Anexa 5 la Procedura de înscriere on-line la examenele de finalizare a studiilor

LUCRARE DE DISERTAȚIE

Îndrumător proiect/Coordonator științific,
Conf. dr. ing. Dan MUNTEANU

Absolvent,
Cristian-Andrei BĂRGAN

Galați
Anul 2025

PROGRAMUL DE STUDII: Tehnologii Informatice Avansate

Aplicație pentru recunoasterea încărcării pubelelor de gunoi din imagini

Îndrumător proiect/Coordonator științific,
Conf. dr. ing. Dan MUNTEANU

Absolvent,
Cristian-Andrei BĂRGAN

Galați
Anul 2025

CUPRINS

Capitolul 1. Analiza problemei 2

Capitolul 2. Specificarea cerințelor 3

2.1 Navigarea în aplicație 3

2.1 Clasificarea imaginilor 3

2.2 Detecție în imagini 3

2.3 Descrierea pe scurt a aplicației 4

Capitolul 3. Proiectarea sistemului 5

3.1 Procesarea informațiilor de către aplicație 5

3.1.1 Procesarea imaginilor în clasificare 5

3.1.2 Procesarea imaginilor în procesul de detecție 6

3.2 Cazurile de utilizare ale aplicației 6

3.2.1 Clasificarea imaginilor 6

3.2.2 Detectarea de coșuri în imagini 7

Capitolul 4. Implementarea cerințelor 8

4.1 Tehnologii utilizate 8

4.2 Crearea și antrenarea rețelelor neuronale 9

4.2.1 Rețele neuronale de clasificare a imaginilor 9

4.2.2 Rețelele neuronale de detectare a obiectelor 15

Capitolul 5. Testarea rețelelor 31

5.1 Rețelele de clasificare a imaginilor 31

5.1.1 CNN-G1 31

5.1.2 CNN-G2 32

5.2 Rețelele de detectare a obiectelor în imagini 32

5.2.1 YOLO11n 33

5.2.2 YOLO11s 36

5.2.1 YOLO11m 39

Capitolul 6. Descriere aplicației 43

6.1 Meniul și paginile aplicației 43

6.2 Detalierea procesului de clasificare al imaginilor 45

6.3 Detecția în imagini 46

Concluzii 49

Bibliografie 50

CAPITOLUL 1. ANALIZA PROBLEMEI

Problema poluării la nivel global a ajuns la un nivel alarmant, care nu mai poate fi ignorat [15]. Combaterea acesteia a atras eforturi însemnate inclusiv din partea sectorului tehnologiilor avansate precum inteligența artificială [11].

Există mai multe proiecte în care programatorii și-au dorit să dezvolte rețele de inteligență artificială „deep-learning" pentru a clasifica tipurile de gunoaie [12, 13]. O altă direcție de acțiune este eficientizarea gestionării deșeurilor.

În România există zone unde s-a investit în eficientizarea colectării deșeurilor. Un articol din 2024 [15] descrie situația din Târgu Lăpuș, din județul Maramureș care are insule digitalizate. Cu ajutorul unui card sau folosind un cod QR oamenii pot utiliza pubelele intelige pentru a depozita separat deșeurile.

Organizare logistică a transportării deșeurilor poate face, din punctul meu de vedere, întregul procesul de reciclare în sine mai eficient și mai curat prin emiterea mai puținor substanțe dăunătoare atât de la mașinile clasice, cât și de la cele electrice care au nevoie de energine electrică.

În cadrul acestei lucrări de disertație mi-am propus să creez o aplicație simplu de utilizat, care este capabilă să clasifice coșurile de gunoi care sunt mai pline de 60% și pe cele goale, care au mai puțin.

La baza funcționării acestora stau modele de inteligență artificială care au fost antrenate pe un set de imagini descărcat de pe platforma Kaggle [14].

CAPITOLUL 2. SPECIFICAREA CERINȚELOR

Aplicaţia are drept scop analiza imaginilor care conţin containere de gunoi. În cadrul acestei lucrări, aplicaţia a fost concepută cu scopul de a pune în valoare cele mai bune reţele neuronale convoluţionale.

În cadrul aplicaţiei va exista un singur tip de utilizator, acela de utilizator ce se poate folosi de aceasta pentru a clasifica imagini şi a detecta obiecte în imagini.

2.1 Navigarea în aplicaţie

Pentru a realiza navigarea, aplicaţia va avea un meniu lateral care va permite utilizatorilor să aleagă către ce pagini vor să se îndrepte. Vor fi create trei optiuni: pagina „Home”, pagina „Clasificare”, şi pagina „Detecţie”.

2.1 Clasificarea imaginilor

Pentru a clasifica imagini utilizatorul va putea să facă următoarele operaţiuni:

- Să acceseze pagina corespunzătoare din meniu
- Să încarce una sau mai multe imagini în format JPG, JPEG sau PNG
- Să clasifice imaginile ca fiind imagini ce conţin un coş de gunoi plin (mai plin de 60%) sau unul gol (sub 60% umplere)
- Să primească drept răspuns un tabel (un dataframe Pandas) vizibil pe ecran
- Să poată descărca rezultatul clasificării imaginilor sub formă de fişier csv.

2.2 Detecţie în imagini

Pentru a susţine operaţiunea de detecţie aplicaţia va permite următoarele operaţiuni:

- Accesare pagini corespunzătoare folosind meniul
- Să încarce una sau mai multe imagini în format JPG, JPEG sau PNG
- Să detecteze în imagini coşuri de gunoi pline mai mult de 60% (coşuri pline) sau cu un grad de umplere de sub 60% (coşuri goale)
- Să poată vizualiza pe ecran rezultatele detecţiei pe fiecare imagine în parte. Coşurile vor fi încadrate într-un dreptunghi, iar pentru fiecare inferenţă va fi afişat gradul de încredere al reţelei în acea inferenţă şi clasificarea acestora în coşuri goale sau pline (dacă au un grad de umplere sub 60% sau nu) şi vor primi etichetele „empty”, respectiv „full”
- Parametrul „Intersection over Union” (iou) al inferenţei va putea fi reglat cu ajutorul unui slider. Pentru fiecare modificare a acestei valori, aplicaţia va reanaliza imaginile încărcate. Acest parametru va avea valori cuprinse între 0 şi 1 cu o precizie de două zecimale. Valoarea sa iniţială va fi 0,7.
- Parametrul „Încrederea predicţiei” va putea fi modificat tot folosind un slider, va avea valori cuprinse între 0 şi 1 cu o precizie de două zecimi. La fiecare modificare a sa, aplicaţia va reanaliza imaginile încărcate. Valoarea sa iniţială va fi de 0,7.

- Rezultatele se vor putea descărca sub forma unei arhive care conține imaginile cu obiectele detectate în interior și cu etichetele aferente descrise anterior

2.3 Descrierea pe scurt a aplicației

La deschiderea acestei aplicații, utilizatorul va fi întâmpinat de o pagină „Home” ce va conține rezultate sumare despre scopul aplicației și modul de utilizare.

CAPITOLUL 3. PROIECTAREA SISTEMULUI

3.1 Procesarea informaţiilor de către aplicaţie

Scopul principal al acestei aplicaţii este acela de a prelua imagini (date de intrare) şi de a reda informaţiile pe care utilizatorul şi le doreşte, fie că vorbim despre clasificarea acelor imagini, fie că vorbim despre afişarea unor imagini în care au fost identificate obiectele de interes (în acest caz, coşurile de gunoi).

Aplicaţia pentru a funcţiona şi pentru a procesa imaginile are nevoie de modele neuronale convoluţionale care să fie integrate în aceasta. În această secţiune vom prezenta modul în care acestea se integrează şi felul în care informaţia este transformată.

3.1.1 Procesarea imaginilor în clasificare

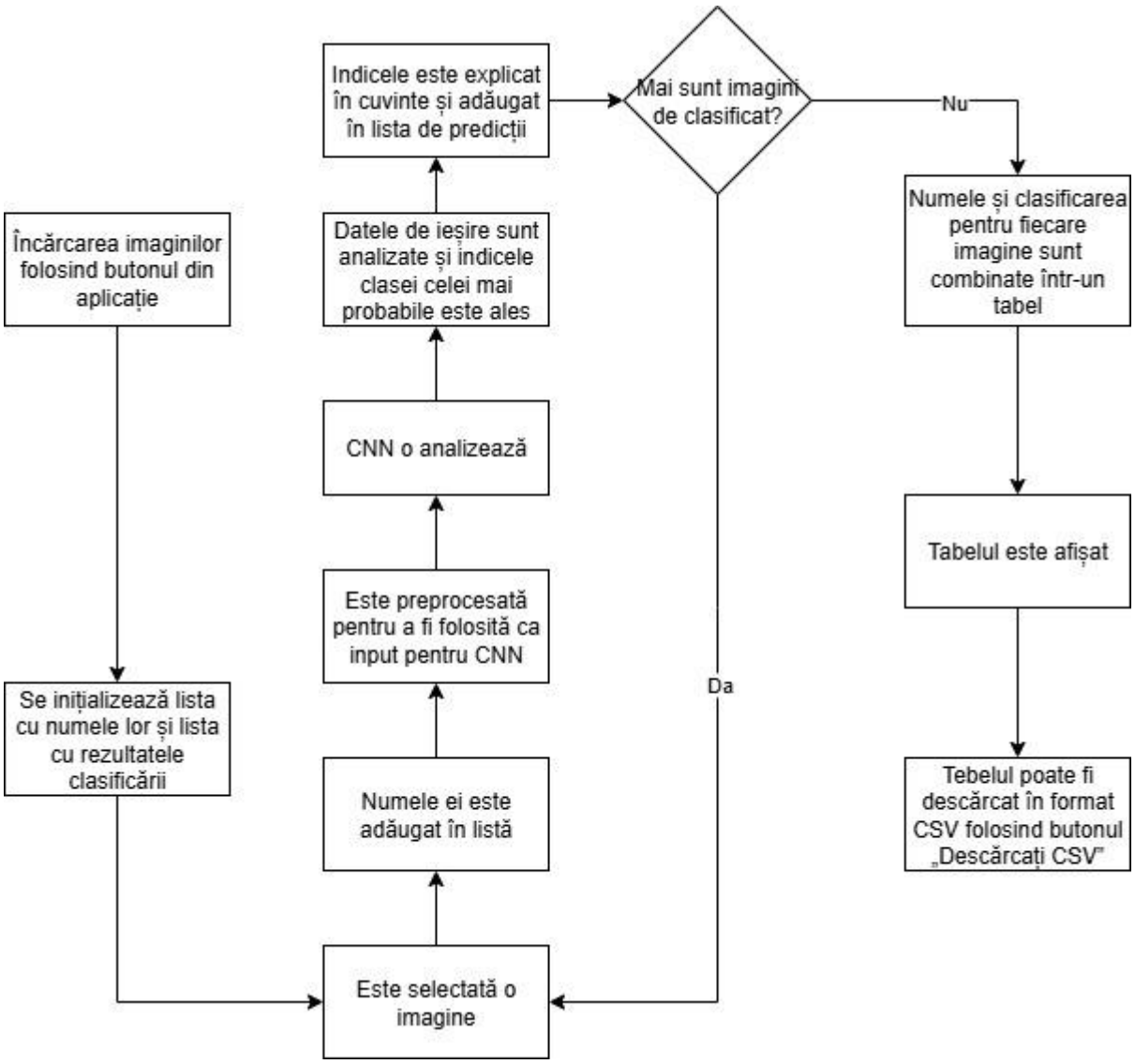

Fig 3.1 Procesul de transformare a imaginii în clasificare

Se poate observa cum fiecare imaginie este clasificată pe rând de către reţeaua neuronală convoluţională şi cum informaţiile sunt colectate „imagine cu imagine".

3.1.2 Procesarea imaginilor în procesul de detecție

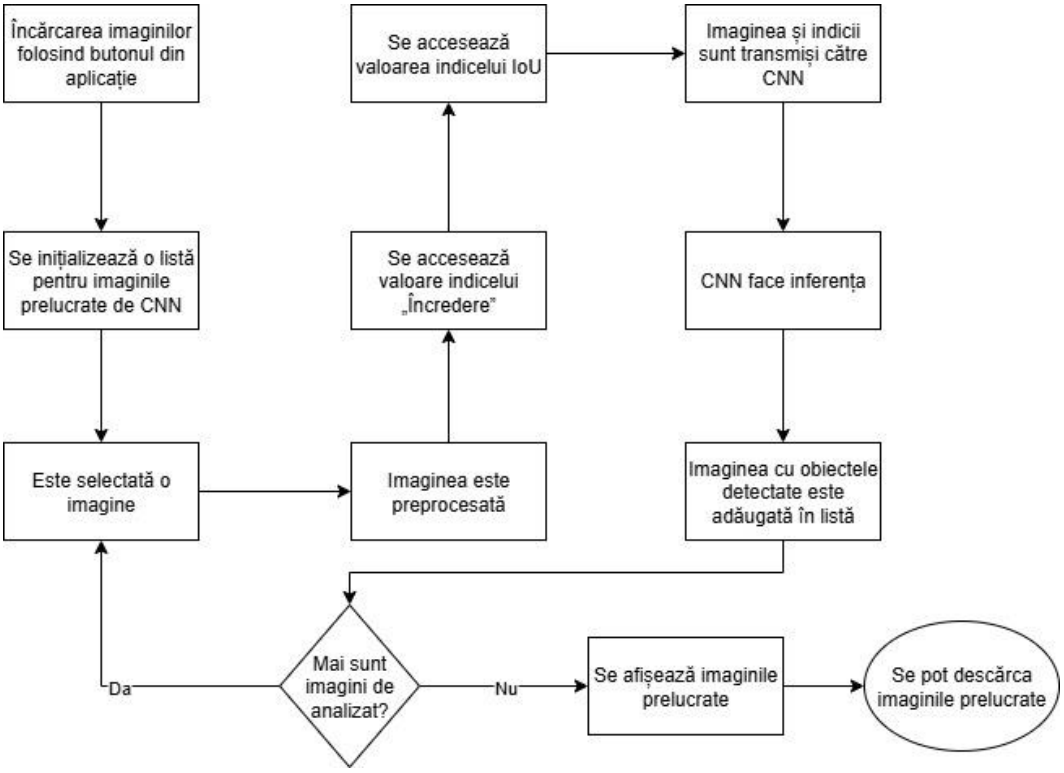

Fig. 3.2 Procesarea imaginilor în procesul de detecție

3.2 Cazurile de utilizare ale aplicației

În cadrul utilizării acestei aplicații există două cazuri de utilizare a aplicației: clasificarea de imagini ce conțin coșuri de gunoi, și detectarea coșurilor de gunoi din imagini. În acest capitol vom prezenta diagramele de funcționare relevante pentru aceste două moduri de utilizare.

3.2.1 Clasificarea imaginilor

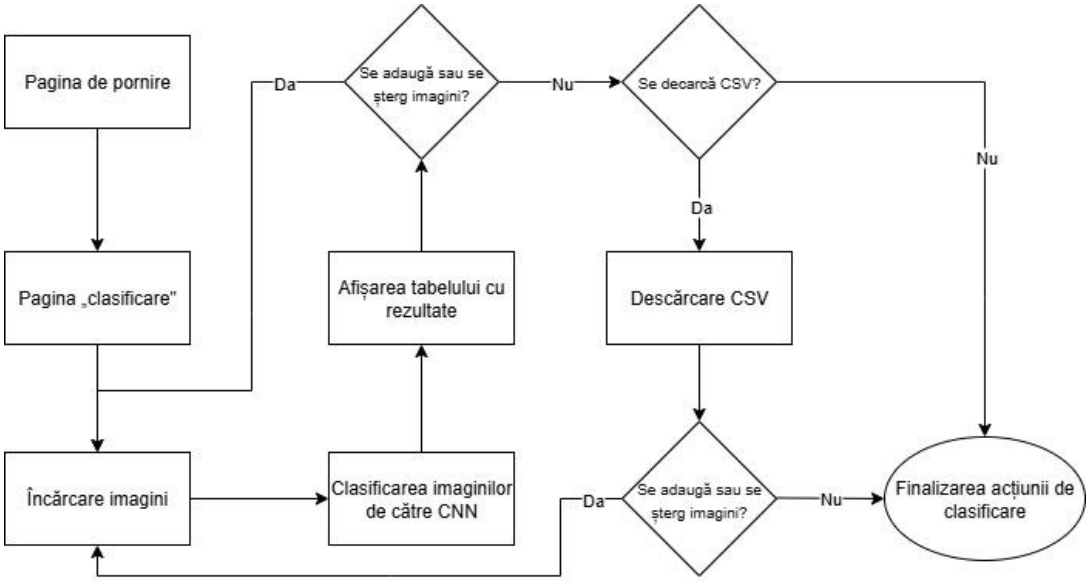

Fig. 3.3 Diagrama de utilizare pentru clasificarea de imagini

Putem observa din diagramă faptul că avem posibilitate de a încărca de mai multe ori pe parcursul procesului noi imagini. Aplicația își va reactualiza rezultatele clasificării după fiecare adăugare sau ștergere de imagini.

De asemenea, putem observa că nu este obligatoriu ca rezultatele să fie descărcate, întregul proces oferă flexibilitate și simplitate pentru utilizator.

3.2.2 Detectarea de coșuri în imagini

În cadrul procesului de detectare a coșurilor de gunoi din imagini, vom avea două elemente in plus: modificarea parametrului IoU și modificarea parametrului de încredere a infernței.

Cei doi parametrii se pot schimba ori de câte ori utilizatorul dorește acest lucru, iar rețeaua neuronală convoluțională va reîncepe de fiecare dată procesul de detectare a coșurilor de gunoi din imagini.

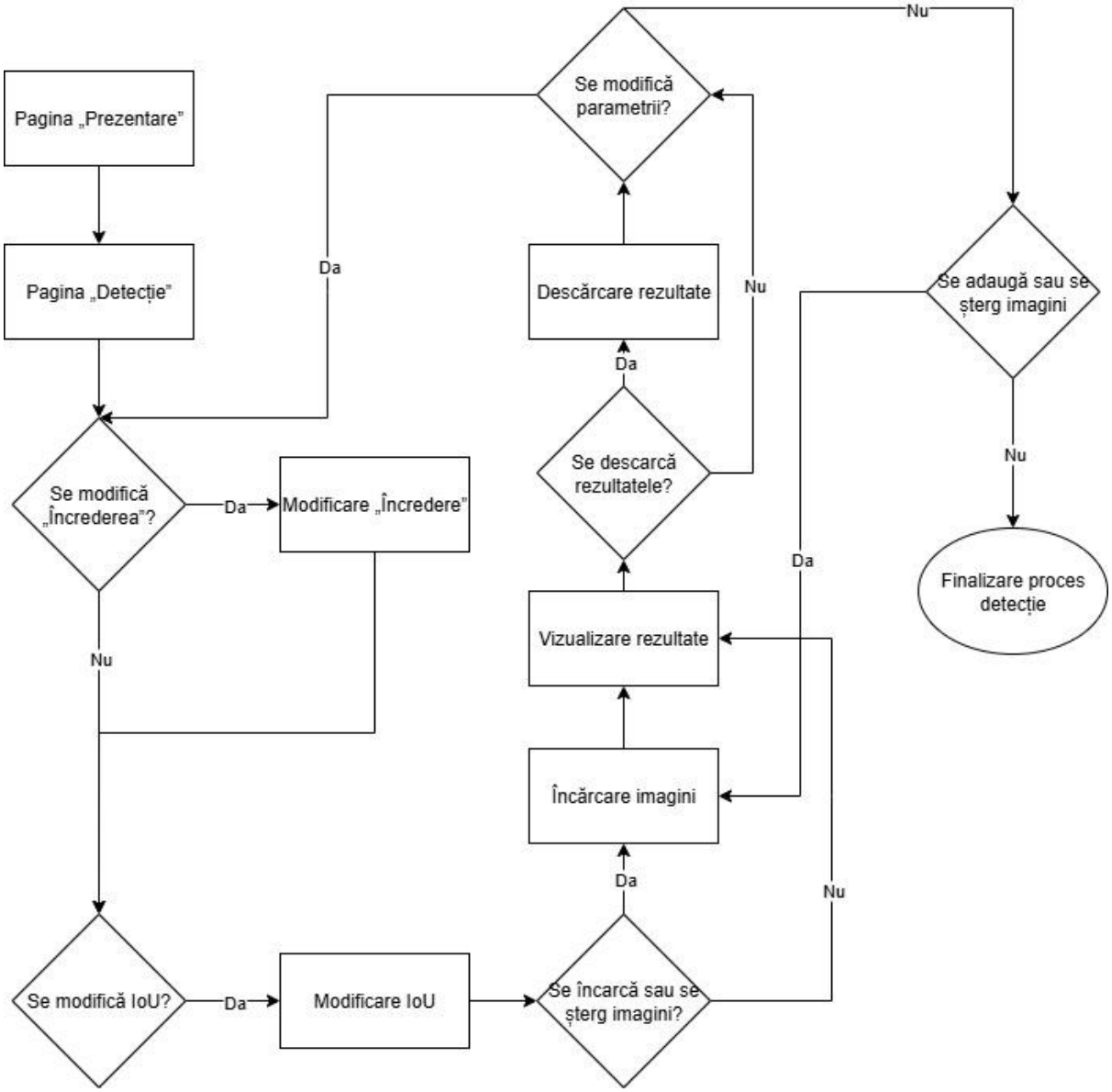

Fig 3.4 Diagrama de utilzare pentru clasificare

CAPITOLUL 4. IMPLEMENTAREA CERINȚELOR

4.1 Tehnologii utilizate

În cadrul acestui proiect s-au folosit librării și framework-uri pentru realizarea aplicației, pentru antrenarea rețelelor neuronale și pentru prezentarea mai multor metrici de performanță ale acestor rețele.

Limbajul de programare folosit la realizarea lucrării este limbajul Python. Am ales acest limbaj de programare deoarece poate fi folosit împreună cu librăriile și framework-urile necesare.

Librăriile și framework-urile folosite în acest proiect:

- Streamlit: Streamlit este un framework folosit pentru reazlizarea de aplicații în mod facil. Am ales acest framework deoarece este „open-source" și se pot realiza „aplicații bazate pe date dinamice cu numai câteva linii de cod" [1].
- Tensorflow este o platformă pentru crearea algoritmilor de „machine learning" (ML) și are instrumente de monitorizare a antrenării. Un aspect important este acela că încorporează API-ul Keras [2].
- Matplotlib: este o librărie python ce are ca scop realizarea de vizualizări grafice a datelor. Am ales-o pentru proiectul meu cu scopul de a reprezenta ușor anumici indici de antrenare și de performanță [3]
- Numpy: este o librărie care poate manipula și efectua operații complexe, în mod rapid, pe vectori multidimensionali [4].
- Keras: este un API folosit pentru antrenarea modelelor ML și, în cadrul acestui proiect, o vom folosi pentru antrenarea rețelelor neuronale de clasificare a imaginilor și va fi accesată prin Tensorflow [2,5]
- VisualKeras: este un pachet folosti pentru a vizualiza arhitectura rețelelor neuronale. Am ales folosirea acestui pachet pentru a reprezenta grafic rețelele pe care le-am construit folosind Keras [6].
- Sklearn: este o librărie de ML „open-source" care poate fi folosită pentru antrenarea de modele. Ea oferă metode de preprocesare a datelor, selectarea modelelor și evaluarea lor [7]. Am ales această librărie pentru capacitatea ei de evaluare a modelelor CNN.
- Seaborn: este o bibliotecă de vizualizare a datelor [8]. Am ales-o pentru acest proiect cu scopul de a o folosi la anumite reprezentări ale metricilor de performanță (cum ar fi matricea de confuzie).
- Ultralytics: este un framework prin care pot fi accesate modelele YOLO, care pot fi folosite pentru sarcini de „computer vision" cum ar fi: clasificare, detectare de obiecte, segemntarea de imagini. Am ales acest framework pentru a folosi modele YOLO11 pentru detectarea de coșuri de gunoi în imagini.

4.2 Crearea și antrenarea rețelelor neuronale

Pentru sarcinile de clasificare a imaginilor am antrenat mai multe rețele neuronale și le-am folosit în aplicație pe cele care au obținut cele mai bune rezultate. În această secțiune vom compara două rețele de clasificare a imaginilor și trei rețele de detectare a obiectelor. Vom prezenta informații generale despre ele, metodele de antrenare și rezultatele antrenării lor.

