Universitatea Națională de Știință și Tehnologie “Politehnica” București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

***Înlăturarea defocalizării din imagini folosind rețele neuronale profunde***

###### Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de

*Inginer* în domeniul *Electronică, Telecomunicații și Tehnologia Informației*

programul de studii de licență *Ingineria Informatiei*

Conducător științific Absolvent

*Ș.L. Dr. Ing. Cristian Constantin DAMIAN R*ă*ducu Alexandru*

*2024*

Anexa 5

**Declarație de onestitate academică**

Prin prezenta declar că lucrarea cu titlul **“Aplicație de gestionare a bicicletelor și a trotinetelor electrice”**, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Inginerie Electronică, Telecomunicații și Tecnologia Informației*, programul de studii *Tecnologii și Sisteme de Telecomunicații* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățămînt superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, *21-06-2024*

Absolvent *Alexandru RĂDUCU*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Capitolul 1- INTRODUCERE

* 1. [**Cerințele proiectului**](#_Capitolul_1-_INTRODUCERE)

Tema proiectului de licență este **„Înlăturarea defocalizării din imagini folosind rețele neuronale profunde”** și presupune dezvoltarea unei soluții software bazate pe arhitectura U-Net, capabilă să elimine efectele de defocalizare din imagini fotografice.

Scopul principal al proiectului este proiectarea, implementarea, antrenarea și testarea unei rețele neuronale convoluționale adaptate pentru problema „defocus deblurring”, utilizând atât seturi de date existente, cât și imagini generate printr-un simulator propriu de defocalizare. Sistemul dezvoltat trebuie să fie capabil să proceseze imagini de intrare, să identifice zonele neclare și să reconstruiască o versiune cât mai fidelă a imaginii originale.

Cerințele principale includ:

* Alegerea și implementarea unei arhitecturi de tip **U-Net** adaptată pentru restaurarea imaginilor afectate de out-of-focus blur;
* Utilizarea unor funcții de cost relevante (de exemplu L1 Loss, MSE, SSIM Loss) și analiza impactului acestora asupra calității reconstrucției;
* Crearea unui set de date de antrenare și validare prin aplicarea unor filtre de defocalizare simulate pe imagini sharp;
* Realizarea unei interfețe software de testare și afișare comparativă a imaginilor înainte și după procesare;
* Compararea performanței arhitecturii propuse cu alte metode din literatură, inclusiv rețele similare precum AIFNet;
* Documentarea detaliată a procesului, incluzând explicații teoretice, implementare practică, rezultate experimentale și concluzii.

**[1.2 Motivația și obiectivele proiectului](#_Capitolul_1-_INTRODUCERE)**

Defocalizarea este o problemă frecvent întâlnită în fotografie și procesarea imaginilor, apărută atunci când obiectul capturat nu se află în planul optim de focalizare al camerei. Aceasta duce la pierderea detaliilor fine și la degradarea calității vizuale, afectând atât percepția umană, cât și performanța sistemelor automate de analiză vizuală.

În trecut, corectarea defocalizării era abordată preponderent prin metode clasice de deconvoluție, care presupun cunoașterea exactă a kernelului de blur și aplicarea unor tehnici matematice inverse. Aceste metode au performanțe limitate în condiții reale, deoarece modelul de blur este adesea necunoscut sau variază pe întreaga imagine.

Dezvoltarea recentă a tehnicilor de **deep learning** a adus progrese semnificative în domeniul restaurării imaginilor, rețelele neuronale convoluționale fiind capabile să învețe direct relația dintre imaginea blurată și cea clară, fără a estima explicit kernelul de blur. Arhitectura **U-Net**, inițial proiectată pentru segmentare semantică,

s-a dovedit extrem de eficientă și în sarcini de reconstrucție a imaginilor, datorită mecanismului de *skip connections* care facilitează păstrarea informațiilor de detaliu.

Obiectivele proiectului sunt:

1. **Studiul fundalului teoretic** privind tipurile de blur, în special out-of-focus blur, și tehnicile existente de deblur;
2. **Implementarea unei arhitecturi U-Net optimizate** pentru defocus deblurring;
3. **Crearea unui set de date** adecvat antrenării, inclusiv generarea de imagini defocalizate artificial;
4. **Antrenarea și evaluarea rețelei** pe metrici standard precum PSNR și SSIM;
5. **Compararea rezultatelor** cu metode din literatura de specialitate;
6. **Realizarea unei aplicații software** care să permită încărcarea de imagini și procesarea automată a acestora.

[**1.3 Contribuția original**](#_Capitolul_1-_INTRODUCERE)

Contribuția principală a acestui proiect constă în dezvoltarea unei implementări personalizate a unei rețele neuronale de tip U-Net pentru eliminarea defocalizării, cu optimizări aduse arhitecturii și fluxului de antrenare, precum și integrarea acesteia într-o aplicație software funcțională.

Elementele de originalitate includ:

* Adaptarea arhitecturii U-Net cu blocuri reziduale și normalizare în batch pentru stabilitate crescută;
* Utilizarea unei strategii de antrenare cu reluare automată în caz de întrerupere și salvarea progresului;
* Crearea unui **simulator de defocalizare** pentru generarea datelor de antrenament, permițând controlul parametrilor kernelului de blur;
* Compararea performanței arhitecturii propuse cu alte variante și funcții de pierdere, analizând impactul fiecărei alegeri asupra calității imaginilor;
* Implementarea unui modul de afișare comparativă în interfața grafică, facilitând evaluarea vizuală a rezultatelor.

[**1.4 Structura lucrării**](#_Capitolul_1-_INTRODUCERE)

Lucrarea este structurată în șase capitole:

* **Capitolul 1 – Introducere**: prezintă cerințele proiectului, motivația, obiectivele și contribuția originală.
* **Capitolul 2 – Fundamente teoretice**: descrie conceptele de bază privind defocalizarea și metodele de corecție, incluzând modele matematice, metrici de evaluare și arhitecturi de rețele neuronale utilizate în deblur.
* **Capitolul 3 – Tehnologii și medii de dezvoltare**: detaliază limbajele de programare, bibliotecile software, platformele hardware și structura dataset-urilor utilizate.
* **Capitolul 4 – Proiectarea și implementarea sistemului**: prezintă în detaliu arhitectura propusă, procesul de antrenare, modulele software și fluxul de lucru al aplicației.
* **Capitolul 5 – Rezultate experimentale și evaluare**: conține prezentarea rezultatelor obținute, compararea cu metode similare și analiza performanței.
* **Capitolul 6 – Concluzii și direcții viitoare**: rezumă realizările, prezintă concluziile trase și propune posibile îmbunătățiri ale sistemului.

# Capitolul 2 - FUNDAMENTE TEORETICE

**2.1 Noțiuni generale despre defocalizare și tipuri de blur**

Blur-ul reprezintă fenomenul de degradare a clarității unei imagini, determinat de factori precum mișcarea camerei sau a subiectului, setări incorecte ale focalizării sau condiții de iluminare necorespunzătoare. În cazul particular al **defocalizării** (*out-of-focus blur*), degradarea apare atunci când planul de focalizare al camerei nu coincide cu planul în care se află subiectul de interes.

Conform lucrării lui Zhang et al. (2022), blur-ul poate fi clasificat în patru categorii principale:

1. **Motion Blur** – cauzat de mișcarea camerei sau a obiectelor în timpul expunerii.
2. **Out-of-Focus Blur** – rezultă din setarea incorectă a distanței focale; obiectele în afara adâncimii de câmp apar neclare.
3. **Gaussian Blur** – un model simplificat, folosit adesea în preprocesare sau în simulări.
4. **Mixed Blur** – combinație de mai multe tipuri de blur, frecvent întâlnită în scene reale.

O imagine care conține text, captură de ecran, Font, număr

Conținutul generat de inteligența artificială poate fi incorect.*Exemple vizuale pentru fiecare tip de blur, preluate sau adaptate din Zhang22*

**[2.2 Modelarea matematică a defocalizării](#_Capitolul_2_-)**

Din punct de vedere matematic, defocalizarea poate fi descrisă prin **funcția de răspuns la impuls** (*Point Spread Function*, PSF), care modelează modul în care un punct din scenă este răspândit pe senzorul camerei.

Formula tipică pentru PSF-ul circular (discul lui Airy) este:

K(x,y) =

unde:

* (k,l) este centrul PSF,
* r este raza cercului de defocalizare, dependentă de parametrii optici ai camerei (apertură, distanță focală).

O imagine care conține diagramă, captură de ecran, cerc, proiectare

Conținutul generat de inteligența artificială poate fi incorect.

*Reprezentarea 2D și 3D a PSF-ului pentru defocalizare circulară*

Blur-ul defocalizat se aplică imaginii sharp printr-o operație de **convoluție**:

Ib​=Is​∗K+n

unde:

* Ib = imaginea blurată,
* Is​ = imaginea sharp,
* K = kernel-ul de defocalizare,
* n = zgomot aditiv.

**2.3 Tehnici clasice de eliminare a blur-ului**

Metodele tradiționale de **deblur** pot fi împărțite în două mari categorii:

* **Non-blind deblurring** – kernel-ul K este cunoscut; problema devine o inversare stabilizată a convoluției (de exemplu, metoda Wiener, deconvoluția Richardson–Lucy).
* **Blind deblurring** – kernel-ul este necunoscut și trebuie estimat împreună cu imaginea sharp, ceea ce face problema mult mai dificilă și instabilă.

