

Sveučilište Jurja Dobrile Pula

Fakultet informatike

Treniranje modela za detekciju semafora

Kolegij: Praktikum

Nositelj: doc. dr. sc. Siniša Miličić

Asistent: dr. sc. Mate Krišto

Student: Jakov Benedikt Ružić

Godina: 3.

Jmbag: 0269158890

Sadržaj

1. Uvod	3
2. Prikupljanje podataka	4
2.1 Prikup licenciranih podataka	4
2.2 Prikup podataka pomoću koda.....	4
3. Opis postupka.....	6
3.1 Anotacija podataka u YOLO formatu.....	6
3.1.1 Anotiranje	7
3.2 Treniranje modela	9
3.3 Detekcija objekta	11
4. Dijagrami.....	13
4.1 Confusion matrix	13
4.2 F1-Confidence	15
5. Zaključak.....	17

1. Uvod

Ovaj projekt je izrađen u sklopu kolegija 'Praktikum'. Zadatak je bio treniranje modela umjetne inteligencije. U ovom seminaru je opisan rad na treniranju modela umjetne inteligencije za detekciju semafora. Ova tema je izabrana zbog toga što se takav oblik UI konstantno pojavljuje na televizoru u filmovima itd. Htio sam isprobati, koliko je zapravo zahtjevno izraditi takav projekt. Stečeno iskustvo je opravdalo očekivanja prema ovom projektu, to je ujedno bio i cilj.

2. Prikupljanje podataka

Za ovaj projekt iskorišteno je 912 slika, s tim da je za validaciju upotrijebljeno 145 slika.

„Redlight“ klasa je klasa s najviše slika, zatim ide klasa „greenlight“, pa „yellowlight“.

Dataset za prepoznavanje boja na semaforu.

Tablica broj 1.

Klasa	Broj slika
Redlight	540
Greenlight	522
yellowlight	48

Izvor: Autor

Ovdje vidite da se sveukupni broj slika ne poklapa s brojem slika u tablici. To je za to što jednoj slici možemo labelirati više klasa. Na primjer, ako je na semaforu crveno svjetlo za naprijed u dosta slučajeva je upaljeno zeleno svjetlo za skretanje desno.

2.1 Prikup licenciranih podataka

Za ovaj projekt podaci su prikupljeni na 2 načina, u ovom dijelu će biti objašnjen prikup licenciranih podataka.

Licencirane podaci zajedno s licencom su prikupljeni s GitHub repozitorija

(<https://github.com/Thinklab-SJTU/S2TLD>).

Skinut je dataset podataka i licenca jer su podaci zaštićeni autorskim pravima.

2.2 Prikup podataka pomoću koda

Za prikup podataka pomoću koda, napravljen je `lights_images_download.py` file u root folderu(projekt_praktikum). Zatim je u navedenom file-u napisan ovaj kod.

Slika broj 1.

```
1 import os
2 import sys
3 from bing_image_downloader import downloader
4
5 # Suppress all print statements by redefining print
6 class SilentPrinter:
7     def __init__(self): # Ispravak metode __init__
8         self.original_stdout = sys.stdout
9
10    def __enter__(self): # Ovo omogućava korištenje 'with' bloka
11        sys.stdout = open(os.devnull, 'w') # Preusmjerava izlaz u /dev
12        /null
13        return self
14
15    def __exit__(self, exc_type, exc_val, exc_tb): # Povratak
16        originalnom izlazu
17        sys.stdout.close()
18        sys.stdout = self.original_stdout
19
20 def download_images(query, num_images):
21     with SilentPrinter(): # Koristi kontekstni manager za tisinu
22         downloader.download(query, limit=num_images, output_dir=query,
23                             adult_filter_off=True, force_replace=False, timeout=60)
24
25 # Download images of Red traffic light
26 download_images('red_traffic_lights', 200)
```

Izvor: Autor

Gore navedeni blok koda bi pokretali tako da prvo otvorimo command prompt, pokrenemo naredbu „python lights_images_download.py“ u željenom direktoriju. Nakon toga će nam se pojaviti novi folder u tom direktoriju gdje će biti slike.