4.2.1 Rețele neuronale de clasificare a imaginilor

Pentru realizarea și antrenarea acestor rețele neuronale convoluționale(CNN) vom folosi framework-ul tensorflow și vom accesa API-ul Keras prin intermediul acestuia. Procesult de antrenare a CNN parcurge următorii pași:

1) Se creează un dataset: imaginile sunt preluate din folder și sunt redimensionate la 180x180 pixeli. Ele mai apoi sunt împărțite în loturi (batch-uri). Aceste imagini sunt convertite în vectori multidimensionali numiti tensori.

2) Acest prim set este împărțit în trei subseturi. Din toate loturile de imagini 70% vor fi folosite pentru antrenare, 20% sunt folosite pentru validare și 10% sunt folosite pentru testare

```
dataset_size = len(dataset)
train_split = 0.7
validation_split = 0.2
test_split = 0.1

train_size = int(dataset_size * train_split)
validation_size = int(dataset_size * validation_split)
test_size = dataset_size - train_size - validation_size

print(dataset_size, train_size, validation_size, test_size)

train_ds = dataset.take(train_size)
val_ds = dataset.skip(train_size).take(validation_size)
test_ds = dataset.skip(train_size + validation_size).take(test_size)
```

Fig 4.1 Funcția de creare a celor trei subseturi de date

3) Seturile de date sunt mai apoi optimizate pentru performanță

4) Se definesc straturile de augmentare a imaginilor (codul în Fig 4.2). Această operațiune are rolul de a face modelele să fie mai robuste (să generalizeze mai bine). Se observă în Fig 4.3 un exemplu de imagine augmentată.

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal",
                          input_shape=(img_height,
                                       img_width,
                                       3)),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.1),
    ]
)
```

Fig 4.2 Definirea straturilor de augmentare a imaginilor

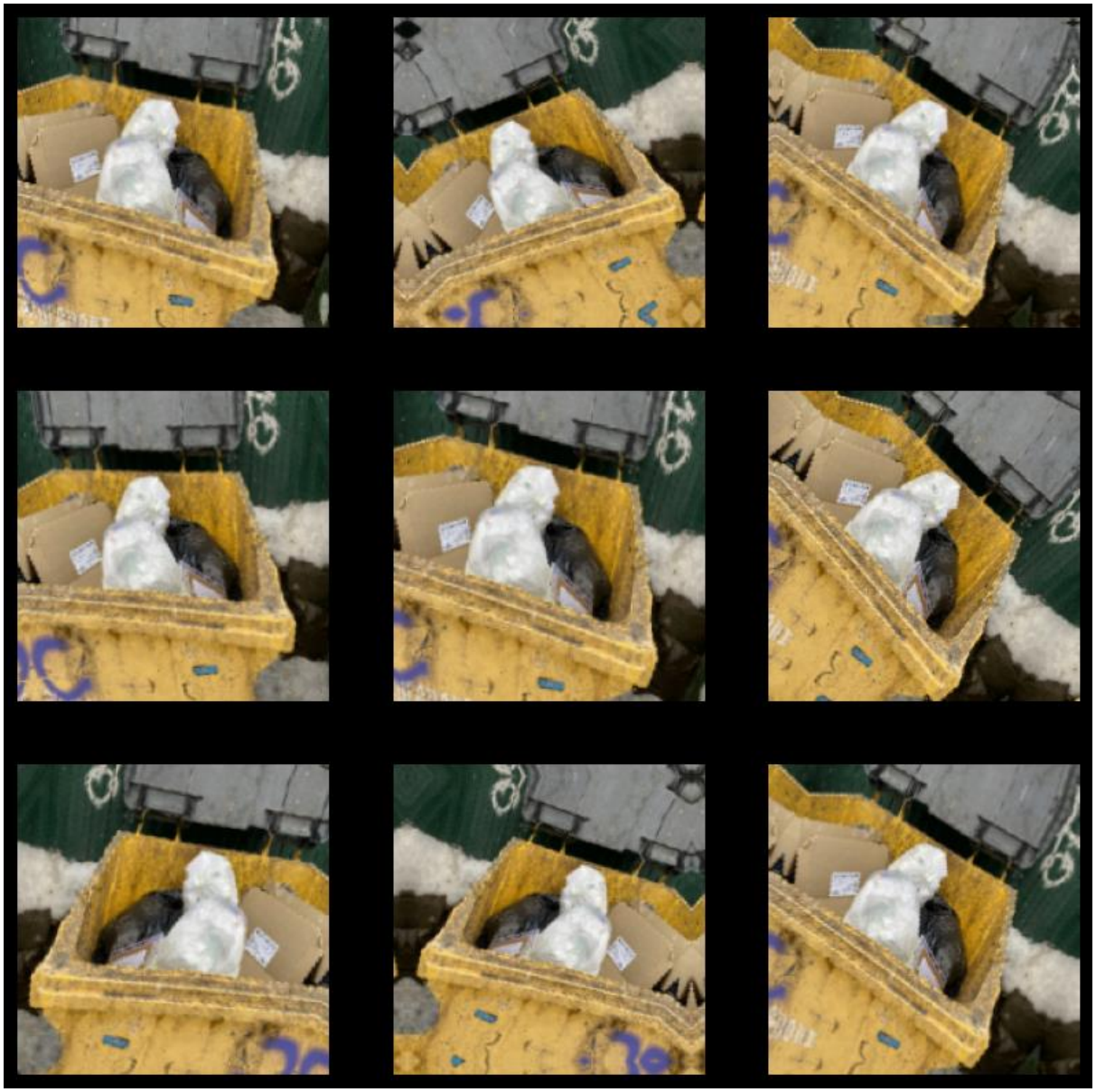

Fig 4.3 Model de imagine augmentată

5) În cazul modelelor care sunt construite, se defineste structura CNN şi se uneşte cu straturile de augmentare.

6) Modelul este compilat

7) Vizualizarea structuriilor reţelelor

8) Modelul este antrenat

9) Sunt realizate graficele ce conţin informaţii din timpul antrenării (acurateţea şi loss pentru antrenare şi validare)

În continuare vom prezenta fiecare reţea în parte împreună cu informaţiile obţinute în timpul antrenării şi validării.

I. Prima reţea convoluţională (CNN-G1)

Această reţea are mai multe straturi convoluţionale, straturi de tip MaxPooling şi straturi de tip Dense. Structura poate fi vizualizată atât în tabelul din Fig 4.4 şi în schema din Fig 4.5. Numărul total al parametrilor antrenabili este de 3.988.898.

Valorile pentru acurateţe şi loss pot fi vizualizate in Fig 4.6 unde sunt prezentate valorile pentru fiecare dintre cele 15 epoci de antrenare.

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 180, 180, 3)	0
rescaling (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 64)	0
dropout (Dropout)	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3,965,056
outputs (Dense)	(None, 2)	258

Fig 4.4 Tabel straturi CNN1

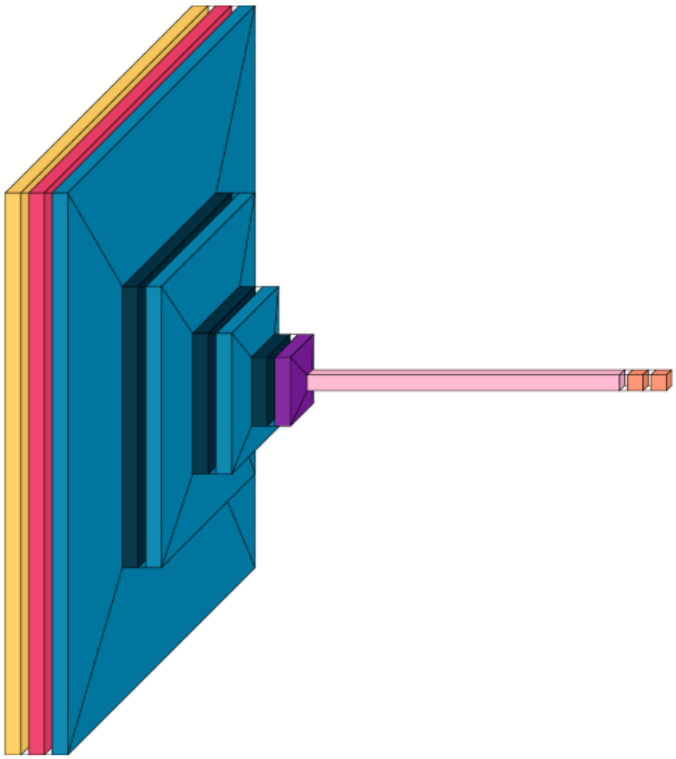

Fig 4.5 Structura CNN1

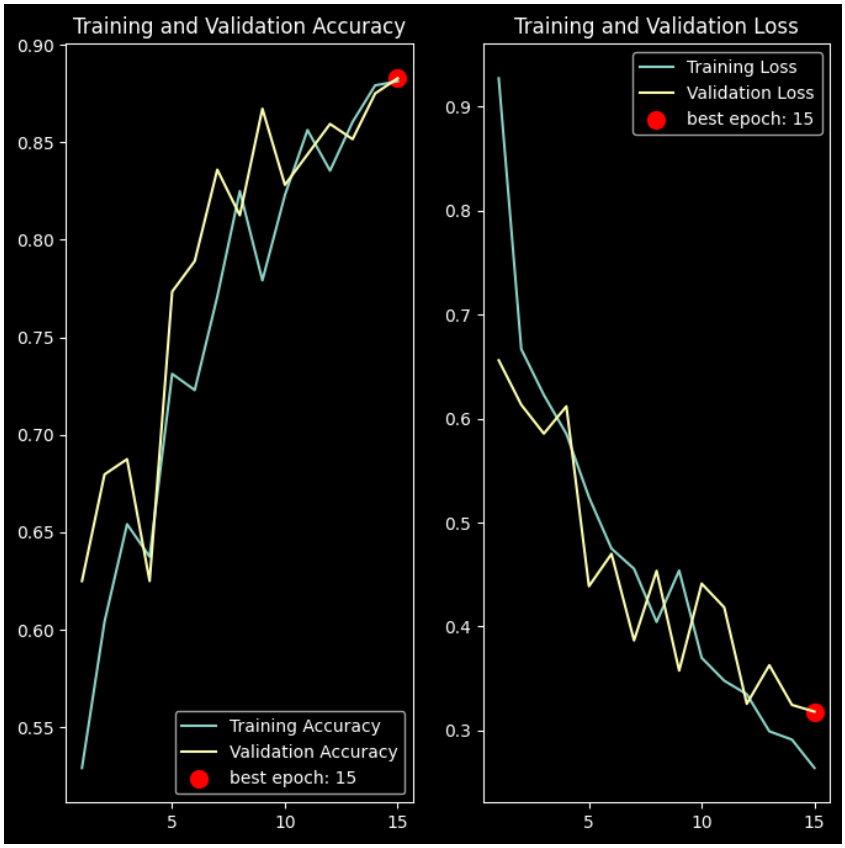

Fig 4.6 Valorile antrenării pentru acurateţe şi funcţia Loss

II. A doua rețea convoluțională (CNN-G2)

Structura poate fi vizualizată atât în tabelul prezentat în Fig 4.7 și Fig 4.8, și în schema din Fig 4.9. Numărul total al parametrilor antrenabili este de 67.589.986.

Valorile pentru acurateță și loss pot fi vizualizate in Fig 4.10 unde sunt prezentate valorile pentru fiecare dintre cele 15 epoci de antrenare.

Layer (type)	Output Shape	Param #
sequential_5 (Sequential)	(None, 180, 180, 3)	0
rescaling_5 (Rescaling)	(None, 180, 180, 3)	0
conv2d_32 (Conv2D)	(None, 180, 180, 128)	3,584
max_pooling2d_16 (MaxPooling2D)	(None, 90, 90, 128)	0
conv2d_33 (Conv2D)	(None, 90, 90, 64)	73,792
max_pooling2d_17 (MaxPooling2D)	(None, 45, 45, 64)	0
conv2d_34 (Conv2D)	(None, 45, 45, 32)	18,464
max_pooling2d_18 (MaxPooling2D)	(None, 22, 22, 32)	0
conv2d_35 (Conv2D)	(None, 22, 22, 16)	4,624
max_pooling2d_19 (MaxPooling2D)	(None, 11, 11, 16)	0
flatten_9 (Flatten)	(None, 1936)	0
dense_8 (Dense)	(None, 128)	247,936
dense_9 (Dense)	(None, 256)	33,024
reshape_4 (Reshape)	(None, 4, 4, 16)	0
conv2d_36 (Conv2D)	(None, 4, 4, 16)	2,320

Fig 4.7 Prima parte a tabelului structurii CNN2

up_sampling2d_16 (UpSampling2D)	(None, 8, 8, 16)	0
conv2d_37 (Conv2D)	(None, 8, 8, 32)	4,640
up_sampling2d_17 (UpSampling2D)	(None, 16, 16, 32)	0
conv2d_38 (Conv2D)	(None, 16, 16, 64)	18,496
up_sampling2d_18 (UpSampling2D)	(None, 32, 32, 64)	0
conv2d_39 (Conv2D)	(None, 32, 32, 128)	73,856
up_sampling2d_19 (UpSampling2D)	(None, 64, 64, 128)	0
dropout (Dropout)	(None, 64, 64, 128)	0
flatten_10 (Flatten)	(None, 524288)	0
dense_10 (Dense)	(None, 128)	67,108,992
outputs (Dense)	(None, 2)	258

Fig 4.8 Partea a doua a tabelului structurii CNN2

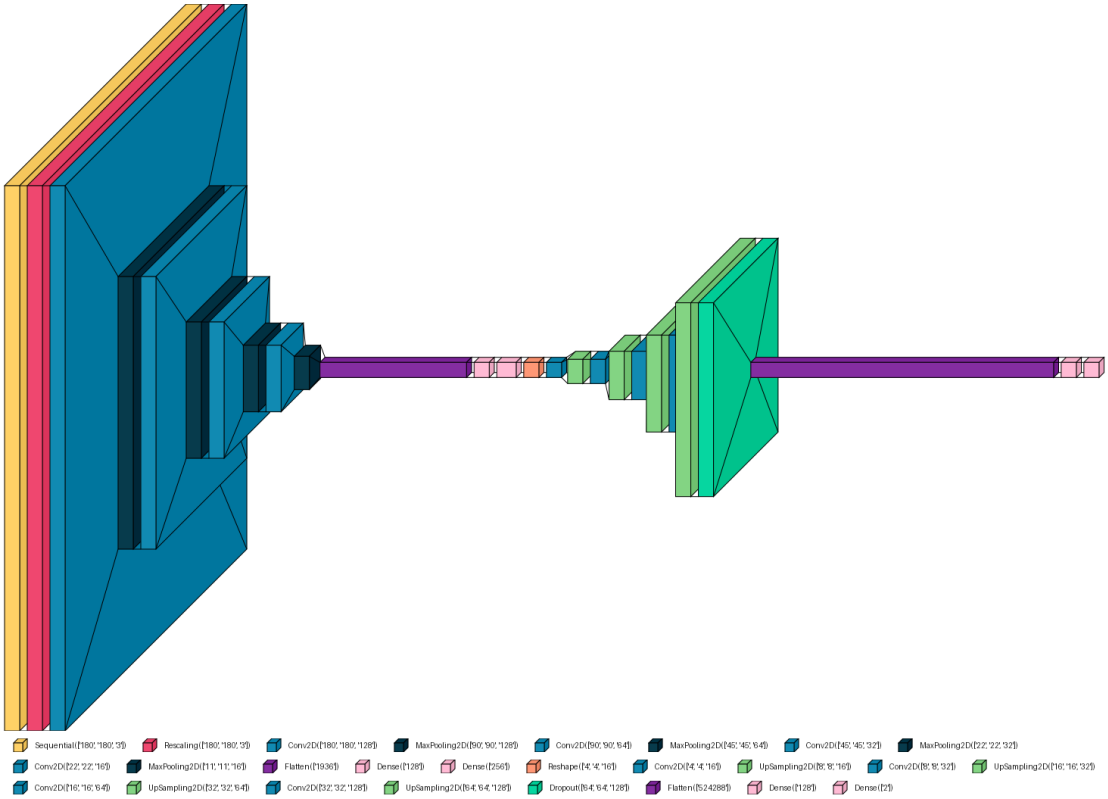

Fig 4.9 Reprezentarea schematică a CNN2

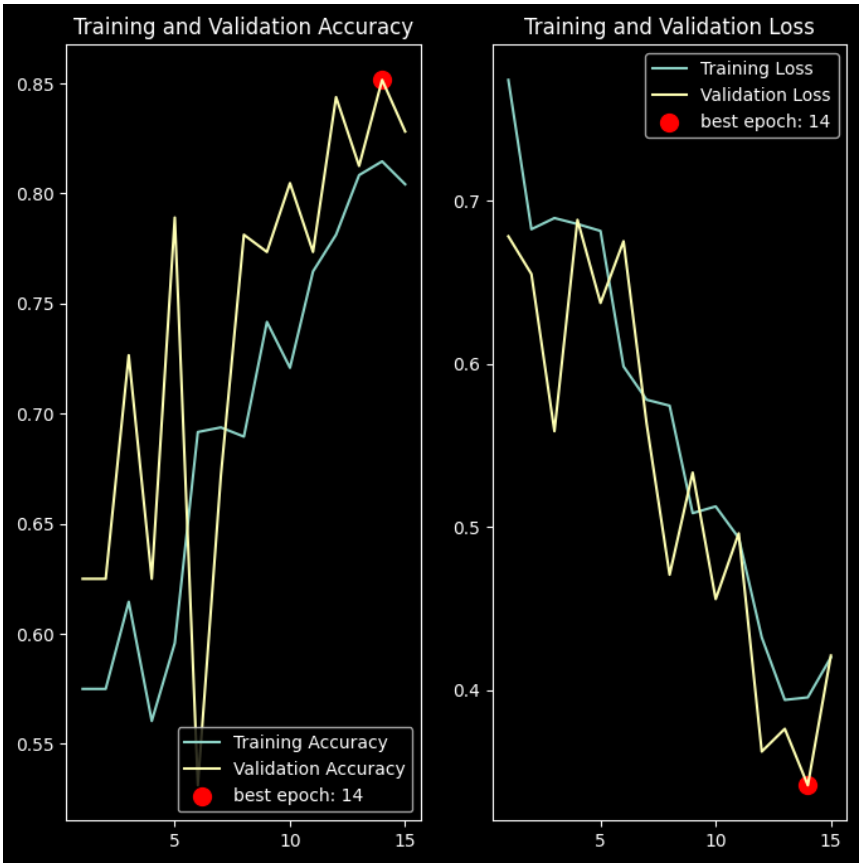

Fig 4.10 Valorile pentru acurateţe şi loss pe seturile de antrenare şi de validare

4.2.2 Rețelele neuronale de detectare a obiectelor

În această secțiune vom analiza rețelele neuronale responsabile cu detectarea obiectelor în imagini (în cazul nostru al coșurilor de gunoi). Fiecare rețea va fi capabilă să identifice coșul de gunoi, să specifice în ce categorie se încadrează (coș gol sau plin) și să specifice gradul de încredere al inferenței.

Pentru realizarea cestei sarcini care ține de „computer vision" vom folosi 3 modele YOLO11 din framework-ul Ultralytics [9].

De asemenea Ultralytics realizează augmentarea imaginilor fără ca noi să îi cerem în mod explicit. Putem modifica acești parametrii sau să îi dezactivăm dacă ne dorim [10].

În cadrul proiectului nostru vom specifica următorii parametrii:

- Schimbarea nuanței culorilor („hue ajustment") prin setarea parametrului hsv_h la valoarea de la valoarea prestabilită de 0,015 la 0,5
- Ajustarea saturației („saturation ajustment") prin modificarea parametrului hsv_s de la 0,7 la 0,5.
- Ajustarea luminozității („brightness ajustment") prin modificarea parametrului hsv_v de la 0,4 la 0,5.
- Rotirea imaginilor se face într-un interval cuprins de la 0° la 90° prin setarea parametrului degrees la 90 (implicit el este 0).

Framework-ul Ultralytics face antrenarea modelelor accesibilă cu doar câteva linii de cod așa cum se poate observa din Fig 4.11. Fiecare rețea va fi antrenată pe o durată de o sută de epoci.

```
from ultralytics import YOLO
import os

# Load a model
model = YOLO("yolo11m.pt")  # load a pretrained model (recommended for training)

# Train the model
results = model.train(data= f"{os.getcwd()}/data.yaml", epochs=100, imgsz=180, hsv_h= 0.5,
hsv_s= 0.5,
hsv_v= 0.5,
degrees = 90)
```

Fig 4.11 Codul necesar pentru antrenarea unui model

Fiecare model de detecție își ia informațiile din fișierul de configurare data.yaml unde sunt specificate căile către setul de antrenare, validare și testare.

```
1  path: output_dataset
2  train: images/train
3  val: images/val
4  test: images/test
5
6  names:
7    0: empty
8    1: full
```

Fig 4.12 Cod fișier de configurare

Codul de împărţire a setului în subseturi de antrenare, testare şi validare a fost realizat cu ajutorul sistemului AI Copilot [16].

În continuare vom prezenta rezultatele antrenării fiecărui model de detecţie împreună cu graficele aferente:

I. Modelul YOLO11m:

- Sumarul antrenării poate fi vizualizat în Fig 4.13

