UNIVERSITATEA „ALEXANDRU IOAN CUZA”

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENŢĂ

Coordonator ştiinţific

Conf. Dr. Anca Ignat

Absolvent

Năstasă-Baraș Luca

IAŞI

2024

UNIVERSITATEA „ALEXANDRU IOAN CUZA”

FACULTATEA DE INFORMATICĂ

SEGMENTAREA SEMANTICĂ A IMAGINILOR

Coordonator ştiinţific

Conf. Dr. Anca Ignat

Absolvent

Năstasă-Baraș Luca

IAŞI

2024

Anexa II

Avizat,

Îndrumător Lucrare de Licenţă/Diplomă/Disertaţie/Absolvire

Titlul, Numele şi prenumele Segmentarea semantică a imaginiilor, Năstasă Baraş Luca

Data 25.06.2024 Semnătura _____

DECLARAȚIE privind autenticitatea conţinutului lucrării de licenţă/diplomă/disertaţie/absolvire

Subsemnatul(a) Năstasă Baraș Luca_____ domiciliat(ă) în _Tecuci,județul Galați_, născut(ă) la data de _10.06.2002_, identificat(ă) prin CNP _5020610394421_, absolvent(ă) al(a) **Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de Informatică**, specializarea _Informatică_, promoţia __2021-2024__, declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă/diplomă/disertaţie/absolvire cu titlul: ____Segmentarea semantică a imaginilor____ elaborată sub îndrumarea dlui/dnei__Ignat Anca___, este autentică, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă/diplomă/ disertaţie/absolvire să fie verificată prin orice modalitate legală pentru confirmarea autenticității, consimțind inclusiv la introducerea conţinutului său într-o bază de date în acest scop. Declar că lucrarea de faţă are exact acelaşi conţinut cu lucrarea în format electronic pe care profesorul îndrumător a verificat-o prin intermediul software-ului de detectare a plagiatului.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice în vederea facilitării falsificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diplomă sau de disertaţie şi în acest sens declar pe proprie răspundere că lucrarea de faţă nu a fost copiată, ci reprezintă rodul cercetării pe care am întreprins-o.

Data __25.06.2024__ Semnătură student _____

_____

Anexa III

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul *„Segmentarea semantică a imaginiilor.*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data 25.06.2024*

Absolvent *Năstasă Baraș Luca*

_____

(semnătura în original)

_____

Cuprins

Introducere

Datorită dezvoltării tehnologiei hardware și a noilor metode de invățare automată din ultima vreme, vederea artificală a cunoscut o îmbunătățire majoră. Segmentarea semantică a imaginilor este un domeniu important în ramura vederii artificiale, deoarece are aplicații variate în domenii cum ar fi conducerea autonomă, analiza medicală, monitorizarea mediului și realitatea augmentată.

Segmentarea semantică reprezintă clasificarea pixelilor dintr-o imagine prestabilită, oferind rezultatul unei înțelegeri concise a scenei.

Metodele tradiționale de segmentare a imaginilor au prezentat rezultate pozitive pentru unele aplicații, insa acestea sunt constrânse de capacitatea de a înțelege contextul global al imaginii.

Rețelele neuronale convoluționale au realizat o performanță deosebita, producand rezultate foarte bune pentru diferite probleme de segmentare semantică. O bibliotecă flexibilă și complexă pentru dezvoltarea și antrenarea modelelor de rețele neuronale este PyTorch. Utilizarea acesteia a permis cercetătorilor să dezvolte eficient și într-un timp relativ scurt arhitecturi și tehnici diverse si complexe.

Prezenta lucrare de licență prezinta metode actualizate de segmentare semantică a imaginilor folosind rețele neuronale convoluționale. Principalele obiective sunt investigarea și prezentarea tehnicilor de bază și a arhitecturilor de rețele neuronale folosite pentru segmentarea semantică, dezvoltarea unei rețele neuronale convoluționale pentru segmentarea semantică utilizând biblioteca PyTorch, examinarea performanței modelului dezvoltat pe un set de date de bază și analiza rezultatelor obținute.

Lucrarea este constituită pe mai multe capitole, pentru a oferi informații cât mai concise fata de subiectul abordat, astfel:

- Capitolul 1: Noțiuni Generale - Acest capitol oferă o prezentare a conceptelor teoretice esențiale pentru înțelegerea segmentării semantice și a rețelelor neuronale.
- Capitolul 2: Studiu al Metodelor de Segmentare Semantică - Sunt prezentate metodele clasice și cele bazate pe învățare profundă, subliniind avantajele și limitările fiecăreia.
- Capitolul 3: Metodologie - Se detaliază procesul de colectare și preprocesare a datelor, arhitectura rețelei neuronale utilizate și metodologia de antrenare și evaluare a modelului.
- Capitolul 4: Implementare Practică - Include detalii despre configurarea mediului de lucru, codul sursă, exemple de segmentare și discuții asupra rezultatelor obținute.
- Capitolul 5: Discuții și Concluzii - Rezumă contribuțiile principale ale lucrării, concluziile finale și propune direcții pentru lucrări viitoare.
- Bibliografie

Capitolul 1 – Noțiuni Generale

1.1 Segmentarea semantică a imaginilor

1.1.1 Definiție și importanță

Segmentarea semantică a imaginilor reprezintă tehnica de bază în domeniul vedere artificiala, care se ocupă cu organizarea fiecărui pixel dintr-o poză într-o categorie desemnată. Față de clasificarea tradițională a imaginilor, care acordă o etichetă, segmentarea semantică distribuie o înțelegere mult mai diversificată a scenei. Tehnica oferă identificarea și localizarea exactă a obiectelor și a contextului lor conform imaginii.[1]

Valoarea segmentării semantice este realizată de diferitele sale aplicații practice:

- Conducerea autonomă: Identificarea drumurilor, vehiculelor, pietonilor și a altor obiecte din mediul înconjurător pentru navigare și evitarea coliziunilor.
- Analiza medicală: Segmentarea imaginilor medicale pentru identificarea și clasificarea diferitelor structuri anatomice și anomalii, cum ar fi tumori sau leziuni.
- Monitorizarea mediului: Analiza imaginilor satelitare pentru identificarea și monitorizarea schimbărilor în peisaj, cum ar fi defrișările sau urbanizarea.
- Realitatea augmentată: Integrarea obiectelor virtuale în medii reale prin segmentarea precisă a elementelor scenei.

1.1.2 Aplicații practice

Segmentarea semantică are aplicații practice sporite în diferite domenii:

- Medicină: Este folosită pentru segmentarea imaginilor medicale (CT, RMN) pentru detectarea automată a tumorilor și a leziunilor.
- Automotive: Este utilizată în sistemele de asistență pentru conducatori și în vehiculele autonome pentru recunoașterea și clasificarea obiectelor din trafic.
- Agricultură: Segmentarea imaginilor aeriene pentru supravegherea culturilor și identificarea problemelor legate de plante.
- Robotica: Oferă roboților posibilitatea naviga și de a interacționa cu mediul înconjurător prin clasificarea și recunoașterea obiectelor.

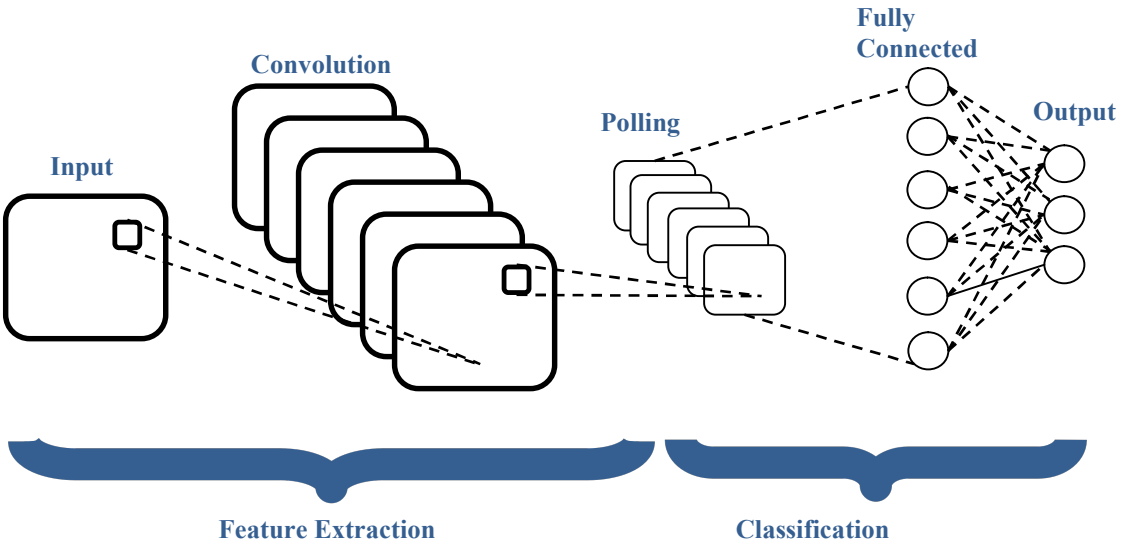
[1] Principles of Digital Image Processing - Advanced Methods Willhelm Burger & Mark J. Burge

1.2 Rețele neuronale

1.2.1 Concepte de rețele neuronale

Rețelele neuronale artificiale reprezintă modele computaționale inspirate din structura și funcționarea creierului uman. Ele sunt alcătuite din neuroni artificiali, dispuși în straturi, care preiau datele de intrare pentru a dezvolta o ieșire in baza schimburilor de informații. Rețelele neuronale convoluționale (CNN) reprezintă o clasă specială de rețele neuronale, concepute pentru a procesa date care au o topologie de grilă, cum ar fi imaginile.[2]

Rețelele neuronale convoluționale folosesc straturi convoluționale pentru a prelua caracteristicile locale din imagini și straturi de pooling pentru a diminua dimensiunea reprezentărilor, folosind în același timp informația necesară. Această arhitectură permite rețelelor neuronale convoluționale să învețe reprezentările descendente ale datelor, de nivel inferior, cum ar fi margini și texturi până la caracteristici superioare, cum ar fi obiecte și scene.[3]



[2] Principles of Digital Image Processing - Core Algorithms Willhelm Burger & Mark J. Burge
[3] Computer Vision Using Deep Learning - Vaibhav Verdhan

1.2.2 Arhitecturi folosite în Segmentarea Semantică

O diversitate de arhitecturi de rețele neuronale convenționale sunt folosite pentru segmentarea semantică a imaginilor, printe cele mai notabile le putem clasifica pe următoarele:

- Fully Convolutional Networks (FCN): Această arhitectură dezvoltă o rețea de clasificare a imaginilor într-o rețea de segmentare prin schimbarea straturilor dense cu straturi convoluționale.
- U-Net: Dezvoltată inițial pentru segmentarea imaginilor biomedicale, U-Net folosește o arhitectură în formă de U, care unește straturile de convoluție și de deconvoluție pentru a realiza segmentări precise.
- SegNet: O rețea de segmentare care este folosită pe o arhitectură de encoder-decoder, care folosește straturi de pooling și unpooling pentru a realiza segmentările de rezoluție superioară.
- DeepLab: Folosește straturi convoluționale dilatate și o metodă de pooling piramidală pentru a capta contextul la diferite scale și a dezvolta segmentarea obiectelor de dimensiuni variate.

1.3 PyTorch

1.3.1 Prezentarea PyTorch

PyTorch este o bibliotecă open-source pentru învățarea automată, dezvoltată de Facebook's AI Research lab (FAIR). Aceasta este renumită pentru flexibilitatea și utilizarea eficientă, oferind dezvoltarea într-un timp scurt a modelelor de învățare profundă. Biblioteca PyTorch furnizează suport amplu pentru operațiuni tensoriale, producția dinamică a grafurilor computaționale și integrarea cu alte biblioteci populare.[15]

Caracteristicile principale ale PyTorch sunt:

- Calculul tensorial: Operații performante pe matrici multidimensionale, similare cu NumPy, dar cu suport pentru GPU.
- Grafuri computaționale dinamice: Permit schimbarea grafului de calcul în timpul execuției, contribuind la eficientizarea calculului si verificarea mai ușoara a erorilor.
- Suport extensiv pentru rețele neuronale: Modul amplu și intuitiv de a realiza și antrena modelele de rețele neuronale prin folosirea modulului torch.nn.
- Interfață prietenoasă: Codul PyTorch este intuitiv de scris și de citit, ceea ce îl face util pentru cercetare și dezvoltare.

