Automatsko zaustavljanje vozila na osnovu prometne signalizacije

Mikulić, Frane

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

Sveučilišni studij

AUTOMATSKO ZAUSTAVLJANJE VOZILA NA OSNOVU PROMETNE SIGNALIZACIJE

Diplomski rad

Frane Mikulić

Osijek, 2019.

SADRŽAJ

1. UVOD

Početak automobilske industrije seže daleko u prošlost, kada je davne 1886. godine, Karl Friedrich Benz izumio motor s unutarnjim izgaranjem [1]. Od toga doba do danas prošlo je mnogo godina, a automobil je značajno napredovao, kako u dizajnu tako i kroz tehnologiju koja se u njemu koristi. Današnji automobili koriste mnogobrojne tehnologije kako bi povećali razinu sigurnosti, udobnosti i unaprijedili način upravljanja vozilom. Sukladno tome, danas se razvijaju napredni sustavi za pomoć vozaču, ADAS (engl. *Advance Driver Asistance Systems*), koji vozačima uvelike olakšavaju upravljanje automobilom te nastoje maksimalno povećati sigurnost svih sudionika u prometu. Implementacijom ADAS u vozila, ali i njihovim napretkom kroz tehnologiju teži se potpuno autonomnom vozilu u budućnosti. Cilj je da autonomno vozilo u potpunosti zamijeni čovjeka kao vozača, a automobil postane samo sredstvo koje će se koristiti za prijevoz putnika do određenog mjesta.

Velika prepreka na putu do potpuno autonomnog vozila svakako je rad na potpunoj sigurnosti vozača, putnika i svih sudionika u prometu. Sigurnost uvelike narušavaju prometne nesreće koje su danas jedan od najvećih uzroka smrti u svijetu. Prema [2], 94% prometnih nesreća uzrokovano je ljudskim faktorom, a glavni je razlog upravo ljudska pogreška, odnosno loša procjena situacije u prometu. Prema istim istraživanjima, samo 2% uzroka prometnih nezgodna odnosi se na mehaničke kvarove automobila i iznenadne otkaze određenih dijelova automobila. Upravo iz tih razloga, pridaje se velika pažnja razvoju ADAS koji će, prema stručnjacima, u budućnosti biti potpuno nepogrešivi [2]. Stoga sve velike kompanije, uključene u rad na ADAS, pred sebe stavljaju veliki izazov i kreiraju revoluciju u automobilskoj industriji. Osim rješavanja problema prometnih nezgoda, ADAS bi trebali osigurati bolji i lakši protok automobila na prometnicama, kao i njihovu međusobnu komunikaciju te značajno smanjiti gužve na prometnicama. Vodeći se ovim ciljevima, ideja je da u budućnosti imamo znatno manje automobila na cestama, a veću iskoristivost svakog od njih. Naime, većini vozača automobil nakon kratke vožnje dugo vremena stoji i ne koristi se pa je ideja da jedan automobil obavi puno više vožnji, umjesto da stoji na parkingu dok vlasnik ne bude ponovo imao potrebu za njim. Automobil umjesto praznog hoda može obaviti nekoliko drugih vožnji i ponovno se vratiti po prvog putnika u određeno vrijeme i na taj način njegova iskoristivost je višestruko veća.

Glavne su zadaće današnjih ADAS razumijevanje scene, praćenje stanja vozača, planiranje kretanja te lokalizacija i mapiranje, odnosno određivanje položaja ostalih objekata u odnosu na vozilo. U rješavanju svih ovih problema autonomne vožnje veliku ulogu ima strojno učenje. Algoritmima strojnog učenja pokušava se naučiti računalo da obavlja sve ove zadatke, na način da mu se ugradi umjetna inteligencija.

U ovom radu naglasak se stavlja na razumijevanje scene i algoritam detekcije i prepoznavanja prometne signalizacije, konkretno onih prometnih znakova i signalizacije koji zahtijevaju obavezno zaustavljanje vozila. Kao što je poznato, lokacije na kojima je potrebno obavezno zaustaviti vozilo najčešće su i najopasnije lokacije (u smislu potencijalnih prometnih nesreća) i kao takve predstavljaju mjesta na kojima nema kompromisa kada je u pitanju točnost njihove detekcije i pravovremenost ispravne reakcije, a to je obavezno zaustavljanje vozila.

U drugom poglavlju ovog rada opisan je problem detekcije i prepoznavanja objekata u okolini koristeći kameru postavljenu na prednjoj strani vozila. Nakon toga opisana su neka postojeća rješenja i metode koje se koriste za detekciju znakova i prometne signalizacije na slikama dobivenim s kamere postavljene na prednjoj strani vozila. U trećem poglavlju opisani su alati i tehnologije korištene za razvoj vlastitog algoritma za detekciju prometnih znakova i signalizacije te sami postupak razvoja algoritma. Naime, osim same detekcije i prepoznavanja znakova i signalizacije od interesa, algoritam daje i procjenu udaljenosti do detektiranog znaka za svaku sliku, tj. za svaki okvir video signala. Nadalje, u četvrtom poglavlju analizirani su rezultati testiranja razvijenog algoritma te ja na koncu dan zaključak ovoga rada u petom poglavlju.

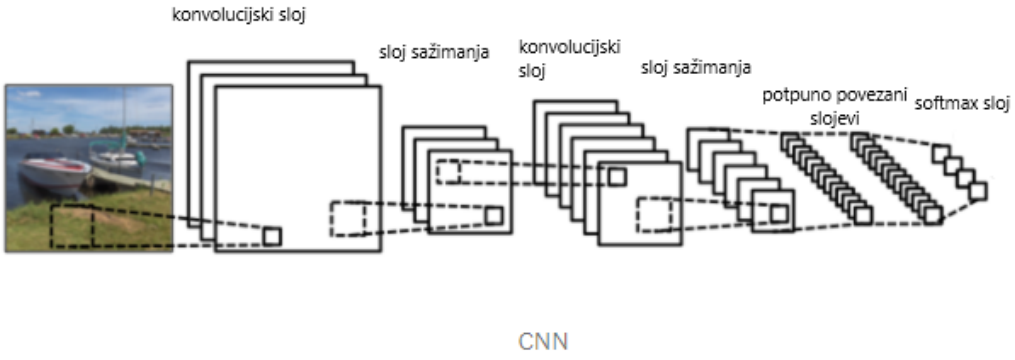
2. PROBLEM PREPOZNAVANJA PROMETNIH ZNAKOVA I PROMETNE SIGNALIZACIJE

Jedna od glavnih zadaća ADAS je prepoznavanje prometne signalizacije u prometnoj sceni. Ovaj rad bavi de detekcijom i prepoznavanjem prometnih znakova i prometne signalizacije u slici dobivenoj s kamere postavljene na prednjem kraju vozila. Ovisno o prepoznatom znaku, određuju se buduće radnje vozila. U skladu s tim dolazi se do problema detekcije objekata u sceni čiji je zadatak odrediti klasu pojedinog objekta od interesa u nekoj dolaznoj slici s kamere. Znakovi i prometna signalizacija na cesti daju informacije vozaču kako se treba ponašati na cesti. Upravljaju regulacijom na raskrižjima, ukazuju na to nalazi li se vozilo na sporednoj ili glavnoj cesti, nalažu kojom je maksimalnom brzinom dopušteno voziti, treba li se zaustaviti na određenom raskrižju, skrenuti lijevo ili desno, itd. Detekcija prometnih znakova i prometne signalizacije u slici iznimno je težak i zahtjevan zadatak. Mnogi su znakovi slični po boji, mogu se preklapati na slici ili biti presitni da ih računalo, odnosno algoritam prepozna, mogu biti pod određenim kutom snimljeni, mogu biti prljavi, itd. Postoje različiti pristupi u rješavanju problema detekcije i prepoznavanja prometnih znakova i signalizacije, a jedan od najčešćih u posljednje su vrijeme svakako konvolucijske neuronske mreže (engl. *Convolutional Neural Network* - CNN) [3]. Pošto se rješenje kreirano u ovom diplomskom radu zasniva upravo na korištenju CNN, u sljedećem podpoglavlju bit će ukratko opisana osnovna koncepcija rada CNN.

2.1. Konvolucijske neuronske mreže i njihova primjena za detekciju prometnih znakova

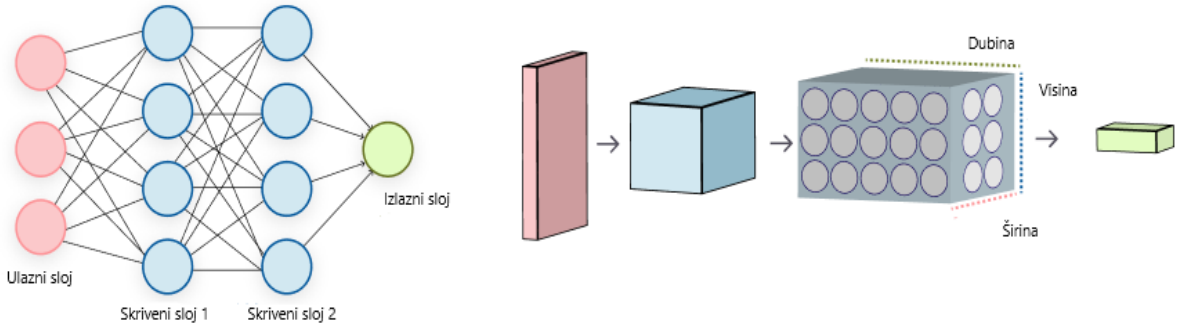
Strojno učenje kao grana umjetne inteligencije ima mnogo različitih pristupa u rješavanju određenih problema. Duboko učenje, koje je podskup strojnog učenja, koristi se za rješavanje mnogih problema u različitim područjima, kao što su obrada slike i računalni vid, obrada prirodnog jezika, robotika, računalne igre, itd. Što se tiče obrade slike, duboko učenje rješava probleme kao što su prepoznavanje objekata, detekcija objekata, segmentacija scene i dr. S druge strane, kod obrade prirodnog jezika, duboko učenje vrlo uspješno rješava probleme poput pretvorbe teksta u govor ili govora u tekst i sl. Kao što je navedeno, duboko učenje veliki značaj ima i u području autonomne vožnje u zadacima detekcije objekata, segmentacije slike, praćenja stanja vozača i sl. Jedna od najznačajnijih metoda primjene dubokog učenja je CNN. CNN je neuronska mreža dizajnirana za rad s dvodimenzionalnim podacima (kao što su slike), iako se

koristi i za rad s jednodimenzionalnim i trodimenzionalnim podacima [3]. Na slici 2.3 dan je izgled CNN.



Sl.2.3. *Primjer izgleda jedne konvolucijske neuronske mreže (CNN) [4].*

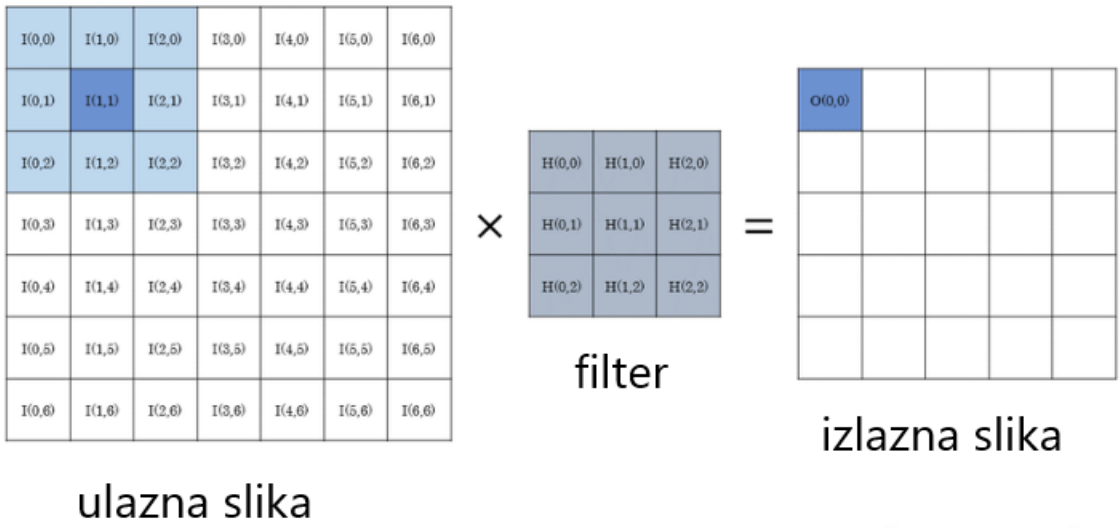
CNN sastoji se od niza slojeva od kojih je najznačajniji konvolucijski sloj, po kojem je ovaj model mreže i dobio naziv. Uvođenjem konvolucijskog sloja nastoji se očuvati dubina i struktura ulaznog, višedimenzionalnog podatka na način da svaka komponenta ulazne slike (npr. crvena, plava, zelena boja u RGB formatu) ostaje u zasebnom višedimenzionalnom obliku, što je velika prednost u odnosu na klasične neuronske mreže. Kod klasičnih neuronskih mreža sva tri vektora ulazne slike, koji označavaju pojedinu komponentu boje, spremaju se u jedan zajednički vektor. Na slici 2.4.(a) nalazi se primjer klasične neuronske mreže, a radi usporedbe s njom, izgled tipične CNN nalazi se na slici 2.4.(b). [3].



(a) (b)

Sl.2.4. *Izgled (a) klasične neuronske mreže, (b) CNN [5].*

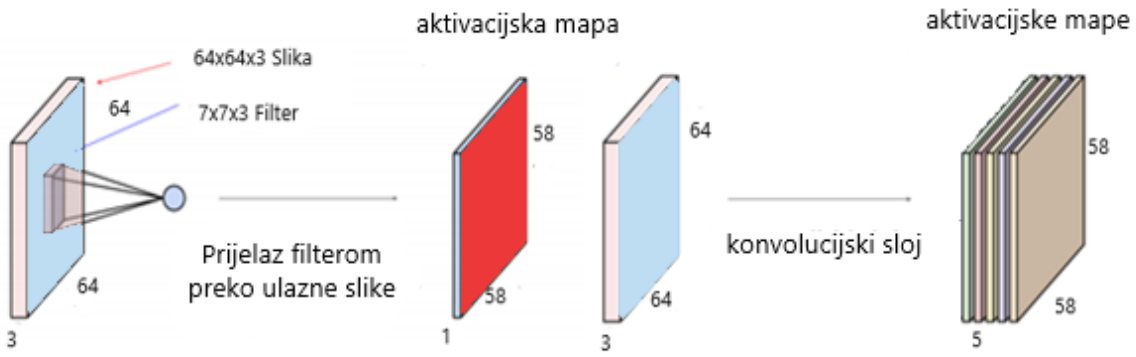
U konvolucijskom sloju provodi se operacija konvolucije nad ulaznim podacima (slikom). U kontekstu CNN-a, konvolucija je linearna operacija koja uključuje primjenu matrice određenih dimenzija koja se naziva filtar ili konvolucijski kernel [6]. Primjenjuje se filtar na određenim dijelovima ulazne slike i rezultat te primjene je zapravo izlaz iz tog sloja, tj. ulaz u sljedeći sloj. Kada je ulaz u konvolucijski sloj npr. slika veličine npr. 64x64x3, gdje 3 označava komponentu pojedine boje (npr. crvene, zelene, plave u RGB modelu boja) u slici, tada filtar mora također biti dubine 3, a najčešće je kvadratnih dimenzija, npr. 7x7x3. Na slici 2.5 nalazi se grafički prikaz primjene filtra na ulazne podatke (na lokaciji ulazne slike označene tamno plavom bojom).



Sl. 2.5. *Primjena filtera na ulazne podatke [6].*

Filter se primjenjuje na način da svaki njegov element množi onaj element ulazne matrice s kojim se preklapa. Zbrojem ovih umnožaka dobije se vrijednost elementa izlazne slike na istoj lokaciji koja se obrađuje u ulaznoj slici. Ova radnja se ponavlja na način da se filter kreće po slici s lijeva na desno, odozgor prema dolje, dok ne obradi sve lokacije slike. Nakon primjene jednog filtra na svim dijelovima ulazne slike dobiva se aktivacijska mapa, koja predstavlja izlaz konvolucijskog sloja za određeni filtar. Aktivacijska mapa sadrži odziv filtra na pojedinim dijelovima slike. Veličina aktivacijske mape ovisi o veličini ulazne slike, o tome dopunjava li se slika prije filtriranja (engl. *padding*), veličini filtra te koraku konvolucije (engl. *stride*) koji određuje za koliko mjesta će se pomicati filter po ulaznoj slici prilikom obrade slike [3]. Primjenom više različitih filtara u istom konvolucijskom sloju dobivaju se zasebne aktivacijske mape čijim se slaganjem u dubinu dobivaju ulazni podaci za sljedeći sloj CNN. Ovakvim

načinom rada konvolucijski slojevi mogu detektirati različite uzorke, odnosno značajke poput rubova, krugova, različitih vrsta uglova, itd. Slaganjem više konvolucijskih slojeva u mrežu na kraju se mogu detektirati jasni oblici, poput prometnog znaka, pješaka, automobila i sl. Na slici 2.6(a) dan je primjer kreiranja jedne aktivacijske mape primjenom jednog filtra, a na slici 2.6(b) primjer dobivanja pet aktivacijskih mapa nakon primjene 5 različitih filtera. Primjerice, od ulazne slike dimenzija 64x64x3, primjenom filtra 7x7x3 dobiva se aktivacijska mapa veličine 58x58x1 (uz *stride* 1). Primjenom pet različitih filtera te slaganjem aktivacijskih mapa u dubinu, na koncu se dobije izlazni podatak veličine 58x58x5 koji se kreće dalje kroz mrežu. U primjerima sa slike 2.6 nije korištena tehnika dopunjavanja (engl. *padding*) ulazne slike i stoga su aktivacijske mape manjih dimenzija od ulazne slike [3]. Dopunjavanjem se rub aktivacijske mape dopuni na određeni način (s nulama ili nekako drugačije) te se na taj način ne gubi na prostornoj dimenziji.



a) b)

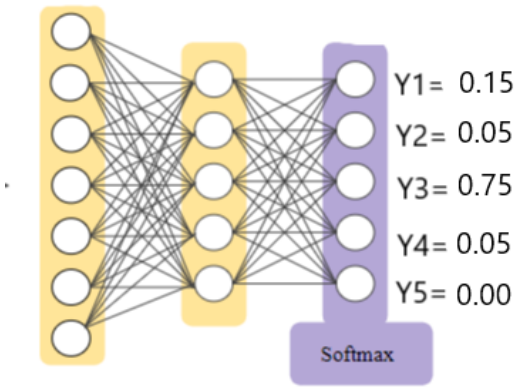
Sl.2.6. *(a) Kreiranje jedne aktivacijske mape primjenom jednog filtra, (b) slaganje pet aktivacijskih mapa primjenom pet različitih filtera [3].*

Sljedeći sloj koji se pojavljuje u CNN naziva se sloj sažimanja (engl. *Pooling layer*). Sloj sažimanja može se primijeniti na svaku aktivacijsku mapu zasebno, a njegov je cilj smanjiti rezoluciju, odnosno dimenziju slike (aktivacijske mape) koja predstavlja ulaz u sloj nakon sloja sažimanja [7]. Sloj sažimanja uzima predefiniranu veličinu podmatrice i koristi jednu od operacija kako bi sažeo podatke. Najčešći način sažimanja je po maksimalnoj vrijednosti (engl. *Max pooling*). Tada se uzima vrijednost najvećeg elementa podmatrice te se njome popunjava nova lokacija u novoj matrici (sažetoj slici) kako bi se saželi podaci. Osim po maksimalnoj

vrijednosti, postoji način sažimanja gdje se uzima srednja vrijednosti elemenata podmatrice, a takav način naziva se sažimanje po srednjoj vrijednosti (engl. *Average pooling*). Na slici 2.7 prikazan je koncept sažimanja po maksimalnoj i srednjoj vrijednosti elemenata podmatrice.

