SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Računarstvo

ANALIZA DOPRINOSA DUBINSKIH INFORMACIJA U DETEKECIJI OBJEKATA

Seminarski rad
Obrada slike i računalni vid

Domagoj Lešković

Osijek, 2025.

SADRŽAJ

1. UVOD

Detekcija objekata predstavlja jedan od temeljnih izazova u računalnom vidu s sustavima za autonomnu vožnju i nadziranje. Tradicionalni pristupi primarno se oslanjaju na RGB (*engl. Red Green Blue*) slike, izdvajajući značajke za prepoznavanje i lokaliziranje objekata unutar scene. Iako je treniranje modela preko RGB slika najčešće korišteni pristup rješavanja problema detekcije objekata, ova metoda često ima problema kada se suoči sa složenim okruženjima u kojima razine svjetlosti variraju, postoje slični objekti koji nisu isti, ali su istog oblika ili ako su objekti sakriveni iza prepreka (odnosno drugih objekata) [1]. Prema [2], u posljednje vrijeme, sve veću pažnju dobivaju alternativni izvori podataka, poput dubinskih slika koje omogućuju modelima da uzmu u obzir prostorne informacije o udaljenosti objekata od kamere. Kombinacijom RGB i dubinskih podataka (RGBD slike) moguće je dodatno unaprijediti performanse detekcije, osobito u izazovnim uvjetima poput slabog osvjetljenja, preklapanja objekata ili scena s puno vizualnog šuma. Dva najpopularnija i najčešće korištena pristupa za detekciju objekata u suvremenim sustavima su algoritmi *Faster R-CNN* (*engl. Faster Region-based Convolutional Neural Network*) i YOLO (*engl. You Only Look Once,* YOLO) [3]. *Faster R-CNN* predstavlja pristup u dvije faze. Prvo se generira prijedlog regija, a zatim se klasificiraju objekti i precizno se određuju granice njihovih položaja. S druge strane, YOLO je model jednofazne detekcije koji cijelu sliku obrađuje kao jedinstvenu cjelinu i odmah predviđa klase i koordinate okvira za više objekata, što ga čini znatno bržim. U ovom radu primjenjuje se YOLO zbog njegove efikasnosti i mogućnosti postizanja dobrog kompromisa između točnosti i brzine detekcije.

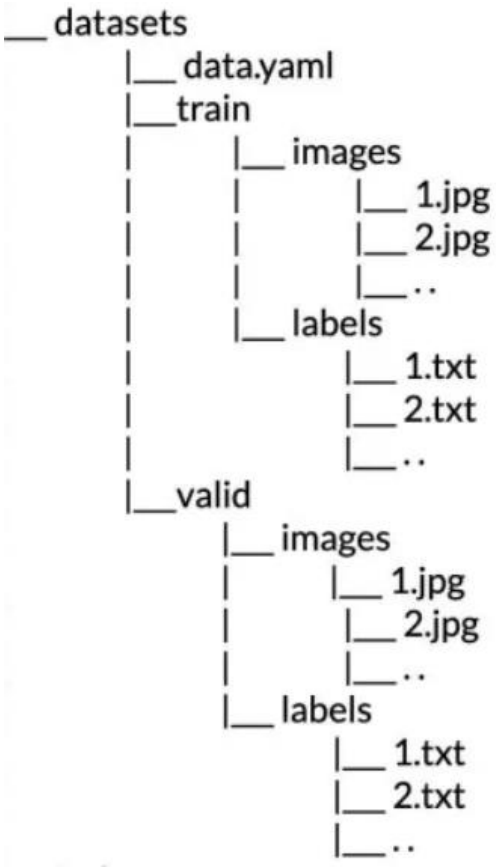
2. PREGLED I PREDOBRADA PODATAKA

Skup podataka korišten u ovom radu je SUN-RGBD, koji sadrži slike unutarnjih prostora snimljene u različitim uvjetima osvjetljenja i konfiguracijama scena, uz pripadajuće dubinske informacije i anotacije objekata. Unutar skupa, dostupno je 10335 slika u obliku RGB i kao jedno kanalne dubinske slike. Prema [6], autori ovog podatkovnog skupa razvili su sustav koji se sastoji od standardne RGB kamere te 4 različita senzora za dobivanje dubinskih informacija. Spajanjem ovih dvaju tipova podataka po kanalu formiraju se četvero kanalne RGBD slike, pri čemu se trima kanalima RGB slike dodaje četvrti kanal koji sadrži dubinske informacije. Na taj način se zadržavaju vizualne značajke scene, dok se istovremeno modelima omogućuje pristup prostornim informacijama. Nadalje, uz skup podataka dolaze i pripadajuće anotacije za svaku sliku. Ti podaci zapisani su u JSON formatu, koji nije izravno prikladan za arhitekturu modela korištenih u ovom seminarskom radu. Kako bi se omogućilo pravilno učenje i evaluacija, potrebno je podatke transformirati u JSON datoteku strukturiranu prema COCO standardu zapisivanja anotacija. Slika 2.1. prikazuje kako izgledaju podaci strukturirani prema COCO standardu. Ovaj primjer je minimalan dozvoljen format zapisa koji obuhvaća samo osnovne podatke. Kao što se sa slike može vidjeti, podaci su organizirani i dalje u JSON obliku, ali na način da se svi podaci nalaze unutar triju glavnih elemenata: *images, annotations* i *categories*.

```json
{
  "images": [
    {
      "id": 1,
      "file_name": "image1.jpg",
      "width": 640,
      "height": 480
    }
  ],
  "annotations": [
    {
      "id": 1,
      "image_id": 1,
      "category_id": 1,
      "bbox": [100, 100, 50, 50],
      "area": 2500,
      "iscrowd": 0
    }
  ],
  "categories": [
    {
      "id": 1,
      "name": "cat"
    }
  ]
}
```

