



Universitatea Politehnica Bucureşti  
Facultatea de Automatică și Calculatoare  
Departamentul de Automatică și Ingineria Sistemelor

## Vedere Artificială

**Recunoasterea plantelor individuale din culturi de tutun utilizând imagini de la distanță și rețelele neuronale**

Autor  
Anghelin Mihai-Robert  
Roșcan Costin-Ștefan

Coordonator  
Prof. dr. ing. Dan Popescu

Bucureşti, 2024

# Cuprins

<b>1</b>	<b>Introducere</b>	<b>3</b>
<b>2</b>	<b>Stadiul Actual</b>	<b>4</b>
<b>3</b>	<b>Materiale și metode</b>	<b>6</b>
3.1	Baza de date . . . . .	6
3.2	YOLO (You Only Look Once) . . . . .	7
3.3	DETR (DEtection TRansformers) . . . . .	10
<b>4</b>	<b>Antrenarea rețelelor și rezultate experimentale</b>	<b>12</b>
4.1	Antrenarea rețelelor . . . . .	12
4.1.1	YOLOv8 . . . . .	12
4.1.2	DETR . . . . .	12
4.2	Rezultate experimentale . . . . .	13
4.2.1	YOLOv8 . . . . .	13
4.2.2	DETR . . . . .	17
<b>5</b>	<b>Concluzie</b>	<b>21</b>
	<b>Bibliografie</b>	<b>22</b>

# 1 Introducere

Recunoașterea plantelor individuale din culturi utilizând imagini de la distanță este importantă din mai multe motive ce țin de agricultură, eficiență economică și sustenabilitate. Pentru acest studiu se vor folosi imagini cu plante de tutun fotografiate individual pentru a testa rețele neuronale pentru a fi recunoscute și diferențiate față de buruieni ce pot afecta creșterea planței.

Folosind aceste rețele neuronale poate ajuta fermierii să monitorizeze starea de sănătate a culturilor în timp real, astfel se pot identifica simptomele bolilor, dăunătorilor sau deficiențelor de nutrienți mult mai rapid și precis decât metodele tradiționale de monitorizare.

Prin monitorizarea atentă a plantelor, fermierii pot maximiza randamentul și calitatea producției de tutun. Tehnologia poate ajuta la identificarea momentului optim pentru recoltare și la asigurarea faptului că plantele cresc în condiții optime. Automatizarea procesului de monitorizare a plantelor reduce timpul și resursele necesare pentru inspecții manuale, eliberând astfel resursele umane pentru alte activități și permite o gestionare mai eficientă a culturii.

Tutunul este cultivat în peste 120 de țări din întreaga lume, acoperind milioane de hectare de teren. În Pakistan, este considerat o cultură importantă, deoarece generează venituri substanțiale. Conform unei estimări, în zonele rurale ale țării se produc anual 80.000–90.000 de tone de tutun Virginia cu uscare prin fum (*Nicotiana Tabacum*). Pe lângă faptul că este o cultură profitabilă, este important de subliniat că frunza de tutun este extrem de suscepțibilă la dăunători și agenți patogeni, iar culturile necesită eforturi și îngrijirimeticuloase pentru a le proteja de insectele sezoniere. [1]

Fermierii se bazează pe utilizarea metodelor convenționale de pulverizare cu produse agrochimice pentru a combate acești dăunători și agenți patogeni. Pesticidele sunt aplicate plantelor de tutun de obicei de cinci până la șase ori într-un sezon (pe parcursul a trei luni), ceea ce face ca această cultură să fie foarte dependentă de pesticide. În ciuda pericolelor, pulverizarea agrochimică este încă practicată frecvent, deoarece reprezintă un mijloc viabil și economic de a proteja cultura de tutun de dăunători și agenți patogeni. Soluția, prin urmare, nu constă în eliminarea utilizării agrochimicalelor, ci mai degrabă în optimizarea aplicării acestora prin adoptarea unor tehnici și metodologii avansate. [1]

Acest studiu se axează pe o detecție automata a plantelor de tutun folosind doi dintre algoritmii consacrați - YOLOv8<sup>1</sup> (You Only Look Once) și DETR<sup>2</sup> (DEtection TRansformers). Se vor ajusta ("fine tune") aceste rețele pentru a crește robustețea și acuratețea modelelor pentru recunoașterea plantelor specifice și a buruienilor în condiții imperfecte. În general YOLO este cunoscut pentru viteza sa de detecție, dar având o acuratețe mai slabă, în schimb DETR necesită resurse mai multe de procesare și este mai lent la detecție, fiind mai precis. De asemenea YOLO este mult mai simplu de antrenat oferind un API ușor de folosit, necesitând hardware nu foarte performant pentru a scoate rezultate bune.

---

<sup>1</sup><https://github.com/ultralytics/ultralytics>

<sup>2</sup>[https://huggingface.co/docs/transformers/main/en/model\\_doc/detr](https://huggingface.co/docs/transformers/main/en/model_doc/detr)

## 2 Stadiul Actual

Recunoașterea plantelor individuale din culturi de tutun utilizând imagini de la distanță și rețele neuronale a evoluat semnificativ datorită necesității de a avea soluții eficiente și cu costuri reduse pentru estimarea automată a producției. Metodele tradiționale de estimare, care se bazează pe identificarea și numărarea manuală, sunt laborioase și consumatoare de timp. Studiile recente folosesc date de teledetectie, însă acestea sunt costisitoare și necesită resurse computaționale semnificative. [7]

În ultimul deceniu, comunitatea științifică a acordat o atenție semnificativă dezvoltării unor soluții precise și țintite de pulverizare pentru reducerea aportului de pesticide/herbicide, cu scopul de a diminua efectele secundare ale acestora. Un aspect esențial al acestor sisteme inteligente de pulverizare de precizie este capacitatea de a discerne scenariul din câmpul agricol, adică de a înțelege necesitățile culturii sau de a identifica locurile unde se află buruienile. Viziunea computerizată, o tehnologie care permite unei mașini să analizeze și să înțeleagă automat lumea vizuală, a fost îmbunătățită enorm în ultimii ani și este acum utilizată pe scară largă în multe probleme ale agriculturii de precizie. [1] [8]

Detectarea și clasificarea plantelor din culturi de tutun reprezintă o provocare, în special, datorită variațiilor semnificative din teren. Aceste variații includ dimensiunile diferite ale frunzelor în diverse stadii de creștere, intensitățile variante ale luminii, texturile diverse ale solului, culorile schimbătoare ale frunzelor cauzate de nivelurile diferite de apă, densitățile ridicate de buruieni și mascare a plantelor de cultură de către buruieni.[1] [8]

Majoritatea tehniciilor existente pentru detectarea plantelor și buruienilor, bazate pe visiune, utilizează metode tradiționale de învățare automată. Deși aceste tehnici au obținut precizii ridicate, formularea caracteristicilor artizanale și generarea unei funcții de decizie pe baza caracteristicilor extrase le fac mai puțin robuste. Astfel, ele nu sunt preferate pentru detectarea plantelor de tutun și a buruienilor în condițiile variante și complexe din câmpurile de tutun, din cauza capacitațiilor lor slabe de generalizare.[1] [8]

