УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Departman za računarstvo i automatiku

Smer računarstvo i automatika

PROJEKTNI ZADATAK

Studenti: **Milica Vojnović RA 59-2020**

Pavle Vasiljević RA 207-2020

Predmet: **Algoritmi digitalne obrade slike**

Tema rada: **Mace**

Novi Sad, jun 2023.

Projektni zadatak

Tema ovogodišnjeg projekta je korišćenje YOLO algoritma kako bi se detektovao izabrani objekat, koristeći znanje stečeno na predmetu Algoritmi digitalne obrade slike.

U prvom delu projektnog zadatka, izabrali smo objekat koji će da bude tema našeg projekta. U našem slučaju, to su bile **mačke**. Sakupili smo fotografije mačaka, koje su bile naš početni skup podataka. Zatim smo izvršili četiri različite augmentacije na ove slike, kako bismo povećali skup podataka. Augmentacije su bile sledeće: *DCT lowpass Unsharp masking, Saturation, Aspect and Cut, Occlusions.* Nakon dobijih slika, na svakoj smo labelirali traženi objekat. Na kraju, dobili smo pripremljeni skup za treniranje YOLO modela.

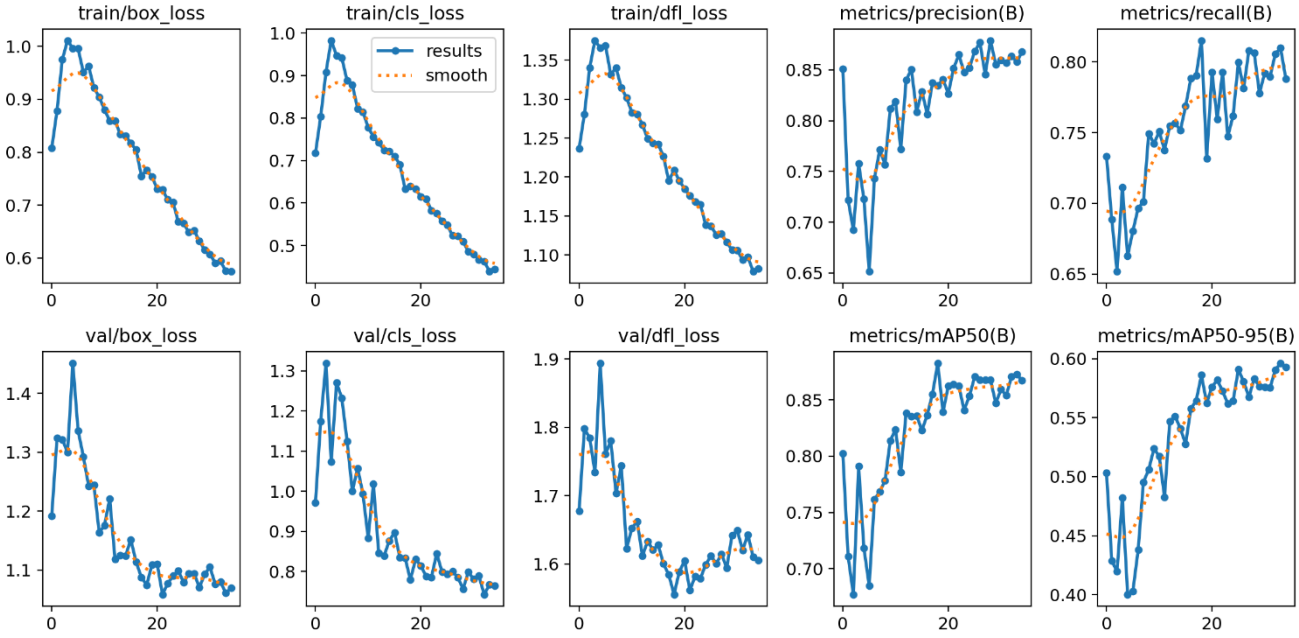
Zadatak drugog dela projektnog zadatka je da istreniramo YOLO model na pripremljenom trening skupu. Nakon treniranja, potrebno je odraditi predikciju tj. detekciju objekta iz skupa podataka koji nije korišćen tokom treniranja i obraditi rezultate predikcije. U nastavku ove dokumentacije, biće zabeleženi koraci drugog dela projekta, kao i dobijeni rezultati.

