(x)\sin(y) + \frac{y}{20} \right]$$

2. Alegem o valoare aleatoare pentru  $x_0$  și una pentru  $y_0$ .

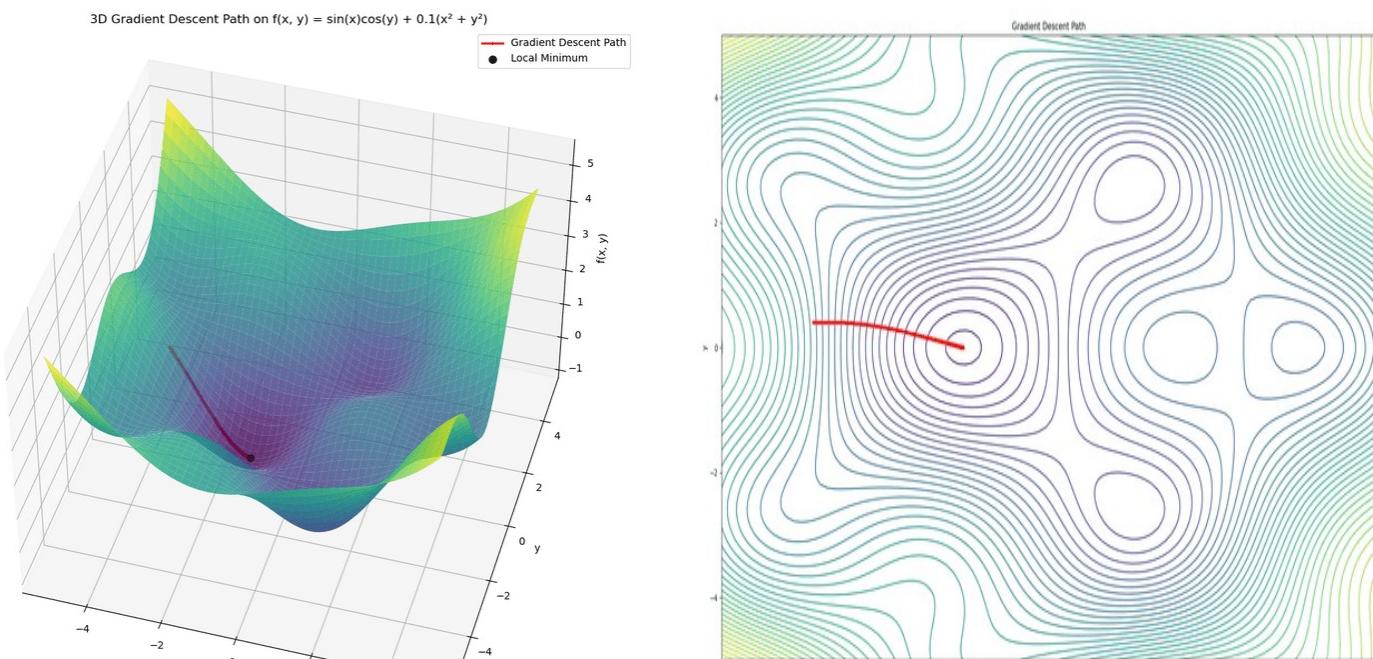
3. Alegem rata de antrenare.

4. Ajustăm valorile.

$$x_0 = x_0 - lr * \nabla_1 \quad y_0 = y_0 - lr * \nabla_2$$

Repetăm până când ajungem la convergență ( valoarea lui  $f(x_0, y_0)$  nu se schimbă semnificativ de la o iterare la alta ) sau până când ajungem la numărul maxim de iterări.

Implementarea în limbajul python al algoritmului se găsește în Algoritm Gradient Descent caz tridimensional.



În urma rulării s-a găsit un minim local în punctul (-1.3071, 0.0002, -0.7946).

Forma generală a algoritmului arată astfel:

$$Fie f: R^n \rightarrow R, f(x=[x_1, x_2, \dots, x_n]) = y$$

$$\nabla = [f'_{x_1}(x), f'_{x_2}(x), f'_{x_3}(x), \dots, f'_{x_n}(x)]$$

$$x_i = x_i - lr * \nabla_i$$

### 1.3.2 Funcția de cost

Funcția de cost este o funcție matematică ce modelează pierderea sau penalizarea asociată unui eveniment. În economie, aceasta este utilizată pentru a reprezenta costul monetar sau riscul asociat unei decizii, iar în domeniul învățării automate, funcția de cost cuantifică diferențele dintre predicțiile unui model și valorile reale, penalizând erorile de predicție. Indiferent de tipul situației pe care vrem să o modelăm, ne dorim maximizarea funcției obiectiv asociate evenimentului și putem realiza asta în mod indirect prin minimizarea funcției inverse, funcția de cost. Algoritmul Gradient Descent este una din metodele prin care putem minimiza funcția de cost. Nu maximizăm direct funcția obiectiv pentru că nu putem ajusta coeficienții pe baza acesteia. Gradientul poate fi calculat doar pentru variabilele funcției, care în funcția obiectiv sunt datele de intrare. În funcția de cost coeficienții și variabilele sunt inversate, ceea ce permite calculul gradientului pentru coeficienții funcției obiectiv.

Regresia liniară este cel mai simplu algoritm de învățare supervizată, expresia ei fiind formată dintr-o singură ecuație liniară. În cadrul regresiei ne dorim minimizarea diferențelor dintre predicții și valoarea reală și putem utiliza MSE ( Mean Square Error ) pentru a reprezenta costul asociat regresiei.

Pentru regresie gradient descent arată astfel:

1. Scriem forma generală a regresiei

$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + \dots + w_n x_{in} + b \text{ sau}$$

$$\hat{y}_i = w x_i + b, unde w este vectorul [w_1, w_2, w_3, \dots, w_n], iar x_i este [x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}]$$

2. Scriem eroarea și costul asociat regresiei

$$\hat{e}_i = y - \hat{y}_i = y_i - w x_i - b$$

$$L = MSE = \frac{1}{m} \sum (y_i - \hat{y}_i)^2 = \frac{1}{m} \sum (y_i - wx_i - b)^2$$

3. Aflăm derivata costului în funcție de  $w_j$

$$L'_{w_j} = \left( \frac{1}{m} \sum (y_i - wx_i - b)^2 \right)'_{w_j} = \textcolor{red}{\cancel{L}}$$

$$\textcolor{red}{\cancel{L}} \frac{1}{2m} \sum (y_i - wx_i - b) (y_i - wx_i - b)'_{w_j} = \textcolor{red}{\cancel{L}}$$

$$\textcolor{red}{\cancel{L}} - \frac{1}{2m} \sum (y_i - wx_i - b) x_i = \textcolor{red}{\cancel{L}}$$

$$-\frac{1}{2m} \sum \hat{e}_i x_i$$

4. Aflăm derivata costului în funcție de temenul liber  $b$

$$L'_{b} = \left( \frac{1}{m} \sum (y_i - wx_i - b)^2 \right)'_b = \textcolor{red}{\cancel{L}}$$

$$\textcolor{red}{\cancel{L}} \frac{1}{2m} \sum (y_i - wx_i - b) (y_i - wx_i - b)'_b = \textcolor{red}{\cancel{L}}$$

$$\textcolor{red}{\cancel{L}} - \frac{1}{2m} \sum (y_i - wx_i - b) = \textcolor{red}{\cancel{L}}$$

$$\textcolor{red}{\cancel{L}} - \frac{1}{2m} \sum \hat{e}_i$$

5. Scriem gradientul funcției de cost

$$\nabla L = [L'_{w_1}, L'_{w_2}, L'_{w_3}, \dots, L'_{w_n}, L'_b]$$

6. Ajustăm parametrii funcției

$$w_i = w_i - lr * \nabla L_i = wi - lr * L'_{wi}$$

$$b = b - lr * \nabla L_{n+1} = b - lr * L'_b$$

Implementarea în limajul C++ se găsește în Logistic and Linear Regression implemenation in C++

Gradient Descent poate fi utilizat pentru a ajusta un model descris de o singură ecuație. Totuși, nu poate fi aplicat direct atunci când avem de-a face cu mai multe ecuații interdependente, aşa cum se întâmplă într-o rețea neuronală, unde cunoaștem costul asociat ultimului strat, dar nu și costurile straturilor anterioare.

### 1.3.3 Regula lanțului

Regula lanțului sau derivarea compusă este o metodă de calcul a derivatelor pentru funcții care depind de alte funcții. În rețelele neuronale, relațiile dintre straturi sunt exprimate matematic prin compunerea funcțiilor, deoarece straturile superioare depind de cele inferioare. George F. Simmons explică conceptul de derivare compusă în CALCULUS WITH ANALYTIC GEOMETRY prin următoarea analogie: „Dacă o mașină se deplasează de două ori mai repede decât o bicicletă și bicicleta este de patru ori mai rapidă decât un om care merge, atunci mașina se deplasează de  $2 \times 4 = 8$  ori mai repede decât omul”.

$$\begin{cases} r_1 = \frac{\text{masina}}{\text{bicicleta}} = 2 \\ r_2 = \frac{\text{bicicleta}}{\text{om}} = 4 \end{cases} \Rightarrow r_3 = \frac{\text{masina}}{\text{om}} = \frac{\frac{\text{masina}}{\text{bicicleta}} \text{bicicleta}}{\frac{\text{om}}{\text{bicicleta}} \text{bicicleta}} = \frac{\text{masina}}{\text{bicicleta}} * \frac{\text{bicicleta}}{\text{om}} = r_1 * r_2 = 2 * 4 = 8,$$

principiu ce poate fi aplicat oricărui raport

Conform definiției de mai devreme, derivata era limita raportului  $\frac{f(x_0+h) - f(x_0)}{(x+h) - x}$ , deci putem scrie  $f'(x) = \frac{dy}{dx}$ , folosind notația lui Leibniz. Dacă avem funcțiile  $z(y)$ ,  $y(x)$ , unde  $z$  depinde de  $y$  și  $y$  depinde de  $x$ , putem determina derivata lui  $z$  în funcție de  $x$  folosind principiul explicitat de George Simmons astfel:

$$\begin{cases} d_1 = \frac{dz}{dy} \\ d_2 = \frac{dy}{dx} \Rightarrow d_3 = d_1 * d_2 \Rightarrow \frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}, \\ d_3 = \frac{dz}{dx} \end{cases} \quad \text{iar} \quad \text{forma generalizată a lanțului este}$$

$$\frac{du_n}{dx} = \frac{du_n}{du_{n-1}} \frac{du_{n-1}}{du_{n-2}} \frac{du_{n-2}}{du_{n-3}} \dots \frac{du_1}{dx}$$

### 1.3.4 Antrenarea rețelei

Backpropagation este combinația celor două metode, Gradient Descent și regula lanțului. Algoritmul folosește regula lanțului pentru a transmite mai departe costul asociat ultimului strat către funcțiile anterioare, după ce am aflat costul vom ajusta parametrii rețelei folosind algoritmul Gradient Descent.

Demonstrațiile din acest document au fost inspirate de Backpropagation calculus | Deep Learning Chapter 4. Pentru a explica modul prin care o rețea neuronală poate să învețe, vom construi o rețea de tip MLP cu 3 straturi ( input-ul nu va fi considerat strat ):

Forma rețelei :{I(2),H1(3),H2(2),O(3)} {W1(2,3),W2(3,2),W3(2,3)} {B1(3),B(2),B(3)}

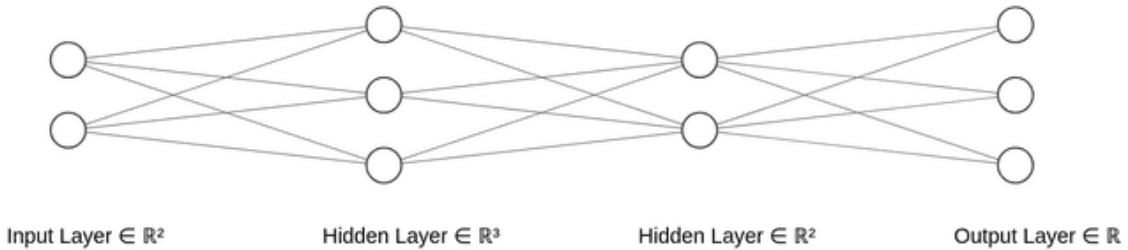


Figure 19: Arhitectura Rețelei

Rețeaua primește un vector de 2 elemente și clasifică vectorul într-una dintre cele 3 categorii (output – un neuron pentru fiecare clasă). Clasele au fost reprezentate sub forma [1, 0, 0], [0, 1, 0], [0, 0, 1], unde  $i = j$  pentru care  $y[j] = 1$  reprezintă clasa. Acest tip de codificare se numește „one-hot encoding” și poate simplifica calculul funcției de cost.

Relațiile dintre straturi sunt date de formula  $l_i = \sigma(l_{i-1}W + B)$ , deși modul de reprezentare este unul atipic, forma corespunde cu modul în care NumPy și Tensorflow reprezintă vectorii ( sub formă de vectori rând ). Stratul curent a fost notat  $l_i$ ,  $l_{i-1}$  fiind stratul anterior lui  $l_i$ ,  $W$  este matricea greutăților, iar  $B$  este vectorul termenilor liberi. Funcția de activare sigma introduce relații non-liniarități în rețea și modelează activarea neuronilor, folosim activarea ReLU pentru straturile intermediare pentru a nu permite valori negative ale activărilor (reprezentarea gradului de activare). Ultimul strat va folosi o activare diferită, deoarece neuronii de pe ultimul strat exprimă nivelul de siguranță al modelului. De

exemplu, dacă neuronul asociat clasei 0 are o activare de 0.20, înseamnă că modelul este 20% sigur că inputul aparține clasei 0, iar restul de 80% este distribuit între celelalte clase posibile.

Pentru ultimul strat funcția de activare aleasă este  $\text{softmax}(x) = \frac{e^{x_i}}{\sum e^{x_k}}$ , unde  $c$  nr de clase, ce

transformă rezultatul ultimului strat într-o distribuție de probabilități.

Având de a face cu distribuții de probabilități, costul asociat va fi  $L = -\sum y_i \log \hat{y}_i$ , unde  $\hat{y}$  este activarea softmax și se numește entropie încrucișată. Entropia este un concept introdus de Claude Shannon în A Mathematical Theory of Communication pentru a măsura gradul de nedeterminare al evenimentelor. În situația curentă entropia modelului indică gradul de incertitudine al modelului și trebuie să minimizăm cât mai mult această incertitudine pentru a obține predicții cât mai apropiate de valoarea reală.

Funcția pare destul de complicată la prima vedere, dar pentru că am reprezentat output-ul rețelei prin metoda „one-hot encoding”, știm că există un singur  $j$ , mai mic sau egal cu  $c$ , pentru care  $y_j$  are valoarea 1, iar restul sunt 0. Funcția de cost devine  $L = -\log p_j$ .

Odată ce am stabilit funcția de cost, trebuie să determinăm derivatele pentru ultimul strat. Cunoaștem următoarele relații:  $L = -\log p_j$ ,  $\hat{y}_j = \frac{e^{z_j}}{\sum e^{z_k}}$ ,  $z = l_2 W_3 + B_3$ , cu  $l_2$  output-ul stratului anterior în rețea curentă. Folosind cele 3 funcții trebuie să formăm lanțul derivatelor pentru fiecare dintre coeficienți.

$$\begin{aligned}\frac{dL}{dW} &= \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dW_3} \\ \frac{dL}{dB} &= \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dB_3}\end{aligned}$$

a)  $\frac{dL}{dp_j} = -\frac{1}{\hat{y}_j}$

b)  $\frac{d\hat{y}_j}{dz}$

Regula de derivare a raportului este  $\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$ .

Există două situații de derivare:

I.  $i = j$

Putem lua ca exemplu  $i = j = 2$

$$(\hat{y}_2)'_{z_2} = \left( \frac{e^{z_2}}{e^{z_0} + e^{z_1} + e^{z_2}} \right)' = \frac{e^{z_2}(e^{z_0} + e^{z_1} + e^{z_2}) - e^{z_2}e^{z_2}}{(e^{z_0} + e^{z_1} + e^{z_2})^2} = \frac{e^{z_2}}{(e^{z_0} + e^{z_1} + e^{z_2})} - \frac{(e^{z_2})^2}{(e^{z_0} + e^{z_1} + e^{z_2})^2} = \hat{y}_2 - \hat{y}_2^2 = \hat{y}_2(1 - \hat{y}_2) \Rightarrow$$

$$(\hat{y}_j)'_{z_j} = \hat{y}_j(1 - \hat{y}_j)$$

Din lanț  $\Rightarrow \frac{dL}{dz_j} = \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_j} = -\frac{1}{\hat{y}_j} \hat{y}_j(1 - \hat{y}_j) = \hat{y}_j - 1$ , dar pentru că  $j = i$  și  $y_i = 1 \Rightarrow$

$$\frac{dL}{dz_j} = \hat{y}_j - 1 = \hat{y}_i - y = \hat{e}_i, \text{ eroarea pt } i$$

II.  $i \neq j$

Vom considera exemplul  $j = 2, i = 1$

$$(\hat{y}_1)'_{z_2} = \left( \frac{e^{z_2}}{e^{z_0} + e^{z_1} + e^{z_2}} \right)'_{z_1} = \frac{0 * e^{z_1} - e^{z_1}e^{z_2}}{(e^{z_0} + e^{z_1} + e^{z_2})^2} = 0 - \frac{e^{z_1}e^{z_2}}{(e^{z_0} + e^{z_1} + e^{z_2})^2} = -\frac{e^{z_1}}{(e^{z_0} + e^{z_1} + e^{z_2})} \frac{e^{z_2}}{(e^{z_0} + e^{z_1} + e^{z_2})} = -\hat{y}_1 \hat{y}_2 \Rightarrow$$

$$(\hat{y}_j)'_{z_i} = -\hat{y}_j \hat{y}_i$$

Din lanț  $\Rightarrow \frac{dL}{dz_i} = \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_i} = -\frac{1}{\hat{y}_j} (-\hat{y}_j \hat{y}_i) = \hat{y}_i$ , dar  $y_i = 0$  pentru  $i \neq j \Rightarrow$

$$\frac{dL}{dz_i} = \hat{y}_i = \hat{y}_i - 0 = \hat{y}_i - y = \hat{e}_i, \text{ eroarea pt } i \neq j$$

Din I și II  $\Rightarrow \frac{dL}{dz_i} = \hat{y}_i - y = \hat{e}_i, \forall i \leq c$

c)  $\frac{dz_i}{dW_i}, \frac{dz}{dB_i}$

Funcția  $z_i$  este ecuația unei regresii  $\Rightarrow \frac{dz_i}{dW_{3_j}} = l_{2j}$ , input strat anterior,  $\frac{dz_i}{dB_{3_i}} = 1$

Având toate derivatele din lanț, putem determina forma gradientului. Deoarece  $W_i$  și  $B_i$ , gradientul se va determina folosind matricea derivatelor parțiale, matricea Jacobiană.

$$\nabla L_{W_3} = \begin{bmatrix} L'_{W_{3_{11}}} & L'_{W_{3_{12}}} & L'_{W_{3_{13}}} \\ L'_{W_{3_{21}}} & L'_{W_{3_{22}}} & L'_{W_{3_{23}}} \end{bmatrix} = \begin{bmatrix} \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_1} \frac{dz_1}{dW_{3_{11}}} & \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_2} \frac{dz_2}{dW_{3_{12}}} & \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_3} \frac{dz_3}{dW_{3_{13}}} \\ \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_1} \frac{dz_1}{dW_{3_{21}}} & \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_2} \frac{dz_2}{dW_{3_{22}}} & \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_3} \frac{dz_3}{dW_{3_{23}}} \end{bmatrix} = \underline{\underline{\epsilon}}$$

$$\begin{bmatrix} (\hat{y}_1 - y_1)l_{21} & (\hat{y}_2 - y_2)l_{21} & (\hat{y}_3 - y_3)l_{21} \\ (\hat{y}_1 - y_1)l_{22} & (\hat{y}_2 - y_2)l_{22} & (\hat{y}_3 - y_3)l_{22} \end{bmatrix} = \begin{bmatrix} \hat{e}_1 l_{21} & \hat{e}_2 l_{21} & \hat{e}_3 l_{21} \\ \hat{e}_1 l_{22} & \hat{e}_2 l_{22} & \hat{e}_3 l_{22} \end{bmatrix} = \begin{bmatrix} l_{21} \\ l_{22} \end{bmatrix} [\hat{e}_1 \quad \hat{e}_2 \quad \hat{e}_3] = l_2^T \hat{e}, \text{gradientul lui } W_3$$

este produsul dintre erori si transpusa output-ului stratului anterior și erori

$$\nabla L_{B_3} = \begin{bmatrix} L'_{B_{3_1}} & L'_{B_{3_2}} & L'_{B_{3_3}} \end{bmatrix} = \begin{bmatrix} \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_1} \frac{dz_1}{dB_{3_1}} & \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_2} \frac{dz_2}{dB_{3_2}} & \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_3} \frac{dz_3}{dB_{3_3}} \end{bmatrix} = \underline{\underline{\epsilon}}$$

$\underline{\underline{\epsilon}} [\hat{y}_1 - y_1 \quad \hat{y}_2 - y_2 \quad \hat{y}_3 - y_3] = [\hat{e}_1 \quad \hat{e}_2 \quad \hat{e}_3] = \hat{e}$ , gradientul vectorului coeficienților liberi  $B_3$  este vectorul erorilor

$B_3$  și  $W_3$  vor fi ajustați astfel:  $W_3 = W_3 - lr * l_2^T \hat{e}$ , unde  $lr$  este rata de antrenare a modelului.

După ce am ajustat ultimul strat vom trece la stratul următor,  $l_2$ . Folosim în continuare  $l_i = \sigma(l_{i-1}^T W + B)$ , dar pentru straturile anterioare funcția de activare este ReLU, de aceea output-ul  $l_2$  este  $l_2 = \text{ReLU}(f_2(l_1))$ , unde  $f_2 = l_1 W_2 + B_2$ . Lanțul derivatelor va avea forma:

$$\frac{dL}{dW_2} = \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dl_2} \frac{dl_2}{d\text{ReLU}} \frac{d\text{ReLU}}{df_2} \frac{df_2}{df_{W_2}}$$

$$\frac{dL}{dB_2} = \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dl_2} \frac{dl_2}{d\text{ReLU}} \frac{d\text{ReLU}}{df_2} \frac{df_2}{df_{B_2}}, \text{ din care cunoaștem primele două}$$

derivate

a)  $\frac{dz_i}{dl_{2j}}$

Folosim  $z = 1, j = 2$

$$\frac{dz_1}{dl_{22}} = (W_{3_{11}} l_{21} + W_{3_{21}} l_{22} + W_{3_{31}} l_{23} + B_1)'_{dl_{22}} = W_{3_{21}} \Rightarrow \frac{dz_i}{dl_{2j}} = W_{3_{ji}}$$

$$\text{b) } \frac{dl_{2j}}{dReLU}$$

Pentru că activarea ReLU a fost utilizată pentru a restrânge valorile numerelor la intervalul  $[0, \infty)$ , când  $l_{2j}$  ia valoarea 0 derivate va fi tot 0, altfel, derivata ia valoarea 1. Ca urmare, parametrii ce nu contribuie la activarea neuronilor care au determinat output-ul curent nu vor fi modificați (costul nu se poate propaga prin neuroni inactivi). În cazul neuronilor activi, costul se va putea propaga prin aceștia și valoarea costului va rămâne neschimbată. Vom definii în cod o funcție ce poate aplica acestă regula direct asupra costului determinat anterior.

$$\frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dl_2} = \text{applyReluDerivative}\left(\frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dl_2}, l_2\right), \text{ iar } l_2 \text{ devine } l_2 = f_2 = l_1 W_2 + B_2$$

Funcția are rolul de a elimina activarea din lanț și de a simplifica calcul gradientului. Noul lanț este:

$$\begin{aligned} \frac{dL}{dW_2} &= \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dl_2} \frac{dl_2}{dW_2} \\ \frac{dL}{dB_2} &= \frac{dL}{d\hat{y}_j} \frac{d\hat{y}_j}{dz} \frac{dz}{dl_2} \frac{dl_2}{dB_2} \end{aligned}$$

Datorită simplificării nu vom mai calcula alte derivate, deoarece știm că  $\frac{dl_{2i}}{dW_{2_{ji}}} = l_{1j}$ ,  $\frac{dl_{2i}}{dB_{2_{ji}}} = 1$ ,

pentru că  $l_2$  este liniară. Acum putem construi gradientul folosind matricea Jacobiană:

$$\nabla L_{W_2} = \begin{bmatrix} L'_{W_{2_{11}}} & L'_{W_{2_{12}}} \\ L'_{W_{2_{21}}} & L'_{W_{2_{22}}} \\ L'_{W_{2_{31}}} & L'_{W_{2_{32}}} \end{bmatrix} = \begin{bmatrix} \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{21}} \frac{dl_{21}}{dW_{2_{11}}} & \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{22}} \frac{dl_{22}}{dW_{2_{12}}} \\ \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{21}} \frac{dl_{21}}{dW_{2_{21}}} & \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{22}} \frac{dl_{22}}{dW_{2_{22}}} \\ \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{21}} \frac{dl_{21}}{dW_{2_{31}}} & \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{22}} \frac{dl_{22}}{dW_{2_{32}}} \end{bmatrix}, \text{ motivul pentru care folosim simbolul}$$

„sumă” este pentru că erorile provin de la 3 neuroni diferenți, eroarea finală va fi cumulul acestor erorilor.

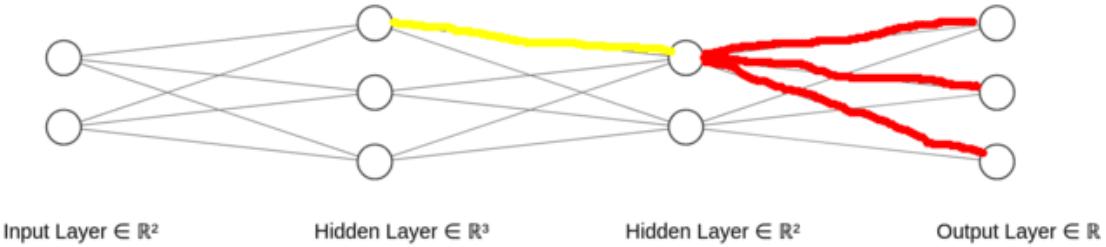


Figure 20: Acumularea erorilor

$$\nabla L_{W_2} = \begin{bmatrix} L'_{W_{2_1}} & L'_{W_{2_2}} \\ L'_{W_{2_1}} & L'_{W_{2_3}} \\ L'_{W_{2_2}} & L'_{W_{2_3}} \end{bmatrix} = \begin{bmatrix} \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{21}} \frac{dl_{21}}{dW_{2_{11}}} & \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{22}} \frac{dl_{22}}{dW_{2_{12}}} \\ \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{21}} \frac{dl_{21}}{dW_{2_{21}}} & \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{22}} \frac{dl_{22}}{dW_{2_{22}}} \\ \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{21}} \frac{dl_{21}}{dW_{2_{31}}} & \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{22}} \frac{dl_{22}}{dW_{2_{32}}} \end{bmatrix} = \textcolor{red}{\dot{e}}$$

$$\textcolor{red}{\dot{e}} \begin{bmatrix} \sum (\hat{y}_k - y_k) w_{3_{1k}} l_{11} & \sum (\hat{y}_k - y_k) w_{3_{2k}} l_{11} \\ \sum (\hat{y}_k - y_k) w_{3_{1k}} l_{12} & \sum (\hat{y}_k - y_k) w_{3_{2k}} l_{12} \\ \sum (\hat{y}_k - y_k) w_{3_{1k}} l_{13} & \sum (\hat{y}_k - y_k) w_{3_{2k}} l_{13} \end{bmatrix} = \begin{bmatrix} \sum w_{3_{1k}} \hat{e}_k l_{11} & \sum w_{3_{2k}} \hat{e}_k l_{11} \\ \sum w_{3_{1k}} \hat{e}_k l_{12} & \sum w_{3_{2k}} \hat{e}_k l_{12} \\ \sum w_{3_{1k}} \hat{e}_k l_{13} & \sum w_{3_{2k}} \hat{e}_k l_{13} \end{bmatrix} = \textcolor{red}{\dot{e}}$$

$$\textcolor{red}{\dot{e}} \begin{bmatrix} l_{11} \\ l_{12} \\ l_{13} \end{bmatrix} \begin{bmatrix} \sum \hat{e}_k w_{3_{1k}} & \sum \hat{e}_k w_{3_{2k}} \end{bmatrix} = \begin{bmatrix} l_{11} \\ l_{12} \\ l_{13} \end{bmatrix} \begin{bmatrix} \hat{e}_1 & \hat{e}_2 & \hat{e}_3 \end{bmatrix} \begin{bmatrix} w_{3_{11}} & w_{3_{21}} \\ w_{3_{12}} & w_{3_{22}} \\ w_{3_{13}} & w_{3_{23}} \end{bmatrix} = l_1^T \hat{e} W_3^T$$

$$\nabla L_{B_2} = \begin{bmatrix} L'_{B_{2_1}} & L'_{B_{2_2}} \end{bmatrix} = \begin{bmatrix} \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{21}} \frac{dl_{21}}{dB_{2_1}} \\ \sum \frac{dL}{dz_K} \frac{dz_K}{dl_{22}} \frac{dl_{22}}{dB_{2_2}} \end{bmatrix} = \left[ \sum (\hat{y}_k - y_k) w_{3_{1k}} \quad \sum (\hat{y}_k - y_k) w_{3_{2k}} \right]$$

$$\textcolor{red}{\dot{e}} \left[ \sum \hat{e}_k w_{3_{1k}} \quad \sum \hat{y}_k w_{3_{2k}} \right] = \begin{bmatrix} \hat{e}_1 & \hat{e}_2 & \hat{e}_3 \end{bmatrix} \begin{bmatrix} w_{3_{11}} & w_{3_{21}} \\ w_{3_{12}} & w_{3_{22}} \\ w_{3_{13}} & w_{3_{23}} \end{bmatrix} = \hat{e} W_3^T \quad \text{și reprezintă eroarea propagată stratului anterior}$$

Parametrii sunt ajustați astfel:

$W_2 = W_2 - lr * l_2^T \hat{e} W_3^T$ , eroarea propagata stratului anterior este produsul dintre transpusa matricei  $B_3 = B_3 - lr * \hat{e} W_3^T$

greutăților stratului superior și eroarea sa. Pentru că toate straturile anterioare lui  $l_2$  au același tip de activare, forma generală pentru straturile anterioare este:

$W_i = W_i - lr * l_i^T \hat{e}_{i+1} W_{i+1}^T$ , eroarea a fost transformată folosind  $\hat{e}_{i+1} = applyReluDerivative(\hat{e}_{i+1}, l_i)$   
 $B_i = B_i - lr * \hat{e}_{i+1} W_{i+1}^T$

Exemplu pentru stratul  $l_1$ :

$$W_1 = W_1 - lr * i^T \hat{e}_2 W_2^T = W_1 - lr * i^T \hat{e} W_3^T W_2^T$$

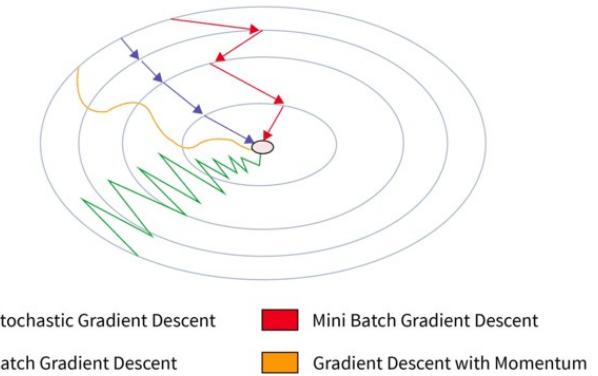
$$B_1 = B_1 - lr * \hat{e}_2 W_2^T = B_1 - lr * \hat{e} W_3^T W_2^T$$

### 1.3.5 Forme ale algoritmului

#### I. Metoda stocastică

Este metoda prezentată în demonstrația anterioară și reprezintă ajustarea parametrilor modelului pentru fiecare instanță a setului de antrenament în parte. Este o metodă ce nu necesită multe resurse de calcul și duce la o convergență mai rapidă a modelului, dar procesul de antrenare este unul instabil, deoarece modelul încearcă să aproximeze cât mai bine ultima instanță din setul de date și nu poate face asta fără să se depărteze de optimul real.

Backpropagation in Neural Network



Cauza acestui fenomen este faptul că graficul se modifică cu fiecare iterație. Aceste fluctuații pot fi un impediment în antrenarea modelului, dar pot avea și efecte pozitive. Unul dintre defectele algoritmului Gradient Descent este posibilitatea ca acesta să rămână blocat într-un platou. Dacă algoritmul întâlnește un minim local pentru o instanță, atunci când graficul se va schimba pentru a doua instanță, este posibil ca acel minim local să dispară, iar procesul de antrenare nu va mai fi blocat.

Backpropagation in Neural Network ROThe Misconception that Almost Stopped AI [How Models Learn Part 1]

Metoda este utilizată pentru modelele care necesită actualizare continuă în contextul modificării datelor existente sau al apariției unor date noi. Exemple relevante includ platformele de comerț electronic care învață preferințele utilizatorilor în timp real, precum și sistemele de recomandare utilizate de site-uri precum YouTube.

[Backpropagation in Neural Network](#)

## 2. Metoda Batch

Presupune calcularea valorilor medii ale output-urilor și ale erorilor, iar mai apoi antrenarea modelului folosind valorile medie. La fiecare pas vom folosi aceeași instanță, reprezentată de valoarea medie. Utilizarea acestei metode duce la un proces de antrenare stabil și se poate apropia mai mult de optimul real, deoarece algoritmul ia în considerare fiecare instanță a setului de date. Un prim dezavantaj este consumul excesiv de memorie, deoarece toate instanțele setului de date trebuie să fie încărcate în memoria calculatorului pentru a putea calcula valorile medii. De asemenea, metoda nu poate rezolva problema platoului. Având doar o singură instanță, graficul funcției cu nu va modifica și algoritmul nu va putea găsi alt minim local.

[ROThe Misconception that Almost Stopped AI \[How Models Learn Part 1\]](#)

[Backpropagation in Neural Network](#)

## 3. Metoda Mini-Batch

Este o combinație a celor două metode și presupune împărțirea setului de date inițial în subseturi. Pentru fiecare din aceste subseturi sunt calculate valorile medii. Vom ajusta parametrii rețelei, o singură dată, folosind valorile subsetului, ca mai apoi să ajustăm parametrii folosind subsetul următor. Este cea mai utilizată metodă de implementare a algoritmului Backpropagation și păstrează avantajele metodelor componente. Dezavantajul metodei este complexitatea. Pentru a obține rezultate cât mai bune folosind această metodă, trebuie să ajustăm mărimea subseturilor în funcție de setul de date și de arhitectura modelului. Dacă folosim prea puține subseturi, ne vom lovi de problema platoului. În schimb, atunci când avem prea multe subseturi, procesul de antrenare va fi instabil și nu vom putea obține rezultate optime. Totodată, dimensiunea subseturilor trebuie ajustată în funcție de limitările resurselor hardware disponibile.

[ROThe Misconception that Almost Stopped AI \[How Models Learn Part 1\]](#)

[Backpropagation in Neural Network](#)

Am implementat metoda mini-batch în Backpropagation from scratch. Am utilizat un model cu 4 straturi (I(784), H1(256), H2(128), O(10)) pentru a clasifica setul de date MNIST (Modified National Institute of Standards and Technology database). MNIST a fost conceput în anul 1988 și a fost utilizat ca standard de măsurare al performanței pentru algoritmii de computer vision. În set se află 70000 de

imagini cu litere scrise de mâna, preluate din scrisori. În urma rulării algoritmului, modelul a obținut o performanță de 96.25%. Figura 23 conține activările primului strat după clasificarea unei imagini.

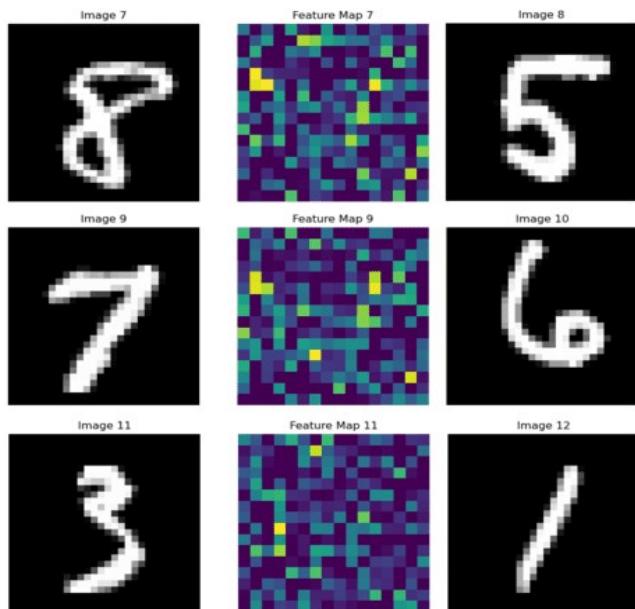


Figure 23: Activări MLP

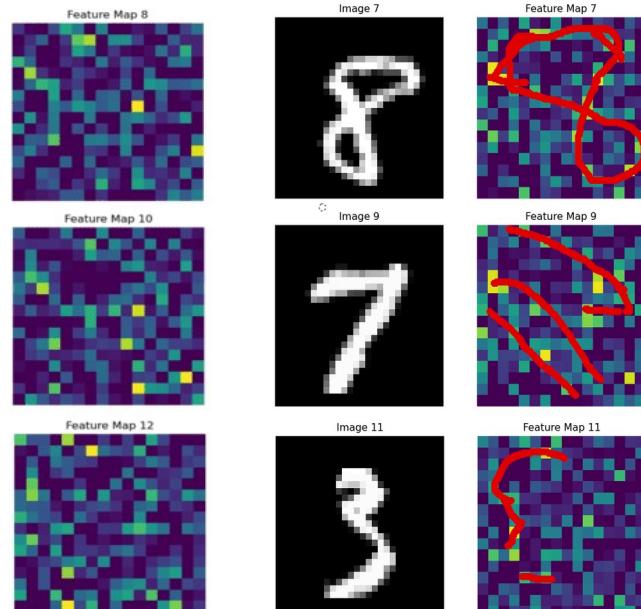


Figure 22: Tiparuri găsite de model

Se observă foarte slab anumite tipare în imaginile din prima coloană, dar pentru imaginile din a 3-a coloană nu se pot observa tipare. Arhitectura MLP nu era un capabilă să extragă informații spațiale din poză. La acea vreme, exista însă conceptul de CNN (Convolutional Neural Network), rețele ce puteau extrage informații spațiale din imagini, dar acestea nu puteau fi implementate din cauza limitărilor hardware. Deși algoritmul backpropagation a revoluționat inteligența artificială, limitările tehnologice din acea vreme au dus la stagnarea domeniului. Doar câțiva ani mai târziu, industria s-a confruntat cu o a doua „iarnă a inteligenței artificiale” (AI Winter), care a luat sfârșit odată cu apariția plăcilor video. But what is a neural network? | Deep learning chapter 1

## 1.4 Inteligența artificială după anul 2000. Apariția plăcilor video

Înainte de apariția plăcilor video, afișarea videoclipurilor și a imaginilor, grafica jocurilor și antrenarea inteligenței artificiale erau realizate exclusiv de către procesor. Totuși, grafica digitală nu putea fi generată eficient de un procesor. Procesoarele sunt unități de calcul cu un număr redus de nuclei, dar capabile să execute rapid secvențe de operații logice sau aritmetice. În ciuda vitezei mari de

calcul, procesoarele nu pot randa eficient imagini, deoarece sunt proiectate să execute secvențe de instrucțiuni, în timp ce generarea imaginilor digitale necesită un volum mare de calcule efectuate în paralel, datorită numărului ridicat de pixeli ale căror valori trebuie actualizate la intervale scurte de timp. Prima placă video a fost lansată în 1999 de către Nvidia, marcând o creștere semnificativă a performanței calculatoarelor din acea perioadă. A Brief History of Neural Networks

Procesorul grafic NVIDIA GeForce 256 nu era însă unul programabil, având proceduri predefinite pentru crearea imaginilor, dar odată cu popularizarea plăcilor video, NVIDIA a urmărit extinderea capacitaților plăcilor video. Programatorul putea interacționa cu placa prin intermediul API-urilor DirectX 8.0 și OpenGL 1.3. API-urile permiteau încărcarea unui vector de puncte geometrice (vertex buffer) și mai apoi încărcarea unui program definit anterior, denumit shader, prin care programatorul indica modul prin care placa video va procesa fiecare punct în parte. Shader-ele ofereau multă flexibilitate în interacțiunea cu placăile video, dar puteau fi folosite doar pentru generarea de imagini. Un shader acceptă date sub un singur format (puncte geometrice), iar output-ul unui shader va fi mereu un pixel afișat pe ecranul calculatorului.

