Primjena modela neuronske mreže na FSOCO skupu slika

Sitar, Nikolina

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Nikolina Sitar

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Nikolina Sitar

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru Tomislavu Stipančiću na iskazanom razumijevanju i strpljenu te pomoći koju mi je pružio tijekom izrade završnog rada.

Zahvaljujem se majci, sestri i prijateljima na potpori tijekom studija. Također bih se zahvalila i kolegama iz FSB Racing Teama, a posebno članovima Driverless podtima.

Nikolina Sitar

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove.
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Nikolina Sitar** JMBAG: **0035218620**

Naslov rada na hrvatskom jeziku: **Primjena modela neuronske mreža na FSOCO skupu slika**

Naslov rada na engleskom jeziku: **Application of neural network models to the FSOCO image set**

Opis zadatka:

Metode strojnog učenja moguće je koristiti prilikom rješavanja problema iz različitih domena ljudskog djelovanja. Neuronske mreže kao dio metoda dubokog učenja su posebno prikladne u primjenama gdje postoji velika količina strukturiranih i nestrukturiranih podataka, kod računalnog prepoznavanja, praćenja, izdvajanja, itd.

U sklopu rada potrebno je:

- proučiti metodologiju konvolucijskih neuronskih mreža te odabrati odgovarajući model ranije trenirane mreže,
- koristeći prikladnu evaluacijsku metriku istražiti rezultate predviđanja odabranog modela na FSOCO skupu koji sadrži slike čunjeva na automobilističkim stazama,
- odabrani model nadopuniti tako da ga se ponovo trenira na FSOCO skupu slika,
- ponoviti postupak evaluacije rezultata na nadopunjenom modelu te usporediti rezultate,
- dati kritički osvrt na mogućnost korištenja predložene metodologije u sklopu modela koji ostvaruju autonomnu vožnju vozila.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:
30. 11. 2021.

Zadatak zadao:
Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 24. 2. 2022.
2. rok (izvanredni): 6. 7. 2022.
3. rok: 22. 9. 2022.

Predviđeni datumi obrane:
1. rok: 28. 2. – 4. 3. 2022.
2. rok (izvanredni): 8. 7. 2022.
3. rok: 26. 9. – 30. 9. 2022.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

SAŽETAK

U ovom radu dan je teorijski pregled strojnog učenja, konvolucijske neuronske mreže te pojmova bitnih za razumijevanje same evaluacije dobivenih rezultata. Nakon prikladnog uvođenja pojmova, rad se nastavlja s praktičnim dijelom zadatka koji kreće s odabirom prikladnog modela postojeće mreže. Nastavno na to, odabrani se model trenira na FSOCO skupu podataka. Navedeni skup podataka je jedini prikladan za ovaj zadatak zbog specifičnosti objekata koje je potrebno detektirati i klasificirati. Na kraju se provodi evaluacija dobivenih rezultata u kontekstu buduće primjene treniranog modela, s fokusom na vrijednosti srednje prosječne preciznosti (mAP).

Ključne riječi: strojno učenje, konvolucijska neuronska mreža, FSOCO, detekcija, klasifikacija, srednja prosječna preciznost.

SUMMARY

In this paper, theoretical background is given regarding the machine learning, convolutional neural networks and main expressions crucial for understanding the evaluation of the results. After appropriatelly introducing key expressions, paper is continued with practical part which starts with choosing the right model of existing network. Afterwards, chosen model is being trained on FSOCO dataset. Mentioned dataset is the only appropriate for this problem due to specific object that need to be detected and classified. At the end, evaluation of the results is made keeping in mind future use of trained model, with focuse on mean average precision (mAP).

Key words: machine learning, convolutional neural networks, FSOCO, detection, classification, mean average precision.

1. UVOD

Pod okriljem udruge Hrvatska studentska asocijacija strojarskih fakulteta krajem 2003. godine osnovan je projekt FSB Racing Team. Tim je do sad izradio sedam bolida s motorom na unutarnje izgaranje a zadnja dva bolida su s električnim pogonom. Koristeći zadnji električni bolid, VulpesR, cilj tima je izraditi prvi autonomni bolid u Hrvatskoj (VulpesD).

Za realizaciju autonomne vožnje bolida potrebno je, među ostalim, razviti i vizijski sustav koji će služiti za detekciju i klasifikaciju čunjeva kojima je označena staza po kojoj se bolid kreće. Korišteni čunjevi definirani su pravilnikom natjecanja te se iz tog razloga kao skup podataka koristi FSOCO s obzirom da on sadrži slike propisanih čunjeva. Postoje 4 vrste propisanih čunjeva, a to su: plavi čunj, žuti čunj, narančasti čunj i veliki narančasti čunj. S obzirom na nabrojane vrste, u daljnjem će radu klase dobiti prikladne nazive.

U svrhu ostvarivanja navedenog ovaj će se rad pozabaviti metodologijom konvolucijskih neuronskih mreža, odabirom prikladnog modela postojeće mreže kao i njegovom nadopunom treniranjem na FSOCO skupu podataka kao i evaluacijom dobivenih rezultata.

2. TEORIJSKA OSNOVA RADA

2.1. Strojno učenje

Strojno učenje je grana umjetne inteligencije koja naglasak stavlja na korištenje podataka i algoritama kako bi se imitirao način na koji uče ljudi, postepeno poboljšavajući točnost.

Postupak učenja se može podijeliti u tri glavna koraka:

- Postupak donošenja odluka – na temelju ulaznih podataka, algoritam donosi procjenu o uzorku u skupu podataka.
- Funkcija greške – funkcija greške služi za evaluaciju predikcije modela. Uz poznate primjere, funkcija greške radi usporedbu kako bi procijenila točnost modela.
- Postupak optimizacije modela – postupak učenja se ponavlja sve dok se usporedbom dobivenih rezultata i željenih rezultata ne dobije zadovoljavajuća točnost. U procesu ponavljanja postupka učenja, bolji rezultati se postižu korekcijom vrijednosti težina.

Strojno učenje se može podijeliti u tri kategorije:

- Nadgledano učenje – karakterizira ga upotreba označenih skupova podataka za treniranje algoritama za dvije grupe problema. Prva grupa se odnosi na slučaj klasifikacije kada je izlaznu varijablu moguće svrstati u određenu kategoriju, dok se druga grupa odnosi na predikciju izlazne varijable. Za procjenu točnosti algoritma koristi se funkcija gubitka koja se prilagođava sve dok se greška ne smanji do prihvatljive granice.
- Nenadgledano učenje – karakterizira ga upotreba neoznačenih skupova podataka. Koristi se za dvije kategorije problema. Prva kategorija je grupiranje, pri kojem je cilj algoritma napraviti grupiranje na temelju neke zajedničke karakteristike. Druga je kategorija problema otkrivanje nekog pravila u skupu podataka koje omogućuje povezivanje informacija.
- Pojačano učenje - koristi sustav nagrade i kazne. Osnovna je razlika naspram nadgledanog učenja ta što kod pojačanog učenja nisu potrebni označeni parovi ulazno izlaznih podataka.

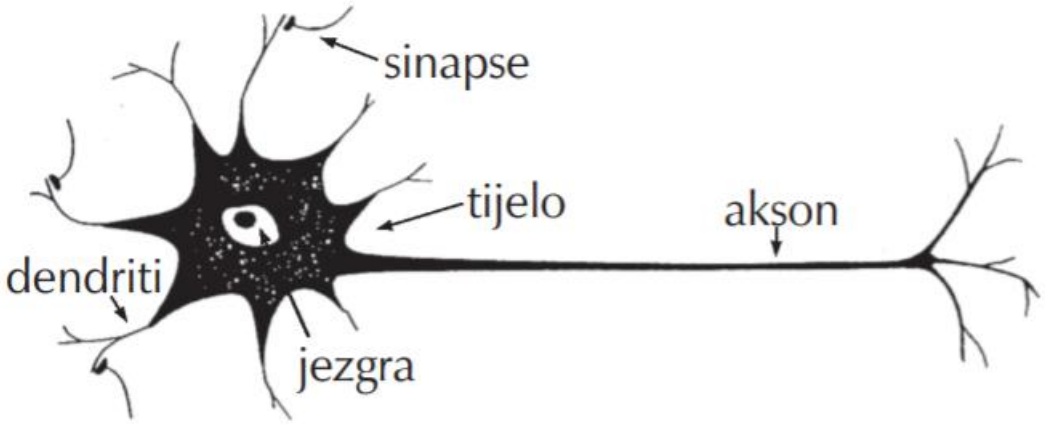
2.2. Duboko učenje

Duboko učenje, također poznato i kao duboka neuronska mreža, je grana strojnog učenja. Umjetne neuronske mreže se temelje na umjetnim neuronima koji imitiraju funkcije ljudskog mozga. Neuroni mogu biti složeni u različite konfiguracije, najčešće u nekoliko slojeva. Ukoliko se mreža sastoji od velikog broja neurona i/ili mnogo slojeva, onda se naziva dubokom neuronskom mrežom.

S obzirom na spomenutu činjenicu da je duboko učenje grana strojnog učenja, važno je naglasiti po čemu je ono specifično. Za razliku od klasičnog strojnog učenja, duboko učenje smanjuje potrebu za sudjelovanjem čovjeka nakon inicijalnog postavljanja. No kao negativna strana veće kompleksnosti dubokog učenja naspram strojnog, postavljaju se veći zahtjevi za korišteni hardver.

