**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**DETEKCIJA OBJEKATA NA SLICI KORIŠTENJEM YOLOv5**

Kolegij: Raspoznavanje uzoraka i strojno učenje

**Robert Pepić**

Osijek, 2022.

Sadržaj

[1. Uvod 1](#_Toc106545287)

[2. YOLO algoritma 2](#_Toc106545288)

[2.1 Razlike među verzijama implementacije YOLO algoritma 3](#_Toc106545289)

[3. Priprema podataka 4](#_Toc106545290)

[4. Ultralytics YOLOv5 5](#_Toc106545291)

[4.1 Instalacija i potrebni paketi 5](#_Toc106545292)

[4.2 Unaprijed istrenirani modeli 5](#_Toc106545293)

[4.3 Podjela podataka na skupove za treniranje i validaciju 7](#_Toc106545294)

[4.4 Prilagodba modela vlastitom skup podataka 7](#_Toc106545295)

[5 Rezultati 9](#_Toc106545296)

[5.4 Evaluacija modela 9](#_Toc106545297)

[5.2 Primjena modela 10](#_Toc106545298)

[5.2 Poboljšanje modela 12](#_Toc106545299)

[6. Zaključak 13](#_Toc106545300)

# Uvod

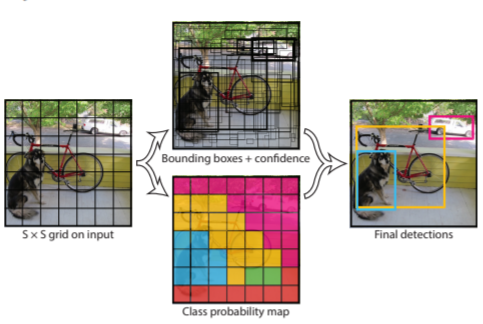
Detekcija objekata na slici predstavlja složeniji zadatak od same klasifikacije objekata. Ne samo da je potrebno klasificirati objekt koji se nalazi na slici, već je potrebno odrediti gdje se isti i nalazi. Problem je dodatno otežan time što je na istoj slici moguće imati više objekata različitih klasa. Implementacija ovakvih algoritama vrlo je složena. Neki od najpoznatijih i najkorištenijih su YOLO, SSD i R-CNN. Ovaj rad fokusirat će se na YOLO algoritam. Postoji nekoliko gotovih implementacija ovog algoritma koji su besplatni za korištenja i dolaze sa već prethodno istreniranim mrežama. Te mreže istrenirane su na COCO skupu podataka, no moguće ih je dodatno prilagoditi, odnosno istrenirati na vlastitim skupovima podataka. Ovakav pristup učenju, gdje se koriste već istrenirane mreže, naziva se transfer učenje. Ideja transfer učenja je da se znanje dobiveno rješavanjem jednog problema iskoristi u rješavanju drugog, sličnog problema. U okviru ovog rada za tu svrhu koristit će se *Ultralytics-ova* YOLOv5 verzija koju je moguće preuzeti s njihovog službenog git repozitorija.

Ideja ovog projektnog zadatka je objasniti princip rada YOLO algoritma, pokazati kako radi gotova implementacija istog te kako se koristiti njime prethodno istreniranim mrežama. Nadalje, pokazati će se kako je te mreže moguće prilagoditi vlastitom skupu podataka te kako je te podatke potrebno pripremiti za treniranje mreže. Kao objekti od interesa odabrani su šalica, sat i naočale.

# YOLO algoritma

YOLO *(engl. You only look once)* je algoritam koji koristi konvolucijske neuronske mreže (CNN) za otkrivanje objekata u stvarnom vremenu. Ovaj algoritam je popularan zbog svoje brzine i točnosti. Koristi se u raznim aplikacijama za otkrivanje objekata kao što su ljudi, životinje, vozila, prometni znakovi i slično. Kao što mu ime sugerira, algoritam zahtijeva samo jedan prolaz kroz neuronsku mrežu da bi otkrio objekte. Kao rezultat vraća sliku na kojoj su označeni objekti, klase kojima ti objekti pripadaju i vjerojatnosti da pripadaju tim klasama.