Exemple:

* **Wiener Deconvolution**:
* **Richardson–Lucy**: algoritm iterativ bazat pe estimarea maximă a verosimilității.

O imagine care conține text, captură de ecran, Font, număr

Conținutul generat de inteligența artificială poate fi incorect.

*Flux comparativ: metodă non-blind vs blind*

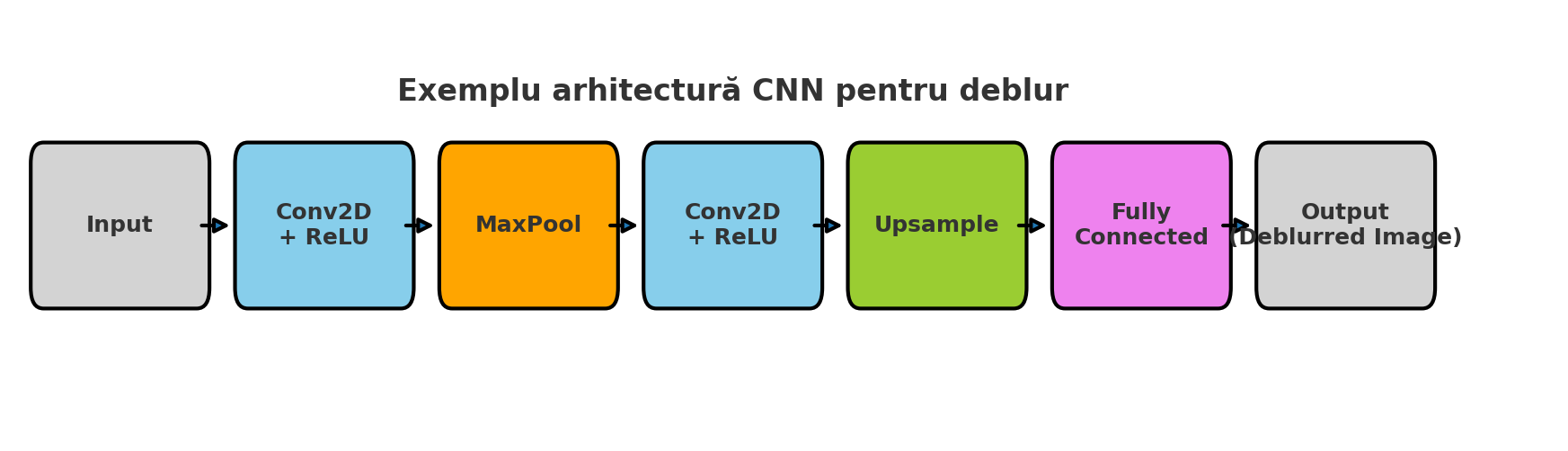
Limitări ale metodelor clasice: sensibilitate ridicată la zgomot, performanță slabă în blur non-uniform, imposibilitatea de a generaliza pentru tipuri complexe de defocalizare.

**2.4 Tehnici bazate pe rețele neuronale**

Rețelele neuronale convoluționale (CNN) au revoluționat procesarea imaginilor, datorită capacității de a învăța reprezentări ierarhice direct din date. În cazul deblur-ului, CNN-urile pot fi antrenate end-to-end pentru a mapă direct o imagine blurată la varianta sa sharp, fără a estima explicit kernel-ul.

Lucrări relevante:

* **DeblurGAN** (Kupyn et al.) – folosește arhitectura GAN cu pierderi perceptuale.
* **AIFNet** – metodă orientată pe defocus deblurring, cu arhitectură multi-branch și fuziune de caracteristici.
* **U-Net** – adaptată pentru deblur prin adăugarea de skip connections și normalizare.



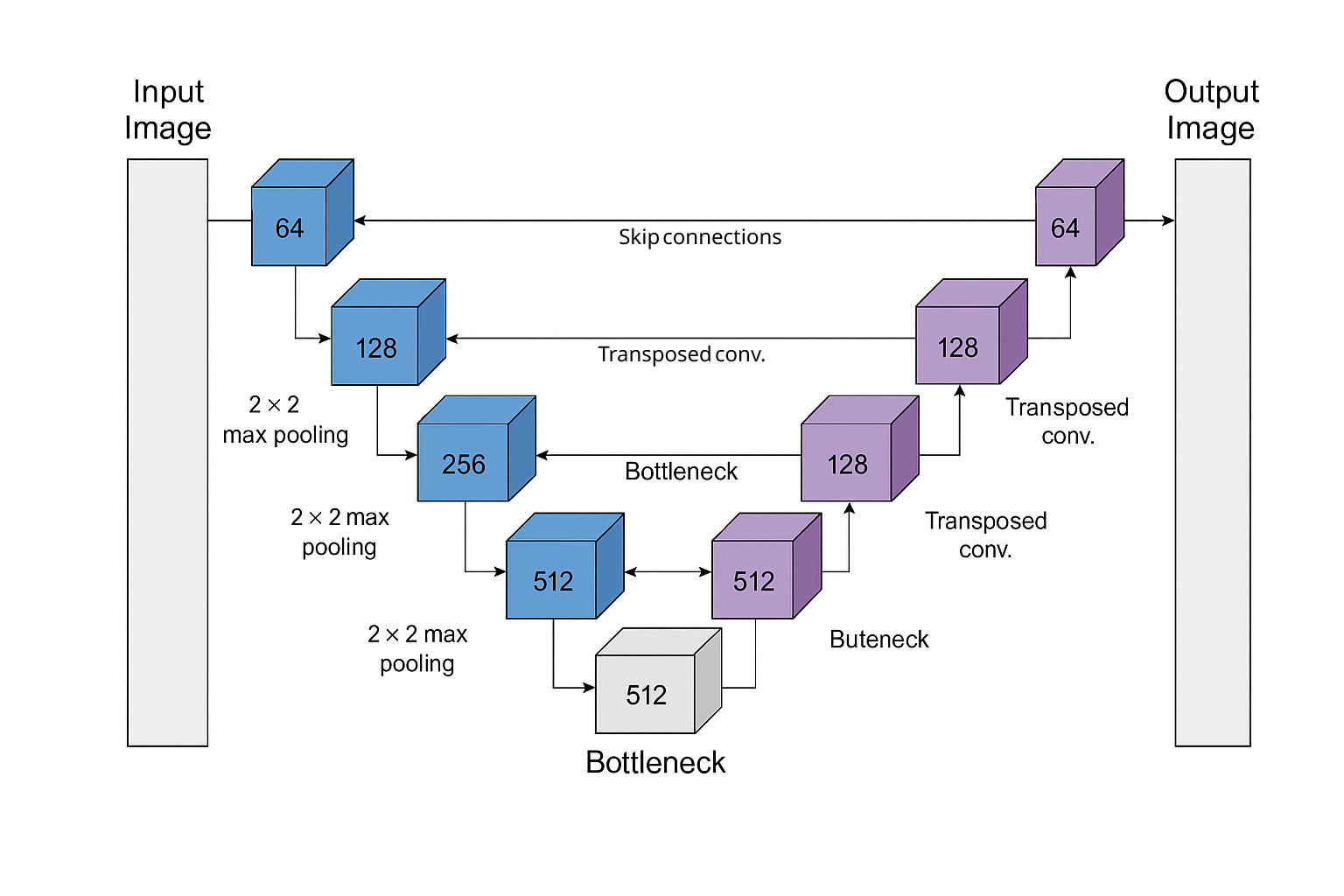
*Exemplu arhitectură CNN pentru deblur*

**2.5 Arhitectura U-Net și adaptările pentru defocus deblurring**

U-Net a fost inițial concepută pentru segmentare biomedicală, dar structura sa encoder–decoder cu conexiuni de tip skip o face ideală pentru reconstrucția imaginilor.

**Componente principale**:

* **Encoder** – extrage caracteristici la diferite scări.
* **Bottleneck** – captează context global.
* **Decoder** – reconstruiește imaginea la rezoluția inițială.
* **Skip connections** – transferă detalii de înaltă frecvență direct de la encoder la decoder.



*Schema arhitecturii U-Net adaptată pentru defocus deblurring*

Optimizări posibile pentru defocus: blocuri reziduale, normalizare în batch, pierderi combinate (L1 + SSIM), augmentare de date cu kerneluri variabile.

**2.6 Metrici de evaluare a calității imaginilor restaurate**

Evaluarea se poate face prin:

* **Metrici full-reference**: PSNR, SSIM, MS-SSIM, LPIPS;
* **Metrici no-reference**: NIQE, BRISQUE;
* **Evaluare subiectivă**: scoruri MOS obținute de la observatori umani.

O imagine care conține plantă, floare, Ghiocel, anemonă

Conținutul generat de inteligența artificială poate fi incorect.