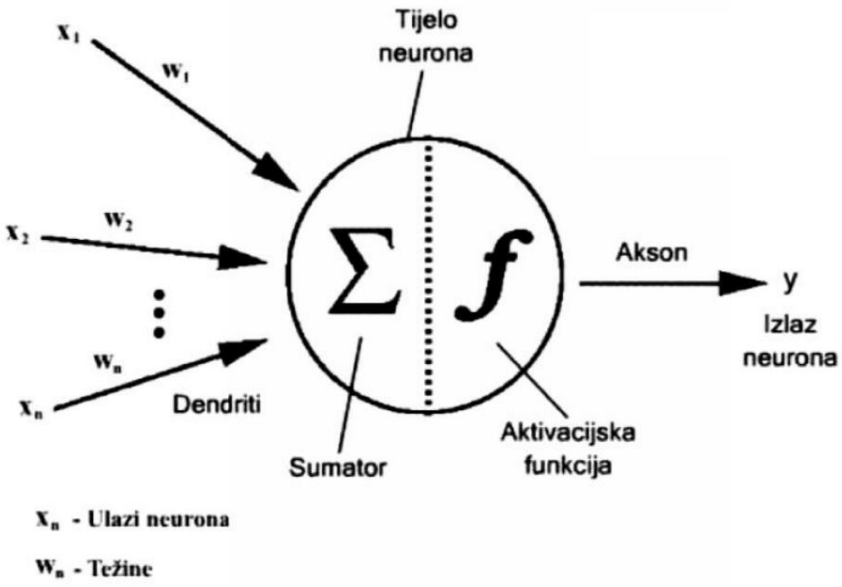
2.3. Građa umjetnog neurona

Umjetne neuronske mreže, kako je prethodno navedeno, imitiraju funkciju ljudskog mozga. Kako bi se to postiglo, njihova je građa temeljena na građi ljudskog mozga točnije na građi bioloških neurona. Svaki biološki neuron sastoji se od tijela, dendrita, aksona, sinapse i jezgre. Njegova je građa predočena na sljedećoj slici:



Slika 1. Građa neurona

Po uzoru na biološki neuron, građa umjetnog neurona prikaza je sljedećom slikom:

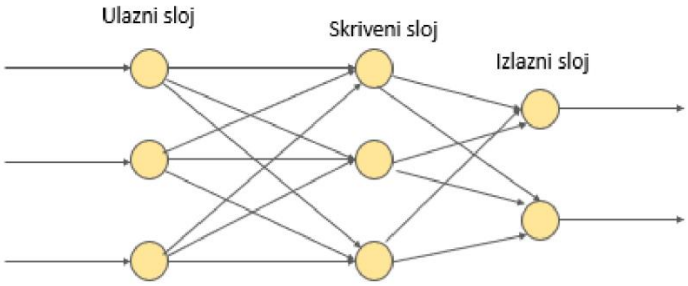


Slika 2. Građa umjetnog neurona

Njega ukratko možemo opisati na sljedeći način: umjetni neuron od susjednih neurona prima ulazne signale koje zatim pretvara u izlazne signale i šalje ih dalje na sljedeći neuron. No za razumijevanje same pretvorbe ulaznih u izlazne signale, važno je znati da svaki spoj između dva neurona ima svoju težinsku vrijednost. Te težinske vrijednosti su parametri koji se množe s vrijednostima ulaznih varijabli. Sumator zbraja sve umnoške ulaznih i težinskih vrijednosti. Primjenom aktivacijske funkcije na vrijednost dobivenu iz sumatora dobiva se vrijednost izlaznog signala.

2.4. Arhitektura neuronske mreže

Prethodno opisani umjetni neuron je osnovna građevna jedinica umjetne neuronske mreže. Variranjem broja i rasporeda umjetnih neurona postiže se željena arhitektura mreže. Ukoliko se ograničimo na skup neurona na istoj dubini mreže, možemo govoriti o jednom sloju neurona a te slojeve možemo podijeliti u tri kategorije: ulazne, skrivene i izlazne.

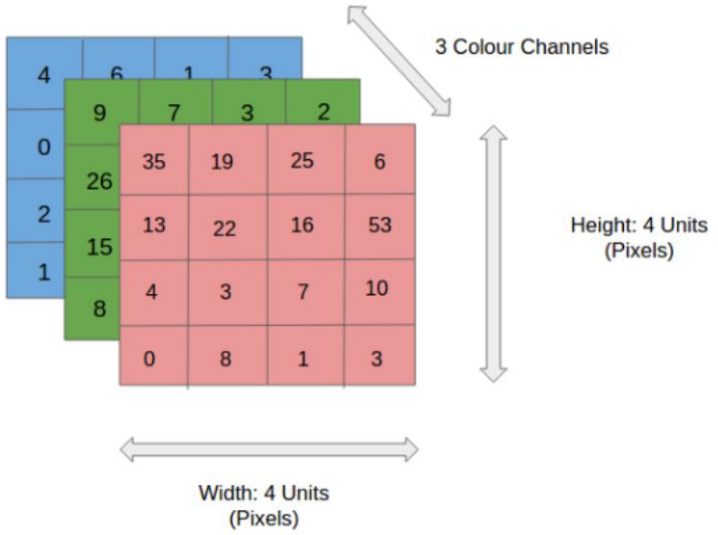


Slika 3. Arhitektura neuronske mreže

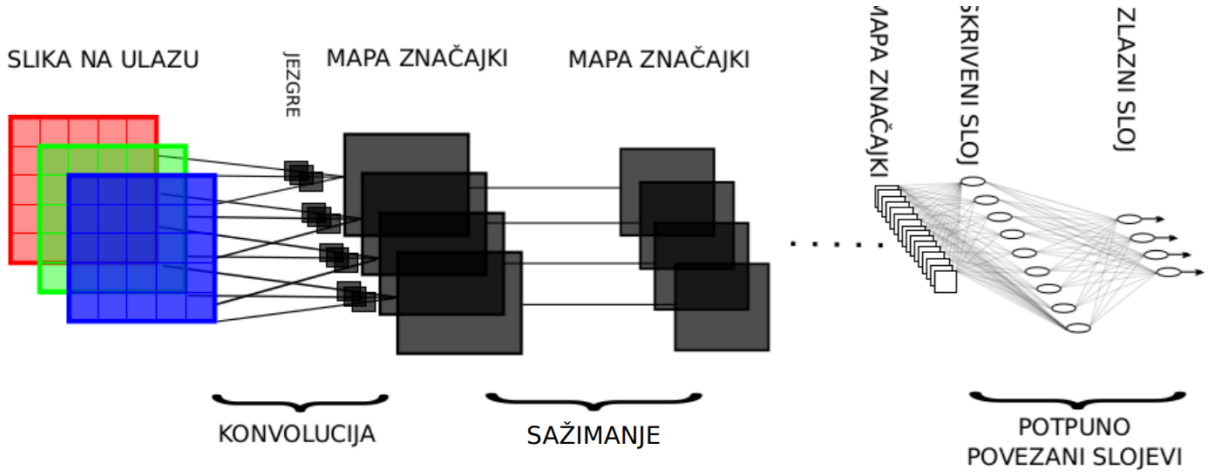
2.5. Konvolucijska neuronska mreža (CNN)

Konvolucijska neuronska mreža se oslanja na činjenicu da je ulaz u mrežu slika, točnije, volumen slike. S time na umu, u arhitekturu mreže implementiraju se specifične karakteristike čime se smanjuje broj parametara potrebnih u mreži. Uloga CNN je da reducira slike u formu koja je lakši za procesiranje, a u isto vrijeme da se ne izgube karakteristike koje su kritične za dobivanje dobre preciznosti. Prednost CNN je u tome što je pretprocesiranje koje ona zahtjeva puno manje od pretprocesiranja potrebnog za ostale klasifikacijske algoritme. Također joj u prilog ide i to što uz dovoljno treniranje ima mogućnost naučiti filtere/karakteristike umjesto da se oni ručno unose. Filter je kvadratna matrica koja prolazi kroz sliku u koracima te detektira razne objekte.

Arhitektura konvolucijske neuronske mreže sastoji se od jednog ulaznog, jednog izlaznog te jednog ili više skrivenih slojeva a neuroni sloja CNN su posloženi u tri dimenzije: širina, visina i dubina. Dimenzija dubine odgovara kanalima boje RGB (crvena, zelena, plava). Ono što je karakteristično za CNN je da se u skrivenom sloju javljaju specifični slojevi od kojih su najčešći konvolucijski slojevi i slojevi sažimanja, a također se koriste i potpuno povezani slojevi. Različitim načinom slaganja prethodno navedenih slojeva, dobivaju se razni modeli te će se u daljnjem radu izabrati jedan od pretreniranih modela.



Slika 4. Primjer slike na ulazu



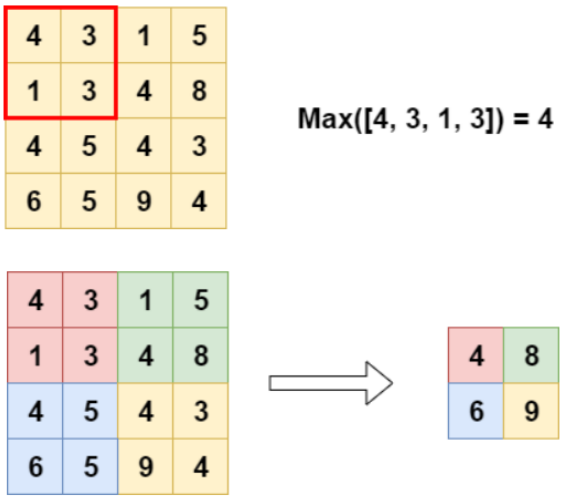
Slika 5. Klasična struktura konvolucijskog modela

2.5.1. Konvolucijski sloj

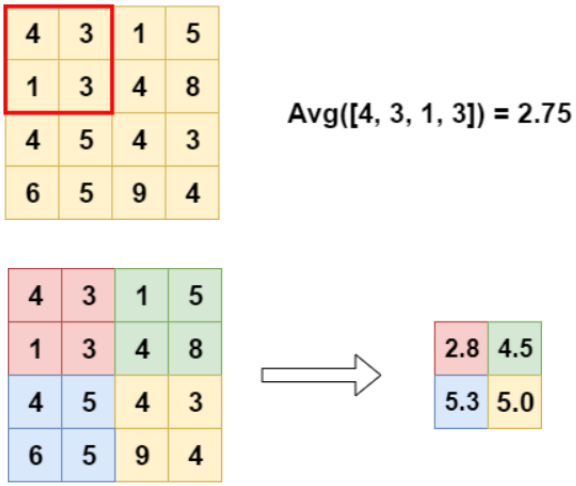
Matematička operacija dviju funkcija naziva se konvolucijom te se njome dobiva treća funkcija koja opisuje ovisnost jedne funkcije o obliku druge. Za provedbu konvolucije su zaduženi prethodno navedeni filteri. Filteri u konvolucijskom sloju se sastoje od težina koje se uče prilikom samog treniranja mreže.

2.5.2. Sloj sažimanja

Ovaj sloj također sadrži filtere a sam naziv sloja indicira njihovu funkciju. Oni služe za sažimanje dimenzija slike. Unutar odabrane okoline bira se maksimalna vrijednost u slučaju korištenja sažimanja maksimalnih vrijednosti ili prosječna vrijednost u slučaju korištenja prosječnog sažimanja.