O lucrare din 2007, publicată de Universitatea Stanford, a demonstrat că plăcile video oferă performanțe mai bune decât procesoarele în efectuarea calculelor algebrice. Această descoperire a evidențiat potențialul plăcilor video de a fi utilizate în domenii de cercetare și în sarcini care necesită astfel de calcule. Lucrarea a atrăs atenția companiei NVIDIA, care, în același an, a lansat platforma CUDA (Compute Unified Device Architecture) pentru programarea plăcilor video în scopuri generale. Lansarea platformei CUDA a făcut iar posibilă dezvoltarea inteligenței artificiale, plăcile video accelerând cu mult procesul de antrenare.

În subcapitolul anterior, am reprezentat componente rețelei sub forma unor vectori și matrice, iar mai apoi am ajustat acești parametrii pe baza unor calcule algebrice. Putem lua exemplul următor pentru a înțelege de ce plăcile video facilitează procesul de antrenare:

$$\text{Produsul matricelor } M_1 = \begin{bmatrix} 3 & 2 \\ 3 & 4 \end{bmatrix}, M_2 = \begin{bmatrix} 5 & 2 \\ 2 & 1 \end{bmatrix} \text{ se calculează astfel: } M_3 = \begin{bmatrix} 3 * 5 + 2 * 2 & 3 * 2 + 2 * 1 \\ 3 * 5 + 4 * 2 & 3 * 2 + 4 * 1 \end{bmatrix}$$

Pentru aflarea rezultatului sunt necesare 4 calcule, câte unul pentru fiecare celula a matricei rezultat ( $2 * 2 - \text{numărul de linii al matricei } M_1 \text{ înmulțit cu numărul de coloane al matricei } M_2$ ). Procesorul va trebui să calculeze pe rând fiecare celulă a matricei, în timp ce placa video le va calcula simultan.

Pentru exemplul dat nu se vor observa creșteri semnificative de performanță, dar rețelele neuronale folosesc milioane sau chiar miliarde de parametri. Să considerăm acum matricele  $M_1(1024,512)$ ,  $M_2(512,128) \Rightarrow M_1 * M_2 = M_3(1024,128) \Rightarrow 1024 * 128 = 134144$  de calcule. O placă video de buget are 3000 de nuclee, iar un procesor din același interval de preț are numai 8 nuclee. Pentru calculul lui  $M_3$  procesorul va face  $134144 / 8 = 16768$  de pași, atunci când folosim thread-uri, iar placa video  $134144 / 8 \sim 45$  de pași, diferența fiind una majoră. Calculele din rețelele neuronale pot deveni cu mult mai complicate, deoarece majoritatea rețelelor moderne folosesc tablouri cu mai mult de două dimensiuni. Aceste tablouri se numesc tensori.

## 1.5 AlexNet

Prima rețea ce a fost antrenată folosind o placă video dedicată a fost AlexNet, în anul 2012. Concursul ILSVRC (ImageNet Large Scale Visual Recognition Challenge) punea la dispoziție un set de date cu 1281167 de imagini împărțite în 1000 de categorii, în funcție de obiectul care se află în imagine. A fost cel mai mare concurs de viziune computerizată de la acea vreme. Modelele erau clasate în funcție de erorile de tipul Top1 și Top5. Top1 este procentul de instanțe clasate greșit, iar Top5 este procentul instanțelor pentru care clasa corectă nu se află printre primele 5 cele mai mari probabilități din distribuția output-ului. Rețeaua convezională AlexNet, proiectată de Alex Krizhevsky, a întrecut cu mult așteptările organizatorilor, având o eroare Top1 de 36.7% și Top5 de 15.3 %. Eroarea Top1 se poate traduce printr-o acuratețe a modelului de 63.3%, în timp acuratețea modelul clasat pe locul al doilea avea o acuratețe de sub 40%. Conceptul de rețea convezională nu era unul nou, acestea fiind utilizate încă din 1980. Alex Krizhevsky a preluat arhitectura rețelelor convezionale și a extins-o conform capacităților tehnologiei moderne. Folosind 2 plăci dedicate GTX 580 cu 3GB VRAM, el a reușit să antreneze un model cu 60 de milioane de parametri și 650000 de neuroni. Antrenarea unui asemenea model era considerat imposibil până la acea vreme. Krizhevsky a fost prima persoană care a demonstrat capacitățile plăcilor video în antrenarea inteligenței artificiale. Până la acea vreme, se credea că arhitecturile vechi nu au capacitatea de-a realiza sarcini mai complexe. În realitate, însă, arhitecturile vechi obțin rezultate mult mai bune atunci când sunt scalate. [AlexNetImageNet Classification with Deep Convolutional Neural Networks](#)

Arhitectura rețelei AlexNet a fost atât de inovativă la acea vreme, încât a dus la o explozie a investițiilor în cercetarea inteligenței artificiale. La scurt timp au apărut primii asistenți vocali, precum

Cortana și Siri, iar numărul de mașini ce se puteau conduce singure a crescut. Tot în aceeași perioadă au apărut și programe ce puteai fi utilizate pentru diagnosticarea automată, precum Zebra Medical Vision.

La trei ani după publicarea lucrării lui Krizhevsky, OpenAI a fost fondat. Tehnologia care a dus la apariția modelului de limbaj ChatGPT, transformatorul, a apărut în 2017. Transformatorul a fost o adaptare modernă a rețelelor neuronale recurente, care simulau memoria de durată scurtă. Transformatoarele au devenit unelte ce pot fi utilizate pentru a rezolva orice tip de problemă ce putea fi rezolvată cu ajutorul inteligenței artificiale, fiind capabile să proceseze date sub formă de text, imagini sau sunete mai bine decât orice altă arhitectură. Cu timpul, transformatoarele au devenit un standard pentru modelele largi, mai ales cele de limbaj. The history of AI

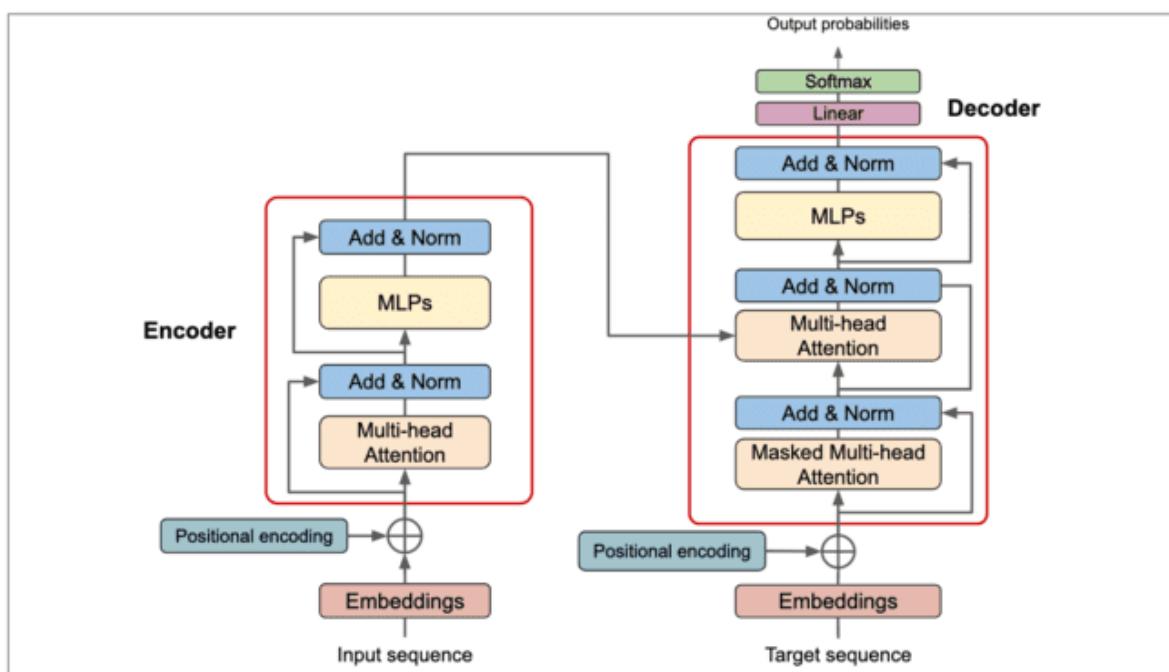


Figure 24: Arhitectura unui transformator

## 1.6 Aplicații ale rețelelor neuronale în medicină

Până la explozia inteligenței artificiale din 2012, datele medicale erau analizate folosind metode clasice de învățare automată, precum vectorii de suport, arborii de decizie, regresia și tehniciile de clusterizare. Odată cu apariția rețelelor neuronale, au fost dezvoltate noi metode de analiză și procesare a datelor medicale, iar în anii următori utilizările acestora în domeniul medicinei au crescut considerabil.

### **1.6.1 Diagnosticare**

Rețelele neuronale pot asista medicii în stabilirea diagnosticului prin analizarea unor seturi mari de date, identificând informațiile relevante. De asemenea, aceste modele pot ajuta la prelucrarea datelor pentru a le face mai ușor de interpretat, de exemplu, prin separarea părților esențiale ale imaginii care sunt importante pentru diagnostic, proces ce poartă numele de segmentare. Inteligența artificială poate, totodată, să reducă numărul cazurilor în care bolile nu sunt detectate la timp, deoarece este capabilă să identifice tipare subtile care pot să-i scape medicului. Deep Neural Networks and Applications in Medical Research

### **1.6.2 Analiză predictivă**

Pe baza analizei informațiilor istorice și genetice, o rețea neuronală poate prezice șansele de apariție a unei boli, precum și modul în care aceasta va evoluă în cazul în care pacientul a fost deja diagnosticat. Inteligența artificială oferă un avantaj major în prevenirea anumitor afecțiuni, deoarece poate analiza un volum mare de date într-un timp foarte scurt. Astfel, medicul nu va mai fi nevoie să desfășoare investigații pe perioade îndelungate. Deep Neural Networks and Applications in Medical Research

### **1.6.3 Tratament personalizat**

Găsirea tratamentului potrivit este un proces dificil, atât pentru medic, cât și pentru pacient. Indiferent de afecțiune, nu există un tratament universal, deoarece eficiența acestuia depinde de particularitățile anatomici și genetice ale fiecărui pacient. Un tratament care a dus la vindecarea unui pacient poate fi ineficient pentru altul sau chiar să provoace efecte adverse. De aceea, alegerea tratamentului adecvat consumă mult timp și poate provoca stres pacientului, mai ales atunci când apar efecte secundare sau când boala se agravează. Folosind datele genetice ale pacientului și istoricul reacțiilor la medicamente, inteligența artificială poate identifica tratamentul optim într-un timp mult mai scurt. De exemplu, reacția la substanțele utilizate în tratarea afecțiunilor psihologice, precum depresia și anxietatea, este influențată de profilul nostru genetic. Din acest motiv, există deja teste automate care pot determina compatibilitatea pacientului cu anumite medicamente pe baza informației genetice. Deep Neural Networks and Applications in Medical Research

#### **1.6.4 Monitorizarea pacientului de la distanță**

Dispozitivele inteligente, cum ar fi ceasurile smart, sunt dotate cu senzori care pot monitoriza constant starea de sănătate a pacientului. În multe cazuri, aceste ceasuri au ajutat pacienții și medicii să detecteze probleme precum aritmia sau apneea în somn. Persoanele care au suferit un infarct miocardic sau un accident vascular cerebral pot purta permanent astfel de dispozitive pentru monitorizarea ritmului cardiac și a tensiunii arteriale. Pe ceasuri sau pe dispozitivele conectate la acestea pot fi instalate aplicații care compară valorile curente ale pulsului și tensiunii arteriale cu cele anterioare, pentru a detecta anomalii. În cazul identificării unui risc, sistemul poate alerta atât medicul, cât și pacientul, contribuind astfel la prevenirea unui al doilea eveniment care ar putea fi fatal.

Deep Neural Networks and Applications in Medical Research

# Capitolul 2. Arhitecturi și tehnologii utilizate

## 2.1 CNN

Pentru a înțelege arhitectura rețelelor conoluționale trebuie să analizăm arhitectura lui AlexNet.

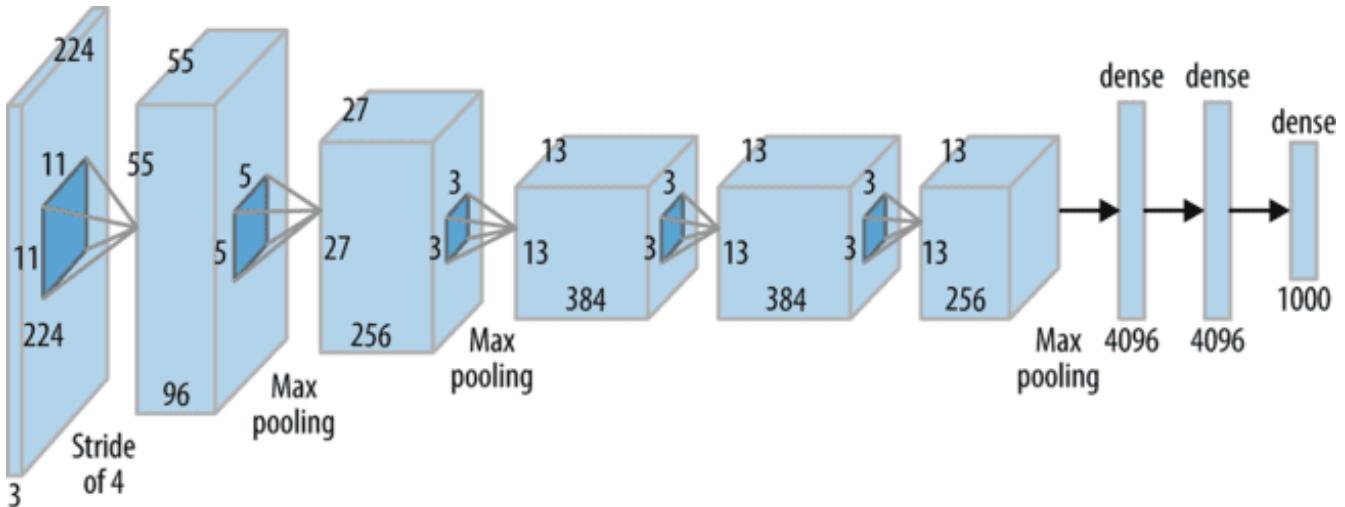


Figure 25: Arhitectură AlexNet

Rețeaua era compusă din 8 blocuri, 5 blocuri conoluționale pentru extragerea tiparelor din imagini și 3 pentru clasificare, formate din straturi perceptron. Blocurile conoluționale cele mai importante și se compun din 3 straturi ce se ocupă de extragerea tiparelor din imagini, subeșantionare, respectiv activare.

### 2.1.1 Straturile conoluționale

În matematică, conoluția este o operație binară care combină două funcții pentru a produce o a treia funcție care exprimă modul în care graficul primei funcții influentează graficul celei de-a doua. Această operație implică reflectarea funcției  $g$  în jurul originii ( $g(\tau) \rightarrow g(-\tau)$ ), urmată de translația acesteia cu o valoare  $t$ , și apoi integrarea produsului dintre funcțiile. Conoluția este utilizată frecvent în analiza semnalelor, procesarea imaginilor, teoria sistemelor liniare și rețelele neuronale conoluționale. Conoluție

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

Ecuația de mai sus este forma conoluției pentru cazul continuu. Pentru că imaginile digitale sunt reprezentă șiruri cu un număr finit de valori, în procesarea imaginilor se folosesc conoluții discrete. Conoluție

Exemplu de conoluție discretă:

Fie funcțiile F: [1, 2, 3, 4] -> [1, 3, 6, 5] și G: [1, 2, 3, 4] -> [4, 2, 2, 1]

Pentru a obține conoluția, trebuie să rotim imaginea lui G. Imaginea rotită va fi plasată sub imaginea funcției F, astfel încât ultimul număr din ImG să se afle sub primul număr din ImF. Apoi, vom deplasa imaginea lui G spre dreapta, câte o unitate, până când ImG trece de ultimul element din ImF. La fiecare pas, vom înmulți elementele aflate unul sub altul, vom aduna rezultatele și vom plasa suma obținută într-un nou șir. ROBut what is a convolution?

Table 2: Caclul conoluții discrete

Pas	F * G	Produs scalar	Sumă
0	[1, 3, 6, 5] [1, 2, 2, 4]	1 * 4	4
1	[1, 3, 6, 5] [1, 2, 2, 4]	1 * 2 + 3 * 4	14
2	[1, 3, 6, 5] [1, 2, 2, 4]	1 * 2 + 3 * 2 + 6 * 4	32
3	[1, 3, 6, 5] [1, 2, 2, 4]	1 * 1 + 3 * 2 + 6 * 2 + 5 * 4	39
4	[1, 3, 6, 5] [1, 2, 2, 4]	3 * 1 + 6 * 2 + 5 * 2	25
5	[1, 3, 6, 5] [1, 2, 2, 4]	6 * 1 + 5 * 2	16
6	[1, 3, 6, 5] [1, 2, 2, 4]	5 * 1	5

Pas	$F * G$	Produs scalar	Sumă
7	[1, 3, 6, 5]	-	-
	[1, 2, 2, 4]		

Și conoluția  $F * G$  este [4, 14, 32, 39, 25, 16, 5].

În cazul procesării imaginilor, conoluțiile sunt utilizate pentru transformarea imaginilor și aplicarea filtrelor, combinând imaginea originală cu o matrice convoluțională. La fiecare deplasare a unei metrice spre dreapta se va obține un pixel nou prin combinarea pixelilor apropiati.

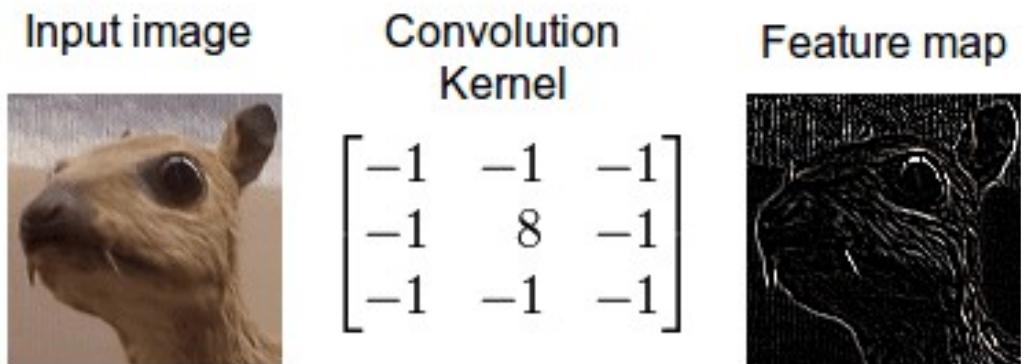


Figure 26: Edge detection

Matricea convoluțională din figura 27 este utilizată pentru a detecta contururile obiectelor din imagine. La prima vedere, nu am putea înțelege cum reușește această matrice să separe obiectele din imagini. Putem aplica o conoluție asupra unui pixel din matrice pentru a înțelege cum funcționează.

$$M_{p_5} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix}, \text{kernel} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \text{prin aplicarea conoluției obținem } p_5' = 8p_5 - \sum p_i$$

, dar valoarea lui  $p_5'$  nu este o valoare validă, pentru că intervalul de valori a devenit  $[-8 * 255, 8 * 255] = [-2040, 2040]$ . Pentru a aduce valoarea lui  $p_5'$  înapoi în intervalul  $[0, 255]$ , trebuie să calculăm modulul lui  $p_5'$  împărțit la 8. Noua valoare a lui  $p_5'$  va fi  $p_5' = \left| p_5 - \frac{\sum p_i}{8} \right|$ . Observăm că termenul din

dreapta este chiar media vecinilor lui  $p_5'$ . Atunci,  $p_5' = |p_5 - \bar{p}|$ , care este modulul abaterii lui  $p_5$  față de media vecinilor săi. În realitatea convoluțiile sunt metode de măsurare a variației din setul de date. Atunci când pixelul din centru diferă semnificativ de ceilalți pixeli, înseamnă că acesta separă două sau mai multe obiecte din imagine. În schimb, atunci când valoarea pixelului este similară cu cea a pixelilor aflați în vecinătatea sa, el se află în interiorul obiectului. Există și alte tipuri de convoluții, de exemplu convoluții pentru aplicarea filtrelor 3D, figura 28, dar toate folosesc același principiu de măsurare al variației.



Figure 27: Filtru 3D folosind convoluții

În exemplul de mai devreme, în care am folosit setul de date MNIST pentru a antrena o rețea MLP, am observat că straturile perceptron nu sunt capabile să surpindă tiparele complexe din imagini. Pentru a rezolva această problemă, au fost introduse convoluții antrenabile în rețelele neuronale pentru a extrage informația utilă din imagini, înainte ca acestea să fie clasificate.

### 2.1.2 Activare ReLU

O perioadă îndelungată de timp, cea mai utilizată funcție de activare a fost tangenta hiperbolică, figura 29. Alex Krizhevsky a observat că rețeaua AlexNet a obținut rezultate mai bune după ce a înlocuit activarea de tip tangentă cu una de tip ReLU. Descoperirea lui Krizhevsky a dus la popularizarea activării ReLU. ImageNet Classification with Deep Convolutional Neural Networks

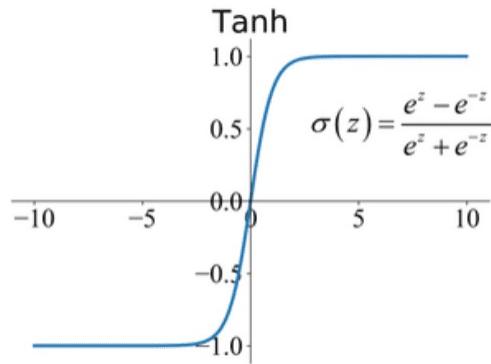


Figure 28: Tangentă hiperbolică

Câştigul de performanță se datorează simplității funcției ReLU. În subcapitolul 1.3, am eliminat activarea ReLU din lanț pentru a simplifica calculul gradientului. Același lucru nu se poate face și în cazul tangentei hiperbolice, fiind adăugate alte două derivate în lanțul final. Funcția se aplică atât straturilor perceptron, cât și straturilor conoluționale.

### 2.1.3 MaxPooling

Straturile de subeșantionare, sau MaxPooling, realizează compresia imaginilor cu scopul de a reduce numărul de parametri ai rețelei. În urma compresiei, sunt păstrate cât mai multe dintre informațiile utile extrase din imagine. Sunt denumite straturi de subeșantionare deoarece compresia imaginilor se realizează prin folosirea unei matrice conoluționale ce extrage valoarea maximă din fiecare vecinătate, informația cea mai semnificativă. Un efect pozitiv al MaxPooling-ului, datorit mecanismului de selecție a datelor, este reducerea redundanței datelor, ceea ce ajută la combaterea fenomenului de overfitting, când modelul are performanțe semnificativ mai slabe pe setul de testare față de cele obținute pe setul de antrenare. ImageNet Classification with Deep Convolutional Neural Networks

Un alt rol al subeșantionării este detectarea trăsăturilor complexe. Subeșantionarea combină treptat detaliile din imagine pentru a reconstrui obiectul inițial. În cazul unui program de detectare facilă, primul bloc poate detecta componente ale organelor de simț, cum ar fi sclera, pupila și irisul în cazul ochiului, în cel de-al doilea strat va combina aceste componente pentru a forma organele de simț, iar ultimul bloc va compune fața persoanei (situație ipotetică, verifică 3.1.3). ImageNet Classification with Deep Convolutional Neural Networks

### **2.1.3 Straturi Perceptron**

Ultimele 3 blocuri din AlexNet formează un perceptron cu mai multe straturi, arhitectură detaliată anterior în 1.3. MLP-ul este partea modelului ce se ocupă de clasificare, iar blocurile anterioare au rolul de a face imaginile mai ușor de interpretat de către MLP. *ImageNet Classification with Deep Convolutional Neural Networks*

### **2.1.4 Controlul procesului de antrenare și combaterea overfitting-ului**

Așa cum am descris anterior, există multe probleme ce pot apărea în timpul procesului de antrenare, cum ar fi instabilitatea procesului, overfitting-ul și problema platoului. Există și alte probleme ce pot apărea, precum problema dispariției gradientului și „The dying ReLU”, ce va fi abordată în capitolul anterior. Alex Krizhevsky, în lucrarea *ImageNet Classification with Deep Convolutional Neural Networks*, a găsit soluții pentru o parte din aceste probleme.

Prima metodă descrisă de Krizhevsky este preprocesarea datelor și are rolul de a reduce overfitting-ul prin aplicarea unor operațiuni asupra imaginilor, astfel încât acestea să fie interpretate mai ușor de către rețea. Operațiile se realizează înainte de începerea procesului de antrenare. Acestea pot fi folosite pentru a amplifica variația atât la nivelul setului de date, cât și la nivelul fiecărei instanțe. Dorim să introducem mai multă diversitate în setul de date atunci când aceleași porțiuni sau trăsături se repetă în cadrul unui număr mare de imagini, dar nu influențează suficient decizia modelului. Operațiuni care pot fi aplicate în această situație sunt rotațiile, deformările, translațiile și modificarea luminozității imaginii. *ImageNet Classification with Deep Convolutional Neural Networks*

La nivelul instanței, lipsa de variație între pixelii imaginii poate afecta performanța modelului. Două operații care pot rezolva această problemă sunt amplificarea contrastului imaginii și corectarea imaginii pe baza histogramei de culori. Există și cazuri în care modelul nu poate învăța eficient din cauza variației mari din interiorul imaginii. În această situație, trebuie să reducem cantitatea de informație din imagini. Deși poate părea contraintuitiv, prin blurare imaginile pot deveni mai ușor de interpretat, deoarece se elimină excesul de informație. *ImageNet Classification with Deep Convolutional Neural Networks*

Krizhevsky a numit preprocesarea în *ImageNet Classification with Deep Convolutional Neural Networks* „cea mai ușoară metodă de-a reduce overfitting-ul atunci când folosim imagini”. Metodele de preprocesare pot fi într-adevăr mai ușor de implementat decât alte metode, dar alegerea greșită a

acestora sau aplicarea excesivă poate afecta negativ setul de date. În situațiile în care clasificarea depinde de orientarea imaginii, operațiile de rotație și oglindire nu pot fi aplicate. Un exemplu este clasificarea setului MNIST: rotirea unui „6” îl poate transforma într-un „9”, iar rotirea unei imagini cu cifra „2” poate determina modelul să confundă imaginea cu un „5”. Atunci când clasificarea depinde de dimensiunea, aspectul sau poziția unui element din imagine, nu putem aplica transformări geometrice, deoarece acestea ar duce la pierderea informațiilor spațiale. Un exemplu în acest sens este clasificarea tumorilor. Detectiona tumorilor depinde de forma structurilor anatomice. Aplicarea unor filtre care distorsionează această formă poate conduce la apariția unor rezultate fals pozitive. Clasificarea tumorilor depinde de poziția și dimensiunea lor. Dacă aplicăm un filtru care micșorează imaginile, dimensiunea tumorilor se va modifica, iar acestea pot fi interpretate greșit ca tumori benigne, ce nu se pot extinde. În plus, translația tumorii poate duce la interpretarea acesteia ca un alt tip. Aplicarea agresivă a filtrelor poate duce fie la pierderea excesivă a datelor relevante, fie la inundarea imaginilor cu zgomot, ceea ce face ca informațiile importante să fie mai greu de detectat. *ImageNet Classification with Deep Convolutional Neural Networks*

A doua metodă introdusă de Krizhevsky, de această dată pentru a rezolva problema dispariției gradienților, este normalizarea batch-urilor. O problemă care apare atunci când folosim batch-uri este schimbarea distribuției datelor de la un batch la altul, ceea ce duce la deplasarea graficului funcției. Modelul va trebui să se adapteze la această schimbare, iar acest lucru se realizează prin mărirea sau micșorarea gradienților, în funcție de modul în care graficul a fost deplasat. În funcție de cât de mult a fost modificată distribuția datelor, parametrii pot fluctua de la valori apropiate de zero până la valori foarte mari. Astfel, modelul nu va putea învăța să clasifice eficient setul de date sau, în cel mai rău caz, nu va putea învăța deloc. Normalizarea datelor din batch-uri asigură faptul că toate batch-urile vor avea aceeași distribuție. Aceasta este o metodă atât de eficientă, încât utilizarea ei elimină necesitatea altor tehnici de reducere a overfitting-ului, în majoritatea cazurilor. *ImageNet Classification with Deep Convolutional Neural Networks*

Ultima metodă folosită în cadrul rețelei AlexNet este „dropout”. O altă problemă ce apare la nivelul rețelelor neuronale este codependența neuronilor. Codependența este fenomenul prin care rețeaua devine prea dependentă de anumiți neuroni pentru clasificare, ceea ce duce la duplicarea acestora în rețea (mai mulți neuroni cu aceeași parametru). Acest fenomen duce la reducerea capacitații de învățare a modelului, din cauza reducerii numărului real de neuroni, neuroni asemănători fiind de fapt același neuron. *ImageNet Classification with Deep Convolutional Neural Networks*

„Dropout”-ul este o tehnică de regularizare a parametrilor ce rezolvă problema codependenței. Tehnica presupune dezactivarea (setarea activării la 0) a unui procent de neuroni în timpul antrenării. Tehnica realizează împărțirea modelului în subrețele și antrenarea fiecărei rețele independent de celelalte subrețele. Neuronii din subrețea vor fi aleși aleatori, ceea ce înseamnă ca la fiecare pas vom antrena o subrețea diferită, ceea ce nu va permite ca neuronii să devină dependenți unii de alții. „Dropout”-ul are și alte efecte pozitive. Krizhevsky, în lucrarea ImageNet Classification with Deep Convolutional Neural Networks, a observat că tehnica duce la „învățarea mai multor trăsături robuste”.

„Dropout”-ul poate avea efectul opus, atunci când numărul de neuroni inactivi este prea mare. Subrețelele cu prea puțini neuroni nu vor fi capabile să extragă informații complexe din setul de date. Ca regulă generală, folosim coeficienți „dropout” mai mici sau egali cu 0.3 în cazul straturile convecționale, pentru a nu fi afectată capacitatea modelului de a interpreta imagini și coeficienți mai mari de 0.3 pentru straturile perceptron, ce nu au un rol la fel de important și sunt mai predispuși la codependență. ImageNet Classification with Deep Convolutional Neural Networks

O problemă similară cu cea descrisă anterior este cea a ponderilor prea mari. Această are același efect ca și codependența, deoarece, doar neuronii cu ponderi suficient de mari vor avea un impact semnificativ asupra activărilor stratului următor, iar rezultatul final va fi o rețea în care o bună parte din neuroni nu contribuie la clasificare. Soluția este reglarea funcției de cost prin termeni aditivi: L1 și L2. How does L1 and L2 regularization prevent overfitting?

Penalizările L1 și L2 nu sunt concepte ce au fost inventate de Krizhevsky și nici nu au fost folosite în arhitectura AlexNet. L1 și L2 sunt penalizări ce au fost folosite în cadrul regresiilor, iar mai apoi au fost preluate și integrate în rețelele neuronale. How does L1 and L2 regularization prevent overfitting?

L1, denumit și regresie Lasso, penalizează modelul proporțional cu suma valorilor absolute ale coeficienților. În urma penalizării, L1 face parametrii prea mari să devină 0. Ca urmare numărul de parametrii activi este redus, obținându-se un model mai simplu, dar care poate să extragă eficient informațiile relevante. Se recomandă L1 pentru imaginile ce conțin multe detalii ce nu influențează clasificarea, fiind extrase doar acele informații relevante. How does L1 and L2 regularization prevent overfitting?

L2, sau regresia Ridge, penalizează modelul în funcție de suma pătratelor greutăților. Față de L1, Ridge nu elimină complet ponderile, ci doar le reduce. Pentru clasificarea imaginilor complexe, ce

conțin multe detalii fine, dar semnificative, se recomandă folosirea penalizării L2. How does L1 and L2 regularization prevent overfitting?

Trebuie să fim atenți atunci când combinăm tehnicele discutate anterior. Mai jos avem câteva principii pentru alegerea modalității de reducere a overfitting-ului:

- Dacă folosim batch-uri suficient de mari, prima tehnică utilizată trebuie să fie normalizarea. Dropout: a simple way to prevent neural networks from overfitting
- Dacă normalizarea nu a fost suficientă pentru a îmbunătății performanțele modelului, se recomandă preprocesarea datelor, pentru că nu interacționează cu normalizarea. Dropout: a simple way to prevent neural networks from overfitting
- „Dropout”-ul se va utiliza atunci când nu avem batch-uri suficient de mari pentru a aplica normalizarea. Dropout: a simple way to prevent neural networks from overfitting
- Tehnica „dropout” poate fi folosit împreună cu normalizarea pentru a duce la performanțe mai bune, dar trebuie să ajustăm cu atenție valoarea coeficientului. Folosirea celor două tehnici în același timp face modelul mai sensibil la „dropout”, chiar dacă poate îmbunătății procesul de antrenare. Dropout: a simple way to prevent neural networks from overfitting
- Se recomandă utilizarea regularizărilor L1 și L2 în locul lui „dropout”-ului atunci când vrem să folosim normalizarea batch-urilor. Dropout: a simple way to prevent neural networks from overfitting
- Evităm folosirea simultană a penalizărilor de tipul L1 sau L2 cu „dropout”, atunci când putem. Dropout: a simple way to prevent neural networks from overfitting

## 2.2 U-Net

U-Net a fost prima arhitectură concepută pentru segmentarea imaginilor medicale. U-Net a fost introdus pentru prima dată în lucrarea „U-Net: Convolutional Networks for Biomedical Image Segmentation”(U-Net: Convolutional Networks for Biomedical Image Segmentation), scrisă de Olaf Ronneberger, Philipp Fischer, Thomas Brox. A fost prima rețea „complet conținutuală” din lume, ceea ce înseamnă că blocurile perceptron din capătul modelului, folosite pentru clasificare, au fost înlocuite cu conoluții. Doarece ultimul strat era unul conținutional, output-ul modelului era o imagine generată de acesta. Deși scopul original al U-Net-ului era segmentarea imaginilor medicale, U-Net-ul a

avut mai mult potențial decât puteau să-și imagineze creatorii acestuia. În prezent U-Net stă la baza modelelor de generare de imagini, cum ar fi DALL-E sau Midjourney, dar vom reveni la segmentare imaginilor medicale, pentru că acesta modul în care va fi utilizat în această lucrare.

## 2.2.1 Limitările segmentării manuale.

Segmentarea este procesul prin care imaginile sunt descompuse în părțile lor componente. Segmentarea imaginilor este utilă atunci când dorim evidențierea sau separarea unor zone de interes din imagine. O altă utilitate a segmentării este crearea de modele interactive 3D, prin conturarea obiectelor din imagini cu mai mult de două dimensiuni, cum ar fi cele obținute prin tomografie. În medicină, segmentarea imaginilor poate fi un pas important în stabilirea corectă a diagnosticului. Segmentarea imaginilor medicale poate fi utilă din mai multe motive. Image Segmentation

Un prim motiv este eliminarea porțiunilor care nu conțin informații relevante pentru diagnostic, permîțându-i medicului să se concentreze exclusiv pe zonele semnificative. De asemenea, anumite regiuni din imagine pot îngreuna procesul de diagnosticare. Un exemplu relevant este reprezentat de tomografe, în care creierul trebuie separat de craniu pentru a permite o analiză corectă a structurii sale. La limita dintre cele două, poate fi greu de determinat ce structuri aparțin craniului și care aparțin creierului, motiv pentru care medicii trasează contururile acestora cu precizie, pentru a evita omisiunea unor eventuale anomalii aflate la marginea dintre cele două structuri.

Un alt motiv important pentru utilizarea segmentării este necesitatea delimitării cu precizie a structurilor anatomici și patologice, întrucât prezența unei boli sau severitatea acesteia poate fi indicată de forma neregulată a acestor structuri, aşa cum este cazul tumorilor, caz discutat în subcapitolul anterior.

