

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA [Informatică]

LUCRARE DE LICENȚĂ

**Evaluarea arhitecturilor de segmentare
a imaginilor pentru jurnalizarea dietei**

**Conducător științific
Asist. dr. Bota Florentin**

*Absolvent
Butacu Ștefan-Alexandru*

2023

ABSTRACT

In the contemporary era, maintaining a well-balanced diet through regular consumption of nutritious food has become increasingly imperative in combating the prevalent issue of obesity among individuals. Within this context, automated food recognition in images has emerged as a promising field with diverse and impactful applications, including nutritional tracking within medical cohorts.

This study comprehensively explores a deep learning model based on neural networks, which aims to classify each pixel in an image into one of the 103 available food categories. By utilizing advanced techniques in image segmentation, the proposed methodology strives to achieve exceptional performance in accordance with established evaluation metrics in the field.

Moreover, the development of a user-friendly graphical interface for the application adds an extra dimension to the overall project. This interface serves as a critical factor in promoting user engagement and adherence to dietary plans, while facilitating the consistent recording of daily eating habits. The intuitive and appealing design of the interface enhances user experience, ensuring higher retention rates and encouraging regular interaction with the system.

In summary, this bachelor thesis delves into the domain of automated food recognition, focusing on the development of a deep learning model for accurate classification of food categories in images. The study not only emphasizes the technical aspects of image segmentation but also places significant importance on the user experience to foster user engagement and facilitate adherence to healthy eating habits. Through this research endeavour, valuable insights and contributions are made towards addressing the challenges associated with nutrition tracking and promoting healthier dietary practices.

Cuprins

1	Introducere	1
1.1	Prezentarea problemei	1
1.2	Structura lucrării	2
2	Context Teoretic	3
2.1	Inteligenta artificiala	3
2.1.1	Deep Learning	5
2.1.2	Rețele neuronale convolutionale	5
2.2	Nutriție: Mănâncă pentru a trăi sau trăiește pentru a mânca	8
2.2.1	Nutrienți: fundația pentru viață	9
2.2.2	Studii in domeniul jurnalizării dietei	13
3	Soluția propusă	15
3.1	Analiza arhitecturală	16
3.1.1	UNet	16
3.1.2	ResNet	17
3.1.3	Vision Transformer	18
3.2	Setul de date	19
3.3	Funcția de loss	20
3.4	Optimizer	22
3.5	Funcții de activare	23
3.6	Metrici de evaluare	24
3.6.1	Coeficientul Sorenson-Dice	24
3.6.2	Acuratețea	25
3.7	Experimente realizate	25
3.7.1	Primul experiment: UNet	25
3.7.2	Al doilea experiment: ResNet Encoder în arhitectura UNet	28
3.7.3	Al treilea experiment: SAM Image Encoder cu Decoder propriu	30
4	Arhitectura și dezvoltarea produsului software	34
4.1	Descrierea cerințelor	34

4.1.1	Cazuri de utilizare	34
4.1.2	Arhitectura obiectuală	37
4.2	Arhitectura client-server	38
4.3	Tehnologii folosite	39
4.3.1	Integrarea Tehnologiei de Segmentare a Imaginilor	41
5	Concluzii și perspective de viitor	47
5.1	Concluzie	47
5.2	Scanarea codului de bare	48
5.3	Integrarea cu alte aplicații și dispozitive	48
Bibliografie		49

Capitolul 1

Introducere

1.1 Prezentarea problemei

Creșterea în greutate la nivel global este o problemă de sănătate publică de proporții care afectează miliarde de oameni. Din ce în ce mai mulți oameni suferă de obezitate și supraponderabilitate, cauzată de o combinație de dietă nesănătoasă, sedentarism și factori genetici. Obezitatea este asociată cu un număr mare de afecțiuni cronice, inclusiv diabet, boli de inimă, accident vascular cerebral și anumite tipuri de cancer. Aceasta poate duce la o scădere semnificativă a calității vieții și a speranței de viață. [29]

Pentru a aborda problema globală a obezității, am dezvoltat o soluție inovațioare care exploatează puterea inteligenței artificiale în monitorizarea și gestionarea dietei. Aceasta este un pas semnificativ în direcția utilizării tehnologiei pentru a îmbunătăți sănătatea și calitatea vieții oamenilor. Aplicația noastră utilizează algoritmi de învățare automată pentru a analiza fotografiile alimentelor consumate de utilizator, identifică tipul de alimente și estimează valoarea calorică și compoziția nutrițională. Prin acestea, oferă utilizatorilor un mod convenabil și precis de a ține evidență consumului lor de alimente, de a înțelege mai bine obiceiurile alimentare și de a face alegeri alimentare mai sănătoase.

Astfel, se individualizează două direcții distincte de dezvoltare: una ce privește identificarea alimentelor și alta privind integrarea algoritmului într-o aplicație pentru dispozitive mobile. Propunem o examinare comparativă a trei modele de arhitecturi neuronale, urmărind două abordări de proiectare, una fundamentată pe convoluții și una fundamentată pe mecanismul de atenție, cu scopul final de a selecta modelul cu cea mai înaltă performanță și de a-l încorpora într-un produs software.

1.2 Structura lucrării

Această lucrare este organizată după cum urmează: în capitolul 2 descriem contextul teoretic al lucrării, incluzând repere în domeniul inteligenței artificiale, precum și informații dietetice relevante. În capitolul 3 discutăm despre arhitecturile existente, care sunt cunoscute pentru rezolvarea problemelor de segmentare a imaginilor, cum le-am integrat și cum le-am măsurat capacitatele. În capitolul următor, 4, intrăm în detaliu asupra componentelor tehnice ale acestui proiect, inclusiv implementarea, precizând ce tehnologii au fost utilizate pentru dezvoltarea acestuia și care sunt caracteristicile sale principale. În final, utilizând capitolul 5 oferim potențiale îmbunătățiri sau modalități de extindere a sistemului.

Capitolul 2

Context Teoretic

2.1 Inteligența artificială

În ceea ce privește relevanța inteligenței noastre, oamenii se referă la ei însiși ca Homo Sapiens, ceea ce înseamnă „om intelligent” sau „om învățat”[28]. Timp de mii de ani, am încercat să înțelegem de ce gândim, cum gândim și ce să gândim, rezultând o legătură între intelligentă și existență, așa cum René Descartes a afirmat principiul de bază al filozofiei sale: „Cogito, ergo sum (Gândesc, deci exist).”[3] Întrebarea de cum poate un creier să perceapă, să înțeleagă, să prevadă și să manipuleze o lume mult mai mare și mai complicată decât el însuși este una fascinantă. Domeniul inteligenței artificiale (IA) merge chiar mai departe: încearcă nu numai să înțeleagă, ci și să construiască entități inteligente.[27] Subiectul inteligenței artificiale de astăzi cuprinde o gamă largă de subdomenii, de la subiecte generale, cum ar fi învățarea și precizarea viitorului, până la cele specifice, cum ar fi mașinile autonome, asistenții vocali, sistemele de recomandare, asistența medicală și multe altele. Orice operație intelectuală poate beneficia de IA; acest lucru face ca acest domeniu să fie cu adevărat global.

Inteligența artificială este un domeniu de studiu care vizează crearea de sisteme informaticе capabile să imite sau să depășească nivelul de inteligență umană, atât în ceea ce privește gândirea, cât și în ceea ce privește comportamentul, prin utilizarea de modele și algoritmi de calcul. Scopul este de a dezvolta agenți inteligenți care pot lua decizii și acțiuni într-un mod rațional și coerent, fiind capabili să efectueze sarcini care necesită inteligență umană, precum perceptia, raționarea, acțiunea și învățarea. IA implică utilizarea de algoritmi și tehnici avansate, precum învățarea automată, procesarea limbajului natural, viziunea artificială și robotică, având ca scop final crearea de sisteme informaticе capabile să efectueze sarcini și să rezolve probleme la fel de eficient ca oamenii, astfel încât să automatizeze activități care necesită intervenția umană sau să completeze și să îmbunătățească capacitatea umană

de a efectua sarcini complexe.

Aceste definiții expun câteva dintre domeniile de dezvoltare a tehnologiei în viitor, bazate pe două aspecte, unul legat de actul de înțelegere, iar celălalt de actul de comportare. Capabilitățile generale ale IA sunt evaluate prin accentuarea succesului în funcție de cât de apropriate sunt de performanța umană sau prin evaluarea succesului față de un standard perfect de performanță, cunoscut sub numele de raționalitate. Un sistem expune raționalitate atunci când efectuează "acțiunea corectă" pe baza informațiilor pe care le detine.

Pe măsură ce sistemele evoluează și devin mai sofisticate, cercetătorii și filosofii se confruntă cu întrebarea dacă aceste sisteme cu adevărat gândesc sau doar simulează aparența gândirii prin acțiunile lor. Această dezbatere are implicări nu numai pentru înțelegerea noastră a IA, ci și pentru întrebările mai largi legate de natura inteligenței, conștiinței și limitele capacitaților mașinilor.

Dintr-o perspectivă evolutivă, gândirea și comportamentul sunt intim legate, deoarece au co-evoluat în organismele biologice pentru a permite adaptarea și supraviețuirea într-un mediu complex și în continuă schimbare. În contextul IA, relația dintre gândire și comportament este, de asemenea, esențială. Sistemele inteligente sunt proiectate pentru a procesa informații, a lua decizii și a efectua acțiuni pe baza acestora, adesea în timp real și cu resurse limitate. Capacitatea de a gândi și de a acționa eficient este, prin urmare, un aspect critic al dezvoltării IA.

Un punct de vedere este că sistemele IA se comportă pur și simplu, urmând un set de reguli predefinite sau algoritmi pentru a efectua sarcini specifice. Se susține că adevărată gândire necesită conștiință, autopercepție și experiențe subiective, pe care sistemele artificiale nu le dețin în prezent. Conform acestei perspective, sistemele sunt instrumente complexe care pot simula aparența gândirii, dar nu posedă în mod autentic capacitatea de a gândi aşa cum o fac oamenii. De exemplu, testul Turing - sugerat de Alan Turing în 1950 - a fost creat pentru a stabili o definiție empirică suficientă a inteligenței, însemnând că un interogator uman nu poate identifica dacă răspunsurile scrise sunt de la o persoană sau de la o mașină după ce a pus câteva întrebări scrise.

Gândirea și comportamentul în IA se extind și la întrebarea cum ar trebui proiectate și dezvoltate aceste sisteme. Unii cercetători se concentrează pe crearea de sisteme inteligente care imită procesele cognitive umane, având ca scop replicarea modului în care oamenii gândesc și raționează. Această abordare, cunoscută sub numele de "Inteligentă artificială cognitivă", își propune să construiască sisteme care sunt mai asemănătoare cu modul de gândire și comportamentul uman. Alții susțin o abordare pragmatică, orientată spre obiective, concentrându-se pe dezvoltarea de sisteme care pot efectua sarcini și rezolva probleme într-un mod eficient, fără a replica neapărat procesele cognitive umane.

Această discuție continuă nu numai informează proiectarea și dezvoltarea sistemelor, ci și provoacă înțelegerea noastră despre inteligență, conștiință și natura gândirii umane. Explorarea diferenței dintre gândire și comportament, uman și rational, poate conduce la o înțelegere mai profundă atât a inteligenței artificiale, cât și a celei umane, evidențiind relația complexă dintre gândire, comportament și dezvoltarea sistemelor inteligente.[27]

2.1.1 Deep Learning

Deep learning este o ramură a inteligenței artificiale care implică antrenarea rețelelor neuronale artificiale pentru a învăța și a face predicții din date complexe. Este o formă de inteligență artificială care folosește straturi de noduri interconectate (neuroni) pentru a modela și analiza datele.

Modelele de învățare pot fi folosite pentru o gamă largă de aplicații, inclusiv pentru viziunea artificială, procesarea limbajului natural, recunoașterea vocală și multe altele. Aceste modele sunt proiectate pentru a învăța automat reprezentări ierarhice ale datelor, permitându-le să extragă caracteristici și tipare la niveluri tot mai complexe.

Eficiența algoritmilor de învățare se datorează capacitatea lor de a identifica automat caracteristici și modele importante în date, fără a fi programate explicit. Această abordare elimină necesitatea unui operator uman de a furniza explicit toate informațiile necesare, deoarece modelul învăță prin expunerea la date. Acest lucru îi face potrivit pentru a rezolva probleme complexe în care abordările traditionale de învățare automată pot fi insuficiente.

Unele dintre tehniciile cheie utilizate în Deep Learning includ rețele neuronale convolutionale (CNN), rețele neuronale recurente (RNN) și rețele adversariale generative (GAN). Aceste tehnici au fost aplicate la o gamă largă de probleme reale, de la recunoașterea imaginilor și a vocilor până la descoperirea de medicamente și mașini autonome.

Deep learning a devenit o unealtă puternică pentru rezolvarea problemelor complexe în multe domenii diferite, iar impactul său este susceptibil să continue să crească pe măsură ce domeniul continuă să avanseze.[6]

2.1.2 Rețele neuronale convolutionale

Rețelele neuronale sunt modele matematice inspirate de modul în care funcționează creierul uman, prin utilizarea unor algoritmi de învățare automată. Acestea sunt alcătuite din mai multe unități numite neuroni artificiali, care primesc informații de intrare, procesează aceste informații și generează o ieșire. Un neuron artificial

este o funcție matematică care poate combina valorile de intrare ponderate cu niște parametri și produce o valoare de ieșire.

Convoluțiile sunt operații matematice care sunt utilizate în procesarea datelor de tip imagine, fiind folosite în special pentru extragerea de caracteristici importante din aceste date. Aceste operații sunt realizate prin combinarea unei mici matrice de filtrare cu datele de intrare, calculând astfel produsul scalar al elementelor corespunzătoare dintre matricea de filtrare și o porțiune a datelor de intrare. Rezultatul este o nouă matrice numită hartă de caracteristici (*feature map*), care conține informații despre prezența sau absența anumitor caracteristici în datele de intrare.

În rețelele neuronale, convoluțiile sunt utilizate în principal în cadrul arhitecturilor numite rețele convolutionale sau CNN (*Convolutional Neural Networks*). Aceste rețele sunt utilizate în special în probleme de clasificare și recunoaștere de obiecte în imagini, fiind alcătuite din mai multe straturi de convoluții și alte tipuri de straturi de prelucrare a datelor. Utilizarea convoluțiilor în aceste rețele permite extragerea de caracteristici relevante din imagini, ceea ce duce la o mai bună performanță în rezolvarea problemelor de prelucrare a imaginilor.

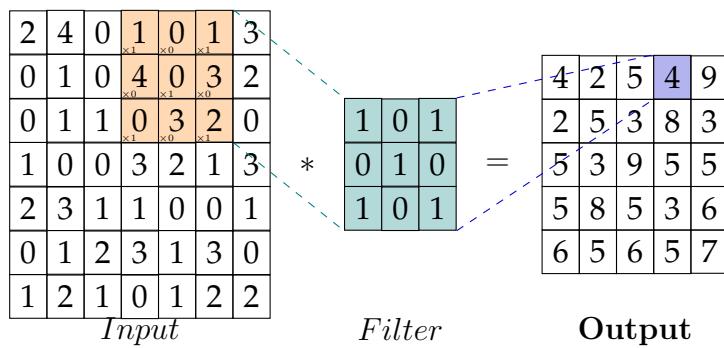


Figura 2.1: Aplicarea unei convoluții

Neuronul: Inspirația biologică a calculului

Neuronii reprezintă unitățile fundamentale de procesare a informațiilor în creier și sunt o componentă esențială în rețelele neuronale artificiale. Există o serie de lucrări științifice care au adus contribuții semnificative în înțelegerea funcționării neuronilor și aplicarea acestora în rețele neuronale artificiale. În cele ce urmează, vom prezenta câteva din cele mai importante lucrări științifice referitoare la neuron.