Slika 6. Primjer sažimanja maksimalnih vrijednosti

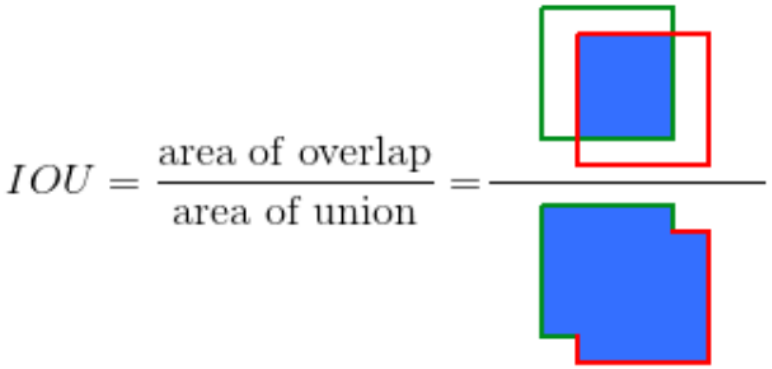


Slika 7. Primjer prosječnog sažimanja

2.6. Evaluacija rezultata

2.6.1. Križanje preko unije (Intersection over Union - IoU)

Radi se o široko primjenjivanoj metodi mjerenja točnosti lokalizacije i računanja greške lokalizacije za modele detekcije objekata. Za vrijednosti IoU koristimo granične kutije dobivene korištenim modelom i istinite granične kutije. IoU je definiran kao omjer površine preklapanja i površine unije tih graničnih kutija kao što je prikazano na sljedećoj slici:



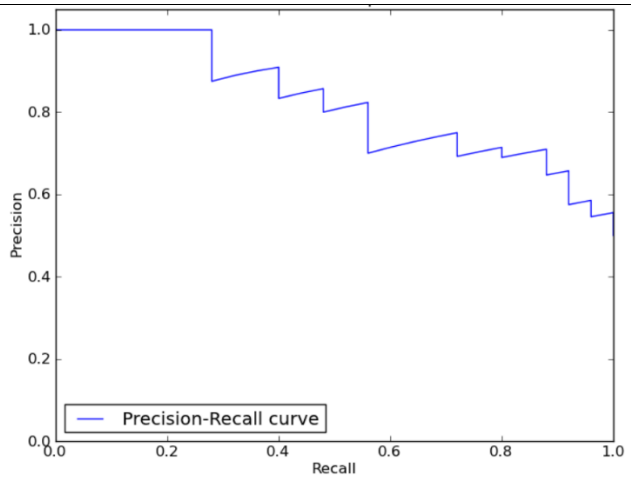
Slika 8. Slikovni prikaz IoU

Za veći iznos IoU predviđanje je bolje.

2.6.2. Srednja prosječna preciznost (mAP)

Za razumijevanje srednje prosječne preciznosti, prvo je potrebno objasniti dva pojma, a to su: krivulja preciznost-odziv i prosječna preciznost.

Krivulja preciznost-odziv prikazuje odnos preciznosti i odziva za različite vrijednosti praga. Ona služi za odabir vrijednosti praga pri kojem dobivamo optimalan odnos preciznosti i odziva. Visoka preciznost je vezana uz nisku stopu lažno pozitivnih detekcija dok je visoka vrijednost odziva vezana uz nisku stopu lažno negativnih detekcija. Na sljedećoj slici možemo vidjeti primjer jedne takve krivulje:



Slika 9. Krivulja odnosa preciznosti i odziva

Odziv se računa kao odnos svih detekcija napravljenih od strane modela za određenu klasu naspram svih postojećih oznaka za tu klasu dok se preciznost računa kao odnos točnih pozitivnih detekcija naspram ukupnog broja detekcija napravljenih od strane modela.

Prosječna preciznost računa se kao površina ispod prethodno navedene krivulje te se odnosi na pojedinu klasu a usrednjavanjem te vrijednosti za sve klase dobivamo srednju prosječnu preciznost.

3. IZVEDBA ZADATKA

3.1. Odabir modela

Odabrani model konvolucijske neuronske mreže je YOLOv4. Razlog za odabir navedenog modela je mogućnost dobivanja zadovoljavajuće točnosti detekcije objekata uz istovremeno ostvarivanje optimalne brzine izvođenja. Optimalnu brzinu izvođenja omogućuje predviđanje vjerojatnosti klasa i lokalizaciju, tj. pozicije graničnih kutija (engl. Bounding Box) u isto vrijeme. U usporedbi s drugim mrežama, za YOLOv4 je dovoljna samo jedna iteracija za jednu sliku po čemu je model i dobio ime (YOLO – You Only Look Once). Najveće ograničenje za YOLOv4 je detekcija malih objekata. Za VulpesD, autonomni bolid na kojem će se primjenjivati navedena mreža, to neće predstavljati problem s obzirom da će se vršiti detekcija čunjeva koji se nalaze bliže samom bolidu te se samim time ne radi o malim objektima. Za detekciju čunjeva na većoj udaljenosti koristit će se VLP-16 Puck Hi-Res LiDAR.

3.2. Rezultati predviđanja odabranog modela na FSOCO skupu

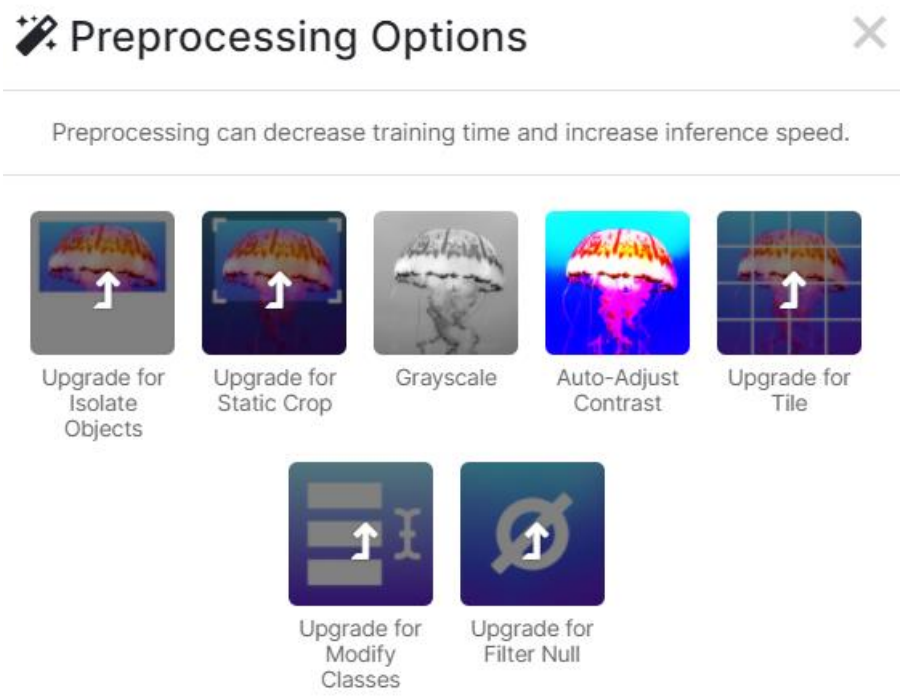
S obzirom na specifične klasa koje je potrebno detektirati, nije moguće evaluirati predviđanja predtreniranog odabranog modela na FSOCO skupu slika.

Iz tog razloga odmah se kreće na nadopunu odabranog modela. Model je prvo nadopunjen korištenjem web platforme pod nazivom Roboflow a nezadovoljavajući rezultati dobiveni njenom primjenom ukazali su na nužnost primjene nekog drugog načina treniranja mreže.

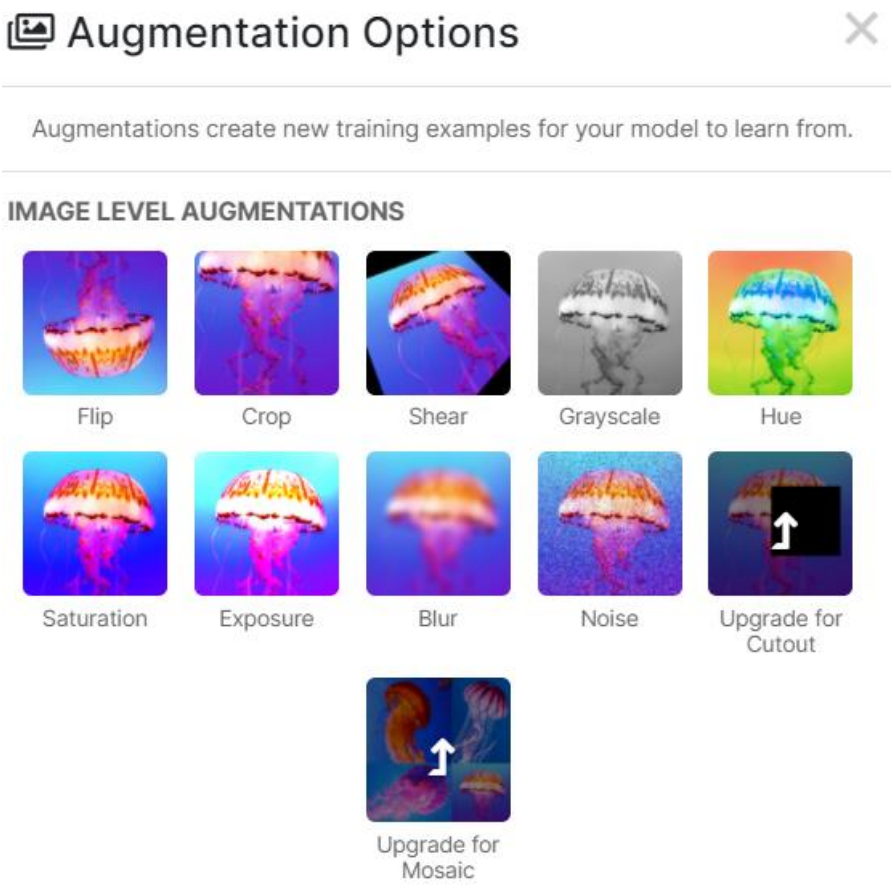
3.3. Nadopuna modela treniranjem na FSOCO skupu slika

3.3.1. Roboflow

Roboflow je web platforma koja ima širok spektar mogućnosti, te je iz tog razloga odabrana za prvi pokušaj nadopune modela. Jedna od njezinih mogućnosti je odabir različitih metoda pretprocesiranja slika (svođenje slika različitih dimenzija na zadanu veličinu, automatsko orijentiranje, ...) kao i različitih metoda augmentacije (rotiranje slika, promjena svjetline, ...)



