Sortiranje plodova pistaća na otvorene i zatvorene - Izvještaj

Matija Šuković, 1/22, D MAS

Prirodno-matematički fakultet
Univerzitet Crne Gore
24. 01. 2023.

Sadržaj

1 Uvod **1**
1.1 Prošireni dataset 1
1.2 Unaprijeđena arhitektura 1

2 Alati i kôd **3**
2.1 Programi za anotiranje dataseta 3
2.2 Roboflow 4
2.3 YOLOv5 by Ultralytics 4

1 Uvod

U ovom izvještaju govoriće se o trenutnom stanju projekta sortiranja pistaća na otvorene i zatvorene primjenom metoda dubokog učenja. Značaj teme i cilj projekta pokriveni su u seminarskom radu [1]. Performanse modela su unaprijeđene širenjem dataset-a kao i korišćenjem moćnije verzije algoritma YOLOv5. Detaljnije će se govoriti u podsekcijama 1.1 i 1.2 respektivno. Takođe će se objasniti alati i resursi korišćeni prilikom izrade modela (sekcija 2).

1.1 Prošireni dataset

Originalni dataset priložen uz rad [2] sadrži 423 anotirane slike na kojima se nalazi ukupno 3927 plodova. Uz dataset priloženo je i 6 videa iz kojih su autori izvukli slike za dataset. Dat je i kôd jednostavnih programa za ekstrakciju frejmova iz videa i za labelisanje slika.

Koristeći se priloženim alatima izvukao sam i anotirao 450 novih slika, iz svakog videa po 75, i time uvećao broj plodova za 3414, pa prošireni dataset sadrži 873 slike na kojima se nalazi ukupno 7341 plod.

Slike su kropovane da bi se smanjila veličina fajla i time ubrzalo treniranje, kao i da bi se otarasili nepotrebnih podataka (slika 1). Nakon toga primjenjena je augmentacija na dataset. Na 33% slika je primjenjen horizontalni ili vertikalni flip (slika 2), čime se dobija vještački dataset od 1296 slika. Podaci su podijeljeni na trening, validacioni i testni set po proporcijama 80%-15%-5%. Testni set se sastoji isključivo od originalnih slika na kojima nije primijenjena augmentacija.

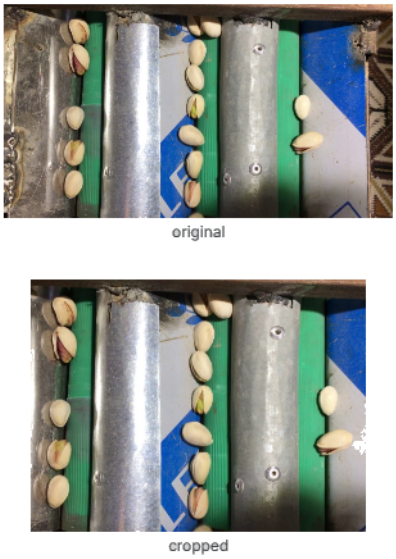
Za pripemu dataseta za treniranje koristio sam platformu roboflow.ai, koja umnogome olakšava manipulaciju podataka za detekciju objekata (sekcija 2.2). Na njoj se takođe nalazi blog koji mi je bio od velike pomoći pri izradi projekta i treniranju modela.

1.2 Unaprijeđena arhitektura

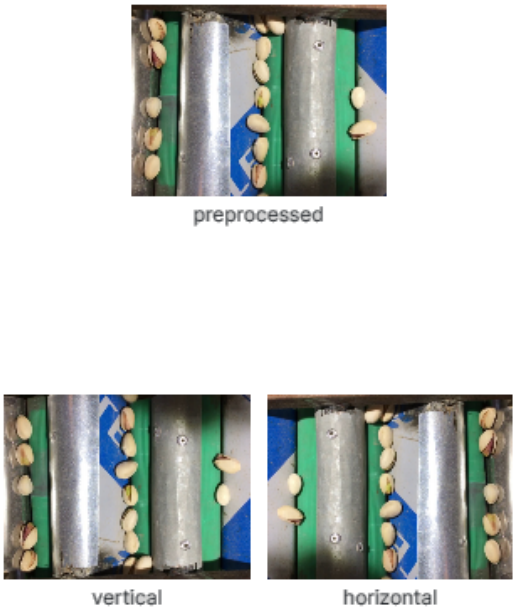
Postoje različite verzije YOLOv5 algoritma koje se razlikuju po kompleksnosti. Svaka verzija je skalirana varijanta iste arhitekture gdje se broj i veličina slojeva mijenjaju. Korišćenjem moćnije verzije poboljšavaju se performanse, ali se pogoršava brzina zaključivanja modela (slika 3).

