Prepoznavanje prometnih znakova za ograničenje maksimalne brzine kretanja za primjenu u naprednim ADAS algoritmima

Strišković, Barbara

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:464953

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

Sveučilišni studij

PREPOZNAVANJE PROMETNIH ZNAKOVA ZA OGRANIČENJE MAKSIMALNE BRZINE KRETANJA ZA PRIMJENU U NAPREDNIM ADAS ALGORITMIMA

Diplomski rad

Barbara Strišković

Osijek, 2020.

SADRŽAJ

1. UVOD

U 2019. godini je u 27 država članica EU zabilježeno 22 800 smrtnih slučajeva u cestovnom prometu. To predstavlja gotovo 7 000 manje smrtnih slučajeva u odnosu na 2010. godinu, odnosno smanjenje za 23%. Iako je broj smrtnih slučajeva i dalje u padu, pad je usporen u većini zemalja od 2013. godine. Zacrtani cilj EU je prepoloviti broj smrtnih slučajeva na cestama do 2020. godine, u odnosu na referentnu vrijednost iz 2010. godine [1]. Kako bi se povećala sigurnost vožnje i općenito sigurnost prometa, automobilska industrija je dizajnirala elektronički sustav u vozilu koji pomaže vozaču u svakodnevnoj vožnji. Takav sustav naziva se napredni sustav za pomoć vozaču (engl. *Advanced driver-assistance systems*, ADAS). ADAS smanjuje utjecaj ljudske pogreške tako da alarmira vozača o opasnim situacijama ili preuzme kontrolu nad vozilom[2]. Automatizirani ADAS smanjuju broj smrtnih slučajeva na cesti smanjujući zapravo broj ljudskih (vozačevih) pogrešaka [3]. Sigurnosne značajke ADAS-a dizajnirane su za izbjegavanje nesreća i sudara, nudeći tehnologije koje vozača upozoravaju na probleme, primjenjujući zaštitne mjere (detekcija drugih vozila, detekcija pješaka, detekcija voznih traka, automatsko kočenje, sustav za nadziranje vozača itd.) i preuzimajući kontrolu nad vozilom ako je potrebno.

Proučavanje prometa pokazalo je da se većina prometnih nesreća događa zbog nepažljivih vozača koji ne obraćaju pažnju na prometne znakove i situaciju oko vozila. Visok postotak prometnih nesreća sa smrtnim posljedicama uzrokovan je pogreškama vozača koji često prekorače najveću dopuštenu brzinu na cesti. Ishod takvih proučavanja dovodi do potrebe automatskog sustava detekcije ograničenja brzine u automobilu, koji vozača obavještava o maksimalnoj dozvoljenoj brzini. Također, sposobnost ljudske vizualne percepcije ovisi o fizičkom i psihičkom stanju pojedinca. U određenim okolnostima, na ove sposobnosti može utjecati mnogo faktora poput umora i napetosti, te je i zbog toga vrlo važno uspostaviti sustav automatskog prepoznavanja prometnih znakova za ograničenje maksimalne brzine, koji vozaču može biti od velikog značaja. Posljednjih godina broj istraživanja vezanih za prepoznavanje i detekciju prometnih znakova za ograničenje maksimalne brzine je porastao, zbog stvarne potrebe za takvim sustavima u vozilima.

Ovaj se diplomski rad bavi problematikom detekcije i ispravnog prepoznavanja prometnih znakova, s naglaskom upravo na znakove maksimalnog ograničenja brzine kretanja vozila. Za ovu se namjenu obrađuje signal s kamere postavljene na prednjoj strani vozila. Iako postoji mnoštvo rješenja koje pokušavaju riješiti ovaj problem, mali broj njih koji su javno dostupni je implementiran na realnu

ugradbenu platformu za rad u stvarnom vremenu, kao što je ADAS razvojna ploča. Programsko rješenje za detekciju prometnih znakova razvijeno je u C programskom jeziku za Texas Instruments (TI) Vision SDK (engl. *Software Development Kit*) razvojno okruženje i konkretnu ADAS razvojnu ploču. Za ispravno prepoznavanje prometnih znakova ograničenja brzine kretanja vozila korištena je konvolucijska neuronska mreža (engl. *Convolutional Neural Network*, CNN ili ConvNet) koja je za potrebe treniranja napisana u programskom jeziku Python i istrenirana na računalu. Tijekom treniranja CNN dobiveni su parametri mreže koji su izvezeni u obliku tekstualne datoteke te kasnije korišteni za implementaciju same CNN napisane u programskom jeziku C na ADAS razvojnu ploču. U drugom poglavlju opisan je problem detekcije i prepoznavanja prometnih znakova te je dan osvrt na postojeće radove koji se bave sličnom tematikom. U trećem poglavlju detaljno je objašnjeno vlastito rješenje za detekciju i prepoznavanje prometnih znakova ograničenja brzine. U četvrtom poglavlju predstavljeni su rezultati ispitivanja performansi stvorenog rješenja i diskusiju o rezultatima. Također je opisana korištena baza podataka za testiranje prepoznavanja prometnih znakova i problemi na koje predloženo rješenje nailazi. U petom poglavlju dan je zaključak diplomskog rada.

2. PROBLEM PREPOZNAVANJA PROMETNIH ZNAKOVA ZA OGRANIČENJE MAKSIMALNE BRZINE

Problem prepoznavanja prometnih znakova sastoji se od dva dijela, problem detekcije prometnog znaka i problem prepoznavanja prometnog znaka, odnosno klasifikacije. Dakle, prvo je potrebno u danoj sceni pronaći prometni znak, a zatim ispravno prepoznati koji je to prometni znak. Predstavljeni problemi u struci se rješavaju pomoću dviju tehnika, tehnika računalnog vida i strojnog učenja.