[15] PyTorch - https://pytorch.org

1.3.2 Avantaje și comparație cu alte biblioteci

Fiind pus comparativ cu alte biblioteci de învățare avansate, precum TensorFlow, PyTorch asigură următoarele avantaje:

- Flexibilitatea și simplitatea: Grafurile computaționale dinamice și API-ul receptiv fac codul PyTorch sa fie corectat usor.
- Performanța: PyTorch asigură performanțe superioare și un grad ridicat de eficiență în folosirea memoriei, obținând un suport diversificat pentru calcul paralel pe GPU.
- Comunitatea și suport-ul PyTorch: Acesta beneficiază de o comunitate care constant se dezvoltă constant și este activă, asigurând numeroase resurse, cât și exemple care deserversc la învățarea și folosirea bilbiotecii.

Astfel, putem spune că biblioteca PyTorch reprezintă un instrument de baza in dezvoltarea și implementarea modelelor de învățare profundă, recomandata fiind pentru cercetarea în vederea artificiala.[15]

[15] PyTorch - https://pytorch.org

Capitolul 2 – STUDIU AL METODELOR DE SEGMENTARE SEMANTICĂ

2.1 Metode clasice

2.1.1 Metode bazate pe regiuni și contururi

Metodele de bază de segmentare ale imaginilor se folosesc în mod special pe tehnici de prelucrare a imaginilor, folosind caracteristici locale ale pixelor pentru a diferenția regiunile ale unor imagini.Printre aceste metode enumeram:

- Algoritmi de dezvoltare a regiunilor: Aceste metode încep de la un set de pixeli și dezvoltă regiunea, introducând pixeli adiționali care realizează criteriile menționate de similaritate. Un exemplu ar putea fi algoritmul de dezvoltare a regiunilor, care se bazează pe similaritatea intensităților pixelilor.
- Segmentarea bazată pe contururi: Această metodă determină contururile obiectelor prin depistarea marginilor în imagine. Algoritmi cum ar fi detectorul de margini Canny sau transformata Hough sunt folosiți pentru a accentua contururile.
- Algoritmii de clustering: Algoritmi cum ar fi K-means și Gaussian Mixture Models sunt utilizați pentru a clasifica pixelii în clase conform caracteristicilor lor, precum culoarea sau textura.[2]

2.1.2 Algoritmi de clasificare a pixelilor

Metodele de încadrare a pixelilor utilizează clasificatori pentru a marca fiecare pixel al unei imagini.Conform caracteristicilor sale, enumerăm următorii algoritmi:

- Clasificatorii Bayesieni: Folosiți pentru a calcula acuratețea ca un pixel să aparțină unei anumite clase, pe baza distribuției condiționale a caracteristicilor.
- Clasificatori pe baza rețelelor neuronale: Anterior erei învățării profunde, rețelele neuronale simple erau folosite pentru a cataloga pixelii individuali pe baza caracteristicilor extrase.
- Support Vector Machines: Utilizat pentru clasificarea pixelilor pe baza unui set de particularități, căutând să maximizeze marginea dintre clasele distincte.

[2] Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution & Fully Connected CRFs - Chen, Liang-Chieh

2.2 Metode bazate pe învățare profundă

Învățarea profundă a dezvoltat segmentarea semantică a imaginilor, prin modele complexe care pot invata caracteristici esențiale din date. Dintre cele mai importante arhitecturi, cunoaștem prezentam metode:

2.2.1 Fully Convolutional Networks

Fully Convolutional Networks (FCN) reprezintă primele rețele neuronale convoluționale elaborate special pentru segmentarea semantică a imaginilor. Diferența de bază față de rețelele de clasificare a imaginilor este faptul că Fully Convoluțional Networks, îndepărtează straturile dense și folosește straturi convoluționale pentru a crea o hartă de segmentare de aceeași mărime ca imaginea inițială.

- Arhitectura: FCN schimbă straturile dense în straturi convoluționale și utilizează straturi deconvoluționale pentru a recrea rezoluția inițială a imaginii.
- Avantajele: Oferă antrenarea rețelei pe imagini de orice dimensiuni și realizează rezultate de segmentare exacte și detaliate.[2]

2.2.2 U-Net

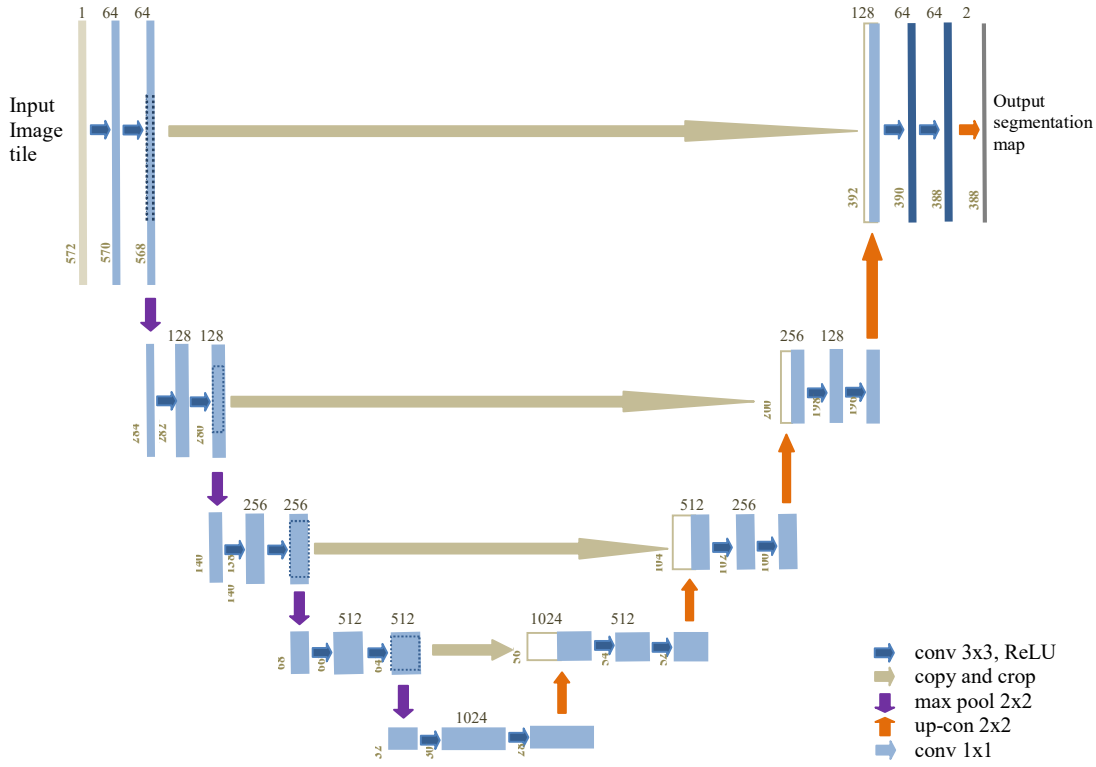
U-Net reprezintă o arhitectură dezvoltată inițial pentru segmentarea imaginilor biomedicale, dar care a devenit este mult mai eficientă pentru o gamă largă de aplicații.

Arhitectura: U-Net are o formă de U, constând dintr-un encoder și un decoder. Encoder-ul extrage caracteristici la diferite scale, iar decoder-ul reconstruieste harta de segmentare folosind straturi de upsampling și conexiuni de tip skip.

Avantajele: Combinația de straturi convoluționale și de upsampling permite segmentarea precisă a detaliilor fine din imagine.[20]

[2] Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution & Fully Connected CRFs - Chen, Liang-Chieh

[20] Towardsdatascience - Cook your First U-Net in PyTorch - Mostafa Wael



2.2.3 SegNet

SegNet reprezintă o arhitectură ce se folosește de un encoder-decoder, identic cu cel al U-Net, dar cu diferențe în modul de reconstrucție a imaginii.[4]

- Arhitectura: Encoder-ul este reprezentat de straturi convoluționale și de pooling, iar decoder-ul folosește straturi de upsampling care realizează indiciile de pooling din encoder pentru a dezvolta reconstrucția.
- Avantajele: SegNet este superior din punct de vedere al memoriei și oferă o reconstrucție precisă a detaliilor de segmentare.

2.2.4 Deeplab

DeepLab reprezintă o familie de arhitecturi ce deservesc la realizarea segmentării semantice prin utilizarea de tehnici avansate.[3]

- Arhitectura: DeepLab utilizează straturi convoluționale dilatate (atrous convolution) pentru a mări câmpul receptiv fără a diminua rezoluția. De asemenea, utilizează un modul de pooling piramidal (PSP) pentru a captura contextul la diferite scale.

[4] OpenCV 4 Computer Vision Application Programming Cookbook - David Millan Escriva & Robert Laganiere

[3] Computer Vision Using Deep Learning - Vaibhav Verdhan

- Avantajele: Permite segmentarea obiectelor de dimensiuni variate și îmbunătățește precizia în detectarea contururilor.[4]

2.3 Compararea metodelor

Metodele de segmentare semantică bazate pe învățarea profundă au dovedit performanțe ridicate față de metodele clasice, datorită capacității lor de a învăța reprezentări complexe și de capta contextul global al imaginii. Astfel, putem spune că folosirea arhitecturii potrivite pentru segmentarea semantică depinde în special de cerințele aplicației, de resursele disponibile și complexitatea datelor. Tabelul de mai jos realizează o comparație a arhitecturilor prezentate și discutate mai sus.[5]

Metodă	Avantaje	Dezavantaje
FCN	Simplitate, flexibilitate	Rezoluție redusă la ieșire
U-Net	Segmentare precisă, conexiuni de skip	Necesită resurse computaționale mari
SegNet	Eficiență de memorie, reconstrucție detaliată	Complexitate în implementare
DeepLab	Câmp receptiv mare, context multiscală	Complexitate ridicată, consum mare de resurse

2.4 Arhitectura U-Net

După cum putem observa mai sus, arhitectura de tip U-Net-ul este extrem de utilă în segmentarea imaginilor întrucât, în mare parte este atent la detalii.U-Net++ este o extensie a arhitecturii folosită pentru păstrarea detaliilor fine și a diferitelor trăsături complexe. Principalele îmbunătățiri sunt aduse de caracteristicile aduse în plus, precum conexiuni de tip skip îmbunătățite, reprezentate de utilizarea unor blocuri dense de convoluție pe aceste conexiuni.Astfel, U-Net++ reduce decalajul semantic dintre hărțile de caracteristici ale encoder-ului și decoder-ului, îmbunătățind calitatea informației propagate si ajutând la reconstrucția mai precisă a detaliilor fine ale obiectelor segmentate.[20]

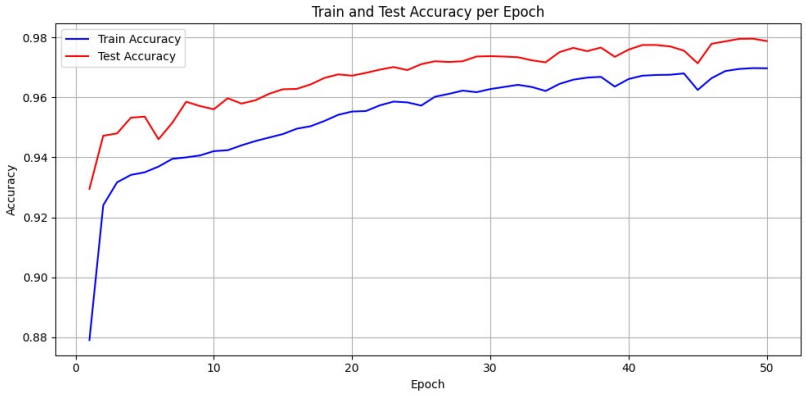
O caracteristică importantă este supervizarea profundă, ce permite rețelei să fie antrenată cât mai eficient, realizand la o convergență mai rapidă și la o generalizare eficientă.