Kao što je pomenuto u seminarskom radu [1], za prvi model sam koristio YOLOv5s verziju i dostigao mAP@.5 = 0.94. Ovaj rezultat je u rangu sa razultatom modela baziranog na RetinaNet arhitekturi kojeg su istrenirali autori rada [2]. Brzina zaključivanja ovog modela je 9.0ms, što je ekvivalentno 111 frejmova u sekundi (FPS).

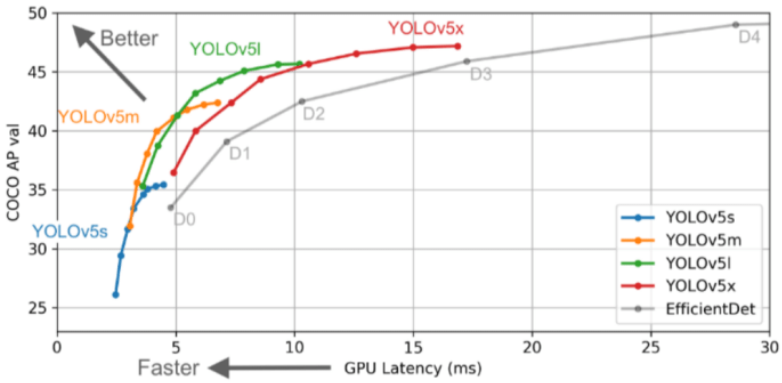
U želji da postignem uspješnost sličnu najboljim mašinama za sortiranje pistaća (98%), počeo sam da eksperimentišem sa ostalim verzijama YOLOv5 algoritma. Odlučio sam se za YOLOv5m verziju jer ona daje najbolji balans između performansi i brzine zaključivanja za



Slika 1: Uklanjanje nepotrebnih djelova slika radi uštede radne memorije



Slika 2: Kreiranje dodatnih primjera flipovanjem originalnih slika



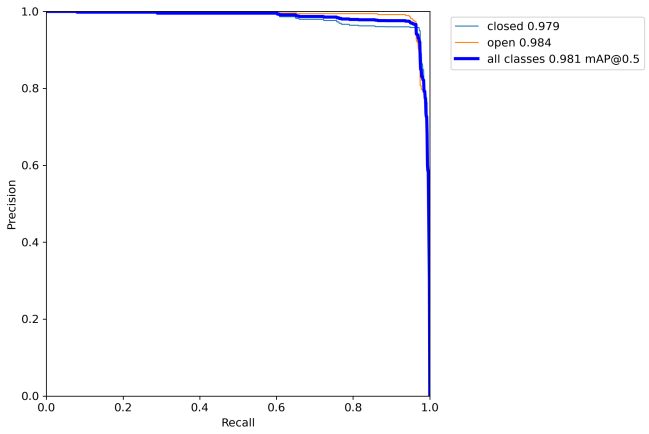
Slika 3: Graf koji prikazuje odnos između brzine algoritma i njegovih performansi na COCO datasetu (izvor: [3])

konkretnu svrhu. Uzeo sam u obzir i činjenicu da bi u praksi algoritam morao da radi na slabom procesoru sa ograničenom memorijom. Veličina modela naglo raste sa naprednijim verzijama modela, ići dalje od verzije 'l' vjerovatno ne bi imalo praktične koristi. Vrijeme treninga takođe raste sa kompleksnijim verzijama, pa sam zbog svega navedenog odlučio da koristim YOLOv5m.

Trening YOLOv5s verzije rađen je kroz 100 epoha i nad slikama koje su skalirane na širinu od 416 piksela. Pošto razlikovanje orvorenih i zatvorenih plodova često zahtijeva fine detalje, pri treniranju verzije 'm' nisam radio skaliranje. Takođe, postavio sam broj epoha na 1000 da bih osigurao potpuno iskorištavanje dataseta.