Slika 2.1. COCO format zapisa podataka

Images sadrži imena svih slika unutar skupa podataka kao i njihove dimenzije. *Annotations* predstavlja skup svih objekata na slici, uključujući pripadajuću klasu, koordinate granica i identifikator slike kojoj anotacija pripada. Na kraju, *categories* sadrži popis svih klasa dostupnih unutar skupa podataka, zajedno s njihovim jedinstvenim identifikatorima. Sljedeći korak je pronaći najzastupljenije objekte u slikama. Ovaj korak je posebno važan jer se u ovom skupu podataka nalazi 1068 unikatnih objekata od kojih velik broj nema dovoljno pojavljivanja za kvalitetno treniranje. Zbog toga bi modeli imali znatno lošiju preciznost i ne bi se dobio dobar uvid u stvarne metrike preciznosti. Da bih se spriječilo negativno utjecanje nedovoljno zastupljenih objekata na učenje modela, odabrano je prvih 45 najzastupljenijih objekata. Od svih odabranih objekata, najviše pojavljivanja imaju stolice (24599 pojavljivanja), a najmanje knjige (208 pojavljivanja). Nadalje, slike se moraju podijeliti prema strukturi koju zahtjeva YOLO model. Slika 2.2. prikazuje datotečnu strukturu koja se koristi za podjelu slika na skup za treniranje i skup za testiranje. Kao što se sa slike može vidjeti, slike namijenjene treniranju smještene su u mapu *train/images,* dok se pripadajuće anotacije pohranjuju u mapu *train/labels*. Isti princip primjenjuje se i na testne podatke.



Slika 2.2. Primjer datotečne strukture za treniranje i validaciju YOLO modela, preuzeto iz [4]

Pravilan naziv za mape nije strogo definiran, ali je uobičajeni standard da se mapa za treniranje naziva *train,* dok se mapa za testiranje (odnosno validaciju) naziva *valid* ili skraćeno *val.* Datoteka *data.yaml* sadrži putanje do trening podataka i validacijskih podataka. Također, sadrži i broj klasa (odnosno objekata) i imena tih klasa. Primjer takve datoteke prikazan je na slici 2.3

```yaml
train: C:/Users/Domagoj/Desktop/MYSUN/content_processed/RGB/train/images
val: C:/Users/Domagoj/Desktop/MYSUN/content_processed/RGB/val/images
nc: 45
names:
- chair
- table
- pillow
- desk
- sofa_chair
- cabinet
- lamp
- sofa
- door
- bed
- box
- garbage_bin
- monitor
- shelf
- counter
- computer
- endtable
- sink
- bookshelf
- night_stand
- drawer
- dresser
```

Slika 2.3. Skraćena verzija *data.yaml* datoteke korištene u ovom seminarskom radu za treniranje RGB modela

Budući da se broj legitimnih (odnosno korištenih) klasa smanjio s 1068 na prvih 45 najzastupljenijih, postoji velika vjerojatnost da neke slike neće sadržavati niti jednu od legitimnih klasa. Takve slike su uklonjene iz skupa podataka prije podjele na skup za treniranje i skup za testiranje. Slike su zatim podijeljene u omjeru 80:20, ali kako bi se osiguralo da modeli dobiju dovoljno informacija o svakoj klasi, dodatno je osigurano da u trening i validacijskom skupu bude proporcionalna zastupljenost svih odabranih klasa. Prilikom podjele, za svaku sliku mora se izraditi tekstualna datoteka unutar mape *labels* koja nosi isti naziv kao i sama slika te sadrži podatke o anotaciji objekata unutar slike. Na slici 2.4. prikazan je primjer jedne takve tekstualne datoteke korištene prilikom treniranja modela u ovom seminarskom radu. U prvom stupcu nalazi se identifikator klase, dok preostali stupci predstavljaju normalizirane vrijednosti pozicije objekata na slici, izražene relativno u odnosu na dimenzije slike. Na primjer, iz prvog retka sa slike može se zaključiti da klasa s identifikatorom 9 ima koordinatu x na 34.25% širine slike, koordinatu y na

62.08% visine slike te da objekt zauzima 68.22% širine slike od te točke prema desno kao i 75.85% visine slike od iste točke prema dolje.