De asemenea altă caracteristică este flexibilitatea timpului de interferență, unde U-Net++

[4] Deep Learning - Ian Goodfellow
[5] Practical Deep Learning - Ron Kneusel

[20] UNet++: A Nested U-Net Architecture for Medical Image Segmentation - Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang

permite ajustarea flexibilă a complexității rețelei în timpul interferenței, ceea ce poate optimiza balanța dintre acuratețe și viteza de procesare, adaptându-se astfel la diverse necesități din domeniul medical.



2.4.1 Conexiuni Skip in Unet++

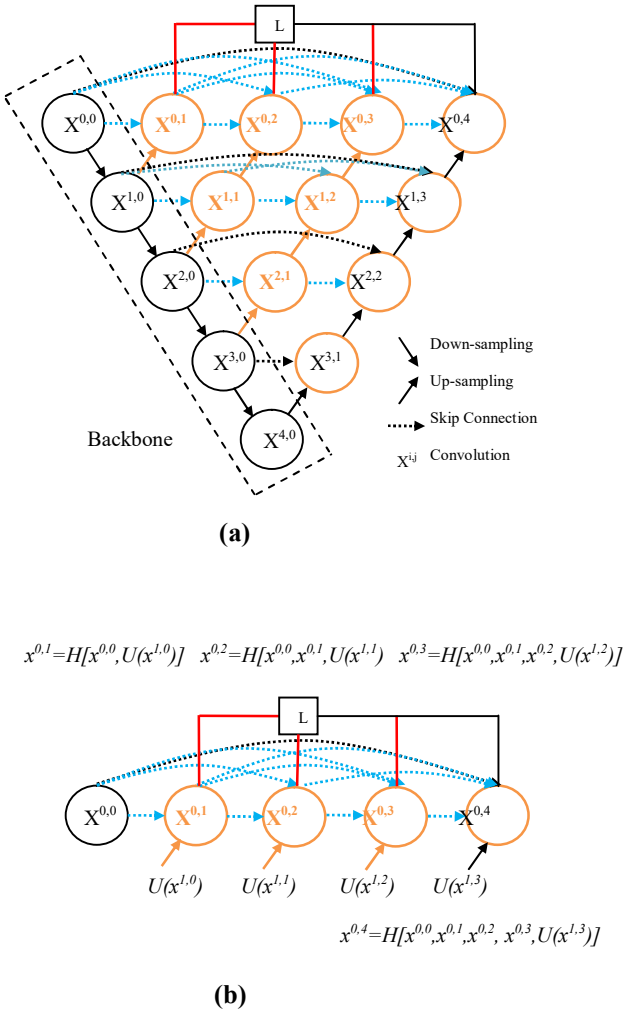
Conexiunile de tip Skip în U-Net++ nu sunt implementate ușor ca fiind legături directe între straturile corespondente ale encoder-ului și decoder-ului, totuși ele sunt configurate ca fiind o serie de blocuri dense de convoluție, care realizează și rafinează hărțile de caracteristici înainte de a putea fi propagate de la encoder la decoder. Această abordare reduce decalajului semantic și îmbunătățește totodată coerența informațiilor transmise. Pentru a reflecta procesul de agregare și rafinare a caracteristicilor pe parcursul rețelei metoda este formulată matematic astfel: [20]

$$x_{ij}=\begin{cases} H\left(D\left(x_{-1,j}\right)\right), \text{pentru } j=0 \\ H\left(\left[x_{i,k}\right]_{k=0}^{j-1}, U\left(x_{i+1,j-1}\right)\right), \text{pentru } j>0 \end{cases}$$

Prin utilizarea acestor conexiuni skip complexe se obține:

- Reducerea decalajului semantic - fiecare strat de convoluție în blocurile dense ajută la alinierea semantică între hărțile de caracteristici ale encoderului și cele ale decoderului, facilitând o reconstrucție mai precisă și mai detaliată a imaginii țintă.
- Îmbunătățirea fluxului de gradient - conexiunile dense asigură un flux robust de gradient pe tot parcursul rețelei, ceea ce este important pentru antrenarea eficientă a rețelelor profunde.
- Flexibilitate în fuzionarea caracteristicilor: - abilitatea de a combina caracteristici de la multiple niveluri și scale diferite permite modelului să capteze atât detalii fine, cât și contextul mai larg necesar pentru segmentarea precisă.

[20] UNet++: A Nested U-Net Architecture for Medical Image Segmentation - Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang



2.4.2 Supervizare Profunda

Putem discuta despre supervizarea profundă a U-Net++ ca fiind realizată prin adăugarea unui strat de convoluție 1x1, cu funcție de activare Sigmoid la ieșirile din nodurile $X^{\{0,1\}}$, $X^{\{0,2\}}$, $X^{\{0,3\}}$ și $X^{\{0,4\}}$.

Astfel, se permite rețelei să genereze rezultate de segmentare la multiple niveluri semantice, fiecare având capacitatea de a contribui independent la funcția de pierdere globală, prin pierderea principală, determinata de output-ul final al rețelei și măștile țintă și pierderile intermediare, obținute pentru fiecare dintre output-urile intermediare redimensionate la dimensiunea corespunzătoare a măștilor țintă.[4]

Formula de calcul este următoarea:

$$L = BCE(Y, P) + \lambda * Dice(Y, P)$$

unde:

- Y sunt măștile binare ale imaginii;

[4] Deep Learning - Ian Goodfellow

- P reprezintă predicțiile modelului;
- λ reprezintă un coeficient de ponderare care echilibrează contribuția componentelor pierderi;

- Funcția BCE este reprezentată de formula:

$$BCE(Y,P) = -\frac{1}{N}\sum_{i=1}^{N} |y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)]$$

- Funcția Dice este reprezentată de formula:[20]

$$Dice(Y,P) = 1 - \frac{2 \cdot \sum_{i=1}^{N} y_i p_i}{\sum_{i=1}^{N} y_i^2 + \sum_{i=1}^{N} p_i^2}$$

Astfel, U-Net++ va putea opera în două moduri denumite modul ensemble și modul pruned. Cu primul mod, rezultatele segmentării de la toate ramurile de segmentare vor fi adunate și mediate, maximizând acuratețea și folosind informații combinate de la toate nivelurile de detalii. Celălalt modul reprezintă rezultatul segmentării selectat de la o singura ramură, a cărei alegere reprezintă amploarea tăierii modelului și gestionarea vitezei.

[20] UNet++: A Nested U-Net Architecture for Medical Image Segmentation - Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang

Capitolul 3 – METODOLOGIE

3.1 Colectarea și pregătirea datelor

3.1.1 Surse de date și seturi de date utilizate

Pentru a pregătirea și analiza rețelei neuronale pentru segmentarea semantică a imaginilor se pot folosi urmatoarele seturi de date de bază:

- Pascal VOC: Un set de date bine-cunoscut în domeniul vederii artificiale, care are la bază imagini etichetate cu 20 de clase de obiecte, de asemenea și o clasă de fundal. Setul de date integrează imagini variate, captate în contexte distincte și condiții de iluminare.
- COCO (Common Objects in Context): Un set de date amplu, care alcătuiește imagini etichetate cu 80 de clase de obiecte. COCO este folosit pentru o diversitate de sarcini de vedere artificială, totodată și segmentarea semantică.
- HAM10000 (Human Against Machine with 10000 training images): Reprezintă un set de date important în cercetarea medicală, în special în dermatologie. Acesta are la bază 10.000 de imagini dermatoscopice utilizate pentru antrenarea algoritmilor de inteligență artificială, cu scopul recunoașterii și clasificării leziunilor cutanate. Setul de date este variat și cuprinde imagini de melanom, nevi și alte tipuri de leziuni ale pielii, marcat pentru a facilita segmentarea exactă și diagnosticarea. Imaginile reies din diverse surse și sunt colectate pentru a furniza o reprezentare complexă a variațiilor clinice întâlnite în practica reală.

3.1.2 Preprocesarea datelor

Preprocesarea datelor reprezintă un pas vital în garanția calității și consistenței datelor de interare pentru rețeaua neuronală, incluzând următorii pași:

- Redimensionarea: Imaginile sunt modificate la o dimensiune fixă pentru a oferi consistența în timpul antrenării. De exemplu, noi am redimensionat in aplicație la dimensiunea de 128x128 pixeli.
- Normalizarea: Valorile pixelilor sunt scalate într-un interval anumit, de obicei între 0 și 1, sau normalizate utilizând media și deviația standard a canalelor de culoare ale setului de date.
- Augmentarea: Tehnicile de augmentare a datelor, precum rotația, translația, scalarea, și modificarea contrastului, sunt folosite pentru a mări diversitatea setului de date și a preveni supraantrenarea.

3.2 Arhitectura rețelei neuronale

Pentru segmentarea semantică a imaginilor, am utilizat arhitectura U-Net deoarece nivelul de performanță a acesteia este crescut și capacitatea sa de a realiza segmentările este precisă.

3.2.1 Descrierea arhitecturii alese

După cum am spus mai sus, am utilizat arhitectura U-Net deoarece performanțele acesteia s-au dovedit a fi eficiente, fiind recunoscută pentru capacitatea de a trata problemele de segmentare biomedicală, deși a fost extinsă și pe alte domenii de segmentare a imaginilor.

U-Net este realizată din două componente de bază:

- Encoder (contracting path): Această parte a rețelei funcționează identic cu un model de convoluție standard, unde imaginea este transpusă printr-o serie de straturi de convoluție și pooling, diminuând dimensiunea spațială dar sporind numărul de canale. Odată cu fiecare pas parcurs, se aplică un nucleu pentru a extrage caracteristicile, apoi de un ReLU și o operație de max-pooling pentru a diminiua dimensiunea.
- Decoder (expanding path): În partea de upsampling, dimensiunea spațială este mărită în timp prin operațiuni numite transpose convolution sau upsampling, astfel reducându-se numărul de canale. Odată cu utilizarea decoder-ului, sunt utilizate conexiunile de tip skip care transmit caracteristicile de la straturile obținute din encoder, alcătuind informațiile de la rezoluții distincte pentru a realiza o segmentare precisă.
- Conexiuni de tip skip: Aceste conexiuni joacă un rol imporant datorită combinării caracteristicilor obtinute la diferite niveluri de rezoluție, oferind rețelei oportunitatea de a recupera detaliile spațiale lipsă din timpul pooling-ului. Conexiunile de tip skip sunt alăturate cu straturile de upsampling, dezvoltând semnificativ performanța de segmentare.

3.2.2 Despre implementare în PyTorch

Pentru implementarea U-Net în PyTorch, am folosit următoarele metode:

- Definirea modelului: Am definit straturile convoluționale și de pooling în encoder și straturile de upsampling și convoluționale în decoder. Encoder-ul rezultă în straturi convoluționale urmate de activări ReLU și operații de pooling, care reduc dimensiunea imaginii și extrag caracteristicile relevante. Decoder-ul utilizează straturi de upsampling urmate de convoluții pentru a reconstrui imaginea segmentată la dimensiunea originală, combinând informațiile din encoder prin conexiuni de tip skip.
- Funcția de pierdere: Am folosit funcția de pierdere Cross-Entropy Loss, potrivita pentru clasificarea multi-clasă. Aceasta măsoară diferența între distribuțiile de probabilitate prezise de model și etichetele reale, minimizând erorile. Este utilizata impreuna cu funcția de activare softmax la ultimul strat, oferind modelul output-ului ca fiind o distribuție de probabilitate
- Optimizator: Am folosit Adam, care ajută la ajustarea dinamică a ratei de învățare pentru fiecare parametru. Adam combină avantajele AdaGrad și RMSProp, crescând ratele de învățare pentru fiecare parametru în parte. Astfel se permite o optimizare mult mai eficientă și o convergență mai rapidă. Folosind momentele de ordinul întâi și doi, Adam realizează procesul de antrenare, evitând oscilațiile bruște în actualizarea parametrilor.
- Antrenarea modelului: Antrenarea modelului s-a concretizat pe un set de date dat, folosind un DataLoader pentru a obține mini-batch-uri de date. Procesul implică trecerea datelor prin rețea, calcularea pierderii utilizând funcția de pierdere, realizarea backpropagation-ului pentru a obține gradientele și actualizarea parametrilor modelului utilizând optimizer-ul Adam. Antrenarea a fost făcută pe multiple epoci, iar performanța modelului a fost supravegheată folosind un set de validare pentru a ne asigura că modelul nu supraînvață și generalizează pe noi date..