Lucrarea lui Hodgkin și Huxley din 1952 a avut un impact major în înțelegerea funcționării neuronilor. Aceștia au propus un model matematic pentru a descrie generarea și propagarea potențialelor de acțiune în neuroni. Modelul lor a fost bazat pe studiul experimentelor electrice efectuate pe axonul caracătăiei, iar rezultatele lor au fost aplicate pentru a înțelege generarea și propagarea potențialelor de acțiune în neuroni, precum și alte aspecte ale funcționării neuronilor. Această lucrare a fost

recunoscută ca fiind una dintre cele mai importante din domeniul neuroștiințelor, fiind considerată un punct de referință în dezvoltarea modelelor neuronale. [12]

O altă lucrare importantă în înțelegerea funcționării neuronilor este cea a lui McCulloch și Pitts din 1943, care a propus un model matematic simplu pentru a descrie activitatea neuronilor. Modelul lor a fost bazat pe ideea că neuronii sunt unități de procesare binare, capabile să primească semnale de intrare și să producă un semnal de ieșire în funcție de un anumit prag de activare. Acest model a fost apoi extins și rafinat de alți cercetători, ducând la dezvoltarea rețelelor neuronale artificiale, care sunt folosite astăzi într-o varietate de aplicații. [20]

Lucrarea lui Hebb din 1949 a fost una dintre primele care a abordat conceptul de plasticitate sinaptică și de învățare în conexiune cu activitatea neuronilor. Hebb a propus că atunci când doi neuroni sunt activați simultan, conexiunea dintre ei se întărește, iar această conexiune puternică poate duce la învățarea unor tipare specifice. Acest concept a fost ulterior dezvoltat în teoria învățării asociative și a jucat un rol important în dezvoltarea rețelelor neuronale artificiale cu învățare automată. [11]

Aceste lucrări științifice au adus contribuții semnificative în înțelegerea funcționării neuronilor și aplicarea lor în dezvoltarea rețelelor neuronale artificiale. Modelul matematic propus de Hodgkin și Huxley a fost un punct de referință în dezvoltarea modelelor neuronale, iar modelul simplu propus de McCulloch și Pitts a dus la dezvoltarea rețelelor neuronale artificiale. Conceptul de plasticitate sinaptică și de învățare propus de Hebb a fost crucial în dezvoltarea teoriei învățării asociative și învățării automate în rețelele neuronale artificiale. Aceste lucrări sunt considerate fundamentale în domeniul neuroștiințelor și reprezintă bazele teoretice ale rețelelor neuronale artificiale moderne.

Overfitting

Un obstacol ce poate apărea în urma procesului de instruire este supramodelarea sau overfitting-ul. Supramodelarea survine atunci când algoritmul învăță datele de antrenament "pe de rost", în loc să identifice o generalizare care să se potrivească întregii colecții de date de antrenament. Astfel, rețeaua nu va fi capabilă să furnizeze predicții precise pentru datele noi, în ciuda faptului că oferă rezultate excelente pe datele de antrenament. Problema supramodelării este una subtilă și nu există o metodă care să o rezolve în mod constant. Cu toate acestea, validarea încrucișată poate fi utilizată pentru a estima un punct potrivit de oprire în cadrul căutării, în vederea minimizării acestui risc. Uneori, reducerea ratei de învățare la o valoare mai mică poate ajuta, de asemenea. O altă posibilă soluție la problema supramodelării este adăugarea de straturi de tip 'Dropout'. Aceasta determină rețeaua neuronală să

ignore anumite unități alese aleatoriu în timpul unor rulări de propagare înapoi sau propagare înainte.

Antrenament

Algorithm 1 Exemplu de antrenare a unei rețele neuronale

```
Set de date de antrenare  $D$ 
rată de învățare  $\alpha$ 
număr de epoci  $N$ 
Inițializare greutăți rețea neuronală
for  $epoch = 1$  până la  $N$  do
    Amestecare date de antrenare
    for fiecare mini-batch  $B$  din  $D$  do
         $\hat{y} = \text{model}(B)$ 
        loss = loss_function( $\hat{y}, y$ )
        Actualizare greutăți și bias folosind gradient descendente cu rata de învățare
         $\alpha$  pe baza loss-ului
    end for
end for
```

2.2 Nutriție: Mănâncă pentru a trăi sau trăiește pentru a mâncă.

Nutriția este știința nutrientilor din alimente și a acțiunilor lor în organism. O definiție mai largă acoperă cercetarea modului în care oamenii se comportă în jurul alimentelor și a mâncatului. Viața noastră a fost întotdeauna semnificativ afectată de nutriție. Alegem alimente de mai multe ori pe zi care afectează bunăstarea corpului nostru. Efectele alegerilor noastre alimentare zilnice pot fi doar minore, dar pe termen lung, ele au un impact semnificativ asupra sănătății noastre. Practicile alimentare bune ajută procesul nostru de îmbătrânire în opozitie cu alegerile alimentare neglijente, care pot dezvolta boli cronice. Desigur, unii oameni se vor îmbolnăvi devreme indiferent de ceea ce consumă, în timp ce alții vor trăi o viață lungă în ciuda deciziilor proaste [30]. Conform Organizației Națiunilor Unite, speranța de viață globală în 2023 este în medie de 73,4 ani, fiind mai mare cu aproape 20 de ani față de anul 1950 [19]. Am putea fi de acord că speranța de viață urmează o tendință ascendentă, iar alegerile pe care le facem cu privire la dieta noastră pot influența semnificativ calitatea anilor noștri mai târzii. În ultimii 25 de ani de viață, un individ poate fi nevoie să petreacă timpul în spital și să depindă de numeroase medicamente zilnice, în timp ce o altă persoană care a fost mai atentă la sănătatea ei poate petrece anii de aur alături de nepoți, rămânând mobilă și menținând agilitatea

mentală. Cu toate că ambele persoane pot ajunge în cele din urmă să trăiască 73 de ani, se crede în mod larg că cea din urmă a avut o viață mai împlinită și mai fericită.

Deși majoritatea oamenilor sunt conștienți că obiceiurile alimentare au un impact asupra sănătății, ei fac adesea alegeri alimentare din diferite motive. În cele din urmă, alimentele aduc plăcere, respectă tradițiile, facilitează socializarea și, de asemenea, oferă nutrientii necesari. Dificultatea constă în a combina mesele preferate și activitățile plăcute cu o dietă echilibrată și sănătoasă. O dietă nu este neapărat un plan alimentar restrictiv orientat spre pierderea în greutate, ci include varietatea de alimente și băuturi consumate de un individ. În funcție de preferințele și alegerile personale, o dietă poate lua diferite forme, cum ar fi o dietă vegetariană, o dietă de slăbire sau orice altă dietă specializată. În cele din urmă, tipul specific de dietă pe care o persoană îl urmează este determinat de selecția ei de alimente și băuturi consumate.

În ceea ce privește alimentația, oamenii iau decizii privind ce să mănânce, când să mănânce, cât să mănânce și chiar dacă să mănânce pe baza unei interacțiuni complexe a variabilelor genetice, comportamentale și sociale, nu doar pe baza cunoștințelor despre importanța nutriției pentru sănătate. O varietate de selecții dietetice pot ajuta la menținerea unei sănătăți excelente, iar înțelegerea nutriției umane permite luarea deciziilor mai bine informate în mod regulat. Folosim alimentele pentru a proiecta o imagine dorită, pentru a forma relații, pentru a demonstra creativitate și pentru a dezvălui emoțiile noastre. Trăim stresul prin consumul de alimente sau prin abținerea de la mâncare; ne răsfățăm cu alimente drept recompensă pentru o realizare remarcabilă; sau, în situații extreme, ne pedepsim prin refuzul de a mâncă. [13]

Emoțiile influențează preferințele alimentare și obiceiurile alimentare. Unele persoane nu pot mâncă când sunt emoțional nefericite. Alții pot mâncă reacționând la diverse stimulente emotionale, cum ar fi plăcerea, furia sau îngrijorarea. O persoană deprimată poate alege să mănânce în loc să sune un prieten. O persoană care s-a întors acasă după o seară evenimentată poate să se relaxeze cu o gustare nocturnă. Unele persoane pot simți consolare emoțională datorită faptului că hrana poate influența chimia creierului. [30]

2.2.1 Nutrienți: fundația pentru viață

Din perspectivă biologică, oamenii mănâncă pentru a primi nutrienți. Alimentele, precum corpul nostru, sunt compuse dintr-un amestec de substanțe chimice, unele dintre acestea fiind esențiale pentru funcționarea normală a organismului. Aceste substanțe esențiale sunt numite nutrienți. Nutrienții sunt necesari pentru creștere și dezvoltare, pentru întreținerea celulelor și țesuturilor, pentru furnizarea de energie

pentru efectuarea sarcinilor fizice și metabolice și pentru controlul multor procese din corpul uman care au loc în fiecare secundă a fiecărei zile. Mai mult, este imperativ ca hrana să furnizeze acești nutrienți, deoarece corpul nu poate produce aceste substanțe esențiale sau nu le poate produce în cantități suficiente. Există trei clase de macro-nutrienți: proteine, lipide și carbohidrați, care constituie baza oricărei diete pentru a crește, dezvoltă și menține corpul uman. Piramida alimentară servește ca ghid util pentru ilustrarea proporțiilor acestor nutrienți care ar trebui consumați zilnic. Prin urmarea recomandărilor, se poate asigura un aport variat și echilibrat de toți nutrienții esențiali pentru sănătate și bunăstare, așa cum este ilustrat în Figura 2.2.[13]

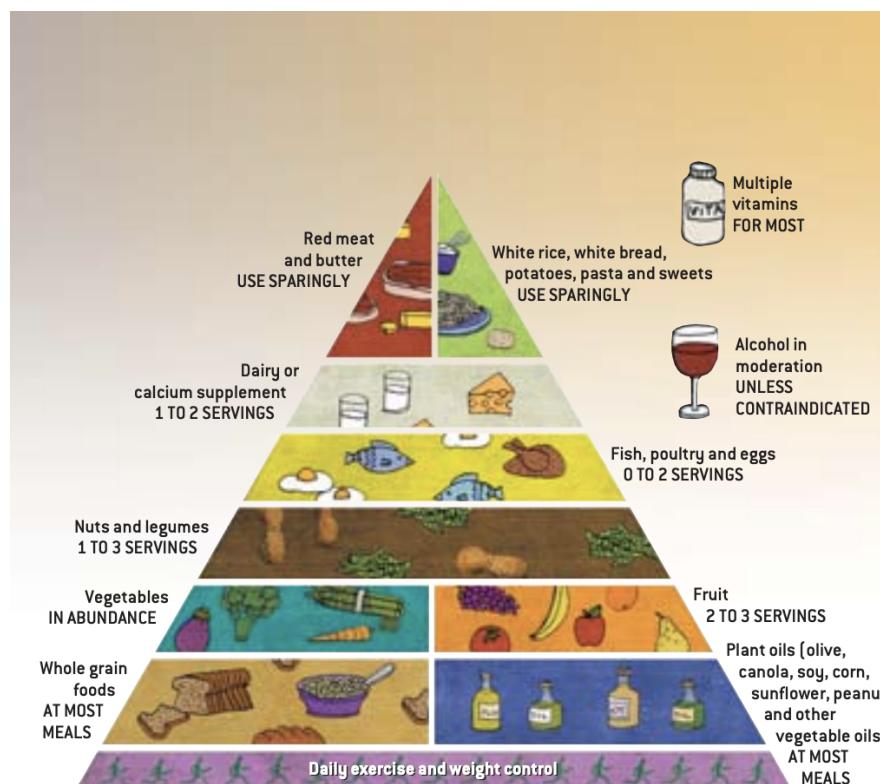


Figura 2.2: Piramida alimentelor [31]

Proteine

Proteinele sunt molecule mari și complexe care au un rol crucial în diferite funcții din organismul uman. Din punct de vedere chimic, proteinele sunt alcătuite din aminoacizi, care sunt blocurile de construcție ale acestora. Aminoacizii au o structură de bază care include un atom de carbon central (C) cu un atom de hidrogen (H), un grup amino (NH_2), un grup acid ($COOH$) și pe al patrulea carbon este legat un atom sau un grup de atomi distinct care diferențiază aminoacizii. Proteinele sunt implicate atunci când organismul se dezvoltă, se vindecă sau înlocuiesc țesutul.

Rolul lor poate varia de la facilitarea sau supravegherea unor procese până la integrarea într-o structură.

Ca material structural, proteina este componenta majoră a tuturor celulelor corpului, cum ar fi mușchii, săngele, pielea și multe alte celule și țesuturi. Exemple includ colagenul, care se găsește în țesuturile conjunctive, actina și miozina, care sunt prezente în mușchi.

Ca enzime, proteinele pot acționa ca catalizatori pentru reacțiile biochimice, accelerând rata de reacție și permitând proceselor metabolice esențiale să aibă loc.

Proteinele pot funcționa ca hormoni, care sunt mesageri chimici care reglează diferite procese fiziologice. De exemplu, insulina este un hormon proteic care reglează nivelurile de zahăr din sânge.

Unele proteine, cum ar fi hemoglobina, sunt responsabile pentru transportul moleculelor, cum ar fi oxigenul și nutrientii, în întregul corp.

Ca anticorpi, unele proteine ajută sistemul imunitar să identifice și să neutralizeze substanțele străine, cum ar fi bacteriile și virusurile.

Recomandările privind consumul de proteine variază în funcție de vârstă, sex și activitatea fizică. Recomandarea generală este ca 20-35% din totalul caloriilor zilnice să provină din proteine. Această rată poate fi mai mare pentru sportivi, femeile însărcinate și persoanele care se recuperează după boală sau intervenții chirurgicale. Pentru a asigura un aport adecvat de toți aminoacizii esențiali, este critic să se consume o varietate de surse de proteine, inclusiv cele de origine animală și vegetală. [30]

Lipide

Lipidele reprezintă o gamă largă de molecule organice solubile în solventi organici și insolubile în apă. Principalele clase de lipide sunt trigliceridele, fosfolipidele și sterolii. Din punct de vedere chimic, lipidele sunt compuse din acizi grași, care sunt lanțuri de atomi de carbon cu o grupă acid organic ($COOH$) la un capăt și metil (CH_3) la celălalt capăt. [13]

Lipidele joacă numeroase roluri critice în corpul uman. Trigliceridele, tipul predominant de lipide, sunt utilizate ca stocare de energie pe termen lung în țesutul adipos. Ele oferă o sursă concentrată de energie, cu fiecare gram de grăsime producând aproximativ 9 kilocalorii, de două ori mai mult decât cele provenite din carbohidrați sau proteine.

În structura membranei celulare, fosfolipidele și colesterolul sunt componente esențiale, oferind permeabilitate, fluiditate și stabilitate structurală. Aceste lipide se adună pentru a crea un dublu strat care servește drept barieră între interiorul celulei și mediul înconjurător și este esențial pentru mișcarea substanțelor înspre și dinspre

celulă.

Lipidele din țesutul adipos acționează ca protecție pentru a ajuta la menținerea corpului la o temperatură constantă și protejează organele interne de deteriorări fizice. Stratul de grăsime subcutanată acționează ca un izolator termic, în timp ce depozitele de grăsime din jurul organelor servesc ca o pernă pentru absorția șocurilor externe.

În producerea hormonilor, colesterolul servește ca precursor pentru sinteza hormonilor steroizi, cum ar fi cortizolul, aldosteronul și hormonii sexuali, cum ar fi testosteronul și estrogenul. Acești hormoni reglează o gamă largă de procese fizio- logice, inclusiv metabolismul, funcția imună și reproducerea.

Ingestia recomandată pentru lipide depinde de factori precum vârstă, sexul și necesitățile calorice. Recomandarea generală este ca 20-35% din totalul calorilor zilnice să provină din grăsimi. Cu toate acestea, tipul de grăsimi consumate este, de asemenea, important. Se recomandă consumul de grăsimi nesaturate și limitarea grăsimilor saturate și trans. [30]

Carbohidrați

Carbohidrați sunt compuși alcătuși din atomi de carbon (*C*), oxigen (*O*) și hidrogen (*H*) sau CH_2O . Cele două tipuri principale de carbohidrați sunt carbohidrații simpli (monozaharide și dizaharide) și carbohidrații complecși (oligozaharide și polizaharide)[9].

Ei servesc drept sursă esențială de energie pentru organismul uman. Glucoza, obținută din carbohidrați, este sursa preferată de energie pentru creier, mușchi și alte organe vitale. Carbohidrații mai joacă un rol în menținerea nivelului de zahăr din sânge, susținerea sănătății gastrointestinale și furnizarea de componente structurale pentru celule și țesuturi. Doza recomandată de carbohidrați pentru adulți este de aproximativ 45-65% din necesarul caloric zilnic total. Această proporție poate varia în funcție de factori individuali, precum vârstă, sexul, nivelul de activitate fizică și starea generală de sănătate. Este esențial să se aleagă carbohidrați de înaltă calitate, precum cerealele integrale, fructele, legumele și leguminoasele, care sunt bogate în nutrienți și fibre alimentare.[30]

Rata metabolică bazală

Rata Metabolică Bazală (RMB) reprezintă cantitatea de energie pe care organismul o utilizează în stare de repaus pentru a susține funcțiile vitale ale corpului, cum ar fi respirația, circulația săngelui și funcționarea organelor interne. Este importantă în determinarea necesarului caloric zilnic al unei persoane.

