Razvoj aplikacije za detekciju slobodnih mjesta na parkiralištu korištenjem dubokog učenja

Babić, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1421

RAZVOJ APLIKACIJE ZA DETEKCIJU SLOBODNIH MJESTA NA PARKIRALIŠTU KORIŠTENJEM DUBOKOG UČENJA

Luka Babić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1421

RAZVOJ APLIKACIJE ZA DETEKCIJU SLOBODNIH MJESTA NA PARKIRALIŠTU KORIŠTENJEM DUBOKOG UČENJA

Luka Babić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1421

Pristupnik:	**Luka Babić (0036538016)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Marko Horvat

Zadatak: **Razvoj aplikacije za detekciju slobodnih mjesta na parkiralištu korištenjem dubokog učenja**

Opis zadatka:

Neuronske mreže dubokog učenja naziv je za skupa modela umjetnih neuronskih mreža koje kroz višeslojnu obradu podataka konstruiraju apstrakcije visoke razine. Ti su se modeli pokazali iznimno učinkoviti u nizu zadataka kao što su obrada prirodnog jezika i računalni vid. Prvenstveno u području računalnog vida, pokazali su veliku primjenjivost u implementaciji različitih sustava za nadzor kroz detekciju objekata u slici. Cilj ovog rada je razviti aplikaciju koja koristi radni okvir zasnovan na dubokom učenju za detekciju slobodnih mjesta u stvarnom okruženju parkiralištu fakulteta. Takva aplikacija bila bi od velike koristi djelatnicima i gostima fakulteta, a posebice osoblju zaduženom za upravljanje infrastrukturom, jer bi u širem sustavu mogla olakšati odabir parkirnog mjesta i nadzor parkirališta. U okviru ovog završnog rada potrebno je upoznati se s radnim okvirom YOLO (You Only Look Once), te koristeći razvojno okruženje u programskom jeziku Python izraditi aplikaciju koja će implementirati traženu funkcionalnost. Snimiti video isječke parkirališta za naknadnu analizu. Provesti eksperimentalno vrednovanje i statistički obraditi rezultate. Prikazati komponente izrađenog sustava, bitne isječke izvornog programskog koda te definirati korištena programska sredstva i potrebne postupke. Radu priložiti izvorni i izvršni kod razvijenog sustava uz potrebna dodatna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem se mentoru, doc. dr. sc. Marku Horvatu, na mentoriranju i svoj pruženoj pomoći tokom pisanja ovog rada.

Zahvaljujem se obitelji i prijateljima na podršci.

Sadržaj

1. Uvod

Pronalaženje slobodnog parkirnog mjesta predstavlja značajan problem u urbanim područjima, što dovodi do gubitka vremena, povećanja stresa i dodatnih troškova za vozače. Osim toga, pretrpane parkirne zone mogu uzrokovati prometne gužve i smanjiti učinkovitost korištenja dostupnog prostora. Razvojem tehnologija računalnog vida i dubokog učenja, otvorila se mogućnost za automatizirano praćenje i detekciju slobodnih parkirnih mjesta.

Cilj ovog rada je razviti aplikaciju koja koristi YOLOv8 (You only look once) [1], napredni model za detekciju objekata, za prepoznavanje slobodnih parkirnih mjesta na parkiralištima oko fakulteta.

Eksperimentalno vrednovanje i statistička analiza rezultata omogućit će procjenu učinkovitosti sustava te ukazati na moguća područja za daljnje istraživanje i poboljšanje.

.

2. Opis problema

Motivacija za aplikaciju razvijenu u sklopu ovog završnog rada proizašla je iz potrebe za učinkovitim upravljanjem parkirnim prostorima na fakultetu. Pronalaženje slobodnog parkirnog mjesta može biti izazovno i frustrirajuće za vozače, posebno tijekom perioda visoke popunjenosti kao što je to razdoblje tokom dolaska na posao, te odlaska kući. Poteškoće u pronalasku slobodnog mjesta ne samo da uzrokuju gubitak vremena i povećavaju stres vozača, već također dovode do prometnih gužvi i smanjuju ukupnu učinkovitost korištenja dostupnog prostora.

Kako bismo riješili ovaj problem, potrebno je razviti sustav koji može automatski detektirati slobodna parkirna mjesta u stvarnom vremenu. Takav sustav treba biti sposoban analizirati videozapise parkirališta, identificirati slobodna i zauzeta mjesta, te pružiti korisnicima ažurirane informacije o dostupnosti parkirnih mjesta. Ključni izazovi uključuju prepoznavanje parkirnih mjesta u različitim uvjetima osvjetljenja, prilagođavanje različitim konfiguracijama parkirališta i osiguravanje visoke točnosti detekcije.

Ovaj sustav će omogućiti kontinuirano praćenje parkirališta, pružajući korisnicima točne i pravovremene informacije o dostupnosti parkirnih mjesta, što će olakšati upravljanje parkirnim prostorom i povećati zadovoljstvo korisnika.

Razvijeni sustav bit će evaluiran kroz eksperimentalno vrednovanje i statističku analizu rezultata kako bi se procijenila njegova učinkovitost i identificirale mogućnosti za daljnje poboljšanje. Na taj način, ovaj rad će doprinijeti boljem razumijevanju i rješavanju problema upravljanja parkirnim prostorima korištenjem suvremenih tehnologija računalnog vida i dubokog učenja.

.

2.1. Duboko učenje

Duboko učenje podskup je strojnog učenja koji koristi višeslojne neuronske mreže za analizu i učenje iz velikih količina podataka [2]. U kontekstu detekcije slobodnih parkirnih mjesta, duboko učenje omogućuje modelima da automatski prepoznaju i klasificiraju objekte na slikama ili video zapisima s visokim stupnjem točnosti. Korištenjem slojeva

neuronskih mreža, model može naučiti složene apstrakcije i značajke koje razlikuju slobodna od zauzetih parkirnih mjesta, čak i u uvjetima različite osvijetljenosti i perspektive.

Ova sposobnost prilagodbe i učenja iz velikih skupa podataka čini duboko učenje idealnim rješenjem za problem detekcije slobodnih parkirnih mjesta, pružajući pouzdane rezultate u stvarnom vremenu.

2.2. Računalni vid

Računalni vid je grana računarske znanosti koja se bavi razumijevanjem i interpretacijom vizualnih informacija iz digitalnih slika ili videozapisa [3]. U kontekstu detekcije slobodnih parkirnih mjesta, računalni vid omogućuje sustavima da analiziraju vizualne podatke s parkirališta kako bi automatski identificirali prisutnost i lokaciju vozila.

Koristeći napredne tehnike obrade slike i dubokog učenja, računalni vid omogućuje modelima da prepoznaju obrasce i karakteristike parkirnih mjesta te ih klasificiraju kao slobodna ili zauzeta. Integracija računalnog vida u sustav detekcije parkirnih mjesta omogućuje brzu i preciznu identifikaciju slobodnih mjesta, pridonoseći efikasnom upravljanju parkirnim prostorom i poboljšavajući iskustvo korisnika.

3. Zahtjevi programske potpore i arhitektura sustava

3.1. Funkcionalni zahtjevi

3.1.1. Učitavanje videozapisa ili fotografije

Aplikacija treba omogućiti učitavanje fotografije ili videozapisa, bilo lokalno ili putem nekog online servisa (npr. Google Drive) kako bi se učitani podatak mogao koristiti za daljnju analizu pomoću dubokog učenja, te kako bi se sačuvala referenca na podatak u bazu podataka.

3.1.2. Estimacija broja automobila na parkiralištu

Aplikacija treba omogućiti prolaz umjetne inteligencije, to jest računalnog vida pomoću dubokog učenja, po učitanom videozapisu ili fotografiji, te vratiti rezultat koji će biti korišten za daljnju obradu.

3.1.3. Prikaz mjera kvalitete performansi

Na temelju rezultata analize rezultata obrade, aplikacija treba vrednovati kvalitetu detekcije duboke neuronske mreže. Mjere koje je potrebno izvesti su sljedeće: točnost (engl. *accuracy*), odziv (engl. *recall*), preciznost (engl. *precision*) i *F1-score* (izrazi su prikazani i numerirani istim redoslijedom ispod) [6]. Ove mjere potrebno je prikazati na ekranu kako bi jednostavno bili vidljivi korisniku aplikacije.

$$ACC = (TP + TN) \,/\, (TP + TN + FP + FN) \quad (1)$$

$$TPR = TP \,/\, (TP + FN) \quad (2)$$

$$PPV = TP / (TP + FP) \quad (3)$$

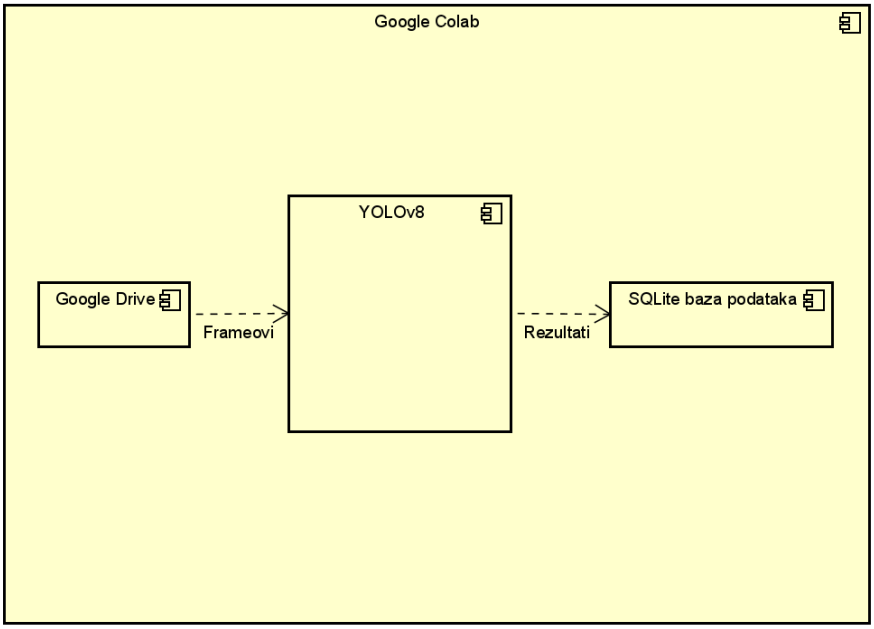
$$F1 = 2 * TP / (2TP + FP + FN) \quad (4)$$

3.1.4. Spremanje obrađenog videozapisa ili fotografije

Nakon što je provedena obrada i analiza učitanog videozapisa ili fotografije, aplikacija treba moći sačuvati obrađeni videozapis ili fotografiju, te detalje analize, lokalno na korisnikovo računalo ili na neki udaljeni servis (npr. Google drive).