Segmentarea manuală este încă practicată în rândul medicilor și este un proces ce prezintă numeroase dezavantaje. În primul rând, nu orice medic poate realiza manual segmentarea imaginilor medicale. Ca imaginile să fie segmentate corect, medicul are nevoie de un nivel ridicat de experiență, precum și cunoștințe detaliate despre patologiile ce pot apărea la nivelul acelei structuri. Din cauza complexității structurilor anatomici, segmentarea corectă necesită ani de pregătire. Totuși, chiar și în cazul specialiștilor cu experiență și competențe avansate, nu există garanția unei delimitări corecte a structurilor din imagine.

Motivul pentru care un medic cu experiență poate segmenta incorect imaginea este prezența leziunilor ce sunt prea mici pentru a fi observate cu ochiul liber, sau prezența bolilor aflate în faza inițială, care nu produc modificări vizibile la nivelul structurii organului afectat. Dacă medicul nu poate detecta aceste anomalii, pacientul nu va putea fi diagnosticat corect.

Și nu în ultimul rând, segmentarea unei singure imagini poate dura de la câteva minute, până la câteva ore, în funcție de complexitatea structurii analizate. Un tomograf poate conține peste 100 de imagini (slice-uri), iar timpul pierdut pentru segmentarea unui singur tomograf putea fi utilizat pentru diagnosticarea și tratarea altor pacienți. Din cauza dezavantajelor explicate anterior, a fost necesară introducerea programelor de segmentare automată și semiautomată în clinici.

Programele de segmentare semiautomată sunt soluții ce realizează segmentarea parțială a imaginilor și îi oferă medicului unelte pentru segmentarea imaginilor. Nu sunt la fel de precise ca cele complet automate și este necesară intervenția medicului, care trebuie să corecteze mai apoi măștile generate de program. Pentru utilizarea unui astfel de program este încă necesar ca medicul să aibă suficientă experiență, însă gradul de experiență necesară este cu mult mai scăzut decât în cazul segmentării manuale.

Programele complet automate nu necesită intervenția directă a medicului și pot fi utilizate de către un medic aflat la începutul carierei. Segmentarea automată reduce timpul de segmentare la numai câteva secunde, oferindu-i mai mult timp medicului pentru diagnosticare și pentru tratarea altor pacienți.

Ambele tipuri de programe au potențialul de a reduce numărul cazurilor în care afecțiunile nu au fost detectate la timp. Eficiența acestor programe se datorează faptului că acestea pot analiza imaginile până la cea mai mică unitate a imaginii (pixeli). Totuși, eficiența programelor nu înseamnă că nu va mai fi necesară contribuția medicului, ci dimpotrivă. Chiar și în cazul segmentării automate, există posibilitatea ca marginile să necesite ajustări. De asemenea, astfel de programe nu au scopul de a diagnostica pacienții. Scopul lor este de a prelucra imaginile pentru a mări claritatea detaliilor din imagine, ceea ce va face imaginile mai ușor de interpretat de către medic.

Așa cum a fost detaliat în „Deep learning approaches to biomedical image segmentation”(Deep learning approaches to biomedical image segmentation), realizarea unui astfel de program poate fi o sarcină dificilă. Zgomotul de imagine, diferențele de contrast între regiuni și lipsa de consistență din setul de date pot afecta performanțele modelelor, mai ales în cazul celor care folosesc tehnici clasice de

învățare automată, cum ar fi vectorii de suport, regresiile, arborii de decizie sau clusterizările. Rețelele neuronale sunt cel mai puțin predispuse să fie afectate de structura setului de date, necesitând un număr minim de preprocesări pentru imaginile din setul de date. De asemenea, se preferă utilizarea rețelelor neuronale datorită posibilității de a accelera procesul de antrenare prin utilizarea plăcilor video.

### **2.2.2 Structura modelului**

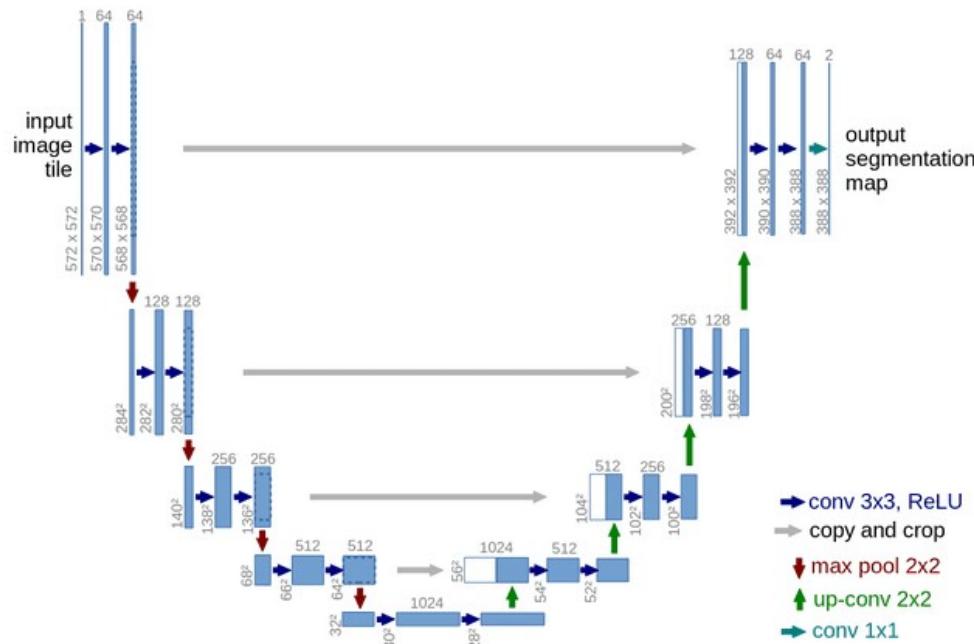


Figure 29: Arhitectura UNET

Numele de U-Net a fost ales din cauza formei de „U” a arhitecturii. U-Net este compus din 4 tipuri de blocuri:

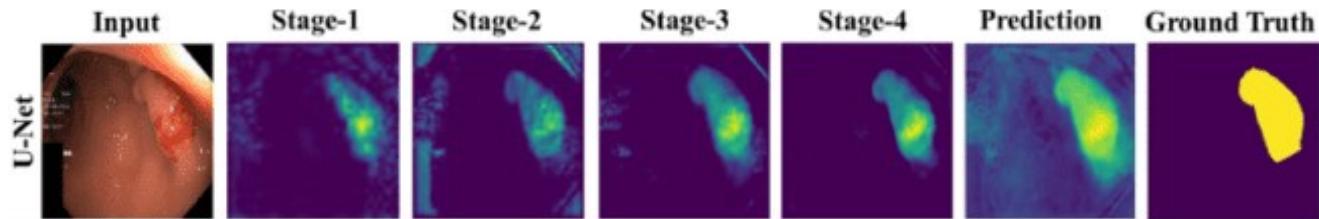
- Codificator (Encoder). În partea stângă a figurii 30 se află cele 4 codificatoare ale rețelei. Sunt formate din blocuri conoluționale clasice, similare cu cele din rețeaua AlexNet. Fiecare bloc conține două straturi conoluționale, ce au rolul de a extrage recunoaște componentele din imagine și un strat de subeșantionare, ce reduce mai apoi dimensiunea imaginilor. Sunt cele mai importante blocuri ale rețelei , fiind cele care se ocupă de extragerea informației necesare segmentării.U-Net: Convolutional Networks for BiomedicalImage Segmentation

- „Bottleneck”. Poartă denumirea de „bottleneck” pentru că este „cea mai îngustă zonă” a rețelei. Există un singur block „bottleneck” în rețea și se află la baza acesteia. Este porțiunea modelului cu cel mai mare nivel de compresie a datelor, fiind „cea mai îngustă zonă” a rețelei pentru că imaginile care ajung aici sunt de dimensiuni foarte mici, conținând doar cele mai relevante detalii din imagini. Rolul decodorului este de a scala imaginea pentru a putea fi pasată primului decodor, fiind componenta ce face legătura dintre partea ce se ocupă de compresia imaginilor și partea ce se ocupă de decompresie.U-Net: Convolutional Networks for BiomedicalImage Segmentation
- Decodorul (decoder). Are rolul de a utiliza informațiile extrase de codificatori pentru a construi imaginea finală ce va fi returnată în output-ul rețelei. Fiecare decodor conține 2 convoluții și un strat ce realizează scalarea imaginilor, pentru a inversa operația MaxPooling. Înainte de a trece prin decodor output-ul stratului anterior este concatenat cu activările codificatorului aflat la același nivel cu decodorul. În cea de-a doua convoluție a decodorului, cele două imagini vor fi îmbinate într-o singură. Îmbinarea reintroduce treptat în imagine informația pierdută în urma compresiei datelor. Figura 31 este un exemplu de pierdere a datelor prin compresie.U-Net: Convolutional Networks for BiomedicalImage Segmentation



*Figure 30: Pierderea fidelității imaginii prin compresie*

- Clasificator. Este compus dintr-o singură convoluție. Pentru a face convoluția să realizeze clasificarea pixelilor în funcție de regiunea din care fac parte, asupra ei vom aplica activări folosite în straturile de clasificare, cum ar fi Softmax sau Sigmoid.U-Net: Convolutional Networks for BiomedicalImage Segmentation



*Figure*

31: Activările unei retele U-Net

### 2.2.3 Implementarea arhitecturii.

Pentru a simplifica crearea rețelelor ce utilizează arhitectura U-Net și pentru a putea modifica într-un mod mai simplu configurația rețelei, am defenit următorul modul pentru implementarea arhitecturii U-Net.

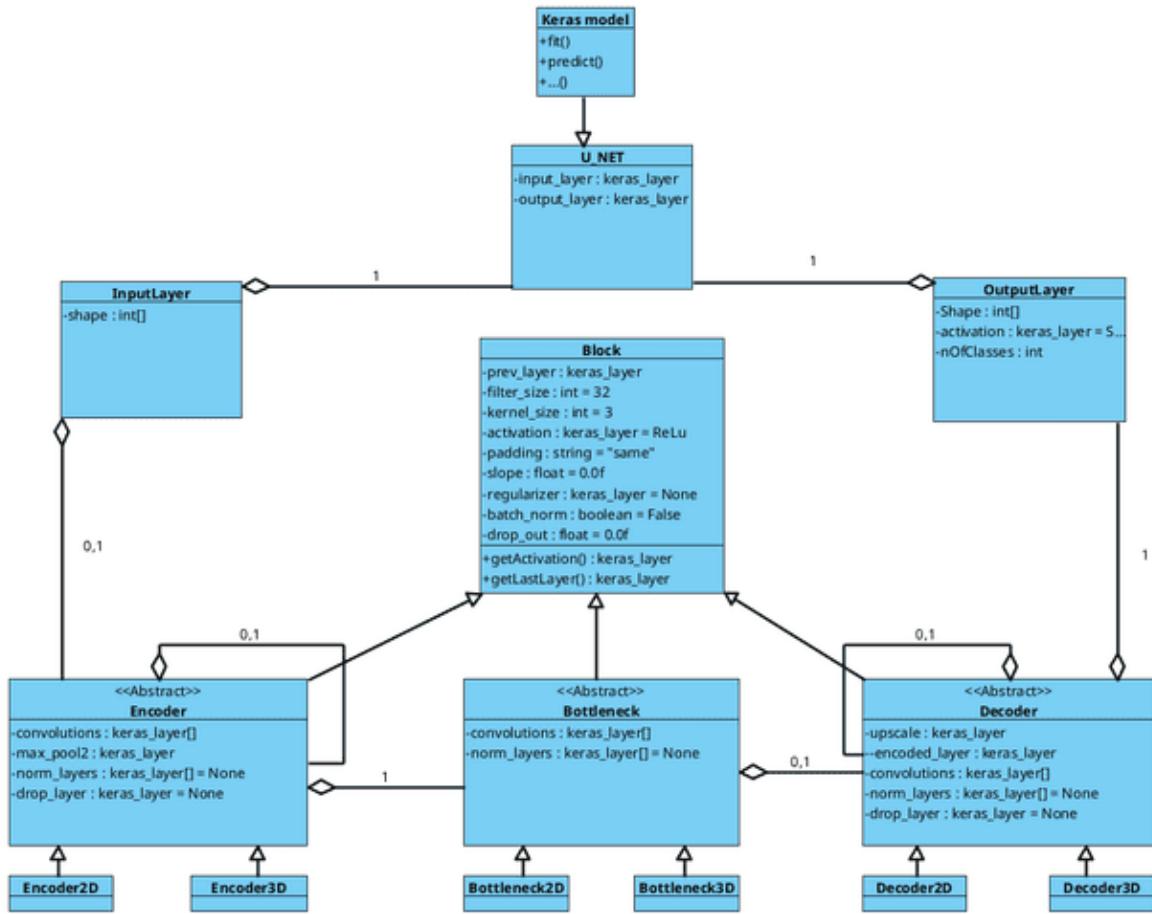


Figure 32: Diagrama claselor – Modul U-Net

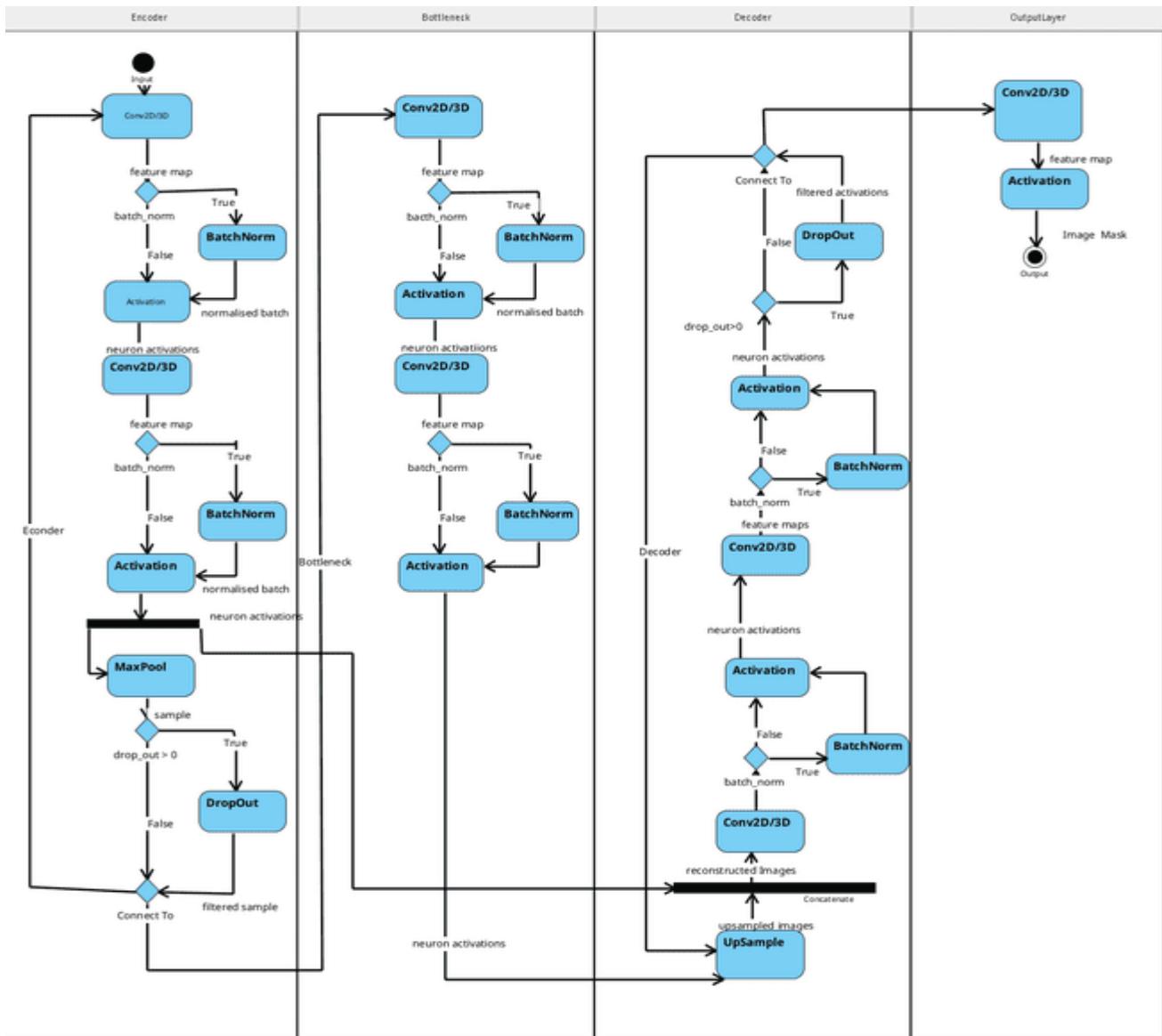


Figure 33: Diagramă de activitate – Modul U-Net

```
inputs = Input(shape=(240,155,1))

conv1,pool1 = EncoderBlock(inputs,filter_size=32,batch_normalization=True,activation='leaky_relu',slope=0.01)
conv2,pool2 = EncoderBlock(pool1,filter_size=64,batch_normalization=True,activation='leaky_relu',slope=0.01)
conv3,pool3 = EncoderBlock(pool2,filter_size=128,batch_normalization=True,activation='leaky_relu',slope=0.01)

neck = BottleneckBlock(pool3,filter_size=256,batch_normalization=True,activation='leaky_relu',slope=0.01)

up1 = AttentionDecoder(neck,conv3,filter_size=128,batch_normalization=True,activation='leaky_relu',slope=0.01)
up2 = AttentionDecoder(up1,conv2,filter_size=64,batch_normalization=True,activation='leaky_relu',slope=0.01)
up3 = AttentionDecoder(up2,conv1,filter_size=32,batch_normalization=True,activation='leaky_relu',slope=0.01)

outputs = OutputBlock(up3,num_of_classes=4,kernel_size=1,activation='softmax')

model = U_NET(inputs=inputs,outputs=outputs)
optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=[dice_coefficient,dice_coef_necrotic,dice_coef_edema,dice_coef_enhancing,dice_coef_background,combined_loss]
)
model.summary()
```

Figura 34: Utilizarea Pythonului în mediul Google Collab

În 2015, tot în cadrul concursului ILSVRC, a fost folosită prima rețea reziduală. Arhitectura, publicată în lucrarea „Deep Residual Learning for Image Recognition”. Lucrarea adresa dezavantajele rețelelor convecționale și oferea soluții pentru acestea. Modelul final, prezentat în lucrare, conținea 152 de convecționale și a obținut o acuratețe de de 80.6%, cu mult peste cea a lui AlexNet de 63% pentru același set de date. Deep Residual Learning for Image Recognition

### 2.3.1 Arhitectură ResNet

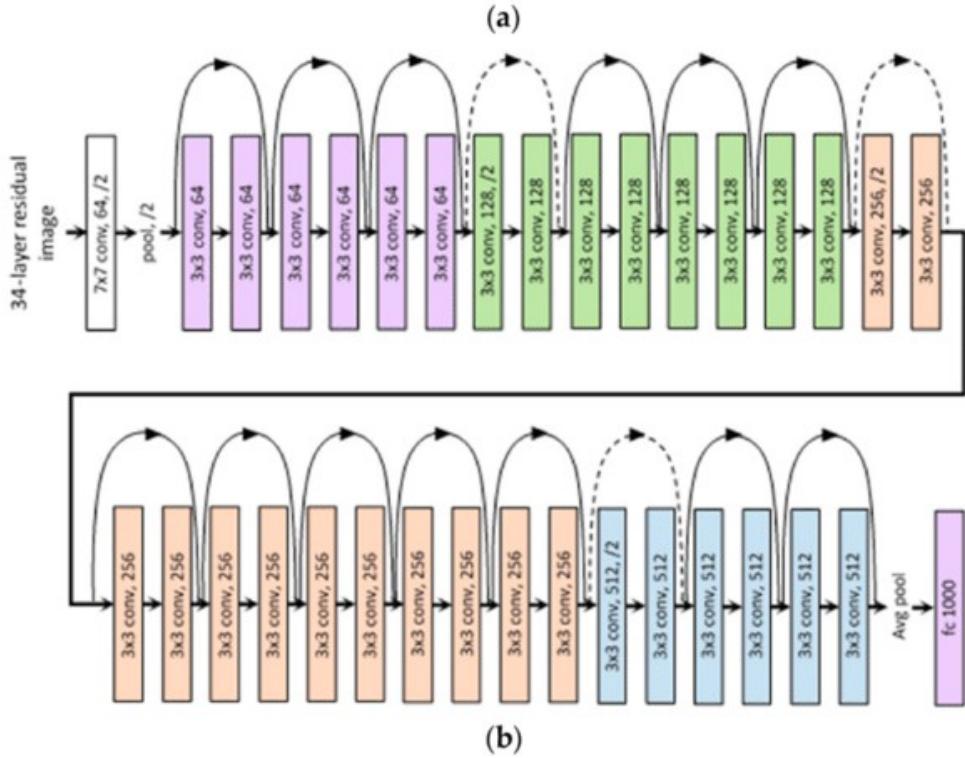


Figure 35: Arhitectură ResNet

Există două noi elemente importante:

- Stratul conoluțional de început. Primul strat al rețelei este o conoluție de 7x7. ResNet utilizează o conoluție mai mare pentru a detecta porțiunile din imagini ce fac parte din același obiect.
- Blocuri reziduale. ResNet rezolvă problema gradienților care dispar prin adaugarea unei conexiuni directe cu activările anterioare. Activările anterioare, ce au fost pierdute din cauza greutăților prea mici sau prea mari, vor fi reintroduse înapoi în rețea prin adunarea acestora la straturile superioare. Există două tipuri de astfel de straturi, ce pot fi văzute în figura 37. Cea de-a doua variantă este o metodă subtilă de-a realiza compresia datelor. Prima conoluție de 1x1 îmbină rezultatele filtrelor aplicate în stratul anterior, ca mai apoi să putem utiliza o conoluție cu mai puțini parametrii pentru extracție. Ultima conoluție de 1x1 are rolul de a activaile pentru a obține numărul inițial de filtre.

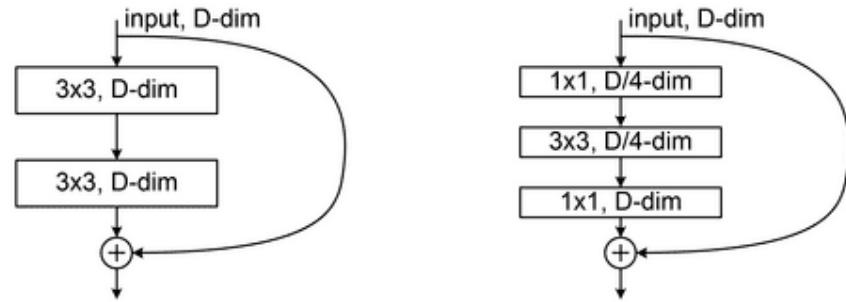
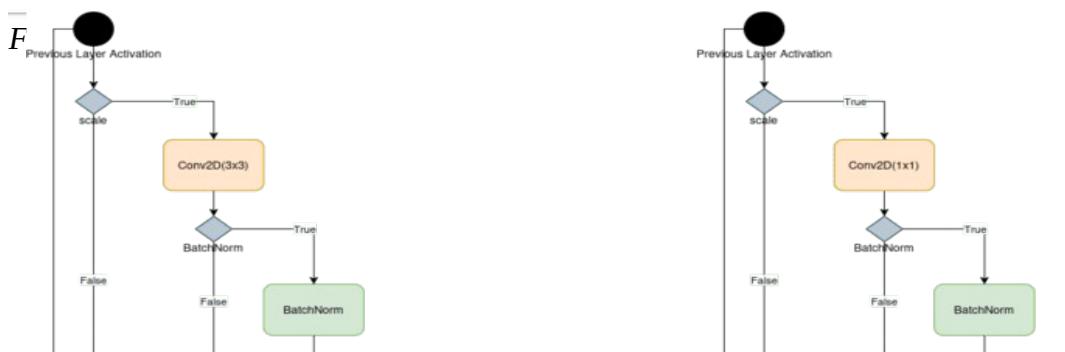
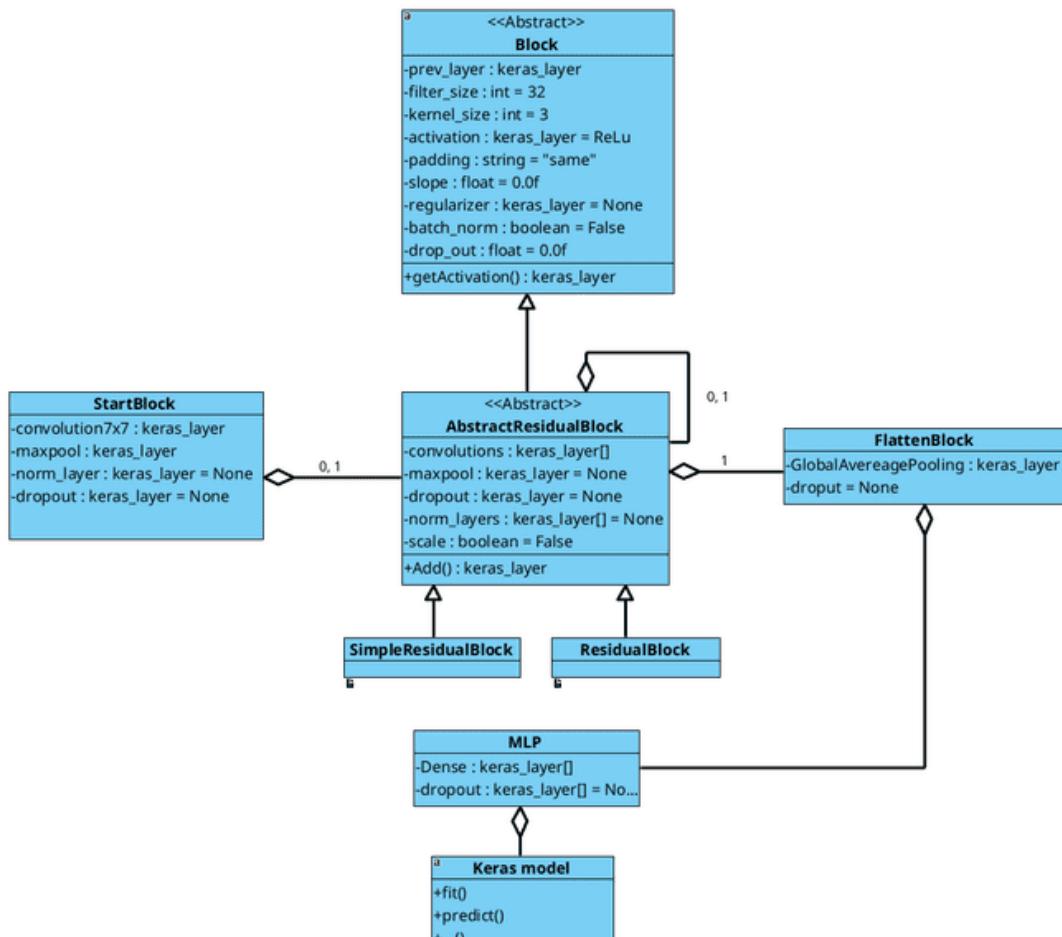


Figure 36: Blocuri reziduale

### 2.3.2 Implementarea arhitecturii

Mai jos avem diagrama de clase pentru modulul ce va fi utilizat:



Deși nu fac parte din arhitectura originală, am oferit posibilitatea implementării straturilor de dropout, normalizare, pooling și posibilitatea de-a alege dacă convoluțiile anterioare vor fi sau nu scalate.

```
In [5]:  
  
input = Input(shape=(256, 256, 3))  
start = StartBlock(input, filter_size=64, kernel_size=(7, 7), strides=2,batch_normalization=True)  
  
residual1 = SimpleResidualBlock(start, filter_size=64, kernel_size=(3, 3), padding="same",  
                                batch_normalization=True)  
residual2 = SimpleResidualBlock(residual1, filter_size=64, kernel_size=(3, 3), padding="same",  
                                batch_normalization=True)  
  
residual3 = SimpleResidualBlock(residual2, filter_size=128, kernel_size=(3, 3), padding="same",scale=True,  
                                batch_normalization=True)  
residual4 = SimpleResidualBlock(residual3, filter_size=128, kernel_size=(3, 3), padding="same",  
                                batch_normalization=True)  
  
residual5 = SimpleResidualBlock(residual4, filter_size=256, kernel_size=(3, 3), padding="same",scale=True,  
                                batch_normalization=True)  
residual6 = SimpleResidualBlock(residual5, filter_size=256, kernel_size=(3, 3), padding="same",  
                                batch_normalization=True)  
  
residual7 = SimpleResidualBlock(residual6, filter_size=512, kernel_size=(3, 3), padding="same",scale=True)  
residual8 = SimpleResidualBlock(residual7, filter_size=512, kernel_size=(3, 3), padding="same",  
                                batch_normalization=True)  
  
flatten = FlattenResidualBlock(residual8,drop_out=0.3)  
mlp = Dense(256, activation='relu')(flatten)  
mlp = Dropout(0.5)(mlp)  
output = Dense(4, activation='softmax')(mlp)  
  
model = Model(inputs=input, outputs=output)|  
  
model.compile(  
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),  
    loss='categorical_crossentropy',  
    metrics=['accuracy', tf.keras.metrics.AUC()]  
)  
  
model.summary()
```

Figure 39: Utilizare în Jupyter Notebooks - ResNet

## 2.4 VGG

Modelul VGG (Visual Geometry Group) este o arhitectură de rețea neuronală convoluçãoanală (CNN) propusă de cercetătorii de la Universitatea Oxford în cadrul grupului Visual Geometry Group. A fost prezentată pentru prima dată în lucrarea "Very Deep Convolutional Networks for Large-Scale Image Recognition" de Simonyan și Zisserman în 2014. Scopul principal al acestui model a fost clasificarea imaginilor în cadrul competiției ImageNet Large Scale Visual Recognition Challenge (ILSVRC). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

VGG este caracterizat prin utilizarea repetitivă a strukturilor conoluționale cu filtre de dimensiune mică ( $3 \times 3$ ) și straturi de pooling ( $2 \times 2$ ), urmate de straturi complet conectate (fully connected) în partea finală. O trăsătură distinctivă este adâncimea mare a rețelei, versiunile populare fiind VGG-16 (16 straturi cu greutăți antrenabile) și VGG-19.VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Datorită folosirii mai multor straturi convoluționale de același tip, rețeaua poate detecta mai ușor straturile ierarhice din imagini și poate extrage mai multe trăsături aflate la același nivel.

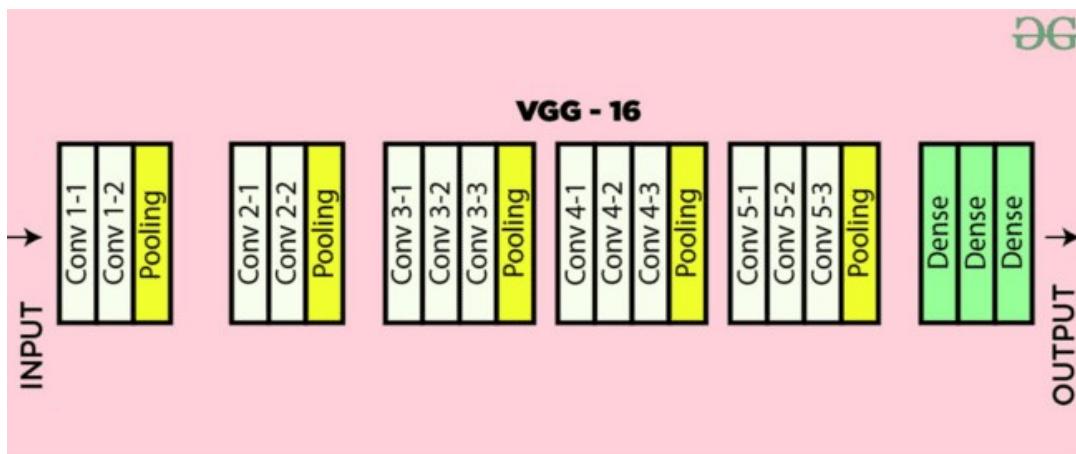


Figure 40: Arhitectură VGG16

**Pentru că nu există prea multe particularități față de rețelele conoluționale obișnuite, definirea unui modul pentru implementarea rețelelor din clasa VGG nu va fi necesară. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION**

## 2.5. Clasificare

U-Net e singurul model ce poate fi utilizat pentru segmentarea imaginilor, celelalte vor fi utilizate pentru clasificare.

Atât în cazul segmentării cât și în cazul diagnosticării , doctorii trebuie să aloce foarte mult timp pentru a analiza o singură imagine dintr-un volum uriaș de date , iar datorită timpului limitat, a lipsei de experiență sau a condițiilor aflate în fază inițială ce pot fi greu de detectat, mulți pacienți vor fi diagnosticați greșiti. Deși evaluarea de către un medic ce a acumulat suficientă experiență de-a lungul timpului reprezintă cea mai sigură variantă pentru diagnosticare celor mai multe condiții , există mulți medici ce nu au dobândit încă experiență necesară. Cu un set suficient de date , un AI poate căpăta experiență necesară în doar câteva ore și mai apoi poate fi antrenat continu cu noi imagini. Există condiții ce nu pot fi diagnosticate ușor de către oameni, de exemplu în Statele Unite 50% dintre femeile care fac în mod regulat ecografii mamare vor primi un diagnostic fals pozitiv pentru cancer de sân , iar 12% dintre cazurile reale sunt nediagnosticate lunar. În cazul acesta s-a observat că modelele ce au fost antrenate folosind ecografii mamare au o acuratețe de diagnosticare cu 10% mai bună decât un medic. Pentru clasificare se folosesc rețele neuronale convoluționale , doarece pot înțelege mai bine tipurile din imagini și activările lor pot fi extrase pentru a fi interpretate .Activările îi vor spune medicului unde se află anomalie și cât de gravă e, astfel scutește timpul necesar unei analize complete asupra imaginii și reduce cazurile în care leziunile nu sunt detectate de medic.

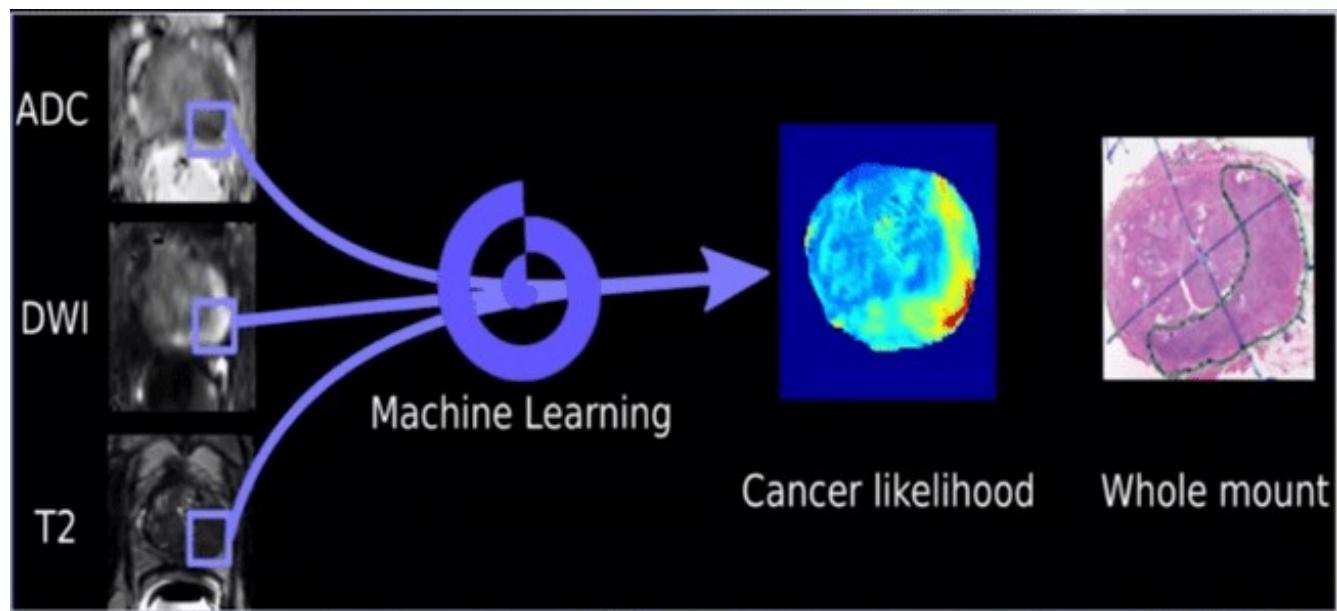


Figure 41: Activările unei rețele convoluționale

## 2.6. Tehnologii

Datorită simplității și a numărului mare de modele, am ales folosire limbajului Python. Mai jos se află descrierile librăriilor ce au fost folosite pentru realizarea lucrării.

- **Pandas.** Pandas este o bibliotecă Python utilizată pentru manipularea și analiza datelor. Aceasta oferă structuri de date eficiente, precum DataFrame, care permite manipularea datelor tabelare. Este frecvent utilizată pentru importul, curățarea, transformarea și analiza datelor numerice și categoriale, fiind o componentă esențială în pregătirea datelor pentru modelele de învățare automata.
- **TensorFlow.** TensorFlow este o platformă open-source dezvoltată de Google pentru construirea și antrenarea modelelor de învățare automata. Este scalabilă și permite rularea pe CPU, GPU și TPU, fiind potrivită pentru sarcini complexe, precum recunoașterea imaginilor, analiza secvențelor și predicția medicală asistată de AI.
- **Keras.** Keras este o interfață de nivel încalt pentru TensorFlow, concepută pentru a facilita definirea rapidă a modelelor de rețele neuronale. Suportă modele de tip Sequential și Functional API, fiind ideală pentru prototipare rapidă și experimente AI, inclusiv pentru clasificarea imaginilor medicale și segmentare.
- **NumPy.** NumPy este o bibliotecă fundamentală pentru calcule numerice în Python. Oferă suport pentru array-uri multidimensionale și un set bogat de funcții matematice. Este utilizată în fundal de alte biblioteci AI (inclusiv Pandas, OpenCV și TensorFlow) pentru a efectua calcule eficiente la nivel de matrici și tensori.
- **OpenCV.** OpenCV este o bibliotecă pentru procesarea imaginilor și viziune computerizată. Permite aplicarea de filtre, detectarea marginilor, transformări geometrice, și segmentarea obiectelor. Este des folosită în domeniul medical pentru preprocesarea imaginilor CT, RMN sau radiografii, în vederea analizelor automate.
- **Nibabel.** Nibabel este o bibliotecă specializată în lucrul cu imagini medicale 3D în formate standard precum NIfTI sau Analyze. Oferă funcții pentru încărcarea, salvarea și manipularea

volumelor de imagini RMN, fiind un instrument esențial în segmentarea și analiza automata a datelor volumetrice.