Algoritam se sastoji od nekoliko koraka. Prvo, slika se dijeli na ćelije pomoću mreže koja ima dimenzije S x S. Zatim se unutar svake ćelije predviđa B graničnih okvira (*engl. Bounding box*) i za svaki se računa njegova pouzdanosti. Granični okvir definiran je koordinatama središta, visinom, širinom i klasom objekta kojeg opisuje. Računanje kvocijenta presjeka i unije površina stvarnog i predviđenog okvira *(engl. Intersection over Union)* osigurava da su predviđeni granični okviri što je moguće sličniji stvarnim okvirima objekata. Ovaj korak eliminira nepotrebne granične okvire koji ne zadovoljavaju karakteristike objekata poput visine i širine, u slučaju da se isti objekt detektira više puta. Dodatno, okviri koji su ispod određenog praga pouzdanosti neće se prikazivati. Konačna detekcija sastojat će se od jedinstvenih graničnih okvira koji savršeno pristaju objektima.



*Slika 2.1 Pojednostavljeni prikaz koraka YOLO algoritma*

Izvor: <https://www.guidetomlandai.com/assets/img/computer_vision/YOLO.PNG>

## Razlike među verzijama implementacije YOLO algoritma

YOLO algoritam ima sveukupno pet službenih verzija, od YOLOv1 do YOLOv5. Sve one rade na sličan način kao što je prethodno objašnjeno, međutim svaka novija verzija dodala je određene inovacije u odnosu na svog prethodnika. Neke od najvažnijih inovacija koje je pojedina verzija uvela:

* **YOLOv1**: Slika se dijeli na ćelije pomoću mreže dimenzija S x S, svaka ćelija odgovorna je za detekciju objekata i računanje pouzdanosti
* **YOLOv2**: Dodan *K-means Anchor,* uvedeno dvo-stupnjasko treniranje, potpuna konvolucijska mreža
* **YOLOv3**: Multiscale detekcija korištenjem FPN-a (*Feature pyramid network*)
* **YOLOv4**: SPP (*Spatial Pyramid Pooling*), MISH aktivacijska funkcija, Mosaic/Mixup poboljšanje podataka, GIOU kriterijska funkcija (*Generalized Intersection over Union*)
* **YOLOv5**: Fleksibilna kontrola veličine modela, primjena Hardswish aktivacijske funkcije i poboljšanja podataka

Kao što je i spomenuto na početku, u okviru ovog rada koristit će se *Ultralytics-ova* YOLOv5 implementacija YOLO algoritma.

# Priprema podataka

Kako bi se trenirao prilagođeni model, potrebno je pripremiti skup podataka reprezentativnih slika. Za svaku sliku potrebno je definirati granične okvire (*engl. Bounding box*) oko objekata koje želimo otkriti te klase kojima ti objekti pripadaju. Ovi podaci pohranjuju se u tekstualnu datoteku koja mora imati isto ime kao slika koju ti podaci opisuju. Ti podaci moraju biti pohranjeni u odgovarajućem YOLO formatu. Svaka linija u tekstualnoj datoteci predstavlja jedan objekt na slici. U svakoj liniji nalazi se pet vrijednosti. Prva vrijednost je broj koji predstavlja klasu kojoj objekt pripada, dok ostale četiri određuju položaj objekta na slici.

Postoje mnogi alati koji olakšavaju ovaj proces. U okviru ovog rada koristit će se aplikacija *Labelimg.* Aplikacija se relativno lagano koristi. Prvo joj je potrebno predati putanju do direktorija u kojem se nalaze slike koje želimo označiti. Zatim putanju direktorija u koji želimo pohraniti tekstualne datoteke s oznakama. Format označavanja potrebno je podesiti na yolo. Nakon toga može se započeti s označavanjem slika. To se radi pritiskom na gumb *CreateRectBox* i označavanjem područja na kojem se nalazi objekt u pitanju. Nakon što je objekt označen potrebno je odrediti kojoj klasi objekt pripada te kliknuti na gumb *save.* Aplikacija će tada generirati odgovarajuću tekstualnu datoteku za tu sliku u kojoj su pohranjeni podaci o objektima koji se na njoj nalaze. Na slici ispod može se vidjeti izgled aplikacije te kako se pomoću nje određuju granični okvir i klasa objekta.