Pentru a calcula RMB, există mai multe ecuații, iar una dintre cele mai utilizate și

precise este ecuația Mifflin-St Jeor. Această ecuație a fost dezvoltată în 1990 de către Mifflin și St Jeor pe baza unui studiu extins pe un eșantion reprezentativ de indivizi sănătoși.

Ecuatia Mifflin-St Jeor[21] este adaptată în funcție de sex, greutate, înălțime și vârstă. Pentru bărbați, ecuația este:

$$RMB = (10 \times \text{greutate în kg}) + (6.25 \times \text{înălțime în cm}) - (5 \times \text{vârstă în ani}) + 5 \quad (2.1)$$

Pentru femei, ecuația devine:

$$RMB = (10 \times \text{greutate în kg}) + (6.25 \times \text{înălțime în cm}) - (5 \times \text{vârstă în ani}) - 161 \quad (2.2)$$

Necesarul caloric se calculează astfel:

$$\text{Necesarul caloric} = RMB \times \text{factor de activitate} \quad (2.3)$$

Unde factorul de activitate poate lua următoarele valori:

- 1.0 - Rată metabolică de bază
- 1.2 - Sedentar sau puține alte activități
- 1.375 - Execuții 1-3 zile pe săptămână
- 1.55 - Execuții 5-6 zile pe săptămână
- 1.725 - Execuții intense zi de zi
- 1.99 - Execuții foarte obositore și îndelungate

Calcularea RMB poate fi utilă în stabilirea unui plan alimentar personalizat și în gestionarea greutății corporale. Este important de menționat că RMB poate varia de la individ la individ în funcție de factori precum masa musculară, compoziția corporală și nivelul de activitate fizică.

2.2.2 Studii în domeniul jurnalizării dietei

The role of self-monitoring in the maintenance of weight loss success

Studiul a fost realizat pe un eșantion de 167 de femei obeze, cu o valoare medie a Indicelui de Masă Corporală de $37.0 \pm 5.1 \text{ kg/m}^2$ și o vârstă medie de 59.9 ± 6.2 ani. Studiul s-a desfășurat în două faze, fiecare durând 6, respectiv 12 luni.

Rezultatele sugerează că monitorizarea reprezintă o componentă vitală în succesul inițial al pierderii în greutate. Acest proces de auto-monitorizare constă adesea în urmărirea detaliată a consumului de alimente și băuturi. S-a constatat că

participantii înscrisi în programele comportamentale tind să piardă între 8 și 10% din greutatea corporală inițială.

Acstea constatări sunt văzute ca favorabile, având la bază dovezile care indică faptul că o pierdere de greutate de 5% sau mai mult poate conduce la îmbunătățiri semnificative în sănătate. Printre acestea se numără reducerea nivelului de trigliceride, a glucozei din sânge și a tensiunii arteriale, îmbunătățirea nivelului de lipide din sânge, precum și scăderea riscului de a dezvolta diabet de tip 2.[17]

A Focused Review of Smartphone Diet-Tracking Apps: Usability, Functionality, Coherence With Behavior Change Theory, and Comparative Validity of Nutrient Intake and Energy Estimates

Studiul este o analiză ce examinează gradul de utilizabilitate al aplicațiilor de urmărire a dietei, măsura în care funcționalitățile acestora corespund cu metode de schimbare a comportamentului alimentar și de a evalua variațiile dintre aplicații în ceea ce privește consistența nutrientilor.

Aplicațiile de jurnalizare a dietei pot contribui la pierderea în greutate a individului, gestionarea afecțiunilor cronice și înțelegerea modelelor dietetice. În anul 2017, un total de 325,000 de aplicații de sănătate (mHealth) erau disponibile în principalele magazine de aplicații. Pentru acest studiu, au fost considerate cele mai populare 7 aplicații din categoria de sănătate precum: *FatSecret*, *Lifesum*, *MyPlate*, *Argus*, *Lose it!*, *MyFitnessPal*, *MyDietCoach*. Disponibilitatea oamenilor de a utiliza aplicațiile mobile pentru sănătate a fost ridicată printre participanții la studiu. Utilizabilitatea cuprinde multiple interacțiuni ale utilizatorului cu o aplicație, care includ ușurința de utilizare, complexitatea, necesitatea de suport, precum și disponibilitatea de a continua accesarea aplicației.

De remarcat este faptul că funcționalități precum scanarea codurilor de bare, introducerea fotografiilor, liste cu alimente frecvent introduse și auto-completarea cresc considerabil ușurința cu care pot fi introduse elementele.

În concluzie, aproape toate aplicațiile de urmărire a dietei examineate au obținut scoruri bune în ceea ce privește utilizabilitatea, au utilizat o varietate de metode de schimbare a comportamentului și au persistat corect caloriile, permitându-le să joace un rol potențial în monitorizarea comportamentului alimentar. [5]

Capitolul 3

Soluția propusă

În această lucrare, ne propunem să comparăm performanța a trei modele pentru sarcinile de segmentare a imaginilor. Primul model este o implementare UNet dezvoltată de la zero, bazată pe arhitectura originală. Al doilea model urmează, de asemenea, arhitectura UNet, dar introduce o modificare cheie în procesul de codare. În locul straturilor tradiționale de downsampling, acest model utilizează un model ResNet pre-antrenat pentru a captura caracteristicile relevante ale imaginii. Pe lângă aceste două abordări, am explorat și utilizarea unui al treilea model bazat pe Vision Transformer, mai precis un Masked Auto Encoder. O modificare esențială a acestui model constă în ascunderea unor zone ale imaginii în timpul procesului de propagare prin rețea. Această abordare ne permite să evidențiem importanța anumitor zone și să obținem segmentări precise. Pentru a evalua performanța modelelor, am experimentat și utilizarea a două funcții de loss. În loc să folosim o singură funcție de loss, am combinat *Cross Entropy Loss* cu *Intersection over Union Loss* printr-o combinație liniară. Această abordare ne permite să obținem un echilibru între acuratețea segmentării și consistența cu obiectele reale din imagine. Prin aceste experimente și comparații, ne propunem să identificăm cel mai eficient și precis model pentru sarcinile de segmentare a alimentelor din imagini. Rezultatele obținute vor contribui la dezvoltarea unei soluții software inovatoare pentru jurnalizarea dietei cu ajutorul inteligenței artificiale și a modelelor de învățare automată.

Această soluție va oferi utilizatorilor un instrument eficient și ușor de utilizat pentru monitorizarea și înregistrarea alimentelor consumate într-un mod precis și convenabil. Prin intermediul acestei soluții, utilizatorii vor putea ține evidența obiecturilor alimentare pentru a-și atinge obiectivele legate de sănătate și nutriție.

3.1 Analiza arhitecturală

3.1.1 UNet

Arhitectura UNet este o rețea neurală convoluțională care a fost introdusă în anul 2015 pentru sarcini de segmentare a imaginilor. Principala particularitate a arhitecturii UNet este capacitatea sa de a gestiona eficient sarcini care necesită localizarea precisă a obiectelor dintr-o imagine.

Arhitectura UNet este o rețea neuronală complet convoluțională compusă din două părți principale: o cale de contractare și o cale de expansiune. Calea de contractare este o serie de straturi convoluționale și de max-pooling utilizate pentru a reduce rezoluția spațială a imaginii de intrare și pentru a crește numărul de straturi de caracteristici.

Pe de altă parte, calea de expansiune este o serie de straturi convoluționale transpuse și de up-sampling utilizate pentru a crește rezoluția spațială a caracteristicilor și pentru a produce o segmentarea imaginii.

Ideea principală din spatele UNet este de a utiliza straturi din calea de contractare ca ghid pentru a ajuta calea de expansiune să localizeze precis obiectele din imagine. Acest lucru se realizează prin concatenarea caracteristicilor extrase în urma contractării imaginii cu caracteristicile corespunzătoare din calea de expansiune.

Arhitectura UNet este utilizată pe scară largă în imagistica medicală, unde localizarea precisă a obiectelor este importantă. De asemenea, este utilizată în alte aplicații, cum ar fi imagistica prin satelit și mașinile autonome. [25]

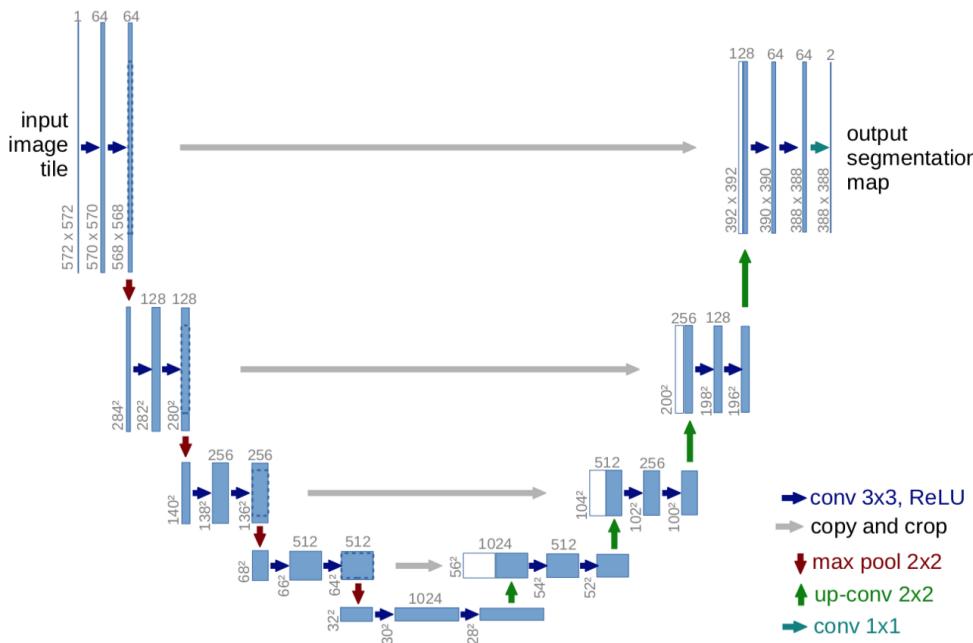


Figura 3.1: Arhitectura UNet [25]

3.1.2 ResNet

ResNet (*Residual Network*, în traducere "Rețea reziduală") este o arhitectură de rețea neuronală convezională introdusă de către Microsoft Research în 2015. Principala specialitate a arhitecturii ResNet este abilitatea de a învăța arhitecturi foarte pro-funde fără a suferi de problema gradientului care dispare.

Problema gradientului care dispare apare atunci se antrenează rețele neuronale foarte adânci, unde gradientul parametrilor devine foarte mic, ceea ce face dificilă învățarea modelului. Arhitectura ResNet abordează această problemă prin introducerea unui nou bloc de construcție numit bloc rezidual.

Un bloc rezidual conține două sau mai multe straturi convezionale, unde ieșirea primului strat convezional este adăugată la ieșirea celui de-al doilea strat convezional, creând o conexiune reziduală. Această conexiune permite rețelei să învețe o mapare de identitate a intrării către ieșire, ceea ce ajută la ameliorarea problemei gradientului care dispare.

Acest lucru permite arhitecturii ResNet să învețe arhitecturi foarte complexe cu sute sau mii de straturi. În plus, arhitectura utilizează o tehnică numită normalizare în lot, care ajută la stabilizarea procesului de antrenare și îmbunătățește performanța generalizării.

Arhitectura ResNet a obținut performanțe majore pe mai multe seturi de date de clasificare a imaginilor și este considerată o descoperire cheie în dezvoltarea rețelelor neuronale convezionale. ResNet este utilizat pe scară largă în multe sarcini de viziune artificială, cum ar fi clasificarea imaginilor, detectarea obiectelor și segmentarea semantică. [10]

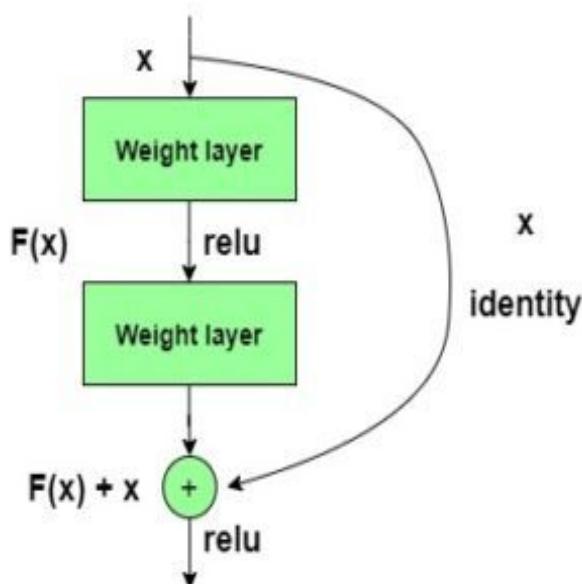


Figura 3.2: Rețea reziduală

3.1.3 Vision Transformer

Vision Transformer este o arhitectură de rețea neurală introdusă în 2020 de cercetătorii de la Google Brain. Această arhitectură a fost dezvoltată pentru a îmbunătăți performanța modelelor de viziune artificială pe sarcini complexe, precum recunoașterea imaginilor și segmentarea semantică.

Arhitectura Vision Transformer se bazează pe ideea de a transforma imaginea de intrare într-o matrice de caracteristici, care este apoi transmisă printr-o serie de straturi de auto-attenție (self-attention) și rețele neurale complet conectate (*fully connected neural networks*). Aceste straturi sunt proiectate pentru a învăța reprezentări ierarhice ale datelor de intrare, începând cu caracteristici simple și progresând spre caracteristici mai complexe.

Deoarece Vision Transformer nu utilizează straturi de convolução, ci se bazează pe auto-attenția straturilor, aceasta poate învăța dependențe de la distanțe mai lungi decât modelele tradiționale de viziune artificială. Aceasta permite modelului să înțeleagă mai bine relațiile dintre caracteristicile de intrare și să producă rezultate mai precise și mai exacte.

Arhitectura Vision Transformer a demonstrat performanțe remarcabile în domeniul viziunii artificiale, depășind alte modele tradiționale, precum rețelele neuronale convoluționale pe unele seturi de date de referință. De asemenea, Vision Transformer este utilizat cu succes în alte domenii, precum procesarea naturală a limbajului și generarea de texte.[4]

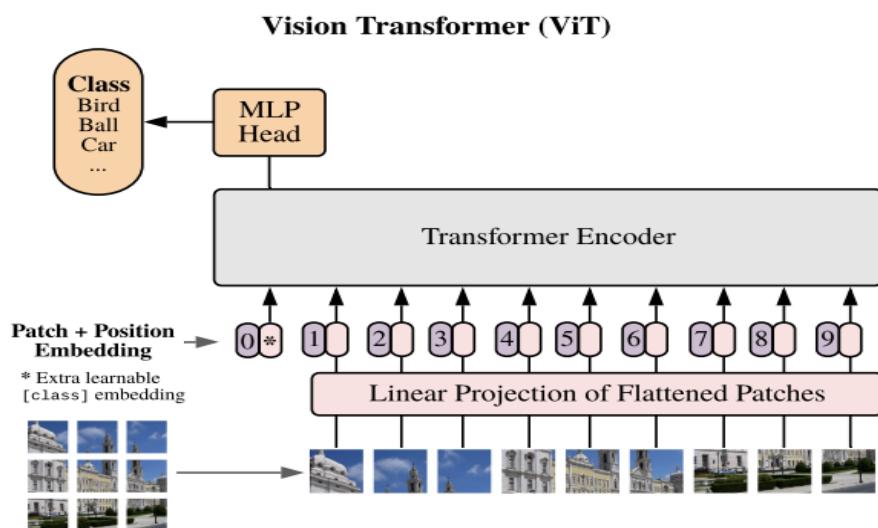


Figura 3.3: Vision Transformer

3.2 Setul de date

Setul de date folosit în antrenarea modelelor este disponibil publicului, lansat alături de lucrarea **“A Large-Scale Benchmark for Food Image Segmentation”**[32]. Acesta conține 103 tipuri de alimente și 7118 imagini segmentate, cu posibilitatea de a avea multiple alimente în aceeași fotografie.