Metodele de detectie a obiectelor au evoluat, pornind de la modelul DPM<sup>1</sup> propus de Felzenszwalb et al. în 2009, care utilizează HOG<sup>2</sup> pentru extragerea caracteristicilor și SVM<sup>3</sup> pentru clasificare, dar este inefficient temporal din cauza căutării în fereastră glisantă. Algoritmi precum 'selective search' și 'edge boxes' au fost dezvoltăți pentru a înlocui etapa de căutare, reducând numărul de propuneri și crescând acuratețea. R-CNN, propus în 2014, a introdus trei etape: obținerea regiunilor de interes prin selective search, extragerea caracteristicilor cu CNN și clasificarea acestor caracteristici cu SVM-uri multiple. Ulterior, selective search a fost înlocuit cu RPN pentru creșterea vitezei și menținerea acurateței. YOLO, propus în 2016, elimină etapa de propunere a regiunilor, fiind fezabil pentru detecția în timp real, dar cu acuratețe mai mică. YOLOv3 a îmbunătățit semnificativ performanțele, păstrând proprietățile de timp real.[7]

În contextul recunoașterii plantelor individuale din culturi de tutun utilizând imagini de la distanță și rețele neuronale, aceste abordări de învățare profundă sunt deosebit de promițătoare. Ele pot aborda provocările complexe ale câmpurilor de tutun, inclusiv variațiile de iluminare,

<sup>1</sup><https://arxiv.org/abs/1409.5403>

<sup>2</sup><https://medium.com/@prantiksen4/training-a-neural-network-with-histogram-of-oriented-gradients-hog-feature-extrac-1f3a2a23a23>

<sup>3</sup><https://scikit-learn.org/stable/modules/svm.html>

texturile solului, densitățile mari de buruieni și asemănările vizuale dintre plantele de tutun și buruieni. Studiul nostru propune și evaluează aplicarea unui clasificator bazat pe rețele neuronale conoluționale pentru detectarea plantelor de tutun și a buruienilor în scenarii reale de câmp, oferind o soluție mai robustă și adaptabilă în comparație cu tehniciile tradiționale de învățare automată.[1] [8]

# 3 Materiale și metode

## 3.1 Baza de date

Pentru acest studiu am găsit și folosit două baze de date cu imagini annoteate.

Prima se numește TobSet<sup>1</sup> și a fost construită pentru un studiu asemănător cu imagini din câmpurile de tutun din Swabi, Khyber Pakhtunkhwa, Pakistan (34°09'07.3" N 72°21'36.2" E). Aceasta conține 7000 de imagini cu plante de tabac și 1000 de imagini cu pământ gol sau cu buruieni care cresc în cultura de tutun. Imaginele au o rezoluție de 640x480px și au fost făcute cu o camera digitală de 13-megapixeli cu un senzor de imagine CMOS (IMX258 Exmor RS de la Sony), distanță focală (focal distance) de 28 mm, unghi de vizualizare orizontal(horizontal FOV) de 65.4° și unghi de vizualizare vertical(vertical FOV) de 51.4°. Acest set de date a fost construit pe o perioadă de 2 luni, adică începând cu prima săptămână de transplantare a răsadurilor de tutun din paturi de semințe până la momentul când plantele ating o înălțime aproximativă de 1.25 m. Toate imaginile din setul de date au fost capturate manual de către cercetători în lunile iunie și iulie 2020. Nu s-au folosit umbre artificiale și surse de iluminat în timpul colectării imaginilor. În timpul achiziției imaginilor, înălțimea camerei a fost ajustată între 1 și 1.5 m. Pentru a menține diversitatea în setul de date, toate imaginile din TobSet sunt capturate sub mai mulți factori de variație: diferite stadii de creștere, diferite momente ale zilei, condiții de iluminare și meteorologice variate (adică în zile normale, însorite și înnorate), și ocaziile vizuale ale frunzelor de cultură de către buruieni, etc.<sup>[1]</sup> Câteva imagini din acest set de date se găsesc în Figura 3.1. Imaginile din setul de date TobSet au fost annoteate folosind LabelImg<sup>2</sup>.



Figura 3.1: Ilustrație a factorilor de variațiune din câmpul de tutun [1]

<sup>1</sup><https://github.com/mshahabalam/TobSet>

<sup>2</sup><https://github.com/HumanSignal/labelImg>

Al doilea set de date<sup>3</sup> folosit a fost găsit pe Roboflow<sup>4</sup>. Acesta oferă 14640 imagini pentru antrenament, 1387 de imagini valide și 729 imagini de test. Acest set este adnotat în mai multe formate, noi folosind formatul YOLOv8, pentru antrenarea rețelei YOLO, iar pentru DETR am folosit formatul COCO JSON. Setul de date a fost încărcat de utilizatorul "heloooooooo"<sup>5</sup> care a mai lucrat și la alte seturi de date asemănătoare. Imaginile au o rezoluție de 640x480px și conțin 8046 de plante de tutun și 4396 buruieni adnotate. [5]. Ultima actualizare a setului de date este versiunea 9 la data 30.11.2022

Am folosit cel de-al doilea set de date pentru antrenarea celor două rețele, iar pe primul l-am folosit pentru testarea lor.

## 3.2 YOLO (You Only Look Once)

Algoritmul de detectare a obiectelor YOLO este în centrul atenției pentru performanța sa remarcabilă, caracterizată de dimensiunea compactă a modelului și viteza de calcul impresionantă. Structura sa simplă permite obținerea directă a poziției și categoriei casetei delimitatoare prin intermediul unei rețele neuronale, facilitând astfel o viteză de procesare rapidă. Abordarea YOLO permite detectarea în timp real a obiectelor în imagini sau videoclipuri, folosind informații globale pentru a minimiza erorile de detectare a fundalului și pentru a codifica eficient caracteristicile esențiale ale obiectelor. Mai mult, YOLO are capacitatea de a generaliza în mod eficient caracteristicile învățate, permitând transferul lor într-o varietate de domenii. Cu toate acestea, performanța YOLO poate fi afectată în cazul obiectelor apropiate sau grupate, precum și în cazul unei poziționări inexacte a acestora, ceea ce necesită îmbunătățiri suplimentare pentru a asigura precizia și robustețea detecției. În contextul îmbunătățirii eficienței detecției, o atenție specială trebuie acordată proiectării funcției de pierdere, iar adaptarea arhitecturii YOLO poate aduce îmbunătățiri semnificative în detectarea și localizarea precisă a obiectelor.[6]

YOLOv1, cunoscut și sub denumirea de YOLO Versiunea 1, modelează detectarea obiectelor ca o problemă de regresie. O singură rețea convolutională prezice simultan mai multe casete delimitatoare și probabilități de clasă pentru acele casete. YOLOv1 împarte imaginea de intrare într-o grilă. Dacă centrul unui obiect cade într-o celulă a grilei, acea celulă este responsabilă pentru detectarea obiectului respectiv. Fiecare celulă a grilei prezice B casete delimitatoare, scoruri de încredere pentru acele casete și C probabilități de clasă pentru celula grilei. Aceste predicții sunt codificate sub formă de tensor. În procesul de testare, YOLOv1 înmulțește probabilitățile condiționate ale claselor cu predicțiile individuale de încredere ale casetelor, ceea ce ne oferă scoruri de încredere specifice fiecărei clase pentru fiecare casetă.[6]

Aceste scoruri codifică atât probabilitatea apariției acelei clase în casetă, cât și cât de bine se potrivește caseta prezisă obiectului. Fiecare casetă delimitatoare constă din 5 predicții: și încredere. Coordonatele reprezintă centrul casetei raportat la limitele celulei grilei. Lățimea și înălțimea sunt prezise raportat la întreaga imagine. De aceea, YOLOv1 utilizează pentru calcularea tensorului. Pentru evaluarea lui YOLO pe Pascal VOC, sunt folosite în mod obișnuit. Pascal VOC are 20 de clase etichetate, astfel că . Predicția finală a lui YOLOv1 este un tensor. Utilizează doar 98 de casete delimitatoare per imagine în comparație cu 2000 de la Selective Search.[6]