Sl.2.7. *Sažimanje po maksimalnoj i srednjoj vrijednosti [3].*

Najčešće posljednji sloj u CNN je *softmax* sloj. *Softmax* je potpuno povezani sloj koji se koristi kao izlazni sloj kod mreže koja vrši višeklasnu klasifikaciju. Broj neurona *softmax* sloja jednak je broju različitih klasa u danom problemu višeklasne klasifikacije. Na izlazu ovog sloja dobiva se vjerojatnost pripadnosti objekta na slici pojedinoj klasi. Ukupan zbroj vjerojatnosti (izlaza svih neurona ovog sloja) jednak je jedan. Na slici 2.8 dan je primjer *softmax* sloja s 5 izlaznih neurona (pet klasa, Y1-Y5) i pripadajućim vjerojatnostima.



Sl.2.8. *Softmax sloj s 5 izlaznih neurona [8].*

2.2. Mjere uspješnosti rada rješenja za detekciju objekata u slici

Kako bi bilo koje rješenje za detekciju i prepoznavanje objekata bilo ispravno evaluirano, koriste se određene mjere koje na smislen način odražavaju performanse rada određenog rješenja na nekom skupu podataka. U ovom dijelu rada opisane su najvažnije mjere korištene pri ocjeni uspješnosti rada rješenja za detekciju i prepoznavanje objekata [9]. Mjere su sljedeće:

- Preciznost (engl. *Precision*)
- Odziv (engl. *Recall*)
- F-1 mjera (engl. *F-1 Score*)
- IoU (engl. *Intersect over union*)
- Prosječna preciznost (engl. *Average Precision*, AP)
- Srednja prosječna preciznost (engl. *mean Average precision*, mAP)

Osim navedenih mjera potrebno je razjasniti pojmove koji se odnose na ispravnost detekcije objekata. Stoga objekt može biti:

- Točno detektiran objekt (engl. *True positive*, TP) – objekt koji je trebao biti detektiran i ispravno je detektiran
- Netočno detektiran objekt (engl. *False positive*, FP) – objekt koji nije trebao biti detektiran, a detektiran je
- Nedetektirani objekt (engl. *False negative*, FN) – objekt koji je trebao biti detektiran, a nije detektiran

Pomoću navedenih parametara (TP, FP, NP) računa se preciznost detekcije (PR) na sljedeći način:

$$PR = \frac{TP}{TP+FP} \tag{2-1}$$

Odziv (OD) označava koliko dobro model pronalazi sve točne detekcije, a računa se kao omjer broja svih točnih detekcija i stvarnog (ispravnog) broja detekcija:

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

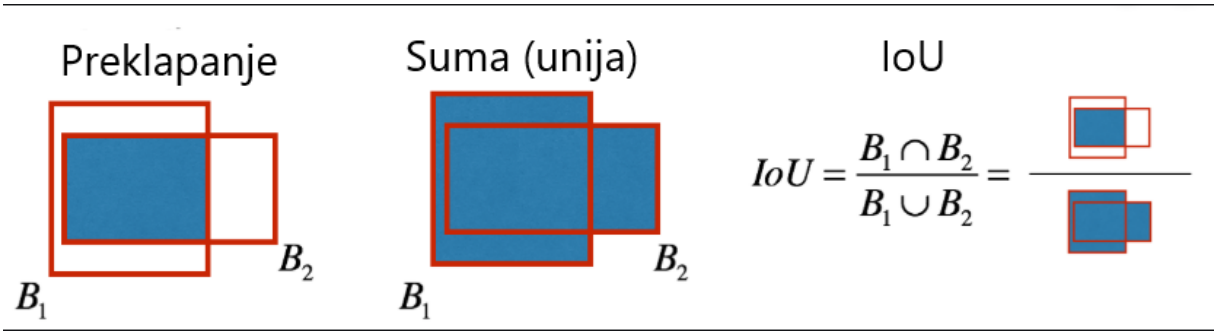
F-1 mjera (F1) označava mjeru točnosti modela. Koristi se kada mjere preciznosti i odziva ne daju dovoljno jasne rezultate. F-1 ovisna je preciznosti i odzivu i računa se kao:

$$F1 = 2 * \frac{PR*OD}{PR+OD} \tag{2-3}$$

Parametar IoU predstavlja omjer površine preklapanja detektiranog i stvarnog graničnog okvira objekta (engl. *area of overlap*) i unije površina detektiranog i stvarnog graničnog okvira objekta (engl. *area of union*), a računa se prema formuli:

$$IoU = \frac{površina\ preklapanja\ detektiranog\ i\ stvarnog\ graničnog\ okvira}{unija\ površina\ detektiranog\ i\ stvarnog\ graničnog\ okvira} \tag{2-4}$$

Na slici 2.1. dan je grafički prikaz objašnjenja parametra IoU. B_1 predstavlja stvarni granični okvir objekta (engl. *ground truth*), a B_2 predstavlja granični okvir objekta kojeg detektira određeni algoritam.

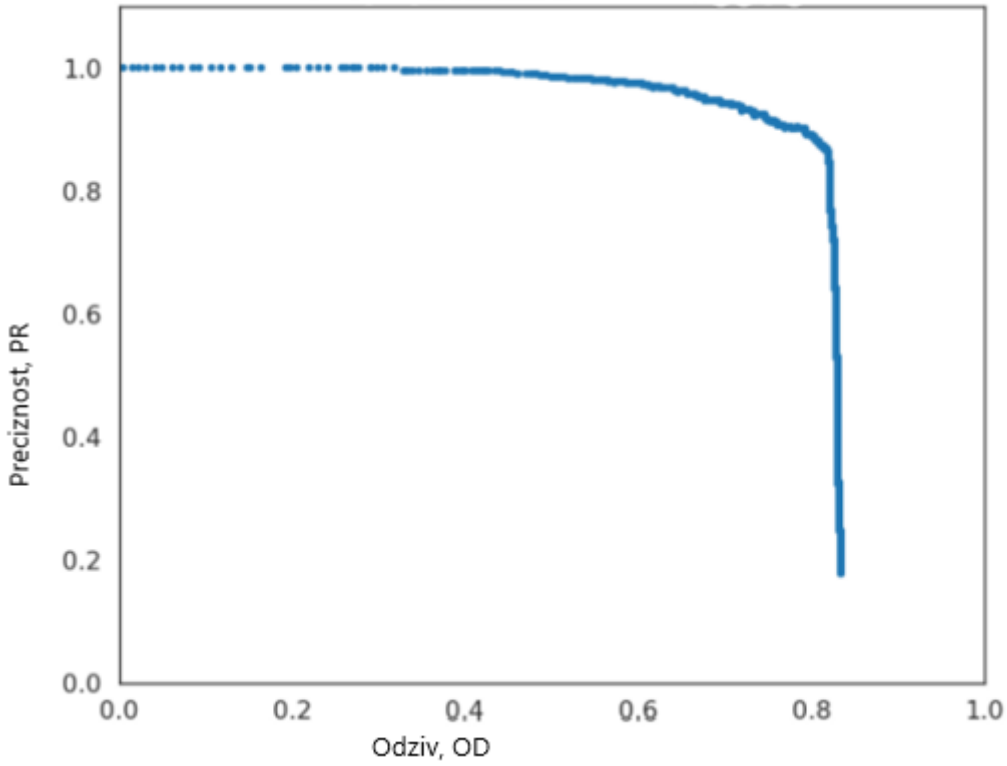


Sl.2.1. *Grafički prikaz Intersect over union (IoU)[9].*

Kako bi se izračunala mjera AP (engl. *average precision*) potrebne su mjere preciznosti i odziva. Računa se kao površina ispod krivulje preciznosti ovisne o odzivu:

$$AP = \int_0^1 p(r)\, dr \tag{2-5}$$

Na slici 2.2 dan je primjer krivulje preciznosti ovisno o odzivu.



Sl.2.2. *Krivulja preciznosti ovisne o odzivu [9]*.

U izrazu (2-5) *p(r)* označava krivulju preciznosti ovisnu o odzivu. Pomoću mjere AP računa se mjera mAP (engl. *mean average precision*) koja čini zbroj prosječnih preciznosti za svaku klasu objekta (a ima ih *N*) koju je potrebno razlikovati, a računa se prema formuli:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2-6}$$

2.2.1 Pregled radova koji se bave primjenom CNN u detekciji prometnih znakova

Postoje mnogi radovi iz područja detekcije prometnih znakova za autonomnu vožnju koji u svom rado koriste CNN. U jednom od takvih radova [10], autori su razvili metodu za detekciju prometnih znakova koja se zasniva na konvolucijskoj neuronskoj mreži. Ulazna slika najprije je transformirana u sliku u skali sive boje (engl. *grayscale image*), a nakon toga se primjenjuje metoda potpornih vektora (engl. *support vector machine, SVM*). SVM je metoda koja se koristi kod problema binarne klasifikacije ili problema regresije. Ideja je naći optimalnu krivulju u određenom N-dimenzionalnom prostoru koja će jasno klasificirati podatke ulazne slike. Nakon toga korištena je CNN za detekciju i prepoznavanje. Mreža je učena na način da se nastoji smanjiti količina interesnih područja te označava granice vrlo blizu granica samih znakova

prometnih znakova. Slojevi koji se mogu učiti mogu značajno povećati točnost detekcije. Osim toga, korištena je *bootstrap* metoda uzorkovanja za procjenu pouzdanosti kojom se poboljšava točnost i izbjegava problem pretjeranog treniranja, tj. prevelikog usklađivanja modela na trening podatke uz gubitak sposobnosti generalizacije (engl. *overfitting*). Testiranja su vršena na skupu slika njemačkih prometnih znakova, a dobiveni su rezultati s preciznošću od 97,62% pri zadatku prepoznavanju znakova.

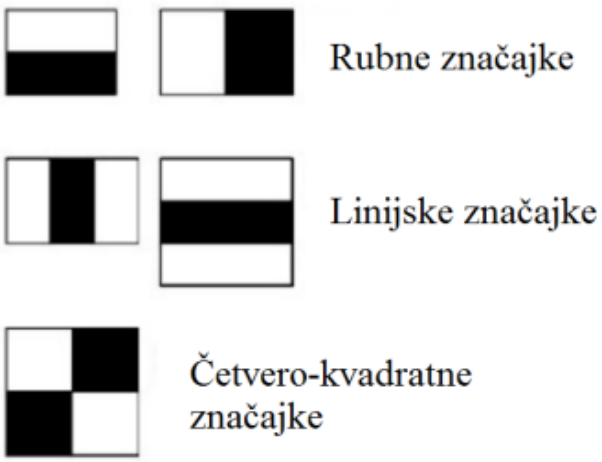
U radu [11] razmatra se implementacija algoritma za klasifikaciju u svrhu prepoznavanja prometnih znakova. U kombinaciji s koracima za prethodnu obradu i lokalizaciju koji koriste Houghovu transformaciju, predložena metoda CNN za klasifikaciju i prepoznavanje prometnih znakova pokazuje vrlo dobre rezultate s mjerom odziva od 0.9994. Predloženo klasifikacijsko rješenje provodi se pomoću programskog okvira TensorFlow, a koristi CUDA tehnologiju za realizaciju rješenja na GPU (engl. *graphic proccesing unit*). Korištenje takvih algoritama za prepoznavanje znakova omogućava obradu video okvira u stvarnom vremenu s velikom razlučivosti pa stoga radi i na većim udaljenostima i s boljom kvalitetom. S obzirom na tu činjenicu moguće je prepoznati prometni znak na udaljenosti i do 50 m. Razvijena metoda implementirana je na uređaj s Nvidia Tegra K1 procesorom. CUDA je navikla ubrzati izvedbu opisanih metoda. U budućim istraživanjima planira se osposobiti CNN da razmatra više klasa prometnih znakova i moguće loše vremenske uvjete. Također, planiraju koristiti CNN ne samo za klasifikaciju već i za detekciju objekata.

Današnje mreže koje se koriste za detekciju i prepoznavanje temelje se na algoritmima koji obrađuju sliku po regijama. Autori rada [12] uvode mrežu „regionalnih prijedloga“ (RPN) čija je svrha predložiti više objekata koje je moguće prepoznati unutar određene slike. RPN dijeli značajke cjelovite slike s detekcijskom mrežom, omogućujući tako prijedloge za regiju. RPN je potpuno konvolucijska mreža koja istodobno predviđa granice objekta i rezultate objekta na svakoj poziciji. RPN je osposobljen za generiranje visokokvalitetnih prijedloga regija, koje primjerice Fast R-CNN koristi za otkrivanje. Nadalje autori rada spajaju RPN i brzi R-CNN u jedinstvenu mrežu, gdje RPN komponenta govori cjelokupnoj mreži gdje treba pogledati. Za vrlo duboki model VGG-16, njihov sustav za otkrivanje ima brzinu slike od 5 FPS (uključujući sve korake) na GPU-u, dok istodobno postiže vrhunsku preciznost otkrivanja objekata na skupovima podataka PASCAL VOC 2007, 2012 i MS COCO sa samo 300 prijedloga po slici. Na natjecanjima ILSVRC i COCO 2015. godine, ubrzani R-CNN i RPN temelj su osvajanja 1. mjesta u nekoliko područja detekcije.

Velika većina ovih algoritama radi s visokim postotkom preciznosti u idealnim uvjetima gdje su korištene jasne slike bez šuma. Autori rada [13] bavili su se detekcijom prometnih znakova u različitim vremenskim uvjetima s mnogim tipovima šuma i smetnji kao što su kiša, noć, snijeg i dr. Za detekciju je korištena CNN arhitekture VGG-16, dok se za problem lokalizacije objekata koristila CNN arhitekture U-net. Testiranja pokazuju da algoritam postiže solidnu razinu preciznosti detekcije i klasifikacije čak i u prisutnosti različitih vremenskih uvjeta, s ukupnom preciznošću 62% i odzivom 42%. Međutim, iz tih brojki može se vidjeti da je prepoznao samo 42 % svih znakova koje je trebao prepoznati i da mu pritom razina preciznosti nije ni blizu željene brojke. Upravo takve i slične su brojke razlog zašto današnja vozila još uvijek nisu potpuno autonomna i zašto je još uvijek vozač neophodan u vožnji.

2.3. Viola-Jones algoritam i njegova primjena u detekciji prometnih znakova

Osim CNN postoje i druge metode zasnovane na strojnom učenja koje se koriste za detekciju objekata u slici, pa tako i prometnih znakova i signalizacije. Jedna od njih je Viola-Jones algoritam[14]. Viola-Jones je metoda za brzo pronalaženje objekata u slici korištenjem kaskade Haarovih klasifikatora, odnosno Haarovih značajki. Haarove značajke slične su konvolucijskim kernelima koji su spomenuti ranije, a koriste se za detekciju određene značajke u danoj slici. Primjenjuju se na način da se određenom značajkom prelazi preko elemenata ulazne slike i u njoj se traže određene odgovarajuće karakteristike željenog objekta. Na slici 2.9 dani su tipovi osnovnih značajki u Viola-Jones algoritmu [15].



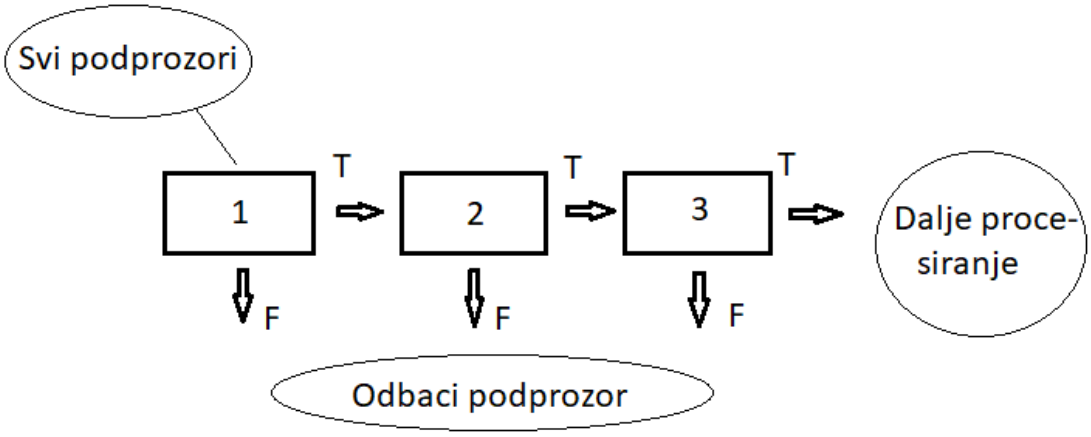
Sl.2.9. *Osnovne Haarove značajke[14]*

Sve ostale značajke dobivaju se translacijom, rotacijom ili skaliranjem osnovnih značajki. Primjena svake značajka rezultira jedinstvenom vrijednošću koja je izračunata oduzimanjem sume intenziteta elemenata slike ispod bijelog pravokutnika od sume intenziteta elemenata slike ispod crnog pravokutnika značajke. Na slici 2.10 prikazan je način primjene Haarovih značajki. Primjerice, linijska značajka namijenjena je za detekciju nosa (tj. za detekciju područja u slici gdje se potencijalno nalazi nos) jer je nos svjetliji od dva ruba nosa, s lijeve i desne strane, što odgovara upravo ovoj značajki. S druge strane rubna će značajka najbolje izdvojiti područje očiju, tj. tamno područje samih očiju te svijetlo područje ispod očiju. Stoga, primjenom značajke za oči na cijeloj slici, trebale bi se dobiti veće vrijednosti na lokacijama onih elemenata slike koji potencijalno pripadaju očima [15].

Sl.2.10. *Primjena Haarove značajke na ulaznu sliku (rubna značajka se koristi npr. za detekciju očiju, a linijska za detekciju nosa) [15].*

Svaka značajka smatra se kao jedan binarni klasifikator, koja za određenu lokaciju slike kaže pripada li potencijalno objektu koji se traži ili ne pripada. Kada se proračunaju sve tražene značajke (svi tipovi) na svim skalama i lokacijama slike, dobiva se jako velik broj rezultata primjene tih značajki, što obradu slike i detekciju objekta čini praktički neizvedivom u stvarnom vremenu. Stoga se uvodi AdaBoost algoritam kojim se eliminiraju sve irelevantne značajke za danu sliku. Osim AdaBoosta uvodi se i tehnika nazvana integralna slika, koja ubrzava sam postupak proračuna značajki. Ideja AdaBoosta je da stvori jedan jaki klasifikator pomoću više slabih klasifikatora. Jaki klasifikator dobiva se linearnom kombinacijom slabih. Na koncu se radi kaskada gdje je ideja koristiti više jakih klasifikatora koji će imati visoki postotak eliminacije lokacija u kojima se ne nalaze traženi objekti. Ulančavanjem jakih klasifikatora u kaskadu znatno se ubrzava postupak detekcije. Na slici 2.11 nalazi se primjer kaskade

klasifikatora. Prvi klasifikator eliminira velik broj negativnih uzoraka uz vrlo malo procesiranja. Sljedeće razine eliminiraju dodatne negativne uzorke, ali uz sve više procesiranja.