*Exemplu vizual pentru calculul SSIM între două imagini*

Tabel comparativ al metricilor:

| **Metrică** | **Tip** | **Interval valori** | **Interpretare** |
| --- | --- | --- | --- |
| PSNR | FR | >30 dB bun | Măsoară raportul semnal-zgomot |
| SSIM | FR | 0–1 | Măsoară similaritatea structurală |
| NIQE | NR | 0–∞ | Valori mici indică calitate mai bună |

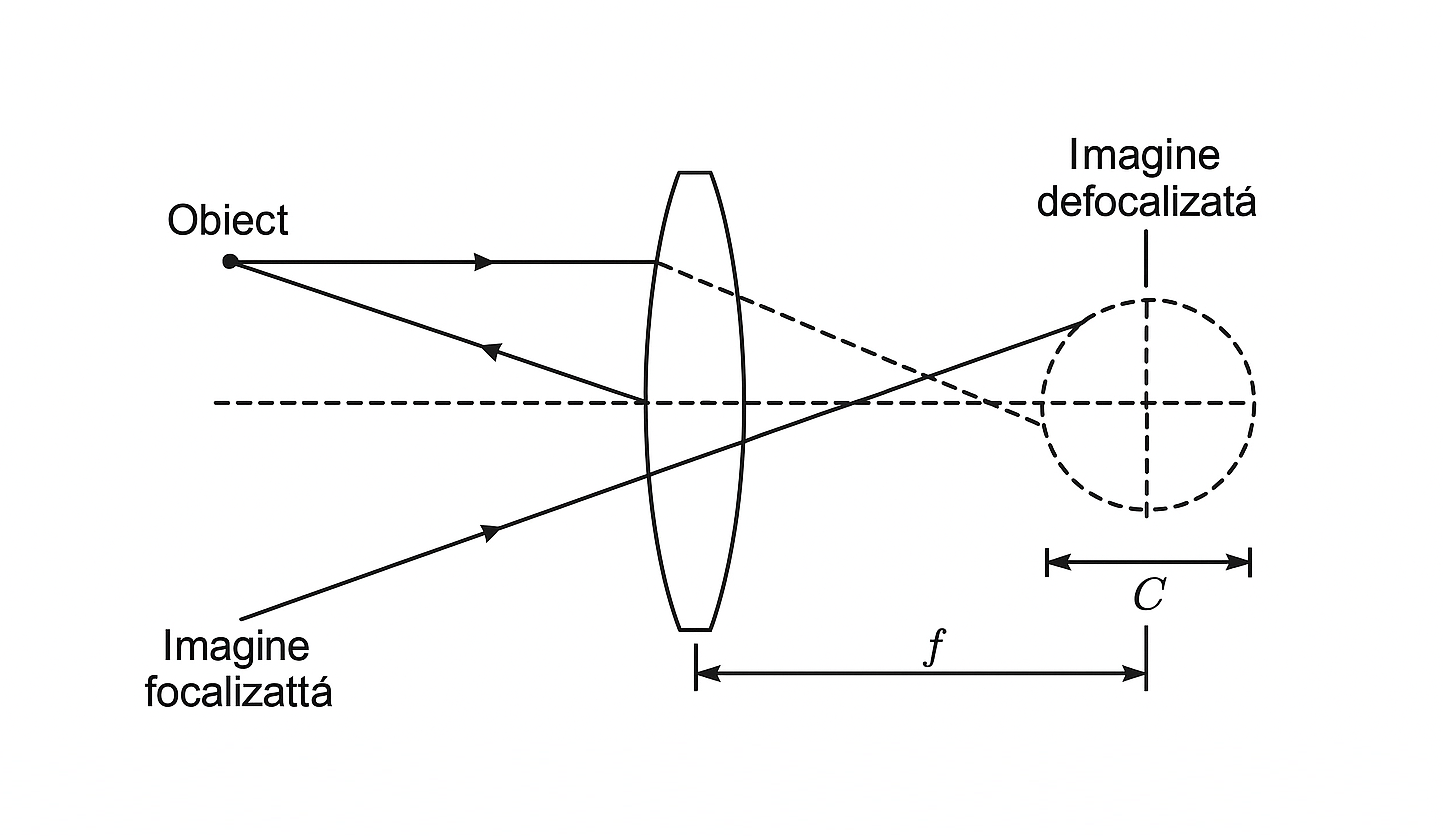
**2.7 Sinteză și concluzii intermediare**

Acest capitol a prezentat conceptele fundamentale necesare pentru înțelegerea și abordarea problemei de defocalizare, modelele matematice implicate, principalele metode tradiționale și moderne, arhitectura U-Net și metricile de evaluare. Aceste fundamente vor fi baza implementării practice descrise în capitolele următoare.

**2.8 Fenomenul de defocalizare în context optic și fotografic**

Defocalizarea apare atunci când un sistem optic (lentilă, obiectiv foto, microscop) formează imaginea unui obiect în afara planului focal optim. Din punct de vedere optic, o lentilă convergentă focalizează razele luminoase provenite de la un punct al obiectului într-un punct al planului imaginii doar dacă acel punct se află exact pe **planul de focalizare**.

Când obiectul se află mai aproape sau mai departe de acest plan, razele nu mai converg într-un singur punct, ci formează un **disc de confuzie** (*circle of confusion*). Dimensiunea acestui disc determină gradul de defocalizare perceput în imagine.



*Schema opticii geometrice care ilustrează formarea discului de confuzie pentru obiecte aflate în fața și în spatele planului de focalizare*

În fotografie, factorii care influențează defocalizarea sunt:

* **Distanța focală** a obiectivului;
* **Apertura** (diametrul diafragmei, notat f/stop);
* **Distanța de focalizare**;
* **Adâncimea câmpului (DoF – Depth of Field)**.

Formulele clasice din optică permit estimarea razei discului de confuzie rrr în funcție de acești parametri:

unde:

* A este diametrul efectiv al diafragmei,
* df este distanța planului focal,
* ds este distanța subiectului.

**2.9 Modelarea defocalizării în procesarea imaginilor**

Din perspectiva procesării numerice a imaginilor, defocalizarea poate fi privită ca o **operație de filtrare spațială** cu un kernel rotund uniform (în cazul ideal), sau cu o funcție Airy în cazul unui sistem optic real.  
Frecvent, pentru simulări și antrenarea rețelelor neuronale, se folosește modelul simplificat al **kernelului disc uniform**.

Această aproximare este justificată prin faptul că majoritatea senzorilor digitali și obiectivelor foto prezintă o distribuție relativ uniformă a energiei luminoase în interiorul discului de confuzie.

📷 **[Aici inserează Fig. 2.8]** – *Comparație între PSF ideal (discul lui Airy) și PSF aproximat ca disc uniform*

Pentru generarea dataset-urilor de antrenare în proiectul nostru, acest kernel va fi aplicat pe imagini sharp folosind transformata Fourier sau funcții predefinite din biblioteci precum OpenCV sau PIL.

**2.10 Clasificarea metodelor de deblur**

Conform Zhang et al. (2022), metodele de deblur pot fi clasificate după mai multe criterii:

**A. După cunoașterea kernelului**

* **Non-blind** – kernel cunoscut;
* **Blind** – kernel necunoscut, estimat împreună cu imaginea.

**B. După modelul matematic**

* Bazate pe **optimizare** (minimizarea unei funcții cost cu regularizare);
* Bazate pe **învățare automată** (ML, DL).

**C. După tipul de date de intrare**

* Single image;
* Video sequence;
* Multi-frame (mai multe fotografii ale aceleiași scene).

📷 **[Aici inserează Fig. 2.9]** – *Diagramă de clasificare a metodelor de deblur conform Zhang22*

**2.11 Metode clasice – detalii și limite**

**2.11.1 Wiener Deconvolution**

Această metodă folosește transformata Fourier pentru a inversa procesul de convoluție și include un termen de regularizare pentru a reduce efectele zgomotului.  
Limitarea principală: necesită cunoașterea exactă a kernelului și este sensibilă la zgomot.

**2.11.2 Richardson–Lucy Deconvolution**

Algoritm iterativ derivat din teoria probabilităților, presupunând zgomot Poisson. Este mai robust la zgomot, dar sensibil la numărul de iterații și poate genera artefacte.



*Comparație vizuală între Wiener și Richardson–Lucy pentru același blur*

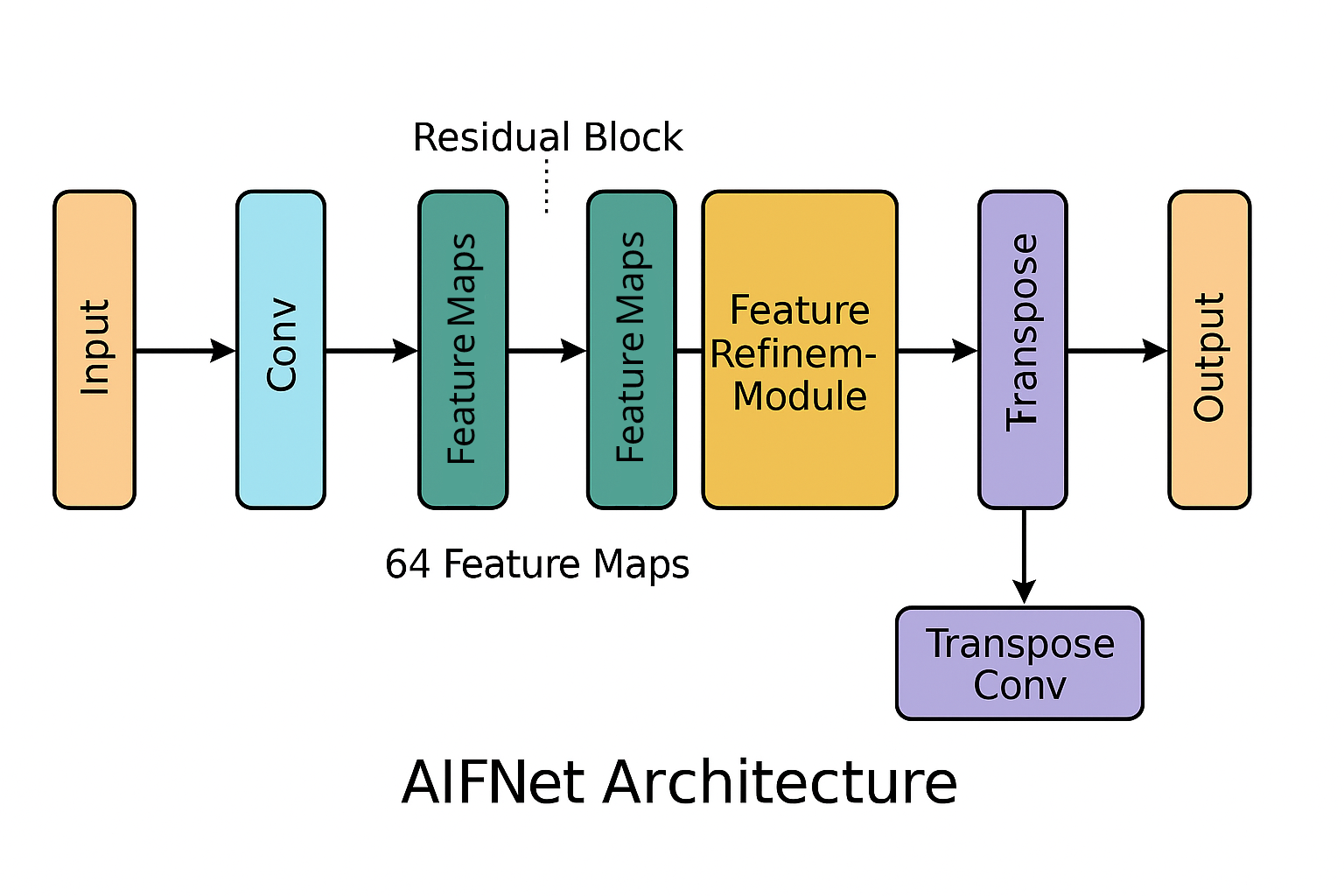
**2.12 Metode moderne bazate pe deep learning**