3.2. Arhitektura sustava

Sustav se sastoji od Google Colab bilježnice koja je glavna komponenta sustava i čini samu aplikaciju, baze podataka koja ostvaruje interakciju između korisnika i aplikacije, komponente za obradu videozapisa ili fotografije koja koristi metode računalnog vida, odnosno duboku neuronsku mrežu.



Slika 3.1 Dijagram komponenti sustava

3.2.1. Google Colab

Za temelj aplikacije odabrano je korištenje Google Colab-a koji omogućava udaljeno korištenje grafičke kartice za provedbu analize (što uvelike ubrzava proces ako korisnik lokalno nema dovoljno jaku grafičku karticu).

Sljedeći razlog zašto je baš Google Colab odabran jest zbog mogućnosti korištenja Python programskog jezika koji nudi razne biblioteke koje će olakšati analizu.

Posljednji razlog odabira upravo Google Colab-a jest taj da je vrlo intuitivan za korištenje, te će biti jednostavan za navigiranje budućim korisnicima.

3.2.2. Baza podataka

Baza podataka sastoji se od 4 tablice, od kojih svaka ima ključnu ulogu za arhitekturu.

ID	INTEGER	Jedinstveni identifikator videozapisa
Name	TEXT	Naziv videozapisa
Path	TEXT	Putanja do videozapisa
Import_date	TIMESTAMP	Vrijeme uvoza videozapisa u bazu podataka

3.1 Tablica „Videos“

ID	INTEGER	Jedinstveni identifikator analize rezultata
Video_id	INTEGER	Vanjski ključ na videozapis ili fotografiju koja je povezana s analizom
Detected_objects	INTEGER	Broj detektiranih vozila

Number_of_parkings	INTEGER	Broj parking mjesta koje unosi korisnik
Analysis_date	TIMESTAMP	Datum analize

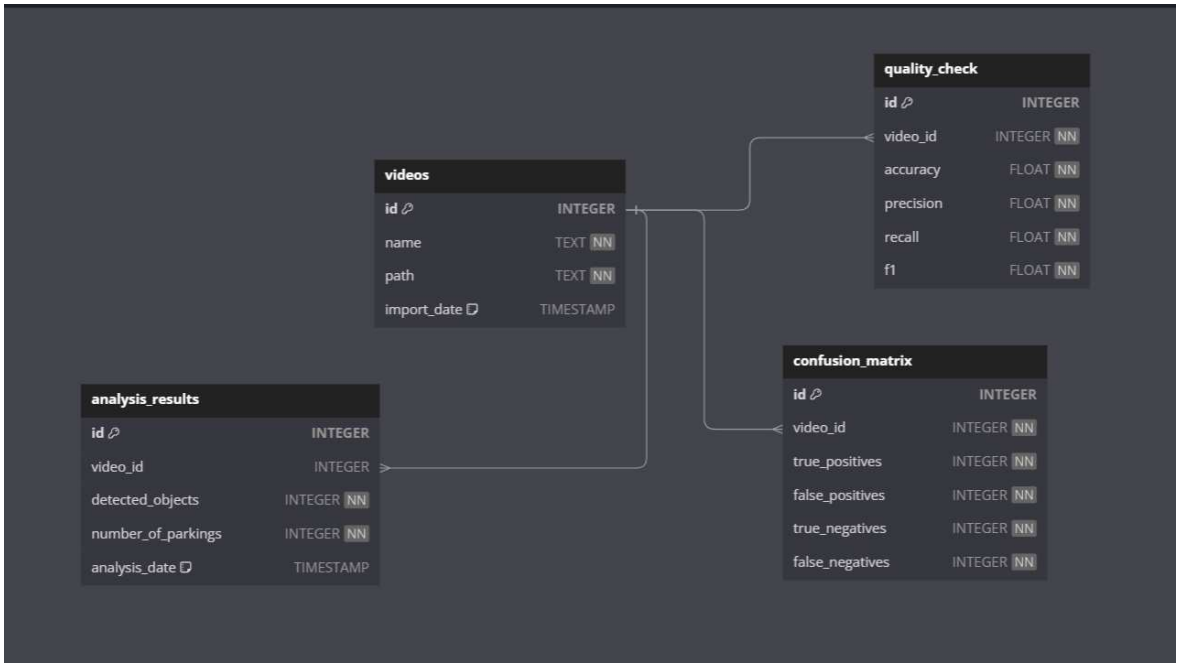
3.2 Tablica „Analysis_results“

ID	INTEGER	Jedinstveni identifikator konfuzijske matrice
Video_id	INTEGER	Vanjski ključ na videozapis ili fotografiju koja je povezana s matricom
True_positives	INTEGER	Broj točno pozitivnih
False_positives	INTEGER	Broj netočno pozitivnih
True_negatives	INTEGER	Broj točno negativnih
False_negatives	INTEGER	Broj netočno negativnih

3.3 Tablica „Confusion_matrix“

ID	INTEGER	Jedinstveni identifikator mjera kvalitete
Video_id	INTEGER	Vanjski ključ na videozapis ili fotografiju koja je povezana s mjerom kvalitete
Accuracy	FLOAT	Iznos točnosti
Precision	FLOAT	Iznos preciznosti
Recall	FLOAT	Iznos odziva
F1	FLOAT	Iznos F1-scorea

3.4 Tablica „Quality_check“



Slika 3.1. Dijagram baze podataka

3.2.3. Baza videozapisa na udaljenom servisu Google Drive

Za spremanje videozapisa parkirališta odabran je udaljeni servis Google Drive. On će omogućiti da korisnik učita svoje videozapise, odnosno da te iste videozapise aplikacija može čitati.

Također, za spremanje obrađenih videozapisa i fotografija aplikacija će koristiti Google Drive kako bi ih spremila.

3.3. Komponenta za obradu videozapisa tehnikama računalnog vida

U ovoj komponenti našeg sustava za obradu videozapisa koristimo tehnike računalnog vida za detekciju i praćenje objekata. Konkretno, koristimo duboku neuronsku mrežu YOLOv8

(You Only Look Once, verzija 8), koja je poznata po svojoj brzini i preciznosti u detekciji objekata u stvarnom vremenu.

YOLOv8 napredna je verzija YOLO algoritma za detekciju objekata [8]. Glavne prednosti YOLOv8 uključuju [8]:

Brzina: YOLOv8 je dizajniran da obrađuje slike i videozapise u stvarnom vremenu, što je ključno za aplikacije koje zahtijevaju brzu analizu.

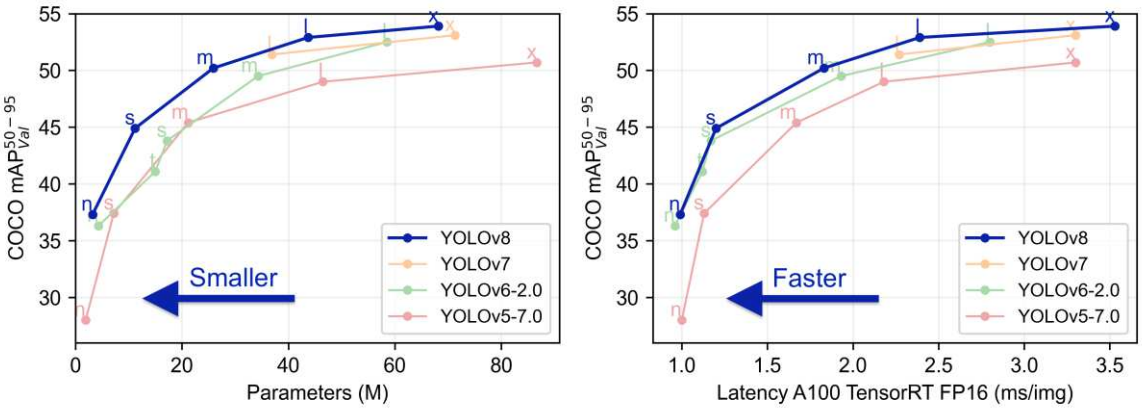
Preciznost: Poboljšane arhitekture neuronskih mreža omogućuju bolju preciznost u detekciji objekata, smanjujući broj lažnih pozitivnih i lažnih negativnih detekcija.

