Algoritam za uključivanje alarmnog sustava prilikom izlaska iz vozila

Ovžetski, Marija

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

Sveučilišni studij

ALGORITAM ZA UKLJUČIVANJE ALARMNOG SUSTAVA PRILIKOM IZLASKA IZ VOZILA

Diplomski rad

Marija Ovžetski

Osijek, 2023.

SADRŽAJ

1. UVOD

Automobilska industrija u zadnjih desetak godina prolazi kroz značajne transformacije, vođena sve većim napretkom umjetne inteligencije. Zahvaljujući njoj, automobili su sposobni obavljati određene vozačke funkcije samostalno, s minimalnom potrebom za intervencijom vozača. Konačan cilj automobilske industrije je razvoj potpuno autonomnih vozila.

Zadatak ovog diplomskog rada je razviti rješenje koje će detektirati sudionika prometa koji dolazi sa stražnje strane vlastitog vozila i upozoriti vozača na potencijalnu koliziju s njim ukoliko otvori svoja vrata. Procjenu opasnosti potrebno je obaviti na temelju udaljenosti detektiranog objekta od kamere i njegove brzine kretanja. Svakodnevno se pješaci, biciklisti i vozači suočavaju sa svim vrstama rizika i opasnosti dok se nalaze u prometu. Jedan od rizika s kojim se suočavaju je kolizija s vozačevim vratima. Sudionicima prometa koji dolaze sa stražnje strane vozila otvorena vrata vozila mogu zapriječiti put i može doći do kolizije. Opasnost nastaje kada se vrata automobila otvore jer tada sudionik prometa koji dolazi sa stražnje strane mora brzo reagirati kako ne bi došlo do mogućeg sudara. Udaljenost i brzina nadolazećeg sudionika prometa može se procijeniti pomoću senzora. Senzori se koriste kako bi automobilu pružili informacije o okolini automobila. Većina prometnih nesreća uzrokovana je ljudskom pogreškom. Ljudska pogreška ne može se kontrolirati i događa se jer vozač može izgubiti fokus zbog unutarnjih čimbenika kao što su umor i slabost ili vanjskih čimbenika poput razgovora s osobom pored sebe. Stoga se za smanjenje broja nesreća koriste senzori i sigurnosni sustavi za pomoć vozaču.

Nastavak ovog rada organiziran je na sljedeći način. U drugom poglavlju su opisani opći problemi koji se mogu pojaviti prilikom detekcije objekata u prometu. Objašnjeno je kako je sve moguće detektirati objekte i spomenute su različite vrste senzora. Obuhvaćene su neke od metoda za detekciju objekata, odnosno YOLO (engl. *You Only Look Once)* algoritam i njegove verzije. Također su obrađene konvolucijske neuronske mreže (engl. *convolutional neural network*, CNN) jer je YOLO algoritam zasnovan na tim mrežama. U trećem poglavlju je opisano vlastito rješenje za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila koji je izrađen u *Python* programskom jeziku pomoću YOLOv7 algoritma za detekciju objekata. U četvrtom poglavlju predstavljena je evaluacija točnosti i brzine rada rješenja pokrenutog na osobnom računalu i na Raspberry Pi 4 platformi. U petom je poglavlju iznesen zaključak rada.

2. PROBLEM DETEKCIJE OBJEKATA U PROMETU I POSTOJEĆA RJEŠENJA ZA UKLJUČIVANJE ALARMNOG SUSTAVA PRILIKOM IZLASKA IZ VOZILA

Detekcija objekata u prometu predstavlja ključni izazov u razvoju sigurnosnih sustava za vozila. Sposobnost precizne identifikacije objekata, kao što su druga vozila, pješaci i biciklisti, od velikog su značaja za pravovremeno reagiranje i sprječavanje nesreća. U potpoglavlju 2.1. predstavljen je problem detekcije objekata općenito. Opisane su neke od najpoznatijih arhitektura CNN-a za detekciju objekata: konvolucijske neuronske mreže zasnovane na regiji (R-CNN engl. *Region Based Convolutional Neural Networks*), YOLO i RetinaNet jedno stupanjski detektor. U potpoglavlju 2.2. predstavljena su neka postojeća rješenja, odnosno rješenja koja omogućuju uključivanje alarmnog sustava prilikom izlaska vozača iz vozila.

2.1. Detekcija objekata u slikama

Detekcija objekata zadatak je u računalnom vidu (engl. *computer vision*) koji uključuje pronalaženje jednog ili više objekata unutar slike i klasificiranje svakog objekta na slici. Lokalizacija identificira mjesto tih objekata odnosno daje lokaciju objekta [1]. Izazovan je zadatak u računalnom vidu uspješno otkriti i lokalizirati više objekata u slici. Međutim, u posljednjem desetljeću došlo je do značajnih postignuća u računalnom vidu. Detekcija objekata u prometu podrazumijeva proces prepoznavanja i lokalizacije objekata na temelju ulaznih podataka dobivenih senzorima. Takvi su sustavi prethodno predloženi na temelju kombinacije senzorskih tehnologija, kao što su radar i LiDAR (engl. *Light Detection and Ranging*). Međutim poboljšanja u strojnom učenju i otkrivanju objekata nude jedinstveni prijedlog jednostavnijeg i jeftinijeg sustava koji se zasnova samo na video sekvenci dobivenoj s kamere u vozilu.

Kamere su široko rasprostranjeni senzori u vozilima, koji daju vizualnu informaciju kako bi se na slici uz npr. duboko učenje detektirali objekti od interesa u prometu. Koristeći različite tehnike računalnog vida, na slikama iz kamere mogu se detektirati objekti na temelju njihovih vizualnih karakteristika, kao što su oblik, boja, tekstura i pokret. U posljednjim godinama, značajan napredak u detekciji objekata postignut je primjenom konvolucijskih neuronskih mreža (CNN) i dubokog učenja. Ove tehnike omogućuju automatsko izvlačenje značajki iz slika i prepoznavanje objekata s visokom točnošću [2].

Dugi niz godina uz kamere koristi se radarska oprema koja radi na fenomenu Dopplerovog pomaka, koja koristi elektromagnetske valove za detekciju objekata u prometu [3]. Radarski

senzori emitiraju elektromagnetske valove i mjeri se vrijeme koje je potrebno da se reflektirani signal vrati na senzor. Na temelju vremena povratka signala i promjene u frekvenciji, radar može odrediti udaljenost, brzinu i smjer objekata. Ova tehnologija omogućuje detekciju objekata bez obzira na uvjete osvjetljenja ili vremenske uvjete [4]. Glavni ciljevi bilo kojeg sustava detekcije sudionika prometa mogu se navesti na sljedeći način [3]: ispitati brzinu sudionika prometa i predvidjeti uzorke kretanja u video sceni, ukloniti šum u slici kako bi se omogućila poboljšana preciznost detektiranja, popuniti prazninu u djelomično zabilježenim događajima i osigurati da izračun brzine bude primjereno blizak stvarnoj brzini.

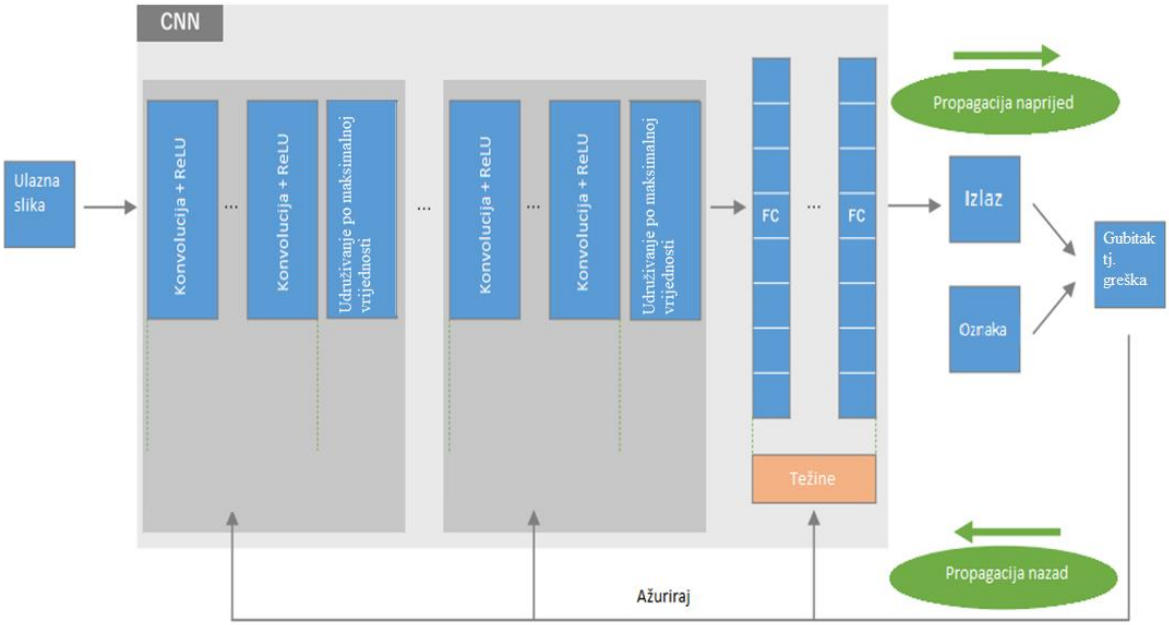
Prije nego što se duboko učenje započelo široko upotrebljavati, koristila se detekcija objekata drugim različitim tehnikama strojnog učenja [5]. Uobičajeno su uključivale tehniku detekcije objekata Viola-Jones algoritmom [6] ili tehniku detekcije objekata korištenjem izdvajanja i uparivanja značajki npr. korištenjem SIFT (engl. *scale-invariant feature transforms*) algoritma ili histograma orijentiranih gradijenta (engl. *Histograms of Oriented Gradients,* HOG). No tehnike zasnovane na dubokom učenju značajno nadmašuju ove metode.

Detekcija objekata u slikama postala je izuzetno uspješna zahvaljujući razvoju konvolucijskih neuronskih mreža i dubokog učenja. Ovi pristupi temelje se na korištenju dubokih neuronskih mreža s mnogo slojeva koje mogu naučiti iz velikih skupova podataka detektirati željene objekte. Osnovna ideja leži u tome da ulazna slika prolazi kroz mrežu koja izlučuje značajke različitih razina apstrakcije. Nakon toga, klasifikacijski slojevi mreže koriste dobivene značajke kako bi klasificirali objekte u različite kategorije. Neke od najpoznatijih arhitektura CNN-a za detekciju objekata uključuju: konvolucijske neuronske mreže zasnovane na regiji (engl. *Region Based Convolutional Neural Networks,* R-CNN) [7], YOLO [8] i RetinaNet [9] jedno stupanjski detektor. Svaka od ovih arhitektura ima svoje prednosti i specifičnosti, ali sve su usmjerene na postizanje visoke točnosti i brzine detekcije objekata u slikama.

2.1.1. Konvolucijske neuronske mreže (CNN)

Svojstvo po kojem se CNN ističe u odnosu na druge modele neuronskih mreža je njena sposobnost da analizira sliku kao 2D signal, a ne da ju rastavljaju u vektor. Postoji nekoliko različitih CNN arhitektura koje se koriste za detektiranje objekata. Važno je napomenuti da postoji razlika između konfiguracije mreže, odnosno broja i vrste korištenih slojeva i arhitekture mreže kada se radi o modelima detekcije objekata. CNN se obično sastoji od tri glavna tipa slojeva: konvolucijski sloj (engl. *convolutional layer*), sloj udruživanja (engl. *pooling layer*) i potpuno povezani sloj (engl. *fully connected layer*) [10]. Konvolucijski sloj izvodi izdvajanje značajki. Sloj

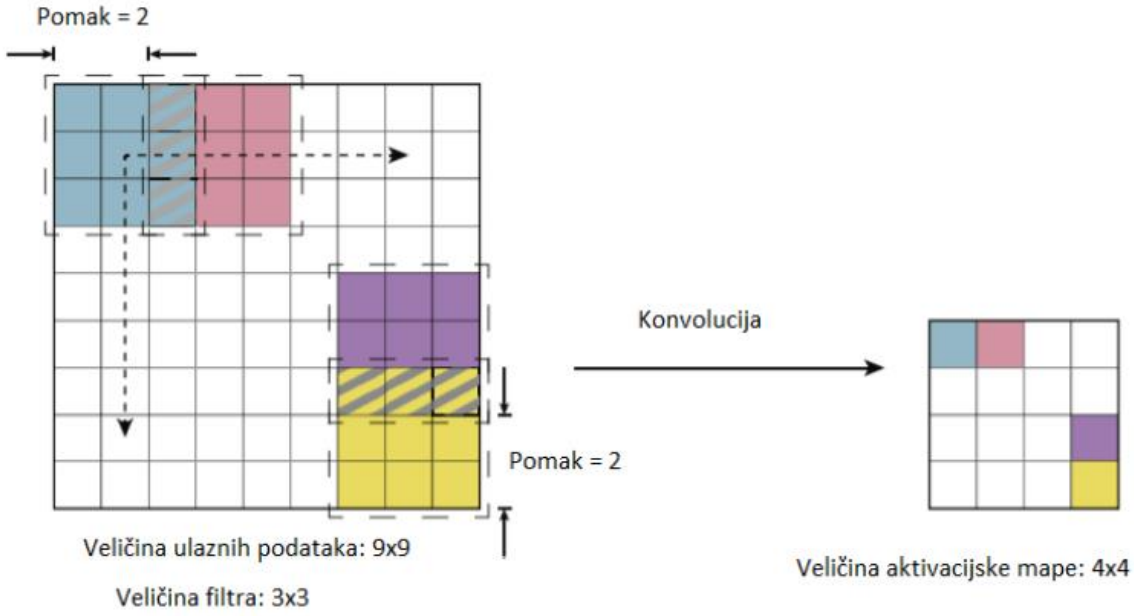
udruživanja smanjuje broj operacija konvolucije koje je potrebno provesti u mreži i donekle unosi invarijantnost na rotaciju. Treći, potpuno povezani sloj, služi da pomoću ranije izdvojenih značajki nauči ispravno klasificirati objekte i da omogući ispravnu klasu objekta na izlazu iz same mreže u zadacima klasifikacije. Konvolucijski sloj igra ključnu ulogu u CNN-u, a konvolucija je specijalizirana vrsta linearne operacije. Proces optimizacije parametara mreže naziva se treniranje, koje se izvodi tako da se minimizira razlika između izlaza (predviđanja) same mreže i stvarne istine (engl. *ground truth*), tj. labela (engl. *labels*) za dane ulazne trening podatke. Algoritam optimizacije se naziva propagacija unazad i koristi metodu gradijentnog spusta [10]. Kao što je već spomenuto, a i prikazano na slici 2.1., CNN se sastoji od slojeva: konvolucijski sloj, sloj udruživanja (npr. udruživanje po maksimalnoj vrijednosti (engl. *max. pooling*)) i potpuno povezani sloj.



Slika 2.1. Pregled arhitektura procesa treniranja konvolucijske neuronske mreže

Konvolucijski sloj osnovna je komponenta CNN arhitekture koja izvodi izdvajanje značajki, koja se obično sastoji od kombinacije linearnih i nelinearnih operacija, odnosno od funkcija konvolucije i aktivacije. Prema [11], konvolucijski sloj je najznačajniji sloj u CNN i u njemu se provodi većina zahtjevnih operacija cjelokupne mreže. Treba imati na umu da je primarna funkcija danog sloja provođenje konvolucije nad elementima slike odnosno pikselima. Može detektirati značajke objekta bez obzira na to gdje se određeni objekt nalazi na slici. Naime, prema [12], u konvolucijskom sloju nalazi se skup filtara, koji se tijekom procesa konvolucije postavljaju u gornji lijevi kut slike. Vrijednosti spomenutih filtara množe se s odgovarajućim elementima na slici, a zatim se navedeni umnošci zbroje. Navedeni postupak se provodi na slici na lokacijama

svih elemenata slike filtriranjem, nakon svake izvedene operacije konvolucije pomiču se za unaprijed određeni broj elemenata udesno. Kad je riječ o posljednjem desnom elementu slike, proces se nastavlja pomicanjem filtra na skup krajnje lijevih elemenata, koji su pomaknuti okomito prema dolje za prethodno određeni broj elemenata (slika 2.2.). Vrijednosti koje se nalaze unutar filtra čine niz parametara, takozvanih težina (engl. *weights*) koje se ažuriraju u svakoj iteraciji tijekom procesa učenja konvolucijske neuronske mreže. Na slici 2.2. može se vidjeti primjer operacije konvolucije nad ulaznom matricom veličine 9x9. Uz primjenu konvolucije i pomak elemenata za 2 te filtera veličine 3x3 kreira se aktivacijska mapa dimenzija 4x4. Važno je da filter ima istu dubinu kao i slika odnosno ulazni volumen na koji se primjenjuje.



Slika 2.2. Primjer operacije konvolucije nad ulaznom matricom 9x9 [11]

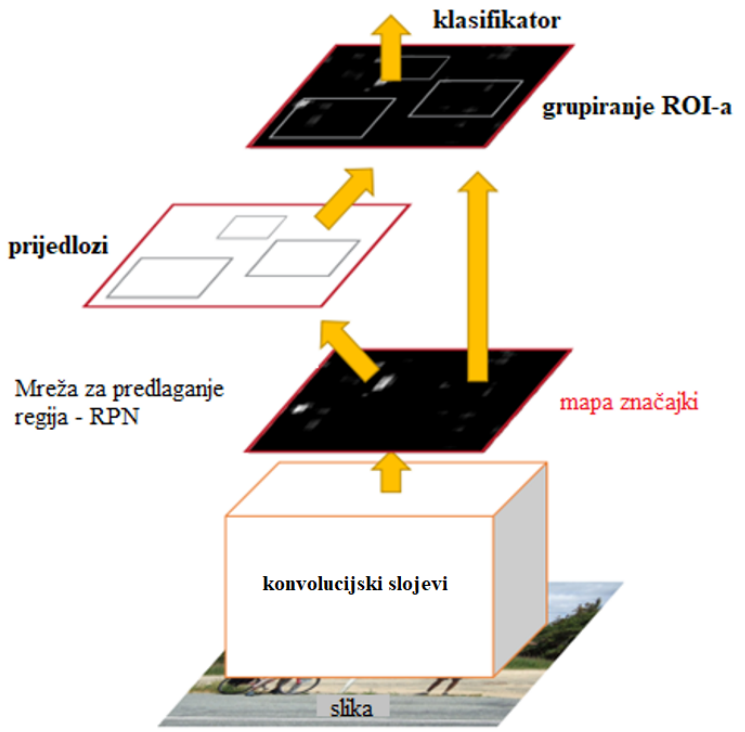
Sloj udruživanja omogućuje tipičnu operaciju smanjivanja dimenzija mapa značajki u prostoru. Smanjuje broj operacija konvolucije koje je potrebno provesti u mreži i donekle unosi invarijantnost na rotaciju. Važno je napomenuti da ne postoji parametar koji se može naučiti ni u jednom od slojeva udruživanja, dok su veličina područja udruživanja, korak i nadopuna hiperparametri u operacijama udruživanja, slično kao kod operacija konvolucije. Iako postoje mnoge vrste udruživanja, najčešće korištena u CNN-u je takozvano udruživanje po maksimalnoj vrijednosti [10].

Izlazne mape značajki posljednjeg konvolucijskog sloja ili sloja udruživanja obično su pretvorene u jednodimenzionalni niz brojeva ili vektora i povezane su s jednim ili više potpuno povezanih slojeva, poznatih i kao gusti slojevi (engl. *dense layers*), u kojima je svaki ulazni neuron povezan sa svakim izlaznim neuronom. Nakon što se stvore značajke izdvojene konvolucijskim

slojem, one se na određeni način kombiniraju podskupom potpuno povezanih slojeva [10], kako bi se dobila ispravna klasa objekta na izlazu iz mreže, tj. da pomoću ranije izdvojenih značajki nauči ispravno klasificirati objekte.

2.1.2. Konvolucijske neuronske mreže zasnovane na regiji (R-CNN)

R-CNN, ili regije sa značajkama CNN-a, se upotrebljava za detekciju objekata koje koriste CNN velikog kapaciteta za prijedloge regija odozdo prema gore kako bi lokalizirao i segmentirao objekte. R-CNN koristi selektivno pretraživanje za identifikaciju brojnih objekata za regiju graničnog pravokutnika (tzv. regije od interesa), a zatim izdvaja značajke iz svake regije. R-CNN uglavnom se koristi kao klasifikator i ne predviđa granice objekata. Brži R-CNN sastoji se od dva modula. Prvi modul je duboka potpuno konvolucijska mreža koja predlaže regije[1], a drugi modul je brzi R-CNN detektor koji koristi predložene regije [7]. Cijeli sustav je jedna, ujedinjena mreža za detekciju objekata prikazana na slici 2.3.



Slika 2.3. Jedinstvena mreža za otkrivanje objekata - Brži R-CNN [13]

U bržem R-CNN-u slika se koristi kao ulaz u konvolucijsku mrežu koja generira mapu značajki. Umjesto korištenja selektivnog algoritma pretraživanja na mapi značajki, za predviđanje

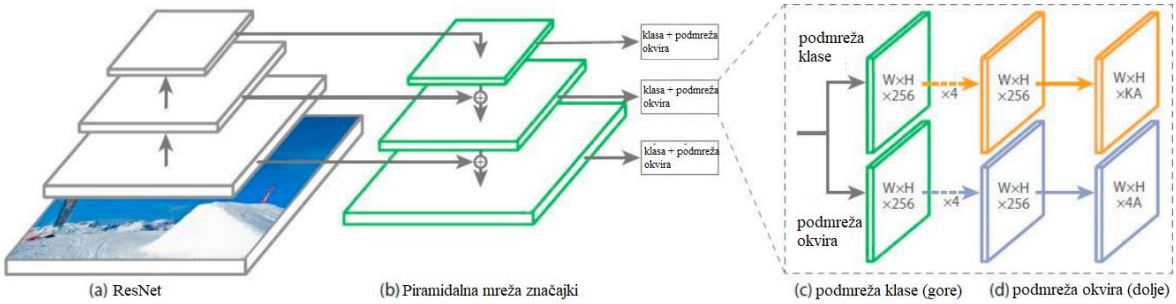
[1] „Regija“ je generički pojam i razmatraju se samo pravokutne regije

predložene regije koristi se zasebna mreža. Predviđene predložene regije zatim se preoblikuju pomoću ROI (engl. *Region of Interest*) sloja za udruživanje koji se koristi za klasificiranje slike unutar predložene regije i predviđanje vrijednosti pomaka za granične okvire (engl. *bounding box*).

R-CNN dijeli proces detektiranja u dvije odvojene faze. Prva faza procjenjuje sliku kako bi se izdvojio mogući ROI. Otprilike 2000 regija interesa je izdvojeno iz ulazne slike pomoću algoritma selektivnog pretraživanja prijedloga regije u prvoj iteraciji R-CNN, koji je prvi predložen u radu [10]. CNN mreža se zatim koristi za izdvajanje i klasificiranje značajki iz tih ROI-ja. Iz svakog ROI-a, ova CNN izvlači korisne značajke. U svrhu klasifikacije objekata, ove značajke se potom kombiniraju s klasifikatorom SVM (engl. *Support Vector Machine*). U sustavima autonomnih vozila, gdje je precizna detekcija ključna za razlikovanje dolazećih automobila i sudionika prometa, ova sposobnost je vrlo značajna. Brži R-CNN uvodi dva ključna koraka u odnosu na originalnu metodu R-CNN kako bi povećao učinkovitost i preciznost. Prvo se primjenjuje konvolucija na cijelu sliku kao dio faze predobrade. Zatim se koristi mreža za predlaganje regija (RPN engl. *Region Proposal Network*) koja je trenirana da izravno generira predložene regije iz konvolucijskih mapa značajki. Ove modifikacije omogućavaju bržem R-CNN-u značajno skraćivanje vremena izračuna, istovremeno povećavajući ukupnu preciznost modela. Dodavanjem učinkovitijeg mehanizma za predlaganje regija i primjenom konvolucijskog sloja na cijelu sliku, brži R-CNN poboljšava originalnu R-CNN arhitekturu. Ovi razvoji pomažu ubrzavanju računanja i poboljšanju preciznosti detekcije objekata.

2.1.3. RetinaNet algoritam

RetinaNet je duboka konvolucijska neuronska mreža implementiran slično kao YOLO algoritam detekcije. Ova vrsta mreže za detektiranje naziva se jedno-stupanjski model detekcije. U odnosu na dvo-stupanjski model detekcije prikazuje lošije rezultate kada je u pitanju točnost detektiranja. Ovaj neuspjeh uglavnom je posljedica neuravnoteženih klasa koji se susreće pri treniranju [9]. U radu [9] je riješen ovaj problem primjenom fokalnog gubitka. Fokalni gubitak je preoblikovanje gubitka unakrsne entropije tako da smanjuje težinu gubitaka dodijeljenim dobro klasificiranim primjerima. Novi fokalni gubitak usredotočuje se na treniranje na rijetkom skupu teških primjera i sprječava ogroman broj lakih negativnih primjera da preplave detektor tijekom treninga. RetinaNet također uspijeva usporediti oko 100 000 različitih segmenata za jednu sliku za razliku od modela kao što je YOLO koji klasificira između 98 i 2000 segmenata, ovisno o verziji. Kao što je prikazano na slici 2.4., ovaj model koristi *ResNet* za duboko izdvajanje značajki, kao i mrežu piramidi značajki (engl. *Feature Pyramid Network,* FPN).