Sl.2.11. *Primjer vezanja kasifikatora u kaskadu.*

Autori rada [16] koristili su Viola-Jones algoritam za razvoj detektora prometnih znakova. Rezultati pokazuju da je detektor primjenjiv za obradu videa u realnom vremenu. Rješenje radi s mjerom preciznosti od 0.72 i obrađuje u prosjeku 21 FPS (engl. *frames per second*). Na snažnijem računalu s po 4 jezgre od 2.2 Ghz preciznost se podiže do 0.82. Nadalje, zaključuju da detektor ima bolje rezultate kada se koristi na video sekvenci, u usporedbi sa standardnim pristupom detekcije prometnih znakova na statičkim slikama. Kao problem navode mali broj uzoraka rijetkih prometnih znakova, a tu je i problem sličnosti između nekih klasa prometnih znakova. Algoritam je treniran samo na znakovima okruglog oblika pa buduća poboljšanja uključuju prepoznavanje znakova različitih oblika te korištenje naprednijih metoda izdvajanja značajki kao što su, primjerice, analiza glavnih komponenata (engl. *Principal Component Analysis* - PCA).

U radu [17] autori se bave detekcijom prometnih znakova najpopularnijim metodama detekcije, a jedna od njih je i Viola-Jones algoritam. Njihov detektor je u osnovi kaskada binarnih linearnih klasifikatora koji se naknadno primjenjuju na ulaz kliznog prozora. Primjer je prošao kroz kaskadu sve dok je pozitivno klasificiran od strane trenutne faze. Svaka se faza sastoji od određenog broja ponderiranih jednodimenzionalnih klasifikatora s pragom koji se odnose na jedno obilježje. Trening set za fazu *n* sastoji se od svih TP i FP rezultata koji su ostali nakon faze *n-1*, pri čemu se oni za prvu fazu biraju nasumično iz cjelovitih slika. Većina kliznih prozora procjenjuje se samo u prvim fazama koje sadrže malo klasifikatora / značajki. Značajke koje se nude tijekom treninga su jednostavni filteri nalik Haar-u koji se mogu lagano procijeniti

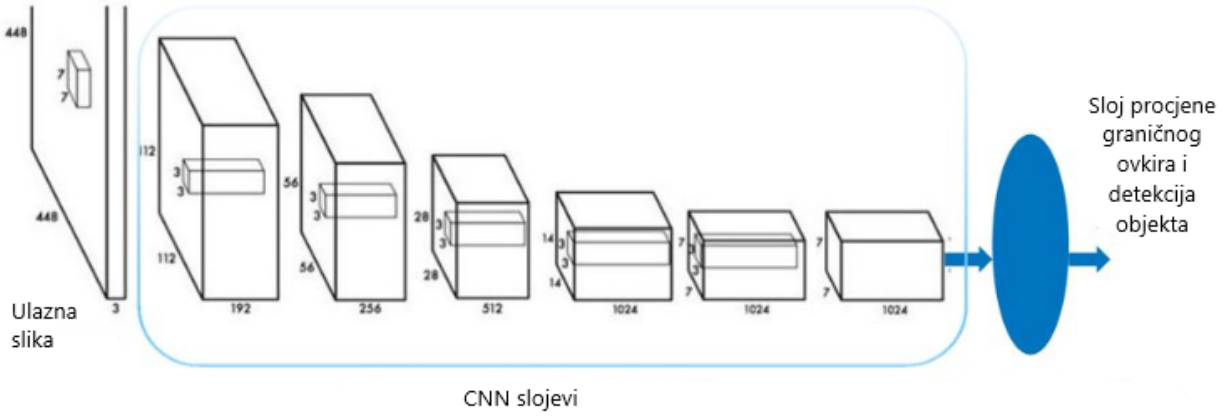
pomoću unaprijed izračunate integralne slike. Njihov skup Haarovih značajki sadrži 5 osnovnih vrsta u 3 razine i sve moguće položaje unutar prozora, što rezultira skupom od 5445 značajki. Na koncu je metoda Viola-Jones detektora bila najuspješnija pri detekciji znakova s mjerom odziva od 0.98 za znakove nailaska na cestu s prednošću prolaska, a 0.74 za znakove obaveznog zaustavljanja.

2.4. YOLO algoritam i njegova primjena u detekciji prometnih znakova

YOLO (engl. *You Only Look Once*) algoritam [18] također je danas široko primjenjiv algoritam i vrlo često korišten. Izuzetno je brz u obavljanju zadataka detekcije objekata u slikama. Poseban je po tome što se zasniva na regresiji. Prethodni sustavi za detekciju primjenjuju klasifikatore ili lokalizatore radi obavljanja detekcije. YOLO radi na način da primjenjuje model CNN-a na ulaznu sliku na više mjesta i kroz više mjerila. Ulazna slika dijeli se na mrežu i ukoliko se traženi objekt nalazi unutar određenog dijela mreže (regije), onda taj određeni dio mreže obavlja detekciju objekta. Područja visoke koncentracije točaka na slici smatraju se detekcijama. YOLO predviđa granične okvire (engl. *Bounding Box*) za svaku regiju posebno i daje vrijednost koja predstavlja vjerojatnost da je unutar određenog okvira traženi objekt. Ukoliko određena regija ne sadrži objekt, vjerojatnost je jednaka nuli. Granični okvir određen je s pet parametra: vjerojatnost predikcije objekta te četiri predikcije *x, y, w,* i *h*. *x* i *y* daju lokaciju središta predviđenog graničnog okvira, a *w* i *h* predstavljaju širinu i visinu graničnog okvira. Na slici 2.12 prikazan je način na koji je mreža primijenjena na sliku. Svaka ćelija ove mreže predstavlja gore spomenute regije i ako objekt padne u određenu regiju, regija je zadužena za detekciju[18].

Sl.2.12. *Primjena mreže na ulaznu sliku.*

Arhitektura CNN koju primjenjuje osnovni YOLO algoritam inspirirana je GoogLeNet modelom za klasifikaciju slika [19]. YOLO CNN ima 24 konvolucijska sloja te dva potpuno povezana sloja. Umjesto početnih modula koje koristi GoogLeNet, YOLO jednostavno upotrebljava 1x1 redukcijske slojeve, nakon kojih slijede 3x3 konvolucijski slojevi. Arhitektura CNN osnovnog YOLO algoritma prikazana je na slici 2.13.



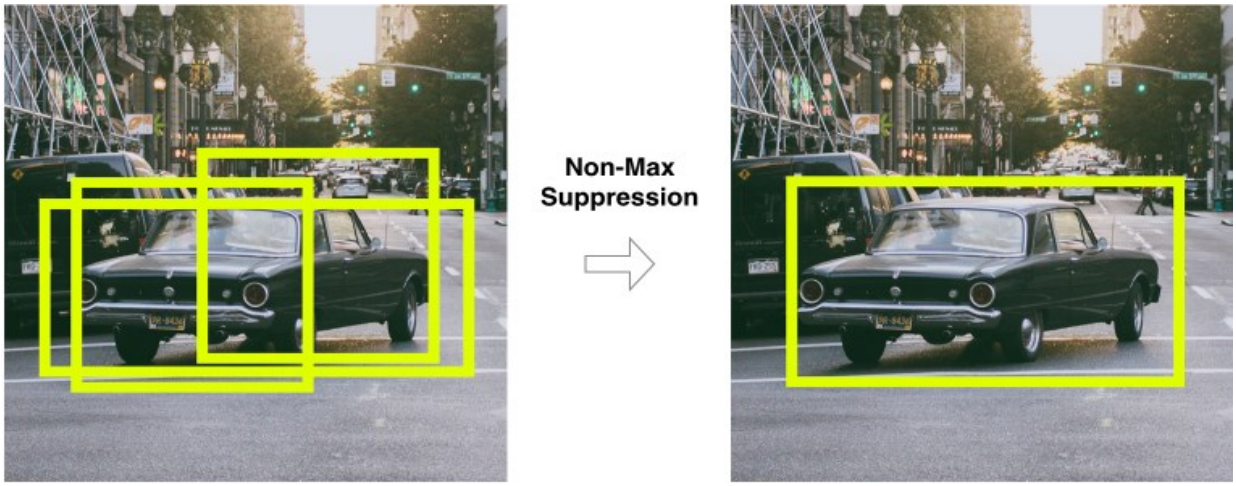
Sl.2.13. *Arhitektura YOLO modela*.

YOLO proces učenja provodi na cijelim slikama na kojima su objekti ručno označeni te koje su spremljene u skup za treniranje. Prije prolaska kroz mrežu, sve slike iz skupa skalirane su na 448 x 448 elemenata slike. Osim originalnog YOLO, dizajniran je i ubrzani YOLO koji sadrži 9 konvolucijskih slojeva s manje filtera po sloju. Ovi modeli prvi put su trenirani i testirani na Pscal VOC 2007 bazi slika te je ustanovljeno da su puno brži od ranije razvijenih CNN kao što su R-CNN (engl. *Regions with CNN features*) i Fast R-CNN (ubrzani R-CNN) [20].

Prvotni YOLO ima nekoliko nedostataka. YOLO nameće snažna prostorna ograničenja pri predviđanju graničnih okvira jer svaka ćelija mreže predviđa samo dva okvira, a može imati samo jednu klasu. Ovo prostorno ograničenje otežava detekciju drugih objekata u blizini koje YOLO može predvidjeti. Model se tako bori s malim objektima koji se pojavljuju u grupama, poput jata ptica. Budući da YOLO uči predvidjeti granične okvire iz podataka, bori se s generalizacijom u novim ili drugačijim uvjetima. Model također koristi relativno grube značajke za predviđanje graničnih okvira jer YOLO arhitektura ima više slojeva poduzorkovanja (engl. *Downsampling*). Osim toga, problem se javlja i jer kriterijska funkcija na isti način tretira pogreške u predikciji malih i velikih graničnih okvira. Greška određene

veličine kod predikcije velikih graničnih okvira ne utječe jednako na IoU kao kod predikcije malih graničnih okvira [18].

Još jedan od problema je kada se veliki objekt na slici nalazi blizu ruba slike, a može biti detektiran kroz više ćelija. Ovo se može riješiti dodavanjem funkcionalnosti *non-maximal supression* koja se može dodati u osnovni model YOLO radi bolje detekcije. *Non-maximal supression* pronalazi detekciju s najvećom vjerojatnošću i proglasi ju objektom, a potom računa njezin IoU s ostalim detekcijama i odbacuje sve detekcije koje imaju preklapanje veće od određenog praga. Postupak se ponavlja sve dok ima detekcija koje se preklapaju. Većina ćelija, odnosno graničnih okvira, neće sadržavati objekt i to je razlog zašto se predviđa vjerojatnost nalazi li se objekt u određenom graničnom okviru [20]. Na slici 2.14 dan je primjer gdje se uklanjaju granični okviri s niskom vjerojatnošću objekta i dobiva se najbolji granični okvir za dani objekt.



Sl.2.14. *Primjena metode non-maximal suppression.*

YOLO algoritam je sve više prisutan u području računalnog vida i automobilske industrije te se mnogi oslanjaju upravo na ovakav način detekcije objekata. U [21] opisana je implementacija YOLO CNN-a na mobilnu GPU platformu NVIDIA Jetson. Istrenirani parametri mreže pokazuju dobre rezultate i potvrđuju da je YOLO dobro rješenje za obradu slika u realnom vremenu na mobilnoj platformi. Testovi su izvršavani na video sekvencama, a pokazuju da je rješenje, implementirano na mobilnom GPU Jetsonu TK1, prilično sporo, ali pokazuje dobru kvalitetu prepoznavanja objekata. Kako bi se zadovoljio rad u stvarnom vremenu, potrebni su snažniji uređaji poput Jetson TX2 ili Jetson Xavier. Ove GPU mobilne platforme u kombinaciji s YOLO dobar su izbor za razvoj kvalitetnog ADAS sustava. Kao baza

slika za treniranje korištena je baza njemačkih znakova, a testiran je broj detektiranih objekata po sekundi. Rezultati na Jetson TK1 su 1.5-2.2 FPS, dok je na snažnijim inačicama poput Xaviera rezultat 18-22 FPS. Također se navodi kako na snažnijim, stolnim računalima, detekcija može biti još uspješnija.

Autori rada [22] predlažu drugu inačicu YOLO algoritma YOLOv2 kao idealno rješenje za detekciju i prepoznavanje prometnih znakova. Algoritam je treniran na kineskim i njemačkim prometnim znakovima na računalu s NVIDIA GTX980Ti GPU i procesorom Intel Core i7-6700K (CPU 4.0 GHz). Najbolja rješenja pokazuju detekciju na slikama s odzivom od 0.82 i preciznosti od 0.97 s vremenom detekcije oko 0.017 sekundi za jedan okvir. U daljnjem radu najavljuju razvijanje rješenja na slikama puno veće rezolucije što samim time zahtjeva i puno veću računalnu moć.

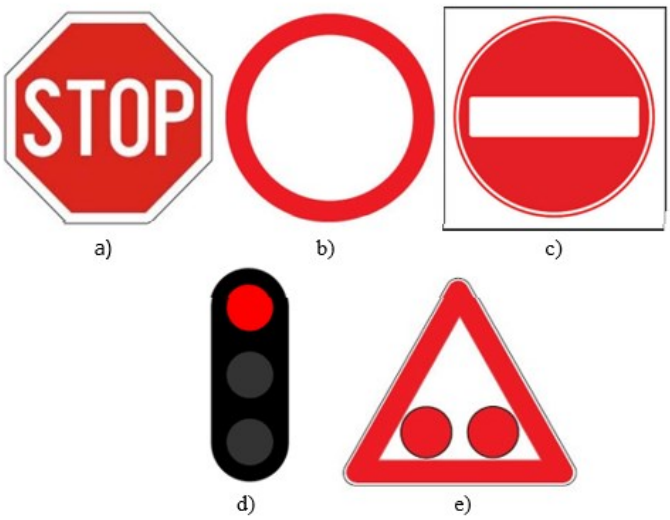
3. VLASTITO RJEŠENJE ZA DETEKCIJU PROMETNIH ZNAKOVA I PROMETNE SIGNALIZACIJE U SLICI

U ovom poglavlju detaljno je opisano rješenje za detekciju i prepoznavanje prometnih znakova i prometne signalizacije koja nalaže izričito zaustavljanje vozila. Osim algoritma za detekciju i prepoznavanje razvijen je i algoritam za procjenu udaljenosti vozila od detektiranog znaka ili semafora, koji je također opisan u ovom poglavlju. Također, u sklopu ovog poglavlja opisani su i alati korišteni pri razvoju algoritama te tehnologija u kojima su razvijani.

Budući da je zadatak ovog rada bio je razviti rješenje koji će detektirati i prepoznati prometne znakove i prometnu signalizaciju koja izričito nalaže zaustavljanje vozila, izabrani su sljedeći znakovi i signalizacija koji su, shodno tome, raspoređeni u pet klasa:

- Obavezno zaustavljanje (Slika 3.1 (a))
- Zabrana prometa u oba smjera(Slika 3.1 (b))
- Zabrana prometa u jednom smjeru (Slika 3.1 (c))
- Semafor s uključenim crvenim svjetlom (Slika 3.1 (d))
- Semafor s uključenim crvenim svjetlom pri nailasku na pružni prijelaz (Slika 3.1 (e))

Na slici 3.1 nalaze se navedeni znakovi.



Sl. 3.1. *Prometni znakovi i prometna signalizacija koji trebaju biti detektirani u sklopu zadatka (a) obavezno zaustavljanje, (b) zabrana prometa u oba smjera. (c) zabrana prometa u jednom smjeru, (d) semafor s uključenim crvenim svjetlom, (e) semafor s uključenim crvenim svjetlom pri nailasku na pružni prijelaz*

3.1. Opis alata korištenih za razvoj rješenja

Za razvoj detektora prometnih znakova i prometne signalizacije korišten je YOLOv3 algoritam koji je razvijan u programskom okviru Darknet koji je otvorenog koda (engl. *open source*)[23].

3.1.1. YOLOv3 algoritam

Za detekciju i prepoznavanje prometnih znakova i prometne signalizacije koristi se YOLO algoritam, čiji je princip rada opisan u dijelu 2.4 [16]. Razlog za to je što je YOLO izuzetno brz algoritam za detekciju objekata u stvarnom vremenu i trenutno je jedan od najrobusnijih i najučinkovitijih algoritama u području detekcije objekata, a sve češće se primjenjuje i u automobilskoj industriji. YOLO ima više inačica osnovnog modela u kojima su se nastojale poboljšati performanse, a neke od njih su YOLO9000 i YOLOv3. U ovom diplomskom radu je korišten YOLOv3, koji je treća po redu inačica ovog algoritma i koji primjenjuje jednaku CNN kao osnovni YOLO, ali uz određena poboljšanja [18].

Ulazna slika podijeljena je na mrežu od *N* x *N* ćelija (engl. *Grid cell*) od kojih je svaka zadužena za detekciju objekta koji se nalazi u njoj. Kao i inačica YOLO9000, YOLOv3 koristi metodu predviđanja graničnih okvira (engl. *bounding box*). Ti okviri su različitih veličina, uz koje se također račun vjerojatnost detekcije koja pokazuje koliko je algoritam siguran da se u ćeliji mreže nalazi određeni objekt te koliko je preklapanje predviđenog graničnog okvira i stvarnog. Mreža daje četiri vrijednosti za svaki granični okvir (t_x,t_y, t_w, t_h). Ako je gornji lijevi ugao ćelije pomaknut od gornjeg lijevog ugla slike za c_x i c_y (kao na slici 3.2), tada granični okvir ima širinu i visina (p_w, p_h, a predviđanja odgovaraju izrazima:

$$b_x = \sigma(t_x) + c_x \tag{3-1}$$

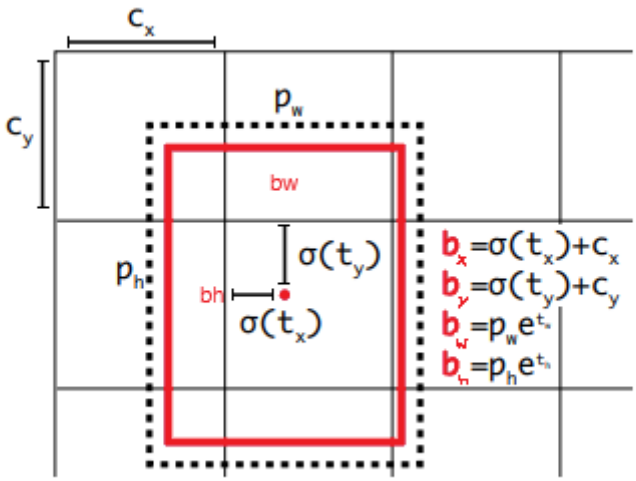
$$b_y = \sigma(t_y) + c_y \tag{3-2}$$

$$b_w = p_w e^{t_w} \tag{3-3}$$

$$b_h = p_h e^{t_h} \tag{3-4}$$

Ovdje b_x i b_y predstavljaju koordinate središta predviđenog graničnog okvira, a b_w i b_h širinu i visinu predviđenog graničnog okvira. Rezultat za svaki granični pravokutnik predviđa se na temelju logičke regresije. Rezultat predviđanja iznosi 1 ako se predviđeni granični okvira

poklapa sa ručno označenim objektom više od bilo kojeg drugog predviđenog graničnog okvira. Ako se pak granični okvir ne preklapa baš najbolje ili se preklapa manje od neke postavljene granice, zanemaruje se predikcija. Ako se predviđeni granični okvir uopće ne preklapa s ručno označenim graničnim okvirom, ne gube se predviđene koordinate ili klase već samo vjerojatnost predikcije.