Rețeaua YOLOv1 are 24 de straturi convolutionale urmate de 2 straturi complet conectate. În locul modulelor de inițiere utilizate de GoogLeNet, YOLOv1 folosește simplu un strat de reducere urmat de straturi convolutionale, similar cu Lin et al. Pe Pascal VOC2007, YOLOv1 procesează imagini la 45 de cadre pe secundă (FPS), ceea ce este de două până la nouă ori

<sup>3</sup><https://universe.roboflow.com/heloooooooo/tobacco-js5vk/dataset/9>

<sup>4</sup><https://universe.roboflow.com>

<sup>5</sup><https://universe.roboflow.com/heloooooooo>

mai rapid decât Faster R-CNN. În special, Fast YOLO, o versiune rapidă a YOLO concepută pentru a împinge limitele detectării rapide a obiectelor, a atins un uimitor de 155 de cadre pe secundă.[6]

YOLO v2 dezvoltată în anul 2016, a abordat unele dintre limitările modelului original YOLO. A introdus casetele ancoră, care au ajutat la prezicerea mai precisă a casetelor delimitatoare de diferite dimensiuni și raporturi de aspect. YOLO v2 a utilizat o rețea de bază mai puternică, Darknet-19, și a fost antrenat nu numai pe setul de date original (PASCAL VOC), ci și pe setul de date COCO, ceea ce a crescut semnificativ numărul de clase detectabile. Combinarea casetelor ancoră și antrenamentul la scară multiplă a contribuit la îmbunătățirea performanței de detectare pentru obiectele mici.[4]

YOLO v3 aparută în 2018, a îmbunătățit în continuare performanța detecției obiectelor. Această versiune a introdus conceptul de rețele de piramidă de caracteristici, cu mai multe straturi de detectare care au permis modelului să detecteze obiecte la diferite scale și rezoluții. YOLO v3 a folosit o arhitectură de rețea mai mare cu 53 de straturi de convoluție, numită Darknet-53, ceea ce a îmbunătățit capacitatea de reprezentare a modelului. YOLO v3 folosește trei scale diferite pentru detectare: grile de 13x13, 26x26 și 52x52. Fiecare scală prezice un număr diferit de casete delimitatoare per celulă a grilei. [4]

Deoarece YOLO și YOLO V2 nu sunt eficiente în detectarea țintelor mici, detectia multi-scală a fost adăugată în YOLO V3. YOLO V3 este considerat un maestru bine primit al generațiilor anterioare. YOLO V4 a examinat și a încercat toate optimizările posibile în întregul proces și a identificat cel mai bun efect în fiecare permutare și combinație. YOLOv4 rulează de două ori mai repede decât EfficientDet cu o performanță comparabilă. Îmbunătățește AP și FPS-ul YOLOv3 cu 10%, respectiv 12%. YOLO V5 poate controla flexibil modele de la 10+M la 200+M, iar modelul său mic este foarte impresionant. Schemele de rețea globale ale YOLO V3 până la YOLO V5 sunt similare, dar se concentrează și pe detectarea obiectelor de diferite dimensiuni din trei scale diferite.[6]

YOLO v5 a fost introdus în 2020 de aceeași echipă care a dezvoltat algoritmul YOLO original, ca proiect open-source și este întreținut de Ultralytics. YOLO v5 se bazează pe succesorul versiunilor anterioare și adaugă mai multe caracteristici noi și îmbunătățiri. Spre deosebire de YOLO, YOLO v5 folosește o arhitectură mai complexă numită EfficientDet, bazată pe arhitectura rețelei EfficientNet. Folosirea unei arhitecturi mai complexe în YOLO v5 îi permite să atingă o precizie mai mare și o generalizare mai bună către o gamă mai largă de categorii de obiecte.[yolo.]

O altă diferență între YOLO și YOLO v5 este setul de date de antrenament folosit pentru a învăța modelul de detectare a obiectelor. YOLO a fost antrenat pe setul de date PASCAL VOC, care constă în 20 de categorii de obiecte. YOLO v5, pe de altă parte, a fost antrenat pe un set de date mai mare și mai divers numit D5, care include un total de 600 de categorii de obiecte. YOLO v5 folosește o nouă metodă pentru generarea casetelor ancoră, numită "casete ancoră dinamice". Aceasta implică utilizarea unui algoritm de clusterizare pentru a grupa casetele delimitatoare ale adevărului teren în clusteri și apoi utilizarea centroidelor clusterilor ca casete ancoră. Aceasta permite casetelor ancoră să fie mai bine aliniate cu dimensiunea și forma obiectelor detectate.[11]

YOLO v6 a fost propus în 2022 de Li și colab. ca o îmbunătățire față de versiunile anterioare. Una dintre diferențele principale între YOLO v5 și YOLO v6 este arhitectura CNN utilizată. YOLO v6 folosește o variantă a arhitecturii EfficientNet numită EfficientNet-L2. Este o arhitectură mai eficientă decât EfficientDet folosită în YOLO v5, cu mai puțini parametri și o eficiență computatională mai mare. Poate obține rezultate de vârf pe diverse benchmark-uri de detectare a obiectelor. De asemenea, YOLO v6 introduce o metodă nouă pentru generarea casetelor ancoră, numită "casete ancoră dense".[11]

YOLO v7, cea mai recentă versiune a YOLO, aduce mai multe îmbunătățiri față de versiunile anterioare. Una dintre îmbunătățirile principale constă în utilizarea casetelor ancoră, acestea fiind un set de casete predefinite cu diferite raporturi de aspect, folosite pentru a detecta obiecte de forme diferite. YOLO v7 utilizează nouă casete ancoră, ceea ce îi permite să detecteze o gamă mai largă de forme și dimensiuni de obiecte în comparație cu versiunile anterioare, contribuind astfel la reducerea numărului de rezultate false.[11]

O îmbunătățire-cheie în YOLO v7 constă în utilizarea unei noi funcții de pierdere numită "focal loss". Versiunile anterioare ale YOLO foloseau o funcție de pierdere standard de entropie încrucișată, care se știe că este mai puțin eficientă în detectarea obiectelor mici. Focal loss abordează această problemă prin reducerea ponderii pierderii pentru exemplele bine clasificate și concentrându-se pe exemplele dificile - obiectele greu de detectat.[11]

YOLO v7 are, de asemenea, o rezoluție mai mare decât versiunile anterioare. Procesează imagini la o rezoluție de 608 pe 608 pixeli, ceea ce este mai mare decât rezoluția de 416 pe 416 utilizată în YOLO v3, fapt care permite lui YOLO v7 să detecteze obiecte mai mici și să aibă o precizie mai mare în general.[11]

Una dintre principalele avantaje ale lui YOLO v7 este viteza sa. Poate procesa imagini la o rată de 155 de cadre pe secundă, mult mai rapid decât alte algoritme de detectare a obiectelor de vârf. Chiar și modelul YOLO inițial de bază era capabil să proceseze la o rată maximă de 45 de cadre pe secundă. Acest lucru îl face potrivit pentru aplicații sensibile în timp real, cum ar fi supravegherea și mașinile autonome, unde vitezele mai mari de procesare sunt cruciale.[11]

În ceea ce privește acuratețea, YOLO v7 se descurcă bine în comparație cu alte algoritme de detecție a obiectelor. Obține o precizie medie de 37.2% la o valoare de suprapunere (IoU - Intersection over Union) de 0.5 pe setul de date COCO, ceea ce este comparabil cu alte algoritme de detecție a obiectelor de ultimă generație. Comparația cantitativă a performanței este prezentată în tabelul de mai jos.[11]