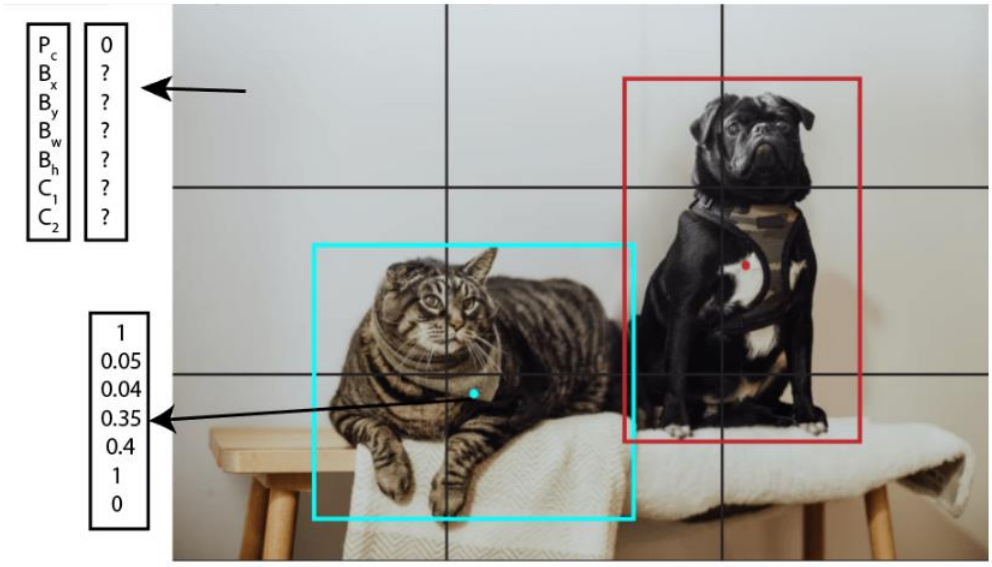
Slika 2.4. Mrežna arhitektura mreže RetinaNet [9]

U radu koriste dodatni sloj FPN-a na vrhu *ResNeta* za izgradnju bogatog skupa značajki koji može razlikovati različite uzorke slika i ispravno klasificirati više objekata [9]. FPN je zapravo konvolucijska neuronska mreža koja kao ulaz uzima sliku koja sadrži objekt koji treba prepoznati. Na temelju istog kreira izlaznu mapu značajki na više razina, čime se na kraju oblikuje piramida značajki (engl. *feature pyramid*). Budući da se svaka razina piramide koristi za detekciju objekta različitih veličina, navedena piramida značajki s više razina omogućuje prepoznavanje objekata različitih veličina na slici [9]. Dodatni sloj FPN-a omogućuje modelu da ima više razina značajki s različitim prostornim rezolucijama, što ga čini učinkovitijim u detektiranju objekata različitih veličina na slikama. To poboljšava sposobnost modela da razlikuje različite uzorke slike i ispravno klasificira više objekata na temelju tih značajki.

2.1.4. YOLO algoritam

YOLO algoritam za detekciju objekata razvijen je 2016. godine [8]. Većina modela detekcije objekata dijeli slike na područja interesa (ROI) ili segmente. Zatim se na te ROI ili segmente primjenjuje algoritam detekcije kako bi se klasificirao bilo koji objekt ili dio objekta. YOLO algoritam primjenjuje jednu neuronsku mrežu na cijelu sliku kao i običan CNN [8]. YOLO algoritam ima za cilj predvidjeti klasu objekta i granični pravokutnik koji definira mjesto objekta na ulaznoj slici. Prepoznaje svaki granični pravokutnik pomoću sljedećih predviđanja: vjerojatnost predviđanja, središte graničnog pravokutnika (x,y), širina pravokutnika (w), visina pravokutnika (h). Predviđena vjerojatnost predstavlja IoU (engl. *Intersection over Union,* IoU) tj. omjer presjeka između predviđenog graničnog pravokutnika i stvarnog graničnog pravokutnika te njihove unije [14].

Prvi korak koji YOLO algoritam radi je ravnomjerna podjela slike na rešetku koja se sastoji na N ćelija, gdje je S *x* S dimenzija rešetke u broju ćelija, a primjer je prikazan na slici 2.5 (S=3).



Slika 2.5. Predviđanje graničnog pravokutnika na slici [14]

Uz postojanje ćelije, moguće je otkriti jedan objekt po ćeliji umjesto jednog objekta po slici. Za svaku ćeliju kodira se vektor koji ju opisuje. Na primjer, prva ćelija odozgo lijevo prikazana na slici 2.5. nema nikakav objekt, a opisujemo je izrazom (2-1):

$$C_{1,1} = (P_c, \mathrm{B}_x, \mathrm{B}_y, \mathrm{B}_w, \mathrm{B}_h, \mathrm{C}_1, \mathrm{C}_2) = (0, ?, ?, ?, ?, ?, ?), \tag{2-1}$$

gdje je P_c vjerojatnost klase objekta, B_x i B_y su koordinate središta graničnog pravokutnika relativno u odnosu na ćeliju, B_w i B_h su širina i visina graničnog pravokutnika relativno u odnosu na cijelu sliku, C_1 i C_2 su 0 ili 1, ovisno o tome koja klasa je predstavljena unutar graničnog pravokutnika (u ovome primjeru C_1 se odnosi na mačku, a C_2 za psa). Vektor $C_{1,1}$ sastoji se od simbola „ *?* “ jer ako je prva komponenta P_c jednaka nuli, tada ostale komponente mogu imati slučajne brojeve ako se ne uzimaju u obzir. Zatim, na slici 2.5. ćelija koja sadrži središte plavog graničnog pravokutnika, prikazuje se izrazom (2-2):

$$C_{3,2} = (1, 0.05, 0.04, 0.35, 0.4, 1, 0), \tag{2-2}$$

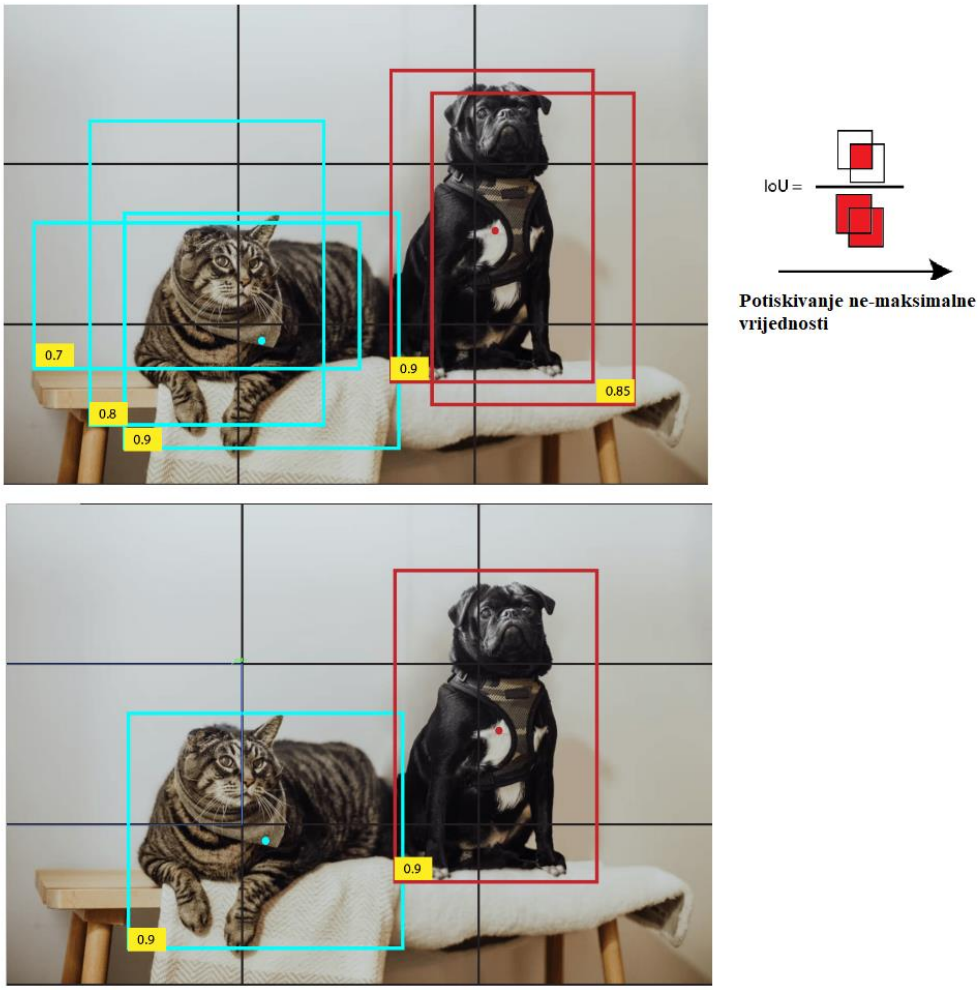
gdje vektor $C_{3,2}$ označava da se radi o ćeliji koja se nalazi u trećem retku i drugom stupcu. Prvi element vektora je vjerojatnost klase objekta (P_c) i on je jednaka jedinici. Koordinate središta graničnog pravokutnika relativno u odnosu na dimenzije ćelije iznose 0.05 i 0.04, gdje je vrijednost *x* koordianate 0.05, a *y* koordinate 0.04. Širina graničnog pravokutnika iznosi 0.35, a visina 0.4 te se promatra u odnosu na cijelu sliku. Klasa koje je predstavljena unutar graničnog pravokutnika je „mačka“ te se sukladno tome označava C_1 s jedinicom, a C_2 s nulom.

Umjesto predviđanja samo dviju klasa, moguće je definirati onoliko klasa koliko je potrebno. Naravno, to utječe na točnost algoritma. Općenito, svaka ćelija se opisuje pomoću vektora s dimenzijom $B \times 5 + C$ gdje je B broj graničnih pravokutnika, a C je broj klasa. Također se može prikazati pomoću $S \times S \times (B \times 5 + C)$ tenzora. Nakon ovog postupka, ako je definiran jedan vektor za svaku ćeliju rešetke, cijela slika 2.5. je predstavljena s devet vektora veličine 7 ili 3 x 3 x 7 tenzora. Broj sedam je rezultat množenja broja graničnih pravokutnika po ćeliji (u ovom primjeru B je jedan) i broja pet te se na rezultat množenja dodaje broj klasa, a u primjeru je C jednak dva. To znači da je u skupu podataka svaki uzorak slike označen jednim tenzorom 3x3x7. Koristeći CNN, YOLO je u stanju predvidjeti sve objekte u jednom prolazu prema naprijed (engl. *forward pass)* i to je razlog njegovog punog imena „*You Only Look Once*“.

Jedan problem koji se može dogoditi je kada algoritam predviđa nekoliko graničnih pravokutnika za jednu klasu. Može odabrati samo jedan pravokutnik po klasi koji ima najveću vjerojatnost, ali na slici može biti više objekata jedne klase. Zbog toga se koristiti algoritam potiskivanja ne-maksimalne vrijednosti (engl. *non-maximal supression*). Prvo se uzima pravokutnik s maksimalnom vjerojatnošću. Nakon toga se uspoređuje pravokutnik sa svim ostalim pravokutnicima te klase pomoću IoU. Općenito, ova metrika je poznata i kao *Jaccardov indeks,* ali u računalnom vidu koristi se naziv IoU. Formula IoU je (2-3):

$$IoU = \frac{\text{prostor presjeka između } B_1 \text{ i } B_2}{\text{prostor unije između } B_1 \text{ i } B_2}, \tag{2-3}$$

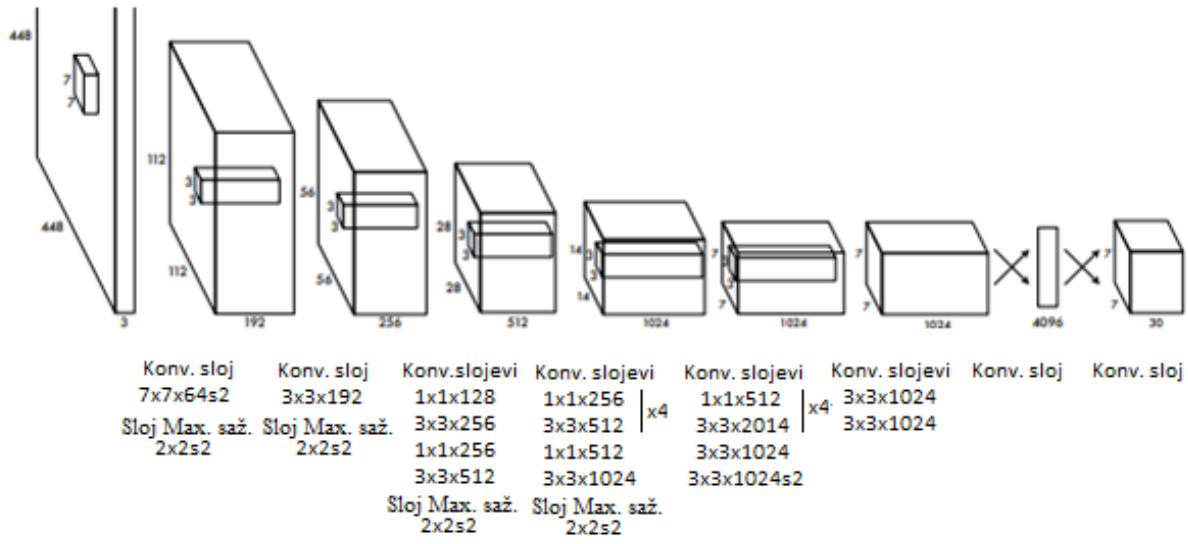
gdje su B_1 i B_2 dva granična pravokutnika. Ako je IoU viši od unaprijed definiranog praga (na primjer 0,5), tada se pravokutnik s manjom vjerojatnošću potiskuje ili isključuje. To znači da dva pravokutnika s visokom IoU vrijednošću vjerojatno označavaju isti objekt na slici, pa se pravokutnik s manjom vjerojatnošću isključuje. Taj se postupak ponavlja sve dok se svi pravokutnici ne uzmu kao predviđanje objekta ili se ne isključe (slika 2.6.). Brojevi na slici koji su označeni žutom bojom (0.7, 0.8, 0.9, 0.9, 0.85) predstavljaju vjerojatnosti da se unutar graničnog pravokutnika nalazi detektirani objekt.



Slika 2.6. Primjer potiskivanja ne-maksimalne vrijednosti [14]

YOLO algoritam koristi CNN čiji je princip rada objašnjena u potpoglavlju 2.1.1. Arhitektura YOLO-a inspirirana je GoogLeNet arhitekturom [8]. Originalni YOLO je implementiran i testiran na VOC 2007 i 2012 bazama podataka. Originalna YOLO mreža za detektiranje ima 24 konvolucijska sloja, nakon čega slijede dva potpuno povezana sloja, a prikazana je na slici 2.7. Od ta 24 konvolucijska sloja, samo nakon četiriju konvolucijskih slojeva slijede udruživanje po maksimalnoj vrijednosti [8].

Prilična pogreška lokalizacije i manji odziv u usporedbi s dvo-stupanjskim detektorima objekata mogu se smatrati dvama značajnim nedostacima prve verzije YOLO algoritma. Autori YOLO algoritma [8] preoblikovali su problem detekcije objekata u regresijski problem umjesto problema klasifikacije. CNN predviđa granične okvire, kao i vjerojatnost klase za sve objekte prikazane na slici. Ovaj algoritam identificira objekte i njihovo pozicioniranje uz pomoć graničnih okvira gledajući sliku samo jednom.



Slika 2.7. Arhitektura CNN osnovnog YOLO algoritma [8][15]

Danas najnovija verzija YOLO-a (YOLOv7) [14], s druge strane, uspijeva točnije otkriti manje objekte zbog nekoliko dodatnih klasifikacijskih koraka uvedenih u mrežu. YOLOv1 je objavljen 2016. godine, no u YOLOv2 napravljeno je nekoliko poboljšanja u odnosu na YOLOv1. Koristio je pristup predviđenih graničnih pravokutnika (engl. *anchor box*) za poboljšanje točnosti detektiranja i uveo sloj za povećanje uzorkovanja (engl. *upsample layer*), što je poboljšalo razlučivost izlazne mape značajki [14].

YOLOv3 objavljen je 2018. godine [16] i također je poboljšao pristup predviđenih graničnih pravokutnika, omogućujući različitim skalama i omjerima slike da bolje odgovaraju veličini i obliku detektiranih objekata. Obilježja YOLOv3 su korištenje FPN-a i funkcije gubitka GHM-a (engl. *Gradient Harmonizing Mechanism*). Bitna obilježja su i korištenje objekata različitih veličina i omjera slika te dolazi do poboljšanja točnosti i stabilnosti. Najistaknutija značajka treće verzije je da detekcije otkriva na tri različite ljestvice. Jedini nedostatak toga je blago smanjenje brzine obrade [16].

YOLOv4, objavljen 2020. godine, uveo je niz poboljšanja u odnosu na YOLOv3, uključujući novu arhitekturu mreže, poboljšanje procesa treniranja i povećani kapacitet modela. YOLOv4 je također predstavio CBN (engl. *Cross-Iteration Batch Normalization*), novu metodu normalizacije dizajniranu za povećanje stabilnosti procesa treniranja [17].

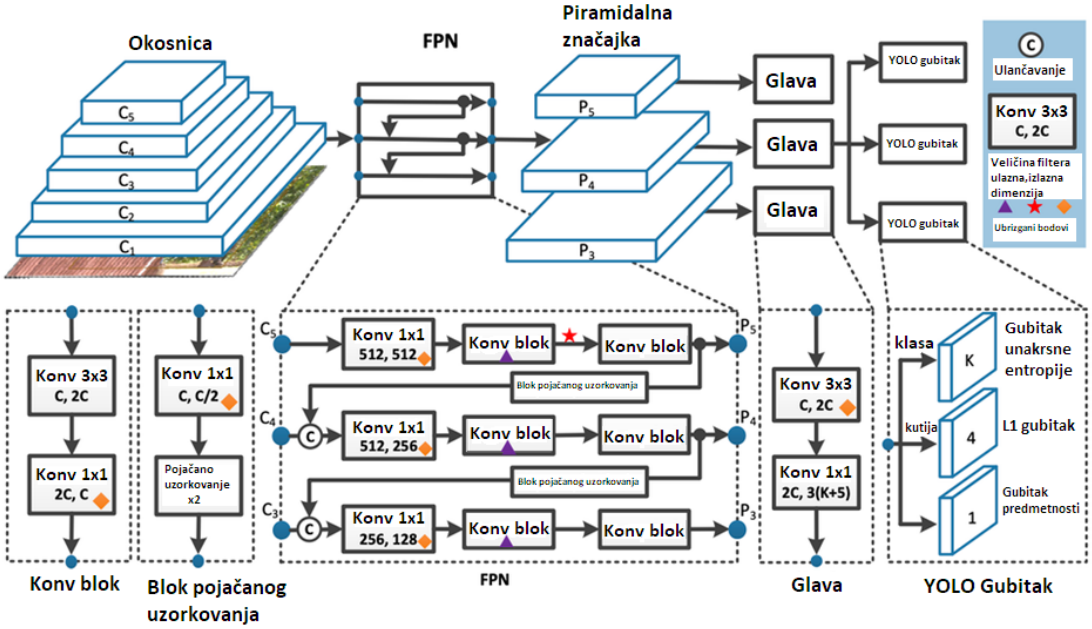
YOLOv5 je također predstavljen 2020. godine, nadovezuje se na uspjeh prethodnih verzija, a tvrtka *Ultralytics* ga je objavila kao projekt otvorenog koda. YOLOv5 je koristio arhitekturu

EfficientDet, zasnovanu na *EfficientNet* mreži i poboljšanu s nekoliko novih značajki, kako bi postigao poboljšane performanse detektiranja objekata [18].

YOLOv6 je objavljen 2022. godine [19]. Usredotočio se na povećanje učinkovitosti sustava i smanjenje njegovog memorijskog otiska. Koristi novu CNN arhitekturu pod nazivom *SPP-Net* (engl. *Spatial Pyramid Pooling Network*). Ova arhitektura dizajnirana je za rukovanje objektima različitih veličina i omjera slike, što je čini idealnom za zadatke detektiranja objekata [19].

YOLOv7 je također predstavljen 2022. godine te je u ovom diplomskom radu fokus na analizi YOLOv7 modela. Jedno od ključnih poboljšanja u YOLOv7 je korištenje nove CNN arhitekture pod nazivom *ResNeXt*. YOLOv7 također uvodi novu strategiju treniranja na više skala (engl. *multi-scale training)*, koja uključuje treniranje modela na slikama na više različitih skala te kombiniranje predviđanja. To pomaže modelu da učinkovitije obradi objekte različitih veličina i oblika. Konačno, YOLOv7 uključuje novu tehniku pod nazivom „*Focal Loss*“, odnosno žarišnog gubitka, koja je dizajnirana za rješavanje problema neravnoteže klase koji se često javlja u zadacima detektiranja objekata. Funkcija žarišnog gubitka daje veću težinu teškim primjerima i smanjuje utjecaj lakih primjera [20].

Općenito, YOLOv7 nudi bržu i robusniju mrežnu arhitekturu koja nudi bolji pristup integraciji značajki, preciznije performanse detektiranja objekata, robusniju funkciju gubitka i poboljšano označavanje objekata i učinkovitost treniranja modela. Ovdje je objašnjeno koje su se veće promjene u arhitekturi YOLOv7 modela dogodile u odnosu na prethodne arhitekture YOLO verzija (slika 2.8.).



Slika 2.8. Arhitektura YOLOv7 mreže [21]

Neke od njih vezane za arhitekturu modela su E-ELAN (engl. *Extended - Efficient Layer Aggregation Network*) i skaliranje veličine modela (engl. *Model Scaling*) za modele zasnovane na ulančavanju. BoF (*Bag of Freebies* je pojam koji predstavlja niz strategija koje povećavaju performanse mreže. Koriste se za postizanje boljih rezultata primijenjenih u treniranju mreže bez povećanja „troškova" treniranja. YOLO arhitektura sastoji se od okosnice (engl. *backbone*) (tj. kralježnice algoritma), vrata (engl. *neck*) i glave (engl. *head*) algoritma. Glava sadrži predviđene izlaze, a YOLOv7 se ne ograničava samo na jednu glavu već ima više glava [20]. Omjer ulazno/izlaznih kanala i rad s elementima utječu na brzinu mreže prema modelima *VovNet* i *CSPVNet*. Konvolucijska neuronska mreža ima za cilj učiniti gustu konvolucijsku mrežu (*DenseNet)* učinkovitijom integriranjem svih značajki samo jednom u posljednju mapu značajki. E-ELAN, koji je YOLOv7 nazvao nakon proširenja ELAN (engl. *Efficient Layer Aggregation Networks*). Glavna prednost ELAN-a je sposobnost boljeg učenja i dublje mreže upravljanjem gradijentnim putem. Cilj ove metode je koristiti grupnu konvoluciju za proširenje kanala. To se postiže primjenom istih grupnih parametara i množitelja kanala na svaki računalni blok u sloju. Blok zatim izračunava mapu značajki i preuređuje u određeni broj grupa, kako je određeno varijablom te se kombinira. Na taj je način količina kanala u svakoj grupi mapa značajki jednaka broju kanala u izvornoj arhitekturi. Samo promjenom arhitekture modela u računalnom bloku, prijelazni sloj ostaje nepromijenjen i gradijentni put je fiksiran. Uobičajeno je da YOLO i drugi modeli detekcije objekata izdaju niz modela koji se skaliraju u veličini, a koji će se koristiti u različitim slučajevima uporabe. Za skaliranje, modeli detekcije objekata moraju znati dubinu mreže, širinu mreže i razlučivost na kojoj je mreža trenirana. YOLOv7 model istovremeno skalira dubinu i širinu mreže dok spaja slojeve. U radu [20] pokazuju da ova tehnika održava arhitekturu modela optimalnom prilikom skaliranja za različite veličine. Nakon treniranja, re-parametrizacija je metoda koja se koristi za poboljšanje modela. Iako je trajanje treninga produljeno, ishodi krajnjeg modela su poboljšani. Ta su poboljšanja dovela do značajnog povećanja sposobnosti. Glavna glava (engl. *lead head*) u YOLOv7 naziva se konačna izlazna glava. A pomoćna glava (engl. *auxiliary head*) je glava koja pomaže u treniranju srednjih slojeva mreže. Dodjeljivanje oznaka (engl. *label assigner*) je metoda koja dodjeljuje meke oznake (engl. *soft labels*) nakon uzimanja u obzir oznake stvarne istine i ishoda predviđanja mreže. Te meke oznake se generiraju na temelju konačnih predviđanja koja proizvodi glava za konačni izlaz. Bitno je da se iste meke oznake koje su generirane koriste za izračunavanje gubitka (engl. *loss*) kako za glavu za konačni izlaz, tako i za pomoćnu glavu [20]. To osigurava da se obje glave treniraju koristeći dosljedne informacije i doprinose cjelokupnom procesu učenja.