- Matplotlib. Matplotlib este o bibliotecă de vizualizare a datelor, folosită pentru a crea grafice și imagini statice sau interactive. Este utilizată pentru reprezentarea vizuală a rezultatelor modelelor AI, curbelor de antrenare, distribuțiilor de date și afișarea imaginilor medicale segmentate.
- Scikit-learn (sklearn). Scikit-learn este o bibliotecă Python ce oferă implementări ale celor mai cunoscuți algoritmi de învățare automata, precum regresie, clasificare, clustering, selecția trăsăturilor și validare. Este ideală pentru dezvoltarea de modele predictive în domeniul medical, cum ar fi predicția riscului de boli sau clasificarea pacienților.
- Scikit-image (skimage). Scikit-image este o extensie a Scikit-learn pentru procesarea imaginilor. Include algoritmi pentru filtrare, segmentare, transformări și extragerea caracteristicilor geometrice. Este des folosită în analiza imaginilor RMN și CT, fiind complementară cu OpenCV și Nibabel în lucrul cu imagini medicale.

## **Capitolul 3. Antrenarea rețelelor folosind imagini medicale**

### **3.1 Leucemie**

Leucemia este un tip de cancer care afectează capacitatea celulelor hematopoietice din măduva osoasă de a produce celule sanguine normale și duce la creșterea excesivă a numărului de celule anormale în sânge. Apariția unui număr atât de mare de celule anormale poate afecta sistemul imunitar sau poate duce la distrugerea țesuturilor organelor, mai ales atunci când și producția de celule roșii este afectată, deoarece oxigenul nu mai poate fi transportat eficient. În general, mutația cauzează dezvoltarea incompletă a celulelor albe. Cu timpul, numărul acestora poate ajunge să-l depășească pe cel al globulelor dezvoltate complet. Automated detection of leukemia in blood microscopic images

using image processing techniques and unique features: Cell count and area ratio Dictionar de afectiuni, simptome, investigatii si tratamente

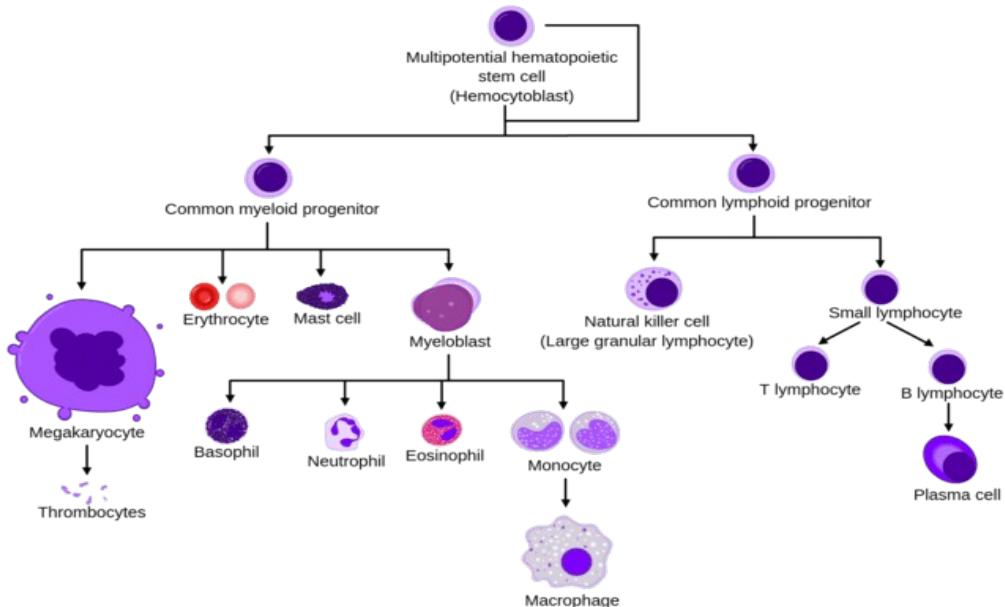


Figure 42: Formarea celulelor sanguine

Leucemia pornește din măduva osoasă, aflată în interiorul oaselor, unde sunt produse celulele stem hematopoietice (hemo = sânge, poietice = creatoare), care se pot transforma fie în celule limfoide, ce vor începe procesul de specializare pentru a deveni limfocite (un tip de globule albe), fie în celule mieloide, care se vor transforma în globule roșii, trombocite sau alte tipuri de celule albe.

Cauza leucemiei este apariția mutațiilor în procesul de formare al acestor celule. În funcție de tipul de celule afectate, leucemia poate fi limfoidă sau mieloidă. Odată cu apariția leucemiei, celulele afectate încep să se reproducă necontrolat, ocupând spațiul necesar formării celulelor sănătoase. În funcție de viteza cu care aceste celule anormale se înmulțesc, leucemia poate fi clasificată drept acută (cu evoluție rapidă) sau cronică (cu evoluție lentă). Automated detection of leukemia in blood microscopic images using image processing techniques and unique features: Cell count and area ratio Dictionar de afectiuni, simptome, investigatii si tratamente

Leucemia acută limfatică este cea mai frecventă formă de cancer la copii și reprezintă, totodată, cea mai întâlnită formă de leucemie. Aproximativ 75% dintre cazuri de leucemie sunt de acest tip. A fast and efficient CNN model for B-ALL diagnosis and its subtypes classification using peripheral blood smear images

Această afecțiune poate avea cauze genetice, de exemplu, copiii care suferă de sindromul Down sau de boli ce afectează sistemul imunitar prezintă un risc mai mare de a dezvolta leucemie. Cu toate acestea, în multe cazuri, apariția leucemiei nu este determinată de factori genetici. Mai degrabă, ea este corelată cu sensibilitatea crescută a copiilor la factorii de mediu, apariția leucemiei fiind favorizată de expunerea la radiații, solvenți, substanțe toxice similare, poluare sau fumul rezultat din arderea țigărilor.

A fast and efficient CNN model for B-ALL diagnosis and its subtypes classification using peripheral blood smear images

Acute Lymphoblastic Leukemia (ALL)

În ciuda incidenței ridicate, rata de supraviețuire în rândul copiilor este de aproximativ 90%, deoarece leucemia limfatică acută determină mutații genetice mai ușoare comparativ cu alte forme de cancer, fiind mai ușor de tratat prin chimioterapie sau transplant medular. Rata ridicată de supraviețuire se datorează și faptului că, de obicei, copiii nu suferă de alte afecțiuni cronice, cum ar fi diabetul sau bolile cardiovasculare, care ar putea complica tratamentul leucemiei. În cazul adulților, incidența cazurilor de leucemie limfatică acută este mult mai mică, dar și rata de supraviețuire. În rândul adulților rate de supraviețuire este de sub 50%

A fast and efficient CNN model for B-ALL diagnosis and its subtypes classification using peripheral blood smear images

Acute Lymphoblastic Leukemia (ALL)

Adulții atribuie adesea simptomele asociate leucemiei, cum ar fi obosalea și infecțiile cronice, altor cauze, precum stresul sau epuizarea, și caută asistență medicală doar atunci când este prea târziu. De asemenea, în cazul în care sunt prezente și alte afecțiuni, sistemul imunitar este slăbit și nu poate opri dezvoltarea altor boli. Deși leucemia este tratabilă, este foarte important să putem identifica la timp cazurile de îmbolnăvire.

Leukemia

În general, leucemia nu formează tumori vizibile și nu poate fi detectată prin tomografii sau radiografii, deoarece măduva se află în interiorul oaselor. În schimb, se pot folosi mostre de sânge sau de măduvă osoasă pentru a confirma prezența bolii.

Leukemia

Pentru detectarea leucemiei am conceput două programe. Unul ce a fost specializat în segmentarea monștrelor de sânge și celălalt în clasificarea lor.

### 3.1.1. Set de date

Setul de datele utilizate a fost publicat de spitalul Teheran și conține 3256 de imagini cu monștre de sânge de la 3256 de pacienți alături de etichetele acestora și imagini segmentate manual pentru antrenament. Clasele setului de date sunt:

- „Benign”. Caracterizează pacienții ce nu prezintă afecțiuni
- „Early”. Sunt cazurile ce au fost detectate în stagile inițiale ale bolii, înainte ca celulele anormale să înceapă înlocuirea celor sănătoase.
- „Pro-B/T ALL” și „Pre-B/T ALL” sunt stadii ale leucemiei limfoblastice acute (ALL – *Acute Lymphoblastic Leukemia*) asociate cu diferite etape de dezvoltare a celulelor limfoide.
  - Clasa „Pro-B/T ALL”, cunoscută și sub denumirea de „Progenitor Acute Lymphoblastic Leukemia”, se referă la cazurile în care mutația a avut loc la nivelul celulelor stem parțial diferențiate, care urmează să devină limfocite (celule limfoide). Este o formă mai agresivă și mai dificil de tratat, deoarece aceste celule se înmulțesc rapid și necontrolat.
  - „Pre-B/T ALL” sau „Precursor Acute Lymphoblastic Leukemia” indică faptul că mutația s-a produs în limfocitele aflate într-un stadiu mai avansat de dezvoltare – ele au devenit leucocite imature, dar nu s-au diferențiat complet în celule T sau B. Deoarece mutația apare într-un stadiu aproape matur, boala are de obicei o evoluție mai lentă și poate fi mai ușor de controlat decât forma Pro-B/T.

### **3.1.2 Modelul pentru segmentare. Rezultate inițiale**

Pentru segmentare am optat pentru o variantă mai simplă a arhitecturii U-Net, cu doar 3 nivele, și nu am aplicat metode de preprocesare. În schimb am utilizat straturi pentru normalizarea datelor pentru a folosit subseturi de simenii mai mari, în cazul acesta de 64 de imagini. Pentru diagnosticarea monștrelor de sânge vrem să separăm leucocitele de restul imaginii, pentru a putea vedea concret forma, mărimea și numărul acestora. Deoarece avem o singură regiune de interes, leucocitele, vom utiliza activarea Sigmoid pentru ultima conoluție a rețelei. Modelul inițial se poate vedea în figura 42.

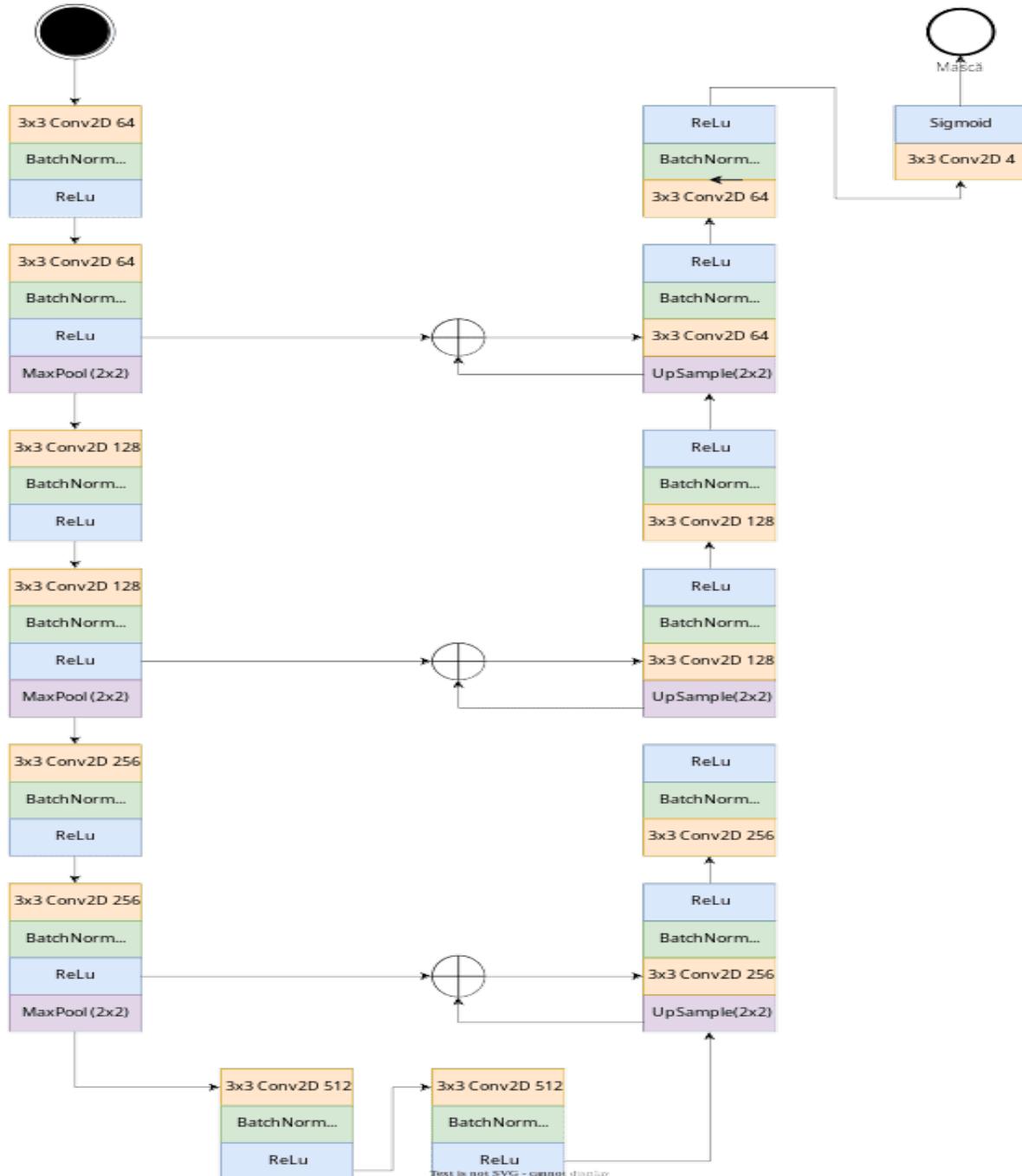


Figure 43: Diagrama Semnetare Leucemie - I

Înainte de antrenare trebuie să ne asigurăm că datele inițiale se află în intervalul [0, 1], pentru a nu avea greutăți prea mari. De asemenea trebuie ca toate imaginile să aibă aceeași mărime, pentru că

```

def load_and_preprocess_image(input_path, output_path, target_size=(256, 256), img_format="png", grayscale=False):
    input_image = tf.io.read_file(input_path)
    output_image = tf.io.read_file(output_path)

    input_image = decoder.get(img_format, "png")(input_image, channels=3)
    output_image = decoder.get(img_format, "png")(output_image, channels=3)

    input_image = tf.image.resize(input_image, target_size)
    output_image = tf.image.resize(output_image, target_size)

    if grayscale:
        input_image = tf.image.rgb_to_grayscale(input_image)
        output_image = tf.image.rgb_to_grayscale(output_image)

    input_image = tf.cast(input_image, tf.float32) / 255.0
    output_image = tf.cast(output_image, tf.float32) / 255.0

    return input_image, output_image

```

Figure 44: Scalarea și încărcarea imaginilor

```

In [5]:
#Building the model
inputs = Input(shape=(224,224,1))

conv1,pool1 = EncoderBlock(inputs,filter_size=64,batch_normalization=True)
conv2,pool2 = EncoderBlock(pool1,filter_size=128,batch_normalization=True)
conv3,pool3 = EncoderBlock(pool2,filter_size=256,batch_normalization=True)

neck = BottleneckBlock(pool3,filter_size=512,batch_normalization=True)

up1 = DecoderBlock(neck,conv3,filter_size=256,batch_normalization=True)
up2 = DecoderBlock(up1,conv2,filter_size=128,batch_normalization=True)
up3 = DecoderBlock(up2,conv1,filter_size=64,batch_normalization=True)

outputs = OutputBlock(up3)

model = U_NET(inputs=inputs,outputs=outputs)
model.summary()

```

Figure 45: Modelul implementat în cod

Pentru antrenarea modelului au fost necesare 20 de epoci de antrenament. Modelul a obținut pe setul de antrenament un cost de 0.08 și o acuratețe a pixelilor de 97%, dar pentru a decide dacă aceste metriki chiar sunt semnificative ne vom uita la modul în care modelul segmentează imagile.

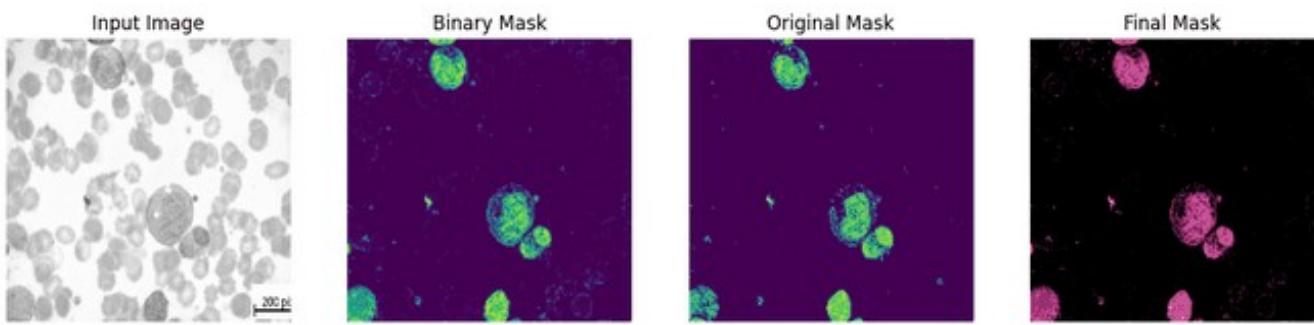


Figure 48: Leucemie - ImgSeg3.png

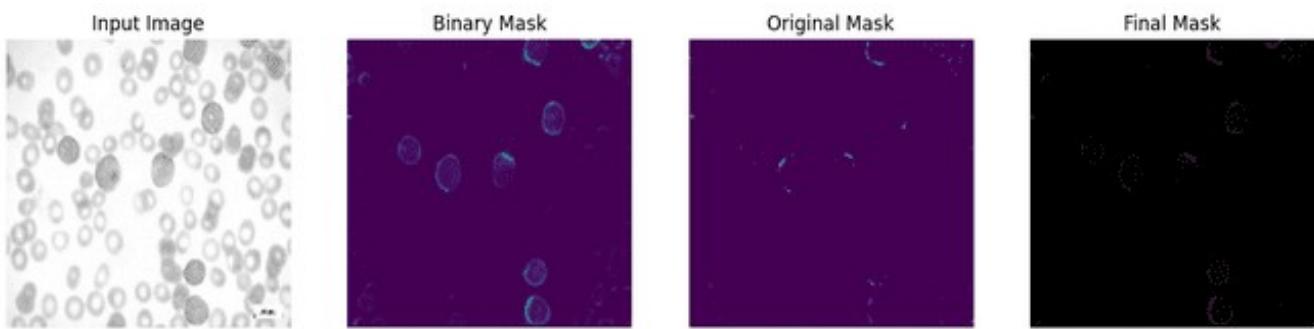


Figure 49: Leucemie – ImgSeg4.png

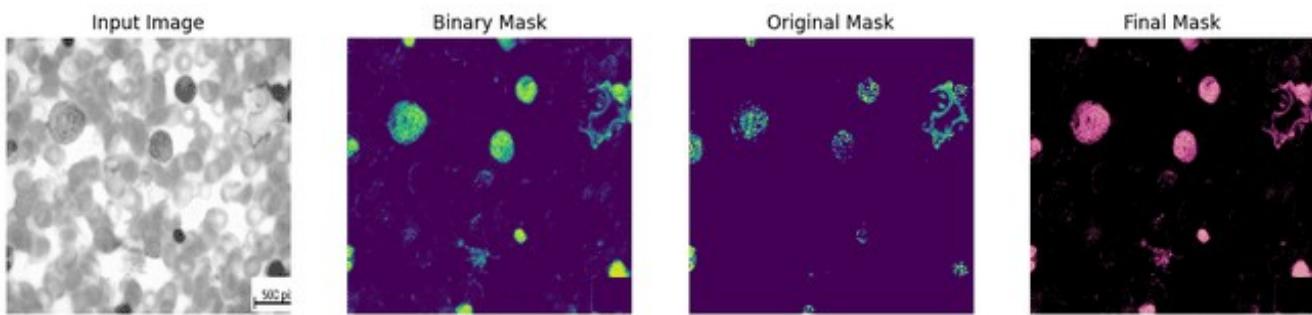


Figure 50: Leucemie - ImgSeg5.png

Cu mici excepții, pentru primele imagini pare că nu există diferențe semnificative între predicții și măștile originale, modelul reușind să extragă informațiile din setul de date. Modelul pare să poată elimina celulele roșii și celelalte tipuri de leucocite.

Dacă imaginile sunt clare, segmentările vor fi suficient de bune pentru a putea fi interpretate de către medici, dar nu toate imaginile au același nivel de claritate. Pentru acest tip de imagini, zgomotul poate fi introdus ușor de lumină inegală sau de deteriorarea/contaminarea probei din cauza păstrării necorespunzătoare. Indiferent de cauză, imaginile afectate nu vor mai putea fi interpretate nici de medici, nici de program.

Programul curent nu știe să trateze corect zgomotul din imagini, de aceea multe informații nerelevante vor fi utilizate în procesul de segmentare. Dacă astfel de imagini ajung în setul de date, procesul de antrenare a modelului va fi afectat. Am putea elimina imaginile ce prezintă zgomot de fundal din setul de date, ceea ce va duce la rezultate mai bune. Din păcate, aşa cum menționat mai devreme, zgomotul de fundal poate fi introdus ușor în monstrele de sânge, imaginile afectate nefiind cu adevărat outlieri. Dacă antrenăm rețeaua doar pe imagini neafectate de zgomot, rezultatele vor fi mai bune, dar nu vor fi reprezentative.

Figura 51 este un exemplu în care modelul nu a reușit să eliminate complet informația irelevantă, aceasta fiind prezente în masca de segmentare. În ultima imagine, zgomotul pare să fie cauzat de artefakte de lumină (iluminare neuniformă) sau de reglarea incorectă a aparaturii.

### 3.1.3 Preprocesarea imaginilor. Rezultatele după preprocesare

Vom aplica transformări ce au scopul de a reduce zgomotul din imagini. Pentru că zgomotul poate fi cauzat de reziduri din monstre sau de către iluminarea neadecvată, am ales următoarele transformări.

- **Eroziune și Dilatare.** Sunt două tipuri de transformări morfologice ( schimbă formă ).
  - Eroziunea separă obiecte din imagine prim reducerea marginilor acestora. În cazul imaginilor alb-negru, marginile sunt eliminate prin transformarea tuturor pixelilor ce nu au doar vechini de același tip în 0. În cazul imaginilor color, procesul este puțin mai complicat. Canelele RGB sunt descompuse în canale de tipul HSV (hue, saturation, value). Valuarea fiecărui pixel va fi înlocuită cu cea mai mică valoare din vecinătatea acestuia, în timp ce canale H (Nuanța) și S (Saturația) rămân nemodificate. După ce am schimbat valoarea pentru fiecare pixel, vom converti poza înapoi la RGB. Morphological Transformations
  - Dilatarea este procesul prin care marginile obiectelor vor fi reconstruite. Algoritmul de dilatare este similar, dar de data aceasta vom alege valorile maxime. Nu există riscul ca obiectele din imagine să fie intersectate iar după aplicarea dilatării, deoarece eroziunea a eliminat deja zgomotul care a cauzat îmbinarea obiectelor. În schimb, poate duce totuși la distorsionarea structurilor anatomici, ceea ce nu ar fi ideal în cazul nostru, dar s-ar putea să fie ce-a mai bună metodă de a elimina zgomotul de imagine. Morphological Transformations
- **Filtrarea imaginilor pe baza medianei.** Este o metodă nelineară de filtrare utilizată des în reducerea zgomotului din semanale, imagini și sunete. Atunci când elimină zgomotul, metoda poate produce distorsionarea imaginilor, dar duce la păstrarea marginilor obiectelor din imagini, figurile 52 și 53. Metoda poate fi aplicată imaginilor pentru că trasarea maginilor este obiectul principal al segmentării, de aceea metoda nu va duce la scăderea performanței. Filtrarea este aplicată folosind o conoluție ce înlocuiește valoarea pixelului din centrul matricei cu mediana pixelilor din vecinătate. Pentru setul nostru de date, filtrarea imaginii pe baza medianei ne ajută să eliminăm zgomotul apărut în urma contaminării monstrelor de sânge.Median Filter



Figure 51: Efecte - Median

Figure 52: Efecte Median Blur II

- Ajustarea contrastului imaginii. Primele metode sunt mult mai eficiente în tratarea zgomotului, dar ajustarea contrastului este cea mai simplă metodă pentru reglarea artefactelor de lumină.

Mai jos avem rezultate după aplicarea filtrelor:

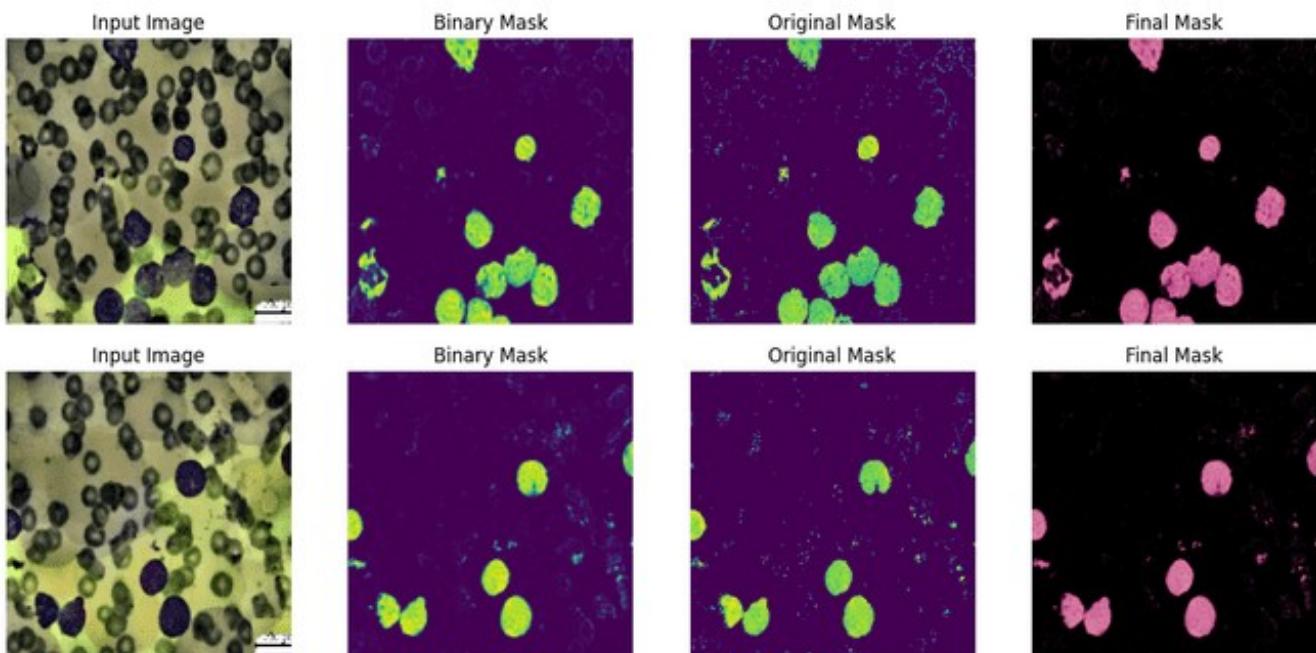


Figure 53: Leucemie - ImgSeg6.png

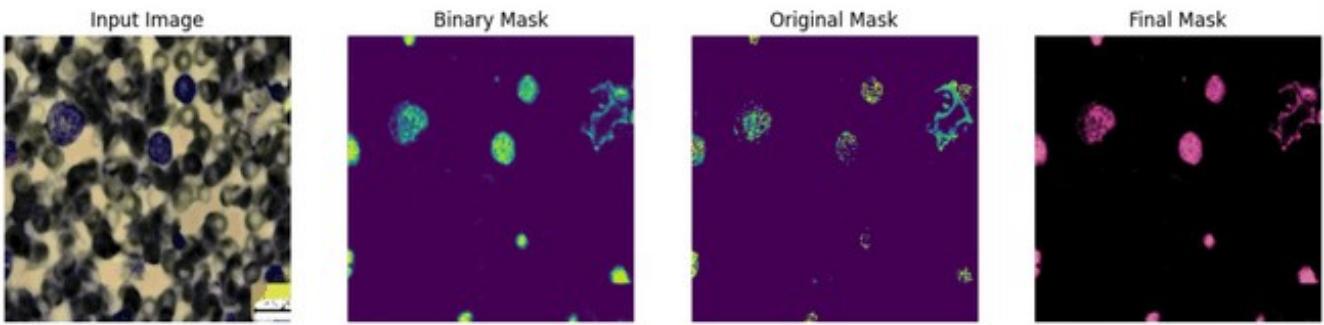
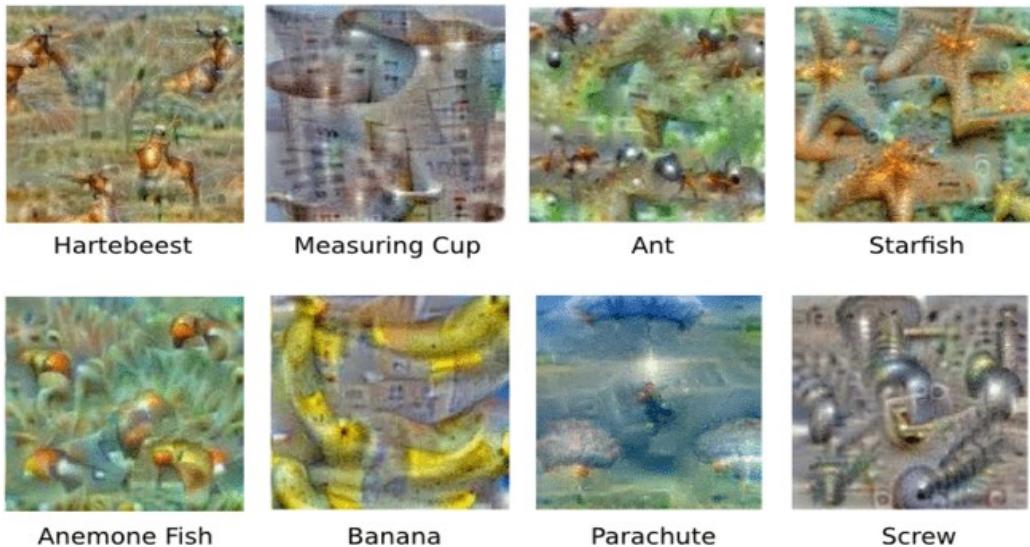


Figure 54: Leucemie - ImgSeg5.png(1)

Modelul elimină mult mai bine zgomotul din imagine, dar blurează prea mult porțiunile din imagine ce au dimensiuni mai mici. O soluție ar fi utilizarea imaginii inițiale pentru a corecta masca, dar o soluție și mai simplă ar fi să folosim ambele rețele în același timp sau să obținem masca finală ca media acestora (Ensemble Model).

La prima vedere, imaginile preprocesate par mult mai zgomotoase și cu singuranță nu sunt mai ușor de interpretat de către ochiul uman. Cu toate acestea, pe alocuri, modelul obține rezultate mai bune decât cel anterior. Preprocesarea nu are întotdeauna rolul de a face imaginile mai ușor de interpretat de către om. Deși imaginea conține mai puține detalii „vizuale” în sens uman (nuante, fundaluri sau culori), acesta conține toate caracteristicile cu adevărat utile pentru segmentarea, ba chiar le amplifică. Inteligența artificială nu vede imaginile în același mod le vedem noi, nu caută context sau semnificație în date, pentru că poate vedea doar numere.

Există o tehnică pentru generarea imaginilor sintetice ce maximizează activările pentru o anumită clasă. Tehnica se numește DeepDream, deoarece imaginile par a fi parte a unui vis. Atunci când tehnica a fost aplicată asupra rețelei AlexNet au fost generate imaginile din figura 56. DeepDream: Reinvent and Applied Research The moment we stopped understanding AI [AlexNet]



*Figure 55: DeepDream - AlexNet*

Putem observa multe caracteristici ale obiectelor concrete, dar sunt de parte de-a se asemăna cu obiectele reale. Deși imaginile prezentate par abstracte, toate trăsăturile obiectelor necesare recunoașterea lor se află în acele imagini.

### **3.1.4 Prima rețea pentru clasificarea imaginilor**

Pentru clasificare am început cu un model mai simplu, cu un număr mic de straturi conoluționale. Am folosit subseturi cu 32 de imagini, de aceea am introdus straturi pentru normalizarea batch-urilor, dar au fost insuficiente pentru a prevenii ovefitting-ul. Adăugarea de „dropout”-uri cu valori mici pentru a remedia acestă problemă.

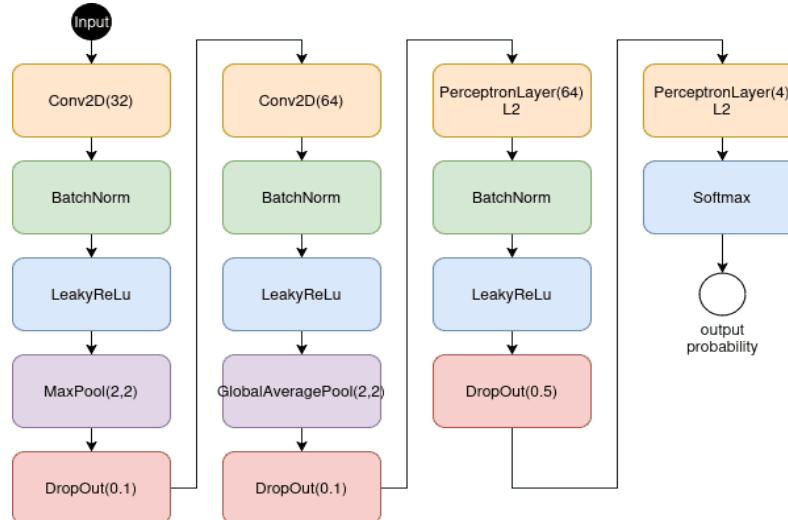


Figure 56: Leucemie - Prima Retea pentru clasificare

```

<model = Sequential([
    Input(shape=(224, 224, 3)),
    data_augmentation,
    Conv2D(filters=32, kernel_size=(3, 3), padding='same'),
    BatchNormalization(),
    LeakyReLU(negative_slope=0.01),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.3),

    Conv2D(filters=64, kernel_size=(3, 3), padding='same'),
    BatchNormalization(),
    LeakyReLU(negative_slope=0.01),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.3),

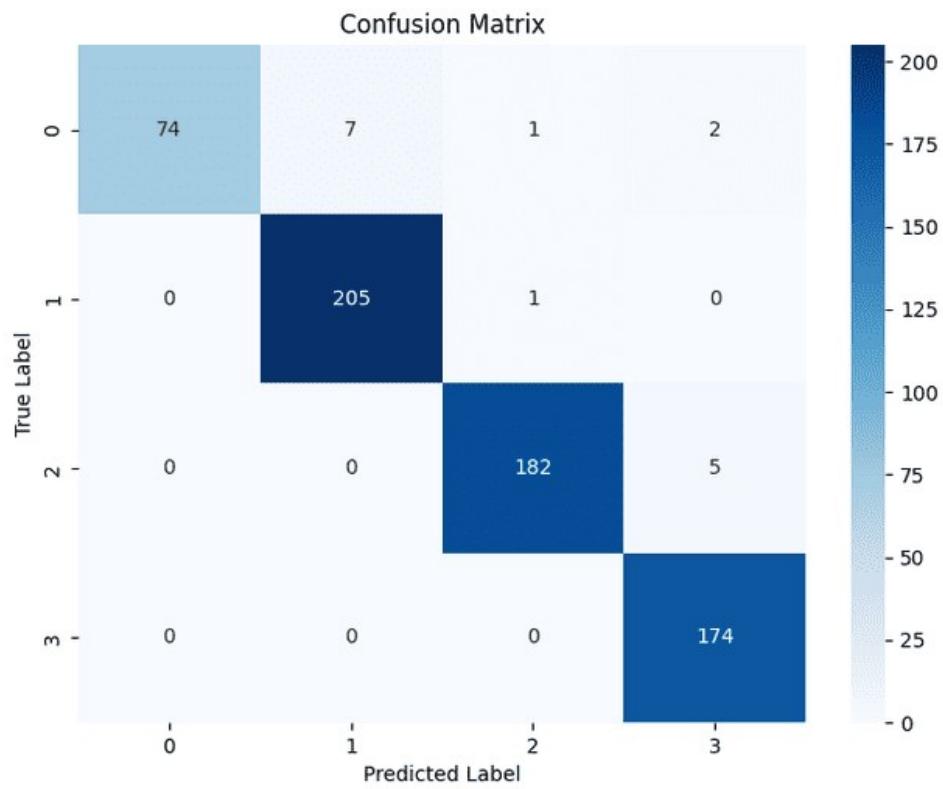
    GlobalAveragePooling2D(),
    Dense(64, kernel_regularizer=L2(0.01)),
    BatchNormalization(),
    LeakyReLU(negative_slope=0.01),
    Dropout(0.5),

    Dense(4, activation='softmax', kernel_regularizer=L2(0.01))
])
)

```

Figure 57: Implementare python a rețelei

Modelul a ajuns la convergență după 20 de epoci. În figura de mai jos regăsim matricea de confuzie.



*Figure 58: Leucemie Clas1 - Matrice de confuzie*

Acuratețea modelului a fost de 97.54%. Nu au fost instanțe în care modelul nu a detectat prezența condiției, dar au fost multe instanțe în care modelul a răspuns cu un fals pozitiv, iar 3 dintre aceste instanțe au fost clasificate ca forme avansate de leucemie. Ne dorim ajustarea modelului pentru a minimiza numărul pacienților diagnosticați incorrect.