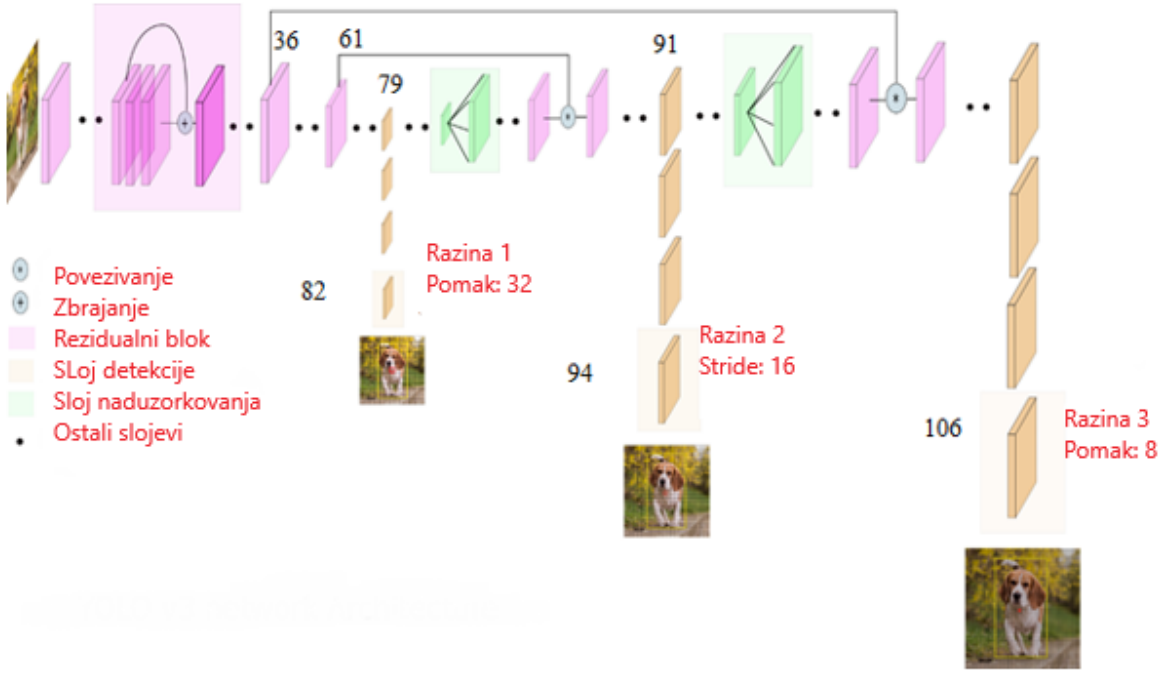
Sl.3.2. *Izgled mreže i procjena točaka koje opisuju predviđeni granični okvir [24].*

YOLOv3 granične okvire predviđa na tri različite razine, što mu omogućava da detektira male, srednje ili velike objekte. Arhitektura YOLOv3 mreže sastoji se od rezidualnih blokova (engl. *Residual Block*) koji sadrže konvolucijske slojeve i puteve za prečac (engl. s*hortcut path*). Kod klasične CNN značajke se uče jedna po jedna, a kako se proces odvija sve dublje kroz slojeve mreže, teže je učiti nove značajke. Slojevi koji se nalaze unutar prečaca uče koji parametar treba nadodati na staru značajku kako bi se stvorile nove. Rezultat je složena značajka *H(x)*, koja je predstavljena pomoću izraza:

$$H(x) = F(x) + x \tag{3-5}$$

gdje je *F(x)* stara i jednostavnija značajka na koju se dodaje posebni naučeni parametar *x*. Na ovaj se način mreži omogućuje lakše učenje značajki, pogotovo u dubljim slojevima mreže. Kako bi se omogućilo višeznačno prepoznavanje, *softmax* sloj zamijenjen je s 1x1 konvolucijskim slojem koji sadrži logističku funkciju. Kod *softmax* sloja, jedan izlaz, tj. jedan neuron mreže pripada samo jednoj klasi, a postoje slučajevi kada jedan objekt može pripasti

većem broju klasa na izlazu (npr. klasa osoba i klasa muškarac) [20]. Arhitektura YOLOv3 mreže dana je na slici 3.3.



Sl.3.3. *Arhitektura YOLOv3 algoritma[24]*

3.1.2. Darknet i CUDA

Darknet je programski okvir otvorenog koda koji sadrži mnoštvo funkcija koje opisuju rad YOLO algoritma, a pisan je u C programskoj jeziku [25]. Jednostavan i brz za instalaciju, podržava izvedbu na središnjoj procesorskoj jedinici (engl. *Central Processing Unit* – CPU) i grafičkoj procesorskoj jedinici (engl. *Graphics processing Unit* – GPU) te dolazi za operacijske sustave Linux i Windows. Darknet podržava dvije međusobno neovisne opcije, OpenCV i CUDA (engl. *Compute Unified Device Architecture*) tehnologiju [26]. OpenCV se koristi ako se traži širi izbor podržanih vrsta slika, a CUDA ako se želi izvedba preko GPU. Darknet na CPU je brz, ali na GPU je i do 500 puta brži [10]. Uvjet je da računalo koje se koristi za razvoj algoritama u Darknetu mora imati GPU (koristi Nvidia GPU). CUDA je paralelna računalna platforma i model programskog sučelja (API) kreiran od strane Nvidia kompanije [26]. Omogućuje programerima i softverskim inženjerima korištenje grafičke procesorske jedinice. Ovaj pristup nazvan je računarstvo opće namjene na jedinicama za obradu grafičkih podataka (engl. *General-purpose computing on graphics processing units*, GPGPU). Platforma CUDA je softverski sloj koji daje izravan pristup virtualnom skupu instrukcija GPU-a i paralelnim

računskim elementima za izvršavanje računalnih kernela. Osmišljena je za rad s programskim jezicima C, C++ i Fortran [26]. U ovom radu korištena je CUDA tehnologija, a nakon instalacije Darknet-a i CUDA-e potrebno je samo u *makefile*-u Darkneta-a parametar *GPU* postaviti na 1. Nakon toga razvoj i izvršavanje cijelog algoritma prebacuje se na GPU razvojnog računala.

3.2. Predstavljanje vlastitog rješenja za detekciju prometnih znakova i prometne signalizacije

Zadaća je ovog algoritma pravilno prepoznati jedan od prezentiranih znakova, nakon čega bi se vozilu trebala dati naredba da se obavezno zaustavi. Treba naglasiti da samo slanje naredbe kao i upravljanje vozilom nisu predmet ovog diplomskog rada. Osim detekcije i klasifikacije znakova/semafora, algoritam također treba dati i procjenu udaljenosti vozila (kamere) od detektiranog znaka/semafora. Algoritam za procjenu udaljenosti opisan je također u ovom poglavlju. Ukoliko vozilo ima procjenu udaljenosti do znaka nakon kojeg je potrebno zaustavljanje vozila i zna svoju trenutnu brzinu, vrlo lako može proračunati model usporavanja, tj. način i količinu kočenja, kako bi na vrijeme stalo.

Cilj je ovog rada što bolje istrenirati parametre mreže YOLOv3 kako bi detekcija i prepoznavanje bile što uspješnije. Osim toga, potrebno je kreirati bazu slika koja je podijeljena u trening skup i validacijski skup. Vremenski uvjeti u kojima bi algoritam trebao ispravno detektirati i klasificirati spomenute znakove su sljedeći:

- Dan (sunčano vrijeme)
- Dan (kišno vrijeme)
- Noć (vedro vrijeme)

Kao što je navedeno u drugom poglavlju ovog rada, postoje mnogi radovi i mnoga istraživanja u ovom području [10], [11], [12], [13] ali budući da je ovo specifičan slučaj detekcije i prepoznavanja, ne postoji gotov model za detekciju i prepoznavanje upravo navedene skupine znakova u ovim vremenskim uvjetima. Razvijeni su algoritmi za detekciju i prepoznavanje koji prepoznaju određene znakove, ali uglavnom u idealnim uvjetima. Cilj je ovog rada postići što uspješniju detekciju, ali ne u idealnim uvjetima. Postoje mnogi radovi, od kojih je jedan opisan u drugom poglavlju, koji su se bavili ovakvim problemom, no rijetki su razvijani pomoću YOLOv3 algoritma.

3.2.1. Baza podataka i postupak treniranja algoritma

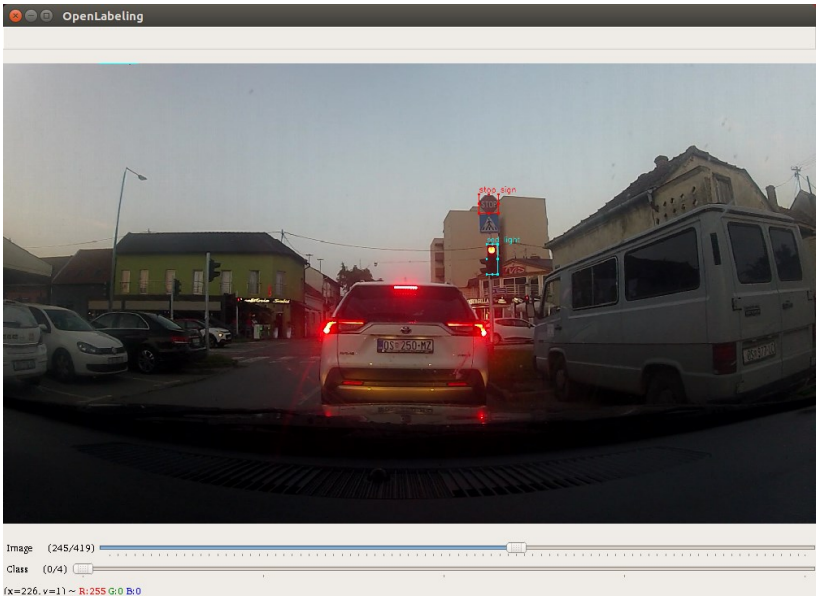
Trening skup podataka, koji će se koristiti pri treniranju parametara mreže YOLOv3 algoritma, sadržava slike znakova u ranije spomenutim uvjetima. Slike sadrže scene iz svakodnevnog prometa na kojima su pozicionirani znakovi i signalizacija. Dakle, znakovi i signalizacija nisu preko cijele slike, nego predstavljaju realan pogled iz vozila. Za treniranje algoritma kreirana je posebna baza slika koja sadrži slike znakova i signalizacije spomenute u prethodnom poglavlju. Najprije su snimani video zapisi na području grada Osijeka kamerom GoPro Hero3 rezolucije 1280x720s 30 FPS [27]. Video zapisi su snimani u ranije spomenutim vremenskim uvjetima, a znakovi i prometna signalizacija su zabilježeni iz različitih pozicija. Prema tome, znakovi i signalizacija su na pojedinim slikama bliži vozilu, na pojedinima dalji od vozila, a karakterizira ih i ugao iz kojega su snimani. Snimljeni video zapisi podijeljeni su nakon toga na okvire koji čine slike namijenjene za treniranje. Na slici 3.4 dan je primjer jedne slike namijenjene skupu slika za treniranje. Trening parametara mreže YOLOv3 proveden je na stolnom računalu s procesorom Intel i7-8700 3.20Ghz, 16 GB RAM memorije te grafičkom karticom Nvidia RTX 2060 6GB, na Ubuntu 16.04 operacijskom sustavu.

Sl.3.4. *Primjer slike iz skupa za treniranje parametara mreže YOLOv3 algoritma.*

Snimljeno je ukupno 5500 slika, od kojih je izdvojeno 4500 slika koji sadrže objekte od interesa. Dodatno, na izdvojene slike primijenjene su i metode augmentacije kako bi se na

umjetan način povećao broj slika za treniranje i kako bi se dobila veća otpornost na šum u slikama. Tako je originalnim slikama mijenjana oštrina. Na ukupno 250 snimljenih slika primijenjena je metoda augmentacije na način da je na 250 slika smanjena oštrina. Na još dodatnih 250 primijenjena je metoda zrcaljenja pod uvjetom da ne mijenja značenje znaka. Tako je dobiveno još 500 dodatnih slika, te je stvoren skup od ukupno 5000 slika (4500 originalnih i 500 dodatnih). Sve slike korištene u procesu treniranja parametara mreže nalaze se u prilogu P.3.1. na elektroničkom mediju uz rad.

Nakon kreiranja baze slika, bilo je potrebno označiti sve znakove od interesa na trening slikama (označiti *ground truth* za objekte od interesa). Za označavanje slika koristio se alat *OpenLabeling* [28]. *OpenLabeling* je jednostavna Python skripta koja služi za ručno kreiranje graničnih okvira oko željenih objekata. Najprije se sve trening slike ubace u mapu *input* nakon čega se pokreće skripta *main.py* koja za svaku sliku automatski generira tekstualne datoteke. U njima se nalaze informacije o označenim objektima. Prva vrijednost odnosi se na broj klase određenog objekta, druge dvije vrijednosti opisuju koordinate središta označenog graničnog okvira, dok posljednje dvije vrijednosti opisuje visinu i širinu graničnog okvira, odnosno samog objekta. Na slici 3.5 dan je primjer označavanja ciljanog objekta u alatu *OpenLabeling*, a na slici 3.6 izgled tekstualne datoteke u kojoj se nalaze podaci koji se spremaju za pojedini ručno označeni granični okvir. Nakon što se označe sve slike, potrebno je generirati najbolje parametre parametre mreže (modela) do kojih će se doći kroz postupak treniranja opisan u nastavku ovog poglavlja.



Sl.3.5. *Označavanje objekata u alatu OpenLabeling*.

Sl.3.6. *Broj klase, koordinate, visina i širina graničnog okvira koji se nalaze u tekstualnoj datoteci.*

Prije samog početka treniranja, slike s ručno označenim graničnim okvirima (njih 5000) su razdvojene u 2 skupa. Kreiran je skup slika za trening i skup za validaciju, u omjeru 4:1 (4000 slika za trening, 1000 slika za validaciju). Skup za trening sadrži 4000 slika koje su korištene samo u procesu treniranja, dok skup za validaciju sadrži 1000 slika na kojima su se ispitivale mjere uspješnosti (preciznost, odziv, mAP, IoU) za vrijeme treniranja. Nakon što se istreniraju parametri mreže, rješenje će se testirati na okvirima video sekvenci iz stvarnog života o kojima će više biti rečeno u četvrtom poglavlju, budući da one predstavljaju treći skup slika, tj. testni skup, i nisu korištene za vrijeme treniranja.

Objekti od interese podijeljeni su u pet klasa: semafor s uključenim crvenim svjetlom, obavezno zaustavljanje, zabrana prometa u oba smjera, zabrana prometa u jednom smjeru, semafor s upaljenim crvenim svjetlom pri nailasku na pružni prijelaz. Na nekim slikama nalazi se više objekata od interesa pa tako skup za treniranje sadrži više označenih znakova i signalizacija nego što ima slika. Kao što je ranije spomenuto, cjelokupni stvoreni skupa slika (njih 5000) podijeljen je na skupove za trening i validaciju. Primijenjena je jednostavna Python skripta koja jednom podijeli ukupni skup na trening skup koji se sprema u *train.txt* tekstualnu datoteku i na skup za validaciju koji se sprema u *validation.txt* tekstualnu datoteku. *Validation.txt* sadrži 20% nasumično odabranih slika iz početnog ukupnog broja sakupljenih slika. U tablici 3.1 dani su podaci o broju označenih objekata po pojedinim klasama i o ukupnom broju označenih objekata u slikama skupa za treniranje i validaciju. Na ukupnom broju od 5000 slika nalazi se 6930 oznaka. Podaci iz tekstualnih datoteka *train.txt* i *validation txt* nalaze se u prilogu P.3.2. na DVD-u uz rad.

Tab.3.1. *Broj označenih objekata po klasama u ukupno 5000 sakupljenih slika koje se koriste za trening i validaciju.*

Tip prometnog znaka	Broj oznaka
Semafor s uključenim crvenim svjetlom	3103
Semafor s uključenim crvenim svjetlom pri nailasku na pružni prijelaz	552
Obavezno zaustavljanje	1629
Zabrana prometa u jednom smjeru	726
Zabrana prometa u oba smjera	920
Ukupno	**6930**

U tablici 3.2 dani su podaci o broju označenih objekata po pojedinim klasama te o ukupnom broju označenih objekata u skupu za treniranje koji sadrži 4000 slika. Podatak o broju označenih objekata na skupu za validaciju od 1000 slika nalazi se u tablici Tab 3.3.

Tab.3.2. *Broj označenih objekata po klasama u ukupno 4000 slika koje se koriste za trening*

Tip prometnog znaka	Broj oznaka
Semafor s uključenim crvenim svjetlom	2712
Semafor s uključenim crvenim svjetlom pri nailasku na pružni prijelaz	369
Obavezno zaustavljanje	1304
Zabrana prometa u jednom smjeru	425
Zabrana prometa u oba smjera	624
Ukupno	**5435**

Nadalje, kreirane su tekstualne datoteke *traffic-sign-yolov3.names* i *traffic-sign-yolov3.data* iz kojih algoritam čita potrebne informacije o označenim slikama. *Traffic-sign-yolov3.names* sadrži imena klasa, a *traffic-sign-yolov3.data* sadrži putanje do datoteka gdje se nalaze trening slike, imena i broj klasa te do datoteke u koju će se spremati težine (*traffic-sign-yolov3.weights*), tj. istrenirani parametri na osnovu kojih algoritam vrši detekciju i klasifikaciju. Nakon toga izmijenjena je konfiguracijska datoteka *traffic-sign-yolov3.cfg* prema vlastitim zahtjevima, kako bi algoritam bio podređen vlastitom trening skupu i njegovim parametrima. Iz konfiguracijske datoteke algoritam čita zapisane parametre i smješta ih u određene funkcije u kodu. *broj_klasa* postavljen je na pet, dok je vrijednost *filteri* u konvolucijskim slojevima izračunat prema formuli [19]:

$$filteri = (broj\_klasa + 5) * 3 \tag{3-6}$$

Prema formuli, *broj_filtera* postavlja se na 30 jer je *broj_klasa* za zadani zadatak 5. Broj 3 predstavlja broj razina na kojima radi YOLOv3. Minimalni broj epoha u procesu treniranja računa se prema formuli [23]:

$$Minimalan\ broj\ epoha = 2000 * broj\_klasa \tag{3-7}$$

Prema formuli, minimalan broj epoha postavljen je na 10000. Maksimalan broj epoha teoretski ne postoji, a može se odrediti prema parametrima (preciznost, odziv, mAP, IoU) koji se prate tijekom postupka treniranja. Prema [23] i njihovom radu s YOLOv3 algoritmom na COCO bazi slika, podešeni su i parametri koji utječu na proces treniranja. *Batch,* odnosno broj slika koji ulazi u slojeve mreže po jednoj epohi podešen je na 64, a dodatna podjela tog ulaza ovisi o parametru *subdivisions* koji je podešen na 8. Ovi parametri ovise i o karakteristikama grafičke kartice koja je korištena u procesu treniranja. Rezolucija slika pri treniranju podešena je u konfiguracijskoj datoteci na 416x416 što ubrzava proces u odnosu na slike veće rezolucije, ali potencijalna detekcija može biti manje precizna. Stopa učenja, tj. parametar *learning rate* postavljen je na 0.001 jer je prema [23] upravo to optimalna vrijednost u procesu treniranja. Kako bi detekcija bila što preciznija računaju se vlastiti predviđeni granični okviri (engl. *anchor box*). Računaju se već gotovim funkcijama programskog okvira *Darknet*, a potrebno je pozicionirati se u *darknet* mapu i pokrenuti naredbu:

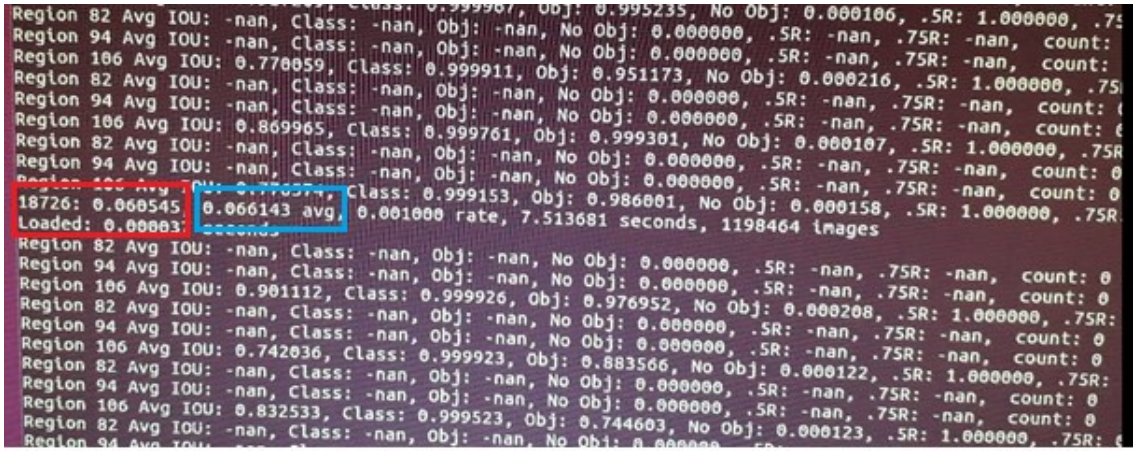
./darknet detector calc_anchors cfg/ traffic-sign-yolov3.data -num_of_clusters 6 -width 416 -height 416 –show

Nakon što su podešeni ključni parametri koji utječu na postupak treniranja potrebno je pokrenuti naredbu kojom počinje proces treniranja modela:

./darknet detector train cfg/ traffic-sign-yolov3.data cfg/ traffic-sign-yolov3.cfg darknet53.conv.74

Darknet53.conv.74 je datoteka koja sadrži početne težine (trenirane parametre za detekciju) prema kojima se detektiraju neke jednostavnije značajke na slici, a koriste se kako bi se ubrzao proces treniranja tj., generiranja vlastitih težina. Sadržaj konfiguracijskih datoteka nalazi se u prilogu P.3.3. na DVD-u uz rad.

