Algoritam za izdvajanje putokaza iz scene i razumijevanje teksta na njima

Čordašić, Ivan

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

Sveučilišni studij

ALGORITAM ZA IZDVAJANJE PUTOKAZA IZ SCENE I RAZUMIJEVANJE TEKSTA NA NJIMA

Diplomski rad

Ivan Čordašić

Osijek, 2023. godina

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 19.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Ivan Čordašić
Studij, smjer:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. Pristupnika, godina upisa:	D-60ARK, 06.10.2021.
OIB studenta:	88705497149
Mentor:	prof. dr. sc. Mario Vranješ
Sumentor:	,
Sumentor iz tvrtke:	Zvonimir Kaprocki
Predsjednik Povjerenstva:	prof. dr. sc. Marijan Herceg
Član Povjerenstva 1:	prof. dr. sc. Mario Vranješ
Član Povjerenstva 2:	izv. prof. dr. sc. Ratko Grbić
Naslov diplomskog rada:	Algoritam za izdvajanje putokaza iz scene i razumijevanje teksta na njima
Znanstvena grana diplomskog rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	U današnjim vozilima dostupni su moderni navigacijski sustavi koji vozaču pomažu u određivanju putanje prilikom dolaska na željenu lokaciju. No i dalje su od velike koristi putokazi koji upućuju vozača u određenom smjeru, pogotovo u izvanrednim situacijama kada nekakve iznenadne situacije u prometu nisu još popraćene pravovremenim ažuriranjem u navigacijskim mapama. U sklopu ovog diplomskog rada potrebno je izraditi računalni algoritam koji prepoznaje putokaze u prometu (žute, zelene, crvene, plave,...), te prepozna tekst na putokazu, kao i smjer na koji određeni putokaz ukazuje. Za detekciju putokaza i teksta na njima mogu se koristiti
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene od strane mentora:	19.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 26.09.2023.

Ime i prezime studenta:	Ivan Čordašić
Studij:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. studenta, godina upisa:	D-60ARK, 06.10.2021.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom **: Algoritam za izdvajanje putokaza iz scene i razumijevanje teksta na njima**

izrađen pod vodstvom mentora prof. dr. sc. Mario Vranješ

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Posljednjih godina automobilska industrija prolazi kroz revolucionarne promjene, potaknute napretkom tehnologije i razvojem umjetne inteligencije (*engl. Artificial Intelligence - AI*). To je utrlo put novoj eri prijevoza, u kojoj vozila mogu obavljati funkcije vožnje uz minimalnu ili nikakvu interakciju s vozačem. Autonomna vozila postupno postaju stvarnost našeg svakodnevnog života. Krajnji cilj automobilske industrije je razviti potpuno autonomna vozila koja su sposobna postići razinu 5 autonomije vožnje. Ova razina autonomije podrazumijeva da vozilo može samostalno voziti bez ikakve ljudske intervencije, obavljajući sve zadatke vožnje u svim uvjetima na cesti ili vremenskim uvjetima. Kako bi se postigao ovaj ambiciozan cilj, mnoštvo komponenti i sustava moraju besprijekorno raditi zajedno. To uključuje fuziju senzora, algoritme autonomne vožnje i aktuatore koji izvršavaju potrebne radnje za autonomno upravljanje vozilom.

Jedan ključni čimbenik omogućavanja autonomije vozila je napredni sustav pomoći vozaču (*engl. Advanced Driver Assistance System - ADAS*). ADAS [1] je računalni sustav koji kontrolira različite aspekte rada vozila i zahtijeva opsežnu količinu informacija za percipiranje i razumijevanje okruženja. Vizija, kao primarno osjetilo za ljude, također je važna za autonomna vozila. Shodno tome, kamere igraju ključnu ulogu u pružanju ključnih vizualnih podataka ADAS-u. Konstantnim snimanjem slika s prednje kamere vozila, sustav može steći sveobuhvatno znanje o objektima ispred njega, olakšavajući pravovremeno donošenje odluka i osiguravajući sigurnu i učinkovitu autonomnu vožnju.Tako npr., autonomno vozilo da bi se pravilno orijentiralo u prometu, mora imati sposobnost točnog detektiranja smjera kretanja koji je naznačen na prometnim putokazima te razumijevanja teksta koji piše na istima.

Vozači mogu izbjeći potencijalne opasnosti, donositi promišljene odluke i kretati se kroz zamršene cestovne mreže točnim prepoznavanjem i razumijevanjem teksta na putokazima. U domeni algoritama [2] za pomoć vozačima, točna lokacija i razumijevanje znakova igraju krucijalnu ulogu u poboljšanju sigurnosti na prometnicama i smanjivanju broja nesreća, jer omogućuju vozačima da ostanu informirani o osnovnim podacima prikazanim na prometnim putokazima. Putokazi su obično prikazani u različitim formatima, bojama, veličinama te na različitim mjestima uz cestu, dok tekst koji je napisan na njima može biti izazovan za pročitati zbog veličine slova te vremenskih otegotnih okolnosti, što sve skupa dovodi do osjetljivog problema.

U ovom radu predložen je računalni algoritam koji detektira putokaz u prometu te prepoznaje tekst na putokazu, kao i smjer na koji određen putokaz ukazuje. U radu je bilo potrebno realizirati

algoritam koristeći metode računalnog vida i/ili strojnog učenja. Algoritam je izrađen u Python programskom jeziku na Windows operacijskom sustavu te je njegova implementacija u konačnici testirana na osobnom računalu i na računalnoj ugradbenoj (*engl. embedded)* platformi.

Struktura preostalog dijela rada organizirana je na sljedeći način. U drugom poglavlju opisane su komponente od kojih se problem ovoga rada sastoji, te su predložene i opisane metode za rješavanje problema. Također, analizirana su postojeća rješenja za neke od tih problema. Nakon toga, u trećem poglavlju objašnjen je cjelovit proces izrade vlastitog računalnog algoritma za detekciju putokaza, smjera i razumijevanje teksta na putokazima. Četvrto poglavlja opisuje način testiranja rada algoritma te rezultate testiranja na odabranom skupu slika. Zaključak je napisan u petom, završnom poglavlju rada.

2. PROBLEM DETEKCIJA PUTOKAZA I RAZUMIJEVANJA TEKSTA NA NJIMA

U domeni računalnog vida i autonomnih sustava, detekcija prometnih znakova, smjerova na njima te naknadno prepoznavanje teksta na njima predstavljaju značajne izazove koji zahtijevaju sofisticirane algoritme i tehnike. Ovaj problem je od iznimne važnosti jer prometni znakovi prenose vitalne informacije vezane uz prometna pravila, propise, upozorenja i upute. Točno otkrivanje i razumijevanje ovih znakova ključni su za omogućavanje autonomnim vozilima da donose ispravne odluke i osiguravaju sigurnu navigaciju na cestama.

Problem se sastoji od tri komponente. Prva komponenta problema je detekcija samog putokaza. Otkrivanje putokaza na slikama složen je zadatak zbog različitih čimbenika kao što su varijacije u uvjetima osvjetljenja, promjene u veličini ili udaljenosti objekta i prostorni šum, odnosno prisutnost irelevantnih objekata. Učinkovite metode detekcije moraju biti sposobne točno i učinkovito locirati prometne putokaze, čak i u izazovnim scenarijima. Druga komponenta problema je detekcije smjera unutar samog putokaza, što također može biti izazovno iz istih razloga koji vrijede za detekciju samog putokaza. Treća komponenta problema uključuje prepoznavanje teksta sadržanog u detektiranom putokazu. Izdvajanje i razumijevanje tekstualnih informacija ključno je za autonomna vozila kako bi protumačila i prikladno odgovorila na prenesene poruke. Ovaj zadatak zahtijeva primjenu OCR (*engl. Optical character recognition - OCR*) [3] tehnike za pretvaranje vizualnog prikaza teksta u strojno čitljive tekstualne podatke. Ovaj problem predstavlja veliki izazov za autonomne sustave jer je sigurnost uvijek na prvome mjestu. Na slici 2.1 prikazan je primjer koji idealno rješava navedeni problem. Svaki detektirani objekt (putokaz, smjer) prikazan je unutar graničnog pravokutnika (*engl. Bounding box*) te mu je dodijeljena ispravna klasa, dok se pročitani tekst nalazi iznad graničnog pravokutnika putokaza.

U potpoglavljima 2.1 i 2.2 detaljnije su opisane komponente od kojih se navedeni problem sastoji, predložene su metode za rješavanje problema i navedena postojeća rješenja.

2.1. Detekcija objekata u slikama

Detekcija objekata jedna je od ključnih oblasti računalnog vida. Osnovni cilj detekcije objekata je detektirati objekte unutar slike, te klasificirati svaki detektirani objekt u odgovarajuće klase. Ovo se postiže pomoću algoritama i tehnika koje analiziraju piksele slike i traže obrasce koji odgovaraju poznatim klasama objekata. Nedavno je detekcija objekata doživjela značajan napredak, a područje se pretežno pomiče prema pristupima koji se zasnivaju na strojnom učenju.

Ove metode iskorištavaju snagu algoritama dubokog učenja, posebice konvolucijskih neuronskih mreža (*engl. Convolutional Neural Network - CNN*) [4], za automatsko učenje i izdvajanje relevantnih značajki iz slika. Ova promjena paradigme dovela je do značajnih poboljšanja u točnosti i robusnosti detekcije. Osim spomenutih CNN zasnovanih na dubokom učenju, implementirano je nekoliko različitih metoda i arhitektura za detekciju objekata u slikama. Viola-Jones algoritam [5] za detekciju objekata i HOG (*engl. Histograms of Oriented Gradients - HOG*) [6] metoda predstavljaju klasične pristupe detekcije objekata. Međutim, s razvojem dubokog učenja, danas se sve više koriste metode zasnovane na dubokom učenju koje postižu veću preciznost. Osim toga, postoje i metode segmentacije slike koje mogu izdvojiti objekte na temelju njihovih prostornih karakteristika. Također, detekcija objekata zasnovana na izdvajanju značajki (*engl. Feature-based object detection*) [7] koristi razne karakteristike, poput boje, teksture ili oblika, kako bi detektirala objekte. Ovi različiti pristupi pružaju širok spektar tehnika za detekciju objekata s različitim prednostima i primjenama, ali i nedostacima.

2.1.1. You Only Look Once algoritam

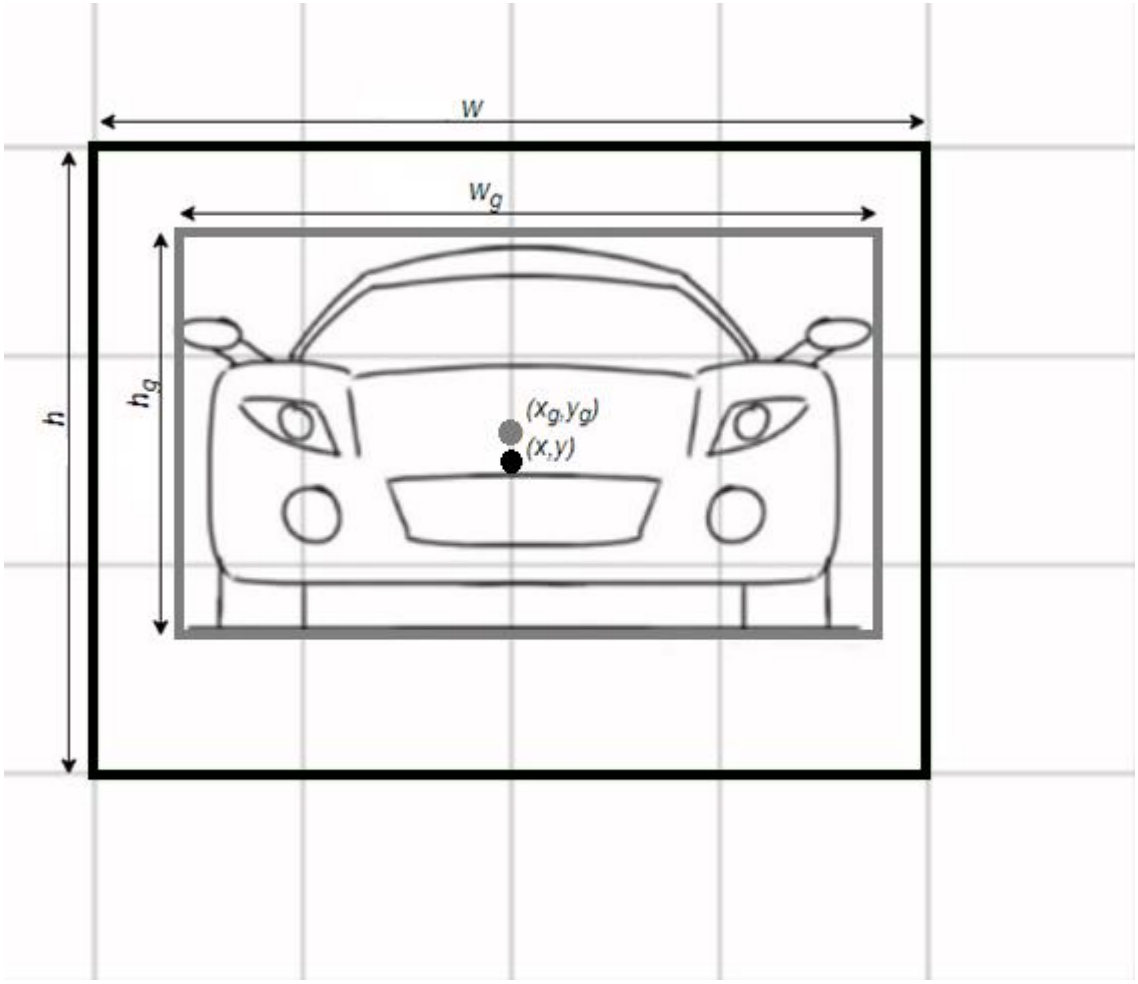
YOLO (*engl. You Only Look Once -YOLO*) algoritam, predstavljen 2016. [8], nudi moćan pristup detekciji objekata u slikama. Revolucionirao je polje detekcije objekata formuliranjem zadatka kao regresijskog problema i objedinjavanjem komponenti detekcije u jednu neuronsku mrežu. YOLO algoritam usvaja pristup zasnovan na mreži, dijeleći ulaznu sliku u matricu ćelija.

Unutar svake ćelije matrice, YOLO predviđa određeni broj graničnih pravokutnika, također poznatih kao granični okviri, popraćenih vjerojatnošću koja predstavlja vjerojatnost da je objekt prisutan unutar tog određenog graničnog pravokutnika. U idealnom slučaju, vjerojatnost bi trebala biti blizu nule ako ćelija ne sadrži objekt od interesa, a u suprotnom blizu jedinice. Algoritam koristi koncept presjeka preko unije (*engl. Intersection over Union - IoU*) [9], koji mjeri preklapanje između predviđenog graničnog pravokutnika i stvarnog graničnog pravokutnika, kako bi se odredila vjerojatnost predviđanja (2-1).

$$IoU = \frac{A \cap B}{A \cup B}, \tag{2-1}$$

gdje A predstavlja površinu detektiranog graničnog okvira, a B površinu stvarnog graničnog okvira (*engl. ground truth*) koji je označen prije početka treninga. Svaki granični pravokutnik donosi nekoliko parametara predikcije: predviđanje vjerojatnosti, te četiri parametara koji su označeni s *x, y, w, h*. Oznake *x* i *y* se koriste za označavanje koordinata središta graničnog pravokutnika, dok *w* i *h* označavaju dimenzije detektiranog objekta, to jest, njegovu širinu i visinu

koje su izražene kao relativne vrijednosti u rasponu od 0 do 1 u odnosu na širinu i visinu cijele slike (slika 2.1.). Vjerojatnost predikcije je zapravo *IoU* između predviđenog pravokutnika s parametrima x, y, w, h (na slici 2.1. prikazan crno) i stvarnog pravokutnika s parametrima x_g, y_g, w_g, h_g (na slici 2.1. prikazan sivo). Na slici 2.1., *IoU* iznosi 0,75.

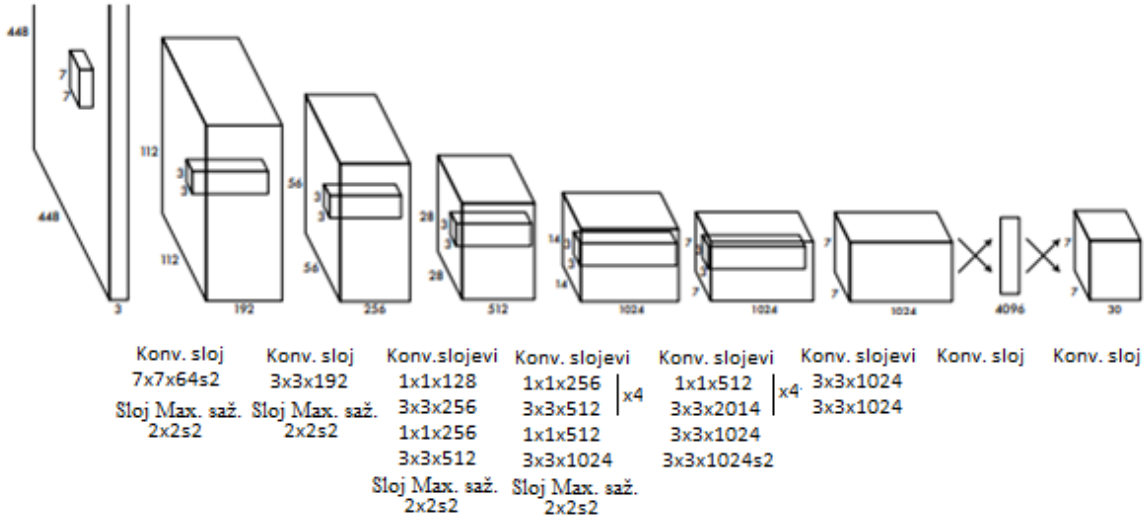


Slika 2.1. *Prikaz stvarnog graničnog pravokutnika i predviđenog graničnog pravokutnika* [10]

Korištenjem CNN i iskorištavanjem njihove sposobnosti izdvajanja bitnih značajki, YOLO učinkovito bilježi važne informacije potrebne za robusnu detekciju objekata. YOLO algoritam koristi CNN arhitekturu koja se sastoji od konvolucijskih slojeva koji izvlače značajke iz ulazne slike i potpuno povezanih slojeva odgovornih za predviđanje izlaznih vjerojatnosti i koordinata graničnog okvira. CNN koju koristi izvorni YOLO model, prikazana na slici 2.2., uključuje 24 konvolucijska sloja, nakon kojih slijede dva potpuno povezana sloja (*engl. Fully connected layer*) [11].

U ovom modelu koristi se tehnika učenja koja radi s cjelovitim slikama. Prije početka procesa učenja potrebno je prilagoditi nekoliko parametara, od kojih su najvažniji:

- Brzina učenja (*engl. Learning rate)* – hiperparametar koji se koristi za određivanje brzine učenja parametara tijekom treninga [12].
- Broj klasa na izlazu – taj broj će ovisiti o tome koliko različitih klasa postoji.



Slika 2.2. *Prikaz arhitekture izvornog YOLO modela* [13]

Model je pripremljen korištenjem organiziranog rasporeda označenih slika. Sve slike skupa podataka šalju se kroz CNN nakon što im se promijeni dimenzija na 448 x 448 piksela na početku procesa treninga. U procesu treninga, svaki granični okvir zadužen je samo za jedan objekt. Originalni YOLO model istreniran je na Pascal VOC 2007 bazi podataka [14]. Kako je prikazano u tablici 2.1., u usporedbi s drugim metodama i arhitekturama dubokog učenja koje se koriste za detekciju objekata, jedino su dva YOLO modela predstavljena u [9] sposobna za rad u stvarnom vremenu, odnosno mogu procesirati više od 30 okvira u sekundi (*engl. Frames per second - FPS*) na NVIDIA GeForce Titan X GPU (*engl. Graphics processing unit – GPU*), uz nižu prosječnu preciznost po klasama (*engl. mean Average Precision - mAP*). Prosječna preciznost detektora objekata mjeri se *mAP*-om. Izračunava se tako da se prvo izračuna srednja vrijednost pojedinačnih preciznosti, a zatim prosječna preciznost za svaku klasu. Osim *mAP*-a, različite metrike kao što su točnost, odziv, preciznost i rezultat F1 dodatno se koriste za vrednovanje modela. Preciznost [15] procjenjuje udio točno identificiranih objekata (*engl. True positives - TP*) u odnosu na ukupan broju objekata detektiranih kao pozitivni primjeri (2-2). Uz *TP*, postoje još:

- FP (*engl. False positive*) – detekcije koje predstavljaju objekt, a objekta nema na tom mjestu,

- FN (*engl. False negative*) – model ne detektira objekt, a objekt postoji na tom mjestu,
- TN (*engl. True negative*) – model ne detektira objekt, a objekta i stvarno nema na tom mjestu.

Tablica 2.1. *Usporedbe performansi različitih modela za detekciju objekata* [20]

NAZIV MODELA	BRZINA OBRADE U FPS	mAP	MOGUĆNOST OBRADE U STVARNOM VREMENU (DA/NE)
Fast YOLO [9]	155.00	52.70 %	DA
YOLO [9]	45.00	63.40 %	DA
YOLO VGG-16 [16]	21.00	66.40 %	NE
Fast R-CNN [17]	0.50	70.00 %	NE
Faster R-CNN VGG-16 [18]	7.00	73.20 %	NE
Faster R-CNN ZF [19]	18.00	62.10 %	NE

Omjer [15] ispravno otkrivenih objekata prema ukupnom broju objekata koji su trebali biti otkriveni naziva se odziv (*engl. Recall – R*) (2-3). Omjer broja točnih odluka i ukupnog broja svih odluka naziva se točnost (2-4) [15]. F1 [15] rezultat je metrika koja objedinjuje preciznost i odziv uzimajući njihovu harmonijsku sredinu (2-5).

$$preciznost = \frac{TP}{TP+FP} \tag{2-2}$$

$$odziv = \frac{TP}{TP+FN} \tag{2-3}$$

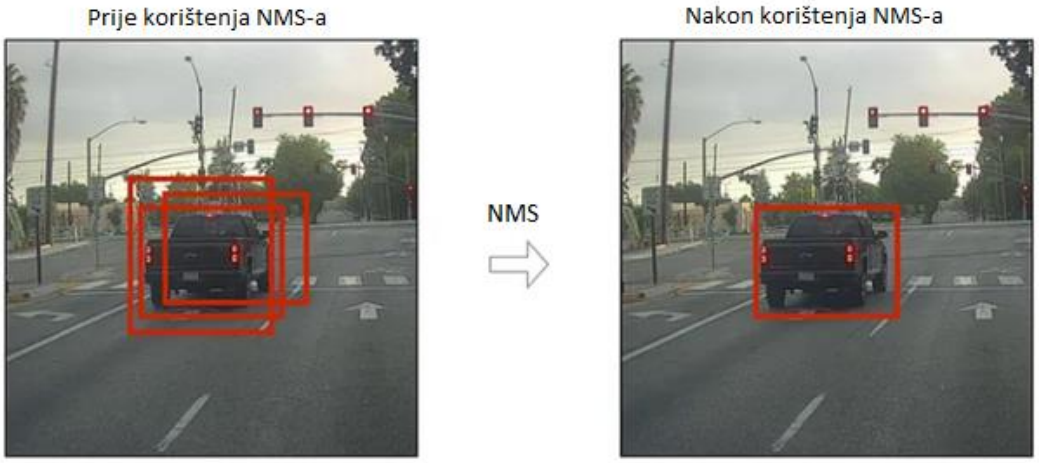
$$točnost = \frac{TP+TN}{TP+FP+FN+TN} \tag{2-4}$$

$$F1 = 2 * \frac{preciznost*odziv}{preciznost+odziv} \tag{2-5}$$

Unatoč svom uspjehu, izvorni model YOLO ima nekoliko ograničenja. Generaliziranje objekata u različitim okruženjima ili objekata s različitim omjerima širine i visine može predstavljati izazov jer se model prvenstveno trenira na određenom skupu podataka. Prostorna ograničenja unutar svake mrežne ćelije ograničavaju broj susjednih objekata koji se mogu točno otkriti, ograničavajući sposobnost algoritma da identificira i locira više objekata na slici, posebno u situacijama s više objekata koji su blizu jedan drugome. Nadalje, kriterijska funkcija jednako

tretira pogreške u predviđanju malih i velikih graničnih pravokutnika, što potencijalno dovodi do lošijih rezultata.