Računalni vid je interdisciplinarno znanstveno područje koje se bavi načinom na koji računala mogu steći razumijevanje visoke razine digitalnih slika ili videozapisa. Programska rješenja koja implementiraju tehnike računalnog vida nastoje razumjeti i automatizirati zadatke koje ljudski vizualni sustav obavlja. Zadaci računalnog vida uključuju pribavljanje, obradu, analizu i razumijevanje digitalnih slika te izdvajanje podataka velikih dimenzija iz stvarnog svijeta, radi dobivanja brojčanih ili simboličkih podataka. Razumijevanje u ovom kontekstu znači transformaciju vizualnih slika u opise svijeta koji imaju smisla za misaone procese i mogu izazvati odgovarajuće djelovanje. To razumijevanje slike može se promatrati kao odvajanje simboličkih informacija od slikovnih podataka pomoću modela izgrađenih pomoću geometrije, fizike, statistike i teorije učenja [4].

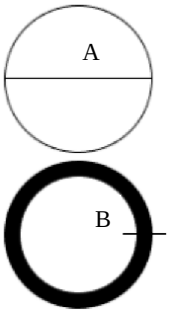
Strojno učenje je znanstveno područje koje se bavi proučavanjem algoritama i statističkih modela koje računalni sustavi koriste za izvršavanje određenog zadatka bez korištenja izričitih uputa, oslanjajući se na obrasce i zaključke[5]. Ono predstavlja podskup umjetne inteligencije. Algoritmi strojnog učenja grade matematički model zasnovan na uzorku podataka, poznat kao trening skup, kako bi se donijela predviđanja ili odluke bez korištenja izričitih uputa. Algoritmi strojnog učenja koriste se u širokom rasponu aplikacija, poput filtriranja e-pošte i računalnog vida, gdje je teško ili nemoguće razviti konvencionalne algoritme za obavljanje potrebnih zadataka.

2.1. Karakteristike prometnog znaka ograničenja brzine

Kako bi se detekcija znaka mogla provesti, nužno je znati karakteristike onoga što je potrebno pronaći. Kao i svaki drugi prometni znak, prometni znakovi ograničenja brzine imaju svoje karakteristike po kojima su specifični. Prometni znakovi ograničenja brzine pripadaju skupini prometnih znakova izričitih naredbi. Prometni znakovi izričitih naredbi su okrugli, bijele pozadine te sadrže dvije koncentrične kružnice čiji je međuprostor ispunjen crvenom bojom, odnosno time je dobiven crveni

prsten. Promjer koncentričnih kružnica varira ovisno o vrsti cesta (lokalna cesta, autocesta itd.) na kojima je prometni znak postavljen te vrijedi za sve prometne znakove izričitih naredbi, a ne samo za znakove ograničenja brzine, što je prikazano u tablici 2.1. [6]. Slika. 2.1. prikazuje prometni znak ograničenja brzine 30km/h.

Tablica 2.1. *Prikaz promjera koncentričnih kružnica za znakove izričitih naredbi (mm)[6]*

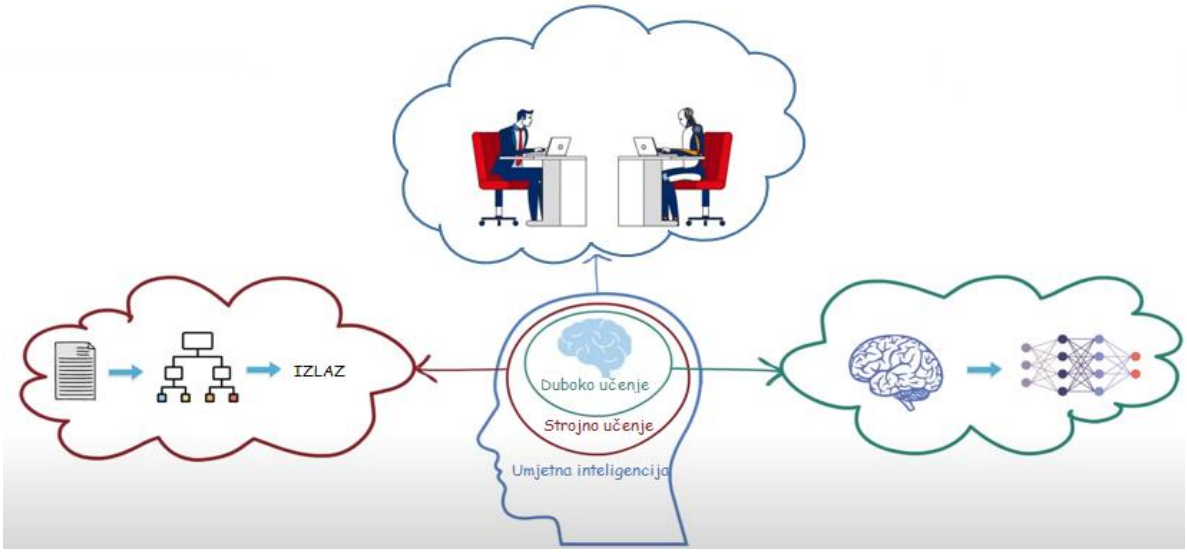
Oblik prometnog znaka	Element znaka	Autoceste i brze ceste	Ostale javne ceste i glavne gradske prometnice	Ostale ceste i prometne površine	Tuneli, galerije i javne garaže (minimalno)
	A	900	600	400	300
	B	90	60	40	30

Prometni znakovi ograničenja brzine na bijeloj pozadini sadrže najčešće dvije znamenke crne boje, a za ograničenja 100km/h i više, tri znamenke. Prometni znakovi postavljaju se s desne strane ceste uz kolnik u smjeru kretanja vozila [6]. Iznimno, ako na mjestu na kojem se postavlja prometni znak postoji opasnost da ga sudionici u prometu neće na vrijeme primijetiti zbog gustoće prometa ili zbog drugih razloga, prometni znak može se postaviti i na suprotnoj, lijevoj strani ceste ili iznad kolnika [6].