2.1.5. Metrike koje se koriste za evaluaciju YOLO algoritma

Metrika koje se vrlo često koristi kod detekcije objekata je *mAP* (engl. *Mean Average Precision*). Izračunava ukupnu prosječnu preciznost za različite vrste objekata. Uspoređuje predviđene granične pravokutnike s pravokutnicima temeljne istine te se računa na temelju preciznosti (engl. *precision*) i odziva (engl. *recall*). Prosječna preciznost se izračunava pomoću krivulje preciznosti u odnosu na odziv, a *mAP* je srednja vrijednost AP (engl. *Average Precision*) vrijednosti za sve klase. Odziv mjeri koliko točno detektiranih objekata postoji u odnosu na broj svih objekata koji su trebali biti detektirani (2-5), dok preciznost mjeri koliko točno detektiranih objekata ima u odnosu na broj svih detektiranih objekata (onih koje je trebalo detektirati i onih koji su pogrešno detektirani kao primjeri neke klase) (2-4). Točnost sustava mjeri se kao omjer broja detektiranih objekata odnosno zbroj TP (engl. *true positive*) i TN (engl. *true negative)* u odnosu na broj svih detekcija (2-6). Za procjenu kompromisa između učinkovitog detektiranja objekata i smanjenja lažno pozitivnih rezultata, preciznost i odziv često se kombiniraju. Harmonijska sredina odziva i preciznosti je F1 rezultat (2-7). Daje vrijednost koja kombinira i preciznost i odziv u jednu mjeru, dajući istu težinu objema mjerama. Navedeni mjerni podaci koriste se za procjenu uspješnosti YOLO algoritma. Oni nude percepciju preciznosti i opće izvrsnosti ishoda detektiranja objekta:

$$preciznost = \frac{TP}{TP+FP}, \tag{2-4}$$

$$odziv = \frac{TP}{TP+FN}, \tag{2-5}$$

$$točnost = \frac{TP+TN}{TP+FP+FN+TN}, \tag{2-6}$$

$$F1 = 2 * \frac{preciznost*odziv}{preciznost+odziv}, \tag{2-7}$$

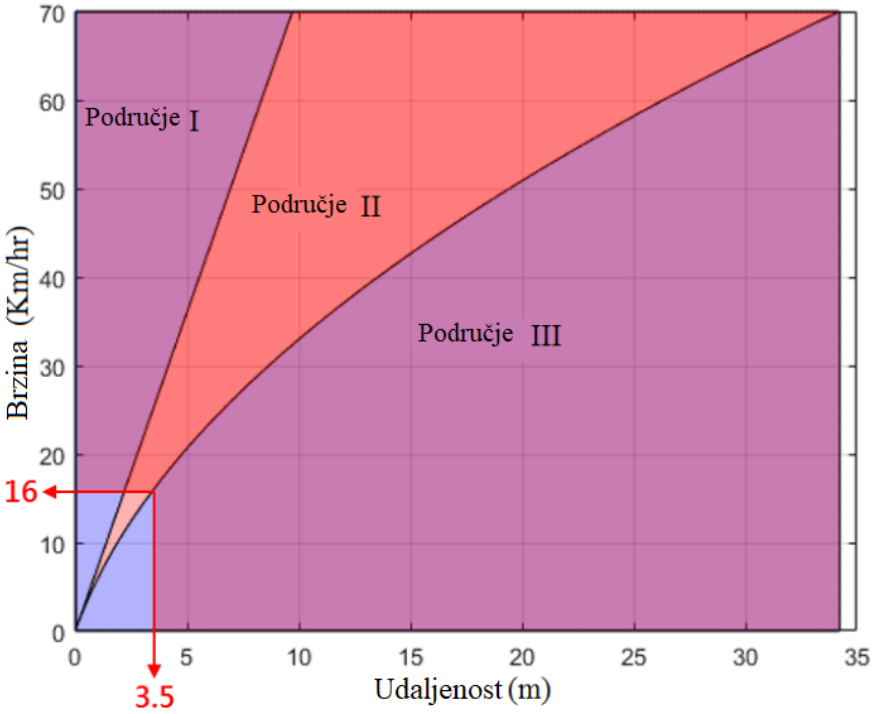
gdje je:

- TP – *true positive,* detekcije koje su proglašene objektom, a one to jesu,
- FP – *false positive,* detekcije koje su proglašene objektom, a one to nisu,
- FN – *false negative,* detekcije koje nisu proglašene objektom, a one to jesu,
- TN – *true negative,* detekcije koje nisu proglašene objektom, a one to nisu [15].

2.2. Postojeća rješenja koja omogućuju uključivanje alarmnog sustava prilikom izlaska iz vozila.

Rješenja za uključivanje alarmnog sustava prilikom izlaska iz vozila proučavaju se već duže vrijeme. Postoji mnogo radova koji različitim pristupom i metodama detektiraju sudionike prometa i na temelju procijenjene udaljenosti i brzine primjenjuju se za različite svrhe. Na početku je važno temeljito istražiti i razumjeti različite probleme koji mogu utjecati na točnost procjene brzine i uključivanja alarmnog sustava prilikom izlaska vozača iz vozila kako bi se smanjile pogreške i poboljšala učinkovitost rješenja. Postojeća rješenje za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila različito su koncipirana, odnosno koriste različite metode za procjenu udaljenosti i brzine kao i različite načine implementacije na različite ugradbene računalne platforme. U nastavku će biti opisano nekoliko postojećih rješenja.

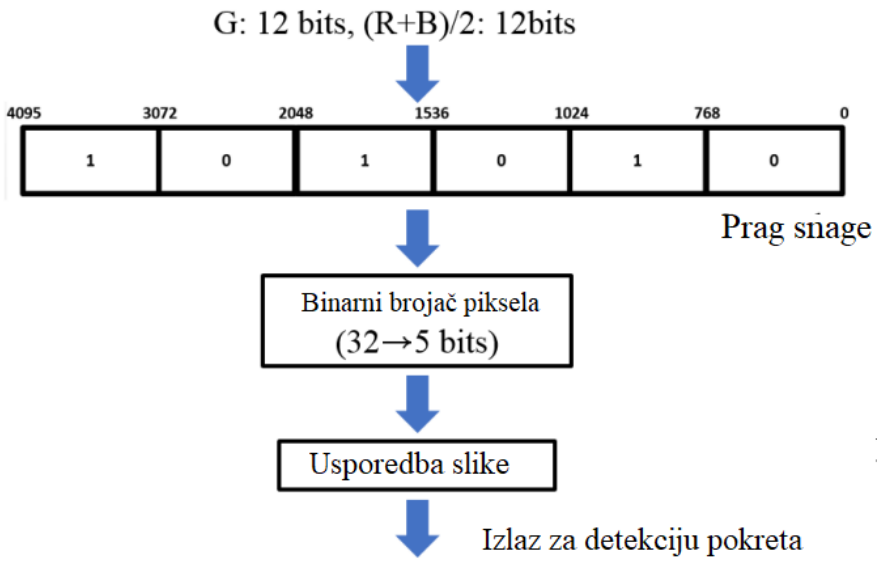
U prvom promatranom postojećem rješenju [22] predstavljen je sustav protiv sudara pomoću jednog digitalnog fotoaparata i programabilnog logičkog sklopa (FPGA engl. *Field programmable gate array*). U radu su definirali prag upozorenja odnosno odredili su koja područja u slici su opasna, a koja sigurna te su prikazana na slici 2.9.



Slika 2.9. Područje upozorenja s različitim brzinama objekta [22]

Prikazali su opasna i sigurna područja s različitim pristupnim brzinama objekta. Područje koje su označili kao „I“ predstavlja sigurnu brzinu u odnosu na udaljenost objekta od kamere. Ovo područje uzima u obzir kašnjenje od 0,5 sekundi kako bi vozač otvorio vrata vozila dok objekt ima

dovoljnu brzinu da sigurno prođe vozilom. Područje „III“ prikazuje sigurnu udaljenost kada je moguće početi s kočenjem u odnosu na brzinu objekta. Stoga je područje „II“ opasno za vozača koji ne može sigurno i na vrijeme zakočiti ili proći pored parkiranog vozila. Postavili su prag upozorenja na 16 km/h za otkrivanje objekta udaljenog od 3,8 do 18,8 metara od kamere. Predloženi sustav prvo detekcijom pokreta detektira približavanje objekata, a zatim izračunava njihovu brzinu kako bi se alarm aktivirao. Procjena udaljenosti s 2D slikom konvencionalno je modelirana trigonometrijskim funkcijama. Međutim u radu su naveli da takva metodologija nameće teške računalne zahtjeve na niskobudžetnom mikroprocesoru. Također je neprikladna i za hardver zato što trigonometrijske funkcije s operacijama pomičnog zareza dramatično povećavaju potrošnju hardverskih resursa. Zato su koristiti *T-S fuzzy* (engl. *Takagi-Sugeno fuzzy*) rezoniranje za procjenu udaljenosti od kamere. Postupak detekcije pokreta usvojen je iz *One-Page-Comparison-a* (OPC) kako bi se izbjegla upotreba vanjske memorije. Svaki ulazni piksel slike najprije prelazi prag snage (engl. *strength threshold*) (slika 2.10.).

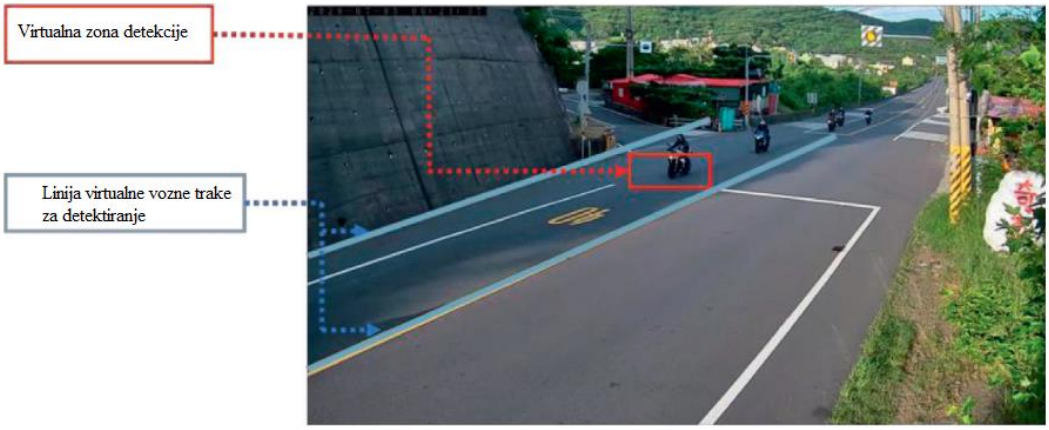


Slika 2.10. OPC algoritam [22]

S jačinom boje (12 bita) koja opada u određenim područjima, boja piksela je označena kao binarna "1", a snaga koja pripada ostalim pragovima označena je kao "0". Skeniranjem kroz piksele OPC broji koliki je broj "1" na svaka 32 bita i pohranjuje binarnu sliku s 5 bitova u memoriju odnosno to je predstavljeno binarnim brojačem piksela na slici 2.10. [22]. Za usporedbu slika, nad dolaznim 5-bitnim podacima i podacima pohranjenim u memoriji se izračunava *bitwise* operacije „*I*“ (engl. *AND*). OPC potvrđuje da se objekt kreće ako je je rezultat "1" tako što se izlazni četvrti ili treći bit izračuna s bitom jedan operacijom *AND*, to se odnosi za detektiranje velike udaljenosti, a "1" na bitu četiri ili tri se odnosi na kratke udaljenosti. Za ostale bitove u 5-bitnim podacima OPC

potvrđuje da se objekt kreće ako je njihov *AND* jednak rezultatu "1". Rezultati su testirani s biciklistom te pokazuju da je predloženi sustav praktičan za dizajn u automobilima s ugrađenom detekcijom potencijalnih sudara s biciklistom.

Drugo proučavano rješenje se sastoji od četiri koraka [23]:(1) prikupljanje videozapisa o prometu s internetskih kamera, (2) izvršavanje brojanja vozila pomoću GMM-a (engl. *Gaussian mixture model*) i virtualne zone detekcije, (3) izvršavanje klasifikacije vozila i (4) procjena brzine pomoću YOLOv3 algoritama. U radu videozapisi o prometu prikupljeni su s internetskih kamera. Baza podataka slika izdvojena je iz prometnih videozapisa pomoću skripte, a označavanje je izvedeno pomoću softverske aplikacije otvorenog koda pod nazivom „*labeling*“. Za realizaciju brojanja vozila u stvarnom vremenu provodi se detekcija i prepoznavanje objekata. Linija virtualne vozne trake za detektiranje i virtualna zona detekcije upotrebljavaju se za brojanje vozila i procjenu brzine (slika 2.11.).



Slika 2.11. Prikaz virtualne zone i linije virtualne vozne trake za detekciju objekta [23]

U ovom rješenju koristili su YOLOv3 algoritam u okviru „*Darkneta“* za klasificiranje vozila u šest klasa. Brzina vozila u stvarnom vremenu također se izračunava u ovom radu. Za izračunavanje paralelnog segmenta linije između točaka A i B, algoritam izračunava udaljenost između njih. Zatim se izračunava stvarna udaljenost iz njezinog odnosa s duljinom segmenta linije *x*, gdje *x* označava prijeđenu udaljenost vozila u virtualnom prostoru. U procesu procjene brzine vozila program koristi broj okvira u sekundi kada vozilo putuje u prostoru linija virtualne vozne trake (*p*) prikazane na slici 2.11. i broj okvira u sekundi u cijelom videozapisu (*fps*). Koristili su jednadžbu (2-8) za pronalaženje vremena putovanja vozila od točke A do točke B u virtualnom prostoru te jednadžbu za određivanje brzine vozila u *m/s*.

$$t = \frac{p}{fps}, \quad v = \frac{x}{t}. \tag{2-8}$$

Za treće proučavano rješenje [24] izračun brzine te praćenje vozila provode se pomoću *Alexey AB*-ovog (engl. *Alexey Bochkovskiy*) *API*-ja (engl. *Application Programming Interface*) odnosno *Darknet*-a za detekciju objekta i modificirane verzije njegovog primjera napisanog u C++ programskom jeziku. Praćenje se koristi za identifikaciju svakog vozila prisutnog u videozapisu i povezivanje liste lokacija graničnih pravokutnika s njim. Praćenje detektiranog objekta provodi se pomoću biblioteke *OpenCV*. Nakon što *Kalmanov* filter utvrdi koji granični pravokutnici odgovaraju određenom vozilu, može se izračunati prosječna brzina svakog vozila. Kamera je postavljena iznad ceste i pokriva samo relativno kratku dionicu ceste. Funkcija ovog algoritma je praćenje uvjeta u prometu, a ne pojedinačnih vozila. Pretpostavili su da se sva vozila kreću ravnom linijom i konstantnom brzinom. To omogućuje jednostavno izračunavanje brzine bez potrebe za ravninom *z*. Da bi se izračunala brzina, predviđeni granični pravokutnici moraju se pretvoriti u položaj na ravnini $z = 0$. To se radi tako da se parametri kamere učitavaju iz datoteke generirane pomoću alata za kalibraciju kamere. Zatim se za svako detektirano vozilo odabire kut graničnog pravokutnika najbližeg ravnini tla. Kutne koordinate (x, y) u ravnini slike zatim se pretvaraju u skup odgovarajućih koordinata (X, Y) na ravnini $z = 0$ na sceni stvarnog svijeta koristeći transformacije. Bilježi se vrijeme otkrivanja novog vozila, njegov ID praćenja, početne koordinate (X, Y) i trenutni broj okvira f_q. Svaki put kada se detektira isto vozilo, izračunava se njegova udaljenost d od (X, Y) koordinata i protekli broj okvira t_f od f_q. Koristeći ih, prosječna brzina v vozila od trenutka kada je detektirano je (2-9):

$$v = \frac{d}{t_f \cdot \frac{1}{FPS}} = d \cdot \frac{FPS}{t_f}, \tag{2-9}$$

gdje je *FPS* broj okvira u sekundi video sekvence. Model procjene brzine postigao je vrlo visoku točnost za zadatak detekcije objekta s *mAP*-om od 98%. Sustav je sposoban za rad u stvarnom vremenu, s tim da model može pratiti videozapis od 30 *FPS*, a procijenjene brzine su u području očekivanih brzina za cestu i tip vozila. U radu su spomenuli da, kako bi se poboljšalo procjenjivanje brzine, treba koristiti algoritam *Deep SORT*, a ne *Kalmanov* filter. Također, uključivanjem značajki objekta, detektirani objekti pratili bi se mnogo uspješnije.

3. VLASTITO RJEŠENJE ZA UKLJUČIVANJE ALARMNOG SUSTAVA PRILIKOM IZLASKA VOZAČA IZ VOZILA

Zadatak, a ujedno i cilj ovog rada je kreirati i implementirati rješenje za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila. Potrebno je detektirati sudionike prometa koji dolaze sa stražnje strane vozila kada vozač napušta parkirano vozilo i u skladu s vremenom preostalim do potencijalne kolizije aktivirati odgovarajući alarmni sustav. Detekcija se izvršava obradom okvira video sekvenci snimljenih kamerom koja je postavljena u bočnom retrovizoru vozila i okrenuta na način da snima prema nazad u odnosu na orijentaciju vozila. Osim detekcije, mehanizam za aktiviranje alarmnog sustava, koji može uključivati zvučni signal, vizualno upozorenje ili neki oblik tekstualne obavijesti, mora biti dizajniran i implementiran kako bi upozorio na opasnost izlaska vozača iz vozila.

Vlastito rješenje je izgrađeno na bazi YOLOv7 algoritma koji se sastoji od nekoliko koraka. Učitava okvir video sekvenci i prosljeđuju potrebne informacije kroz algoritam za detektiranje kako bi se naposljetku prikazali detektirani sudionici prometa koji će u sljedećim okvirima biti praćeni te se procjenjuje ujedno i njihova udaljenost, brzina te vrijeme preostalo do potencijalne kolizije s vratima vozila. Alarmni sustav koji treba obavijestiti vozača o potencijalnoj koliziji s vratima vozila napravljen je u obliku tekstualne poruke, a aktivira se ako je vrijeme preostalo do potencijalne kolizije manje od tri sekunde. Rješenje je izrađeno u *Python* programskom jeziku koristeći YOLOv7 algoritam. U potpoglavlju 3.1. detaljno su opisani korišteni alati i tehnologije za izradu rješenja, odnosno potrebne biblioteke za njegovo korištenje te *Deep Sort* algoritam za praćenje detektiranih objekata. Koraci uključeni u postupak kreiranja baze podataka slika objašnjeni su u potpoglavlju 3.2 i naposljetku proces treniranja mreže temeljito je objašnjen u potpoglavlju 3.3. Glavni dio rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila zasnovanog na YOLOv7 algoritmu detaljno je opisan u potpoglavlju 3.4. U potpoglavlju 3.4 objašnjeno je kako odrediti udaljenost sudionika u prometu od mjesta kamere, brzinu i vrijeme koje će mu trebati da stigne do mjesta kamere odnosno do vozačevih vrata parkiranog vozila. Upute za pokretanje rješenja dane su u potpoglavlju 3.5. odnosno koje verzije korištenih biblioteka je potrebno instalirati za pokretanje rješenja i koje naredbe upisati za pokretanje. U potpoglavlju 3.6. opisan je postupak implementacije i pokretanje rješenja na ugradbenoj računalnoj platformi Raspberry Pi 4.

Cilj rješenja je detektirati sudionike prometa koji pripadaju jednoj od sljedećih klasa: automobil, autobus ili kamion, pješak, biciklist i motociklist te na temelju njihove udaljenosti i brzine procijeniti vrijeme preostalo do potencijalne kolizije s vratima vozila (slika 3.1.).

(a) (b) (c)

(d) (e)

Slika 3.1. Primjeri slika objekata iz baze koje je potrebno detektirati i klasificirati u sklopu diplomskog rada: (a) automobil, (b) autobus i kamion, (c) pješak, (d) biciklist, (e) motociklist

3.1. Opis korištenih alata i tehnologija za izradu rješenja

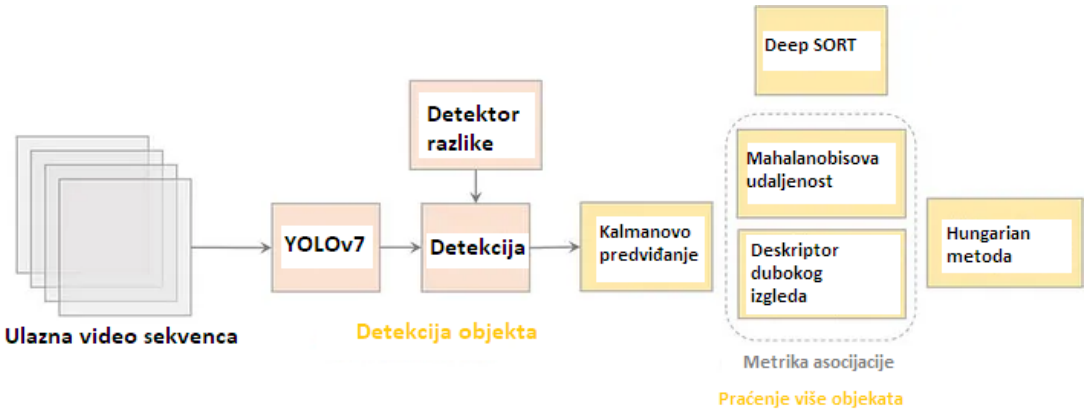
Vlastito rješenje implementirano je pomoću YOLOv7 algoritma za detekciju i *Deep SORT* algoritma za praćenje detektiranog objekta. Napisano je u programskom jeziku *Python* (verzija 3.9.16) uz najvažnije biblioteke *OpenCv* (verzija 4.6.0) i *NumPy* (verzija 1.23.5). Treniranje i implementacija rješenja izgrađena je u *Anaconda Navigator*-u, desktop aplikacija koja omogućuje upravljanje paketima i okruženjima iz grafičkog korisničkog sučelja (engl. *graphical user interface,* GUI). Sučelje naredbenog retka (engl. *command line interface,* CLI) program je na računalu koji obrađuje tekstualne naredbe za obavljanje različitih zadataka [25]. *Conda* je CLI program, što znači da se može koristiti samo putem naredbenog retka. Rješenje je implementirano na *Windows* računalu pa se s *Anaconda Navigator-om* koristio CLI *Anaconda Prompt.*

Algoritam za detekciju objekata u stvarnom vremenu za zadatke računalnog vida s najvećom točnošću i brzinom je YOLOv7 te je iz tog razloga korišten za detektiranje sudionika prometa. Detalji o YOLOv7 opisani su u dijelu 2.1.4. YOLOv7 i općenito YOLO zasnovan na CNN-u zahtijeva veliku količinu slika za treniranje. Slike moraju imati mnogo izgleda objekta u sebi kako bi bile dovoljno robusne (u ovom slučaju varijabilnost u vrstama navedenih klasa sudionika prometa, uvjetima osvjetljenja, kutovima gledanja itd.). Stoga je kod treniranja modela korišteno preneseno učenje (engl. *transfer learning*) gdje je korišten unaprijed istrenirani model, a treniranje novog modela počinje od unaprijed istreniranog modela, točnije korišteni su YOLOv7 i YOLOv7 *Tiny PyTorch* modeli. *PyTorch* je okruženje za strojno učenje otvorenog koda [26]. U ugradbenoj računalnoj platformi Raspberry Pi 4, računalni resursi su ograničeni. Stoga je korišten YOLOv7 *Tiny* koji zauzima manje memorije te je potrebno manje parametara. YOLOv7 *Tiny* u usporedbi s YOLOv7 ima brže vrijeme obrade odnosno može obrađivati veći broj FPS-a, ali *mAP* je manji. Ideja je da prvih nekoliko slojeva već nauči koje značajke izdvojiti odnosno što tražiti na slikama. Samo treba naučiti posljednje ili posljednjih nekoliko slojeva koji će raditi za određeni slučaj upotrebe. PT datoteka je model strojnog učenja stvoren pomoću *PyTorch*-a, biblioteke strojnog učenja otvorenog koda koju je razvila *Meta*. Sadrži algoritme koji se koriste za automatsko izvršavanje zadatka, kao što su obrada prirodnog jezika ili identifikacija objekata [26]. PT datoteka kao rezultat treniranja korištena je u vlastitom rješenju.

Praćenje objekata ima nekoliko prednosti u odnosu na tradicionalne zadatke računalnog vida kao što su detektiranje i segmentacija objekata. Neke od prednosti praćenja objekata uključuju: kontinuitet, brzinu, efikasnost i preciznost. Praćenje objekata osigurava kontinuitet u detektiranju i praćenju objekata tijekom vremena, što je važno za primjene kao što su nadzor i autonomna vožnja. Algoritmi za praćenje objekata mogu obrađivati videozapise u stvarnom vremenu, što ih čini prikladnim za aplikacije koje zahtijevaju obradu u stvarnom vremenu. Mogu istovremeno pratiti više objekata, što ih čini efikasnim. Mogu precizno pratiti objekte u izazovnim okruženjima kao što su pretrpane pozadine. Praćenje objekata postupak je lokalizacije prethodno detektiranog objekta u sljedećim okvirima. Detektor objekata otkriva različite objekte, njihova mjesta odnosno označava granični pravokutnik i vrstu klase iz jednog okvira videozapisa. Rješenje u sklopu ovog diplomskog rada se sastoji od algoritma za detekciju, praćenje i procjenu udaljenosti. Pri pokretanju rješenja učitan je prvi video okvir i vrši se detekcija objekta u prvom video okviru te klasificira objekt u jednu od klasa sudionika prometa. Detektirani objekt je dobio jedinstvenu oznaku te predstavlja referentni, odnosno stari objekt. Zatim se učita sljedeći video okvir i opet je odrađena detekcija objekta pomoću algoritma za detekciju. Taj detektirani objekt predstavlja novi

objekt. Također je napravljena klasifikacija objekta. Izračunati su odgovarajući rezultati (engl. *matching score*) između svakog starog objekta i novog objekta i na temelju rezultata se pronalazi najbolje podudaranje novog objekta sa starim objektom. Ako postoji više starih detektiranih objekata odnosno više objekata u video okviru i nisu se pronašli na temelju rezultata podudaranja s novim objektom, oni se smatraju propuštenim objektima, a ne podudarani novi objekti se smatraju novim objektima u sceni. Propušteni objekti su uklonjeni ako nedostaju u određenom graničnom razdoblju (engl. *threshold period*). Novim detektiranim objektima se dodjeljuje jedinstvena oznaka. Novi i praćeni objekti se spremaju za sljedeću iteraciju. Pohranjeni su rezultati praćenih objekta s njihovim jedinstvenim oznakama.

Za praćenje objekta koristi se *DeepSORT* algoritam koji je proširenje *SORT* algoritma (engl. *Simple Online Realtime Tracking*) te je dijagram toka praćenja objekta pomoću *DeepSORT* algoritama prikazan na slici 3.2 [27].



