

Sortiranje pistaća na otvorene i zatvorene korišćenjem metoda dubokog učenja

Matija Šuković, 1/22, D MAS



Prirodno-matematički fakultet
Univerzitet Crne Gore
13.06.2022.

Sadržaj

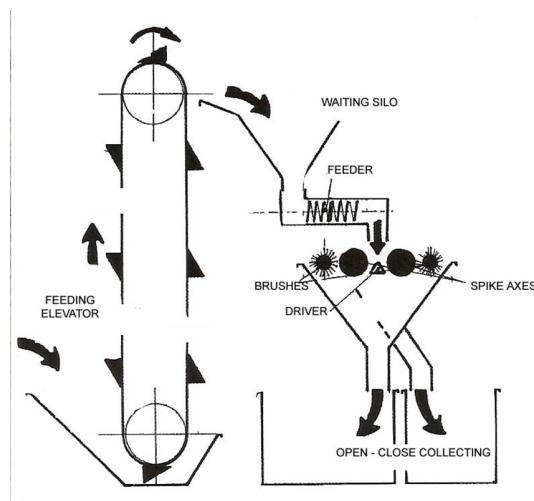
1	Uvod u temu	1
2	Pregled podataka	2
3	Tehnike	2
3.1	Model RetinaNet	3
3.2	Model YOLOv5	4
4	Metrike	5
5	Trenutno stanje projekta	6
5.1	Verzije dataset-a i rezultati	6
5.2	Vizualizacija trenutnih rezultata	6

1 Uvod u temu

Pistaći su koštunjavo voće poznato i voljeno zbog njihovog specifičnog ukusa. Pistaći su, takođe, čuveni po tome da su drastično skuplji od ostalog koštunjavog voća. Njihova visoka vrijednost je uzrokovana brojnim faktorima. Jedan dio tih faktora dolazi iz činjenice da je farma pistaća izuzetno teška za formiranje i održavanje. Sa druge strane, pistaći zahtijevaju dalju obradu nakon berbe. Najskuplji dio ove faze je sortiranje.

Pistaći se sortiraju po veličini, kvalitetu, izgledu i po tome da li je ljska otvorena ili ne. Postoje sistemi koji koriste laserske i slične tehnologije da automatski sortiraju plodove. Ovi sistemi se fokusiraju na otklanjanje otpada i sortiranje plodova na one sa i bez ljske.

Sortiranje na otvorene i zatvorene pistaće se automatizuje mašinama koje fizički detektuju otvorene plodove (slika 1). U suštini, one kukicama 'hvataju' otvorene pistaće za njihovu ljsku. Mana ovakvih mašina je da kukice nerijetko oštete plod, kao i da mogu da skinu ljsku sa ploda. Ovo predstavlja problem jer se uklanjanje otpada i sortiranje pistaća bez ljske radi prije sortiranja na otvorene i zatvorene plodove.



Slika 1: Skica mašine za mehaničko sortiranje pistaća na otvorene i zatvorene

Navedene mašine rade sa procentom uspješnosti do 98%, ali su veoma skupe. Mnogi farmeri locirani u siromašnijim zemljama ne mogu da ih priušte, pa se i danas sortiranje većinom radi ručno, što zahtijeva mnogo vremena i doprinosi visokoj cijeni proizvoda. Napredovanjem

tehnika mašinskog učenja i kompjuterske vizije nastaju novi prijedlozi za rješenje problema sortiranja pistaća.

Tema kojom sam odlučio da se bavim jeste sortiranje pistaća na otvorene i zatvorene korišćenjem metoda dubokog učenja. Cilj je kreiranje modela koji će moći da detektuje pistaće sa video snimka i da ih klasificuje na otvorene i zatvorene, u realnom vremenu, sa uspješnoću sličnom današnjim standardima.

2 Pregled podataka

U potrazi za sličnim projektima, naišao sam na rad [1], koji se bavi rješavanjem problema sortiranja pistaća na otvorene i zatvorene. Uz rad objavljen je i set od 432 slike i 6 videa. Slike su rezolucije 1070×600 piksela i u pitanju su slike maštine koja sprovodi plodove sa jedne strane na drugu, istovremeno ih rotirajući velikom brzinom pomoću valjaka (slika 2). Svaka slika sadrži 1 - 27 plodova, ukupno 3927. Video klipovi sadrže 561 plod i ukupno su dužine 167 sekundi.