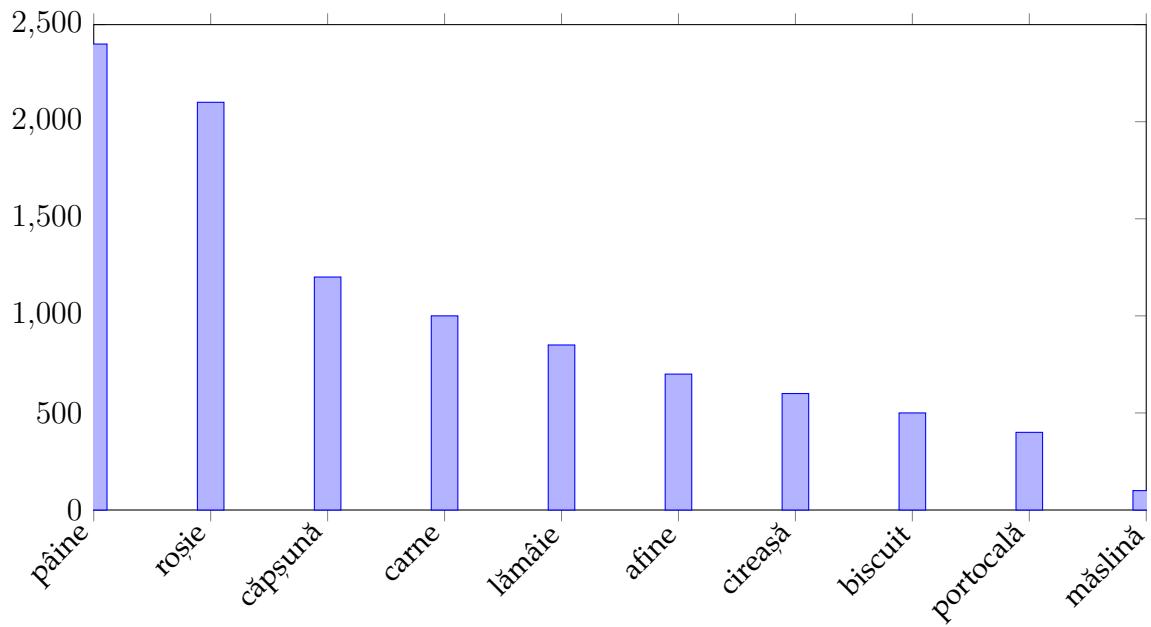


Figura 3.4: Distribuția tipurilor de alimente prezente în setul de date [32]

Arhitecturile menționate anterior sunt cunoscute pentru faptul că necesită un volum mare de date de intrare, de aceea am apelat la o îmbunătățire a setului de date prin aplicarea anumitor transformări din librăria Python **albumentations**[1], prezentate în figurile 3.5, 3.6



Figura 3.5: Transformările aplicate unei imagini

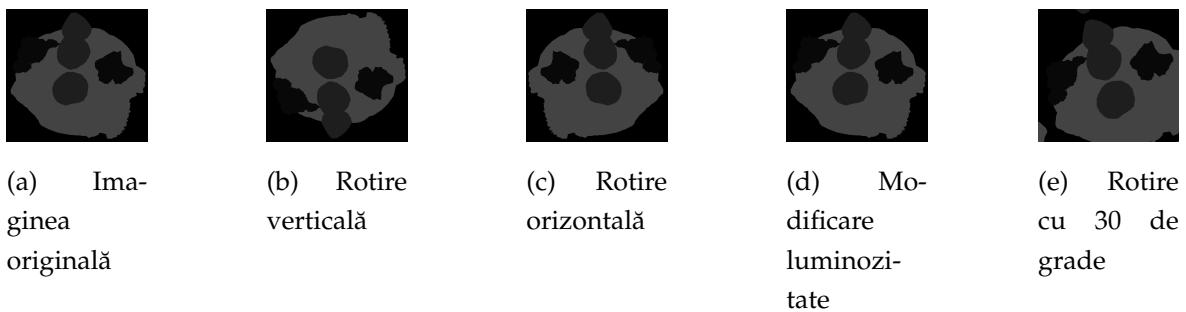


Figura 3.6: Transformările aplicate măștii asociate imaginii de mai sus

Astfel, am reușit să mărim de 4 ori dimensiunea setului de date. Antrenamentul a avut loc pe o mașină Asus TUF Gaming A15, cu sistem de operare Windows 10, CPU AMD Ryzen 7 4800H, 16 GB RAM, GPU nVidia GeForce RTX 2060 6GB memorie, permitându-se utilizarea tehnologiei CUDA(**Compute Unified Device Architecture**) în procesul de antrenament. Cu toate acestea, timpul necesar unei epoci de antrenament în cazul arhitecturilor bazate pe convoluții ajungea până la 4 ore, am recurs la o micșorare a setului de date, la aproximativ 12000 de exemple, realizând o diversitate a datelor în schimbul dimensiunii setului. Această problemă nu a fost întâlnită la arhitectura ce folosea Vision Transformer, pentru întreg setul de date a fost necesar aproximativ o oră pentru parcurgerea unei epoci. În acest caz, nu am alterat dimensiunea setului de date, antrenând cu cele 24000 de imagini.

Pentru partea de antrenament datele au fost împărțite în mod aleator în date de antrenament și date de test, conform unei proporții de 80:20. Înainte de a antrena rețeaua, am aplicat un alt set de transformări: redimensionare și normalizare. Redimensionarea la 256px x 256px asigură procesarea tuturor imaginilor la aceeași mărime, iar prin normalizare fiecare canal de culoare (roșu, verde și albastru) al imaginii este normalizat individual, prin scăderea valorii medii (0.5) și apoi împărțirea la deviația standard (0.5). Rezultatul este că valorile pixelilor pentru fiecare canal de culoare sunt acum distribuite în mod uniform în intervalul [-1, 1].

3.3 Funcția de loss

Cross Entropy Loss

Funcția de loss Cross Entropy, adesea denumită și Log Loss, este o măsură larg utilizată pentru performanța unui model de clasificare. În contextul Deep Learning, unde modelul este antrenat pentru a învăța o distribuție de probabilitate peste clasele de predicție, Cross Entropy devine o alegere naturală ca funcție de loss. Aceasta măsoară discrepanța dintre distribuția reală de probabilitate și distribuția de probabilitate estimată de model. În mod formal, dacă y este eticheta adevărată și \hat{y} este

probabilitatea estimată de model, Cross Entropy Loss pentru un singur exemplu poate fi definită ca $L = -y \log(\hat{y})$. Aceasta se generalizează la cazul cu mai multe clase însușind pierderea pentru fiecare clasă. Scopul antrenării modelului este să minimizeze această pierdere, aducând astfel distribuția estimată mai aproape de distribuția reală. În practică, aceasta conduce la o mai bună acuratețe de clasificare.

```
loss_fn = torch.nn.CrossEntropyLoss()
```

Intersection over Union Loss

Intersection over Union (IoU), cunoscut și sub denumirea de Jaccard Index, este o metrică comună utilizată în problemele de segmentare semantică și de detectare a obiectelor pentru a măsura acuratețea predicțiilor modelului. În termeni matematici, IoU este calculat ca raportul dintre intersecția și uniunea a două mulțimi. Pentru o predicție de segmentare A și realitatea B , IoU este definit ca:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (3.1)$$

Unde:

- $|A \cap B|$ reprezintă numărul de pixeli comuni între predicție și adevăr
- $|A \cup B|$ reprezintă numărul total de pixeli acoperiți de ambele mulțimi

Cu cât valoarea IoU este mai mare, cu atât predicția este mai bună. O reprezentare grafică a formulei se poate vedea în Figura 3.7.

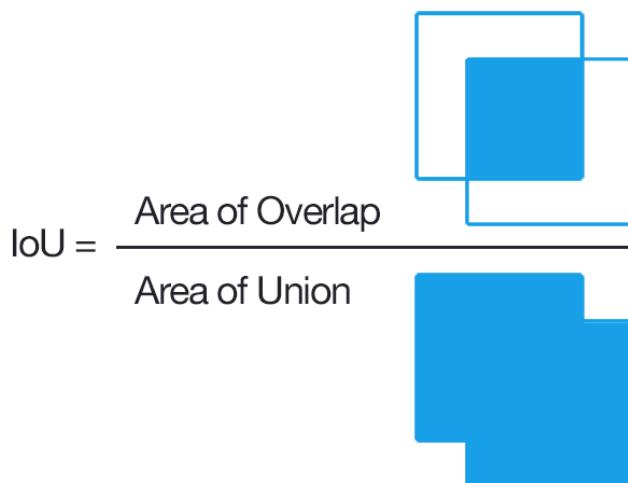


Figura 3.7: Vizualizare IoU [26]

```
# iou_loss.py
class IoULoss(nn.Module):
    def __init__(self, eps=1e-6):
        super(IoULoss, self).__init__()
        self.eps = eps

    def forward(self, predictions, targets):
        # Convert predictions to probabilities using softmax
        probabilities = torch.softmax(predictions, dim=1)
        # One-hot encode the targets
        targets_one_hot = torch.zeros_like(predictions).scatter_(1,
            → targets.unsqueeze(1), 1)
        # Compute intersection and union
        intersection = torch.sum(probabilities * targets_one_hot, dim=(2,
            → 3))
        union = torch.sum(probabilities, dim=(2, 3)) +
            → torch.sum(targets_one_hot, dim=(2, 3)) - intersection
        # Calculate IoU and then IoU loss
        iou = (intersection + self.eps) / (union + self.eps)
        iou_loss = 1 - torch.mean(iou)
        return iou_loss
```

Cross Entropy Loss + Intersection over Union Loss

```
# ce_iou_loss.py
class CE_IOU_Loss(nn.Module):
    def __init__(self, alpha=0.5):
        super(CE_IOU_Loss, self).__init__()
        self.alpha = alpha
        self.cross_entropy_loss = CrossEntropyLoss()
        self.iou_loss = IoULoss()

    def forward(self, predictions, targets):
        ce_loss = self.cross_entropy_loss(predictions, targets)
        iou_loss = self.iou_loss(predictions, targets)
        combined_loss = self.alpha * ce_loss + (1 - self.alpha) *
            → iou_loss
        return combined_loss
```

3.4 Optimizer

Optimizatorii sunt algoritmi sau metode utilizate pentru a modifica atributele rețelei neurale, cum ar fi ponderile și rata de învățare, cu scopul de a reduce pierderile.

Adam

Adam [15] este un algoritm de optimizare care poate fi utilizat în locul procedurii clasice a gradientului descendente stochastic pentru a actualiza ponderile rețelei în mod iterativ, bazându-se pe datele de antrenament. Numele Adam provine de la estimare adaptivă a momentului (*adaptive moment estimation*). Ecuațiile pentru algoritmul Adam sunt următoarele:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)dW \quad (3.2)$$

Aceasta este o medie mobilă exponențială a gradientului.

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)dW^2 \quad (3.3)$$

Aceasta este o medie mobilă exponențială a pătratului gradientului, ceea ce servește ca estimare a variabilității gradientului.

$$W_t = W_{t-1} - \frac{\alpha}{\sqrt{v_t} + \epsilon} m_t \quad (3.4)$$

Aceasta este formula de actualizare a ponderilor.

Unde:

- α este rata de învățare
- m_t este estimarea momentului de ordinul întâi
- v_t este estimarea momentului de ordinul al doilea
- dW reprezintă gradientul curent, derivata funcției de loss în raport cu parametrii rețelei
- ϵ este o constantă mică pentru a preveni împărțirea la zero.

3.5 Funcții de activare

O funcție de activare este o funcție care este adăugată într-o rețea neurală artificială pentru a ajuta rețeaua să învețe modele complexe în date. Comparând cu un model bazat pe neuroni care se află în creierul nostru, funcția de activare decide în final ce urmează să fie transmis către neuronul următor. Aceasta preia semnalul de ieșire de la neuronul precedent și îl convertește într-o formă care poate fi considerată intrare pentru neuronul următor.

ReLU

Funcția de activare liniară rectificată (ReLU) este o funcție liniară pe ramuri care va reda intrarea direct dacă aceasta este pozitivă, altfel, va reda zero. A devenit funcția de activare implicită pentru multe tipuri de rețele neuronale deoarece un model care o folosește este mai ușor de antrenat și adesea obține performanțe mai bune.

$$ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \quad (3.5)$$

Softmax

Softmax este o funcție care convertește un vector de numere reale într-un vector de valori de probabilitate. Fiecare element al vectorului de ieșire al funcției softmax reprezintă probabilitatea ca intrarea să aparțină unei anumite clase. Softmax este adesea utilizată în stratul final al unei rețele neuronale care trebuie să efectueze clasificarea. În acest caz, fiecare nod din stratul softmax corespunde unei categorii pe care modelul a fost instruit să o clasifice. Ieșirea acestui nod (după aplicarea funcției softmax) este interpretată ca probabilitatea ca intrarea să aparțină acelei categorii.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^N e^{z_k}} \quad (3.6)$$

Unde:

- $\sigma(\mathbf{z})_j$ este softmax-ul componentei a j -a a vectorului de intrare \mathbf{z}
- z_j este componenta a j -a a vectorului de intrare
- N este numărul de clase

3.6 Metrici de evaluare

3.6.1 Coeficientul Sørensen-Dice

Coeficientul Sørensen-Dice este o măsură a similarității între multimi, definit ca fiind de două ori mărimea intersecției împărțită la reuniunea celor două.

$$\text{Dice score} = 2 * \frac{|A \cap B|}{|A| + |B|} \quad (3.7)$$

O reprezentare grafică a formulei se poate vedea în Figura 3.8.

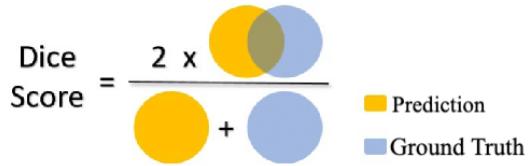


Figura 3.8: Vizualizare coeficient Sorensen-Dice [24]

3.6.2 Acuratețea

Acuratețea este una dintre metodele de evaluare a modelelor de clasificare. Informal, acuratețea reprezintă fracția predictiilor pe care modelul nostru le-a realizat corect. Formal, are următoarea definiție [7]:

$$\text{Acuratețea} = \frac{\text{Numărul de prediții corecte}}{\text{Numărul total de prediții}} \quad (3.8)$$

$$\text{Acuratețea} = \sum_{i=1}^N \frac{TP_i}{\sum_{j=1}^N TP_j + FP_j + FN_j}$$

Unde:

- TP_j este numărul de rezultate adevărat pozitive pentru clasa j
- FP_j este numărul de rezultate fals pozitive pentru clasa j
- FN_j este numărul de rezultate fals negative pentru clasa j
- N este numărul total de clase

3.7 Experimente realizate

3.7.1 Primul experiment: UNet

Configurare

O primă abordare a presupus implementarea arhitecturii UNet, fiind o alegere populară pentru sarcinile de segmentare a imaginilor datorită capacității sale de a localiza precis obiectele într-o imagine. Arhitectura UNet este alcătuită dintr-o parte de codare, responsabilă cu extragerea caracteristicilor, și o parte de decodare, care utilizează caracteristicile extrase pentru a realiza o clasificare precisă. Particularitatea arhitecturii este reprezentată de *skip-connections* ce fac legătura orizontală dintre rezultatul codării cu stratul corespondent decodării imaginii.

Parametrii

Experimentul desfășurat a avut ca scop găsirea unei funcții de loss, care prin minimizarea acesteia să producă metrii cât mai performante. Astfel, am rulat $N = 20$ epoci de antrenament, dimensiunea unui mini-batch $B = 16$ și cele 2 funcții de loss exemplificate anterior, *CrossEntropyLoss* și *IoUCrossEntropyLoss*, cu $\alpha = 0.5$ (ponderile pentru importanța *CrossEntropyLoss* și *IoULoss* fiind egale). Timpul de antrenament pentru o epoca este de aproximativ 2 ore. Numărul parametrilor antrenabili al rețelei este aproximativ 31 de milioane, ponderile fiind calculate prin intermediul optimizer-ului *Adam*, cu rata de învățare $\alpha = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. *GradScaler* este o clasă în PyTorch care ajută la scalarea gradientului în timpul antrenamentului cu precizie mixtă. Scopul său este de a evita problemele de instabilitate numerică a gradientului în timpul antrenamentului. *GradScaler* asigură că gradientele sunt într-un interval adecvat, îmbunătățind stabilitatea și convergența antrenamentului pentru rețeaua neuronală.