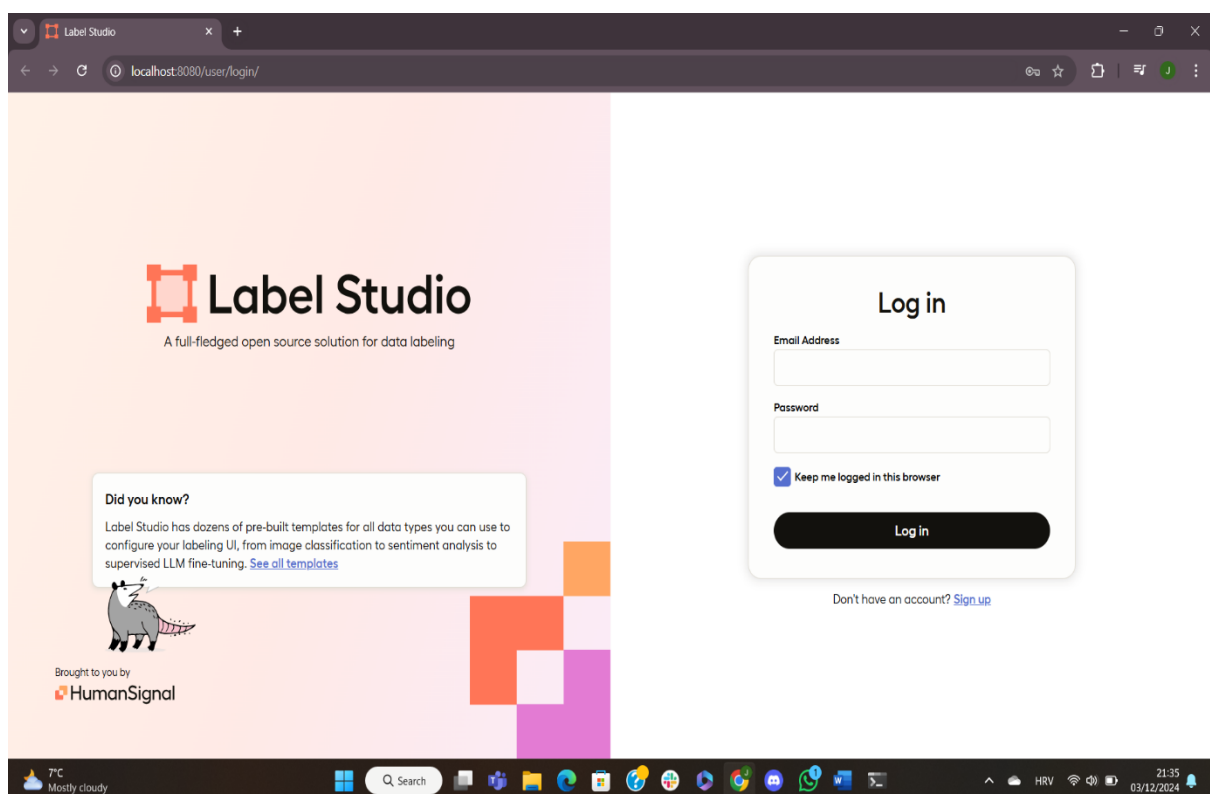
3. Opis postupka

U ovom poglavlju je općenito opisan postupak cijelog projekta. Gore je objašnjen postupak prikupljanja podataka, dok ćemo u ovom dijelu govoriti više o samom postupku treniranja modela.

3.1 Anotacija podataka u YOLO formatu

Za početak ćemo napraviti root folder, u ovom slučaju „projekt_praktikum“. Na taj folder ćemo se bazirati i u tom folderu će biti pohranjeno sve što ćemo napraviti. Nakon toga u root folderu radimo još jedan folder, u ovom slučaju „images“. U „images“ folder spremamo sve slike koje ćemo koristiti u daljnjem radu na projektu. Sada kroz comand prompt instaliramo label studio. Label studio je alat koji ćemo koristiti za labeliranje slika. Instalirat ćemo ga pomoću komande „pip install label-studio“. Pokrećemo label studio pomoću komande „label-studio“ i onda nam se label studio otvara u browser-u.

Slika broj 2.