Efikasnost: Optimizacije u YOLOv8 omogućuju efikasnije korištenje hardverskih resursa, što je važno za implementacije na uređajima s ograničenim resursima.

Korištenjem YOLOv8 modela, ovaj sustav može efikasno detektirati i pratiti objekte u videozapisima u stvarnom vremenu. To omogućuje pouzdanu analizu i izvještavanje, što je ključno za ovu aplikaciju.

Prvi grafikon na slici 3.2 prikazuje usporedbu točnosti (eng. *mAP - mean Average Precision*) modela YOLOv8 s prethodnim YOLO modelima. YOLOv8 postiže veću točnost u detekciji objekata na skupu podataka COCO u usporedbi s YOLOv5, YOLOv6, i YOLOv7.

Drugi grafikon na slici 3.2 uspoređuje brzinu izvođenja (eng. *latency*) modela YOLOv8 s prethodnim verzijama. YOLOv8 ima nižu latenciju, što znači brže izvođenje, u odnosu na YOLOv5, YOLOv6, i YOLOv7, čineći ga efikasnijim za primjene u stvarnom vremenu.



Slika 3.2. Usporedba YOLOv8 sa starijim izvedbama [8]

4. Implementacija sustava

Dosad već opisanu aplikaciju moguće je implementirati na razne načine, no odlučeno je koristiti online implementaciju putem Google Drive-a i udaljenog radnog okruženja Google Colab, zbog mogućnosti korištenja grafičke kartice.

4.1. Python

Kao što je već spomenuto, korišteni programski jezik za implementaciju jest Python (verzija `3.10.12`), radi svoje jednostavnosti, te zbog toga što je besplatan za korištenje. Prikladan je za implementaciju složenih algoritama na jednostavan način.

Osim toga, bogat ekosustav biblioteka omogućava jednostavno rukovanje različitim aspektima razvoja sustava. Specifično, korištene su sljedeće biblioteke: *numpy, sqlite3, google.colab, IPython, ultralytics, supervision, time, os, scikit-learn*. Kombinacija ovih biblioteka omogućava razvoj sustava po zadanim specifikacijama, s fokusom na jednostavnost i efikasnost.

4.2. Ultralytics

Biblioteka otvorenog koda pomoću koje aplikacija dolazi to YOLOv8 modela. YOLOv8 je najnovija iteracija u YOLO seriji *real-time* detektora objekata, koja nudi vrhunske performanse u smislu točnosti i brzine [8]. Nadovezujući se na napretke prethodnih verzija YOLO-a, YOLOv8 uvodi nove značajke i optimizacije koje ga čine idealnim izborom za razne zadatke detekcije objekata u širokom rasponu primjena.

Važne značajke

Napredni dizajni backbone i neck: YOLOv8 koristi najmodernije dizajne backbone i neck kako bi poboljšao performanse za identifikaciju objekata i izvlačenje značajki.

Anchor-free Split Ultralytics Head: Za razliku od metoda temeljenih na sidrištima, YOLOv8 koristi anchor-free split Ultralytics head, što poboljšava točnost i pojednostavljuje proces detekcije.

Optimizirani omjer točnosti i brzine: YOLOv8 je prikladan za poslove identifikacije objekata u stvarnom vremenu u raznim područjima primjene, jer stavlja jak naglasak na očuvanje idealne ravnoteže između točnosti i brzine.

Brojni unaprijed trenirani modeli: YOLOv8 olakšava odabir idealnog modela za vašu specifičnu primjenu pružajući izbor unaprijed treniranih modela koji su prilagođeni za razne zadatke i potrebe performansi.

4.3. Supervision

Supervision je moćna biblioteka za Python koja omogućava napredne mogućnosti obrade i analize videozapisa i slika. Korištenjem Supervision biblioteke, moguće je implementirati različite tehnike računalnog vida, kao što su detekcija objekata, praćenje objekata i analiza kretanja [9].

Važne značajke

Detekcija i anotacija: Supervision omogućava anotaciju predikcija iz raznih modela za detekciju i segmentaciju objekata.

Praćenje objekata: Biblioteka nudi alate za poboljšanje video analize implementacijom besprijekornog praćenja objekata.

Detekcija malih objekata: Izvedba rješenja kako detektirati male objekte na slikama pomoću Supervision alata.

Brojanje objekata koji prelaze liniju: Metode za precizno brojanje i analizu objekata koji prelaze unaprijed definiranu liniju.

Filtriranje objekata u zoni: Tehnike za selektivno filtriranje i fokusiranje na objekte unutar specifične zone.

4.4. NumPy

Glavna Python biblioteka za znanstveno računanje naziva se NumPy. Ova biblioteka pruža višedimenzionalni niz objekata, različite izvedene objekte (poput maskiranih nizova i matrica) te razne rutine za brze operacije nad nizovima, kao što su sortiranje, odabir, ulazno-izlazne operacije, diskretne Fourierove transformacije, osnovna linearna algebra, osnovne statističke operacije, simulacije nasumičnosti i mnoge druge [10].

4.5. Opis vlastitog skupa podataka

U samoj implementaciji ovog rada korišten je vlastiti skup podataka koji je snimljen mobitelom iPhone 12, te tronožac za snimanje, visine 60 centimetara. Skup se sastoji od videozapisa parkirališnih prostora iza C zgrade FER-a, te ispred B zgrade FER-a.

Kadrovi videozapisa parkirališta ispred B zgrade snimljeni su sa zapadne strane zgrade, te s trećeg kata A zgrade. Ti videozapisi pokrivaju kadrove takozvane žablje perspektive, a drugi dio skupa pokriva kadrove polu-ptičje perspektive. Prvi dio sniman je tronošcem iz 3 kuta stubišta ulaza B zgrade, kako bi se pokrio cijeli parkirališni prostor.

Također dodatno, parkiralište je snimljeno iz A zgrade kako bi se dobio ukupni uvid u prostor.

Kadrovi videozapisa parkirališta iza C zgrade snimljeni su sa sjeverne strane zgrade, točnije s dvanaestog, osmog i šestog kata. Ovim kadrovima pokušano je dobiti oblik ptičje perspektive parkirališta.

Skup podataka učitan je na Google-ov servis *Google Drive,* te će taj isti skup u ovom radu biti korišten u obliku videozapisa i slika zaslona (eng. *screenshots*). U nastavku će biti prikazani određeni *screenshot-ovi* skupa podataka, a kasnije u vrednovanju sustava bit će prikazane specifične situacije ovog skupa podataka vezano za konkretan problem koji se obrađuje u ovom radu.

Tablica 4.1 Reprezentativni primjeri parkirališta C zgrade

Tablica 4.2 Reprezentativni primjeri parkirališta B zgrade, snimljeno na stubištu

Tablica 4.3 Reprezentativni primjeri parkirališta B zgrade, snimljeno iz A zgrade

4.6. Implementacija konvolucijske neuronske mreže

Implementacija YOLOv8 izvedena je uz veliku pomoć programskog repozitorija autora Piotra Skalskija [11] koji je razvio podršku za treniranje modela, te isporučio već istrenirane modele. U ovom radu korišteni su dijelovi tih kodova, uz naravno prilagodbu na konkretan problem ovog završnog rada.

Nadalje, ovakav odabir implementacije omogućuje korištenje dostupne grafičke kartice unutar Google Colab radnog okruženja, što će uvelike olakšati provedbu analize slika i videozapisa.

Za sami početak, potrebno je učitati Ultralytics biblioteku pomoću koje ćemo kasnije učitati konkretan YOLOv8 model.

```
!pip install ultralytics
```

```
from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

Kôd 4.1 – Kod za učitavanje Ultralytics biblioteke

Zatim je potrebno učitati konkretan YOLOv8 model, za potrebe ovog završnog rada korišten je YOLOv8x.pt. Neki od razloga korištenja upravo YOLOv8x modela u izradi ovog rada su sljedeći:

Veća točnost: YOLOv8x je dizajniran da postigne veću točnost u usporedbi s manjim varijantama kao što su YOLOv8n, YOLOv8s i YOLOv8m. Viša točnost može rezultirati boljim prepoznavanjem vozila na slikama i videozapisima parkirališta.

Veći kapacitet modela: YOLOv8x ima više parametara i slojeva, što mu omogućava bolje učenje kompleksnih značajki objekata. To je korisno za složene zadatke detekcije gdje su objekti blizu jedan drugom ili su djelomično zaklonjeni.

Prikladnost za veće resurse: Ako su dostupni veći računalni resursi (kao što je primjerice grafička kartica, ili više njih), YOLOv8x može efikasno koristiti te resurse za brže i preciznije detekcije. [13]

Model	Filenames	Task	Inference	Validation	Training	Export

YOLOv8	`yolov8n.pt`
	`yolov8s.pt`
	`yolov8m.pt`
	`yolov8l.pt`
	`yolov8x.pt`
		Detection	✅	✅	✅	✅

YOLOv8-seg	`yolov8n-seg.pt`
	`yolov8s-seg.pt`
	`yolov8m-seg.pt`
	`yolov8l-seg.pt`
	`yolov8x-seg.pt`
		Instance Segmentation	✅	✅	✅	✅

YOLOv8-pose	`yolov8n-pose.pt`
	`yolov8s-pose.pt`
	`yolov8m-pose.pt`
	`yolov8l-pose.pt`
	`yolov8x-pose.pt`
	`yolov8x-pose-p6.pt`
		Pose/Keypoints	✅	✅	✅	✅

YOLOv8-obb	`yolov8n-obb.pt`
	`yolov8s-obb.pt`
	`yolov8m-obb.pt`
	`yolov8l-obb.pt`
	`yolov8x-obb.pt`
		Oriented Detection	✅	✅	✅	✅

YOLOv8-cls	`yolov8n-cls.pt`
	`yolov8s-cls.pt`
	`yolov8m-cls.pt`
	`yolov8l-cls.pt`
	`yolov8x-cls.pt`
		Classification	✅	✅	✅	✅

Slika 4.1 Popis svih dostupnih YOLOv8 modela [8]

```
MODEL = "yolov8x.pt"
from ultralytics import YOLO

model = YOLO(MODEL)
model.fuse()
```

Kôd 4.2 – Kod za učitavanje YOLOv8x.pt modela u radno okruženje

Konkretno ovaj model bit će korišten za raspoznavanje automobila, motocikala, buseva i kamiona iz učitanih videozapisa i fotografija stvarnog svijeta. Četiri prethodno navedena vozila samo su neke od klasa za koje je ovaj model naučen, tj. istreniran. Ostatak, tj. sveukupno 80 različitih klasa moguće je vidjeti koristeći sljedeći kod.