```
Validating runs\detect\train9\weights\best.pt...
Ultralytics 8.3.163  Python-3.12.9 torch-2.7.1+cpu CPU (Intel Core(TM) i7-8750H 2.20GHz)
YOLO11m summary (fused): 125 layers, 20,031,574 parameters, 0 gradients, 67.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        135        185      0.901      0.838      0.912      0.777
                 empty         67         91      0.871      0.824      0.892      0.765
                  full         85         94      0.931      0.851      0.931      0.789
Speed: 0.1ms preprocess, 46.0ms inference, 0.0ms loss, 0.8ms postprocess per image
```

Fig 4.13 Rezumatul antrenării

- Graficul scorului F1 în raport cu încrederea, graficul preciziei în raport cu încrederea, graficul preciziei în raport cu „Recall” şi graficul funcţiei „Recall” în raport cu încrederea

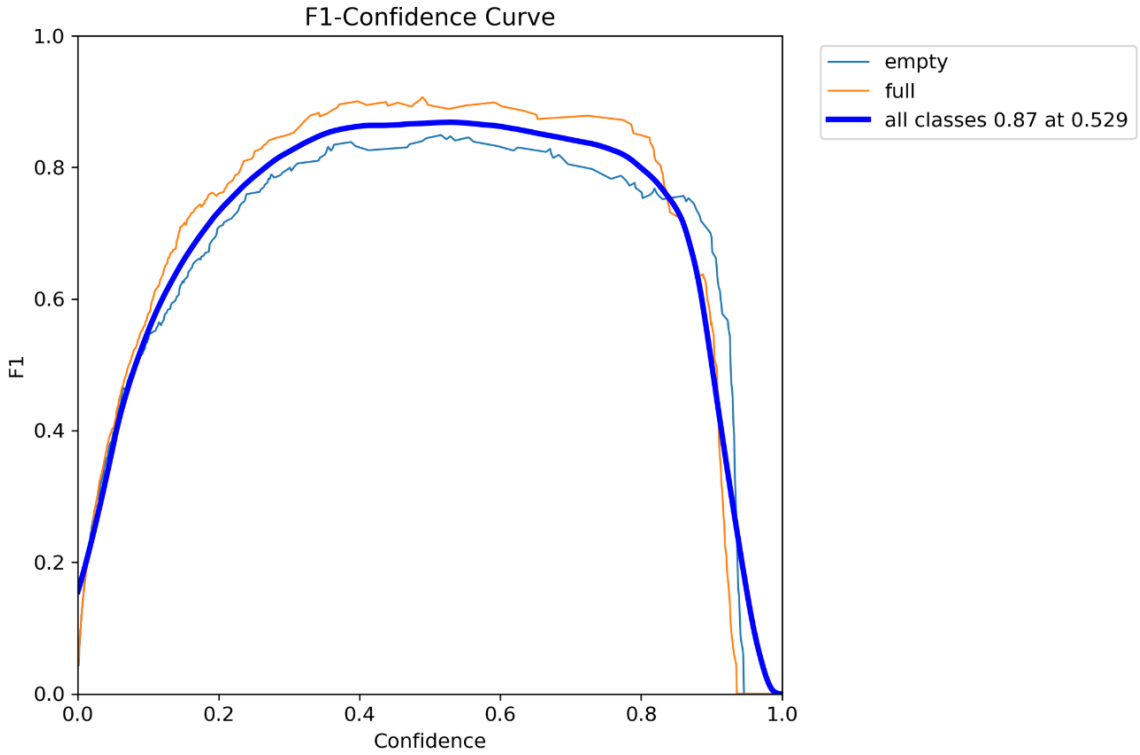

Fig4.14 Graficul F1 raportat la încredere

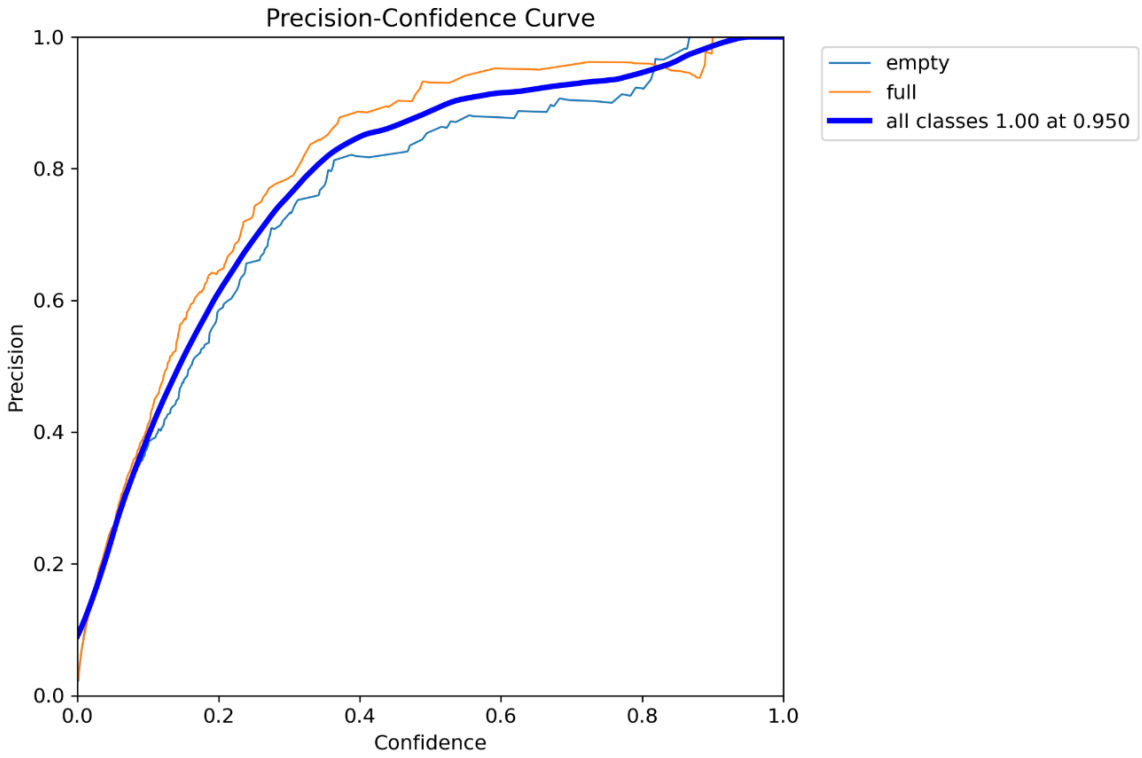

Fig 4.15 Precizie raportată la încredere

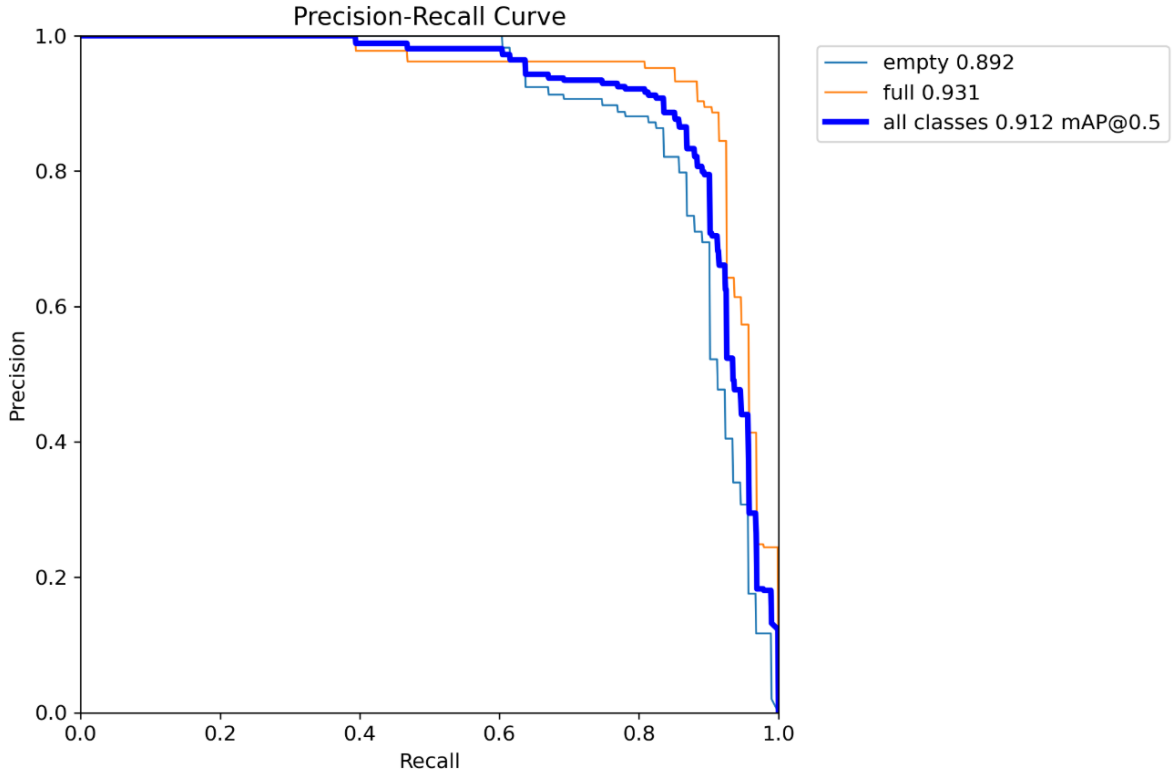

Fig 4.16 Precizie raportată la Recall

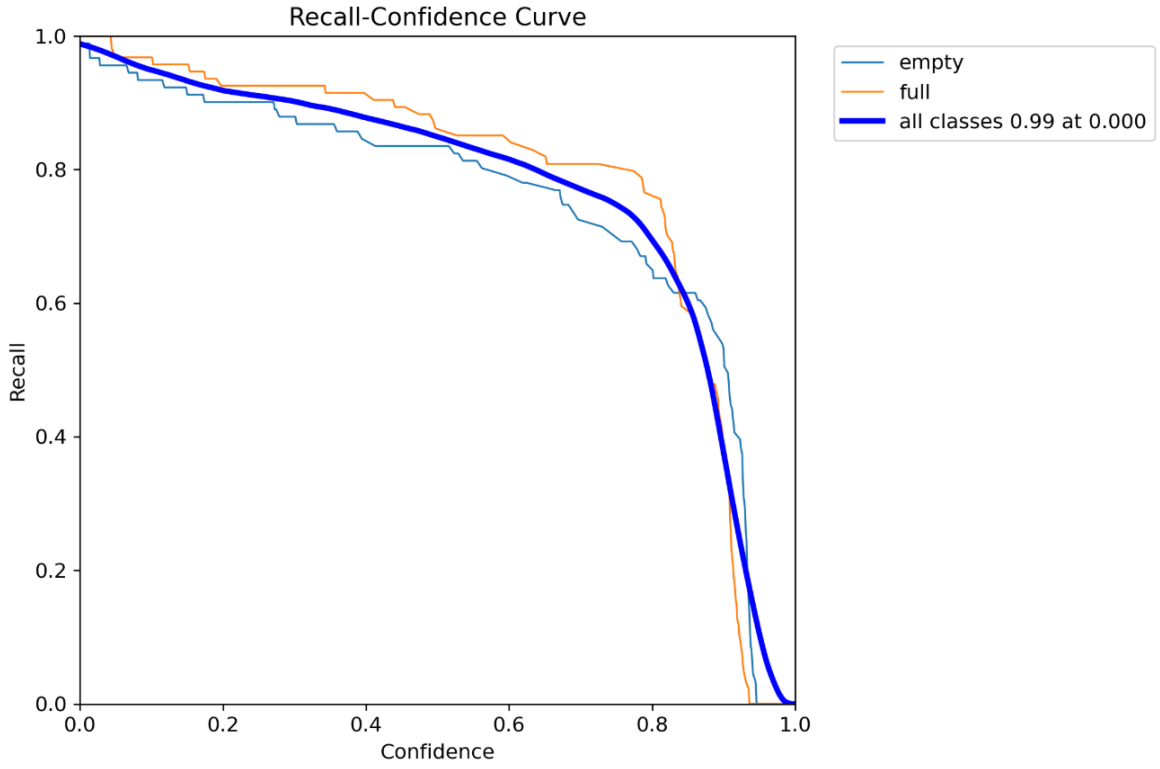

Fig 4.17 Recall raportat la încredere

- Matricea de confuzie şi matricea de confuzie normalizată

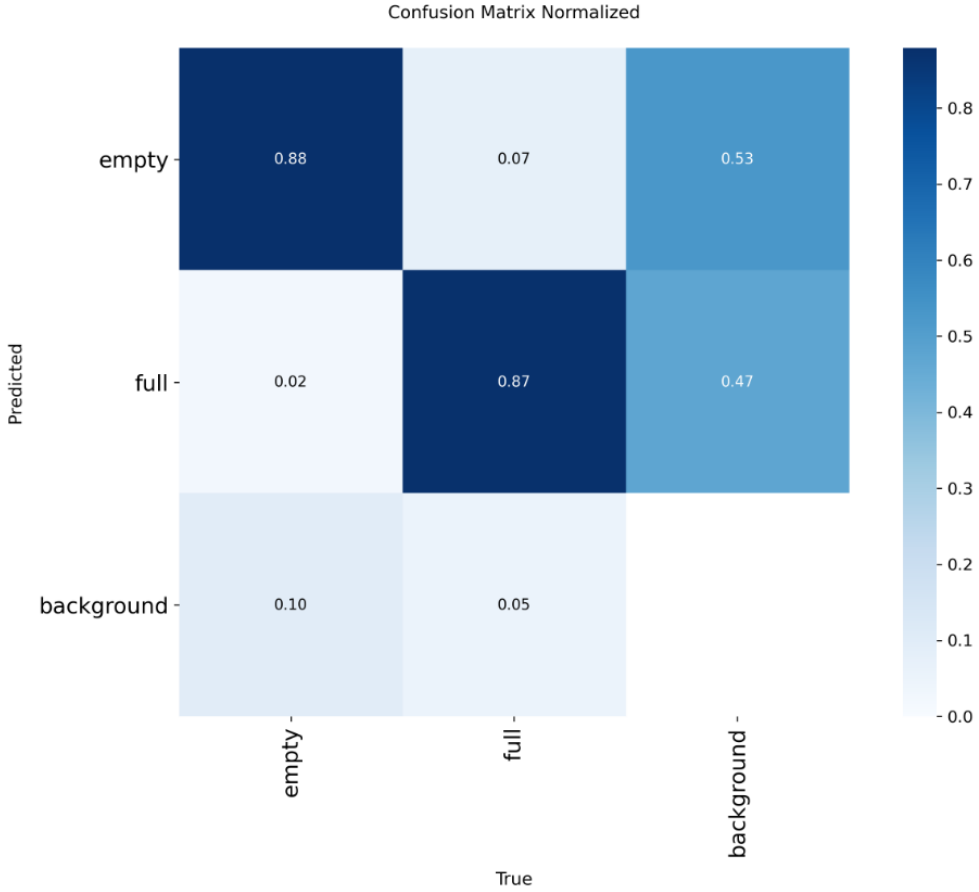

Fig 4.18 Matricea de confuzie normalizată

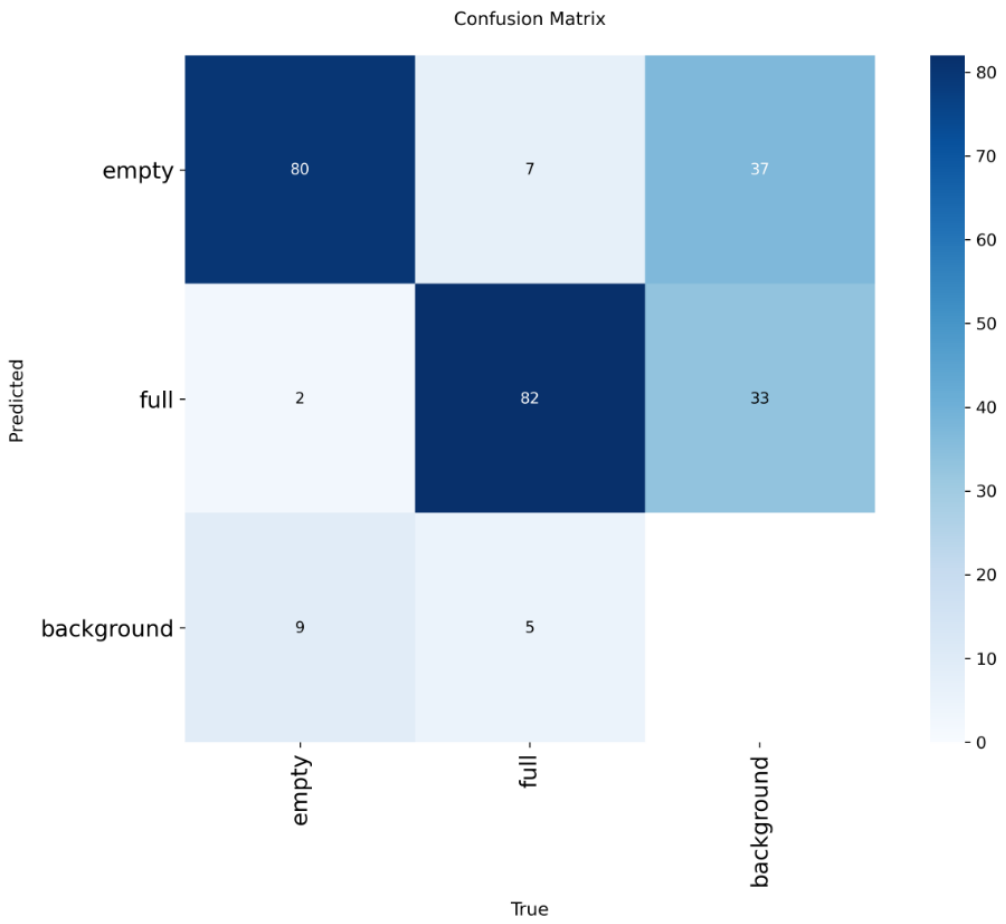

Fig 4.19 Matricea de confuzie

- „labels”

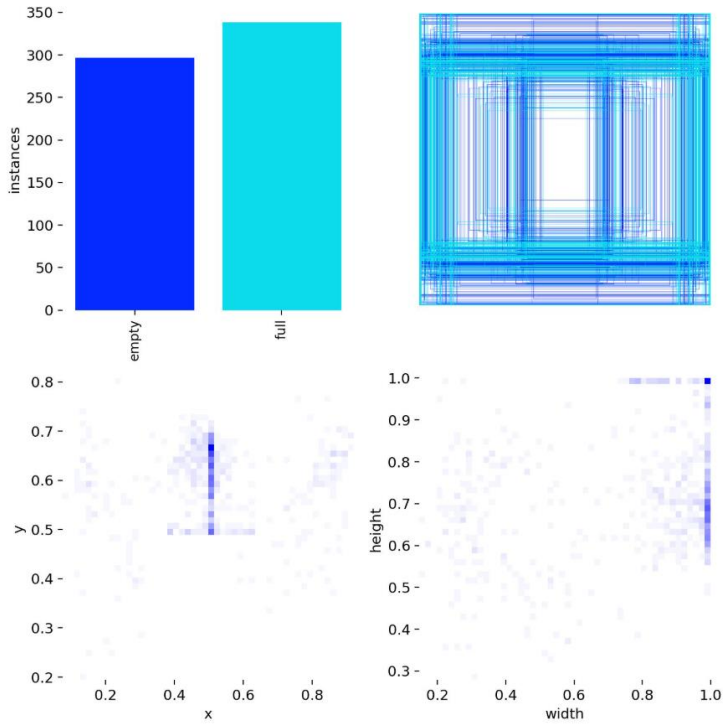

Fig 4.20 „labels”

- Alte grafice de testare și validare

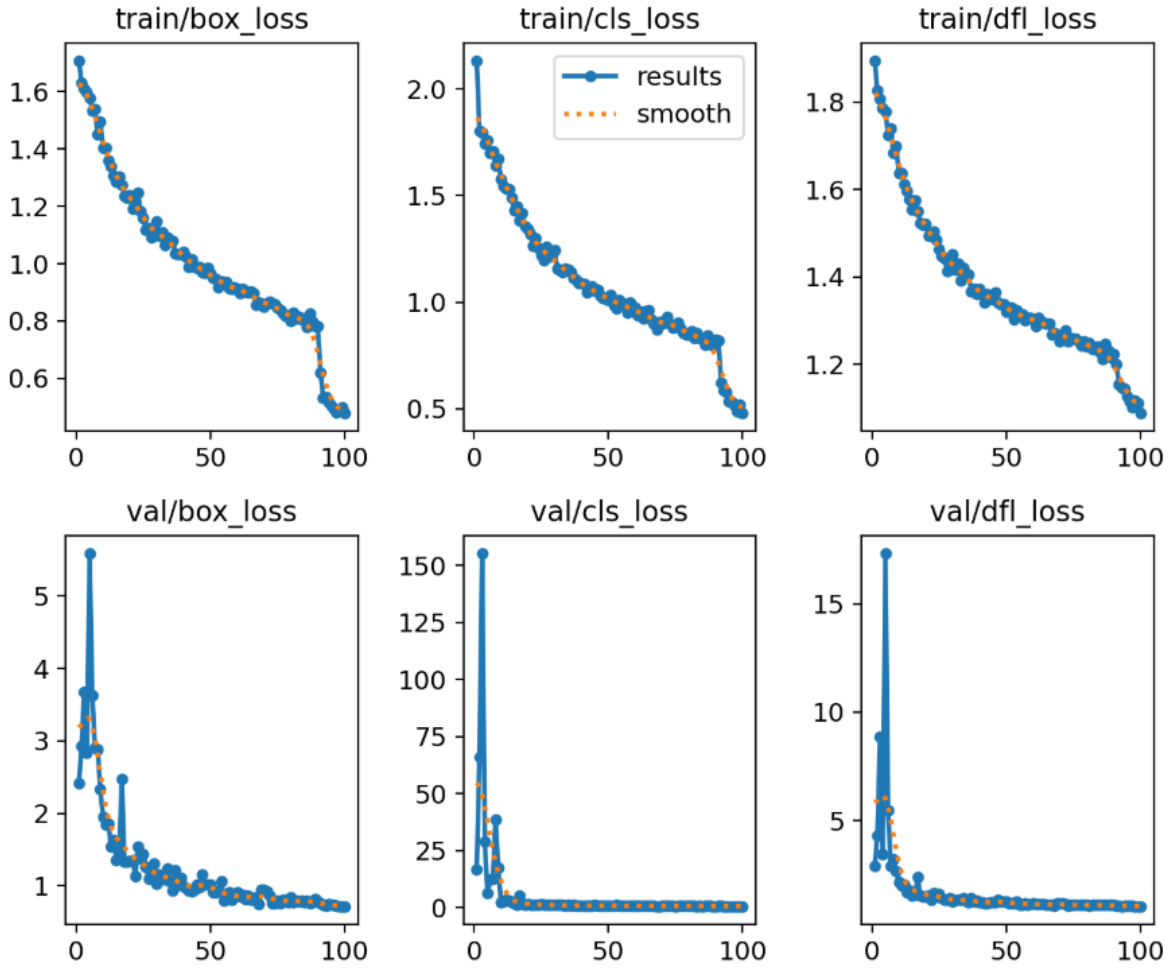

Fig 4.21 Metrici antrenare și validare

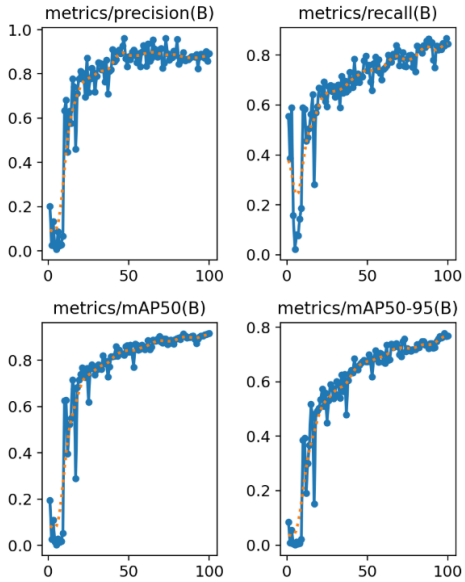

Fig 4.22 precision(B), recall(B), mAP50(B), mAP50-95(B)

II. Modelul YOLO11s:

- Sumarul antrenării poate fi vizualizat în Fig 4.23