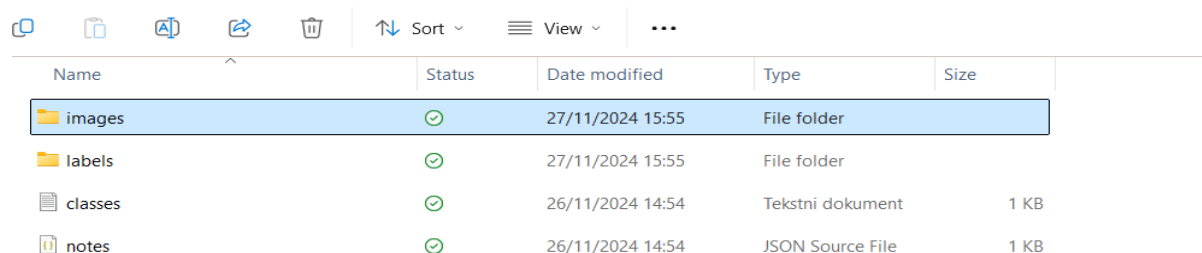


Izvor: Autor

3.1.1 Anotiranje

Prijavljujemo se u Label Studio, tamo odabiremo novi projekt, kliknemo na import da upload-amo naš dataset. Nakon toga Label Studio nas vodi na „labeling setup“, tu izabiremo „computer vision“, a onda „object detection with bounding boxes“ jer je u ovom slučaju tako bilo najlakše anotirati svijetla na semaforu. Kad smo sve to odabrali moramo označiti naše klase, u ovom slučaju to su bile: klasa „redlight“, klasa „greenlight“ i klasa „yellowlight“. Nakon toga počinjemo anotirati naše slike. Njih anotiramo tako da razvučemo pravokutnik točno onoliko koliko nam treba da pokrijemo svijetlo na semaforu. Ovaj proces je dug i iscrpljujuć, ali bez njega ne možemo dovršiti naš projekt. Na kraju se trud isplati. Kad smo završili naše anotiranje, kliknemo na „export“. Bitno je da export-amo u YOLO formatu. Anotirani dataset će se download-ati u zip. file-u, taj zip. file spremamo u naš glavni(root) folder. Kad smo ga spremili, više ne trebamo stari folder „images“, zato ga brišemo. Extract-amo novi zip. file i onda dobivamo „images“ folder, „labels“ folder, classes.txt i notes.json.

Slika broj 3.



Name	Status	Date modified	Type	Size
images	✓	27/11/2024 15:55	File folder	
labels	✓	27/11/2024 15:55	File folder	
classes	✓	26/11/2024 14:54	Tekstni dokument	1 KB
notes	✓	26/11/2024 14:54	JSON Source File	1 KB

Izvor: Autor

Radimo nova dva folder u root folderu, to su „train“ folder i „val“ folder. Kopiramo „images“ i „labels“ folder i stavljamo ih u „train“ folder. Za validaciju je korišteno oko 20% dataset-a, to jest 145 slika. Tih 145 slika i njihovih pripadajućih oznaka to jest labels-a, prebacujemo iz „train“ foldera u „val“ folder, u kojem smo također stvorili „images“ i „labels“ folder kako bi ih pravilno spremili. Stvaramo nova 2 file-a, to su „train.py“ i „dataset_custom.yaml“. „dataset_custom.yaml“ je file koji sadrži informacije o našem dataset-u, koje će nam trebati da istreniramo naš model.

Slika broj 4.

```
train: C:\Users\38595\OneDrive\Radna površina\projekt_praktikum\train
val: C:\Users\38595\OneDrive\Radna površina\projekt_praktikum\val

nc: 3

names: ["greenlight", "redlight", "yellowlight"]
```

Izvor: Autor

- „Train“ i „val“ su naši folderi, a pokraj njih upisujemo putanju do tih foldera.
- „nc“ predstavlja broj naših klasa, mi ih imamo 3.
- „names“ su imena naših klasa koje smo označili u Label Studiu.

3.2 Treniranje modela

„Train.py“ file koji smo stvorili sada otvaramo. Prva naredba nam je „from ultralytics import YOLO“, s ovom naredbom učitavamo YOLO model i python biblioteke ultralytics. Zatim naredbom „model = YOLO("yolo11m.pt")“, definiramo model. U ovom projektu korišten je yolo11m. Modele možemo pronaći na ultralytics GitHub repozitoriju. Treća, ujedno i zadnja naredba u train file-u je „model.train(data = "dataset_custom.yaml", imgsz = 640, batch = 8, epochs = 45, workers = 1, device = 0)“. Ova naredba pokreće treniranje YOLO modela.