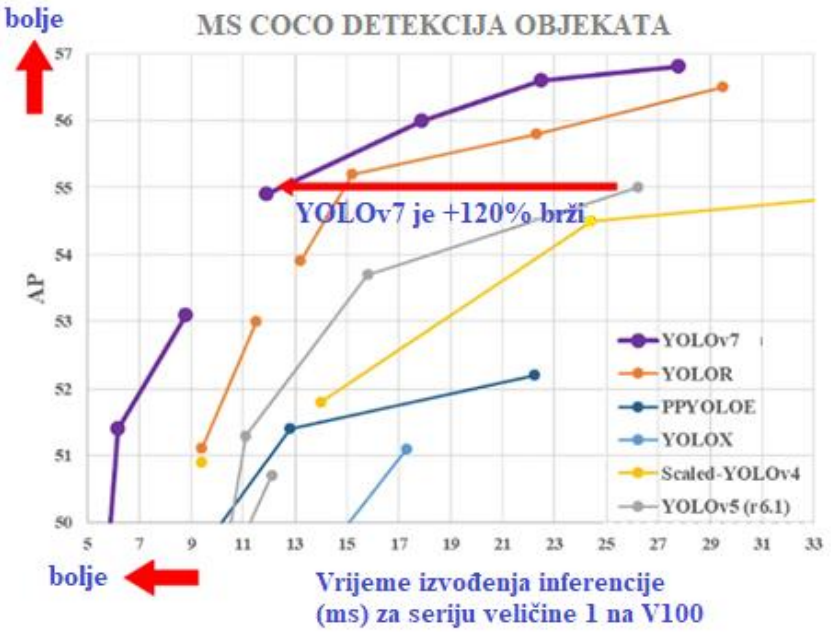
Potiskivanje ne-maksimalne vrijednosti (*engl. Non-Maximum Supression - NMS*) [21] je važna tehnika u procesu detekcije objekata YOLO algoritma. Korištenjem NMS-a rješava se problem otkrivanja velikih objekata u više ćelija. NMS algoritam počinje odabirom detekcije s najvećom vjerojatnošću i proglasi ju objektom. Zatim uspoređuje tu detekciju s ostalim detekcijama, izračunavajući njihov međusobni *IoU* kako bi izmjerio njihovo preklapanje. Ako *IoU* premašuje određeni prag, detekcija s manjom vjerojatnošću se odbacuje te se proces nastavlja sve dok ima detekcija. Rezultat svega toga je samo jedan granični okvir oko objekta od interesa, a ne mnoštvo njih, kako bi bilo da se ne koristi navedena tehnika. Slika 2.3. ilustrira učinkovitost primjene NMS-a. Na lijevoj slici detektirano je više automobila s različitim vjerojatnostima detekcije, ali oni predstavljaju isti automobil u stvarnosti. Primjenom NMS-a zadržava se granični okvir s najvećom vjerojatnošću detekcije, poboljšavajući točnost i eliminirajući suvišne detekcije.



Slika 2.3. *Primjer primjene NMS-a* [22]

Previđeni granični okviri (*engl. Anchor box*) igraju ključnu ulogu u najsuvremenijim sustavima detekcije objekata, pa tako i u YOLO-u. Ovi sustavi generiraju previđene granične okvire za svaki prediktor, predstavljajući očekivanu lokaciju, oblik i veličinu objekta za čiju su detekciju specijalizirani. Proces uključuje izračunavanje *IoU-a* između svakog previđenog graničnog okvira i stvarnog graničnog okvira objekta. Ako najveći izračunati *IoU* prelazi 50%, previđeni granični okvir treba detektirati objekt povezan s tim *IoU*. U slučajevima kada najveći izračunati *IoU* padne između 40% i 50%, otkrivanje se smatra dvojbenim, a neuronska mreža tada ne uči iz tog konkretnog primjera. Međutim, ako je najveći izračunati *IoU* ispod 40%, predviđeni granični okvir predviđa odsutnost objekta u toj regiji. Ovaj se pristup pokazao učinkovitim u praksi, budući da brojni prediktori točno određuju postoji li na slici njihova specifična vrsta objekta.

Unutar domene YOLO algoritma postoji više verzija, svaka sa svojim poboljšanjima u odnosu na verziju prije nje. Za potrebe ovog rada koristit će se verzija YOLOv7 [23]. U vrijeme izrade praktičnog dijela rada, YOLOv7 predstavljao je najnoviju i najnapredniju verziju, koja uključuje različita poboljšanja i optimizacije. Performanse YOLOv7 procijenjene su na temelju prethodnih verzija YOLOv4 [24], YOLOv5 [25] i YOLOR-a [26]. YOLOv7 je pokazao najbolji omjer brzine i točnosti među uspoređenim modelima detekcije objekata. Općenito, YOLOv7 nadmašuje sve dosadašnje detektore objekata u pogledu brzine i točnosti, u rasponu od 5 FPS-a do čak 160 FPS-a. YOLO v7 algoritam postiže najveću točnost među svim ostalim modelima detekcije objekata u stvarnom vremenu (slika 2.4.), dok postiže 30 FPS ili više koristeći GPU V100.



Slika 2.4. *Usporedba YOLOv7 modela sa starijim verzijama YOLO modela* [27]

Arhitektura YOLOv7 [28] nadovezuje se na napredak prethodnih YOLO modela, uključujući YOLOv4, Scaled YOLOv4 i YOLOR. Uvodi nekoliko ključnih značajki i poboljšanja za unaprijeđenje performansi i mogućnosti modela. Jedna važna komponenta YOLOv7 je E-ELAN (*engl. Extended Efficient Layer Aggregation Network - E-ELAN*). E-ELAN koristi razne tehnike za poboljšanje sposobnosti učenja mreže bez ometanja gradijentnog toka. To omogućuje YOLOv7 da kontinuirano poboljšava svoj kapacitet učenja. Kako bi zadovoljio zahtjeve različitih aplikacija, YOLOv7 uključuje složeno skaliranje modela. Ovaj pristup omogućuje neovisnu prilagodbu atributa modela kao što su širina (broj kanala), dubina (broj stupnjeva) i razlučivost (veličina ulazne slike). Za razliku od tradicionalnih pristupa, koji faktore skaliranja razmatraju zajedno,

YOLOv7 složena metoda skaliranja čuva optimalnu strukturu modela zadržavajući svojstva početnog dizajna. YOLO arhitektura sastoji se od kičme (*engl. backbone*), vrata (*engl. neck*) i glave (*engl. head*). U YOLOv7, glava sadrži predviđene izlaze modela. Model koristi više glava, uključujući glavnu glavu odgovornu za konačni rezultat i pomoćnu glavu koja se koristi za pomoć u obuci u srednjim slojevima. Ovaj pristup poboljšava proces treninga i ukupnu izvedbu duboke mreže. Tehnika poznata kao ponovna parametrizacija uključuje izračunavanje prosjeka skupa težina modela kako bi se proizveo model koji je otporniji na općenite obrasce koje pokušava modelirati.

ADAS u praksi ima ograničene računalne resurse. Kao rezultat toga, razvijena je manja verzija YOLOv7 poznata kao YOLOv7-tiny. Ova verzija koristi manje memorije i posjeduje manje parametara. Ovaj je model sličan izvornom ubrzanom modelu YOLO po tome što ima niži *mAP*, ali može obraditi više okvira u sekundi. I YOLOv7 i YOLOv7-tiny koristit će se pri izradi vlastitog računalnog algoritma za detekciju objekata u ovom radu, kao što je opisano u trećem poglavlju.

2.1.2. Dodatna postojeća rješenja za detekciju objekata

Detekcija objekata igra ključnu ulogu u računalnom vidu, omogućujući računalima da prepoznaju i lociraju objekte unutar slika ili videa. Tijekom godina, uvođenjem inovativnih pristupa i tehnika u ovom su području napravljeni značajni pomaci, posebno u području prometnih znakova. U radu [29] predstavljena je metoda za detekciju i klasifikaciju prometnih znakova u video zapisima. U ovom radu se ističe upotreba HOG i SVM (*engl. Support Vector Machine - SVM*) tehnika, zajedno s CNN za prepoznavanje prometnih znakova. Proces detekcije i klasifikacije sastoji se od dvije glavne faze. Prvo se primjenjuje HSV (*engl. Hue Saturation Value - HSV*) tehnika segmentacije boja kako bi se očistila slika od šuma, a zatim se koristi SVM za detekciju znakova. U drugoj fazi, CNN se koristi za klasifikaciju detektiranih prometnih znakova. Sustav je testiran na velikom broju video okvira iz različitih okruženja kako bi se osigurala sveobuhvatna evaluacija sustava. Skup podataka za treniranje modela dobiven je iz GTSRB (*engl. German Traffic Sign Recognition Benchmark - GTSRB*) [30] , gdje su prometni znakovi svrstani u 43 kategorije. Iako rad ne pruža podatke o točnosti algoritma, prednosti ovog rješenja uključuju snažan i ekonomičan sustav za detekciju znakova, koji se može implementirati u vozila uz potrebne nadogradnje. Međutim, važno je napomenuti da će algoritam raditi samo s njemačkim prometnim znakovima, budući da su samo oni korišteni za treniranje modela. Također, u radu se ne spominje korištenje algoritama u lošim vremenskim uvjetima kao što su magla ili loše osvjetljenje. U [31] je predstavljena metoda dubokog učenja za prepoznavanje i detekciju prometnih znakova, koja donosi nekoliko poboljšanja u odnosu na prethodno opisani pristup. Glavna novost je korištenje

poboljšane verzije LeNet-5 CNN arhitekture. Autori su analizirali postojeću arhitekturu i unaprijedili je korištenjem Gabor kernela za provjeru efikasnosti algoritma. Gabor kernel je linearni filter koji se koristi za analizu teksture i pokazao se uspješnim u ovom kontekstu. Metoda se zasniva na slikama dobivenim s kamere montirane na vozilu. Slike prolaze kroz postupak predprocesiranja, koji uključuje konverziju u HSV prostor boja i odbacivanje nepotrebnih dijelova slike. Također se primjenjuju tehnike poboljšanja slike zbog uklanjanja šuma i poboljšanja same kvalitete. Testiranje rješenja je provedeno u stvarnom vremenu, tijekom vožnje vozila. Rezultati su impresivni, s točnošću algoritma koja prelazi 99% (s manjim odstupanjima za različite vrste znakova), dok je prosječna brzina obrade okvira 8 ms. Ovaj algoritam ima visoku učinkovitost i stopu prepoznavanja, uz značajno smanjenje vremenskog intervala obrade okvira. Autori u [32] istražuju primjenu najnovijeg modela u YOLO obitelji modela, YOLOv7, za otkrivanje bolesti biljaka. Autori su koristili model obučen na skupu podataka PlantDoc [33] i postigli značajan napredak u odnosu na prethodno korištene modele iz YOLO obitelji za isti problem. YOLOv7 je postigao visoku preciznost od 72,8%, odziv od 68,5% i *mAP* od 71%. Unatoč postignutom napretku, izvedba modela i dalje nije bila potpuno zadovoljavajuća zbog raznolikosti skupa podataka koji je sadržavao slike lišća oboljelog i nezaraženog iz različitih prirodnih okruženja.

2.1.3. Detekcija putokaza u slikama

Otkrivanje putokaza ključan je korak u raznim aplikacijama, uključujući autonomnu vožnju, analizu prometa i navigacijske sustave. Točna i pouzdana detekcija putokaza (slika 2.5.) ključna je za osiguranje sigurnosti i učinkovitosti ovih sustava. Detekcija putokaza i smjera na slikama sadrži nekoliko izazova zbog čimbenika kao što su različiti uvjeti osvjetljenja, loša kvalitete slike s kamere u automobilu, složena pozadine te različiti oblici i veličine putokaza (prikazano na slici 2.6.). Stoga je razvoj robusnih i učinkovitih algoritama za detekciju od vitalnog značaja. Algoritam korišten za detekciju putokazu i smjera u ovom radu detaljno je opisan u dijelu 2.1.1., dok generalni proces obično uključuje sljedeće korake [34]:

- Predobrada podataka: ulazna slika prolazi kroz tehnike predobrade kako bi se poboljšala vidljivost prometnih znakova. Uobičajeni koraci predobrade uključuju promjenu rezolucije slike, podešavanje kontrasta i uklanjanje šuma ili nevažnih detalja.
- Izdvajanje značajki: Različite značajke izdvajaju se iz prethodno obrađene slike kako bi se dohvatile karakteristike putokaza i smjera. Ove značajke mogu uključivati oblik,

Slika 2.5. *Primjer točno detektiranih objekata od interesa te pročitanog teksta*

boju, teksturu ili druge vizualne atribute koji izdvajaju objekte od interesa od pozadine i drugih objekata.

- Klasifikacija: Izdvojene značajke zatim se koriste za treniranje modela klasifikacije koji može razlikovati putokaze i smjer od objekata koji to nisu. Tehnike klasifikacije, kao što su algoritmi strojnog učenja ili modeli dubokog učenja (*engl. Deep learning)*, koriste se za klasifikaciju izdvojenih značajki i utvrđivanje kojoj klasi pripada detektirano područje.
- Naknadna obrada: Nakon što se klasifikacijskim modelom identificiraju područja koja će vjerojatno sadržavati objekte od interesa, primjenjuju se tehnike naknadne obrade za pročišćavanje i filtriranje rezultata. Ove tehnike, npr., mogu uključivati NMS kako bi se uklonile detekcije koje se preklapaju.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

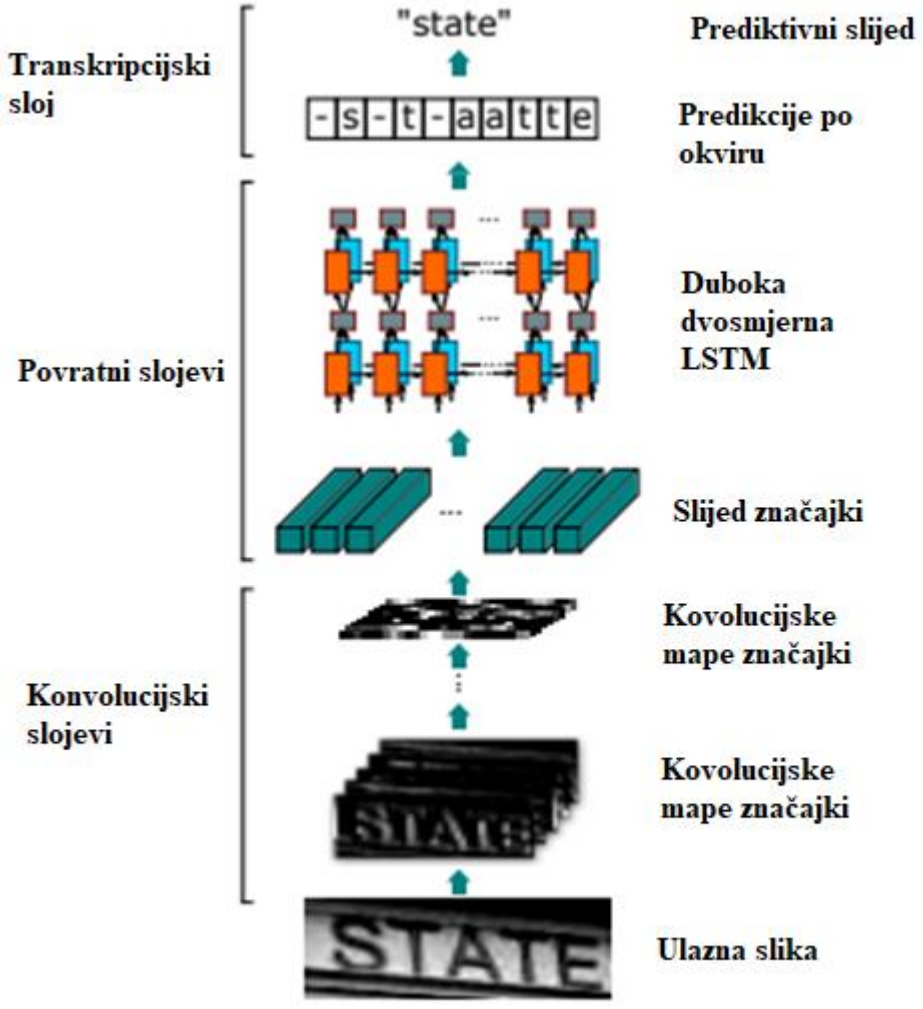
Slika 2.6. *Primjeri slika ((a), (b), (c), (d), (e), (f), (g), (h)) objekata od interesa koje je potrebno ispravno detektirati i klasificirati u sklopu rada. Slike prikazuju objekte u različitim bojama i uvjetima (osvjetljenje, loša kvaliteta slike).*

2.2. Prepoznavanje teksta u izdvojenom putokazu

Prepoznavanje teksta, također poznato kao OCR (*engl. Optical Character Recognition - OCR*), jedan je od osnovnih zadatak u području računalnog vida i obrade prirodnog jezika. Uključuje proces pretvaranja tiskanog ili rukom pisanog teksta na slikama u strojno čitljiv tekst. Prepoznavanje teksta ima brojne primjene, uključujući digitalizaciju dokumenata, izdvajanje teksta iz slika, automatsko prepoznavanje registarskih pločica i još mnogo toga. Problem prepoznavanja teksta može biti prilično izazovan zbog varijacija u fontovima, stilovima,

veličinama, usmjerenjima, uvjetima osvjetljenja i pozadini. Osim toga, prisutnost šuma i izobličenja dodatno komplicira zadatak. Međutim, napredak u tehnikama dubokog učenja i računalnog vida značajno je poboljšao točnost i robusnost sustava za prepoznavanje teksta.

Jedan od popularnih pristupa rješavanju problema prepoznavanja teksta je kombinirano korištenje CNN i povratnih neuronskih mreža (*engl. Recurrent neural network - RNN*), što zajednički tvori CRNN (*engl. Convolutional Recurrent neural network - CRNN*), prikazano na slici 2.7. [35]. CNN-ovi su učinkoviti u izvlačenju značajki iz ulaznih slika, dok su RNN-ovi, posebno mreže dugog kratkoročnog pamćenja (*engl. Long short-term memory - LSTM*) [36], prikladne za modeliranje sekvencijskih podataka kao što je tekst.



Slika 2.7. *Arhitektura CRNN mreže građena od tri dijela: konvolucijski slojevi, povratni slojevi te transkripcijski sloj*

Proces prepoznavanja teksta obično uključuje sljedeće korake:

- Lokalizacija teksta: U početku je potrebno lokalizirati područja teksta na ulaznoj slici. To se može učiniti pomoću tehnika kao što je detekcija teksta, gdje se identificiraju područja koja vjerojatno sadrže tekst.
- Segmentacija teksta: Nakon što se identificiraju područja teksta, pojedinačne znakove ili retke teksta potrebno je segmentirati kako bi se izolirali za daljnju obradu.
- Izvlačenje značajki: U ovom koraku značajke se izdvajaju iz segmentiranih tekstualnih područja pomoću CNN-ova. CNN-ovi se mogu unaprijed obučiti na skupovima podataka velikih razmjera, kao što je skup podataka ImageNet, kako bi naučili generičke vizualne reprezentacije.
- Modeliranje sekvence: Izdvojene značajke zatim se unose u RNN, obično LSTM mrežu, koja modelira sekvencijalne ovisnosti unutar teksta. To omogućuje mreži da uhvati kontekstualne informacije i poboljša točnost prepoznavanja.
- Dekodiranje: Izlaz LSTM mreže dekodira se u konačni prikaz teksta.

2.2.1. PaddleOCR

PaddleOCR [37], izgrađen na platformi za duboko učenje PaddlePaddle, OCR je alat otvorenog koda koji ima za cilj pružiti najsuvremenije mogućnosti prepoznavanja teksta. Njegov glavni cilj je pojednostaviti razvoj i implementaciju OCR sustava uz postizanje visoke točnosti i učinkovitosti. Ključne značajke i mogućnosti su mu:

- Svestrano prepoznavanje teksta: PaddleOCR podržava širok raspon zadataka prepoznavanja teksta, uključujući prepoznavanje teksta scene, analizu izgleda dokumenta, prepoznavanje tablice i prepoznavanje rukopisa. Ova svestranost ga čini prikladnim za različite primjene.
- Unaprijed trenirani (*engl. pretrained*) modeli: PaddleOCR nudi kolekciju unaprijed obučenih modela koji su obučeni na velikim skupovima podataka, kao što su SynthText [38], ICDAR [39] i COCO-Text [40]. Ovi modeli pokrivaju više jezika i stilova teksta, omogućujući korisnicima da ih odmah iskoriste za svoje specifične zadatke.
- Jednostavna integracija i implementacija: PaddleOCR pruža API (*engl. Application program interface*) jednostavan za korištenje i jasnu dokumentaciju, što ga čini dostupnim programerima i istraživačima. Podržava više programskih jezika, uključujući Python, C++ i Java-u, olakšavajući besprijekornu integraciju u različite aplikacije. Dodatno, podržava implementaciju na različitim platformama, uključujući CPU i GPU.

- Optimizacija modela: PaddleOCR fokusiran je na optimizaciju modela za scenarije stvarnog svijeta. Nudi lagane modele, kao što su MobileNetV3 i Slim, koji pružaju dobar kompromis između točnosti i računalne učinkovitosti. Ova optimizacija omogućuje brže vrijeme zaključivanja, što ga čini prikladnim za aplikacije u stvarnom vremenu.
- Povećanje skupa podataka i trening: PaddleOCR podržava tehnike povećanja skupa podataka za poboljšanje generalizacije modela. Pruža alate za generiranje sintetičkih podataka, geometrijske transformacije i sintezu teksta. Ovo pomaže u treningu modela koji su otporni na varijacije u izgledu teksta, orijentaciji i uvjetima osvjetljenja.

2.2.2. Tesseract OCR

Tesseract OCR [41], razvijen od strane Google-a, OCR je mehanizam otvorenog koda koji je stekao popularnost zahvaljujući svojoj točnosti, fleksibilnosti i mogućnosti proširenja. Iskorištava tehnike dubokog učenja i sveobuhvatan jezični model za točno izdvajanje teksta iz slika i dokumenata. Ključne značajke i mogućnosti su mu:

- Svestrano prepoznavanje teksta: Tesseract podržava širok raspon zadataka prepoznavanja teksta, uključujući prepoznavanje tiskanog teksta, analizu izgleda dokumenta i ekstrakciju višejezičnog teksta. Omogućuje jezične pakete za više od 100 jezika, što ga čini vrlo svestranim alatom za različite primjene.
- Točnost i jezična podrška: Tesseract koristi napredni OCR mehanizam koji kombinira tehnike dubokog učenja i statističkog modeliranja jezika. Njegovi jezični modeli uvježbani su na opsežnim skupovima podataka, omogućujući točno prepoznavanje različitih fontova, veličina i stilova.
- Otvoreni kod i proširivost: Tesseract je projekt otvorenog koda koji razvojnim programerima omogućuje pristup i izmjenu izvornog koda prema njihovim zahtjevima. Ova proširivost omogućuje prilagodbu i dodavanje novih značajki, čineći Tesseract prilagodljivim različitim slučajevima upotrebe.
- Trening i fino podešavanje (*engl. fine tuning*): Tesseract nudi mogućnosti treninga, omogućujući programerima fino podešavanje OCR mehanizma za određene domene ili aplikacije. Ova fleksibilnost omogućuje stvaranje specijaliziranih modela koji daju veću točnost i izvedbu za specifične zadatke prepoznavanja teksta.
- Sučelje naredbenog retka i API (*engl. Application Programming Interface - API*): Tesseract pruža sučelje naredbenog retka (*engl. Command-Line Interface - CLI*) za jednostavnu integraciju i skupnu obradu. Osim toga, nudi API-je za integraciju u različite

programske jezike, kao što su Python, Java i C++, olakšavajući besprijekornu integraciju u različite softverske aplikacije.