3.3 Antrenarea rețelei

3.3.1 Configurații și hiperparametri

Pentru a realiza antrenarea rețelei, am configurat diferiți hiperparametrii, precum rata de învățarea, dimensiunea lotului și numărul de epoci folosite. Hiperparametrii utilizați în antrenarea modelului cuprind rata de învățare ca fiind de 0.001, batch size-ul de 16 imagini, pentru a avea un echilibru între folosirea memoriei și stabilizarea antrenării, numărul de epoci de 50, pentru a oferi rețelei capacitatea de a învăța caracteristicile esențiale, funcția de pierdere cross-entropy loss și optimizatorul Adam.

Antrenarea rețelei, s-a realizat prin urmatorii pași: pasul forward, care reprezintă trecerea datelor de intrare prin rețeaua neuronală pentru a realiza predicțiile si calculul pierderii, care reprezintă distingerea predicțiilor cu marcările sau etichetele reale pentru a rezolva funcția de pierdere; pasul backward, pentru propagarea erorii prin rețea înapoi cu scopul de a obtine gradientele. Ultimul pas, este reprezentat de actualizarea parametrilor, ce ajustează ponderea rețelei minimizând funcția de pierdere.[14]

3.3.2 Optimizarea și funcțiile de pierdere

După cum am menționat mai sus, m-am folosit de algoritmul Adam pentru optimizare, iar pentru funcția de pierdere, am utilizat pasul cross-entropy loss. Algoritmul Adam este unul dintre cei mai cunoscuți algoritmi de optimizare pentru rețelele neuronale. Motivele principale pentru care am ales acest algoritm sunt adaptarea ratei de învățare, prin care algoritmul combină avantajele AdaGrad și RMSProp, mărind ratele de învățare pentru fiecare parametru în parte și permițând o optimizare superioară și o convergență eficientă, păstrarea datelor prin care sunt folosite momentele de ordin primar și de ordin secundar, care ajută la stabilizarea procesului de antrenare.[22]

Formulele utilizate în algoritmul Adam sunt următoarele:

$$m_t = \beta_1 \, m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 \, v_{t-1} + (1 - \beta_2) g_t^2$$

$$\widehat{m}_t = \frac{m_2}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{\hat{v}_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

unde:

- g_t reprezintă gradientul la pasul t,
- m_t și v_t sunt estimările momentelor de ordinul întâi și doi,

[14] TensorFlow & PyTorch Documentation

[22] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." MICCAI, 2015.

- β_1 și β_2 sunt hiperparametri care controlează ratele de decădere ale momentelor (de obicei setate la 0.9 și 0.999),
- $\hat{m}_t$ și $\hat{v}_t$sunt momentele corectate pentru bias,
- η este rata de învățare,
- ε este un termen de regularizare pentru a preveni divizarea cu zero (de obicei setat la 1e-8),
- θ_t reprezintă parametrii modelului la pasul t.

Funcția de pierdere Cross-Entropy Loss este folosită în problemele de clasificare, a măsura diferența dintre distribuțiile de probabilitate prezise de model și etichetele reale. Aceasta este redată prin formula:

$$L = -\sum_{i=1}^{N} y_i \log(p_i)$$

unde:

- N este numărul de exemple din setul de date,
- y_i este eticheta reală pentru exemplul i (1 pentru clasa corectă și 0 pentru celelalte clase),
- p_i este probabilitatea prezisă de model pentru eticheta corectă.

Funcția Cross-Entropy Loss este folosită special folosită pentru sancționarea mai strictă a erorilor mari, oferindu-i modelului posibilitatea de a învăța într-un tip cât mai scurt și eficient. Totodată funcția este des utilizată împreună cu funcția de activare softmax la ultimul strat, garantând că output-ul modelului reprezintă o distribuție de probabilitate adecvată.[7]

3.4 Evaluarea performanței

Performanța modelului a fost evaluată folosind mai multe metrici, printre care precizia, desemnată ca fiind proporția pixelilor corect clasificați, IoU sau Intersection over Union, cunoscut ca și raportul dintre intersecția și uniunea seturilor de pixeli prezișI și cei reali, obținut pentru fiecare clasă și nu în ultimul rând, Dice Coefficient, identic cu IoU, care măsoară suprapunerea dintre seturile de pixeli estimați și cei reali.

3.4.1 Metode de evaluare și metrici utilizate

Evaluarea performanței unui model de segmentare semantică este necesară pentru a realiza cât de

[7] Deep Learning - Ian Goodfellow

bine poate identifica și segmenta în mod corect obiectele din imagini. Precizia fiind una dintre cele mai simple și folosite metrici de evaluare, aceasta măsoară proporția pixelilor clasificați amplu din totalul pixelilor, calculându-se după formula:

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

unde:

- TP (True Positives) este numărul de pixeli corect clasificați ca aparținând unei clase;
- TN (True Negatives) este numărul de pixeli corect clasificați ca neapărținând unei clase;
- FP (False Positives) este numărul de pixeli clasificați greșit ca aparținând unei clase;
- FN (False Negatives) este numărul de pixeli clasificați greșit ca neapărținând unei clase;
- Deși precizia este utilă, poate fi înșelătoare în cazul dataseturilor dezechilibrate, unde numărul de pixeli dintr-o clasă este mult mai mare decât din alta;

Intersection over Union (IoU), numită de alții și ca Jaccard Index, reprezintă o formă standard metrică pentru evaluarea segmentării semantice, măsurând suprapunerea între segmentarea estimată și segmentarea reală, formula acesteia fiind:

$$IoU = \frac{A \cap B}{A \cup B}$$

unde:

- A este setul de pixeli prezis ca aparținând unei clase;
- B este setul de pixeli etichetat ca aparținând aceleași clase;
- ∩ reprezintă intersectia dintre seturile de pixeli;
- ∪ reprezintă reuniunea seturilor de pixeli;

Daca acesta indică seminficativ(apropiat de valoarea 1) atunci s-a obținut o performanță mai bună.

Coeficientul Dice, știut și sub numele de F1 Score pentru segmentare, este o altă metrică utilizată pentru evaluarea segmentării. Este identic cu IoU, înă accentuează în mode deosebit suprapunerea dintre segmentări, formula acestuia fiind următoarea:[13]

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

unde:

- A și B sunt aceleași seturi de pixeli ca la IoU;
- Coeficientul Dice variază între 0 și 1, unde 1 indică o suprapunere perfectă. Este deosebit de util pentru seturi de date dezechilibrate, deoarece tratează mai echitabil clasele minore;
- Precizia și Revocarea sunt metrici esențiale în evaluarea performanței clasificatorilor,

[13] Fast.ai Practical Deep Learning for Coders

inclusiv în segmentarea semantică;

Precizia (Precision): măsoară acuratețea predicțiilor pozitive ale modelului.

$$Precizie = \frac{TP}{TP+F}$$

Revocarea (Recall): măsoară capacitatea modelului de a identifica toate instanțele pozitive.

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

Matricea de Confuzie oferă o vizualizare detaliată a performanței modelului prin afișarea valorilor TP, TN, FP și FN pentru fiecare clasă. Aceasta ajută la identificarea claselor pentru care modelul performează bine sau slab, oferind o imagine clară asupra modului în care predicțiile sunt distribuite.

3.4.2 Rezultate obținute și interpretate

O precizie globală ridicată arată faptul că în marea parte din pixelii unor imagini au fost clasificați corect, deși această metrică poate fi greșită dacă setul de date este echilibrat. Un exemplu ar fi, dacă un număr mare de pixeli sunt reprezentanți ai unei singure clase dominante atunci modelul are posibilitatea de a realiza o precizie ridicată prin simpla clasificare a numărului mare de pixeli în acea clasă, fără a segmenta în mod potrivit obiectele mai rare.[6]

Clasa 1: IoU = Y1%
Clasa 2: IoU = Y2%
...
Clasa N: IoU = YN%

Valorile IoU diferă între clase, presupunând că modelul performează, ceea ce recomandă că modelul performează mai eficient pentru anumite clase decât pentru altele. Clasele cu IoU superior sunt segmentate mai precis, în timp ce clasele cu IoU mai scăzut dezvăluie dificultăți în segmentarea corectă a acestor obiecte. Posibile cauze pentru aceste variații reprezintă complexitatea vizuală a claselor, dezechilibrul în numărul de instanțe și similitudinea vizuală dintre diferite clase.[13]

Clasa 1: Dice = Z1%

[6] Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution & Fully Connected CRFs - Chen, Liang-Chieh
[13] Fast.ai Practical Deep Learning for Coders

Clasa 2: Dice = Z2%

...

Clasa N: Dice = ZN%

Coeficientul Dice furnizează o măsură a suprapunerii între segmentările presupuse și cele reale. Clasele cu valori ridicate ale coeficientului Dice au o suprapunere eficientă, oferind o segmentare eficientă. Clasele cu valori mici ale coeficientului Dice pot solicita ajustări în model sau tehnici suplimentare de preprocesare a datelor pentru a dezvolta performanța.

Clasa 1: Precizie = P1%, Revocare = R1%

Clasa 2: Precizie = P2%, Revocare = R2%

...

Clasa N: Precizie = PN%, Revocare = RN%

Examinând valorile preciziei și revocării, putem observa că dacă modelul nostru obișnuiește să producă positive false sau negative, clasele cu precizie superioară și revocare mică denotă faptul că modelul este conservator și distribuie corect majoritatea obiectelor detectate, însă ignoră multe instanțe. Totodată, clasele cu revocare mărită și precizie mică relevă o tendință de a include mai multe suprasegmentări.

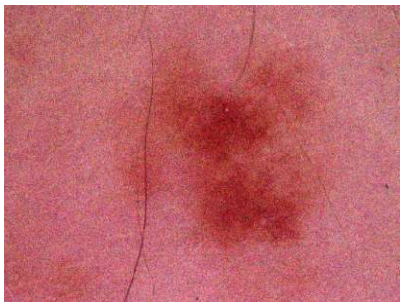
Clasa adevărată/clasa prezisă	Clasa 1	Clasa 2	Clasa 3
Clasa 1	90	5	5
Clasa 2	10	80	10
Clasa 3	15	5	80

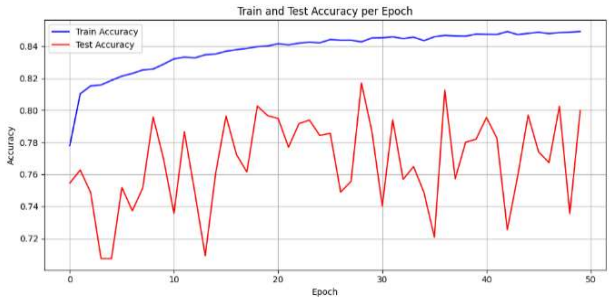
Matricea de confuzie ne furnizează o vizualizare amplă a modulului în care modelul nostru realizează fiecare clasă, un exemplu ar fi că dacă există un număr mai mare de confuzii între clasele similare de mai sus, am putea îmbunătăți preprocesarea sau ajustarea datelor sau a arhitecturii modelului pentru a diminua aceste erori.