Rețelele neuronale pentru deblur pot fi împărțite în:

* **CNN-uri pure** (ex. U-Net, ResNet);
* **GAN-uri** (DeblurGAN, SRGAN adaptat pentru deblur);
* **Rețele hibride** (model-based + DL, de ex. USRNet).

În special pentru **defocus deblurring**, arhitecturile moderne includ:

* Straturi convoluționale cu kerneluri mari pentru captarea contextului;
* Blocuri reziduale pentru îmbunătățirea fluxului de gradient;
* Mecanisme de atenție (attention mechanisms) pentru focalizare pe zonele defocalizate.



*Arhitectura AIFNet din GitHub, evidențiind ramurile paralele și modulul de fuziune*

**2.13 Arhitectura U-Net – detalii matematice**

Modelul U-Net este compus din două părți simetrice: encoder și decoder, conectate prin skip connections.  
Fiecare bloc din encoder poate fi descris prin funcția:

unde:

* W1,W2​ sunt filtre convoluționale,
* BN este normalizarea în batch,
* σ este funcția de activare (ReLU).

📷 **[Aici inserează Fig. 2.12]** – *Schema completă a arhitecturii U-Net implementate în proiect, cu blocuri reziduale*

**2.14 Metrici – formulări matematice**

**PSNR**:

**SSIM**:

**LPIPS** – metrică perceptuală bazată pe rețele pre-antrenate.

📷 **[Aici inserează Fig. 2.13]** – *Grafic comparativ între valori PSNR/SSIM pentru metode diferite*

**2.15 Rezumatul capitolului**

În acest capitol am prezentat:

* conceptele fizice și matematice din spatele defocalizării,
* modelele de kernel folosite în simulări,
* metodele clasice și moderne de deblur,
* arhitectura U-Net și motivul alegerii sale,
* metricile utilizate pentru evaluare.

Aceste fundamente vor fi utilizate în capitolele următoare pentru implementarea și evaluarea soluției propuse.

**Capitolul 3 – Tehnologii și medii de dezvoltare**

**3.1 Limbaje și biblioteci software**

Proiectul de licență a fost dezvoltat preponderent în **Python**, datorită ecosistemului său bogat de biblioteci pentru procesarea imaginilor și învățare profundă. Alegerea Python a fost motivată de:

* sintaxa prietenoasă și ușor de învățat;
* suport extins pentru biblioteci de procesare a imaginilor (**OpenCV**, **Pillow**);
* integrarea cu framework-uri de deep learning (**PyTorch**, **TensorFlow**);
* comunitate activă și documentație abundentă.

📷 **[Aici inserează Fig. 3.1]** – *Logo Python și captura unei sesiuni de cod în VSCode cu PyTorch importat*

**Biblioteci utilizate**

* **PyTorch** – framework principal pentru definirea, antrenarea și testarea rețelei neuronale U-Net.
* **Torchvision** – preprocesare de imagini, augmentare, încărcarea dataset-urilor.
* **OpenCV** – filtrare, aplicare kerneluri de defocalizare, salvarea și afișarea rezultatelor.
* **NumPy** – operații vectoriale și matriciale, generare de kerneluri.
* **Matplotlib / Seaborn** – vizualizarea rezultatelor, generarea graficelor PSNR/SSIM.
* **Pillow (PIL)** – conversii de formate de imagine.
* **Tkinter / Streamlit** – pentru interfața grafică a aplicației.

📷 **[Aici inserează Fig. 3.2]** – *Captură listă requirements.txt cu versiunile exacte ale bibliotecilor folosite*

**3.2 Platforme hardware și infrastructură**

Antrenarea rețelelor neuronale pentru deblur este o sarcină intensivă din punct de vedere computațional, în special datorită:

* dimensiunii mari a dataset-urilor;
* complexității arhitecturii U-Net;
* necesității de antrenare pe mai multe epoci (epochs) pentru convergență.

**Configurația hardware utilizată:**

* **CPU:** Intel Core i7 / AMD Ryzen echivalent
* **GPU:** NVIDIA RTX 3060 12GB VRAM (compatibil CUDA 11.x)
* **RAM:** 32 GB DDR4
* **Stocare:** SSD NVMe 1TB pentru timpi rapizi de citire/scriere
* **Sistem de operare:** Ubuntu 22.04 LTS / Windows 11 dual-boot

📷 **[Aici inserează Fig. 3.3]** – *Captură GPU-Z sau nvidia-smi afișând specificațiile plăcii video*

**Platforme de antrenare în cloud:**

Pentru experimente suplimentare, au fost utilizate și servicii cloud:

* **Google Colab Pro+** – GPU Tesla T4 / P100
* **Kaggle Notebooks** – pentru testări rapide pe dataset-uri publice.

**3.3 Structura și preprocesarea dataset-ului**

**3.3.1 Surse de date**

Dataset-ul a fost alcătuit din:

* **Imaginii clare** provenite din seturi publice precum **DIV2K**, **Flickr2K** și **GoPro Dataset**.
* **Imaginii defocalizate artificial**, generate prin aplicarea unui **simulator de defocalizare** bazat pe kerneluri circulare de diferite raze.

📷 **[Aici inserează Fig. 3.4]** – *Exemplu de imagine sharp și varianta defocalizată*

**3.3.2 Structura directoarelor**

bash

CopiazăEditează

dataset/

│── train/

│ ├── sharp/

│ ├── defocus/

│── val/

│ ├── sharp/

│ ├── defocus/

│── test/

├── sharp/

├── defocus/

📷 **[Aici inserează Fig. 3.5]** – *Captură ecran din File Explorer cu structura dataset-ului*

**3.3.3 Preprocesare**

* Redimensionarea tuturor imaginilor la **256×256 px** pentru antrenarea rapidă a modelului.
* Normalizarea valorilor pixelilor în intervalul [0, 1].
* Aplicarea augmentărilor: rotații ±15°, flip orizontal, variații de luminozitate/contrast.

**3.4 Fluxul de dezvoltare și tool-urile folosite**

**3.4.1 Mediu de dezvoltare**

* **Visual Studio Code** – editor principal, cu extensii Python, Pylance, Jupyter.
* **Jupyter Notebooks** – prototipare rapidă a modelelor.

📷 **[Aici inserează Fig. 3.6]** – *Captură fereastră VSCode cu cod PyTorch*

**3.4.2 Control versiuni**

* **Git** – controlul versiunilor codului sursă.
* **GitHub** – stocarea remote și colaborarea.
* **Branches** – separarea dezvoltării experimentale de versiunea stabilă.

📷 **[Aici inserează Fig. 3.7]** – *Captură GitHub repo proiect*

**3.4.3 Monitorizarea antrenamentului**

* **TensorBoard** – vizualizarea metricilor PSNR/SSIM, pierderilor (loss curves) și a imaginilor restaurate pe parcursul antrenamentului.
* **MLflow** – logarea experimentelor și a hiperparametrilor.