- Kontinuirano poboljšanje: Tesseract ima koristi od aktivne zajednice programera i suradnika. Redovita ažuriranja i poboljšanja osiguravaju da algoritam ostane u tijeku s najnovijim dostignućima u prepoznavanju teksta. Zajednica također pruža podršku, dokumentaciju i dodatne jezične pakete, proširujući Tesseractove mogućnosti i pristupačnost.

Tesseract OCR nalazi primjenu u širokom rasponu područja, uključujući digitalizaciju dokumenata, ekstrakciju teksta za rudarenje podataka, arhivsku digitalizaciju i još mnogo toga. Sa stalnim napretkom i doprinosima zajednice otvorenog koda, Tesseract se kontinuirano razvija kako bi mu se poboljšali točnost, brzina i jezična podrška. Budući razvoj može uključivati poboljšanja u modelima dubokog učenja, tehnikama optimizacije i integraciji s novim tehnologijama.

2.2.3. Postojeća rješenja za razumijevanje teksta

U [42] autori su predstavili rješenje za čitanje znakova s kineskih prometnih znakova koji sadrže i okomitu i vodoravnu orijentaciju teksta. Metoda detektira i prepoznaje kineske znakove koristeći RGB komponente i uzimajući u obzir karakteristike nepovezanih kineskih znakova. Tehnika kaskadne segmentacije boja primjenjuje se za prepoznavanje područja teksta, a koraci predobrade koriste se za filtriranje teksta koji ne pripada prometnom znaku. Za klasifikaciju se koristi kombinacija BoVW-a (*engl. Bag of Visual Words - BoVW*) i HOG-a. Predloženom metodom postiže se 94,20 % preciznosti u prepoznavanju prisutnosti kineskih znakova na prometnim znakovima. Predloženi algoritam naposljetku koristi model dubokog učenja za razumijevanje teksta. Procjene pokazuju njegovu učinkovitost u sustavima montiranim na automobilu i u stvarnom prometnom okruženju. Otkrivanje i prepoznavanje teksta na slikama i videozapisima predstavljaju izazove, osobito kada se radi o proizvoljnim orijentacijama i složenim pozadinama. Postojeće metode često imaju problema s točnim otkrivanjem i prepoznavanjem teksta u takvim scenarijima. Kako bi se to riješilo, predložena je metoda opisana u [43] koja kombinira model YOLOv5 za detekciju teksta i TesseractOCR za prepoznavanje teksta. Sustav koristi model YOLOv5s za slike manjih dimenzija i YOLOv5x za slike velikih dimenzija. Obučavanjem modela na različitim skupovima podataka postignuto je učinkovito otkrivanje teksta. TesseractOCR koristi se za pretvaranje otkrivenog teksta u format niza i njegovo pohranjivanje u CSV datoteku. Prilagođeni YOLOv5x model omogućuje točnu detekciju teksta u scenama stvarnog svijeta, dok TesseractOCR učinkovito prepoznaje detektirani tekst. Sustav pokazuje obećavajuće rezultate, posebno u rukovanju gotovo vodoravnim i složenim pozadinskim

slikama. Točnost koje je rješenje uspjelo postići na bazama podataka ICDAR2013 [44] i ICDAR2015 [45] iznosi 74 %, što ukazuje na to da je potreban daljni rad na poboljšanju modela, posebno za rad s višejezičnim skupovima podataka.

3. IZRADA ALGORITMA ZA IZDVAJANJE PUTOKAZA IZ SCENE I RAZUMIJEVANJE TEKSTA NA NJIMA

U ovom poglavlju opisana je izrada vlastitog računalnog algoritma za izdvajanje putokaza iz scene i razumijevanje teksta na njima. Istrenirana su dva modela, jedan za osobno računalo, drugi za ugradbenu računalnu platformu, jer je cilj usporediti ponašanje modela u dva različita okruženja. Algoritam je pisan u Python programskom jeziku. U potpoglavlju 3.1 opisan je proces postavljanja radnog okruženja u kojem će se model trenirati. U potpoglavlju 3.2 opisan je proces kreiranja baze slika koja će služiti za treniranje i testiranje. U potpogljavlju 3.3 detaljno je opisan sam postupak treniranja modela za dvije različite platforme, dok je u potpogljavlju 3.4 opisan proces izrade algoritma za razumijevanje teksta na putokazima. Korištena su dva različita algoritma, za svaku platformu po jedan te su rezultati uspoređivani u četvrtom poglavlju. Konačni cilj modela je ispravno detektirati i klasificirati objekte od interesa.

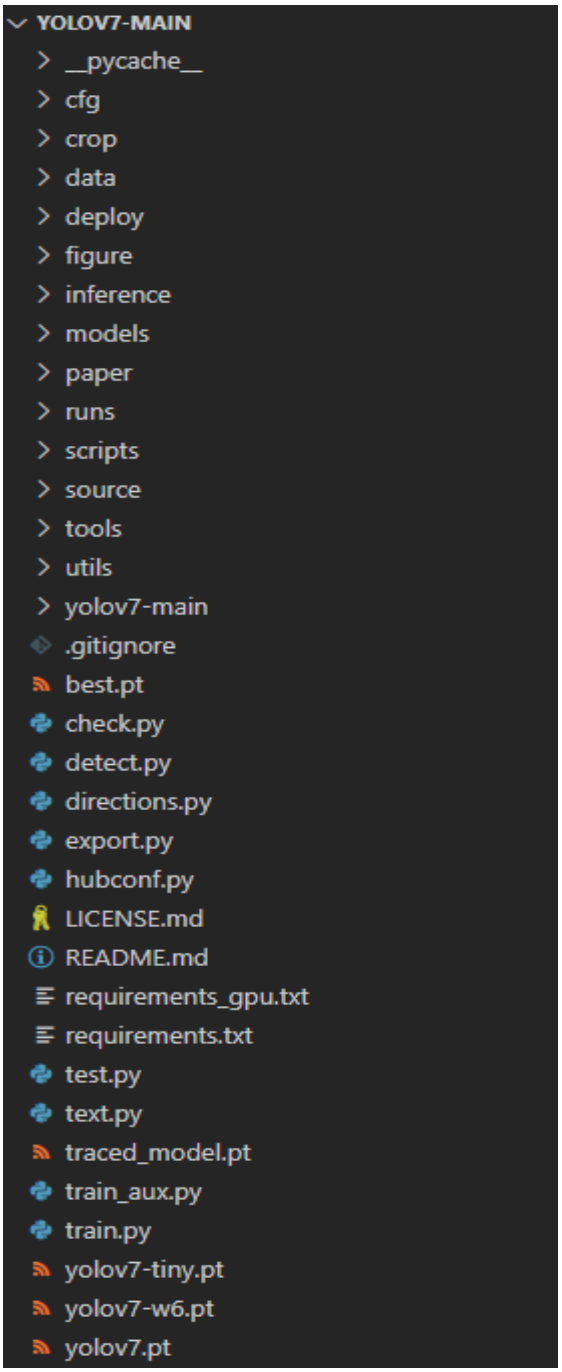
3.1. Postavljanje radnog okruženja

Postavljanje radnog okruženja bitan je korak za osiguravanje glatkog i učinkovitog toka rada. Proces započinje preuzimanjem cjelokupnog direktorija YOLOv7 modela s Github repozitorija [28]. Struktura direktorija, prikazana na slici 3.1., sadrži sve potrebno za treniranje i kasnije testiranje modela, uključujući osnovni kod YOLOv7 modela, unaprijed trenirane težine, konfiguracijske datoteke i relevantne baze podataka. Ova dobro organizirana struktura omogućuje laganu integraciju s razvojnim okruženjem. Nadalje, potrebno je instalirati *Anaconda*-u, moćan alat koji služi za upravljanje paketima i bibliotekama, nudi pristup ekosustavu alata i pojednostavljuje proces upravljanja projektom te osigurava kompatibilnost s verzijama paketa i biblioteka. Nakon što je *Anaconda* instalirana, potrebno je kreirati okruženje (*engl. environment*). Okruženje u Anacondi samostalni je radni prostor koji programerima omogućuje izolaciju i upravljanje ovisnostima za specifične projekte ili zadatke. Omogućuje kontrolirano okruženje u kojem različite verzije programskih (*engl. software*) paketa mogu postojati bez narušavanja odnosa, osiguravajući kompatibilnost među projektima. Stvaranjem zasebnih okruženja, programeri se mogu jednostavno prebacivati između različitih konfiguracija i verzija paketa bez utjecaja na stabilnost njihovog cjelokupnog sustava. Ova fleksibilnost posebno je vrijedna kada se, primjerice, radi na više projekata. Stvaranje okruženja u *Anaconda*-i, jednostavan je proces i ostvaruje se sljedećom naredbom:

conda create -n yolov7 python=3.9

dok se isto okruženje aktivira sljedećom naredbom:

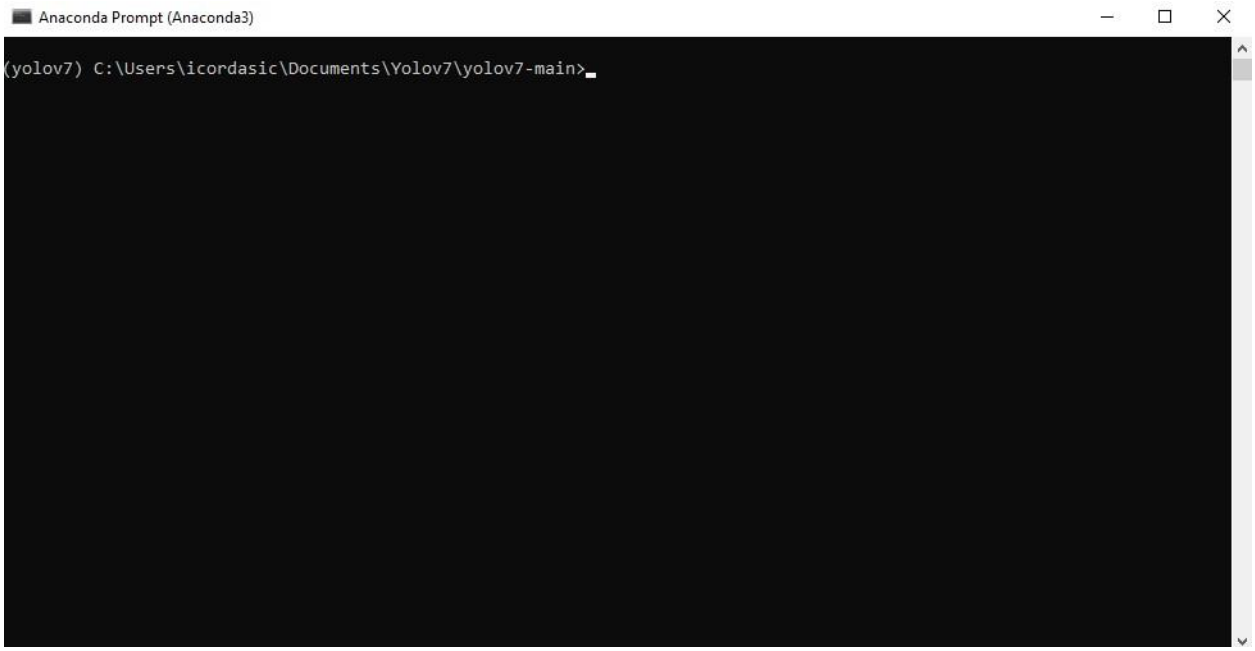
conda activate yolov7



Slika 3.1. *Struktura YOLOv7 direktorija*

Koristeći sučelje naredbenog retka *Anaconda*-e (prikazano na slici 3.2.), odabire se naziv okruženja i Python verzija. Kasnije se mogu i instalirati potrebni paketi i ovisnosti. Jednom stvoreno okruženje može se aktivirati ili deaktivirati, ovisno o zahtjevima projekta. Kada je

okruženje aktivno, putanja (*engl. path*) sustava se mijenja kako bi se odredio prioritet paketa i ovisnosti instaliranih unutar tog okruženja, osiguravajući korištenje ispravnih verzija tijekom izvođenja. Ovo pomaže u sprječavanju sukoba između različitih verzija paketa koji mogu biti instalirani u različitim okruženjima.



Slika 3.2. *Sučelje Acaconda alata*

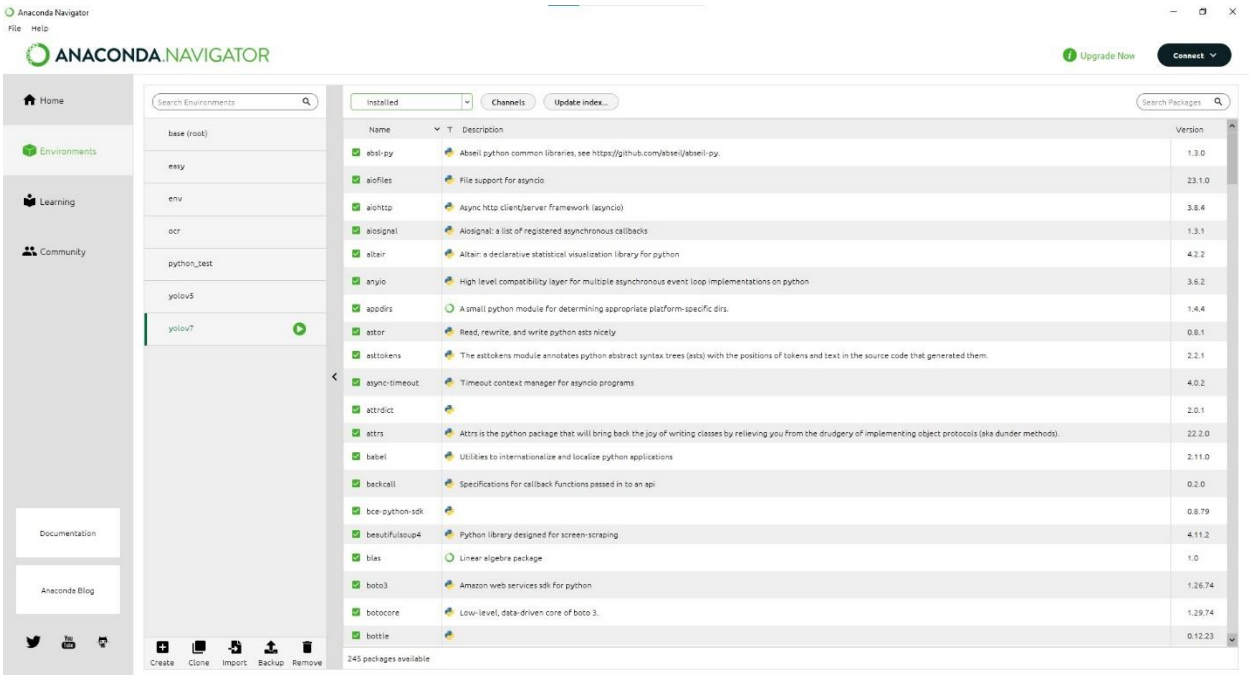
Instaliranje potrebnih paketa i ovisnosti navedenih u datotekama *requirements.txt* i *requirements-gpu.txt* ključan je korak u postavljanju radnog okruženja. Ove datoteke pružaju popis Python paketa i njihovih verzija potrebnih za pokretanje modela YOLOv7 i obavljanje raznih zadataka. Za instalaciju paketa koristi se naredba *pip* s oznakom *-r* iza koje slijedi putanja do odgovarajuće datoteke. Na primjer, za instalaciju paketa navedenih u *requirements.txt*, izvršava se sljedeća naredba u terminalu:

pip install -r requirements.txt

Za instalaciju paketa navedenih u *requirements-gpu.txt* koristi se slična naredba:

pip install -r requirements-gpu.txt

Ova naredba će automatski preuzeti i instalirati navedene pakete i njihove ovisnosti iz indeksa Python paketa (PyPI). Instalirani paketi, zajedno s pripadajućom verzijom, prikazani su u alatu *Anaconda Navigator*, što je prikazano na slici 3.3.



Slika 3.3. *Pregled dijela postojećih paketa u Anaconda Navigatoru*

Neki od ključnih paketa spomenutih u datoteci su:

- *matplotlib* [46]: Ovaj paket se široko koristi za vizualizaciju podataka i crtanje. Pruža sveobuhvatan skup funkcija za stvaranje raznih vrsta dijagrama.
- *numpy* [47]: *numpy* je temeljni paket za znanstveno računarstvo u Pythonu. Omogućuje razne numeričke operacije, što ga čini ključnim za rukovanje i obradu numeričkih podataka.
- *opencv-python* [48]: OpenCV je popularna biblioteka računalnog vida koja se koristi za zadatke obrade slika i videa. Paket *opencv-python* donosi funkcionalnosti OpenCV-a u Python, omogućujući programerima rad sa slikama i izvođenje raznih operacija računalnog vida.
- *Pillow* [49]: *pillow* je moćna biblioteka za obradu slika u Pythonu. Omogućuje alate za manipulaciju slikama, promjenu veličine, izrezivanje i još mnogo toga.
- *PyYAML* [50]: *PyYAML* je YAML (*engl. Yet Another Markup Language - YAML*) parser i emiter za Python. YAML je format serijalizacije podataka čitljiv za čovjeka, često se koristi za konfiguracijske datoteke.
- *Requests* [51]: *requests* je naširoko korišten paket za izradu HTTP (*engl. Hypertext Transfer Protocol - HTTP*) zahtjeva u Python-u. Pojednostavljuje proces interakcije s web API-jima i dohvaćanje podataka s udaljenih poslužitelja.

- *Scipy* [52]: *scipy* je biblioteka koja pruža širok raspon modula za optimizaciju, linearnu algebru, obradu signala i više.
- *Tqdm* [53]: *tdqm* je koristan paket za praćenje napretka iterativnih zadataka.

Dodatno, *requirements.txt* datoteka također uključuje pakete za bilježenje, iscrtavanje i dodatke kao što su *IPython* za interaktivna prijenosna računala, *psutil* za praćenje korištenja sustava te brojne druge. S druge strane, *requirements-gpu.txt* datoteka uključuje specifične pakete za GPU ubrzanje, kao što je naznačeno oznakom *-i* koja pokazuje na NVIDIA CUDA (*engl. Compute Unified Device Architecture*) *Toolkit.* Instalira pakete *torch* i *torchvision* s podrškom za CUDA, omogućujući GPU ubrzanje za zadatke dubokog učenja. Instaliranjem paketa navedenih u ovim datotekama, osiguravaju se sve potrebne ovisnosti za učinkovito pokretanje YOLOv7 modela i besprijekorno obavljanje raznih zadataka računalnog vida.

3.2. Kreiranje baze slika

Stvaranje sveobuhvatne baze podataka slika ključni je korak u treningu i validiranju sustava za detekciju putokaza i smjera na njima. U ovom konkretnom radu, zbog nedostupnosti relevantne baze podataka putokaza, korišteni su alternativni resursi za izradu baze podataka slika. Jedna takva opcija bila je baza podataka o znakovima [54], koja je pružala vrijednu zbirku slika prometnih znakova. Iz te baze podataka odabrano je 280 slika koje su uključene u bazu slika za daljnju analizu i trening. Primjer izabranih slika može se vidjeti na slici 3.4.

(a)

(b)

Slika 3.4. *Dva primjera slika preuzetih iz* [54]

Nadalje, YouTube ([55]–[60]) se pokazao kao neprocjenjiv resurs u proširenju baze slika. Brojni videozapisi vožnje pažljivo su analizirani, a posebna pažnja bila je usmjerena na okvire koji sadrže putokaze. Kada bi se takav okvir pojavio, taj okvir bio je izdvojen, uz zadržavanje odgovarajuće pozadine. Na taj način prikupljene su 683 slike, od kojih su neke prikazane na slici 3.5.

(a)

(b)

Slika 3.5. *Dva primjera slika preuzetih iz* [55]–[60]

Ovaj je pristup osigurao da putokazi dobiveni s YouTubea predstavljaju scenarije iz stvarnog svijeta i budu usklađeni s kontekstom u kojem se pojavljuju na cestama. Uključivanjem ovih putokaza izdvojenih s YouTube-a u bazu podataka slika, uključen je raznolik raspon varijacija znakova i kontekstualne pozadine, poboljšavajući sposobnost sustava da se nosi s izazovima otkrivanja prometnih putokaza u stvarnom svijetu. Uz ručno označavanje, upotrijebljena je tehnika augmentacije podataka kako bi se dodatno proširila izgrađena baza podataka slika za detekciju putokaza i smjera na njima. Augmentacija podataka je proces primjene različitih transformacija i modifikacija na postojeće slike, kako bi se stvorili dodatni uzorci slika. Svih 280 slika iz [54] koje su se koristile u procesu augmentacije transformirane su na sljedeći način:

- rotacija: svaka slika rotirana je za slučajno odabrani kut između -30° i 30°,
- skaliranje: originalne dimenzije slika promijenjene su za faktor između 0.8 i 1.2,
- podešavanje svjetline: intenzitet svjetline na slikama je varirao između 75% i 125% od originalnog intenziteta.

Svaka od navedenih transformacija primjenjivana je samostalno na svaku od odabranih 280 slika iz [54], što je rezultiralo generiranjem triju dodatnih augmentiranih slika za svaku originalnu

sliku, što dovodi do broja od 840 novih slika. Međutim, neke su bile nepovoljne (izobličenja, gubitak bitnih detalja, narušavanje kvalitete slike) pa je odabrano ukupno 648 augmentiranih slika koje će se koristiti u daljnjem procesu. Na slici 3.6. prikazana je primjer izvorne i augmentirane slike (dobivena rotacijom za 30°). Ove transformacije simuliraju varijacije stvarnog svijeta u uvjetima osvjetljenja, točkama gledišta i kvaliteti slike, čineći model robusnijim na takve varijacije. Kombinirano korištenje navedenih resursa omogućilo je opsežniju i raznovrsniju bazu slika. Ova proširena baza podataka pridonijela je robusnijem i preciznijem uvježbavanju sustava za detekciju putokaza i smjera na njima. To je omogućilo sposobnost sustava da dobro generalizira za različite vrste putokaza, pozadine i uvjeta okoline, što je dovelo do poboljšanih performansi kada se suoče sa zadatkom detekcije putokaza i smjera na njima u stvarnom svijetu. Ukupan broj slika dobivenih opisanim procesima iznosi 1611 (280 + 683 + 648).

Izrada baze podataka slika uključivala je kvalitetnu organizaciju i odvajanje slika u različite skupove. Skup podataka podijeljen je na 746 slika za trening, koje su poslužile kao osnova za trening sustava za detekciju putokaza i smjera na njima. Odabran je zaseban skup od 297 validacijskih slika za fino podešavanje (*engl. fine-tuning*) parametara sustava i procjenu njegove izvedbe tijekom razvojne faze. Konačno, odvojeno je 568 slika za testiranje kako bi se procijenila učinkovitost sustava za nove podatke koji nisu korišteni u fazi treninga i validacije. Sve slike su pozitivne i sadrže minimalno jedan objekt neke od klasa koje treba detektirati u ovom radu.