Rezultate

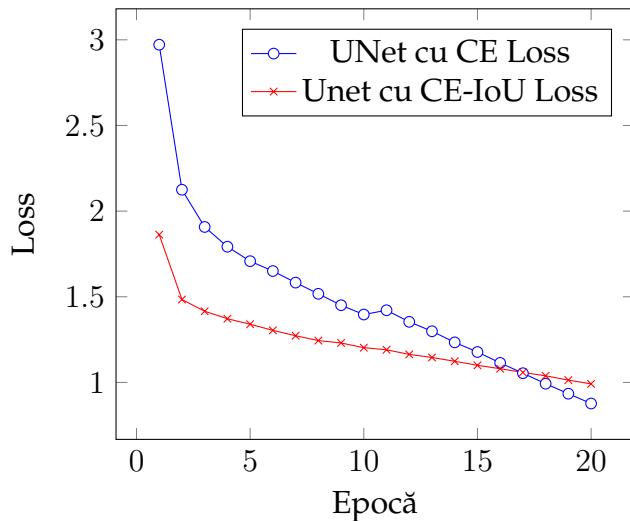


Figura 3.9: Evoluția Loss-ului în funcție de epocă

Model	Epoch	Loss	Acc	Dice	IoU	#Parametrii	
						#Total	#Antrenabili
UNet_CE_Loss	20	0.8769	67.27	57.48	87.92	31M	31M
UNet_CE_IoU_Loss	20	0.9915	68.64	57.71	89.48	31M	31M

Tabela 3.1: Metrici de performanță UNet

Funcția de loss *IoU-CE* prezintă o valoare inițială semnificativ mai mică în comparație cu funcția de loss *CE*, dar arată o rată de convergență mult mai lentă. Intersecția

dintre aceste două funcții de loss devine evidentă după aproximativ 17 epoci de antrenament. În ciuda acestei convergențe mai lente, metricile calculate (Acuratețe, scor Dice, Intersection over Union) indică o performanță superioară pentru funcția de loss *IoU-CE*. Prin urmare, funcția de loss *IoU-CE* va fi folosită pentru evaluarea performanței modelelor ulterioare. Pentru a reduce numărul de categorii segmentate am aplicat un prag astfel încât numărul de pixeli dintr-o categorie să fie cel puțin 2% din numărul total. Așa cum se poate observa în Figura 3.10, cel de al doilea model oferă segmentări mai bune, reducând atât numărul de categorii false și asigurând o mască mai precisă.

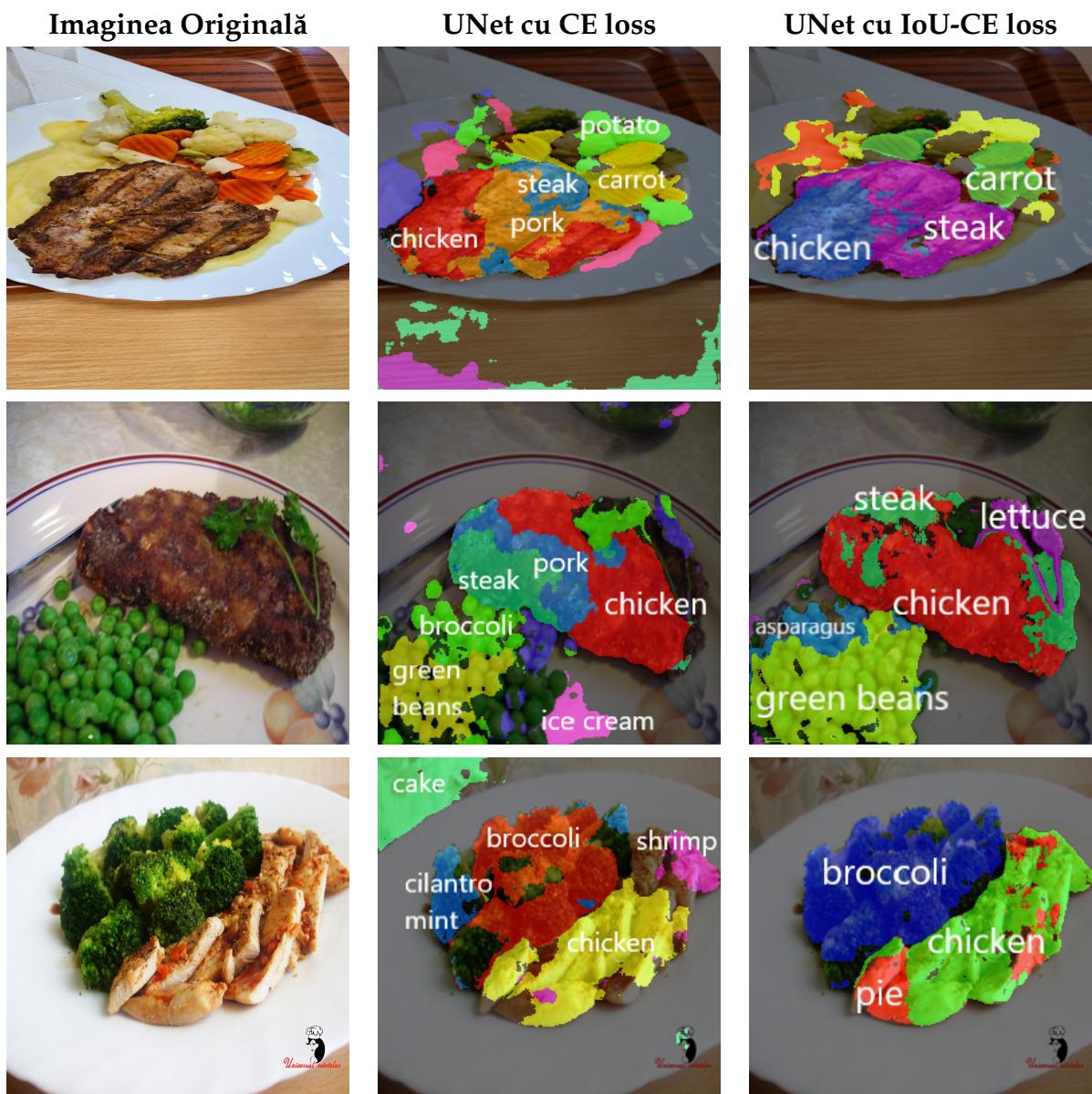


Figura 3.10: Măștile de segmentare rezultate de UNet cu CrossEntropy Loss și IoU-CrossEntropy Loss

3.7.2 Al doilea experiment: ResNet Encoder în arhitectura UNet

Configurare

În cadrul acestui experiment, am dorit să îmbunătățim arhitectura UNet prin integrarea caracteristicilor de extragere a informațiilor din rețeaua ResNet-152. În loc să utilizăm partea de codare originală din UNet, am înlocuit-o cu primele 4 straturi ale rețelei ResNet-152. Aceste straturi sunt pre-antrenate pe setul de date ImageNet[2], ceea ce înseamnă că ele au fost antrenate pentru a extrage caracteristici semnificative din imagini. Prin înlocuirea părții de codare a arhitecturii UNet cu un model ResNet pre-antrenat, ne propunem să profităm de punctele forte ale ambelor arhitecturi. În mod specific, ne așteptăm ca arhitectura UNet să continue să ofere localizarea precisă a obiectelor într-o imagine, în timp ce modelul ResNet pre-antrenat va extrage caracteristicile semnificative.

Parametrii

Arhitectura rezultată are o complexitate mai mare, având un total de 248 de milioane de parametri, dintre care 190 de milioane sunt antrenabili. Cele 4 straturi ale ResNet-152 adaugă aproximativ 58 de milioane de parametrii neantrenabili. Menținerea conexiunilor orizontale în partea de decodare ne permite să păstrăm informațiile spațiale importante în procesul de clasificare. Astfel, am rulat $N = 20$ epoci de antrenament, dimensiunea unui mini-batch $B = 8$, funcția de loss *IoUCrossEntropyLoss*, optimizer-ul *Adam*, cu rata de învățare $\alpha = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ și scaler-ul *GradScaler*. Timpul de antrenament pentru o epocă a fost de aproximativ 2 ore și 10 minute, diferență față de primul experiment fiind insesizabilă, chiar dacă numărul de parametri este de 7 ori mai mare, demonstrând, de asemenea, o performanță mai ridicată.

Rezultate

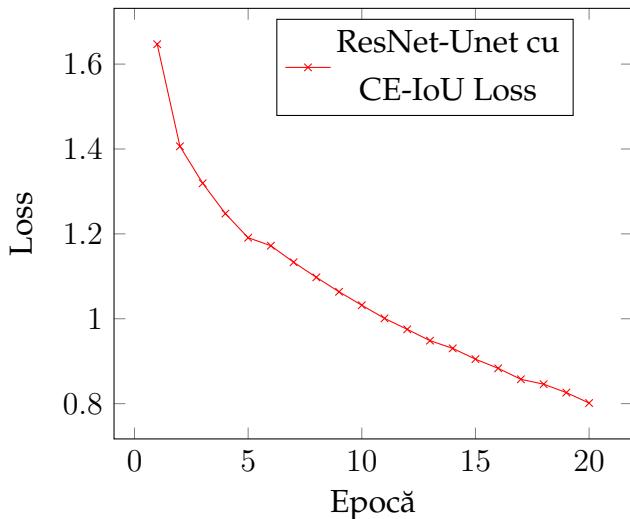


Figura 3.11: Evoluția Loss-ului în funcție de epocă

Model	Epoch	Loss	Acc	Dice	IoU	#Parametrii	
						#Total	#Antrenabili
UNet cu ResNet	20	0.8574	77.92	60.32	92.73	247M	190M

Tabela 3.2: Metrici de performanță ResNet Encoder în arhitectura UNet

Putem observa o creștere semnificativă a metricilor calculate datorată faptului că am folosit un număr de 58 de milioane de parametri "înghețăți", nefiind afectați în cadrul antrenamentelor desfășurate. Alături de faptul că aceștia au fost calculați pe baza setului ImageNet[2], am putut avea în arhitectură hărți de caracteristici relevante în identificarea obiectelor din imagine. În continuare am aplicat același prag de 2% pentru a reduce din categoriile puțin probabile prezise.

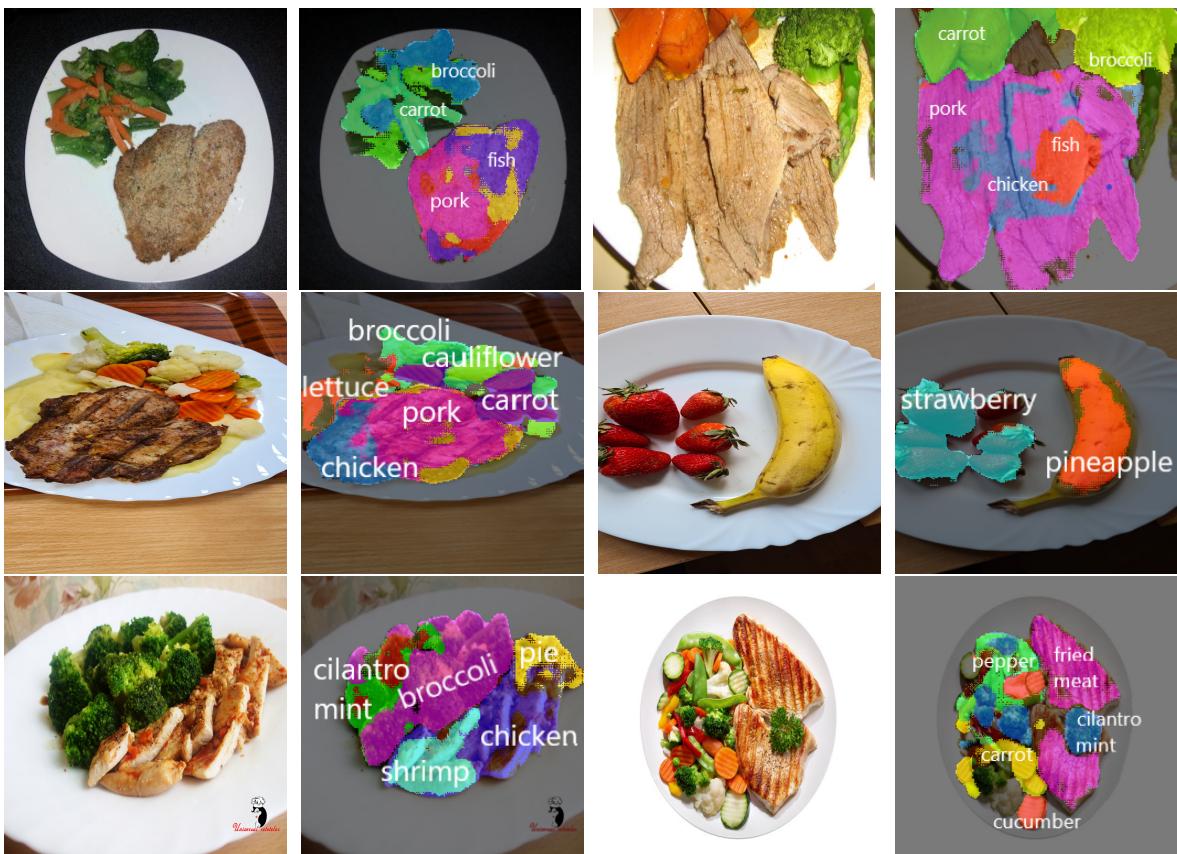


Figura 3.12: Măștile de segmentare rezultate de ResNet,UNet

3.7.3 Al treilea experiment: SAM Image Encoder cu Decoder propriu

Configurare

Cea de a treia abordare schimba cu totul paradigma și utilizează un Vision Transformer, concret un Masked Autoencoder, pentru partea de extragere a caracteristicilor, iar convoluțiile se folosesc cu scopul de a mări dimensiunea output-ului. Această arhitectură este inspirată din lucrarea Segment Anything[16]. Modelul menționat este conceput pentru a segmenta un obiect de interes dintr-o imagine, având în vedere anumite prompt-uri furnizate de un utilizator. Modelul este solicitat să returneze o mască de segmentare validă chiar și în prezența ambiguității obiectelor. Ideea generală din spatele abordării este că modelul a învățat conceptul unui obiect și poate segmenta orice obiect care este indicat. Acest lucru rezultă într-un potențial ridicat pentru a putea segmenta obiecte de tipuri pe care nu le-a văzut fără nicio formare suplimentară, adică performanță ridicată în regimul de învățare zero-shot.[16] Am extras ImageEncoder-ul folosit în antrenarea modelului SAM, rezultând 128 de caracteristici de dimensiune 16×16 . Având setat ca dimensiunea intrării la 256×256 px, modificarea a fost una minimală, pur experimentală adăugând doar 4 straturi de

convolutii transpuse (*ConvTranspose2d*), cu scopul de a dubla dimensiunea la fiecare pas.

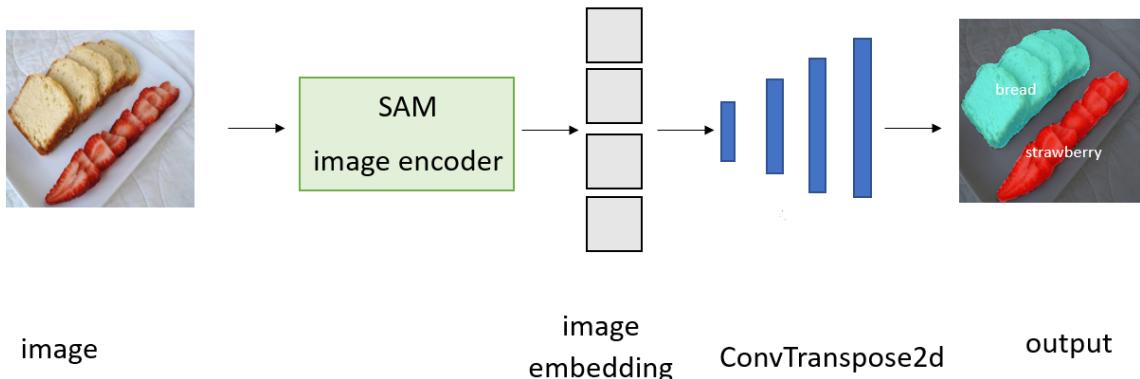


Figura 3.13: Arhitectura SAM Image Encoder + Decoder

Parametrii

Astfel, am rulat $N = 40$ epoci de antrenament, dimensiunea unui mini-batch $B = 8$, funcția de loss *IoUCrossEntropyLoss*, optimizer-ul *Adam*, cu rata de învățare $\alpha = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ și scalarul *GradScaler*. Funcția de activare utilizată la fiecare pas a fost *ReLU*. Timpul de antrenament pentru o epocă a fost de aproximativ 45 de minute, permitându-ne rularea a mai multor epoci față de experimentele precedente.

Rezultate

O constatare notabilă este îmbunătățirea semnificativă a acurateței, cu un procentaj majorat cu 13% comparativ cu modelul UNet și cu 5% mai mare decât modelul UNet integrat cu ResNet Encoder. De asemenea, trebuie subliniat faptul că antrenamentul a fost implementat pe întregul set de date, care cuprinde 24000 de imagini, din care aproximativ 5000 au fost dedicate pentru testare. Chiar dacă valorile celorlalte mătrici au înregistrat o scădere în cadrul acestui experiment, figuri reprezentative prezентate în 3.15 evidențiază performanța superioară a modelului în cadrul aplicației. Într-adevăr, acest model constituie alegerea optimă pentru integrarea în aplicația de jurnalizare a dietei în curs de dezvoltare.