- **data** = "dataset_custom.yaml" – povlačimo naš dataset, koji se nalazi u spomenutom yaml. file-u,
- **imgsz** = 640 – postavljamo veličinu ulaznih slike, slike se zatim automatski prilagođavaju,,
- **batch** = 8 – označava broj slika koji se istovremeno obrađuju,
- **epochs** = 45 – određujemo broj epoha za treniranje modela, odnosno koliko puta model prolazi kroz naš dataset,
- **workers** = 1 – broj workers-a za učitavanje podataka,
- **device** = 0 – označava koji resurs koristimo, ako je prisutna „0“ onda koristimo GPU

Slika broj 5.

```
1 from ultralytics import YOLO
2
3 model = YOLO("yolo11m.pt")
4
5 model.train(data = "dataset_custom.yaml", imgsz = 640, batch = 8,
6             epochs = 45, workers = 1, device = 0)
```

Izvor: Autor

Počinjemo treniranje s naredbom „python train.py“. Taj di je odrađen na Google Colabu, jer želimo pokrenuti model pomoću GPU-a, jer je CPU inače dosta sporiji. Dataset je stavljen na drive da mu možemo pristupiti preko colab-a. Prvo smo s ovom naredbom „!pip install ultralytics“ dohvatili biblioteku, zatim smo s ovom naredbom „from google.colab import drive drive.mount('/content/drive')“ pristupili folder-u.

U colabu to otprilike izgleda ovako:

Slika broj 6.



```
!pip install ultralytics
from ultralytics import YOLO

model = YOLO("/content/drive/MyDrive/projekt_praktikum/yolo11m.pt")

model.train(data = "/content/drive/MyDrive/projekt_praktikum/dataset_custom.yaml", imgsz = 640, batch = 8,
            epochs = 45, workers = 1, device = 0)
```

Izvor: Autor

3.3 Detekcija objekta

Kad model završi svoj trening, dobit ćemo „runs“ folder i „yolo11n.pt“ file, njih ćemo download-ati i spremiti u naš root folder. Sad kad sve to imamo onda slijedi detekcija objekta, to možemo obaviti na slikama, videu ili web kameri. Ovdje smo odabrali slike i video. Sad ćemo pronaći „best.pt“ file, putanja do ovog file-a je obično „runs/detect/train/weights“. Kopirat ćemo file i prebacit ga u root folder, preimenovali smo ga u „yolo11_custom.pt“. Trening je s tim potpuno gotov. Sada ćemo napraviti „predict.py“ file, s njime ćemo obaviti detekciju objekta. Prva naredba je opet „from ultralytics import YOLO“, zatim „model = YOLO(““““, između navodnika ćemo staviti preimenovani best.pt file. Inače best.pt file je datoteka koja sadrži najbolji trenirani model tijekom procesa treniranja. Zadnja naredba je model.predict(source = “““, show=True, save=True), tu ćemo staviti sliku ili video na kojima želimo vršiti detekciju, slika ili video se trebaju nalaziti negdje u folderu.

show=True – omogućava da vidimo rezultate predikcije na ekranu,

save=True – sprema rezultate predikcije.

Slika broj 7.













```
1 from ultralytics import YOLO
2
3 model = YOLO("yolo11_custom.pt")
4
5 model.predict(source = "", show=True, save=True)
```

Izvor: Autor

Kad je predikcija gotova datoteka se obično sprema u “predict” folder koji se obično nalazi na ovoj putanji “\runs\detect”.

Na kraju naš folder izgleda ovako:

Slika broj 8.

Name	Status	Date modified	Type
 project-3-at-2024-11-26-14-54-34f1664f	✓	27/11/2024 15:55	File folder
 runs	✓	01/12/2024 16:57	File folder
 train	✓	27/11/2024 16:00	File folder
 val	✓	27/11/2024 16:12	File folder
 dataset_custom	✓	03/12/2024 22:51	Yaml Sourc
 licenca	✓	26/11/2024 15:05	Tekstni dok
 lights_images_download	✓	22/11/2024 22:25	Python File
 predict	✓	06/12/2024 15:24	Python File
 train	✓	01/12/2024 20:55	Python File
 yolo11m.pt	✓	13/11/2024 17:26	PT File
 yolo11n .pt	✓	01/12/2024 01:28	PT File
 yolov11_custom.pt	✓	01/12/2024 16:57	PT File

Izvor: Autor

4. Dijagrami

U ovom poglavlju pokazat ćemo 2 dijagrama, confusion matrix dijagram i F1-Confidence dijagram. Ovi dijagrami pokazuju koliko je pouzdan i učinkovit model.