content_processed > Depth > train > labels > 00001.txt

```
1   9 0.3424657534246575 0.620754716981132 0.6821917808219178 0.7584905660377359
2   19 0.7678082191780822 0.7556603773584906 0.26986301369863014 0.36792452830188677
3   2 0.553626242726898 0.548494983277592 0.1786721729110981 0.13691218916668235
4   6 0.7994491848156039 0.48388396454472204 0.1023809786628107 0.28444802578565676
5   2 0.3994364777569066 0.5016722408026756 0.13779951436294496 0.1103678929765886
6   2 0.26770742658175656 0.4498327759197324 0.15176157969487333 0.09698996655518398
7   6 0.06252577083428781 0.19481605351170567 0.10319787419251386 0.24581939799331104
8   19 0.1371924680441655 0.41555183946488294 0.1456911165070784 0.09197324414715717
```

Slika. 2.4. Anotacija objekata unutar slike 00001.png

Na slici 2.5. prikazan je jedan primjer označene slike iz obrađenog skupa podataka. Sa slike se

Slika. 2.5. Označena slika iz skupa podataka

može vidjeti kako neki objekti na slici nisu označeni. Razlog tome je prijašnje uklanjanje klasa s malim stupnjem zastupljenosti unutar cijelog skupa podataka.

3. ODABIR MODELA

Tijekom faze odabira modela, razmatrane su dvije vodeće arhitekture za detekciju objekata: YOLO i *Faster R-CNN*. Tradicionalno, YOLO je pružao nešto slabije performanse u usporedbi s *Faster R-CNN-om*, ali je istovremeno bio znatno brži u detekciji objekata. Međutim, zbog značajnih ulaganja u razvoj i unapređenje YOLO arhitekture, već s verzijom YOLOv8 postignute su bolje performanse u odnosu na *Faster R-CNN*, kako u brzini, tako i u točnosti detekcije objekata [5]. Kako je tijekom pisanja ovog seminarskog rada najnovija verzija YOLOv12, upravo je ona i odabrana za daljnju primjenu.

3.1. Ultralytics YOLO

Ultralytics YOLO predstavlja jednu od najpopularnijih i najraširenijih implementacija YOLO arhitekture. Razvijen od strane tvrtke *Ultralytics*, ovaj projekt otvorenog koga poznat je po jednostavnosti korištenja, podršci za najnovije YOLO verzije, te bogatoj dokumentaciji i aktivnoj zajednici. Implementacija nudi više verzija YOLO algoritma koje se razlikuju po složenosti i performansama, prilagođene različitim potrebama korisnika. Dostupne opcije su: n (*nano*), s (*small*), m (*medium*), l (*large*) i x (*extra large*), pri čemu svaka verzija predstavlja kompromis između brzine izvođenja i točnosti detekcije. Na taj način moguće je izgraditi izrazito brze i lagane modele koji su pogodniji za uređaje s ograničenim računalnim resursima, kao i preciznije i zahtjevnije modele prikladne za složenije zadatke. U sklopu ovog seminarskog rada, korištena je *small* verzija YOLO algoritma zbog ograničenih računalnih resursa. Također, moguće je koristiti prijenosno učenje prilikom poziva funkcije za treniranje modela. Ukoliko korisnik postavi argument *pretrained* na *True*, parametri istreniranog modela na COCO skupu podataka se učitaju tijekom treniranja što je preporučeno radi brže konvergencije modela i poboljšanja performansi tokom treniranja.

3.2. Treniranje modela

Slika 3.1. prikazuje postupak pokretanja treniranja modela. Uz prikazane parametre, biblioteka nudi širok raspon argumenata kojima se može promijeniti postupak treniranja modela. Pošto nisu sve slike istih dimenzija, zadnji koraci obrade podataka vrše se putem poziva *train* metode. Kao što se sa slike može vidjeti, svi podaci unutar skupa podataka se transformiraju na dimenzije 640x640 piksela. Interno, biblioteka promjeni prvi sloj u mreži da podržava takve slike. Osim toga, ukoliko bi parametar *augment* bio postavljen na *True*, slike bih se nasumično izmijenile. Izmjene uključuju: rotaciju, translaciju, promjenu nagiba i slične. To u većini slučajeva poboljšava

performanse modela, no u kontekstu ovog seminarskog rada to nije bilo moguće omogućiti. Naime, trenutna najnovija verzija biblioteke ne podržava direktno treniranje modela na slikama s više od tri kanala. Sve operacije augmentacije (osim mozaika, koji se automatski primjenjuje) dizajnirane su isključivo za RGB slike, zbog čega se svi dodatni kanali nakon osnovna tri automatski odbacuju. Radi toga, augmentacija je isključena kako bih se svi modeli mogli uspješno istrenirati na istim parametrima i kasnije međusobno usporedili pod jednakim uvjetima treniranja. Također, ukoliko se *pretrained* parametar stavi na *True,* biblioteka automatski postavi prvi sloj da ima dimenzije 640x640x3, odnosno da prima slike koje imaju 3 kanala, te je zbog toga isključen iako je preporučeno ukoliko se radi model s 3 kanala da ovaj parametar bude postavljen na *True.*

```
# RGB

model = YOLO('./models_yaml/RGB/yolov12s.yaml')

results = model.train(data='./models_data/RGB/data.yaml', epochs=100, imgsz=640, batch=64, augment=False, pretrained = False)
```