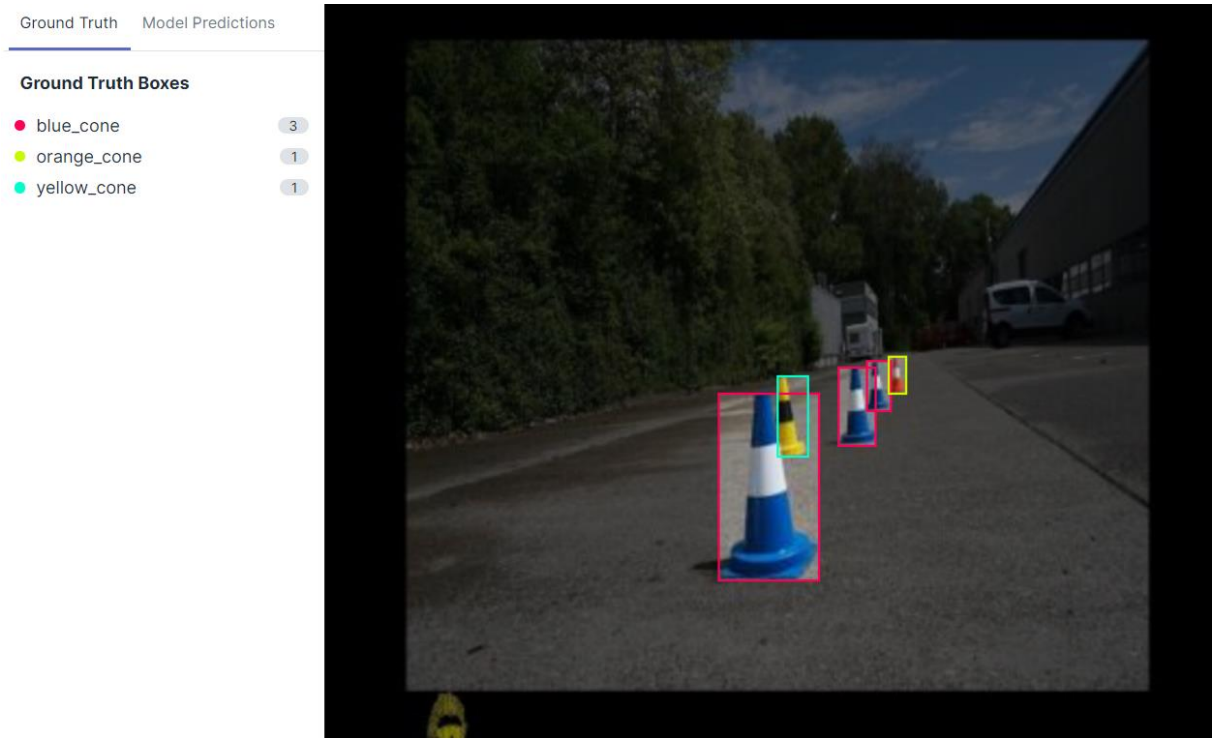
Slika 10. Mogućnosti pretprocesiranja



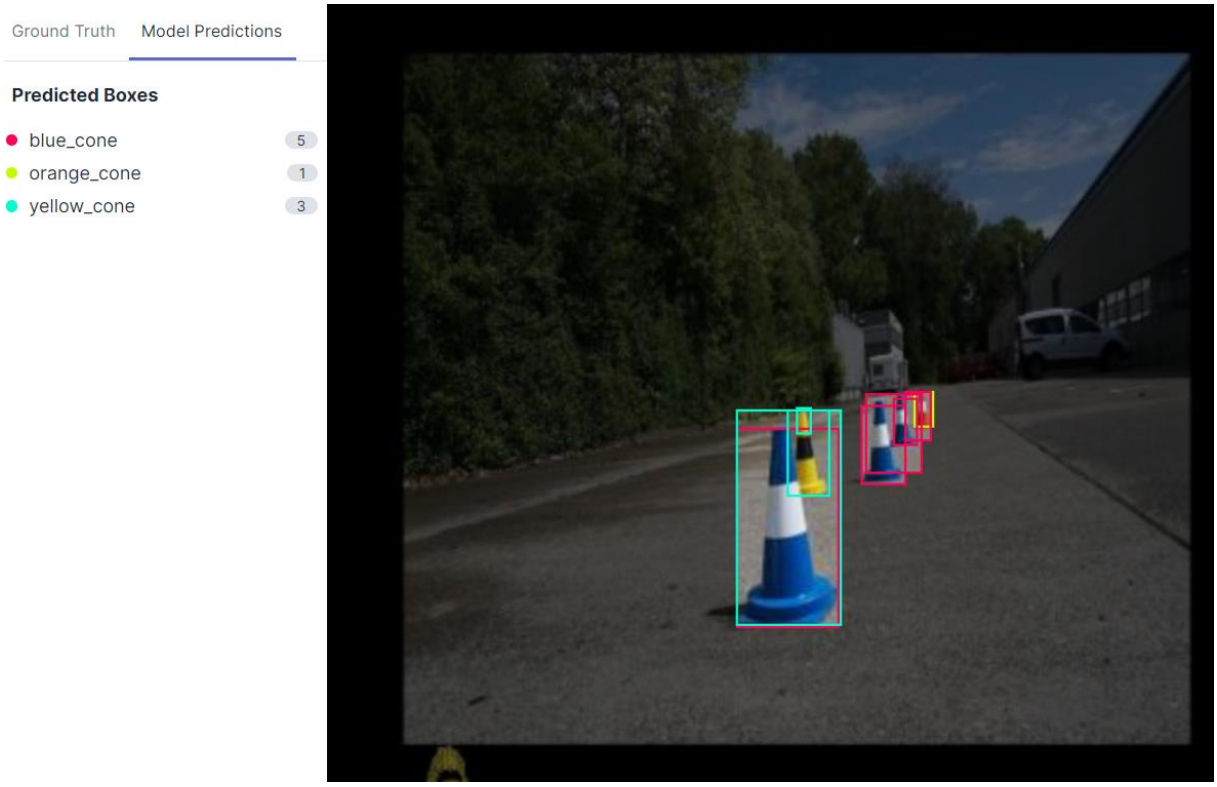
Slika 11. Mogućnosti augmentacije

S obzirom da je FSOCO skup slika zajedno s datotekama s oznakama objekata preuzet s platforme Supervise.ly, navedene datoteke s oznakama su formata JSON. Za treniranje pomoću modela YOLOv4 potrebno je pretvoriti JSON format u odgovarajući format za naš model. Ta pretvorba je također jedna od mogućnosti Robofow web platforme koju ćemo iskoristiti. Pri početku treniranja potrebno je odabrati skup podataka ne kojem ćemo model trenirati te se ovdje susrećemo s ograničenjem od maksimalno 1000 slika za besplatnu verziju platforme. Od tog skupa slika, 10% je izdvojeno za validacijski set, 20% za set za testiranje dok je preostalih 70% (ukupno 700 slika) preostalo za set za treniranje. Set za treniranje ćemo povećati prethodno navedenom mogućnosti augmentacije nakon čega u setu imamo ukupno 2100 slika.

Nakon završetka treniranja mreže, dobivena vrijednost za mAP je 45% te s obzirom da se radi o niskoj vrijednosti, odmah možemo zaključiti da dobiveni rezultati neće biti zadovoljavajući. Pregledom rezultata na validacijskom setu to možemo i potvrditi.



Slika 12. Primjer označenih objekata na slici iz validacijskog seta



Slika 13. Primjer detekcije objekta nadopunjenim modelom

3.3.2. Darknet radni okvir

S obzirom na nezadovoljavajuće rezultate dobivene nadogradnjom modela YOLOv4 korištenjem ograničenog broja slika za skup podataka, ponovit ćemo postupak nadogradnje modela ali ovaj put korištenjem Darknet radnog okvira kako bismo mogli iskoristiti cijeli dostupan skup podataka. Za provedbu treniranja modela u sklopu Darknet radnog okvira, potrebno nam je razumijevanje Supervise.ly i Darknet formata oznaka kako bismo mogli napraviti potrebnu pretvorbu iz jednog u drugi.

3.3.2.1. Supervise.ly format oznaka

Preuzimanjem našeg skupa podataka sa Supervise.ly platforme dobivamo slike zajedno s oznakama objekata za odgovarajuće slike. Spomenute oznake objekata se nalaze u txt datoteci u JSON formatu.


```
{
    "description": "",
    "tags": [
        {
            "id": 93798581,
            "tagId": 28921802,
            "name": "train",
            "value": null,
            "labelerLogin": "fsocov2",
            "createdAt": "2020-06-04T10:03:42.978Z",
            "updatedAt": "2020-06-04T10:03:42.978Z"
        }
    ],
    "size": {
        "height": 1480,
        "width": 2872
    },
    "objects": [
        {
            "id": 588041414,
            "classId": 2744363,
            "description": "",
            "geometryType": "rectangle",
            "labelerLogin": "fsocov2",
            "createdAt": "2020-06-04T10:03:42.979Z",
            "updatedAt": "2020-06-04T10:03:42.979Z",
            "tags": [],
            "classTitle": "blue_cone",
            "points": {
                "exterior": [
                    [
                        905,
                        565
                    ],
                    [
                        1079,
                        768
                    ]
                ],
                "interior": []
            }
        }
    ]
}
```


Slika 14. Primjer JSON datoteke za sliku s jednim čunjem – plavim čunjem

Prema priloženoj slici, možemo vidjeti da JSON datoteka sadrži informacije kao što su nazivi klasa kojima objekti na slici pripadaju (u ovom slučaju na slici je samo jedan objekt koji pripada klasi „blue_cone“), sadrži koordinate točaka koje se odnose na x i y koordinate za lijevu donju točku te desnu gornju točku granične kutije te također sadrži i visinu i širinu slike. U nastavku ćemo vidjeti zašto su nam baš navedene informacije bitne.