(a)

(*b)*

Slika 3.6. *(a) Originalna slika, (b) augmentirana slika*

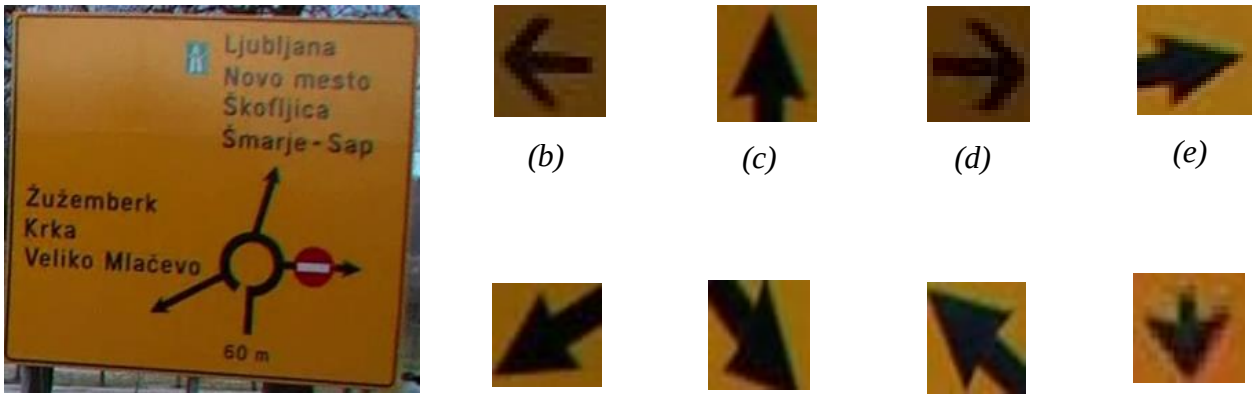
Korištene slike mogu se pronaći u elektroničkom prilogu na DVD-u priloženom uz rad u mapi Prilog P.3.1.

Klase koje je potrebno detektirati u ovom radu su (sve klase osim prve odnose se na strelice):

- klasa *putokaz*
- klasa *L* (lijevo)
- klasa *G* (gore)
- klasa *D* (desno)
- klasa *GD* (gore desno)
- klasa *DL* (dolje lijevo)
- klasa *DD* (dolje desno)
- klasa *GL* (gore lijevo)
- klasa *d* (dolje)

Primjeri po jedne slike za svaku od klasa prikazani su na slici 3.7.

Anotiranje slika igra bitnu ulogu u strojnom učenju, pružanjem označenih informacija o određenim objektima ili područjima od interesa na slici. U kontekstu detekcije putokaza i smjera na njima, označavanje uključuje označavanje lokacije objekta i klase unutar slika, kako bi se



(a) *(b)* *(c)* *(d)* *(e)* *(f)* *(g)* *(h)* *(i)*

Slika 3.7. *Primjeri objekata koje je potrebno detektirati: (a) klasa putokaz, (b) klasa L, (c) klasa G (d) klasa D, (e) klasa GD, (f) klasa DL, (g) klasa DD, (h) - klasa GL, (i) klasa d*

stvorili temeljni istiniti podaci za treniranje modela. Za anotiranje slika korišten je alat *labelImg* [61]. *labelImg* je popularan alat otvorenog koda za anotiranje slika koji korisnicima omogućuje crtanje graničnih okvira oko objekata od interesa i dodjeljivanje odgovarajućih oznaka. Pomoću navedenog alata ručno je označena svaka slika u bazi podataka crtanjem graničnih okvira oko putokaza i smjera na njima prisutnih na slikama. Izlaz svake anotirane slike je tekstualni zapis pojedinačnog objekta na slici u obliku:

klasa_objekta x_koordinata_središta y_koordinata_središta širina visina

gdje posljednje 4 vrijednosti tekstualnog zapisa predstavljaju brojeve u rasponu [0.0, 1.1] jer su vrijednosti skalirane kako bi modelu bilo lakše učiti, dok prva vrijednost počinje s oznakom 0 pa nastavlja dalje redom, ovisno o tome koliko ima klasa (ukupno 9 klasa, pa oznake idu do broja 8). Kako bi se izračunale vrijednosti *x_koordinata_središta* (3-1), *y_koordinata_središta* (3-2), *širina* (3-3) i *visina* (3-4) za YOLO format tekstualnog zapisa, potrebno je koristiti sljedeće izraze:

$$x\_koordinata\_središta = \frac{x\_središte\_graničnog\_okvira}{širina\_slike} \quad (3\text{-}1)$$

$$y\_koordinata\_središta = \frac{y\_središte\_graničnog\_okvira}{visina\_slike} \quad (3\text{-}2)$$

$$širina = \frac{širina\_graničnog\_okvira}{širina\_slike} \quad (3\text{-}3)$$

$$visina = \frac{visina\_graničnog\_okvira}{visina\_slike} \quad (3\text{-}4)$$

gdje su *x_središte_graničnog_okvira* i *y_središte_ graničnog_okvira* koordinate središnjeg elementa slike graničnog okvira objekta, a *širina_ graničnog_okvira* i *visina_ graničnog_okvira* dimenzije graničnog okvira objekta. Ukoliko je slika rezolucije 500x500 piksela, a označeni granični okvir objekta od interesa ima središte s koordinatama u (250,250) i dimenzija je 100x200 piksela, tada bi računanje vrijednosti za YOLO format tekstualnog zapisa, za klasu objekta 0, kao i sam zapis izgledalo kako je prikazano u nastavku:

$$klasa\_objekta = 0$$

$$x\_koordinata\_središta = 250 / 500 = 0.5$$

$$y\_koordinata\_središta = 250 / 500 = 0.5$$

$$širina = 100 / 500 = 0.2$$

$$visina = 200 / 500 = 0.4$$

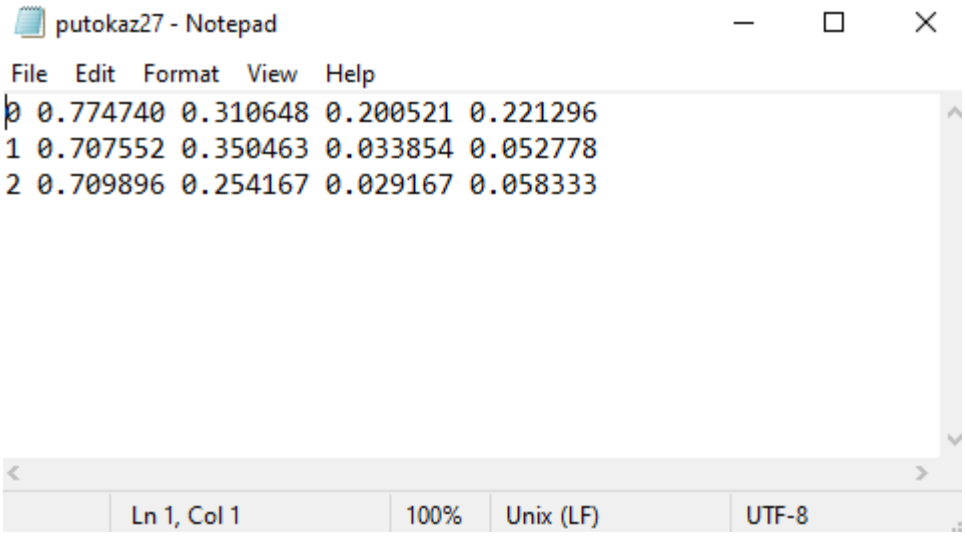
$$tekstualni\ zapis: 0\ 0.5\ 0.5\ 0.2\ 0.4$$

Svaki objekt na istoj slici zapisan je u novu liniju. Klasa putokaz je logično nazvana putokaz, dok su smjerovi strelica nazvani prema početnom slovu smjera na koji pokazuju, kako ispisani tekst ne bi zauzimao puno prostora prilikom inferencije. Popis korištenih klasa s pripadajućim ID-ovima (*engl. Identification Number - ID*) i broju oznaka po klasama prikazan je u tablici 3.1. Na slici 3.8. prikazana je potpuno označena slika, dok je tekstualni zapis te iste slike prikazan na slici 3.9.

Tablica 3.1. *Prikaz klasa sa pripadujućim ID-ovima i broju oznaka po pojedinom skupu*

KLASA	ID	TRENING SKUP	VALIDACIJSKI SKUP	TESTNI SKUP
putokaz	0	984	406	731
L	1	158	51	111
G	2	437	139	340
D	3	362	174	257
GD	4	415	173	314
DL	5	33	33	28
DD	6	29	29	24
GL	7	79	73	63
d	8	23	15	18

Slika 3.8. *Potpuno označena slika u labelImg alatu*



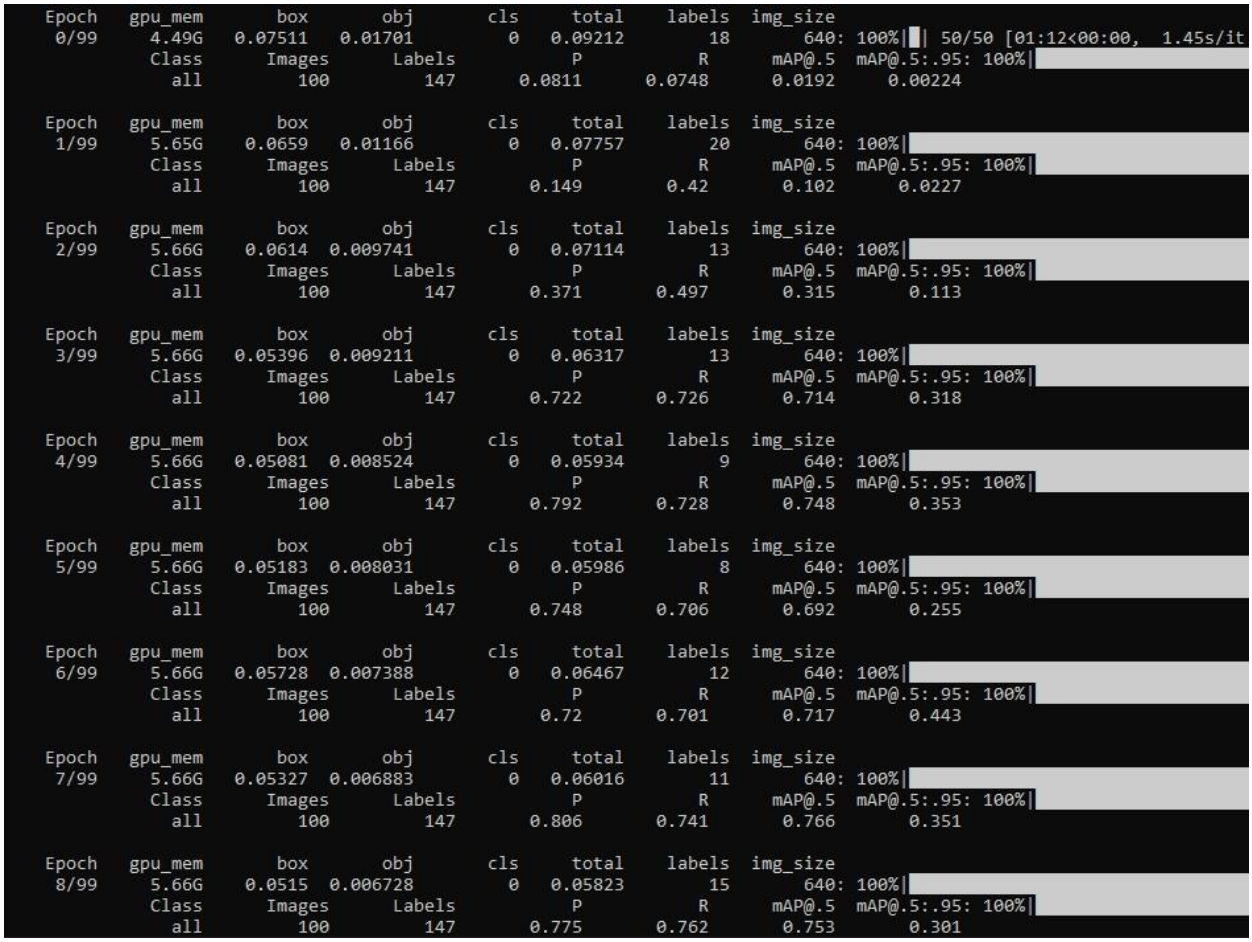
Slika 3.9. *Tekstualni zapis potpuno označene slike*

3.3. Proces treniranja algoritma zasnovanog na YOLOv7 modelu za detekciju putokaza i smjera na njima

Proces treninga za YOLOv7 model proveden je na osobnom računalu opremljenom Intel Core i7 procesorom šeste generacije, 16 GB RAM-a (*engl. Random Access* Memory - *RAM*) i NVIDIA GeForce GTX 1060 s 6GB GPU memorije. Sustav je radio na Windows operacijskom sustavu,

pružajući prikladno okruženje za osposobljavanje modela za detektiranje putokaza i smjerova prikazanih na njima. YOLOv7, u trenutku pisanja rada, najsuvremeniji model za otkrivanje objekata, korišten je za trening modela. YOLOv7 je poznat po svojoj visokoj točnosti i performansama u stvarnom vremenu, što ga čini prikladnim izborom za ovaj zadatak. Model koristi CNN koja istovremeno predviđa granične okvire i vjerojatnosti klasa za više objekata na slici. Tijekom procesa treninga, anotirane slike, njih 500, iz konstruirane baze slika korištene su za trening modela YOLOv7. Početna ideja bila je da se pomoću YOLOv7 samo detektiraju putokazi u slici, a da se nakon toga određenim dodatnim tehnikama detektiraju smjerovi na putokazima. Ovakav pristup isproban je sa spomenutih 500 slika i ukratko je opisan u nastavku. Trening je uključivao optimizaciju parametara modela prilagodbom težina iterativnim procesom. Ovaj iterativni proces, poznat kao algoritam unazadne propagacije (*engl. backpropagation*), omogućuje modelu da uči iz svojih pogrešaka i kontinuirano poboljšava svoju izvedbu [62]. Proces treninga uključivao je prilagodbu različitih hiperparametara, kao što su stopa učenja, veličina serije (*engl. batch size*) i broj ponavljanja (*engl. epoch*), kako bi se optimizirala izvedba modela. GPU ubrzanje koje pruža NVIDIA GeForce GTX 1060 značajno je ubrzalo proces treninga, omogućujući bržu konvergenciju i učinkovitije ažuriranje modela. Fokus treninga prvotno je stavljan isključivo na detekciju putokaza. Model je osposobljen za precizno detektiranje putokaza prisutnih na slikama. To je uključivalo treninga modela YOLOv7 više puta, ponavljanje i usavršavanje procesa treninga kako bi se poboljšali rezultati. U početnim fazama treninga rezultati su bili manje zadovoljavajući jer je model bio u ranoj fazi učenja. Međutim, sa svakim sljedećim ciklusom treninga, izvedba modela postupno se poboljšavala. Ovaj iterativni pristup omogućio je algoritmu da uči iz svojih pogrešaka i izvrši prilagodbe svojih predviđanja, što je dovelo do točnijeg otkrivanja putokaza. Kako bi se olakšao proces treninga, težine *yolov7.pt* [27] (prethodno uvježbani model) korištene su kao inicijalne težine postavljene na početku treninga za problem koji se rješava u ovom radu. Proces treninga proveden je 39 puta, uključujući neprestano mijenjanje hiperparametara. Svako ponavljanje treninga poboljšalo je sposobnost modela da nauči karakteristike putokaza, poboljšavajući njegovu sposobnost točnog detektiranja. Na slici 3.10. prikazan je ispis za vrijeme treniranja (*engl. training logs*) kroz sučelje *Anaconda* alata. Ispis uključuje razne informacije o trenutnom statusu procesa treninga, uključujući broj trenutnog ponavljanja, zauzeće memorije, trenutni gubitak, preciznost, odziv, *mAP* i brojne druge. Tijekom svakog novog treninga različiti hiperparametri prilagođeni su za fino podešavanje modela i optimizaciju njegove izvedbe.

Hiperparametri igraju ključnu ulogu u oblikovanju procesa učenja algoritma i određivanju učinkovitosti obučenog modela. Nekoliko je hiperparametara izmijenjeno u svakom novom



Slika 3.10. *Prikaz ispisa treniranja modela za detekciju putokaza kroz sučelje Anaconde*.

treningu kako bi se istražile različite konfiguracije i pronašle optimalne postavke za otkrivanje putokaza. Parametri kao što su stopa učenja, veličina serije, veličina slike i broj epoha treninga prilagođeni su kako bi se postigla ravnoteža između složenosti modela i računalne učinkovitosti. Sustavnim variranjem ovih hiperparametara bilo je moguće promatrati njihov utjecaj na konvergenciju, točnost i sposobnost generalizacije modela. Ovaj iterativni pristup omogućio je kontinuirano eksperimentiranje i poboljšavanje vrijednosti hiperparametara, omogućujući modelu da učinkovitije uči iz podataka o obuci i poboljša svoju izvedbu. Sa svakim novim treningom, uvidi dobiveni iz prethodnih izvođenja treninga korišteni su za informiranje o prilagodbi hiperparametara u sljedećim iteracijama. Ovaj iterativni proces modificiranja hiperparametara i evaluacije performansi modela osigurao je postupno poboljšanje sposobnosti modela da točno detektira putokaze. Radi jednostavnosti i zorne ilustracije ishoda treninga, analizirat će se rezultati posljednjeg treninga koji se pokazao najuspješnijim. Posljednja iteracija treninga pokazala je

značajna poboljšanja u točnosti otkrivanja putokaza i pokazala sposobnost modela da učinkovito detektira putokaze. Naredba koja se koristila za taj trening je:

python train.py --weights yolov7.pt --cfg cfg\training\yolov7_custom.yaml --data data\data.yaml --hyp data\hyp.scratch.custom.yaml --epochs 100 --batch-size 32 --img-size 192 --device 0 --name nove_slike_tr

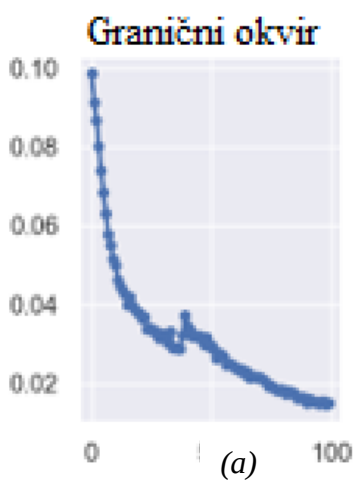
train.py je naziv skripte koja se pokreće. Datoteka *yolov7_custom.yaml* daje detaljan opis mrežne arhitekture za YOLOv7, specificirajući konfiguraciju modela. U *data.yaml* datoteci navedene su putanje direktorija za skupove podataka za trening, validaciju i testiranje. Direktoriji *train*, *val* i *test* označavaju lokacije na kojima su pohranjeni odgovarajući skupovi podataka. Također su dani broj i naziv klase za koju se model trenira. Datoteka *hyp.scratch.custom.yaml* sadrži različite hiperparametre koji se koriste tijekom treninga YOLOv7 modela. Ovi hiperparametri kontroliraju različite aspekte procesa treninga i mogu se prilagoditi za optimizaciju izvedbe modela. Broj epoha označava koliko puta će se kompletno proći kroz cijeli trening set slika. Veličina serija je postavljena na 32 i određuje broj uzoraka slika koje se obrađuju zajedno prije ažuriranja težina modela. Dimenzija ulazne slike slike je postavljena na 192x192 piksela, dok je uređaj na kojem se trening izvodi označen sa '0', što označava GPU. Rezultati treninga za čije je izvođenje bilo potrebno 2.47h (prikazani u tablici 3.2.), dobiveni su treniranjem modela na trening skupu slika, dok je validacija provedena na validacijskom skupu slika nakon svake epohe. Nije implementirana metoda ranog zaustavljanja (*engl. early stopping*), već je za odabir najboljeg modela korišten pristup temeljen na najmanjoj vrijednosti validacijskog gubitka (*engl. loss*) tijekom treninga. YOLOv7, tijekom treninga, automatski sprema težine modela koje postižu najbolje performanse na validacijskom skupu kao *best.pt* datoteku. Ovaj pristup osigurava da konačni model koji se koristi za testiranje predstavlja onaj koji je postigao najbolje performanse na validacijskom skupu tijekom cijelog procesa treninga.

Tablica 3.2. *Rezultati treniranja za klasu 'putokaz' s prikazanim pripadajućim vrijednostima preciznosti, odziva i F1 mjere*

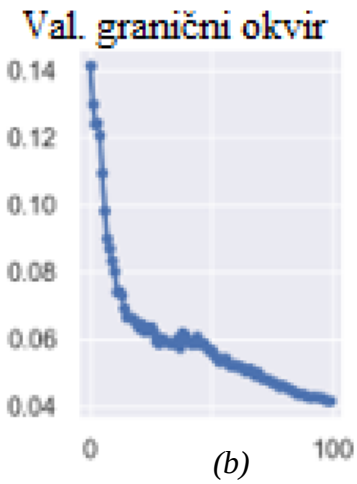
Klasa	Preciznost	Odziv	F1 mjera
putokaz	0.951	0.938	0.944

Nakon svakog treninga generiraju se grafovi koji prikazuju performanse modela kroz broj epoha tijekom treninga. Broj epoha je prikazan sa apscisi, dok se na ordinati nalaze brojčane vrijednosti onoga što sam graf prikazuje. Grafovi koji se generiraju su sljedeći:

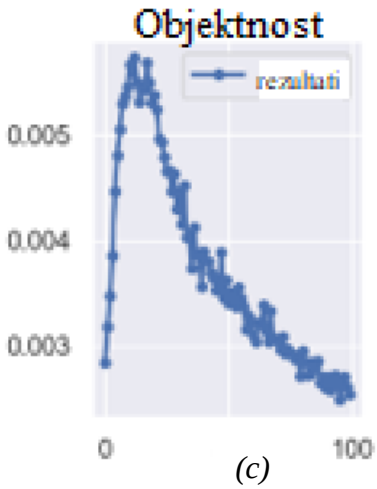
- Granični okvir (slika 3.11. (a)) i validacijski granični okvir (slika 3.11. (b)): prikazuju koliko dobro model predviđa granične okvire objekta. Vrijednosti na ordinati označavaju prosječnu pogrešku modela u predviđanju položaja i veličine graničnog okvira objekta. Pogreška se izračunava kao razlika između stvarnih i predviđenih koordinata središta graničnih okvira, uz dodatne metrike koje mogu obuhvaćati omjer dimenzija, površinu ili udaljenost. Niže vrijednosti na ovom grafu označavaju bolje performanse modela.
- Objektnost (engl. *objectness*) (slika 3.11. (c)) i validacijska objektnost (slika 3.11. (d)): prikazuju koliko dobro model prepoznaje gdje su objekti, bez obzira na klasu. Vrijednosti na ordinati označavaju prosječnu pogrešku modela u predviđanju prisutnosti objekata na slici. Pogreška se računa kao razlika između predviđenih vjerojatnosti prisutnosti objekta koje generira model i stvarne prisutnosti objekta (100 %). Niže vrijednosti na ovom grafu označavaju bolje performanse modela.
- Preciznost (slika 3.11. (e)): prikazuje preciznost modela. Na ordinati se nalaze vrijednosti preciznosti. Veće vrijednosti na ovom grafu označavaju bolje performanse modela.
- Odziv (slika 3.11. (f)): prikazuje odziv modela. Na ordinati se nalaze vrijednosti odziva. Veće vrijednosti na ovom grafu označavaju bolje performanse modela.
- mAP@0.5 (slika 3.11. (g)): prikazuje *mAP* modela pri *IoU* pragu od 0.5. Na ordinati se nalaze vrijednost *mAP-a*. Veće vrijednosti na ovom grafu označavaju bolje performanse modela.
- mAP@0.5:0.95 (slika 3.11. (h)): prikazuje *mAP* modela za *IoU* pragove od 0.5 do 0.95 s korakom 0.05. Na ordinati se nalaze vrijednosti *mAP-a*. Veće vrijednosti na ovom grafu označavaju bolje performanse modela.