Tabelul 3.1: Compararea performanței modelelor YOLO și YOLOR [9]

Model	#Param.	FLOPs	Size	APval	APval50	APval75	APvalS	APvalM	APvalL
YOLOv4	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1)	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	51.2%	69.7%	55.5%	35.2%	56.0%	66.7%
YOLOR-CSP-X	96.9M	226.8G	640	52.7%	71.3%	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	52.9%	71.1%	57.5%	36.9%	57.7%	68.6%
YOLOv4-tiny	6.1M	6.9G	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2M	5.8G	416	35.2%	52.8%	37.3%	15.7%	38.0%	53.4%
YOLOv4-tiny-3l	8.7M	5.2G	320	30.8%	47.3%	32.2%	10.9%	31.9%	51.5%
YOLOR-E6	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	60.4%	69.2%
YOLOv7-E6	97.2M	515.2G	1280	55.9%	73.5%	61.1%	40.6%	60.3%	70.0%
YOLOR-D6	151.7M	935.6G	1280	56.1%	73.9%	61.2%	42.4%	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	56.8%	74.4%	62.1%	40.8%	62.1%	70.6%

La o rezoluție de 416 x 416, YOLO v1 prezice  $7 \times 7 = 49$  de casete. YOLO v2 prezice  $13 \times 13 \times 5 = 845$  de casete. Pentru YOLO v2, la fiecare celulă a grilei, sunt detectate 5 casete folosind 5 ancoraje. Pe de altă parte, YOLO v3 prezice casete la 3 scale diferite. Pentru aceeași imagine de 416 x 416, numărul de casete prezise este  $13 \times 13 \times 3 + 26 \times 26 \times 3 + 52 \times 52 \times 3 = 10,647$ , rezultatele îmbunătățindu-se pe măsură ce versiunile de YOLO au avansat, astfel pe baza calculelor anterioare se poate observa evoluția de care a avut parte algoritmul YOLO.[4]

Tabelul de mai jos ne oferă numărul de articole academice de cercetare pentru fiecare versiune. Detalierea arată că numărul de articole de cercetare a crescut semnificativ în anii

2019 și 2020. În plus, versiunile YOLO V3 și V2 au atras cea mai mare atenție din partea cercetătorilor, deși factorul timp poate fi și el un element. Numărul de articole pentru versiunile V4 și V5 este mai mic pentru că sunt foarte noi.[6]

Tabelul 3.2: Numărul de articole academice pentru fiecare versiune YOLO în diferiți ani [6]

An	YOLO V2	YOLO V3	YOLO V4	YOLO V5	Total
2016	0	0	0	0	0
2017	5	0	0	0	5
2018	47	19	0	0	66
2019	48	210	0	0	258
2020	36	496	81	13	626
<b>Total</b>	<b>136</b>	<b>725</b>	<b>81</b>	<b>13</b>	<b>955</b>

### 3.3 DETR (DEtection TRansformers)

Detectia obiectelor cu transformatori (DETR) este un cadru recent care tratează detectia obiectelor ca o problemă de predicție directă printr-un encoder-decoder cu transformatori [39]. Fără selecție manuală a eșantioanelor și fără suprimare a nonmaximelor, DETR atinge o performanță competitivă cu Faster R-CNN. Cu toate acestea, DETR se confruntă cu provocări de antrenament și optimizare, care necesită date de antrenament la scară mare și un program de antrenament extrem de lung. [3]

Lucrările curente par să sugereze că transformatorii de detectie sunt superioari detectorilor de obiecte bazate pe CNN, precum Faster R-CNN, atât în simplitate, cât și în performanță modelului. Cu toate acestea, constatăm că transformatorii de detectie arată o performanță superioară doar pe seturi de date cu date de antrenament bogate, cum ar fi COCO 2017 (118K imagini de antrenament), în timp ce performanța majorității transformatorilor de detectie scade semnificativ atunci când cantitatea de date de antrenament este mică. De exemplu, pe setul de date de conducere autonomă Cityscapes (3K imagini de antrenament), preciziile medii (AP) ale majorității transformatorilor de detectie sunt mai mici de jumătate din performanța AP a Faster R-CNN. Mai mult, deși diferențele de performanță între diferiți transformatori de detectie pe setul de date COCO sunt mai mici de 3 AP, există o diferență semnificativă de peste 15 AP pe setul de date Cityscapes de dimensiuni mici.[10]

Arhitectura globală DETR este surprinzătoare de simplă, ea conține trei componente principale: un suport CNN pentru a extrage o reprezentare compactă a caracteristicilor, un transformator encoder-decoder și o rețea de feed-forward (FFN) simplă care face predicția finală de detectie. Spre deosebire de multe detectoare moderne, DETR poate fi implementat în orice cadru de învățare profundă care furnizează un suport CNN comun și o implementare a arhitecturii transformatorului cu doar câteva sute de linii de cod.[2]

În ceea ce privește rețelele de feed-forward pentru predicție (FFN), predicția finală este calculată printr-un perceptron cu 3 straturi cu funcție de activare ReLU și dimensiune ascunsă d, și un strat de proiecție liniar. FFN-ul prezice coordonatele centrale normalizate, înălțimea și lățimea cutiei în raport cu imaginea de intrare, iar stratul liniar prezice eticheta clasei folosind o funcție softmax. Deoarece prezicem un set de N cutii de delimitare de dimensiune fixă, unde N este în mod obișnuit mult mai mare decât numărul real de obiecte de interes într-o imagine, o etichetă specială de clasă este folosită pentru a reprezenta faptul că nu este detectat niciun obiect într-un slot. Această clasă joacă un rol similar cu clasa "fundal" în abordările standard.[2]

Tabelul de mai jos prezintă o comparație între diferite modele de detectare a obiectelor, evidențiind performanța acestora în termeni de acuratețe și eficiență computatională. Fiecare

model este caracterizat prin arhitectura sa, numărul de parametri, performanța în detecția obiectelor exprimată prin diverse metrii precum acuratețea medie a punctelor (AP), precizia la diferite niveluri de suprapunere între detectate și ground truth (AP50, AP75), precum și scalarea performanței în funcție de dimensiunea obiectelor (APS, APM, APL). De asemenea, sunt furnizate informații despre puterea de calcul a fiecărui model, exprimată în giga-operații pe secundă (GFLOPS) și cadre pe secundă (FPS). Aceste date oferă o imagine cuprinzătoare asupra performanțelor și trade-off-urilor între precizie și eficiență computatională a diferitelor modele de detectare a obiectelor.

Tabelul 3.3: Compararea performanței modelelor de detectare a obiectelor [2]

Model	GFLOPS/FPS	#Params	AP	AP50	AP75	APS	APM	APL
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	47.8	27.2	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	44.9	64.7	47.7	23.7	49.5	62.3

Comparând performanța modelului DETR cu valorile din tabel, observăm că DETR obține scoruri competitive în ceea ce privește acuratețea medie a punctelor (AP) și alte metrii de evaluare a performanței, în special în comparație cu modelele Faster R-CNN. Spre exemplu, DETR-DC5-R101 are un AP de 44.9, care este comparabil cu cele mai bune performanțe ale modelelor Faster R-CNN-R101-FPN+ și DETR-R101, care au AP-uri de 44.0 și, respectiv, 43.5. Acest lucru indică faptul că DETR se descurcă bine în detecția obiectelor într-o varietate de scenarii, obținând rezultate solide chiar și în comparație cu modelele tradiționale care sunt bine stabilite în domeniul detectiei de obiecte.