Slike su anotirane u csv fajlu čija je struktura prikazana u tabeli 1. Video klipovi nisu anotirani, autori podstiču čitaoce da ih iskoriste za širenje dataset-a. Na GitHub repozitorijumu projekta može se naći kôd za izvlačenje frejmova iz videa, kao i za program za anotiranje.

Prva kolona tabele anotacija je ime fajla, tj. slike. Zatim slijede X i Y koordinate centra, visina i širina pravougaonika koji predstavlja 'bounding box' jednog od plodova na slici. Poslednja kolona sadrži podatak o klasi selekcije, nosi vrijednost 0 ako je plod zatvorene ljske, odnosno 1 ako je ljska otvorena.



Slika 2: Primjer slike iz dataset-a

filename	X	Y	width	height	class
50-20_199.jpg	740	109	825	178	0
50-20_199.jpg	485	136	545	185	1
...

Tabela 1: Format podataka u priloženom csv fajlu

3 Tehnike

Problem sortiranja pistaća na otvorene i zatvorene je tipičan problem detekcije objekata. Zadatak kod ovih problema je nalaženje svih objekata od interesa na datoj slici i dodjeljivanje

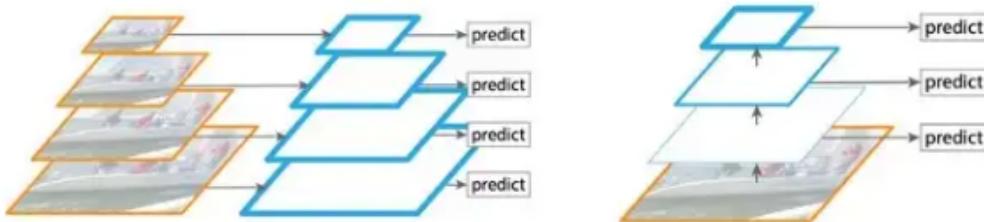
klase svakom od nađenih objekata. U suštini to su dva problema u jednom, i u začetku mašinskog učenja su se tretirali odvojeno. Danas imamo nekoliko vrsta modela za detekciju objekata koji vrlo dobro rješavaju probleme ovog tipa.

U radu [1] korišćen je model RetinaNet, treniran na priloženim slikama i testiran na 6 video klipova dao je uspješnost 94.75%. Ja sam izabrao model YOLO (verzija 5) koji je danas veoma popularan na polju detekcije objekata. Na kraju projekta uporediće uspješnost YOLO modela sa uspješnoću modela RetinaNet. O metrikama govorim u sekiciji 4.

3.1 Model RetinaNet

RetinaNet je jedan od najpopularnijih modela za detekciju objekata koji daje izuzetno dobre rezultate na gustim i sitnim objektima.

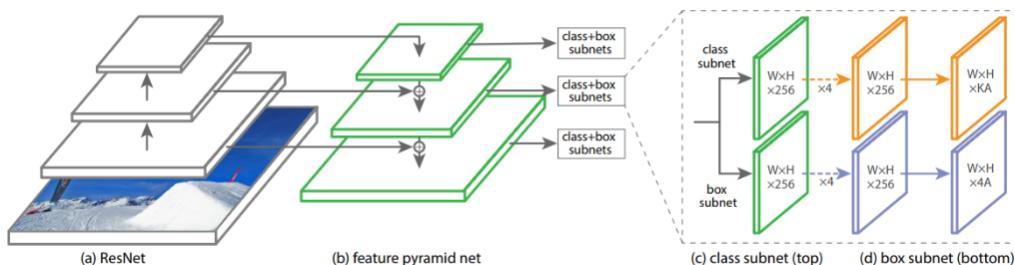
Glavno poboljšanje arhitekture RetinaNet-a u odnosu na ostale modele jeste 'Feature pyramid network' (FPN). Piramide odlike su tradicionalno korišćene za detekciju objekata koji variraju u veličini. U suštini, na ulaznoj slici radi se 'downsampling' i formira se piramida (slika 3). Ručno zadate odlike (features) se, zatim, izvlače iz svakog sloja da bi se detektovali objekti. Ovaj pristup omogućava laku detekciju objekta nezavisno od njegove veličine, ali je komputaciono zahtjevan. [2]



Slika 3: Piramida slika (lijevo) i piramida 'feature' mapa (desno)

Kako je duboko učenje napredovalo, tako su ručno zadate odlike počele da se zamjenjuju konvolucinim neuronskim mrežama, takva struktura se naziva FPN.