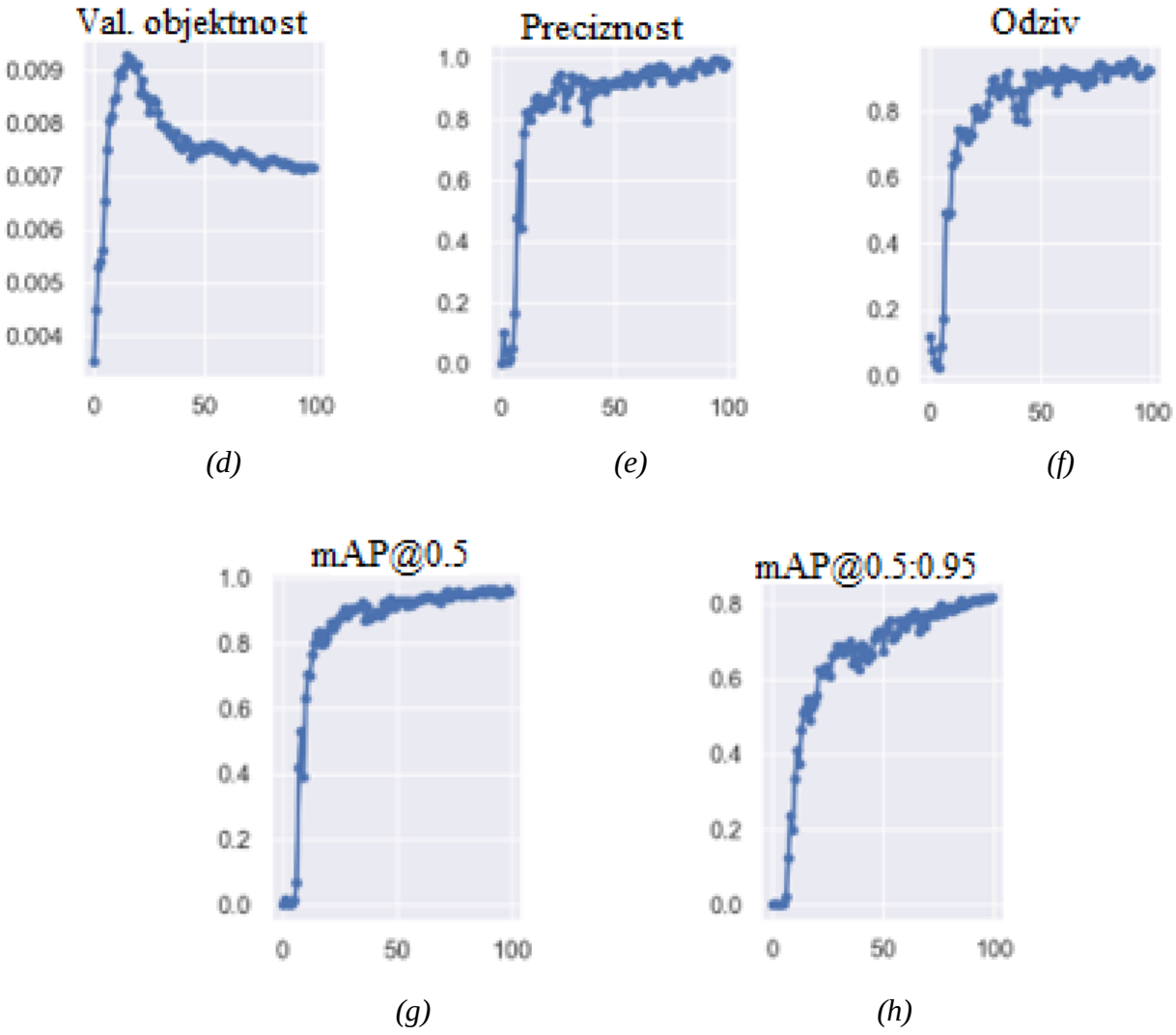
(a)



(b)



(c)



(d) *(e)* *(f)*

(g) *(h)*

Slika 3.11. *Generirani grafovi nakon treninga modela za detekciju putokaza koji se pokazao najboljim: (a) graf graničnih okvira, (b) graf validacijskih graničnih okvira, (c) graf objektnosti, (d) graf validacijske objektivnosti, (e) graf preciznosti, (f) graf odziva, (g) mAP@0.5 graf, (h) mAP@0.5:0.95 graf*

U pokušaju detektiranja smjera na putokazu, istražena je tehnika uparivanja predložaka (*engl. template matching*) [64]. Uparivanje predložaka je popularna tehnika obrade slika koja se koristi za pronalaženje prisutnosti određenog predloška ili uzorka unutar slike. Ideja je bila izdvojiti putokaz, te na temelju jednog predloška smjera tražiti slične uzorke na slici. Međutim, tijekom faze eksperimentiranja nekoliko izazova je pronađeno s pristupom uparivanja predložaka. Iako bi promjena mjerila ulazne slike mogla djelomično riješiti problem, problem neotpornosti algoritma na rotaciju je i dalje prisutan (prikazano na slici 3.12.). Algoritam za uparivanje predložaka nije inherentno otporan na rotaciju predloška, što otežava točno podudaranje izvornog predloška s rotiranim smjerovima koje je potrebno detektirani.

S obzirom da navedeni pristup nije bilo moguće koristiti, u cilju što boljeg konačnog rezultata,

Slika 3.12. *Neuspjeh detekcije svih smjerova na izrezanom putokazu zbog ograničene otpornosti predloška (označen plavim graničnim okvirom) na rotaciju*

ponovno je za isti problem primijenjen YOLOv7 algoritam. Nadovezujući se na prvotni model za detektiranje putokaza koji je uspješno izgrađen i istreniran, model je proširen dodavanjem novih klasa, od kojih svaka predstavlja različitu stranu svijeta, odnosno smjer na koji putokaz pokazuje. Umjesto da se ograniči samo na detektiranje putokaza, novi, unaprijeđeni model sada može detektirati ukupno devet različitih klasa - putokaz i osam različitih smjerova. Sve slike iz trening skupa slika (746), ali i iz validacijskog (297) i testnog (568), zbog kasnijeg verificiranja i testiranja, ponovno su anotirane kako bi uključile oznake smjera. Model je, u konačnici, postao precizniji i robusniji na razne prepreke (varijacije u kutovima gledanja, različiti smjerovi i dizajn putokaza) na koje se može naići u stvarnim situacijama. Trening, za ovaj prošireni model, izveden je 17 puta, te su rezultati svaki put bivali sve bolji i bolji. Odluka o zaustavljanju treniranja nakon sedamnaestog pokušaja donesena je na temelju pažljive analize. Iako se čini da bi se dodatnim treninzima mogli možda postići bolji rezultati, u posljednja tri treninga nije bilo značajnog napretka u performansama modela. Za odabir najboljeg modela koristila se metoda temeljena na minimizaciji gubitka validacije tijekom procesa treniranja. Konkretno, tijekom treniranja YOLOv7 modela, automatski su zabilježene težine modela koje su rezultirale najmanjim gubitkom na validacijskom skupu. Ove optimalne težine su pohranjene u datoteku nazvanu *best.pt*. Ovaj pristup osigurava da model koji se koristi za testiranje predstavlja onaj koji je ostvario najbolje performanse na validacijskom skupu tijekom cijelog procesa treniranja. Rezultati najuspješnijeg treninga za čije je izvođenje bilo potrebno 8.396 h prikazani su u tablici 3.3. Pojedinačni grafovi

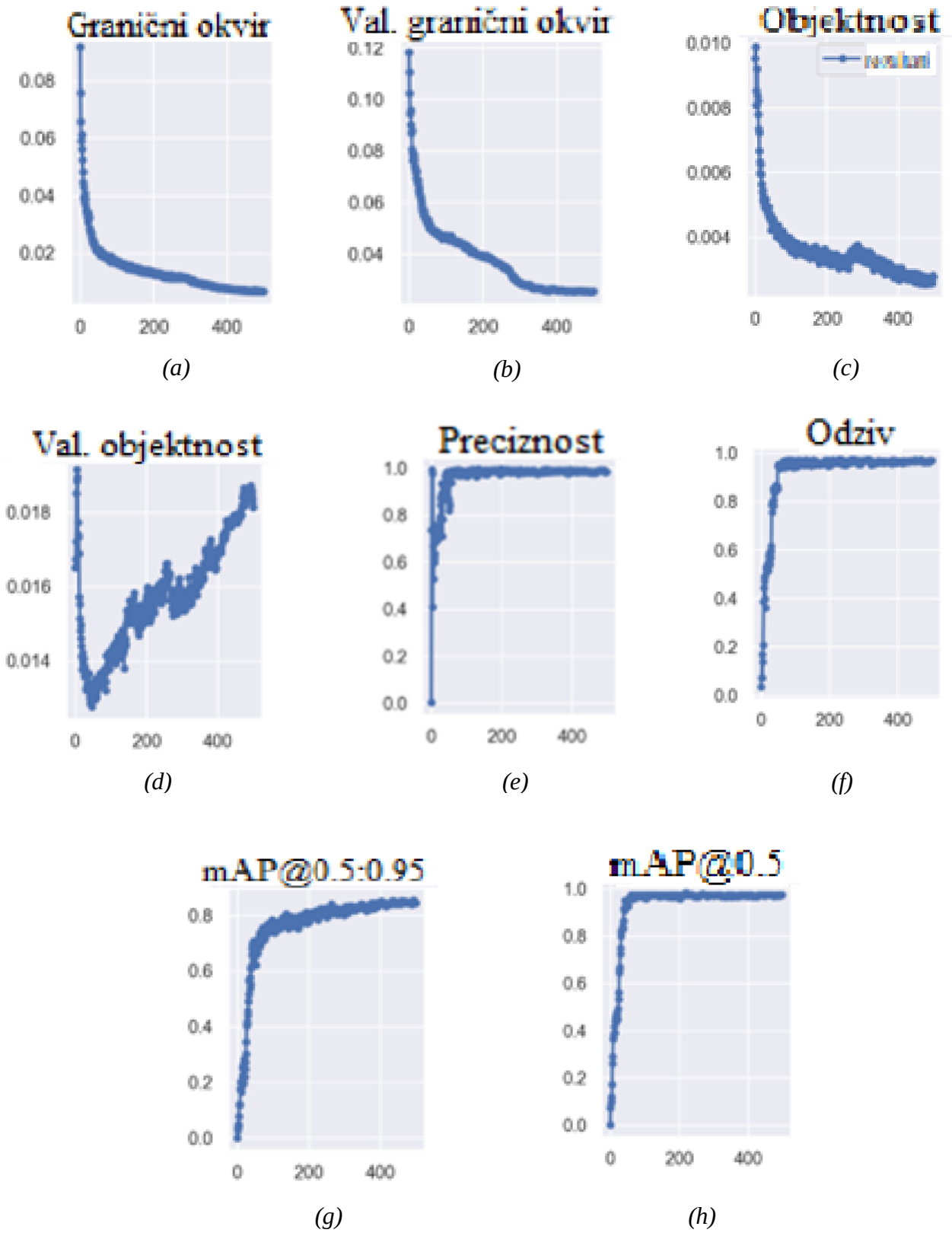
treninga prikazani su na slici 3.13., dok je matrica zabune (*engl. confusion matrix*) prikazana na slici 3.14. Matrica zabune [63] je tablični prikaz koji daje detaljan sažetak izvedbe modela. Omogućuje procjenu predviđanja modela uspoređujući ih sa stvarnim oznakama. Matrica se sastoji od četiri ključna pokazatelja: TP, TN, FP i FN. Pozadinski FN predstavlja primjere kada model propusti detektirati objekt jer ih pogrešno klasificira kao pozadinu, dok pozadinski FP predstavlja primjere kada model pogrešno klasificira dijelove pozadine kao objekte. Stupci matrice zabune predstavljaju stvarne klase, dok redovi predstavljaju predviđene klase. Brojevi unutar matrice pokazuju kako su primjeri svake stvarne klase klasificirani u svaku predviđenu klasu. Npr., u prvom retku (putokaz) i prvom stupcu (putokaz), broj je 0.99, što znači da model točno klasificira 99% svih stvarnih putokaza kao putokaze. Naredba koja se koristila za ovaj trening:

python train.py --weights yolov7.pt --cfg cfg\training\yolov7_custom.yaml --data data\data.yaml --hyp data\hyp.scratch.custom.yaml --epochs 500 --batch-size 16 --img-size 420 --workers 4 --device 0

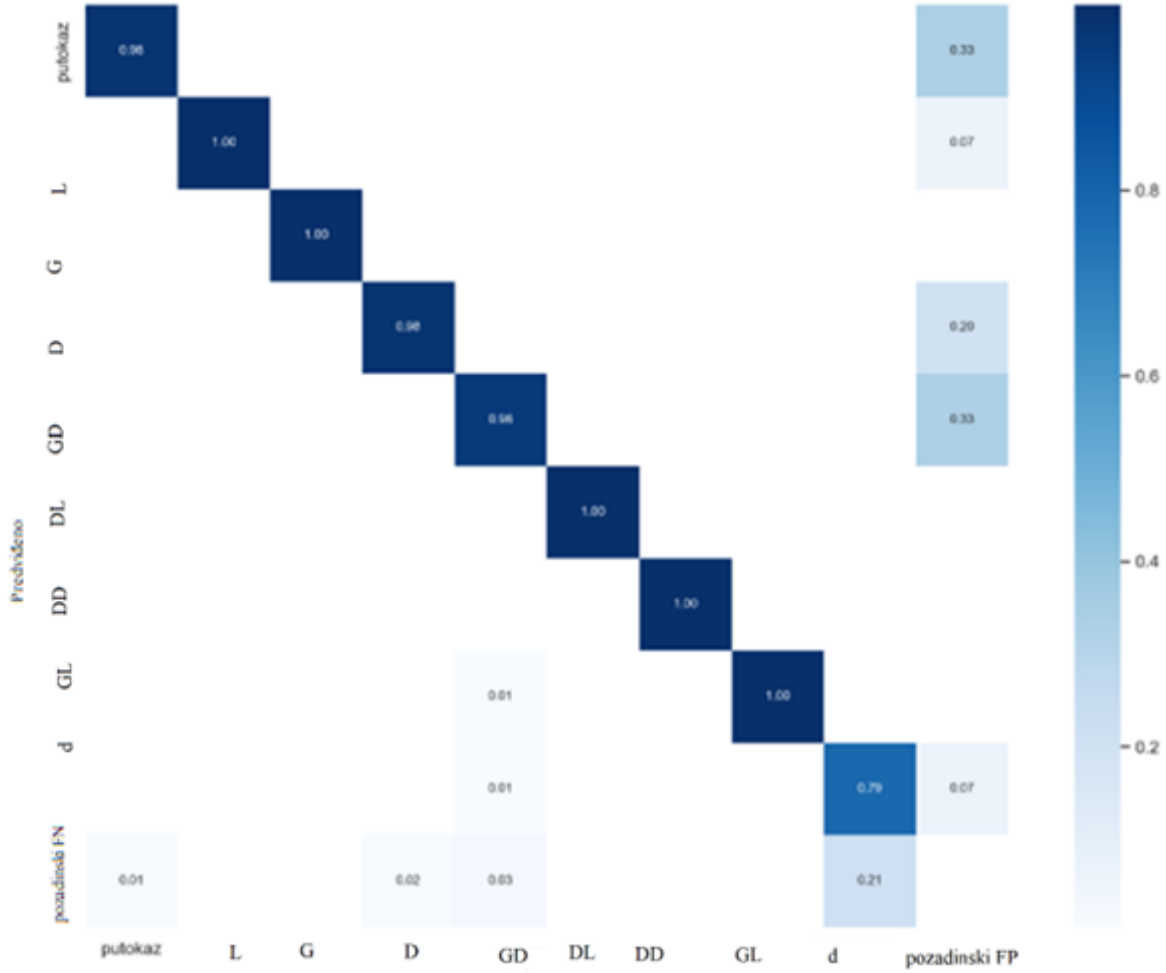
Osim poznatih parametara treninga koji se prosljeđuju u naredbenom retku *Anaconda*-e, dodan je parametar radnika (*engl. workers*), koji označava broj jezgri CPU (*engl. Central Processor Unit - CPU*) koje se koriste prilikom treninga modela.

Tablica 3.3. *Konačni rezultati treninga proširenog modela sa 9 klasa na trening skupu podataka*

Klasa	Preciznost	Odziv	F1 mjera	TP broj	FP broj	FN broj
putokaz (984 primjera)	0.750	0.989	0.853	974	324	10
L (158 primjera)	0.935	1.000	0.966	158	11	0
G (437 primjera)	1.000	1.000	1.000	437	0	0
D (362 primjera)	0.831	0.978	0.898	354	72	8
GD (415 primjera)	0.743	0.949	0.833	394	136	21
DL (33 primjera)	1.000	1.000	1.000	33	0	0
DD (29 primjera)	1.000	1.000	1.000	29	0	0
GL (79 primjera)	1.000	1.000	1.000	79	0	0
d (23 primjera)	0.947	0.783	0.857	18	1	5



Slika 3.13. *Generirani grafovi nakon treninga modela za detekciju putokaza i smjerova na njima koji se pokazao najboljim: (a) graf graničnih okvira, (b) graf validacijskih graničnih okvira, (c) graf objektnosti, (d) graf validacijske objektivnosti, (e) graf preciznosti, (f) graf odziva, (g) mAP@0.5 graf, (h) mAP@0.5:0.95 graf*



Slika 3.14. *Matrica zabune proširenog modela s 9 klasa dobivena iz trening skupa podataka.*

Prema rezultatima, primjetno je da je odziv za klasu *d* značajno manji u usporedbi s ostalim klasama. Razlog za ovu pojavu može ukazivati na moguće probleme s anotiranjem slika ili kvalitetom samih slika. U nekim primjerima gdje se pojavljuje klasa *d*, slike su loše anotirane, odnosno objekt od interesa (klasa *d*) nije pravilno anotiran. Također, loša kvaliteta slika može biti drugi ključni faktor navedenog problema. U tom slučaju model može imati poteškoće u izdvajanju relevantnih karakteristika i obilježja klase *d* što može rezultirati nižim performansama. Primjer gdje model nije detektirao klasu *d* (izvorna slika nije dobro anotirana) prikazan je na slici 3.15.

YOLOv7 provodi proces validacije nakon svake epohe tijeom treninga. Tijekom tog procesa, model uspoređuje svoje performanse na validacijskom skupu i bilježi najmanju vrijednost validacijskog gubitka. Optimalne težine modela koje su rezultirale najboljim performansama pohranjuju se u datoteku *best.pt*. U tablici 3.4. prikazani su rezultati validacije za epohu (423. epoha) u kojoj se model ostvario najbolje rezultate na validacijskom skupu podataka u procesu treninga.

Slika 3.15. *Primjer kada model ne detektira klasu d*

Tablica 3.4. *Rezultati validacije proširenog modela sa 9 klasa na validacijskom skupu podataka u najboljoj epohi tijekom treninga*

Klasa	Preciznost	Odziv	F1 mjera	TP broj	FP broj	FN broj
putokaz (406 primjera)	0.667	0.988	0.796	401	203	5
L (51 primjer)	0.944	1.000	0.971	51	3	0
G (139 primjera)	1.000	1.000	1.000	139	0	0
D (174 primjera)	0.933	0.965	0.949	168	12	6
GD (173 primjera)	0.768	0.959	0.853	166	50	7
DL (33 primjera)	1.000	1.000	1.000	33	0	0
DD (29 primjera)	1.000	1.000	1.000	29	0	0

GL(73 primjera)	1.000	1.000	1.000	73	0	0
d (15 primjera)	0.917	0.733	0.815	11	1	4

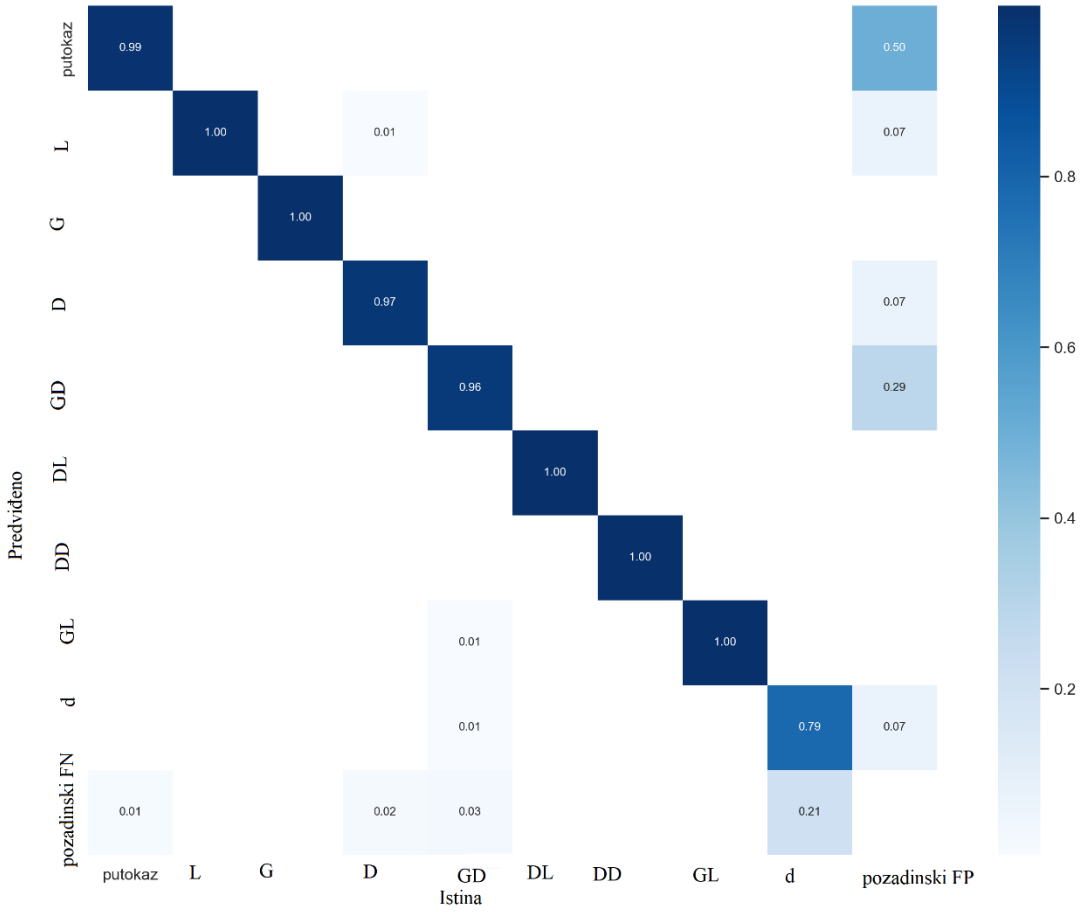
Također, u svrhu dodatne provjere, nakon samog treninga može se provesti dodatna validacija koristeći generiranu *best.pt* datoteku. Validacija je važan korak u razvoju modela strojnog učenja jer omogućuje provjeru performansi modela izvan okvira trening skupa podataka, odnosno na validacijskom skupu podataka, što omogućuje bolje razumijevanje njegove stvarne sposobnosti generalizacije. Postoje tri glavna razloga zbog kojih se provodi validacija:

- Procjena performansi: validacija pruža objektivan način za procjenu izvedbe modela na stvarnim podacima izvan trening skupa. To je bitno jer visoka točnost na trening skupu ne jamči nužno dobre performanse na novim podacima.
- Sprječavanje prenaučenosti (*engl. overfitting*): validacija pomaže u otkrivanju prenaučenosti. Model koji je prenaučen na trening skupu može loše generalizirati i ostvarivati niske performanse na novim podacima. Rezultati validacije pomažu u prepoznavanju i sprječavanju ovakvih situacija.
- Podešavanje hiperparametara: validacija omogućuje podešavanje hiperparametara modela kako bi se postigle bolje performanse modela. Ovo uključuje podešavanje parametara kao što su stopa učenja ili broj epoha, kako bi se postigla optimalna konfiguracija modela.