Slika 3.2. *DeepSORT* algoritam [27]

DeepSORT uvodi duboko učenje u *SORT* algoritam dodavanjem deskriptora izgleda kako bi se smanjili parametri identiteta, što praćenje čini učinkovitijim. *SORT* je pristup praćenja objekata gdje se *Kalmanovi* filtri i *Hungarian* algoritam koriste za praćenje objekata [28]. *SORT* se sastoji od tri koraka. Prvi korak je procjena gdje se detekcije prenose iz trenutnog okvira u sljedeći okvir kako bi se procijenio položaj objekta u sljedećem okviru pomoću *Gaussove* distribucije i modela konstantne brzine. U drugom koraku matrica se izračunava kao predviđena vjerojatnost (*IoU*) udaljenosti između svake detekcije i svih predviđenih graničnih pravokutnika od postojećih ciljanih objekta. Posljednji korak je kada objekt uđe ili napusti okvir u video sekvenci. Tada se stvara jedinstveni *ID* objekta i uništava u skladu s tim [28]. *SORT* algoritam vrlo dobro funkcionira u smislu preciznosti i točnosti praćenja. Problemi u algoritmu *SORT* su nedostatak u praćenju odnosno neuspijevanju detektiranja objekta u slučaju kada se objekt snimljen iz određenih kutova gledanja. Unatoč učinkovitosti *Kalman* filtra, relativno često dovodi do promjene ID oznaka. Ti

su problemi posljedica korištene metrike pridruživanja. U *DeepSORT* algoritam se uvodi još jedna metrika udaljenosti na temelju izgleda objekta, a to je vektor značajke izgleda (engl. *Deep Appearance Descriptor*). *DeepSORT* koristi bolje mjerne podatke o pridruživanju koji kombinira deskriptore pokreta i izgleda. *DeepSORT* se može definirati kao algoritam praćenja koji prati objekt ne samo na temelju brzine i gibanja objekta, već i na temelju izgleda objekta.

Za treniranje YOLOv7 potrebno je instalirati određene biblioteke prikazane na slici 3.3. i biblioteke za podršku grafičkog procesora (engl. *graphics processing unit* GPU) prikazane na slici 3.4. Sve biblioteke koje se nalaze u *requirements.txt* datoteci instalirane su u kreiranom virtualnom okruženju (engl. *virutal environment*) u *anaconda* sučelju s verzijom *Python-a* 3.9 pomoću naredbe *pip install -r requirements.txt.* Biblioteke koje su dio *PyTorch* projekta instalirane su pomoću naredbe *pip install -r requirements_gpu.txt*.

U kreiranom virtualnom okruženju u *anaconda* sučelju instalirane su sve biblioteke koje se nalaze u *requirements.txt* datoteci naredbom: *pip install -r requirements.txt*. S naredbom *pip install -r requirements_gpu.txt* instalirane su biblioteke koje su dio *PyTorch* projekta.

```
# Base ----------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.1
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
tqdm>=4.41.0
protobuf<4.21.3

# Logging -------------------------
tensorboard>=2.4.1
# wandb

# Plotting ------------------------
pandas>=1.1.4
seaborn>=0.11.0
```

Slika 3.3. Sadržaj datoteke requirements.txt

```
# For Torch GPU

-i https://download.pytorch.org/whl/cu113

torch==1.11.0+cu113
torchvision==0.12.0+cu113
```

Slika 3.4. Sadržaj datoteke requirements_gpu.txt

3.2. Opis korištene baze podataka za treniranje algoritma i validaciju ispravnosti rada algoritma

Za kreiranje baze slika u svrhu treniranja i validacije korištene su već gotove baze slika koje su kreirane u sklopu diplomskih radova [15] i [29] te slike iz *online* dostupnih baza: *BDD100K Berkeley Deep Drive Dataset* [30], *KITTI Vision Benchmark Suite* [31], *The EuroCity Dataset* [32], *nuScenes* [33] i *Parallel Domain* [34]. Iz tih svih baza slika izdvojeno je ukupno 2307 slika, tako da su izdvojene samo one slike koje sadrže objekte koje je potrebno detektirati i to fotografirane sprijeda, iz razloga jer se pretpostavlja da sudionik prometa dolazi predmetnom vozilu straga (kamera u retrovizoru snima prednju stranu nadolazećeg sudionika prometa). Također su iz svih korištenih baza izdvojene slike koja sadrže raznolika okruženja. Sadrže dnevne i noćne uvjete vožnje kao i različite vremenske uvjete te se nalaze u ukupnom broju od 2307 slika. Baza od 2307 slika je podijeljena za trening, validaciju i test u omjeru 81.27 : 9.36 : 9.36 te sadrži 1875 slika za trening, 216 za validaciju i 216 za test. Broj oznaka po klasama za treniranje CNN za algoritme zasnovane na YOLOv7 i YOLOv7 *Tiny* CNN prikazan je u tablici 3.1., za validaciju u tablici 3.2., dok je za test prikazan u tablici 3.3. Primjeri slika za svaku klasu iz kreirane baze slika korištene za proces treniranja prikazani su na slici 3.5.

Tablica 3.1. Broj označenih objekata u skupu od 1875 slika za treniranje YOLOv7 CNN

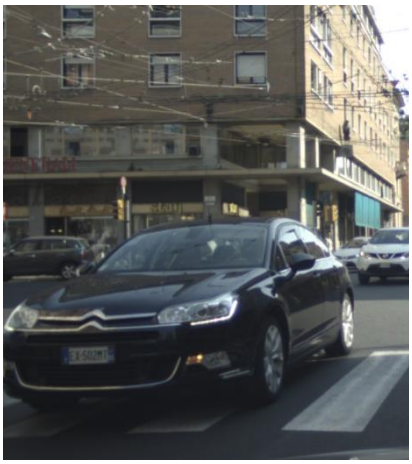
TRENIRANJE	
KLASA	**BROJ OZNAKA**
AUTOMOBIL	2350
KAMION I AUTOBUS	321
PJEŠAK	1260
BICIKLIST	686
MOTOCIKLIST	262
UKUPNO:	**4617**

Tablica 3.2. Broj označenih objekata u skupu od 216 slika za validaciju YOLOv7 CNN

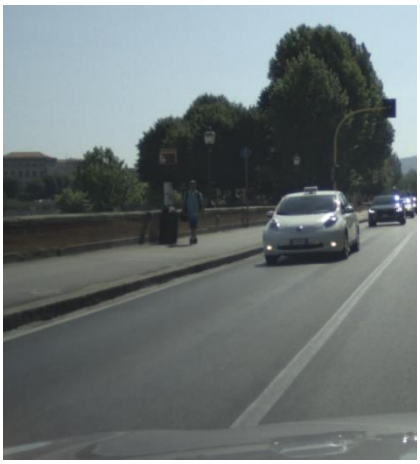
VALIDACIJA	
KLASA	**BROJ OZNAKA**
AUTOMOBIL	283
KAMION I AUTOBUS	37
PJEŠAK	173
BICIKLIST	50
MOTOCIKLIST	61
UKUPNO:	**604**

Tablica 3.3. Broj označenih objekata u skupu od 216 za testiranje YOLOv7 CNN

TEST	
KLASA	**BROJ OZNAKA**
AUTOMOBIL	194
KAMION I AUTOBUS	74
PJEŠAK	147
BICIKLIST	45
MOTOCIKLIST	46
UKUPNO:	**460**

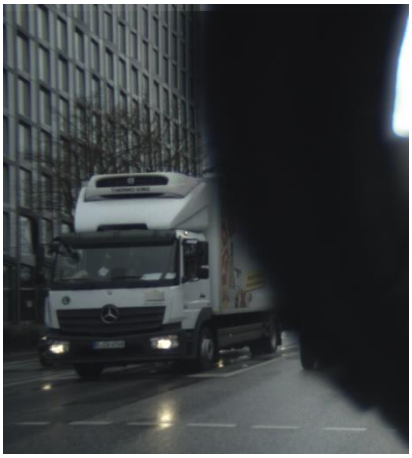


(a)



(b)

(c)



(d)

(e)

(f)

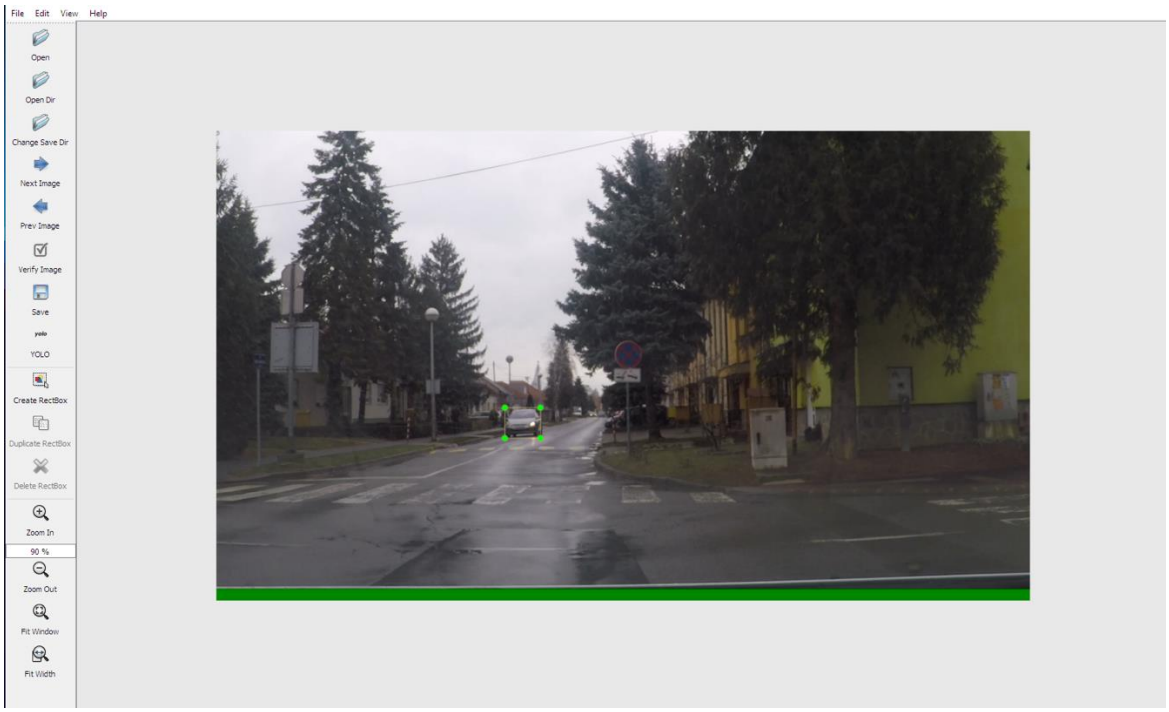
(g) (h) (i)

(j) (k) (l)

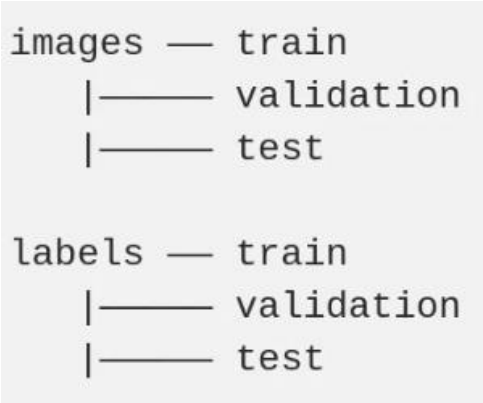
(m) (n) (o)

Slika 3.5. Primjeri slika iz baze koje je potrebno detektirati i klasificirati u sklopu diplomskog rada: (a)(b)(c) automobil, (d)(e)(f) kamion i autobus, (g)(h)(i) pješak, (j)(k)(l) biciklist, (m)(n)(o) motociklist

Kreirana baza slika sadrži sveukupno 2307 slika, od kojih je 905 slika iz baza kreiranih u sklopu diplomskih radova [15] i [29], 575 slika iz *EuroCity* baze, 498 iz *Berkeley Deep Drive* baze, 150 iz *KITTI* baze, 123 iz *Parallel Domain* baze i 56 iz *nuScenes* baze slika. Sve slike koje su se koristile za proces treniranja detektora algoritma zasnovanog na YOLOv7, odnosno YOLOv7 *Tiny* algoritmu mogu se pronaći u elektroničkom prilogu P.3.1. Za trening je korištena baza od 1875 slika, za validaciju baza od 216 slika te za testiranje baza od 216 slika. Sve slike sadrže barem jedan objekt od navedenih klasa koje je potrebno detektirati. Prije početka procesa treniranja potrebno je označiti sve objekte od interesa po klasama zbog načina rada YOLO algoritma. Pri označavanju objekata od interesa po klasama je potrebno paziti da se označe samo prednji dijelovi objekata te se za tu svrhu koristio *labelImg* alat [35], a potpuno označavanje slike prikazano je na slici 3.6. U ovom alatu za označavanje stvara se tekstualna datoteka istog naziva za svaku slikovnu datoteku u direktoriju pod nazivom *labels*, prikaz konačnog izgleda strukture direktorija slika koji je podijeljen na tri skupa, skup za treniranje, skup za validaciju i skup za testiranje (slika 3.7.). Svaka tekstualna datoteka sadrži zapis za odgovarajuću slikovnu datoteku, odnosno klasu objekta, koordinate središta objekta, visinu i širinu. Koordinate središta objekta, visina i širina su decimalni brojevi od nula do jedan, budući da predstavljaju relativnu vrijednost širine i visine objekta u odnosu na širinu i visinu ulazne slike.



Slika 3.6. Potpuno označena slika u *labelImg* alatu



Slika 3.7. Struktura direktorija za proces treniranja

3.3. Opis postupka treniranja algoritma za detekciju nadolazećih sudionika prometa pomoću YOLOv7 algoritma

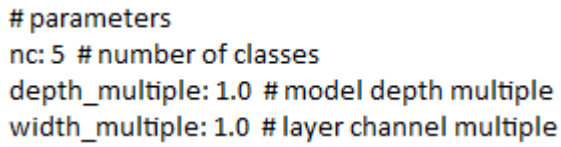
Nakon izrađene baze slika potrebno je postaviti nekoliko konfiguracijski datoteka. Konfiguracije za proces treniranja podijeljene su na tri *YAML* datoteke, a te se datoteke prilagođavaju ovisno o zadatku kako bi odgovarale potrebama. Datoteka konfiguracije podataka opisuje parametre skupa podataka. Trenirano je na prilagođenom skupu podataka o sudionicima prometa, dodana je putanja do skupa podataka za treniranje, validaciju i testiranje; broj klasa (*nc*); i nazive klasa istim redoslijedom kao i njihov indeks. Datoteka konfiguracije podataka nazvana je „*custom_data.yaml*“ i premještena je u direktorij „*data*“. Sadržaj ove datoteke je prikazan na slici 3.8.

```
train: ./data/train
val: ./data/val
test: ./data/test
# number of classes
nc: 5

# class names
names: [ 'car', 'truck', 'person', 'cyclist', 'motorcycle' ]
```

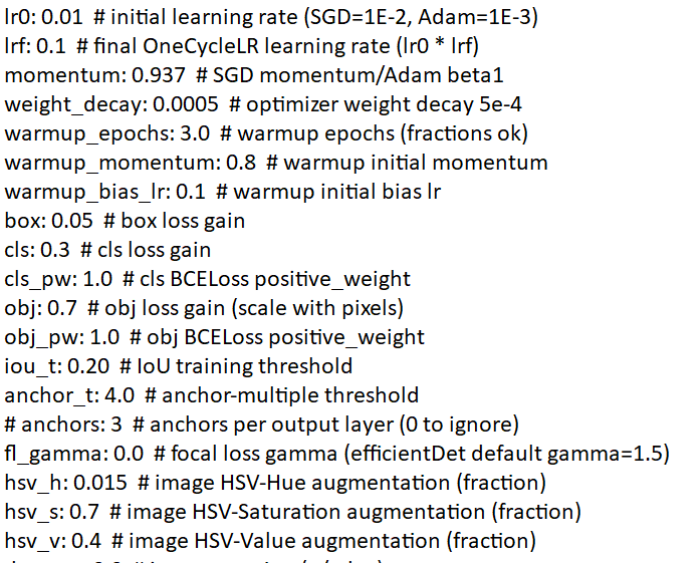
Slika 3.8. Sadržaj datoteke *custom_data.yaml*

Datoteka konfiguracije modela diktira arhitekturu modela. Podržava nekoliko YOLOv7 arhitektura, koje se uglavnom razlikuju ovisno o broju parametara: YOLOv7n (nano), YOLOv7s (mali), YOLOv7m (srednji), YOLOv7l (veliki), YOLOv7x (iznimno velik). YOLOv7 pruža ugrađene konfiguracijske datoteke modela za svaku od gore navedenih arhitektura smještenih u direktorij „*models*“. U datoteci *yolov7-custom.yaml* promijenjen je parametar broj klasa (*nc*) na pet (prikazano na slici 3.9). Za treniranje se koriste unaprijed istrenirane težine *yolov7.pt* i *yolov7-tiny.pt* koje su preuzete s repozitorija odnosno s GitHub stranice [20].



Slika 3.9. Sadržaj datoteke *yolov7-custom.yaml*

Datoteka konfiguracije hiperparametara definira hiperparametre za trening, uključujući stopu učenja (engl. *learning rate*), moment (engl. *momentum*), gubitak (engl. *loss*), augmentaciju (engl. *augmentation*) itd. Zadana datoteka hiperparametara nalazi se u direktoriju „*data*“ pod nazivom „*hyp.scratch.custom.yaml*“, a isječak sadržaja datoteke je prikazan na slici 3.10. Preporučuje se započeti trening sa zadanim hiperparametrima kako bi se uspostavila polazna vrijednost performansi. Sve tri datoteke mogu se pronaći priložene uz ovaj rad kao elektronički prilog u mapi P.3.2.



Slika 3.10. Isječak sadržaja datoteke *hyp.scratch.custom.yaml*

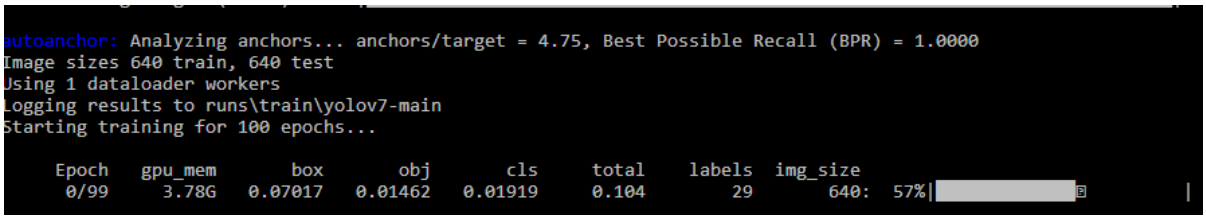
Započeto je treniranje s osnovnom konfiguracijom koje je trajalo približno 30h na NVIDIA GeForce GTX 1060 6GB grafičkom procesoru. Treniranje se pokreće naredbom:

```
python train.py --workers 1 --device 0 –img-size 640 640 –batch-size 8 --epochs 100 --data data/custom_data.yaml --hyp data/hyp.scratch.custom.yaml --cfg cfg/training/yolov7-custom.yaml --name yolov7-custom --weights yolov7.pt –cache
```

u upravljačkom prozoru *Anaconda* sučelja. Parametar *workers* predstavlja koliko pod procesa treba paralizirati tijekom treniranja. *Device* označava da li se trenira pomoću grafičkog procesora u ovisnosti koliko ih ima. Ako je jedan onda se označava parametar „*0*“ te analogno za ostale, jedino ako se radi o procesoru postavlja se oznaka *cpu*. Parametar *img-size* predstavlja rezoluciju slika. Prije početka treniranja potrebno je postaviti parametre epoha (engl. *epochs*) i veličina serije (engl. *batch size*) koji se mijenjaju tijekom procesa treniranja, najbolji parametri su prikazani u

navedenoj naredbi. Epoha koja se određuje pri treniranju postavljena je na 100, a za treniranje YOLOv7 *Tiny* postavljena je na 150. Prolaz kroz cijelu bazu slika koju mreža prođe tijekom treniranja naziva se epoha [36]. Model obrađuje cijeli skup trening podataka svaku epohu kako bi promijenio unutarnje parametre i poboljšao performanse. Baza slika ili podataka se dijeli na manje grupe (engl. *batch*) koji se prosljeđuju mreži jer se ne može odjednom cijela baza proslijediti. Da mreža prođe kroz cijelu bazu slika mora proći kroz određeni broj iteracija (engl. *steps*) nužnih da obradi sve grupe slika [36]. Veličina serije je hiperparametar gradijentnog spuštanja (engl. *gradient descent*) koji kontrolira broj uzoraka za treniranje koje treba obraditi prije ažuriranja internih parametara modela. Veličina serije omogućuje podešavanje parametara modela na temelju prosječnog gradijenta izračunatog na seriji uzoraka za razliku od podešavanja nakon svakog pojedinog primjera. Ukoliko je veličina serije postavljena na jedan onda model dohvaća sliku po sliku i ući na osnovu svake te se naziva stohastički gradijentni pad (engl. *stohastic gradient descent,* SGD). Kada je veličina serije veća od jedan i manja od veličine baze podataka za treniranje tada ažuriranja provodi za svaki mali skup podataka, naziva se metoda gradijentnog pada malog skupa (engl. *mini-batch gradient descent*). U slučaju gradijentnog pada malog skupa, često korištene veličine za veličinu serije uključuju 32, 64 i 128 [36]. Model za zadani broj epoha na osnovu slika učitanih iz baze trenira model, tj. predviđa koristeći svoje trenutne parametre, izračunava gubitak (metriku koliko model dobro radi), a zatim mijenja svoje parametre pomoću postupka optimizacije, poput spuštanja gradijentom, kako bi se smanjio gubitak.

Ostali parametri odnosno hiperparametri, konfiguracijski parametri i težine su objašnjeni ranije, potrebno je dodati putanje do njihovih datoteka. Također, s parametrom *names* se određuje naziv direktorija gdje se spremaju svi podaci od treniranja. Tijekom treniranja trening je zaustavljen zbog nedostatka memorije. Iz tog razloga je dodan parametar *cache* da prisili grafički procesor da što prije oslobodi memoriju. Tijekom treninga, model će emitirati memoriju rezerviranu za trening, broj pregledanih slika, ukupan broj predviđenih oznaka, preciznost, odziv i *mAP @.5* na kraju svake epohe, kako je prikazano na slici 3.11. Te podatke je moguće koristiti radi prepoznavanja spremnosti modela za dovršetak treniranja i za razumijevanje učinkovitost modela na skupu za provjeru valjanosti. Na kraju treninga, najbolje (engl. *best*) i posljednje (engl. *last*) težine modela spremit će se u odgovarajući direktorij. Također će spremiti relevantni podaci o procesu treniranja.



Slika 3.11. Početak treniranja

Rezultati treniranja konačne CNN za YOLOv7 algoritam prikazani su u obliku matrice zabune u tablici 3.4., dok su preciznost i odziv za pojedinu klasu i za cijelu bazu slika za treniranje prikazani u tablici 3.5. Graf vrijednosti preciznosti u odnosu na odziv dobiven u postupku treniranja prikazan je na slici 3.12.

Cjelobrojne oznake klasa odgovaraju redoslijedu navođenja klasa prilikom treniranja YOLOv7 CNN:

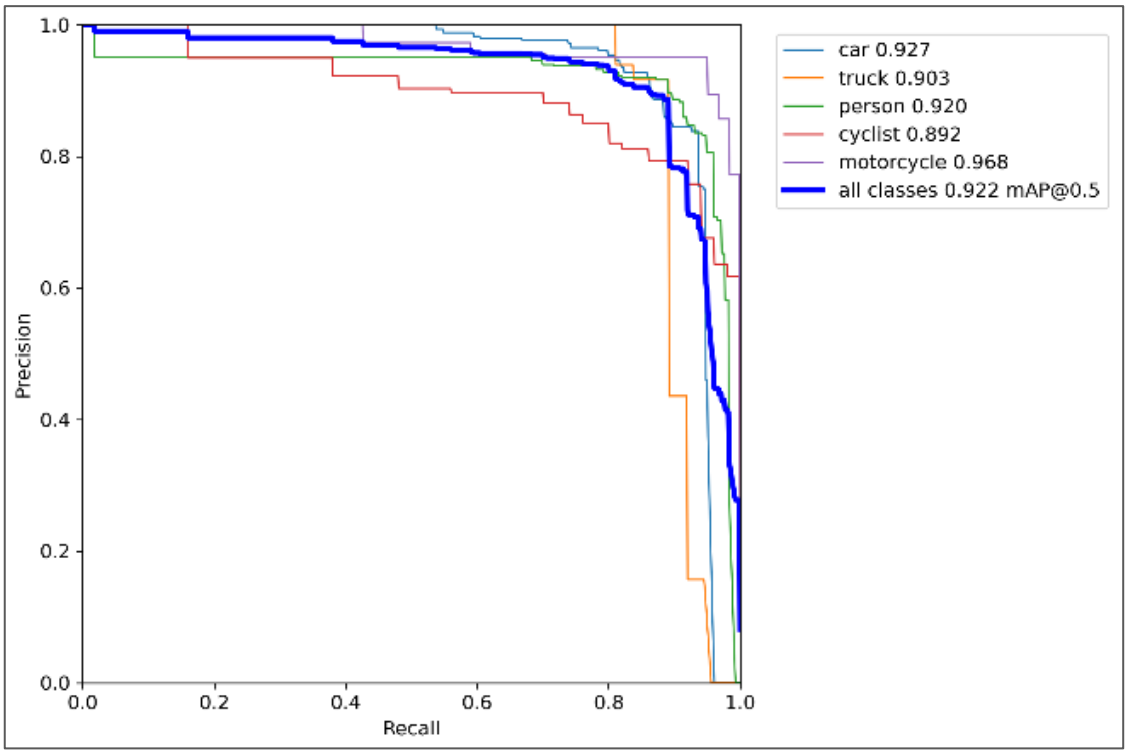
- 0 – automobil,
- 1 – kamion i autobus,
- 2 – pješak,
- 3 – biciklist,
- 4 – motociklist.