RetinaNet koristi 'backbone' FPN pod imenom ResNet za ekstrakciju odlike (slika 4). Zatim prolazi kroz piramidu odozgo nadolje, radeći 'upsampling' sa ciljem da se odlike sa viših slojeva izoštire. Detektovane odlike se onda provlače kroz podmrežu za klasifikaciju, gdje se svakoj odlici dodjeljuje vjerovatnoća da ona pripada nekoj od traženih klasa. Na kraju se prolazi kroz podmrežu za regresiju 'bounding box'-ova, da bi se doobile što tačnije granice oko detektovanih objekata.[3]

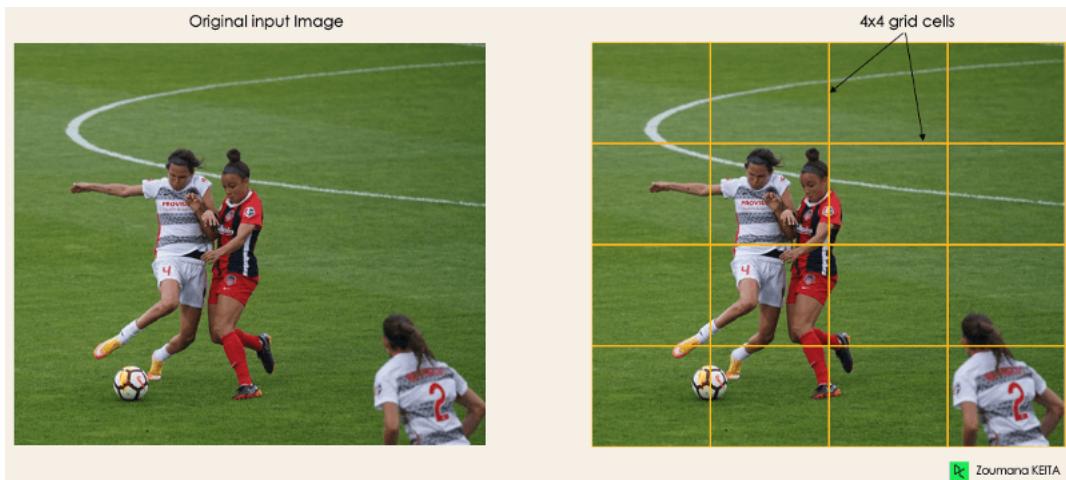


Slika 4: Arhitektura RetinaNet modela

3.2 Model YOLOv5

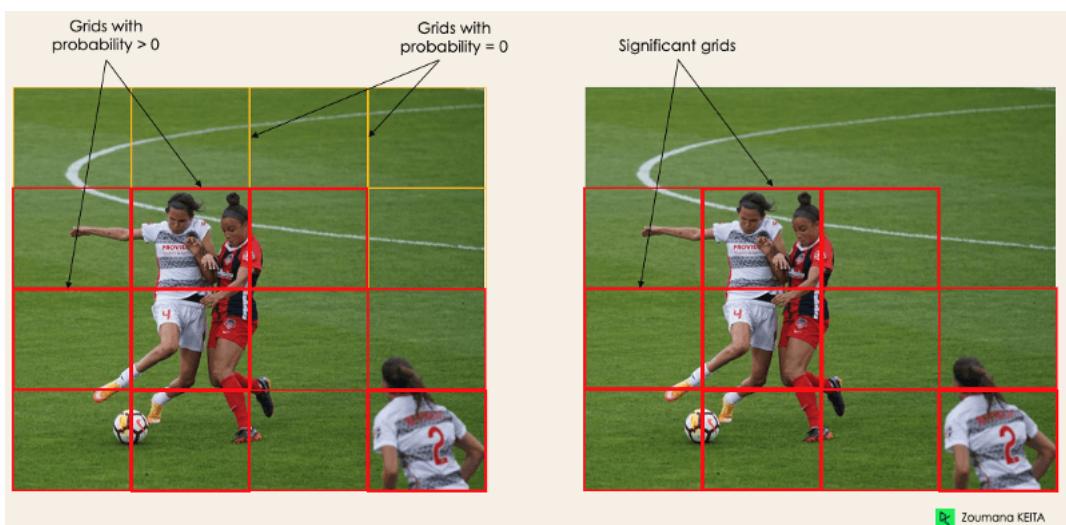
YOLO je 'state of the art' model za detekciju objekata koji je stekao veliku popularnost zahvaljujući njegovoj brzini i preciznošću. Pristup problemu kod YOLO modela je u potpunosti drugačiji od tradicionalnog modela, jer YOLO problem detekcije objekata svodi na problem regresije. Trenutno je najbrži od modela za detekciju objekata, ali ima problema sa detektovanjem velikih grupa sitnih objekata. [4]