În mod normal, rezultatele oferite, propun faptul că modelul nostru de segmentare este realizat conform majorității claselor, dar se pot regăsi oportunități de dezvoltare. Clasele cu performanțe minime ar putea avea parte de tehnici suplimentare de augmentare a datelor, sau de modificări ale arhitecturii rețelei, sau de antrenament suplimentar cu seturi de date mult mai echilibrate. Analiza făcută detaliat a metricilor de evaluare, ne furnizează identificarea punctelor importante pozitive și slăbiciunile modelului nostru, asigurând direcții clare pentru dezvoltarea performanței segmentării semantice a imaginilor.

3.5 Problema Zgomotului în Imagini

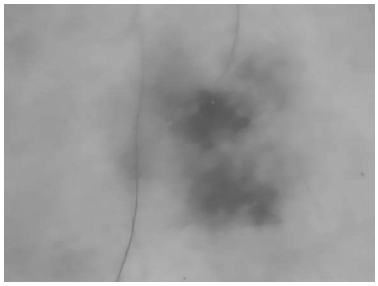
Conform procesului de segmentare a imaginilor, prezența zgomotului poate afecta în mod evident performanța modelului. Zgomotul poate fi introdus din diverse surse, precum senzorii camerei, compresia imaginii sau condițiile de mediu. Acest zgomot poate degrada calitatea imaginilor și poate induce rezultate incorecte în segmentare, fiind de mai multe tipuri: gaussian, zgomotul de sare și piper sau zgomotul uniform. În prezența unui zgomot evident, modelul U-Net poate avea probleme în a segmenta corect regiunile de interes, pentru că zgomotul poate distorsiona caracteristicile vitale ale imaginii.[9] Mai jos regăsim un exemplu de imagine afectată de zgomot și cum performanța rețelei a fost afectată.

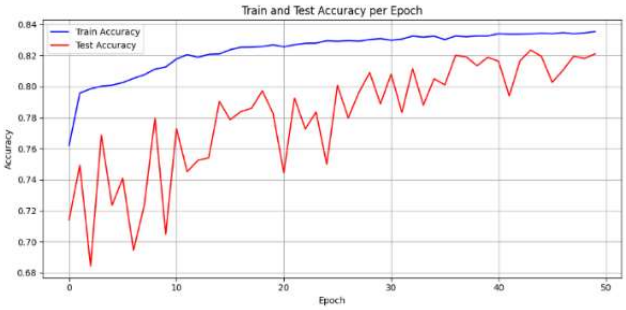




3.6 Influenta zgomotului

Pentru a putea evalua și dezvolta performanța modelului în prezența zgomotului, am folosit o strategie de adaguare/reducere zgomot, aceasta adăugând un zgomot artificial în imagini și aplicarea de tehnici de reducere zgomot pentru a putea elimina zgomotul de dinaintea efectuării segmentării. Prin aceasta putem observa cât de robust este modelul folosit la zgomot și cât de eficient poate fi procesul de reducere zgomot, cu scopul îmbunătățirii segmentării. Un exemplu se regăsește mai jos prin aplicarea reducerii zgomotului pe o imagine, împreună cu un grafic alăturat.





[9] Computer Vision: Algorithms and Applications - Richard Szeliski

3.8 Modelul de Reducere zgomot Chambolle

Pentru a putea elimina zgomotul din imagini, este folosit modelul Chambolle de reducere zgomot, aceasta bazându-se pe regularizarea totală a variației, fiind o metodă populară. Regularizarea totală a variației are ca și obiectiv minimizarea variației totale a imaginii, păstrând totodată și marginile importante. Acest model utilizează pentru reducerea zgomotului imaginilor algoritmul propriu de proiecție, bazându-se pe minimizarea variației totale a imaginii, abordarea fiind populară pentru capacitatea acesteia de a prezerva marginile netede ale obiectelor. Fundamentul matematic al acestui model se bazează pe problemele de optimizare convexă, în care soluția este găsită prin metode iterative de proiecție, formularea matematică a reducerii zgomotului prin variație totală fiind dată de problema de minimizare.[717]

$$\min_{u} \int_{\Omega} |\nabla u(x)| dx$$

$$\int_{\Omega} u(x)\, dx = \int_{\Omega} f(x) dx\,, \int_{\Omega} |u(x) - f(x)|^2 dx = a^2\, |\Omega|$$

unde u preprezintă imaginea in care zgomotul este eliminat, f este imaginea afectata de zgomot, a^2reprezintă varianța zgomotului, și Ω este domeniul imaginii.

Algoritmul de proiecție Chambolle este datorat formulării duale a problemei TV. și folosește un operator de divergență discretizat pentru a putea oferi gradientul imaginii. Pașii pentru acest lucru au în vedere folosirea gradientului discret și folosirea unui procedeu de proiecție la fiecare iterație pentru a putea oferi soluția în setul de funcții de variație limitată.

Inițializează p la zero (vectorul de gradient dual).

Repetă până la convergență:

- Calculează divergența div p și actualizează p folosind:

$$p \leftarrow \frac{p + \tau \nabla (div\, p - \lambda f)}{1 + \tau |\nabla (div\, p - \lambda f)}$$

- Actualizează u ca:

[17] Ipol : Chambolle's Projection Algorithm for Total Variation Denoising - Joan Duran, Bartomeu Coll, Catalina Sbert

$$u \leftarrow \mathrm{f} - \frac{1}{\lambda} div\, p$$

- Parametrul **λ** controlează echilibrul între fidelitatea la date și netezire, iar τ este pasul de timp în schemele iterative, care trebuie ales cu grijă pentru a asigura convergența.

Astfel, modelul a fost extins și pentru reducerea zgomotului imaginilor nu doar alb-negru, prin adaptarea la variația totală vectorială, care se folosește de canalele de culoare într-un mod integrat pentru a obține coerența cromatică și detaliile necesare structurale ale imaginii, folosind o abordare similară, însă în context ultidimensional, unde variația este calculată pe orice canal de culoare.

- Gradientul discret al unui pixel u(i,j)u(i, j)u(i,j) este definit ca un vector de diferențe finite:

$$\nabla u(i,j) = (\partial_x u(i,j), \partial_y u(i,j))$$

unde:

$$\partial_x p_x(i,j) = p_x(i,j) - p_x(i-1,j),\ \partial_y p_y(i,j) = p_y(i,j) - p_y(i,j-1)$$

Aceste diferențe sunt calculate pe axele x și y ale imaginii, respectiv.

- Divergența pentru un câmp vectorial *p(i,j)*=$p_x(i,j), p_y(i,j)$ este calculată ca suma diferențelor finite ale componentelor vectoriale:[817]

$$div\, p(i,j) = \partial_x\, p_x(i,j) + \partial_y\, p_y(i,j)$$

unde:

$$\partial_x\, \mathrm{p}_x(i,j) = \mathrm{p}_x(i,j) - \mathrm{p}_x(i-1,j), \partial_y\, \mathrm{p}_y(i,j) = \mathrm{p}_y(i,j) - \mathrm{p}_y(i,j-1)$$

Actualizarea iterativă în algoritmul Chambolle: Cu aceste definiții, actualizările în cadrul algoritmului Chambolle devin:

- Actualizare p conform:

[17] Ipol : Chambolle's Projection Algorithm for Total Variation Denoising - Joan Duran, Bartomeu Coll, Catalina Sbert

$$p^{n+1}(i,j) = \frac{p^n(i,j) + \tau\nabla(div\, p^n(i,j) - \lambda f(i,j))}{1 + \tau|\nabla\big(div\, p^n(i,j) - \lambda f(i,j)\big)|}$$

- Actualizare u ca:

$$u^{n+1}(i,j) = f(i,j) - \frac{1}{\lambda} divp^{n+1}(i,j)$$

Astfel, prin aceste actualizări, algoritmul poate minimiza energia variației totale în timp, păstrându-și fidelitatea față de imaginea originală afectata de zgomot f(i,j), folosind o abordare bazată pe problemele de optimizare convexă, aceste tehnici asigurând variația totală a imaginii, fiind minimizată, fără a netezi excesiv marginile și detaliile importante.[917]

3.9 Aplicarea și evaluarea reducerii zgomotului

Pentru a putea calcula eficiența modelului de reducere al zgomotului Chambolle, am utilizat tehnica pe imagini zgomotoase pentru a compara rezultatele segmentării cu și fără reducere zgomot.

```python
import numpy as np
import matplotlib.pyplot as plt

def add_noise(image, noise_factor=0.2):
    noise = torch.randn_like(image) * noise_factor
    noisy_image = image + noise
    noisy_image = torch.clamp(noisy_image, 0., 1.)
    return noisy_image

num_images = 5
plt.figure(figsize=(15, num_images * 5))

model.eval()
with torch.no_grad():
    for i, (inputs, labels) in enumerate(test_loader):
        if i >= num_images:
            break

        noisy_inputs = add_noise(inputs)
        denoised_inputs = torch.stack([denoise_image(img) for img in noisy_inputs])

        noisy_outputs = model(noisy_inputs.to(device))
        denoised_outputs = model(denoised_inputs.to(device))

        for j in range(inputs.size(0)):
            plt.subplot(num_images, 3, i * 3 + 1)
            plt.imshow(noisy_inputs[j].cpu().permute(1, 2, 0))
            plt.title('Noisy Image')
            plt.axis('off')

            plt.subplot(num_images, 3, i * 3 + 2)
            plt.imshow(noisy_outputs[j].cpu().squeeze(), cmap='gray')
            plt.title('Noisy Predicted Mask')
            plt.axis('off')

            plt.subplot(num_images, 3, i * 3 + 3)
            plt.imshow(denoised_outputs[j].cpu().squeeze(), cmap='gray')
            plt.title('Denoised Predicted Mask')
            plt.axis('off')

plt.savefig('noisy_denoised_segmentation_results.png')
```

[17] Ipol : Chambolle's Projection Algorithm for Total Variation Denoising - Joan Duran, Bartomeu Coll, Catalina Sbert

Astfel, fiecare rând reprezintă un exemplu de segmentare, incluzând imaginea afectata de zgomot, masca utilizată pe imaginea afectata de zgomot și masca pe imaginea in care am eliminat zgomotul, aceste vizualizări permițându-ne să putem nota vizual eficiența procedeului de eliminare al zgomotului.

3.10 Concluzii privind adaugare/reducere zgomot

Dacă ar fi să tragem o concluzie privind adaugare zgomot și eliminare zgomot folosind algoritmul Chambolle, rezultatele și observațiile obținute denotă o îmbunătățire vizibilă în obținerea segmentării modelului U-Net, în condiții de zgomot. Implementarea algoritmului, facilitează obținerea unor imagini mult mai lizibile, ce permit, o mai bună deosebire și segmentare a regiunilor de interes, subliniind capacitatea algoritmului Chambolle de a perfecționa acuratețea în procesele de analiză a fiecărei imagini, o componentă necesară în aplicațiile medicale sau alte domenii unde calitatea imaginii este obligatorie și fundamentală.

Astfel, aplicarea ineficientă a reducerii zgomotului poate induce în pierderea detaliilor necesare sau poate introduce artefacte care distrug informația de bază a imaginii. O valoare prea mare a parametrului λ poate cauza o supranetezirea imaginii, estompând detaliile sublime esențiale pentru segmentarea precisă, în timp ce o valoare mult prea mică este inutilă pentru a reduce zgomotul eficient, făcând segmentarea să devină vulnerabilă la fluctuațiile aleatorii ale intensității pixelilor.

Putem spune că selecția atentă și calibrarea parametrilor de reducere a zgomotului sunt esențiali pentru a duce la un bun sfârșit beneficiile acestuia fără a periclita integritatea structurală a imaginilor procesate, fiind necesar să efectueze teste dezvoltate și ajustări are parametrilor conform caracteristicilor datelor și ale zgomotului utilizat.

Astfel, abordarea de reducere zgomot folosită cu ajutorul algoritmului Chambolle este o strategie esențială pentru dezvoltarea performanței modelului U-Net în segmentare, sporind robustețea modelului față de variațiile de calitate ale imaginii, extinzând aplicabilitatea practică a acestuia în scenarii reale, gestionând zgomutul ca fiind o provocare constantă.