Kao što je već spomenuto, testovi su vršeni na računalu s operacijskim sustavom Ubuntu 16.04, procesorom Intel i7-8700 3.20Ghz, 16 GB RAM i grafičkom karticom Nvidia RTX 2060 6GB. Inače trajanje procesa treniranja nije vremenski točno određeno kao ni točan broj iteracija. Prema [23] dovoljno je 2000 iteracija po jednoj klasi, ali ne manje od 4000 iteracija za sve klase zajedno. Kako bi se preciznije odredio kraj procesa treniranja, prate se određeni parametri i različiti pokazatelji pogreške. Na slici 3.7 dan je prikaz procesa treniranja iz kojeg iščitavamo potrebne parametre.



Sl.3.7. *Parametri praćeni u procesu treniranja.*

Broj *18726,* označen crvenim pravokutnikom, predstavlja broj iteracija, odnosno epoha kroz koje je proces treniranja trenutno prošao. Prema tome može se zaključiti da je broj iteracija po jednoj klasi jednak 3745, što je dovoljan broj iteracija prema [23]. Parametar *0.066143 avg*, označen plavim pravokutnikom na slici 3.7, predstavlja prosječan gubitak, tj. pogrešku (engl. *average loss*). Cilj je da je prosječni gubitak što manji, a kada se prestane smanjivati (tj. ustali se na određenoj vrijednosti) nakon određenog broja iteracija ili ako čak počne ponovo rasti, trebao bi se prekinuti proces treniranja. Konačni prosječni gubitak može biti od 0.05 (za manji

model i jednostavniji skup podataka za treniranje) do 3.0 (za veliki model i zahtjevnije skupove podataka za treniranje) [23]. Prema praćenim parametrima procijenjeno je da je prošlo dovoljno iteracija i da je pogreška od 0.066143 dovoljno mala pa se zaustavlja trening nakon 18726 iteracija. Kada se trening zaustavi, proces treniranja je završen i generirani su parametri CNN (engl. *weights*) koji će se u daljnjem radu testirati. YOLOv3 također pohranjuje parametre pri svakoj tisućitoj iteraciji pa se tako i one mogu koristiti prilikom validacije. Za svaku iteraciju zapisana je i prosječna pogreška te se može provjeriti za koju iteraciju je ona najmanja. Primjerice, trening je prestao nakon 18726 ponavljanja, ali najbolji rezultat na ipak daje jedna od prethodnih razina (npr. 7000, 8000) (to nije bio slučaj u ovom radu). Ova situacija može se javiti zbog pretreniranosti (engl. *overfitting*). Pretreniranost je slučaj kada se jako dobro ili čak idealno detektiraju objekti na slikama iz skupa podataka za trening, ali se ne mogu dovoljno dobro detektirati na novim nepoznatim slikama (gubi se sposobnost generalizacije).

3.3. Odabir najboljih parametara algoritma nakon procesa treniranja koristeći validacijski skup podataka

Nakon što se odredi završetak procesa treniranja (provede se 18726 iteracija) potrebno je provjeriti performanse algoritma za detekciju na validacijskom skupu podataka. Za provjeru performansi na validacijskom skupu i za kasnije testiranje na video sekvencama koriste se istrenirani parametri iz 18726. iteracije Tablica 3.3 prikazuje ukupan broj oznaka na 1000 slika validacijskog skupa i broj oznaka po klasama.

Tab.3.3. *Broj oznaka u validacijskom skupu podataka koji sadrži 1000 slika.*

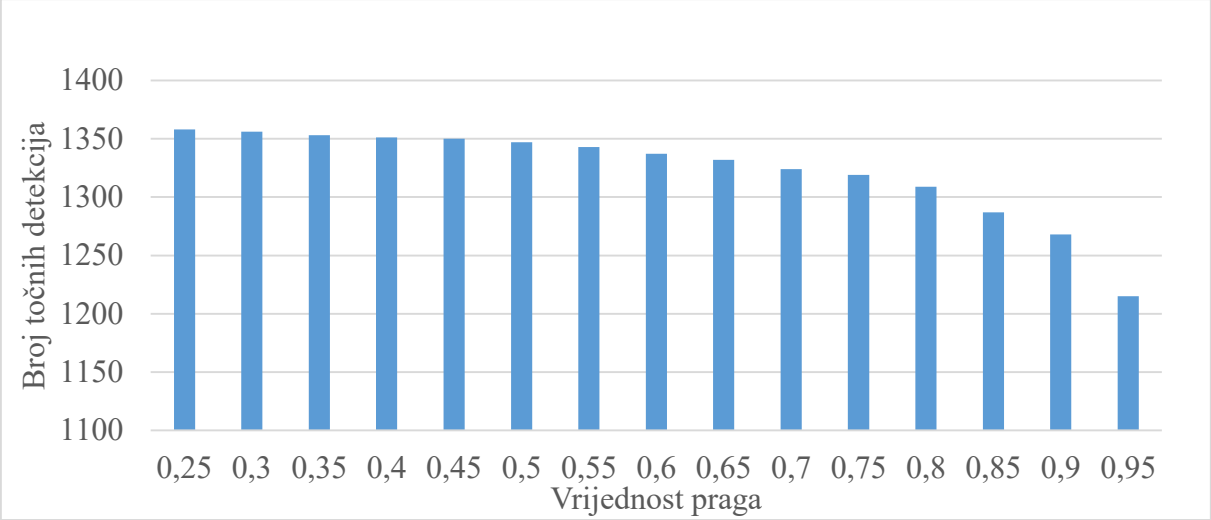
Tip prometnog znaka	Broj oznaka
Semafor s uključenim crvenim svjetlom	390
Semafor s uključenim crvenim svjetlom pri nailasku na pružni prijelaz	183
Obavezno zaustavljanje	325
Zabrana prometa u jednom smjeru	301
Zabrana prometa u oba smjera	296
Ukupno	**1495**

Validacija se zasniva na vrijednosti praga (engl. *threshold*), tj. minimalnoj vrijednosti praga sigurnosti detekcije s kojom algoritam detektira znakove. Ako je prag, primjerice, postavljen na 0.7, algoritam će prikazati one detekcije koje su detektirane sa sigurnošću većom od 70%. Ukupan broj oznaka znakova i signalizacije je 1495, a prag se mijenjao od 0.25 do 0.99, sa stopom povećanja praga od 0.05. Tablica 3.3. prikazuje broj TP, FP i FN detekcija, a za detekciju su korišteni parametri istrenirani nakon 18726 iteracija. Ove mjere općenito se koriste pri testiranju algoritama za detekciju objekata [9]. Tablica 3.4. prikazuje i rezultate ostalih razmatranih performansi mreže na validacijskom skupu podataka. Na osnovu rezultata iz tablice procjenjuje se najbolji parametri mreže ovisno o vrijednostima praga, preciznosti, odzivu i F1 mjere.

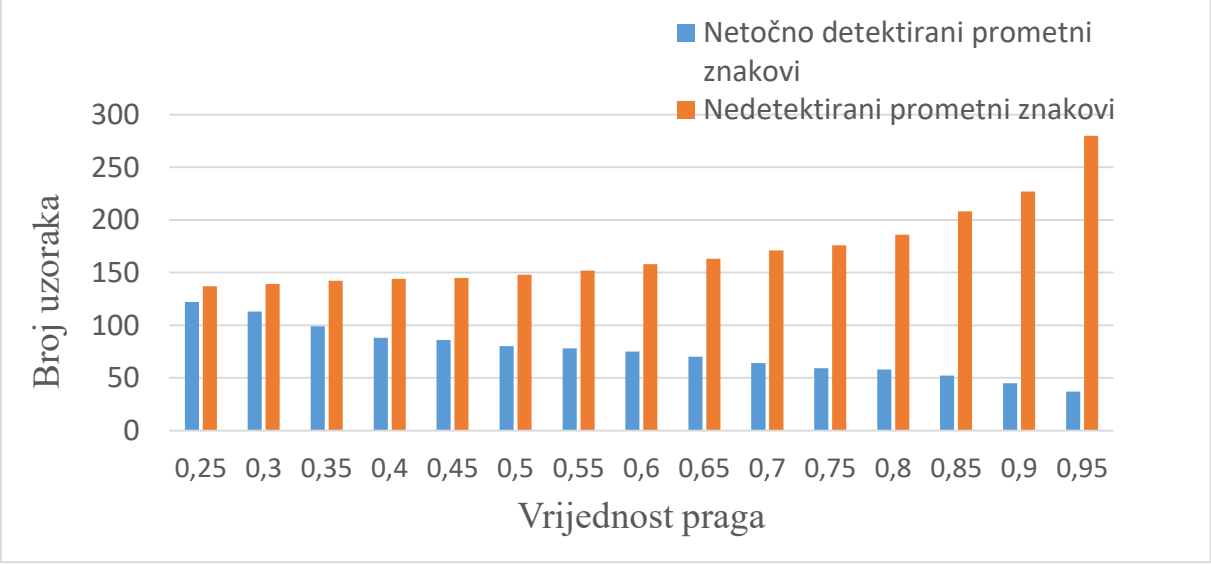
Tab.3.4. *Rezultati provjere performansi algoritma za detekciju i klasifikaciju znakova i signalizacije na validacijskom skupu podataka.*

Vrijednost praga sigurnosti detekcije	Preciznost	Odziv	F-1 vrijednost	Točne detekcije (TP)	Netočne detekcije (FP)	Objekti koji nisu detektirani (FN)	Prosječni IoU %
0.25	0.92	0.93	0.91	1358	122	137	73.60
0.30	0.92	0.93	0.91	1356	113	139	74.04
0.35	0.93	0.92	0.92	1353	99	142	74.78
0.40	0.94	0.91	0.92	1351	88	144	75.35
0.45	0.94	0.91	0.92	1350	86	145	75.47
0.50	0.94	0.91	0.92	1347	80	148	75.77
0.55	0.95	0.91	0.92	1343	78	152	75.88
0.60	0.95	0.90	0.92	1337	75	158	76.02
0.65	0.95	0.90	0.92	1332	70	163	76.32
0.70	0.95	0.90	0.93	1324	64	171	76.75
0.75	0.96	0.90	0.92	1319	59	176	76.95
0.80	0.96	0.89	0.91	1309	58	186	77.01
0.85	0.96	0.86	0.91	1287	52	208	77.41
0.90	0.97	0.85	0.90	1268	45	227	77.89
0.95	0.97	0.81	0.88	1215	37	280	78.52

Na slici 3.8 dan je graf koji prikazuje broj točnih detekcija (TP) za pojedine vrijednosti praga sigurnosti detekcije, dok je na slici 3.9 dana promjena broja netočno detektiranih prometnih znakova (FP) i nedetektiranih prometnih znakova (FN) za pojedine vrijednosti praga.



Sl.3.8. *Grafički prikaz broja točnih detekcija prometnih znakova u odnosu na vrijednost praga sigurnosti detekcije.*



Sl.3.9. *Grafički prikaz broja netočno detektiranih prometnih znakova i nedetektiranih prometnih znakova u odnosu na vrijednost praga sigurnosti detekcije.*

Prije analize samih rezultata, važno je napomenuti kako validacijski skup sadrži velik broj različitih znakova u različitim vremenskim uvjetima s jasnom (slika 3.10(a)) ili manje jasnom vidljivošću (slika 3.10(b)). Stoga se može zaključiti kako je skup podataka vrlo izazovan za

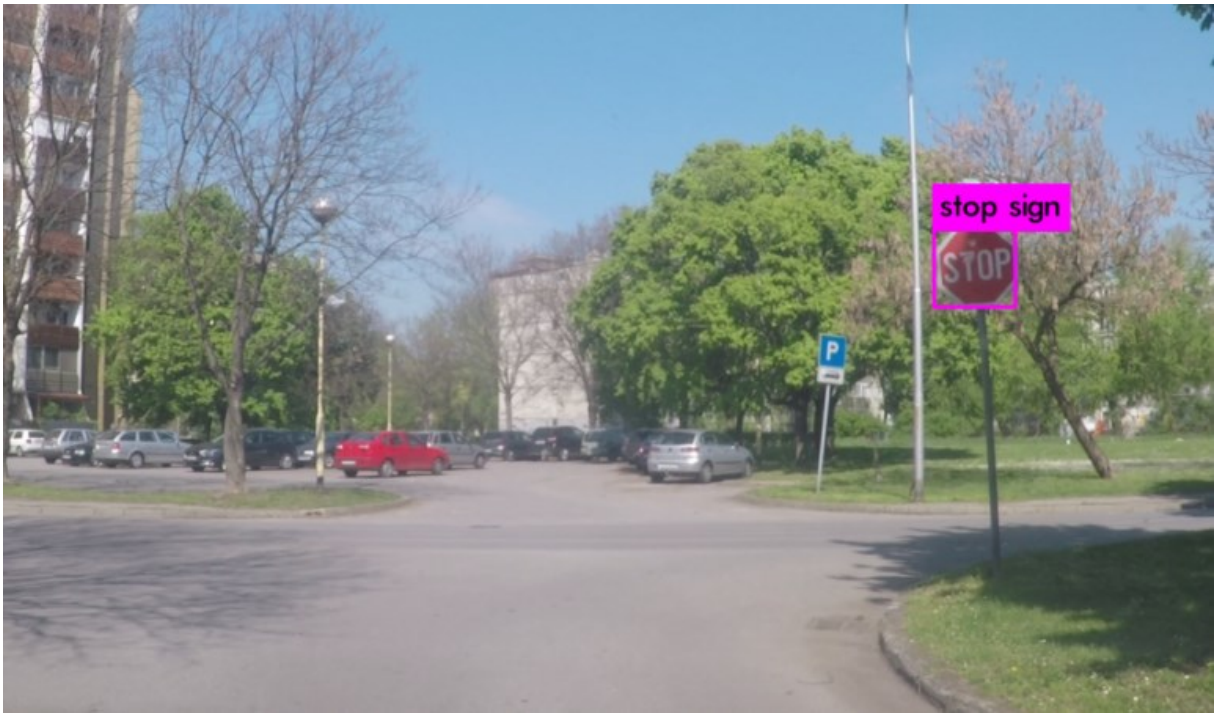
detekciju. U samom skupu nalaze se slike snimljene danju u idealnim uvjetima, noću i po kiši (što predstavlja teže uvjete).

(a)

(b)

Sl.3.10 *Slika znaka stop (a) po danjem svjetlu s jasnom vidljivošću (b) po noći s manje jasnom vidljivošću.*

Iz rezultata validacije može se zaključiti kako povećanjem vrijednosti praga raste preciznost, ali se smanjuje odziv. Ovo se događa iz razloga što će algoritam detektirati samo one objekte za koje je siguran minimalno onoliko kolika je vrijednost praga. Stoga, pri visokim vrijednostima ne detektira neke znakove jer nije dovoljno siguran, a ranije bi ih detektirao. S druge strane, pri nižim vrijednostima praga previše je netočno detektiranih (FP) objekata, za razliku od slučaja s visokim vrijednostima praga, gdje je prevelik broj nedetektiranih objekata (FN). Budući da je teško kreirati idealan algoritam, treba se odlučiti za neku od inačica koja će najbolje raditi na stvarnim testnim video sekvencama. S obzirom na to važno je da svaki znak bude detektiran, ali da nema krivih detekcija kako se automobil ne bi na koncu nepotrebno zaustavio. Analizom rezultata odlučeno je da će krajnja inačica biti ona pri vrijednosti praga 0.7 budući da pri toj vrijednosti algoritam ima najviši parametar F1 koji iznosi 0.93, visok odziv od 0.9 i prosječnu preciznost od 0.95. Na slici 3.11 nalazi se jedan primjer uspješne detekcije objekta uz vrijednost praga sigurnosti detekcije od 0.7.



Sl.3.11. *Primjer uspješne detekcije znaka stop uz odabranu vrijednost praga sigurnosti detekcije 0.7.*

Na slici 3.12 nalazi se primjer netočne detekcije odnosno FP pri nižem pragu sigurnosti detekcije od 0.3. Ovdje algoritam okrugli znak na plakatu pored ceste prepoznaje kao znak obaveznog zaustavljanja (znak stop). Ovo se može dogoditi ako algoritam nije treniran kroz

dovoljan broj iteracija ili ne postoji dovoljno različitih slika znaka stop u skupu za treniranje pa algoritmu slični znakovi pored ceste djeluju kao jedan od znakova iz skupa. Također ako je odabrani prag sigurnosti detekcije nizak za određeni slučaj, algoritam će ovakve znakove detektirati kao traženi objekt, tj. znak stop u ovom slučaju.