Naredba koja se koristila za validaciju je:

python test.py --weights runs\train\exp4\weights\best.pt --data data\data.yaml --img 420 --batch-size 16 --device 0 --task val

Dodatna validacija proširenog modela s 9 klasa, koji je rezultat najboljeg treninga prikazanog u prethodnom dijelu rada, bila je izvedena kako bi se provjerila pouzdanost rezultata dobivenih na *best.pt* modelu. Rezultati ove dodatne validacije predstavljeni su putem matrice zabune, koja je prikazana na slici 3.16. Korištenje matrice zabune omogućilo je detaljno analiziranje performansi modela na validacijskom skupu podataka. Vrijednosti koje su prikazane u matrici zabune odgovaraju onima koje su navedene u tablici 3.4. Ovo potvrđuje dosljednost rezultata između dvije različite metode validacije i dodatno jača vjerodostojnost performansi modela u toj najboljoj epohi. Primjetno je da matrica zabune i vrijednosti koje su dobivene iz tablice na ovom validacijskom skupu (koji se sastoji od 297 slika) nisu značajno odstupale od onih koje su zabilježene na trening



Slika 3.16. *Matrica zabune proširenog modela s 9 klasa dobivena iz validacijskog skupa podataka*

skupu podataka. Ovaj postupak dodatne validacije bio je ključan za potvrdu dosljednosti i pouzdanosti rezultata dobivenih na *best.pt* modelu, koji je izabran na temelju performansi tijekom treninga. To ukazuje na to da je model dobro naučio karakteristike svake pojedine klase tijekom treninga. Manje razlike u vrijednostima u tablicama i matricama zabune dobivenih iz treninga i validacije znače da model uspješno generalizira svoje znanje na novim podacima. Smanjenje točnosti ili drugih mjera performansi na validacijskom skupu u usporedbi s treningom je očekivano jer validacijski podaci nisu korišteni tijekom treninga. No, ključno je da su te razlike relativno male. Ovo sugerira da model nije prenaučen na trening podacima i da može ispravno klasificirati nove, neviđene primjere. Takvo konzistentno ponašanje modela na trening i validacijskom skupu podataka ukazuje na kvalitetan postupak treniranja. Model je uspio naučiti općenite karakteristike podataka koje se mogu primijeniti i na novim situacijama.

Za testiranje konačnog rješenja na ugradbenoj računalnoj platformi, s obzirom na to da ista ima određene nedostatke u vidu snage za rukovanjem YOLOv7 modelom, što je detaljnije objašnjeno u četvrtom poglavlju, bilo je potrebno istrenirati model koji će moći vršiti detekciju na

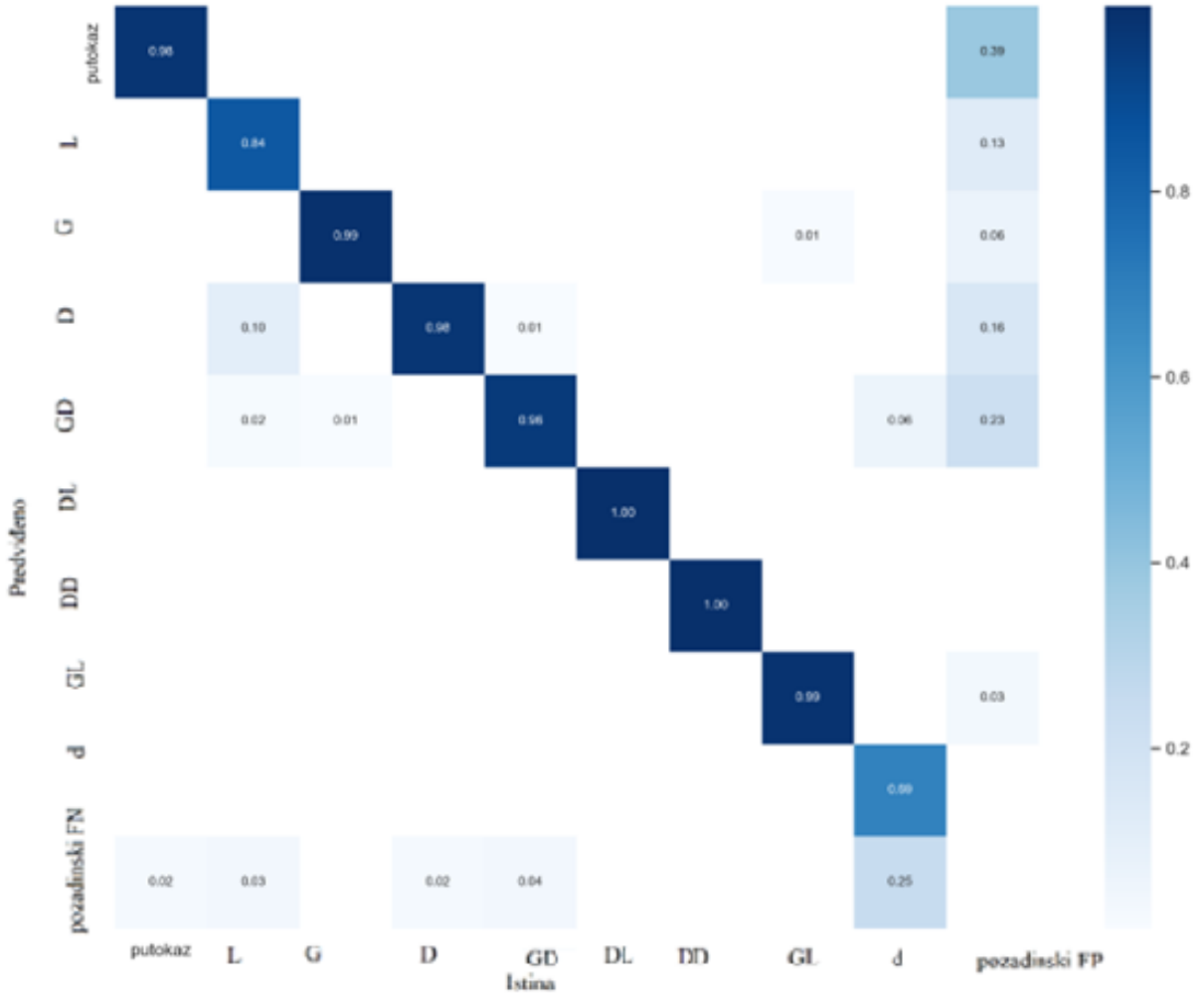
ugradbenoj računalnoj platformi. Kao rezultat toga, uzet je alternativni pristup treniranjem modela pomoću *yolov7-tiny.pt* težina. *yolov7-tiny.pt* je varijanta modela YOLOv7 koja je posebno dizajnirana da bude manja i brža, što ju čini prikladnijom za implementaciju na uređaje sa smanjenim resursima. Korištenje težina *yolov7-tiny.pt* pružilo je praktično rješenje koje je zadržalo bit izvornog modela YOLOv7, uz istovremeno prilagođavanje ograničenjima platforme. Naredba koja je korištena za trening ove verzije modela je:

python train.py --weights yolov7-tiny.pt --cfg cfg\training\yolov7_custom.yaml --data data\data.yaml --hyp data\hyp.scratch.custom.yaml --epochs 500 --batch-size 16 --img-size 420 --workers 4 --device 0 --name tiny

Trening je proveden ukupno 9 puta, a bitno je za naglasiti kako model i dalje radi odlično, unatoč tomu što je arhitektura pojednostavljena. Nakon devetog treninga, performanse modela se nisu poboljšale pa je to ujedno bio i posljednji trening. Za izbor najboljeg modela primijenjena je strategija temeljena na minimizaciji gubitka na validacijskom skupu tijekom treninga. Konkretno, tijekom treniranja modela, sustav je automatski zadržao težine modela koje su rezultirale najmanjim gubicima na validacijskom skupu. Te optimalne težine su spremljene u datoteku imena *best.pt*. Trening koji se pokazao kao najuspješniji izvodio se 7.435 h. U tablici 3.5. prikazani su rezultati treninga *yolov7-tiny.pt* modela, dok je na slici 3.17. prikazana matrica zabune za isti model. Na slici 3.18. prikazani su pojedinačni grafovi generirani nakon treninga.

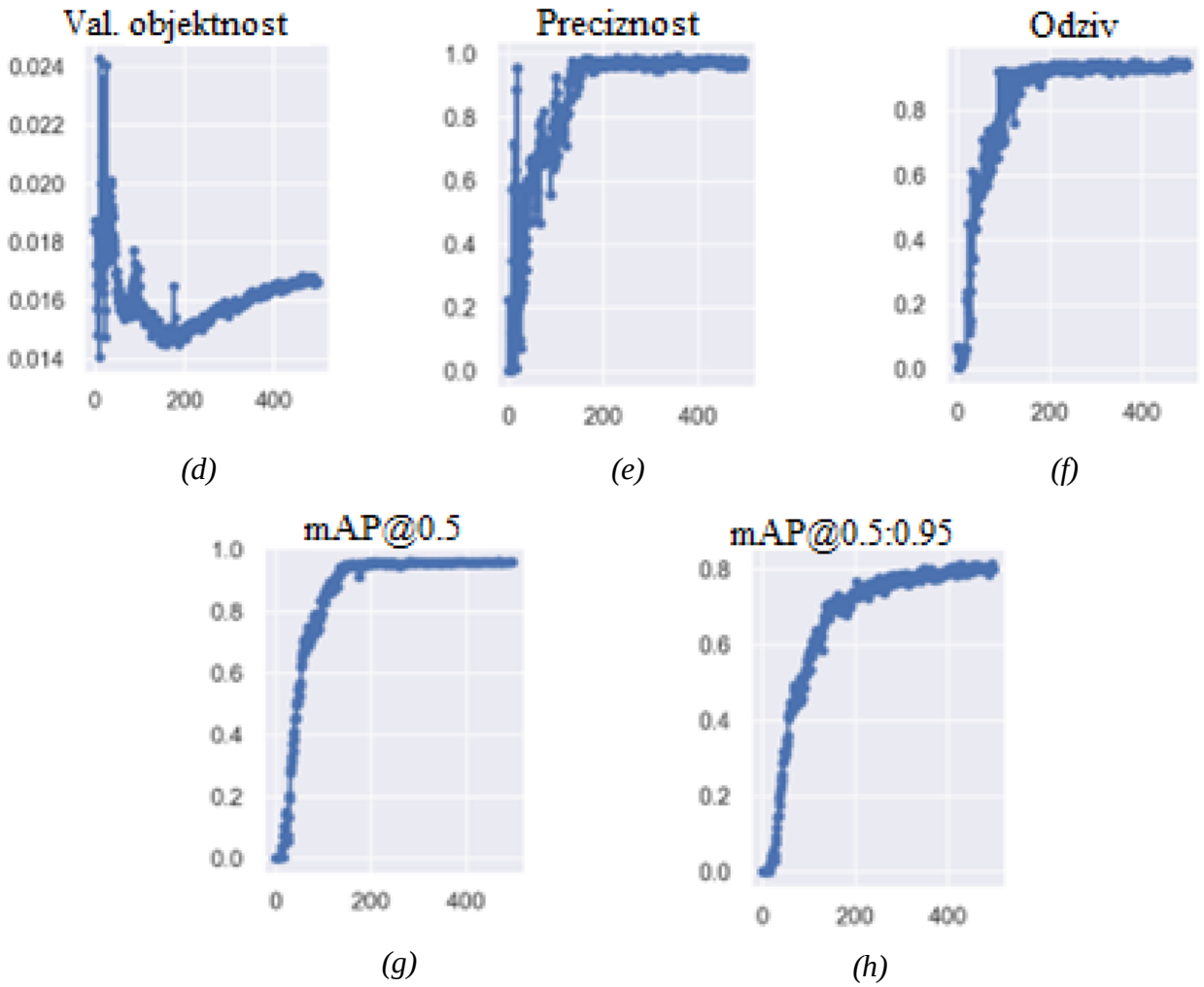
Tablica 3.5. *Konačni rezultati treninga yolov7-tiny.pt modela za detekciju putokaza i smjerova na njima na trening skupu podataka*

Klasa	Preciznost	Odziv	F1 mjera	TP broj	FP broj	FN broj
putokaz (984 primjera)	0.716	0.980	0.827	964	383	20
L (158 primjera)	0.868	0.835	0.851	132	20	26
G (437 primjera)	0.943	0.988	0.965	432	26	5
D (362 primjera)	0.861	0.978	0.916	354	57	8
GD (415 primjera)	0.807	0.959	0.876	398	95	17
DL (33 primjera)	1.000	1.000	1.000	33	0	0
DD (29 primjera)	1.000	1.000	1.000	29	0	0
GL (79 primjera)	0.975	0.987	0.981	78	2	1
d (23 primjera)	1.000	0.652	0.789	15	0	8



Slika 3.17. *Matrica zabune za yolov7-tiny.pt model s 9 klasa na trening skupu podataka*

(a) *(b)* *(c)*



(d) *(e)* *(f)*

(g) *(h)*

Slika 3.18. *Generirani grafovi nakon treninga yolov7-tiny.pt modela za detekciju putokaza i smjerova na njima koji se pokazao najboljim: (a) graf graničnih okvira, (b) graf validacijskih graničnih okvira, (c) graf objektnosti, (d) graf validacijske objektivnosti, (e) graf preciznosti, (f) graf odziva, (g) mAP@0.5 graf, (h) mAP@0.5:0.95 graf*

U tablici 3.6. prikazani su rezultati validacije za epohu (465. epoha) u kojoj se model ostvario najbolje rezultate na validacijskom skupu podataka tijekom procesa treninga.

Tablica 3.6. *Rezultati validacije yolov7-tiny.pt modela sa 9 klasa na validacijskom skupu podataka*

Klasa	Preciznost	Odziv	F1 mjera	TP broj	FP broj	FN broj
putokaz (406 primjera)	0.749	0.978	0.848	397	133	9
L (51 primjer)	0.898	0.863	0.880	44	5	7
G (139 primjera)	0.938	0.986	0.961	137	9	2

D (174 primjera)	0.853	0.965	0.905	168	29	6
GD (173 primjera)	0.781	0.948	0.856	164	46	9
DL (33 primjera)	1.000	1.000	1.000	33	0	0
DD (29 primjera)	1.000	1.000	1.000	29	0	0
GL(73 primjera)	0.935	0.986	0.960	72	5	1
d (15 primjera)	1.000	0.667	0.800	10	0	5

Dodatna validacija proširenog *yolov7-tiny.pt* modela s 9 klasa, koji je rezultat najboljeg treninga prikazanog u prethodnom dijelu rada, bila je izvršena kako bi se potvrdila pouzdanost rezultata dobivenih iz *best.pt* modela. Rezultati ove dodatne validacije su predstavljeni u obliku matrice zabune, prikazane na slici 3.19, što je omogućilo detaljnu analizu performansi modela na validacijskom skupu podataka. Vrijednosti iz matrice zabune su uspoređene s onima navedenima u tablici 3.6, što potvrđuje dosljednost rezultata između dvije različite metode validacije i dodatno jača vjerodostojnost performansi modela u najboljoj epohi. Važno je napomenuti da nema odstupanja između rezultata matrice zabune i vrijednosti iz tablice na ovom validacijskom skupu, koji se sastoji od 297 slika. Konzistentnost performansi modela na treningu i validaciji ukazuje na to da je model pravilno naučio karakteristike svake klase tijekom treninga. Manje razlike između treninga i validacije su očekivane, jer validacijski podaci nisu korišteni tijekom treniranja. No, važno je napomenuti da su te razlike relativno male, što sugerira da model nije prenaučen i sposoban je ispravno klasificirati nove primjere. Takva dosljednost modela na treningu i validaciji ukazuje na visoku kvalitetu procesa treniranja i sposobnost modela da općenito generalizira svoje znanje na nove situacije. Nakon što je model istreniran na trening skupu podataka, ključno je vizualno provjeriti i analizirati rezultate kako bismo razumjeli njegovu sposobnost detekcije putokaza i smjera na njima. Iako su kvantitativni rezultati, poput odziva i preciznosti, važni za ocjenu performansi modela, kvalitativna analiza, odnosno stvarno gledanje rezultata detekcije, može pružiti dublji uvid u to kako model djeluje i gdje može doći do potencijalnih problema. Na slici 3.20. prikazani su primjeri ispravne detekcije putokaza i smjera na njima. Na slici 3.21. prikazan su primjeri gdje putokazi i smjerovi na njima nisu ispravno ili potpuno detektirani, čiji uzrok može biti ograničenje trening skupa slika te slični objekti (na primjer, ako su na slici prisutni objekti slični putokazima).

Slika 3.19. *Matrica zabune za yolov7-tiny.pt model s 9 klasa na validacijskom skupu podataka*

(a)

(b)

Slika 3.20. *Primjer ispravno detektiranih putokaza i smjerova na njima*

Slika 3.21. *Primjer detektiranog putokaza koji nije putokaz. Također, smjerovi na ispravno detektiranom putokazu nisu detektirani*

Cjelokupan kod koji se koristio za izradu algoritma zasnovanog na YOLOv7 modelu za detekciju putokaza i smjera na njima može se pronaći u elektroničkom prilogu na DVD-u u mapi Prilog P.3.2.

3.4. Algoritam za razumijevanje teksta na putokazima

U procesu implementacije komponente razumijevanja teksta, evaluirane su različite OCR biblioteke. U početku su testirani easyOCR i TesseractOCR, ali nisu dali zadovoljavajuće rezultate za specifične zahtjeve projekta. Slijedom toga, donesena je odluka da se PaddleOCR s podrškom za engleski jezik integrira u *detect.py* skriptu modela YOLOv7. Čitanje teksta vrši se samo na području detektiranog putokaza ili više njih, a to je omogućeno tako da izrezani putokaz bude ulaz funkciji za razumijevanje teksta. Sami tekst prilikom detekcije poravnat je sa oznakom 'putokaz' te prelazi u novi red, a da pritom ne ide preko samog graničnog okvira. Izlaz PaddleOCR-a su koordinate detektiranog teksta, tekst i vjerojatnost točnosti. Upravo zbog ovog posljednjeg, moguće je odvojiti netočno pročitani tekst od ispravnog postavljanjem praga (*engl. threshold),* pa je tako na slici 3.22. prikazana usporedba korištenja PaddleOCR-a prije i nakon postavljanja praga. Na slici 3.22. (a) PaddleOCR iščitava tekst čak i ako nije potpuno siguran u njegovu ispravnost, dok se na slici 3.22. (b), zbog uvođenja praga razumijevanja teksta, netočan tekst ne pokazuje („rO“). Također, tekst se zadržava unutar širine graničnog okvira putokaza. PaddleOCR nije prikladan za implementaciju na korištenoj ugradbenoj računalnoj platformi zbog ograničenja arhitekture platforme. Hardverska i softverska ograničenja ugradbene računalne platforme ne podržavaju zahtjeve i ovisnosti PaddleOCR-a te je zbog toga kao alternativa korišten i implementiran TesseractOCR. TesseractOCR nije toliko zahtjevan i može se implementirati na ugradbene računalne platforme s ograničenim resursima, pružajući samo tekst kao izlazni rezultat. Lokacija i logika ispisa teksta jednaka je kao i u prvom slučaju.

(a)

(b)

Slika 3.22. *(a) Prikaz rezultata PaddleOCR-a bez postavljenog praga razumijevanja teksta, (b) prikaz rezultata PaddleOCR-a sa postavljenim pragom razumijevanja teksta i implementiranom logikom ispisa teksta.*

. Kod korišten za izradu algoritama za razumijevanje teksta njima može se pronaći u elektroničkom prilogu na DVD-u u mapi Prilog P.3.3.

4. VERIFIKACIJA RADA IZRAĐENIH ALGORITAMA ZA DETEKCIJU PUTOKAZA U SCENI I RAZUMIJEVANJE TEKSTA NA NJIMA

Verifikacija i evaluacija rješenja je ključni korak u procjeni performansi i pouzdanosti implementiranog algoritma. Ovaj proces uključuje temeljito ispitivanje mogućnosti detekcije modela YOLOv7, zajedno s integriranom komponentom za razumijevanje teksta pomoću PaddleOCR-a, odnosno Tesseract OCR-a, na slikama s kojima se model do sada nije suočio. Provođenjem temeljite provjere i evaluacije, postaje moguće dobiti uvid u jake i slabe strane algoritma i ukupnu učinkovitost u scenarijima iz stvarnog svijeta. Kroz proces verifikacije i evaluacije rješenja može se objektivno procijeniti njegova učinkovitost i pouzdanost, čime se dobivaju vrijedni uvidi za daljnje usavršavanje i optimizaciju. U potpoglavlju 4.1. opisan je proces verifikacije i testiranja na osobnom računalu, dok je u 4.2. potpoglavlju opisan proces verifikacije i testiranja na ugradbenoj računalnoj platformi, odnosno na Raspberry Pi 4 platformi. Prilikom verifikacije i evaluacije rješenja, ključno je provesti odvojene testove na različitim arhitekturama sustava. Ovo je neophodno jer izvedba i ponašanje rješenja mogu varirati ovisno o osnovnim konfiguracijama hardvera i softvera. Testiranjem rješenja na različitim arhitekturama sustava može se procijeniti njegova robusnost i osigurati da dosljedno radi u različitim okruženjima. Ovaj pristup omogućuje prepoznavanje potencijalnih ograničenja ili ovisnosti koje mogu nastati zbog specifičnih hardverskih ili softverskih konfiguracija.

4.1. Verifikacija i evaluacija rada na osobnom računalu

Provjera i testiranje rješenja provedeno je na osobnom računalu sa specifičnim hardverskim i softverskim konfiguracijama. Računalo korišteno u tu svrhu bilo je opremljeno Intel Core i7 procesorom šeste generacije, 16 GB RAM-a i NVIDIA GeForce GTX 1060 GPU s 6 GB memorije. Korišteni operacijski sustav bio je Windows 10.