# 4 Antrenarea rețelelor și rezultate experimentale

## 4.1 Antrenarea rețelelor

Pentru antrenarea celor două rețele am folosit un calculator echipat cu un procesor Intel i5-13600KF, 32BG RAM și o placă video AMD ATI Radeon RX 6700 XT. Pentru a folosi placă grafică pentru antrenare am folosit driverul ROCm 6.0<sup>1</sup> de la AMD pe un sistem Debian 12 cu biblioteca PyTorch<sup>2</sup>.

### 4.1.1 YOLOv8

Pentru antrenarea rețelei YOLOv8 am folosit modelul pre-antrenat yolov8n, pe 10 epoci. Am folosit API-ul de antrenare de la Ultralytics<sup>3</sup> în Python cu biblioteca Pytorch.

```
1 model = YOLO("yolov8n.pt")
2
3 results = model.train(
4     data="tobacco.v9i.yolov8/data.yaml",
5     epochs=10,
6     batch=8,
7     imgsz=640,
8     name="tobacco_v1"
9 )
```

Antrenarea a durat 25 de minute pe 10 epoci folosind setul de date "tobacco Dataset" (de pe Roboflow), iar apoi am testat modelul obținut folosind setul de date "TobSet".

### 4.1.2 DETR

Pentru antrenarea rețelei DETR am folosit un notebook<sup>4</sup> bazat pe cel al lui Niels Rogge<sup>5</sup>, în care se antrenează (fine tune) DETR pe un set de date specific. Am folosit același set de date folosit anterior la YOLOv8, pe o singura epoca. Antrenarea a durat 17 minute. Pentru prima antrenare am încercat cu 10 epoci, însă la cea de a doua am rămas fară suficient VRAM și am decis să folosim o singură epocă pentru antrenare.

<sup>1</sup><https://rocm.docs.amd.com/en/latest/>

<sup>2</sup><https://pytorch.org/>

<sup>3</sup><https://github.com/ultralytics/ultralytics>

<sup>4</sup>Notebook - How to Train DETR with Transformers on a Custom Dataset

<sup>5</sup><https://github.com/NielsRogge>

## 4.2 Rezultate experimentale

### 4.2.1 YOLOv8

Metrici de performanță pentru modelul YOLOv8:

- **epoch**: Aceasta este numărul epocii de antrenament. Fiecare epocă reprezintă o trecere completă prin întregul set de date de antrenament.
- **train/box\_loss, train/cls\_loss, train/dfl\_loss**: Acestea sunt pierderile pentru "bounding box", clasă și lungimea caracteristicii de direcție (DFL - Direction Feature Length), calculate în timpul antrenamentului. Valori mai mici indică o performanță mai bună.
- **metrics/precision(B), metrics/recall(B), metrics/mAP50(B), metrics/mAP50-95(B)**: Acestea sunt metricile de evaluare calculate pe setul de date de validare. Precizia este raportul dintre observațiile pozitive corect prezise la totalul pozitivelor prezise. Recall (Sensibilitatea) este raportul dintre observațiile pozitive corect prezise la toate observațiile din clasa actuală. mAP50 este media Preciziei Medii la IoU (Intersecția peste Uniune) de 0.50. mAP50-95 este media Preciziei Medii la IoU de la 0.50 la 0.95.
- **val/box\_loss, val/cls\_loss, val/dfl\_loss**: Acestea sunt pierderile pentru "bounding box", clasă și lungimea caracteristicii de direcție (DFL), calculate pe setul de date de validare. Valori mai mici indică o performanță mai bună.
- **lr/pg0, lr/pg1, lr/pg2**: Acestea sunt ratele de învățare pentru diferite grupuri de parametri în optimizator.

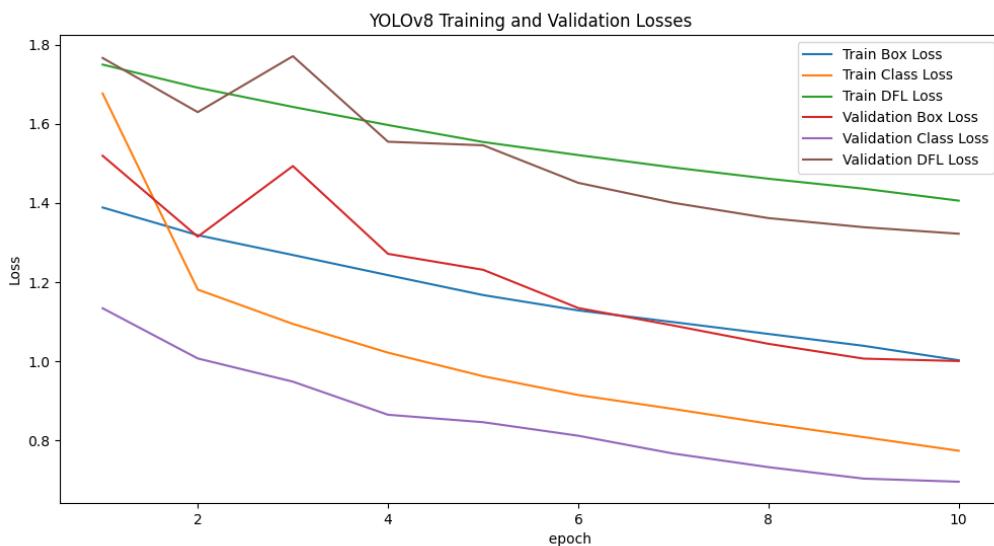


Figura 4.1: Pierderile în timpul antrenamentului și validării pentru YOLOv8

În figura 4.1, observăm evoluția pierderilor în timpul antrenamentului și validării pentru modelul YOLOv8 pe parcursul a 10 epoci. Toate tipurile de pierderi (*train/box\_loss*, *train/cls\_loss*, *train/dfl\_loss*, *val/box\_loss*, *val/cls\_loss* și *val/dfl\_loss*) încep de la valori inițiale ridicate și scad pe măsură ce epocile avansează.

## Observații generale

- **Trend descendente**: Toate pierderile prezintă un trend descendente pe parcursul celor 10 epoci. Acest lucru indică faptul că modelul învăță și se îmbunătățește, ajustându-și parametrii pentru a minimiza pierderile.
- **Reducerea pierderilor de antrenament și validare**: Atât pierderile de antrenament (train loss) cât și cele de validare (validation loss) scad, ceea ce sugerează că modelul se adaptează bine atât pe datele de antrenament cât și pe cele de validare.

## Interpretare specifică

### Pierderile de antrenament (Train Loss):

- *Train Box Loss*: Scăderea acestui parametru indică faptul că modelul devine din ce în ce mai precis în predicția limitelor obiectelor.
- *Train Class Loss*: Scăderea acestui parametru sugerează că modelul îmbunătățește acuratețea în clasificarea obiectelor.
- *Train DFL Loss*: Scăderea acestui parametru arată o îmbunătățire în predicția lungimii caracteristicii de direcție, ceea ce poate contribui la o localizare mai precisă a obiectelor.

### Pierderile de validare (Validation Loss):

- *Validation Box Loss*: Scăderea indică o îmbunătățire a performanței modelului pe setul de date de validare în ceea ce privește localizarea obiectelor.
- *Validation Class Loss*: O scădere în acest parametru indică faptul că modelul generalizează bine în clasificarea obiectelor și pe datele de validare.
- *Validation DFL Loss*: Reducerea acestui parametru sugerează că modelul reușește să păstreze precizia în predicția lungimii caracteristicii de direcție și pe setul de validare.