```
Validating runs\detect\train8\weights\best.pt...
Ultralytics 8.3.163  Python-3.12.9 torch-2.7.1+cpu CPU (Intel Core(TM) i7-8750H 2.20GHz)
YOLO11s summary (fused): 100 layers, 9,413,574 parameters, 0 gradients, 21.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        135        185      0.944      0.862      0.912      0.762
                 empty         67         91      0.933      0.835       0.89      0.739
                  full         85         94      0.954       0.89      0.934      0.784
Speed: 0.1ms preprocess, 16.2ms inference, 0.0ms loss, 0.7ms postprocess per image
```

Fig 4.23 Rezumatul antrenării

- Graficul scorului F1 în raport cu încrederea, graficul preciziei în raport cu încrederea, graficul preciziei în raport cu „Recall” şi graficul funcţiei „Recall” în raport cu încrederea

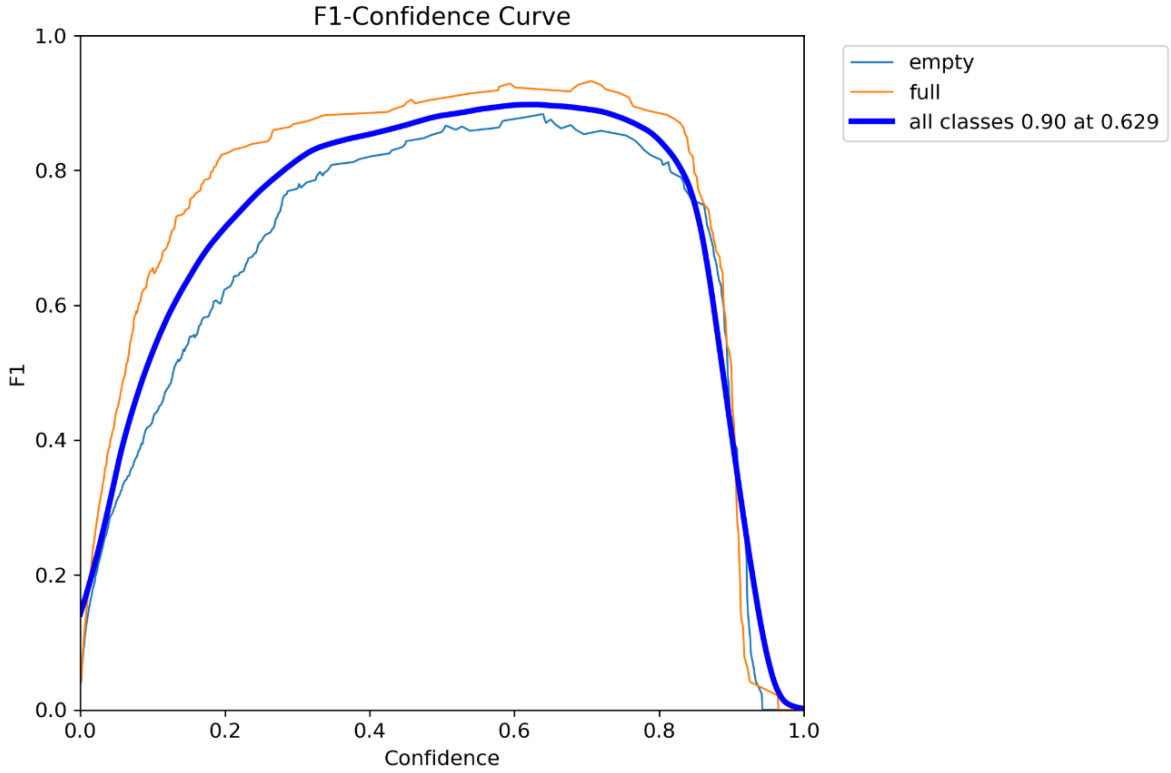

Fig4.24 Graficul F1 raportat la încredere

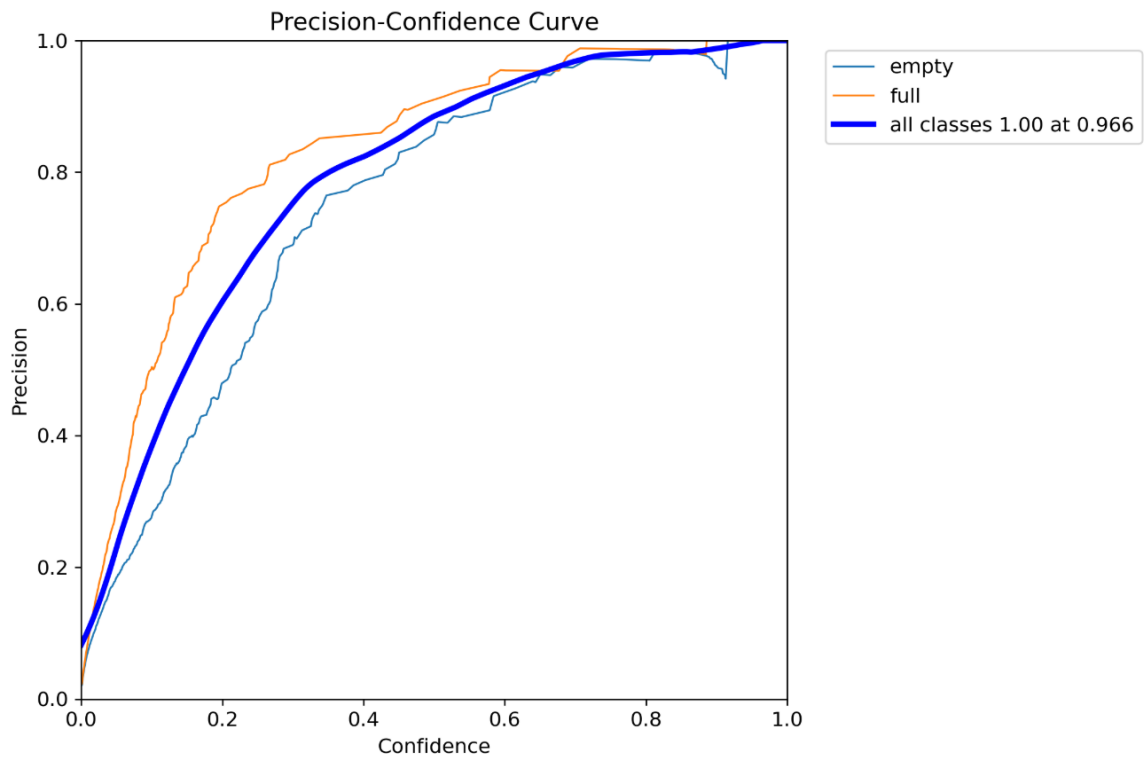

Fig 4.25 Precizie raportată la încredere

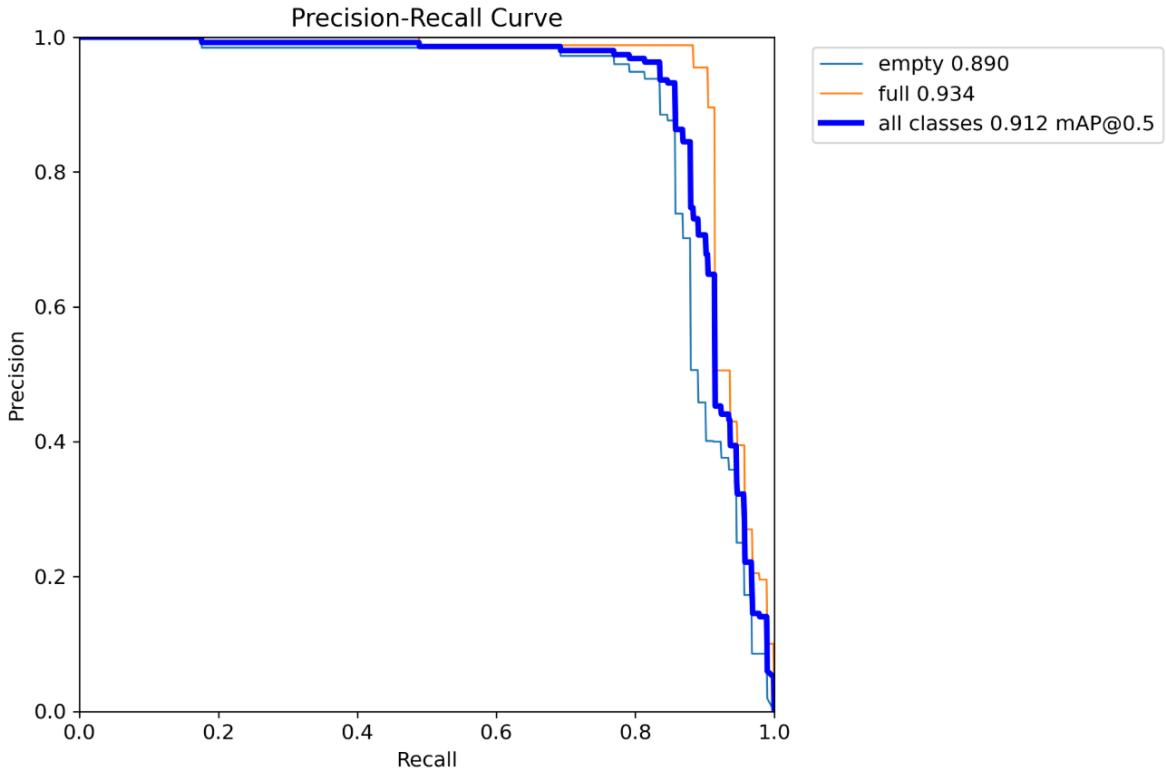

Fig 4.26 Precizie raportată la Recall

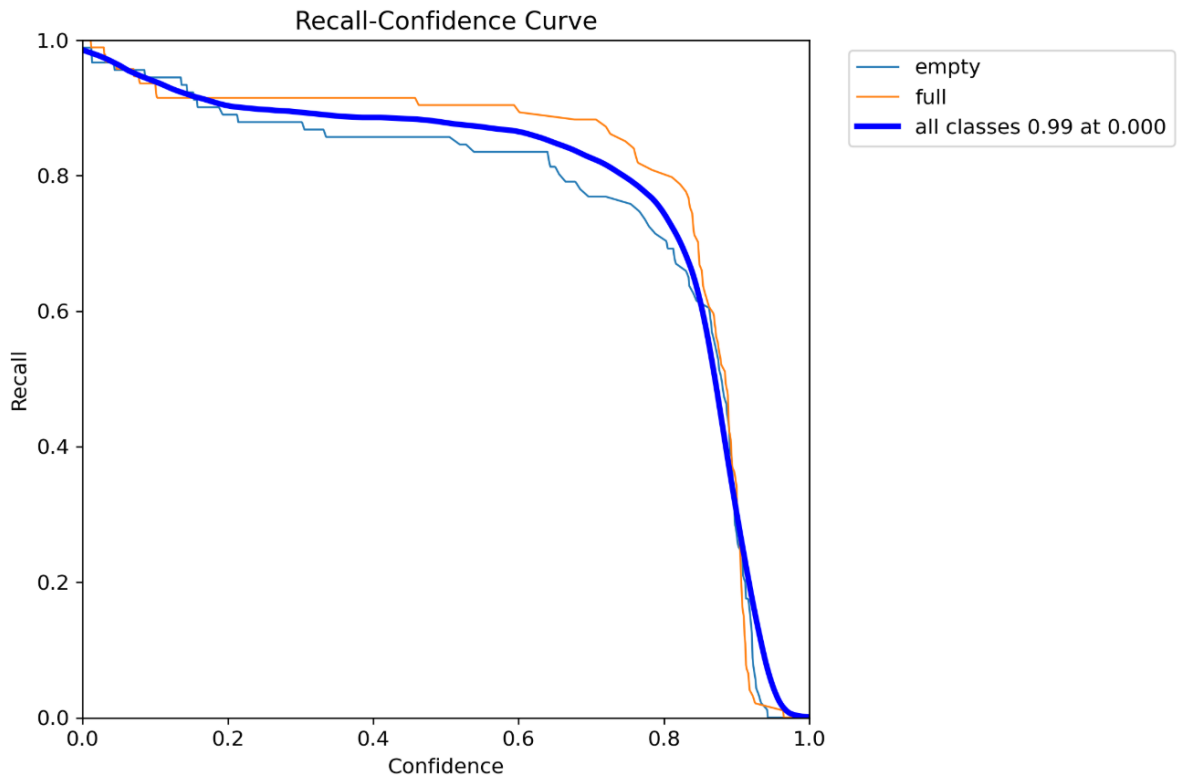

Fig 4.27 Recall raportat la încredere

- Matricea de confuzie şi matricea de confuzie normalizată

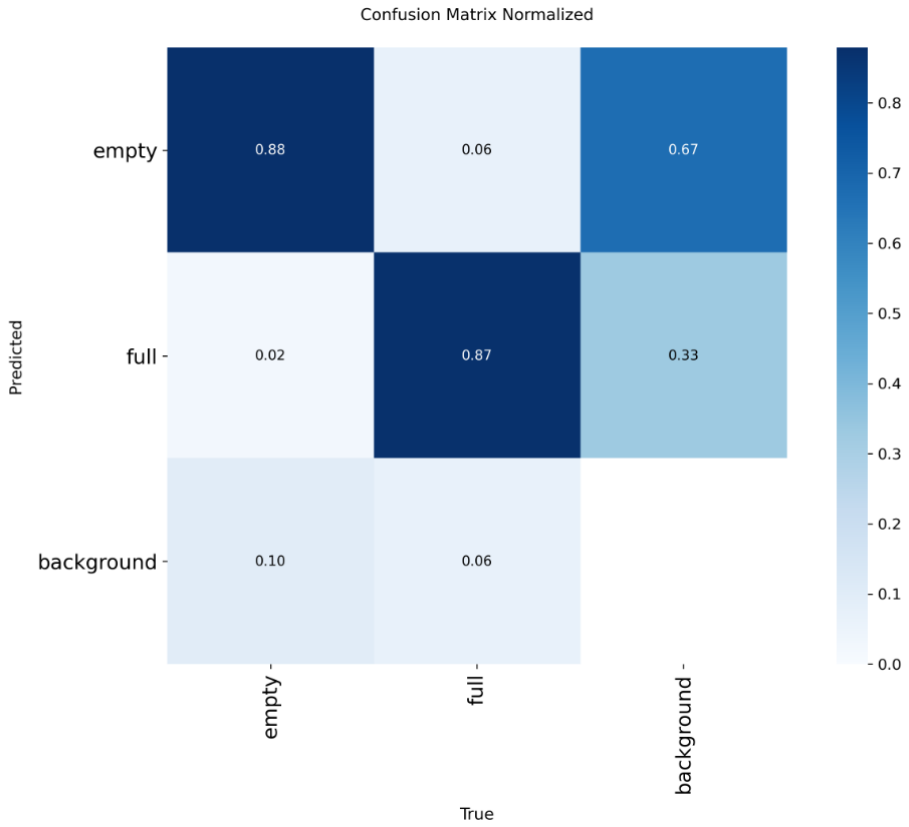

Fig 4.28 Matricea de confuzie normalizată

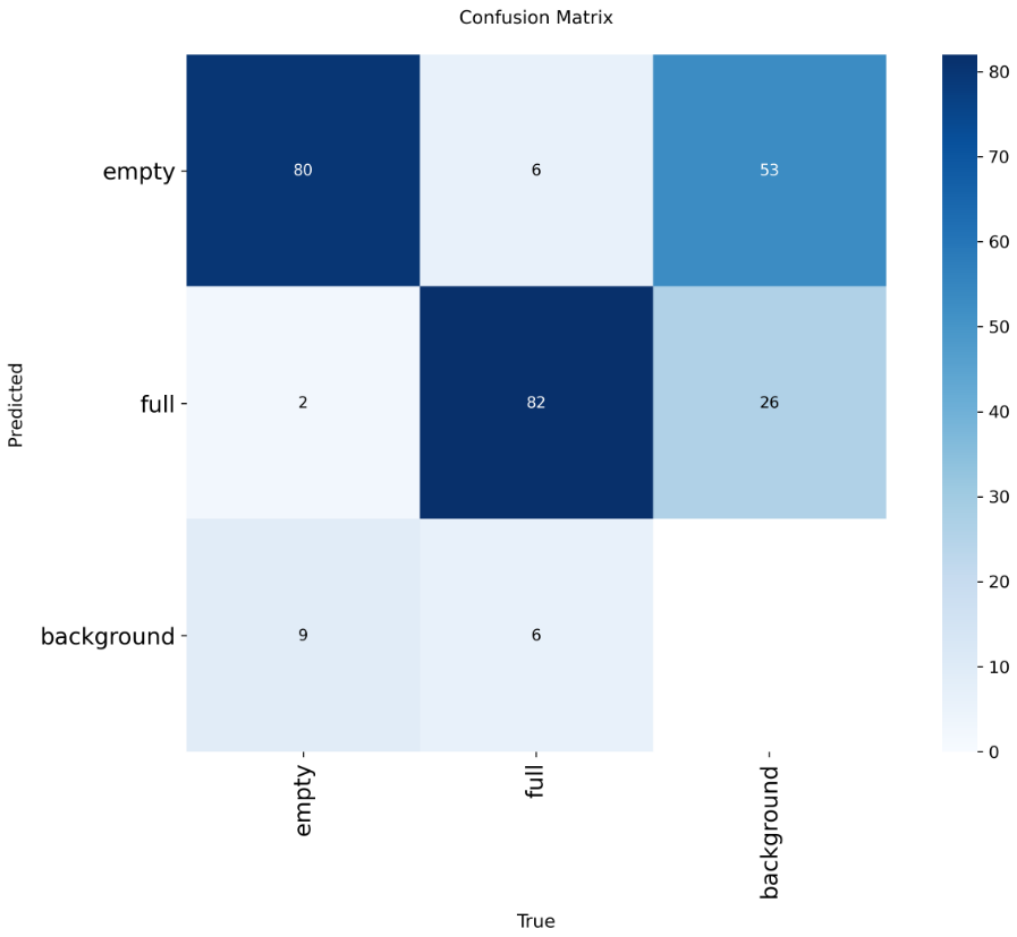

Fig 4.29 Matricea de confuzie

- „labels”

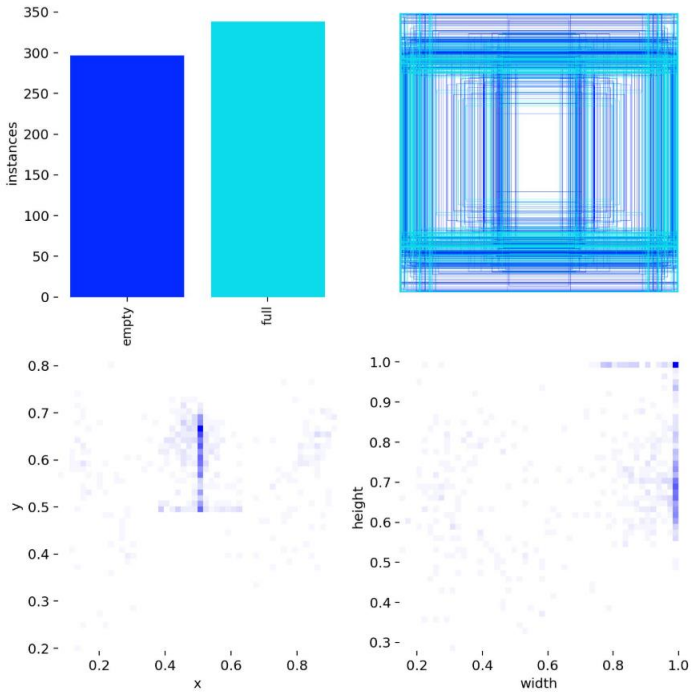

Fig 4.30 „labels”

- Alte grafice de testare şi validare

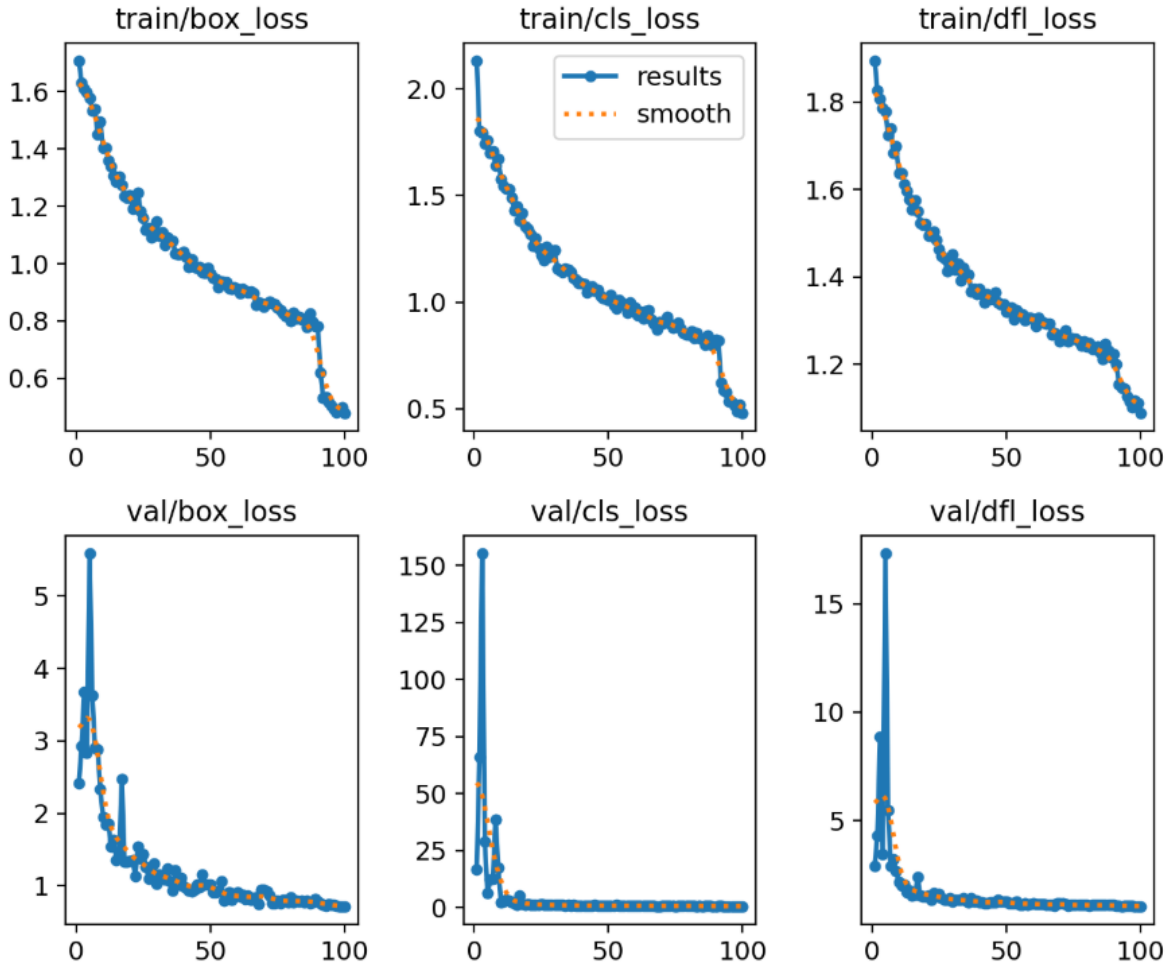

Fig 4.31 Metrici antrenare şi validare

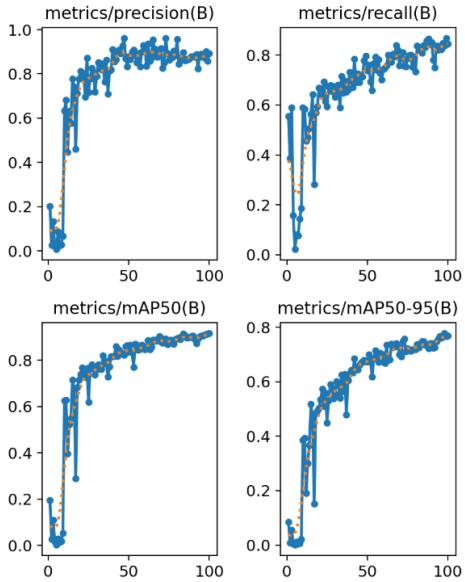

Fig 4.32 precision(B), recall(B), mAP50(B), mAP50-95(B)

III. Modelul YOLO11n:

- Sumarul antrenării poate fi vizualizat în Fig 4.33