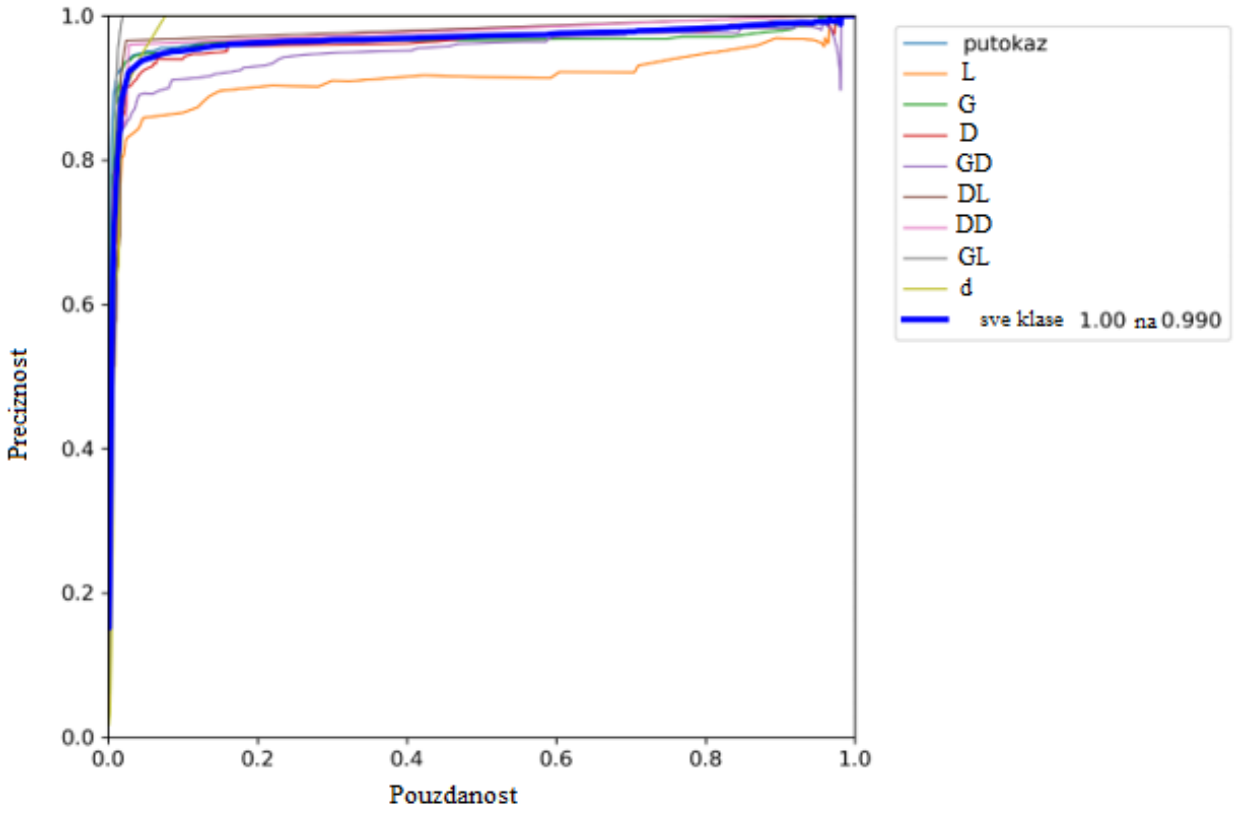
Nakon dobivanja najboljeg modela iz procesa treninga, provedeno je temeljito testiranje modela na namjenskom skupu testnih slika. Ove testne slike pažljivo su odabrane kako bi predstavljale scenarije iz stvarnog svijeta i obuhvatile različite zahtjevne uvjete s kojima bi se model mogao susresti. Svrha testiranja modela bila je procijeniti performanse i sposobnosti generalizacije obučenog modela na neviđenim podacima. Tijekom faze testiranja, istrenirani model primijenjen je na testne slike (568 slika), a njegovi rezultati su analizirani. Izlazi su uključivali otkrivene granične okvire oko putokaza i smjerova na njima te odgovarajuće oznake

klasa. Ti su rezultati zatim uspoređeni s istinitim graničnim okvirima testnih slika kako bi se procijenila točnost i robusnost modela. Naredba korištena za testiranje modela je:

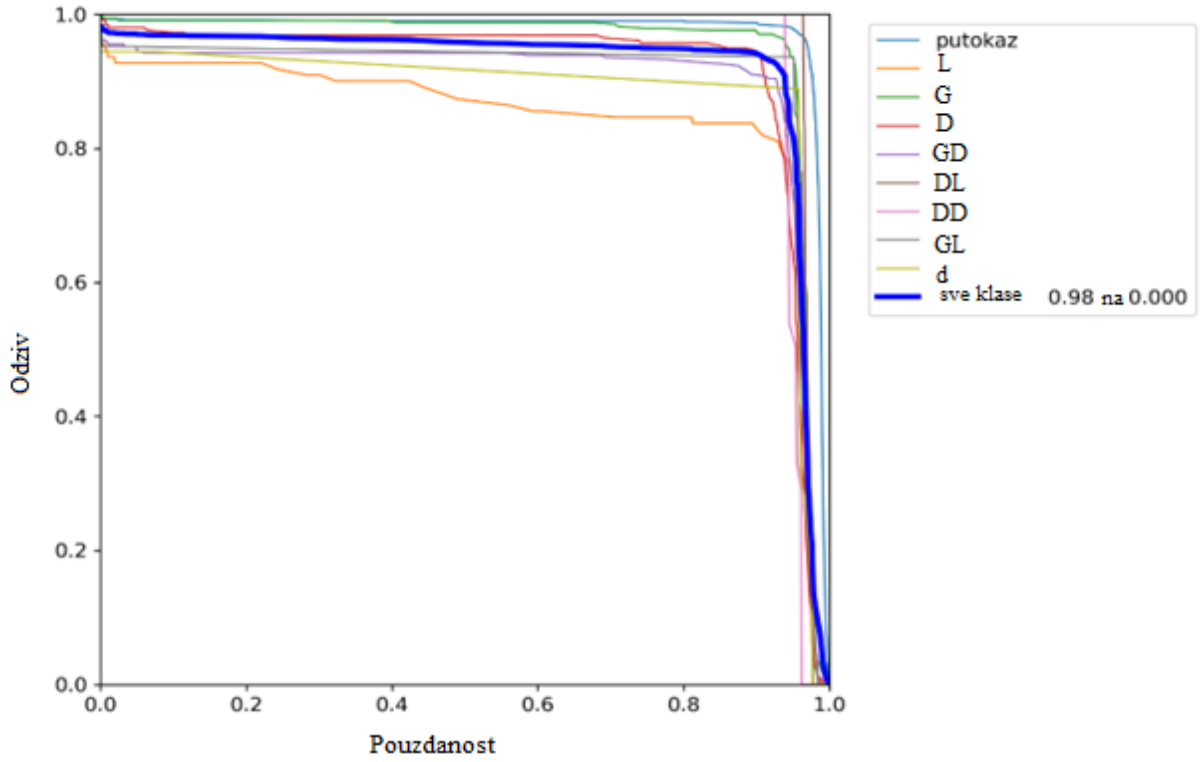
python test.py --weights runs\train\exp4\weights\best.pt --data data\data.yaml --img 420 --batch-size 16 --device 0 --name finale_test --task test

Mjerila evaluacije kao što su preciznost i odziv grafički su oblikovana za procjenu izvedbe modela za detekciju putokaza i smjerova na njima i prikazana na slici 4.1 (na apscisi su vrijednosti pouzdanosti, dok se na ordinati nalaze vrijednosti preciznosti i odziva), dok su pojedinačni rezultati po klasama prikazani u tablici 4.1. Također treba uzeti u obzir i vrijeme potrebno za obradu slike. Prilikom detekcije prisutan je i algoritam obrade teksta, što usporava proces. U procesu detekcije putokaza i smjera na njima s integriranim PaddleOCR-om, vrijeme obrade jedne slike iznosi 1.79 ms. Bez PaddleOCR-a, dakle samo za detekciju putokaza i smjera na njima, vrijeme obrade jedne slike iznosi 0.11 ms. Ti rezultati idu u prilog činjenici da se model može koristiti u stvarnom vremenu.

Također, u svrhu dubljeg testiranja isprobana je i detekcija u kombinaciji sa TesseractOCR-om, te tu vrijeme potrebno za obradu jedne slike iznosi 0.62 ms. Vrijeme za obradu slike bez TesseractOCR-a, i dalje iznosi 0.11 ms, što uključuje detekciju putokaza i smjera na njima. Prikaz



(a)



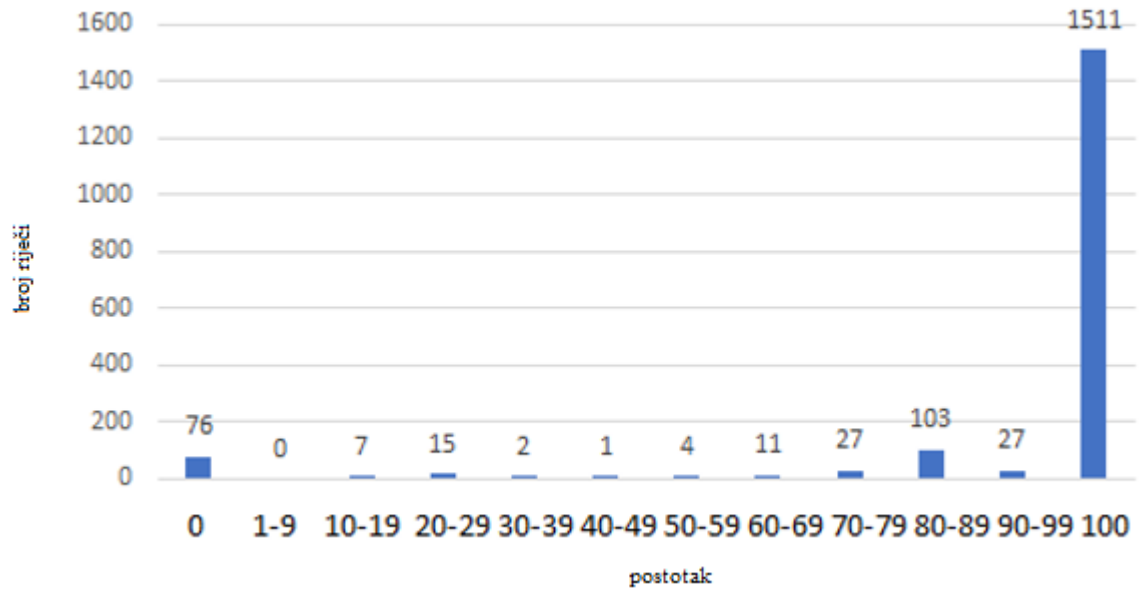
(b)

Slika 4.1. *(a) Graf preciznosti i (b) graf odziva dobiveni nakon testiranja modela na testnom skupu slika.*

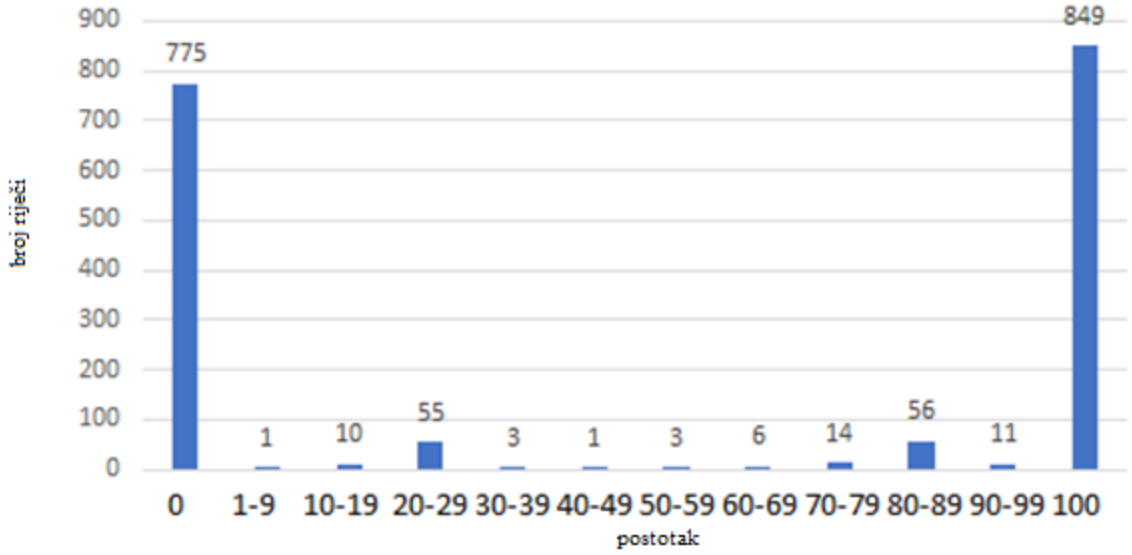
Tablica 4.1. *Prikaz rezultata testiranja modela za detekciju putokaza i smjera po klasama*

Klasa	Preciznost	Odziv	F1 mjera	TP broj	FP broj	FN broj
putokaz (731 primjer)	0.712	0.989	0.828	723	292	8
L (111 primjer)	0.933	0.874	0.903	97	7	14
G (340 primjera)	0.854	0.979	0.912	333	57	7
D (257 primjera)	0.877	0.969	0.921	249	35	8
GD (314 primjera)	0.816	0.917	0.864	288	65	26
DL (28 primjera)	1.000	1.000	1.000	28	0	0
DD (24 primjera)	1.000	1.000	1.000	24	0	0
GL(63 primjera)	1.000	1.000	1.000	63	0	0
d (18 primjera)	0.947	1.000	0.973	18	1	0

rezultata detekcije s TesseractOCR algoritmom razumijevanja teksta na osobnom računalu nalazi se na slici 4.2. Na apscisi se nalaze intervali postotka u kojima pročitana riječ odgovara izvornoj riječi, dok se na ordinati nalazi ukupan broj riječi u svim putokazima na testnom skupu slika. Za provjeru intervala postotka unutar kojega se nalazi pročitana riječ, korišten je Levenshteinov algoritam udaljenosti [65]. Algoritam je mjera razlike između dvaju nizova, odnosno dviju riječi. Izračunava minimalni broj uređivanja jednog znaka (umetanja, brisanja ili zamjene) potrebnih za transformaciju jednog niza u drugi. U kontekstu provjere sličnosti između pročitane riječi i izvorne riječi, Levenshteinov algoritam udaljenosti može biti izuzetno koristan. Ako je udaljenost između dviju riječi mala, to obično znači da su riječi vrlo slične ili čak identične. Ako je udaljenost veća, to ukazuje na veću razliku između dviju riječi. Prema slici 4.2. (a), vidljivo je da PaddleOCR pokazuje puno bolje rezultate od TesseractOCR-a (slika 4.2. (b)) jer od svih riječi, 85% pročitanih riječi odgovaraju izvornima, dok kod TesseractOCR-a, taj postotak iznosi 48%. U tablici 4.2. prikazani su primjeri usporedbe pročitane i izvorne riječi po intervalima postotaka za korištene algoritme razumijevanja teksta. Simbol „-“ u tablici kod izvorne riječi označava da ne postoji niti jedna riječ u tom intervalu postotka podudaranja između izvorne i pročitane riječi, dok navedeni simbol kod pročitane riječi ukazuje na to da ništa nije pročitano, odnosno da korišteni OCR ništa nije uspio pročitati. Učinkovitost OCR-a se smanjuje kada je tekst na putokazu nejasan, ili kad je sam putokaz malih dimenzija.



(a)



(b)

Slika 4.2. *(a) Rezultati PaddleOCR-a prilikom izvođenja detekcije na osobnom računalu, (b) Rezultati TesseractOCR-a prilikom izvođenja detekcije na osobnom računalu*

Tablica 4.2. *Primjeri pročitanih riječi po intervalima postotka podudaranja između izvorne i pročitane riječi za PaddleOCR i TesseractOCR algoritme.*

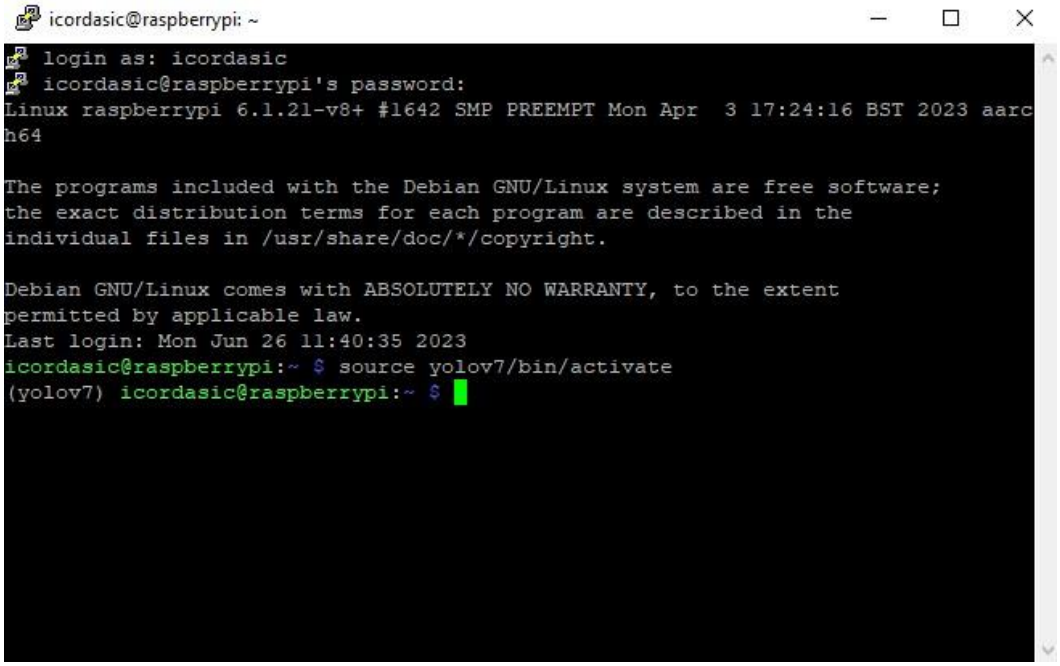
Interval postotka	PaddleOCR algoritam		TesseractOCR algoritam	
	Izvorna riječ	Pročitana riječ	Izvorna riječ	Pročitana riječ
0%	Zagreb	-	Ljubljana	-
1-9%	-	-	22:00-06:00h	0
10-19%	Split - centar	nra	1000 m	0
20-29%	Makarska	Ma	200 m	0
30-39%	E65	62	50 m	0
40-49%	11 km	km	11 km	km
50-59%	A1	AT	Trogir	Treggo
60-69%	Mostar	Mostar en	Split	TSplit
70-79%	Mostar	Mostaram	550 m	350 m
80-89%	Opuzen	OpuzenQ	Trieste	Triesteo
90-99%	Dubrovnik	1Dubrovnik	Ljubljana	Ljubljana@
100%	Slano	Slano	Opatija	Opatija

4.2. Verifikacija i evaluacija rada na Raspberry Pi 4 ugradbenoj računalnoj platformi

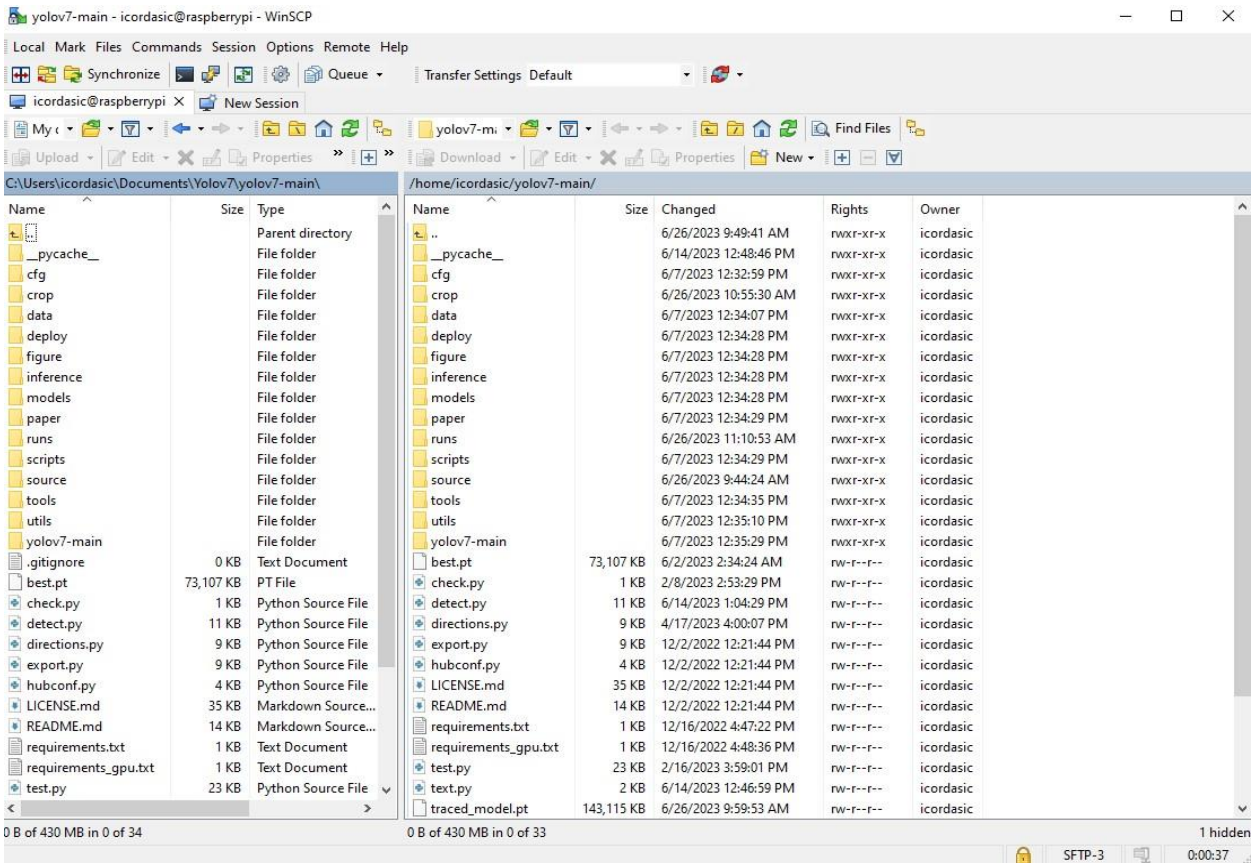
Verifikacija i evalucija razvijenog rješenja provedena je na Raspberry Pi 4 ugradbenoj računalnoj platformi kako bi se rješenje testiralo u uvjetima platforme sa smanjenim resursima.

Raspberry Pi 4 [66] je računalo malih dimenzija s jednom pločom koje nudi jeftino i energetski učinkovito rješenje za različite računalne zadatke. Sadrži četverojezgreni ARM Cortex-A72 procesor, u rasponu od 1,5 GHz do 2,0 GHz, i dostupan je u različitim konfiguracijama RAM-a, kao što su 2 GB, 4 GB ili 8 GB. Pruža dovoljno računalne snage za mnoge aplikacije, uključujući računalni vid i zadatke strojnog učenja. Za spajanje na Raspberry Pi 4 korišten je softver *PuTTY*, prikazan na slici 4.3. *PuTTY* je popularan SSH (*engl. Secure Shell - SSH*) klijent koji omogućuje siguran udaljeni pristup računalu ili uređaju preko mreže. Korisnicima omogućuje uspostavljanje sigurne veze s uređajem, pružajući sučelje naredbenog retka za izvršavanje naredbi i daljinsko upravljanje sustavom. Za prijenos datoteka između osobnog računala i ugradbene računalne platforme korišten je *WinSCP*, čije je sučelje prikazano na slici 4.4. Lijevi dio slike prikazuje datotečnu hijerarhiju na osobnom računalu, dok desni dio pripada Raspberry Pi 4. *WinSCP* je besplatni SFTP (*engl. SSH File Transfer Protocol - SFTP*) klijent otvorenog koda za Windows. Omogućuje siguran prijenos datoteka između lokalnog Windows stroja i udaljenog poslužitelja, kao što je Raspberry Pi 4. Uz *WinSCP*, datoteke se mogu lako kopirati, uređivati te pojednostavljuje proces prijenosa potrebnih podataka i skripti. Također, prije pokretanja YOLOv7 modela, bilo je potrebno instalirati sve potrebne ovisnosti i biblioteke čije verzije moraju odgovarati istoimenima s osobnog računala.

Zbog ograničenih resursa i hardverskih ograničenja Raspberry Pi 4, nije bilo moguće provesti stvarno testiranje koje bi uključivalo detaljne metrike za procjenu performansi, kao što su preciznost, odziv, TP, FN i FP vrijednosti.