### 3.1.5 Al doilea model pentru clasificare

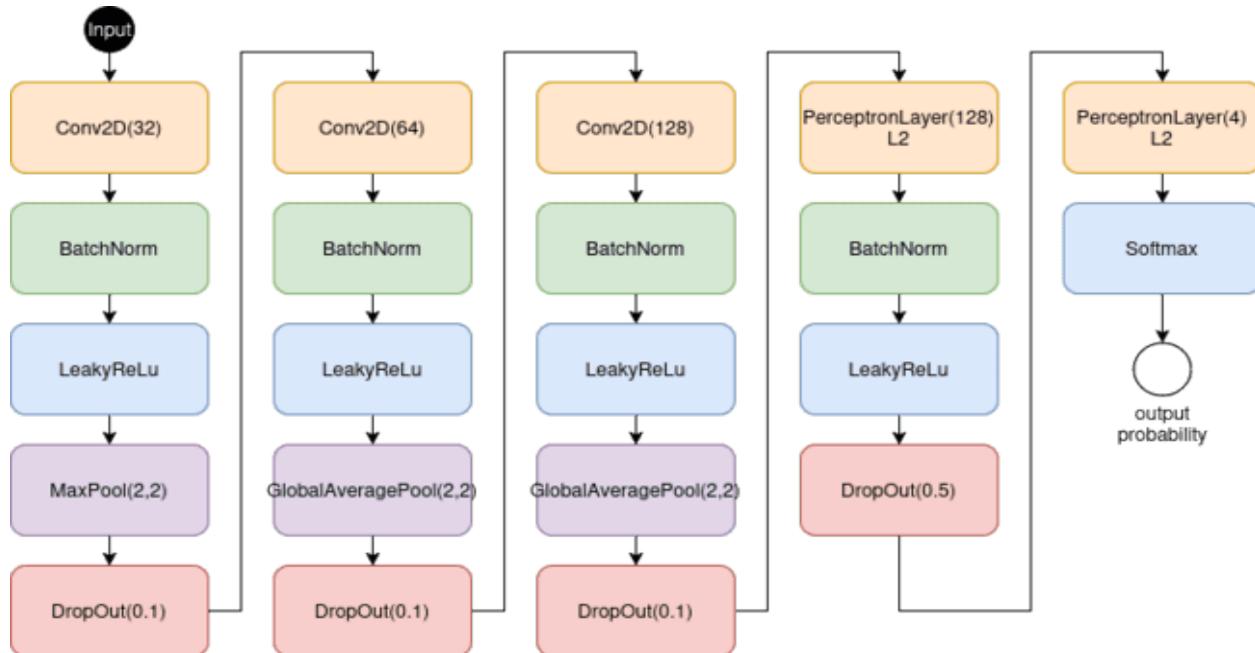
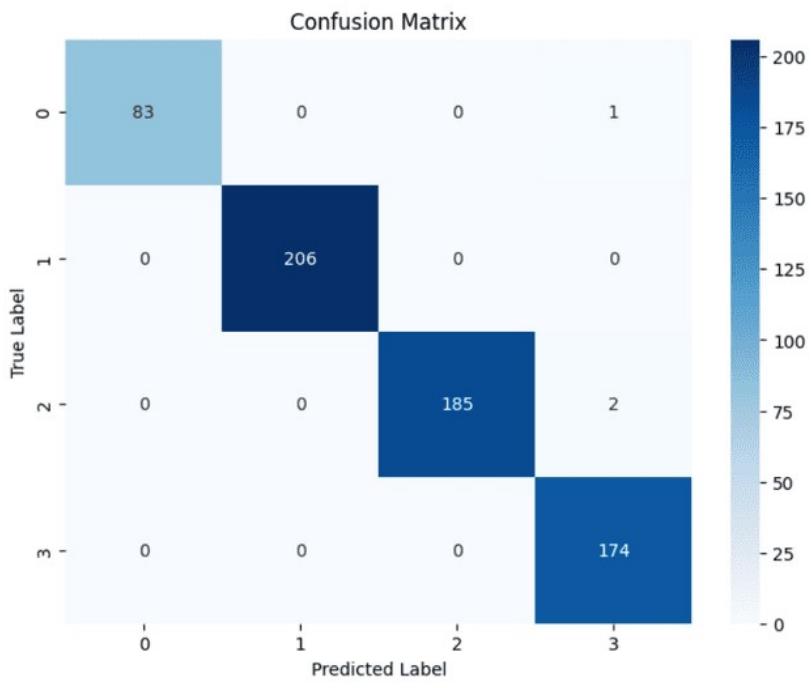


Figure 59: Leucemie Clas2

Figura 60 conține arhitectura finală a rețelei. Pentru obține rezultate mai bune pentru setul de test am introdus o a treia conoluție pentru a extinde capacitatele modelului. Pe lângă extinderea rețelei, am încercat să utilizez o funcție de activare diferită pentru a vedea dacă aceasta va duce la o precizie mai bună. Am ales funcția LeakyReLU pentru că era cea mai apropiată de activarea ReLU. Subcapitolul următor explică avantajele funcției LeakyReLU.

În figura 56 se regăsesc rezultatele celui de-al doilea model. Acuratețea a fost cu mult mai bună, ajungând la 99.54%. Există încă 3 cazuri clasificate greșit, printre care se află și un fals pozitiv pentru tipul progenitor, dar acestea sunt neglijabile considerând acuratețea modelului.



*Figure 60: Lucemie Class2 - Matricea de confuzie*

*Table 3: Măsurători class 1*

Clasa	Precizie	Recall	F1-Score	Suport
0	1.000	0.881	0.937	84
1	0.967	0.995	0.981	206
2	0.989	0.973	0.981	187
3	0.961	1.000	0.980	174

*Table 4: Măsurători class 2*

Clasă	Precizie	Recall	F1-Score	Suport
0	1.000	0.988	0.994	84
1	1.000	1.000	1.000	206
2	1.000	0.989	0.995	187
3	0.983	1.000	0.991	174

Conform tabelelor 3 și 4, cel de-al doilea model oferă performanțe cu mult mai bune, de aceea modelul 2 va fi cel pe care îl vom alege pentru clasificare. Precizia și recall pot fi metrii foarte utili pentru datele noastre. Precizia este utilă atunci când target-ul este micșorarea numărului fals pozitivelor, iar recall este util când atunci când vrem să reducem numărul fals negativelor.

### **3.1.6 Compararea activărilor cu filtrele de segmentare**

Am ales la întâmplare filtre din activările primului strat convoluțional, pentru a obține imagini ce pot fi comparate cu măștile am realizat corecția măștilor prin normalizare și mai apoi prin funcția modul aplicată măștilor. O parte semnificativă din fiecare mască se regăsește și în activarea respectivă. Însă activările specifice măștii au valori mai mici. Se pare că modelul a învățat clasifică folosind și alte criterii, cum ar fi numărul de globule roșii ce poate fi afectat de leucemie. Putem extrage insight-uri și din măști, fiind amplificate zonele cu informații utile. De asemenea observăm că activările modelului sunt slabe atunci când leucemia este cauzată de celule progenitor.

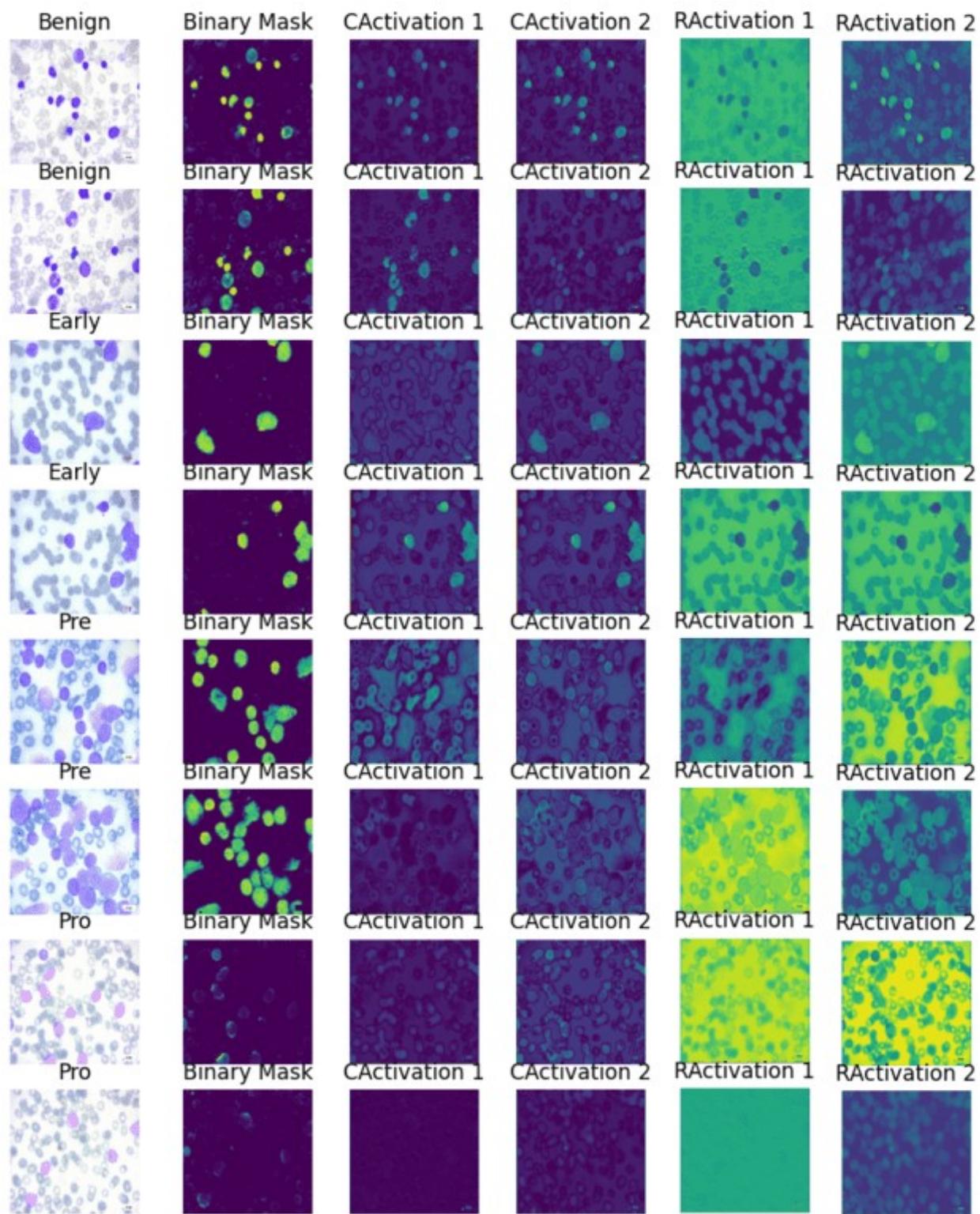


Figure 61: Leucemie - Măști și Feature Map-uri

## 3.2 Tumori cerebrale

Printre cele mai mortale tipuri de cancer se află tumorile cerebrale, 20% din persoane vor supraviețui mai mult de 2 ani după diagnosticare, în timp ce 90% din persoanele diagnostice cu cancer de prostată sau de sân vor supraviețui mai mult de 5 ani. Cea mai comună și mai agresivă formă de cancer cerebral este glioma care pornește de la celulele gliale ce au rolul de a proteja și de a hrăni neuronii. Mortalitatea cazurilor a scăzut în ultimii ani, dar glioma rămâne totuși o afecțiune dificil de detectată și mai apoi mult mai dificil de tratat, deoarece nu răspunde la metode convenționale de tratare. The Brain Tumor Segmentation (BraTS) Challenge 2023: Glioma Segmentation in Sub-Saharan Africa Patient Population (BraTS-Africa)

Tomografia prin rezonanță magnetică (RMN) reprezintă o metodă imagistică non-invazivă ce permite obținerea unor imagini tridimensionale detaliate ale structurilor anatomici, fără a provoca afectarea țesuturilor organelor. Aceasta utilizează proprietățile magnetice ale nucleelor atomilor de hidrogen, fiind o tehnică ce nu implică radiații ionizante, ceea ce o face sigură pentru utilizarea repetată în diagnostic. Atomii de hidrogen sunt particulele țintă în RMN datorită abundenței lor în organismul uman (aproximativ 63% din atomii din corp sunt hidrogen) și structurii simple a nucleului acestora, format dintr-un singur proton. Protonii au două stări posibile de spin, fapt ce facilitează măsurarea și detectarea semnalului generat de aceștia în prezența unui câmp magnetic puternic. În timpul efectuării examinării RMN, pacientul este plasat într-un câmp magnetic puternic generat de tomograf. Sub acțiunea acestui câmp, vectorii de spin ai protonilor se aliniază cu axa câmpului magnetic, fie paralel (stare de energie joasă), fie antiparalel (stare de energie înaltă). Ulterior, aparatul emite pulsuri de radiofrecvență care perturbează această aliniere, determinând protonii să-și schimbe orientarea și să intre într-o stare de excitare. După întreruperea impulsurilor radio, protonii încep să-și revină către starea de echilibru inițială, eliberând energia absorbită sub formă de semnal detectabil. Caracteristicile acestui semnal, în special timpul în care protonii revin la starea de echilibru, variază în funcție de tipul țesutului și compoziția acestuia, oferind astfel informații esențiale pentru reconstrucția imaginii. Există două tipuri principale de tempi de relaxare care stau la baza contrastului în imaginile RMN: timpul longitudinal de relaxare (T1) și timpul transversal de relaxare (T2). Magnetic Properties Of Hydrogen MRI Physics Made Easy Magnetic Resonance Imaging (MRI) of the Brain and Spine: Basics How are T1 and T2 weighted images generated? T1 vs T2 MRI

Timpul de relaxare longitudinală (T1) reprezintă intervalul necesar pentru ca protonii să-și realinieze spin-ul de-a lungul axei câmpului magnetic principal. Acest timp depinde de transferul de energie între protoni și mediul înconjurător, fiind mai scurt în țesuturile cu o densitate mai mare de atomi de hidrogen, cum ar fi țesuturile grase. Aceasta se datorează faptului că protonii din astfel de medii pot transmite energia mai eficient între ei..Magnetic Properties Of HydrogenMRI Physics Made EasyMagnetic Resonance Imaging (MRI) of the Brain and Spine: BasicsHow are T1 and T2 weighted images generated ?T1 vs T2 MRI

Timpul de relaxare transversală (T2) reflectă intervalul de timp în care protonii își pierd coerența în planul transversal față de axa câmpului magnetic, revenind la spin-ul inițial. T2 este mai scurt în țesuturile cu densitate scăzută de atomi de hidrogen, cum ar fi țesuturile bogate în apă, deoarece protonii sunt mai puțin influențați de interacțiunile magnetice locale..Magnetic Properties Of HydrogenMRI Physics Made EasyMagnetic Resonance Imaging (MRI) of the Brain and Spine: BasicsHow are T1 and T2 weighted images generated ?T1 vs T2 MRI

Deoarece au acțiuni opuse, RMN-ul nu poate măsura concomitent ambele durate și este necesară alegerea uneia dintre ele în funcție de situație. T1 este detecteză cel mai bine țesuturile adipose, ceea ce înseamnă că T1 este cel mai util atunci când vrem să vizualizăm anatomia generală a creierului. T2 detecteză țesuturile cu conținut mai mare de apă, pentru anomaliiile ce cauzează acumularea de fluide T2 trebuie folosit. Tumorile și leziunile pot duce la acumularea de apă în țesuturi (edem) sau anomalii de vascularizare și pot fi detectate mai ușor de T2. .Magnetic Properties Of HydrogenMRI Physics Made EasyMagnetic Resonance Imaging (MRI) of the Brain and Spine: BasicsHow are T1 and T2 weighted images generated ?T1 vs T2 MRI

FLAIR (Fluid-Attenuated Inversion Recovery) este o variantă a masurătorii T2, în care semnalul de la fluidele libere este suprimat. În cazul creierului, lichidul cefalorahidian care se găsește între celule creierului poate acoperi leziunile sau anomaliiile ușoare. FLAIR poate fi utilizat pentru a dezvălui leziunile subtile în materia albă a creierului..Magnetic Properties Of HydrogenMRI Physics Made EasyMagnetic Resonance Imaging (MRI) of the Brain and Spine: BasicsHow are T1 and T2 weighted images generated ?T1 vs T2 MRI





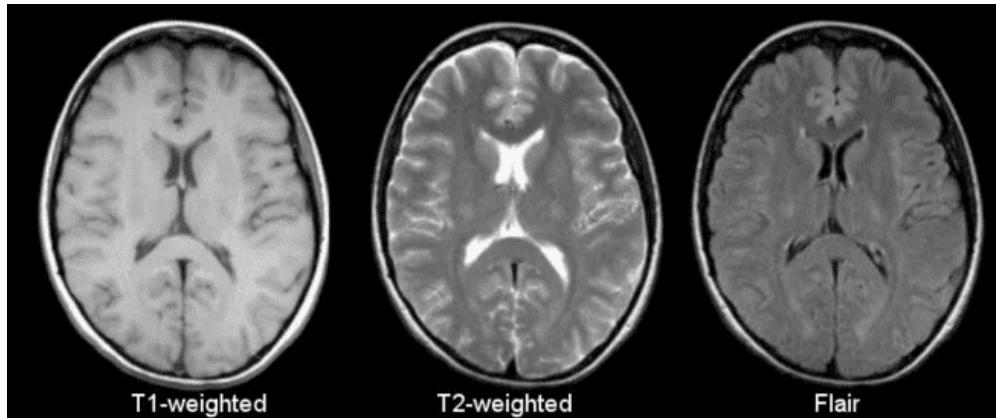


Figure 62: Metodote RMN

### 3.2.1. Setul de date folosit pentru segmentare

Brain Tumor Segmentation (BraTS) Challenge este o competiție internațională de detectare și de clasificare a cancerului la nivelul celulelor gliale. Competiția este organizată anual pentru a găsi soluții ce pot contribui la reducerea numărului de cazuri ce nu au fost detectate la timp și pentru a simplifica procesul de diagnosticare. Setul de date a fost creat prin colaborarea mai multor instituții și conține RMN realizate folosind următoarele metode T1, T1GD, T2 și FLAIR. T1DG reprezintă utilizarea metodei T1 după ce pacientului i-a fost administrat un lichid(godoniu) ce poate mării contrastul pentru tumorile ce nu pot fi detectate ușor .MICCAI 2025 Lighthouse Challenge: Brain Tumor Segmentation Cluster of Challenges (BraTS)

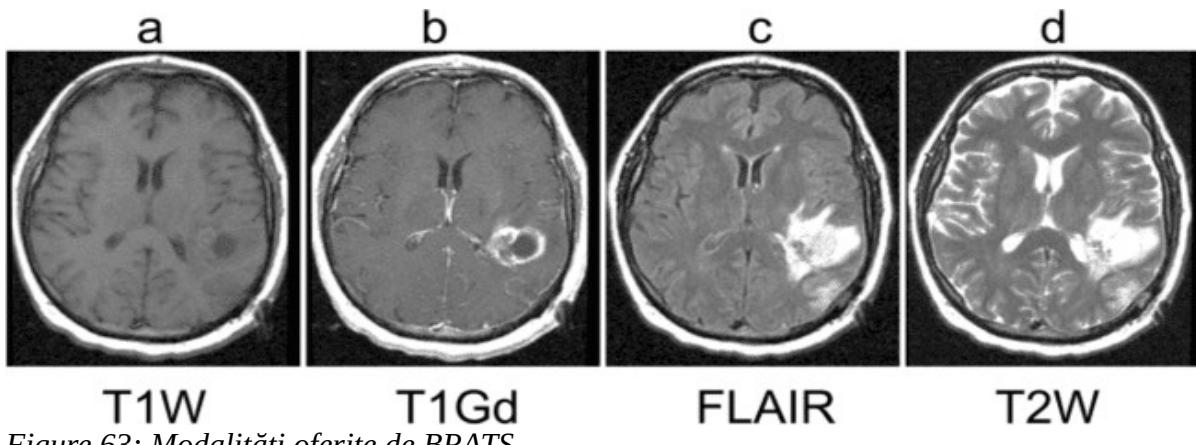


Figure 63: Modalități oferite de BRATS

Pentru a putea clasifica tumorile în funcție de gravitate, BraTS pune la dispoziție imagini segmentate în 3 regiuni.

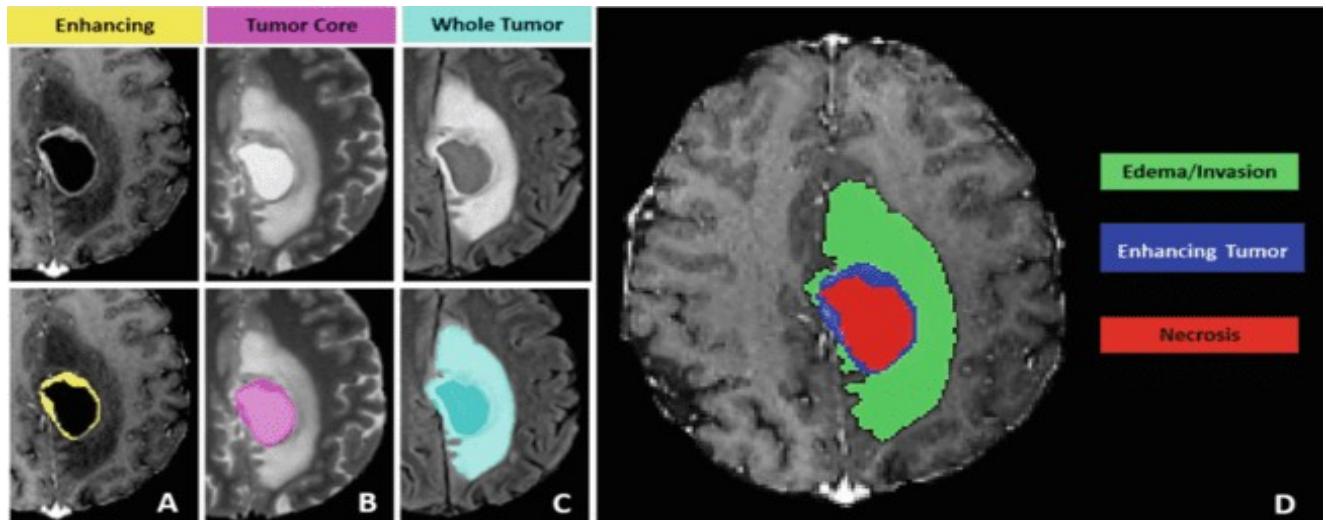


Figure 64: Măstile din setul de segmentare

- Necroza tumorala sau Centrul tumorii: Parte a tumorii formate din celule ce au murit datorită vascularizării scăzute cauzate de extinderea tumorii
- Edemul: Este o inflamație a țesutului nervos cauzată de extinderea tumorii și duce la acumularea de lichid în creieri. Poate apărea și în lipsa unei tumori atunci când este consecința unui atac cerebral sau a unei leziuni. Presiunea și infiltrarea lichidului în alte părți ale creierului

duce la dureri de cap, crize de epilepsie, amețeli, probleme de vedere sau de vorbire și la distrugerea celulelor sănătoase ce nu au fost afectate încă de tumoră.

- Enhancing(tumora activă): Este zona ce nu poate fi detectată ușor de către RMN și adesea necesită administrarea de fluid de contrast. Apare ca urmare a extinderii tumorilor și ne poate oferi informații despre viteza cu care se extinde tumora către țesuturile sănătoase și despre gravitatea tumorii.

Corelațiile dintre clase și metode:

Componenta (clasa)	tumорii	Cea mai bună secvență RMN	Explicație
Edem		T2 / FLAIR	T2/ FLAIR oferă cea mai bună vizualizare a fluidului cauzat de edem. Preferăm însă FLAIR pentru a separa fluidul cauzat de inflamație de fluidul cefalorahidian
Tumoră activă		T1 cu gadoliniu (T1Gd)	Contrastul cu gadoliniu evidențiază zonele unde bariera hematoencefalică este compromisă și face tumora activă mai vizibilă
Necroză/Nucleu		T1 / T1Gd	Datorită lipsei de vascularizare țesutul este mai dens și poate fi mai ușor de observat folosind T1, unde contrastul va duce la formarea unei zone întunecate acolo unde se află necroza
Tumoră totală		FLAIR (cea mai bună modalitate)	Deși nu oferă vizibilitate maximă în cazul necrozei și a tumorii active. FLAIR oferă cel mai bun echilibru dintre metode și este singura metodă care surprinde toate structurile tumorii

### 3.2.2 Preprocesarea imaginilor pentru setul de segmentare

Tensorflow nu poate încărca RMN-uri, de aceea trebuie să ne folosim de librăria nibabel și de tensorflow pentru a genera setul de date.

Pentru început am ales să folosesc doar metoda FLAIR pentru că oferă informații despre fiecare clasă. Înainte de etapa proiectare a modelului trebuie să analizăm setul de date.

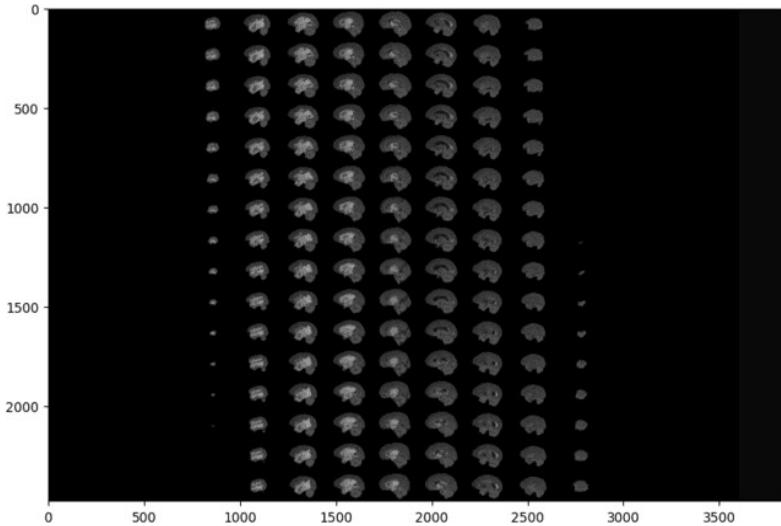


Figure 65: Vizualizarea conținutului slice-urilor

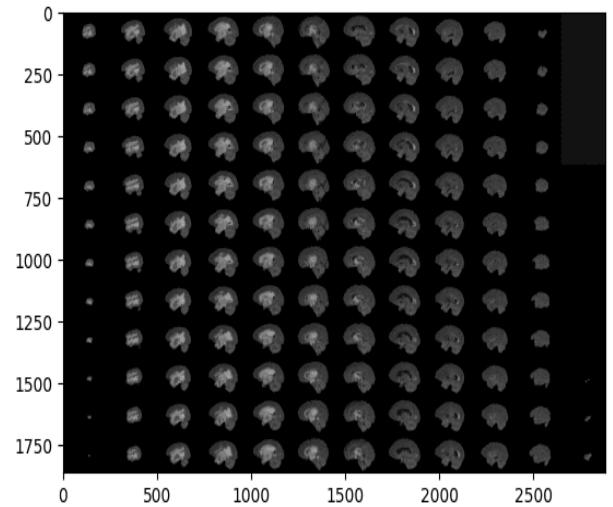
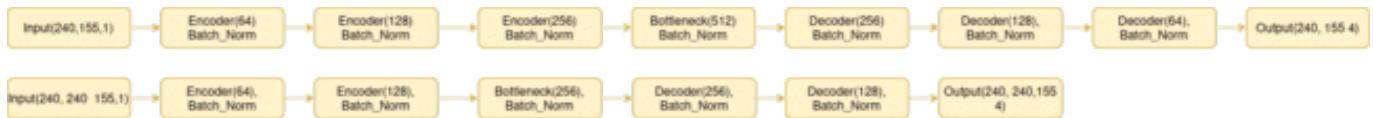


Figure 66: Slice-urile ce conțin informații

O primă observație asupra setului de date ar fi că fiecare din axele tomografului conține slice-uri goale ce pot afecta procesul de învățare prin crearea de „bias” care ar face modelul să prezică mai des fundalul decât celelalte clase pentru a obține un scor de acuratețe mai bun. După ce am eliminat slice-urile goale am creat un al doilea set de date ce conține doar slice-urile sagitale din tomografe. Am ales slice-urile sagitale pentru că ofereau o claritate mai bună decât celelalte slice-uri și sunt în număr mai mare (dimensiunile tomografului sunt 240x240x155). Pentru preprocesare am aplicat rotații, oglindiri pe ambele axe și am amplificat contrastul imaginilor în cazul setului de slice-uri, iar în cazul setului de tomografe doar contrastul a fost a modificat.

### 3.2.3 Segmentarea imaginilor



În prima etapă a procesului de dezvoltare, am optat pentru utilizarea unor rețele neuronale cu o arhitectură mai simplă ca mai apoi să măresc complexitatea acestora. Am ales această abordare cu scopul de a identifica configurația optimă a modelului. Modelul pentru tomografe folosește mai puține, dar straturile au mai mulți parametrii deoarece se mai adaugă o dimensiune fiecărui layer pentru a reprezenta dependența dintre slice-uri. După antrenare metrica de acuratețe avea valori de 97% pentru modelul 3D și 95% pentru cel 2D, dar prediicările erau departe de a fi corecte. Prediicările pentru modelul 2D păreau aleatorii, în timp ce prediicările modelul pentru setul original prezicea doar clasa „background”. Pentru a înțelege mai bine cauza am verificat distribuția claselor.

Clasă	Background	Necrotic	Edema	Enhancing
Prob	0.98651249	0.00299638	0.00783444	0.00265669

Din tabelul de distribuții putem observa diferențe uriașe între probabilitățile de apariție a claselor. Aproximativ 98.65% din numărul total de pixeli sunt de tip „background”, ceea ce înseamnă că metrica de acuratețe nu este de încredere, deoarece modelul va prezice mai des această clasă pentru a obține rezultate cât mai bune pentru cost și metriki, deși prediicările pentru celelalte clase vor avea de suferit, de acea va trebui să găsim o metodă de rezolvare a acestei probleme.

- Pentru că majoritatea pixelilor aparțin clasei 0 ne confruntăm cu „dying ReLU problem”. „The dying ReLU problem” presupune existența unui număr mare de neuroni ale căror activări sunt constant 0. Deoarece acești neuroni sunt mereu 0 nu pot fi antrenați, fiind considerați „neuroni morți” pentru că erorile nu se pot propaga înapoi prin aceștia. Leaky ReLU este o variantă a funcției ReLU ce rezolvă problema aproximând valorile negative la o valoare negativă apropiată de 0. Deoarece valoarea este diferită de 0 eroarea se va putea propaga mai departe, dar parametrii vor fi ajustați într-o măsură mai mică. După ce am înlocuit activările de tip ReLU cu activări de tip Leaky ReLU, modelul a început să prezică și celelalte clase, însă păstra tiparul aleator prezent și la celălalt model. Schimbarea activărilor a reprezentat un pas important în ajustarea modelului, însă nu a fost suficient pentru a mări în mod semnificativ acuratețea modelului.
- Coeficientul Dice-Sørensen (cunoscut și sub denumirea simplă de Dice Score sau Dice Similarity Coefficient - DSC) este o metrică utilizată frecvent pentru evaluarea performanței modelelor de segmentare, în special în domeniul procesării imaginilor medicale. Această

metrică măsoară suprapunerea între două seturi de date binare: segmentarea prezisă de model și segmentarea de referință (ground truth).

Matematic, coeficientul Dice este definit ca:

$$Dice = 2 \times \frac{|A \cap B|}{|A| + |B|}$$

Unde A reprezintă mulțimea pixelilor clasificați pozitiv de model, iar B este mulțimea pixelilor pozitivi din segmentarea reală. Termenul  $|A \cap B|$  indică numărul de pixeli comuni între cele două mulțimi, iar  $|A|$  și  $|B|$  reprezintă cardinalitatea (numărul de elemente) a fiecărei mulțimi.

Valoarea coeficientului Dice variază între 0 și 1, unde 1 indică o suprapunere perfectă, iar 0 indică lipsa oricărei suprapunerii. Astfel, o valoare mai mare a Dice Score reflectă o segmentare mai precisă și o concordanță mai bună între predicția modelului și segmentarea de referință.

Avantajul utilizării coeficientului Dice constă în faptul că acesta penalizează puternic fals pozitivele și fals negativele, fiind deosebit de util în cazul datelor dezechilibrate, cum este frecvent cazul în segmentarea medicală, unde clasele de interes ocupă o proporție mică din totalul imaginii (de exemplu, tumori sau anomalii). Pentru a evalua performanțele modelului vom utiliza o metrică pentru fiecare clasă și una generală.

- În seturile de date în care clasele au frecvențe de apariții sunt inegale, modelele vor prefera să prezică clasele majoritare, clasificând incorrect sau chiar ignorând complet clasele minoritare. Pentru a rezolva această problemă trebuie să adaugăm ponderi de penalizare pentru clasificarea greșită a claselor pentru a-i oferi modelului informații despre relevanța claselor. Ne putem folosi de distribuția claselor pentru a obține niște ponderi inițiale și ne vom folosi de următoarea formulă pentru transformarea distribuției în ponderi:

$$w_i = \frac{\frac{1}{p_i}}{\sum \frac{1}{p_j}}$$

Și obținem următorul tabel de ponderi:

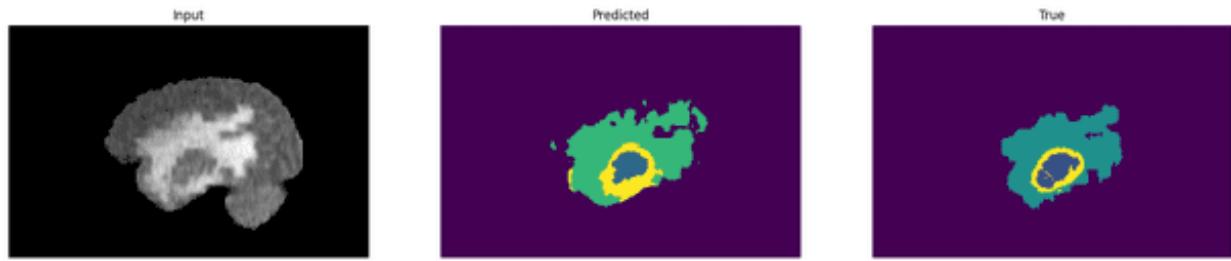
Clasă	Background	Necrotic	Edema	Enhancing
Pondere Calculată	0.00120848	0.39787306	0.15217176	0.4487467

Unde clasele cu ponderi mai mici vor fi penalizate cel mai puternic. Am început totuși cu ponderi mai slabe (0.1, 1.0, 0.4, 1.0)

Rezultate după modificări :

*Table 5: Comparații implementări*

	Dice	Dice_Bg	Dice_Necrotic	Dice_Edema	Dice_Enhancing
U_NET	95.05	98.41	10.57	28.82	16.49
U_NET_3D	76.58	91.28	9.7	21.97	12.5



*Figure 67: Mască prezisă folosind primul U-Net*

Modelul nu mai generează aleatori măști, ceea ce înseamnă că am rezolvat o parte din limitările modelului. Dar performanțele nu sunt suficient de bune. Am îmbunătățit rețeaua prin următoarele metode:

- Adăugarea de mecanisme de atenție. În contextul segmentării imaginilor, mecanismele de atenție reprezintă componente arhitecturale menite să permită rețelei neuronale să aloce ponderi diferite pixelilor sau regiunilor din imagine, în funcție de relevanța acestora pentru sarcina de învățare. Astfel, modelul devine capabil să distingă între zonele de interes (structuri anatomicice, tumori sau alte anomalii) și regiunile mai puțin semnificative din punct de vedere clinic sau contextual. Pentru implementarea unui astfel de mecanism, se pot introduce porți de atenție în decodoarele rețelei de tip encoder-decoder. O poartă de atenție urmează, în general, următoarea structură funcțională:

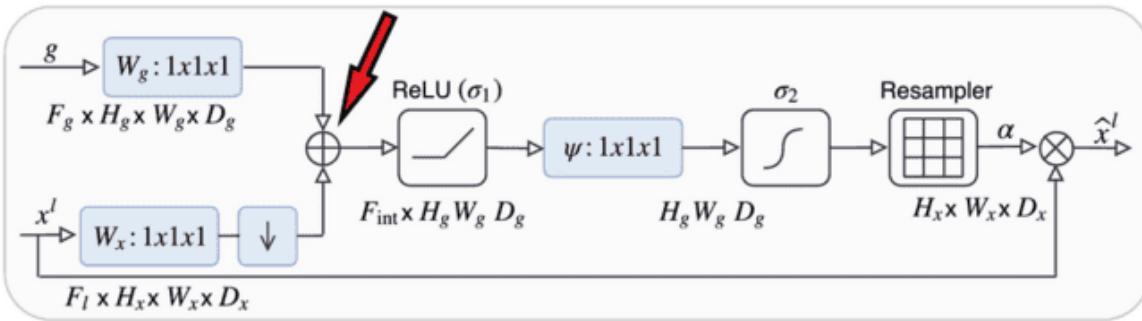


Figure 68: Portă de atenție

- Se îmbină (de obicei prin concatenare) activările provenite din stratul de codificare (encoder) cu cele din stratul anterior al decodorului, urmate de aplicarea unei funcții de activare, cum ar fi ReLU.
- Se aplică o funcție sigmoid asupra resultantului pentru a obține o matrice de atenție, care reflectă importanța relativă a fiecărui pixel. Valoarea fiecărui element din această matrice se află între 0 și 1, unde 0 semnalează o contribuție neimportantă, iar 1 indică o trăsătură esențială.
- Această matrice este apoi folosită pentru a modula activările codificatorului prin înmulțire element cu element (element-wise), reducând influența trăsăturilor irelevante și amplificând impactul celor relevante în procesul de decodificare.

Prin introducerea acestor mecanisme, rețeaua devine mai eficientă în alocarea resurselor computaționale, concentrându-se pe regiunile critice ale imaginii. Acest lucru are drept consecință o îmbunătățire a acurateței segmentării și o mai bună capacitate de generalizare a modelului în fața datelor noi. Totuși, această îmbunătățire vine cu un cost computațional considerabil. Adăugarea porțiilor de atenție implică un număr suplimentar de operații și parametri, ceea ce conduce la un consum crescut de memorie și la un timp de antrenare mai mare. Prin urmare, utilizarea mecanismelor de atenție trebuie echilibrată atent în funcție de resursele disponibile și de complexitatea sarcinii abordate.