Scăderea continuă a pierderilor de antrenament și validare de-a lungul epocilor indică faptul că modelul YOLOv8 se antrenează eficient și se îmbunătățește în predicția limitelor, clasificarea obiectelor și predicția lungimii caracteristicii de direcție. Acest lucru sugerează o performanță în creștere și o bună capacitate de generalizare a modelului.

Este important de menționat că, deși o scădere a pierderilor este un semn pozitiv, este necesară monitorizarea constantă pentru a evita probleme precum supraînvățarea (*overfitting*), unde performanța pe setul de antrenament continuă să se îmbunătățească, dar cea pe setul de validare începe să se degradeze.

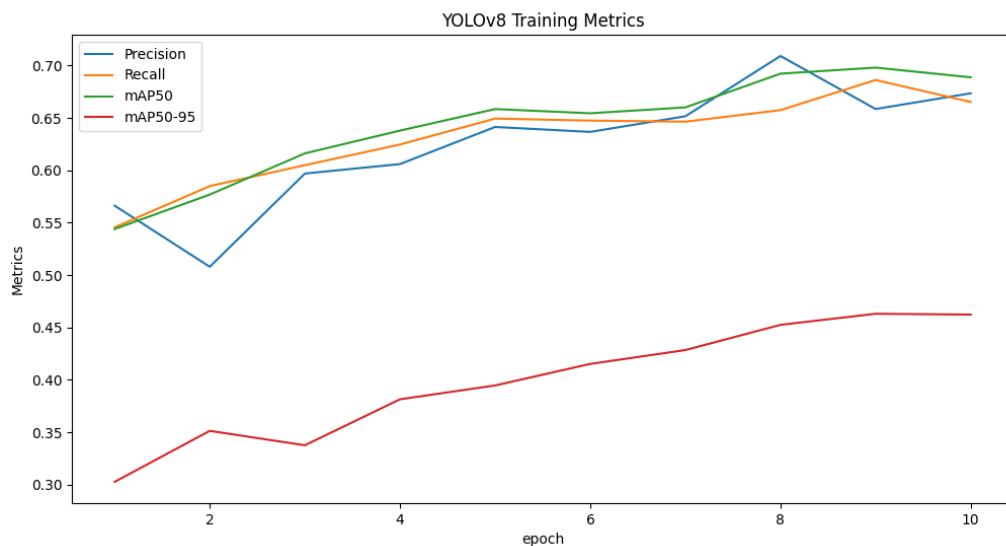


Figura 4.2: Metricile de antrenament pentru YOLOv8

În figura 4.2, sunt prezentate evoluțiile metricilor de antrenament pentru YOLOv8 pe parcursul a 10 epoci. Metricile afișate includ *precision*, *recall*, *mAP50* și *mAP50-95*. Toate aceste metriki cresc pe măsură ce epocile avansează.

### Observații

- **Creșterea preciziei:** Valoarea *precision* crește pe parcursul celor 10 epoci. Aceasta indică o îmbunătățire a modelului în identificarea corectă a obiectelor prezise, reducând numărul de elemente fals pozitive.
- **Creșterea sensibilității (recall):** Valoarea *recall* crește de asemenea, sugerând că modelul reușește să identifice un număr tot mai mare de obiecte corecte pe măsură ce epocile avansează, reducând numărul de elemente fals negative.
- **Îmbunătățirea mAP50:** Creșterea valorii *mAP50* indică faptul că modelul obține o medie a preciziei mai bună la un IoU (Intersecția peste Uniune) de 0.50, ceea ce semnalează o precizie crescută în predicțiile obiectelor.
- **Îmbunătățirea mAP50-95:** Creșterea valorii *mAP50-95* reflectă o îmbunătățire generală a modelului pe o gamă de valori IoU de la 0.50 la 0.95, indicând o performanță robustă și consistentă a modelului în diferite scenarii de suprapunere a obiectelor.

Cel mai bun model YOLO este cel din epoca 9. Având cea mai mare valoare mAP50 și mAP50-95.