Treniranje YOLO modela

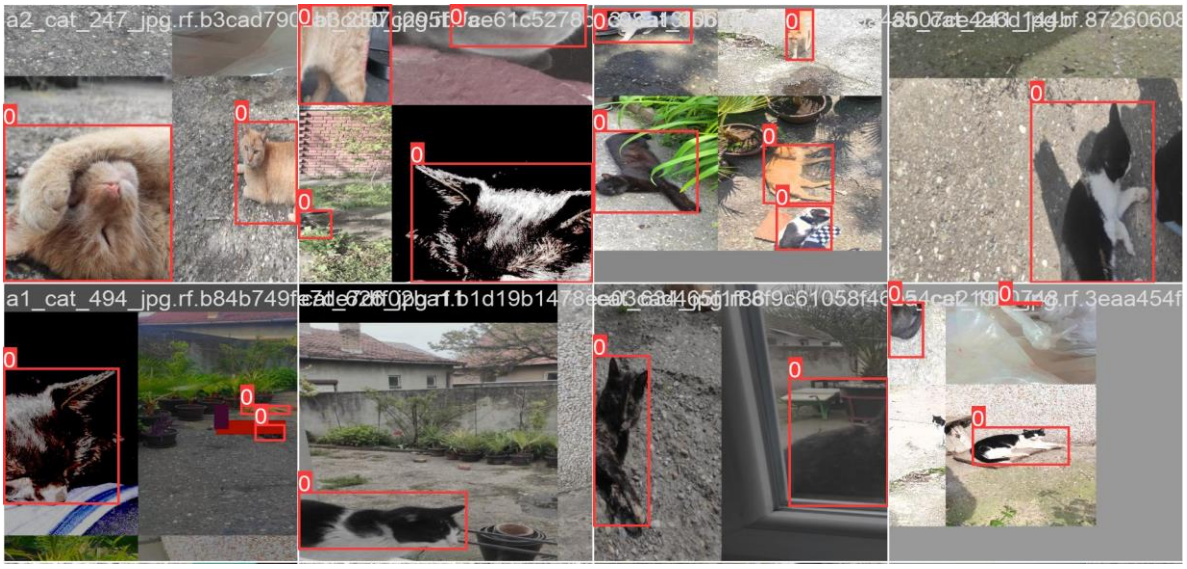
Pre treniranja, pripremili smo **data.yaml** datoteku, tako da sadrži putanju do trening i validacionog skupa podataka, broj klasa i nazive klasa. U našem slučaju, imamo jednu klasu i ona se zove **cat**. Zatim, koristimo python skriptu u kojoj pozivamo funkciju za treniranje YOLO modela iz ultralytics biblioteke. Kao argument za model postavili smo yolov8s.pt.

```
model.train(task='detect', data='data.yaml',
                    epochs=50, imgsz=640, cache=True)
```