- Utilizarea altor funcții de cost. Atunci când folosim Dice-Sørensen ca și metrică de evaluare a modelului (funcție obiectiv) putem folosi atât funcția de entropie încrucișată, cât și funcția de cost Dice. Costul Dice este 1-Dice, iar dacă funcția obiectiv Dice-Sørensen arată cât de similare sunt două imagini, atunci costul pentru funcția obiectiv arată cât de diferite sunt. Atunci când am utilizat ponderile inițiale calculate în costul Cross-Entropy am obținut rezultate mai slabe,

deși scorurile Dice pentru clase erau egale. Deoarece penalizăm prea agresiv predicția greșită a anumitor clase, modelul nu mai poate înțelege relația spațială dintre clase și procesul de învățare stagnează. Utilizarea costului Dice rezolvă această problemă, dar creează diferențe prea mari între clase. Cum nici utilizarea funcției de cost Dice nu a rezolvat această problemă, decis că o combinație dintre cele 2 funcție ar fi cea mai potrivită pentru rezolvarea acestor funcții. Costul a devenit:

$$\text{Combined Loss} = a * (1 - \text{Dice}) + b * \text{Cross Entropy}, a+b=1$$

- Combinarea mai multor tipuri de tomografe. Deși am evitat utilizarea mai multor tipuri de tomografe pentru că am vrut ca rețea finală să fie cât mai optimă din punct de vedere al memoriei, mi-am dat seamă că utilizarea unei singure metode nu este suficient pentru a segmenta precis imaginea. Am modificat mai apoi rețea finală pentru a putea primi ca input câte un tomograf de fiecare tip. T1/T1gd ne vor ofere mai multe informații despre tumoră activă și despre necroză, iar T2 ne va oferi mai multe informații pentru segmentarea edem-ului.



Figure 69:

Arhitectura finală brats

În figura 65 avem configurația finală a rețelei. Deoarece datele sunt de dimensiuni mai mari, a trebuit să reduc dimensiunea batch-urilor la 2 instanțe. Având de a face cu batch-uri de dimensiuni mici nu vom normaliza batch-urile în timpul antrenării. În schimb vom utiliza L2 pentru a reduce overfitting-ul pentru setul de date de antrenament.

Table 6: U-NET final

	Dice	Dice_Bg	Dice_Necrotic	Dice_Edema	Dice_Enhancing
U_NET_3D	99.19	98.41	47.09	63.49	65.64

Se observă o performanță mult mai bună a modelului, forma se apropie însă anumite poze nu respectă ierarhia claselor. În setul de date am găsit imagini ce nu respectă această ierarhie datorită suprapunerii mei multor tumori sau datorită tumorilor sau datorită tipului de tumoră. Gliomele sunt tumori complexe din punct de vedere biologic și pot cauza o gamă largă de mutație, de aceea răspund

greu la tratament. În unele cazuri pot invada alte ţesuturi fără a cauza edem , iar tumorile active și necrozele se pot extinde abnormal și pot exista în lipsa celeilalte.

În setul de date de mai jos observăm o astfel de formăjune neregulată.

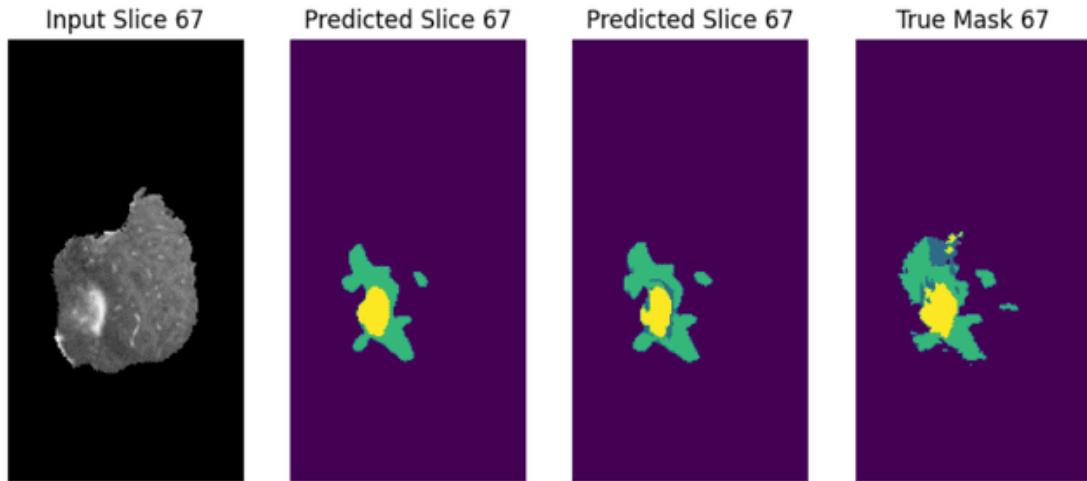


Figure 70: Tumoră atipică

Formațiunea din imagine cauzează probleme și la predicția altor slice-uri din același tomograf, deoarece slice-urile sunt interdependente. Diferența dintre slide-urile 2 și 3 este timpul de antrenare al modelului. Pentru primul slide modelul a fost antrenat în 10 epoci, iar în al doilea a fost antrenat 20 de epoci.

Imaginea aleasă este cea mai dificil de segmentat din setul de date.

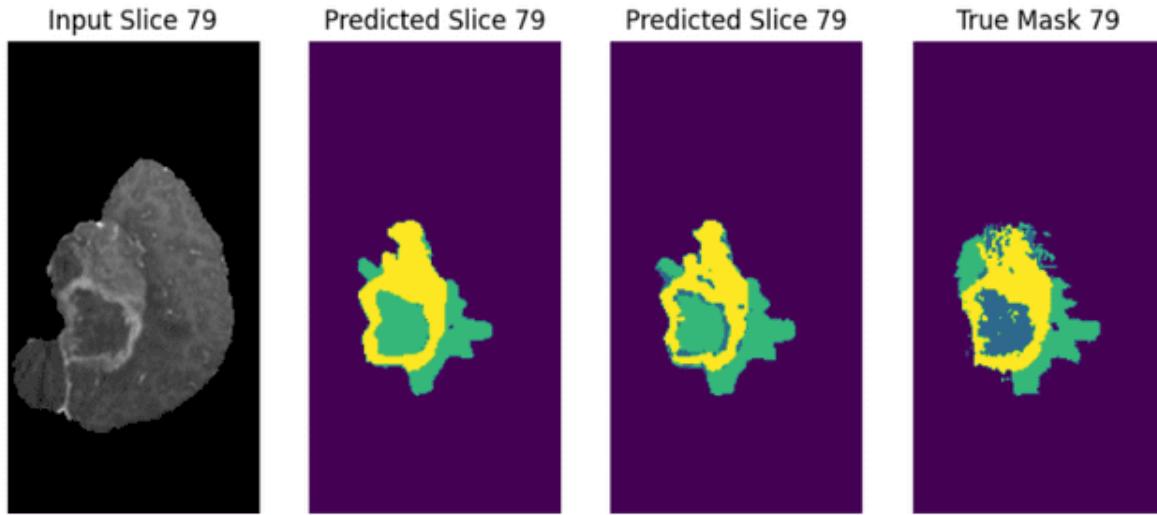


Figure 71: Pierderea informației despre structura ierarhică

Prima metodă prin care am putea face programul să înțeleagă ierarhia claselor este introducea de penalizări. Penalizările sunt termeni pe care îi putem adăuga la funcția de cost pentru a reprezenta reguli pe care modelul trebuie să le respecte. Pot îmbunătății modelul și putem adăuga oricât de multe penalizări dorim, dar atunci când numărul lor este prea mare sau atunci când contrazic reguli deja existente acestea pot duce la performanțe mai slabe.

Pentru a ne construii penalizările trebuie mai întâi să ne creăm un set de reguli pentru reprezentarea ierarhiei:

- R1. Dacă există zona C3 (edema), atunci zona C2(enhancing) trebuie să se afle în interiorul ei.
- R2. Dacă există zona C2(enhancing), atunci zona C1(necrotic) se află în interiorul ei. Dacă nu există zona C2, dar există zona C3, zona C1 se află în interiorul ei

$$C1 \subseteq C2 \subseteq C3$$

Dar putem folosi și regulile inversate:

- $\sim$ R1. C3 nu se află în C2 sau C1
- $\sim$ R2. C2 nu se află în C1

$$C3 \not\subseteq C2 \not\subseteq C1$$

Ambele seturi de reguli prezintă dezavantaje. În regulă inițială C3 poate să se afle în interiorul lui C1 și C2, iar C2 poate să se afle în C1 pentru că nu există reguli care să impună să fie în afara lor. În regula inversată C1 poate fi în afara lui C2 și C3, iar C2 poate fi în afara lui C3. În ciuda limitărilor nu vom aplica simultan ambele seturi reguli. Scopul penalizărilor este de a ghida modelul, nu de a impune

reguli stricte de structură, în plus aplicarea ambelor seturi de reguli nu va duce la performanțe mai bune, deoarece regulile încearcă să obțină același rezultat prin metode diametral opuse. Un alt motiv pentru care nu vrem ierarhii stricte de clase este pentru a putea clasifica în continuare acele anomalii care par outlier la prima vedere, dar reprezintă cazuri reale ce trebuie să fie luate în considerare de către model. Mai jos regăsim implementarea primei penalizări

```

@register_keras_serializable()
def containment_penalty(y_pred):
    class_1 = y_pred[..., 1]
    class_2 = y_pred[..., 2]
    class_3 = y_pred[..., 3]

    kernel = tf.ones((3, 3, 3, 1, 1), dtype=tf.float32)
    strides = (1, 1, 1, 1, 1)
    padding = 'SAME'

    c2_exp = tf.expand_dims(class_2, -1)
    c3_exp = tf.expand_dims(class_3, -1)

    class2_shell = tf.nn.max_pool3d(c2_exp, ksize=3, strides=1, padding='SAME')
    class3_shell = tf.nn.max_pool3d(c3_exp, ksize=3, strides=1, padding='SAME')

    class2_shell = tf.squeeze(class2_shell, -1)
    class3_shell = tf.squeeze(class3_shell, -1)

    penalty_1 = tf.nn.relu(class_1 - class2_shell)

    mask_1_exists = tf.cast(tf.reduce_sum(class_1, axis=[1,2,3]) > 0, tf.float32)
    mask_1_exists = tf.reshape(mask_1_exists, [-1,1,1,1])

    target = class_1 * mask_1_exists + class_2 * (1 - mask_1_exists)
    penalty_2 = tf.nn.relu(target - class3_shell)

    penalty = tf.reduce_mean(penalty_1 + penalty_2)

    return penalty

```

Figure 72: Penalizare pentru nerrespectarea ierarhiei

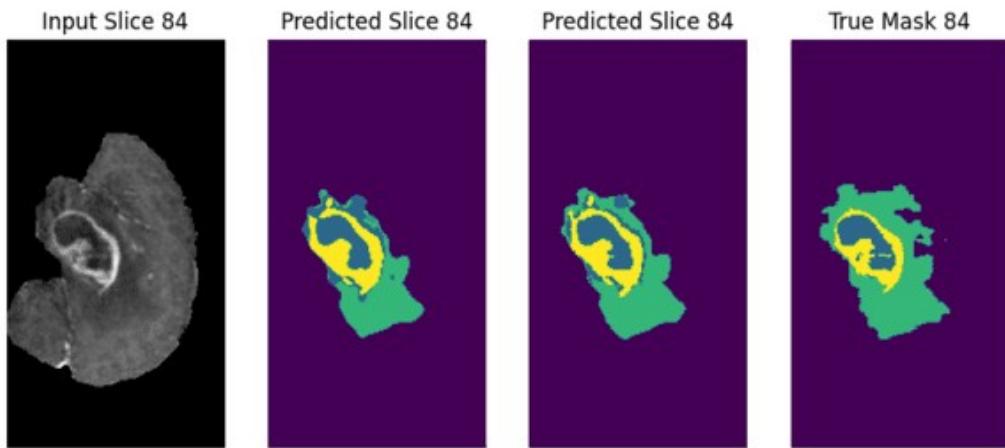
Funcția de mai sus penalizează output-urile ce nu respectă regulile inițiale. Dacă penalizarea devine prea agresivă putem lăsa o margine de eroare (un threshold) pentru a permite apariția zonelor anormale(ce nu respectă ierarhia claselor). Mecanismul este similar cu cel de „Soft Margin” din SVM (Support Vector Machine) și putem pune un prag pentru incluziunea (Dacă 70% din C1 se află în C2, atunci C1 se află în C2) sau pentru dimensiunea zonei (Dacă aria lui C1 este mai mică decât 10 pixeli<sup>2</sup> atunci regulile nu se aplică pentru C1).

Noul cost devine:

*Combined Loss=a \* (1 - Dice) + b \* Cross Entropy + c \* Containment Penalty, a+b+c=1, iar valorile optime găsite au fost:*

a	b	c
0.65	0.35	0.1

Penalizarea prezentată în document este corectă, însă atunci când am creat funcția am inversat accidental clasele din penalizare. Această greșeală a dus la formarea unui contur albastru în jurul claselor.



După ce am aplicat corect penalizarea s-au obținut următoarele rezultate, din figura.

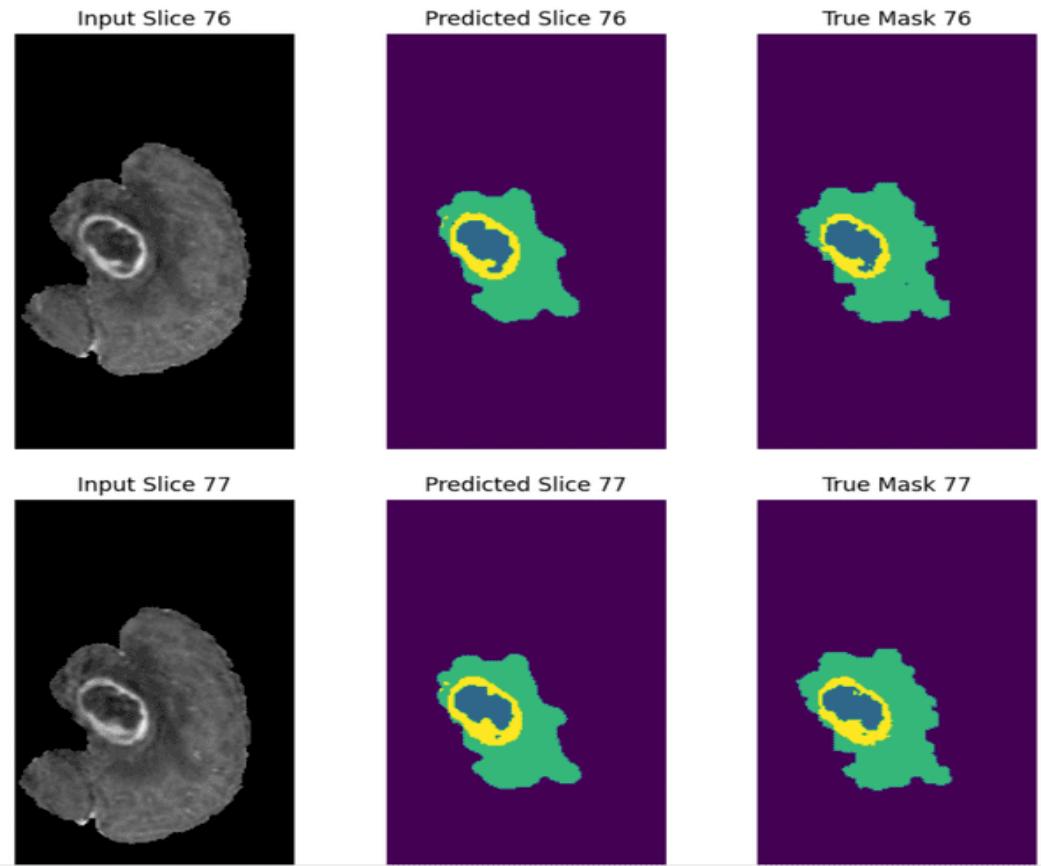


Figure 74: Segmentarea după corectarea penalizării

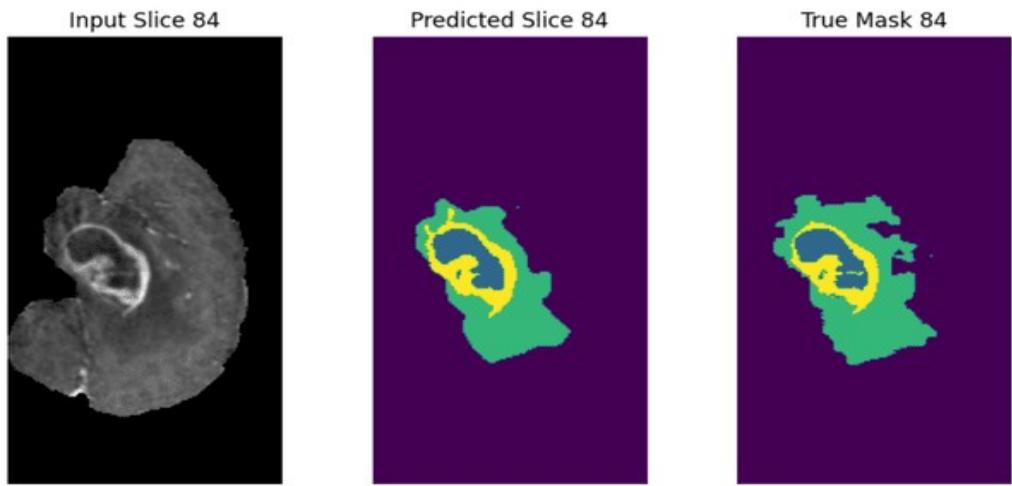


Figure 75: Segmentarea după corectarea penalizării 2

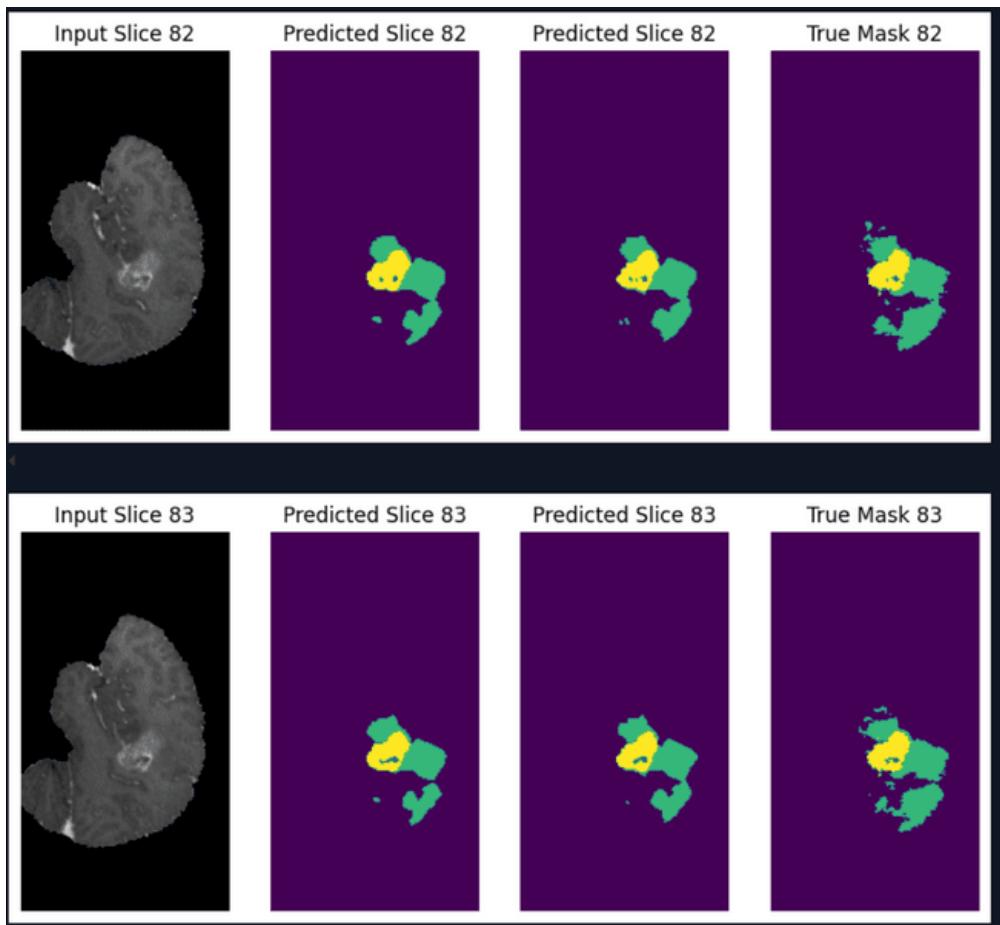


Figure 76: Segmentarea după corectarea penalizării 3

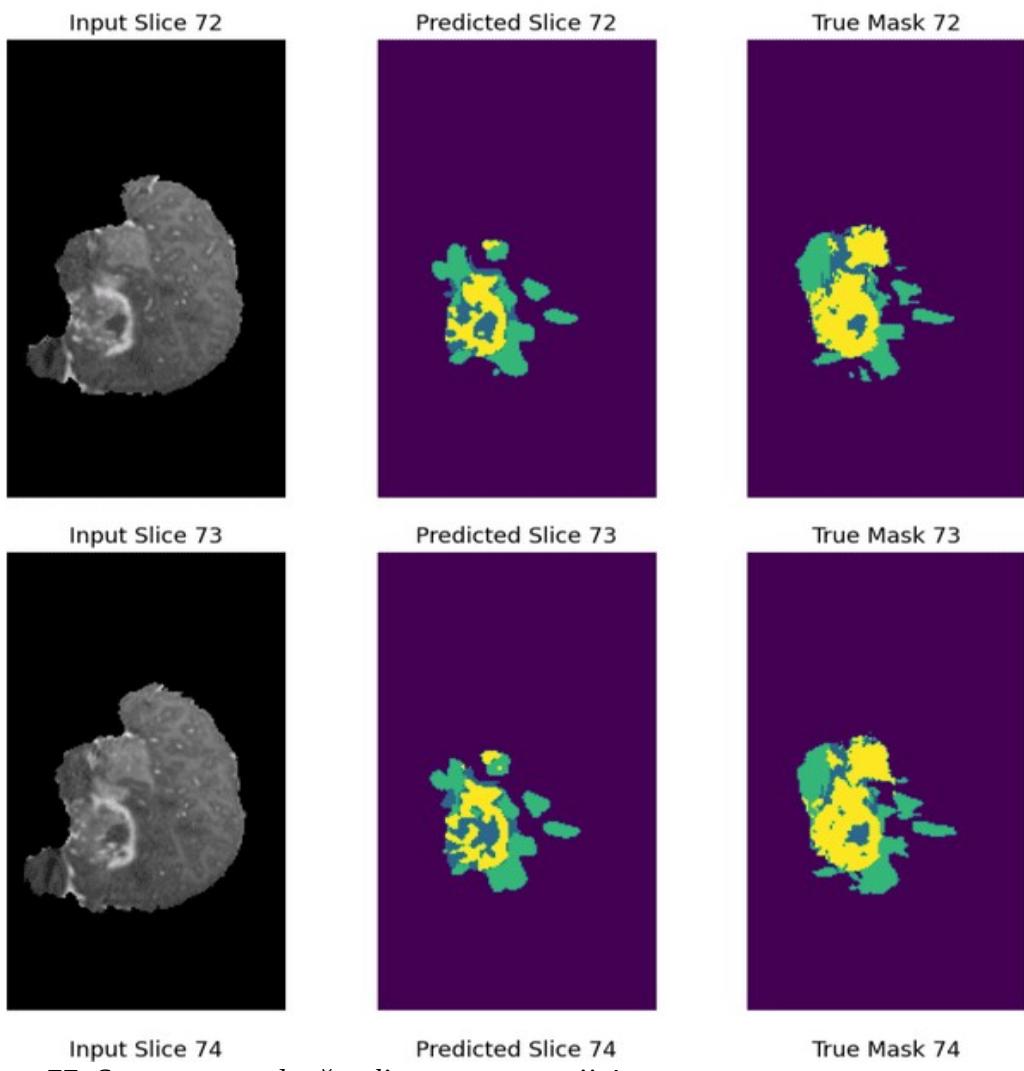


Figure 77: Segmentarea după aplicarea corecturii 4

Table 7: Măsurătorile după aplicarea panalizărilor

	Dice	Dice_Bg	Dice_Necrotic	Dice_Edema	Dice_Enhancing
U_NET_3D	99.38	98.87	60.02	72.69	75.96

Rezultatele obținute sunt mult mai bune pentru fiecare clasă. Pe cazuri generale modelul forma măștilor și predicțiile sunt similare, însă multe părți din imagine rămân nedetectate. Pe cazul celei mai dificile segmentări pare că se păstrează mai bine ierarhia claselor, însă forma imaginii are de suferit. Modelul poate fi util în segmentarea semiautomată și chiar dacă rezultatul nu este suficient de bun, este totuși unul satisfăcător. Am reușit să ridic scorul de similaritate pentru clase de la 10.57, 28.82, respectiv 16.49 până la 60.02, 72.69 și 75.96. Putem ridica obține rezultate și mai bune prin utilizarea

unei rețele cu mai multe straturi, adăugarea marginilor de eroare pentru penalizare sau prin combinarea cu alte seturi de date. Procesul de antrenare pentru rețea curentă nu a fost unul simplu totuși. Rețea curentă conține 20 de milioane de parametrii ce au fost antrenați în 50 de epoci de-a lungul a 8 ore. Resursele de calcul pe care le dețin nu mi-ar permite să antrenez o rețea mai complexă și consider că modelul curent oferă cea mai bună performanță pe care aş putea să o obțin cu resursele pe care le dețin.

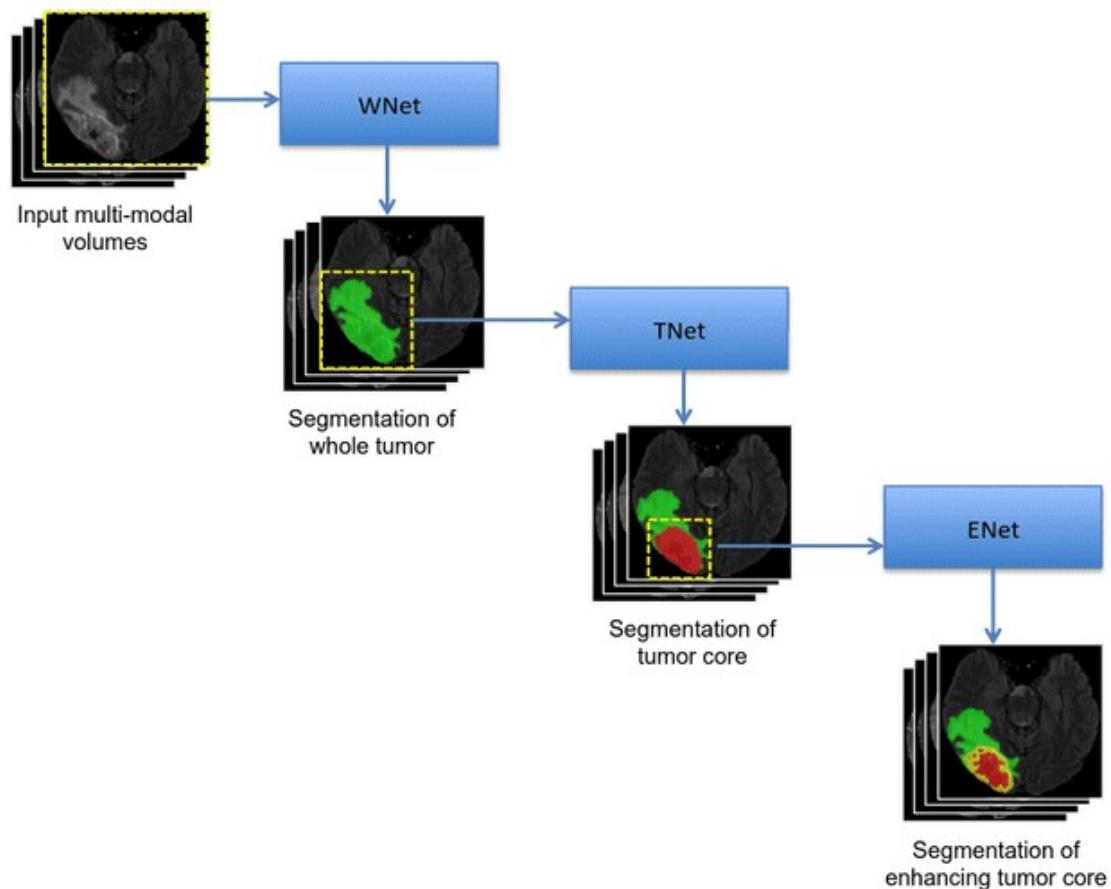


Figure 78: Ierarhie de rețele

O altă opțiune, care nu necesită la fel multe resurse hardware, ar fi codificarea structurii claselor direct în structura modelului. Putem face asta prin crearea unei ierarhii de rețele, care vor segmenta pe rând fiecare nivel din ierarhie. Vom crea măști binare prin îmbinarea straturilor : prima mască va conține clasele C3,C2,C1 și va reprezenta zona afectată, a doua mască va conține clasele C2, C1 și va reprezenta tumora, iar ultima mască va conține C1 și va reprezenta în continuare necroza. Fiecare rețea va elimina un strat din mască și mai apoi vom putea obține măștile finale astfel: C3 = OutputM1 – OutputM2, C2 = OutputM2 – OutputM3. Nu am obținut rezultate mai bune folosind acest proces, dar

am găsit lucrarea Automatic Brain Tumor Segmentation using Cascaded Anisotropic Convolutional Neural Networks, publicată de Colegiul din Londra. Guotai Wang, Wenqi Li, Sébastien Ourselin și Tom Vercauteren au reușit să implementeze corect această arhitectură și au obținut măsurători mai bune decât cele obținute în cazul modelului anterior.

*Table 8: Rezultate obținute de CACNN*

	Dice_Necrotic	Dice_Edema	Dice_Enhancing
CACNN	78.31	87.39	77.74

### 3.2.4 Setul de date pentru clasificare

Pe lângă gliomă, există și alte tipuri de tumori cerebrale cum ar fi tumorile pineale și cele meningiomele. Sunt mai puțin severe decât glioma, dar pot produce complicații în timp. Tumorile pineale apar la nivelul glandei pineale ce produce melatonina, hormonul pentru a regla ritmul circadian (somnul). Tumorile pineale pot afecta somnul, iar atunci când se extind produc și alte complicații, cum ar fi probleme în reglarea ritmului inimii, prin afectarea structurilor aflate în vecinătatea glandei. Tumorile pineale variază mult în funcție de severitate, dar de obicei răspund la tratamentul prime chimioterapie sau pot fi înlăturate. Meningioma este o tumoră la nivelul meninges, învelișul ce acoperă și protejează creierul. Meningiomele sunt tumori care se extind lent și nu cauzează complicații în cele mai multe cazuri. Pot provoca leziuni datorită presiunii pe care o pun asupra creierului, ceea ce va duce la pierderea memoriei, dureri de cap și afectarea capacitaților cognitive ale pacienților. Rareori pot fi tratate prin chimioterapie, dar pot fi îndepărtate ușor, mai ales datorită faptului că se află în afara creierului, ceea ce ușurează intervențiile chirurgicale. Mi-am propus realizarea unui program general ce poate detecta oricare dintre aceste tipuri de tumori. Setul de date ales conține slice-uri extrase din mai multe tomografe, pe oricare dintre cele 3 dimensiuni. În total setul de date conține 7023 de poze ce pot fi clasificate în 4 clase: gliomă, meningiomă, „notumor” și tumoră pineală.

### 3.2.5 Clasificarea imaginilor

În figura 75 avem prima arhitectură utilizată pentru clasificarea imaginilor.

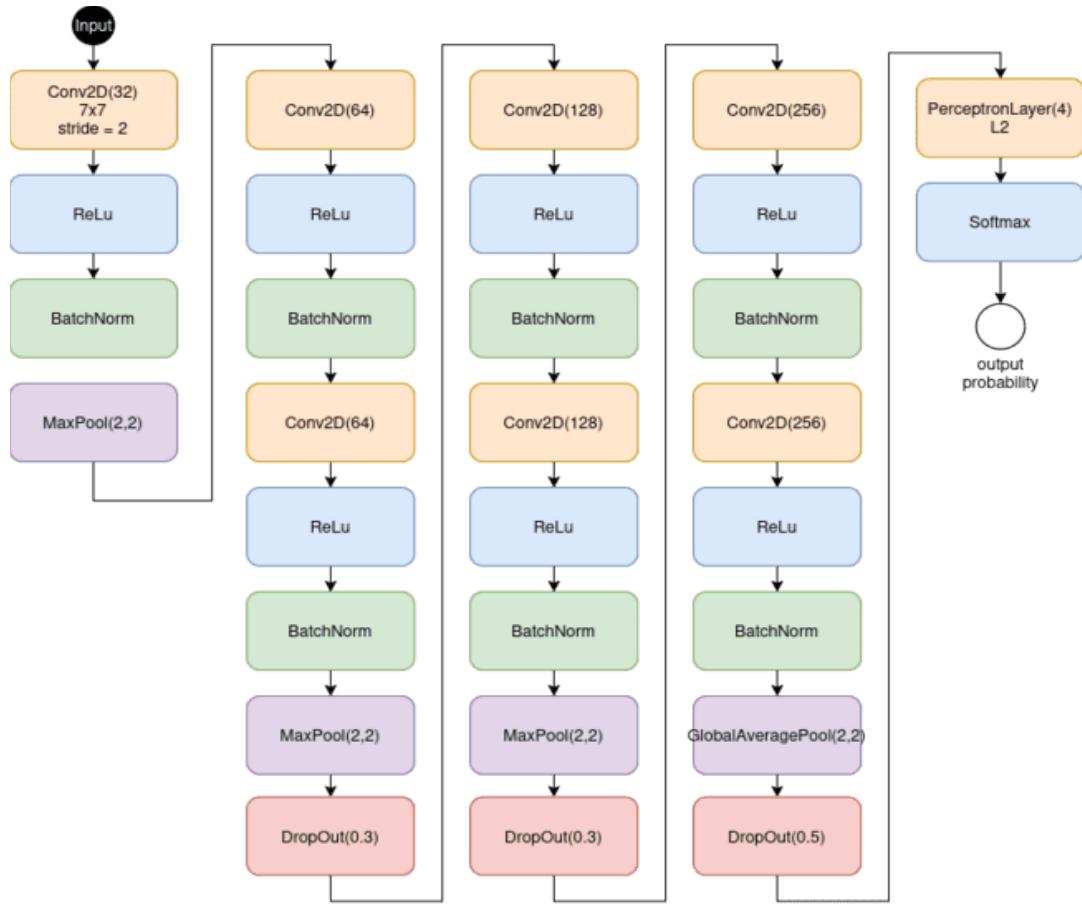


Figure 79: Clasificare Tumori - Model 1

Modelul din imagine preia elemente atât de la ResNet, cât și de la VGG, dar are cu mult mai puțini parametrii decât ambele model, ceea ce face modelul mai ușor de antrenat. Pentru că am vrut ca modelul să poată analiza structura completă a tumorilor, am adăugat o conoluție de  $7 \times 7$  utilizată de obicei în rețelele reziduale. Fiecare bloc are două conoluții de același tip înainte de pooling, ceea ce duce la o mai bună extracție a tiparelor de la același nivel. În documentația inițială a normalizării batch-urilor este recomandată normalizarea înainte de activare pentru că poate duce la rezultate ușor mai bune. Există și rețele ce utilizează normalizarea după calcularea activărilor. În documentația originală a rețelelor reziduale, normalizarea se realizează după calcularea activărilor.

Acuratețea modelului a fost una de 98.4%, obținută după 53 de epoci. În cazul gliomei nu există cazuri în care tumoră nu a fost detectată, dar în 4 instanțe modelul a prezis un alt tip de tumoră, o astfel de greșală poate fi specifică pozelor în care glioma pornește din extremitățile creierului sau aproape de glanda pineală. Menigioma nu este detectată la fel de ușor de către model ca restul claselor, având cele

mai multe cazuri nedetectate și cele mai multe cazuri clasificate incorect. Modelul are rezultate bune pentru tumorile pineale, doar un caz a fost clasificat ca un alt tip de tumoră.

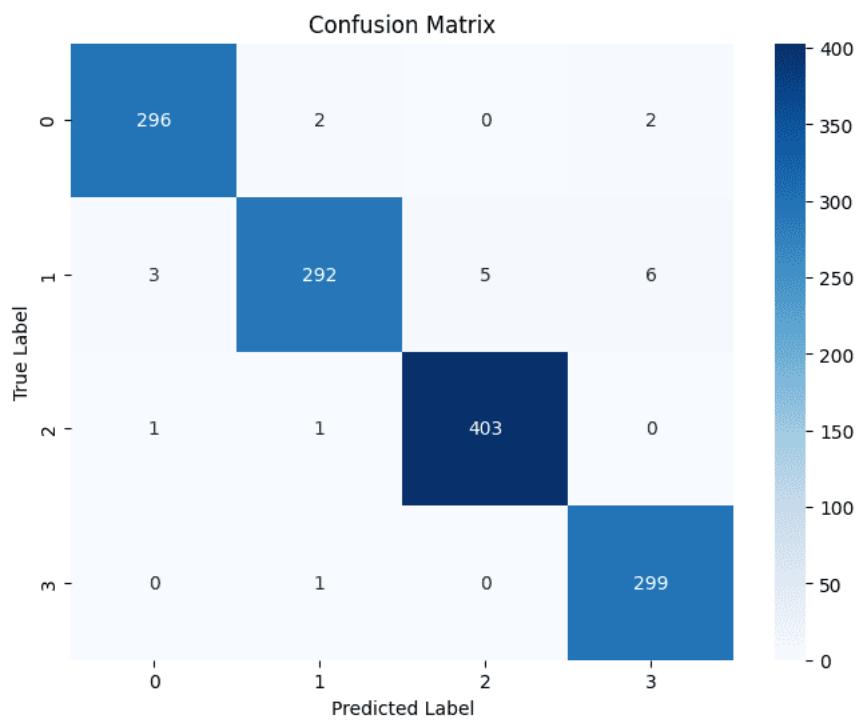
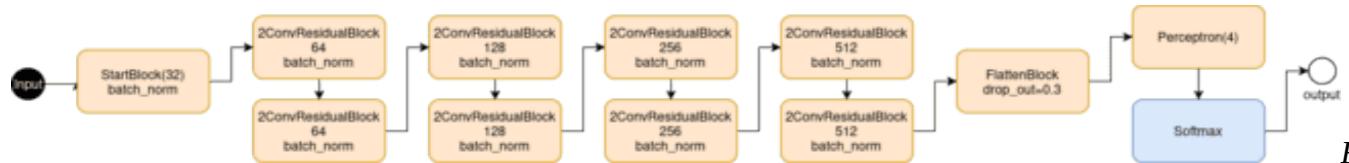


Figure 80: Clasificarea tumorilor - Matrice de confuzie model 1

Table 9: Rezultatele primului model

<b>Class</b>	<b>Precision</b>	<b>Recall</b>	<b>F1 Score</b>	<b>Support</b>
0	0.9899	0.9867	0.9883	300
1	0.9865	0.9574	0.9717	305
2	0.9878	0.9951	0.9914	405
3	0.9739	0.9967	0.9851	300

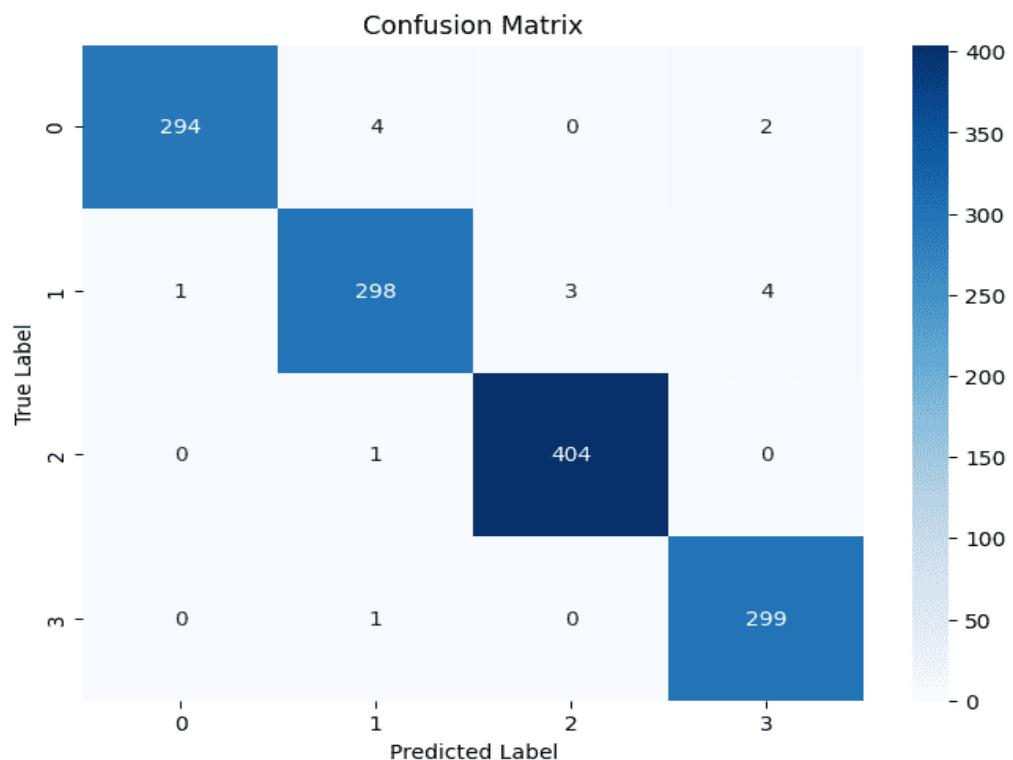
De data acesta vom utiliza o rețea reziduală completă.