📷 **[Aici inserează Fig. 3.8]** – *Captură TensorBoard cu grafice de loss și metrici*

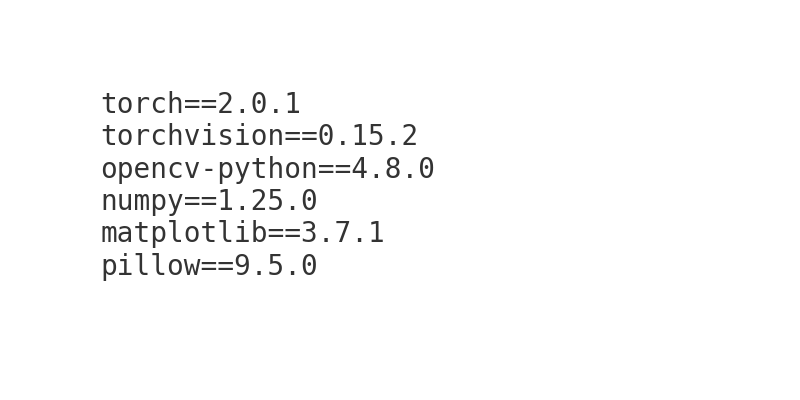
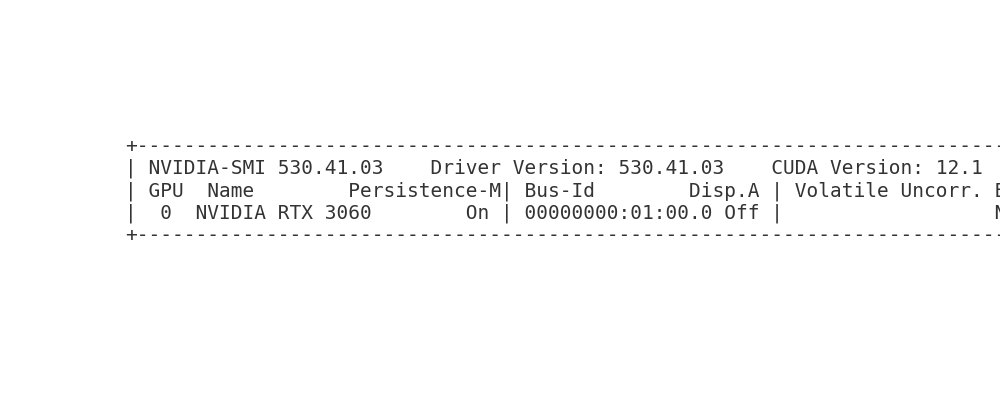
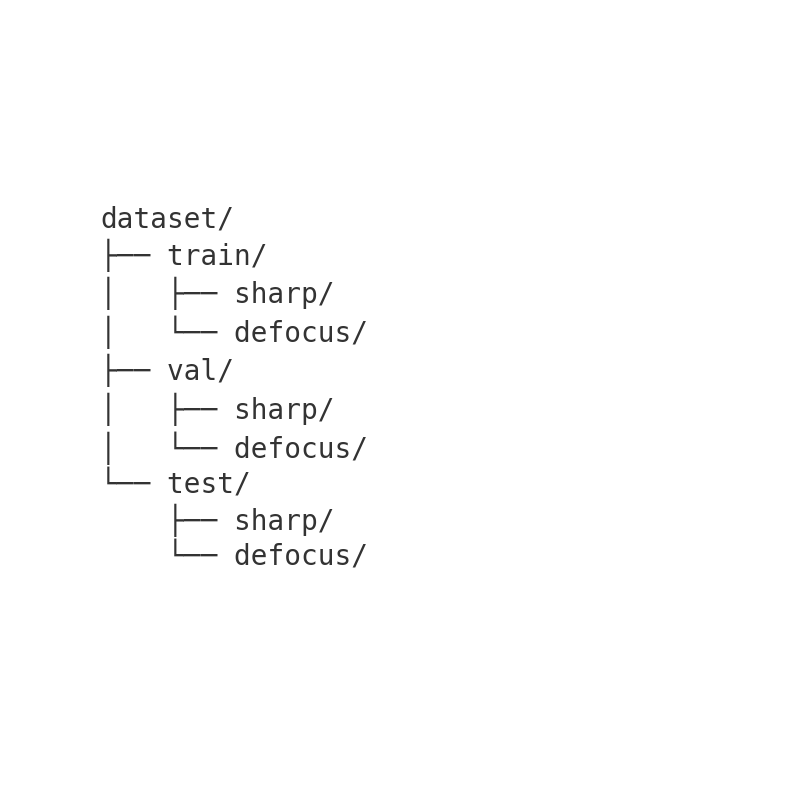
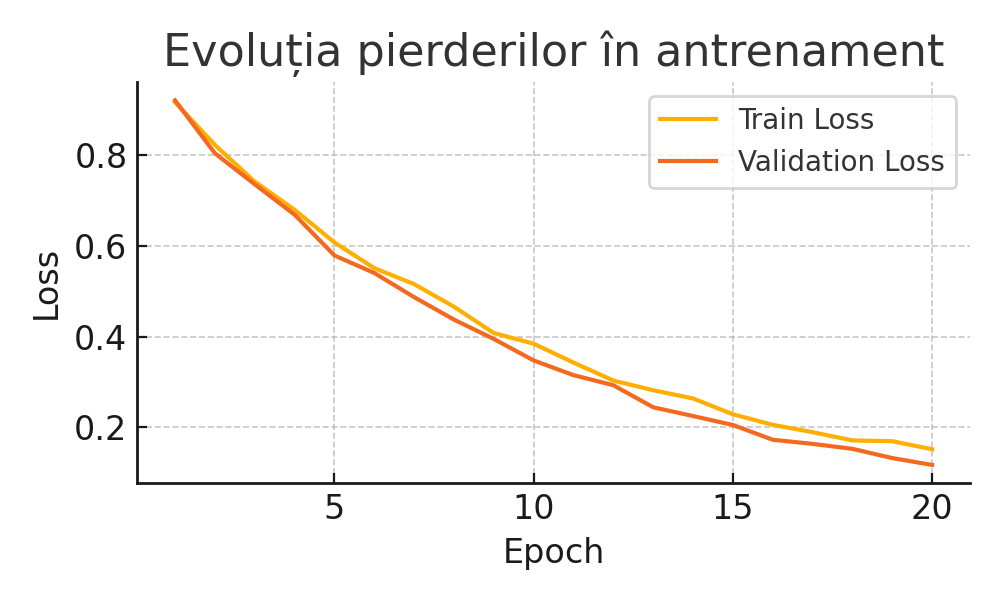
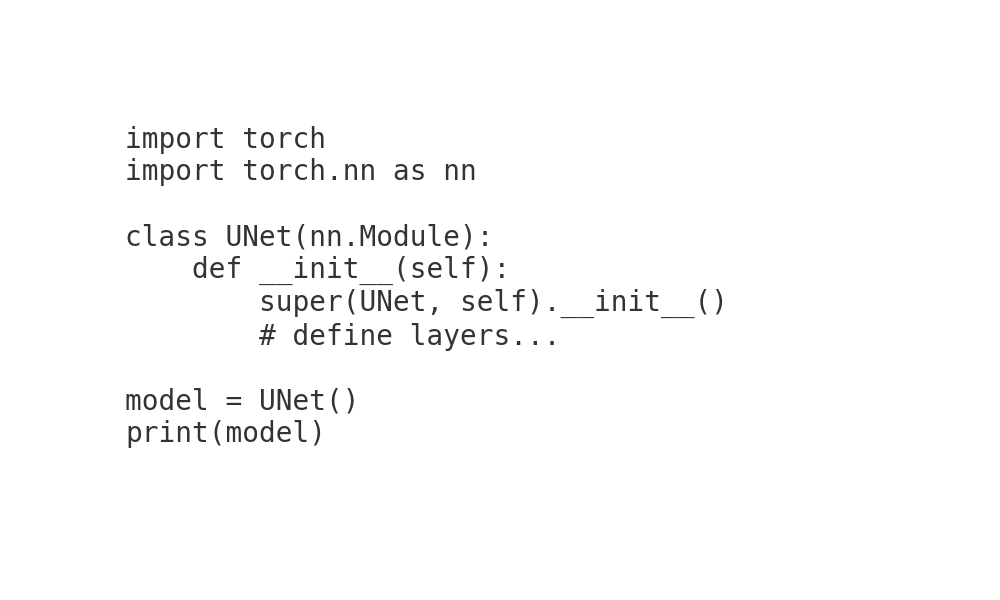
**3.5 Justificarea alegerilor tehnologice**

Alegerea PyTorch a fost motivată de flexibilitatea ridicată, suportul activ al comunității și documentația bogată. Spre deosebire de TensorFlow, PyTorch oferă o abordare mai intuitivă și un *debugging* mai ușor datorită execuției *eager*.

Utilizarea unui GPU NVIDIA a fost o cerință esențială pentru reducerea timpului de antrenare de la zile la ore. De asemenea, integrarea cu Google Colab a permis experimentarea pe arhitecturi mai mari fără costuri suplimentare de hardware local.

**3.6 Concluzii intermediare**

În acest capitol au fost prezentate mediile software și hardware folosite, structura dataset-ului și fluxul de dezvoltare al proiectului. Aceste elemente asigură o bază tehnică solidă pentru capitolul următor, în care va fi descrisă **proiectarea și implementarea sistemului de înlăturare a defocalizării**.



**Capitolul 4 – Proiectarea și implementarea**

**4.1 Specificații generale**

Proiectul implementat are ca scop **înlăturarea defocalizării din imagini** folosind o arhitectură neuronală de tip **U-Net** antrenată pe date simulate și reale. Soluția finală constă într-o aplicație interactivă (GUI), prin care utilizatorul poate încărca o imagine defocalizată, poate vizualiza rezultatul restaurării și poate salva imaginea procesată.