YOLO algoritam ima četiri ključna koraka [5]. U prvom, ulazna slika se dijeli na NxN ćelija jednakog oblika (slika 5). Vrijednost N je obično 18, ali je korisno mijenjati tu vrijednost u slučaju koji će biti pomenut kasnije.



Slika 5: Podjela slike na ćelije

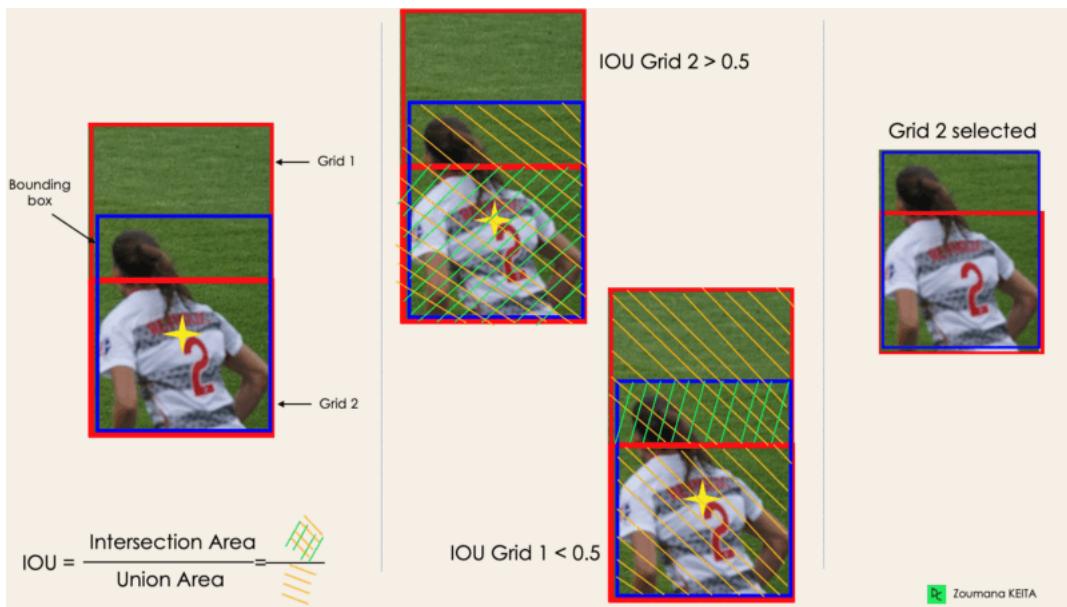
U drugom koraku svaka od ćelija daje neki broj predikcija. Jedna predikcija je formata: $Y = [pc, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_n]$. Sastoje se od koordinata centra objekta (b_x, b_y), širine i visine 'bounding box'-a (b_h, b_w), vjerovatnoća da se u ćeliji nalazi objekat pc , a za svaku od klasa daje se vjerovatnoća da je objekat u ćeliji te klase (c_1, \dots, c_n). Razmatramo samo ćelije u kojima postoji objekat, tj. $pc > 0$ (slika 6).



Slika 6: Ćelije u kojima je $pc > 0$

U koraku 3 isključuju se sve predikcije koje se ne podudaraju dovoljno sa anotiranim 'bounding box'-om (slika 7). Metrika koja se za ovo koristi je IOU (Intersection over union).