Sl.3.12. *Detekcija nepoznatog objekta pored ceste kao znaka obaveznog zaustavljanja. Primjer FP.*

Ako je algoritam treniran kroz dovoljno iteracija i sadrži velik broj znakova stop u različitim uvjetima i pozicijama, a ovakvi FP se ipak događaju, onda se ovakav problem rješava podizanjem praga sigurnosti pri detekciji. Nakon što se digne prag sigurnosti na, primjerice, 0.7 algoritam više ne detektira ovaj znak kao znak stop. Ovaj slučaj dan je na slici 3.13

Sl.3.13. *Promjenom praga sigurnosti detekcije, nepoznati objekt više nije detektiran kao traženi objekt.*

3.4. Procjena udaljenosti vozila od detektiranog objekta

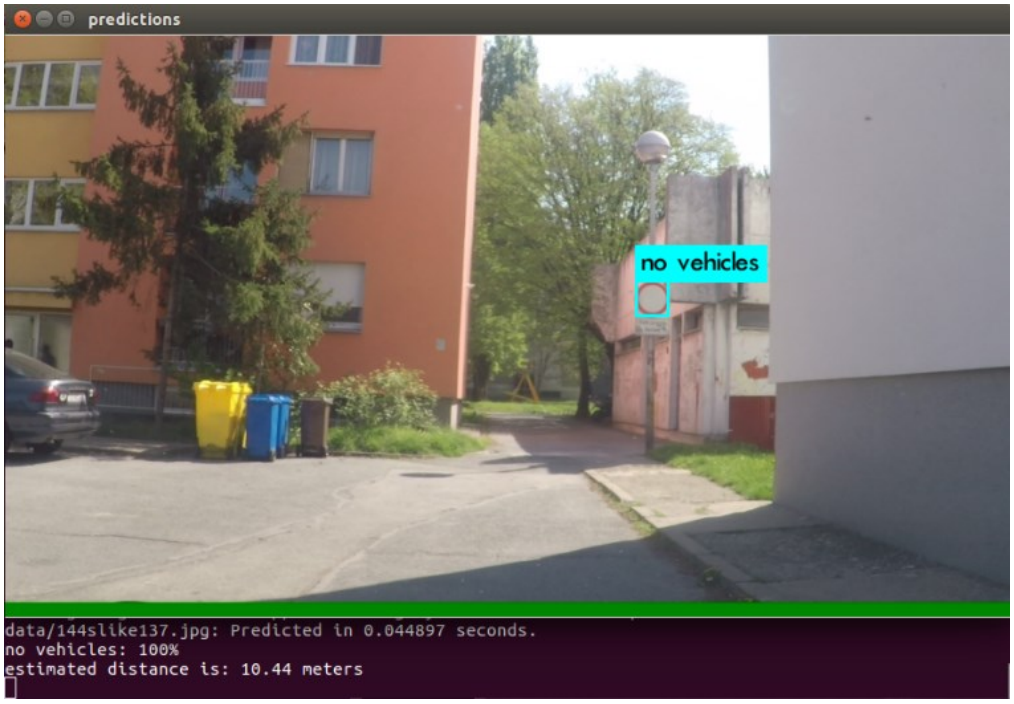
Druga zadaća ovog algoritma je procjena udaljenosti vozila od detektiranog objekta. Ovo je samo još jedan od načina kako smanjiti broj prometnih nesreća na prometnicama jer vozači često loše procjene udaljenost od raskrižja, znaka ili semafora pa pri velikim brzinama jednostavno ne stignu na vrijeme stati. Uz podatak o udaljenosti od znaka i podatak o brzini vožnje, vozač može znati koliko vremena ima za reakciju, a autonomno vozilo može znati kojim intenzitetom kočiti. Udaljenost vozila od detektiranog objekta može se računati i kao udaljenost kamere od detektiranog objekta, s tim što u tom slučaju treba uzeti u obzir i položaj kamere u vozilu kako bi se dobila točna udaljenost vozila od objekta. U nastavku je dan način proračuna udaljenosti kamere od detektiranog objekta. Potrebni parametri pri procjeni udaljenosti su parametri kamere kojom su slike snimane. Pri snimanju slika u ovom diplomskom radu korištena je kamera GoPro Hero3 prema kojoj su podešeni parametri u algoritmu za procjenu udaljenosti. Parametri kamere koji su potrebni za izračun udaljenosti su žarišna udaljenost leće (engl. *focal length*) i širina senzora (engl. *sensor width*). Kamera GoPro Hero3 ima 0.005376 m širinu senzora. Karakteristike leće koju kamera koristi nisu definirane, ali je njezina žarišna duljina jednaka 0.0036 m [27]. Udaljenost kamere od detektiranog objekta računa se na temelju formule [29]:

$$d[mm] = \frac{f[mm] * h[mm] * hslike[elementslike]}{h'[element\ slike] * hsenzor[mm]} \tag{3-7}$$

- d – udaljenost objekta od kamere na automobilu u milimetrima
- f – žarište duljina leće u milimetrima
- h – visina znaka u milimetrima u stvarnom svijetu
- h_{slike} – visina video okvira u broju elemenata slike
- h' – visina objekta u video okviru u broju elemenata slike, tj. visina graničnog okvira
- h_{senzor} – visina senzora kamere u milimetrima

Na slici 3.14 vidi se rezultat procjene udaljenosti detektiranog znaka. Procjena udaljenosti se radi u svakom video okviru. Važno je napomenuti da formula (3-7) vrijedi samo ako se objekt nalazi u sredini leće, ali kako je leća kamere koja se koristila dovoljno mala, moguće je koristiti navedenu formulu. Također je važno napomenuti kako je visina znakova korištenih u algoritmu 0.6 m za gradske ulice, a visina semafora 0.8 m što je uvršteno u formulu. Svi parametri čitaju

se iz specifikacija osim parametra h' koji predstavlja visinu graničnog okvira detektiranog objekta, a računa se kao udaljenost u elementima slike od gornje do donje linije graničnog okvira.



Sl.3.14. *Rezultat procjene udaljenosti kamere od detektiranog znaka.*

4. VERIFIKACIJA IPRAVNOSTI RADA ALGORITMA ZA DETEKCIJU PROMETNIH ZNAKOVA I PROMETNE SIGNALIZACIJE

U ovom poglavlju testirana je ispravnost rada algoritma za detekciju prometnih znakova i prometne signalizacije na video sekvencama iz stvarnog života. Za potrebe testiranja snimane su nove video sekvence koje ne sadrže okvire korištene pri treniranju parametara mreže, a snimane su također kamerom GoPro Hero3 na ulicama gradova Osijeka i Đakova. Testovi su vršeni na stolnom računalu s procesorom Intel i7-8700 3.20Ghz, 16 GB RAM memorije te grafičkom karticom Nvidia RTX 2060 6GB, na Ubuntu 16.04 operacijskom sustavu. Test će biti podijeljen u tri grupe od kojih svaka sadrži po tri dodatna pod-testa. Testni video materijali na kojima su ujedno dani i rezultati algoritma za pojedini test nalaze se na elektroničkom prilogu ovom radu, označeni kao prilog P.4.1. Sekvence su snimane na rezoluciji 1280x720 elemenata slike, ali su spuštene na rezoluciju 416x416 elemenata slike i imaju 24 FPS jer algoritam sporo radi pri većim rezolucijama. Testovi su podijeljeni na sljedeći način:

- Test 1: Detekcija i prepoznavanje semafora s uključenim crvenim svjetlom
    - Detekcija i prepoznavanje po danu (sunčano vrijeme)
    - Detekcija i prepoznavanje po noći
    - Detekcija i prepoznavanje po danu (kišno vrijeme)
- Test 2: Detekcija i prepoznavanje znakova obaveznog zaustavljanja, zabrane prometa u oba smjera, zabrane prometa u jednom smjeru
    - Detekcija i prepoznavanje znakova po danu (sunčano vrijeme)
    - Detekcija i prepoznavanje znakova po noći
    - Detekcija i prepoznavanje znakova po danu (kišno vrijeme)
- Test 3: Detekcija i prepoznavanje semafora s uključenim crvenim svjetlom pri nailasku na pružni prijelaz
    - Detekcija i prepoznavanje semafora po danu (sunčano vrijeme)
    - Detekcija i prepoznavanje semafora po noći
    - Detekcija i prepoznavanje semafora po danu (kišno vrijeme)

U daljnjem dijelu ovog poglavlja dana je analiza rezultata detekcije znakova i signalizacije po navedenim testovima te procjena udaljenosti kamere (vozila) od pojedinog znaka.

4.1. Test 1 - Detekcija i prepoznavanje semafora s uključenim crvenim svjetlom

U ovom testu vozilo se kreće prometnicom prema semaforu određenom brzinom. Test sadrži posebne situacije, odnosno slučajeve u kojima se semafor može nalaziti, kako bi se ispitala ispravnost detekcije u različitim uvjetima. Stoga test uključuje sljedeće situacije:

- Semafor pokazuje zeleno svjetlo
- Semafor pokazuje crveno svjetlo
- Semafor je ugašen
- Semafor treperi žutim svjetlom

Algoritam treba detektirati semafor samo kada je na njemu crveno svjetlo, što bi bio znak za zaustavljanje vozila. U svim ostalim situacijama algoritam ne treba detektira ništa. U tablici 4.1 dani su rezultati ovog testa. Testom se utvrđuju F-1 mjera, odziv i preciznost rada algoritma za tri različita vremenska uvjeta. Stupac *Broj objekata u videu* odnosi se na broj objekata koji moraju biti detektirani u ovom videu. Točnije rečeno, u video sekvenci iz tablice 4.1, snimanoj po danu nalazi se 27 semafora, u video sekvenci snimanoj po noći nalazi se 18 semafora, a u onoj snimanoj po kišnom vremenu nalazi se 19 semafora. Važno je napomenuti da se objekt, u ovom slučaju semafor, smatra detektiranim ako je detektiran u najmanje 16 FPS. Ovaj test, dakle, sadrži tri video sekvence koje imaju različito trajanje. Testna video sekvenca za dan traje 3 minute i 16 sekundi, za noć 1 minutu i 52 sekunde, a za kišu 3 minute i 10 sekundi.

Tab. 4.1. *Rezultati detekcije semafora s uključenim crvenim svjetlom u video sekvencama*

Vrijeme	Broj objekata u videu	Detektirani objekti (TP)	Netočno detektirani objekti (FP)	Objekti koji nisu detektirani (FN)	Preciznost	Odziv	F1 mjera
Dan	27	26	1	1	0.96	0.96	0.96
Noć	18	11	2	7	0.84	0.61	0.70
Kiša	19	15	1	4	0.93	0.78	0.84

Iz vrijednosti preciznosti (0.96), odziva (0.96) i F1 mjere (0.96) može se zaključiti kako algoritam prilično zadovoljavajuće detektira semafore danju po sunčanom vremenu. Problem stvara noć, kada semafori više ne svijetle jasnim i oštro vidljivim crvenim okruglim svjetlom, već se svjetlo previše raspršuje pa algoritam takav semafor ne vidi kao objekt od interesa. Ovaj problem se događa jer su slike lošije kvalitete pa algoritam teže raspoznaje, odnosno pravi razliku između semafora i nekog drugog svjetla na ulici (primjerice svjetlo automobila). Iz tog razloga ovaj test ima 2 FP i čak 7 FN u noćnim uvjetima, što donosi 0.84 preciznost i 0.61 odziv. Kada bi slike noćnog svjetla bile veće rezolucije, pravila bi se jasnija razlika između svjetla semafora i ostalih svjetala na ulici, a detekcija bi također bila točnija. No, problem je u tome što algoritam ne može obrađivati fotografije visoke rezolucije, tj. potrebna mu je veća računalna moć. Kada bi se algoritam vrtio na snažnijem računalu, a baza slika bila veća i sadržavala slike veće rezolucije, detekcija objekata u noćnim uvjetima bila bi puno bolja. Na slici 4.1 dan je primjer FN detekcije gdje algoritam ne detektira sve semafore.

Sl. 4.1. *Primjer FN u noćnim uvjetima pri detekciji semafora s uključenim crvenim svjetlom.*

Što se tiče kišnog vremena, algoritam radi s preciznošću od 0.93 te odzivom 0.78. Nešto je slabiji odziv jer zbog onečišćenog stakla pri kišnom vremenu, algoritam propusti detektirati određene semafore (4 FN). Kako bi uspio detektirati semafore i u takvim situacijama, potrebno je bazu slika za trening proširiti s puno više ovakvih primjera gdje se kroz kapljice kiše teže vidi traženi objekt. Na slici 4.2 dan je primjer FN detekcije po kišnom vremenu.

Sl. 4.2. *Primjer FN detekcije po kišnom vremenu pri detekciju semafora s uključenim crvenim svjetlom.*

FP detekcija događa se u slučajevima kada algoritam detektira semafor za pješake s uključenim crvenim svjetlom. Na slici 4.3 nalazi se primjer FP detekcije po kišnom vremenu gdje algoritam detektira crveno svjetlo na semaforu za pješake.

Sl. 4.3. *FP detekcija semafora za pješake s uključenim crvenim svjetlom po kišnom vremenu.*

4.2. Test 2: Detekcija i prepoznavanje znakova obaveznog zaustavljanja, zabrane prometa u oba smjera i zabrane prometa u jednom smjeru

U ovom testu vozilo se kreće prometnicom prema znaku određenom brzinom. Test sadrži tražene znakove i ostale znakove koje algoritam ne treba detektirati kako bi detekcija bila izazovnija. Traženi znakovi (koje treba detektirati) su:

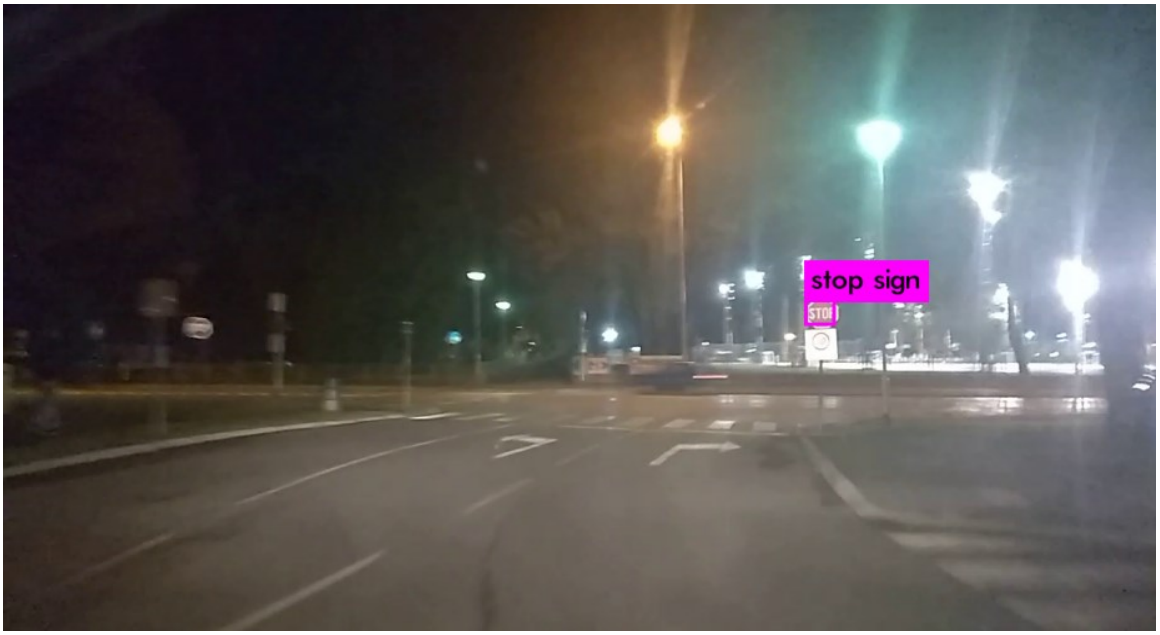
- Obavezno zaustavljanje
- Zabrana prometa u jednom smjeru
- Zabrana prometa u oba smjera

Algoritam treba detektirati navedene znakove koji nalažu obavezno zaustavljanje vozila kada ono na njih naiđe. U svim ostalim situacijama algoritam ne treba detektirati ništa. U tablicama 4.2, 4.3 i 4.4 dani su rezultati ovog testa za svaku klasu posebno. Tablica 4.2 prikazuje rezultate detekcije za znakove obaveznog zaustavljanja. Stupac *Broj objekata u videu* odnosi se na broj objekata koji moraju biti detektiranu u ovom videu, tj. u video sekvenci snimanoj po danu nalazi se 12 znakova, u video sekvenci snimanoj po noći nalazi se 11 znakova, a u onoj snimanoj po kišnom vremenu nalazi se 11 znakova. Važno je napomenuti da se objekt, u ovom slučaju znak zaustavljanja, smatra detektiranim ako je detektiran u najmanje 16 FPS. Ovaj test, dakle, sadrži tri video sekvence koje imaju različito trajanje. Testna video sekvenca za dan traje 4 minut i 30 sekundi, za noć 3 minute i 9 sekundi, a za kišu 6 minuta.

Tab. 4.2. *Rezultati detekcije znakova obaveznog zaustavljanja u video sekvencama*

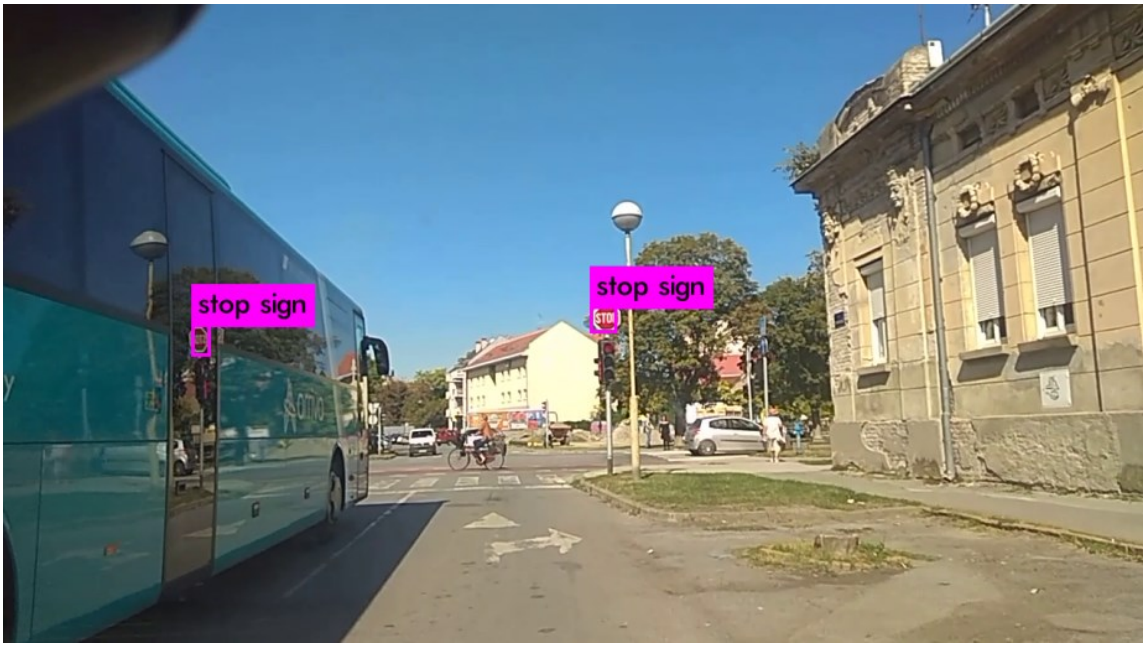
Vrijeme	Broj objekata u videu	Detektirani objekti (TP)	Netočno detektirani objekti (FP)	Objekti koji nisu detektirani (FN)	Preciznost	Odziv	F-1 mjera
Dan	12	12	1	0	0.92	1	0.95
Noć	11	9	0	2	1	0.81	0.89
Kiša	11	11	1	0	0.91	1	0.95

Analizom rezultata zaključuje se da algoritam gotovo bez greške detektira znakove u sva tri vremenska uvjeta. Najbolje rezultate postiže po danu s preciznošću 0.92 i odzivom 1. Problem se ponovo javlja po noći. U noćnim uvjetima ima 2 FN iz razloga što algoritam teže prepoznaje znakove koji se nalaze u slabo osvjetljenim ulicama. Na slici 4.4 dan je primjer FN u noćnim uvjetima.