Kompleksniji model, sirovi podaci i ogroman broj epoha drastično će uticati na trajanje treninga. Dok je treniranje YOLOv5s modela trajalo svega 20 minuta, za treniranje modela m trebalo je 9 sati. Trening sam zaustavio kod epohe broj 639 jer se uspješnost modela nije značajno mijenjala kroz podlednjih 100 epoha.

Model istreniran na ovaj način je evaluiran sa mAP@.5 = 0.981, dok je brzina zaključivanja 16.2ms, što je ekvivalentno 61.7 frejmova po sekundi. Ovaj model dostigao je željene performanse, ali 61 FPS je 'previše brzo' za konkretan zadatak. Obično je kroz par frejmova jasno da li je plod otvoren ili zatvoren, a u snimcima sa kojim radimo jedan plod se nalazi na ekranu nekoliko desetina do stotina frejmova. Vjerovatno bi se isplatilo koristiti kompleksniju verziju modela i žrtvovati brzinu zaključivanja za još bolje performanse.



Slika 4: PR kriva modela baziranog na YOLOv5m verziji

2 Alati i kôd

U ovoj sekciji proći ćemo, u kratkim crtama, kroz kôd i izvore koje sam koristio za izradu projekta.

2.1 Programi za anotiranje dataseta

Uz rad [2] dolazi GitHub repozitorijum koji ,pored dataseta, sadrži i jednostavne programe za manipulaciju datim podacima. `frame_generator.py` koristi OpenCV funkciju `VideoCapture` da video pretvori u niz frejmova koji se onda čuvaju kao slike. Zatim se folder sa slikama predaje programu `label_maker.py`, koji koristi OpenCV funkciju `SelectROI` za odabir regiona od interesa na slici. Prolazi se kroz svaku sliku u datom folderu i kreira se .csv fajl sa unesenim podacima. `visualization.py` je program koji crta regione od interesa na slici i koristan je za validaciju unesenih podataka. Navedene programe sam modifikovao kako

bih sebi olakšao posao. Modifikovane skripte, kao i druge koje sam napravio i koristio, dostupne su na mom GitHub repozitorijumu [1].

2.2 Roboflow

Roboflow je online platforma čiji je cilj da kompjutersku viziju učini što pristupačnijom. Imaju veoma informativan blog i brojne tutorijale. Takođe imaju alat koji olakšava pripremu dataseta za trening. Nakon uploadovanja dataseta, lako se mogu uređivati klase i anotacije, dodatvati preprocesing, raditi augmentacija i slično. Više različitih verzija može se čuvati na njihovom serveru i zatim lako importovati u projekat koristeći njihov Python paket:

```
from roboflow import Roboflow
rf = Roboflow(api_key="YOUR API KEY HERE")
project = rf.workspace().project("YOUR PROJECT")
dataset = project.version("YOUR VERSION").download("yolov5")
```

Kao osnovu za treniranje modela koristio sam objavu [4] na njihovom blogu i Colab Notebook koji ide uz nju.

2.3 YOLOv5 by Ultralytics

YOLOv5 je inicijalno napravljen za Darknet, framework za neuronske mreže napisan u jeziku C. Sa ciljem da arhitekturu učini pristupačnijom, organizacija Ultralytics je napravila repozitorijum koji implementira YoloV5 za PyTorch. Koristio sam ovaj repozitorijum za izradu projekta.

Skripta `train.py` je zadužena za treniranje modela. Predaje joj se konfiguracija modela i podaci o datasetu, a ima i veliki broj opcionih argumenata. Poziv koji sam koristio za treniranje je:

```
python3 train.py
    --batch 64
    --epochs 1000
    --data {dataset.location}/data.yaml
    --cfg ./models/custom_yolov5m.yaml
    --name yolov5m_results
    --cache
```

`--data` je argument kome se predaje .yaml fajl koji sadrži detalje o datasetu, kao što je broj klasa i putanje do testnog, validacionog i trening seta. Moj fajl, `data.yaml`, izgleda ovako:

```
names:
- closed
- open
nc: 2
test: Pistachios-4/test/images
train: Pistachios-4/train/images
val: Pistachios-4/valid/images
```