```
CLASS_NAMES_DICT = model.model.names
# class_ids of interest - car, motorcycle, bus and truck
selected_classes = [2, 3, 5, 7]
```

```
# Print all available classes
for class_id, class_name in CLASS_NAMES_DICT.items():
    print(f"Class ID: {class_id}, Class Name: {class_name}")
```

Kôd 4.3 – Kod za ispis svih klasa koje je moguće prepoznati putem učitanog modela

4.7. Implementacija obrade fotografija i videozapisa

S obzirom da je zahtjev sustava bio da se korisniku mora omogućiti obrada videozapisa, te obrada fotografija, implementirana su dva posebna odjeljka. Svaki od navedenih ima zadatak za obraditi ili fotografije, ili videozapise.

4.7.1. Komponenta za obradu fotografija

Ova komponenta služi za obradu videozapisa i detekciju objekata unutar pojedinačnih snimaka zaslona (eng. *frame*) koristeći unaprijed istrenirani YOLOv8 model, pri čemu su sve klase objedinjene u jednu "*vehicle*" klasu radi lakše obrade podataka. Prvo, kreira se generator *frame-ova* iz zadanog videozapisa pomoću funkcije get_video_frames_generator, što omogućava iteraciju kroz svaki *frame* videozapisa. Konkretno, u ovoj implementaciji uzima se prvi *frame*. Zatim se inicijalizira objekt *BoxAnnotator* koji će služiti za dodavanje okvira i oznaka na detektirane objekte, pri čemu su parametri debljina okvira, debljina teksta i skala teksta.

Iz generatora se uzima prvi *frame* videozapisa, na kojem se provodi model za detekciju objekata. Model, unaprijed istreniran na COCO *dataset*-u, detektira objekte na *frame-u* i vraća rezultate detekcije. Rezultati se konvertiraju u format koji koristi supervision biblioteka, omogućavajući lakšu manipulaciju i analizu detekcija.

Od detekcija se zadržavaju samo one koje odgovaraju odabranim klasama objekata (automobili, motocikli, autobusi i kamioni). Svaka detekcija dobiva prilagođenu oznaku koja uključuje ime klase (sve klase objedinjene kao "*vehicle*") i razinu povjerenja u detekciju. Oznake se kreiraju u formatu koji sadrži naziv klase i stupanj povjerenja u detekciju (npr. "vehicle 0.85").

Nadalje, frame se anotira s okvirima i oznakama za detektirane objekte pomoću *BoxAnnotator* objekta, te se anotirani *frame* prikazuje koristeći matplotlib za vizualizaciju rezultata. Ovo omogućava vizualnu inspekciju detekcija i provjeru točnosti modela u stvarnom vremenu.

Dodatno, kod ispisuje ukupan broj detektiranih vozila u frameu, što omogućuje korisnicima kvantifikaciju broja detektiranih vozila u videozapisu.

```
# create frame generator
generator = sv.get_video_frames_generator(SOURCE_VIDEO_PATH)
# create instance of BoxAnnotator
box_annotator = sv.BoxAnnotator(thickness=2,
text_thickness=2, text_scale=1)
# acquire first video frame
iterator = iter(generator)
frame = next(iterator)
# model prediction on single frame and conversion to
supervision Detections
results = model(frame, verbose=False)[0]

# convert to Detections
detections = sv.Detections.from_ultralytics(results)
# only consider class id from selected_classes define above
detections = detections[np.isin(detections.class_id,
selected_classes)]

# format custom labels
labels = [
    f"{CLASS_MAPPING[class_id]} {confidence:0.2f}"
    for confidence, class_id in zip(detections.confidence,
detections.class_id)
]

# annotate and display frame
anotated_frame=box_annotator.annotate(scene=frame,
detections=detections, labels=labels)

%matplotlib inline
```

```
sv.plot_image(anotated_frame, (16,16))
# print the number of detected vehicles
num_detected_vehicles = len(detections)
print(f"Number of detected vehicles:
{num_detected_vehicles}")
```

Kôd 4.4 – Kod za obradu fotografije, odnosno *frame-a*

4.7.2. Komponenta za obradu videozapisa

Ova komponenta služi za obradu videozapisa i detekciju vozila koristeći unaprijed istrenirani YOLO model i *BYTETrack* algoritam za praćenje objekata. Na početku se kreira generator frameova iz zadanog videozapisa te se inicijaliziraju različiti anotatori za dodavanje okvira, oznaka i praćenja tragova detektiranih objekata. Također se kreira instanca *BYTETracker-a* za praćenje objekata kroz frameove.

Za svaki frame iz videozapisa, model provodi detekciju vozila, a detekcije se konvertiraju u format koji koristi supervision biblioteka. Detekcije se potom prosljeđuju BYTETrackeru za praćenje objekata kroz više frameova, a zatim se anotiraju okviri, oznake i tragovi za svako detektirano vozilo.

U okviru callback funkcije, svaki detektirani objekt se dodaje u skup jedinstvenih ID-eva vozila kako bi se prebrojila ukupna jedinstvena vozila u videozapisu. Na kraju obrade, ispisuje se ukupan broj jedinstvenih vozila te vrijeme potrebno za obradu cijelog videozapisa.

```
import time
# Create BYTETracker instance with updated arguments
byte_tracker = sv.ByteTrack(track_activation_threshold=0.5,
lost_track_buffer=30, minimum_matching_threshold=0.8,
frame_rate=30)
video_info = sv.VideoInfo.from_video_path(SOURCE_VIDEO_PATH)
generator = sv.get_video_frames_generator(SOURCE_VIDEO_PATH)

bounding_box_annotator = sv.BoundingBoxAnnotator(thickness=4)
label_annotator = sv.LabelAnnotator(text_thickness=4,
text_scale=1)
trace_annotator = sv.TraceAnnotator(thickness=4,
trace_length=50)
```

```
unique_car_ids = set()

# define call back function to be used in video processing
def callback(frame: np.ndarray, index:int) -> np.ndarray:
    #global car_count
    # model prediction on single frame and conversion to
supervision Detections
    results = model(frame, verbose=False)[0]
    detections = sv.Detections.from_ultralytics(results)
    # only consider class id from selected_classes define
above
    detections = detections[np.isin(detections.class_id,
selected_classes)]
    # tracking detections
    detections =
byte_tracker.update_with_detections(detections)

    # Add detected car ids to the set of unique car ids
    for tracker_id in detections.tracker_id:
        unique_car_ids.add(tracker_id)
    annotated_frame =
trace_annotator.annotate(scene=frame.copy(),
detections=detections)
    annotated_frame =
bounding_box_annotator.annotate(scene=annotated_frame,
detections=detections)
    annotated_frame =
label_annotator.annotate(scene=annotated_frame,
detections=detections, labels=[
        f"#{tracker_id} {model.model.names[class_id]}
{confidence:.2f}"
        for confidence, class_id, tracker_id in
zip(detections.confidence, detections.class_id,
detections.tracker_id)
    ])
    return annotated_frame
# Measure the time taken to process the video
start_time = time.time()

sv.process_video(
    source_path = SOURCE_VIDEO_PATH,
```

```
    target_path = TARGET_VIDEO_PATH,
    callback=callback
)

print(f"Total number of unique cars detected:
{len(unique_car_ids)}")
end_time = time.time()
elapsed_time = end_time - start_time
print(f"Time taken to process the video: {elapsed_time:.2f}
seconds")
```

Kôd 4.5 – Kod za obradu fotografije, odnosno *frame-a*

4.8. Implementacija baze podataka

Za samu provedbu implementacije baze podataka postoji mnogo izbora, no odabir implementacije baze podataka sveo sa na vrlo jednostavan izbor, odnosno na SQLite3. Razlog tomu bio je vrlo intuitivan uvoz i implementacija u radno okruženje Google Colab, tj. na kraju krajeva samog programskog jezika Python.

Kao što je već u ranijim poglavljima spomenuto, baza podataka se sastoji od četiri tablice koje su međusobno povezane preko vanjskih ključeva i međusobno se nadopunjuju.