Sl.4.4. *Primjer FN u noćnim uvjetima*

Na slici 4.4 detektiran je jedan znak obaveznog zaustavljanja dok je drugi ostao nedetekitran iz razloga što pod noćnim svjetlom postaje zamućen i teško ga je detektirati. Kako bi se ovaj slučaj popravio, potrebno je algoritam istrenirati na puno više slika snimljenih u mraku, pod ovakvim uvjetima. Također bi pomogla i viša rezolucija videa, no u tom slučaju algoritam radi puno sporije pa u pitanje dolazi njegova funkcionalnost u realnom vremenu. Na slici 4.5 nalazi se primjer FP po danu. Na ovom primjeru može se vidjeti kako je algoritam pogrešno detektirao znak obaveznog zaustavljanja kao odraz točno detektiranog znaka na prozoru autobusa. Ovakvi slučajevi su rijetki i teško se sprječavaju, no kada bi se dodali ovakvi okviri kao negativni primjeri u skup za treniranje, algoritam ne bi više detektirao ovakve primjere. Što se tiče kišnog vremena algoritam postiže rezultate preciznosti 0.91 i odziva 1 pa se zaključuje da kiša ne utječe loše na detekciju znakova.



Sl.4.5. *Primjer FP detekcije po danu.*

Tablica 4.3 prikazuje rezultate detekcije znakova zabrane prometa u jednom smjeru.

Tab. 4.3. *Rezultati detekcije znakova zabrane prometa u jednom smjeru u video sekvencama*

Vrijeme	Broj objekata u videu	Detektirani objekti (TP)	Netočno detektirani objekti (FP)	Objekti koji nisu detektirani (FN)	Preciznost	Odziv	F-1 mjera
Dan	5	5	0	0	1	1	1
Noć	5	4	0	1	1	0.80	0.88
Kiša	4	4	1	0	0.80	1	0.88

Iz rezultata se može vidjeti kako algoritam radi savršeno po danu sa gdje su mjere preciznosti, odziva i F-1 mjera jednake 1. Detekcija znakova po kišnom vremenu postiže nešto slabije rezultate od detekcije po danu, ali s druge strane jednake kao detekcija po noći. Na slici 4.6 nalazi se primjer FP detekcije po kišnom vremenu gdje algoritam izdaleka prepoznaje znak obaveznog zaustavljanja kao znak zabrane prometa u jednom smjeru. Ovo se događa iz razloga

što je natpis znaka obaveznog zaustavljanja djeluje kao ravna bijela crta u sredini znaka zabrane prometa u oba smjera. Kada se vozilo približi znaku, detektira ga točno.

Sl. 4.6. *Primjer FP detekcije po kišnom vremenu.*

Nadalje, u tablici 4.4 dani su rezultati detekcije znakova zabrane prometa u oba smjera.

Tab. 4.4. *Rezultati detekcije znakova zabrane prometa u oba smjera u video sekvencama*

Vrijeme	Broj objekata u videu	Detektirani objekti (TP)	Netočno detektirani objekti (FP)	Objekti koji nisu detektirani (FN)	Preciznost	Odziv	F-1 mjera
Dan	4	4	1	0	0.80	1	0.88
Noć	4	4	1	0	0.80	1	0.88
Kiša	4	3	0	1	1	0.75	0.85

Rezultati su približno isti u sva tri vremenska uvjeta s 2 FP i 1 FN. Sljedeći primjer na slici 4.7 primjer je FP u noćnim uvjetima. Prikazuje detekciju poleđine jednog znaka kao detekciju znaka zabrane prometa u oba smjera. Zbog prejakog obasjavanja svjetlima auta,

poleđina detektiranog znaka postaje potpuno bijela, pa algoritam ovaj znak prepoznaje kao znak zabrane prometa u oba smjera. Ovakvi primjeri također se rješavaju dodavanjem ovakvih i sličnih okvira kao negativan primjer u skup za treniranje.

Sl.4.7.. *Primjer FP detekcije u noćnim uvjetima.*

Primjer FN detekcije po kišnom vremenu dan je na slici 4.8 gdje algoritam ne detektira znak zabrane prometa u oba smjera kroz najmanje 16 FPS jer ga ometa rad brisača.

Sl. 4.8. *Primjer FN detekcije po kišnom vremenu.*

Analizom rezultata dolazi se do zaključka da algoritam vrlo dobro detektira znakove u sva tri vremenska uvjeta s visokim mjerama preciznosti, odziva i F-1 mjere. Razlog tomu je što skup slika za trening sadrži jako puno ovakvih slika pa je algoritam vrlo dobro istrenirao parametre CNN

4.3. Test 3 - Detekcija i prepoznavanje semafora na pružnom prijelazu

U ovom testu vozilo se kreće prometnicom određenom brzinom prema semaforu s uključenim crvenim svjetlom za regulaciju prelaska vozila preko pruge. Test sadrži posebne situacije, odnosno slučajeve u kojima se semafor može nalaziti kako bi detekcija bila otežana. Stoga test uključuje sljedeće situacije:

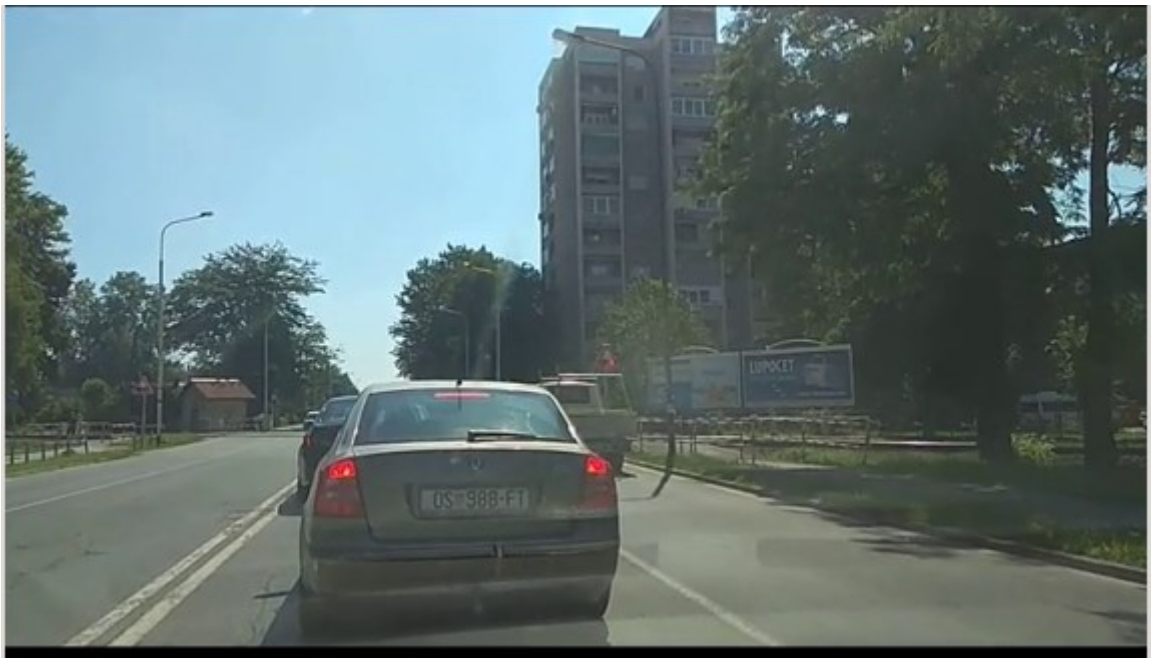
- Semafori su ugašeni i rampa je podignuta
- Na semaforima je uključeno crveno svjetlo i rampa je podignuta
- Na semaforima je uključeno crveno svjetlo i rampa je spuštena

Algoritam treba detektirati semafor samo kada je na njemu uključeno crveno svjetlo i rampa je ili podignuta ili spuštena, što bi bio znak za zaustavljanje vozila. U svim ostalim situacijama algoritam ne treba detektirati ništa. U tablici 4.5 dani su rezultati ovog testa. Stupac *Broj objekata u videu* odnosi se na broj objekata koji moraju biti detektiranu u ovom videu, tj. u video sekvenci snimanoj po danu nalazi se 12 semafora, u video sekvenci snimanoj po noći nalazi se 12 semafora, a u onoj snimanoj po kišnom vremenu nalazi se 11 semafora. Važno je napomenuti da se objekt, u ovom slučaju semafor smatra detektiranim, ako je detektiran u najmanje 16 FPS. Ovaj test, dakle, sadrži tri video sekvence koje imaju različito trajanje. Testna video sekvenca za dan traje 1 minutu i 40 sekundi, za noć 1 minutu i 52 sekunde, a za kišu minute i 9 sekundi.

Ovaj test je pokazao da algoritam najslabije performanse postiže upravo za ovu skupinu znakova. Budući da u gradu Osijeku ima samo nekoliko raskrižja gdje regulaciju vrše ovakvi semafori, a auto većinu vremena stoji ispred semafora, teško je skupiti puno slika za trening skup iz različitih uglova. Kada automobil u testnoj sekvenci naiđe na ovakvo raskrižje, kamera na semafor gleda iz potpuno drugačijeg ugla nego na trening slikama i stoga algoritam, primjerice u noćnim uvjetima, ima loš odziv od 0.50, dok mu je preciznost nešto veća 0.66. Ovaj problem mogao bi se djelomično riješiti dodavanjem puno više različitih slika u trening skup kako bi algoritam učio na što više različitih primjera. Na slici 4.9 dan je primjer FN detekcije gdje je vozilo previše udaljeno pa stoga „ne vidi“ semafor.

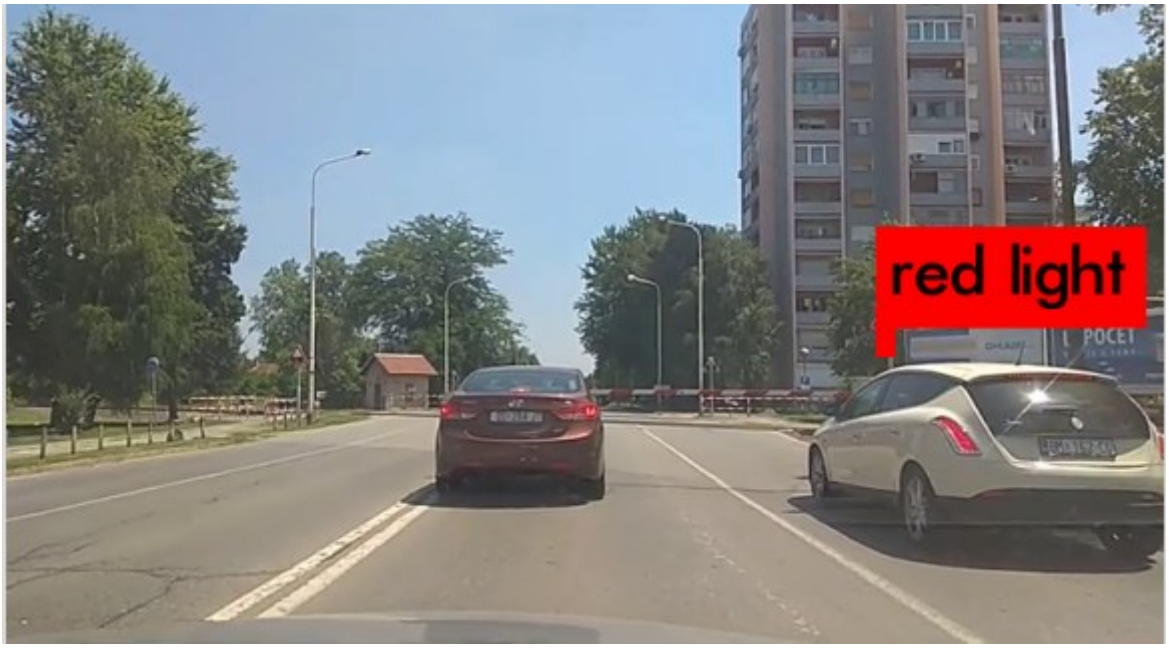
Tab. 4.5. *Rezultati detekcije semafora s uključenim crvenim svjetlom pri nailasku na pružni prijelaz u video sekvencama*

Vrijeme	Broj objekata u videu	Detektirani objekti (TP)	Netočno detektirani objekti (FP)	Objekti koji nisu detektirani (FN)	Preciznost	Odziv	F-1 mjera
Dan	12	8	0	4	1	0.66	0.79
Noć	12	6	3	6	0.66	0.50	0.56
Kiša	9	5	1	4	0.83	0.55	0.66



Sl. 4.9. *Primjer FN detekcije semafora s uključenim crvenim svjetlom pri nailasku na pružni prijelaz.*

FP detekcije se događaju iz razloga što algoritam detektira semafor s uključenim crvenim svjetlom umjesto semafora za regulaciju prometa preko pruge. Na slici 4.10 nalazi se primjer FP detekcije. Ovi problemi rješavaju se dodavanjem puno većeg broja slika za trening čije prikupljanje zahtjeva puno više vremena zbog manjka ovakvih prijelaza u gradu Osijeku te zbog gužvi na samim prijelazima.

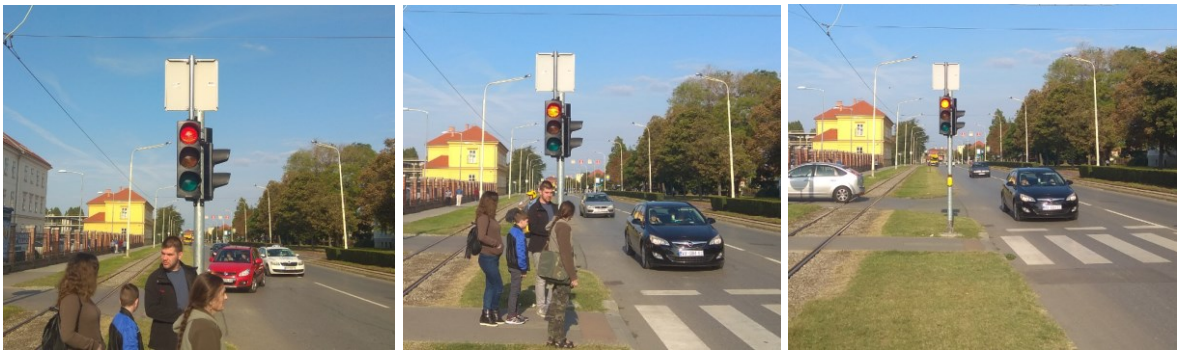


Sl. 4.10. *Primjer FP detekcije semafora s uključenim crvenim svjetlom pri nailasku na pružni prijelaz.*

4.4. Provjera točnosti proračuna udaljenosti vozila od detektiranog objekta

Za testiranje algoritma za procjenu udaljenosti snimljeno je deset znakova (4 znaka obaveznog zaustavljanja, 3 znaka zabrane prometa u jednom smjeru te 3 znaka zabrane prometa u oba smjera), na deset različitih lokacija i od toga za svaki znak po deset slika na definiranim udaljenostima. Osim znakova snimljeno je i deset semafora na deset različitih lokacija, za svaki također po deset slika na definiranim udaljenostima. Znakovi i semafori snimani su počevši od udaljenosti 5 m, a udaljenost se povećavala s korakom 5 m sve do udaljenosti od 50 m. Za ovaj test nisu korišteni video okviri iz testnih sekvenci jer se na njima nije znala udaljenost do objekata od interesa. Snimani su novi i računata je udaljenost. Na slici 4.11 nalazi se primjer semafora s uključenim crvenim svjetlom koji je sniman po navedenom principu.

Sve slike koje su se koristile u ovom testu prikladno su imenovane i nalaze se u elektroničkom prilogu ovog rada u prilogu P.4.2 Za svaku sliku poznata je stvarna udaljenost od znaka ili semafora, a znakovi i semafori snimani su pješice, kamerom GoPro Hero3 koja je postavljane u ravnini sa znakom. Nakon toga primjenjuje se algoritam za detekciju, koji na ovom uzorku od 100 slika ima 27 FN, te procjenjuje udaljenost. Rezultati procjene udaljenosti znakova od kamere nalaze se u tablici 4.4.



a) *b)* *c)*

d) *e)* *f)*

g) h) i)

j)

Sl. 4.11. *Semafor s uključenim crvenim svjetlom snimljen s deset različitih udaljenosti: a) 5m, b) 10m, c) 15m, d) 20m, e) 25m, f) 30m, g) 35m, h) 40m, i) 45m, j) 50m*

Izmjerena je apsolutna pogreška svake pojedine procjene i u tablici 4.4 je dana srednja apsolutna pogreška za svih 10 procjena na istoj udaljenosti. Apsolutna pogreška se računa prema formuli:

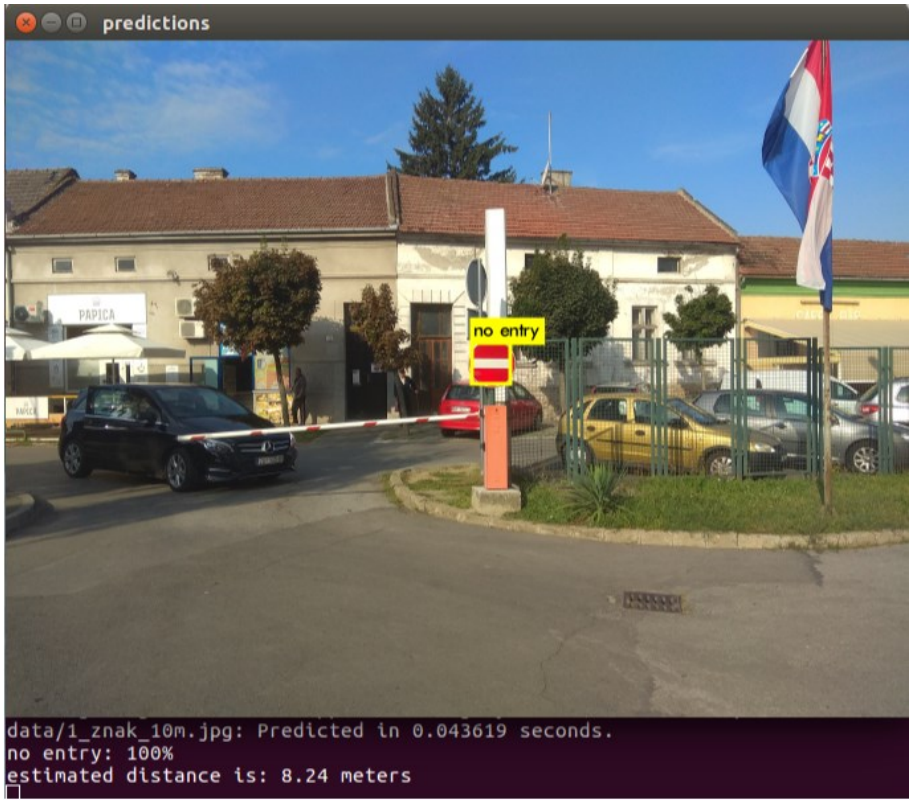
$$\Delta x = |x_i - X| \tag{4-1}$$

gdje je: x_i iznos pojedine procjene, a X iznos stvarne udaljenosti u stvarnom svijetu. Testovi 1-10 iz tablice odnose se na deset snimljenih znakova, a svaki test ima po 10 slika za 10 različitih udaljenosti (od 5m do 50m). Testovi 1-4 odnose se na znak obaveznog zaustavljanja, 5-7 na znak zabrane prometa u oba smjera, a testovi 8-10 na znak zabrane prometa u jednom smjeru. U tablicu se upisuje srednja apsolutna pogreška procjene udaljenosti dobivena za svih 10 mjerenje na jednoj udaljenosti.