Kao što ime naglašava, IOU je razmjera između presjeka i unije predikcije i anotacije. Prihvataju se sva predviđanja čiji je IOU veći od zadatog praga.



Slika 7: Proces odabira najboljeg 'bounding-box'-a jedne ćelije

Ako je objekat dovoljno velik da pokriva više ćelija, jasno je da će za jedan objekat postojati više predikcija. Ovo može biti slučaj i ako jedna ćelija pokriva cijeli objekat, jer više predikcija te ćelije može imati IOU iznad zadatog praga. Zato se u koraku 4, primjenom 'non-max suppression' tehničke, od dobijenih predikcija uzima ona sa najvećom vjerovatnoćom da se objekat ispravne klase nalazi u predviđenom 'bounding box'-u.

Navedeni koraci račujanu se jednim prolaskom kroz neuronsku mrežu modela, što je i razlog njegove izuzetne brzine. Problem kod YOLO modela je što, ako su objekti sitni, više njih može da bude pokriveno u cijelosti jednom ćelijom. Pošto se od svake ćelije uzima samo jedna predikcija, modelu lako mogu promaći sitni objekti koji formiraju grupe. Ukoliko je ovo slučaj, vrijedno je eksperimentisati sa vrijednošću N , tj. brojem ćelija, ali treba imati na umu da će broj računanja rasti eksponencijalno u odnosu na N , i time negirati glavnu prednost YOLO modela - brzinu.

4 Metrike

U radu [1], mjera uspješnosti modela od 94.75% izračunata je metrikom 'mean Average Precision' (mAP). U ovoj sekciji objašnjeno je šta ova metrika predstavlja.

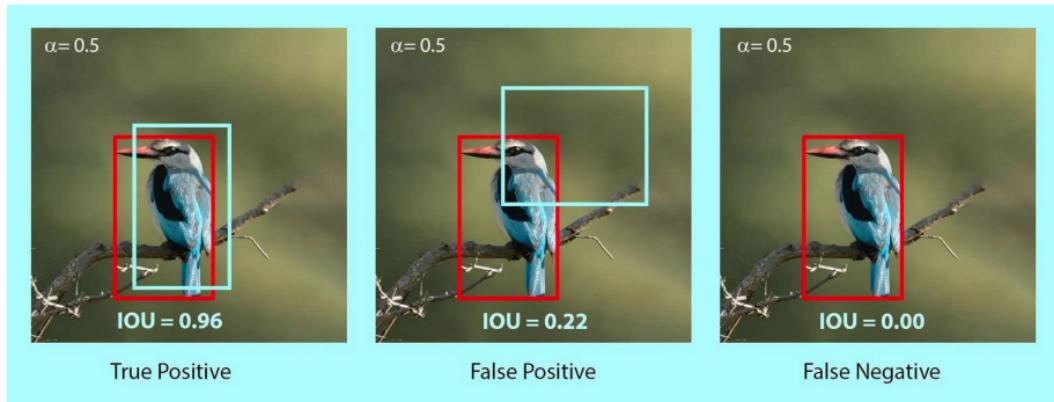
U sekciji 3.2 bilo je pomena o metriči IoU. Ova metrika se koristi kao pomoćna mjera za mAP. Ako odredimo neki prag α (slika 8), svaku IoU vrijednost možemo označiti kao 'True Positive' (TP), 'False Positive' (FP) i 'False Negative' (FN). Pomoću ovih vrijednosti možemo izračunati 'Precision' (P) i 'Recall' (R), na sljedeći način:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}$$

'Average Precision' (AP) računa se za svaku klasu posebno, kao aproksimirana površina ispod 'Precision-Recall' krive dobijene za tu klasu. mAP se dobija kao prosjek AP vrijednosti po svim kategorijama.

Mijenjanjem vrijednosti α , tj. pomijeranjem praga nakon kog neki IoU smatramo kao TP, nastaju varijante metrike mAP. Po običaju, mAP ima vrijednost praga $\alpha = 0.5$, i ovaj prag je

korišten u radu [1]. Često se koristi i mAP@[0.5:0.05:0.95], gdje se računa prosjek 10 mAP vrijednosti, za $\alpha = 0.5, 0.55, \dots, 0.95$.