Sl. 2.1. *Prometni znak ograničenja brzine 30km/h*

2.2. Umjetne neuronske mreže

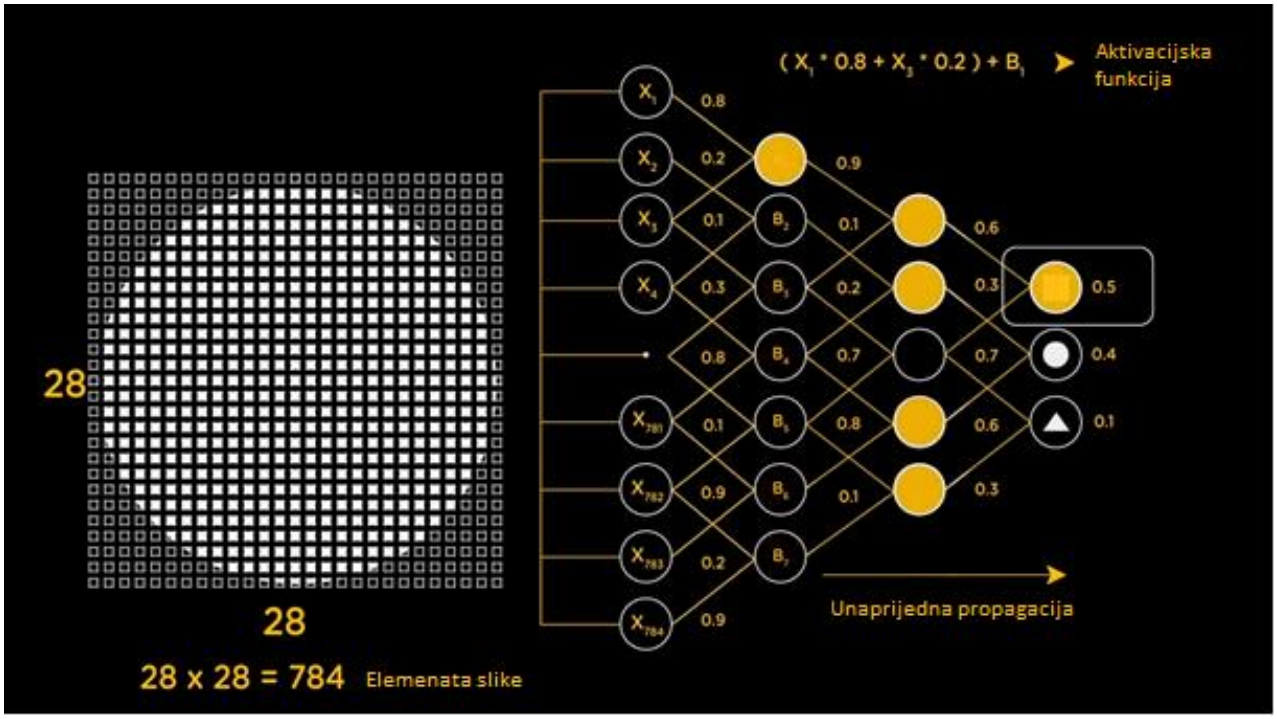
Umjetna inteligencija (engl. *artificial intelligence*) odnosi se na simulaciju ljudske inteligencije u strojevima koji se pokušavaju programirati da razmišljaju kao ljudi i oponašaju njihovo djelovanje. Izraz se također može primijeniti na bilo koji stroj koji pokazuje osobine povezane s ljudskim umom, poput učenja i rješavanja problema [7]. Strojno učenje (engl. *machine learning*) je znanstveno područje koje realizira umjetnu inteligenciju kroz algoritme i statističke modele koje računalni sustavi koriste za izvršavanje određenog zadatka. Nadalje, duboko učenje (engl. *deep learning*) je oblik strojnog učenja inspiriran biološkim živčanim sustavom. Na slici 2.2. prikazan je odnos umjetne inteligencije, strojnog učenja i dubokog učenja [8]. Duboko učenje računalima omogućuje učenje iz iskustva i razumijevanje svijeta, što se postiže višeslojnim umjetnim neuronskim mrežama. Duboko učenje se također može definirati kao funkcija umjetne inteligencije koja oponaša rad ljudskog mozga u obradi podataka i stvaranju obrazaca za korištenje u odlučivanju. Ono što razlikuje duboko učenje od strojnog učenja je način na koji se dobiva rezultat. Da bi se strojnim učenjem postigao rezultat, potrebno je odrediti značajke po kojima će se odrediti rezultat (potrebna je ljudska intervencija), dok kod dubokog učenja umjetna neuronska mreža (engl. *Artifical Neural Networks*, ANN) sama određuje značajke po kojima će odrediti rezultat. No duboko učenje upravo zbog ove karakteristike zahtjeva veći obujam podataka na kojima će se izvršiti učenje. [8] Za klasifikaciju prometnih znakova u sklopu ovog rada korišteno je duboko učenje.