Slika na kojoj se prikazuje tekst, na zatvorenom, šalica, kava

Opis je automatski generiran

*Slika 3.1 Određivanje graničnih okvira objekta korištenjem Labelimg aplikacije*

# Ultralytics YOLOv5

## Instalacija i potrebni paketi

Prva stvar koju je potrebno napraviti prije nego što bi se uopće mogao koristit YOLOv5 je instalirati *Pytorch* framework za strojno učenje. Instalacija je vrlo jednostavna. Na službenoj *Pytorch* stranici potrebno je odabrati verziju, operacijski sustav, programski jezik i platformu za računanje, nakon čega će se generirati komanda za instalaciju paketa. Istu je tada samo potrebno pokrenuti u razvojnom okruženju u kojem se radi. U okviru ovog rada koristit će se python programski jezik i jupyter notebook. Zadnja stvar koju je potrebno napraviti je klonirati Ultralytics YOLOv5 repozitorij i instalirati sve njemu potrebne pakete, što se može postići slijedećim naredbama.

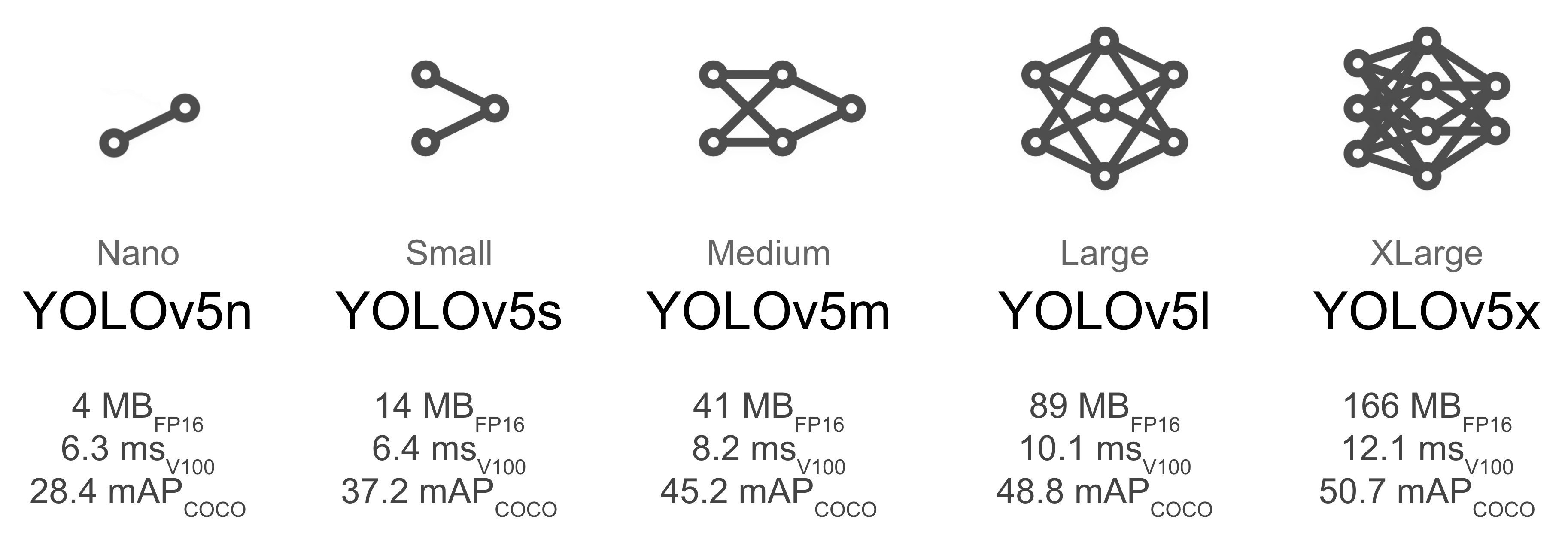
**Slika na kojoj se prikazuje tekst

Opis je automatski generiran**

*Slika 4.1 Naredbe za kloniranje YOLOv5 repozitorija i instalaciju potrebnih paketa*