```
import sqlite3
conn = sqlite3.connect('app_db.sqlite')

cursor = conn.cursor()

cursor.execute('''
CREATE TABLE IF NOT EXISTS videos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    path TEXT NOT NULL,
    import_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
''')

cursor.execute('''
```

```
CREATE TABLE IF NOT EXISTS analysis_results (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    video_id INTEGER,
    detected_objects INTEGER NOT NULL,
    number_of_parkings INTEGER NOT NULL,
    analysis_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (video_id) REFERENCES videos(id)
);
''')
cursor.execute('''
CREATE TABLE IF NOT EXISTS confusion_matrix (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    video_id INTEGER NOT NULL,
    true_positives INTEGER NOT NULL,
    false_positives INTEGER NOT NULL,
    true_negatives INTEGER NOT NULL,
    false_negatives INTEGER NOT NULL,
    FOREIGN KEY (video_id) REFERENCES videos(id)
);
''')
cursor.execute('''
CREATE TABLE IF NOT EXISTS quality_check (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    video_id INTEGER NOT NULL,
    accuracy FLOAT NOT NULL,
    precision FLOAT NOT NULL,
    recall FLOAT NOT NULL,
    f1 FLOAT NOT NULL,
    FOREIGN KEY (video_id) REFERENCES videos(id)
);
''')
conn.commit()
conn.close()
```

Kôd 4.4 – Kod za učitavanje baze podataka u radno okruženje

5. Analiza i vrednovanje sustava

Kako bi se dobio dojam koliko je ovakav sustav zapravo koristan i upotrebljiv, potrebno je provesti vrednovanje istog. Za vrednovanje razvijenog sustava za detekciju objekata korištenjem YOLOv8 modela, provedena je validacija na zasebnom skupu podataka.

Izazov koji se javio tokom provedbe analize jest detekcija slobodnih parkirnih mjesta. S obzirom da YOLOv8 nije istreniran da samostalno prepoznaje slobodna parkirna mjesta, korisnik aplikacije morao bi imati unaprijed označiti parkirna mjesta za svaki skup podataka što dosta otežava obradu jer ovisno o kutu i konfiguraciji parkirališta model neće biti u mogućnosti ispravno prepoznavati vozila i parkirna mjesta.

No, za početak opišimo skup podataka koji je korišten za treniranje YOLOv8x mreže koja se koristi u ovom radu.

5.1. Skup podataka COCO

Opsežan skup podataka (eng. *dataset*) za detekciju objekata, segmentaciju i dodavanje opisa naziva se COCO (eng. *Common Objects in Context*). Često se koristi za evaluaciju modela računalnog vida i namijenjen je za poticanje istraživanja u širokom spektru kategorija objekata. Za istraživače i *developer-e* koji rade na zadacima vezanim uz detekciju objekata, segmentaciju i procjenu položaja, ovaj skup podataka je neophodan.

Ključne značajke

COCO sadrži 330.000 slika, pri čemu 200.000 slika ima anotacije za zadatke detekcije objekata, segmentacije i dodavanja opisa (opisivanja konteksta slike riječima). Preostalih 130.000 slika koje nisu anotirane koriste se za različite svrhe, uključujući dodatno testiranje modela, korištenje neoznačenih podataka, pozadinske slike i augmentaciju podataka.

Skup podataka obuhvaća 80 kategorija objekata, uključujući uobičajene objekte poput automobila, bicikala i životinja, kao i specifičnije kategorije kao što su kišobrani, torbice i sportska oprema.

Anotacije uključuju granice objekata, maske za segmentaciju i opise za svaku sliku.

COCO pruža standardizirane metrike evaluacije poput srednje prosječne preciznosti za detekciju objekata i srednje prosječnog odziva za zadatke segmentacije, što ga čini pogodnim za usporedbu performansi modela. [5]

Struktura *dataset*-a

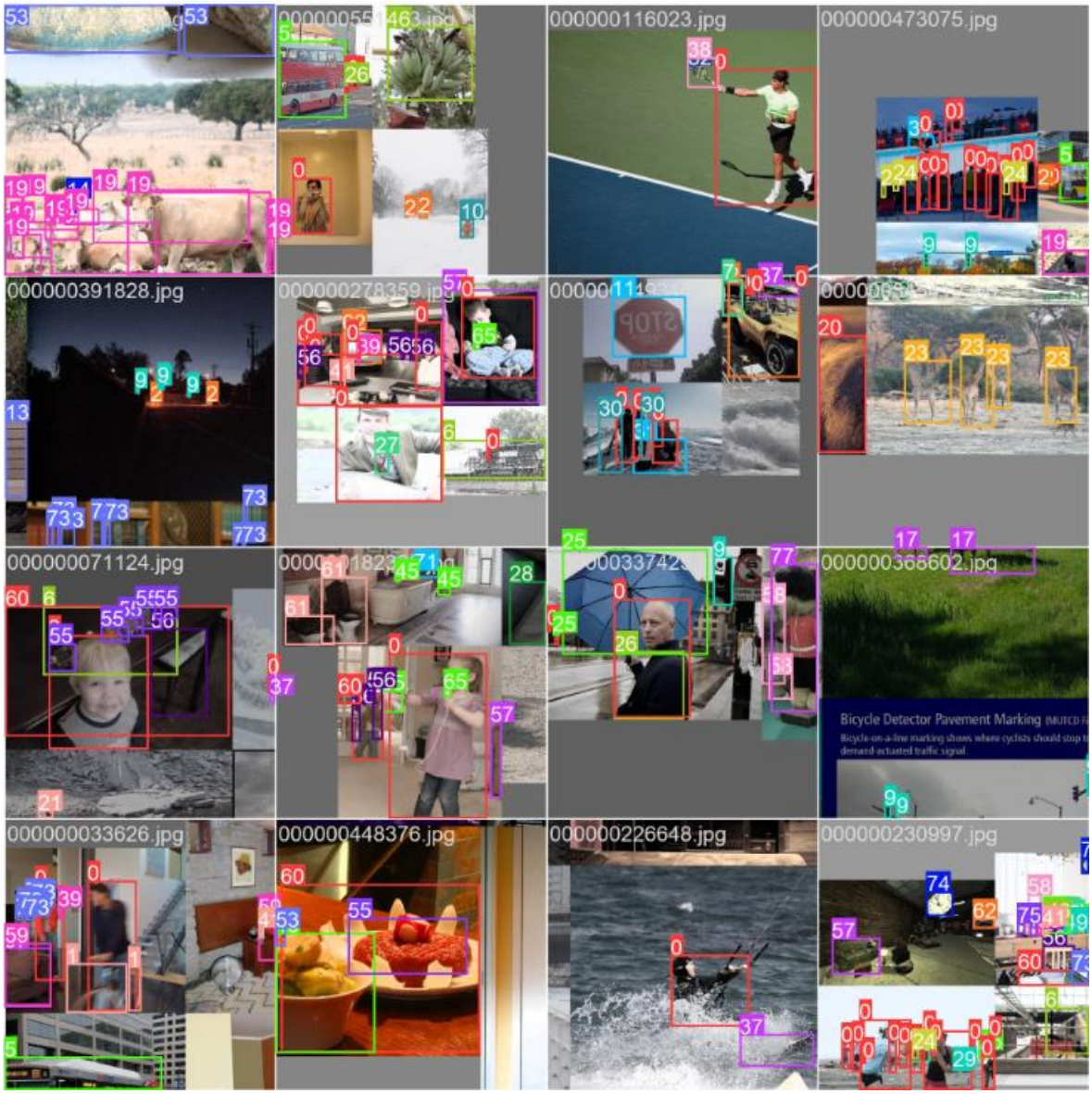
Iz COCO *dataset-a* izdvojena su tri podskupa:

Train2017: Ovaj podskup uključuje 118 tisuća fotografija koje se koriste za treniranje modela za segmentaciju, označavanje i identifikaciju objekata.

Val2017: Tijekom treniranja modela, 5 tisuća fotografija iz ovog podskupa koristi se za validaciju.

Test2017: U ovom podskupu nalazi se 20 tisuća fotografija koje se koriste za procjenu i testiranje treniranih modela. Rezultati se šalju na COCO evaluacijski poslužitelj radi procjene performansi, a istinite oznake (eng. *ground truth*) za ovaj podskup nisu dostupne javnosti.

Prethodno navedena struktura *dataset-a* uobičajena je za skupove za treniranje u umjetnoj inteligenciji. Cijeli *dataset* podijeljen je na tri dijela, od kojih svaki ima svoju ulogu u dobivanju modela koji je naučen za specifični problem.



Slika 5.1 Reprezentativni primjeri *COCO dataset-a* [4]