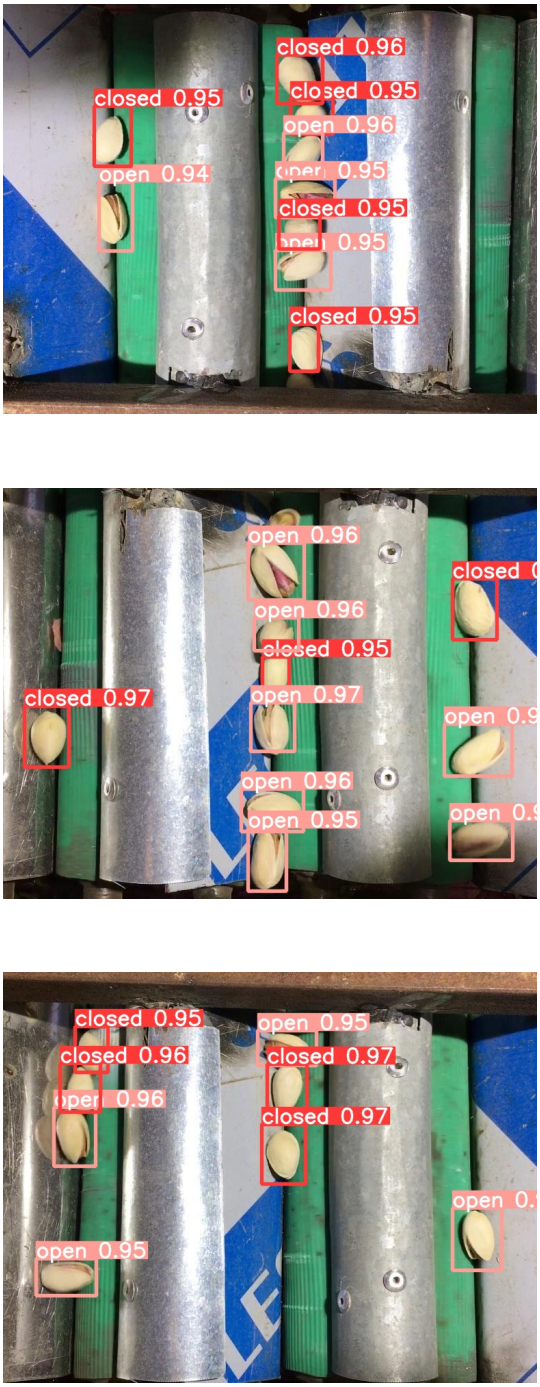
`--config` argumentu se predaje putanja do konfiguracije modela. U mom slučaju, to je konfiguracija YOLOv5m verzije.

Po završetku svake epohe, skripta `train.py` čuva najnovije parametre modela kao `last.pt`. Ukoliko su ti parametri dali model sa najboljim performansama do sada, oni se takođe čuvaju kao `best.pt`.

Skripta `val.py` se pokreće od strane `train.py` nakon svake epohe da bi se trenutni model evaluirao na validacionom skupu. Može i manualno da se pozove, a postavljanjem argumenta `--task test` model se može evaluirati na testnom skupu.

Nakon treniranja može se pokrenuti skripta `detect.py`. Njoj se prosljeđuju slike nad kojima se radi predikcija kao i parametri modela (`best.pt`). Skripta vraća predikcije, slike sa nacrtanim predikcijama kao i brzinu zaključivanja modela.

Kompletan kôd koji sam koristio za kreiranje i evaluaciju krajnjeg modela nalazi se na mom GitHub repozitorijumu [1].



Slika 5: Predikcije modela na testnim slikama

Reference

[1] Šuković, Matija (2022). *Pistachio sorting using deep learning and YOLO algorithm*; [https://github.com/matijasukovic/Pistachio-sorting-with-deep-learning-YOLO].

[2] Rahimzadeh, Mohammad & Attar, Abolfazl (2021). *Detecting and counting pistachios based on deep learning*; [https://link.springer.com/article/10.1007/s42044-021-00090-6].

[3] Rakkshab, Iyer; Priyansh Shashikant, Ringe & Bhensdadiya, Kevin Prabhulal (2021). *Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection*; [https://tinyurl.com/yc5w957c].

[4] Solawetz, Jacob & Nelson, Joseph (2020). *How to Train YOLOv5 On a Custom Dataset*; [https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/].