Slika 4.3 *Putty terminal.*



Slika 4.4. *Sučelje WinSCP-a, alata koji omogućuje jednostavno dijeljenje datoteka između klijenta i poslužitelja.*

Umjesto toga, korišten je pojednostavljeni pristup usmjeren na otkrivanje, gdje je naglasak bio na detektiranju objekata i procjeni performansi na temelju vizualnog pregleda, a ne na generiranju detaljnih grafikona ili usporedbi detektiranih graničnih okvira s istinitim graničnim okvirima. Nadalje, kao što je već prethodno spomenuto u trećem poglavlju, arhitektura ugradbene računalne platforme ne podržava korištenje PaddleOCR-a. Stoga je bilo potrebno alternativno rješenje koje je dovelo do upotrebe TesseractOCR-a. Iako može imati drugačije karakteristike izvedbe u usporedbi s PaddleOCR-om, poslužio je kao održiva opcija za prepoznavanje teksta na Raspberry Pi 4. Obrada jedne slike potrajala je 5.84 ms, što je u usporedbi s vremenom obrade slike na osobnom računalu znatno sporije (0.62 ms za detekciju u kombinaciji sa TesseractOCR-om). Ukoliko se vrši detekcije bez TesseractOCR algoritma za razumijevanje teksta, vrijeme za obradu jedne slike iznosi 3.48 ms, što je u usporedbi s vremenom za obradu jedne slike na računalu bez algoritma za razumijevanje teksta znatno sporije (0.11 ms). Naredba koja se koristila za provođenje detekcije je:

python detect.py --weights best.pt --source source/ --img-size 448

gdje *source* predstavlja putanju do mape u kojoj su smještene slike na kojima će se detekcija provoditi. U ovom slučaju, sve testne slike su prebačene u istoimenu datoteku. Zbog toga što je nemoguće izvesti testiranje te uslijed toga nema grafičkog prikaza rezultata, ne može se previše toga reći o samoj izvedbi modela, osim što se može napraviti usporedba razumijevanja teksta. Rezultati razumijevanja teksta su konzistentni s onima dobivenima pomoću TesseractOCR-a na osobnom računalu. Vizualnim pregledom uočavaju se određene razlike u pokretanju modela na osobnom računalu i ugradbenoj računalnoj platformi, ali model generalno dobro izvršava detekciju putokaza i smjera na njima. Na slici 4.5. prikazana je usporedba detekcije putokaza i smjera na njima na osobnom računalu sa PaddleOCR-om i Raspberry Pi 4 sa TesseractOCR-om.

(a)

(b)

Slika 4.5. *(a) Rezultat detekcije putokaza i smjera na njima na osobnom računalu uz korištenje PaddleOCR-a, (b) rezultat detekcije putokaza i smjera na njima na Raspberry Pi 4 ugradbenoj računalnoj platformi uz korištenje TesseractOCR-a.*

Analizirajući rezultate detekcije putokaza i smjera na njima u kombinaciji s algoritmom za razumijevanjem teksta na osobnom računalu i Raspberry Pi 4 ugradbenoj računalnoj platformi, jasno je da postoji niz čimbenika koji utječu na konačne performanse rješenja. Na osobnom računalu algoritam za detekciju putokaza i smjera na njima je pokazao odlične performanse, posebno za klasu *putokaz*, dok je isti i za druge klase postigao visoku preciznost te visok odziv. Međutim, zanimljiva je pojava u klasi *d*, gdje, iako je preciznost bila 100%, odziv je bio znatno niži, što sugerira da algoritam često propušta detektirati instance te klase. To može ukazivati na potencijalne slabosti u trening skupu slika ili modeliranju koje treba dalje istražiti. Međutim, prava kompleksnost ove analize dolazi prilikom usporedbe performansi rješenja na osobnom računalu i na Raspberry Pi 4 ugradbenoj računalnoj platformi. Raspberry Pi 4, iako je cijenjen zbog svoje portabilnosti i cjenovne pristupačnosti, ima inherentna hardverska ograničenja u odnosu na osobna računala. Te razlike u hardverskoj snazi mogu rezultirati značajnim padom u performansama rješenja, posebno kada se koristi YOLOv7. Dodatno, korištenje različitih OCR algoritama dodalo je još jedan sloj kompleksnosti u analizu. Iako su oba OCR algoritma, PaddleOCR i TesseractOCR, široko korišteni, rezultati su pokazali značajne razlike u njihovoj točnosti. Ovo ukazuje na važnost pažljivog odabira algoritma i prilagodbe prema specifičnim zahtjevima projekta. Nedostatak potpunog testiranja na Raspberry Pi platformi također je velika prepreka. Oslanjajući se samo na vizualni pregled, postoji rizik od propuštanja ključnih informacija o izvedbi rješenja, što može dovesti do pogrešnih zaključaka. Dok su početni rezultati rješenja na osobnom računalu obećavajući, prijenos rješenja na platforme s manje računalne snage zahtijeva dodatnu pažnju i optimizaciju. Ova analiza ukazuje na važnost razumijevanja kako različiti hardver i softverski alati mogu utjecati na performanse i naglašava potrebu za sveobuhvatnim testiranjem i validacijom u različitim okruženjima.

5. ZAKLJUČAK

Cilj rada bio je napraviti algoritam za izdvajanje putokaza iz scene i razumijevanje teksta na njima. Dodatno, unutar detektiranog putokaza, bilo je potrebno detektirati smjer na koji putokaz pokazuje. Rad se zasniva na dubokom učenju, konkretno na YOLOv7 modelu, korištenom za detekciju putokaza i smjera na njima, u kombinaciji s OCR-om korištenim za razumijevanje teksta. Algoritam se razvijao u Python programskom jeziku. Algoritam je, nakon obrade slike i detekcije putokaza, koristio OCR za čitanje i razumijevanje teksta na detektiranim putokazima. Osim detekcije putokaza i razumijevanja teksta, algoritam je vrlo uspješno detektirao i smjer na koji putokaz pokazuje. Testiranje je pokazalo da algoritam u velikoj mjeri točno detektira putokaze i smjer na njima te čita tekst na njima, ali ima poteškoća s detekcijom u situacijama kada je putokaz djelomično zaklonjen ili kada je kvaliteta slike niska. Učinkovitost OCR-a u čitanju teksta smanjuje se kod malih putokaza ili kada je tekst na putokazu nejasan. Rješenje je pokazalo potencijal za upotrebu u sustavima navigacije, ali zahtijeva daljnje optimizacije i poboljšanja kako bi bilo potpuno funkcionalno u realnom vremenu i svim uvjetima na cesti. Rješenje se pokazalo korisnim u različitim uvjetima osvjetljenja, ali bi se moglo još poboljšati u smislu veće otpornosti na različite uvjete osvjetljenja i vremenske uvjete. Zaključno, rješenje pokazuje obećavajuće rezultate, ali zahtijeva daljnje prilagodbe i poboljšanja kako bi se povećala njegova točnost i robusnost u različitim uvjetima.

LITERATURA

[1] „What is ADAS (Advanced Driver Assistance Systems)? – Overview of ADAS Applications | Synopsys“. https://www.synopsys.com/automotive/what-is-adas.html (pristupljeno 17. srpanj 2023.).

[2] M. Ahmad, „ADAS is back in the driver's seat at CES 2023“, *EDN*, 09. siječanj 2023. https://www.edn.com/adas-is-back-in-driving-seat-at-ces-2023/ (pristupljeno 01. kolovoz 2023.).

[3] „What is OCR? - Optical Character Recognition Explained - AWS“, *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/ocr/ (pristupljeno 17. srpanj 2023.).

[4] „Object Detection with Convolutional Neural Networks | by Yağmur Çiğdem Aktaş | Towards Data Science“. https://towardsdatascience.com/object-detection-with-convolutional-neural-networks-c9d729eedc18 (pristupljeno 17. srpanj 2023.).

[5] baeldung, „The Viola-Jones Algorithm | Baeldung on Computer Science“, 14. listopad 2022. https://www.baeldung.com/cs/viola-jones-algorithm (pristupljeno 17. srpanj 2023.).

[6] A. Rosebrock, „Histogram of Oriented Gradients and Object Detection“, *PyImageSearch*, 10. studeni 2014. https://pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/ (pristupljeno 17. srpanj 2023.).

[7] J. W. Howarth, H. H. C. Bakker, i R. C. Flemmer, „Feature-based Object Recognition“, u *2009 4th International Conference on Autonomous Robots and Agents*, velj. 2009, str. 375–379. doi: 10.1109/ICARA.2000.4804015.

[8] J. Redmon, S. Divvala, R. Girshick, i A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection“, u *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, lip. 2016, str. 779–788. doi: 10.1109/CVPR.2016.91.

[9] „IOU: What It Is, How It Works, and Examples“, *Investopedia*. https://www.investopedia.com/terms/i/iou.asp (pristupljeno 17. srpanj 2023.).

[10] „paper.pdf“. Pristupljeno: 01. kolovoz 2023. [Na internetu]. Dostupno na: https://ddd.uab.cat/pub/tfg/2017/tfg_71066/paper.pdf

[11] „YOLO Object Detection Explained: A Beginner's Guide“. https://www.datacamp.com/blog/yolo-object-detection-explained (pristupljeno 17. srpanj 2023.).

[12] „What is Learning Rate in Machine Learning“, *Deepchecks*. https://deepchecks.com/glossary/learning-rate-in-machine-learning/ (pristupljeno 17. srpanj 2023.).

[13] R. Alghifari, „When In Doubt, Go YOLO“, *Medium*, 05. lipanj 2021. https://rayfazt.medium.com/when-in-doubt-go-yolo-9ffc19ec2ac7 (pristupljeno 17. srpanj 2023.).

[14] „The History of YOLO Object Detection Models from YOLOv1 to YOLOv8 | Deci“. https://deci.ai/blog/history-yolo-object-detection-models-from-yolov1-yolov8/ (pristupljeno 17. srpanj 2023.).

[15] „What is Accuracy, Precision, Recall and F1 Score?“ https://www.labelf.ai/blog/what-is-accuracy-precision-recall-and-f1-score (pristupljeno 17. srpanj 2023.).

[16] „The history of YOLO: The origin of the YOLOv1 algorithm | SuperAnnotate“. https://www.superannotate.com/blog/yolov1-algorithm (pristupljeno 01. kolovoz 2023.).

[17] R. Girshick, „Fast R-CNN“. arXiv, 27. rujan 2015. Pristupljeno: 01. kolovoz 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1504.08083

[18] A. Pahinkar, P. Mohan, A. Mandal, i K. A, „Faster Region Based Convolutional Neural Network and VGG 16 for Multi-Class Tyre Defect Detection“, u *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, srp. 2021, str. 01–07. doi: 10.1109/ICCCNT51525.2021.9579855.

[19] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, i O. Buyukozturk, „Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types“, *Comput.-Aided Civ. Infrastruct. Eng.*, sv. 00, str. 1–17, stu. 2017, doi: 10.1111/mice.12334.
[20] „05 Duboko ucenje.pdf“. Pristupljeno: 01. kolovoz 2023. [Na internetu]. Dostupno na: https://moodle.srce.hr/2022-2023/pluginfile.php/7336899/mod_resource/content/3/05%20Duboko%20ucenje.pdf
[21] A. Singh, „Selecting the Right Bounding Box Using Non-Max Suppression (with implementation)“, *Analytics Vidhya*, 02. kolovoz 2020. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/ (pristupljeno 17. srpanj 2023.).
[22] „Inverse.AI“. https://inverseai.com/ (pristupljeno 17. srpanj 2023.).
[23] G. Boesch, „YOLOv7: The Most Powerful Object Detection Algorithm (2023 Guide)“, *viso.ai*, 21. svibanj 2023. https://viso.ai/deep-learning/yolov7-guide/ (pristupljeno 01. kolovoz 2023.).
[24] A. Bochkovskiy, C.-Y. Wang, i H.-Y. M. Liao, „YOLOv4: Optimal Speed and Accuracy of Object Detection“. arXiv, 22. travanj 2020. doi: 10.48550/arXiv.2004.10934.
[25] S.-H. Tsang, „Brief Review: YOLOv5 for Object Detection“, *Medium*, 11. lipanj 2023. https://sh-tsang.medium.com/brief-review-yolov5-for-object-detection-84cc6c6a0e3a (pristupljeno 01. kolovoz 2023.).
[26] V. Meel, „YOLOR - You Only Learn One Representation (What’s new, 2022)“, *viso.ai*, 06. kolovoz 2021. https://viso.ai/deep-learning/yolor/ (pristupljeno 01. kolovoz 2023.).
[27] K.-Y. Wong, „Official YOLOv7“. 17. srpanj 2023. Pristupljeno: 17. srpanj 2023. [Na internetu]. Dostupno na: https://github.com/WongKinYiu/yolov7
[28] „YOLOv7 Architecture Explanation“. https://www.cameralyze.co/blog/yolov7-architecture-explanation (pristupljeno 17. srpanj 2023.).
[29] Y. Swapna, M. S. Reddy, J. V. Sai, N. S. S. Krishna, i M. V. Teja, „Deep Learning Based Road Traffic Sign Detection and Recognition“, u *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, ruj. 2021, str. 1–4. doi: 10.1109/ICIRCA51532.2021.9545080.
[30] „GTSRB - German Traffic Sign Recognition Benchmark“. https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign (pristupljeno 01. kolovoz 2023.).
[31] A. Suriya Prakash, D. Vigneshwaran, R. Seenivasaga Ayyalu, i S. Jayanthi Sree, „Traffic Sign Recognition using Deeplearning for Autonomous Driverless Vehicles“, u *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, tra. 2021, str. 1569–1572. doi: 10.1109/ICCMC51019.2021.9418437.
[32] S. Vaidya, S. Kavthekar, i A. Joshi, „Leveraging YOLOv7 for Plant Disease Detection“, u *2023 4th International Conference on Innovative Trends in Information Technology (ICITIIT)*, velj. 2023, str. 1–6. doi: 10.1109/ICITIIT57246.2023.10068590.
[33] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, i N. Batra, „PlantDoc: A Dataset for Visual Plant Disease Detection“, u *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, sij. 2020, str. 249–253. doi: 10.1145/3371158.3371196.
[34] „Object Detection: Models, Architectures & Tutorial [2023]“. https://www.v7labs.com/blog/object-detection-guide, https://www.v7labs.com/blog/object-detection-guide (pristupljeno 17. srpanj 2023.).
[35] Q. Wu, Y. Zhou, i G. Liang, „A Text Detection and Recognition System Based on an End-to-End Trainable Framework from UAV Imagery“, ožu. 2019. doi: 10.1109/ROBIO.2018.8665259.

[36] „What is LSTM - Introduction to Long Short Term Memory“. https://intellipaat.com/blog/what-is-lstm/?US (pristupljeno 17. srpanj 2023.).
[37] „Optical Character Recognition using PaddleOCR | LearnOpenCV“, 14. lipanj 2022. https://learnopencv.com/optical-character-recognition-using-paddleocr/ (pristupljeno 17. srpanj 2023.).
[38] „Visual Geometry Group - University of Oxford“. https://www.robots.ox.ac.uk/~vgg/data/scenetext/ (pristupljeno 02. kolovoz 2023.).
[39] „ds4sd/icdar2023-doclaynet · Datasets at Hugging Face“. https://huggingface.co/datasets/ds4sd/icdar2023-doclaynet (pristupljeno 02. kolovoz 2023.).
[40] „COCO-Text V2.0“. https://bgshih.github.io/cocotext/ (pristupljeno 02. kolovoz 2023.).
[41] „Tesseract User Manual“, *tessdoc*. https://tesseract-ocr.github.io/tessdoc/ (pristupljeno 17. srpanj 2023.).
[42] J. Guo, R. You, i L. Huang, „Mixed Vertical-and-Horizontal-Text Traffic Sign Detection and Recognition for Street-Level Scene“, *IEEE Access*, sv. 8, str. 69413–69425, 2020, doi: 10.1109/ACCESS.2020.2986500.
[43] Y. L. Chaitra, R. Dinesh, M. Jeevan, M. Arpitha, V. Aishwarya, i K. Akshitha, „An Impact of YOLOv5 on Text Detection and Recognition System using TesseractOCR in Images/Video Frames“, u *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, srp. 2022, str. 1–6. doi: 10.1109/ICDSIS55133.2022.9915927.
[44] „Papers with Code - ICDAR 2013 Dataset“. https://paperswithcode.com/dataset/icdar-2013 (pristupljeno 24. kolovoz 2023.).
[45] „Papers with Code - ICDAR 2015 Dataset“. https://paperswithcode.com/dataset/icdar-2015 (pristupljeno 24. kolovoz 2023.).
[46] J. D. H. Droettboom Michael, „matplotlib: Python plotting package“. Pristupljeno: 17. srpanj 2023. [Na internetu]. Dostupno na: https://matplotlib.org
[47] „Introduction to NumPy“. https://www.w3schools.com/python/numpy/numpy_intro.asp (pristupljeno 17. srpanj 2023.).
[48] „opencv-python: Wrapper package for OpenCV python bindings.“ Pristupljeno: 17. srpanj 2023. [MacOS, Microsoft :: Windows, POSIX, Unix]. Dostupno na: https://github.com/opencv/opencv-python
[49] R. Python, „Image Processing With the Python Pillow Library – Real Python“. https://realpython.com/image-processing-with-the-python-pillow-library/ (pristupljeno 17. srpanj 2023.).
[50] K. Simonov, „PyYAML: YAML parser and emitter for Python“. Pristupljeno: 17. srpanj 2023. [OS Independent]. Dostupno na: https://pyyaml.org/
[51] K. Reitz, „requests: Python HTTP for Humans.“ Pristupljeno: 17. srpanj 2023. [OS Independent]. Dostupno na: https://requests.readthedocs.io
[52] „SciPy Tutorial | Beginners Guide to Python SciPy with Examples“, *Edureka*, 04. rujan 2019. https://www.edureka.co/blog/scipy-tutorial/ (pristupljeno 17. srpanj 2023.).
[53] „tqdm/tqdm“. tqdm developers, 17. srpanj 2023. Pristupljeno: 17. srpanj 2023. [Na internetu]. Dostupno na: https://github.com/tqdm/tqdm
[54] „DFG Traffic Sign Data Set“. https://www.vicos.si/resources/dfg/ (pristupljeno 02. kolovoz 2023.).
[55] *[4K 60fps] Scenic Drive with Panoramic Sea Views - Croatia Scenic Byway*, (29. veljača 2020.). Pristupljeno: 17. srpanj 2023. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=Ejq9iwZRR7Q
[56] *Driving Croatia: A7 around Rijeka - 4K drive on a spectacular croatian motorway*, (28. studeni 2020.). Pristupljeno: 17. srpanj 2023. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=aXGYBgjyx7g

[57] *Driving from Zagreb to Istra CROATIA HR*, (01. listopad 2022.). Pristupljeno: 17. srpanj 2023. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=rW71b1MzAFk
[58] *Driving in Croatia in 4K - Along the Adriatic Highway to Split*, (25. studeni 2021.). Pristupljeno: 17. srpanj 2023. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=Kst83z1EtrA
[59] *Driving in Croatia in 4K - Croatian highways | Baška Voda to Zadar.*, (28. studeni 2021.). Pristupljeno: 17. srpanj 2023. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=63TjiuwKvoI
[60] *Driving the Adriatic Coast Highway in Croatia to Dubrovnik*, (19. studeni 2021.). Pristupljeno: 17. srpanj 2023. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=x75pHN5Wnn4
[61] G. Boesch, „LabelImg for Image Annotation“, *viso.ai*, 11. veljača 2022. https://viso.ai/computer-vision/labelimg-for-image-annotation/ (pristupljeno 02. kolovoz 2023.).
[62] S. Kostadinov, „Understanding Backpropagation Algorithm“, *Medium*, 12. kolovoz 2019. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd (pristupljeno 17. srpanj 2023.).
[63] „Confusion Matrix in Machine Learning“, *GeeksforGeeks*, 15. listopad 2017. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (pristupljeno 17. srpanj 2023.).
[64] „Template Matching“. https://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html (pristupljeno 17. srpanj 2023.).
[65] E. Nam, „Understanding the Levenshtein Distance Equation for Beginners“, *Medium*, 27. veljača 2019. https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0 (pristupljeno 09. kolovoz 2023.).
[66] „Raspberry-Pi-4-Product-Brief.pdf“. Pristupljeno: 17. srpanj 2023. [Na internetu]. Dostupno na: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf

SAŽETAK

U sklopu ovog rada razvijen je algoritam za detekciju putokaza u sceni i razumijevanje teksta na njima, koristeći vlastitu bazu podataka. Cilj je bio izolirati putokaze iz scene i razumjeti tekst na njima, uključujući smjer na koji pokazuju. Algoritam je izgrađen koristeći metode dubokog učenja, odnosno YOLOv7 za detekciju putokaza i smjera na njima, u kombinaciji s OCR alatima za razumijevanje teksta. Razvoj algoritma proveden je u Python programskom okruženju. Testiranja su provedena na različitim slikama, koje su snimljene u različitim uvjetima. Rezultati su pokazali visoku točnost detekcije putokaza i smjera na njime te razumijevanja teksta, ali i prostor za poboljšanje u situacijama kada su putokazi djelomično zaklonjeni ili kada je kvaliteta slike niska. Algoritam predstavlja dobar korak prema poboljšanju navigacijskih sustava kroz detekciju i interpretaciju putokaza. Iako su rezultati obećavajući, daljnja optimizacija je potrebna kako bi se poboljšala performansa u realnom vremenu i u različitim uvjetima na cesti. U okviru testiranja, posebno je važno istaknuti primjenu implementiranih algoritama na Raspberry Pi 4 ugradbenoj računalnoj platformi. Iako su početni rezultati obećavajući, primjetna je potreba za dodatnom optimizacijom kako bi se osigurala pouzdana i brza obrada slika u stvarnom vremenu zbog softverskih i hardverskih ograničenja ugradbene računalne platforme. To uključuje optimizaciju hardverskih resursa, kao i prilagodbu algoritama kako bi se što efikasnije koristili ograničeni resursi ugradbenih računalnih platformi. Činjenica koja ide u prilog tom zaključku dolazi iz usporedbe performansi rada na osobnom računalu i ugradbenoj računalnoj platformi, gdje osobno računalo pokazuje bolje performanse te je veći spektar algoritama koji se mogu koristiti.

Ključne riječi: *detekcija, putokaz, smjer, duboko učenje, YOLOv7, PaddleOCR, TesseractOCR ugradbena računalna platforma*

ALGORITHM FOR EXTRACTING SIGNPOSTS FROM THE SCENE AND UNDERSTANDING THE TEXT ON THEM

ABSTRACT

As part of this work, an algorithm was developed for the detection of road signs in the scene and the understanding of the text on them, using our own database. The goal was to isolate the signposts from the scene and understand the text on them, including the direction they point to. The algorithm was built using deep learning methods, namely YOLOv7 to detect road signs and directions on them, combined with OCR tools for text understanding. The development of the algorithm was carried out in the Python programming environment. The tests were carried out on different photos, which were taken in different conditions. The results showed a high accuracy of the detection of road signs and the direction on them, as well as understanding of the text, but also room for improvement in situations where the road signs are partially obscured or when the image quality is low. The algorithm represents a good step towards the improvement of navigation systems through the detection and interpretation of road signs. Although the results are promising, further optimization is needed to improve real-time performance under different road conditions. As part of the testing, it is particularly important to highlight the application of the implemented algorithms on the Raspberry Pi 4 embedded computer platform. Although the initial results are promising, there is a noticeable need for further optimization to ensure reliable and fast real-time image processing due to the software and hardware limitations of the embedded computer platform. This includes the optimization of hardware resources, as well as the adaptation of algorithms to make the most efficient use of the limited resources of embedded computer platforms. The fact that supports this conclusion comes from a comparison of the performance of work on a personal computer and on an embedded computer platform, where the personal computer shows better performance and the spectrum of algorithms that can be used is greater.

Keywords: *detection, signpost, direction, deep learning, YOLOv7, PaddleOCR, TesseractOCR, embedded computer platform*

ŽIVOTOPIS

Ivan Čordašić rođen je 24. rujna 1999. u Vinkovcima. Završio je opću gimnaziju M. A. Reljkovića u Vinkovcima s odličnim uspjehom. Nakon srednje škole upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2021. godine stječe akademski naziv sveučilišni prvostupnik (*lat. baccalauereus*) inženjer računarstva. Iste godine upisuje diplomski sveučilišni studij automobilskog računarstva i komunikacija i postaje stipendist tvrtke TTTech Auto d.o.o. Aktivno se služi engleskim jezikom u govoru i pismu.

_____
Potpis autora

PRILOZI

- P.3.1. Slike korištene za proces izrade rješenja (priloženo na DVD-u uz rad)
- P.3.2. cjelokupan kod sa svim popratnim datotekama potrebnim za izradu algoritma za detekciju putokaza i smjera na njima (priloženu na DVD-u uz rad)
- P.3.3. kod za izradu algoritama za razumijevanje teksta (priloženu na DVD-u uz rad)