5.2. Metode vrednovanja detekcije objekata

Vrednovanje sustava izvršeno je pomoću sljedećih metrika:

Preciznost (eng. *precision*): Omjer točno predviđenih pozitivnih primjera u odnosu na sve predviđene pozitivne primjere.

Odziv (eng. *recall*): Omjer točno predviđenih pozitivnih primjera u odnosu na sve stvarne pozitivne primjere.

F1-mjera (eng. *F1-score*): Harmonijska sredina preciznosti i odziva, koja daje uravnoteženu mjeru performansi sustava.

Točnost (eng. *accuracy*): Omjer točno predviđenih primjera (i pozitivnih i negativnih) u odnosu na ukupan broj primjera.

Također, procesu vrednovanja, bitno je razumjeti osnovne komponente koje čine metrike:

***True Positives* (TP)**: Ispravno identificirani pozitivni primjeri. To su slučajevi kada model ispravno prepozna prisutnost objekta u slici.

***False Positives* (FP)**: Neispravno identificirani pozitivni primjeri. To su slučajevi kada model pogrešno prepozna objekt koji nije prisutan u slici.

***True Negatives* (TN)**: Ispravno identificirani negativni primjeri. To su slučajevi kada model ispravno prepozna odsutnost objekta u slici.

***False Negatives* (FN)**: Neispravno identificirani negativni primjeri. To su slučajevi kada model ne prepozna objekt koji je prisutan u slici.

5.3. Analiza obrađenih fotografija vlastitog skupa podataka

S obzirom da je za obradu videozapisa potrebno mnogo vremena zbog svakog pojedinačnog *frame-a,* a videozapisi su bili snimani sa 60 sličica u sekundi te u prosječnom trajanju od oko 15 sekundi, za vrednovanje sustava provedena je analiza nad snimkama zaslona (eng. *screenshots*) početnih *frame-ova* videozapisa.

U nastavku bit će prikazani *screenshot*-i obrađenih fotografija koji su korišteni za analizu.

5.3.1. Parkiralište ispred B zgrade, žablja perspektiva

Za početak, prikazat ćemo rezultate parkirališta ispred B zgrade, koristeći prvu, žablju perspektivu. Naime, problem koji se ovdje stvorio jest da je zbog ograničenosti perspektive

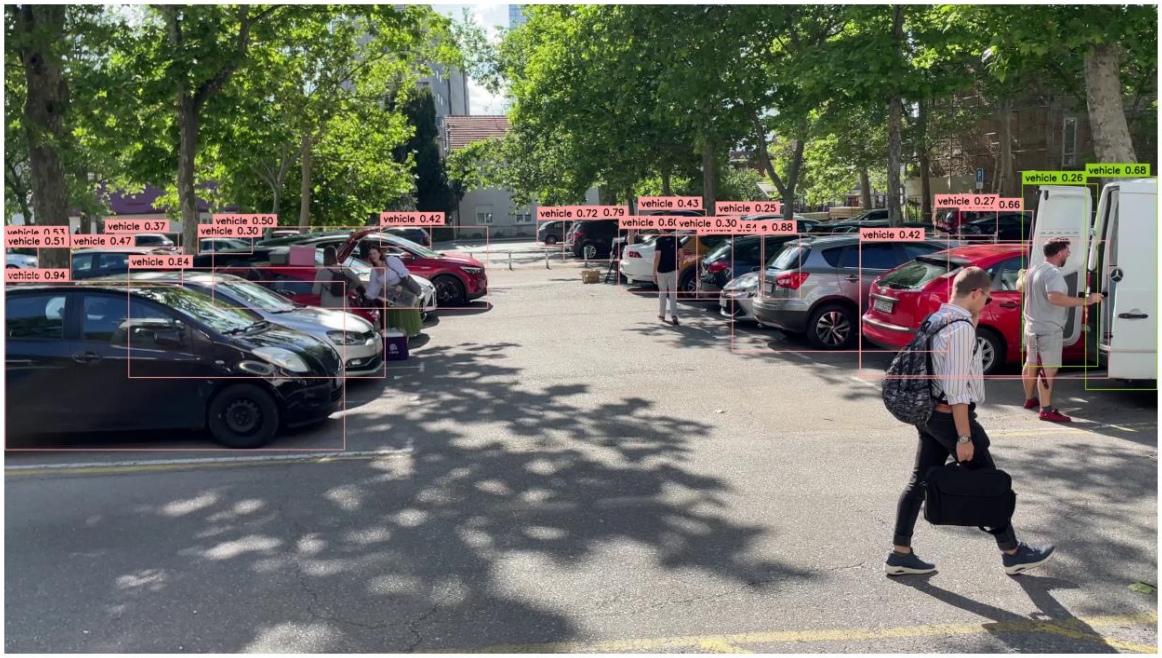
modelu bilo poprilično teško raspoznati vozila koja su na krajnjim rubovima parkirališta, čak do te razine da model uopće nije prepoznavao neka vozila. Također, izazov koji je model imao jest taj da u neke očite prikaze vozila nije bio siguran s velikim postotkom.

Spletom okolnosti, tokom snimanja parkirališta, na samom početku parkirališta nalazio se kombi kojeg je model označio kao 2 vozila, a u stvarnosti su tom kombiju bila otvorena vrata prtljažnika. Opisanu situaciju moguće je vidjeti na slici 5.3.

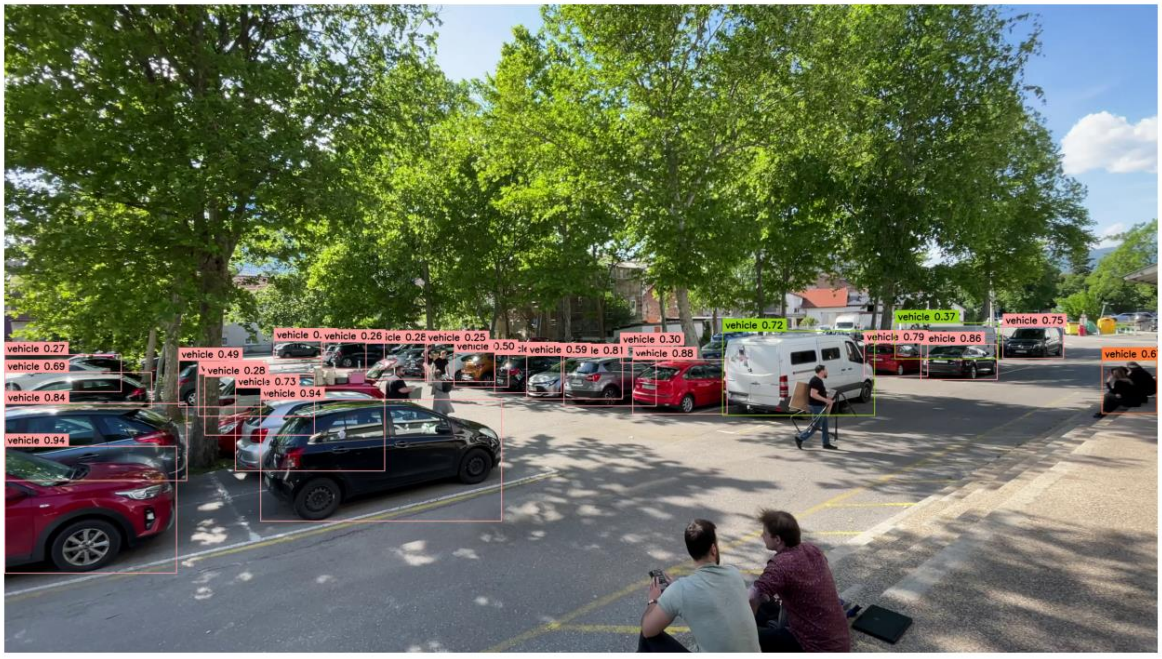
Još jedna pogreška koja je nastala u obradi jest ta da je model u jednoj situaciji skup od dvoje ljudi koji sjede na stubištu označio kao vozilo. Opisanu situaciju moguće je vidjeti na slici 5.4.

Spomenute situacije upućuju na to da model nije dovoljno dobro istreniran na situacije kada je perspektiva kamere niska.

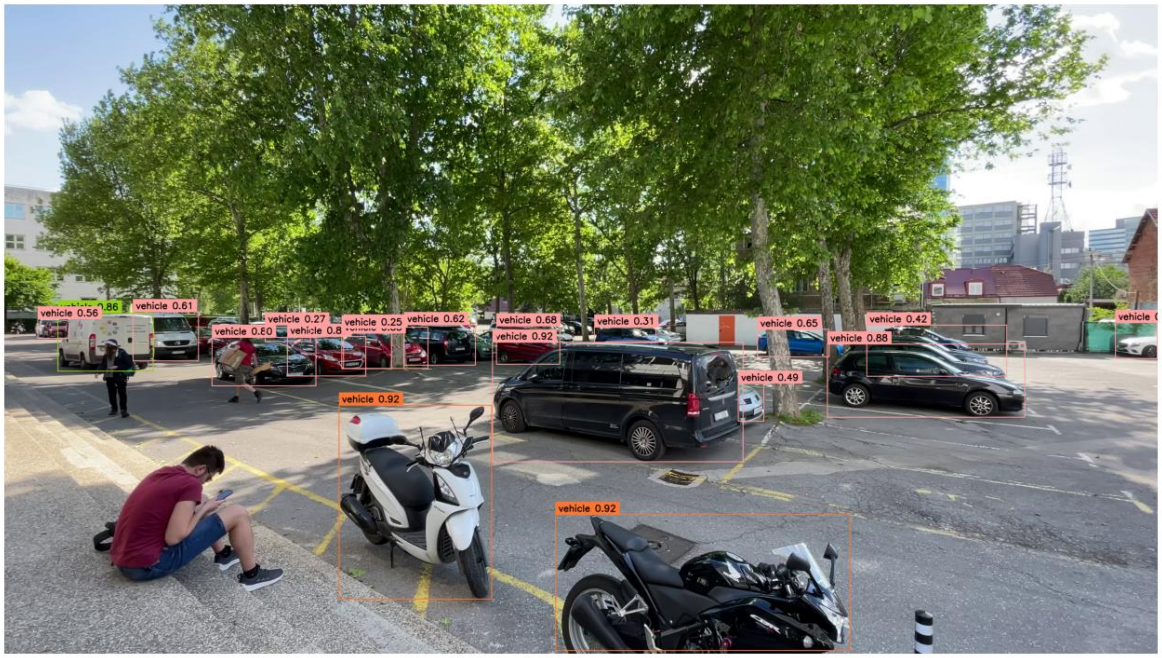
Slika 5.2 Prikaz obrade parkirališta ispred B zgrade, žablja perspektiva, kadar prvi



Slika 5.3 Prikaz obrade parkirališta ispred B zgrade, žablja perspektiva, kadar drugi



Slika 5.4 Prikaz obrade parkirališta ispred B zgrade, žablja perspektiva, kadar treći, širokokutna kamera



Slika 5.5 Prikaz obrade parkirališta ispred B zgrade, žablja perspektiva, kadar četvrti, širokokutna kamera

- Matrica konfuzije:

	Predicted Positive (Vehicle)	Predicted Negative (No Vehicle)
Actual Positive (Vehicle)	21	13
Actual Negative (No Vehicle)	1	8

- Accuracy (Točnost): 0.6744
- Precision (Preciznost): 0.9545
- Recall (Odziv): 0.6176
- F1-Score: 0.7491

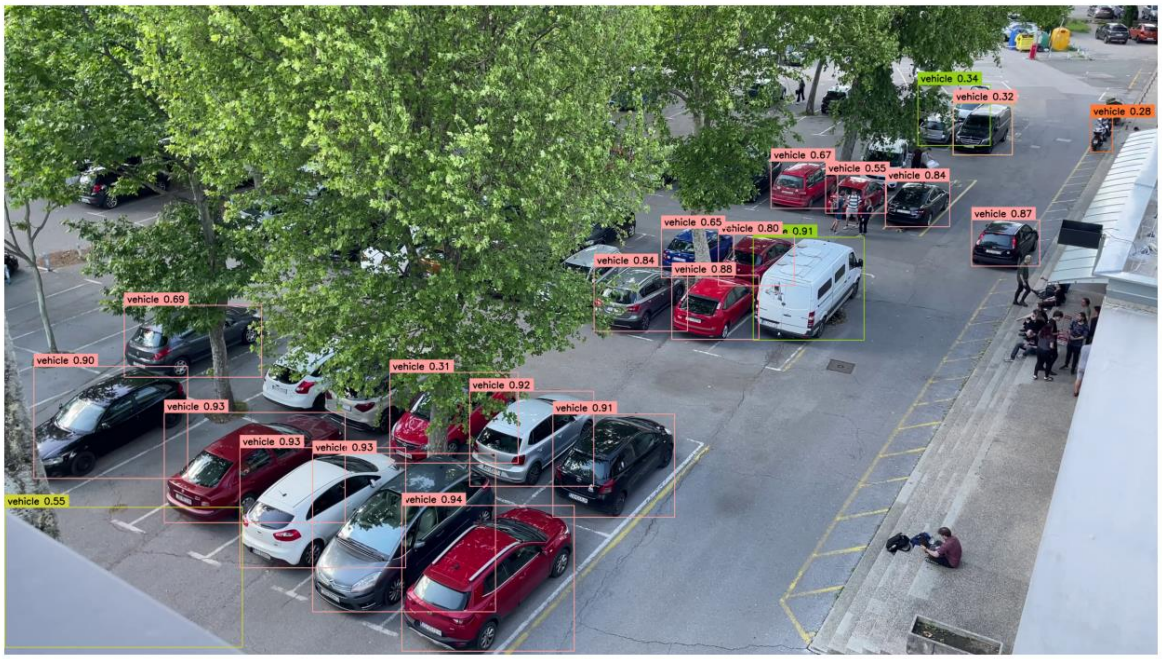
Slika 5.6 Matrica konfuzije i mjere kvalitete performansi za situaciju na slici 5.3

5.3.2. Parkiralište ispred B zgrade, ptičja perspektiva

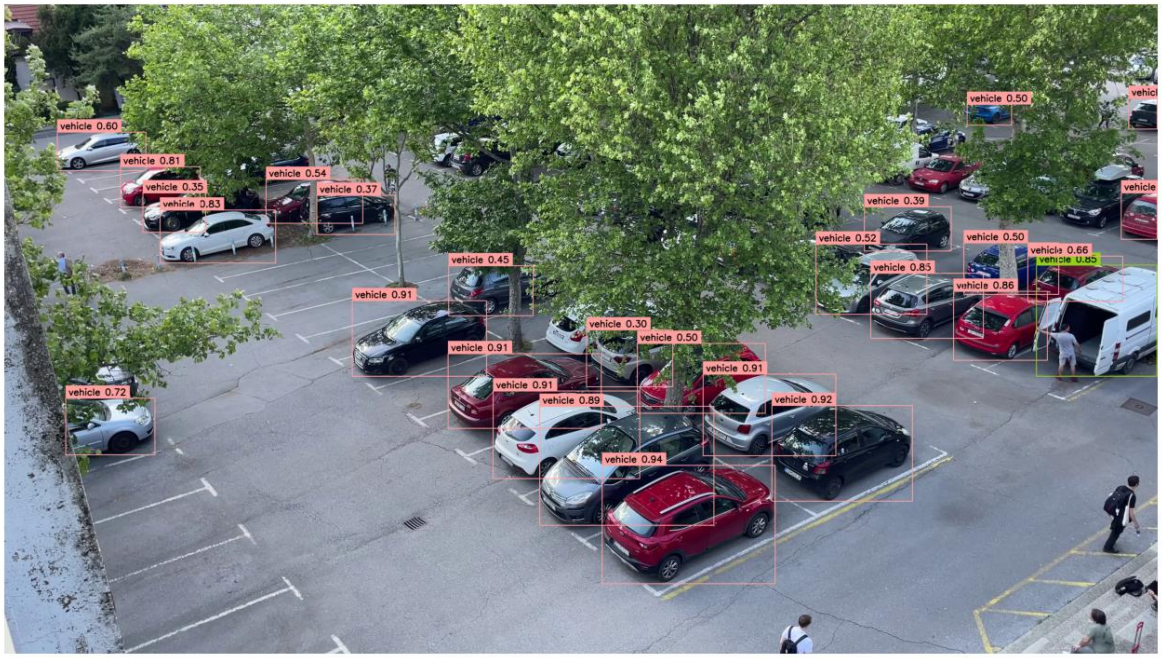
Kako bi se poboljšao rezultat obrade, podignuta je perspektiva na treći kat A zgrade. Pretpostavljeno je da će viša perspektiva donijeti bolje rezultate, što je i bio slučaj.

Model je bio sigurniji, te je uspio prepoznati skoro pa sva vozila koja su u tom trenutku bila na parkiralištu. Problem mu je stvaralo drveće koje je pokrivalo određeni prostor parkirališta.

No, u jednoj situaciji ponovno se dogodio problem. Naime, model je u jednom trenutku rub zida A zgrade klasificirao kao vozilo, što dovodi do pitanja „Je li perspektiva problem ili model nije dovoljno dobro istreniran da prepoznaje vozila?“. Spomenuto je moguće vidjeti na slici 5.6.