Sl. 2.2. *Slikoviti prikaz odnosa umjetne inteligencije, strojnog učenja i dubokog učenja, preuzeto iz videozapisa [8]*

ANN su dizajnirane kako bi simulirale način na koji ljudski mozak analizira i obrađuje informacije. To je temelj umjetne inteligencije i rješava čak i probleme koji su se pokazali nerješivim ili teškim ljudskim ili statističkim standardima. Izgrađene su od velikog broja međusobno povezanih procesnih elemenata (neurona), s uzorom na građu ljudskog mozga. Svaki je neuron sumirajući element, povezan s aktivacijskom funkcijom. Neuroni su strukturirani u slojeve (ulazni, jedan ili više skrivenih slojeva i izlazni sloj). Izlazne vrijednosti neurona jednog sloja predstavljaju ulazne vrijednosti neurona za sljedeći sloj, transformiraju te vrijednosti uz pomoć parametara mreže, te šalju izlazne vrijednosti u sljedeći sloj. Da bi se neuronska mreža definirala, pored osnovnih parametara koji opisuju oblik i tip mreže, odnosno arhitekturu, potrebno je odrediti i način učenja. Proces učenja proces je optimizacije parametara mreže pomoću određenog algoritma, gdje se pronalaze najbolji parametri mreže, koji daju najbolje rješenje problema.

Kako bi se što bolje pojasnio princip rada ANN, u daljnjem tekstu će biti objašnjen jednostavan primjer ANN koja treba klasificirati dani ulaz kao jedan od 3 moguća izlaza: kvadrat, krug ili trokut. Na slici 2.3.[9] vidi se slikoviti prikaz neuronske mreže koja predviđa što se nalazi na ulaznoj slici. Ulaz u prikazanu neuronsku mrežu je slika koja je veličine 28x28 elemenata slike, što ukupno daje



Sl. 2.3. *Strukturni prikaz jednostavne neuronske mreže za klasifikaciju oblika, preuzeto iz videozapisa sa [9]*

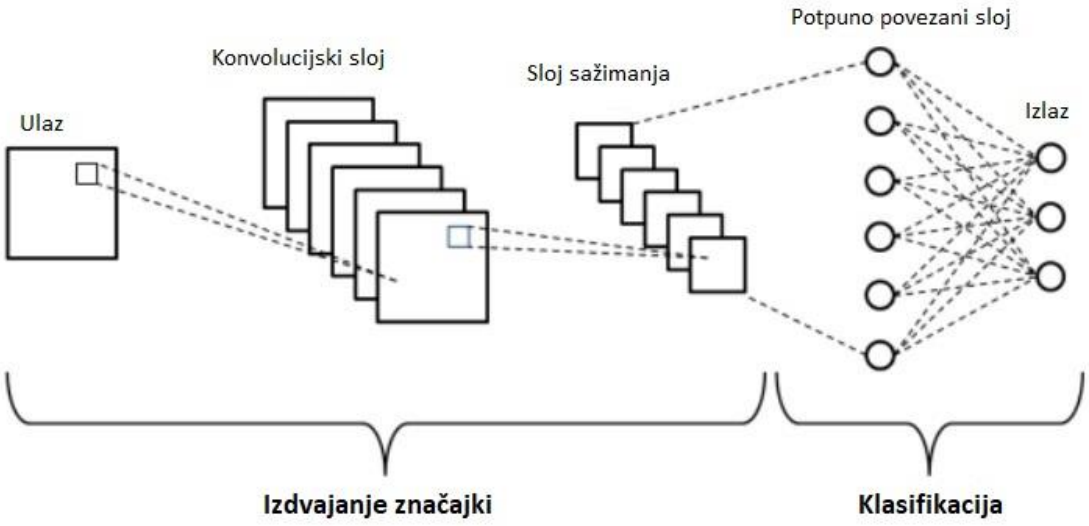
784 elemenata slike. Svaki taj element slike unosi se u mrežu na način da jedan element slike odlazi u jedan neuron prvog sloja mreže. Neuroni prvog sloja su povezani s neuronima drugog sloja mreže putem kanala. Svaki kanal ima dodijeljenu numeričku vrijednost koja se naziva težina. Svaki ulazni čvor, u kojem se nalazi vrijednost jednog elementa slike, množi se s odgovarajućom težinom kanala i zbraja s određenim drugim ulaznim čvorom koji je također pomnožen s njemu odgovarajućom težinom kanala. Tako zbrojeni predstavljaju ulaz za neurone u prvom skrivenom sloju. Za sliku 3.4 to znači da za prvi neuron u prvom skrivenom sloju, zbrajaju se umnošci ulaza i težina za 1. i 3. ulazni čvor (x_1 i x_3). Svaki neuron u skrivenom sloju ima svoju numeričku vrijednost koja se naziva *bias* i ona se nadodaje na ulaznu sumu. Dobivena vrijednost se prosljeđuje aktivacijskoj funkciji. Uloga aktivacijske funkcije je odrediti hoće li određeni neuron biti aktiviran ili neće. Ako je neuron aktiviran, on prenosi podatke sljedećem sloju putem sljedećih kanala koji također imaju svoje težine. Ovakav način propagacije podataka kroz mrežu se naziva unaprijedna propagacija. Na izlaznom sloju se aktivira neuron s najvećom vrijednošću i tako određuje izlaz mreže za trenutne parametre, pri čemu spomenuta vrijednost predstavlja vjerojatnost da je određeni izlaz točan. Na slici 3.4. vidi se da je mreža krivo predvidjela izlaz. Osim što je mreža dobila ulaz, također je dobila i stvarni izlaz (u procesu treniranja). Stoga mreža može predviđeni izlaz usporediti sa stvarnim izlazom kako bi uvidjela grešku. Za trenutni ulaz slike kruga, pretpostavljeni izlaz bi trebao biti 0,1,0 što predstavlja da izlaz nije kvadrat i nije trokut, ali je krug (100 %). S obzirom da je prethodno rečeno da vrijednost koja je na izlazu predstavlja vjerojatnost da je određeni izlaz točan, sada se može reći da ta vrijednost predstavlja veličinu greške. Dakle, za prvi izlaz pretpostavljena vrijednost je 0, a mreža je dobila 0.5, što znači da je predviđena vrijednost veća od stvarne i ukazuje na to da ju je potrebno smanjiti kako bi se smanjila greška. Za drugi izlaz predviđena vrijednost je 1, a dobivena 0.4, što znači da je predviđena vrijednost manja od stvarne i ukazuje na to da je potrebno ju povećati. Ove informacije se tada distribuiraju unazad kroz slojeve mreže (engl. *the backward propagation of errors*) i s obzirom na te informacije težine kanala se podešavaju kako bi dale bolje rezultate u sljedećoj iteraciji učenja. Ovako objašnjen ciklus unaprijedne i unazadne propagacije se iterativno izvršava s višestrukim ulazima. Proces se ponavlja dok težine nisu podešene tako da mreža ispravno predviđa oblike odnosno dok se ne postignu željene performanse mreže