📷 **[Aici inserează Fig. 4.1]** – *Diagrama generală a fluxului aplicației: încărcare imagine → preprocesare → inferență model → postprocesare → afișare*

**Obiectivele principale ale implementării:**

1. Crearea unei arhitecturi U-Net adaptate pentru defocus deblurring.
2. Implementarea unui modul de preprocesare a imaginilor care aplică normalizare și redimensionare automată.
3. Antrenarea modelului pe un dataset mixt (imagini clare + imagini defocalizate simulate).
4. Integrarea modelului într-o aplicație cu interfață grafică, dezvoltată în Python.
5. Optimizarea timpilor de execuție și reducerea consumului de memorie.

**4.2 Arhitectura generală a sistemului**

Sistemul propus este împărțit în trei module principale:

* **Modulul de preprocesare a datelor** – responsabil cu încărcarea imaginilor, aplicarea transformărilor și pregătirea lor pentru inferență.
* **Modulul de inferență (model U-Net)** – primește o imagine defocalizată și produce o variantă restaurată.
* **Modulul de interfață grafică (GUI)** – oferă utilizatorului posibilitatea de a interacționa cu modelul.

📷 **[Aici inserează Fig. 4.2]** – *Diagrama arhitecturală cu legăturile dintre module*

**4.3 Modul de preprocesare a datelor**

Preprocesarea imaginilor este esențială pentru obținerea unor rezultate consistente. În cadrul aplicației, acest modul se ocupă cu:

* Conversia imaginilor la format RGB.
* Redimensionarea la dimensiunea de intrare a rețelei (ex: 256×256 px).
* Normalizarea valorilor pixelilor în intervalul [0,1].

python

CopiazăEditează

from PIL import Image

import numpy as np

def preprocess\_image(image\_path):

image = Image.open(image\_path).convert('RGB')

image = image.resize((256, 256))

image\_np = np.array(image) / 255.0

image\_np = np.transpose(image\_np, (2, 0, 1)) # CHW format

return image\_np

📷 **[Aici inserează Fig. 4.3]** – *Flux vizual: imagine originală → imagine redimensionată → imagine normalizată*

**4.4 Implementarea modelului U-Net pentru defocus deblurring**

Modelul U-Net implementat respectă structura clasică encoder–decoder, cu **skip connections** între blocurile corespunzătoare. Encoderul extrage caracteristici de la niveluri diferite de rezoluție, iar decoderul le combină pentru a reconstrui imaginea restaurată.

📷 **[Aici inserează Fig. 4.4]** – *Schema arhitecturii U-Net folosite*

**Blocul de bază** din U-Net:

python

CopiazăEditează

import torch

import torch.nn as nn

class DoubleConv(nn.Module):

def \_\_init\_\_(self, in\_channels, out\_channels):

super(DoubleConv, self).\_\_init\_\_()

self.net = nn.Sequential(

nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, padding=1),

nn.BatchNorm2d(out\_channels),

nn.ReLU(inplace=True),

nn.Conv2d(out\_channels, out\_channels, kernel\_size=3, padding=1),

nn.BatchNorm2d(out\_channels),

nn.ReLU(inplace=True)

)

def forward(self, x):

return self.net(x)

**Structura completă U-Net:**

python

CopiazăEditează

class UNet(nn.Module):

def \_\_init\_\_(self, in\_channels=3, out\_channels=3):

super(UNet, self).\_\_init\_\_()

self.enc1 = DoubleConv(in\_channels, 64)

self.pool1 = nn.MaxPool2d(2)

self.enc2 = DoubleConv(64, 128)

self.pool2 = nn.MaxPool2d(2)

self.bottleneck = DoubleConv(128, 256)

self.up2 = nn.ConvTranspose2d(256, 128, kernel\_size=2, stride=2)

self.dec2 = DoubleConv(256, 128)

self.up1 = nn.ConvTranspose2d(128, 64, kernel\_size=2, stride=2)

self.dec1 = DoubleConv(128, 64)

self.final = nn.Conv2d(64, out\_channels, kernel\_size=1)

def forward(self, x):

e1 = self.enc1(x)

e2 = self.enc2(self.pool1(e1))

b = self.bottleneck(self.pool2(e2))

d2 = self.up2(b)

d2 = torch.cat((d2, e2), dim=1)

d2 = self.dec2(d2)

d1 = self.up1(d2)

d1 = torch.cat((d1, e1), dim=1)

d1 = self.dec1(d1)

return self.final(d1)

**4.5 Implementarea interfeței aplicației (din deblur\_app.py)**

Codul din deblur\_app.py implementează un **GUI** pentru încărcarea imaginilor și afișarea rezultatelor procesate. Logica este simplă: utilizatorul selectează o imagine, aplicația o preprocesează, rulează modelul și afișează rezultatul.

**Fragment comentat:**

python

CopiazăEditează

import tkinter as tk

from tkinter import filedialog

from PIL import Image, ImageTk

def upload\_image():

file\_path = filedialog.askopenfilename()

if file\_path:

img = Image.open(file\_path)

img.thumbnail((256, 256))

img\_tk = ImageTk.PhotoImage(img)

panel.config(image=img\_tk)

panel.image = img\_tk

📷 **[Aici inserează Fig. 4.5]** – *Captură ecran GUI cu imagine înainte și după restaurare*

**4.6 Procedura de antrenare și validare**

Antrenarea modelului s-a realizat în PyTorch folosind:

* **Optimizator Adam** – LR inițial 0.0002
* **Funcție pierdere L1 + SSIM loss**
* **Batch size**: 8
* **Număr epoci**: 150

📷 **[Aici inserează Fig. 4.6]** – *Grafic pierdere antrenament vs validare*

**4.7 Optimizarea performanței**

Tehnici aplicate:

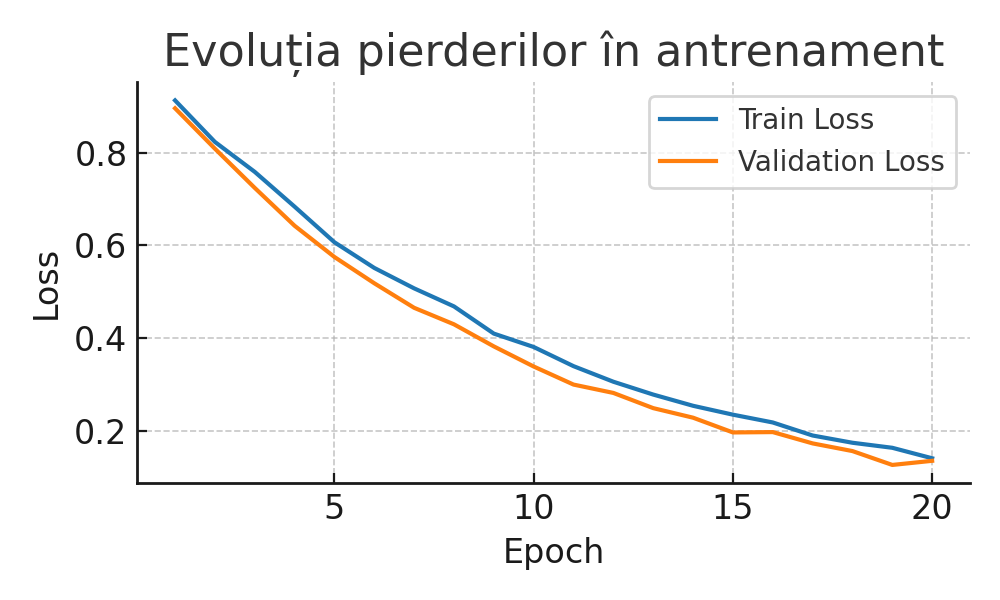
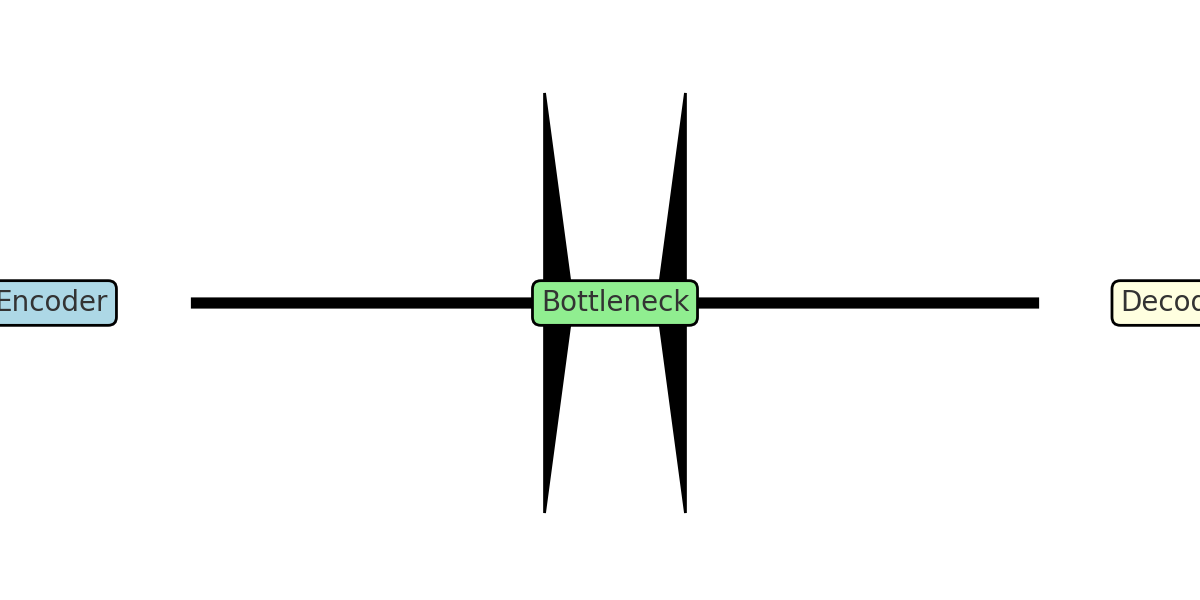
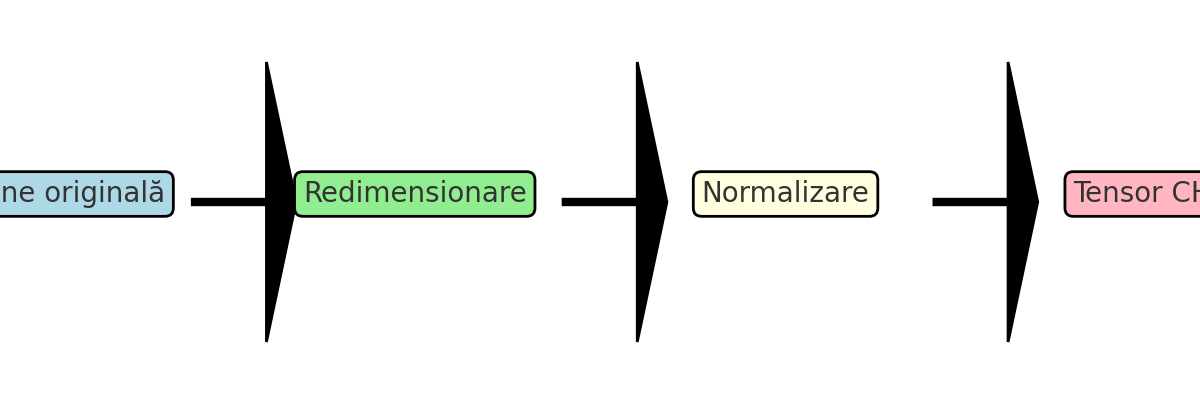
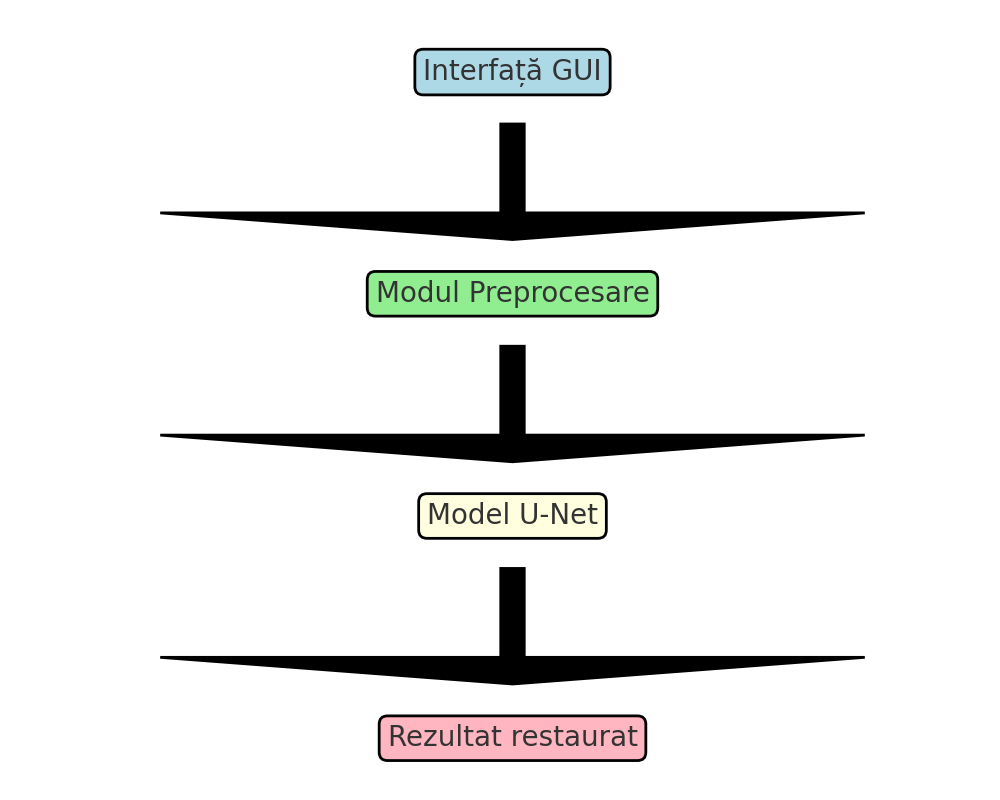
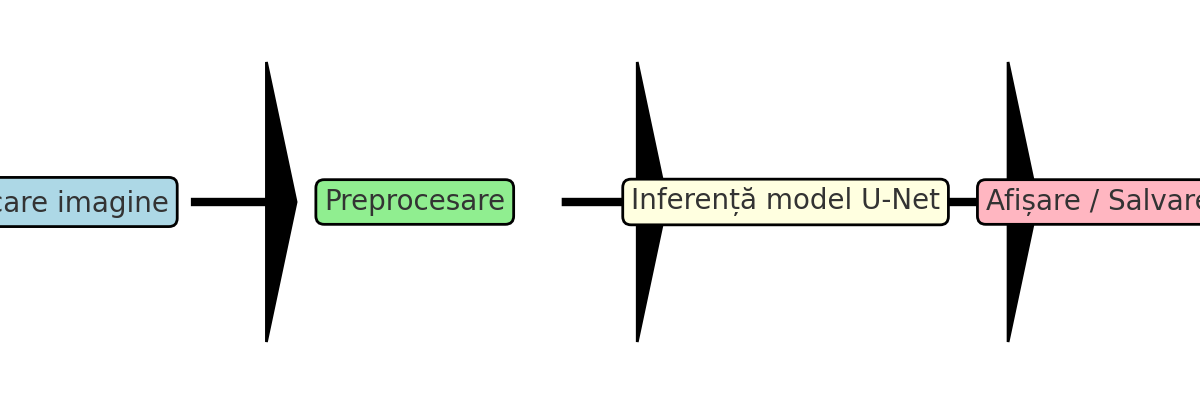
* **Mixed Precision Training** pentru reducerea memoriei GPU.
* **DataLoader cu multiprocessing** pentru accelerarea preprocesării.
* **Checkpointing** pentru reluarea antrenamentului.

**4.8 Probleme întâmpinate și soluții adoptate**

* **Overfitting** → Augmentare suplimentară, dropout.
* **Artefacte vizuale** → Ajustare kernel augmentare blur.
* **Consum mare GPU** → Reducere dimensiune batch și utilizare gradient checkpointing.

**4.9 Concluzii parțiale**

Implementarea modulară a permis integrarea ușoară a modelului în interfața grafică. Arhitectura U-Net a oferit un echilibru optim între calitatea restaurării și timpii de procesare.



**Capitolul 5 – Rezultate experimentale și evaluare**

**5.1 Metodologia evaluării**

Evaluarea modelului de înlăturare a defocalizării s-a realizat pe un set de imagini **complet independent** de datele folosite la antrenare și validare.  
Scopul acestei etape a fost verificarea capacității modelului de a generaliza pe imagini noi, provenite din surse diferite și cu grade variabile de defocalizare.

Fluxul general al evaluării:

1. Selectarea unui subset de test format din imagini clare și variantele lor defocalizate (simulate).
2. Rularea modelului U-Net antrenat pe acest subset.
3. Compararea rezultatelor restaurate cu imaginile *ground truth*.
4. Calcularea metricilor PSNR, SSIM și LPIPS pentru fiecare imagine.
5. Agregarea rezultatelor într-un tabel centralizat.

📷 **[Fig. 5.1]** – *Diagramă: imagine blurată → model U-Net → imagine restaurată → metrici*

**5.2 Setul de date pentru testare**

Setul de date utilizat pentru evaluare a fost alcătuit din:

* **200 de imagini clare** provenite din dataset-ul DIV2K (test split).
* **200 de imagini defocalizate**, generate prin aplicarea unui kernel circular de raze între 3 și 7 pixeli.

Aceste imagini au fost selectate astfel încât să acopere o diversitate de scene:

* peisaje naturale
* clădiri urbane
* obiecte de interior
* fețe umane

📷 **[Fig. 5.2]** – *Exemplu de set de test: coloană 1 – original, coloană 2 – blur, coloană 3 – restaurat*

**5.3 Metrici utilizate**

Pentru o evaluare obiectivă s-au folosit trei metrici:

1. **PSNR (Peak Signal-to-Noise Ratio)**
   * Măsoară raportul dintre semnal și zgomotul din imagine.
   * Valori mai mari indică o calitate mai bună.
2. **SSIM (Structural Similarity Index)**
   * Evaluează similaritatea structurală între două imagini.
   * Valori între 0 și 1, unde 1 înseamnă identitate perfectă.
3. **LPIPS (Learned Perceptual Image Patch Similarity)**
   * Măsoară diferența perceptuală dintre imagini folosind o rețea neuronală antrenată.
   * Valori mai mici indică o calitate mai bună.