Tablica 3.4. Matrica zabune za najbolji YOLOv7 model prilikom treninga na trening skupu podataka

Predviđeno							
	0	2068	0	0	0	0	1081
	1	24	292	0	0	0	13
	2	0	0	1172	14	0	416
	3	0	0	0	569	0	75
	4	0	0	0	14	254	16
	pozadina	258	29	88	89	8	
		0	1	2	3	4	pozadina
		Istina					

Tablica 3.5. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli trening skup za najbolji YOLOv7 model prilikom treninga na trening skupu podataka

KLASA	0	1	2	3	4	CIJELI TRENING SKUP ZA TRENIRANJE
PRECIZNOST	65.67%	88.94%	73.18%	88.30%	89.62%	72.50%
ODZIV	88.00%	91.00%	93.00%	83.00%	97.00%	89.27%



Slika 3.12. Graf vrijednosti preciznosti u odnosu na odziv dobiven u postupku treniranja za najbolji YOLOv7 model

Matrica zabune je tablica koja sumira performanse klasifikacijskog modela prikazujući brojeve istinski pozitivnih, istinski negativnih, lažno pozitivnih i lažno negativnih predviđanja. Matrice zabune u ovom slučaju su dimenzije 5 x 5 jer predstavljaju broj klasa koje treba predvidjeti. Iznimno, zbog YOLO algoritma je izvedena klasa „pozadina“ koja ne utječe na performanse modela jer nije označena pri označavanju željenih klasa za detekciju. Klasa „pozadina“ u matrici zabune predstavlja pozadinske objekte koji ne pripadaju nijednoj od klasa. Klasa „pozadina“ predstavlja objekte koje je detektor propustio detektirati i koji se smatraju nekim drugim objektom, tj. odnosi se na detekcije koje nisu proglašene objektom, a one to jesu [20]. To je prikazano u tablici 3.4. rezultatom od 258 propuštenih detekcija koje zapravo predstavljaju klasu automobil. Uključivanje pozadinskih slika (neoznačenih slika) u proces treniranja je korisno za procjenu performansi modela u razlikovanju objekata i pozadinskih scena. Pomaže u procjeni sposobnosti modela da izbjegne lažno pozitivne rezultate. Što se tiče postotka pozadinskih slika u skupu podataka, dokumentacija predlaže da čine između 0% i 10% od ukupnog skupa podataka [20]. Uključivanje previše pozadinskih slika utječe na sposobnost modela da učinkovito detektira objekte. Dijagonalni elementi predstavljaju broj za koje je predviđena oznaka jednaka pravoj oznaci (TP), dok su izvan-dijagonalni elementi (FP i FN) oni koje klasifikator pogrešno označava. Što su dijagonalne vrijednosti matrice zabune veće to ukazuje na točnija predviđanja. Kao rezultat, zbroj svakog od stupaca trebao bi biti jednak broju označenih objekta za svaku klasu tj. u matrici

prikazanoj u tablici 3.4. zbroj svakog stupca od svake pojedinačne klase predstavlja broj označenih objekata svake pojedine klase prikazane u tablici 3.1. Na slici 3.12. prikazan je graf vrijednosti preciznosti u odnosu na odziv provedenih u postupku treniranja. Tijekom treniranja procijenjeno je područje ispod krivulje kao prosječnu preciznost, *AP*. Krivulja bi u idealnom slučaju trebala ići od preciznosti jednakoj jedan i odzivu jednakom nula u gornjem lijevom kutu te prema preciznosti jednakoj nula i odzivu jednakom jedan u donjem desnom kutu kako bi se zabilježilo područje ispod krivulje. Graf je silazan za svaku klasu kao i za sve klase zajedno jer, što se preciznost više smanjuje to se više pretpostavki napravi (engl. *recall*). Različitim vrijednostima parametra *conf-thres* može se odabrati jedna točka na krivulji pri pokretanju modela, tijekom cijelog procesa treniranja, ali i pri pokretanju krajnjeg rješenja. Vrijednost parametra je postavljena na 0.65 kako bi se dala prednost odzivu u odnosu na preciznost.

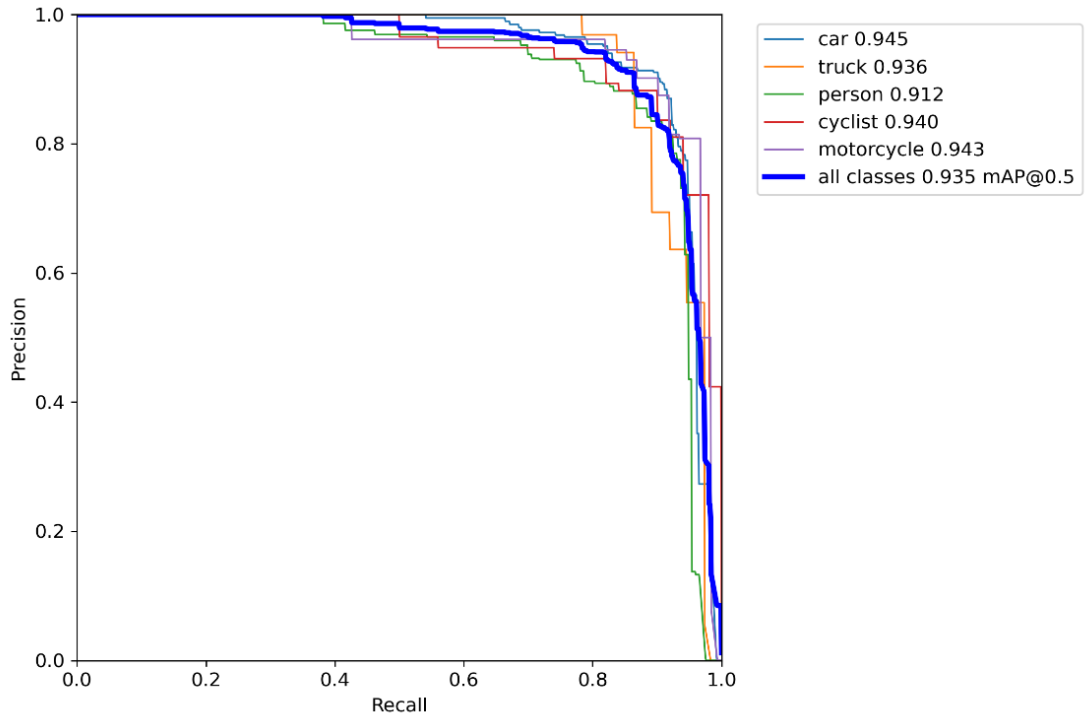
Također, matrica zabune za validacijski skup opisan tablicom 3.2 dana je u tablici 3.6. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli validacijski skup dan je u tablici 3.7. te je graf vrijednosti preciznosti u odnosu na odziv dobiven u postupku validacije prikazan na slici 3.13.

Tablica 3.6. Matrica zabune za najbolji YOLOv7 model prilikom validacije na validacijskom skupu podataka

Predviđeno							
	0	263	1	0	0	0	127
	1	3	34	0	0	0	3
	2	0	0	164	1	0	52
	3	0	0	0	45	0	6
	4	0	0	0	1	57	4
	pozadina	17	2	9	3	4	
		0	1	2	3	4	pozadina
		Istina					

Tablica 3.7. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli validacijski skup za najbolji YOLOv7 model prilikom validacije na validacijskom skupu podataka

KLASA	0	1	2	3	4	CIJELI VALIDACIJSKI SKUP ZA VALIDACIJU
PRECIZNOST	67.20%	86.26%	75.65%	89.11%	91.50%	74.04%
ODZIV	93.00%	91.09%	95.00%	90.00%	93.00%	93.21%



Slika 3.13. Graf vrijednosti preciznosti u odnosu na odziv dobiven u postupku validacije za najbolji YOLOv7 model

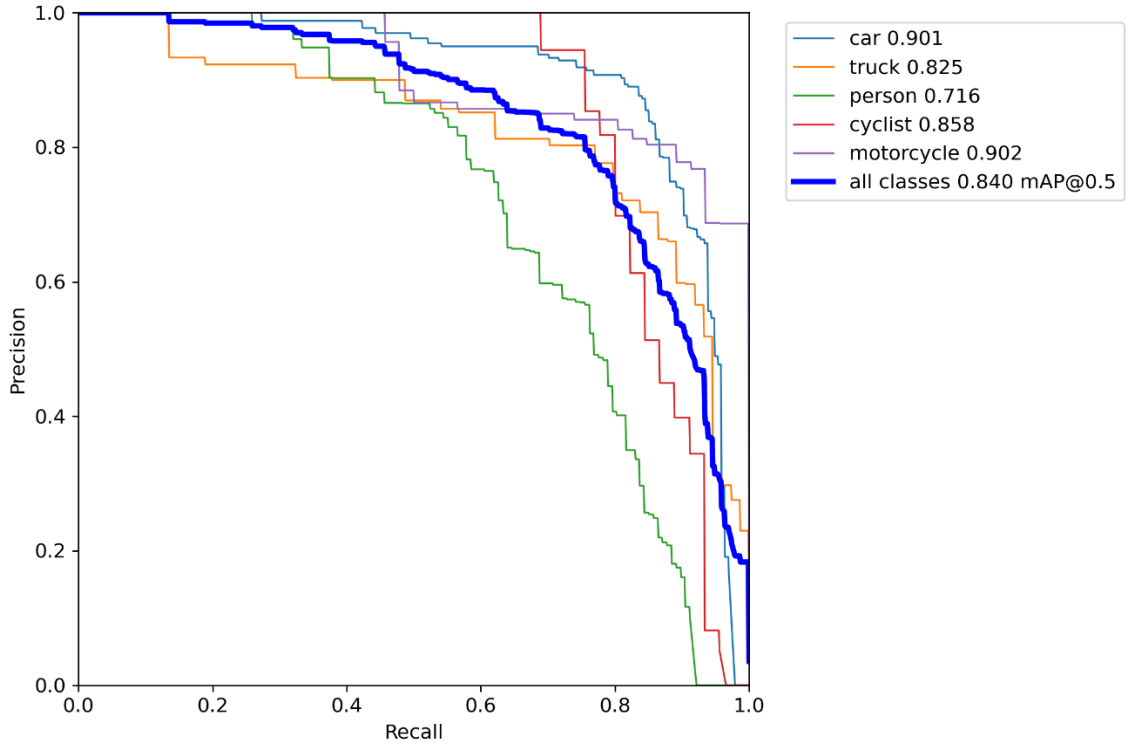
Testiranje je napravljeno na novom skupu slika. Broj označenih objekata za svaku klasu prikazan je u tablici 3.3. Za testni skup matrica zabune je prikazana u tablici 3.8. Prikazuje da je zbroj svakog od stupaca jednak broju označenih objekta za pojedinu klasu u tablici 3.3. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli skup za testiranje nalazi se u tablici 3.9. te je graf vrijednosti preciznosti u odnosu na odziv prikazan na slici 3.14.

Tablica 3.8. Matrica zabune za najbolji YOLOv7 model prilikom testiranja na testnom skupu podataka

Predviđeno							
	0	165	6	0	0	0	29
	1	4	62	0	0	0	13
	2	0	0	101	2	0	63
	3	0	0	1	36	1	3
	4	0	0	0	0	45	8
	pozadina	25	6	44	7	0	
		0	1	2	3	4	pozadina
		Istina					

Tablica 3.9. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli testni skup za najbolji YOLOv7 model prilikom testiranja na testnom skupu podatak

KLASA	0	1	2	3	4	CIJELI TESTNI SKUP ZA TESTIRANJE
PRECIZNOST	82.48%	78.33%	60.94%	87.61%	84.48%	75.82%
ODZIV	85.00%	84.00%	69.00%	80.00%	98.00%	80.94%



Slika 3.14. Graf vrijednosti preciznosti u odnosu na odziv dobiven u postupku testiranja za najbolji YOLOv7 model

Rezultati treniranja konačne CNN za YOLOv7 *Tiny* algoritam prikazani su u obliku matrice zabune u tablici 3.10., dok su preciznost i odziv za pojedinu klasu i za cijeli trening skup prikazani su u tablici 3.11. Broj označenih objekata za svaku klasu prikazan je u tablici 3.1. Visoke vrijednosti u matrici zabune nalaze se duž dijagonale za prvu i drugu klasu. To ukazuje na to da je model uspješan u predviđanju velikog broja slučajeva koji pripadaju tim klasama. Međutim, postoje neke pogrešne klasifikacije, što zahtijeva povećanje baze slika i daljnje treniranje modela. Matrice zabune pružaju uvid u uspješnost modela klasifikacije i pomažu u usmjeravanju daljnjih poboljšanja modela.

Tablica 3.10. Matrica zabune za najbolji YOLOv7 *Tiny* model prilikom treniranja na trening skupu podataka

Predviđeno							
	0	2045	0	0	0	0	1457
	1	23	247	0	0	0	10
	2	0	0	869	82	0	202
	3	0	0	0	364	5	82
	4	0	0	0	41	197	16
	pozadina	282	74	391	199	60	
		0	1	2	3	4	pozadina
		Istina					

Tablica 3.11. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli trening skup za najbolji YOLOv7 *Tiny* model prilikom treninga na trening skupu podataka

KLASA	0	1	2	3	4	CIJELI TRENING SKUP ZA TRENIRANJE
PRECIZNOST	58.39%	88.34%	75.38%	80.59%	77.55%	65.99%
ODZIV	87.02%	77.00%	69.00%	53.00%	76.53%	76.36%

Također, rezultati za validaciju YOLOv7 *Tiny* modela prikazani su matricom zabune u tablici 3.12. Zbroj svakog od stupaca jednak je broju označenih objekta za pojedinu klasu koje su prikazane u tablici 3.2. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli validacijski skup dan je u tablici 3.13.

Tablica 3.12. Matrica zabune za najbolji YOLOv7 *Tiny* model prilikom validacije na validacijskom skupu podataka

Predviđeno							
	0	249	0	0	0	0	158
	1	3	32	0	0	0	1
	2	0	0	126	3	0	43
	3	0	0	4	13	1	4
	4	0	0	0	4	43	4
	pozadina	31	5	43	30	17	
		0	1	2	3	4	pozadina
		Istina					

Tablica 3.13. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli validacijski skup za najbolji *Tiny* YOLOv7 model prilikom validacije na validacijskom skupu podataka

KLASA	0	1	2	3	4	CIJELI VALIDACIJSKI SKUP ZA VALIDACIJU
PRECIZNOST	61.11%	88.07%	73.19%	59.96%	84.79%	67.25%
ODZIV	88.00%	86.00%	73.00%	26.00%	70.00%	76.63%

Testiranje za YOLOv7 *Tiny* je napravljeno na istom skupu slika kao i testiranje za YOLOv7. Rezultat testiranja su prikazani u obliku matrice zabune u tablici 3.14. Prikazuje da je zbroj svakog od stupaca jednak broju označenih objekta za pojedinu klasu u tablici 3.3. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli testni skup nalazi se u tablici 3.15.

Tablica 3.14. Matrica zabune za najbolji YOLOv7 *Tiny* model prilikom testiranja na testnom skupu podataka

Predviđeno							
	0	146	1	0	0	0	81
	1	17	59	0	0	0	9
	2	0	0	83	9	0	40
	3	0	0	1	25	2	5
	4	0	0	0	0	39	4
	pozadina	31	13	63	11	5	
		0	1	2	3	4	pozadina
		Istina					

Tablica 3.15. Prikaz preciznosti i odziva za pojedinu klasu i za cijeli testni skup za najbolji YOLOv7 *Tiny* model prilikom testiranja na testnom skupu podatak

KLASA	0	1	2	3	4	CIJELI TESTNI SKUP ZA TESTIRANJE
PRECIZNOST	63.69%	69.21%	63.46%	75.31%	91.40%	67.55%
ODZIV	75.00%	80.00%	56.44%	56.00%	85.86%	69.58%

Usporedbom rezultata vidljivo je da YOLOv7 *Tiny* model ima lošije rezultate preciznosti i odziva na svim skupovima podataka (treniranje, validacije, testiranje) u usporedbi s običnim YOLOv7 modelom. Rezultati YOLOv7 *Tiny* modela posljedica su smanjene računalne kompleksnosti modela, što dovodi do gubitka u detekcijama i praćenju objekata. Iako je YOLOv7 *Tiny* model brži za obradu, njegova točnost u detekciji nije toliko velika. Iz rezultata je vidljivo da preciznost YOLOv7 modela 75.82 % (vidljivo je iz 3.9. tablice), dok je za YOLOv7 *Tiny* model postigao preciznost od 67.55% (vidljivo je iz 3.15. tablice) na cijelom testnom skupu. Slično tome, odziv običnog YOLOv7 modela je 80.94% (vidljivo je iz 3.9. tablice) dok YOLOv7 *Tiny* model postiže odziv od 69.58% (vidljivo je iz 3.15. tablice) na cijelom testnom skupu.

Usporedbom rezultata YOLOv7 i YOLOv7 *Tiny* modela vidljivi su značajni padovi u preciznosti i odzivu za određene klase. Za običan YOLOv7 i YOLOv7 model *Tiny*, klase kod kojih je najveći pad vidljiv u preciznosti i odzivu su biciklist (3) i pješak (2). U odnosu na obični YOLOv7 model, YOLOv7 *Tiny* model pokazuje značajan pad vidljiv u preciznosti i odzivu kod tih klasa na svim skupovima podataka. Razlozi pada preciznosti i odziva su manja sposobnost YOLOv7 *Tiny* modela da uhvati detalje i karakteristike tih klasa, kao i manja složenost same arhitekture modela. Moguće je da bi povećanje trening skupa podataka za navedene klase dovelo do boljih rezultata za YOLOv7 *Tiny*. Za klasu automobil (0), klasu kamion i autobus (1) i klasu motociklist (4) pad vidljiv u preciznosti i odzivu je manji. YOLOv7 model ima veću preciznost od YOLOv7 *Tiny* modela za sve tri klase. Posebno, razlika u preciznosti najviše je izražena kod klase kamion i autobus (1), gdje YOLOv7 model ostvaruje značajno bolje rezultate. Odziv YOLOv7 modela također je veći u usporedbi s YOLOv7 *Tiny* modelom za sve tri klase. Za klase automobil te kamion i autobus vidljivi su slični rezultati u odzivu između modela, dok je značajna razlika u rezultatu odziva kod klase motociklist. U slučaju klase motociklist YOLOv7 *Tiny* model ima višu preciznost, ali niži odziv u usporedbi s običnim YOLOv7 modelom. To ukazuje na to da YOLOv7 *Tiny* ispravno, tj. pouzdano, detektira motocikliste, ali zato često propusti detektirati sve instance te klase.

U konačnici, iako YOLOv7 *Tiny* model može brže detektirati objekte i zahtijeva manje resursa, obični YOLOv7 model nadmašuje ga u preciznosti i odzivu. Prilagodbe parametara, veći trening skup podataka ili upotreba naprednijeg modela može dodatno poboljšati performanse detekcije.

3.4. Procjena udaljenosti, brzine sudionika prometa i vremena preostalog do potencijalne kolizije

Kako bi se smanjile prometne nesreće, sve je veći fokus na tehnologijama koje pomažu vozaču u današnjim automobilima. Prilikom izlaska iz vozila uobičajeno je da vozači pogrešno procjenjuju brzinu i udaljenost sudionika prometa. Ovo rješenje pomaže u predviđanju brzine i udaljenosti za sigurniji izlazak iz vozila. Brzina promjene okvira video sekvenci jedan je od ključnih elementa za izračun jer omogućuje da se odredi koliko je vremena prošlo između dviju lokacija sudionika prometa u video sekvenci, a on se izračunava automatski za svaku video sekvencu posebno. Vrsta korištene kamere i specifikacije objektiva koji se koristi utječu na proračun uz brzinu promjene okvira. Također, jedan od bitnijih parametra koji utječu na procjenu udaljenosti i brzine je i visina samih sudionika prometa. Za sve objekte uzeta je prosječna visina

za svaku klasu koju model može detektirati. Visina objekta je uzeta zato što je širina neprecizna tj. veći auto može imati sličnu širinu kao kamion. Kako bi se vozaču pružilo više informacija prilikom odlučivanja o tome hoće li moći sigurno izaći iz vozila, dodana je informacije o udaljenosti i vremenu preostalom do potencijalnog sudara s vratima vozila, jer sama brzina sudionika prometa ne pruža mnogo informacija. Udaljenost identificiranog sudionika prometa do vrata vozila, njegova brzina i vrijeme potrebno da stigne na mjesto kamere izračunavaju se na način koji je objašnjen u nastavku.

3.4.1. Procjena udaljenosti sudionika prometa

Cannon EOS 750d je kamera korištena u razvoju rješenja, a korišteni objektiv je Cannon EF-S 18-55mm F/3.5-5.6 IS STM. Budući da kamera ima jedinstvena svojstva, rješenje se mora prilagoditi za korištenu kameru. Parametri vezani za kameru se moraju promijeniti ako se želi koristiti rješenje da funkcionira s drugom kamerom. Žarišna duljina objektiva i visina senzora bitni su elementi za kameru koji su korišteni. Visina senzora kamere je 0.0149 m, a žarišna duljina objektiva je 0,0018 m [37]. Udaljenost detektiranog sudionika prometa i brzina s kojom se približava ključni su podaci za vozača jer mu pomažu da odluči je li sigurno izaći iz vozila ili ne. U formuli (3-1) korištena je visina sudionika prometa, a ne širina sudionika prometa jer se više razlikuju po visini. Varijabla visine sudionika prometa u okviru videozapisa također se ažurira. Umjesto da se koristi širina video okvira i širina senzora kamere, koristi se visina video okvira i visina senzora kamere. Procijenjena udaljenosti između kamere i sudionika prometa koji se približava izračunava se na temelju formule (3-1):

$$d = \frac{f[m] \cdot h[m] \cdot h_{slike}[element\ slike]}{h'[element\ slike] \cdot h_{senzor}[m]}[m], \tag{3-1}$$

gdje je:

- d - udaljenost objekta od kamere na automobilu u metrima,
- f - žarišna duljina leće u metrima,
- h - visina sudionika prometa u metrima u stvarnom svijetu,
- h_{slike}- visina video okvira u elementima slike,
- h' - visina objekta u video okviru u broju elemenata slike,
- h_{senzor} - visina senzora kamere u metrima.

Prilikom određivanja brzine svjesno se uvodi pogreška izračunom udaljenosti. Izračun uzima u obzir prosječnu stvarnu visinu svakog sudionika posebno: za automobil 1.46 m [38], za kamion 2.985 m [39], za pješaka 1.75 m [40], za biciklista 1.82 m [41] i za motociklista 1.27 m

[42]. Tijekom procjene udaljenosti sudionika prometa dolazi do netočnosti ako je visina sudionika prometa niža ili viša od prosjeka. Objekti koji nisu detektirani do desetog video okvira obrisani su odnosno premješteni su u listu propuštenih objekata.

3.4.2. Procjena brzine sudionika prometa

Procjena brzine sudionika prometa koji se približava vratima vozača objašnjena je u nastavku. Automatska procjena brzine omogućava sigurniji izlazak iz vozila. Budući da se procjena provodi na osnovu prethodno izračunate udaljenosti sudionika prometa koji se približava, procjena brzine ovisi i o kameri koja se koristi. Procjena brzine nadolazećeg sudionika prometa neprecizna je zbog nedostataka u metodi koja se koristi za mjerenje udaljenosti. Formula (3-2) koristi se za izračunavanje brzine sudionika prometa koji se približava:

$$v = \frac{d_{i-1} - d_i}{t} \left[\frac{m}{s}\right], \tag{3-2}$$

gdje je:

- v - brzina nadolazećeg sudionika prometa u metrima u sekundi,
- d_{i-1} - udaljenost do sudionika prometa na prethodnom okviru u metrima,
- d_i - udaljenost do sudionika prometa na trenutnom okviru u metrima,
- t - period između dvaju uzastopnih video okvira u sekundama.

Period između dvaju uzastopnih video okvira se izračunava kao recipročna vrijednost broja okvira u sekundi. Pogreška koja proizlazi iz procjene udaljenosti i detekcija sudionika prometa utječe na procjenu brzinu. Uz poznatu udaljenost detektiranog sudionika prometa od kamere u prethodnom i trenutnom okviru određuje se prevaljeni put tako da se oduzme udaljenost prethodnog i trenutnog video okvira. Medijan zadnjih deset procijenjenih brzina koristi se za rješavanje problema oscilacija u procjeni brzine.

3.4.3. Procjena vremena preostalog do potencijalne kolizije

Vrijeme potrebno detektiranom sudioniku prometa da stigne do mjesta kamere mjeri se kao vrijeme preostalo do potencijalne kolizije (engl. *time to colision*, TTC). Ova informacija daje vozaču do znanja koliko će vremena proći prije nego što se pojavi sudionik prometa kod vrata automobila. Procjena TTC-a generira se korištenjem informacija o udaljenosti i predviđenoj brzini (3-3).

$$TTC = \frac{d}{v}[s], \tag{3-3}$$

gdje je:

- TTC - vrijeme koje je potrebno nadolazećem sudioniku prometa da dođe do mjesta kamere u sekundama,
- d - udaljenost do nadolazećeg sudionika prometa u metrima,
- v - procijenjena brzina nadolazećeg sudionika prometa u metrima u sekundi.

Vozač može procijeniti ima li dovoljno vremena da sigurno otvori vrata vozila pomoću TTC podataka. Upravo je on korišten za aktiviranje alarmnog sustava u obliku tekstualne poruke na video sekvenci, a aktivira se kada je TTC manji od tri sekunde. Predstojeće pogreške u procjeni udaljenosti i brzine sudionika prometa dovode do netočnosti. Ove informacije pružaju grubu predodžbu o tome koliko će vremena vozač imati jer, ako sudionik prometa koji se približava iznenada ubrza, vozačevo vrijeme za izlazak iz vozila brzo će se smanjiti, a pravo je pitanje koliko brzo predloženo rješenje može reagirati na to.