Slika. 3.1. Treniranje modela putem Ultralytics YOLO biblioteke

Model treniran na dubinskim slikama prati isti postupak treniranja kao i model treniran na RGB slikama. Međutim, za model treniran na RGBD slikama bilo je potrebno prilagoditi kod biblioteke kako bi se omogućila podrška za slike s nastavkom .tiff. TIFF format omogućuje spremanje slika s većim brojem kanala. Također, bilo je potrebno izmijeniti prvi sloj mreže kako bi podržavao ulazne slike s 4 kanala. Na slici 3.2. prikazana je jedna od funkcija unutar biblioteke koja se morala modificirati da učitava .tiff slike umjesto ostalih formata koji podržavaju 3 kanalne slike. Konkretno, funkcija je odgovorna za dohvat i pripremu pojedinačnih uzoraka iz skupa podataka tijekom treniranja i evaluacije modela. Prilikom svake epohe, tijekom prolaska kroz skup podataka, ova funkcija se poziva za svaki indeks uzorka. Umjesto *cv2.imread()* funkcije korištene u originalnoj verziji, ovaj poziv zamijenjen je s *tiff.imread()*. Ostale izmjene funkcija u kodu biblioteke nisu prikazane s objašnjenima jer prate istu logiku izmjene kao i izmjena funkcije sa slike 3.2. Dodatno, kako bih se promijenio ulazni sloj da podržava slike s 4 kanala, unutar pripadajuće *data.yaml* datoteke korištene za konfiguraciju RGBD modela, potrebno je postaviti parametar *ch* na vrijednost 4. Time se modelu eksplicitno daje do znanja da ulazne slike imaju 4 kanala. Osim tih prilagodbi, pozivom koda prikazanog na slici 3.1. postupak treniranje ostaje identičan kao i za prethodne modele. Treniranje svakog modela provedeno je tijekom 100 epoha, s veličinom *batcha* od 64, koristeći uslugu *Google Colab* i njihovu A100 grafičku karticu. Za *small* arhitekuru YOLO algoritma, između 80. i 90. epohe modeli su postigli najbolje rezultate, nakon čega je uočeno postupno smanjenje preciznosti s nastavkom treniranja u svakoj sljedećoj epohi.

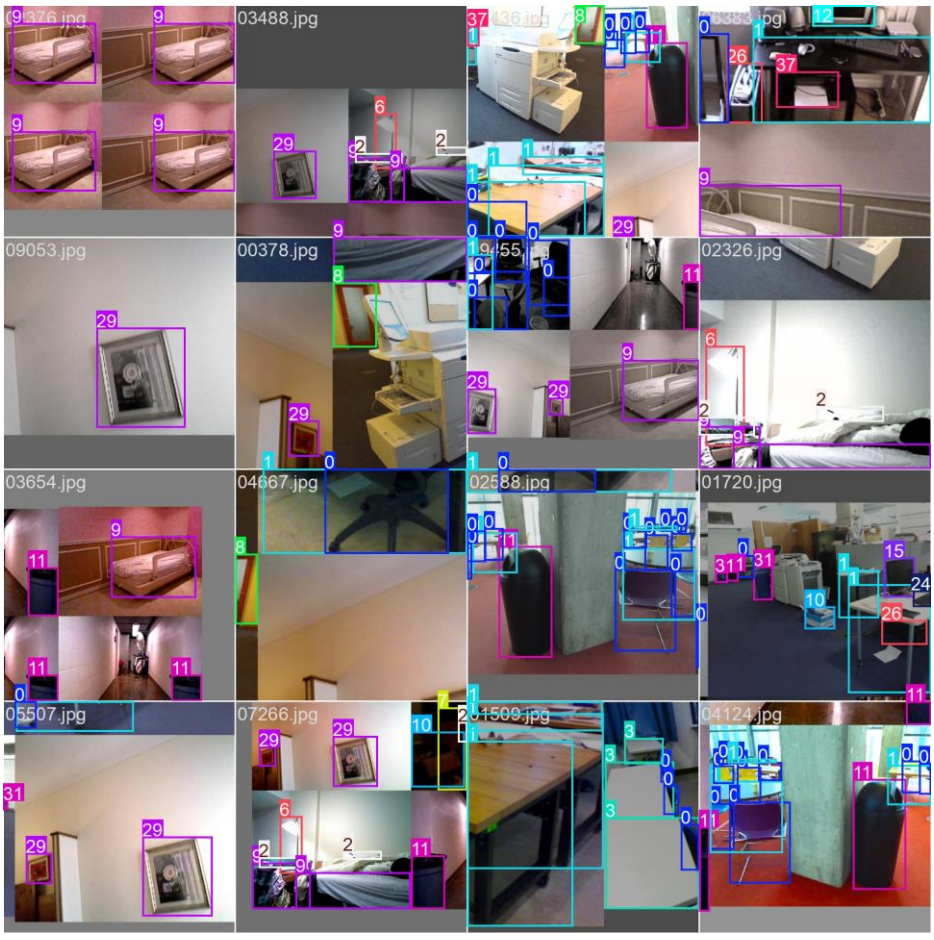
```python
def __getitem__(self, i):
    """
    Returns subset of data and targets corresponding to given indices.

    Args:
        i (int): Index of the sample to retrieve.

    Returns:
        (dict): Dictionary containing the image and its class index.
    """
    f, j, fn, im = self.samples[i]  # filename, index, filename.with_suffix('.npy'), image
    if self.cache_ram:
        if im is None:  # Warning: two separate if statements required here, do not combine this with previous line
            im = self.samples[i][3] = tiff.imread(f)
    elif self.cache_disk:
        if not fn.exists():  # load npy
            np.save(fn.as_posix(), tiff.imread(f), allow_pickle=False)
        im = np.load(fn)
    else:  # read image
        im = tiff.imread(f)  # BGR
    # Convert NumPy array to PIL image
    im = Image.fromarray(cv2.cvtColor(im, cv2.COLOR_BGR2RGB))
    sample = self.torch_transforms(im)
    return {"img": sample, "cls": j}
```