4.1 Confusion matrix

Na slici je prikazana normalizirana matrica zabune (confusion matrix) modela klasifikacije prometnih svjetala i pozadine. Okomita os označava predviđene klase (Predicted), dok vodoravna os predstavlja stvarne klase (True). Kategorije uključuju:

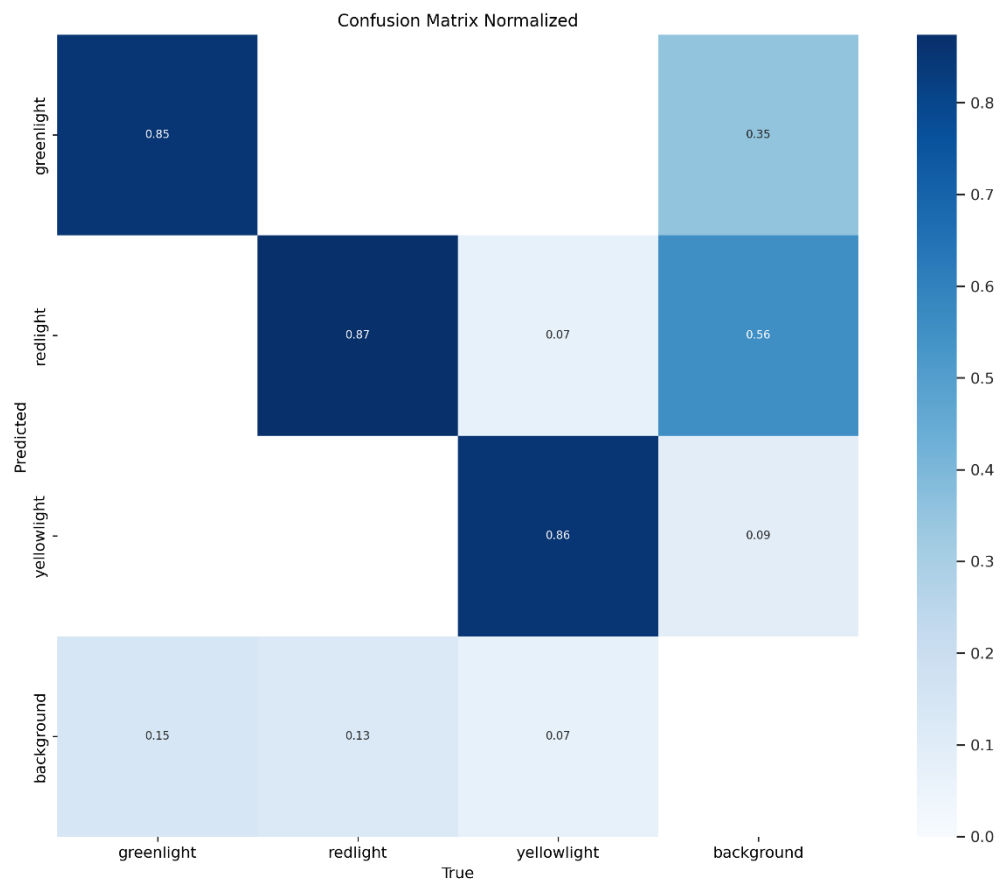
- **greenlight (zeleno svjetlo)**
- **redlight (crveno svjetlo)**
- **yellowlight (žuto svjetlo)**
- **background (pozadina)**

Dijagonalni elementi predstavljaju točno klasificirane uzorke, dok nedijagonalni elementi označavaju pogrešne klasifikacije. Intenzitet boja odražava učestalost klasifikacija – tamniji tonovi označavaju veću učestalost.

Ključne točke analize:

- Model postiže visoku točnost za sve klase prometnih svjetala:
 - Zeleno svjetlo (**0,85**)
 - Crveno svjetlo (**0,87**)
 - Žuto svjetlo (**0,86**)
- Kategorija **background (pozadina)** pokazuje veću stopu pogrešnih klasifikacija, osobito kao:
 - Crveno svjetlo (**0,13**)
 - Zeleno svjetlo (**0,15**)

Slika broj 9.



Izvor: Autor

4.2 F1-Confidence

Na slici je prikazana F1-Confidence krivulja, koja pokazuje odnos između F1-mjere i razine pouzdanosti modela za klasifikaciju prometnih svjetala. Krivulja je prikazana za tri klase:

- **greenlight (zeleno svjetlo) – plava tanka linija**
- **redlight (crveno svjetlo) – narančasta linija**
- **yellowlight (žuto svjetlo) – zelena linija**
- **Srednja vrijednost za sve klase – plava debela linija**

Os x predstavlja prag pouzdanosti predikcije, dok os y prikazuje F1-mjeru, koja balansira preciznost (precision) i odziv (recall) klasifikacije.