Tab.4.6. *Rezultati algoritma za procjenu udaljenosti detektiranog objekta*.

Stvarna udaljenost (m)	Test 1 (m)	Test 2 (m)	Test 3 (m)	Test 4 (m)	Test 5 (m)	Test 6 (m)	Test 7 (m)	Test 8 (m)	Test 9 (m)	Test 10 (m)	Srednja apsolutna pogreška (m)
5	4.26	4.44	4.32	4.14	4.16	5.96	3.95	5.87	4.01	3.87	0.86
10	9.72	8.94	9.60	9.15	9.42	9.32	9.68	8.24	7.75	9.35	0.78
15	14.12	14.06	14.08	14.44	13.96	14.18	14.19	14.51	14.16	14.05	0.82
20	18.26	18.15	18.32	18.24	17.85	17.17	18.29	18.05	18.06	18.11	1.95
25	23.17	22.67	23.11	22.17	23.21	22.32	22.45	22.07	22.57	22.01	2.42
30	26.96	27.56	26.36	26.87	27.11	27.16	26.96	26.72	27.02	26.45	3.08
35	31.23	31.43	30.22	30.54	32.02	30.17	31.73	31.23	30.18	30.16	4.11
40	-	32.25	-	-	33.76	32.45	-	-	-	-	7.18
45	-	-	-	-	-	-	-	-	-	-	
50	-	-	-	-	-	-	-	-	-	-	

Iz dobivenih rezultata može se zaključiti kako je odstupanje najmanje pri udaljenostima od 10 do 20 metara. Ovo se događa jer je na tim udaljenostima znak najjasnije vidljiv i nalazi se direktno ispred leće kamere. Također kako udaljenost raste, raste i pogreška u mjerenju, a iznad 40 metara algoritam „ne vidi“ više znak. Odstupanja se događaju jer algoritam ne postavi savršeno granični okvir pa i sama visina okvira odstupa, a budući da se izračun zasniva na visini graničnog okvira u elementima slike, dobije se greška pri izračunu. Problem također uzrokuje rezolucija koju koristi algoritam pri detekciji jer pri većim udaljenostima znak se ne prikazuje vjerodostojno pa je i samim time granični okvir neprecizno postavljen. Također postoji mogućnost pogreške zbog neprecizno postavljene kamere. Naime kamera se nastoji postaviti u ravnini sa znakom, ali budući da se ručno postavlja postoji mogućnost pogreške izračuna koja je najizraženija pri velikim udaljenostima. Na slikama 4.12. (a) i 4.12 (b) nalazi se usporedba dvaju rezultata detekcije znaka zabrane prometa u jednom smjeru na stvarnoj udaljenosti od 10 metara.



(a)

b)

Sl.4.12. (a) i (b) *Rezultat detekcije znaka zabrane prometa u jednom smjeru na stvarnoj udaljenosti od 10 metara u dvije različite situacije*

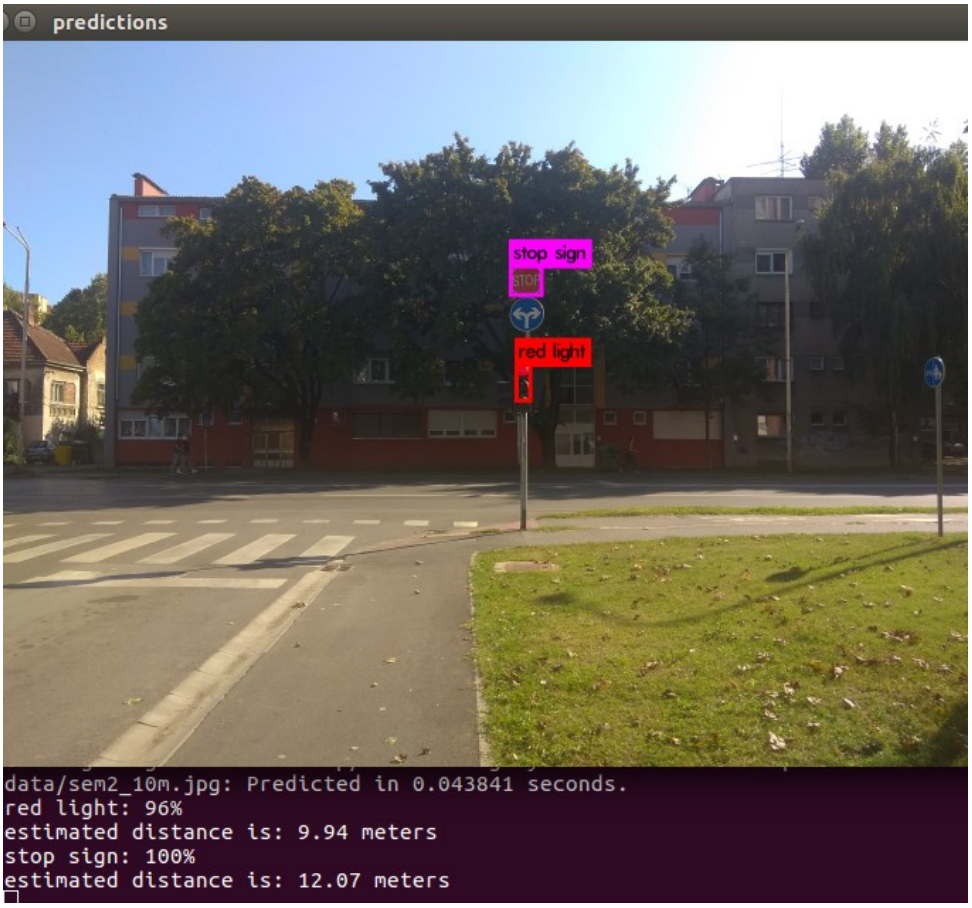
Može se vidjeti da je na slici 4.12.(b) granični okvir nešto kraći nego sama visina znaka pa algoritam procjenjuje manju udaljenost nego li na primjeru (a). U tablici 4.6 dani su rezultati procjene udaljenosti semafora od kamere. Testovi 1-10 predstavljaju deset različitih semafora gdje je svaki snimljen na udaljenostima od 5 do 50 metara. Kao i kod tablice 4.6. računa se srednja apsolutna pogreška procjene udaljenosti dobivena za svih 10 mjerenja na jednoj udaljenosti.

Iz dobivenih rezultata može se zaključiti kako su odstupanja najmanja od 10 do 20 metara kao i kod procjene za znakove. Ovo se događa jer je na tim udaljenostima semafor najjasnije vidljiv i nalazi se direktno ispred leće kamere. Prema dobivenim rezultatima može se vidjeti da i pri procjeni udaljenosti algoritam „ne vidi“ dalje od 35 metara. Postiže tek tri detekcije na udaljenosti od 40 metara. U ovom slučaju algoritam ima 28 FN detekcija testnih semafora od ukupno 100. Jednako kao i kod procjene udaljenosti kamere od znaka, odstupanja se događaju jer algoritam ne postavi savršeno granični okvir pa i sama visina okvira odstupa, a

budući da se izračun temelji na visini graničnog okvira u elementima slike, dobije se greška pri izračunu. Problem stvaraju i vrlo visoko postavljeni semafori ili znakovi pa jako teško padaju u sredinu leće na kameri što dodatno narušava preciznost detekcije. Na slici 4.13. dan je primjer detekcije znaka i semafora u isto vrijeme gdje je znak postavljen nešto iznad semafora i samim time rezultat procjene je puno veći.

Tab.4.7. *Rezultati algoritma za procjenu udaljenosti detektiranog objekta*

Stvarna udaljenost (m)	Test 1 (m)	Test 2 (m)	Test 3 (m)	Test 4 (m)	Test 5 (m)	Test 6 (m)	Test 7 (m)	Test 8 (m)	Test 9 (m)	Test 10 (m)	Srednja apsolutna Pogreška (m)
5	3.82	4.21	4.11	3.75	3.98	4.89	4.25	3.89	4.32	4.08	0.87
10	8.22	8.65	8.17	11.56	10.98	9.10	8.98	10.58	9.45	10.09	1.06
15	16.32	17.52	16.52	14.10	14.10	15.15	15.78	16.11	14.11	13.25	1.18
20	18.65	19.10	22.56	20.85	19.21	23.14	21.56	19.25	22.11	21.51	1.55
25	29.45	28.11	24.51	25.98	27.25	24.51	26.52	27.69	28.12	27.56	2.16
30	32.78	33.15	29.78	33.95	28.96	35.25	29.68	33.58	33.12	29.65	2.37
35	36.45	37.21	33.87	35.68	33.27	37.45	34.12	34.58	36.56	32.15	1.63
40	-	-	-	37.25	-	-	36.56	-	-	-	3.10
45	-	-	-	-	-	-	-	-	-	-	
50	-	-	-	-	-	-	-	-	-	-	



Sl.4.13. *Rezultat procjene udaljenosti kamere od znaka obaveznog zaustavljanja i semafora s uključenim crvenim svjetlom koji se nalaze na istoj stvarnoj udaljenosti od 10m.*

5. ZAKLJUČAK

U ovom radu razvijen je algoritam za detekciju i klasifikaciju prometnih znakova i prometne signalizacije koji nalažu izričito zaustavljanje vozila te procjenu udaljenosti vozila od detektiranog objekta. Za treniranje algoritma korišten je YOLOv3 model na skupu od 5000 slika sa 6930 oznaka namijenjenih za trening i validaciju. Baza slika sadrži slike znaka obaveznog zaustavljanja, znaka zabrane prometa u oba smjera, znaka zabrane promet au jednom smjeru, slike semafora s uključenim crvenim svjetlom te slike semafora za s uključenim crvenim svjetlom pri nailasku na pružni prijelaz. Slike su snimljene u različitim vremenskim uvjetima (dan, noć, kiša) što dodatno otežava detekciju. Trening skup sadrži 4000 slika s 5435 oznaka. Na testnim video sekvencama postignuti su rezultati iz kojih se zaključuje da algoritam u uvjetima danjeg svjetlu radi s odzivom 1 za znakove obaveznog zaustavljanja, 0.96 za semafore i 0.66 za semafore pružnog prijelaza. Rezultati detekcije znakova u ostalim vremenskim uvjetima nešto su lošiji, ali i dalje s visokim odzivom i preciznošću. Detekcija semafora u noćnim uvjetima radi najlošije, s odzivom od 0.61 što je zbog nedostatka trening slika u takvim uvjetima. Da je u sklopu rada načinjen bolji i širi skup slika, vjerojatno bi performanse samog algoritma bile bolje. Performanse algoritma bi mogle biti više dodavanjem dodatnih slika u skup za treniranje te snimanjem testnih video sekvenci kvalitetnije rezolucije, za koju bi algoritam mogao biti prepodešen. Algoritam za procjenu udaljenosti vozila od znaka radi s pogreškom od oko 10% pri udaljenostima do 20 metara. Povećanjem udaljenosti pogreška raste, a znakove „vidi“ do otprilike 40 metara. Algoritam radi na testnom računalu pri rezoluciji 416x416 uz 22-24 FPS i stoga se zaključuje da zadovoljava uvjete rada u stvarnom vremenu.

LITERATURA

[1] Prvi automobil, https://en.wikipedia.org/wiki/Car (pristupljeno: srpnja 2019.)

[2] KPMG, „I see. I think. I drive. (I learn). How Deep Learning is revolutionizing the way we interact with out cars“, https://assets.kpmg/content/dam/kpmg/se/pdf/komm/2016/se-isee-ithink-idrive-ilearn.pdf (pristupljeno: srpanj 2019.)

[3] M. Vranješ, R. Grbić, Predavanja iz kolegija Strojno učenje u sustavima autonomnih i umreženih vozila, studij Automobilsko računarstvo i komunikacije, ak.god. 2018/2019., https://loomen.carnet.hr/pluginfile.php/1946392/mod_resource/content/1/DeepLearning.pdf (srpanj 2019)

[4] S. Saha, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (pristupljeno srpanj 2019.)

[5] Convolutional Neural Network, https://missinglink.ai/guides/neural-network-concepts/convolutional-neural-network-build-one-keras-pytorch/ (pristupljeno: srpanj 2019.)

[6] C. Baskim, A. Mendelson, N. Liss, Streaming Architecture for Large-Scale Quantized Neural Networks, July 2017. https://www.researchgate.net/publication/318849314_Streaming_Architecture_for_Large-Scale_Quantized_Neural_Networks_on_an_FPGA-Based_Dataflow_Platform (pristupljeno: srpanj 2019.)

[7] Understanding of Convolutional Neural Network (CNN) – Deep Learning, https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 (pristupljeno: srpanj 2019.)

[8] Multi-Class Neural Networks: Softmax, https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax (pristupljeno: srpanj 2019.)

[9] J. Hui, mAP (mean Average Precision) for Object Detection, https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173 (pristupljeno: srpanj 2019.)

[10] Y. Wu, Y. Liu, J. Li, H. Liu, X. Hu, Traffic Sign Detection based on Convolutional Neural Networks, Proceedings of International Joint Conference on Neural Networks, Dallas, Texas, USA, August 4-9, 2013, http://xlhu.cn/papers/Wu13.pdf

[11] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection eith Region Proposal Networks, 6.Jan 2016

[12] A. Shustanov, P Yakimov, CNN Design for Real-Time Traffic Sign Recognition, April 2017, Samara, Rusija

[13] U. Kamal, S. Das, A. Abrar, K. Hasan, Traffic-Sign Detection and Clasification under challenging conditions: A deep Neural network based approach, Bangladesh University of Engineering and technology, September 2017, https://www.researchgate.net/publication/329999718_Traffic-Sign_Detection_and_Classification_Under_Challenging_Conditions_A_Deep_Neural_Network_Based_Approach

[14] P. Viola, M. Jones, „*Rapid Object Detection using a Boosted Cascade of Simple Features*“, Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Sjedinjene Američke Države, 2001.

[15] S. Rimac-Drlje, M. Vranješ, R. Grbić, predavanja iz kolegija Digitalna obrada slike i videa za autonomna vozila, studij Automobilsko računarstvo i komunikacije, ak.god. 2018/2019., https://loomen.carnet.hr/course/view.php?id=9844

[16] A. Martinović, G. Glavaš, M. Juribašić, D. Sutić, Z. Kalafatić, Real-time detection and recognition of traffic signs based on Viola-Jones Algorithm

[17] S. Houben, J. Stallkamp, J. Salmen. M. Schlipsing, C. Igel, Detection of traffic signs in real-world images, Dallas, Sjedinjene Američke Države, 2014

[18] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, *"You Only Look Once: Unified, Real-Time Object Detection",* 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, Sjedinjene Američke Države, 2016., str. 779-788.

[19] YOLO: Real-Time Object Detection, https://pjreddie.com/darknet/yolo/ (pristupljeno: kolovoz 2019.)

[20] Joseph Redmon, Ali Farhadi, YOLOv3: An incremental Improvement, University of Washington, https://pjreddie.com/darknet/yolo/

[21] N. S. Artamonov, P. Y. Yakimov, Towards Real-Time Traffic Sign Recognition via YOLO on Mobile GPU, 2018. Samara, Rusija

[22] J. Zhang, M. Huang, X. Jin, X. Li, A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2, September 2017.

[23] Github, AlexeyAB, https://github.com/AlexeyAB/darknet (pristupljeno: kolovoz 2019.)

[24] YOLOv3 Architecture, https://supervise.ly/explore/models/yolo-v-3-coco-1849/overview, (pristupljeno: kolovoz 2019.)

[25] Darknet: Open Source neural Network sin C, https://pjreddie.com/darknet/ (pristupljeno: kolovoz 2019.)

[26] CUDA, https://developer.nvidia.com/cuda-zone (pristupljeno: kolovoz 2019.)

[27] GoPro Hero3, https://www.cnet.com/products/gopro-hd-Hero3/specs/ (pristupljeno: kolovoz 2019.)

[28] Open Labeling: Label images and video for Computer Vision applications, https://github.com/Cartucho/OpenLabeling (pristupljeno: kolovoz 2019.)

[29] L.Ćosić, Procjena brzine vozila na temelju snimke kamere koja zamjenjuje retrovizor, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija, Osijek 2018.

SAŽETAK

U ovom radu razvijan je algoritam za detekciju i klasifikaciju prometnih znakova i prometne signalizacije, koji nalažu izričito zaustavljanje vozila, u različitim vremenskim uvjetima. Algoritam ja razvijan u svrhu primjene u sustavima autonomne vožnje. Za detekciju trenirao se YOLOv3 algoritam na označenim slikama iz trening skupa s područja grada Osijeka. Za razvoj algoritma koristio se programski okvir Darknet. Osim algoritma detekcije razvijen je i algoritam za procjenu udaljenosti vozila od detektiranog objekta. Algoritam postiže visoke performanse pri radu u uvjetima danjeg osvjetljenja, dok pri kišnom vremenu i noćnim uvjetima postiže lošije performanse.

Ključne riječi: detekcija i prepoznavanje prometnih znakova, YOLO, Darknet, autonomna vožnja

ABSTRACT

AUTOMATIC VEHICLE STOPPING BASED ON TRAFFIC SIGNALING

In this paper, an algorithm for the detection and recognition of traffic signs and traffic signalization was developed, which requires explicit stopping of vehicles under different weather conditions. Algorithm is for application in autonomous driving systems. For detection, the YOLOv3 algorithm was trained on tagged images from a training session in the Osijek city. The Darknet programming framework was used to develop the algorithm. In addition to the detection algorithm, an algorithm was developed to estimate the distance of the vehicle from the detected object. The algorithm achieves high performance when operating in daylight conditions, while in poor weather and night conditions it achieves poor performance.

Keywords: traffic sign detection and recognition, YOLO, Darknet, autonomous driving

ŽIVOTOPIS

Frane Mikulić rođen je 10.12.1994. godine u Đakovu. U Đakovu završava osnovnu školu „Ivan Goran Kovačić“ i 2009. upisuje Opću gimnaziju Antun Gustav Matoš u Đakovu. Nakon gimnazije upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Nakon završenog preddiplomskog studija 2017. godine upisuje diplomski studij – Automobilsko računarstvo i komunikacije na istom fakultetu.

Potpis:

_____

PRILOZI

- P.3.1. Slike korištene u procesu treniranja i validacije parametara algoritma za detekciju prometnih znakova i signalizacije, zasnovanih na YOLOv3 algoritmu. (priloženo na DVD-u uz rad)
- P.3.2. Tekstualne datoteke s nazivom slika i zapisom putanja do slika na testnom računalu. (priloženo na DVD-u uz rad)
- P.3.3. Konfiguracijske datoteke *cfg*, *names* i *data* s postavkama korištenim pri treniranju parametara YOLOv3 algoritma. (priloženo na DVD-u uz rad)
- P.4.1. Video sekvence korištene u testiranju istreniranih parametara YOLOv3 algoritma. (priloženo na DVD-u uz rad)
- P.4.2. Slike korištene pri testiranju udaljenosti vozila od pojedinog objekta. (priloženo na DVD-u uz rad)