Slika. 3.2. Funkcija za dohvaćanje slika tijekom treniranja i evaluacije

Slika 3.3. prikazuje izgled jednog *batcha*, tijekom treniranja RGB modela. Kao što se sa slike može vidjeti, umjesto da se kroz mrežu šalju slike zasebno, one su kombinirane u mozaik što generalno poboljšava metrike preciznosti modela. *Ultralytics* YOLO implementacija algoritma



Slika. 3.3. Primjer jednog trening batcha za RGB model

automatski pretvara ulazne slike u mozaik te trenira model na tako generiranim slikama. Zadana vrijednost parametra funkcije *train*, imena *close_mosaic*, postavljena je na broj 20, što znači da će se mozaik koristiti tijekom prvih 80% epoha trenirana. Tako je primjerice, prilikom treniranja modela u ovom seminarskom radu, od 81. epohe nadalje model treniran na originalnim ulaznim slikama.

3.3. Evaluacija modela

Za evaluaciju modela, koriste se dvije metrike, IoU (*engl. Intersection over Union*, IoU) i mAP(*engl. mean average precision*, mAP).

3.3.1. IoU

IoU metrika mjeri koliko se predviđeni okvir (*engl. bounding box*) modela preklapa s istinitim, odnosno stvarnim okvirom objekta [7]. Izračunava se kao omjer površine presjeka predviđenog i stvarnog okvira i njihove ukupne površine. Vrijednosti mogu biti između 0 i 1 pri čemu 0 znači da nema preklapanja dok 1 označava savršeno preklapanje. Najčešće se postavlja prag, primjeri 0.5, pri čemu se detekcija smatra ispravnom ako IoU prelazi tu vrijednost. U kontekstu ovog seminarskog rada, IoU metrika nije izravno korištena, već služi kao temelj za izračun mAP metrike kroz različite pragove preklapanja. Obično se za procjenu učinkovitosti modela koristi više pragova IoU kako bi se preciznije procijenila robusnost detekcije.

3.3.2. mAP

Standardna metrika za evaluaciju performansi modela detekcije objekata je mAP. Ona kvantificira koliko je model precizan u detekciji i klasifikaciju objekata na slikama, uzimajući u obzir i točnost lokalizacije odnosno pozicije okvira i ispravnost klase objekta. Prvi korak pri izračunu ove metrike je računanje prosječne preciznosti za svaku klasu (*engl. average precision*, AP). Prosječna preciznost predstavlja površina ispod krivulje preciznost-odziv za tu klasu. Nakon toga, izračunava se aritmetička sredina AP vrijednosti svih klasa u skupu podataka, čime se dobiva mAP metrika. Prilikom izračuna preciznosti i odziva, uobičajeno je koristiti prag od 0.5 ili više kako bi se detekcija smatrala ispravnom. Isto kao i IoU, vrijednosti mAP-a nalaze se u intervalu između 0 i 1. Što je vrijednost bliža 1, to model preciznije prepoznaje i lokalizira objekte te bolje generalizira nove podatke.