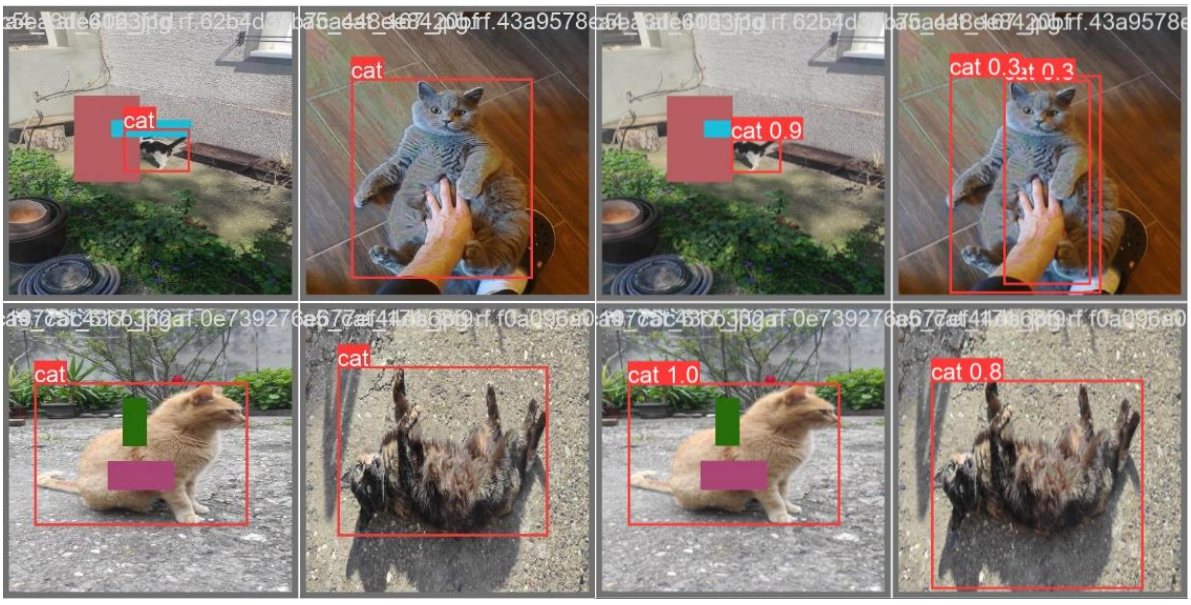
Nakon treniranja, dobijamo rezultate i statistiku o tome kako je treniranje prošlo.



Ovde možemo da primetimo, da se tokom treniranja svi parametri teže svom optimumu, kao npr. broj gubitaka u labeliranju (box_loss), broj gubitaka u klasama (cls_loss), preciznost.. Važno je napomenuti, da se nakon 20 epoha, prilikom validacije, funkcija koja nam pokazuje napredak modela počne da se normalizuje i zaravnjuje. To nam govori da daljim treniranjem, nećemo postići znatno bolje rezultate.



Ovde je prikazan jedan primer gomile za treniranje (train batch) koju YOLO formira, kako bi na njima trenirao.



Takođe kao izlaz koristi slike za validaciju i njihove labele, vrši predikciju na istim slikama i dobijamo poređenje naših labela i YOLO detekcije.

Odrađivanje predikcije

Sada kada imamo istrenirani model, sledeći korak je da ga koristimo. Kao izlaz iz treniranja imamo dva različita rezultata, **best.pt** i **last.pt**. Best.pt je težina neuronske mreže kada je imala najbolje rezultate nad validacionim skupom podataka, a last.pt je težina neuronske mreže iz poslednje epohe. Uradimo predikciju nad oba test skupa i poredimo njihove rezultate. Za predikciju koristimo python skriptu u kojoj pozivamo funkciju za predikciju iz ultralytics biblioteke nad svim primercima iz seta za test.

Takođe, pozivamo detekciju koristeći **stock YOLO model**, kako bi imali da poredimo rezultate sa već postojećim modelom za detekciju mačaka. Prilikom poziva, moramo da prosledimo parameter `classes=15.` Svaka klasa u stock YOLO modelu ima dodeljeni broj, a u našem slučaju klasa cat je broj 15.