## Unaprijed istrenirani modeli

Unaprijed istrenirani modeli, koje želimo prilagoditi novom skupu podataka, dostupni su na *PyTorch Hub-u*. *PyTorch Hub* je repozitorij unaprijed istreniranih modele dizajniran u svrhu istraživanja. Na *ultralytics\_yolov5 hub-u* dostupno je pet unaprijed istreniranih modela: Nano, Small, Medium, Large i xLarge. Specifikacije pojedinih modela su slijedeće.



*Slika 4.2 Specifikacije unaprijed treniranih YOLOv5 modela*

Izvor: <https://github.com/ultralytics/yolov5/releases/download/v1.0/model_comparison.png>

Može se primijetiti da što je model veći to mu je i preciznost *(engl. mAP – mean average precision)* veća, ali mu je zato zauzvrat brzina manja. U okviru ovog rada koristit će se YOLOv5l model. Kako bi ga se učitalo i koristilo unutar koda prvo je potrebno importirati *torch* biblioteku te zatim pokrenuti slijedeću liniju koda.



*Slika 4.3 Učitavanje YOLOv5l modela*

Ovaj model unaprijed je istreniran na COCO *(engl. Common Objects in Context)* skupu podataka, te ga se može odam koristit za detekciju čak do 80 različitih objekata. Korištenje modela vrlo je jednostavno i može se primijeniti na slici, videu ili izravno na snimci kamere korištenjem *webcam-a*. Na slikama ispod može se vidjeti kod potreban za detekciju objekta na slici, ispis rezultata te sliku na kojoj su označeni granični okviri oko detektiranih objekata.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

7

*Slika 4.4 Testiranje unaprijed istreniranog modela na slici*

Slika na kojoj se prikazuje tekst, osoba, na otvorenom

Opis je automatski generiran

*Slika 4.5 Rezultati testiranja unaprijed treniranog modela na proizvoljnoj slici*

## 4.3 Podjela podataka na skupove za treniranje i validaciju

Kako bi istrenirali model na vlastitom skupu podataka prvo je isti potrebno i napraviti. Kao objekti od interesa odabrani su sat, šalica i naočale. Za svaki objekt prikupljeno je stotinjak slika korištenjem *google images* usluge te su iste opisane korištenjem *Labelimg* aplikacije. Postupak je objašnjen u prethodnom poglavlju. Nakon što su na svim slikama označeni objekti, skup podataka potrebno je podijeliti na dva skupa. Jedan za treniranje modela, a drugi za validaciju modela. Kako je za svaki objekt od interesa prikupljeno sto slika, slike će se podijeliti u omjeru 80:20 u direktorije *images\_train* i *images\_test* unutar *yolov5/dataset/images* direktorija. Istu stvar potrebno je napraviti i s datotekama u kojima se nalaze opisi pojedine slike. Direktoriji u koje dijelimo opise slika moraju biti istoimeni kao i folderi u kojima su same slike. Jedina razlika je u tome što ih moramo pohraniti unutar *yolov5/dataset/labels* foldera. Nakon podjele imamo skup za treniranje s 240 opisanih slika i skup za validaciju s 60 opisanih slika. Ovime smo uspješno kreirali i podijelili skup podatak.