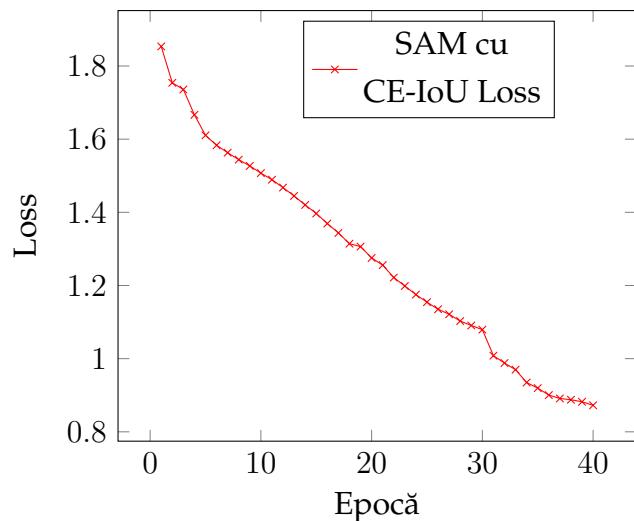


Figura 3.14: Evoluția Loss-ului în funcție de epocă

Model	Epoch	Loss	Acc	Dice	IoU	#Parametrii	
						#Total	#Antrenabili
SAM IE cu Convoluții	40	0.8725	83.29	56.99	76.11	86M	86M
UNet_CE_Loss	20	0.8769	67.27	57.48	87.92	31M	31M
UNet_CE_IoU_Loss	20	0.9915	68.64	57.71	89.48	31M	31M
UNet cu ResNet	20	0.8574	77.92	60.32	92.73	247M	190M

Tabela 3.3: Comparația dintre modele

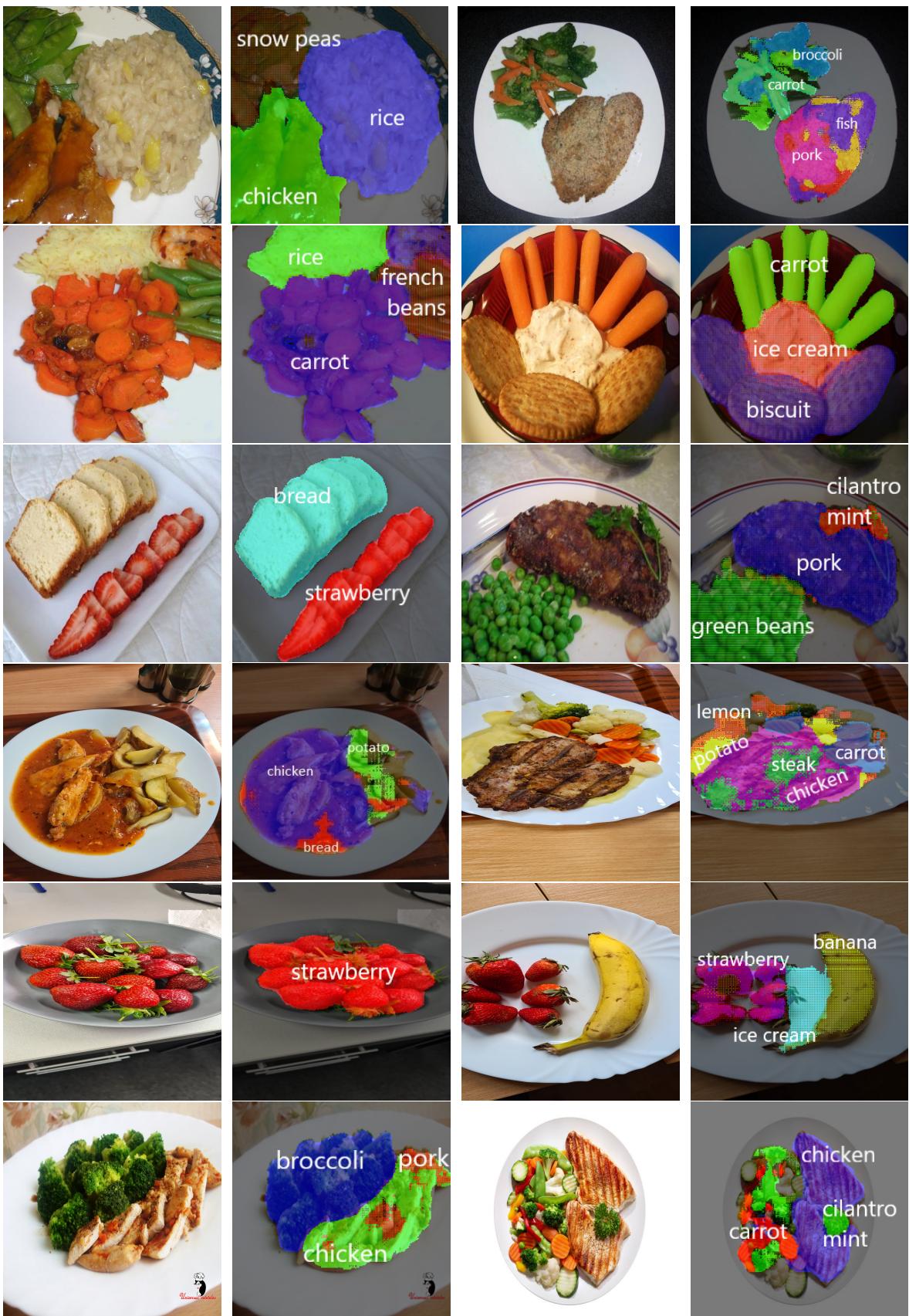


Figura 3.15: Măștile de segmentare rezultate SAM Image Encoder cu Convoluții

Capitolul 4

Arhitectura și dezvoltarea produsului software

4.1 Descrierea cerințelor

Prin intermediul aplicației, utilizatorul își va putea monitoriza comportamentul alimentar. La fiecare masă, aplicația va putea identifica alimentele utilizate cu ajutorul unei fotografii realizate pe moment sau alegerea din galeria dispozitivului. În cele din urmă, aplicația va ține evidență caloriilor consumate, identificând o mare parte din alimentele prezente. În plus, asigurăm facilități pentru introducerea manuală a alimentelor care nu au fost recunoscute anterior, stocându-le în baza noastră de date și făcându-le accesibile tuturor utilizatorilor. Această abordare servește ca ultimă soluție în cazul identificărilor eronate generate de modelul nostru. Această caracteristică contribuie nu doar la îmbogățirea bazei noastre de date, dar permite și o flexibilitate crescută pentru utilizatori, sporind eficacitatea aplicației în procesul de monitorizare a dietei.

Un alt aspect al procesului nostru de înregistrare constă în oferirea diverselor diete de referință, care sunt personalizate pe baza unui algoritm care consideră o serie de factori importanți, inclusiv rata metabolică de bază, nivelul de activitate și scopul utilizatorului. În plus, luăm în considerare și distribuția nutrientilor - raportul de carbohidrați, proteine și lipide necesare pentru o alimentație optimă - pentru a asigura că beneficiarul primește o dietă echilibrată și nutritivă.

4.1.1 Cazuri de utilizare

Principala funcționalitate a aplicației constă în **adăugarea alimentelor** la o masă, proces care poate fi realizat în mai multe moduri. Un prim mod de adăugare este prin selectarea alimentului căutat dintr-o listă sau introducerea manuală a acestuia în cazul în care nu se găsește în listă. Cu scopul de a extinde funcționalitatea și de a

oferi o experiență inovatoare, am implementat o alternativă care implică încărcarea unei fotografii în timp real sau din galeria de imagini a farfuriei cu mâncare. Prin utilizarea algoritmului specific, aplicația poate furniza sugestii privind alimentele prezente în fotografie și utilizatorul poate adăuga aceste alimente în modul corespunzător în cadrul mesei. De asemenea, am implementat operații de ștergere sau modificare a cantității alimentului introdus. Conform ilustrației prezentate în Figura 4.1, se poate observa că singura funcționalitate care poate fi accesată fără a finaliza procesul de autentificare este *Înregistrarea*. Toate celelalte funcționalități, necesită finalizarea cu succes a cazului de *Autentificare* înainte de a fi utilizate.

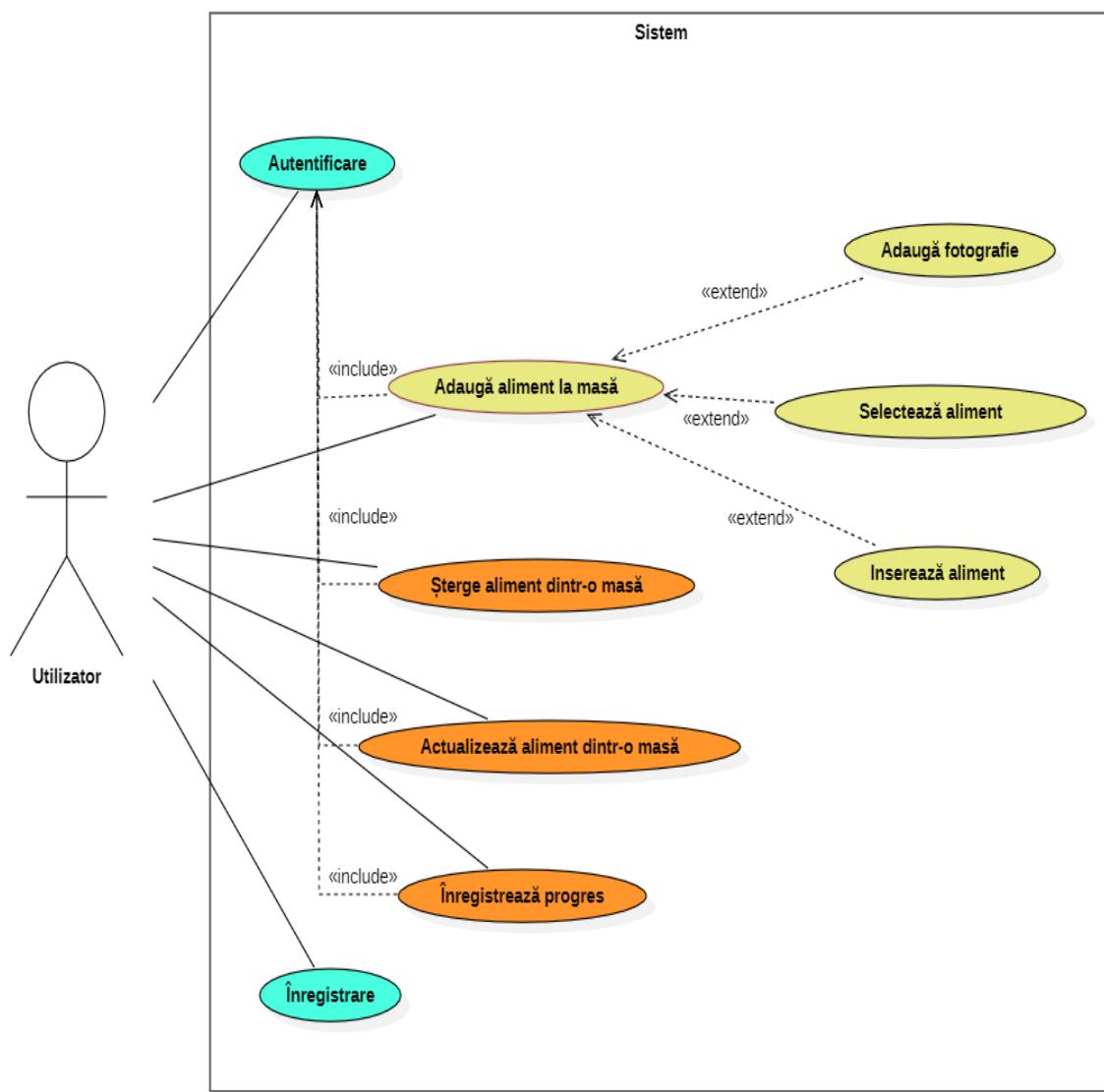
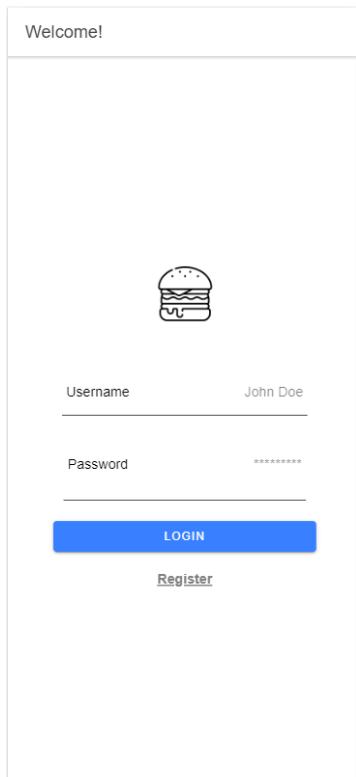
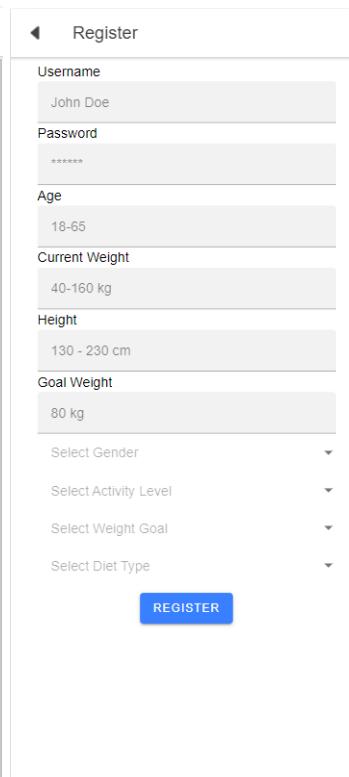


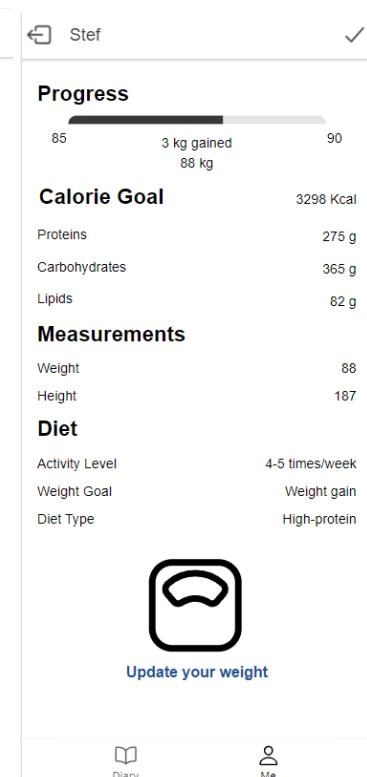
Figura 4.1: Diagrama cazurilor de utilizare



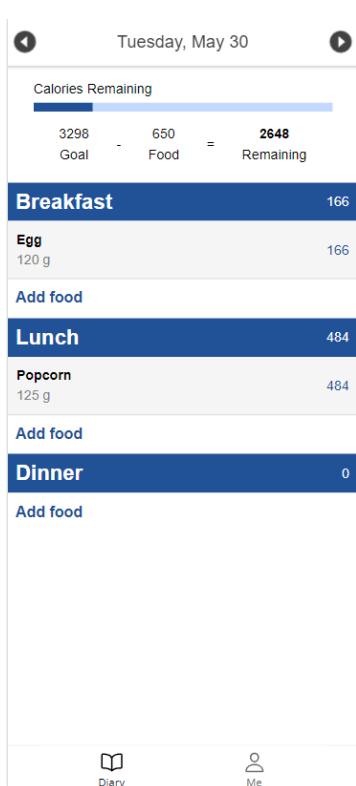
(a) Autentificare



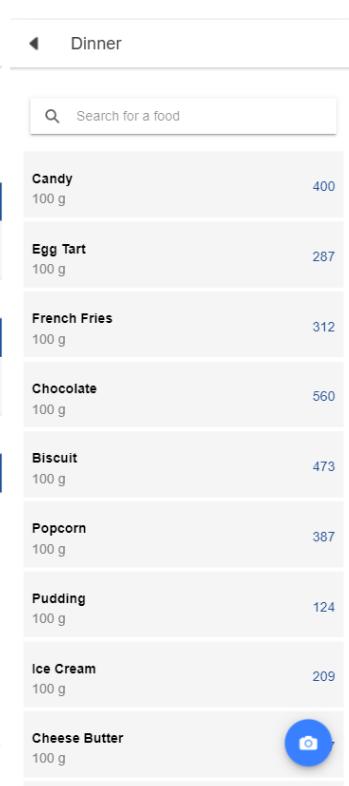
(b) Înregistrare



(c) Profilul Meu



(d) Zi de jurnal



(e) Afisare alimente



(f) Predictie

Figura 4.2: Capturi de ecran din aplicatie

4.1.2 Arhitectura obiectuală

Pentru a implementa modelul relational bazat pe modelul obiectual în aplicația noastră, am folosit un ORM (Object-Relational Mapping) numit Spring JPA. Aceasta ne permite să mapăm obiectele din clasele noastre de domeniu direct în tabelele corespunzătoare din baza de date relatională, MySQL. Obiectul central al aplicației este reprezentat de clasa FoodMeal, fiind o tabelă de legături de $n - m$ (*many to many*) dintre toate clasele aplicației. Esența unui obiect de acest tip este reprezentată prin faptul că un utilizator A, a adăugat alimentul B, la masa C, la data D, având cantitatea E (în grame).