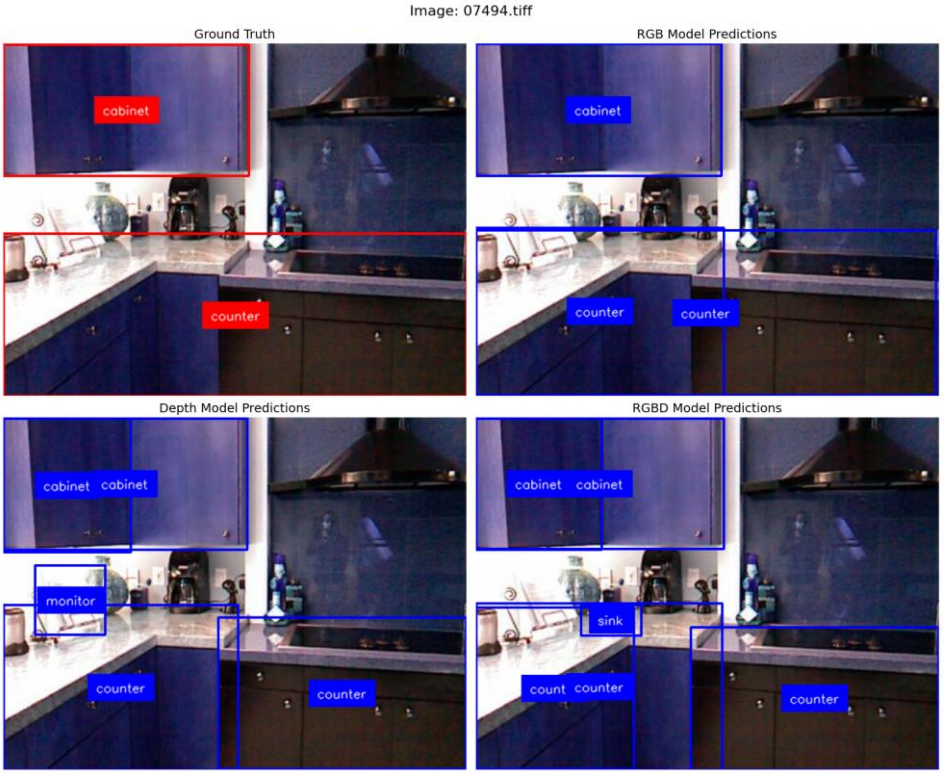
3.3.3. Rezultati treniranja

Nakon treniranja modela kroz 100 epoha, sačuvani su oni s najboljim rezultatima za svaku vrstu ulaznih slika prema mAP metrici. Tablica 3.1. prikazuje završne rezultate treniranja. Modeli su poredani od najnižeg prema najvišem ostvarenome mAP-u. Iz tablice se može vidjeti kako model istreniran na dubinskim informacijama postiže nešto slabije performanse u odnosu na RGB model, koji predstavlja referentnu točku za usporedbu rezultata. Iako model treniran isključivo na dubinskim informacijama u izolaciji ne postiže zadovoljavajuće rezultate, može se primijetiti da

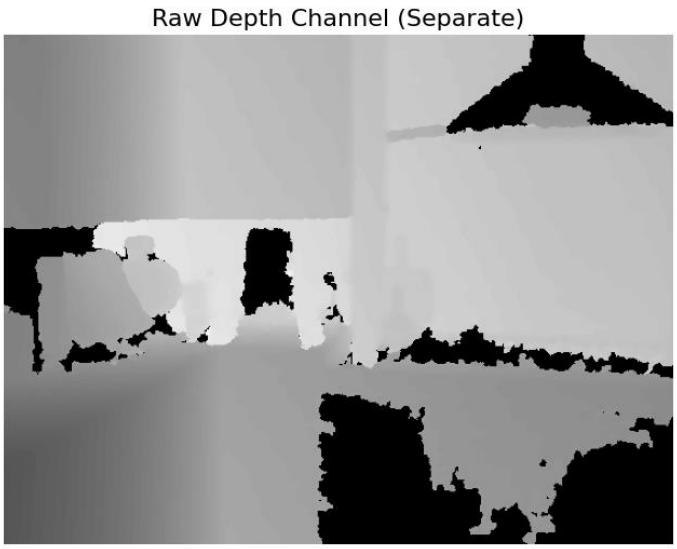
Tablica 3.1. Rezultati treniranja modela

Model	**Preciznost**	**Odziv**	**mAP@50**	**mAP@50-95**
Dubinski	0.496	0.473	0.459	0.327
RGB	0.570	0.476	0.499	0.372
RGBD	0.576	0.525	0.546	0.413

RGBD model ostvaruje bolje rezultate od RGB modela, što ukazuje na to da dodatne dubinske informacije pozitivno utječu na sposobnost modela da bolje i točnije detektira objekte u sceni. To je bila i pretpostavka, budući da dubinske informacije omogućuju modelu bolju percepciju oblika i prostornog rasporeda objekata, čime se olakšava razlikovanje pojedinih klasa i smanjuje broj pogrešnih detekcija. Prilikom istraživanja utjecaja dubinskih informacija na točnost detekcije objekata na slikama, autori članaka [8] i [9] dolaze do zaključka da su dubinske informacije zanemarive prirode ili da pomažu u slučajevima preklapanja objekata, lošeg osvjetljenja u sceni te kada su objekti sličnih boja. Iako postoji korelacija između tih pojmova i točnosti RGBD modela, rezultati ovog seminarskog rada sugeriraju da to ne mora biti uvijek točno. Ukoliko je osvjetljenje u sceni optimalno, ali dubinska informacija sadrži značajnu količinu šuma, RGBD model može postići lošije rezultate od modela treniranog isključivo na RGB slikama kao što je prikazano na slici 3.4. Nadalje, slika 3.5. prikazuje kako izgleda kanal RGBD slike koji sadrži dubinske informacije. Kao što se sa slike može vidjeti, prisutno je puno crnih dijelova unutar scene. Ovi crni pikseli označavaju područja u kojima dubinski senzor nije uspio izmjeriti ili aproksimirati udaljenost između kamere i objekta. Takve „rupe“ u dubinskim podacima najčešće nastaju zbog reflektirajućih, prozirnih ili vro tamnih površina, kao i zbog položaja objekata izvan efektivnog dometa senzora. Siva područja na slici predstavljaju važeće dubinske vrijednosti, odnosno piksele gdje je senzor uspješno izmjerio udaljenost do objekta. Različite nijanse sive boje odgovaraju različitim udaljenostima.