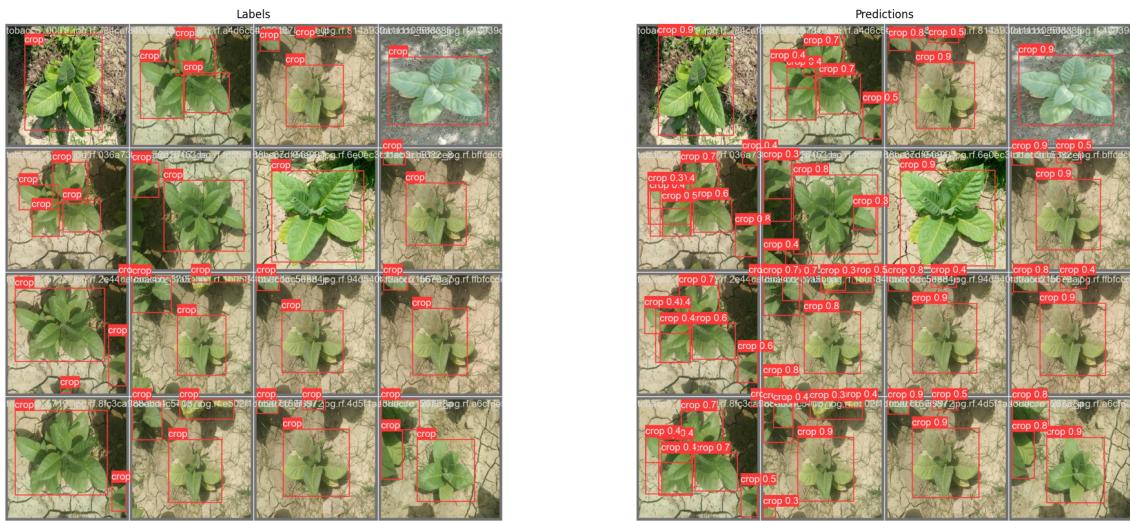


Figura 4.3: Rezultate YOLOv8 Batch 0

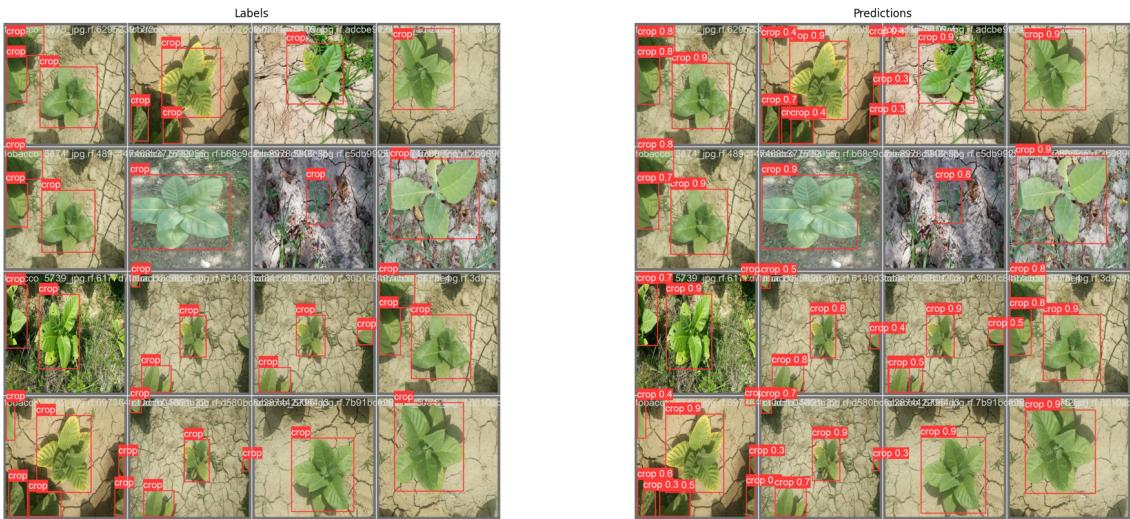


Figura 4.4: Rezultate YOLOv8 Batch 1



Figura 4.5: Rezultate YOLOv8 Batch 2

- **Batch 0:** În figura 4.3, vedem primele rezultate ale modelului YOLOv8 pe batch-ul 0. Modelul începe să învețe structura și caracteristicile obiectelor, iar predicțiile sale sunt în general bune. Deși pot exista unele erori inițiale, acestea sunt normale în primele etape de antrenament.
- **Batch 1:** Figura 4.4 arată rezultatele pe batch-ul 1. Se observă o îmbunătățire în precizia și acuratețea predicțiilor comparativ cu batch-ul 0. Modelul continuă să învețe și să se ajusteze pe baza datelor de antrenament.
- **Batch 2:** În figura 4.5, rezultatele pe batch-ul 2 arată o performanță și mai bună, cu predicții mai precise și mai puține erori. Modelul YOLOv8 demonstrează o capacitate crescută de a identifica și localiza obiectele corect.

Rezultatele prezentate în figurile 4.3, 4.4 și 4.5 demonstrează că modelul YOLOv8 învață eficient din datele de antrenament. Pe măsură ce avansează prin batch-uri, modelul îmbunătățește constant precizia și acuratețea predicțiilor sale. Performanța generală este bună, iar modelul arată o capacitate promițătoare de a generaliza pe seturi de date noi.

#### 4.2.2 DETR

Rezultatele obținute reprezintă evaluarea performanței modelului DEtection TRansformer (DETR) utilizând metricile IoU (Intersection over Union) pentru bounding box-uri. Aceste metriki sunt folosite pentru a măsura cât de bine modelul detectează și localizează obiectele în imagini.

##### Average Precision (AP):

- $\text{AP}@\text{[IoU=0.50:0.95 | area=all | maxDets=100]} = 0.331$ : Aceasta este media preciziei pentru valori IoU între 0.50 și 0.95 pentru toate dimensiunile obiectelor și un număr maxim de 100 de detectii per imagine. Aceasta este considerată AP medie și este o măsură generală a performanței modelului.
- $\text{AP}@\text{[IoU=0.50 | area=all | maxDets=100]} = 0.488$ : Aceasta este precizia medie la un IoU de 0.50 pentru toate dimensiunile obiectelor și un număr maxim de 100 de detectii per imagine. IoU de 0.50 este un criteriu mai permisiv și este mai ușor de atins.
- $\text{AP}@\text{[IoU=0.75 | area=all | maxDets=100]} = 0.377$ : Aceasta este precizia medie la un IoU de 0.75 pentru toate dimensiunile obiectelor și un număr maxim de 100 de detectii per imagine. IoU de 0.75 este un criteriu mai strict.
- $\text{AP}@\text{[IoU=0.50:0.95 | area=small | maxDets=100]} = 0.000$ : Precizia medie pentru obiectele mici, cu valori IoU între 0.50 și 0.95. În acest caz, modelul nu reușește să detecteze obiectele mici.
- $\text{AP}@\text{[IoU=0.50:0.95 | area=medium | maxDets=100]} = 0.100$ : Precizia medie pentru obiectele de dimensiune medie, cu valori IoU între 0.50 și 0.95. Performanța este relativ scăzută pentru obiectele medii.
- $\text{AP}@\text{[IoU=0.50:0.95 | area=large | maxDets=100]} = 0.351$ : Precizia medie pentru obiectele mari, cu valori IoU între 0.50 și 0.95. Modelul are performanțe mai bune pentru obiectele mari.

### Average Recall (AR):

- AR@[IoU=0.50:0.95 | area=all | maxDets=1] = 0.299: Aceasta este recall-ul mediu pentru un număr maxim de 1 detectie per imagine, pentru toate dimensiunile obiectelor și valori IoU între 0.50 și 0.95.
- AR@[IoU=0.50:0.95 | area=all | maxDets=10] = 0.428: Recall-ul mediu pentru un număr maxim de 10 detectii per imagine, pentru toate dimensiunile obiectelor și valori IoU între 0.50 și 0.95.
- AR@[IoU=0.50:0.95 | area=all | maxDets=100] = 0.453: Recall-ul mediu pentru un număr maxim de 100 detectii per imagine, pentru toate dimensiunile obiectelor și valori IoU între 0.50 și 0.95. Aceasta este recall-ul mediu general.
- AR@[IoU=0.50:0.95 | area=small | maxDets=100] = 0.000: Recall-ul mediu pentru obiectele mici, cu valori IoU între 0.50 și 0.95. Modelul nu detectează obiectele mici.
- AR@[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.269: Recall-ul mediu pentru obiectele de dimensiune medie, cu valori IoU între 0.50 și 0.95. Performanța este mai scăzută pentru obiectele medii.
- AR@[IoU=0.50:0.95 | area=large | maxDets=100] = 0.475: Recall-ul mediu pentru obiectele mari, cu valori IoU între 0.50 și 0.95. Modelul detectează mai bine obiectele mari.

### Interpretarea Rezultatelor

Modelul DETR are o performanță moderată, cu un AP mediu de 0.331 pentru toate dimensiunile obiectelor și o gamă de IoU între 0.50 și 0.95. Modelul performează mai bine pentru obiectele mari (AP = 0.351, AR = 0.475) și mai slab pentru obiectele de dimensiune medie (AP = 0.100, AR = 0.269) și foarte slab pentru obiectele mici (AP și AR ambele 0.000). Performanța scade pe măsură ce crește pragul IoU (0.488 la IoU = 0.50, scăzând la 0.377 la IoU = 0.75), ceea ce indică faptul că modelul are dificultăți în a localiza obiectele cu precizie foarte mare. Recall-ul crește cu numărul maxim de detectii permise (0.299 pentru maxDets=1, crescând la 0.453 pentru maxDets=100), sugerând că modelul poate detecta mai multe obiecte corect atunci când are permisiunea de a face mai multe predicții.