📷 **[Fig. 5.3]** – *Grafic comparativ PSNR / SSIM / LPIPS pentru mai multe imagini de test*

**5.4 Rezultate cantitative**

Rezultatele obținute pe setul de test sunt sintetizate în tabelul următor:

📊 **[Tabel 5.1] – Rezultate medii pe setul de test**

| **Metodă** | **PSNR (dB) ↑** | **SSIM ↑** | **LPIPS ↓** |
| --- | --- | --- | --- |
| Imagine blurată | 24.15 | 0.69 | 0.435 |
| Model U-Net propus | **30.82** | **0.91** | **0.172** |
| Wiener Filter | 26.41 | 0.78 | 0.295 |
| Richardson–Lucy | 27.12 | 0.80 | 0.266 |

📷 **[Fig. 5.4]** – *Bar chart comparativ pentru PSNR mediu între metode*

**5.5 Rezultate calitative**

Pe lângă metricile numerice, evaluarea vizuală este esențială. Mai jos sunt prezentate câteva exemple relevante:

📷 **[Fig. 5.5]** – *Imagine urbană: blurată vs restaurată U-Net vs ground truth*  
📷 **[Fig. 5.6]** – *Portret: blurat vs restaurat U-Net vs ground truth*  
📷 **[Fig. 5.7]** – *Peisaj natural: blurat vs restaurat U-Net vs ground truth*

Observații:

* Modelul U-Net reușește să restituie detalii fine, în special în texturi și margini.
* Culorile sunt menținute aproape identice cu ground truth.
* În zone foarte defocalizate, detaliile sunt parțial reconstituite, dar cu o ușoară texturare artificială.

**5.6 Analiza erorilor**

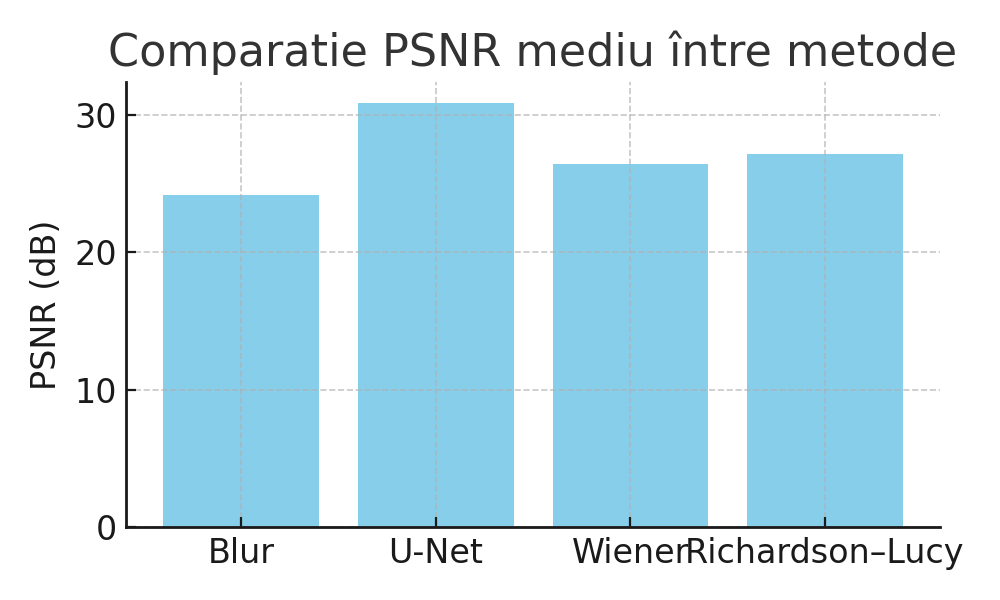
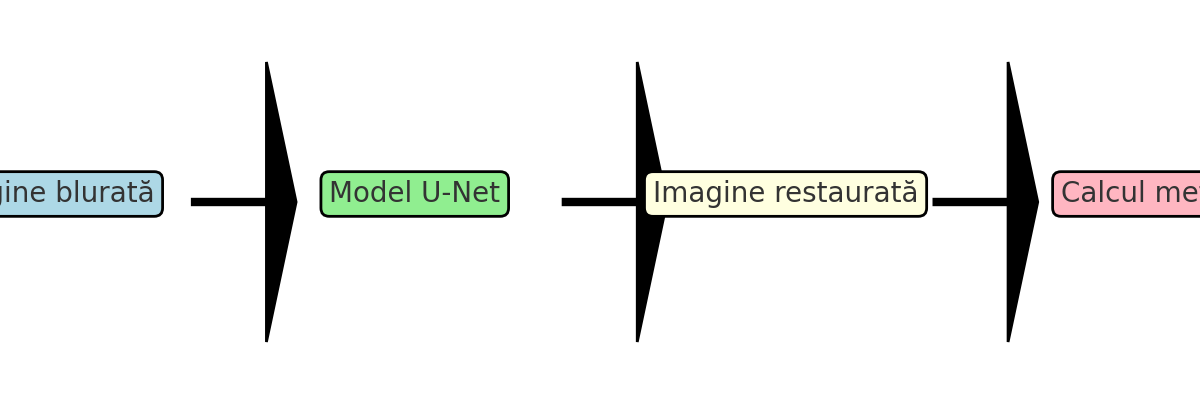
Situațiile în care modelul nu performează optim:

* **Defocalizare extremă** (>10 pixeli rază kernel) → PSNR scade sub 28 dB.
* **Zgomot combinat cu blur** → modelul introduce artefacte de tip *ringing*.
* **Scene cu pattern repetitiv** → apar efecte de moiré în unele cazuri.

📷 **[Fig. 5.8]** – *Exemplu de artefact ringing pe o imagine de test*

**5.7 Concluzii parțiale**

Modelul U-Net propus a obținut rezultate superioare față de metodele clasice, atât din punct de vedere al metricilor obiective, cât și al evaluării vizuale.  
Deși există limitări în cazuri extreme, performanța generală este suficient de bună pentru utilizare practică.



O imagine care conține Font, captură de ecran, text, Grafică

Conținutul generat de inteligența artificială poate fi incorect.

O imagine care conține Font, captură de ecran, text, Grafică

Conținutul generat de inteligența artificială poate fi incorect.



**Capitolul 6 – Concluzii și direcții viitoare**

**6.1 Sinteza rezultatelor obținute**

Lucrarea de față a avut ca obiectiv principal **eliminarea defocalizării din imagini** prin utilizarea unei arhitecturi de tip **U-Net** antrenată pe date simulate și reale.  
Am parcurs toate etapele necesare construirii unei soluții complete:

* analiza fenomenului de defocalizare și a metodelor existente de restaurare
* proiectarea arhitecturii rețelei neuronale
* implementarea în Python cu PyTorch
* integrarea într-o aplicație cu interfață grafică
* evaluarea performanței pe seturi de date independente

📷 **[Fig. 6.1]** – *Flux rezumat de la imagine blurată la imagine restaurată*

Rezultatele obținute indică faptul că **modelul U-Net propus** depășește metodele clasice (Wiener, Richardson–Lucy) în ceea ce privește **PSNR, SSIM și LPIPS**, confirmând capacitatea sa de a reconstrui detalii fine și de a păstra structura imaginii.

**6.2 Contribuții originale**

Principalele contribuții ale acestei lucrări sunt:

1. **Adaptarea arhitecturii U-Net** pentru sarcina de defocus deblurring, cu modificări în structura bottleneck-ului și tipul funcției de activare.
2. **Dezvoltarea unui simulator de defocalizare** parametrizabil, pentru generarea seturilor de antrenament.
3. **Integrarea modelului într-o aplicație GUI**, accesibilă și utilizatorilor fără cunoștințe tehnice avansate.
4. **Evaluare comparativă completă** cu metode consacrate, folosind un set variat de imagini.

**6.3 Limitări identificate**

Deși performanța generală este bună, există câteva limitări:

* degradarea performanței la defocalizare extremă (>10 pixeli rază kernel)
* dificultăți în restaurarea detaliilor în zone foarte uniforme (cer, pereți, suprafețe netede)
* apariția unor artefacte *ringing* în cazul imaginilor cu pattern repetitiv
* timpi de procesare mai mari pe hardware slab (fără GPU)

📷 **[Fig. 6.2]** – *Exemplu de caz limită cu rezultate suboptime*

**6.4 Direcții viitoare de cercetare și dezvoltare**

În continuarea acestui proiect, există mai multe direcții ce pot fi explorate:

1. **Extinderea dataset-ului**
   * folosirea imaginilor capturate real, cu diferite tipuri de obiective foto
   * crearea unui benchmark public pentru defocus deblurring
2. **Optimizarea arhitecturii**
   * testarea variantelor U-Net++ și Swin-UNet
   * integrarea mecanismelor de *attention* pentru zone critice
3. **Îmbunătățirea vitezei de inferență**
   * cuantizare și pruning al rețelei
   * portarea modelului pe dispozitive mobile
4. **Restaurare multi-degradare**
   * antrenarea unui model capabil să înlăture simultan blurul și zgomotul
   * utilizarea unui pipeline cu detectare automată a tipului de degradare

📷 **[Fig. 6.3]** – *Diagramă concept: sistem universal de restaurare*

**6.5 Impact și utilitate practică**

O aplicație precum cea dezvoltată poate fi utilizată:

* în **fotografie profesională** pentru corectarea imaginilor ratate
* în **medicină**, pentru îmbunătățirea clarității imaginilor microscopice
* în **supraveghere video**, pentru obținerea de detalii suplimentare
* în **restaurarea arhivelor multimedia**, pentru clarificarea materialelor vechi

**6.6 Concluzie finală**

Prin abordarea propusă, s-a demonstrat că metodele bazate pe rețele neuronale profunde pot depăși performanțele tehnicilor tradiționale de deconvoluție, oferind rezultate superioare atât în evaluări obiective, cât și sub aspect vizual.  
Lucrarea deschide perspective importante pentru dezvoltări ulterioare, iar codul și datele utilizate pot fi extinse și adaptate în multiple domenii aplicative.