## Prilagodba modela vlastitom skup podataka

Nakon što je sastavljen skup podataka, unutar yolov5 direktorija potrebno je kreirati datoteku *dataset.yaml.* Unutar nje potrebno je prvo definirati lokaciju skupa podataka kojim želimo trenirati model. Zatim je potrebno odrediti koji dio podataka je za treniranje a koji za validaciju. Direktorij koja sadrži datoteke s graničnim okvirima *yolov5* automatski traži tako što u putanji jednostavno zamjeni *images* za *labels*, stoga lokaciju istog nije potrebno posebno navoditi ukoliko se nalazi tamo. Na kraju je još potrebno definirati broj i nazive klasa. Konačni sadržaj datoteke prikazan je na slici ispod.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

*Slika 4.6 Sadržaj dataset.yaml datoteke*

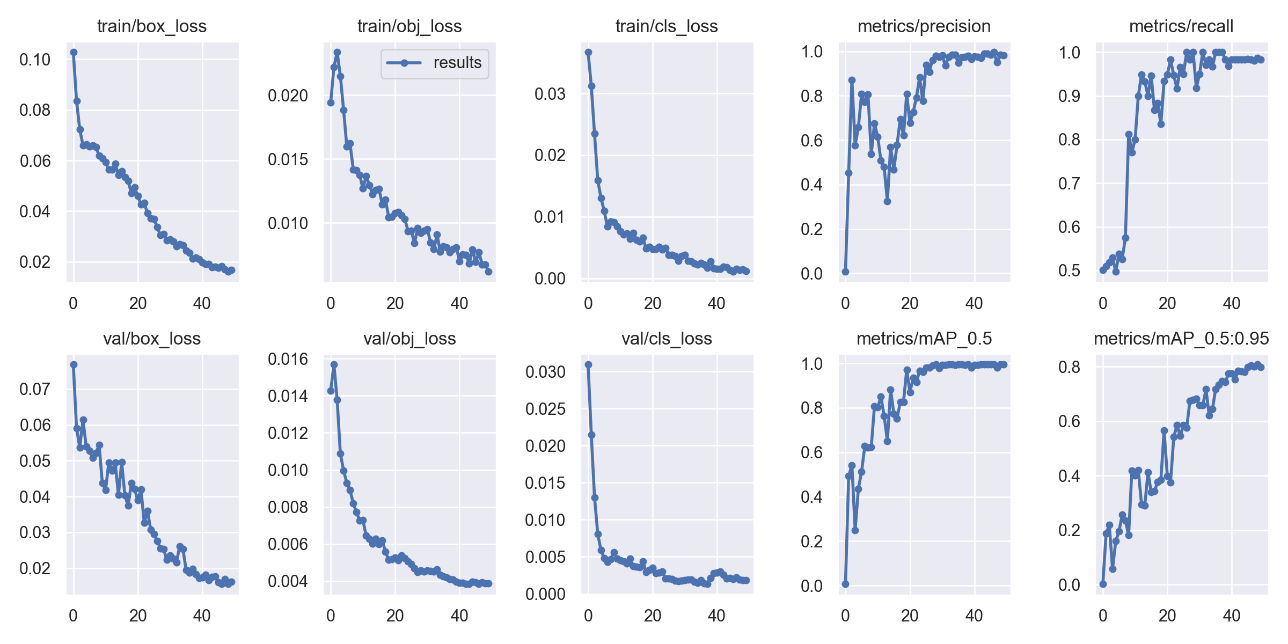
Nakon što je definirana *dataset.yaml* datoteka može se početi s prilagođavanjem odnosno treniranjem modela. Postupak je vrlo jednostavan. Prvo je potrebno ući u yolov5 direktorij, a zatim unutar njega pokrenuti train.py skriptu. Međutim istoj je prvo potrebno predati nekoliko parametara, od kojih su najvažniji *data* i *weights* parametri. Kao *data* parametar predaje se *dataset.yamal* datoteka kreirana u prošlom koraku, a ako *weights* parametar predajemo unaprijed istrenirani model koji smo preuzeli s *PyTorch Hub-a*. Ostali parametri, kao što su rezolucija slike, *batch size* i broj epoha, opcionalni su jer imaju predefinirane vrijednosti. Međutim, radi tehničkih ograničenja (zbog treniranja na vlastitom računalu) rezoluciju smanjujemo s zadanih 640 na 320, a *batch size* sa zadane 64 slike na 4 slika. Ovo je bilo potrebno napraviti jer se u suprotnom treniranje nije moglo uspješno dovršiti radi nedostatka memorije. Za sve navedeno potrebno je svega jedna linija koda, odnosno jedna naredba prikazana na slici 4.7. Pokretanjem iste započinje treniranje odnosno prilagođavanje gotovog modela vlastitom skupu podataka. Samo treniranje trajalo je oko dva sata.