Dakle, duboko učenje otkriva zamršenu strukturu u velikim skupinama podataka koristeći unazadnu propagaciju, kako bi pokazao kako računalo treba mijenjati svoje interne parametre koji se koriste za izračunavanje reprezentacije u svakom sloju od reprezentacije u prethodnom sloju. Duboke neuronske

mreže donijele su napredak u obradi slike, videozapisa, govora i zvuka. Klasa dubokih neuronskih mreža koja je korištena u ovom radu je CNN.

2.2.1. Konvolucijske neuronske mreže

CNN je klasa dubokih neuronskih mreža, koja se najčešće primjenjuje za analizu slika. Ovakve mreže primjenjuju se u klasifikaciji slika, analizama medicinskih slika, obradi prirodnog jezika i slično [9]. CNN je tip unaprijedne neuronske mreže koja sadrži jedan ili više konvolucijskih slojeva. Na slici 2.4.[10] nalazi se primjer jednostavnog shematskog prikaza osnovne CNN. Prikazana mreža sastoji



Sl. 2.4. *Prikaz CNN sa karakterističnim slojevima*

se od pet različitih slojeva: ulazni sloj, sloj konvolucije (konvolucijski sloj), sloj sažimanja, potpuno povezan sloj i izlazni sloj. Slojevi su podijeljeni u dva dijela: oni koji služe za izdvajanje značajki i koji služe za klasifikacija. Slojevi izdvajanja značajki su ulazni sloj, sloja konvolucije i sloj sažimanja (engl. *pooling layer*), dok su slojevi za klasifikaciju potpuno povezani sloj (engl. *film connected layer*) i izlazni sloj. Ulazni sloj određuje fiksnu veličinu za ulazne slike, koja se po potrebi mijenja. Zatim se na sliku primjenjuje konvolucija s više naučenih kernela pomoću zajedničkih težina po sloju konvolucije. Nadalje, sloj sažimanja smanjuje veličinu slike dok pokušava zadržati sadržane informacije. Izlazi izdvajanja značajki poznati su kao mape značajki. Klasifikacija kombinira izdvojene značajke u potpuno povezanim slojevima. Postoji po jedan izlazni neuron za svaku

kategoriju objekata u izlaznom sloju. Rezultat klasifikacijskog dijela rezultat je tražene klasifikacije [10].

Unaprijedne neuronske mreže su mreže u kojima se propagacija signala odvija samo u jednom smjeru, od ulaznih veličina prema izlaznoj veličini, što znači da mreža ne sadrži povratne veze. Ovakve mreže, za razliku od standardnih unaprijednih mreža zasnovanih na potpuno povezanim slojevima, uzimaju u obzir da je ulaz slika određene rezolucije te su neuroni u konvolucijskim slojevima organizirani po visini, širini, ali i dubini. Uobičajena konvolucijska mreža za zadaće klasifikacije slika sadrži, osim konvolucijskih slojeva i slojeve sažimanja, potpuno povezane slojeve te *softmax* sloj na izlazu. Konvolucijski sloj sastoji se od jednog ili više konvolucijskih filtara koji imaju podesive težine. Svaki filter ima određenu prostornu veličinu (npr. 5x5), ali se proteže kroz cijelu dubinu ulaznog volumena. Filtri konvolucijskih slojeva pomiču se po ulaznom volumenu po širini i visini, pri čemu se svaki puta izračuna skalarni produkt težina filtera i odgovarajućih dijelova slike. Kako se filter pomiče po ulaznoj slici po visini i širini, na rezultat skalarnog produkta se primjenjuje aktivacijska funkcija (npr. ReLU) te se rezultat sprema u izlaznu dvodimenzionalnu aktivacijsku mapu. Pretpostavlja se da filtri filtri imaju svoju određenu visinu i širinu, a dubina je određena dubinom samih podataka na koje ti filtri trebaju biti primijenjeni. Veličinu izlazne mape moguće je izračunati pomoću sljedećeg izraza:

$$N_{out} = (N_{in} - F) \cdot S + 1, \quad (2.1.)$$

gdje je N_{in} visina/širina ulaznog volumena (aktivacijske mape), N_{out} visina/širina izlazne aktivacijske mape, F je visina/širina filtra, a S predstavlja *stride* odnosno korak pomaka filtra tijekom izračunavanja aktivacijske mape[11].

2.2.2. Učenje neuronske mreže

Neuronska mreža može se promatrati kao matematička struktura kojom se pokušava aproksimirati funkcionalna ovisnost između ulaznih veličina x i izlazne veličine y. Pod pojmom učenje neuronske mreže podrazumijeva se procjena nepoznatih parametara neuronske mreže, tj. težina mreže i *bias*-a.