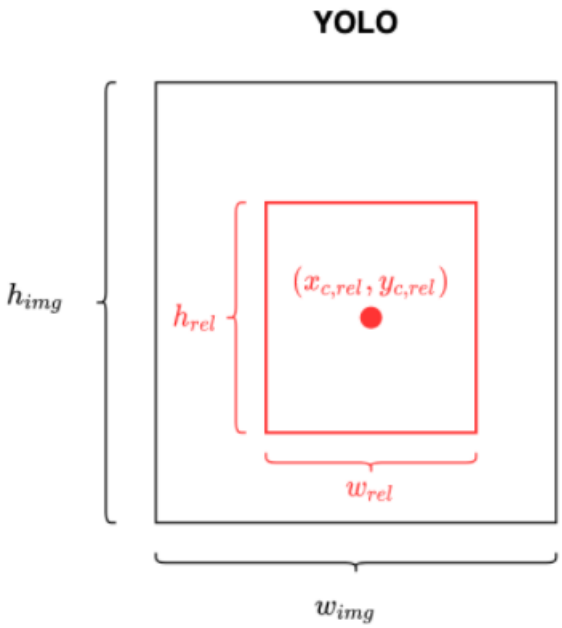
Također je bitno za spomenuti da osim prethodno objašnjenih JSON datoteka koje se odnose na pojedine slike, preuzimanjem skupa podataka također dobivamo i meta.json datoteku koja sadrži informacije o klasama svih objekata koji se pojavljuju u našem skupu.


```
{
    "classes": [
        {
            "title": "blue_cone",
            "shape": "rectangle",
            "color": "#2A00FF",
            "geometry_config": {},
            "id": 2744363,
            "hotkey": ""
        },
        {
            "title": "orange_cone",
            "shape": "rectangle",
            "color": "#FF8000",
            "geometry_config": {},
            "id": 2744364,
            "hotkey": ""
        },
       .
       .
       .
    ],
    "tags": [
        {
            "name": "truncated",
            "value_type": "none",
            "color": "#AC5FC4",
            "id": 28921800,
            "hotkey": "",
            "applicable_type": "objectsOnly",
            "classes": []
        },
        {
            "name": "knocked_over",
            "value_type": "none",
            "color": "#8421ED",
            "id": 28921801,
            "hotkey": "",
            "applicable_type": "objectsOnly",
            "classes": []
        },
      .
      .
      .
    ],
    "projectType": "images"
}
```


Slika 15. Skraćeni prikaz strukture meta.json datoteke s nekim od klasa i oznaka

3.3.2.2. YOLOv4 format oznaka

Txt datoteka s oznakama objekata za svaku pojedinu sliku karakteristična je po tome da se u jednom njenom retku nalaze potrebne informacije za po jedan objekt koji se nalazi na slici, te sukladno tome datoteka ima onoliko redaka koliko na slici ima objekata. Informacije koje se nalaze u jednom retku su redom: indeks klase, x koordinata središta granične kutije, y koordinata središta granične kutije, širina i visina granične kutije.



Slika 16. YOLOv4 koordinate

Umjesto jedne meta.json datoteke koju dobivamo preuzimanjem skupa podataka sa supervise.ly stranice, kod rada s YOLOv4 modelom potrebne su nam 4 datoteke: train.txt, valid.txt, data.obj i classes.names.

Train.txt i valid.txt sadrže put do direktorija u kojem se nalaze sve slike u podskupu za treniranje tj. za validaciju.

Data.obj sadrži 4 retka sa sljedećim informacijama: broj klasa objekata, put do datoteke train.txt i put do datoteke valid.txt te put do direktorija u kojem će se tijekom treniranja spremati dobivene težine.

3.3.2.3. Skripta za normiranje i konverziju

S obzirom na spomenute razlike u JSON formatu i formatu potrebnom za treniranje YOLOv4 modela, napravljena je python skripta za konverziju jednog formata u drugi. Uzevši u obzir da naš skup podataka sadrži i .png i .jpeg format slika, napravljena skripta također provodi normiranje.

Skripta prvo učitava JSON datoteku, jednu po jednu.

```
import json
import os
import glob

for filename in glob.glob('*.json'):
    with open(os.path.join(os.getcwd(), filename), 'r') as f:

        d = json.load(f)
```

Slika 17. Učitavanje datoteke

Zatim iz učitane JSON datoteke uzimamo potrebne informacije i spremamo ih u prikladne varijable.

```
size = d.get("size")
size_list = list(size.values())
all_cones_info = d.get("objects")
cone_number = len(all_cones_info)

class_list = []
all_points_list = []
for i in range(0, cone_number):
    cone_info = all_cones_info[i]
    cone_points = cone_info.get("points")
    cone_class = cone_info.get("classTitle")
    points_list = list(cone_points.values())
    exterior_points = points_list[0]
    all_points_list.append(exterior_points)
    class_list.append(cone_class)

Y = size_list[0]; X = size_list[1]
```

Slika 18. Uzimanje informacija

Potom koristeći definirane varijable provodimo pretvorbu iz JSON formata oznaka u YOLOv4 format oznaka, pritom normirajući dobivene vrijednosti.

```
for j in range(0, cone_number):
    x1 = all_points_list[j][0][0]
    y1 = all_points_list[j][0][1]
    x2 = all_points_list[j][1][0]
    y2 = all_points_list[j][1][1]
    x_sr=(x1+x2)/2
    y_sr=(y1+y2)/2
    w=x2-x1
    h=y2-y1
    x_norm_1 = x_sr / X
    y_norm_1 = y_sr / Y
    x_norm_2 = w / X
    y_norm_2 = h / Y
    if class_list[j]=="blue_cone":
        klasa=0
    elif class_list[j]=="large_orange_cone":
        klasa=1
    if class_list[j]=="orange_cone":
        klasa=2
    if class_list[j]=="yellow_cone":
        klasa=3
    outp = [str(klasa), " ", str(x_norm_1), " ", str(y_norm_1), " ", str(x_norm_2), " ", str(y_norm_2), '\n']
```

Slika 19. Pretvaranje

Na kraju tako dobivene vrijednosti spremamo u txt datoteku s odgovarajućim nazivom kako bi mreža pri treniranju mogla povezati sliku s odgovarajućom datotekom oznaka.

```
ime = "%s.txt" %filename
le_im=len(ime)
lista=[]
for l in range(0,le_im-13):
    k=ime[l]
    lista.append(k)
o="".join(lista)
puno_ime=o+".txt"
with open(puno_ime, 'a') as ff:
   for line in outp:
      ff.write(line)
```

Slika 20. Spremanje

```
2 0.7792479108635098 0.2576086956521739 0.7963091922005571 0.32065217391304346
2 0.807799442896936 0.21739130434782608 0.8199860724233984 0.2608695652173913
2 0.7036908077994429 0.3641304347826087 0.737116991643454 0.48478260869565215
2 0.9188718662952646 0.21195652173913043 0.9321030640668524 0.2576086956521739
2 0.8607242339832869 0.21521739130434783 0.8729108635097493 0.2576086956521739
4 0.8485376044568245 0.20217391304347826 0.8561977715877437 0.2326086956521739
```

Slika 21. Primjer txt datoteke s oznakama objekata nakon korištenja skripte

3.3.2.4. Izgradnja radnog okvira

Darknet radni okvir je javno dostupan na GitHub repozitoriju a pisan je u C programsko jeziku. Za ovaj ćemo rad koristiti račvu originalnog repozitorija u koju su dodana nova poboljšanja algoritma koju je napravio Alexey Bochkovskiy.

Izgradnja samog okvira je jednostavna. Prvo je potrebno preuzeti željeni repozitorij što ćemo u ovom slučaju napraviti korištenjem naredbe git clone. Nakon preuzimanja, Makefile datoteku moguće je podesiti na način da vrijednost za GPU, CUDNN i/ili OPNENCV stavimo na 1 i time omogućimo njihovo korištenje. Zadnji korak je pokretanje naredbe make u korijenskom direktoriju preuzetog repozitorija.

```
nikolina@nikolina-pc:~$ git clone https://github.com/AlexeyAB/darknet.git
nikolina@nikolina-pc:~$ cd darknet
nikolina@nikolina-pc:~/darknet$ make
```

Slika 22. Izgradnja radnog okvira

3.3.2.5. Postupak treniranja modela

Nakon uspješne izgradnje radnog okvira potrebno je osigurati sve potrebne datoteke za treniranje naše mreže:

- yolo-obj.cfg je datoteka u koju ćemo kopirati sav sadržaj iz datoteke yolov4-custom.cfg koja se nalazi u preuzetom repozitoriju. Zatim je u toj datoteci potrebno promijeniti sljedeće:
    - batch smo stavili na 64 što znači da se uzima skup od 64 slike te smo postavili subdivisions na 16 čime smo odredili da se od jednom učitavaju po 4 slike. No nakon pokretanja treniranja dobili smo grešku zbog pomanjkanja memorije te smo iz tog razloga i subdivisions stavili na 64
    - max_batches je potrebno postaviti na vrijednost 8000. To se određuje na način da se broj klasa pomnoži s 2000. Pri tome treba imati na umu da je 6000 minimalna vrijednost na koju se max_batches postavlja. Također, ta vrijednost ne smije biti manja od broja slika koje se nalaze u našem setu za treniranje
    - steps odgovara vrijednostima 80% i 90% od max_batches tj. steps=6400, 7200
    - width i height smo stavili na 416 (može se staviti i neki drugi broj koji je višekratnik od 32 npr. 608x608 čime ćemo dobiti veću preciznost)
    - u linijama 812, 887 i 962 postavljamo classes na 4
    - prema formuli filters=(classes+5)x3 vrijednost za filters postavljamo na 27 ali samo u linijama 807, 882 i 957
- train.txt, valid.txt, obj.names i obj.data datoteke koje su prethodno opisane u poglavlju

```
1 blue_cone
2 large_orange_cone
3 orange_cone
4 yellow_cone
```

Slika 23. obj.names

```
1 classes = 4
2 train = /home/nikolina/darknet/data/train.txt
3 valid = /home/nikolina/darknet/data/valid.txt
4 names = /home/nikolina/darknet/data/obj.names
5 backup = /home/nikolina/darknet/backup/
```

Slika 24. obj.data

Za samo generiranje train.txt datoteke (analogno i za valid.txt), napisana je sljedeća skripta:

```
import json
import os
import glob

for filename in glob.glob('*.jpg'):
        outp = ["/home/nikolina/darknet/data/obj/""%s" %filename, '\n']
        with open("train.txt", 'a') as ff:
                for line in outp:
                        ff.write(line)

for filename in glob.glob('*.png'):
        outp = ["/home/nikolina/darknet/data/obj/""%s" %filename, '\n']
        with open("train.txt", 'a') as ff:
                for line in outp:
                        ff.write(line)

```

Slika 25. Skripta za generiranje train.txt

Nakon toga, sa samim učenjem kreće se na sljedeći način:

```
nikolina@nikolina-pc:~/darknet$ ./darknet detector train data/obj.data yolo-obj.cfg yolov4.conv.137 -map
```

Slika 26. Pokretanje učenja mreže

Tijekom učenja u terminalu u kojem je pokrenuto učenje možemo kontinuirano pratiti tijek.