*Slika 4.7 Naredba za pokretanje treniranja modela*

# Rezultati

## Evaluacija modela

Za evaluaciju modela mogu se koristiti razne metrike, međutim kod detekcije objekata od posebne važnosti su *precision*, *recall* i mAP (mean Averege Precision). Preciznost se računa kao omjer između broja pozitivnih uzoraka koji su ispravno klasificirani i ukupnog broja uzoraka klasificiranih kao pozitivni. *Recall* se računa kao omjer između broja pozitivnih uzoraka koji su ispravno klasificirani kao pozitivni i ukupnog broja pozitivnih uzoraka. *Recall* mjeri sposobnost modela da otkrije pozitivne uzorke, a preciznost mjeri točnost modela u klasificiranju uzorka kao pozitivnog. PR-krivulja (eng. *precission-recall curve*) prikazuje odnos između vrijednosti preciznosti i *recall*-a za različite *threshold* vrijednosti, a mAP je način sažimanja PR-krivulje u jednu vrijednost koja predstavlja prosjek svih preciznosti. Treniranje modela za detekciju objekata obično zahtijeva dva ulaza, a to su slika i okviri koji označavaju položaj objekata na slici. Kada model predvidi granični okvir, očekivano je da se predviđeni okvir neće točno podudarati s pravim okvirom (eng. *ground-truth*). IoU (Presjek preko unije) se izračunava dijeljenjem površine presjeka između dva okvira s površinom njihove unije. Općenito, što je veći IoU to su bolja predviđanja. Tokom treniranja, za svaku epohu YOLOv5 automatski prati i bilježi važne metrike te ih pohranjuje u *results.csv* datoteku. Također nakon što je treniranje gotovo te rezultate prikaže grafički i pohrani ih u *results.png.* Preciznost i recall računaju se na temelju trening skupa, doke se mAP računa na temelju validacijskog skupa. Slika 5.1 grafički prikazuje promjene vrijednosti važnih metrika tokom treniranja.

*Slika 5.1 Evaluacija modela*

Nakon evaluacije modela konačne vrijednosti preciznosti i *recall*-a su 0.982 i 0.983. Vrijednosti mAP-a prikazane su za dva slučaja. Prvi slučaj prikazuje mAP vrijednosti kada *threshold* za detekciju (IoU) iznosi 0.5, a drugi kada je *threshold* za detekciju 0.95. Njihove konačne vrijednosti su 0.995 i 0.79, na temelju kojih bi se moglo zaključiti da model dobro funkcionira.

## 5.2 Primjena modela

Nakon što je treniranje modela završilo, istrenirani model pohranjen je unutar *yolov5/runs/train/exp/weights* direktorija. Unutar njega nalaze se *last.pt* i *best.pt* datoteke. Prva od njih predstavlja model nakon zadnje izvršene epohe, dok druga predstavlja model koji daje najbolje rezultate. U svrhe ovog rada koristit će se *best.pt* model. Kako bi se isti moglo koristiti unutar koda potrebno je izvršiti slijedeću naredbu koja će ga učitati u varijablu *model*.