Figure

81: Clasificarea tumorilor - Model 2

Modelul a fost antrenat în 61 de epoci și a obținut următoarele rezultate:



Acuratețea înregistrată nu a fost cu mult mai bună. Utilizând rețele reziduale performanța a crescut de la 98.4% la 98.8%. Chiar și aşa rețeaua nu a fost suficient de adâncă pentru a mări suficient performanțele modelului. Rețeaua curentă este aproape la fel de complexă ca cea folosită în cazul segmentării gliomei, deci nu putem mări numărul de straturi din cauza limitărilor hardware. Dar pentru a depășii limitările hardware ne putem folosi de învățarea prin transfer. Învățarea prin transfer

reprezintă reantrenarea unei rețele ce a fost antrenată pentru un caz general sau integrarea acesteia în altă rețea, ca mai apoi să fie specializată pentru un caz particular. Poate fi foarte utilă pentru că putem prelua rețele cu multe straturi și mai apoi să antrenăm doar o parte dintre acestea pentru a reduce consumul de resurse și timpul de antrenare. Am preluat o rețea reziduală cu 50 de straturi ce a fost antrenată cu setul ImageNet. Setul ImageNet este setul cu care a fost antrenat modelul AlexNet și poate recunoaște 1000 de obiecte. În setul de date nu se regăsesc imagini medicale, dar fiind antrenat pe un număr mare de obiecte cu forme și trăsături diferite rețeaua știe să separe și să distingă orice tip de obiect, chiar dacă nu le poate recunoaște. Vom preluă ultimele 21 de straturi din rețea (ResNet50 folosește blocuri reziduale cu 3 convoluții, alegem un număr de straturi divizibil cu 3) și pentru clasificare vom adăuga un perceptron cu 2 straturi.

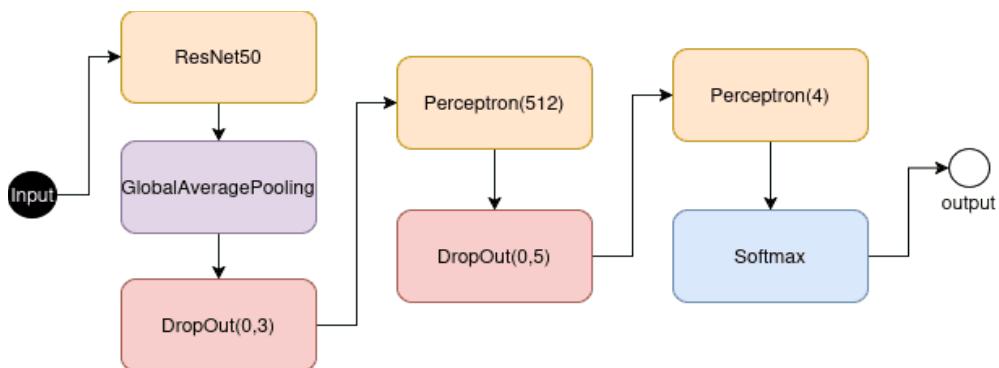
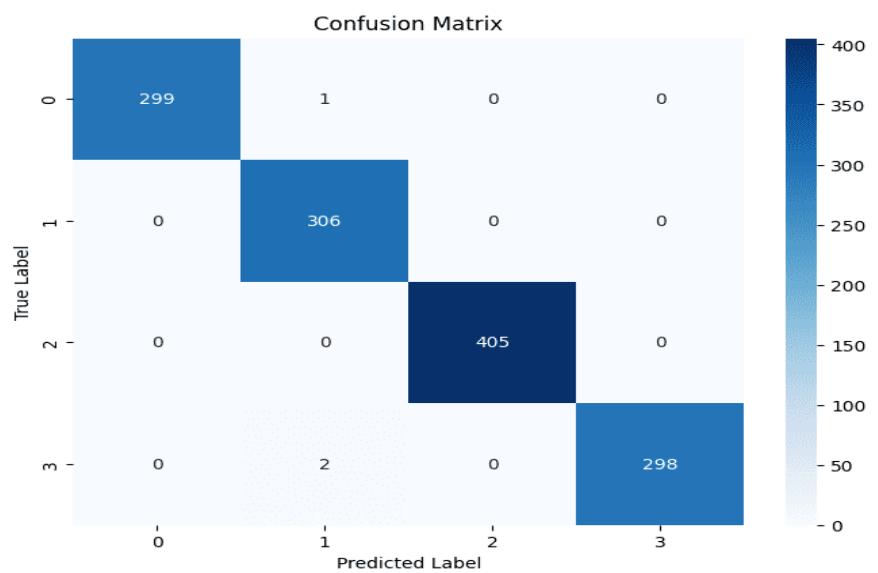


Figure 82: Clasificarea Tumorilor - ResNet50

Cu o acuratețe de 99.77%, rezultatele modelului au fost aproape perfecte. Doar 3 instanțe nu au fost clasificate corect, toate fiind clasificate incorect ca menigiomă. Dar nu există cazuri de false pozitiv și nu există nici cazuri în care prezența anomaliei nu a fost detectată.



Activări

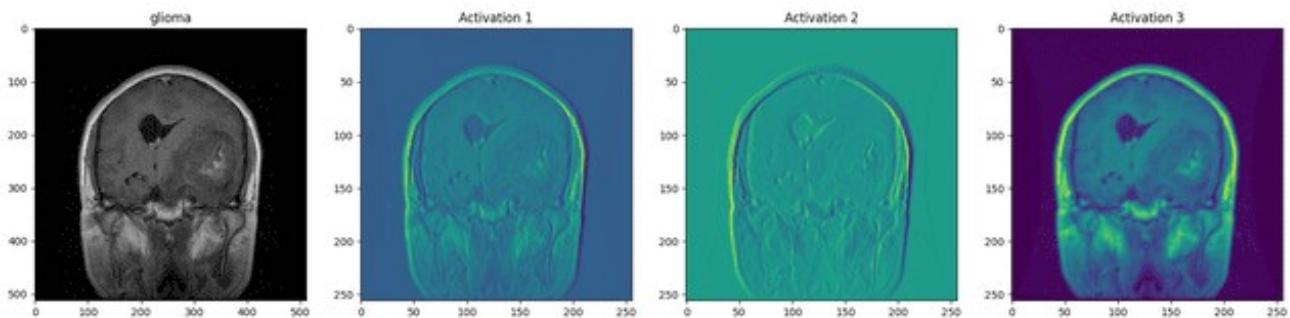


Figure 83: Glioma P1

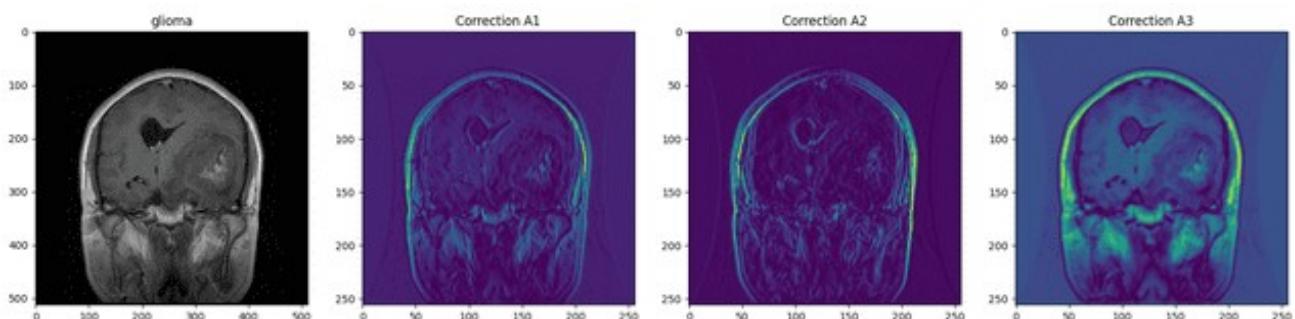


Figure 84: Glioma P1 – Corecție aplicată

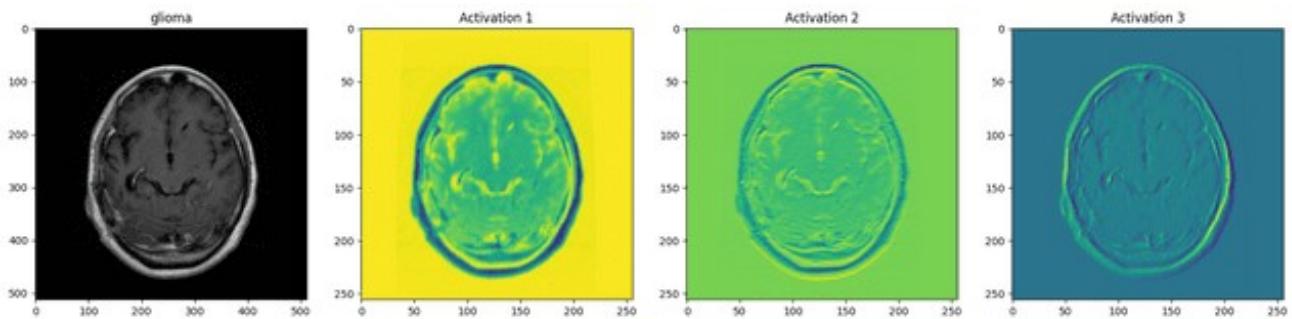


Figure 85: Glioma P2

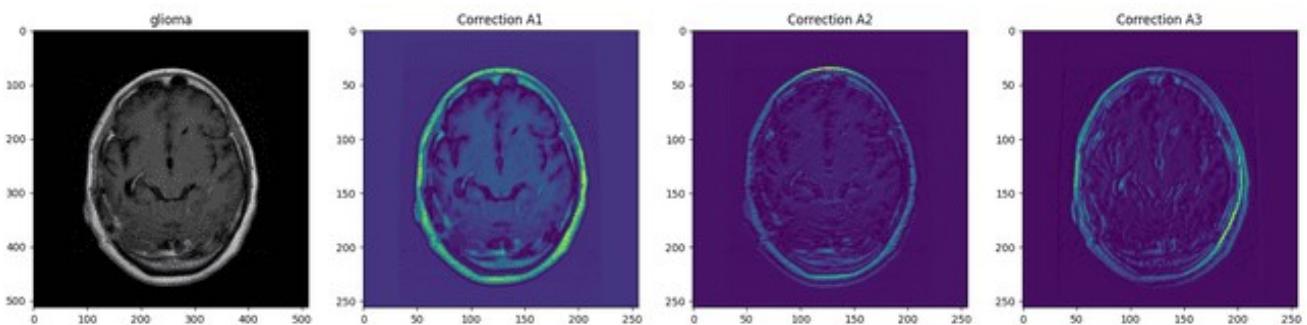


Figure 86: Glioma P2 - Corecție aplicată

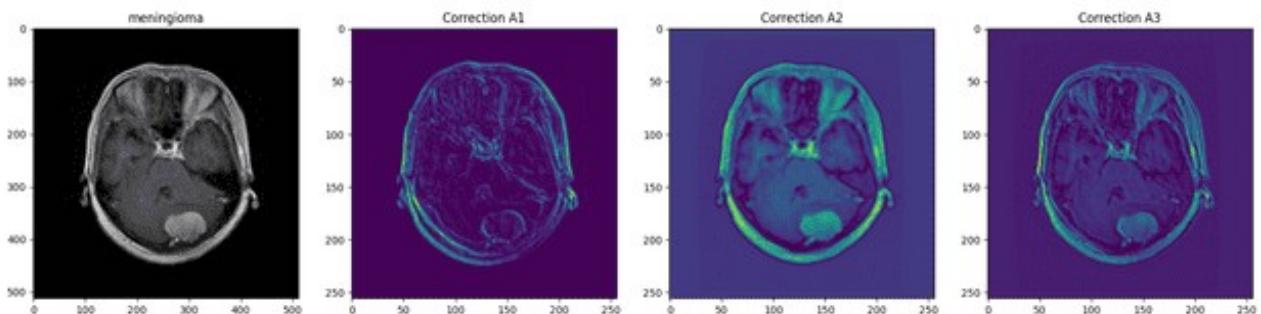


Figure 88: Meningiomă P1 – Corecție aplicată

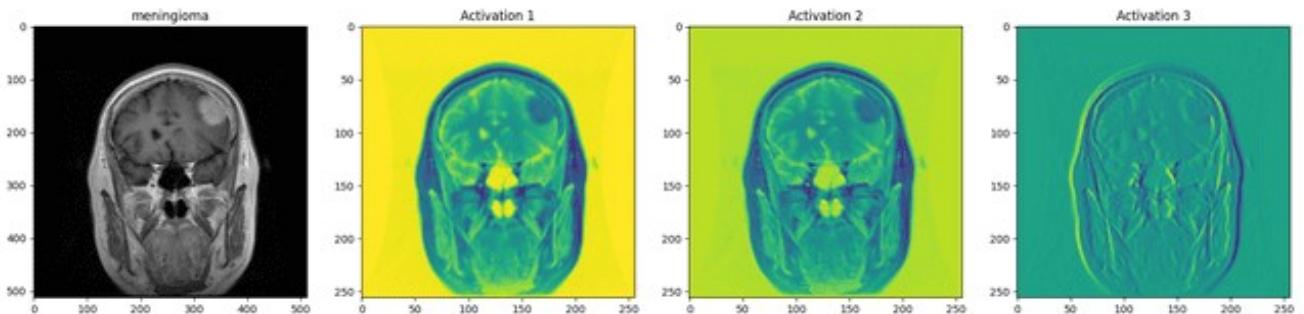


Figure 89: Meningiomă P2

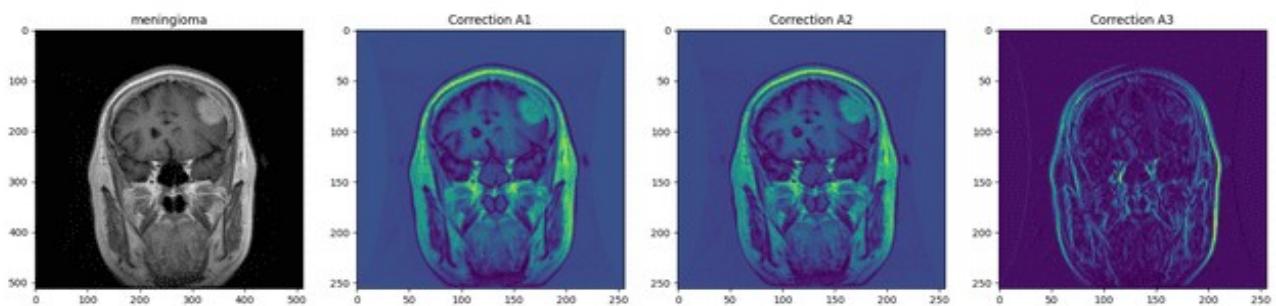


Figure 90: Meningiomă P2 – Corecție aplicată

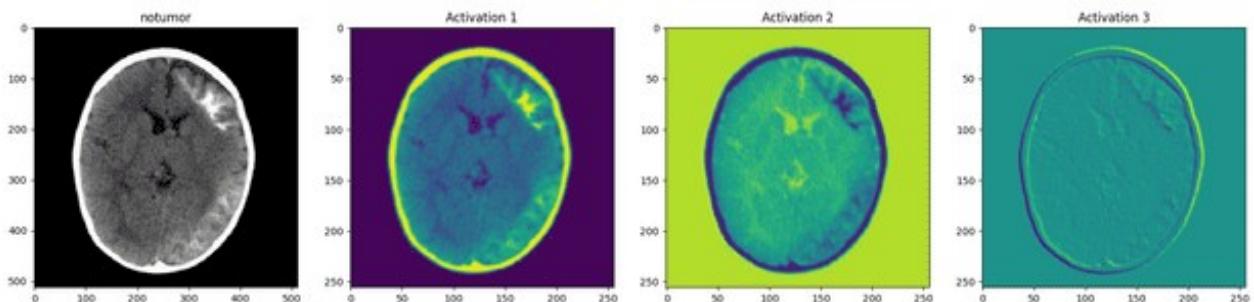


Figure 91: Notumor - P1

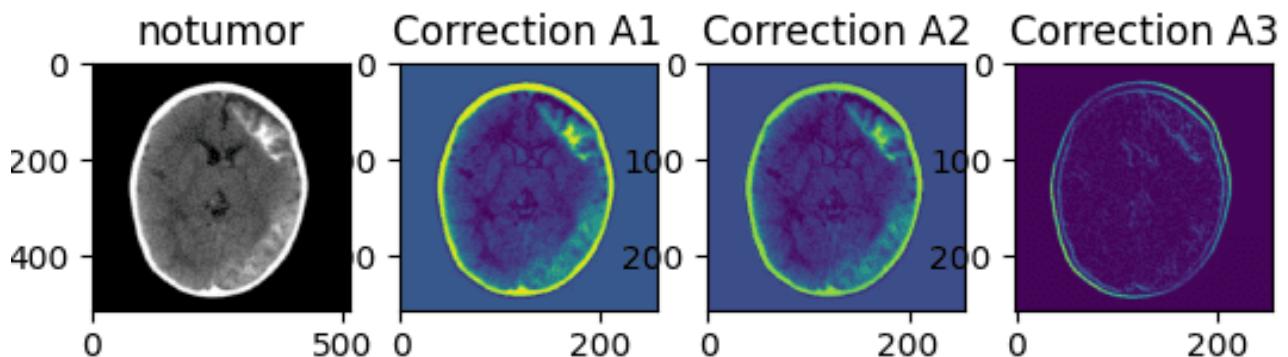


Figure 92: Notumor P1 - Corecție aplicată

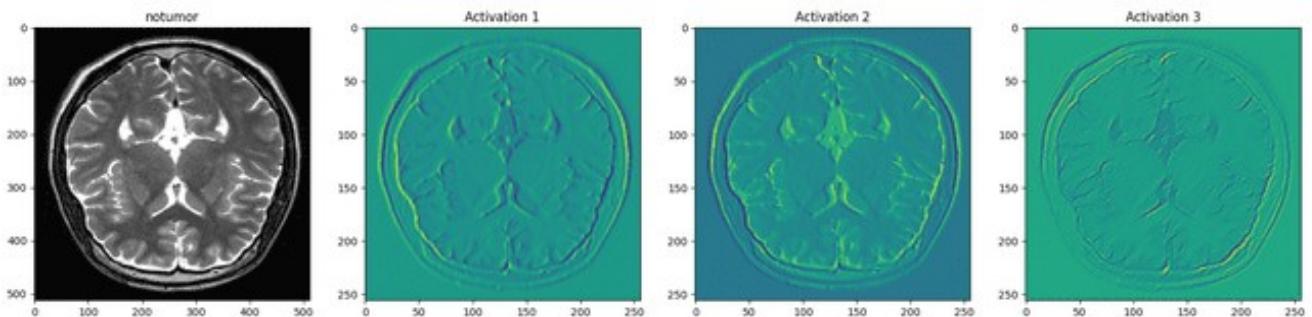


Figure 93: Notumor P2

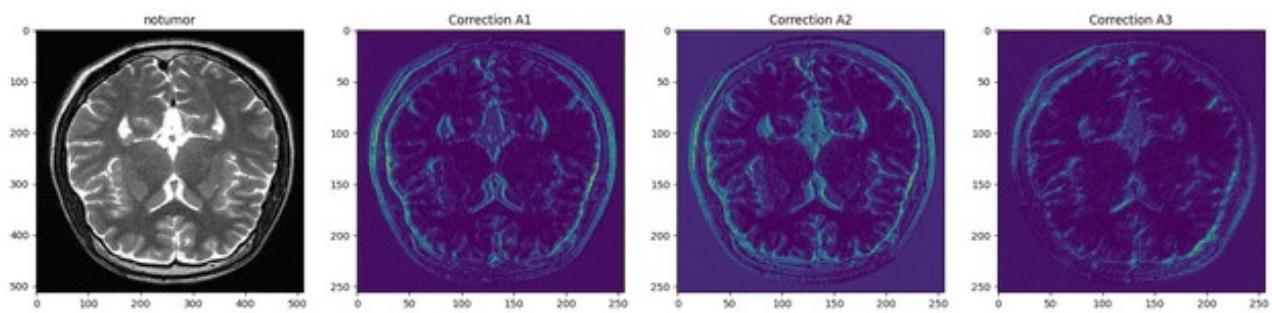


Figure 94: Notumor P2 - Corecție aplicată

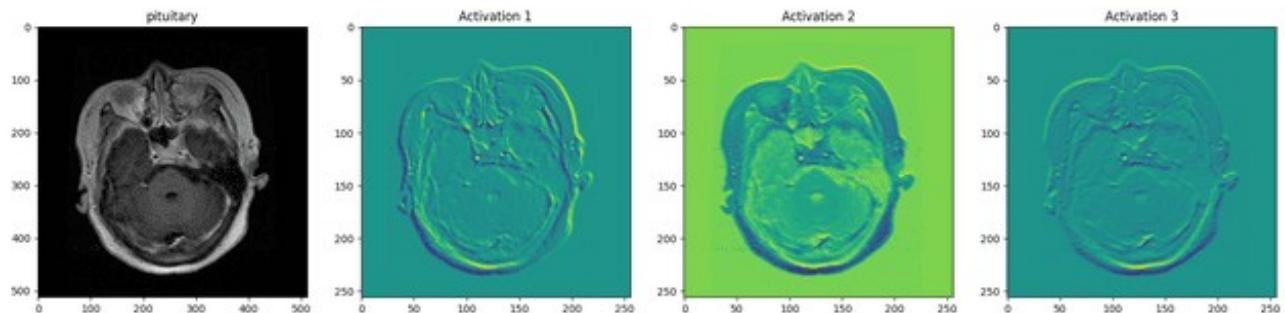


Figure 95: Tumoră Pineală P1

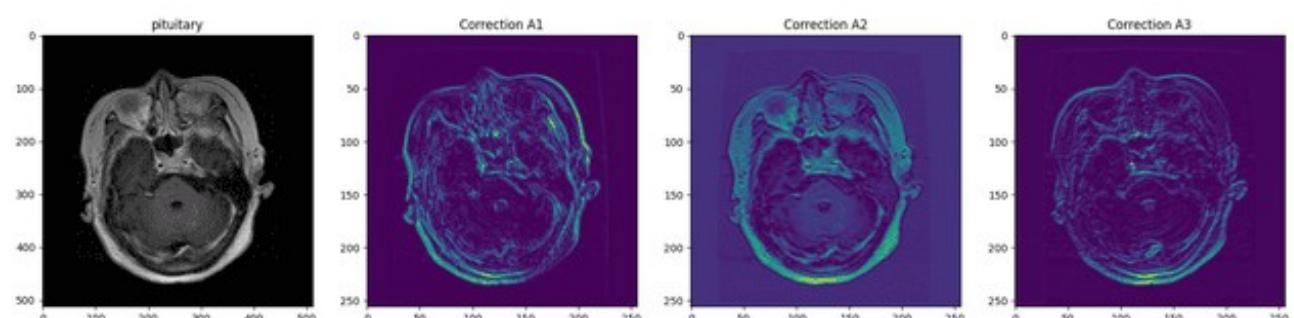


Figure 96: Tumoră pineală P1 - corecție aplicată

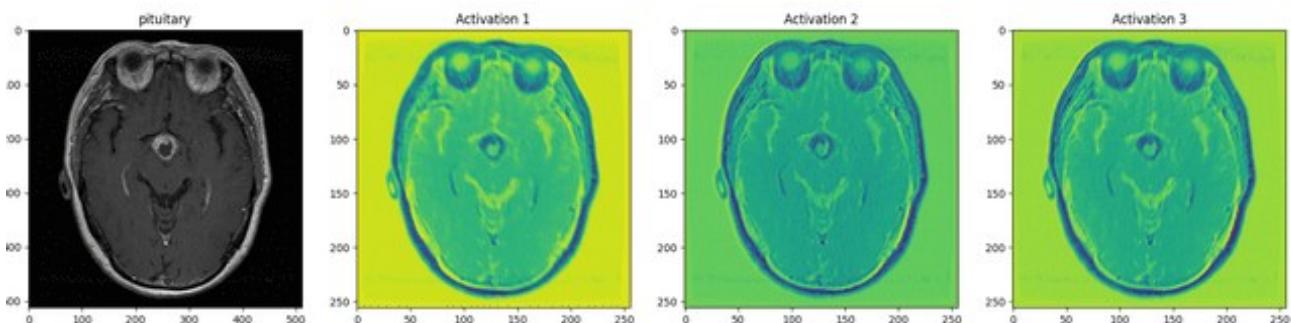


Figure 97: Tumoră pineală P2

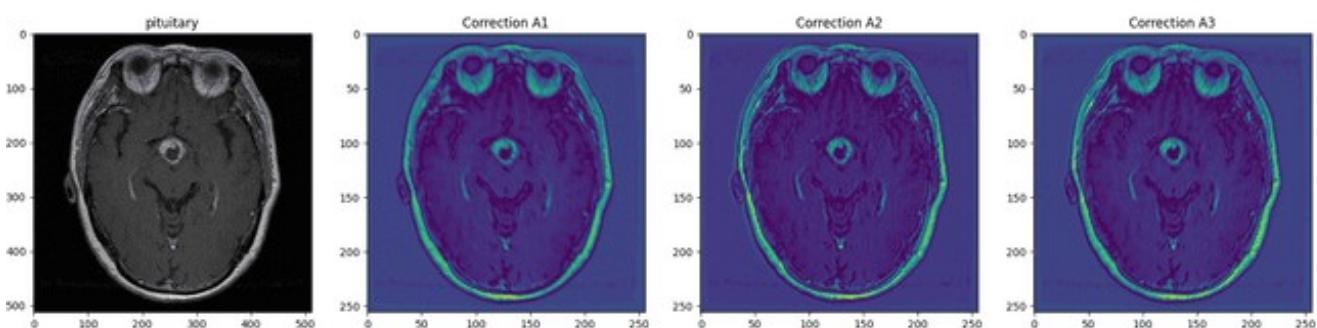


Figure 98: Tumoră pineală P2 - corecție aplicată

### 3.3 COVID-19

Bolile respiratorii, cum ar fi COVID-19 și pneumonia virală asociată, continuă să fie o preocupare majoră în sănătatea publică, deși severitatea infecției cu SARS-CoV-2 a scăzut față de perioada inițială a pandemiei, din cauza imunizării populației fie prin contractarea virusului sau prin vaccinare. Scăderea agresivității virusului este și o consecință adaptării virusului la organismul uman. Cu timpul, virusurile tend să devină mai ușor de transmis, dar mai puțin letale, deoarece decesul gazdei ar afecta răspândirea virusului.

Cu toate acestea, COVID-19 poate provoca complicații în rândul persoanele în vîrstă sau care suferă de alte afecțiuni. Una dintre complicații este pneumonia virală, virusul fiind capabil să ducă la inflamarea alveolelor pulmonare. Pneumonia virală este o inflamație a parenchimului pulmonar provocată direct de virusul SARS-CoV-2. Spre deosebire de pneumoniile bacteriene, care răspund la antibiotice, pneumoniile virale necesită tratament de susținere și, în unele cazuri, terapie antivirală.

În cazul peroanelor vulnerabile, boala se poate agrava într-un timp foarte scurt și este necesar ca afecțiunile să fie detectate într-un timp cât mai scurt.

### 3.3.1 Setul de date

Setul de date utilizat a fost conceput de Universitatea din Qatar și conține aproximativ 33000 de radiografii pulmonare. Radiografiile au fost împărțite în 4 clase: COVID-19, opacitate, normal și pneumonie virală. Cumulul de virusuri sau bacterii din plămâni îngreunează trecerea radiației prin aceștia. De aceea, infecțiile apar ca o ceată în interiorul plămânilor în radiografiile, făcând plămânul să devină opac (nu puteam vedea dincolo de el. Clasa „opacitate” arată faptul că opacitatea plămânlui nu a fost cauzată de infecția cu COVID-19, ci din cauza unei afecțiuni pe care modelul nu o poate recunoaște.

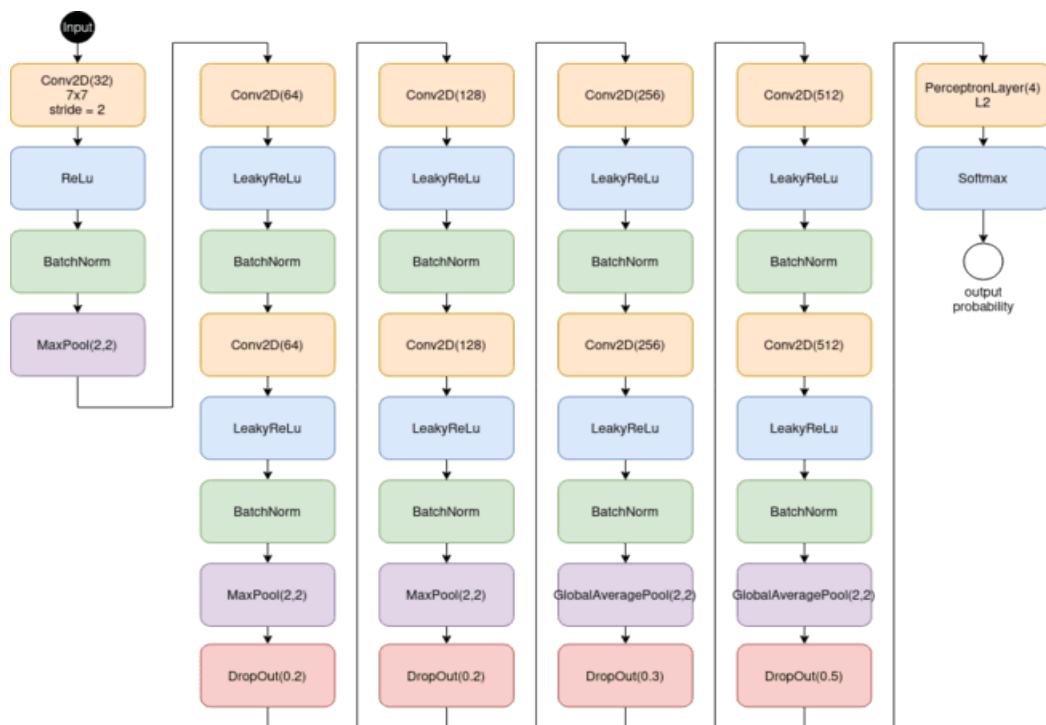


Figure 99: Covid19 - Model 1

Inițial modelul a obținut performanțe destul de slabe în ciuda complexității sale. Acuratețea era una de 88%. Diferențele dintre imaginile ce prezintă opacitate pot fi greu de sesizat. Ne trebuie o metodă pentru a face acele diferențe mai ușor de observat. Pentru că diagnosticarea infecțiilor pulmonare nu depinde atât de de mult de forma sau de poziția structurilor din imagine, am început prin aplicarea transformărilor geometrice. Augmentările sunt prezentate în figura 96. Aplicarea lor a dus la

o creștere a acurateții cu 2%. Dar nu rezolva problema lipsei de contrast specifică radiografiilor pulmonare.

```
'data_augmentation =Sequential([
    RandomFlip("horizontal"),
    RandomRotation(factor=2/360),
    RandomZoom(0.02),
    RandomTranslation(height_factor=0.02, width_factor=0.02),
])
```

multă de distribuția uniformă; aceasta duce la creșterea variației pixelilor din setul de date.

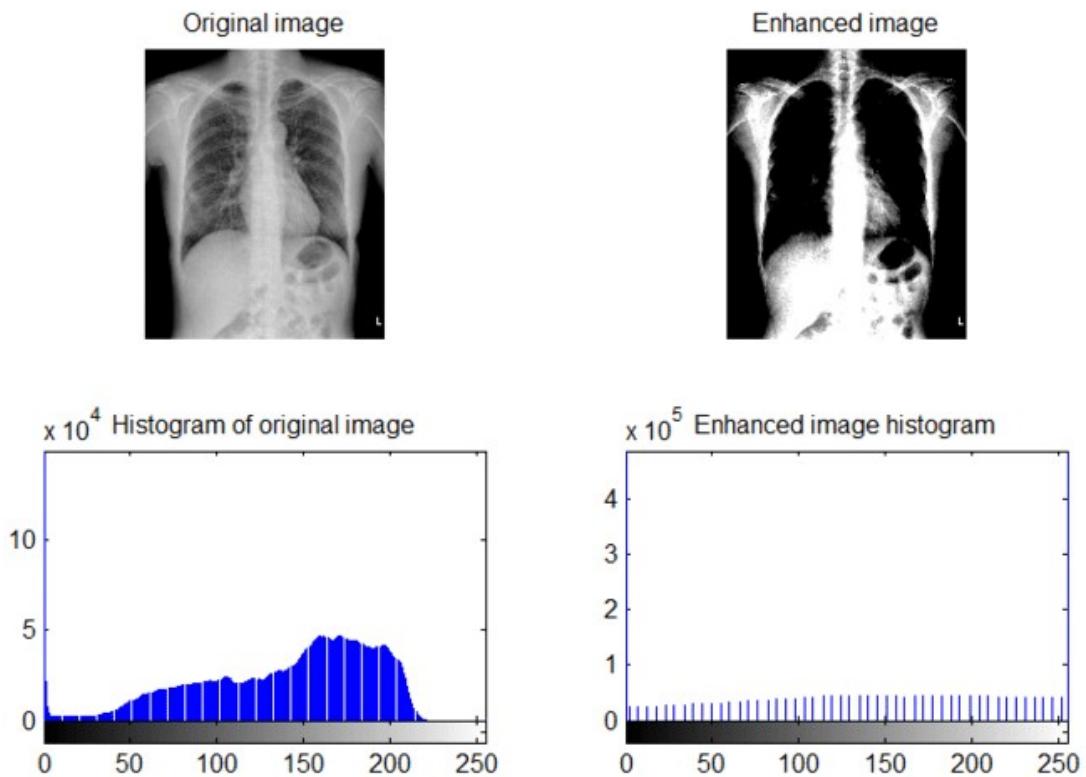


Figure 101: Uniformizarea histogramei în cazul radiografiilor pulmonare

```

def histogram_equalization(image):
    def _equalize(img):
        img = img.astype(np.uint8)
        r, g, b = cv2.split(img)
        r = cv2.equalizeHist(r)
        g = cv2.equalizeHist(g)
        b = cv2.equalizeHist(b)
        return cv2.merge([r, g, b])

    equalized = tf.numpy_function(_equalize, [image], tf.uint8)
    equalized.set_shape([256, 256, 3])
    equalized = tf.cast(equalized, tf.float32)

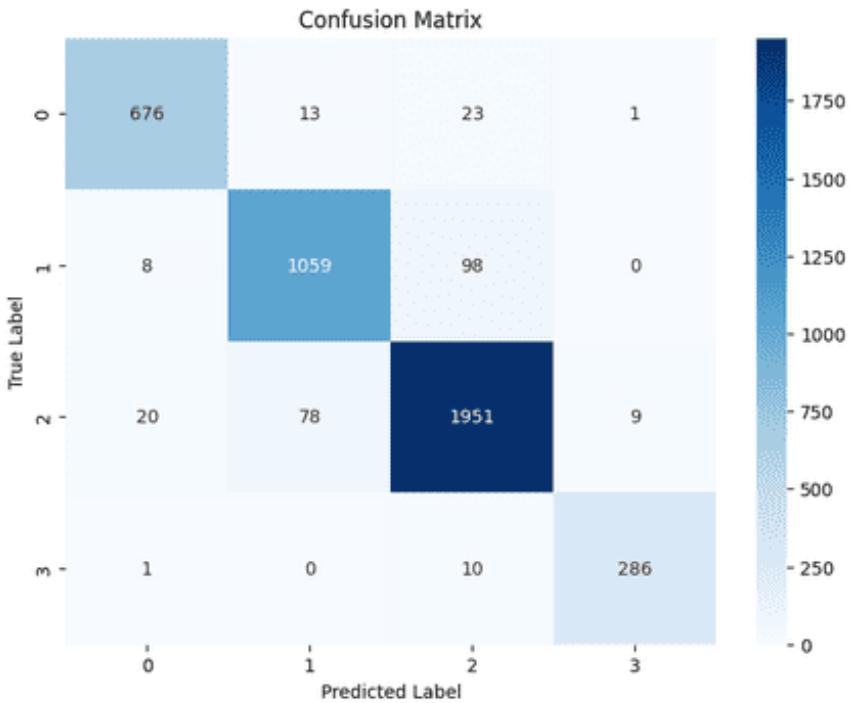
    return equalized

```

*Figure 102: Funcția pentru aplicarea uniformizării*

Cum pozele din setul de date sunt în formatul RGB, trebuie să aplicăm histograma de culori asupra fiecărui canal din imagine în parte, pentru că fiecare canal poate avea o distribuție diferită, figura 98.