3.4. Usporedba rezultata dobivenih primjenom Roboflow web platforme i rezultata dobivenih primjenom Darknet radnog okvira

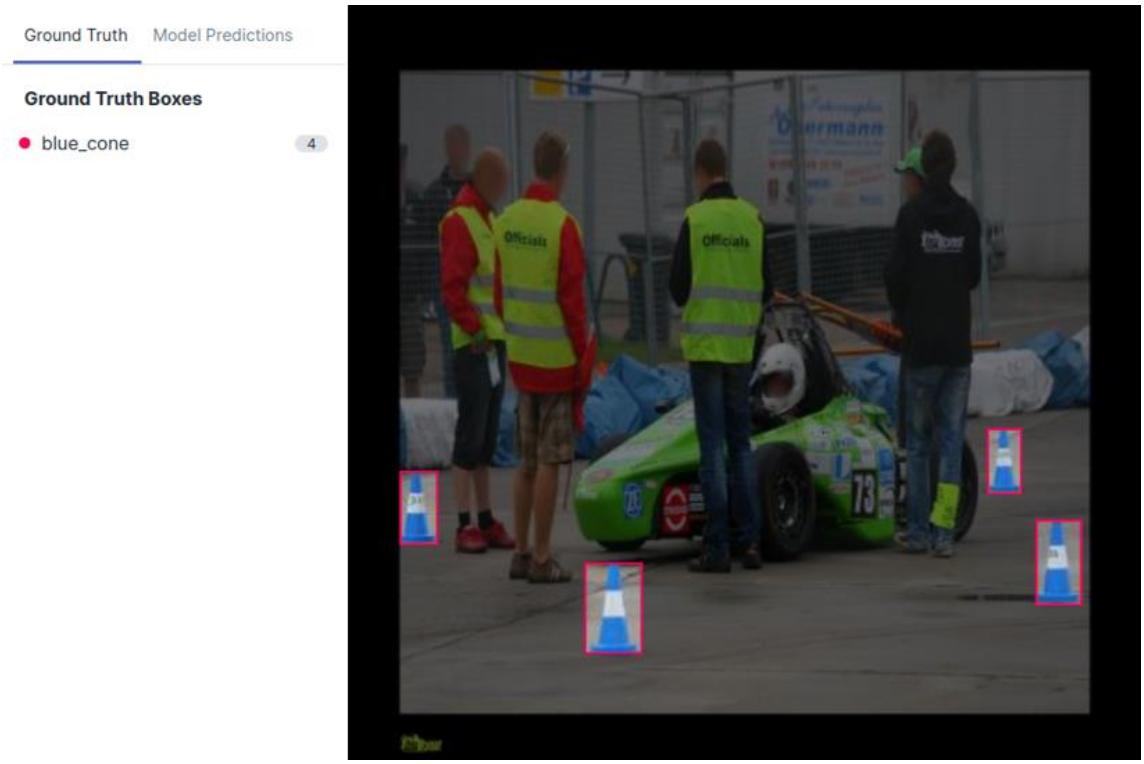
Procjenom rezultata detekcije koristeći težine dobivene prethodnim treniranjem modela, dobivamo sljedeće rezultate:

```
 detections_count = 12236, unique_truth_count = 7019
class_id = 0, name = blue_cone, ap = 67.00%       (TP = 2044, FP = 466)
class_id = 1, name = large_orange_cone, ap = 58.45%       (TP = 101, FP = 35)
class_id = 2, name = orange_cone, ap = 67.38%             (TP = 320, FP = 201)
class_id = 3, name = yellow_cone, ap = 69.11%             (TP = 2139, FP = 349)

 for conf_thresh = 0.25, precision = 0.81, recall = 0.66, F1-score = 0.73
 for conf_thresh = 0.25, TP = 4604, FP = 1051, FN = 2415, average IoU = 64.96 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.654865, or 65.49 %
```

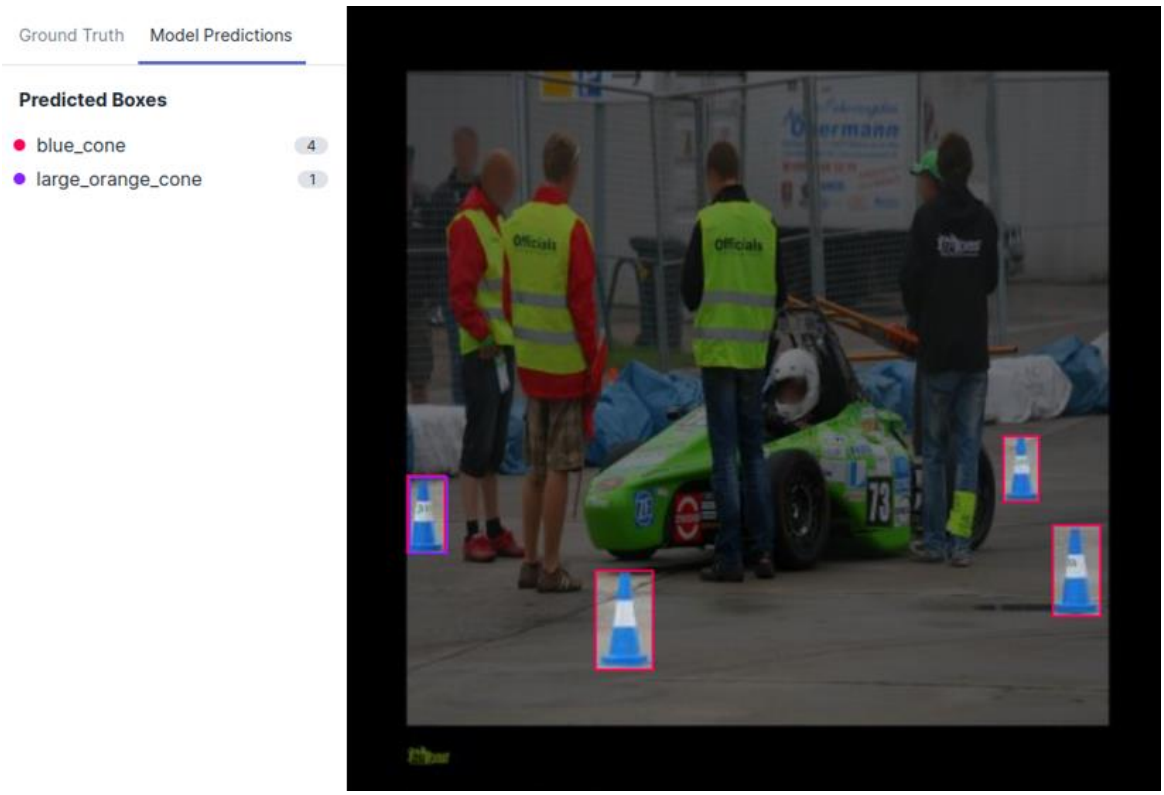
Slika 27. Rezultati - mAP

Usporedbom dobivene vrijednosti mAP-a, možemo vidjeti da je iznos značajno veći nego onaj dobiven korištenjem web platforme Roboflow. Vizualno rezultate možemo predočiti usporedbom rezultata na slikama iz validacijskog seta.

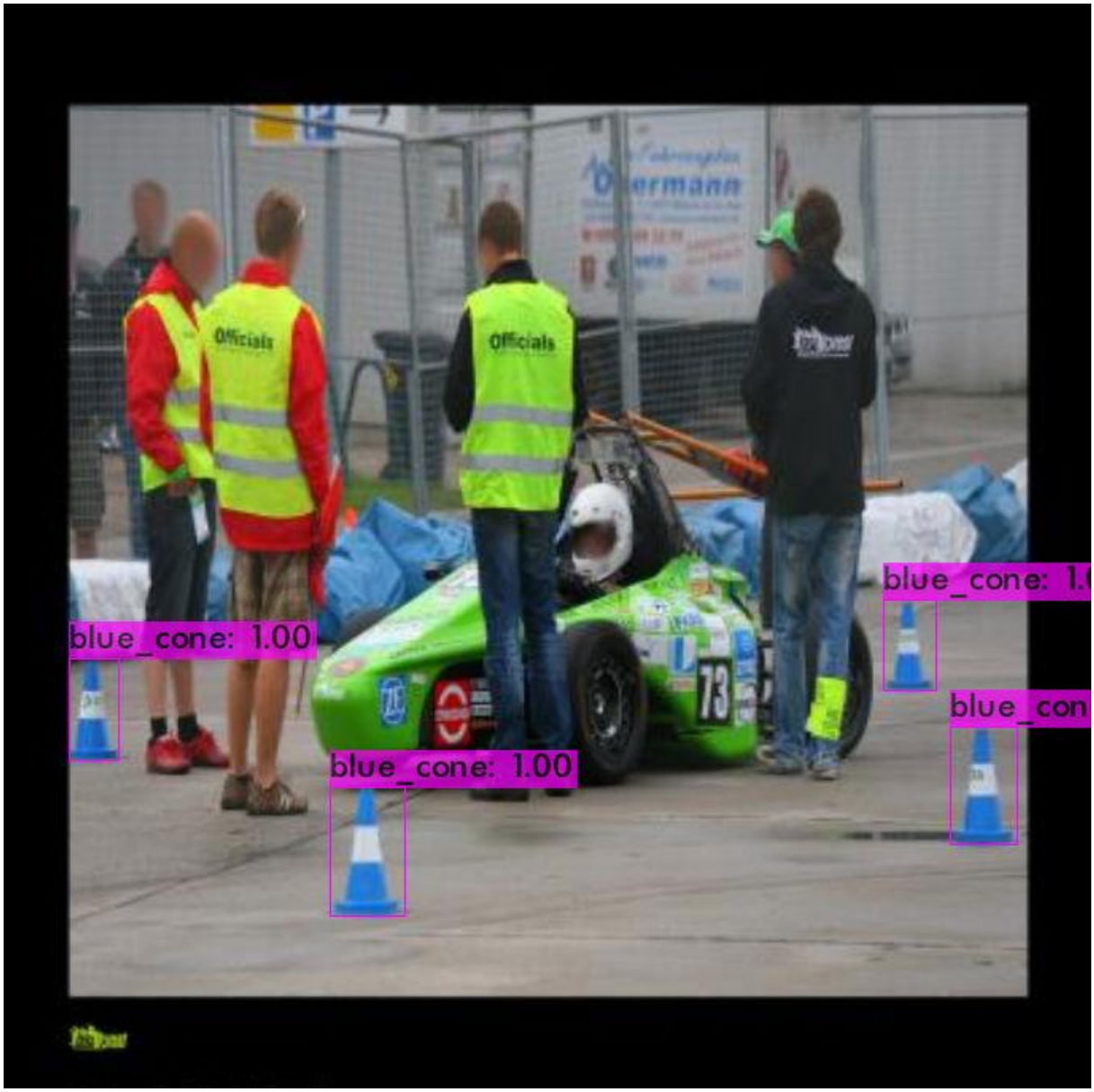
Primjer 1.