3.5. Upute za pokretanje rješenja

Za pokretanje rješenja potrebno je instalirati biblioteke navedene u potpoglavlju 3.1. vidljive na slikama 3.3. i 3.4. Za pokretanje rješenja potrebno je otvoriti terminal *Anaconda Prompt* te se pozicionirati u direktorij gdje se nalazi skripta rješenja i pokrenuti virtualno okuženje. Zatim je potrebno upisati sljedeću naredbu:

```
Python object_tracking_v2.py --view-img –source <putanja_do_video_sekvence> --project <naziv_direktorija_spremanja_generirane_video_sekvence> --name <naziv_video_sekvence>
```

Skripta *object_tracking_v2.py* se nalazi u elektroničkom prilogu P.3.3. kao i ostale potrebne Python skripte za pokretanje rješenja.

3.6. Implementacija rješenja na ugradbenu računalnu platformu Raspberry Pi 4

Razvijeno rješenje potrebno je implementirati na ugradbenu računalnu platformu Raspberry Pi. Raspberry Pi 4 platforma u potpunosti je sposobna za pokretanje 64-bitnih operacijskih sustava, za razliku od Raspberry Pi 3 platforme koji podržava samo 1 GB RAM-a (engl. *Random Access Memory*). Zbog toga se i Raspberry Pi 4 platforma koristi za implementaciju u sklopu ovog diplomskog rada, jer omogućuje instalaciju 64-bitnog Raspberry Pi operacijskog sustava koji je potreban za instalaciju *PyTorch* biblioteke koja je instalirana pomoću „terminala“. *PyTorch* je softverska biblioteka posebno razvijena za duboko učenje, ali troši puno resursa Raspberry Pi platforme [43]. Unutar *PyTorch*-a je instalirana biblioteka *TorchVision* koja je potrebna za

pokretanje rješenja pa nju nije potrebno zasebno instalirati. Potrebno je instalirati unutar „terminala“ svaku biblioteku posebno (slika 3.3. u potpoglavlju 3.1.), kako bi se rješenje uspješno moglo pokrenuti na Raspberry Pi platformi. Za pokretanje rješenja na Raspberry Pi platformi istreniran je model YOLOv7 *Tiny* na istom skupu slika koji je korišten i za YOLOv7 model, a rezultati modela su prikazani u potpoglavlju 3.3. Nakon instalacije navedenih biblioteka za pokretanje je potrebno napraviti isti postupak kao što je i prethodno objašnjeno u potpoglavlju 3.4., ali je potrebno koristiti *Tiny* model. Parametar *weights* predstavlja model koji se koristi, a može se promijeniti u skripti *object_tracking_v2.py* ili se pri pokretanju može dodati parametar *weights* i pored njega napisati ime *Tiny* modela.

4. EVALUACIJA PERFORMANSI RJEŠENJA ZA UKLJUČIVANJE ALARMNOG SUSTAVA PRILIKOM IZLASKA VOZAČA IZ VOZILA

U ovom poglavlju opisan je način evaluacije kompletnog izgrađenog rješenja na testnom skupu video sekvenci koje su napravljene u sklopu ovog diplomskog rada. U potpoglavlju 4.1. opisan je postupak izrade video sekvenci za sve sudionike prometa odnosno klase koje su istrenirane. Evaluacija točnosti i brzine rada rješenja obavljena je koristeći 34 video sekvence te su iste korištene za evaluaciju na Raspberry Pi 4 platformi. Rezultati evaluacije su prikazani u potpoglavljima 4.2. i 4.3. U potpoglavlju 4.4. dana je analiza rezultata rješenja te je provedena diskusija.

4.1. Izrada skupa video sekvenci za testiranje rada rješenja

U ovom potpoglavlju opisana je izrada skupa podataka odnosno video sekvenci koje su korištene prilikom evaluacije performansi rješenja. Snimljene su 34 video sekvence pomoću kamere Cannon EOS 750d. Prilikom snimanja video sekvenci kamera miruje te je postavljena u visini retrovizora automobila. Izmjerene su udaljenosti od mjesta kamere 5 m, 10 m, 15 m, 20 m i 25 m i označene su na cesti markerima prije snimanja video sekvenci. Udaljenost detektiranog sudionika prometa od kamere, brzina detektiranog sudionika prometa i vrijednosti TTC-a procijenjene su na zadanim udaljenostima. Rezolucija video okvira namještena je na 640 x 480 piksela prije početka snimanja video sekvenci, a navedena rezolucija je korištena zbog manjeg vremena obrade video okvira. Stvarne brzine za klase automobil, kamion i motociklist u video sekvencama bile su 20 km/h, 30 km/h, 40 km/h i 50 km/h. Za klasu automobil i kamion koristio se *tempomat*[2] pri snimanju video sekvenci sa stvarnim brzinama: 30 km/h, 40 km/h i 50 km/h. Za klasu automobil odnosno za video sekvence od car20.mp4 do car50.mp4 *tempomat* nije korišten pri snimanju video sekvenci jer nije bio dio opreme automobila.

Za klase pješak i biciklist su korištene srednje vrijednosti njihove brzine kretanja i vožnje koje su izmjerene pomoću aplikacija *Speedometer Simple* [44] i *Samsung Health* [45]. Dvije video sekvence s pješacima su snimljene pri njihovom trčanju s brzinama od 13 km/h do 15 km/h. Treba imati na umu da brzinomjer na kontrolnoj ploči vozila također ima određenu netočnost. Jedan od

[2] *Tempomat* je sustav koji automatski kontrolira brzinu automobila. Sustav je servomehanizam koji preuzima gas automobila kako bi održao ujednačenu brzinu koju je postavio vozač.

razloga je to je što se brzina vozila mjeri brzinom okretaja kotača i to svakih 50 milisekundi kako bi prikazana brzina uvijek bila aktualna. No kako gume mogu biti različitih dimenzija i troše se, njihov dijametar nije uvijek isti pa ni izmjerena brzina, te se odstupanja mogu događati i kod preslabo ili previše napumpanih guma [46]. Još jedan od bitnih podataka je da prema procjenama, netočnost prikaza brzine automobila iznosi 10% stvarne brzine. Propisima je uvedeno da brzinomjeri maksimalno pokazuju 10% + 6 km/h veću brzinu od stvarne brzine za automobile, autobuse, kamione i druga gospodarska vozila, a za motocikle 10% + 8 km/h veću brzinu od stvarne brzine [47]. To znači da brzinomjer u automobilu može maksimalno pokazivati 50 km/h, a zapravo automobil doista vozi brzinom od 40 km/h. Originalne testne video sekvence se nalaze u elektroničkom prilogu P.4.1. Rezultati evaluacije su prikazani u nastavku.

4.2. Evaluacija točnosti i brzine rada razvijenog rješenja pokrenutog na osobnom računalu

U ovom potpoglavlju opisana je evaluacija rada rješenja u pogledu točnosti i brzine rada na testnim video sekvencama pokrenutim na osobnom računalu. Slika 4.1. prikazuje rezultate za video sekvencu za klasu automobil koji se kreće konstantnom stvarnom brzinom od 30 km/h. Odabrani kut snimanja utječe na ishode procjene udaljenosti, brzine i procjena TTC-a zato što smjer snimanja mora biti paralelan smjeru kretanja automobila. Nedostatak koji utječe na rezultat detektiranja sudionika prometa je visina graničnog pravokutnika, koja se koristi pri izračunu udaljenosti, te se stalno mijenja odnosno nije u točnim granicama prikazanog automobila na slici 4.1. Rezultati procjene udaljenosti, brzine i TTC-a unatoč navedenim problemima i nedostacima ne odstupaju znatno od stvarnih rezultata. Procijenjena brzina vozila na svim udaljenostima je precijenjena. To se događa zbog oscilacija detektora koje se događaju prilikom detekcije iz okvira u okvir u video sekvenci. Iz okvira u okvir visina detektiranog vozila se mijenja više nego što bi se trebala promijeniti te je stoga procijenjena brzina veća od stvarne.

Slika 4.2. prikazuje rezultate za video sekvencu za klasu kamion koji se kreće konstantnom stvarnom brzinom od 30 km/h. Nedostatak u procjeni navedenih veličina je stvarna visina i visina graničnog pravokutnika detektiranog kamiona. Precijenjene su procijenjene brzine na udaljenostima 25, 20 i 15 metara zbog visine graničnog pravokutnika koja treba biti viša. Granični pravokutnik je u okviru detektiranog vozila na video sekvenci na navedenim udaljenostima dok je na udaljenosti 10 i 5 metara on viši, ali su udaljenosti i brzine točnije.

(a)

(b)

(c)

(d)

(e)

Slika 4.1. Rezultati rada rješenja za video sekvencu car4_30.mp4, stvarna brzina 30 km/h, automobil na (a) 25 m, (b) 20 m, (c) 15 m, (d) 10 m, (e) 5 m udaljenosti od kamere

Na rezultat je utjecalo što je stvarna visina detektiranog kamiona manja od prosječne stvarne visine za klasu kamion i autobus. Samo je jedan kamion korišten za snimanje video sekvenci za klasu kamion i autobus. S obzirom na navedene nedostatke rješenje je uspjelo zadovoljavajuće, uz ne preveliku grešku, procijeniti sve tri tražene veličine. Zbog potrebe za uključivanjem alarmnog sustava ispisuje se poruka „*ALERT!*“ na video sekvencama ako je vrijednost TTC-a manja od tri sekunde.

(a)

(b)

(c)

(d)

(e)

Slika 4.2. Rezultati rada rješenja za video sekvencu truck1_30.mp4, stvarna brzina 30 km/h, kamion na (a) 25 m, (b) 20 m, (c) 15 m, (d) 10 m, (e) 5 m udaljenosti od kamere

Slika 4.3. prikazuje rezultate za video sekvencu za klasu pješak koji se kreće prosječnom brzinom između 5 km/h i 15 km/h. Prilikom snimanja korištena je aplikacija za mjerenje brzine

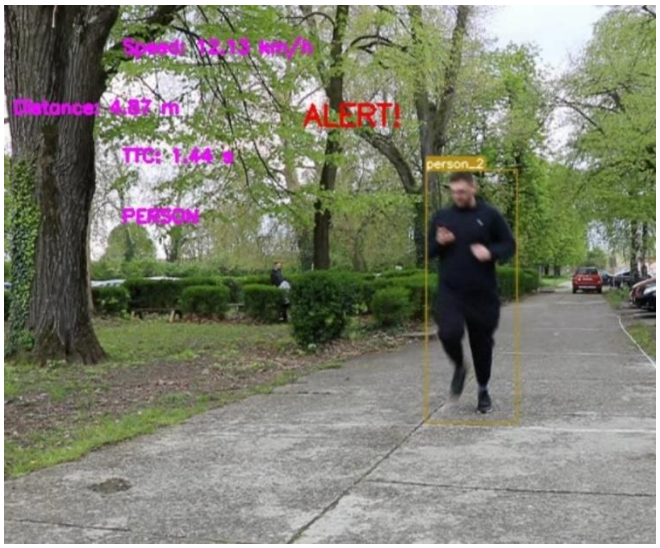
koja koristi GPS (engl. *Global Positioning System*) sustav koji određuje relevantnu brzinu objekta na temelju informacije koliku je udaljenost prešao u određenom vremenu. Opet se javlja nedostatak u visina graničnog pravokutnika detektiranog pješaka. S obzirom na navedene nedostatke rješenje je uspjelo bez prevelike greške procijeniti stvarnu udaljenost, brzinu pješaka i TTC.

(a)

(b)

(c)

(d)



(e)

Slika 4.3. Rezultati rada rješenja za video sekvencu person5_15.mp4, stvarna brzina od 5 km/h do 15 km/h, pješak na (a) 25 m, (b) 20 m, (c) 15 m, (d) 10 m, (e) 5 m udaljenosti od kamere

Slika 4.4. prikazuje rezultate za video sekvencu za klasu biciklist koji se kreće prosječnom stvarnom brzinom od 10 km/h koja je izmjerena kao prosječna brzina biciklista pomoću aplikacije. Odabrani kut snimanja, za razliku od prethodnih video sekvenci, je bolje odabran i visina graničnog pravokutnika detektiranog biciklista je preciznije određena. Rješenje je uspjelo procijeniti stvarnu udaljenost i brzinu biciklista te TTC.

(a)

(b)

(c)

(d)

(e)

Slika 4.4. Rezultati rada rješenja za video sekvencu cyclist1_10.mp4, stvarna brzina 10 km/h, biciklist na (a) 25 m, (b) 20 m, (c) 15 m, (d) 10 m, (e) 5 m udaljenosti od kamere

Slika 4.5. prikazuje rezultate za video sekvencu za klasu motociklist koji se kreće stvarnom brzinom od 30 km/h, no kako motocikl nema *tempomat*, zasigurno u nekim trenucima ne vozi točno tom brzinom. Kut snimanja je dobro postavljen te ne utječe na ishode udaljenosti, brzine i procjene TTC-a. Visina graničnog pravokutnika na navedenim udaljenostima je manja, što je utjecalo na procjenu udaljenosti koja je veća. Na udaljenosti pet metara od kamere motociklist je počeo s kočenjem, što je utjecalo na stvarnu brzinu te ujedno i na procijenjenu brzinu. Greška kod procjene udaljenosti od kamere je utjecala na procjenu brzine jer odstupa od stvarne udaljenosti od kamere, a to se događa zbog postavljene prosječne stvarne visina motociklista koja je postavljena na 1.27 *m*. Motociklist je veći od prosjeka te utječe na rezultate procijenjenih udaljenosti odnosno procijenjene udaljenosti su manje što je vidljivo na slici 4.5. (c), (d) i (e). S obzirom na navedene nedostatke rješenje je uspjelo bez prevelike greške procijeniti stvarnu udaljenost, brzinu motocikla i TTC i također upozoriti vozača na potencijalnu koliziju s vratima vozila.

(a)

(b)

(c)

(d)

(e)

Slika 4.5. Rezultati rada rješenja za video sekvencu motorcycle-1_30.MP4, stvarna brzina 30 km/h, motociklist na (a) 25 m, (b) 20 m, (c) 15 m, (d) 10 m, (e) 5 m udaljenosti od kamere

Cjelokupni rezultati procjene udaljenosti rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutih na osobnom računalu prikazani su u tablici 4.1.

Tablica 4.1. Rezultati procjene udaljenosti rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na osobnom računalu

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Procijenjena udaljenost na 25 m od kamere [m]	Procijenjena udaljenost na 20 m od kamere [m]	Procijenjena udaljenost na 15 m od kamere [m]	Procijenjena udaljenost na 10 m od kamere [m]	Procijenjena udaljenost na 5 m od kamere [m]
20	truck1_20.mp4	25.32	20.32	15.6	10.03	5.38
30	truck1_30.mp4	24.91	20.06	14.57	10.03	5.99
40	truck1_40.mp4	22.71	19.55	14.43	9.08	6.35
50	truck1_50.mp4	25.74	19.07	14.99	10.58	5.38
20	car1_20.mp4	25.33	20.27	15.01	10.26	5.71
30	car1_30.mp4	21.22	18.06	13.69	8.94	5.02
40	car1_40.mp4	23.58	19.29	13.26	9.33	5.05
50	car1_50.mp4	24.25	19.29	13.06	8.01	4.35
20	car20.mp4	22.94	18.86	14.89	9.65	5.81
30	car30.mp4	22.94	18.45	13.26	9.76	5.14
40	car40.mp4	25.72	19.74	14.89	9.65	5.77
50	car50.mp4	24.97	20.71	13.92	9.76	5.70
20	car3_20.mp4	25.72	19.29	15.43	10.8	5.70
30	car3_30.mp4	23.58	20.21	15.72	10.88	5.74
40	car3_40.mp4	24.97	22.34	16.02	9.99	4.47
50	car3_50.mp4	24.25	20.21	15.72	11.96	5.18
20	car4_20.mp4	24.97	19.29	14.64	10.11	5.98
30	car4_30.mp4	25.72	20.71	14.89	9.65	5.51
40	car4_40.mp4	25.72	19.74	15.16	11.47	5.51
50	car4_50.mp4	25.73	19.29	15.33	10.87	5.62
20	motorcycle-1_20.MP4	24.55	20.45	14.44	10.99	5.41
30	motorcycle-1_30.MP4	27.28	23.76	13.64	7.85	4.23
40	motorcycle-1_40.MP4	24.55	19.38	12.27	7.75	5.56
50	motorcycle-1_50.MP4	24.44	18.41	14.16	7.59	4.72
10	cyclist1_10.mp4	25.52	20.73	15.08	10.26	5.32
10	cyclist2_10.mp4	24.88	20.31	15.08	10.05	5.53
10	cyclist3_10.mp4	22.62	19.51	13.67	9.76	4.95
11	cyclist4_11.mp4	22.12	19.51	15.08	10.59	5.96
19	cyclist5_19.mp4	25.52	20.31	15.55	9.95	5.85
4-5	person1_4-5.mp4	24.11	19.94	15.18	10.27	4.94
4-5	person2_4-5.mp4	24.81	19.94	15.41	10.07	4.69
13-15	person3_13-15.mp4	24.22	21.19	13.74	10.17	4.84
5	person4_5.mp4	22.11	19.19	13.74	8.77	4.48

15	person5_15.mp4	25.43	19.19	13.74	9.51	4.87

Rješenje je uspjelo prepoznati sve sudionike prometa u video sekvencama. Na svim udaljenostima od 25 do 5 metara tj. na pet udaljenosti i 34 video sekvence izvršeno je 170 ispravnih detekcija. Zbog aproksimacije tijekom izračuna udaljenosti, procijenjene udaljenosti razlikuju se od stvarne udaljenosti sudionika prometa od kamere i prikazane su u obliku postotne pogreške za svaku klasu u tablici 4.2.

Tablica 4.2. Rezultati postotne pogreške procijenjene udaljenosti rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na osobnom računalu

KLASA	Postotak pogreške udaljenosti na 25 m od kamere [%]	Postotak pogreške udaljenosti na 20 m od kamere [%]	Postotak pogreške udaljenosti na 15 m od kamere [%]	Postotak pogreške udaljenosti na 10 m od kamere [%]	Postotak pogreške udaljenosti na 5 m od kamere [%]
AUTOMOBIL	od -9% do +3%	od -5% do +2%	od -4% do +4%	od -9% do +6%	od +8% do +27%
KAMION I AUTOBUS	od -15% do +3%	od -10% do +12%	od -13% do +7%	od -20% do +20%	od -13% do +20%
PJEŠAK	od -2% do +9%	od -8% do +19%	od -18% do -4%	od -24% do +10%	od -15% do +11%
BICIKLIST	od -12% do +2%	od -2% do +4%	od -9% do +4%	od -2% do +6%	od -1% do +19%
MOTOCIKLIST	od -16% do +2%	od -4% do +6%	od -8% do +3%	od -12% do +3%	od -10% do -1%

Analizirajući tablicu 4.2. u smjeru gore-dolje (različite klase) i lijevo-desno (različite udaljenosti), vidljivo je da postotne pogreške udaljenosti variraju prema klasama i udaljenostima, što ukazuje na različite razine preciznosti za različite uvjete. Iz rezultata je vidljivo da je najmanji raspon postotne pogreške udaljenosti kod klase pješak. Rasponi postotne pogreške udaljenosti prilično su mali, posebno na udaljenostima od 15 m i 10 m. To ukazuje na relativno preciznu procjenu udaljenosti za pješake. Najveći raspon postotne pogreške udaljenosti je za klasu kamion i autobus. Navedena klasa ima najveći raspon postotne pogreške udaljenosti na većim udaljenostima (25 m i 20 m). To ukazuje da rješenje ima veće teškoće u preciznoj procjeni udaljenosti za navedenu klasu na većim udaljenostima. Na temelju analize rezultata, može se reći da rješenje najbolje procjenjuje udaljenost za klasu pješak. Rasponi postotne pogreške udaljenosti za ovu klasu su relativno mali na svim udaljenostima. Za klasu automobil na većim udaljenostima (25 m i 20 m) postotak pogreške ima manji raspon, što ukazuje na to da rješenje bolje radi na većim udaljenostima. Vrijednosti postotne pogreške udaljenosti relativno su bliže jedna drugoj na većim udaljenostima te to ukazuje na veću dosljednost i preciznost u procjeni udaljenosti vozila. Za klasu kamion i autobus rezultati prikazuju blagi porast postotne pogreške na udaljenosti 5 m od kamere. To je povezano s kompaktnim izgledom tih vozila na manjim udaljenostima, tj. na rezultate je utjecalo precizno određivanje graničnog pravokutnika. Veća postotna pogreška na

srednjim udaljenostima (15 m i 10 m) i najmanja postotna pogreška na najbližoj udaljenosti (5 m) ukazuje na izazove u detektiranju specifičnih karakteristika pješaka na većim udaljenostima. Postotak pogreške na najbližoj udaljenosti manji je jer je detekcija pješaka često lakša na kraćim rasponima gdje je veći kontrast između pješaka i okoline. Za klasu biciklist rezultati ukazuju na relativno niske postotne pogreške na većim udaljenostima, te rješenje daje bolje rezultate na većim udaljenostima. Za klasu motociklist povećanje postotne pogreške na većim udaljenostima (25 m i 20 m) uzrokovano je sličnošću motociklista s okolinom na tim udaljenostima, što stvara izazov u preciznom određivanju graničnog pravokutnika.

Pri udaljenosti 25 m od kamere za video sekvencu *car1_30.mp4* procijenjena udaljenost je manja od stvarne zbog visine graničnog pravokutnika detektiranog automobila koja je viša te je procijenjena udaljenost manja. Također za istu video sekvencu procijenjena udaljenost na 20 m od kamere je također manja od stvarne udaljenosti. Na tu procijenjenu udaljenosti je isto utjecao granični pravokutnika koji treba biti manji. Za video sekvencu *car1_50.mp4* procijenjena udaljenost je manja od stvarne udaljenosti na 10 m od kamere pri brzini 50km/h. Kamera je postavljena u pravcu iz kojeg automobil dolazi. Razlika u procijenjenoj udaljenosti se najviše vidi kod klase motociklist jer prosječna visina ne odgovara njegovoj stvarnoj visini. Video sekvence su snimljene samo s jednim motociklistom te je to razlog zašto rezultati odstupaju od stvarnih. Za klasu motociklist postotak pogrešne procjene udaljenosti je većinom negativan, što upućuje na to da su procijenjene udaljenosti manje od stvarnih udaljenosti. Navedeni rezultati vidljivi su i u tablici 4.1. za klasu motociklist pri brzinama 30 i 40 km/h na udaljenosti manjoj od 15 m od kamere. Na temelju rezultata postotne pogreške, prosječna visina motociklista ne odgovara stvarnoj visini snimanog motociklista te ju je potrebno povećati kako bi se povećala procijenjena udaljenosti. Također za klasu motociklist pri udaljenosti 20 m od kamere i brzini 30 km/h odstupa procijenjena udaljenosti odnosno ona je precijenjena. Na slici 4.5. (b) je vidljivo da je granični pravokutnik detektiranog motociklista manji te je utjecao na precijenjenu udaljenost. Za klasu biciklist odnosno za video sekvencu *cyclist3_10.mp4* procijenjena udaljenost je manja od stvarne udaljenosti na 25 m od kamere. Visina graničnog pravokutnika odgovara visini detektiranog biciklista u video sekvenci te pogreška koja je utjecala na manju udaljenost je stvarna visina detektiranog biciklista koja je viša od prosječne definirane za klasu biciklist. Pri istoj stvarnoj udaljenosti za video sekvencu *cyclist4_11.mp4* procijenjena udaljenost je manja, a to je rezultat višeg graničnog pravokutnika detektiranog biciklista. Za video sekvencu *person4_5.mp4* procijenjena udaljenost je manja od stvarne udaljenosti na 25 m od kamere. Na pogrešnu procjenu

je utjecala stvarna visina pješaka koja je viša od prosječne visine te rezultirala manjom procijenjenom udaljenosti.

U tablici 4.3. nalaze se rezultati procjene brzine sudionika prometa rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila. Rezultati postotne pogreške procijenjene brzine sudionika prometa nalaze se u tablici 4.4.