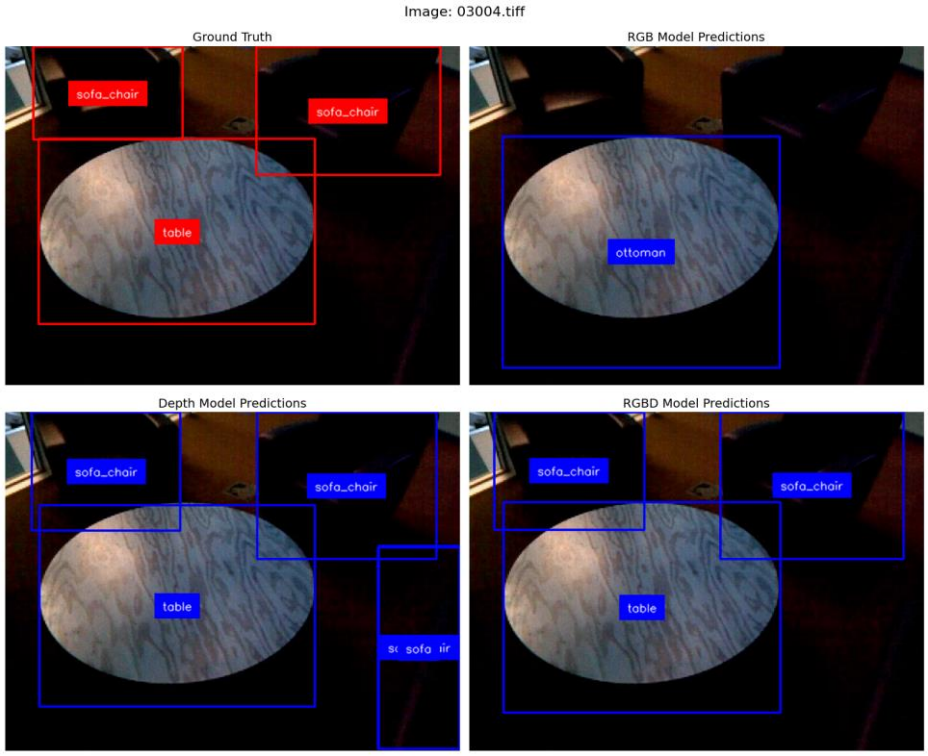


Slika. 3.4. Stvarna anotacija i predikcije ostalih modela

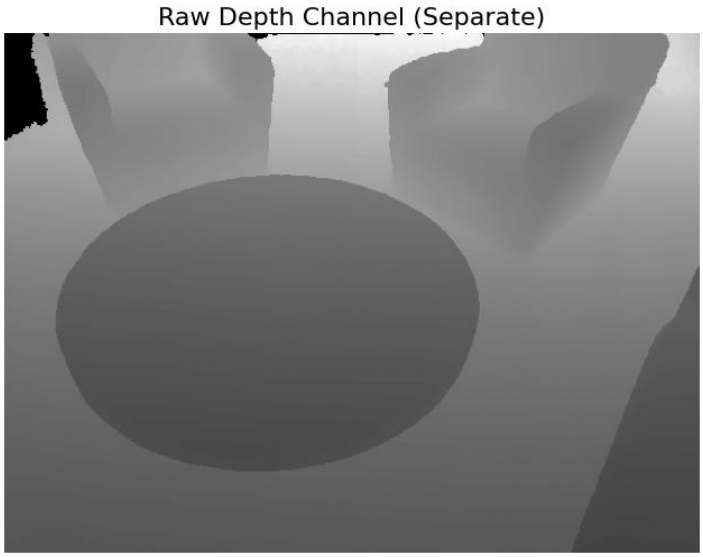


Slika. 3.5. Dubinska informacija za sliku 3.4.

Na slici 3.6. prikazan je scenarij u kojemu je RGBD model precizno odredio okvir i klasifikaciju objekata na sceni dok RGB model nije. Za razliku od prijašnjeg primjera, kao što se može vidjeti sa slike 3.7., dubinska informacija je zapisana bez značajnih distorzija u slici. Osim toga, prostorija na slici snimljena je u tamnijem okruženju, gdje RBG model u većini slučajeva nema dovoljno informacija za točno predviđanje objekata u sceni.



Slika. 3.6. Stvarna anotacija i predikcije ostalih modela



Slika. 3.7. Dubinska informacija za sliku 3.6.