Capitolul 4 – IMPLEMENTARE PRACTICĂ

4.1 Configurarea mediului de lucru

Mediul de lucru a fost realizat folosind un mediu virtual de Python în care am importat librării specializate pe înmulțiri de matrici, procesare set de date și imagini, utilizând în principal librăriile Numpy și PyTorch care dispune un suport dezvoltat pentru operații tensoriale și rețele neuronale.

4.1.1 Instalarea PyTorch și a altor dependențe

PyTorch reprezintă o bibliotecă de machine learning open-source bază pe Torch, folosită pentru aplicații de tip computer vision și procesare de limbaj natural, este constant dezvoltată de Facebook AI Research Lab și este renumită pentru modul ușor și flexibil de utilizare în dezvoltarea rețelei neuronale.

Pentru a putea utiliza accelerarea GPU, trebuie instalată versiunea PyTorch care poate fi folosită împreună cu CUDA, acesta fiind un model de programare și o platformă de calcul paralel realizată de NVIDIA, ce oferă utilizarea GPU-urilor pentru realizarea de date masive și complexe. Astfel ne folosim de resursele oferite de placa video, ideală pentru înmulțiri de matrici.

4.2 Codul sursă și explicații

4.2.1 Procesarea și încărcarea datelor

Datele sunt adăugate cu extensia JPEG, JPG și PNG, constituind imagini de intrare dar și măștile de segmentare necesare, setul de date fiind organizat într-o structură ierarhică pentru a obține accesul și procesarea acestuia.

```
dataset/
├── images/
│   ├── ISIC_0000000.jpg
│   ├── ISIC_0000001.jpg
│   └── ...
└── masks/
    ├── ISIC_0000000_segmentation.png
    ├── ISIC_0000001_segmentation.png
    └── ...
```

Adăugarea si convertirea datelor s-a realizat cu torchvision.transforms pentru preprocesarea imaginilor și măștilor, incluzând si redimensionarea și conversia în tensor.

```python
import torchvision.transforms as transforms
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
])

from torch.utils.data import Dataset
from PIL import Image
import os

class CustomDataset(Dataset):
    def __init__(self, ids, transform=None):
        self.ids = ids
        self.transform = transform

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

    def __getitem__(self, idx):
        formatted_id = 'ISIC_{:07d}'.format(self.ids[idx])
        base_path = 'path_to_dataset'
        image_path = os.path.join(base_path, 'images', f'{formatted_id}.jpg')
        mask_path = os.path.join(base_path, 'masks', f'{formatted_id}_segmentation.png')

        image = Image.open(image_path).convert("RGB")
        mask = Image.open(mask_path).convert("L")

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

        return image, mask

```

```python
import random

TRAIN_SIZE = 7000
TEST_SIZE = 3000
total_ids = random.sample(range(24306, 34321), TRAIN_SIZE + TEST_SIZE)
train_ids = total_ids[:TRAIN_SIZE]
test_ids = total_ids[TRAIN_SIZE:]

train_dataset = CustomDataset(ids=train_ids, transform=transform)
test_dataset = CustomDataset(ids=test_ids, transform=transform)

from torch.utils.data import DataLoader

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

```

Totalitatea acestor etape privind procesarea și încărcarea datelor, sunt necesare pentru antrenarea modelului de segmentare a imaginilor. Folosind aceste tehnici, ne asigurăm ca datele sunt pregătite în mod adecvat pentru a putea fi utilizate în mod eficient de modelul nostru de deep learning.[19]

[19] Stackoverflow - https://stackoverflow.com/questions/75159444/pytorch-loading-a-custom-dataset

4.2.2 Definirea arhitecturii rețelei

Am utilizat mai multe arhitecturi pentru a realiza teste cu privire la setul nostru de date de tip Ham10000 constituit din imagini cu tipuri de alunițe pe piele diferite, arhitectura care a produs cele mai bune rezultate a fost cea utilizată de U-Net, in comparatie cu celelalte metode de tip CNN, FCN, DeepLab, GAN și SegNET, după cum reiese și din tabelul de mai jos.

Network	Precision	Pixel Accuracy	Jaccard Index	Dice Coefficient	F1 Score	Sensitivity	Specificity
U-Net	0.919	0.902	0.712	0.830	0.830	0.846	0.980
U-Net++	0.950	0.978	0.923	0.959	0.959	0.970	0.982
SegNet	0.891	0.941	0.912	0.897	0.897	0.898	0.862
DeepLabv3	0.884	0.943	0.916	0.894	0.894	0.905	0.957
FCN	0.866	0.883	0.603	0.751	0.751	0.665	0.962
CNN	0.892	0.870	0.574	0.727	0.727	0.615	0.971

4.2.3 Antrenarea modelului

Pentru folosirea antrenării modelului U-Net, am parcurs un set de pași esențiali in realizarea modelului și a funcției de pierdere, până la îmbunătățirea hiperparametrilor și păstrarea punctelor de control si determinarea progresul. Acesta este definit, folosind PyTorch și este configurat să poată rula pe GPU, dacă acesta este disponibil. Constă într-o parte de encoder și una de decoder si sunt utilizate straturi convoluționale și straturi de deconvoluție pentru realizarea segmentări imaginilor.

```
class UNet(nn.Module):
    def __init__(self):
        super(UNet, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(64, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 64, kernel_size=3, stride=2, padding=1, output_padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(64, 1, kernel_size=3, padding=1),
            nn.Sigmoid()
        )

    def forward(self, x):
        x1 = self.encoder(x)
        x2 = self.decoder(x1)
        return x2
```

Antrenarea modelului, s-a realizat folosind funcția de pierdere Binary Cross-Entropy Loss și optimizatorul Adam, fiind configurate astfel:

```
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters())
```

Dezvoltarea eficientă a antrenamentului s-a realizat cu datele procesate în batch-uri, antrenarea acestora având mai multe avantaje precum utilizarea eficientă a memoriei, nefiind necesar încarcarea întregului set de date în memorie, procesând astfel datele în părți mai mici, ideale pentru GPU. Accelerarea procesului de antrenament, prin calcularea gradientelor pe batch-uri mici este mult mai eficientă și oferă actualizarea mai bună a parametrilor modelului și, nu în ultimul rând, stabilizarea antrenamentului. Batch-urile mici adaugă zgomot în estimarea gradientelor, ceea ce poate oferi o evitare a minimelor locale în timpul optimizării. Lotul de antrenament se realizează pe parcursul mai multor epoci, în fiecare dintre acestea datele sunt adăugate în batch-uri și trecute prin model pentru a realiza pierderea și a actualiza parametrii săi.[21]

```
BATCH_SIZE = 64
EPOCHS = 50
BATCHES_PER_EPOCH = 10

train_accuracies = []

for epoch in range(EPOCHS):
    running_loss = 0.0
    total_correct_train = 0
    total_train_samples = 0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

        predicted = outputs > 0.5
        total_correct_train += (predicted == labels).sum().item()
        total_train_samples += labels.numel()

        if i % BATCHES_PER_EPOCH == BATCHES_PER_EPOCH - 1:
            print("[{}, {}] loss: {:.6f}".format(epoch + 1, i + 1, running_loss / BATCHES_PER_EPOCH))
            running_loss = 0.0

    train_accuracy = total_correct_train / total_train_samples
    train_accuracies.append(train_accuracy)

model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['loss']

print("Finished training")
```

Recuperarea și continuitatea antrenamentului, am salvat modelul și a starii optimizatorului la fiecare epocă, utilizând PyTorch's torch.save și torch.load.

[21] Towardsdatascience - Cook your First U-Net in PyTorch - Mostafa Wael

```
model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['loss']

print("Finished training")


torch.save({
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
    'epoch': epoch,
    'loss': loss,
    'train_accuracy': train_accuracies
}, 'model_checkpoint.pth')

checkpoint = torch.load('model_checkpoint.pth', map_location=device)
model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['loss']
```

Prin aceste metode, s-a realizat antrenarea modelului U-Net pentru segmentarea imaginilor, supraveghind performanța acestuia pe tot parcursul epocilor și asigurandu-ne faptul că datele și parametrii optimi sunt gestionați în mod corect.[21]

4.2.4 Evaluarea modelului și interpretarea rezultatelor

Despre evaluarea modelului de segmentare a imaginilor, putem spune că este un pas vital pentru a realiza performanța acestuia și a putea înțelege capacitatea acestuia de generalizare. Evaluarea modelului s-a realizat prin analiza performanțelor pe setul de testare și interpretarea rezultatelor, folosind mai mulți metrici de evaluare. Totodată, s-a asigurat faptul că modelul suferă de overfitting prin compararea performanțelor pe seturile de antrenament și testare. Pentru a putea evalua performanța de segmentare au fost utilizați următorii metrici:

- Pixel Accuracy**:** Proporția de pixeli corect clasificați.
- Jaccard Index**:** Măsoară suprapunerea între segmentarea prezisă și segmentarea reală.
- Dice Coefficient**:** Similar cu indicele Jaccard, dar pune accent pe identificarea zonelor de suprapunere.

Pentru a analiza modelul, s-a adăugat setul de testare și s-au calculat metrici de performanță pentru fiecare batch, modelul salvat la fiecare epocă a fost adăugat pentru a putea verifica evoluția performanței de-a lungul antrenamentului.[21]

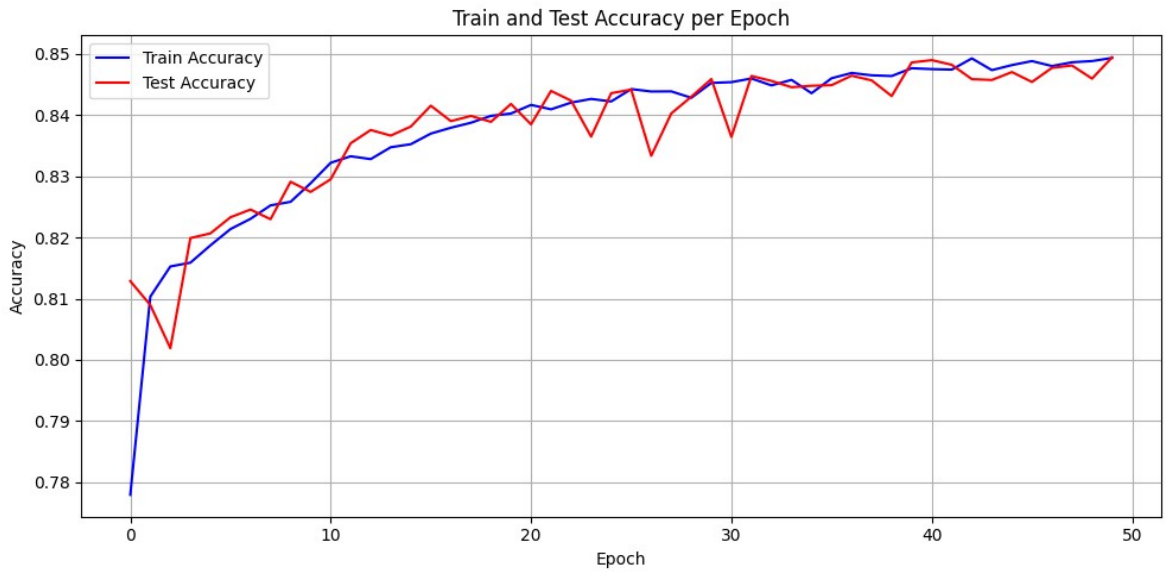
[21] Towardsdatascience - Cook your First U-Net in PyTorch - Mostafa Wael

```python
test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False)
def pixel_accuracy(predicted, target):
    correct_pixels = (predicted == target).sum().item()
    total_pixels = target.numel()
    return correct_pixels / total_pixels

def jaccard_index(predicted, target):
    intersection = torch.logical_and(predicted, target).sum().item()
    union = torch.logical_or(predicted, target).sum().item()
    return intersection / union

def dice_coefficient(predicted, target):
    intersection = torch.logical_and(predicted, target).sum().item()
    dice = (2 * intersection) / (predicted.sum().item() + target.sum().item())
    return dice

pixel_accuracies = []
jaccard_indexes = []
dice_coefficients = []

model.eval()
with torch.no_grad():
    for inputs, labels in test_loader:
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        predicted = outputs > 0.5
        pixel_accuracies.append(pixel_accuracy(predicted, labels))
        jaccard_indexes.append(jaccard_index(predicted, labels))
        dice_coefficients.append(dice_coefficient(predicted, labels))

print(f"Average Pixel Accuracy: {np.mean(pixel_accuracies):.4f}")
print(f"Average Jaccard Index: {np.mean(jaccard_indexes):.4f}")
print(f"Average Dice Coefficient: {np.mean(dice_coefficients):.4f}")
```

Testul de overfitting s-a facut verificând performanța modelului pe seturile de antrenament și testare, salvând și adăugând checkpoint-uri la fiecare epocă pentru a monitoriza evoluția acurateței.