Tablica 4.3. Rezultati procjene brzine sudionika prometa rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na osobnom računalu

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Procijenjena brzina na 25 m od kamere [km/h]	Procijenjena brzina na 20 m od kamere [km/h]	Procijenjena brzina na 15 m od kamere [km/h]	Procijenjena brzina na 10 m od kamere [km/h]	Procijenjena brzina na 5 m od kamere [km/h]
20	truck1_20.mp4	20.93	15.59	20.59	20.93	17.33
30	truck1_30.mp4	34.91	34.91	34.91	31.39	31.39
40	truck1_40.mp4	34.09	45.02	40.33	33.87	39.14
50	truck1_50.mp4	52.50	49.56	51.31	42.72	43.08
20	car1_20.mp4	18.77	19.38	18.77	18.76	18.56
30	car1_30.mp4	41.44	26.70	21.36	30.08	29.14
40	car1_40.mp4	45.53	40.32	25.09	36.66	39.92
50	car1_50.mp4	58.6	46.22	42.79	41.2	52.48
20	car20.mp4	20.90	24.16	19.63	22.72	14.49
30	car30.mp4	54.63	38.37	22.33	32.18	29.33
40	car40.mp4	44.19	35.39	36.14	42.21	27.31
50	car50.mp4	48.37	40.04	35.04	48.20	46.63
20	car3_20.mp4	16.61	21.8	24.04	22.04	20.08
30	car3_30.mp4	35.98	30.07	28.92	30.58	31.63
40	car3_40.mp4	44.02	39.76	34.20	38.09	39.29
50	car3_50.mp4	50.82	41.36	50.64	53.76	41.95
20	car4_20.mp4	20.79	20.79	18.54	19.07	20.79
30	car4_30.mp4	31.50	34.67	34.58	34.58	34.58
40	car4_40.mp4	52.91	49.75	41.51	48.27	39.28
50	car4_50.mp4	62.18	55.76	45.90	42.60	42.39
20	motorcycle-1_20.MP4	22.26	22.51	28.22	16.19	14.59
30	motorcycle-1_30.MP4	27.20	27.20	30.50	36.70	26.07
40	motorcycle-1_40.MP4	46.02	42.13	44.67	33.31	43.79
50	motorcycle-1_50.MP4	46.04	57.88	45.99	38.03	34.33
10	cyclist1_10.mp4	10.64	10.64	10.73	10.73	10.73
10	cyclist2_10.mp4	10.45	10.43	10.43	10.43	10.42
10	cyclist3_10.mp4	12.24	10.08	14.87	14.56	10.08
11	cyclist4_11.mp4	12.71	13.14	20.41	22.36	22.83
19	cyclist5_19.mp4	13.14	11.5	23.36	20.33	13.5
4-5	person1_4-5.mp4	4.65	4.65	4.65	3.94	4.47

4-5	person2_4-5.mp4	4.65	4.65	4.65	4.25	3.45
13-15	person3_13-15.mp4	15.51	11.64	13.51	11.08	13.58
5	person4_5.mp4	5.40	7.17	5.39	5.40	5.75
15	person5_15.mp4	12.79	20.85	18.83	17.10	12.13

Tablica 4.4. Rezultati postotne pogreške procijenjene brzine sudionika prometa rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na osobnom računalu

KLASA	Postotak pogreške brzine na 25 m od kamere [%]	Postotak pogreške brzine na 20 m od kamere [%]	Postotak pogreške brzine na 15 m od kamere [%]	Postotak pogreške brzine na 10 m od kamere [%]	Postotak pogreške brzine na 5 m od kamere [%]
AUTOMOBIL	od -17% do +45%	od -20% do +28%	od -37% do +20%	od -18% do +21%	od -32% do +15%
KAMION I AUTOBUS	od -15% do +16%	od -22% do +16%	od +1% do +16%	od -15% do +5%	od -14% do +5%
PJEŠAK	od -15% do +31%	od -17% do +39%	od -7% do +26%	od -21% do +14%	od -31% do +13%
BICIKLIST	od -31% do +18%	od -39% do +16%	od +4% do +46%	od +4% do +51%	od -29% do +52%
MOTOCIKLIST	od -9% do +15%	od -9% do +16%	od -8% do +41%	od -24% do +22%	od -31% do +9%

Analizirajući tablicu 4.4. u smjeru gore-dolje (različite klase) i lijevo-desno (različite udaljenosti) vidljivo je da postotne pogreške brzine variraju prema klasama i udaljenostima, što ukazuje na različite razine preciznosti za različite uvjete. Najmanji raspon postotne pogreške brzine pojavljuje se za klasu pješak. To ukazuje na relativno dosljedne rezultate u procjeni brzine pješaka na različitim udaljenostima. Ovo može biti posljedica manje varijabilnosti u obliku i veličini pješaka u video okvirima detekcije. Najveći raspon postotne pogreške brzine javlja se za klasu biciklist. To ukazuje da je oblik biciklista teže prepoznatljiv i interpretiran, što rezultira većom varijabilnošću u rezultatima. Analizom rezultata rješenje najbolje procjenjuje brzinu za klasu pješak, zato što ispravno detektira pješake u video okvirima te su pješaci lakši za praćenje. Rezultati variraju ovisno o klasama, ali općenito, rješenje pokazuje bolje rezultate u postotnoj procjeni pogreške brzine na većim udaljenostima (25 m i 20 m). Postotna pogreška procjene brzine za klasu automobil pokazuje značajnu varijabilnost, s rasponom koji seže od -17% do +45%. Ovi rezultati ukazuju da rješenje nije uvijek precizno u procjeni brzine vozila za ovu klasu. Rezultati za klasu kamion i autobus također variraju, ali su manje varijabilni nego za klasu automobil. Ovo ukazuje da rješenje ima bolje performanse u procjeni brzine većih vozila, odnosno za klasu kamion i autobus. Postotna pogreška procjene brzine za klasu pješak varira prema udaljenostima, s najmanjom varijabilnošću na udaljenosti 15 m od kamere. Klasa biciklist pokazuju najveću varijabilnost postotne pogreške procjene brzine, s rasponima koji su posebno veliki na udaljenostima od 10 m i 5 m. Ukazuje da rješenje može imati izazove u preciznoj procjeni brzine biciklista na tim udaljenostima i da se rezultati mogu značajno razlikovati. Postotna pogreška procjene brzine za klasu motociklist ne varira u usporedbi s drugim klasama, a najmanja

varijabilnost je na udaljenosti 25 m od kamere. Ovi rezultati ukazuju na stabilnije performanse rješenja u procjeni brzine motociklista na toj udaljenosti.

Procijenjene brzine ne odstupaju mnogo od stvarnih brzina sudionika prometa, ako dodamo potencijalne pogreške brzinomjera koje su navedene u potpoglavlju 4.1. Ne može se nikad garantirati da se objekt kreće upravo željenom brzinom zbog niza faktora koji mogu na to utjecati.

U tablici 4.4. za klasu autobus i kamion prikazana je postotna pogreška procijenjene brzine na udaljenosti 15 m od kamere. Prikazuje da su sve procijenjene brzine veće od stvarnih odnosno pogreška iznosi od +1% do +16%. Procijenjena brzina kretanja na udaljenosti 10 m od kamere za video sekvencu *truck1_40.mp4* je manja od stvarne brzine kamiona koja je 40 km/h. Procijenjena brzina kamiona je zadovoljavajuća s obzirom na vjerojatnu pogrešku prikazivanja brzine na brzinomjeru. Procijenjena brzina na 25 m od kamere za video sekvencu *car1_30.mp4* je veća od stvarne brzine automobila koja je 30 km/h zbog krivo procijenjene udaljenosti. Za istu video sekvencu na udaljenosti 15 m od kamere procijenjena brzina kretanja je manja od stvarne. Za video sekvencu *car1_40.mp4* procijenjena brzina na 15 m od kamere je manja od stvarne brzine 40 km/h. Na procijenjenu brzinu je utjecala oscilacija procijenjenih brzina odnosno, zbog lošeg detektiranja iz okvira u okvir dimenzije graničnog okvira su se drastičnije mijenjale. Procijenjena brzina na 25 m od kamere je veća od stvarne brzine 30 km/h za video sekvencu *car30.mp4*. Pri snimanju navedene video sekvence *tempomat* nije korišten jer nije dio opreme snimanog automobila, što je rezultiralo da se automobil vjerojatno kretao brzinom većom od 30km/h i utjecalo je na netočnost pri procijene brzine. Za video sekvencu *motorcycle-1_50.MP4* procijenjena brzina na 5 m od kamere je manja od stvarne brzine 50 km/h. Na rezultat pogrešno procijenjene brzine motociklista je utjecala procijenjena udaljenost. Procijenjena brzina biciklista na 20 m od kamere je manja od prosječne stvarne brzine za video sekvencu cyclist5_19.mp4. U tablici 4.4. postotne pogreške procijenjene brzine su od -39% do +16% za klasu biciklist na navedenoj udaljenosti 20 m od kamere. Do oscilacija dolazi i zbog potencijalno varijabilne brzine biciklista tijekom snimanja video sekvenci.

U tablici 4.5. prikazane su referentne odnosno stvarne vrijednosti TTC-a s obzirom na stvarnu udaljenost i brzinu za svaku klasu. Rezultati procjene TTC-a rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila prikazani su u tablici 4.6. Budući da je TTC izveden iz omjera udaljenosti sudionika prometa koji se približava i njegove brzine, netočnosti u procjeni tih parametara utječu su za netočnost procjene TTC-a. Vozač je svjestan koliko vremena ima za izlazak iz vozila na temelju procjene TTC-a. U svim bi slučajevima sustav je uspješno obavijestio vozača za moguću potencijalnu koliziju s vratima vozila.

Tablica 4.5. Stvarni TTC za svaku klasu

KLASA	BRZINA [km/h]	Stvarni TTC na udaljenosti 25 m od kamere [s]	Stvarni TTC na udaljenosti 20 m od kamere [s]	Stvarni TTC na udaljenosti 15 m od kamere [s]	Stvarni TTC na udaljenosti 10 m od kamere [s]	Stvarni TTC na udaljenosti 5 m od kamere [s]
VOZILA (AUTOMOBIL, KAMION I AUTOBUS, MOTOCIKL)	20 km/h	4.50	3.60	2.70	1.80	0.90
VOZILA (AUTOMOBIL, KAMION I AUTOBUS, MOTOCIKL)	30 km/h	3.00	2.40	1.80	1.20	0.60
VOZILA (AUTOMOBIL, KAMION I AUTOBUS, MOTOCIKL)	40 km/h	2.25	1.80	1.35	0.90	0.45
VOZILA (AUTOMOBIL, KAMION I AUTOBUS, MOTOCIKL)	50 km/h	1.80	1.44	1.08	0.72	0.36
BICIKLIST	10 km/h	9.00	7.20	5.40	3.60	1.80
BICIKLIST	11 km/h	8.18	6.55	4.91	3.27	1.64
BICIKLIST	19 km/h	4.74	3.79	2.84	1.89	0.95
PJEŠAK	4 km/h	22.50	18.00	13.50	9.00	4.50
PJEŠAK	14 km/h	6.43	5.14	3.86	2.57	1.29
PJEŠAK	15 km/h	6.00	4.80	3.60	2.40	1.20

Tablica 4.6. Rezultati procjene TTC-a rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na osobnom računalu

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Procijenjen TTC na 25 m od kamere [s]	Procijenjen TTC na 20 m od kamere [s]	Procijenjen TTC na 15 m od kamere [s]	Procijenjen TTC na 10 m od kamere [s]	Procijenjen TTC na 5 m od kamere [s]
20	truck1_20.mp4	4.35	4.69	2.73	1.72	1.12
30	truck1_30.mp4	2.87	2.53	1.46	1.18	0.78
40	truck1_40.mp4	2.4	1.56	1.29	0.97	0.68
50	truck1_50.mp4	1.78	1.38	1.05	0.89	0.53

20	car1_20.mp4	4.88	3.76	2.83	1.97	1.11
30	car1_30.mp4	1.84	2.44	2.31	1.07	0.62
40	car1_40.mp4	1.86	1.72	1.69	0.92	0.46
50	car1_50.mp4	1.49	1.5	1.1	0.7	0.3
20	car20.mp4	3.95	2.81	2.73	1.53	1.44
30	car30.mp4	1.51	1.73	2.14	1.09	0.63
40	car40.mp4	2.1	2.01	1.43	0.82	0.76
50	car50.mp4	1.86	1.86	1.43	0.73	0.44
20	car3_20.mp4	5.58	3.19	2.31	1.75	1.02
30	car3_30.mp4	2.36	2.42	1.96	1.28	0.65
40	car3_40.mp4	2.04	2.02	1.75	0.94	0.41
50	car3_50.mp4	1.72	1.76	1.12	0.8	0.6
20	car4_20.mp4	4.32	3.34	2.84	1.91	1.04
30	car4_30.mp4	2.63	2.12	1.52	0.9	0.52
40	car4_40.mp4	1.75	1.43	1.31	0.86	0.59
50	car4_50.mp4	1.49	1.06	1.28	0.83	0.48
20	motorcycle-1_20.MP4	3.97	3.27	1.84	2.44	1.34
30	motorcycle-1_30.MP4	4.37	3.81	1.49	0.73	0.44
40	motorcycle-1_40.MP4	1.92	1.66	0.99	0.84	0.46
50	motorcycle-1_50.MP4	1.92	1.15	1.11	0.72	0.49
10	cyclist1_10.mp4	8.81	7.15	6.01	3.54	1.85
10	cyclist2_10.mp4	8.57	7.01	5.2	3.47	2.01
10	cyclist3_10.mp4	6.65	6.17	3.3	2.41	1.77
11	cyclist4_11.mp4	6.27	5.35	2.66	1.7	0.94
19	cyclist5_19.mp4	6.99	6.36	2.4	1.76	1.56
4-5	person1_4-5.mp4	17.11	15.43	11.75	9.39	3.98
4-5	person2_4-5.mp4	19.2	15.45	11.93	8.53	4.89
13-15	person3_13-15.mp4	5.62	6.56	3.19	3.3	1.29
5	person4_5.mp4	14.74	9.54	9.19	5.86	2.8
15	person5_15.mp4	7.13	3.43	2.72	2.28	1.8

Rezultati procijenjenog TTC-a za vozila kod vrijednosti za veće brzine (30 km/h, 40 km/h i 50 km/h) obično su manje od stvarnih TTC vrijednosti, što ukazuje da rješenje predviđa da će se potencijalna kolizija s vozačevim vratima dogoditi ranije nego što to stvarno jest. To je bolji slučaj nego da rješenje procjenjuje drugačije jer će ostaviti vozaču manje vremena da napravi pogrešku. Za brzine 10 km/h i 11 km/h kod klase biciklist, procijenjene TTC vrijednosti su obično manje od stvarnih vrijednosti, što znači da rješenje predviđa potencijalnu koliziju s vozačevim vratima ranije nego što se stvarno događa. Međutim, za brzinu 19 km/h, procijenjene TTC vrijednosti su veće od stvarnih, što ukazuje da rješenje predviđa potencijalnu koliziju s vozačevim vratima kasnije nego što se stvarno događa. Na većim udaljenostima procijenjeni TTC je veći od stvarnog jer rješenje nije uspjelo precizno procijeniti brzinu biciklista koji se kreću brzinom od 19 km/h. Za brzine od

4 km/h i 14 km/h kod klase pješak, procijenjene TTC vrijednosti su obično manje od stvarnih, što znači da rješenje predviđa potencijalnu koliziju s vozačevim vratima ranije nego što se stvarno događa. Za brzinu pješaka od 15 km/h, procijenjene TTC vrijednosti su bliske stvarnim.

U tablici 4.7. vidljivi su rezultati vremena potrebnog za izvršavanja rješenja u sekundama. Jasno je da se s većim brojem video okvira u video sekvenci povećava i vrijeme izvršavanja. Iz postignutog broja FPS, što je vidljivo u tablici 4.7., može se primijetiti smanjenje brzine obrade okvira. Rješenje je pokrenuto na procesoru te je to jedan od glavnih razloga dulje obrade video sekvenci. Osobno računalo sastoji se od procesora Intel Core i7-6700 s frekvencijom takta od 3.41 GHz i 16 GB radne memorije.

Tablica 4.7. Vrijeme izvršavanja rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na osobnom računalu

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Vrijeme izvršavanja [s]	Broj video okvira	Postignuti broj fps-a
20	truck1_20.mp4	155.69	117	0.75
30	truck1_30.mp4	126.54	130	1.03
40	truck1_40.mp4	244.88	281	1.15
50	truck1_50.mp4	113.39	124	1.09
20	car1_20.mp4	217.81	178	0.82
30	car1_30.mp4	311.42	212	0.68
40	car1_40.mp4	106.33	108	1.02
50	car1_50.mp4	90.47	99	1.09
20	car20.mp4	311.90	358	1.15
30	car30.mp4	205.45	218	1.06
40	car40.mp4	214.99	220	1.02
50	car50.mp4	186.89	189	1.01
20	car3_20.mp4	202.75	223	1.10
30	car3_30.mp4	189.85	194	1.02
40	car3_40.mp4	253	224	0.89
50	car3_50.mp4	267.04	278	1.04
20	car4_20.mp4	301.51	334	1.11
30	car4_30.mp4	151.78	159	1.05
40	car4_40.mp4	250.01	216	0.86
50	car4_50.mp4	269.70	276	1.02
20	motorcycle-1_20.MP4	270.70	263	0.97
30	motorcycle-1_30.MP4	400.49	268	0.67
40	motorcycle-1_40.MP4	275.22	279	1.01
50	motorcycle-1_50.MP4	117.24	114	0.97
10	cyclist1_10.mp4	179.15	182	1.02
10	cyclist2_10.mp4	243.27	268	1.10

10	cyclist3_10.mp4	232.03	251	1.08
11	cyclist4_11.mp4	113.52	116	1.02
19	cyclist5_19.mp4	391.79	437	1.12
4-5	person1_4-5.mp4	327.99	336	1.02
4-5	person2_4-5.mp4	349.21	392	1.12
13-15	person3_13-15.mp4	133.67	139	1.04
5	person4_5.mp4	305.99	345	1.13
15	person5_15.mp4	214.39	223	1.04

4.3. Evaluacija točnosti i brzina rada razvijenog rješenja pokrenutog na Raspberry Pi 4 platformi

Rezultati procjene udaljenosti rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenuti na Raspberry Pi 4 platformi prikazani su u tablici 4.8., a koristile su se iste video sekvence kao i na osobnom računalu. Algoritam za detekciju nije uspio prepoznati sve sudionike prometa u video sekvencama odnosno na udaljenosti 5m za klasu kamion i na udaljenosti 25m za klase pješak i biciklist zato što se koristio YOLOv7 *Tiny* model kojeg bi trebalo još bolje istrenirati dodavanjem više slika. Zbog aproksimacije tijekom izračuna udaljenosti, procijenjene udaljenosti razlikuju se od stvarne udaljenosti sudionika prometa od kamere prikazane u obliku postotne pogreške udaljenosti u tablici 4.9. Najveće odstupanje od stvarne udaljenosti je pri udaljenosti 15 m od kamere za klasu pješak od -42% do +2% postotne pogreške. Prikazuje da je procijenjena udaljenost manja od stvarne udaljenosti. Za klasu automobil pri udaljenosti 5 m od kamere postotna pogreška udaljenosti iznosi od -7% do +28% te predočava da je procijenjena udaljenost veća od stvarne udaljenosti. Svi problemi navedeni u poglavlju 4.2. također vrijede i za rezultate procjene udaljenosti rješenja pokrenutog na Raspberry Pi 4 platformi.

Tablica 4.8. Rezultati procjene udaljenosti rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Procijenjena udaljenost na 25 m od kamere [m]	Procijenjena udaljenost na 20 m od kamere [m]	Procijenjena udaljenost na 15 m od kamere [m]	Procijenjena udaljenost na 10 m od kamere [m]	Procijenjena udaljenost na 5 m od kamere [m]
20	truck1_20.mp4	27.04	23.78	18.15	10.95	/
30	truck1_30.mp4	25.08	18.64	16.42	10.53	/
40	truck1_40.mp4	24.63	19.55	14.43	9.08	6.35
50	truck1_50.mp4	25.74	19.07	14.99	10.58	5.38
20	car1_20.mp4	25.72	20.21	15.43	11.17	5.70
30	car1_30.mp4	24.97	21.22	15.72	10.48	5.21
40	car1_40.mp4	24.25	19.29	16.33	12.48	5.34
50	car1_50.mp4	24.97	19.74	15.72	9.87	5.41
20	car20.mp4	24.25	20.71	16.02	11.02	5.02

30	car30.mp4	24.97	20.21	15.43	10.35	5.24
40	car40.mp4	24.97	19.74	16.65	9.99	5.27
50	car50.mp4	21.77	18.06	17.32	9.99	5.41
20	car3_20.mp4	25.72	21.22	16.02	10.75	5.74
30	car3_30.mp4	24.98	20.31	16.33	11.02	5.02
40	car3_40.mp4	23.58	20.21	15.72	10.88	5.70
50	car3_50.mp4	24.97	20.49	16.46	11.02	5.37
20	car4_20.mp4	24.25	21.22	14.89	11.17	5.48
30	car4_30.mp4	24.25	20.71	16.02	10.88	4.66
40	car4_40.mp4	24.25	20.21	15.72	9.65	5.66
50	car4_50.mp4	25.72	21.22	15.43	11.17	6.96
20	motorcycle-1_20.MP4	22.32	16.37	13.15	9.44	5.66
30	motorcycle-1_30.MP4	23.76	16.01	12.70	9.44	4.98
40	motorcycle-1_40.MP4	24.55	21.66	13.89	8.87	5.08
50	motorcycle-1_50.MP4	23.76	19.90	12.48	7.67	5.19
10	cyclist1_10.mp4	24.27	20.73	15.08	10.59	5.56
10	cyclist2_10.mp4	25.52	20.31	15.80	9.95	5.53
10	cyclist3_10.mp4	21.18	19.51	14.22	9.76	4.74
11	cyclist4_11.mp4	/	18.08	15.08	10.82	4.98
19	cyclist5_19.mp4	23.15	19.14	14.85	11.71	6.11
4-5	person1_4-5.mp4	22.11	19.94	14.13	9.87	5.27
4-5	person2_4-5.mp4	24.22	17.54	15.18	8.77	5.38
13-15	person3_13-15.mp4	/	20.73	14.53	10.38	4.82
5	person4_5.mp4	21.76	18.49	8.69	8.48	4.78
15	person5_15.mp4	/	17.24	14.74	9.6	4.78

Tablica 4.9. Rezultati postotne pogreške udaljenosti rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi

KLASA	Postotak pogreške udaljenosti na 25 m od kamere [%]	Postotak pogreške udaljenosti na 20 m od kamere [%]	Postotak pogreške udaljenosti na 15 m od kamere [%]	Postotak pogreške udaljenosti na 10 m od kamere [%]	Postotak pogreške udaljenosti na 5 m od kamere [%]
AUTOMOBIL	od -15% do +3%	od -11% do +6%	od -1% do +13%	od -4% do +20%	od -7% do +28%
KAMION I AUTOBUS	od -2% do +8%	od -7% do +16%	od -4% do +17%	od -10% do +9%	od +7% do +21%
PJEŠAK	od -15% do -3%	od -16% do +4%	od -42% do +1%	od -15% do +4%	od -4% do +7%
BICIKLIST	od -18% do +2%	od -11% do +4%	od -5% do +5%	od -2% do +15%	od -5% do +18%
MOTOCIKLIST	od -12% do -2%	od -25% do +8%	od -20% do -8%	od -30% do -6%	od 0% do +12%

U tablici 4.10. nalaze se rezultati procjene brzine sudionika prometa rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi. Kako rješenje nije moglo procijeniti udaljenost za klasu kamion na udaljenosti 5 m od kamere i za klase pješak i biciklist na udaljenosti 25 m od kamere, na temelju toga nije mogao procijeniti ni brzinu. Također zbog krive procjene udaljenosti rješenje je krivo procijenilo i brzinu nadolazećeg sudionika prometa. Rezultati postotne pogreške procijenjene brzine sudionika prometa nalaze su

u tablici 4.11. Najveće postotne pogreške u odstupanju od stvarne vrijednosti brzine su za klase pješak i biciklist. Visina graničnog pravokutnika koja se koristi za procjenu udaljenosti od kamere razlikuje se u okvirima pri detekciji te se odrazila na procjenu brzine sudionika prometa. Za ostale klase procijenjene brzine se nalaze u okviru stvarnih rezultata.

Tablica 4.10. Rezultati procjene brzine sudionika prometa rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Procijenjena brzina na 25 m od kamere [km/h]	Procijenjena brzina na 20 m od kamere [km/h]	Procijenjena brzina na 15 m od kamere [km/h]	Procijenjena brzina na 10 m od kamere [km/h]	Procijenjena brzina na 5 m od kamere [km/h]
20	truck1_20.mp4	25.54	25.6	26.52	22.07	/
30	truck1_30.mp4	37.79	37.92	36.83	37.74	/
40	truck1_40.mp4	48	45.02	40.33	33.87	39.14
50	truck1_50.mp4	52.5	49.56	51.31	42.72	43.08
20	car1_20.mp4	27.53	27.73	28.27	26.38	17.22
30	car1_30.mp4	28.08	27.8	32.48	34.05	32.41
40	car1_40.mp4	41.28	36.52	38.73	39.06	42.24
50	car1_50.mp4	54.53	46.08	45.85	38.69	41.05
20	car20.mp4	23.28	18.52	30.65	20.03	27.93
30	car30.mp4	36.53	28.08	30.86	28.56	33.07
40	car40.mp4	45.53	39.69	39.69	36.28	40.92
50	car50.mp4	48.54	46.08	43.12	53.24	45.74
20	car3_20.mp4	19.08	19.23	25.33	26.46	24.86
30	car3_30.mp4	36.53	28.08	31.57	29.94	27.89
40	car3_40.mp4	37.38	37.08	43.38	40.81	37.34
50	car3_50.mp4	46.08	46.08	48.53	51.85	54.87
20	car4_20.mp4	23.28	19.23	26.7	18.47	21.12
30	car4_30.mp4	36.53	29.31	31.95	33.05	27.73
40	car4_40.mp4	41.28	36.52	39.49	34.41	37.48
50	car4_50.mp4	50.28	48.81	46.08	46.08	46.08
20	motorcycle-1_20.MP4	25.42	25.4	26.59	17.61	16.17
30	motorcycle-1_30.MP4	34.43	34.43	39.31	27.26	26.61
40	motorcycle-1_40.MP4	43.43	43.43	47.09	35.29	30.6
50	motorcycle-1_50.MP4	52.43	52.43	46.62	40.14	46.89
10	cyclist1_10.mp4	9.83	9	10.05	9.83	9.83
10	cyclist2_10.mp4	9.83	9.83	11.52	9.95	9.83
10	cyclist3_10.mp4	6.33	9.83	14.74	9.83	9.83
11	cyclist4_11.mp4	/	18.54	20.88	16.15	20.6
19	cyclist5_19.mp4	12.16	9.83	9.83	13.13	17.02
4-5	person1_4-5.mp4	9.96	5.77	8.42	9.34	6.35
4-5	person2_4-5.mp4	9.11	9.12	6.11	8.83	8.62

13-15	person3_13-15.mp4	/	9.85	10.97	14.97	13.33
5	person4_5.mp4	9.11	9.11	8.59	6.69	6.23
15	person5_15.mp4	/	21.36	18.05	16.93	15.42

Tablica 4.11. Rezultati postotne pogreške procijenjene brzine sudionika prometa rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi

KLASA	Postotak pogreške brzine na 25 m od kamere [%]	Postotak pogreške brzine na 20 m od kamere [%]	Postotak pogreške brzine na 15 m od kamere [%]	Postotak pogreške brzine na 10 m od kamere [%]	Postotak pogreške brzine na 5 m od kamere [%]
AUTOMOBIL	od -8% do +38%	od -9% do +39%	od -16% do +29%	od -29% do +24%	od -18% do +40%
KAMION I AUTOBUS	od +5% do +28%	od -1% do +28%	od +1% do +25%	od -18% do +21%	od -14% do -2%
PJEŠAK	od 0% do +50%	od -42% do +45%	od -22% do +42%	od +6% do +46%	od -5% do +42%
BICIKLIST	od -37% do 0%	od -48% do +69%	od -48% do +47%	od -45% do +32%	od -10% do +47%
MOTOCIKLIST	od +5% do +27%	od +5% do +27%	od -7% do +25%	od -25% do -10%	od -24% do -6%

U tablici 4.12. nalaze se rezultati procjene TTC-a rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi. Na rezultate TTC-a su utjecale pogrešno procijenjene udaljenosti i brzine. Alarm se aktivira pri TTC-u manjem od tri sekunde kao i kod pokretanja rješenja na osobnom računalu. Osim na onim udaljenostima gdje sudionik prometa nije detektiran zbog lošeg istreniranog YOLOv7 *Tiny* modela.