Slika 8: Primjer TP, FP i FN predikcija

5 Trenutno stanje projekta

Tim Roboflow kreirao je tutorial Jupyter Notebook za YOLOv5 [6], pomoću koga sam trenirao model na dvije verzije dataset-a. U sekciji 5.1 govorim o ovim verzijama i njihovim rezultatima. U zavisnosti od 'backbone' modela, YOLOv5 ima nekoliko verzija. Trening je do sada vršen na najmanjoj, 'yolov5s'. 'Batch size' je podešen na 16, i broj epoha je 100. Obično su ovo mnogo veće vrijednosti, ali pošto radim sa malim brojem podataka, benefiti vjerovatno ne bi bili značajni. Svakako ostaje da se eksperimentiše sa ovim vrijednostima.

5.1 Verzije dataset-a i rezultati

Prvi pokušaj treniranja rađen je na originalnom dataset-u koji je objavljen u radu [1]. Slike (njih 423) podijeljene su na setove za trening, validaciju i testiranje po proporciji 70% - 20% - 10%. Nakon treninga, mAP nad testnim setom je 0.595%. Gledajući vizualizaciju rezultata, zaključio sam da je detekcija pistaća veoma dobra, ali klasifikacija je marginalno bolja od nasumičnog pogađanja. Pokušao sam da poboljšam ove rezultate tako što sam proširio dataset.

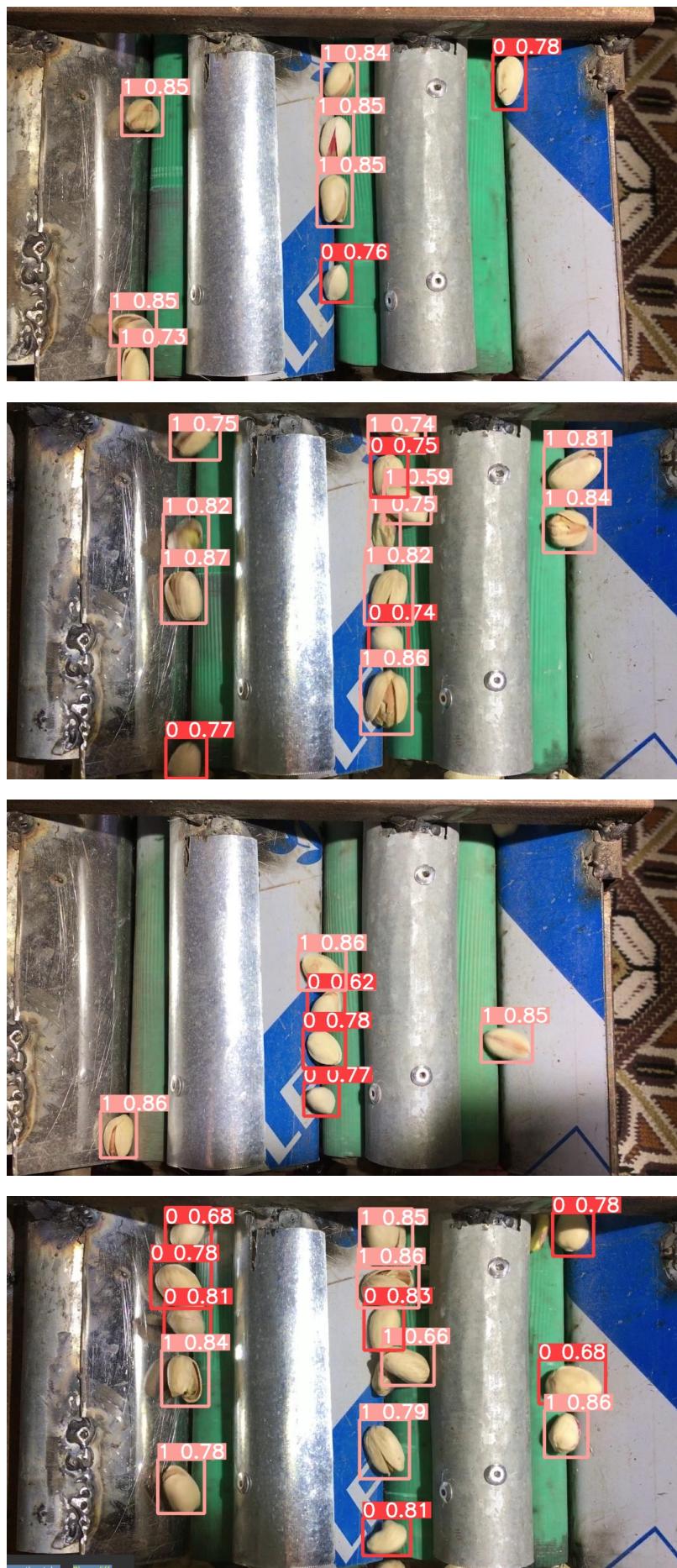
Primjenom augmentacija na dataset-u, njegova veličina je skoro duplirana u drugoj verziji, gdje broji 819 slika. Augmentacija nije rađena nad slikama iz testnog seta. Tip augmentacije je horizontalni i vertikalni flip originalne slike, što ne mijenja oblik ploda i daje modelu više primjera da bi mogao lakše da nauči da razaznaje klase. Treniranje na novom setu podataka pod istom konfiguracijom modela daje značajno bolje rezultate. U ovom slučaju mAP dostiže 94%, što dovodi ovaj YOLO model na nivo RetinaNet modela kreiranog u radu [1].