Prva faza unaprijedne propagacije događa se kada je mreža izložena podacima za trening i oni prolaze kroz čitavu neuronsku mrežu kako bi se predvidio izlazni rezultat na osnovu trenutnog (inicijalnog) skupa parametara mreže. Ulazni podaci prosljeđuju se kroz mrežu na takav način da svi neuroni primjenjuju svoju transformaciju na informacije koje dobivaju od prethodnog sloja i šalju ih sljedećem sloju. Kad su podaci prešli sve slojeve, konačni sloj će biti okinut rezultatom predviđanja

za određene ulazne podatke. Zatim se koristi funkcija gubitaka za procjenu gubitka (ili pogreške) te za usporedbu i mjerenje koliko je dobar / loš rezultat predviđanja u odnosu na točan rezultat. U idealnom slučaju poželjno je da greška bude nula, odnosno bez odstupanja između procijenjene i očekivane vrijednosti. Stoga će se, kako se model trenira, težine međusobnih veza neurona postupno prilagođavati sve dok se ne dobiju dobra predviđanja. Nakon što se izračuna gubitak, te se informacije šire prema natrag. Polazeći od izlaznog sloja, ta se informacija o gubitku širi na sve neurone u skrivenom sloju koji izravno doprinose izlazu. Međutim, neuroni skrivenog sloja primaju samo djelić ukupnog signala gubitka, na temelju relativnog doprinosa koji je svaki neuron pridonio izvornom ishodu. Taj se postupak ponavlja, sloj po sloj, sve dok svi neuroni u mreži nisu primili signal gubitka koji opisuje njihov relativni doprinos ukupnom gubitku. Kada su podaci poslani natrag, može se prilagoditi težina veze između neurona. Cilj je da gubitak bude manji u sljedećem pokušaju i što bliži nuli sljedeći put kad se mreža okine. Kako bi se to postiglo, koristi se gradijentna metoda. Ova metoda mijenja težine grana u malim koracima pomoću izračunavanja gradijenta funkcije gubitaka, što nam omogućava da se vidi u kojem se smjeru „spušta" prema globalnom minimumu. Ovo se provodi općenito u skupinama podataka u uzastopnim ponavljanjima (epohama) svih skupa podataka koji se prenose u mreži u svakoj iteraciji. S obzirom da CNN strukturno ne sadrže neurone već filtre, kod CNN se prilikom treniranja korigiraju parametri filtara u konvolucijskim slojevima.

2.3. Postojeća rješenja detekcije i klasifikacije prometnih znakova

Prije samog predstavljanja postojećih rješenja, potrebno je definirati pojmove koji se koriste prilikom prikazivanja rezultata testiranja u relevantnim radovima. Postoje četiri izraza kojima se vrednuje klasifikator te su prikazani u tablici 2.2. Ako se primjenjuju navedeni izrazi za vrednovanje klasifikatora tada je ukupan broj primjera = TP + TN + FP + FN, broj točno klasificiranih primjera = TP + TN, broj pozitivnih primjera = TP + FN i broj negativnih primjera = TN + FP[12].

Tablica 2.2. *Prikaz izraza kojima se vrednuje klasifikator*

		Stvarne vrijednosti	
		Pozitivno	**Negativno**
Predviđene vrijednosti	**Pozitivno**	Potvrdno pozitivan (engl. *True Positive*, TP)	Lažno pozitivan (engl. *False Positive*, FP)
	Negativno	Lažno negativan (engl. *False Negative*, FN)	Potvrdno negativan (engl. *True Negative*, TN)

Točnost (engl. *accuracy*) je udio točno klasificiranih primjera u skupu svih primjera, izražava se u postotcima (%) i definira se kao[12]:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (2.2.)$$

Preciznost (engl. *precision*) je udio točno klasificiranih primjera u skupu pozitivno klasificiranih primjera, izražava se u postotcima (%) i definira se kao[12]:

$$P = \frac{TP}{TP + FP} \times 100 \quad (2.3.)$$

Odziv (engl. *recall*) je udio točno klasificiranih primjera u skupu svih pozitivnih primjera, izražava se u postotcima (%) i definira se kao[12]:

$$R = \frac{TP}{TP + FN} \times 100 \quad (2.4.)$$

Specifičnost (engl. *specificity*) je udio točno klasificiranih primjera u skupu svih negativnih primjera, izražava se u postotcima (%) i definira se kao[12]:

$$S = \frac{TN}{TN + FP} \times 100 \quad (2.5.)$$

Prema [13], detekcija i prepoznavanje prometnih znakova za brzinu odvija se u tri koraka. Prvo se radi segmentacija slike pomoću formule:

$$S = R - g, \quad (2.6.)$$

gdje je *R* crvena komponenta slike prikazane u RGB formatu boja, *g* ulazna slika prikazana u skali sive boje (engl. *grayscale*) i **S** segmentirana slika. Dakle, segmentirana slika **S**, dobivena je oduzimanjem elemenata slike (engl. *pixel*) iz slike u skali sive boje od odgovarajućih elemenata slike crvene komponente slike. Prednost ove obrade jest prilično efikasno uklanjanje svih neželjenih predmeta na slici i zadržavanje malog broja kandidata, koji imaju karakteristike kao znak ograničenja brzine. Drugi korak je pronalaženje područja interesa (engl. *Region of Interest*, ROI*)* primjenjujući maksimalno stabilne ekstremne regije (engl. *Maximally Stable Extremal Regions*, MSER). Otkrivaju se MSER-ovi radi pronalaska najvjerojatnijeg područja znaka ograničenja brzine. Prednost MSER detekcije je u tome što je robusna u raznim uvjetima okoline. Dobiveni ROI se izdvaja iz ulazne slike