Slika 28. Primjer 1. – ispravne oznake

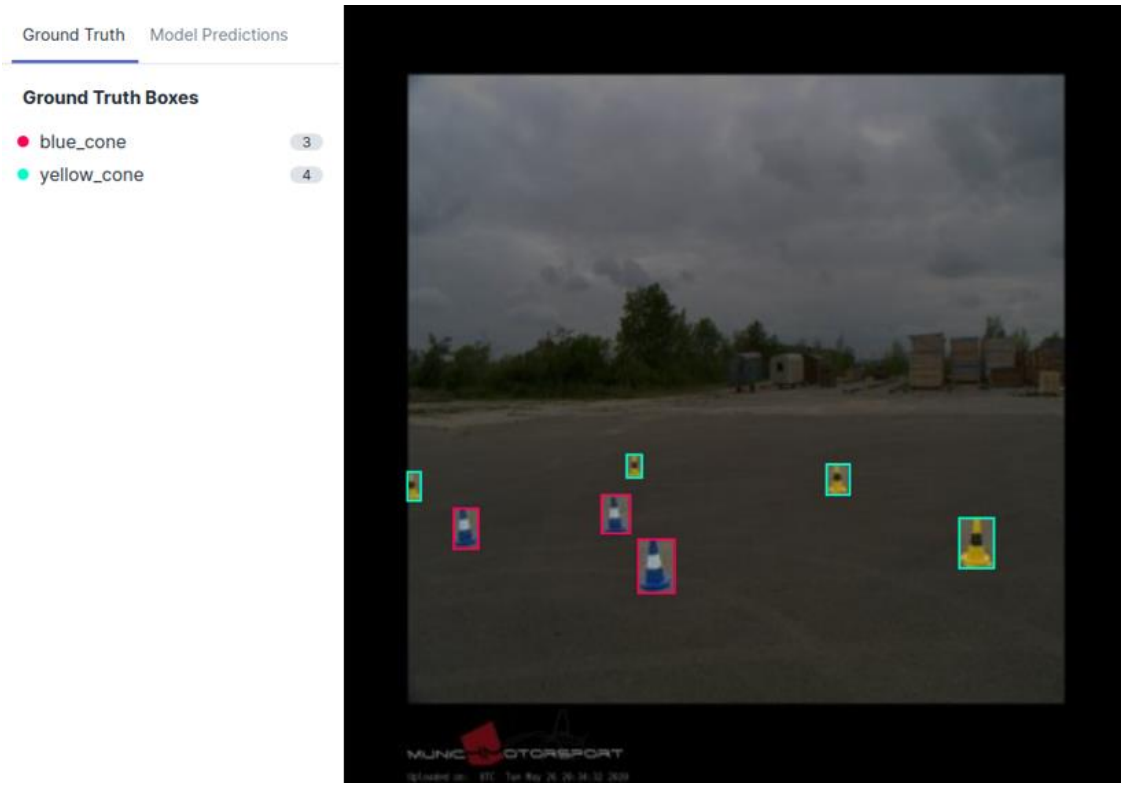


Slika 29. Primjer 1. – Roboflow detekcija

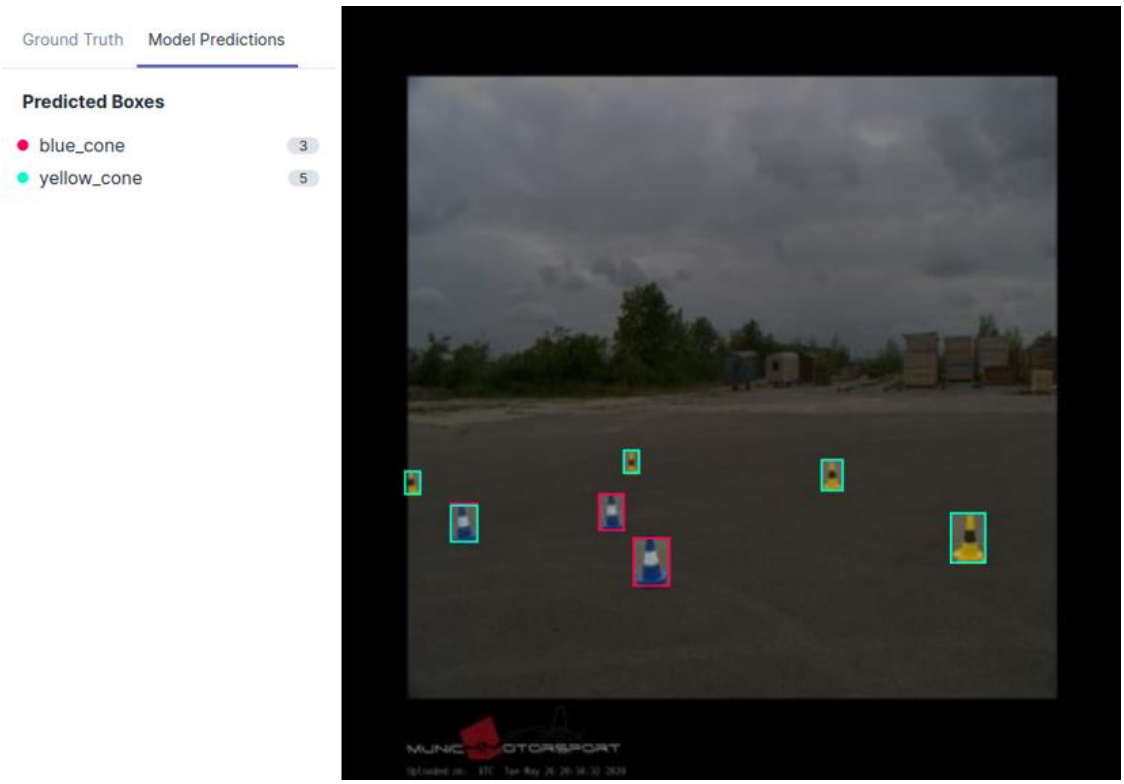


Slika 30. Primjer 1. – detekcija mreže trenirane u Darknet radnom okviru

Primjer 2.



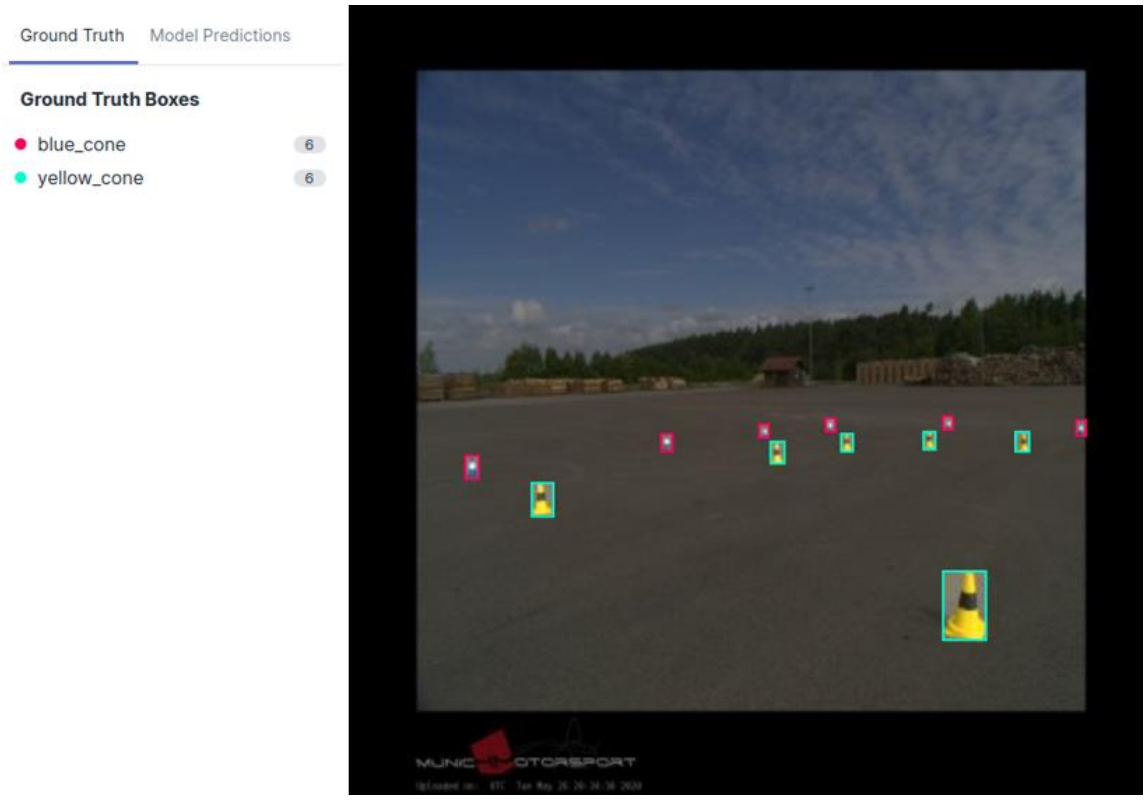
Slika 31. Primjer 2. - ispravne oznake



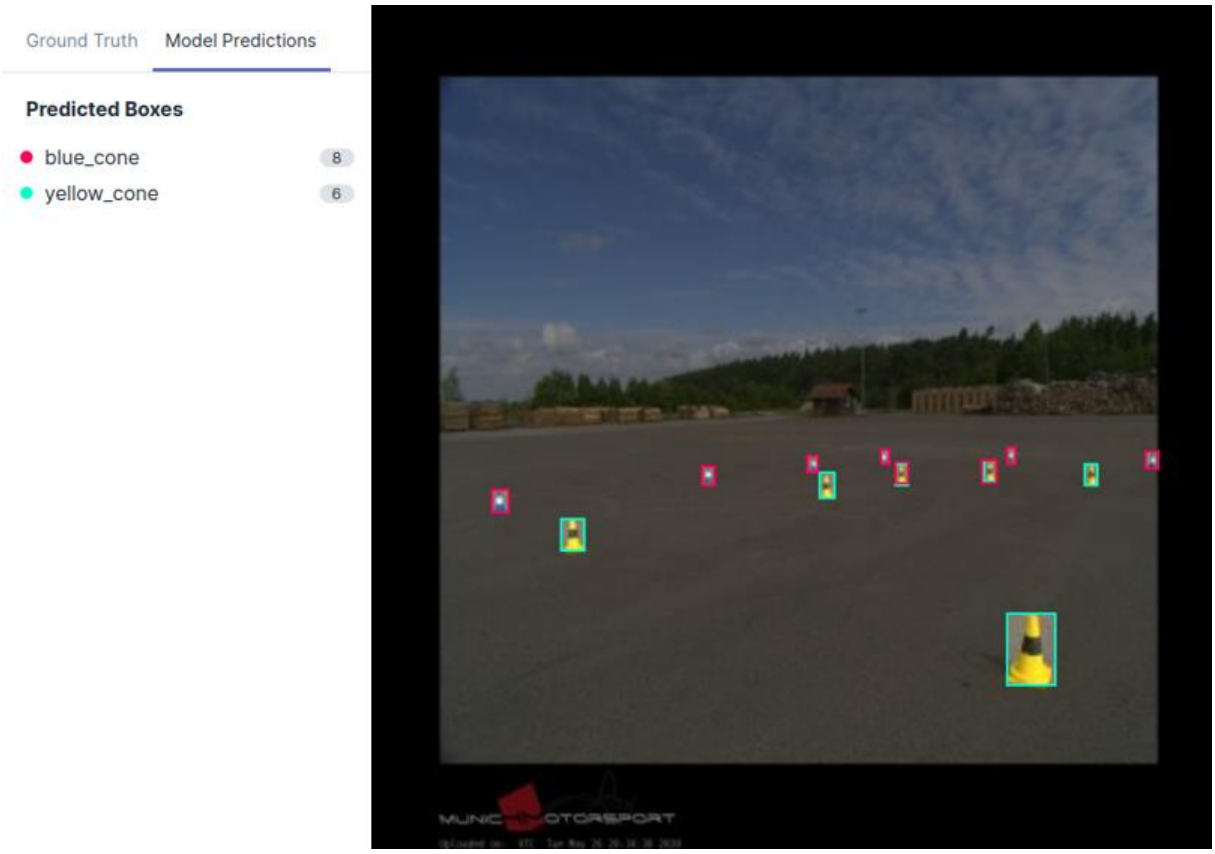
Slika 32. Primjer 2. – Roboflow detekcija