Rezultatele analizei dezvăluie că modelul U-Net antrenat a realizat o performanță puternică pe setul de testare, arătând capacitatea acestuia de a segmenta eficient imaginile. Prin analizarea graficelor de acurateţe pe seturile de antrenament și testare, a reieșit faptul că modelul a reușit în mod eficient să învețe din datele de antrenament sau chiar să facă overfitting, acesta fiind indicat de o mare discrepanță între acuratețea pe setul de antrenament și de testare. In caz de overfitting, acuratețea continua să crească pe setul de antrenament, în timp ce pe setul de testare să stagneze sau să scadă, sugerând că modelul a reușit să memoreze datele de antrenament, în loc să generalizeze la date noi. Prin monitorizarea acestor metrici, a rezultat ajustarea și îmbunătățirea performanței modelului nostru de segmentare, asigurându-ne că funcționează corect, nu doar pe datele de pe antrenament, ci și pe altele noi și neutilizate. [11]Mai jos se regăsește un grafic pentru overfitt-ul rețelei noastre U-Net:

[11] U-Net: Convolutional Networks for Biomedical Image Segmentation - Ronneberger, Olaf, Philip Fischer & Thomas Brox



4.3 Exemple de segmentare

Segmentarea imaginilor este o componentă importantă, fiind folosită pentru a identifica și distinge regiunile de interes dintr-o imagine, existând mai multe tipuri de segmentare, fiecare având aplicații și abordări diferite și specifice.

Sunt cunoscute segmentări de tip Thresholding, implicând selectarea unui prag care separă pixelii obiectelor de fond care poate fi global sau adaptiv, utilizat în mod obișnuit pentru imagini cu contrast ridicat între obiect și fundal, cum ar fi documentele scanate sau imagini medicale simple.[12]

Totodată cunoaștem și segmentarea Region-based Segmentation, care are la bază metode pentru creșterea regiunii, divizarea și combinarea, unde imaginea este divizată în regiuni bazate pe criterii predeterminate, fiind eficientă pentru imagini unde regiunile de interes sunt relativ omogene în ceea ce privește intensitatea sau textura.

Segmentarea Edge Detection, care detectează marginile obiectelor din imagine prin identificarea discontinuităților în intensitatea pixelilor, incluzând detectoarele de margini Sobel, Canny și Prewitt, acestea fiind comune, este utilizată pentru a extrage forma și conturul obiectelor și este importantă în detectarea formelor și în analiza de mișcare.

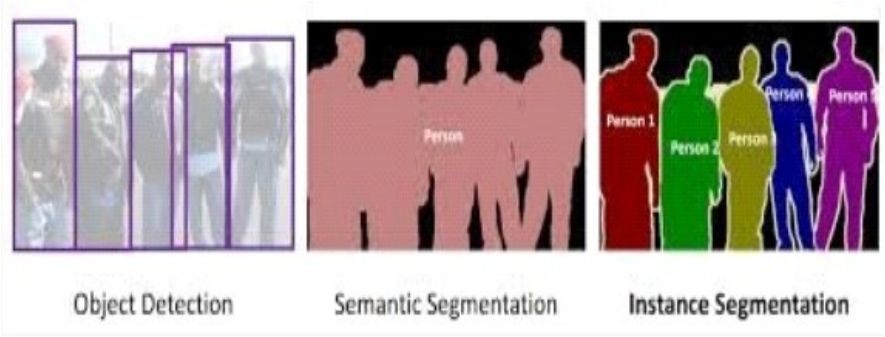
Alte tip de segmentare notabilă ar fi segmentarea semantică, care reprezintă o formă avansată de segmentare si care clasifică fiecare pixel din imagine într-o categorie specifică, asigurând o înțelegere în detaliu a scenei. Este extrem de relevantă în domeniul vehiculele autonome, navigația roboților și în aplicațiile de realitate augmentată, unde înțelegerea contextului scenei este necesară.

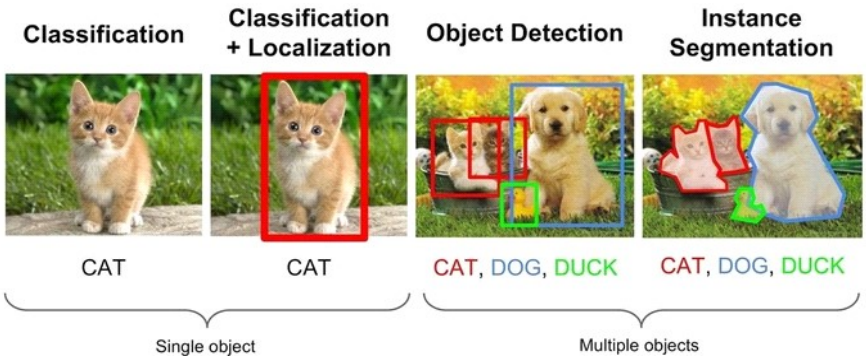
Ultimul tip segmentare este segmentarea instanțelor, similară cu cea semantică. Se distinge prin diferențierea instanțelor obiecte din aceeași categorie și este utilizată în aplicații industriale și de producție, unde obiecte multiple de același tip sunt necesare a fi identificate și urmărite în mod

[12] Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution & Fully Connected CRFs - Chen, Liang-Chieh

individual.

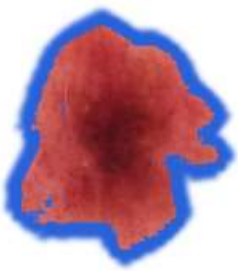
În contextul prezent, metodele folosite pe învățare specială, în special rețelele neuronale convoluționale și arhitectura derivată din acestea, domină segmentarea avansată. Aceste tehnici pot oferi performanțe superioare datorită posibilității de a învăța caracteristici complexe în datele de antrenament, fără a fi necesară presetarea manuală sau ajustarea complexă a parametrilor.

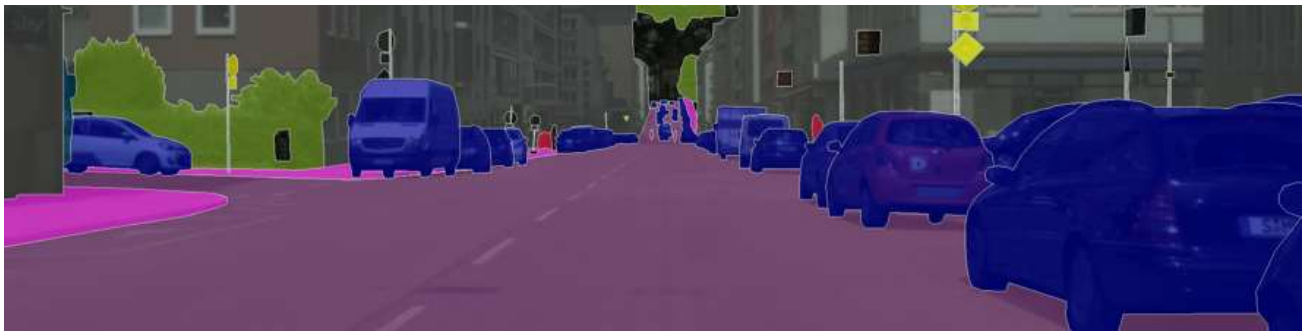




4.3.1 Vizualizări ale rezultatelor segmentării

În această subcabitol, am prezentat unele exemple vizuale ale rezultatelor obținute prin metoda de segmentare semantică.





1. Imaginea originală - Arată o fotografie sau o imagine medicală (în funcție de aplicație) fără procesare.
2. Rezultatele segmentării - Pentru fiecare tehnică de segmentare, prezintă o imagine comparativă:
    - Segmentarea prin metoda U-Net**:** O imagine care arată regiunile de interes clar delimitate.
    - Segmentarea prin FCN (Fully Convolutional Networks): O imagine care compară regiunile segmentate cu cele reale, posibil printr-o suprapunere de culori.

4.3.2 Discuții asupra performanței și limitărilor

Dacă este să vorbim despre performanță și limitări, putem discuta despre precizia segmentării, aceasta oferind detalii cu privire la acuratețea cu care orice tehnică a identificat regiunile de interes, incluzând statistici, precum IoU sau scorurile F1, dacă acestea sunt disponibile, eficiența procesării, unde putem comenta despre timpul necesar pentru procesarea imaginilor de către fiecare algoritm. Acest aspect este necesar în aplicații, unde timpul de răspuns este critic în special în cadrul analizelor medicale de urgență și nu în ultimul rând despre limitări, unde unele tehnici ar putea fi mai puțin eficiente în prezența zgomotului din imagini sau pot necesita ajustări manuale ale parametrilor pentru diferite seturi de date.

Capitolul 5 – DISCUȚII ȘI CONCLUZII

5.1 Rezultatul contribuțiilor

Rezultatele noastre se bazează pe analiza comparativă a mai multor arhitecturi de rețele neuronale convoluționale pentru o segmentare semantică, subliniind performanțele cât și avantajele U-Net față de alte arhitecturi, totodată și alegerea optimizatorului cel mai eficient pentru contextul de față. Contribuția noastră se concentrează pe analiza comparativă a mai multor arhitecturi de rețele neuronale convoluționale pentru segmentarea semantică, evidențiind performanțele și avantajele modelului U-Net comparativ cu alte arhitecturi, precum și studierea comportamentului modelului matematic Chambolle folosit in reducerea zgomotului.

Totuși, Fully Convolutional Networks pot transforma absolut orice rețea de clasificare într-una de segmentare, gestionând imagini de diferite dimensiuni, care tind să fie infim de eficiente în reconstruirea detaliilor simple ale imaginilor, datorită conexiunilor dintre encoder și decoder. Dacă e să vorbim despre SegNet, acesta excelează prin reconstrucția detaliilor de rezoluție înaltă prin straturi de unpooling, păstrând informații despre locația pixelilor, deși este limitat de performanțele inferioare zgomotului din imagine, comparativ cu U-Net. Despre DeepLab putem spune că oferă o segmentare impresionantă pe diferite scale deoarece straturile acestuia convețională sunt dilatate și pooling-ul este piramidal. Complexitatea acestuia poate deveni un dezavantaj în aplicații, timpul de răspuns fiind critic.[10]

U-Net în schimb, a arătat că este cel mai eficient pentru aplicația pe care am elaborat-o, datorită arhitecturii acestuia simetrică, permițând un transfer ideal de informații și o reconstrucție exactă a detaliilor, folosind conexiuni de tip skip între straturile de encoder și decoder, ajutând la păstrarea detaliilor critice ale imaginilor, făcându-l perfect pentru o segmentare precisă împotriva zgomotului. Astfel acesta a fost ales ca fiind cel mai potrivit pentru segmentarea imagistică în condiții de zgomot variabil.[11]

În vreme ce m-am axat pe optimizare, am folosit mai mulți optimizatori precum Adam, RMSprop sau SGD pentru a putea obține maximizarea modelului U-Net. Primul despre care vom discuta este Stochastic Gradient Descent, fiind un optimizator destul de robust care necesită o ajustare sub supraveghere a ratei de învățare, deși avantajul lui este că este lent în convergența către minimul funcției. RMSprop este mult mai adaptabil la schimbările gradientului, mult mai eficient în navigarea contextului complex al funcțiilor, însă devine destul de instabil în fazele de sfârșit ale

[10] Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution & Fully Connected CRFs - Chen, Liang-Chieh
[11] U-Net: Convolutional Networks for Biomedical Image Segmentation - Ronneberger, Olaf, Philip Fischer & Thomas Brox

antrenării. Astfel, Adam a dovedit a fi o îmbinare între primul ordin și adaptabilitatea ratei de operare, făcând ca acesta să fie cel mai eficient pentru U-Net în contextul lucrării elaborate, accelerând convergența inițială și gestionând mult mai eficient dispariițiile gradientului, făcându-l perfect pentru segmentarea împotriva zgomotului și a datelor variabile. Chambolle, popular pentru eficiența acestuia în minimizare a variației totale fără a fi necesară netezirea în mod excesiv a marginilor și detaliilor esențiale, a fost utilizat datorită avantajelor diferite față de celelalte metode enumerate mai sus de reducere a zgomotului.