i mijenja mu se rezolucija na 64x64 elemenata slike, budući da je slika te rezolucije potrebna kao ulaz u fazu prepoznavanja. Za treći korak su predložena dva pristupa, HOG-SVM pristup (engl. *Histogram of Oriented Gradients - Support vector machine*) i prepoznavanje ograničenja brzine pomoću CNN. HOG izdvajanje značajki prvo dijeli sliku u ćelije 4x4 elemenata slike, a zatim se pomoću operatora maske izračunavaju gradijenti svake ćelije, pri čemu gradijenti označavaju veličinu i smjer za svaku ćeliju. Histogram gradijenata svake ćelije je generiran s devet spremnika, koristeći vrijednosti magnitude i smjera gradijenta. U konačnici, ćelije se kombiniraju u blokove veličine 2x2 ćelije i normalizacija bloka primjenjuje se na svaki blok pomoću L^2 norme, s veličinom preklapanja bloka jedan. Na kraju, izračunati karakteristični vektor je veličine 8100 (15x15x2x2x9), što predstavlja značajku HOG kandidata za ograničenje brzine i to je ulaz u obučeni SVM za klasifikaciju znakova. Prepoznavanje prometnog znaka ograničenja brzine pomoću CNN ulaznu RGB sliku šalje konvolucijskom sloju. Konvolucijski sloj koristi skup filtara koji se mogu naučiti (čiji se parametri procesom učenja zapravo podešavaju), kako bi otkrio prisutnost određenih značajki ili uzoraka prisutnih na izvornoj slici. Koriste se filtri veličine 3x3 s dubinom jednakom ulaznoj slici te nakon filtra se primjenjuje aktivacijska funkcija. Sljedeći sloj sažimanja koristi se za smanjenje količine parametara i izračunavanja u mreži. Ovi slojevi se koriste kontinuirano za dobivanje sve specifičnijih značajki slike. Slojevi konvolucije i sažimanja izdvajaju značajke i smanjuju broj parametara iz ulaznih slika. Potpuno povezani sloj i izlazni slojevi koriste se za klasifikaciju izlaza. Povezani sloj klasificira se na temelju značajki izdvojenih u prethodnim slojevima. Tablica 2.3. prikazuje detaljnu strukturu korištene CNN.

Tablica 2.3. *Detaljna struktura korištene CNN[13]*

Ime sloja	**Dimenzija**	**Broj težina**	**Broj *bias*-a**	**Ukupan broj parametara**
Ulazni sloj	64x64x3	0	0	0
Konvolucijski sloj 1 (16)	64x64x16	432	16	448
Sloj sažimanja 1	32x32x16	0	0	0
Konvolucijski sloj 2 (32)	32x32x32	4608	32	4640
Sloj sažimanja 2	16x16x32	0	0	0
Konvolucijski sloj 3 (64)	16x16x64	18432	64	18496
Sloj sažimanja 3	8x8x64	0	0	0
Konvolucijski sloj 4 (64)	8x8x64	36864	64	36928
Sloj sažimanja 4	4x4x64	0	0	0
Konvolucijski sloj 5 (64)	4x4x64	36864	64	36928
Potpuno povezani sloj (6)	6x1	40960	6	40967

Za treniranje je korišten skup podataka GTSRB (engl. *German Traffic Sign Recognition Benchmark*), dok je za testiranje korišten set podatka GTSDB (engl. *German Traffic Sign Detection Benchmark*). Iz svakog navedenog skupa podataka korišteno je šest klasa prometnih znakova, a svaka klasa sadrži 150 slika. Tablica 2.4. prikazuju rezultate testiranja za obje metode, odnosno točnost predloženog rješenja, pri čemu je točnost izračunata prema formuli 2.2.

Tablica 2.4. *Rezultati testiranja predloženog rješenja[13]*

Klasifikator/znak	80	50	100	70	60	30
HOG-SVM	100%	100%	100%	100%	100%	100%
CNN	97%	97.8%	99.2%	99.8%	98.6%	99.1%

Prema [14], detekcija i klasifikacija znakova za ograničenje brzine izvršava se u sljedeće tri faze. Prvi korak prve faze obrade obuhvaća pronalazak kružnog oblika na danoj slici. Slika se prvo obrađuje tako da se smanjuje prostor za pretraživanje željenog prometnog znaka na slici metodom segmentacije pomoću formule (2.6.), koja je prethodno objašnjena. Sljedeći korak u otkrivanju prometnih znakova izričitih naredbi je primjena Canny detektora, kako bi se pronašli rubovi svih objekata na slici. Nakon Canny detektora primjenjuje se Houghovoa transformacija za detekciju krugova za pronalazak kružnih oblika. Druga faza uključuje segmentaciju znamenki. Za segmentaciju znamenki potrebno je izdvojiti odabrani objekt iz scene, binarizirati ga pomoću Otsu algoritma i promijeniti rezoluciju izdvojenog objekta na 50x50 elemenata slike. Zatim se izračunava suma svakog stupca izrezane slike. Sume stupaca se koriste da bi se odredile pozicije na x-osi izdvojene slike na kojima svaka znamenka počinje i završava. Ovaj korak je ključan za segmentaciju svake znamenke prometnog znaka. Jednom kada se znamenke segmentiraju, one se normiraju i izračunava se krivulja svojstva svake znamenke. Krivulja svojstva predstavlja raspodjelu gustoće elemenata slike u svakom stupcu kojim je prikazan dio znamenke. Krivulje se izglađuju lokalnom regresijom s težinskim faktorima (engl. *Locally Weighted Regression*) koja je objašnjenja u [15]. Izglađene krivulje segmentiranih znamenki se koriste za treniranje SVM klasifikatora u svrhu prepoznavanja znamenki. Obučeni SVM klasifikator koristi se za razvrstavanje znamenki segmentiranog prometnog znaka ograničenja brzine u nepoznatim scenama. Predloženo rješenje testirano je na setu od 210 slika koji sadrži 213 prometnih znakova ograničenja brzine i 288 ostalih prometnih znakova. Slike korištene za testiranje predloženog rješenja prikupljene su pod različitim uvjetima okoline (sunčano, oblačno, maglovito i snježno) i različitim pozadinama. Tablica 2.5. prikazuje rezultate testiranja. Točnost predloženog rješenja je

97,81%, preciznost je 99,51% i odziv je 95,31%. Predloženi pristup stvorio je samo 1 FP rezultat i 10 FN rezultata od 501 znaka prisutnih na testiranju.