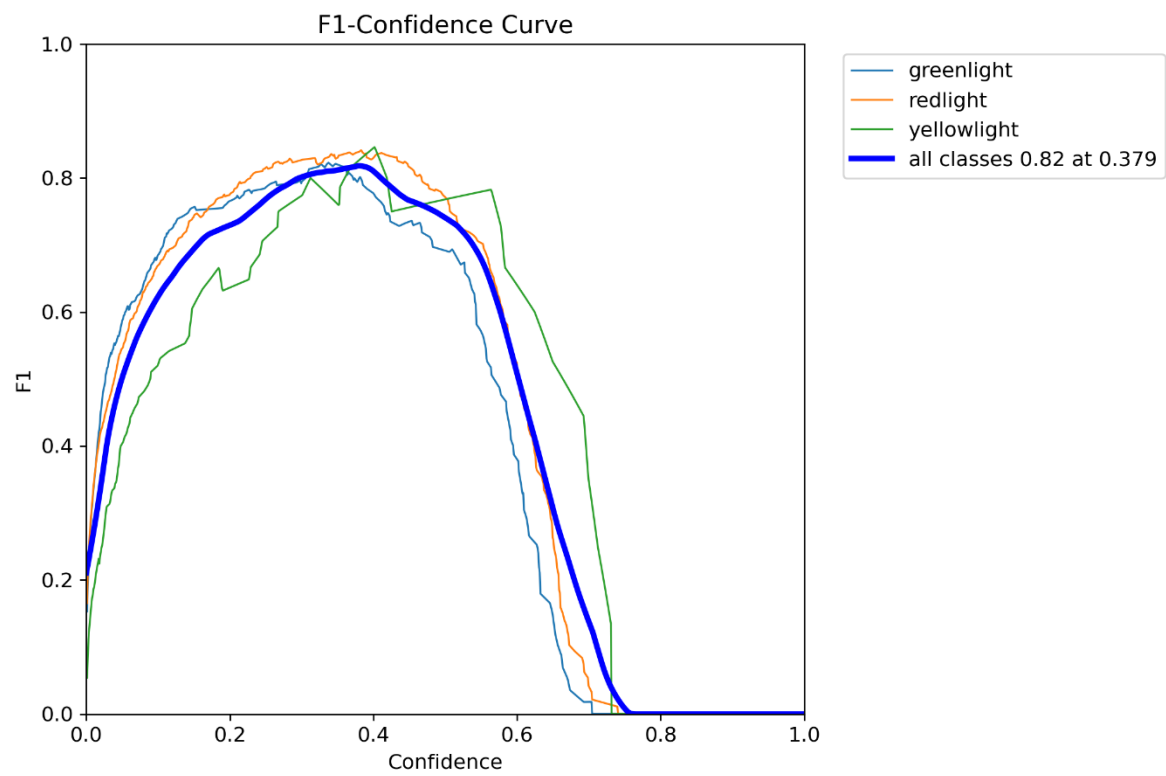
Ključne točke analize:

Maksimalna F1-mjera za sve klase iznosi 0,82 pri pouzdanosti od 0,379, što je naznačeno na legendi.

Krivulje za pojedine klase pokazuju različite performanse:

- Crveno svjetlo (redlight) pokazuje konzistentno dobre rezultate, s visokim F1-vrijednostima u širokom rasponu pouzdanosti.
- Zeleno svjetlo (greenlight) ima nešto veću varijabilnost, osobito pri višim pragovima pouzdanosti.
- Žuto svjetlo (yellowlight) pokazuje najviše oscilacija, što može ukazivati na poteškoće modela u preciznoj detekciji ove klase.

Slika broj 10.



Izvor: Autor

5. Zaključak

Iako se trening modela, a samim time i detekcija može dosta popraviti ubacivanjem još dataset-a, treniranje na više epoha itd. Smatramo da je ovaj model dosta dobar s obzirom da smo ovako nešto radili po prvi put i u nekim dijelovima imali ograničenja poput „GPU“. Rad na ovom projektu je bio dosta zanimljiv i zabavan, vjerujemo da ćemo napraviti još sličnih modela. Nadamo se da će se i Vama svidjeti.