Au fost necesare 58 de epoci pentru a ajunge la convergența și modelul a avut o acuratețe de 93.83%.



Accuracy: 93.83%

```
In [12]: from sklearn.metrics import classification_report
print(classification_report(real_labels, predictions))

      precision    recall  f1-score   support
0       0.96     0.95     0.95      713
1       0.92     0.91     0.91     1165
2       0.94     0.95     0.94     2058
3       0.97     0.96     0.96      297

   accuracy                           0.94      4233
  macro avg       0.95     0.94     0.94      4233
weighted avg       0.94     0.94     0.94      4233
```

Figure 103: Măsurători după uniformizare

După cum se poate observa modelul prezintă dificultăți în separarea cazurilor de opacitate de cele normale. Am încercat să reutilizez arhitectura ResNet utilizată anterior pentru clasificarea tumorilor cerebrale, ce utilizează modulul definit în capitolul 2. Deși nu a fost obținut cele mai bune rezultate, a dus la îmbunătățirea performanțelor pentru setul de date utilizat anterior.

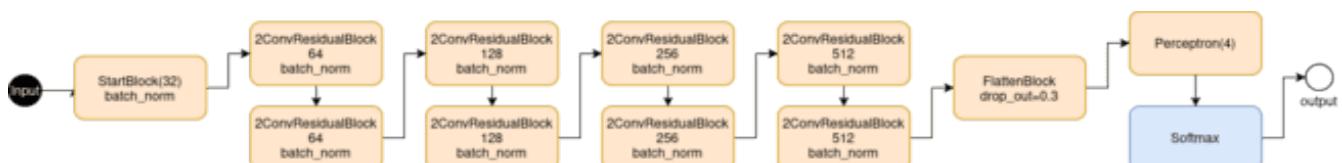


Figure 104: Arhitectura utilizată în cazul clasificării tumorilor.

În figura 101 avem rezultatele finale pentru clasificarea radiografiilor pulmonare. De data asta convergența a fost atinsă sub 20 de epoci de antrenament. Performanțele pentru clasele COVID-19 și pneumonie virală au fost mult mai bune decât cele ale modelului precedent, dar în continuare avem două clase ce nu pot fi separate complet. Rezultatul este însă unul destul de bun. Fără a folosi modele cu un număr foarte mare de parametrii, cel mai bun model avea o acuratețe de 96.3%.

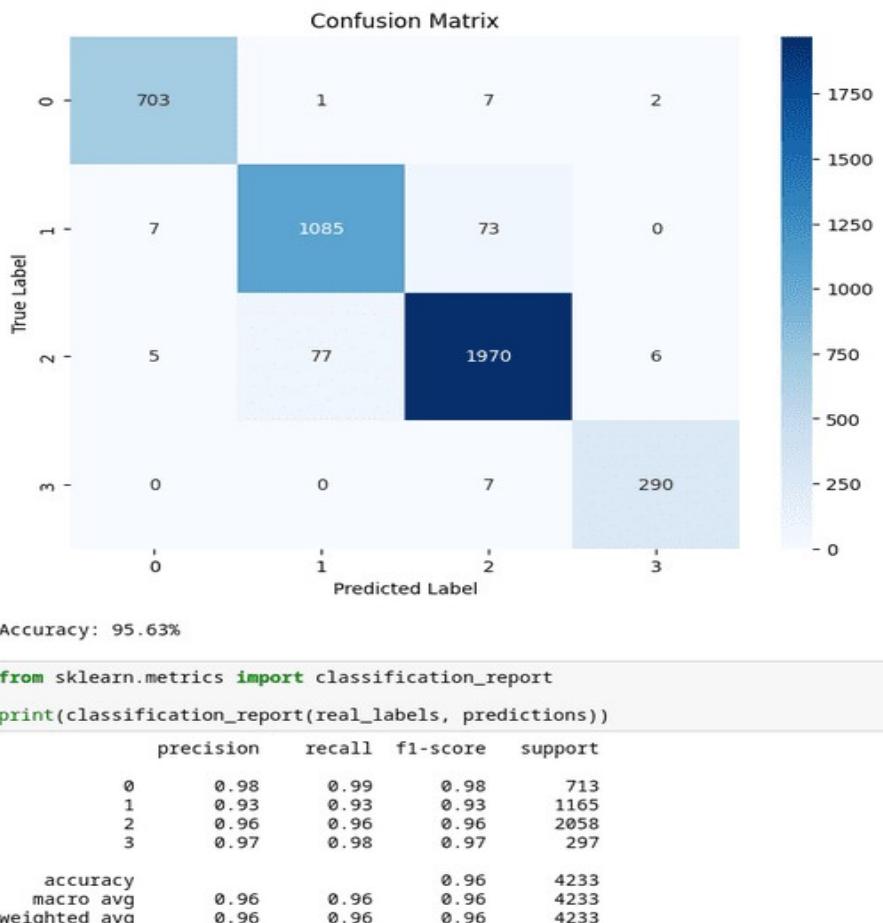


Figura 105. Rezultate finale - clasificarea radiografiilor pulmonare

### 3.4 OCT Retinal

OCT (Optical Coherence Tomography) este o tehnică imagistică non-invazivă de înaltă rezoluție, utilizată pentru vizualizarea structurilor interne ale retinei și ale segmentului anterior al ochiului. OCT-ul funcționează pe baza interferometriei optice, folosind lumina coerentă pentru a produce secțiuni transversale ale țesuturilor, asemănătoare cu cele obținute prin ecografie, dar la o rezoluție mult superioară (până la 5-10 μm).

OCT-ul măsoară reflexia luminii din diferite straturi ale retinei. Un fascicul de lumină cu coerență scurtă este dirijat către ochi, iar reflexiile de la fiecare strat retinal sunt analizate pentru a crea o imagine 2D sau 3D detaliată a retinei.

OCT-ul este esențial în oftalmologie pentru:

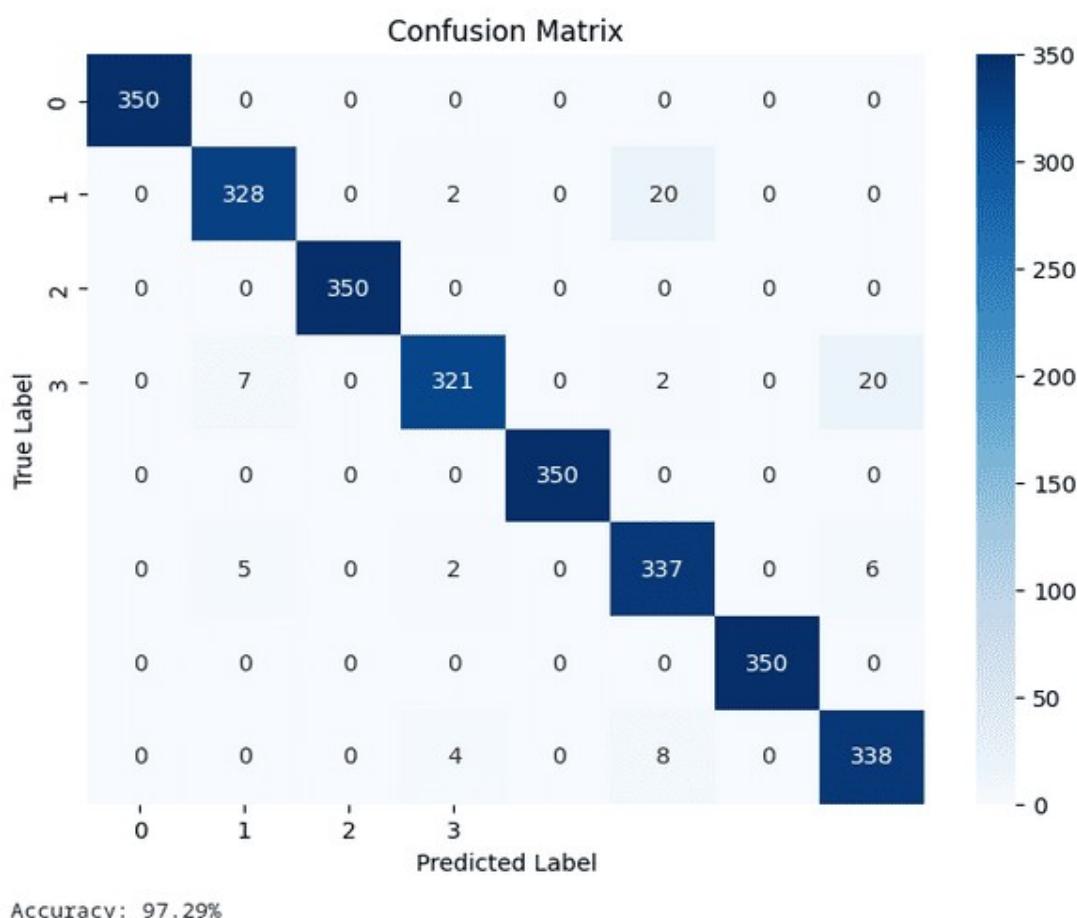
- Diagnostic precoce al bolilor retiniene
- Monitorizarea progresiei bolii
- Evaluarea răspunsului la tratament

În tabelul de mai jos sunt afecțiunile ce vor fi detectate.

<b>Clasă</b>	<b>Afecțiune</b>	<b>Descriere medicală</b>	<b>Semnătura în OCT</b>
<b>AMD</b>	Degenerescență Maculară Legată de Vârstă	Afectează macula (zona centrală a retinei) la persoanele vîrstnice. Poate duce la pierdere drusen, atrofie retiniană centrală a vederii.	Modificări ale grosimii retinei, vârstnice. Poate duce la pierdere drusen, atrofie retiniană centrală a vederii.
<b>CNV</b>	Neovascularizație Coroidiană	Apare în AMD umedă; vase de sânge anormale cresc sub retină subretinian, îngroșări anormale	În OCT: acumulare de fluid sângelui sub retină subretinian, îngroșări anormale
<b>CSR</b>	Retinopatie Centrală	Serosă Acumulare de lichid sub retina centrală	În OCT: decolări seroase ale retinei, spații clare subretiniene
<b>DME</b>	Edem Diabetic	Complicație a diabetului în care retina se umflă din cauza surgerii lichidelor	În OCT: chisturi intraretiniene, îngroșare retinală difuză
<b>DR</b>	Retinopatie Diabetică	Afecțiune progresivă a retinei cauzată de diabet	Microanevrisme, exsudate dure, edem, hemoragii, în OCT: distorsiuni ale arhitecturii retinei
<b>DRUSEN</b>	Depozite galbene sub retină	Drusenii sunt acumulații de materiale celulare sub RPE (retinal pigment epithelium)	În OCT: protuberanțe sub RPE, modificări topografice subtile
<b>MH</b>	Gaură Maculară	Ruptura în centrul maculei, ducând la vedere încețoșată sau separare evidentă a straturilor	În OCT: defect clar în fovee, separare evidentă a straturilor

Clasă	Afectiune	Descriere medicală	Semnătura în OCT
<b>NORMAL</b> Ochi sănătoși		lipsă de vedere centrală Retină normală, fără semne de boală	retiniene Structură stratificată uniformă, fără acumulații de lichid sau rupturi

Am decis să reutilizez primul model folosit pentru clasificarea tumorilor cerebrale ca punct de placare. Suprinzător, a obținut rezultate foarte bune pentru setul de tomografe retinale.



*Figure 106: Matrice de confuzie OCT*

Pietrele renale, denumite în sistemul urinar. Acestea pot varia în ceea ce privește dimensiunea, compoziția chimică și localizarea, fiind clasificate în funcție de substanță predominantă (oxalat de calciu, fosfat de calciu, acid uric, struvit sau cistină). Formarea lor este influențată de factori precum hidratarea insuficientă, aportul

ridicat de sodiu și oxalați din dietă, afecțiuni metabolice (hipercalciurie, hiperuricemie) sau predispoziții genetice. Simptomatologia include dureri lombare severe (colică renală), hematurie, grețuri și disurie, iar complicațiile pot duce la obstrucție ureterală sau infecții urinare recurente.

Diagnosticul precis al calculilor renali se bazează pe investigații imagistice avansate. Ecografia renală reprezintă o metodă accesibilă și non-invazivă, dar are limitări în detectarea calculilor de dimensiuni reduse sau localizați în uretere. Tomografia computerizată (CT) fără contrast este considerată *gold standard*-ul datorită sensibilității și specificității ridicate, permășând identificarea chiar a calculilor submillimetri. Radiografia abdominală convențională are o utilitate redusă, deoarece unele tipuri de calculi (în special cei urici) sunt radiotransparenți. Rezonanța magnetică nucleară (RMN) este utilizată mai rar, dar constituie o alternativă valoroasă pentru pacienții care necesită evitarea expunerii la radiații ionizante, cum ar fi gravidele sau copiii.

Abordarea terapeutică este stabilită în funcție de dimensiunea, localizarea și compoziția calculilor. Pentru calculi de dimensiuni mici (sub 5 mm), managementul conservator este de preferat, incluzând hidratare intensă, administrare de antiinflamatoare nesteroidiene și terapie medicală expulsivă (alfa-blocante). Modificările dietetice joacă un rol esențial în prevenirea recurenței, prin reducerea consumului de sodiu, oxalați și proteine animale. În cazul calculilor mari sau complicați, sunt indicate intervenții invazive, precum litotriția extracorporeală cu unde de soc (ESWL), ureteroscopia cu laser sau nefrolitotomia percutaneousă (PCNL).

```

from keras.src.models import Model

input = Input(shape=(256, 256, 3))
start = StartBlock(input, filter_size=64, kernel_size=(7, 7), strides=2,batch_normalization=True)

residual1 = SimpleResidualBlock(start, filter_size=64, kernel_size=(3, 3), padding="same",
                                batch_normalization=True)
residual2 = SimpleResidualBlock(residual1, filter_size=64, kernel_size=(3, 3), padding="same",
                                batch_normalization=True,drop_out=0.2)

residual3 = SimpleResidualBlock(residual2, filter_size=128, kernel_size=(3, 3), padding="same",scale=True,
                                batch_normalization=True)
residual4 = SimpleResidualBlock(residual3, filter_size=128, kernel_size=(3, 3), padding="same",
                                batch_normalization=True,drop_out=0.2)

residual5 = SimpleResidualBlock(residual4, filter_size=256, kernel_size=(3, 3), padding="same",scale=True,
                                batch_normalization=True)
residual6 = SimpleResidualBlock(residual5, filter_size=256, kernel_size=(3, 3), padding="same",
                                batch_normalization=True,drop_out=0.2)

residual5 = SimpleResidualBlock(residual4, filter_size=512, kernel_size=(3, 3), padding="same",scale=True,
                                batch_normalization=True)
residual6 = SimpleResidualBlock(residual5, filter_size=512, kernel_size=(3, 3), padding="same",
                                batch_normalization=True,drop_out=0.2)

flatten = FlattenResidualBlock(residual6,drop_out=0.5)
output = Dense(1, activation='sigmoid',kernel_regularizer=L2(0.001))(flatten)

model = Model(inputs=input, outputs=output)

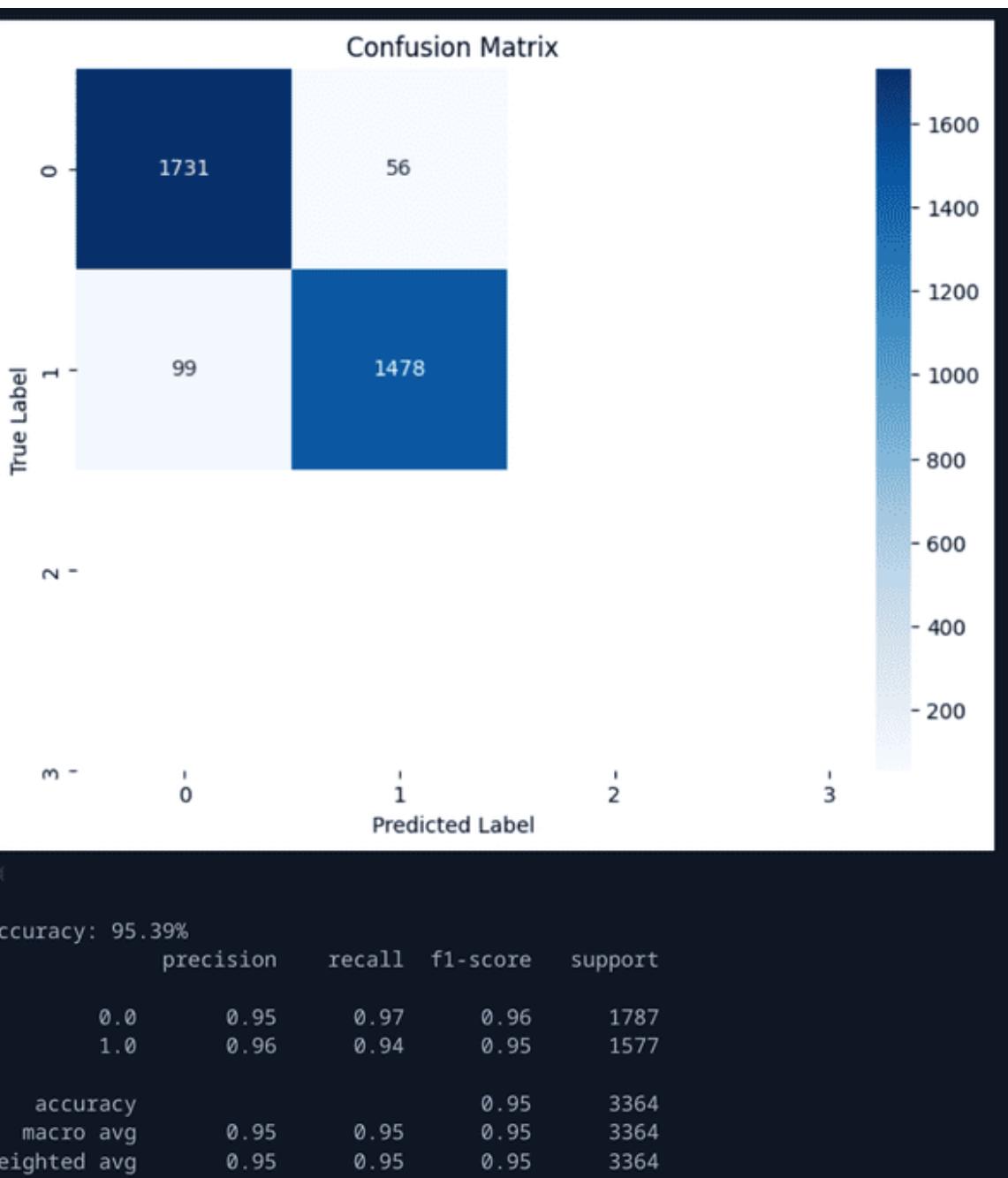
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
    loss="binary_crossentropy",
    metrics=['accuracy', tf.keras.metrics.AUC()])
)

model.summary()

```

Figure 107: Rețea reziduală

Pentru detectarea pietrelor renale am utilizat modelul din figura 108. Am utilizat și alte rețele, cum ar fi arhitecturi VGG și ResNet deja antrenate, însă toate au dus la aceleași valori în matricea de confuzie. Setul de date utilizat conținea doar slice-uri din tomografe, ceea ce a dus la pierderea informațiilor necesare pentru clasificare. Mai jos avem, avem matricea de confuzie. Raportat la numărul de instanțe din setul de test, rezultatul pare a fi decent, dar nu suficient pentru o astfel de clasificare binară.



## Concluzii

Programele de inteligență artificială pot asista medicii în procesul de diagnosticare și pot micșora numărul de cazuri ce nu au fost diagnosticate corect, din cauza erorii umane. De asemenea, pot reduce timpul necesar diagnosticării, pentru a-i permite medicului să se ocupe de alți pacienți. Programele pentru segmentarea și clasificarea leucemiei, clasificarea leucemiei, cel pentru analiza tomografiilor oculare, dar și cel pentru analiza radiografiilor pulmonare, atunci când ne interesează strict detectarea

virusului COVID-19 sau a pneumoniei virale cauzată de acesta, ci nu ne interesează detectarea altor afecțiuni, oferă un grad ridicat de încredere și pot avea multe aplicații în clinicile medicale. Utilizarea programului de segmentare pentru tumorile cerebrale poate fi totuși mai dificil de utilizat, deoarece acesta detectează prezența gliomelor, dar nu poate delimita complet strutura acestora. Ca un medic să înțeleagă mai bine structura gliomei, acesta va trebui să ajusteze manual marginile generate de model

# Biografie

WNorb: Wikipedia, , ,

YNW: Newthink, The Mathematician Forced to become a genius, 2025,

<https://www.youtube.com/watch?v=MrvePmj20eo>

NW2: MPNeuro, From Cybernetics to AI: the pioneering work of Norbert Wiener, ,

<https://maxplanckneuroscience.org/from-cybernetics-to-ai-the-pioneering-work-of-norbert-wiener/>

YS1: Newthink, ROThe Man Who Should Be As Famous As Einstein , 2025,

[https://www.youtube.com/watch?v=0wlmzvf8\\_gI](https://www.youtube.com/watch?v=0wlmzvf8_gI)

WS: Wikipedia, Claude Shannon, , [https://en.wikipedia.org/wiki/Claude\\_Shannon](https://en.wikipedia.org/wiki/Claude_Shannon)

ASAR: Claude Shannon, A Symbolic Analysis of Relay and Switching Circuits, 1938

BL: Wikipedia, Bell Labs, , [https://en.wikipedia.org/wiki/Bell\\_Labs](https://en.wikipedia.org/wiki/Bell_Labs)

AMAP: Jimmy Soni, Rob Goodman, A Mind At Play,

AMTC: Claude E. Shannon, A Mathematical Theory of Communication, 1948

WMC: Wikipedia, Mechanical Calculator, , [https://en.wikipedia.org/wiki/Mechanical\\_calculator](https://en.wikipedia.org/wiki/Mechanical_calculator)

WDP: Wikipedia, The Decision Problem, , [https://en.wikipedia.org/wiki/Decision\\_problem](https://en.wikipedia.org/wiki/Decision_problem)

WEDP: Wikipedia, Entscheidungsproblem, , <https://en.wikipedia.org/wiki/Entscheidungsproblem>

ALCG: James Grime, ROAlan Turing - Celebrating the life of a genius , 2012,

<https://www.youtube.com/watch?v=gtRLmL70TH0>

YT1: Up and Atom, ROTuring Machines - How Computer Science Was Created By Accident , ,

<https://www.youtube.com/watch?v=v=PLVCscCY4xI>

CTT: Alonzo Church, Alan Turing, The Curch-Turing Thesis, 1936

OCN: Alan Turing, ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE

ENTSCHEIDUNGSPROBLEM, 1936

YT2: Computerphile, ROTuring Machines Explained - Computerphile , 2014,

<https://www.youtube.com/watch?v=dNRDvLACg5Q>

WENIG: Wikipedia, Enigma Machine, , [https://en.wikipedia.org/wiki/Enigma\\_machine](https://en.wikipedia.org/wiki/Enigma_machine)

YENIG: Jared Owen, Cum funcționa Mașina Enigma? , 2021, <https://www.youtube.com/watch?v=ybkkiGtJmkM>

YENIG2: Numberphile, ROFlaw in the Enigma Code - Numberphile , ,

<https://www.youtube.com/watch?v=V4V2bpZlqx8>

BOMBE: Wikipedia, Bombe, , <https://en.wikipedia.org/wiki/Bombe>

ACMI: Alan Turing, COMPUTING MACHINERY AND INTELLIGENCE, 1950

TCHAMP: Wikipedia, Turochamp, , <https://en.wikipedia.org/wiki/Bombe>

PWPWM: Walter Pitts, Warren S. McCulloch , A LOGICAL CALCULUS OF THE IDEAS  
IMMANENT INNERVOUS ACTIVITY, 1943

YPFR: Asianometry, The First Neural Networks, 2024, <https://www.youtube.com/watch?v=e5dVSygXbAE>

PPN: Frank Rosenblatt, Principles of Neurodynamics, 1957

YPFTN: Spanning Tree, ROPerceptrons: The First Trainable Neural Networks | Teaching Computers to Learn, Part 3 , , <https://www.youtube.com/watch?v=Ip6RIHwi21c>

PMPP: Marvin Minsky, Seymour Papert, Perceptrons. An Introduction to Computational Geometry, 1969

WBPROP: Wikipedia, Backpropagation, , <https://en.wikipedia.org/wiki/Backpropagation>

BPLRBP: David E. Rumelhart, Geoffrey Hinton, Ronald J. Williams,, Learning representations by back-propagation errors, 1986

TRBGRAD: , Implementare Gradient Descent pe caz bidiimensional, ,  
[https://github.com/TRBogdann/Licenta/blob/main/Exemple/gradient\\_descent\\_1.py](https://github.com/TRBogdann/Licenta/blob/main/Exemple/gradient_descent_1.py)

TRGRAD2: , Algoritm Gradient Descent caz tridimensional, ,  
[https://github.com/TRBogdann/Licenta/blob/main/Exemple/gradient\\_descent\\_2.py](https://github.com/TRBogdann/Licenta/blob/main/Exemple/gradient_descent_2.py)

BCF: Wikipedia, Loss Function, , [https://en.wikipedia.org/wiki/Loss\\_function](https://en.wikipedia.org/wiki/Loss_function)

TRBREG: , Logistic and Linear Regression implemenation in C++, ,  
[https://github.com/TRBogdann/ML-II-Regressions-CPP\\_Python/tree/main/DataScience\(C%2B%2B\)](https://github.com/TRBogdann/ML-II-Regressions-CPP_Python/tree/main/DataScience(C%2B%2B))

NCAG: Goerge F. Simmons, CALCULUS WITH ANALYTIC GEOMETRY, 1985

YBP3: 3blue1brown, Backpropagation calculus | Deep Learning Chapter 4, ,  
<https://www.youtube.com/watch?v=tIeHLnjs5U8>

BGFG: geeksforgeeks, Backpropagation in Neural Network, , <https://www.geeksforgeeks.org/machine-learning/backpropagation-in-neural-network/>

YGDM: Welch Labs, ROThe Misconception that Almost Stopped AI [How Models Learn Part 1] , 2025, <https://www.youtube.com/watch?v=NrO20Jb-hy0>

TRBBP: , Backpropagation from scratch, ,  
<https://github.com/TRBogdann/Licenta/tree/main/Exemple/backpropagation>

YBGRB1: 3blue1brown, But what is a neural network? | Deep learning chapter 1, ,  
<https://www.youtube.com/watch?v=aircAruvnKk>

GPUBHNN: Keith B. Foote, A Brief History of Neural Networks, 2021

WN256: Wikipedia, Nvidia 256, , [https://en.wikipedia.org/wiki/GeForce\\_256](https://en.wikipedia.org/wiki/GeForce_256)

GLWIKI1: Khronos Group, History of programmability,

BSGPU: Ian Buck, Tim Foley, Daniel Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, Pat Hanrahan, Brook for GPUs: Stream Computing on Graphics Hardware,

WANET: Wikipedia, AlexNet, , <https://en.wikipedia.org/wiki/AlexNet>

ANET: Alex Krizhevsky, ImageNet Classification with Deep Convolutional Neural Networks,

TIHOAI: Tim Mucci, The history of AI,

DNAMR: Serdar Abut, Hayrettin Okut, Rosey Zackula, Ken James Kallail, Deep Neural Networks and Applications in Medical Research, 2023

WIKICONV: Wikipedia, Convoluție, , <https://ro.wikipedia.org/wiki/Convoluție>

YCONV: 3blue1brown, ROBut what is a convolution? , , <https://www.youtube.com/watch?v=KuXjwB4LzSA>

GL1L2: geeks4geeks, How does L1 and L2 regularization prevent overfitting?, ,  
<https://www.geeksforgeeks.org/machine-learning/how-does-l1-and-l2-regularization-prevent-overfitting/>

DSNNO: Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, 2014

AUNET: Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

WISEG: Wikipedia, Image Segmentation, , [https://en.wikipedia.org/wiki/Image\\_segmentation](https://en.wikipedia.org/wiki/Image_segmentation)

DLABIS: Intisar Rizwan I Haque, J. Neubert , Deep learning approaches to biomedical image segmentation, 2020

DRLIR: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, 2015

VDCN: Karen Simonyan, Andrew Zisserman, VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, 2014

LEUC1: Al-Ghraibah, Amani , Automated detection of leukemia in blood microscopic images using image processing techniques and unique features: Cell count and area ratio,

LDAIT: ReginaMaria, Dictionar de afectiuni, simptome, investigatii si tratamente, ,  
<https://www.reginamaria.ro/utile/dictionar-de-afectiuni/leucemia>

LEUC3: Mustafa Ghaderzadeh, Mehrad Aria, Azamossadat Hosseini, Farkhondeh Asadi, Davood Bashash, Hassan Abolghasemi, A fast and efficient CNN model for B-ALL diagnosis and its subtypes classification using peripheral blood smear images,

LEUC4: Ashkan Emadi, Acute Lymphoblastic Leukemia (ALL), 2023

CLEVELEUC: Cleveland Clinic, Leukemia, , <https://my.clevelandclinic.org/health/diseases/4365-leukemia>

MTED: Willow Garage, Itseez, Morphological Transformations,

WMEDF: Wikipedia, Median Filter, , [https://en.wikipedia.org/wiki/Median\\_filter](https://en.wikipedia.org/wiki/Median_filter)

DDRAR: ART AND MACHINE LEARNINGCMU 2022 SPRINGPROJECT 2DeepDream: Reinvent and Applied ResearchShijun, LiuHaoran, LyuNingxin Xu, DeepDream: Reinvent and Applied Research, 2022

YANAT: Welch Labs, The moment we stopped understanding AI [AlexNet], 2024,  
<https://www.youtube.com/watch?v=UZDiGooFs54>

NLMBT: Maruf Adewole, The Brain Tumor Segmentation (BraTS) Challenge 2023: Glioma Segmentation in Sub-Saharan Africa Patient Population (BraTS-Africa), 2023

RMNHID: Naomi Bolton , Magnetic Properties Of Hydrogen, 2022

MRIPME: epomedicine, MRI Physics Made Easy, 2020

MRIBSB: David C Preston, Magnetic Resonance Imaging (MRI) of the Brain and Spine: Basics, 2006,  
<https://case.edu/med/neurology/NR/MRI%20Basics.htm>

T1T2WIG: medreadons, How are T1 and T2 weighted images generated ?, ,  
<https://medreasons.com/index.php/posts/radiology/t1-and-t2-weighted-mri/>

MRIMAS: MRIMASTER, T1 vs T2 MRI, , <https://mrimaster.com/t1-vs-t2-mri/>

MICCAICC: Aboian, Mariam Anazodo, Udunna Baid, Ujjwal Bakas, Spyridon Calabrese, Evan Marco Conte, Gian Kazerooni, Anahita Kirchhoff, Yannick Kofler, Florian LaBella, Dominic Li, Hongwei Linguraru, Marius George Menze, Bjoern Rudie, Jeff Vollmuth, Philipp Wiestler, Benedikt, MICCAI 2025 Lighthouse Challenge: Brain Tumor Segmentation Cluster of Challenges (BraTS), 2024

ABTSCA: Guotai Wang, Wenqi Li, Sébastien Ourselin, Tom Vercauteren, Automatic Brain Tumor Segmentation using Cascaded Anisotropic Convolutional Neural Networks,

## Anexe

Figure 1: Norbert Wiener(1), Norbert Wiener și Claude Shannon(2).....	2
Figure 2: Ștefan Odobleja.....	4
Figure 3: Circuit Inițial.....	6
Figure 4: Circuit Transformat.....	6
Figure 5: Detectarea și corectarea erorilor prin biți de paritate.....	8
Figure 6: Theseus, șoarecele electric.....	9
Figure 7: Alan Turing.....	10
Figure 8: Paradoxul lui Turing.....	11
Figure 9: Bombe.....	12
Figure 10: Aparatul Enigma.....	12
Figure 11: Neuron Pitts-McCulloch.....	13
Figure 12: Funcții de activare.....	16
Figure 13: Grafic Derivată.....	17
Figure 14: Rezultat Gradient Descent pentru $g(x)$ .....	19
Figure 15: Graficul lui $f(x,y)$ + graficele derivatelor parțiale.....	20
Figure 16: Graficul lui $f(x,y)$ .....	20
Figure 17: Grafic văzut de deasupra.....	22
Figure 18: Pașii algoritmului gradient descent.....	22
Figure 19: Arhitectura Rețelei.....	25
Figure 20: Acumularea erorilor.....	31
Figure 21: Metode de implementare Backpropagation.....	32
Figure 22: Metode de implementare Backpropagation.....	32
Figure 23: Tiparuri găsite de model.....	34
Figure 24: Activări MLP.....	34
Figure 25: Arhitectura unui transformator.....	37
Figure 26: Arhitectură AlexNet.....	40
Figure 27: Edge detection.....	42
Figure 28: Filtru 3D folosind conoluții.....	43
Figure 29: Tangentă hiperbolică.....	44
Figure 30: Arhitectura UNET.....	51
Figure 31: Pierderea fidelității imaginii prin compresie.....	52
Figure 32: Activările unei rețele U-Net.....	53
Figure 33: Diagrama claselor – Modul U-Net.....	53
Figure 34: Diagramă de activitate – Modul U-Net.....	54
Figure 35: Utilizarea modulului în mediul Google Collab.....	55
Figure 36: Arhitectură ResNet.....	56
Figure 37: Blocuri reziduale.....	57
Figure 38: Diagramă clase - ResNet.....	57
Figure 39: Diagrama Blocurilor.....	59
Figure 40: Utilizare în Jupyter Notebooks - ResNet.....	60
Figure 41: Arhitectură VGG16.....	61
Figure 42: Activările unei rețele convoluționale.....	62
Figure 43: Formarea celulelor sanguine.....	65
Figure 44: Diagrama Semnetare Leucemie - I.....	68

Figure 45: Scalarea și încărcarea imaginilor.....	69
Figure 46: Modelul implementat în cod.....	69
Figure 47: Leucemie - ImgSeg1.png.....	70
Figure 48: Leucemie - ImgSeg2.png.....	70
Figure 49: Leucemie - ImgSeg3.png.....	70
Figure 50: Leucemie – ImgSeg4.png.....	70
Figure 51: Leucemie - ImgSeg5.png.....	71
Figure 52: Efecte - Median Blur.....	73
Figure 53: Efecte Median Blur II.....	73
Figure 54: Leucemie - ImgSeg6.png.....	73
Figure 55: Leucemie - ImgSeg5.png(1).....	74
Figure 56: DeepDream - AlexNet.....	75
Figure 57: Leucemie - Prima Retea pentru clasificare.....	76
Figure 58: Implementare python a rețelei.....	76
Figure 59: Leucemie Clas1 - Matrice de confuzie.....	77
Figure 60: Leucemie Clas2.....	78
Figure 61: Lucemie Class2 - Matricea de confuzie.....	79
Figure 62: Leucemie - Măști și Feature Map-uri.....	81
Figure 63: Metodote RMN.....	83
Figure 64: Modalități oferite de BRATS.....	84
Figure 65: Măștile din setul de segmentare.....	84
Figure 66: Vizualizarea conținutului slice-urilor.....	86
Figure 67: Slice-urile ce conțin informații.....	86
Figure 68: Mască prezisă folosind primul U-Net.....	89
Figure 69: Portă de atenție.....	90
Figure 70: Arhitectura finală brats.....	91
Figure 71: Tumoră atipică.....	92
Figure 72: Pierderea informației despre structura ierarhică.....	93
Figure 73: Penalizare pentru nerespectarea ierarhiei.....	94
Figure 74: Aplicarea greșită a penalizărilor.....	95
Figure 75: Segmentarea după corectarea penalizării.....	96
Figure 76: Segmentarea după corectarea penalizării 2.....	96
Figure 77: Segmentarea după corectarea penalizării 3.....	97
Figure 78: Segmentarea după aplicarea corecturii 4.....	98
Figure 79: Ierarhie de rețele.....	99
Figure 80: Clasificare Tumori - Model 1.....	101
Figure 81: Clasificarea tumorilor - Matrice de confuzie model 1.....	102
Figure 82: Clasificarea tumorilor - Model 2.....	103
Figure 83: Clasificarea Tumorilor - ResNet50.....	104
Figure 84: Glioma P1.....	105
Figure 85: Glioma P1 – Corecție aplicată.....	105
Figure 86: Glioma P2.....	105
Figure 87: Glioma P2 - Corecție aplicată.....	105
Figure 88: Meningioma P1.....	106
Figure 89: Meningiomă P1 – Corecție aplicată.....	106
Figure 90: Meningiomă P2.....	106
Figure 91: Meningiomă P2 – Corecție aplicată.....	106

Figure 92: Notumor - P1.....	107
Figure 93: Notumor P1 - Corecție aplicată.....	107
Figure 94: Notumor P2.....	107
Figure 95: Notumor P2 - Corecție aplicată.....	107
Figure 96: Tumoră Pineală P1.....	108
Figure 97: Tumoră pineală P1 - corecție aplicată.....	108
Figure 98: Tumoră pineală P2.....	108
Figure 99: Tumoră pineală P2 - corecție aplicată.....	108
Figure 100: Covid19 - Model 1.....	110
Figure 101: Preprocesarea radiografiilor pulmonare.....	110
Figure 102: Uniformizarea histogramei în cazul radiografiilor pulmonare.....	111
Figure 103: Funcția pentru aplicarea uniformizării.....	111
Figure 104: Măsurători după uniformizare.....	112
Figure 105: Arhitectura utilizată în cazul clasificării tumorilor.....	113
Figure 106: Rezultate finale - clasificarea radiografiilor pulmonare.....	113
Figure 107: Matrice de confuzie OCT.....	115
Figure 108: Rețea reziduală.....	117
Figure 109: Rezultate - Pietre Renale.....	118