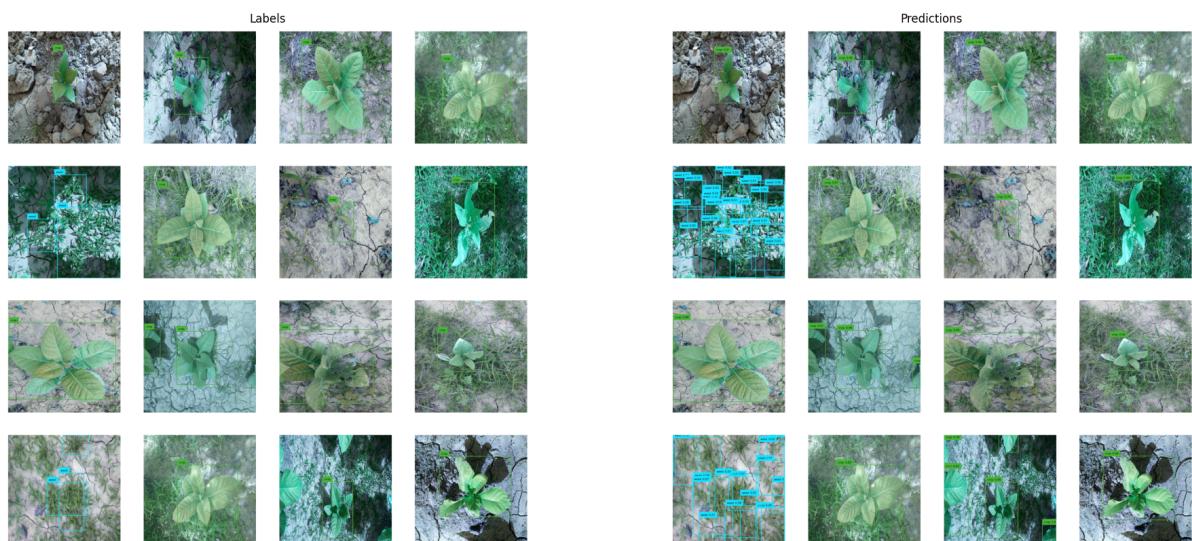


Figura 4.6: Rezultate DETR Batch 0



Figura 4.7: Rezultate DETR Batch 1



Figura 4.8: Rezultate DETR Batch 2

### YOLOv8 vs DETR

Rezultatele obținute pentru modelele YOLOv8 și DEtection TRansformer (DETR) arată o comparație interesantă a performanțelor acestora. Pentru modelul YOLOv8, pierderile în timpul validării au scăzut constant de la epoca 1 la epoca 10: val/cls\_loss a scăzut de la 1.3888 la 0.77408, iar val/dfl\_loss a scăzut de la 1.6769 la 1.4061. Ratele de învățare ( $lr/pg0$ ,  $lr/pg1$ ,  $lr/pg2$ ) au scăzut și ele de-a lungul epocilor, de la 0.00055536 la 0.0001817. Mai mult se observă că precizia, reprezentată de metrica metrics/precision(B), a variat între 0.56631 și 0.70917, manifestând o tendință de creștere inițială, urmată de o stabilizare ulterioară. De asemenea, recall-ul, exprimat prin metrica metrics/recall(B), a variat între 0.54561 și 0.68628, evidențiind o creștere inițială, după care s-a menținut constant. În ceea ce privește mAP50 și mAP50-95, reprezentând media preciziei medii la diferite valori de IoU, ambele metriki au înregistrat o creștere constantă de la 0.544 și 0.30269 la 0.68882 și 0.46233, respectiv. Aceste observații indică o îmbunătățire a performanței modelului YOLOv8 pe parcursul antrenamentului, cu o tendință generală de stabilizare a metricilor de evaluare la epoci ulterioare.

Pe de altă parte, rezultatele pentru DETR arată că modelul a obținut un AP de 0.331

pentru IoU între 0.50 și 0.95 și un AP de 0.488 pentru IoU=0.50. La IoU=0.75, DETR a obținut un AP de 0.377, indicând o performanță rezonabilă la un prag mai strict. Totuși, DETR prezintă dificultăți majore în detectarea obiectelor mici, cu un AP și AR de 0.000, și performanțe scăzute pentru obiectele medii (AP=0.100, AR=0.269), dar mai bune pentru obiectele mari (AP=0.351, AR=0.475). Recall-ul DETR variază între 0.299 pentru un număr maxim de 1 detectie și 0.453 pentru un număr maxim de 100 detectii, indicând o capacitate crescută de a detecta mai multe obiecte când sunt permise mai multe detectii.

În concluzie, YOLOv8 și DETR au performanțe generale comparabile în ceea ce privește precizia medie la diferite valori IoU, dar YOLOv8 pare să fie mai robust în detectarea obiectelor de dimensiuni variante. DETR necesită îmbunătățiri pentru detectarea obiectelor mici și medii, în timp ce YOLOv8 are o performanță mai consistentă pe toate dimensiunile obiectelor.

## 5 Concluzie

Implementarea și testarea algoritmilor DETR și YOLO au demonstrat clar eficiența acestora în detectarea obiectelor în imagini și videoclipuri. Rezultatele obținute au evidențiat o îmbunătățire semnificativă a preciziei și vitezei de procesare în comparație cu metodele tradiționale de detectie a obiectelor. Algoritmul DETR, rețele neurale transformers, a evidențiat capacitatea sa de a detecta și localiza obiecte într-un mod precis și robust, fără a fi nevoie de metode suplimentare de post-procesare. Pe de altă parte, YOLO a excelat în viteza de procesare, asigurând detectarea rapidă a obiectelor, cu o acuratețe bună.

Următoarea etapă în evoluția proiectului constă în integrarea celor două modele pentru a forma un sistem combinat de rețele neuronale. Această abordare combinată ne va permite să valorificăm beneficiile fiecărui algoritm, cum ar fi precizia și robustețea lui DETR, împreună cu viteza și performanța în timp real oferite de YOLO. Prin această integrare, ne așteptăm să obținem o creștere semnificativă a performanței generale a detecției de obiecte, consolidând astfel poziția platformei noastre ca un instrument de încredere pentru sarcinile de vizualizare și identificare a obiectelor în diverse scenarii.

# Bibliografie

- [1] Muhammad Shahab Alam și alții. „TobSet: A New Tobacco Crop and Weeds Image Dataset and Its Utilization for Vision-Based Spraying by Agricultural Robots“. În: *Applied Sciences* 12.3 (2022). ISSN: 2076-3417. URL: <https://www.mdpi.com/2076-3417/12/3/1308>.
- [2] Nicolas Carion și alții. „End-to-End Object Detection with Transformers“. În: nov. 2020, pag. 213–229. ISBN: 978-3-030-58451-1. DOI: [10.1007/978-3-030-58452-8\\_13](https://doi.org/10.1007/978-3-030-58452-8_13).
- [3] Zhigang Dai și alții. „Up-detr: Unsupervised pre-training for object detection with transformers“. În: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pag. 1601–1610.
- [4] DETection TRansformer (DETR) vs. YOLO for object detection. <https://medium.com/@faheemrustamy/detection-transformer-detr-vs-yolo-for-object-detection-baeb3c50bc3>. Accessed: 2024-05-19.
- [5] hellooooo. tobacco Dataset. <https://universe.roboflow.com/hellooooo/tobacco-js5vk>. Open Source Dataset. visited on 2024-05-19. Nov. 2022. URL: <https://universe.roboflow.com/hellooooo/tobacco-js5vk>.
- [6] Peiyuan Jiang și alții. „A Review of Yolo algorithm developments“. În: *Procedia computer science* 199 (2022), pag. 1066–1073.
- [7] Xingping Sun și alții. „Tobacco plant detection in RGB aerial images“. În: *Agriculture* 10.3 (2020), pag. 57.
- [8] Muhammad Tufail și alții. „Identification of Tobacco Crop Based on Machine Learning for a Precision Agricultural Sprayer“. În: *IEEE Access* 9 (2021), pag. 23814–23825. DOI: [10.1109/ACCESS.2021.3056577](https://doi.org/10.1109/ACCESS.2021.3056577).
- [9] Chien-Yao Wang, Alexey Bochkovskiy și Hong-Yuan Mark Liao. „YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors“. În: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pag. 7464–7475. DOI: [10.1109/CVPR52729.2023.00721](https://doi.org/10.1109/CVPR52729.2023.00721).
- [10] Wen Wang și alții. „Towards data-efficient detection transformers“. În: *European conference on computer vision*. Springer. 2022, pag. 88–105.
- [11] YOLO: Algorithm for Object Detection Explained [+Examples]. <https://www.v7labs.com/blog/yolo-object-detection>. Accessed: 2024-05-19.