Slika 5.7 Prikaz obrade parkirališta ispred B zgrade, ptičja perspektiva, kadar prvi



Slika 5.8 Prikaz obrade parkirališta ispred B zgrade, ptičja perspektiva, kadar drugi

- Matrica konfuzije:

	Predicted Positive (Vehicle)	Predicted Negative (No Vehicle)
Actual Positive (Vehicle)	21	27
Actual Negative (No Vehicle)	1	25

- Accuracy (Točnost): 0.6216
- Precision (Preciznost): 0.9545
- Recall (Odziv): 0.4375
- F1-Score: 0.6

Slika 5.9 Matrica konfuzije i mjere kvalitete performansi za situaciju na slici 5.7

5.3.3. Parkiralište ispred C zgrade, ptičja perspektiva

Kako bi se utvrdila hipoteza da viša perspektiva zadaje bolji rezultat obrade, provedeno je snimanje parkirališta iza C zgrade. Videozapisi su prikupljeni sa šestog, osmog, te na kraju dvanaestog kata.

Za početak pogledajmo rezultate snimki sa šestog kata. Naime, hipoteza se pokazuje istinitom, model prepoznaje gotovo pa sva vozila na slici, ništa što nije vozilo nije označio pod vozilo. Možemo pretpostaviti da će povećanjem visine perspektive rezultat obrade biti sve bolji i bolji.

No kod ove perspektive također se pokazuje problem drveta koje pokriva gotovo pa pola kadra, što smanjuje prikaz trenutne stvarne sitaucije na parkiralištu.

Slika 5.10 Prikaz obrade parkirališta ispred C zgrade, ptičja perspektiva, kadar prvi

- **Matrica konfuzije**:

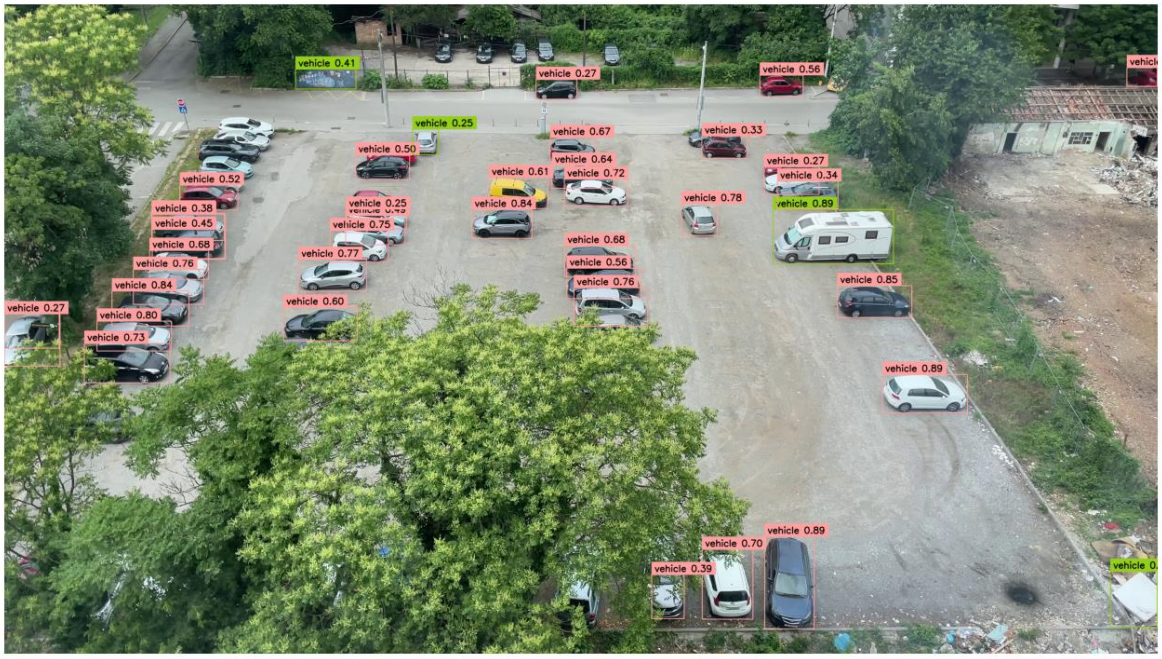
	Predicted Positive (Vehicle)	Predicted Negative (No Vehicle)
Actual Positive (Vehicle)	34	40
Actual Negative (No Vehicle)	0	50

- **Accuracy (Točnost)**: 0.6774
- **Precision (Preciznost)**: 1.0
- **Recall (Odziv)**: 0.4595
- **F1-Score**: 0.6296

Slika 5.10 Matrica konfuzije i mjere kvalitete performansi za situaciju na slici 5.10

Kako bi se potvrdila hipoteza da viši kadar dovodi do boljeg rezultata obrade, za sljedeći dio analize kamera je premještena na osmi kat C zgrade. Tu već vidimo da model dolazi do malo lošijih rezultata nego sa šestog kata, te dolazi sumnja da je kadar previsok, te da su vozila postala presitna.

Model ne prepoznaje vozila u gornjem lijevom kutu parkirališta, te vozila na vrhu fotografije, odnosno nekoliko vozila izvan parkirališta.



Slika 5.12 Prikaz obrade parkirališta ispred C zgrade, ptičja perspektiva, kadar drugi

- **Matrica konfuzije**:

	Predicted Positive (Vehicle)	Predicted Negative (No Vehicle)
Actual Positive (Vehicle)	37	22
Actual Negative (No Vehicle)	2	65

- **Accuracy (Točnost)**: 0.8095
- **Precision (Preciznost)**: 0.9487
- **Recall (Odziv)**: 0.6271
- **F1-Score**: 0.7550

Slika 5.13 Matrica konfuzije i mjere kvalitete performansi za situaciju na slici 5.12

Za sami kraj analize, obrađene su snimke s dvanaestog kata C zgrade. Ovdje je već moguće vidjeti da model podosta slabo prepoznaje vozila, gotovo pa do te mjere da trećinu vozila koje bi ljudsko oko očito prepoznalo nije uspio prepoznati.

Slika 5.14 Prikaz obrade parkirališta ispred C zgrade, ptičja perspektiva, kadar treći

- **Matrica konfuzije**:

	Predicted Positive (Vehicle)	Predicted Negative (No Vehicle)
Actual Positive (Vehicle)	38	58
Actual Negative (No Vehicle)	0	35

- **Accuracy (Točnost)**: 0.5573
- **Precision (Preciznost)**: 1.0
- **Recall (Odziv)**: 0.3958
- **F1-Score**: 0.5667

Slika 5.15 Matrica konfuzije i mjere kvalitete performansi za situaciju na slici 5.14

Zaključak

U ovom završnom radu, analizirana je implementacija konvolucijske neuronske mreže za detekciju i praćenje vozila u stvarnom vremenu, te je provedena analiza i evaluacija različitim tehnikama.

Korištenje unaprijed istreniranog modela i algoritma za praćenje objekata omogućilo je brzu i učinkovitu obradu videozapisa, što je ključno za primjene u stvarnom vremenu poput prometnog nadzora i sigurnosti cesta. Osim toga, sustav je pokazao visoku razinu skalabilnosti i prilagodljivosti za različite scenarije i okoline.

Rezultati analize provedene iz različitih perspektiva pružaju uvid u učinkovitost i ograničenja predloženog sustava za detekciju i praćenje vozila. Iz žablje perspektive, primijećeno je da model nije uspješno prepoznavao sva vozila, posebno kada su bila jedno iza drugoga. Ovo sugerira da postoji prostor za poboljšanje u detekciji objekata sličnih oblika i veličina.

Povećanjem perspektive na treći kat primijećeno je poboljšanje u rezultatima, iako su i dalje postojale nepravilnosti. S druge strane, s visine od šestog kata, situacija se značajno poboljšala, sugerirajući da je udaljenost kamere od parkirališta ključan faktor za preciznost detekcije.