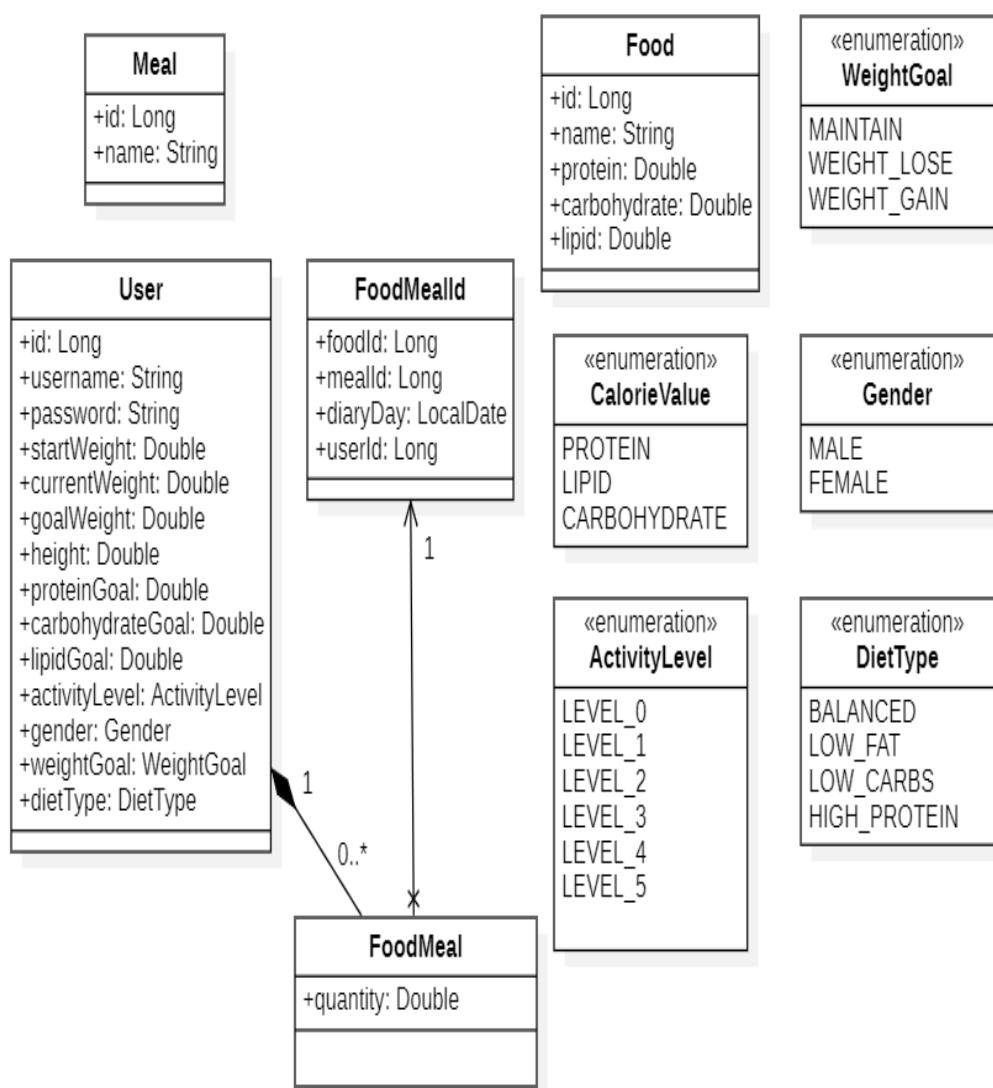


Figura 4.3: Diagrama de clase

4.2 Arhitectura client-server

Arhitectura client-server este un model în care serverul găzduiește, furnizează și gestionează majoritatea resurselor și serviciilor consumate de către client. Acest tip de arhitectură are unul sau mai multe calculatoare client conectate la un server central printr-o rețea sau o conexiune la internet. Arhitectura client-server este cunoscută și sub numele de model de calcul în rețea sau rețea client-server, deoarece toate solicitările și serviciile sunt furnizate prin intermediul unei rețele.

Arhitectura client-server este cea mai utilizată pentru aplicațiile care necesită o separare sau abstractizare a responsabilităților între client și server; este destinată sistemelor cu o inter-operabilitate ridicată. Stilul arhitectural client-server ajută aplicațiile să îmbunătățească performanța în ceea ce privește scalabilitatea. Astfel, prin intermediul acestui model, se permite distribuirea sarcinilor și a resurselor de procesare între servere și clienti, ceea ce duce la o utilizare eficientă a resurselor și la îmbunătățirea performanțelor sistemului în ansamblu. [22]

Servicii REST

Serviciile REST (*REpresentational State Transfer*) sunt un set de principii de arhitectură care definesc modul în care serviciile web pot fi proiectate. Ele utilizează protocoalele standard HTTP.

Un serviciu REST utilizează metodele precum GET pentru a prelua date, POST pentru a crea date, PUT pentru a actualiza date și DELETE pentru a șterge date. Acestea oferă o abordare fără stări, în care fiecare solicitare de la client către server trebuie să conțină toate informațiile necesare pentru a o procesa.

Arhitectură stratificată

Arhitectura stratificată reprezintă unul dintre cele mai recurante stiluri arhitecturale. Conceptul de bază care stă la fundația arhitecturii stratificate este organizarea modulară, în straturi orizontale, a funcționalităților similare. În acest sens, fiecare strat îndeplinește un rol specific în cadrul aplicației. Astfel, se obține o claritate structurală și o eficiență funcțională optimizată, fiecare strat având o autonomie și o responsabilitate clar delimitată în economia generală a aplicației.

De exemplu, stratul de prezentare se însărcinează cu administrarea întregii comunicări cu utilizatorul, în timp ce stratul de business se focalizează pe implementarea funcționalităților esențiale ale aplicației. Stratul de persistență, pe de altă parte, este destinat să faciliteze interacțiunea cu baza de date, furnizând datele necesare pentru procesare către stratul de business. Fiecare strat în cadrul acestei arhitecturi formează o abstractizare a sarcinilor ce trebuie realizate pentru a îndeplini o anu-

mită solicitare. Astfel, stratul de prezentare nu are nevoie să cunoască modalitatea de obținere a datelor, rolul său fiind de a media comunicarea între utilizator și stratul de business prin primirea și răspunderea de cereri. Stratul de business, la rândul său, nu cunoaște detaliile legate de stocarea informațiilor, având drept sarcină să preia datele de la stratul de persistență și să execute logica specifică de business.

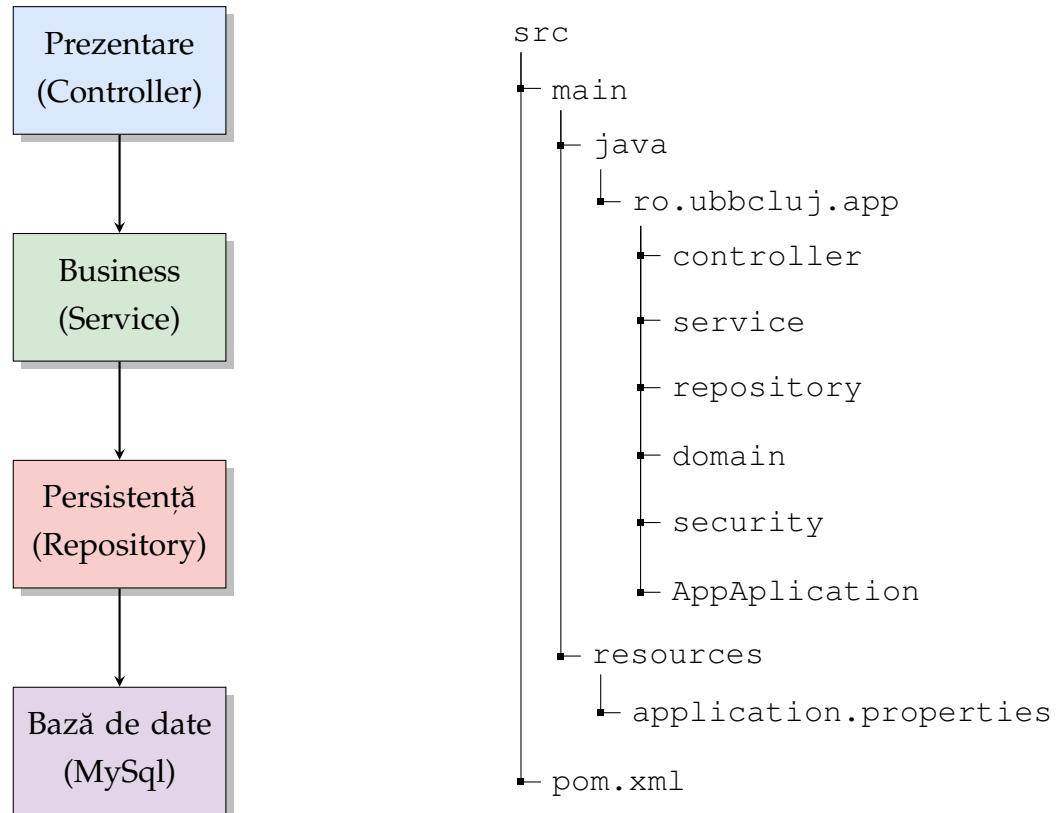


Figura 4.4: Arhitectura stratificată

4.3 Tehnologii folosite

Spring Boot [23]

Spring Boot este un framework de dezvoltare a aplicațiilor Java care facilitează crearea rapidă și eficientă a aplicațiilor web și a serviciilor bazate pe cloud. Aceasta oferă un set puternic de funcționalități și abstracții care permit dezvoltatorilor să se concentreze mai mult asupra logicii aplicației și mai puțin asupra configurațiilor și detaliilor tehnice.

Unul dintre principalele avantaje ale utilizării Spring Boot constă în faptul că acesta aduce un model de programare convențional în dezvoltarea aplicațiilor Java, oferind în același timp o experiență modernă și simplificată. Framework-ul vine cu un sistem avansat de gestionare a dependențelor și permite dezvoltatorilor să creeze rapid structura de bază a aplicației, astfel încât aceștia să poată concentra eforturile

pe implementarea funcționalităților cheie.

De asemenea, Spring Boot vine cu un set integrat de module și componente care acoperă o gamă largă de aspecte, cum ar fi securitatea, gestionarea bazelor de date, comunicarea în rețea și multe altele. Aceste module predefinite facilitează dezvoltarea aplicațiilor robuste și scalabile, reducând timpul și efortul necesare pentru implementarea funcționalităților comune.

Spring Boot este o tehnologie de dezvoltare a aplicațiilor Java care aduce un nivel înalt de productivitate și eficiență, permitând dezvoltatorilor să creeze rapid și ușor aplicații web și servicii bazate pe cloud. Prin abstractii puternice, configurare automată și un set integrat de module, Spring Boot facilitează dezvoltarea aplicațiilor robuste și scalabile, reducând timpul și efortul necesare pentru implementarea funcționalităților complexe.

Ionic Framework [18]

Ionic este un framework open-source pentru dezvoltarea de aplicații mobile și web bazate pe tehnologii web, cum ar fi HTML, CSS și JavaScript. Aceasta oferă un set de componente și instrumente care permit dezvoltatorilor să creeze aplicații cu interfețe utilizator atrăgătoare și o experiență fluidă.

Prin intermediul Ionic, dezvoltatorii pot crea aplicații mobile pentru platforme precum Android și iOS, utilizând un singur cod sursă. Acest lucru facilitează dezvoltarea rapidă și eficientă a aplicațiilor mobile, reducând timpul și efortul necesare pentru a crea aplicații pentru fiecare platformă în parte.

Unul dintre avantajele majore ale utilizării Ionic constă în faptul că acesta oferă o bibliotecă extinsă de componente predefinite, cum ar fi butoane, liste, formulare și navigație. Aceste componente pot fi personalizate și combinate pentru a crea interfețe utilizator adaptabile și ușor de utilizat.

Un alt aspect important al Ionic este suportul său pentru integrarea cu alte tehnologii și servicii, cum ar fi Angular, React și Cordova. Aceste integrări permit dezvoltatorilor să extindă funcționalitatea aplicațiilor lor și să utilizeze funcționalități avansate, cum ar fi autentificare, bază de date în timp real și acces la funcționalitățile hardware ale dispozitivelor mobile.

Ionic este un framework matur și versatil pentru dezvoltarea de aplicații mobile și web. Prin utilizarea tehnologiilor web familiare, componente predefinite și integrări cu alte tehnologii, Ionic facilitează dezvoltarea rapidă și eficientă a aplicațiilor mobile, oferind în același timp flexibilitate și posibilități de personalizare.

Flask[8]

Flask reprezintă un framework, dedicat limbajului de programare Python, construit cu scopul de a furniza un nucleu esențial de instrumente necesare dezvoltării de aplicații web. Deși este etichetat drept "micro" datorită abordării sale minimalistă, acesta nu compromite în ceea ce privește performanța sau scalabilitatea, oferind, în același timp, o capacitate considerabilă de adaptabilitate, ceea ce permite dezvoltatorilor să creeze aplicații cu grade variate de complexitate.

Microframework-ul Flask este definit prin filosofia sa de a furniza doar componentele fundamentale pentru realizarea unei aplicații web, lăsând astfel libertatea dezvoltatorilor de a selecta bibliotecile suplimentare în funcție de nevoile specifice ale proiectului. Această flexibilitate face ca Flask să fie adecvat pentru o varietate largă de proiecte, de la site-uri web de bază la aplicații web avansate și complexe.

În plus, Flask încorporează practici consacrate în domeniul dezvoltării web. Acesta oferă suport pentru modelul arhitectural *Model-View-Controller* (MVC) și este compatibil cu *Web Server Gateway Interface* (WSGI). De asemenea, Flask dispune de suport nativ pentru gestionarea rutelor URL, administrarea sesiunilor, procesarea formularelor și manipularea cererilor și răspunsurilor HTTP.

4.3.1 Integrarea Tehnologiei de Segmentare a Imaginilor

Securizarea aplicației

În sfera securității web, necesitatea unor metode eficiente, securizate și scalabile pentru gestionarea și transmiterea datelor de autentificare ale utilizatorilor este esențială. O metodă care a devenit un standard popular este Tokenul Web JSON (JWT), specificat oficial în IETF RFC 7519 [14].

Tokenurile Web JSON oferă un mijloc de a reprezenta datele care vor fi transfrate între două părți ale comunicării. În esență, aceste token-uri constau din obiecte JSON care sunt semnate digital folosind Semnătura Web JSON (JWS).

În scenariul de utilizare tipic, la autentificarea cu succes a utilizatorului, serverul generează un JWT care codifică informații relevante despre utilizator, cunoscute sub numele de *claims*. Acestea sunt apoi semnate digital de către server și returnate către client. Clientul stochează token-ul local și îl include în cererile ulterioare către server, de obicei în header-ul *Authorization*.

Serverul, la primirea unei cereri, poate verifica semnătura digitală folosind cheia sa secretă. Dacă semnătura este validă, serverul poate avea încredere în datele codificate în token (deoarece acestea ar fi putut fi semnate numai de către serverul însuși) și poate continua să proceseze cererea.

Această abordare oferă mai multe beneficii. În primul rând, elimină necesitatea

ca serverul să stocheze datele de sesiune, deoarece toate informațiile necesare sunt încapsulate în token. Aceasta este benefică pentru scalabilitate și performanță. În al doilea rând, facilitează autentificarea securizată, care poate fi deosebit de utilă în arhitecturi de microservicii.