3.3.4. Usporedba rezultata s ostalim radovima

Tijekom pisanja ovog seminarskog rada, prema [10] postoje sedam znanstvenih članaka baziranih na detekciji i lokalizaciji objekata na SUN-RGBD skupu podataka. Autori u članku [11], postigli su mAP od 58.4% za IoU prag od 0.5 koristeći model treniran na RGBD slikama. Model je treniran na prvih 10 najzastupljenijih klasa unutar skupa podataka. U usporedbi s najboljim modelom razvijenim u okviru ovog seminarskog rada, to predstavlja povećanje od 3.8% u odnosu na ostvareni rezultat, iako pritom treba uzeti u obzir manji broj klasa na kojima je taj model bio treniran u usporedbi s modelom razvijenim u okviru ovog rada. Zanimljivo je također usporediti rezultate iz rada [13] gdje su također korištene RGBD slike za detekciju i lokalizaciju objekata na slici, ali nije korišten YOLO algoritam nego u to vrijeme najpopularniji, R-CNN. Ovaj rad je objavljen 2014. godine, godinu dana prije pojave novije i učinkovitije implementacije, *Faster R-CNN*. U navedenom istraživanju, autori su postigli mAP od 44.2% koristeći RGBD slike. Za usporedbu, u ovom radu je ostvaren rezultat od 49.9% mAP za model treniran na RGB slikama, što predstavlja povećanje od 5.7%. U odnosu na izravno usporediv model treniran na RGBD slikama, u ovom radu postignut je rezultat od 54.6% na relativno ne zahtjevnoj verziji YOLO arhitekture koja uz bolju preciznost treba i manje vremena za detekciju i lokalizaciju. Ovaj rezultat dodatno naglašava napredak u točnosti detekcije objekata kroz godine, unatoč korištenju modela koji nije među najkompleksnijim unutar YOLO obitelji.

ZAKLJUČAK

U ovom radu, istrenirala su se tri modela za detekciju i lokalizaciju objekata unutar scene. Prvi model treniran je isključivo na dubinskim informacijama, odnosno slikama s jednim kanalom, drugi model treniran je na RGB slikama, a treći model treniran je na kombinaciji prijašnja dva, odnosno na RGBD slikama. Cilj rada bio je istražiti utjecaj dubinskih informacija na preciznost detekcije i lokalizacije objekata u sceni. Kao što se iz prikazanih rezultata može zaključiti, RGBD model se pokazao najboljim prema vrijednostima mAP metrike, što upućuje na zaključak da su dubinske informacije u ovom kontekstu bile ključne za bolju detekciju i lokalizaciju objekata. Međutim, važno je naglasiti kako postoje situacije u kojima dubinske informacije ne doprinose značajnom poboljšanju performansi modela. U slučajevima kada je kvaliteta dubinskog kanala narušena šumom ili artefaktima, rezultati mogu biti podjednaki, a u nekim slučajevima čak i lošiji nego kod modela treniranih samo na RGB slikama. Unatoč tim ograničenjima, provedeni eksperimenti pokazuju da integracija dubinskih podataka u većini slučajeva pozitivno utječe na performanse detekcije, osobito u zahtjevnim uvjetima kao što su loše osvjetljenje ili preklapanje objekata. RGBD model istreniran u ovom radu postigao je zadovoljavajuće rezultate u usporedbi sa trenutno dostupnim znanstvenim radovima i implementacijama temeljenima na skupu podataka SUN-RGBD. Za poboljšanje performansi predikcija bilo bi korisno sačekati da autori biblioteke implementiraju dodatnu podršku za treniranje modela na više kanalnim slikama ili pristupiti manualnom proučavanju izvornog koda biblioteke i izmijeniti postojeće funkcionalnosti. Osim toga, ukoliko brzina detekcije nije ključna, treniranje modela na složenijoj arhitekturi moglo bi dodatno poboljšati točnost detekcije i lokalizacije objekata.

LITERATURA

[1] S. Hou, Z. Wang, F. Wu, Deeply exploit depth information for object detection, arXiv prepring, svibanj 2016., dostupno na: https://arxiv.org/pdf/1605.02260

[2] H. Zeng, B. Yang, X. Wang, J. Liu, D. Fu, RGB-D object recognition using multi-modal deep neural network and DS evidence theory, Sensors (Basel), sv. 19, br. 3, str. 529, siječanj 2019.

[3] G. Boesch, Object: The Definitive Guide, viso.ai [online], dostupno na: https://viso.ai/deep-learning/object-detection/

[4] V. Vignesh, YOLOv8: Efficient and Accurate Object Detection Made Easy, Medium [online], dostupno na: https://levelup.gitconnected.com/yolov8-efficient-and-accurate-object-detection-made-easy-648a5debc704

[5] Keylabs.ai, YOLOv8 vs Faster R-CNN: A Comparative Analysis, Keylabs [online], dostupno na: https://keylabs.ai/blog/yolov8-vs-faster-r-cnn-a-comparative-analysis/

[6] S. Song, S. P. Lichtenberg, J. Xiao, SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite, Princeton University, 2015.

[7] D. Shah, Intersection over Union (IoU): Definition Calculation, Code, v7labs [online], dostupno na: https://www.v7labs.com/blog/intersection-over-union-guide

[8] E. Orfaig, I. Stainvas, I. Bilik, Enhanced automotive object detection via RGB-D fusion in a DiffusionDet framework, airXiv prepring, lipanj 2024.

[9] M. Bartolo, D. Seychell, Correlation of object detection performance with visual saliency and depth estimation, arXiv preprint, studeni 2024.

[10] Paperswithcode.com, Object Detection In Indoor Scenes on SUN RGB-D, paperswithcode [online], dostupno na: https://paperswithcode.com/sota/object-detection-in-indoor-scenes-on-sun-rgb

[11] X. Shen, I. Stamos, Unified object detector for different modalities based on vision transformers, arXiv preprint, svibanj 2023.

[12] M. Isaksson, Automatic primitvie detection in point clouds using machine learning, arXiv preprint, srpanj 2014.