Međutim, kako se visina perspektive podizala, rezultati su postajali lošiji zbog povećane udaljenosti kamere od promatranog područja. Ove informacije sugeriraju da postoji omjer između udaljenosti kamere od parkirališta i pruženih performansi.

U skladu s rezultatima istraživanja, preporučuje se daljnje istraživanje i proširenje sustava za integraciju s naprednijim tehnologijama poput dubokog učenja s pojačanjem ili obradom prirodnog jezika. Ovaj istraživački rad pruža temelje za daljnji napredak u području detekcije i praćenja objekata, što ima široku primjenu u područjima kao što su prometna sigurnost, analitika prometa i autonomna vožnja.

Literatura

[1] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 (Version 8.0.0). Retrieved from https://github.com/ultralytics/ultralytics

[2] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[3] Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. Computational intelligence and neuroscience, 2018(1), 7068349..

[4] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., … Dollár, P. (2015). Microsoft COCO: Common Objects in Context. arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/1405.0312

[5] Jocher, G., Munawar, R., & Q, L. (2024, June 2). *Coco*. COCO - Ultralytics YOLO Docs. https://docs.ultralytics.com/datasets/detect/coco/

[6] Doljanin, D. (2021). Računalni sustav za procjenu položaja tijela temeljen na biblioteci OpenPose (Završni rad). Zagreb: Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva. Preuzeto s https://urn.nsk.hr/urn:nbn:hr:168:307946

[7] Jelečević, Lj. (2021). Aplikacija za kontinuiranu detekciju osoba u videosignalu (Undergraduate thesis). Zagreb: University of Zagreb, Faculty of Electrical Engineering and Computing. Retrieved from https://urn.nsk.hr/urn:nbn:hr:168:982466

[8] Jocher, G., Q, B., Q, L., Fcakyon, & Exel, A. (2024, June 2). *Yolov8*. Ultralytics YOLO Docs. https://docs.ultralytics.com/models/yolov8

[9] Roboflow. (2024, March 28). Supervision. https://supervision.roboflow.com/latest/#quickstart

[10] *NumPy documentation#*. NumPy documentation - NumPy v1.26 Manual. (n.d.). https://numpy.org/doc/stable/

[11] https://github.com/roboflow/notebooks

[12] Horvat, M., Jelečević, L., & Gledec, G. (09 2022). A comparative study of YOLOv5 models performance for image localization and classification.

[13] Sohan, M., Sai Ram, T., & Rami Reddy, C. V. (2024). A Review on YOLOv8 and Its Advancements. In I. J. Jacob, S. Piramuthu, & P. Falkowski-Gilski (Eds.), Data Intelligence and Cognitive Informatics (pp. 529–545). Singapore: Springer Nature Singapore.

Sažetak

Razvoj aplikacije za detekciju slobodnih mjesta na parkiralištu korištenjem dubokog učenja

Detekcija vozila postupak je računalnog vida pomoću korištenja alata poput dubokih neuronskih mreža. Postoje različite mreže koje je moguće primijeniti na ovakav ili sličan problem, a u ovom radu korištena je duboka neuronska mreža YOLOv8.

U implementaciji ovog rješenja, korišteno je radno okruženje Google Colab, te Python programski jezik uz biblioteku *ultralytics* i *upervision*.

Aplikacija omogućava učitavanje fotografija i videozapisa s Google Drive-a, procjenu položaja vozila na fotografijama, te spremanje rezultata nazad na Google Drive.

Analiza je provedena pomoću standardne konfuzijske matrice, te pratećih mjera kvalitete performansi: točnost (engl. *accuracy*), odziv (engl. *recall*), preciznost (engl. *precision*) i *F1-score*.

Ključne riječi: konvolucijske neuronske mreže, duboko učenje, računalni vid, detekcija parking mjesta, YOLOv8

Summary

Development of deep learning-based application for detecting vacant parking spaces

Vehicle detection is a computer vision process using tools like deep neural networks. There are different networks that can be applied to this or a similar problem, and in this work the YOLOv8 deep neural network was used.

In the implementation of this solution, the Google Colab working environment and the Python programming language with the Ultralytics and Supervision library were used.

The application allows uploading photos and videos from Google Drive, estimating the position of the car in the photos, and saving the results back to Google Drive.

The analysis was carried out using the standard confusion matrix, and accompanying performance quality measures: accuracy, recall, precision and F1-score.

Keywords: convolutional neural networks, deep learning, computer vision, parking space detection, YOLOv8