```
Validating runs\detect\train7\weights\best.pt...
Ultralytics 8.3.163  Python-3.12.9 torch-2.7.1+cpu CPU (Intel Core(TM) i7-8750H 2.20GHz)
YOLO11n summary (fused): 100 layers, 2,582,542 parameters, 0 gradients, 6.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all        135        185      0.906      0.796      0.907      0.772
                 empty         67         91      0.873      0.769      0.885      0.752
                  full         85         94      0.939      0.823      0.929      0.793
Speed: 0.1ms preprocess, 7.9ms inference, 0.0ms loss, 0.8ms postprocess per image
```

Fig 4.33 Rezumatul antrenării

- Graficul scorului F1 în raport cu încrederea, graficul preciziei în raport cu încrederea, graficul preciziei în raport cu „Recall” şi graficul funcţiei „Recall” în raport cu încrederea

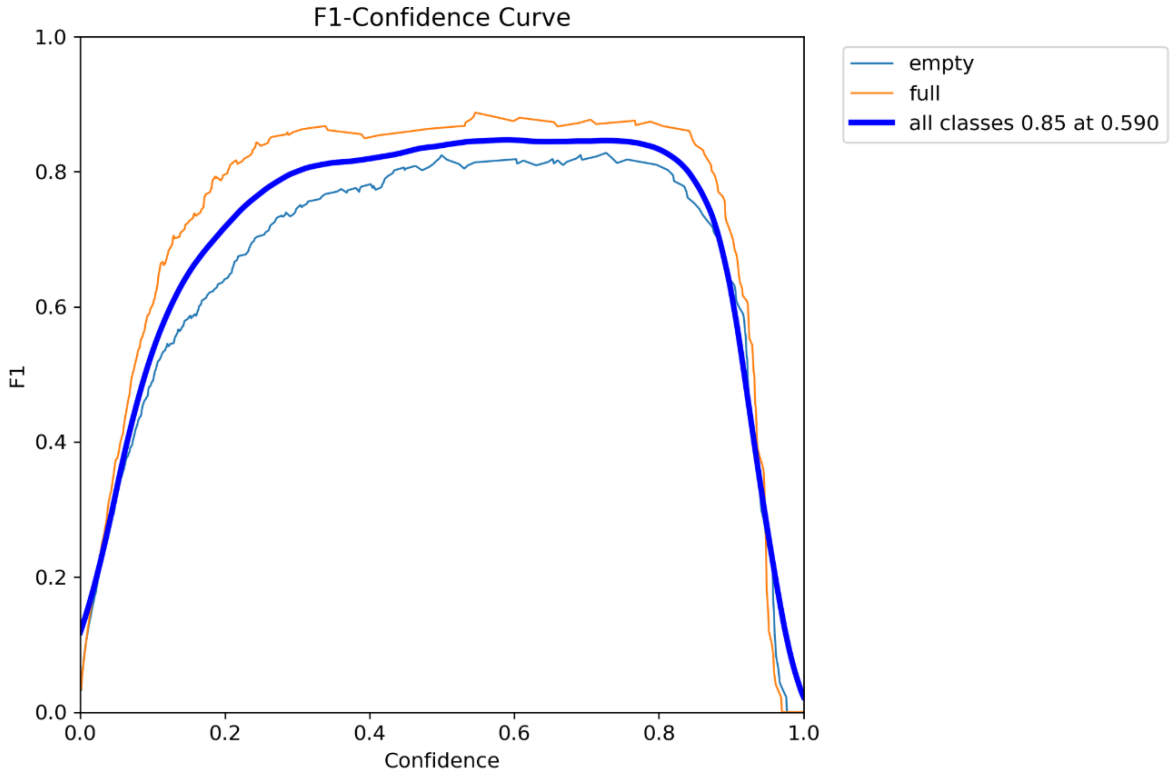

Fig4.34 Graficul F1 raportat la încredere

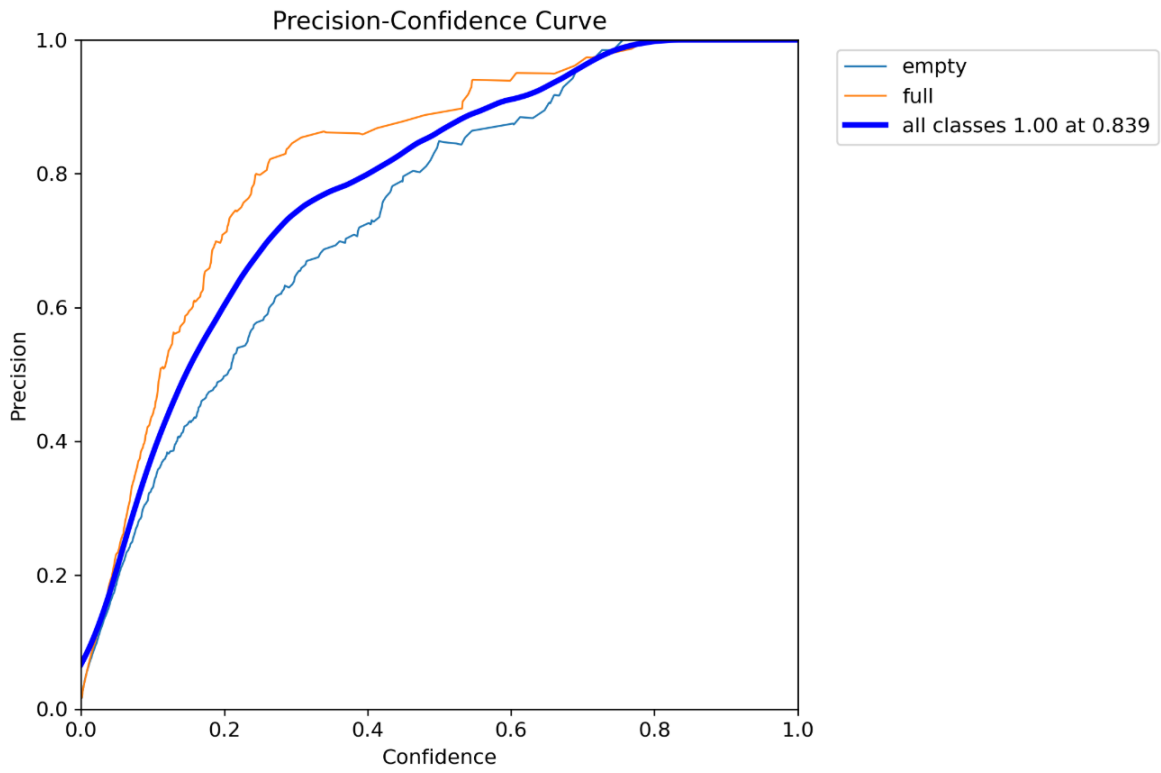

Fig 4.35 Precizie raportată la încredere

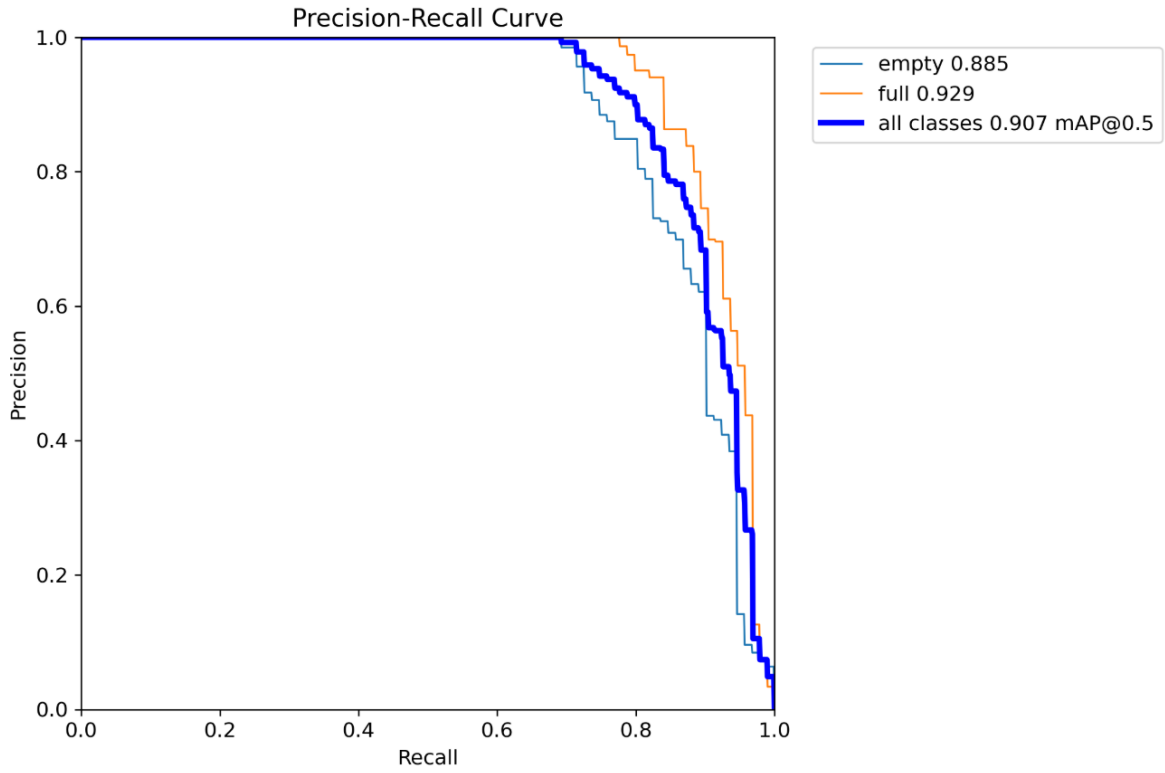

Fig 4.36 Precizie raportată la Recall

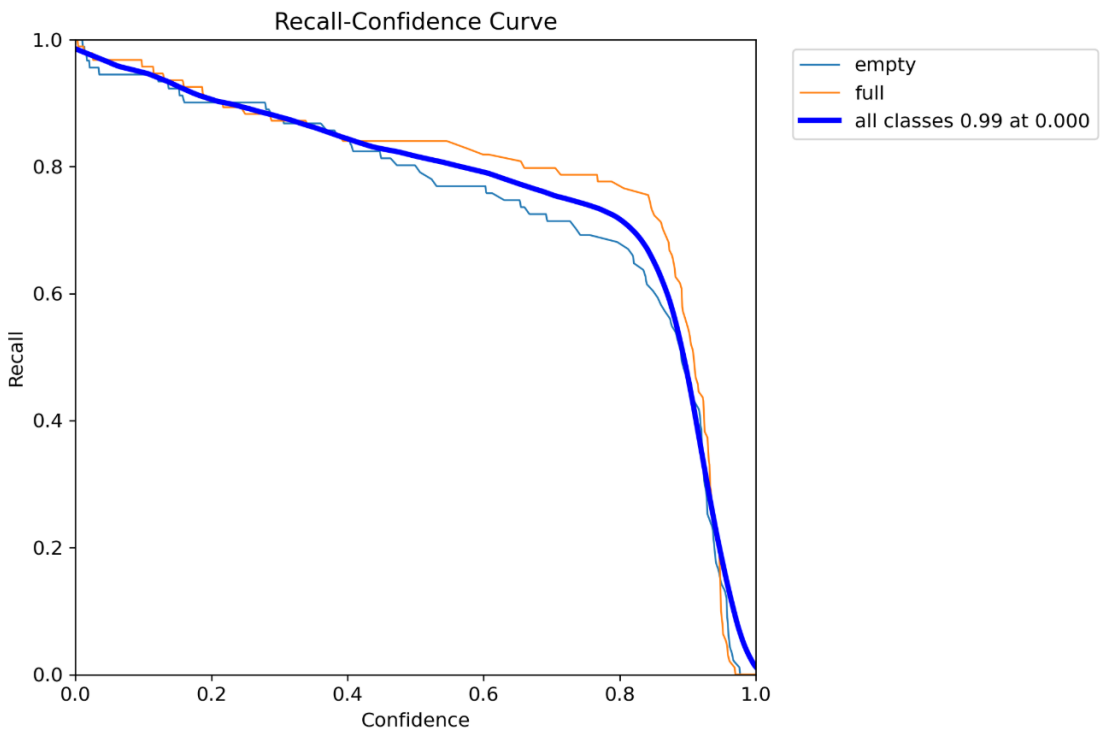

Fig 4.37 Recall raportat la încredere

- Matricea de confuzie şi matricea de confuzie normalizată

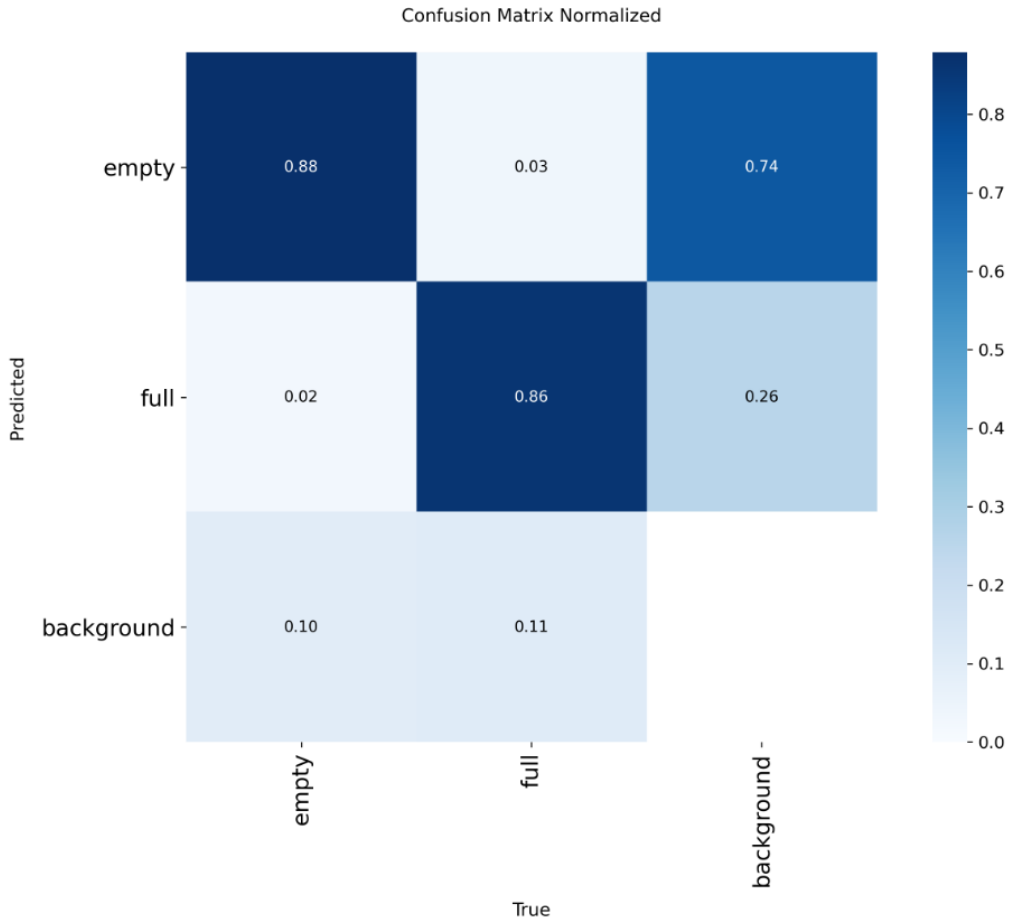

Fig 4.28 Matricea de confuzie normalizată

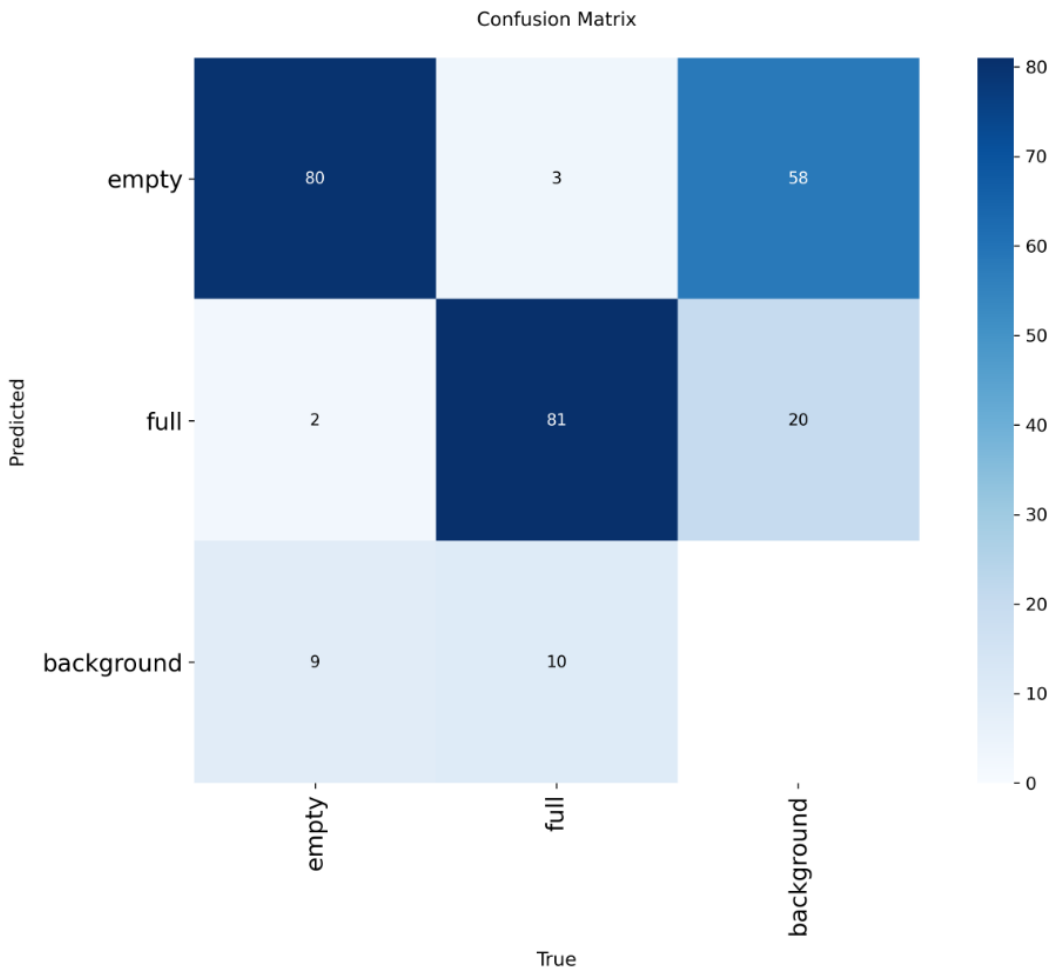

Fig 4.29 Matricea de confuzie

- „labels”

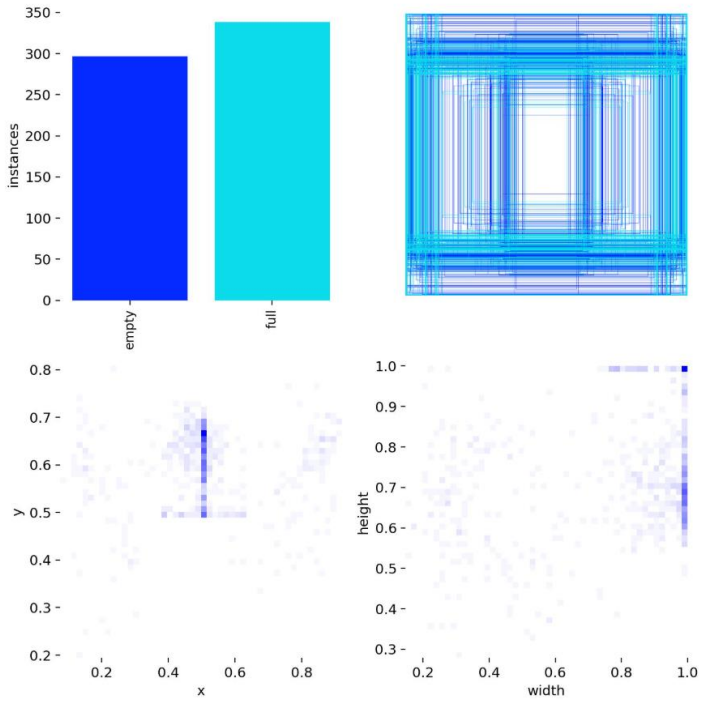

Fig 4.40 „labels”

- Alte grafice de testare și validare

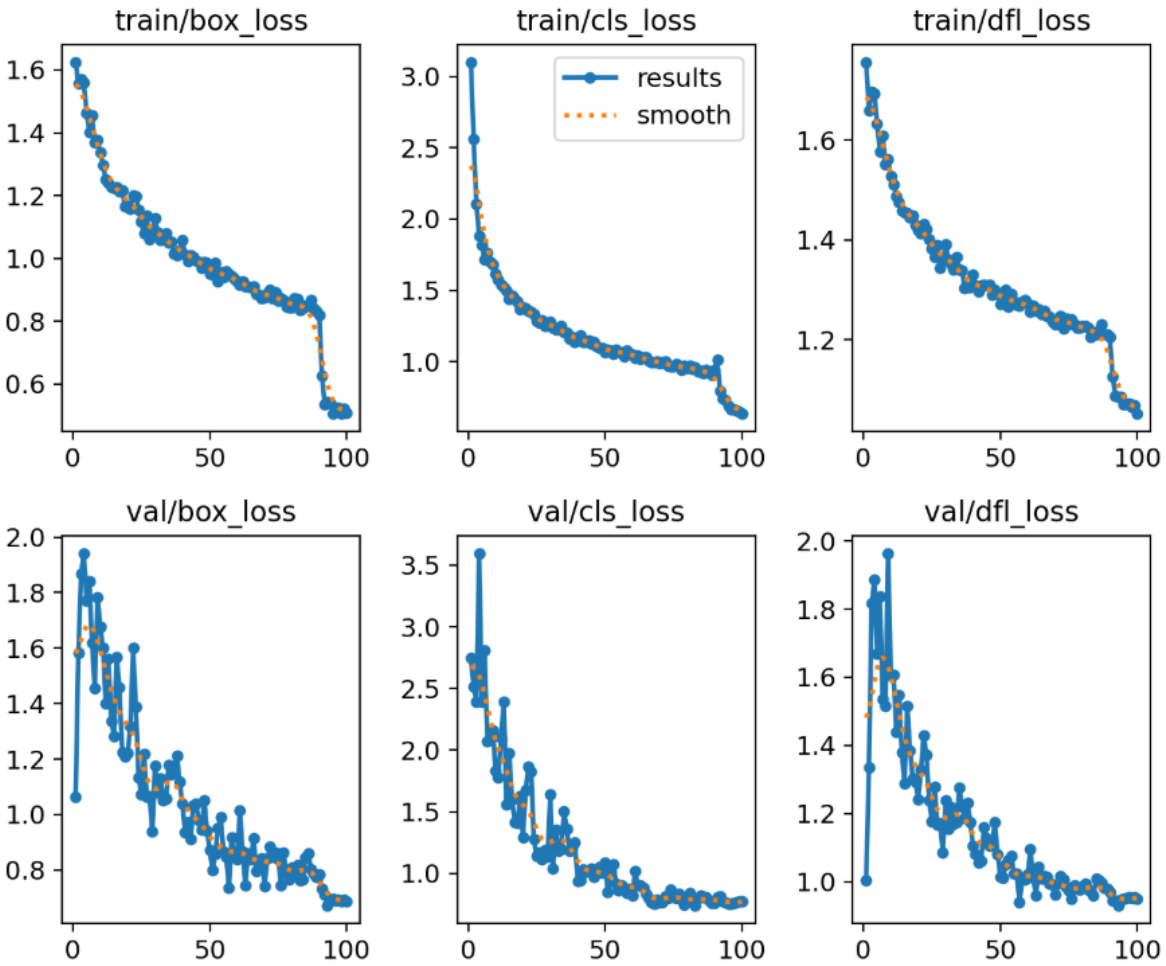

Fig 4.41 Metrici antrenare și validare

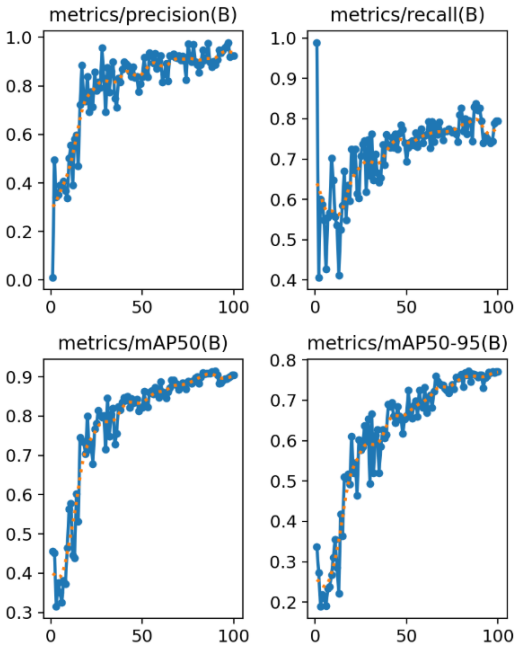

Fig 4.42 precision(B), recall(B), mAP50(B), mAP50-95(B)

CAPITOLUL 5. TESTAREA REȚELELOR

În acest capitol vom prezenta rapoartele testării rețelelor de inteligență artificială pe seturi de date speciale, care nu au mai fost „văzute” pentru a vede cât de bine generalizează.

5.1 Rețelele de clasificare a imaginilor

În această secțiune vom prezenta rezultatele obținute de fiecare rețea de clasificare a imginilor.

5.1.1 CNN-G1

- Raportul de clasificare

	precision	recall	f1-score	support
0	0.79	0.85	0.82	27
1	0.89	0.85	0.87	40
accuracy			0.85	67
macro avg	0.84	0.85	0.85	67
weighted avg	0.85	0.85	0.85	67

Fig 5.1 Raportul de clasificare

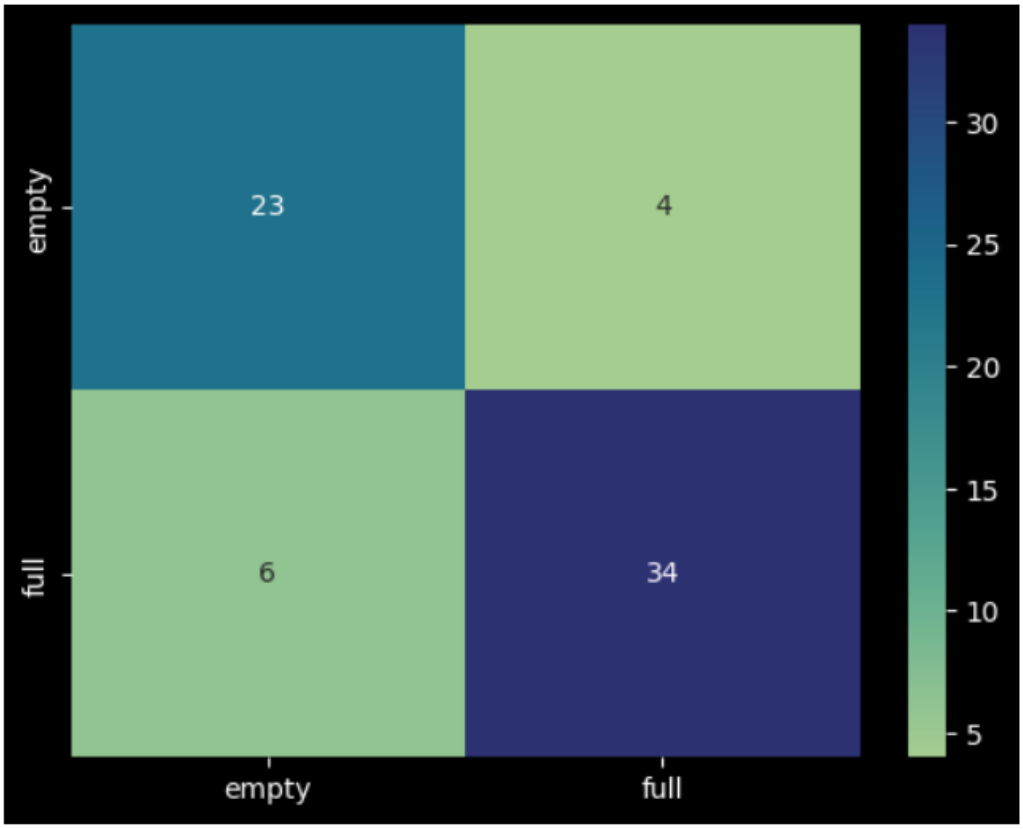

Fig 5.2 Matricea de confuzie

5.1.2 CNN-G2

- Raportul de clasificare

	precision	recall	f1-score	support
0	0.66	0.85	0.74	27
1	0.88	0.70	0.78	40
accuracy			0.76	67
macro avg	0.77	0.78	0.76	67
weighted avg	0.79	0.76	0.76	67

Fig 5.3 Raportul de clasificare

- Matricea de confuzie

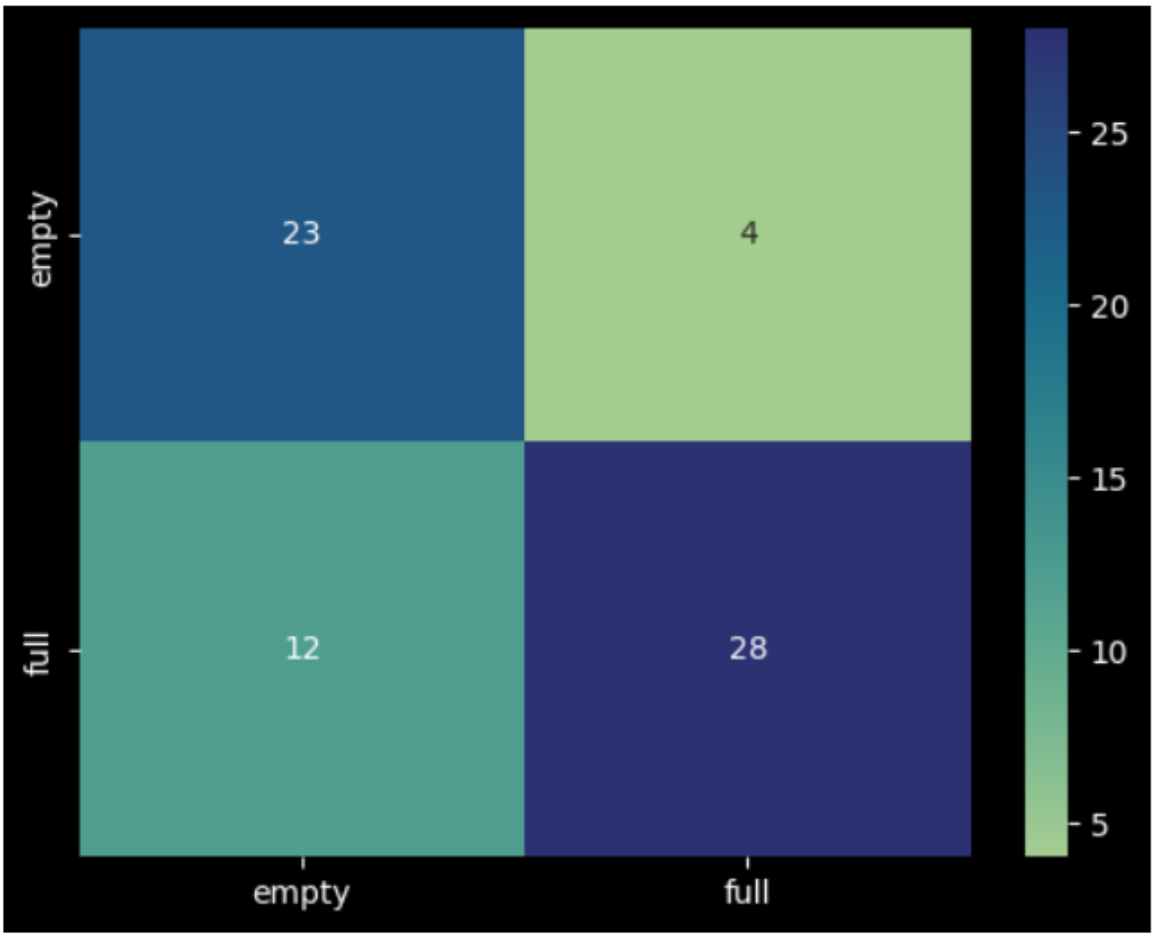

Fig 5.4 Matricea de confuzie

5.2 Reţelele de detectare a obiectelor în imagini

În aceasta vom prezenta mai multe grafice care reflectă performanţa fiecărei reţele de detectare a obiectelor în parte.

5.2.1 YOLO11n

- Raportul F1 și încredere

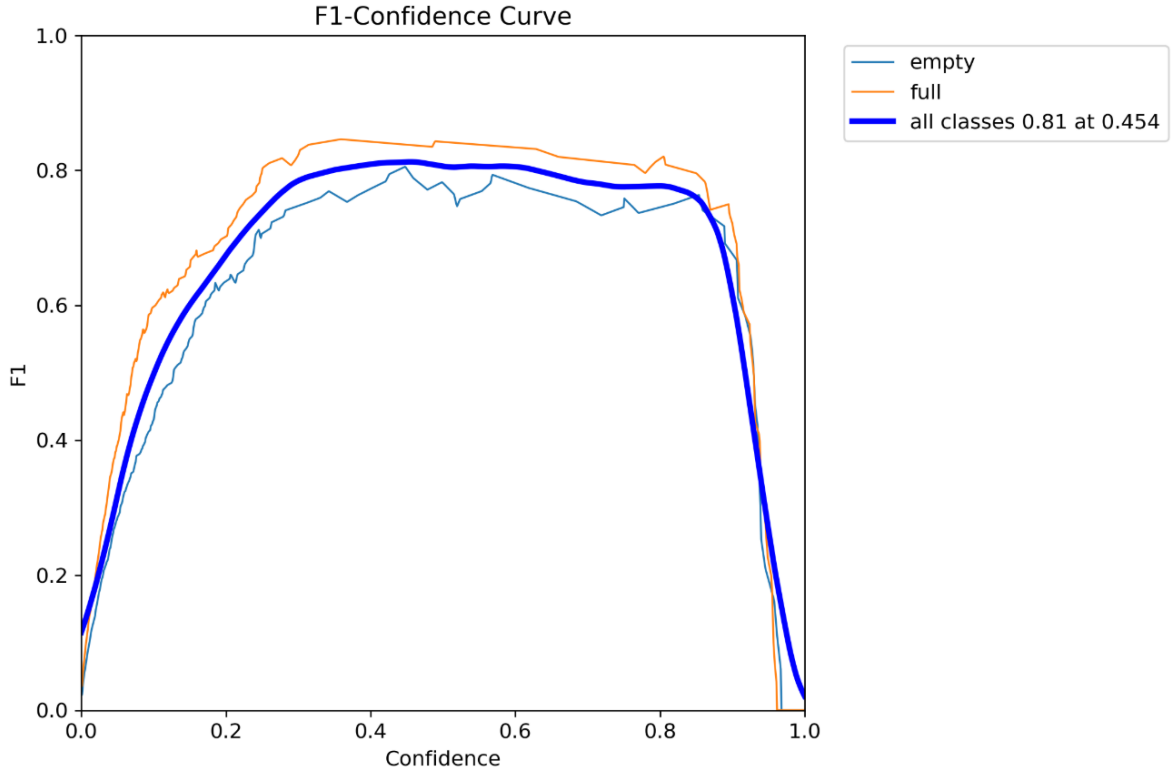

Fig 5.5 Raportul dintre F1 și încredere

- Raportul dintre precizie și încredere

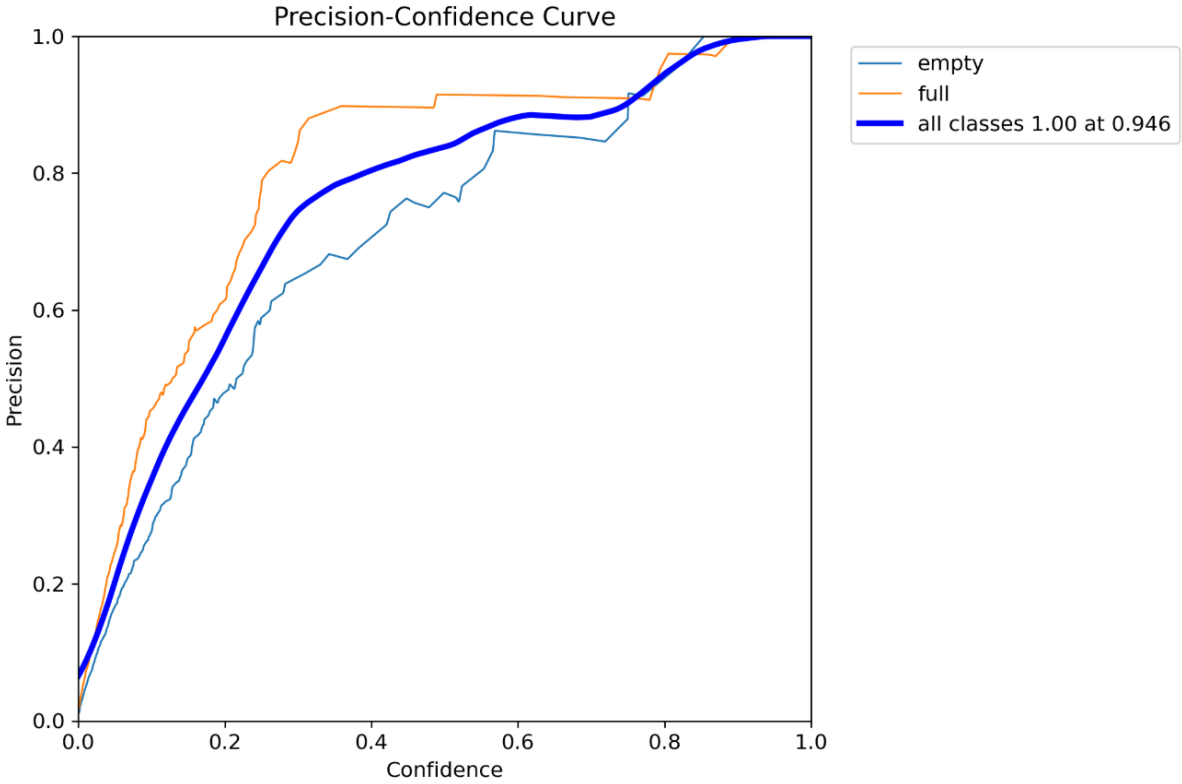

Fig 5.6 Raportul dintre precizie și încredere

- Raportul dintre precizie şi recall

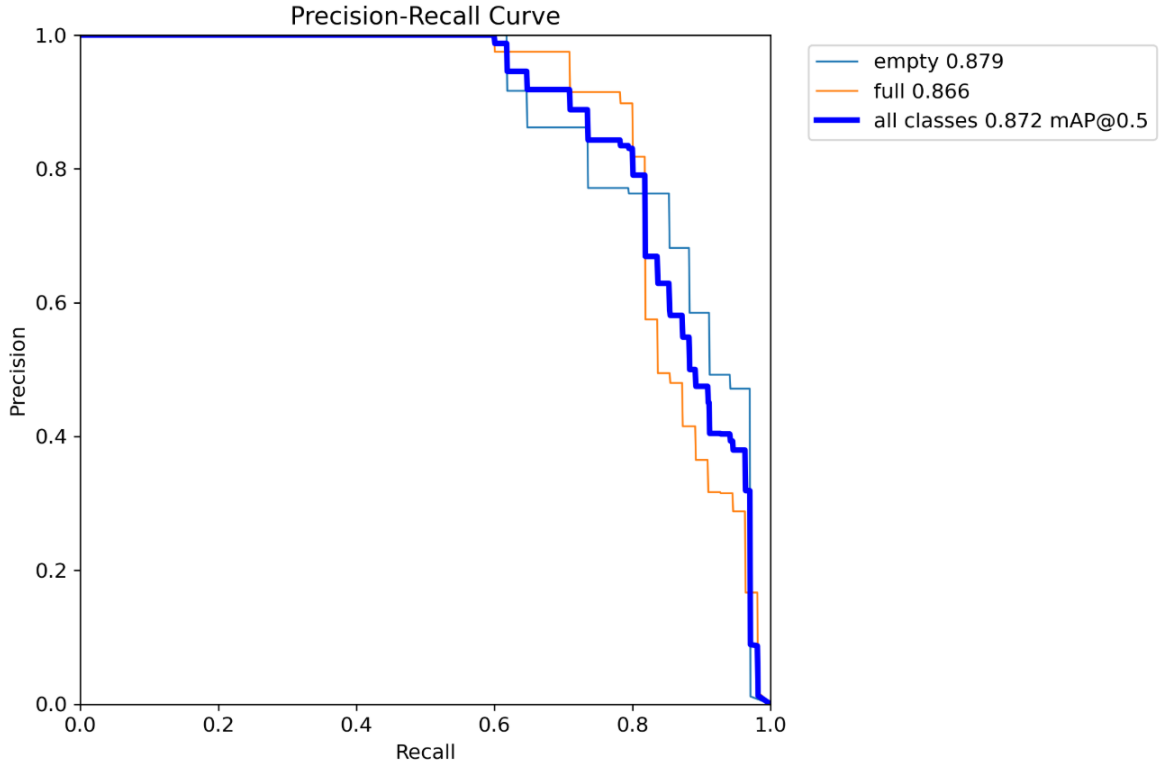

Fig 5.7 Raportul dintre precizie şi recall

- Raportul dintre recall şi încredere

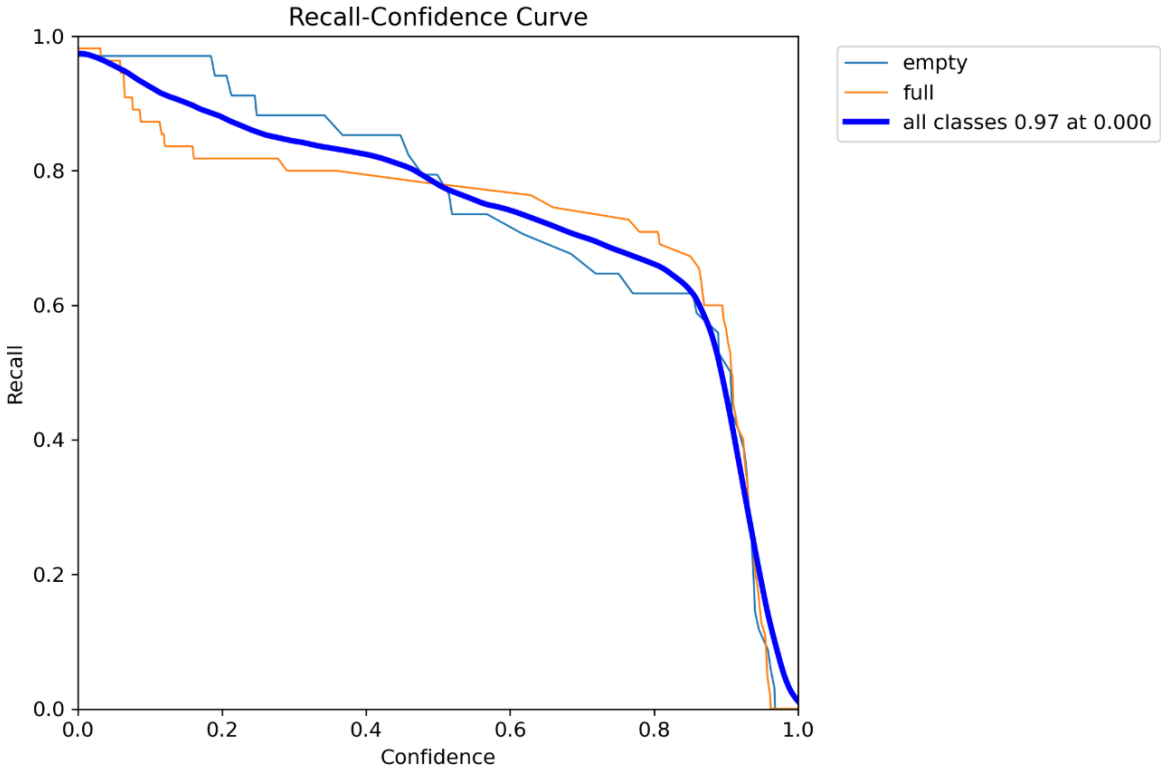

Fig 5.8 Raportul dintre Recall şi încredere

- Matricea de confuzie simplă şi normalizată

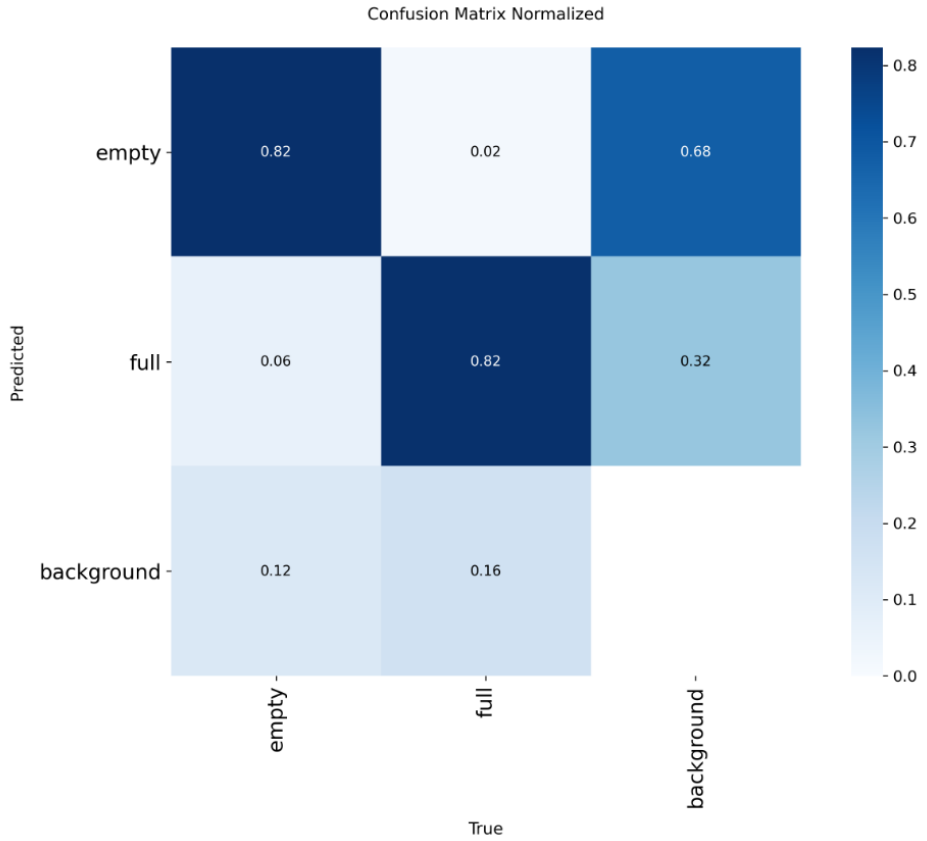

Fig 5.9 Matricea de confuzie normalizată

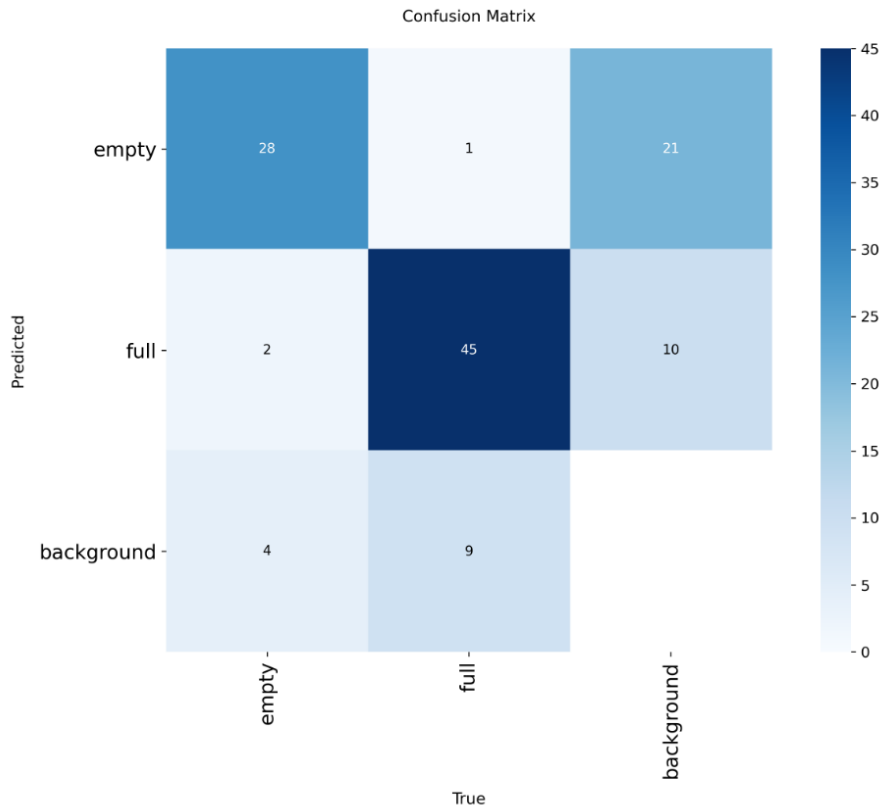

Fig 5.10 Matricea de confuzie simplă

- Prezentarea raportului general