*Slika 5.2 pristupanje modelu istreniranom na vlastitom skupu podataka*

Za testiranje rada modela koristit će se web kamera laptopa. Web kameri pristupit će se korištenjem *openCV* biblioteke*.* Ona sadrži funkcije kao što su *videoCapture()* za pristup web kameri i *read()* koja kao rezultat vrača snimku web kamere. Snimku koja funkcija *read()* vrati konačno predajemo našem modelu koji će na njoj tražiti objekte od interesa. Ukoliko ih nađe oko istih će iscrtati granične okvire i vjerojatnosti. Rezultate možemo vidjeti korištenjem *imshow()* funkcije. Kako bi ovo radilo i uživo na video snimci, a ne samo na snimljenoj slici, cijeli ovaj postupak stavit će se u jednu *while* petlju s *waitKey(30)*. Na taj način će se svakih trideset milisekundi snimiti nova slika, na njoj pronaći i označiti objekti od interesa te rezultati prikazati na ekranu. Na slici 5.3 može se vidjeti isječak koda kojim se implementira ova funkcionalnost, a na slikama u nastavku rada mogu se vidjeti dobiveni rezultati.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

*Slika 5.3 Primjena modela na snimku web kamere*

Slika na kojoj se prikazuje osoba, muškarac, zid, na zatvorenom

Opis je automatski generiran Slika na kojoj se prikazuje osoba, na zatvorenom, sat, ruka

Opis je automatski generiranSlika na kojoj se prikazuje tekst, šalica, kava, na zatvorenom

Opis je automatski generiran Slika na kojoj se prikazuje tekst, osoba, muškarac

Opis je automatski generiran

*Slika 5.3 Rezultati*

Slika na kojoj se prikazuje osoba, muškarac, zid, na zatvorenom

Opis je automatski generiran Slika na kojoj se prikazuje osoba, muškarac, zid, na zatvorenom

Opis je automatski generiran

*Slika 5.4 Pogrešna detekcija/klasifikacija objekta*

## 5.2 Poboljšanje modela

Iz rezultata prikazanih na slici 5.3 može se primijetiti kako model poprilično dobro detektira i klasificira objekte od interesa ako su oni na slici. Međutim, iz rezultata prikazanih na slici 5.4 vidljivo je da model pogrešno detektira i klasificira ljudske oči kao naočale. Ispostavilo se da je razlog ovakvog ponašanja nepravilno sastavljen skup podataka. Naime, u trening skupu nije bilo slika na kojima su ljudi koji ne nose naočale. Kako bi se smanjila, odnosno uklonila pojava ovakvog *false positive*-a, u trening skup dodano je dvadeset slika na kojima su ljudi koji ne nose naočale te je model ponovno istreniran na novom skupu podataka. Istrenirani model opet je testiran te su rezultati prikazani ispod na slikama. Može se primijetiti kako model više ne klasificira ljudske oči kao naočale.

Slika na kojoj se prikazuje osoba, muškarac, zid, na zatvorenom

Opis je automatski generiran Slika na kojoj se prikazuje osoba, muškarac, zid, na zatvorenom

Opis je automatski generiran

Slika na kojoj se prikazuje osoba, muškarac, zid, na zatvorenom

Opis je automatski generiran Slika na kojoj se prikazuje osoba, muškarac, zid, na zatvorenom

Opis je automatski generiran

*Slika 5.5 Rezultati nakon treniranja modela na*

*poboljšanom skupu podatak*

# 6. Zaključak

Ovim radom objasnile su se osnove rada YOLO algoritma te je prikazano kako se koristi *Ultralyitcs*-ova YOLOv5 implementacija istog. Prikazan je način na koji se sastavlja i označava vlastiti skup podataka korištenjem *LabelImg* aplikacije. Nadalje, objašnjeno je kako se prethodno istrenirani modeli mogu prilagoditi vlastitom skupu podataka transfer učenjem. Zatim je testiran je rad modela istreniranog na vlastitom skupu podataka te su prokomentirani rezultati. Uočene su pogreške u radu modela, objašnjen je razlog njihovog nastajanja i način na koji su se iste uklonile.