Slika 33. Primjer 2. – detekcija mreže trenirane u Darkent radnom okviru

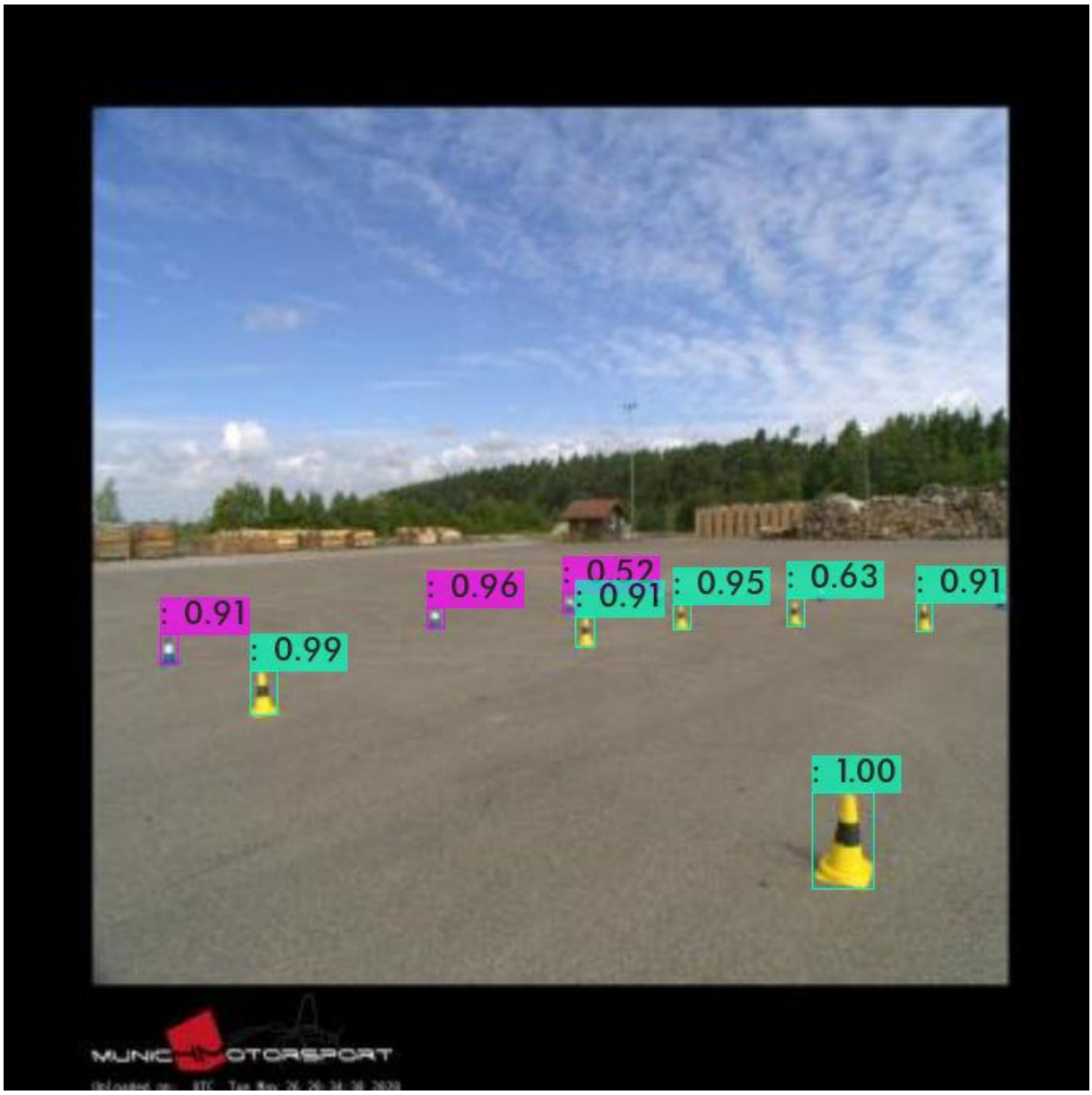
Primjer 3.



Slika 34. Primjer 3. – ispravne oznake



Slika 35. Primjer 3. – Roboflow detekcija



Slika 36. Primjer 3. – detekcija mreže trenirane u Darknet radnom okviru

U ovom su primjeru kod detekcije čunjeva primjenom Darknet radnog okvira maknuti nazivi klasa radi preglednosti slike.

4. ZAKLJUČAK

Generalno gledajući rezultate dobivenih detekcija možemo zaključiti da je odabrani model konvolucijske neuronske mreže prikladan za našu primjenu. Nadalje, pomoću kvalitetnog skupa podataka koji ujedno zadovoljava i kvantitetom možemo vidjeti da se rezultati značajno poboljšavaju (u našem primjeru ponajprije u obliku klasifikacije). Prilikom samog razmatranja primjene neuronske mreže za vizijski sustav, potrebno je uzeti u obzir da se cjelokupni sustav u bolidu izvodi naglašavajući robusnost i redundanciju. Iz tog razloga možemo pristati na kompromis i biti zadovoljni s mogućom nižom razinom točnosti detekcije primjenom YOLOv4 mreže naspram nekog drugog modela s obzirom da nam to omogućuje veću brzinu detekcije koja je za vozilo u pokretu iznimno bitna.

LITERATURA

[1] https://www.ibm.com/cloud/learn/machine-learning, pristupljeno: 9.1.2022.

[2] https://towardsdatascience.com/what-is-machine-learning-a-short-note-on-supervised-unsupervised-semi-supervised-and-aed1573ae9bb, pristupljeno: 12.1.2022.

[3] https://www.ibm.com/cloud/learn/neural-networks, pristupljeno: 4.2.2022.

[4] https://hrcak.srce.hr/file/322233, pristupljeno: 11.1.2022.

[5]http://matematika.fkit.hr/novo/izborni/referati/Popcevic_Varga_Zuvela_Neuronske_mreze.pdf, pristupljeno: 7.2.2022.

[6] https://repozitorij.unipu.hr/islandora/object/unipu%3A4735/datastream/PDF/view, pristupljeno: 3.2.2022.

[7]http://ferko.fer.hr/ferko/EPortfolio!dlFile.action;jsessionid=42FAF2BB2A7E915516D462A818D598F3?id=350, pristupljeno: 3.2.2022.

[8] https://repozitorij.fsb.unizg.hr/islandora/object/fsb%3A5657/datastream/PDF/view, pristupljeno: 19.2.2022.

[9] http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf, pristupljeno: 10.1.2022.

[10] https://github.com/christianversloot/machine-learning-articles/blob/main/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling.md, pristupljeno: 14.1.2022.

[11] https://github.com/AlexeyAB/darknet, pristupljeno: 2.11.2021.

[12]https://docs.supervise.ly/data-organization/00_ann_format_navi/04_supervisely_format_objects, pristupljeno: 24.10.2021.

[13] https://haobin-tan.netlify.app/ai/computer-vision/object-detection/coco-json-to-yolo-txt/, pristupljeno: 4.11.2021.

PRILOZI

I. Python kod - skripta za konverziju formata

II. Python kod - skripta za generiranje train.txt

I. Python kod – skripta za konverziju formata

```
import json
import os
import glob

for filename in glob.glob('*.json'):
    with open(os.path.join(os.getcwd(), filename), 'r') as f:

        d = json.load(f)
        size = d.get("size")
        size_list = list(size.values())
        all_cones_info = d.get("objects")
        cone_number = len(all_cones_info)

        class_list = []
        all_points_list = []
        for i in range(0, cone_number):
            cone_info = all_cones_info[i]
            cone_points = cone_info.get("points")
            cone_class = cone_info.get("classTitle")
            points_list = list(cone_points.values())
            exterior_points = points_list[0]
            all_points_list.append(exterior_points)
            class_list.append(cone_class)

        Y = size_list[0]; X = size_list[1]

        for j in range(0, cone_number):
            x1 = all_points_list[j][0][0]
            y1 = all_points_list[j][0][1]
            x2 = all_points_list[j][1][0]
            y2 = all_points_list[j][1][1]
            x_sr=(x1+x2)/2
            y_sr=(y1+y2)/2
            w=x2-x1
            h=y2-y1
            x_norm_1 = x_sr / X
            y_norm_1 = y_sr / Y
            x_norm_2 = w / X
            y_norm_2 = h / Y
            if class_list[j]=="blue_cone":
                klasa=0
            elif class_list[j]=="large_orange_cone":
                klasa=1
            if class_list[j]=="orange_cone":
                klasa=2
            if class_list[j]=="yellow_cone":
                klasa=3
            outp = [str(klasa), " ", str(x_norm_1), " ", str(y_norm_1), " ", str(x_norm_2), " ", str(y_norm_2), '\n']
            ime = "%s.txt" %filename
            le_im=len(ime)
            lista=[]
            for l in range(0,le_im-13):
                k=ime[l]
                lista.append(k)
            o="".join(lista)
            puno_ime=o+".txt"
            with open(puno_ime, 'a') as ff:
               for line in outp:
                  ff.write(line)
```

II. Python kod – skripta za generiranje train.txt

```
import json
import os
import glob

for filename in glob.glob('*.jpg'):
        outp = ["/home/nikolina/darknet/data/obj/""%s" %filename, '\n']
        with open("train.txt", 'a') as ff:
                for line in outp:
                        ff.write(line)

for filename in glob.glob('*.png'):
        outp = ["/home/nikolina/darknet/data/obj/""%s" %filename, '\n']
        with open("train.txt", 'a') as ff:
                for line in outp:
                        ff.write(line)

```