```
     Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
       all         68         89      0.828      0.811      0.872      0.738
     empty         29         34      0.759      0.835      0.879      0.779
      full         47         55      0.896      0.786      0.866      0.698
Speed: 0.2ms preprocess, 12.1ms inference, 0.0ms loss, 1.4ms postprocess per image
```

Fig 5.11 Raportul general

5.2.2 YOLO11s

- Raportul F1 şi încredere

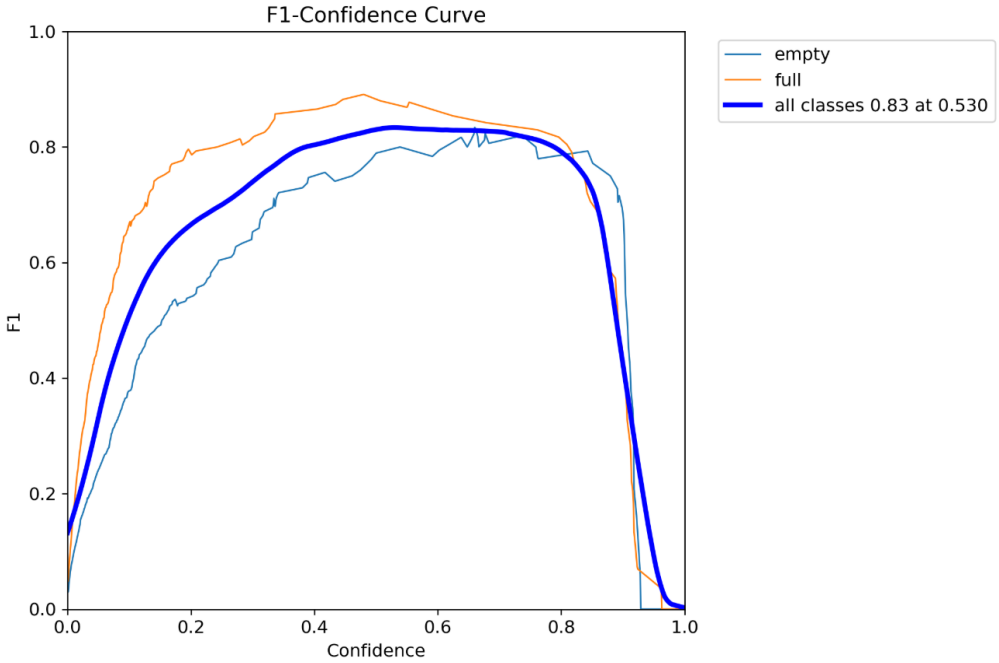

Fig 5.12 Raportul F1 şi încredere

- Raportul dintre precizie şi încredere

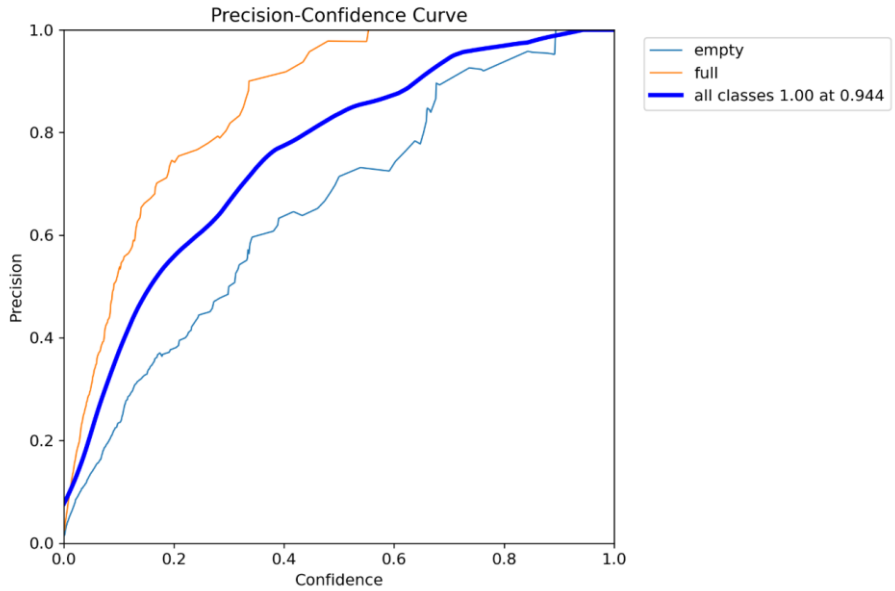

Fig 5.13 Raportul dintre precizie şi încredere

- Raportul dintre precizie şi recall

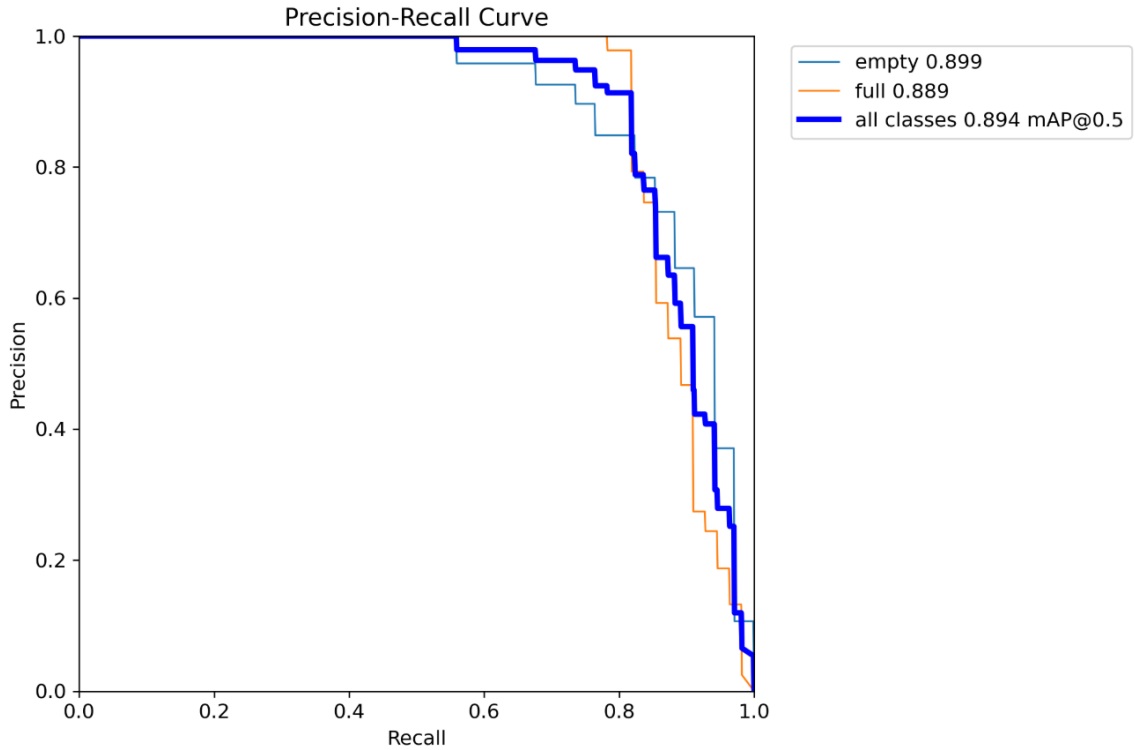

Fig 5.14 Raportul dintre precizie şi recall

- Raportul dintre recall şi încredere

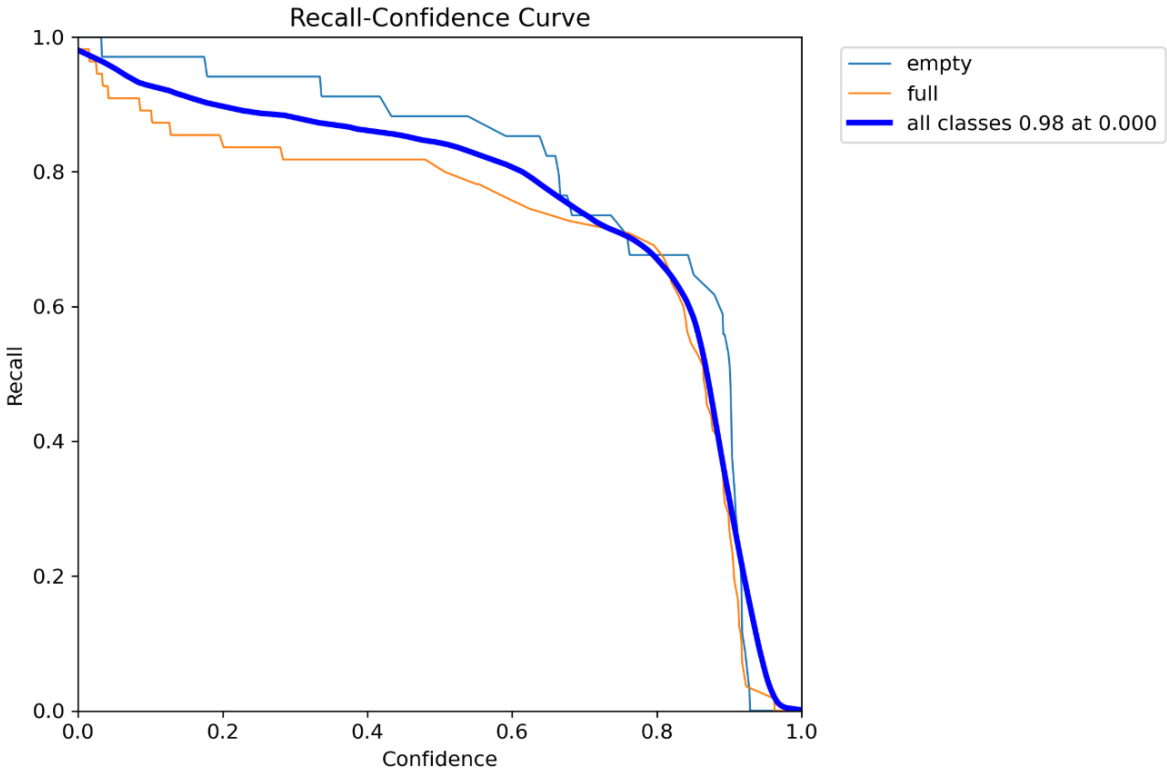

Fig 5.15 Raportul dintre recall şi încredere

- Matricea de confuzie simplă şi normalizată

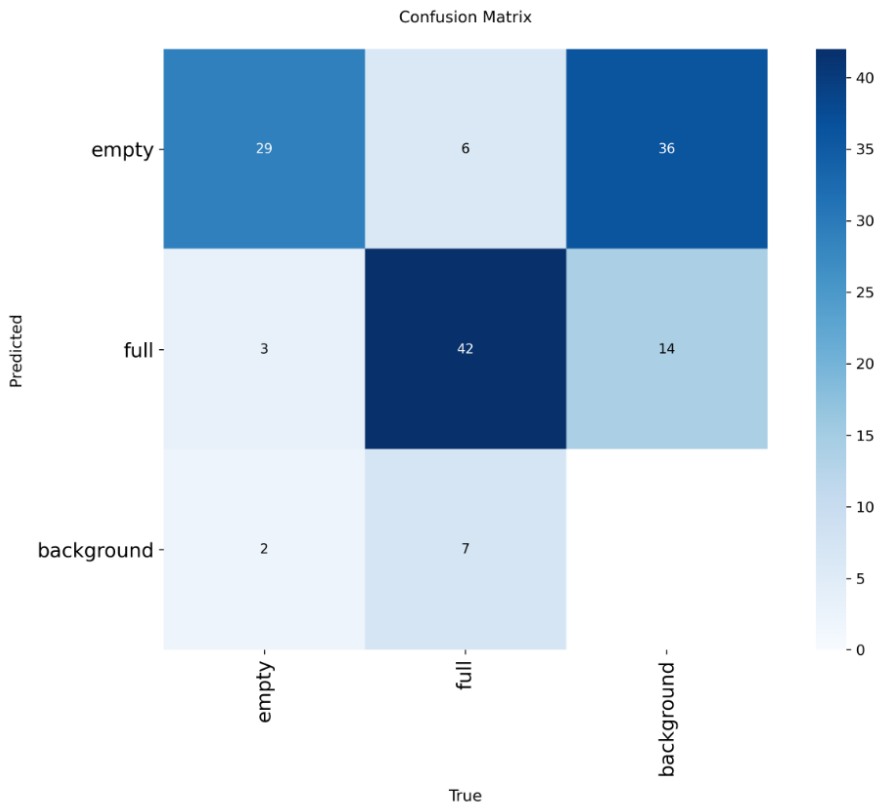

Fig 5.16 Matricea de confuzie simplă

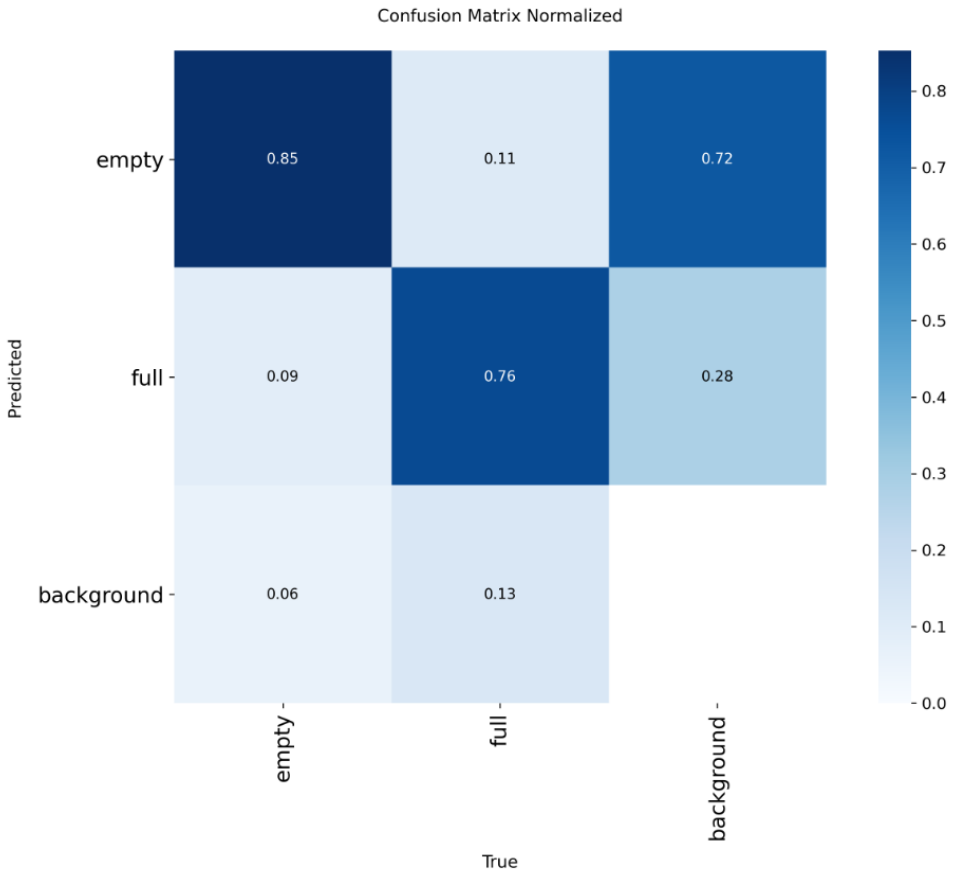

Fig 5.17 Matricea de confuzie normalizată

- Prezentarea raportului general