Una dintre cele mai mari provocări ale reducerii zgomotului imagistic constă păstrarea clarității și acurateței marginilor și a detaliilor simple, algoritmul excelând la acest aspect, conform abordării minimizării variației totale, preservând discontinuitățile în date, în timp ce netezește zonele omogene. Abordarea este esențială pentru aplicațiile de segmentare unde acuratețea marginilor poate impacta în mod exact performanța modelului.

Când vorbim despre eficiența computațională, deși sunt diferite tehnici cât mai actuale ce se bazează pe rețelele neuronale, acestea pot promite rezultate extraordinare, agloritmul Chambolle este o metodă tradițională și independentă de resurse.[12]

Astfel, acesta oferă rezultate sigure și predictibile, fiind vital în aplicații medicale sau alte domenii unde fiabilitatea este unul din factorii primi, metodele acestuia de învățare automată, pot fi imprevizibile, mai exact în prezența datelor care se abat de la distribuțiile obținute în timpul antrenamentului. Dacă e să-l comparăm cu alte metode dezvoltate, algoritmul Chambolle este mult mai ușor de utilizat și integrat în fluxul de lucru existent, totodată ajustarea și calibrarea parametrilor este destul de intuitivă.

5.2 Concluzii finale

Rezultatele obtinute in lucrarea redactată au condus la concluzia ca U-Net-ul este una dintre cele mai eficiente structuri pentru segmentarea imagistică, conform abilităților acestuia de a păstra detaliile de bază ale imaginii, facilitând o recuperare exactă a caracteristicilor în funcție de nivelurile de adâncime, fiind eficient în combinație cu tehnicile avansate de reducere zgomot, indiferent de variațiile de zgomot. Implementarea algoritmului Chambolle impreuna cu optimizatorul Adam a arătat superioritatea acestuia în rolul de a menține marginile și detaliile în condiții de zgomot, în comparație cu alte metode menționate și testate. [13]

Discutând despre optimizatorul Adam, acesta s-a diferențiat fată de restul optimizatoriilor ca fiind cel mai bun datorită faptului că accelereaza convergența, dezvoltă stabilitatea în momente critice de antrenament. Rezultatele noastre denotă aplicabilitatea tehnologiei utilizate în scenarii reale, chiar și-n diagnosticarea unui medic, unde segmentarea este realizată precis, într-un timp scurt

[12] OpenCV 4 Computer Vision Application Programming Cookbook - David Millan Escriva & Robert Laganiere
[13] Computer Vision: Algorithms and Applications - Richard Szeliski

și poate contribui la detectarea din scurt și posibilitatea de a trata problemele respective. Metodele noastre pot fi extinse și adaptate chiar și pentru alte aplicații de procesare vizuală, datorită versatilității și scalabilității soluțiilor care au fost propuse.

Astfel, această lucrare nu doar că poate ajuta la progresarea în domeniul procesării imagistice, ci asigură și soluții ideale pentru provocările cotidiene din industrie, în special medicină. Astfel am putut obține combinația optimă de arhitecturi de rețele, metode de reducere zgomot și optimizări ce pot duce la progrese dacă ne raportăm la calitate și acuratețea segmentării.

5.3 Imbunătățiri posibile

Dacă ar fi să discutăm despre imbunătățiri posibile ne putem gândi la integrarea rețelelor neuronale avansate, cu referire la WINNet (Wavelet-inspired Invertible Network). Această abordare inovatoare are efecte superioare față de modelele matematice în reducerea zgomotului bazându-se pe puterea de invățare si flexibilitatea rețelelor neuronale inversibile.[16]

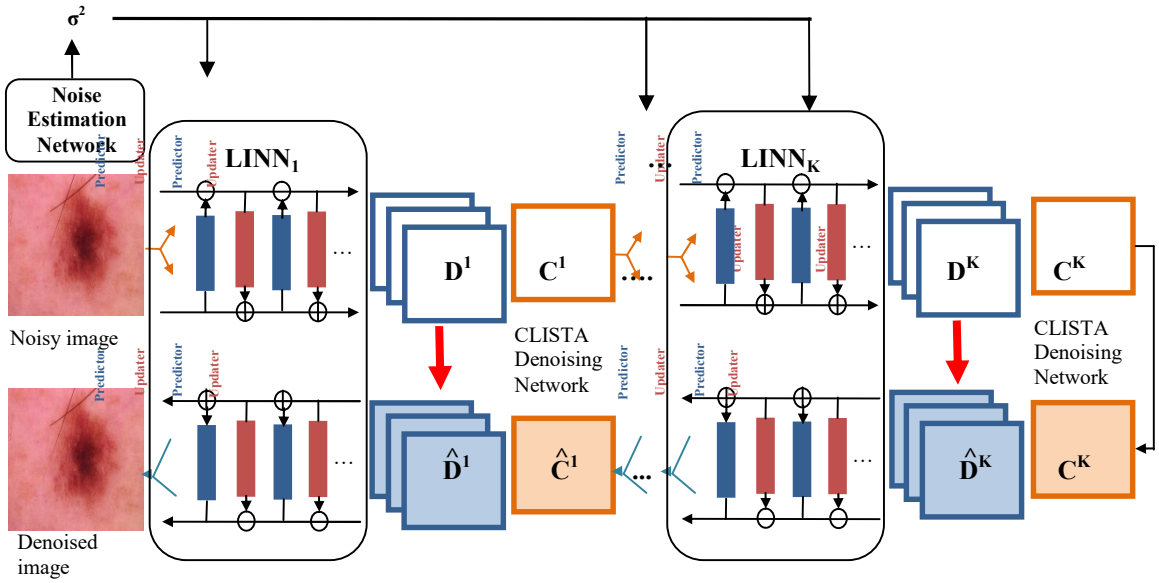
Astfel, câteva avantaje ale acestei metode ar putea fi :

- Obținerea unei performanțe cu mult superioară modelelor datorată trasformariilor walvet pentru a descompune imaginea în componente detaliate combinate cu obordatea de rețea neuronala in scopul eliminării zgomotului.
- Această abordare se poate adapta la diferite tipuri de zgomot deoarece include o rețea de estimare a zgomotului ce are ca scop ajustarea parametriilor pentru un anumit nivel de zgomot. Putem spune că avem o generalizare pentru diferite niveluri de zgomot.
- Alt avantaj ar mai fi reconstrucția imaginii întrucât din trasformariile walvet folosite imaginea poate fi interpretată foarte bine, păstrându-se detaliile esențiale și se elimină zgomotul nedorit

Acum dacă ar fi să discutăm depre straturiile LINN_k, putem spune că acestea sunt esențiale datorită:

- descompunerii imaginii în informații generale și detalii fine
- celor doua tipuri de operatori (split si merge)
- operația de predecție și actualizare ce are rol în separarea zgomotului; în acest caz se îmbunătățesc continuu descompuneriile generale si respectiv detaliate, folosind operatori neliniari pe cele două rețele (predict network si update network) pentru a beneficia de relațiile complexe intre date obținute din convoluție.
- reconstrucția perfectă în sensul că nu se pierde informație in procesul de transformare.

[16] Ieeexplore : WINNet: Wavelet-Inspired Invertible Network for Image Denoising - Jun-Jie Huang,Pier Luigi Dragotti



Predict și Update Networks

Predict și Update Networks sunt folosite în ajustarea componentelor generale și detaliate în transformarea și reconstrucția rețelei.

1. Convoluții separabile depth-wise**:**
   - Se aplică separat pe fiecare canal pentru reducerea numărului de parametri și numărului de computații;
   - Convoluția 1x1 are rolul de identificare și folosire a relațiilor complexe într-un mod eficient, folosiind operatori neliniari.
2. Soft-thresholding**:**
   - Reprezintă un operator neliniar ce folosește o funcție de atenuare pe coeficienții obținuți din transformarea walvet. Astfel, zgomotul este elimintat prin suprimarea coeficienților mici, care sunt asociați cu zgomotul: [16]

$$T_\lambda(x) = \mathrm{sgn}(x) \cdot \max(|x| - \lambda, 0)$$

Ca o concluzie putem spune ca afirma ca extinderea rețelei poate rezulta în mai multe scale, așa cum recomandă rezultatele simulărilor prezentate mai sus, oferind o reprezentare mult mai succintă și o separare vizibilă a zgomotului de semnalul util, îmbunătățind rafinamentul și ajustarea scalelor. WINNet ar putea trata diferite moduri de zgomot, dezvoltând calitatea restaurării imaginilor fără a pierde detalii necesare. Astfel adoptarea unei abordări care poate combina modelarea bazată pe rețelele neuronale cu tehnicile de inferență avansate, pot deschide uși noi pentru dezvoltare, pentru reducerea zgomotului mai robust, prin intermediul diferitor aplicații în procesarea imaginilor medicale precum și în alte domenii.

[16] Ieeexplore : WINNet: Wavelet-Inspired Invertible Network for Image Denoising - Jun-Jie Huang,Pier Luigi Dragotti

BIBLIOGRAFIE

1. Principles of Digital Image Processing - Advanced Methods Willhelm Burger & Mark J. Burge
2. Principles of Digital Image Processing - Core Algorithms Willhelm Burger & Mark J. Burge
3. Computer Vision Using Deep Learning - Vaibhav Verdhan
4. OpenCV 4 Computer Vision Application Programming Cookbook - David Millan Escriva & Robert Laganiere
5. Practical Deep Learning - Ron Kneusel
6. Learn Computer Vision Using OpenCV - Sunila Gollapudi & V Laxmikanth
7. Deep Learning - Ian Goodfellow
8. Pattern Recognition and Machine Learning - Cristopher Bishop
9. Computer Vision: Algorithms and Applications - Richard Szeliski
10. Fully Convolutional Networks for Semantic Segmentation - Long, Jonathan, Evan Shelhamer & Trevor Darrell
11. U-Net: Convolutional Networks for Biomedical Image Segmentation - Ronneberger, Olaf, Philip Fischer & Thomas Brox
12. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution & Fully Connected CRFs - Chen, Liang-Chieh
13. Fast.ai Practical Deep Learning for Coders
14. TensorFlow & PyTorch Documentation
15. PyTorch - https://pytorch.org
16. Ieeexplore : WINNet: Wavelet-Inspired Invertible Network for Image Denoising - Jun-Jie Huang,Pier Luigi Dragotti
17. Ipol : Chambolle's Projection Algorithm for Total Variation Denoising - Joan Duran, Bartomeu Coll, Catalina Sbert
18. Arhxiv : Diff-UNet: A Diffusion Embedded Network for Volumetric Segmentation - Zhaohu Xing, Liang Wan, Huazhu Fu,Guang Yang and Lei Zhu
19. Stackoverflow - https://stackoverflow.com/questions/75159444/pytorch-loading-a-custom-dataset
20. Arhxiv : UNet++: A Nested U-Net Architecture for Medical Image Segmentation - Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang
21. Towardsdatascience - Cook your First U-Net in PyTorch - Mostafa Wael
22. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." MICCAI, 2015.