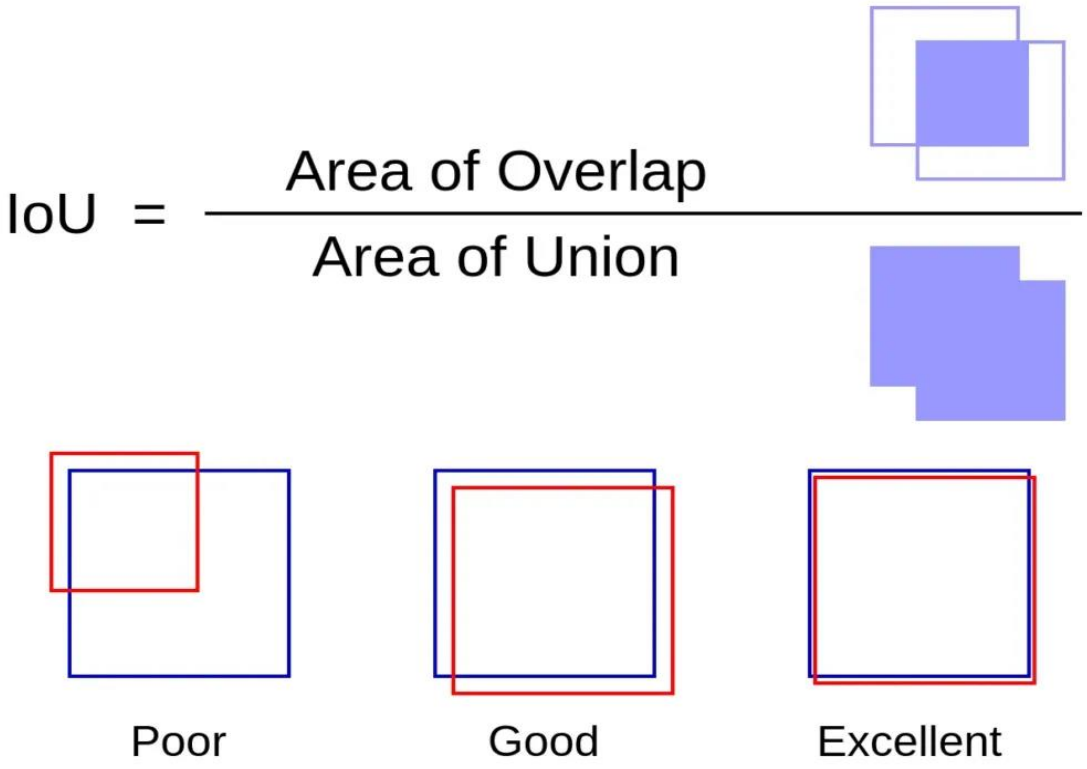
```
model = YOLO('best.pt')
model.predict(source='datasets/test',
      save_txt=True, save=True,
      project="predictionResultDumpBest")
```

Kao rezultat izvršenja dobijamo kordinate labela i sliku sa označenom labelom.