```
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all         68         89      0.853      0.836      0.894      0.741
                 empty         29         34      0.728      0.882      0.899      0.774
                  full         47         55      0.978       0.79      0.889      0.708
Speed: 0.1ms preprocess, 18.7ms inference, 0.0ms loss, 0.8ms postprocess per image
```

Fig 5.18 Raportul general

5.2.1 YOLO11m

- Raportul F1 și încredere

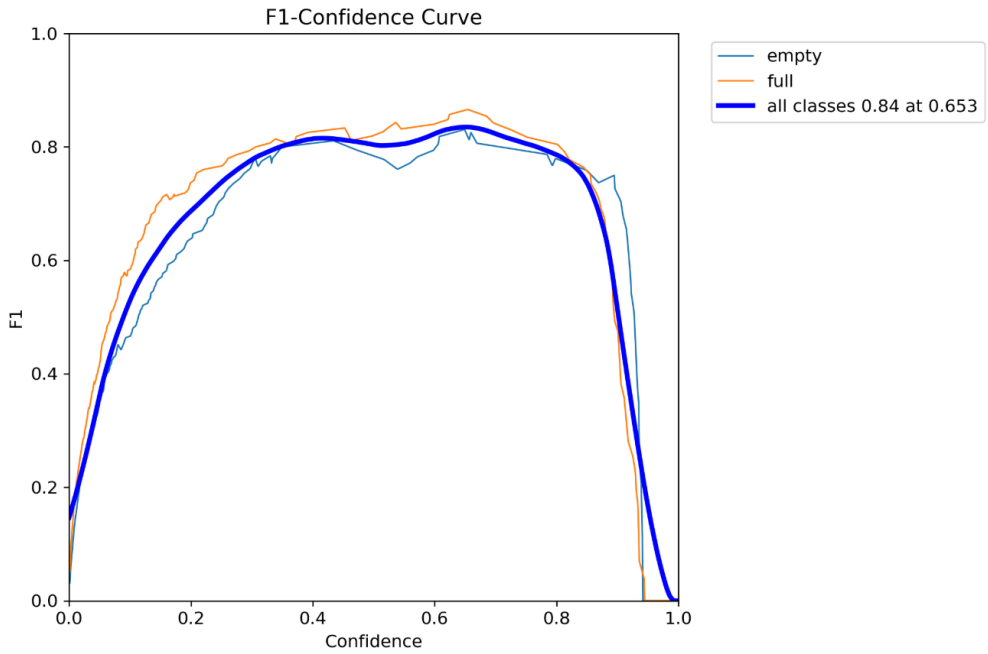

Fig 5.19 Raportul F1 și încredere

- Raportul dintre precizie și încredere

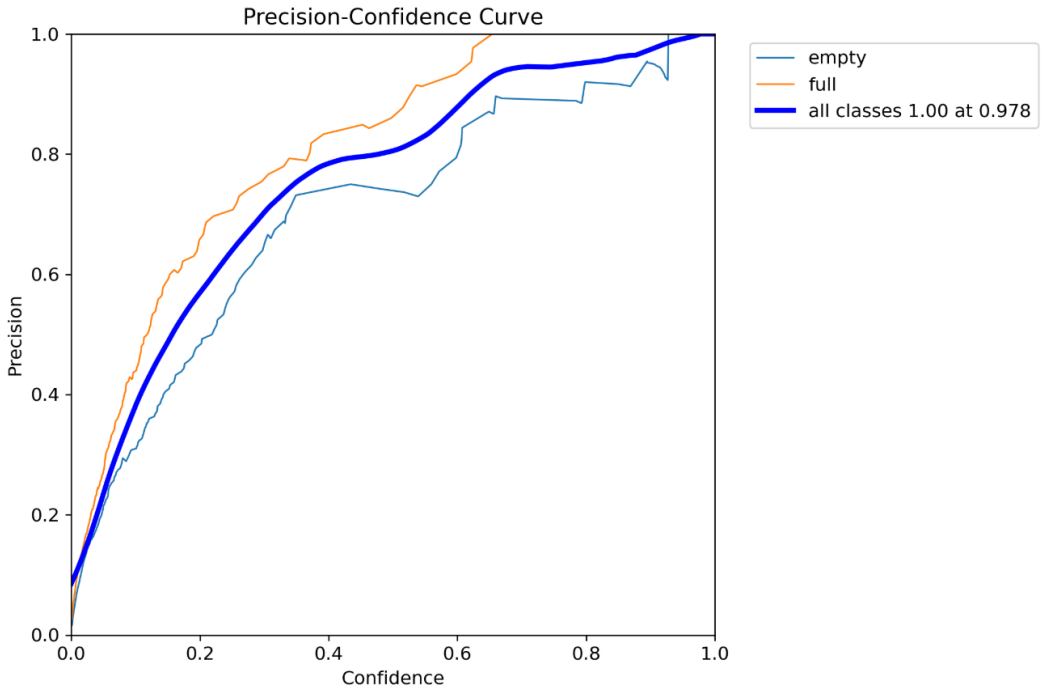

Fig 5.20 Raportul dintre precizie și încredere

- Raportul dintre precizie şi recall

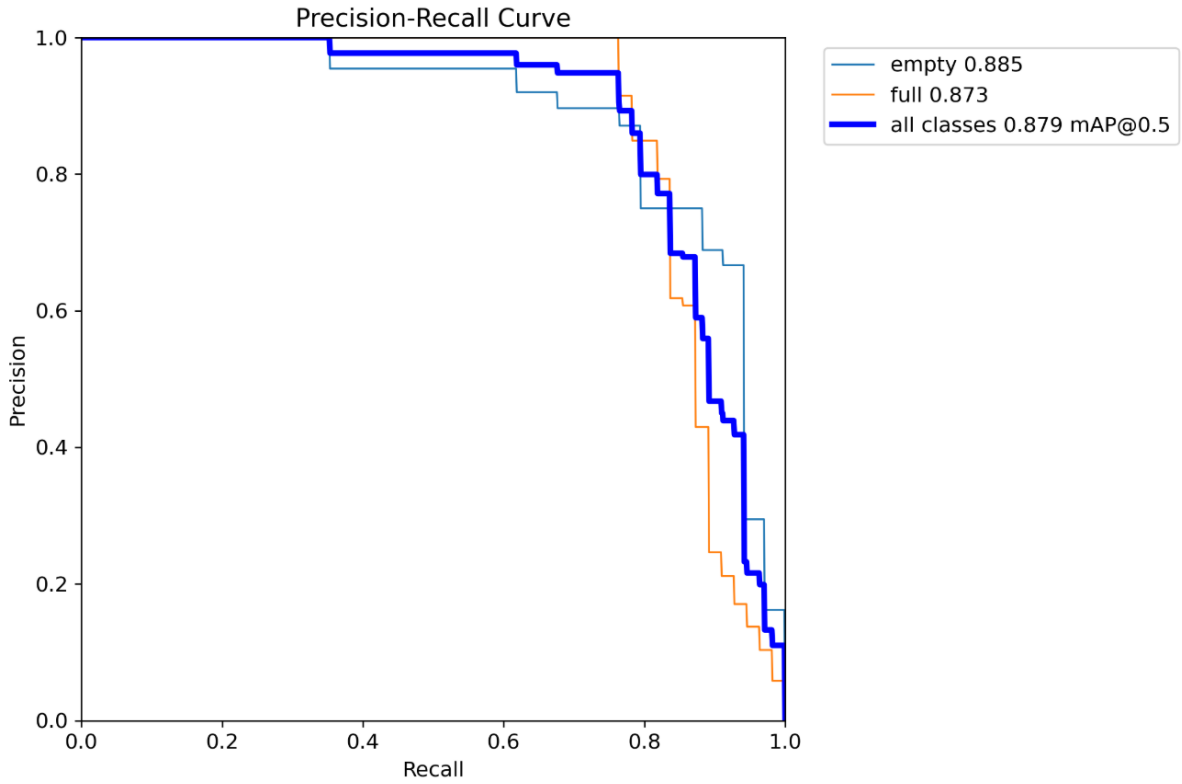

Fig 5.21 Raportul dintre precizie şi recall

- Raportul dintre recall şi încredere

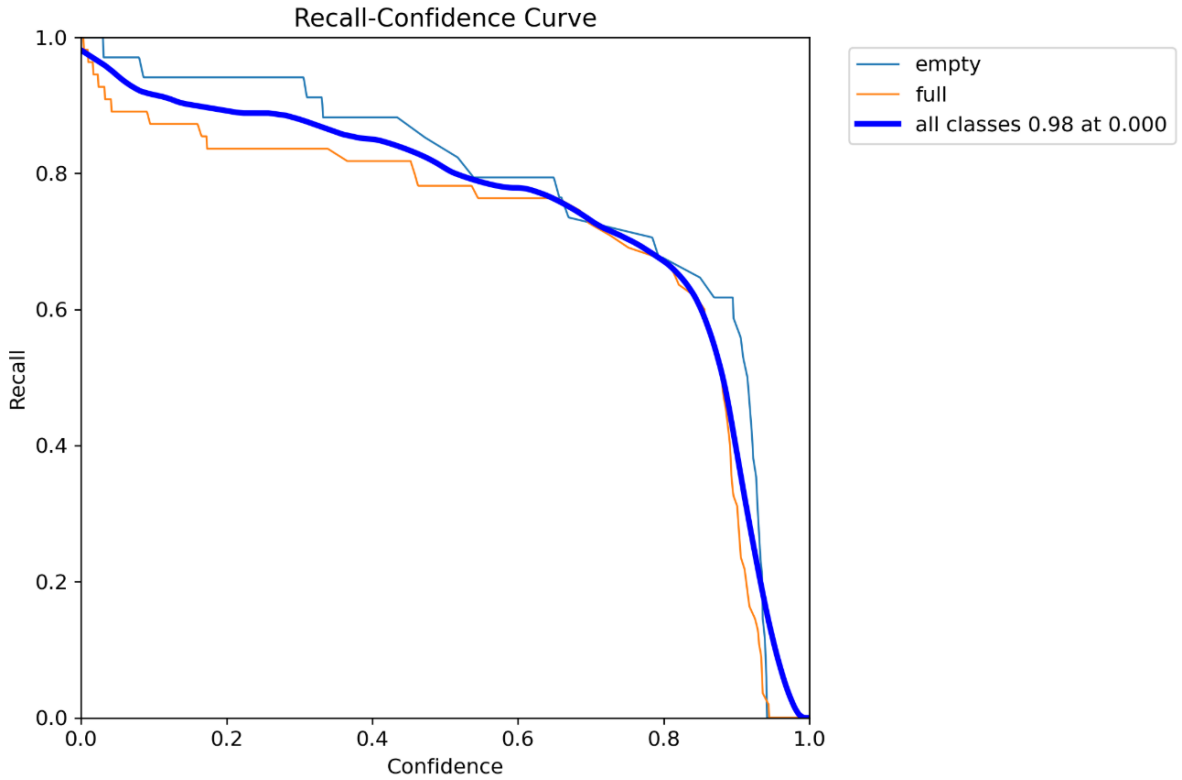

Fig 5.22 Raportul dintre recall şi încredere

- Matricea de confuzie simplă şi normalizată

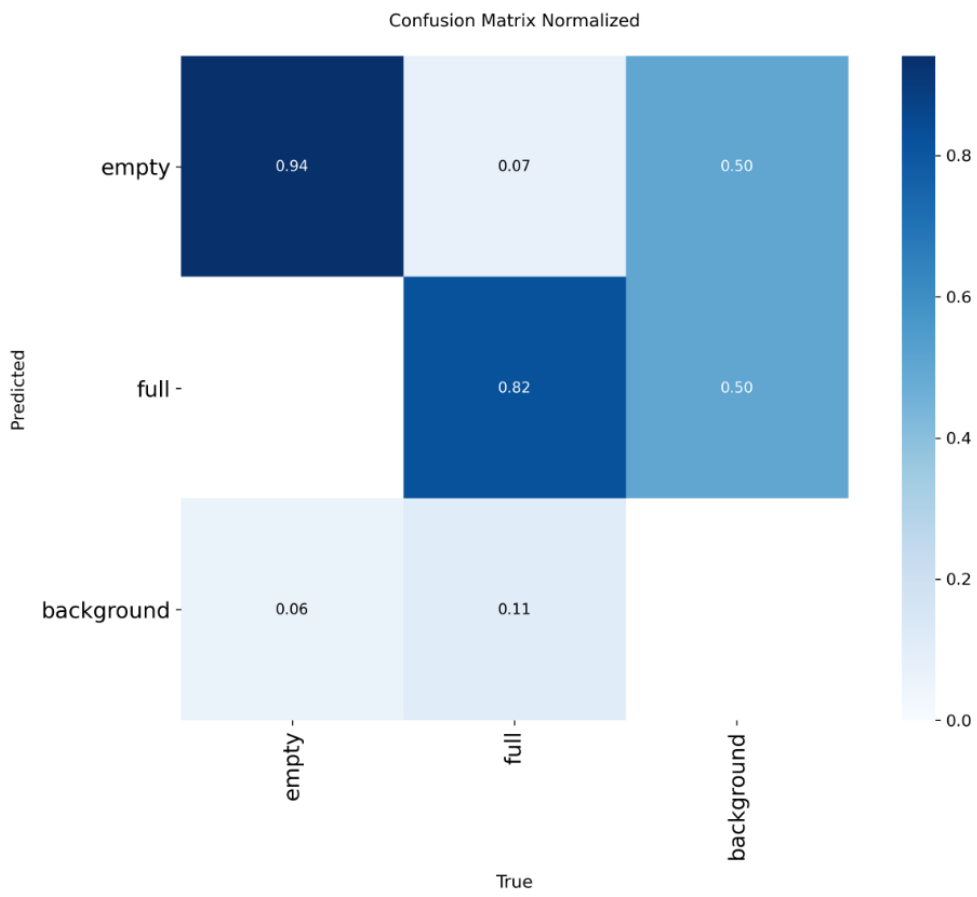

Fig 5.23 Matricea de confuzie simplă normalizată

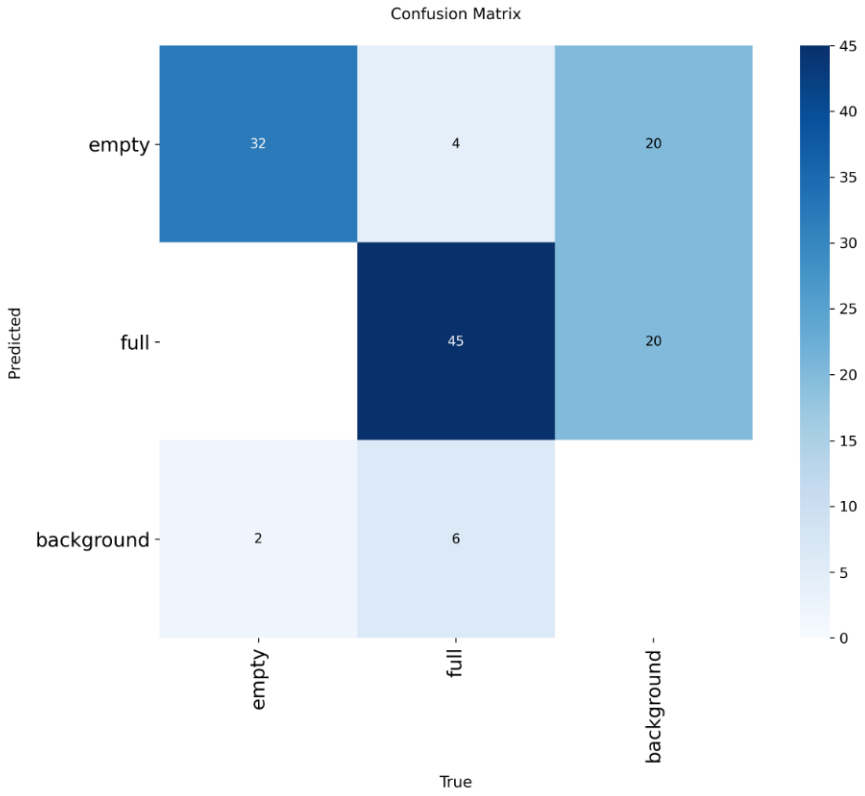

Fig 5.24 Matricea de confuzie

- Prezentarea raportului general