Cu toate acestea, există motive de insecuritate de care trebuie să ținem cont când folosim JWT-uri. Tokenurile ar trebui să fie confidentiale, deoarece oricine obține acces la token îl poate folosi pentru a se pretinde utilizator. Prin urmare, ar trebui folosite metode de transmisie securizate, cum ar fi HTTPS. În plus, token-urile ar trebui să fie semnate pentru a preveni falsificarea și ar trebui să aibă un timp rezonabil de expirare pentru a reduce riscul de furt al token-ului.

```
// LoginController.java
@PostMapping()
public ResponseEntity<?> login(@RequestBody AuthenticationRequest
→ autheticationRequest) {
    User user = userService.login(autheticationRequest.getUsername(),
    → autheticationRequest.getPassword());
    if (user == null) {
        return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
    }
    String jwtToken = jwtTokenService.generateToken(user);
    return ResponseEntity.ok(jwtToken);
}

// AuthorizationFilter.java
protected void doFilterInternal(HttpServletRequest request,
→ HttpServletResponse response, FilterChain filterChain) {
    /*
    */
    String authorizationHeader = request.getHeader("Authorization");
    if (authorizationHeader == null || !authorizationHeader.startsWith("Bearer ")) {
        response.sendError(HttpStatus.SC_UNAUTHORIZED,
        → "Authorization header must be provided");
        return;
    } else {
        String token = authorizationHeader.substring(7);
        if (jwtTokenService.isTokenValid(token)) {
            response.sendError(HttpStatus.SC_UNAUTHORIZED,
            → "Authorization header expired");
            return;
        }
    }
}
```

```
// index.ts
export const authConfig = (token?: string) => ({
  headers: {
    'Content-Type': 'application/json',
    'Authorization': `Bearer ${token}`,
  }
})
```

Astfel, am detaliat procesul de comunicare securizată între client și serverul Spring Boot. Este de menționat, însă, că acesta din urmă colaborează și cu serverul Flask pentru accesarea modulului de segmentare a imaginii.

Un alt aspect esențial al securității sistemului nostru este generarea unei chei secrete comune, care este cunoscută atât de serverul Spring Boot și de cel Flask. Această cheie secretă servește ca un element de autentificare în comunicarea dintre cele două servere. În practică, atunci când serverul Spring Boot face o solicitare către serverul Flask, el include această cheie secretă în antetul solicitării. Când serverul Flask primește o astfel de solicitare, verifică cheia din antet cu propria cheie secretă. Dacă cele două corespund, serverul Flask va procesa solicitarea. Acest mecanism asigură că doar serverul Spring Boot, care deține cheia secretă, poate face solicitări către serverul Flask, oferind astfel un nivel suplimentar de securitate în sistemul nostru.

```
// ImageService.java
public OverlayCategoryDTO getSegmentedImage(byte[] imageBytes) throws
  InterruptedException {
  OverlayCategoryDTO overlayCategoryDTO = new OverlayCategoryDTO();
  try {
    Map<String, String> map = new HashMap<>();
    map.put("content", bytesToJsonArray(imageBytes));
    JSONObject jsonObject = new JSONObject(map);
    HttpRequest request = HttpRequest.newBuilder()
      .uri(URI.create(IMAGE_URL))
      .header("Content-Type", "application/json")
      .header("X-Api-Key", IMAGE_API_KEY) // @Value from
      // application.properties
      .POST(HttpRequest.BodyPublishers.ofString
        .valueOf(jsonObject)))
      .build();
    HttpClient client = HttpClient.newHttpClient();
    HttpResponse<String> response = client.send(request,
      HttpResponse.BodyHandlers.ofString());
    overlayCategoryDTO =
      parseImageResponseToOverlay(response.body());
  }
```

```
        } catch (IOException e) {
            logger.error("Error segmenting the image");
        }
        return overlayCategoryDTO;
    }

```

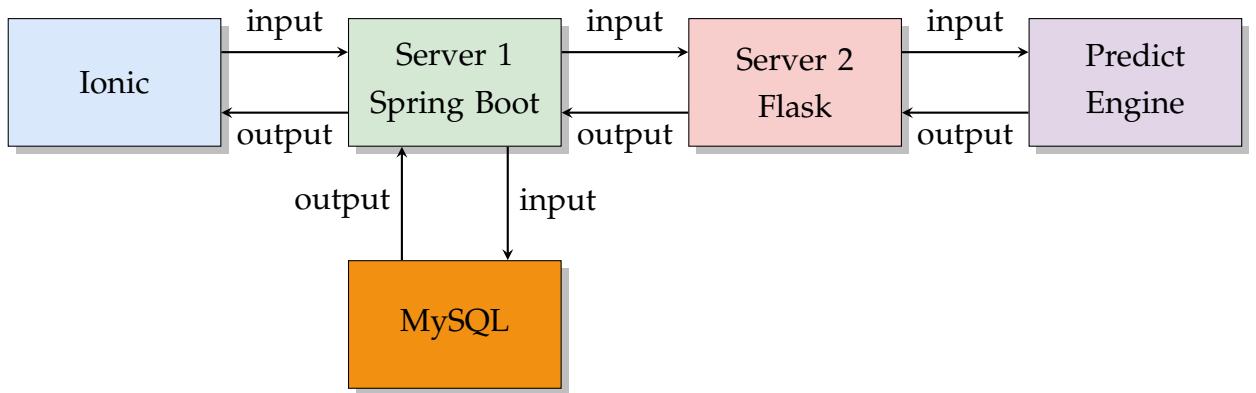
```
# app.py
def require_api_key(view_function):
    @wraps(view_function)
    def decorated_function(*args, **kwargs):
        if request.headers.get('X-Api-Key') and
            request.headers.get('X-Api-Key') == API_KEY: #API_KEY is an
            enviroment variable
            return view_function(*args, **kwargs)
        else:
            return jsonify({"message": "Unauthorized"}), 403
    return decorated_function

```

Servirea cererilor

În următorul fragment, ilustrăm fluxul de date în cadrul arhitecturii sistemului nostru, caracterizat printr-o interconexiune complexă a componentelor. Toate solicitările emise de clienți sunt inițial îndreptate către serverul Spring Boot. Aici, acestea sunt supuse unei inspecții riguroase, urmând a fi procesate în funcție de validitatea tokenului JWT atașat. Excepție o reprezintă solicitările legate de creația unui nou cont de utilizator, care sunt acceptate fără necesitatea verificării unui token.

În situația în care serverul Spring Boot dispune de informațiile necesare, va genera un răspuns imediat către client. În schimb, pentru cererile care implică procesarea de imagini, serverul Spring Boot va iniția o solicitare către serverul Flask, adăugând un header de autorizare cu numele X-Api-Key. La rândul său, serverul Flask va accesa modulul de rețea neuronală, va calcula predictia și va aplica masca de segmentare pe imaginea primită. În final, clientul Ionic este deservit cu informațiile solicitate.



În următoarele paragrafe, vom explora în detaliu două endpoint-uri care facilitează comunicarea între aplicațiile noastre implementate în Spring Boot și Flask. Aceste endpoint-uri reprezintă puncte de legătură între cele două servicii, permitând un flux bidirectional de informații și contribuind la funcționalitatea de ansamblu a sistemului, cu precădere la trimiterea de imagini.

```

// ImageController.java
@PostMapping("/image")
public ResponseEntity<String> segmentImage(@RequestParam("image")
    ↳ MultipartFile multipartFile) throws IOException,
    ↳ InterruptedException {
    byte[] imageBytes = multipartFile.getBytes(); // convert the
    ↳ MultipartFile to byte array
    OverlayCategoryDTO overlayCategoryDTO =
    ↳ imageService.getSegmentedImage(imageBytes); // methods which
    ↳ handles the call to the flask server
    String base64 =
    ↳ imageService.convertImageToBase64(overlayCategoryDTO.getOverlayMap());
    ↳ // convert the image to the base 64 format
    JSONObject returnResponse = new JSONObject();
    returnResponse.put("overlay", base64);
    returnResponse.put("category",
    ↳ overlayCategoryDTO.getCategoryColors());
    return
    ↳ ResponseEntity.status(HttpStatus.OK).body(returnResponse.toString());
}

```

```

# app.py
@app.route('/image', methods=['POST'])
@require_api_key
def get_image_prediction():
    unsigned_bytes = extract_image_bytes(request) # extract the image
    ↳ bytes from the request

```

```
store_image(path_to_save_image, unsigned_bytes) # store the image on
→ the server to preserve the bytes number
image = Image.open('%s' % path_to_save_image).resize((256,256))
preds = generate_prediction(image) # call the prediction engine
image_with_transform = np.array(image)
masked_image, color_map =
→ service.get_overlay_with_map(image_with_transform, preds) # apply
→ the predicted output over the image
data_str_keys = {str(key): value for key, value in color_map.items()}
json_data = json.dumps(data_str_keys)
return jsonify(
    {"mask": masked_image,
     "color_map": json_data}), 200
```

Capitolul 5

Concluzii și perspective de viitor

5.1 Concluzie

Această lucrare și-a îndeplinit obiectivele prin implementarea și evaluarea a trei arhitecturi diferite de rețele neuronale pentru segmentarea imaginilor, precum și prin dezvoltarea unei aplicații software cu un design concentrat pe utilizator.

În primă fază, am implementat arhitectura UNet și am efectuat două experimente cu diferite funcții de loss: *CE Loss* și *IoU-CE Loss*. Deși *IoU-CE Loss* a prezentat valori mai mari, aceasta a oferit măști de segmentare mai fidele, demonstrând că interpretarea adecvată a rezultatelor merge dincolo de simpla comparare a cifrelor.

Ulterior, am extins arhitectura UNet prin integrarea unui encoder derivat din arhitectura ResNet-152, care a fost pre-încărcat cu ponderi obținute din antrenamentul pe setul de date ImageNet. Acest pas a demonstrat valoarea adăugată a transferului de învățare în îmbunătățirea performanței segmentării.

În etapa finală, am implementat un Vision Transformer, în mod specific un Masked Auto Encoder, îmbogățit cu patru straturi conlovuționale suplimentare pentru a crește dimensiunea rezultatelor transformerului. Această strategie a demonstrat că experimentarea și ajustarea arhitecturilor pot avea rezultate semnificative în contextul învățării automate, obținând cele mai bune performanțe în generarea de măști de segmentare. Punctul forte al acestui model constă în faptul că este mult mai "light-weight", caracteristică care ne-a permis să eficientizăm procesul de antrenament. În comparație cu experimentele anterioare, am reușit să dublăm numărul de epoci de antrenament, mărind în același timp de două ori cantitatea de date folosite pentru antrenament. Astfel, am demonstrat că un model mai eficient din punct de vedere al resurselor nu numai că poate obține rezultate superioare, dar poate și îmbunătăți procesul de antrenament, permitându-ne să explorăm mai profund și mai extensiv spațiul de învățare.

În concluzie, această lucrare a ilustrat cum combinarea abordărilor din învățarea

automată cu principiile solide de dezvoltare software poate duce la crearea de soluții performante și centrate pe utilizator în domeniul sănătății.

5.2 Scanarea codului de bare

Una dintre perspectivele cheie pentru dezvoltarea ulterioară a aplicației este integrarea funcționalității de scanare a codului de bare. Această caracteristică ar permite utilizatorilor să scaneze codurile de bare ale produselor alimentare și să obțină automat informații detaliate despre conținutul nutritiv al acestora. În plus, scanarea codului de bare ar putea oferi și alte informații importante, cum ar fi lista de ingrediente, alergeni, informații despre producător și data de expirare. Aceste informații ar permite utilizatorilor să ia decizii alimentare mai informate și să urmărească mai eficient dieta și obiectivele lor nutriționale.

Pentru implementarea scanării codului de bare, aplicația ar putea utiliza tehnologii de recunoaștere optică a caracterelor (OCR) și baze de date cu informații despre produse alimentare. Prin intermediul acestei funcționalități, utilizatorii ar putea beneficia de o experiență îmbunătățită în ceea ce privește monitorizarea și gestionarea dietei lor, aducând un plus de eficiență și precizie în procesul de înregistrare a alimentelor consumate.

5.3 Integrarea cu alte aplicații și dispozitive

O direcție promițătoare pentru dezvoltarea ulterioară a aplicației noastre este integrarea cu alte aplicații, de tipul (Samsung Health) și dispozitive, cum ar fi smartwatch-uri și smartband-uri. Acestea ar permite utilizatorilor să monitorizeze în timp real nivelul de activitate fizică, pulsul cardiac, caloriile arse și calitatea somnului. Prin conectarea și sincronizarea acestor date cu aplicația noastră, utilizatorii ar putea avea o imagine mai completă și mai precisă a nivelului lor de activitate fizică și a impactului acestora asupra consumului alimentar și a sănătății generale. Prin colaborarea cu alte aplicații și dispozitive, putem crea o platformă complexă și versatilă pentru îmbunătățirea sănătății și bunăstării generale a utilizatorilor noștri.

Bibliografie

- [1] E. K. V. I. I. A. Buslaev, A. Parinov and A. A. Kalinin. Albumentations: fast and flexible image augmentations. *ArXiv e-prints*, 2018.
- [2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [3] R. Descartes. *Meditations on First Philosophy*. Michael Soly, 1641.
- [4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.
- [5] G. Ferrara, J. Kim, S. Lin, J. Hua, E. Seto, et al. A focused review of smartphone diet-tracking apps: usability, functionality, coherence with behavior change theory, and comparative validity of nutrient intake and energy estimates. *JMIR mHealth and uHealth*, 7(5):e9232, 2019.
- [6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- [7] M. Grandini, E. Bagli, and G. Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.
- [8] M. Grinberg. *Flask web development: developing web applications with python*. "O'Reilly Media, Inc.", 2018.
- [9] S. S. Gropper and J. L. Smith. *Advanced nutrition and human metabolism*. Cengage Learning, 2012.
- [10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [11] D. O. Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

- [12] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.
- [13] P. M. Insel. *Nutrition*. Jones & Bartlett Publishers, 2014.
- [14] M. B. Jones, J. Bradley, and N. Sakimura. JSON Web Token (JWT). RFC 7519, May 2015. URL <https://www.rfc-editor.org/info/rfc7519>.
- [15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.
- [16] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.
- [17] M. H. Laitner, S. A. Minski, and M. G. Perri. The role of self-monitoring in the maintenance of weight loss success. *Eating behaviors*, 21:193–197, 2016.
- [18] M. Lynch, B. Sperry, and A. Bradley. Ionic framework, 2013. URL <https://ionicframework.com/>.
- [19] MacroTrends. World life expectancy, n.d. URL <https://www.macrotrends.net/countries/WLD/world/life-expectancy>.
- [20] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 52:99–115, 1990.
- [21] M. D. Mifflin, S. T. St Jeor, L. A. Hill, B. J. Scott, S. A. Daugherty, and Y. O. Koh. A new predictive equation for resting energy expenditure in healthy individuals. *The American journal of clinical nutrition*, 51(2):241–247, 1990.
- [22] M. N. A. R. K. R. A. V. W. Z. N. Shahid, M. Zelembaba. Client-server architecture, n.d. URL https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/ClientServer.pdf.
- [23] J. L. S. N. R. W. A. W. M. O. C. D. S. D. M. S. V. P. J. B. M. B. E. M. S. F. M. H. Phillip Webb, Dave Syer. Spring Boot Reference Documentation — docs.spring.io. <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. [Accessed 28-May-2023].
- [24] D. Rao, P. K, R. Singh, and V. J. Automated segmentation of the larynx on computed tomography images: a review. *Biomedical Engineering Letters*, 12:1–9, 03 2022. doi: 10.1007/s13534-022-00221-3.

- [25] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.
- [26] A. Rosebrock. <http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>. [Accessed 28-May-2023].
- [27] S. J. Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.
- [28] E. E. Spamer. Know thyself: responsible science and the lectotype of homo sapiens linnaeus, 1758. *Proceedings of the Academy of Natural Sciences of Philadelphia*, 149:109–114, 1999.
- [29] A. Tiwari and P. Balasundaram. Public health considerations regarding obesity. 2021.
- [30] E. N. Whitney and S. R. Rolfes. *Understanding nutrition*. Cengage Learning, 2015.
- [31] W. C. Willett and M. J. Stampfer. Rebuilding the food pyramid. *Scientific American*, 288(1):64–71, 2003. ISSN 00368733, 19467087. URL <http://www.jstor.org/stable/26060127>.
- [32] X. Wu, X. Fu, Y. Liu, E.-P. Lim, S. C. Hoi, and Q. Sun. A large-scale benchmark for food image segmentation. In *Proceedings of ACM international conference on Multimedia*, 2021.