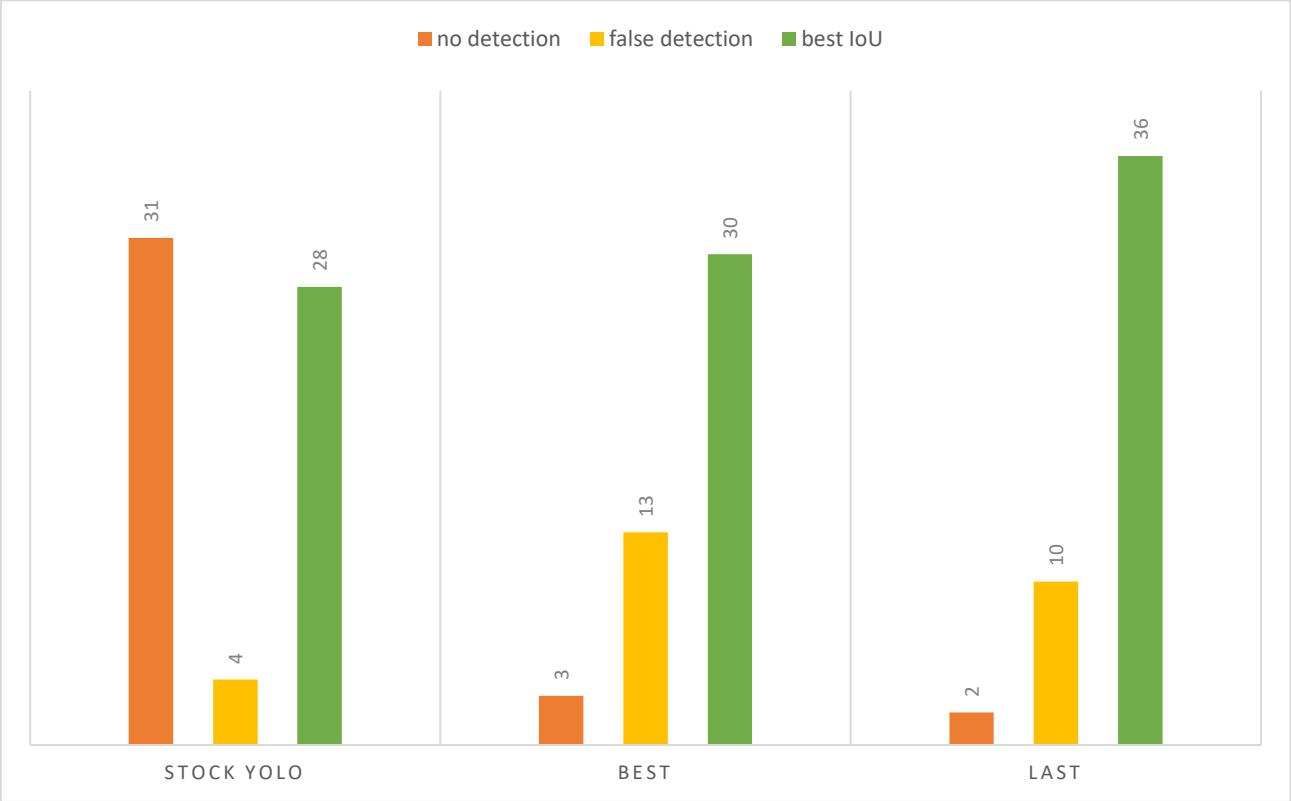
Poređenje i statistika

Poslednji korak ovog projekta je da uzmemo dobijene izlaze iz tri različite detekcije, na istom setu slika, i uporedimo njihove performanse. Jedan od parametara koji možemo izračunati je **Intersection over Union**, tj. IoU metriku za Koliko se poklapaju s originalnom labelom i uporediti. Za računanje IoU, potrebna nam je **Area of Overlap**, koja nam govori kolika je površina preklapanja labelirane i detektovane oblasti (preklapanje), kao i **Area of Union**, koja je kombinovana površina labelirane i detektovane oblasti (unija).

IoU(A, B) = A ∩ B / A ∪ B



Nakon što smo koristili skriptu koja računa IoU za sva tri modela, imamo sl. rezultate:



Ovaj grafik je izveden na osnovu informacija iz excel tabele. Bitno je napomenuti da smo koristi set test slika koje su izvedene na slučajan način iz prikupljenih slika za validaciju. Stock YOLO ubedljivo ima najveći broj nedetektovanih objekata u odnosu na druga dva. Takođe možemo da primetimo veći broj lažnih detekcija, koje se obično javljaju na slikama koje imaju više objekata. Nedotreniran YOLO model ponekad prepozna te slike, ali umesto lažnih detekcija ih češće ne detektuje uopšte. Najveći broj najboljih detekcija ima model Last, što je bilo i očekivano. Dodatnim treniranjem modela i većim setom za treniranje bi se dodatno povećala razlika između ova tri modela.

Izračunali smo prosečne IoU posebno za svaki model, sa i bez uračunatih kaznenih vrednosti za nedetektovanje objekta. U oba slučaja se Last pokazao kao najbolji. Stock YOLO je u slučaju kada se ne koriste kaznene vrednosti gotovo isto dobar koliko i Last, ali kada su one uračunate, znatno opada njegova performansa.