```
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all         68         89      0.934      0.772      0.879      0.738
                 empty         29         34      0.869       0.78      0.885      0.766
                  full         47         55      0.999      0.764      0.873       0.71
Speed: 0.1ms preprocess, 45.0ms inference, 0.0ms loss, 0.8ms postprocess per image
```

Fig 5.24 Prezentarea raportului general

CAPITOLUL 6. DESCRIERE APLICAȚIEI

În acest capitol vom prezenta aplicația și vom introduce capturi de ecran corespunzătoare pentru fiecare secțiune și caracterisitcă a acesteia.

6.1 Meniul și paginile aplicației

Fig 6.1 Pagina de descriere „Home" cu meniul activat

Fig 6.2 Pagina de descriere „Home" cu meniul dezactivat

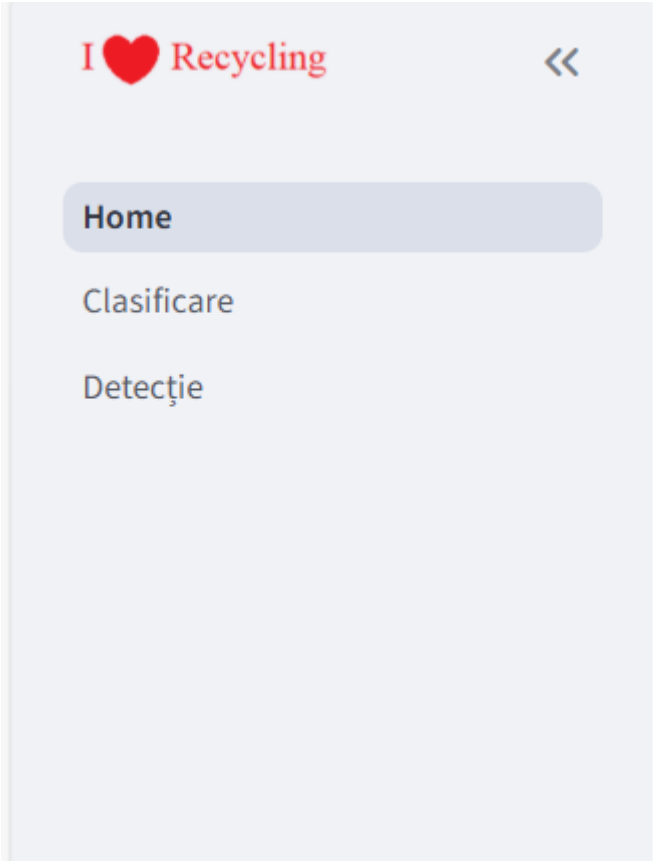

Fig 6.3 Privire detaliată asupra meniului

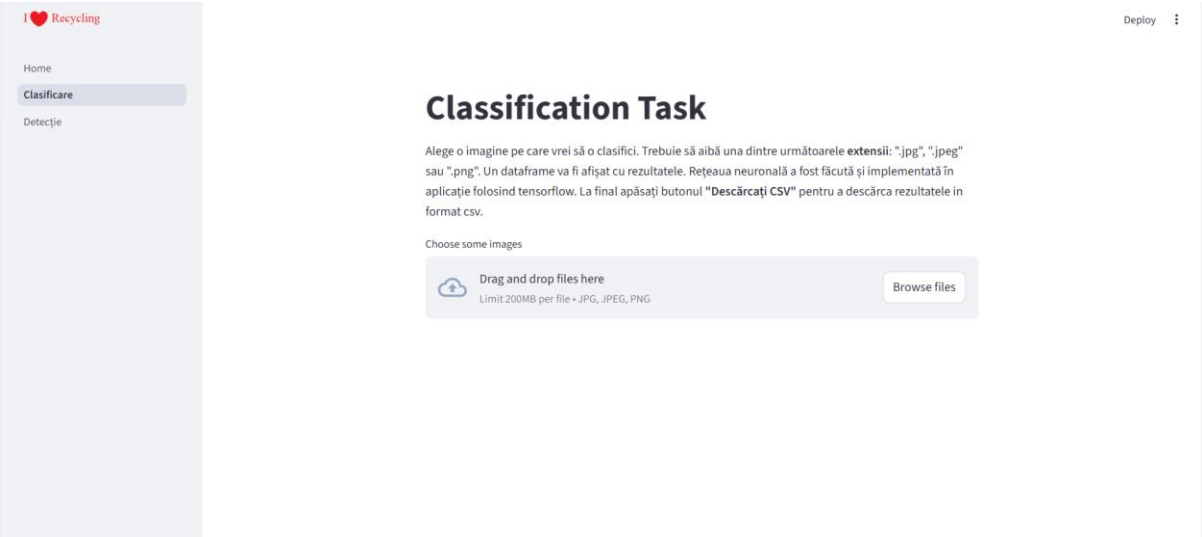

Fig 6.4 Pagina de clasificare a imaginilor

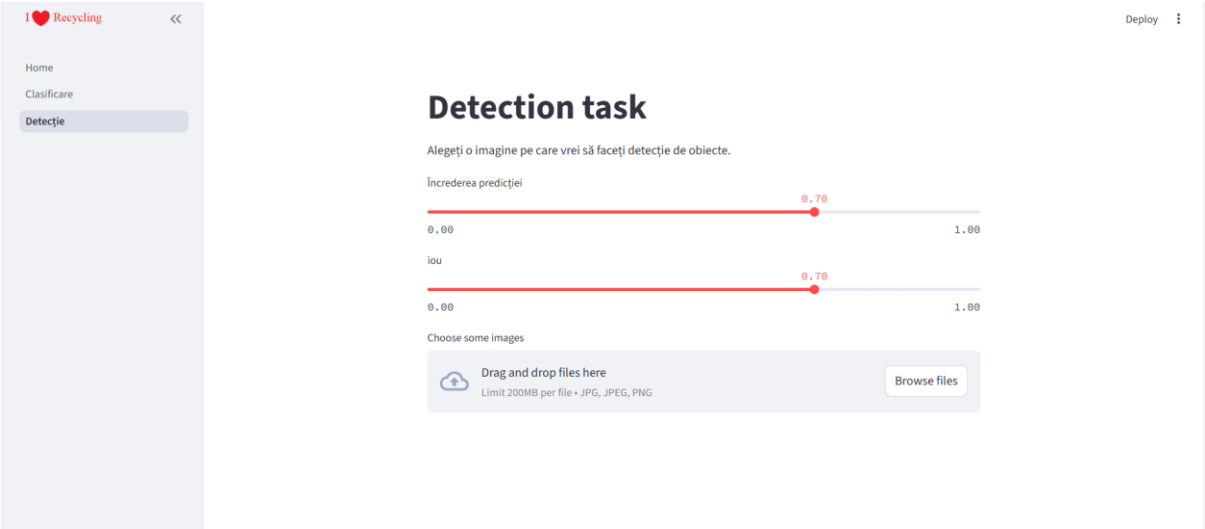

Fig 6.5 Pagina de detectare a coșurilor de gunoi în imagini

6.2 Detalierea procesului de clasificare al imaginilor

În această secţiune vom prezenta elementele grafice din procesul de clasificare al imaginilor:

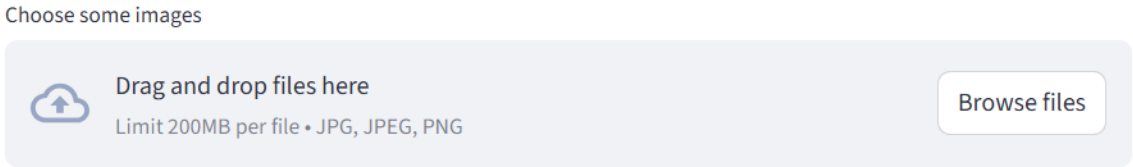

Fig 6.6 Butonul de încărcare al imaginilor

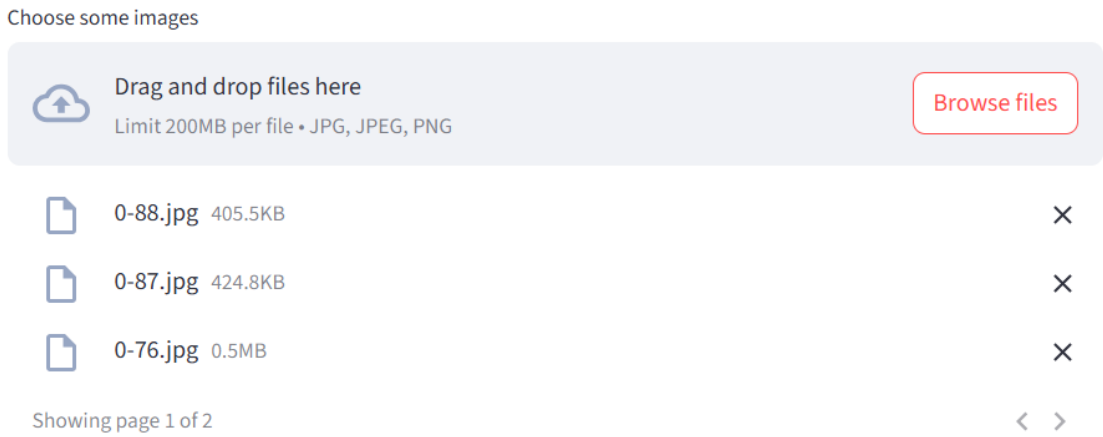

Fig 6.7 Butonul după încărcarea unor imagini

	nume_imagine	clasificare
0	0-20.jpg	imagine cos gol
1	0-21.jpg	imagine cos gol
2	0-76.jpg	imagine cos gol
3	0-87.jpg	imagine cos gol
4	0-88.jpg	imagine cos gol

Fig 6.8 Tabelul cu prezentarea predicţiilor

Fig 6.9 Butonul de descărcare a predicţiilor afişate în tabel

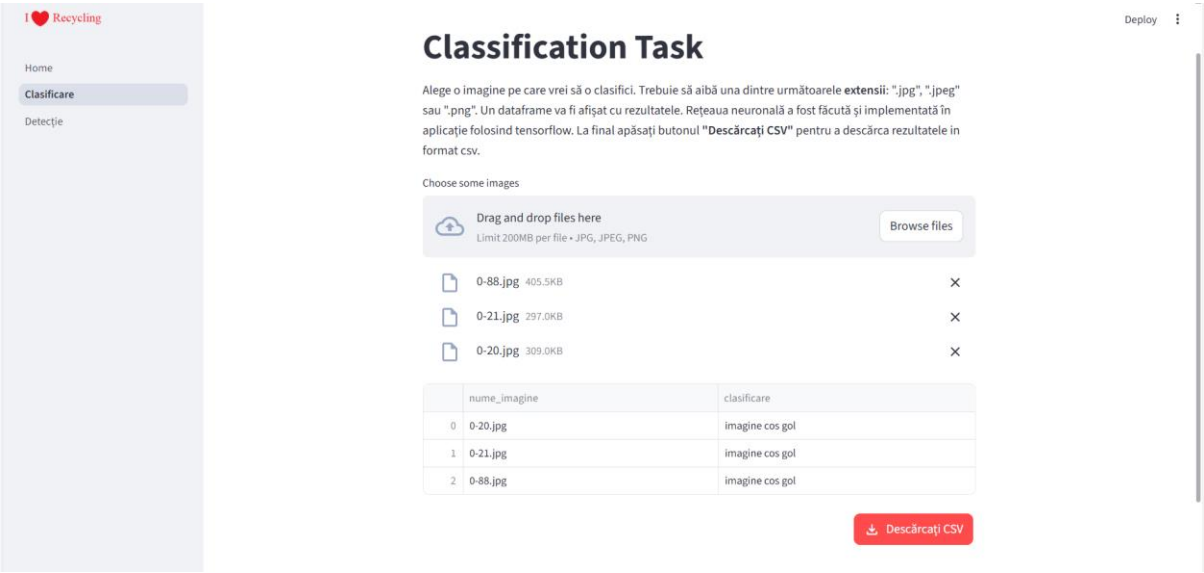

Fig 6.10 Aspectul paginii de clasificare cu imagini încărcate şi predicţii realizate

6.3 Detecţia în imagini

În această secţiune vom prezenta elemente grafice din procesul de detectare a obiectelor (coşurilor de gunoi) în imagini.

Fig 6.11 Slidebar de selectare al nivelului de încredere

Fig 6.12 Slidebar al selecţiei indicelui „Intersection over Union" (IoU)

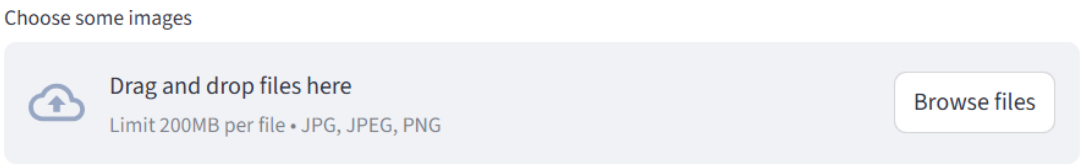

Fig 6.13 Buton de încărcare al imaginilor

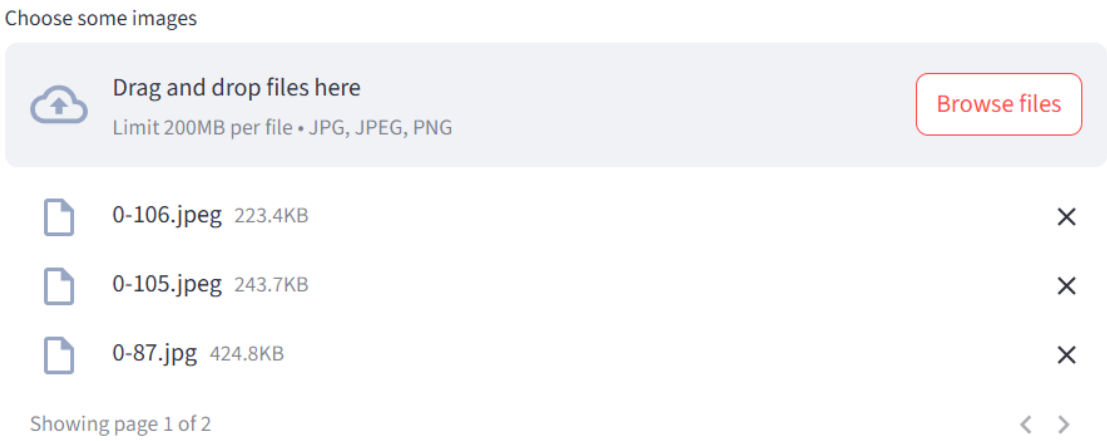

Fig 6.14 Buton de selecţie al imaginilor cu imagini încărcate

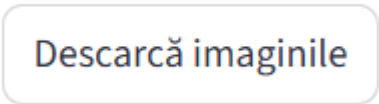

Fig 6.15 Buton de descărcare al imaginilor cu obiecte detectate

Fig 6.16 Exemplu de imagine după detecție

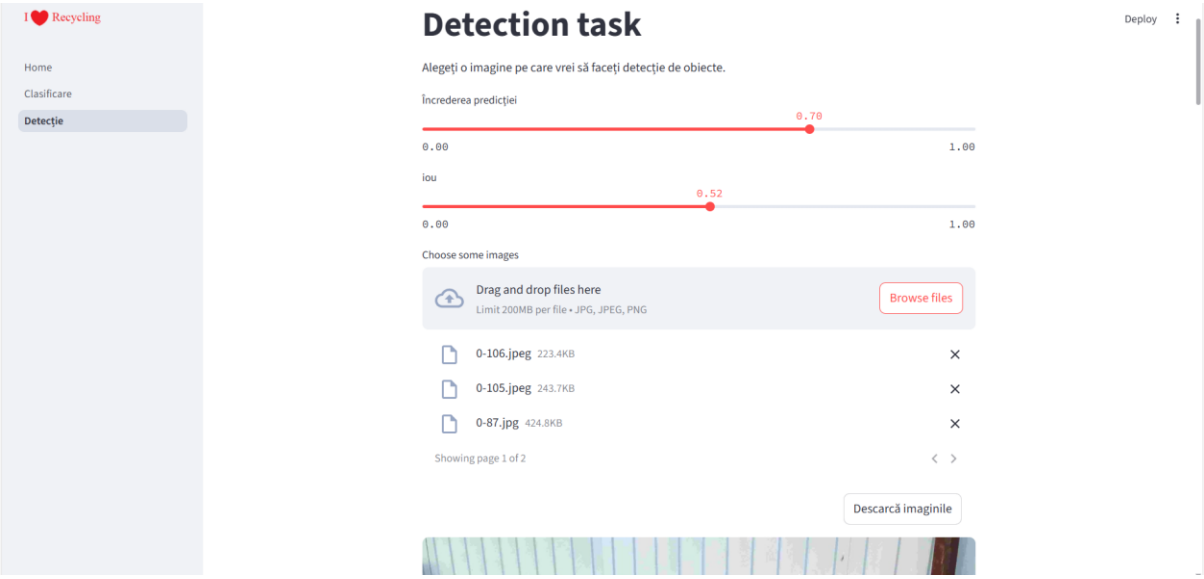

Fig 6.17 Aspectul paginii după ce sunt încărcate imagini

CONCLUZII

În cadrul acestei lucrări de disertaţie a fost realizată o aplicaţie care poate efectua operaţia de clasificare a imaginilor (coş gol/ coş plin) şi care poate detecta coşurile de gunoi din imagine. Pentru a fi funcţională, au fost antrenate mai multe modele de inteligenţă artificială pentru a clasifica imagini şi pentru a detecta obiecte.

În urma antrenamentelor, reţelele neuronale au ajuns la rezultate bune. Reţeaua CNN cu cea mai bună acurateţe la clasificare a fost prima reţea(CNN-G1).

În cazul detectării de obiecte, reţeaua care a avut cel mai bun indicator mAP50-95 este YOLO11s.

Această aplicaţie poate fi dezvoltată în funcţie de nevoile celor care vor să o utilizeze.

Se poate crea un sistem bazat pe roluri de utilizator pentru mai multă securitate.

O altă direcţie posibilă ar fi antrenarea reţelelor neuronale astfel încât să facă inferenţe în timp real şi informaţiile să fie transmise către firma de salubritate a oraşului, care poate trimite ţintit camioane pentru colectarea deşeurilor.

BIBLIOGRAFIE

[1] https://docs.streamlit.io/ , 13.07.2025
[2] https://www.tensorflow.org/learn , 13.07.2025
[3] https://matplotlib.org/ , 13.07.2025
[4] https://numpy.org/doc/stable/user/whatisnumpy.html , 13.07.2025
[5] https://keras.io/, 13.07.2025
[6] https://pypi.org/project/visualkeras/ 13.07.2025
[7] https://scikit-learn.org/stable/getting_started.html , 13.07.2025
[8] https://seaborn.pydata.org/ , 13.07.2025
[9] https://www.ultralytics.com/glossary/object-detection , 13.07.2025
[10] https://docs.ultralytics.com/guides/yolo-data-augmentation/, 13.07.2025
[11] https://www.purpose.jobs/blog/top-climate-tech-companies-2025, 14.07.2025
[12] Diqi, Mohammad. (2022). Waste Classification using CNN Algorithm. International Conference on Information Science and Technology Innovation (ICoSTEC). 1. 130-135. 10.35842/icostec.v1i1.17.
[13] Kumar Lilhore, Dr & Simaiya, Sarita & Dalal, Surjeet & Damaševičius, Robertas. (2023). A smart waste classification model using hybrid CNN-LSTM with transfer learning for sustainable environment. Multimedia Tools and Applications. 83. 10.1007/s11042-023-16677-z.
[14] https://www.kaggle.com/datasets/lenferdetroud/trash-bins-dataset , 14.07.2025
[15] https://www.unrefugees.org/news/a-crisis-that-can-no-longer-be-ignored/ , 14.07.2025
[16] https://www.microsoft.com/en-us/microsoft-copilot/for-individuals/do-more-with-ai/general-ai/what-is-copilot?msockid=256ca2d8ecdd64211cf6b73eed1c65cb&form=MA13KP , 14.07.2025