Tablica 4.12. Rezultati procjene TTC-a rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Procijenjen TTC na 25 m od kamere [s]	Procijenjen TTC na 20 m od kamere [s]	Procijenjen TTC na 15 m od kamere [s]	Procijenjen TTC na 10 m od kamere [s]	Procijenjen TTC na 5 m od kamere [s]
20	truck1_20.mp4	3.61	3.34	2.46	1.79	/
30	truck1_30.mp4	2.39	1.77	1.6	1	/
40	truck1_40.mp4	1.85	1.56	1.29	0.97	0.68
50	truck1_50.mp4	1.78	1.38	1.05	0.89	0.53
20	car1_20.mp4	3.36	2.62	1.97	1.52	1.19
30	car1_30.mp4	3.2	2.75	1.74	1.11	0.58
40	car1_40.mp4	2.12	1.9	1.52	1.15	0.46
50	car1_50.mp4	1.65	1.54	1.23	0.92	0.47
20	car20.mp4	3.75	4.02	1.88	1.98	0.65
30	car30.mp4	2.46	2.59	1.8	1.31	0.57
40	car40.mp4	1.97	1.79	1.51	0.99	0.46
50	car50.mp4	1.61	1.41	1.45	0.68	0.43
20	car3_20.mp4	4.85	3.97	2.28	1.46	0.83

30	car3_30.mp4	2.46	2.65	1.86	1.33	0.65
40	car3_40.mp4	2.27	1.96	1.3	0.96	0.53
50	car3_50.mp4	1.95	1.62	1.12	0.77	0.35
20	car4_20.mp4	3.75	3.97	2.01	2.18	0.93
30	car4_30.mp4	2.39	2.54	1.8	1.19	0.61
40	car4_40.mp4	2.12	1.99	1.43	1.01	0.54
50	car4_50.mp4	1.74	1.57	1.21	0.87	0.54
20	motorcycle-1_20.MP4	3.16	2.37	1.78	1.93	1.34
30	motorcycle-1_30.MP4	2.48	1.67	1.17	1.26	0.67
40	motorcycle-1_40.MP4	2.04	1.8	1.06	0.91	0.6
50	motorcycle-1_50.MP4	1.63	1.37	0.96	0.69	0.4
10	cyclist1_10.mp4	8.89	8.29	5.4	3.88	3.04
10	cyclist2_10.mp4	9.35	7.44	4.94	3.65	2.27
10	cyclist3_10.mp4	12.04	7.15	3.47	3.57	1.74
11	cyclist4_11.mp4	/	2.78	2.6	2.41	0.87
19	cyclist5_19.mp4	6.85	7.01	5.44	3.21	1.29
4-5	person1_4-5.mp4	8.83	12.45	6.04	3.81	2.99
4-5	person2_4-5.mp4	9.57	6.93	6.2	3.58	2.25
13-15	person3_13-15.mp4	/	7.59	4.77	2.5	1.3
5	person4_5.mp4	8.04	7.31	3.64	4.56	2.76
15	person5_15.mp4	/	2.9	2.94	2.04	1.12

Tablica 4.13. prikazuje vrijeme izvršavanja rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi. Vrijeme izvršavanja je približno isto kao i vrijeme izvršavanja na osobnom računalu vidljivog u potpoglavlju 4.2. zbog korištenja YOLOv7 *Tiny* modela za detekciju. Izvršavanje rješenja na Raspberry Pi 4 platformi s YOLOv7 modelom trajalo je predugo te je zbog toga korišten YOLOv7 *Tiny* model. Iako se koristio manje zahtjevan model za detekciju, vrijeme izvršavanje je i dalje za pojedine video sekvence dosta dugo.

Tablica 4.13. Vrijeme izvršavanja rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila pokrenutog na Raspberry Pi 4 platformi

Stvarna brzina [km/h]	NAZIV VIDEO SEKVENCE	Vrijeme izvršavanja [s]	Broj video okvira	Postignuti broj fps-a
20	truck1_20.mp4	249.82	177	0.71
30	truck1_30.mp4	174.97	130	0.74
40	truck1_40.mp4	364.83	281	0.77
50	truck1_50.mp4	165.41	124	0.75
20	car1_20.mp4	348.87	268	0.77
30	car1_30.mp4	327.47	254	0.78

40	car1_40.mp4	160.53	108	0.67
50	car1_50.mp4	134.34	99	0.74
20	car20.mp4	466.39	321	0.69
30	car30.mp4	279.84	212	0.76
40	car40.mp4	285.36	220	0.77
50	car50.mp4	248.31	189	0.76
20	car3_20.mp4	288.21	223	0.77
30	car3_30.mp4	251.56	194	0.77
40	car3_40.mp4	337.52	244	0.72
50	car3_50.mp4	383.52	278	0.72
20	car4_20.mp4	431.58	334	0.77
30	car4_30.mp4	211.22	159	0.75
40	car4_40.mp4	300.96	216	0.72
50	car4_50.mp4	381.09	276	0.72
20	motorcycle-1_20.MP4	339.16	263	0.78
30	motorcycle-1_30.MP4	345.44	268	0.78
40	motorcycle-1_40.MP4	354.23	279	0.79
50	motorcycle-1_50.MP4	163.81	114	0.70
10	cyclist1_10.mp4	254.11	182	0.72
10	cyclist2_10.mp4	368.42	268	0.73
10	cyclist3_10.mp4	327.11	251	0.77
11	cyclist4_11.mp4	172.7	116	0.67
19	cyclist5_19.mp4	240.85	214	0.89
4-5	person1_4-5.mp4	466.19	366	0.79
4-5	person2_4-5.mp4	532.34	392	0.74
13-15	person3_13-15.mp4	170.71	147	0.86
5	person4_5.mp4	400.53	368	0.92
15	person5_15.mp4	292.09	223	0.76

4.4. Diskusija o rezultatima

Pokretanjem rješenja na osobnom računalu, sustav je uspješno otkrio svakog sudionika u prometu. Procijenjene udaljenosti, međutim, razlikuju od stvarnih mjerenja te su prikazane u obliku postotne pogreške za svaku klasu u tablici 4.2. u potpoglavlju 4.2. Loše određivanje visine graničnog pravokutnika detektiranog sudionika prometa čini pogrešku pri procjeni udaljenosti. Na točnost procjene udaljenosti mogu utjecati sudionici u prometu koji nisu odmah u vidnom polju kamere odnosno koji ne dolaze u pravcu kamere. Osim toga, detektirani granični pravokutnici povremeno su viši od visine koja bi trebali biti za sudionike prometa u video sekvenci, što također utječe na ishode procjene udaljenosti. Stvarne brzine i procijenjene brzine sudionika u prometu prilično su blizu. Ključno je da su netočnosti u mjerenju udaljenosti također utjecale na procjenu brzine. Budući da rješenje temelji svoju procjenu brzine na procijenjenoj udaljenosti, pogreške u

udaljenosti uzrokuju neke varijacije u izračunu brzine. Međutim, kao što je navedeno u potpoglavlju 4.1., pogreške u brzinomjeru svih vozila utjecale su na stvarnu brzinu tijekom snimanja video sekvenci, što je i utjecalo na rezultate procjene brzine. Međutim rezultati procijenjenih brzina su zadovoljavajući i nemaju preveliku postotnu pogrešku koja je prikazana u tablici 4.4. u potpoglavlju 4.2. Budući da se procijenjena udaljenost i brzina koriste za izračunavanje TTC-a, pogreške u tim parametrima uzrokuju odstupanje procjene TTC-a. Unatoč tim pogreškama, rješenje je uspjelo pouzdano pokrenuti alarmni sustav za TTC vrijednosti manje od tri sekunde, pružajući vozaču rano upozorenje na potencijalni sudar njegovih vrata sa sudionicima prometa koji mu dolaze odostraga. Loše istreniran YOLOv7 *Tiny* model korišten za evaluaciju rješenja pokretanjem na Raspberry Pi 4 platformi, ima nešto nižu stopu uspjeha za detektiranje sudionika u prometu od istreniranog modela YOLOv7 korištenog za pokretanje na osobnom računalu. Postoje varijacije i u procijenjenim udaljenostima, koje variraju od stvarnih. Prikazane su u obliku postotne pogreške u tablici 4.9. u potpoglavlju 4.3 te su također prikazane i postotne pogreške za brzinu sudionika prometa u tablici 4.11. Evaluacija rješenja za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila dala je značajan uvid u to kolika je njegova točnost. Rješenje je pokazalo obećavajuće rezultate u detektiranju sudionika u prometu i predviđanju udaljenosti, brzine i TTC-a unatoč određenim ograničenjima. Kako bi se rješenje dodatno poboljšalo, bit će potreban daljnji rad, posebno na modelu YOLOv7 *Tiny* za Raspberry Pi 4 platformu, kako bi se povećala točnost detektiranja i procjena udaljenosti i brzine. Testne video sekvence nakon pokretanja rješenja na njima mogu se pronaći u elektroničkom prilogu P.4.2.

5. ZAKLJUČAK

Kako bi se povećala sigurnost vozača tijekom izlaska iz vozila, u ovom diplomskom radu napravljeno je rješenje koji aktivira alarm kada se odostraga automobilu približava sudionik prometa dovoljno velikom brzinom da bi ugrozio sigurnost vozača prilikom izlaska iz vozila. Rješenje za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila zasnovan je na YOLOv7 algoritmu. Kreirana je vlastita baza podataka za treniranje, validaciju i testiranje algoritma za detekciju sudionika prometa. Istrenirani model klasificira 5 klasa: automobil, kamion/autobus, pješak, biciklist i motocikl. Rješenje je izrađeno u programskom jeziku *Python.* Primarni dio rješenja je sposobnost procjene udaljenosti i brzine kako bi se izračunao TTC koji je osnova za aktiviranje alarmnog sustava i obavještavanje o vremenu potencijalne koliziji s vozačevim vratima. Rješenje može pratiti detektiranog sudionika prometa između okvira jer to zahtijeva znatno manje vremena obrade od detektiranja za svaki okvir. Rezultati rješenja u smislu detekcije sudionika u prometu, izračunavanja udaljenosti i brzine te aktiviranja upozorenja na temelju vremena do potencijalnog sudara s vozačevim vratima (TTC) su zadovoljavajući. Međutim, postoje nedostaci i problemi koji su utjecali na rezultate. Rezultati rješenja na Raspberry Pi 4 platformi imaju veliku postotnu pogrešku za udaljenost koja propagira na sve rezultate rješenja (procjenu brzine i TTC-a). Rezultati postotne pogreške udaljenosti i brzine su manje varijabilni za rješenje pokrenuto na osobnom računalu. Kod implementacije na Raspberry Pi 4 platformi koristio se YOLOv7 *Tiny* model koji je utjecao na rezultate te ga je potrebno istrenirati ponovno s više slika. Vrijeme izvršavanja rješenja na Raspberry pi 4 platformi je približno jednako kao i izvršavanje na osobnom računalu upravo zbog korištenja ubrzanog YOLOv7 *Tiny* modela.

Mogućnost nezgoda pri izlasku iz automobila može se značajno smanjiti pomoću ovog rješenja. Ovo rješenje može biti korisno i za vozače koji imaju posebne zahtjeve ili ograničenu pokretljivost. Prilikom izlaska vozača iz automobila, više znanja o blizini i brzini drugih sudionika u prometu moglo bi ih učiniti sigurnijima i samopouzdanijima.

LITERATURA

[1] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V. K. Papastathis, i M. G. Strintzis, „*Knowledge assisted semantic video object detection,*“ IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 10, str. 1210–1224, 2005.

[2] Z. Luo, „*Traffic analysis of low and ultra-low frame-rate videos*“, Doctoral dissertation. Université de Sherbrooke., 2018.

[3] J. Kulandai Josephine Julina, T. Sree Sharmila i S. Joseph Gladwin „*Vehicle Speed Detection System using Motion Vector Interpolation*“, Global Conference for Advancement in Technology (GCAT), Bangalore, India, str. 1-2, 2019.

[4] M. Moriyama, K. Minemura, i K. Wong, „*Moving Object Detection in HEVC Video by Frame Sub-sampling*“ in International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), str. 48 - 52, 2015.

[5] v7labs, https://www.v7labs.com, [22.6.2023.]

[6] 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.

[7] R. Girshick, „*Fast R-CNN*“, IEEE International Conference on Computer Vision (ICCV), 2015.

[8] 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.

[9] T.Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, S. Belongie, „*Feature pyramid networks for object detection“,* The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, „*ImageNet classification with deep convolutional neural networks*“, Proc. Adv. Neural Inf. Process. Syst., 2012, str. 1097–1105.

[11] W. H. Lopez Pinaya, S. Vieira, R. Garcia-Diaz, A. Mechelli, „*Convolutional neural networks, Machine Learning, Methods and Applications to Brain Disorder*“, Academic Press, str. 173-190, London, UK, 2019.

[12] U. Michelucci, „*Advanced Applied Deep Learning, Convolutional Neural Networks and Object Detection*“, Apress, New York, 2019.

[13] S. Ren, K. He, R. Girshick i J. Sun, „*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*“, Advances in neural information processing systems, 2016., str. 2-7

[14] N. Buhl, „*YOLO models for Object Detection Explained*“, 2023., https://encord.com/blog/yolo-object-detection-guide/, [27.06.2023.]

[15] B. Jelić, „*Identifikacija objekata u slici dobivenoj s kamere u vozilu uz korištenje ros-a*“, Diplomski rad, Fakultet elektrotehnike, računarstva i informacijskih tehnologija, Osijek, 2019.

[16] S.H. Tsang, *„Review: YOLOv3 - You Only Look Once (Object Detection)“,* 2019., https://towardsdatascience.com/review-yolov3-you-only-look-once-object-detection-eab75d7a1ba6, [11.07.2023.]

[17] A. Bochkovskiy, C.Y. Wang i H.Y.M. Liao, „*YOLOv4: Optimal Speed and Accuracy of Object Detection“,* Computer Vision and Pattern Recognition, 2020.

[18] Ultralytics HUB, https://ultralytics.com, [27.06.2023.]

[19] S. Rath, *„YOLOv6 Object Detection – Paper Explanation and Inference“, 2022., https://learnopencv.com/yolov6-object-detection/*, [11.07.2023.]

[20] C.Y. Wang , A. Bochkovskiy i H.Y.M. Liao, „*YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors“,* Institute of Information Science, Academia Sinica, Taiwan, 2022., https://github.com/WongKinYiu/yolov7, [27.06.2023.]

[21] J. Solawetz, „*What is YOLOv7? A Complete Guide.*“, 2022., https://blog.roboflow.com/yolov7-breakdown/, [28.06.2023.]

[22] Z.Y Zhung, K.C. Chen, Y.H. Yu i N. Kwok, „*Chip-based Anti-collision System for Car Door Opening*“, The 4th International Conference on Intelligent Transportation Engineerin, 2019.

[23] C.J. Lin, S.Y. Jeng i H.W. Lioa, „*A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO*“, Hindawi Mathematical Problems in Engineering, 2021.

[24] V. Miles, F. Gurr, S. Giani, „*CameraBased System for the Automatic Detection of Vehicle Axle Count and Speed Using Convolutional Neural Networks*“, International Journal of Intelligent Transportation Systems Research, 2022.

[25] Dokumentacija *Anaconda*, https://docs.anaconda.com, [29.06.2023.]

[26] PyTorch, https://pytorch.org/, [29.06.2023.]

[27] B. Le, „*An Introduction to BYTETrack: Multi-Object Tracking by Associating Every Detection Box*“, 2023., https://www.datature.io/blog/introduction-to-bytetrack-multi-object-tracking-by-associating-every-detection-box , [29.06.2023.]

[28] N. Wojke, A. Bewley, and D. Paulus, „*Simple Online and Realtime Tracking with a Deep Association Metric*“, 2017 IEEE International Conference on Image Processing (ICIP), 2017.

[29] E. Šimara, „*Detekcija rizika od sudara i pametna kontrola brzine*“, Diplomski rad, Fakultet elektrotehnike, računarstva i informacijskih tehnologija, Osijek, 2020.

[30] BDD100K baza slika, Berkeley Artificial Intelligence Research Lab of the University of California, Berkeley, https://bdd-data.berkeley.edu , [30.06.2023.]

[31] KITTI Vision Benchmark Suite, Karlsruhe Institute of Technology and Toyota Technological Institute, http://www.cvlibs.net/datasets/kitti/, [28.06.2023.]

[32] M. Braun, S. Krebs, F. B. Flohr I D. M. Gavrila, „*The EuroCity Persons Dataset: A Novel Benchmark for Object Detection“,* https://eurocity-dataset.tudelft.nl/eval/overview/statistics, [28.06.2023.]

[33] nuScene baza slika , Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan i O. Beijbom, „*A multimodal dataset for autonomous driving*”, CVPR, 2020., https://www.nuscenes.org/nuimages, [29.06.2023.]

[34] Parallel Domain, https://paralleldomain.com/, [29.06.2023.]

[35] Alat *labelImg*, https://pypi.org/project/labelImg/, [29.06.2023.]

[36] J. Brownlee, „*Difference Between a Batch and an Epoch in a Neural Network*“, 2020., https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/, [17.7.2023.]

[37] Parametri kamere, Canon EOS 750D, https://www.photoreview.com.au/reviews/dslr-cameras/entry-level/canon-eos-750d/, [3.7.2023]

[38] Fergus, „*Understanding Car Size and Dimensions*“, 2022., https://www.nationwidevehiclecontracts.co.uk/guides/ask-nvc/understanding-car-size-and-dimensions, [18.7.2023.]

[39] ENPEKS, https://www.enpeks.rs/truck-dimensions.html, [18.7.2023.]

[40] Dr. Y. Yurttas, „*The Average Height of Men and Women Worldwide*“, 2022., https://www.dryukselyurttas.com/post/average-height-men-and-women, [18.7.2023.]

[41] Quora, https://www.quora.com/What-is-the-average-height-of-a-male-professional-cyclist, [18.7.2023.]

[42] PowerSportsGuide, https://powersportsguide.com/average-motorcycle-dimensions/, [18.7.2023.]

[43] *PyTorch*, https://qengineering.eu/install-pytorch-on-raspberry-pi-4.html, [3.7.2023.]

[44] Aplikacija *Speedometer Simple*, https://apps.apple.com/us/app/speedometer-simple/id939507215, [4.7.2023.]

[45] Aplikacija *Samsung Health*, https://www.samsung.com/global/galaxy/apps/samsung-health/, [4.7.2023.]

[46] D. Mikulić, „*MOTORNA VOZILA Teorija kretanja i konstrukcija“*, Veleučilište Velika Gorica, Velika Gorica, 2020., str. 42

[47] ORYX, „*Zašto brzinomjeri u autima pokazuju veću brzinu od stvarne?*“,2023. , https://www.oryx-asistencija.hr/savjeti-za-vozace/aktualno/zasto-brzinomjeri-u-autima-pokazuju-vecu-brzinu-od-stvarne-14821, [4.7.2023.]

SAŽETAK

U ovom diplomskom radu opisano je rješenje za uključivanje alarmnog sustava prilikom izlaska vozača iz vozila s ciljem povećanja sigurnosti vozača i sudionika prometa koji mu dolaze odostraga. Rješenje se sastoji od YOLOv7 algoritma za detekciju sudionika prometa u okviru video sekvence, algoritma za praćenje sudionika prometa te algoritma za procjenu udaljenosti koji je potreban za procjenu brzine i vremena do potencijalne kolizije s vozačevim vratima na temelju kojeg se uključuje alarmni sustav. Rješenje je implementirano na osobnom računalu i na Raspberry Pi 4 platformi. Osobno računalo ima jači procesor i grafičku karticu, što omogućuje brže izvršavanje rješenja. S druge strane, Raspberry Pi 4 platforma ima ograničenje u računalnoj snazi, što utječe na vrijeme izvršavanja rješenja i točnost rezultata. Na Raspberry Pi 4 platformi zbog ograničenih resursa detekcija je napravljena pomoću YOLOv7 *Tiny* modela. Rješenje je testirano na vlastitoj bazi od 34 video sekvence. Detekcijom je određen položaj sudionika prometa u video okviru pomoću istreniranog YOLOv7 modela koji klasificira pet klasa: automobil, kamion/autobus, pješak, biciklist i motociklist. Brža obrada omogućena je praćenjem detektiranog sudionika prometa između okvira pomoću *DeepSort* algoritma. Procjena brzine sudionika prometa se izvršava na temelju prijeđene udaljenosti između dvaju okvira video sekvenci. Na 34 testne video sekvence rezultati su pokazali da se postotna pogreška za udaljenost na osobnom računalu kreće se od -24% do +27%, dok na Raspberry Pi 4 platformi varira od -42% do +28%. Za brzinu, postotna pogreška na osobnom računalu kreće se od -39% do +52%, dok na Raspberry Pi 4 platformi iznosi od -48% do +69%. Važno je napomenuti da se postotne pogreške razlikuju ovisno o klasi sudionika prometa, udaljenosti i brzini na kojoj su testirane. Međutim, unatoč navedenim rezultatima, rješenje kao takvo ima prostora za doradu kako bi se povećala preciznost u procjenu udaljenosti, brzine i TTC-a.

Ključne riječi: *detekcija, procjena udaljenosti, procjena brzine, YOLOv7, Raspberry Pi 4*

ALGORITHM FOR TURNING ON THE ALARM SYSTEM WHEN LEAVING THE VEHICLE

ABSTRACT

This thesis describes a solution for turning on the alarm system when the driver exits the vehicle, with the aim of enhancing the safety of both the driver and the traffic participants approaching from the rear. The solution consists of the YOLOv7 algorithm for detecting traffic participants in video sequences, a tracking algorithm for tracking traffic participants, and a distance estimation algorithm needed for calculating speed and time-to-collision with the driver's door, which triggers the alarm system. The solution is implemented on a personal computer and on the Raspberry Pi 4 platform. The personal computer has a more powerful processor and graphics card, enabling faster execution of the solution. On the other hand, the Raspberry Pi 4 platform has limited computational resources, affecting the execution time and accuracy of the results. On the Raspberry Pi 4 platform, detection is performed using the YOLOv7 Tiny model due to a lack of resources. The solution is tested on a custom dataset consisting of 34 video sequences. Detection determines the position of traffic participants in the video frame using the trained YOLOv7 model that classifies five classes: car, truck/bus, pedestrian, cyclist, and motorcyclist. Faster processing is achieved by tracking the detected traffic participant between frames using the DeepSort algorithm. Speed estimation of traffic participants is performed based on the distance traveled between two frames of the video sequence. Over the 34 test video sequences, the results showed that the percentage error for distance on the personal computer ranges from -24% to +27%, while on the Raspberry Pi 4 platform, it varies from -42% to +28%. For speed, the percentage error on the personal computer ranges from -39% to +52%, while on the Raspberry Pi 4 platform, it ranges from -48% to +69%. It is important to note that the error percentage varies depending on the class of traffic participant, distance, and speed at which they were tested. However, despite these results, the solution has room for improvement to increase the accuracy of distance, speed, and TTC (Time-to-Collision) estimation.

Keywords: *detection, distance estimation, speed estimation, YOLOv7, Raspberry Pi 4*

ŽIVOTOPIS

Marija Ovžetski rođena je u Našicama 28. veljače 1999. Odrasla je u Donjem Miholjcu, a trenutačno živi u Osijeku. Školovanje je započela 2005. godine u Osnovnoj školi „August Harambašić“ u Donjem Miholjcu te je sve razrede osnovne škole završila s odličnim uspjehom. Godine 2013. upisala je opću gimnaziju u Srednjoj školi Donji Miholjac. Fakultet elektrotehnike, računarstva, i informacijskih tehnologija Osijek (FERIT) upisala se 2017. godine, gdje je sudjelovala na projektu „Slavonska STEM evolucija“ kao član jednog od timova u edukaciji djece o STEM području u osnovnim školama i u organizaciji LABUS sajma. 2020. godine upisala je sveučilišni diplomski studij razlikovne obveze, smjer komunikacije i informatika. Zatim 2021. upisuje diplomski sveučilišni studij Elektrotehnika, smjer komunikacije i informatika.

Potpis:

_____

PRILOZI (elektronički)

Prilog P.3.1. Slike korištene za proces treniranja, validacije i testiranja CNN algoritama zasnovanih na YOLOv7 i YOLOv7 *Tiny* algoritmima za detekciju objekata (elektronički prilog).

Prilog P.3.2. Konfiguracijska, names i data datoteke korištene u radu CNN algoritama zasnovanih na YOLOv7 i YOLOv7 *Tiny* algoritmima (elektronički prilog).

Prilog P.3.3. Kompletan programski kod predloženog rješenja sa svim potrebnim datotekama za ispravan rad (elektronički prilog).

Prilog P.4.1. Skup video sekvenci korištene za testiranje performansi rada rješenja (elektronički prilog).

Prilog P.4.2. Video sekvence nakon obrade implementiranim rješenjem (električni prilog).