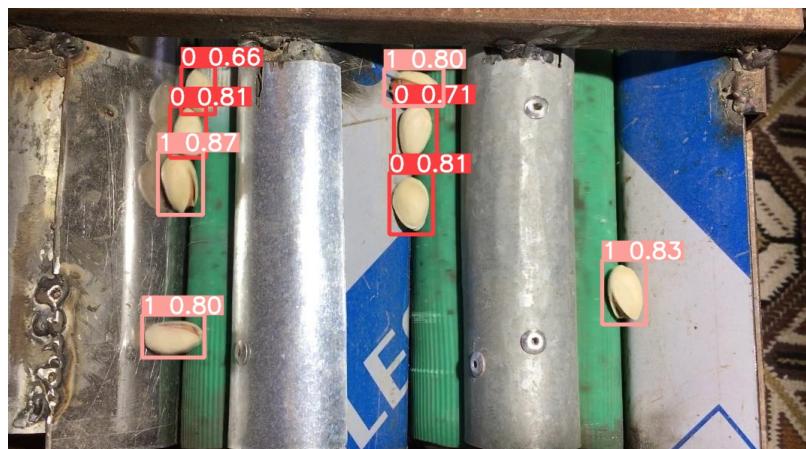
Ovo ne znači da je YOLO bolji od RetinaNet-a za datu svrhu, ali daje naznaku da bi veća verzija modela mogla da dâ slične ili bolje performanse, primjenjena na istim podacima kao i RetinaNet.

Za završni projekat pripremiću rezultate jačeg YOLOv5 modela testiranog na 6 video klipova. Takođe će dalje širiti dataset izvlačenjem i anotacijom frejmova iz video klipova, sa ciljem da dobijem najbolje moguće performanse.

5.2 Vizualizacija trenutnih rezultata

Slike iz testnog seta, sa vizualizovanim predikcijama Yolo modela.





Reference

- [1] Rahimzadeh, Mohammad & Attar, Abolfazl (2021). *Detecting and counting pistachios based on deep learning*; [<https://link.springer.com/article/10.1007/s42044-021-00090-6>].
- [2] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He & Bharath Hariharan (2016). *Feature Pyramid Networks for Object Detection*, [<http://arxiv.org/abs/1612.03144> arXiv:1612.03144].
- [3] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He & Piotr Dollár (2017). *Focal Loss for Dense Object Detection*; [<https://arxiv.org/abs/1708.02002>].
- [4] Joseph Redmon , Santosh Divvala, Ross Girshick& Ali Farhadi (2016). *You Only Look Once: Unified, Real-Time Object Detection*; [<https://arxiv.org/pdf/1506.02640v5.pdf>].
- [5] Zoumana Keitta (2022). *YOLO object detection explained*; [<https://www.datacamp.com/blog/yolo-object-detection-explained>].
- [6] Jacob Solawetz & Joseph Nelson (2020). *How to Train YOLOv5 On a Custom Dataset*; [<https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/>].