Tablica 2.5. *Rezultati predloženog rješenja[14]*

	Broj znakova	**Broj TP**	**Broj FN**	**Broj TN**	**Broj FP**
Znak ograničenja brzine 30km/h	47	22	0	24	1
Znak ograničenja brzine 50km/h	42	19	1	22	0
Znak ograničenja brzine 60km/h	51	23	0	28	0
Znak ograničenja brzine 70km/h	87	30	2	55	0
Znak ograničenja brzine 80km/h	51	24	0	27	0
Znak ograničenja brzine 90km/h	69	29	1	39	0
Znak ograničenja brzine 100km/h	27	10	0	17	0
Znak ograničenja brzine 110km/h	38	13	2	23	0
Znak ograničenja brzine 120km/h	29	8	2	19	0
Znakovi koji nisu znakovi ograničenja brzine	60	25	2	33	0
Ukupmo	501	203	10	287	1

U radu [16] opisana je metoda za automatsko prepoznavanje ograničenja brzine na prometnim znakovima ograničenja brzine snimljenih pomoću kamere. Ova se metoda sastoji od sljedeće tri faze: prva faza obuhvaća detekciju znakova ograničenja brzine pomoću strojnog učenja, korištenjem značajki lokalnih binarnih uzoraka kao informacija koja pomaže u identifikaciji. Zatim se slika obrađuje u prostora boja HSV (engl. *Hue, Saturation and Value*), kako bi se izdvojili brojevi ograničenja brzine na identificiranim znakovima ograničenja brzine. Konačno, za prepoznavanje izdvojenih brojeva koristi se neuronska mreža. Rad [16] usredotočen je na razmatranje sljedeća tri zadatka:

- omogućiti otkrivanje samo znaka ograničenja brzine na slici scene pomoću jednog postupka koji se fokusira na lokalne obrasce znakova ograničenja brzine.
- omogućiti odvajanje i izdvajanje dvoznamenkastih brojeva na znaku ograničenja brzine u slučajevima kada su dvocifreni brojevi pogrešno izdvojeni kao jedno područje zbog svjetlosnog okruženja.

- omogućiti prepoznavanje brojeva pomoću neuronske mreže fokusiranjem na tri veličine svojstva.

Rad [17] predstavlja sustav prepoznavanja prometnih znakova koji se zasniva na strojnom učenju za otkrivanje, segmentaciju i klasifikaciju znakova u stvarnom vremenu. U ovom radu prikazan je algoritam za prepoznavanje predložaka prometnih znakova pomoću tehnike optičkog prepoznavanja znakova (engl. *Optical Character Recognition,* OCR). Prvi korak je detekcija znaka. Kako je crvena boja interesna boja, budući da znakovi imaju crveni obrub, prije svega se crvene komponente izdvajaju od ostalih komponenata boja. Ovo izdvajanje crvene boje iz ostatka boja temeljni je korak u fazi detekcije. Nakon ekstrakcije crvene boje, primjenjuje se medijan filter za uklanjanje šuma. Rubovi postaju oštriji i jasniji, što pomaže u otkrivanju prometnih znakova. Nakon medijan filtera primijenjena je funkcija praga (engl. *threshold*) čime je dobivena binarna slika. Sljedeći korak u fazi detekcije je primjena morfoloških operacija (dilatacija i erozija). Morfološke operacije ovise samo o relativnom redoslijedu vrijednosti elemenata slike, a ne o njihovim numeričkim vrijednostima i zbog su pogodne za obradu binarnih slika. Nakon primjene filtriranja i morfoloških operacija, dobiva se maska koja se koristi za detekciju prometnog znaka. Na izvornu sliku se primjenjuje operacija konvolucije pomoću prethodno spomenute maske, kako bi se detektirao prometni znak. Nakon uspješnog otkrivanja prometnog znaka, sljedeći korak je prepoznavanje znaka. Tehnika koja se koristi u tu svrhu je OCR. Usklađivanje predložaka (engl. *template matching*) je prototip sustava koji je važan za prepoznavanje znakova ili abecede usporedbom dviju slika znaka. Za savršeno otkrivanje znaka uz minimalan šum provodi se daljnje filtriranje. Ti filtri uključuju *Laplacian*, *Sobel* i *Average* filtre, a filteri se uglavnom koriste kako bi poboljšala vidljivost rubova otkrivenih znakova i uklonili svi mali preostali šumovi. Ovaj pristup određuje sličnost danog predloška i prozora iste veličine na slici i pronalaženje prozora koji stvara najveći omjer sličnosti. Testiranje predloženog rješenja je izvršeno na slikama i videozapisima Pakistanskog urbanog i ruralnog područja, rezolucije 640x480 elemenata slike. Testni skup sadrži 115 slika i 20 videozapisa s različitih cesta s različitim uvjetima okoline. Sve prikupljene slike snimljene su iz fiksnog položaja u vozilu. Točnost sustava je izračunata prema formuli 2.2. i iznosi 97,5%. Predloženo rješenje klasificira 314 slika kao TP i 271 kao FP slike.

U radu [18] razvijeno je rješenje za detekciju i prepoznavanje prometnih znakova ograničenja brzine koristeći Visual Studio i OpenCV biblioteke. Na ulaznu sliku koja prolazi kroz sustav prvo je potrebno primijeniti osnovne operacije predobrade kako bi se poboljšala njena kvaliteta. Tada se vjerojatnost

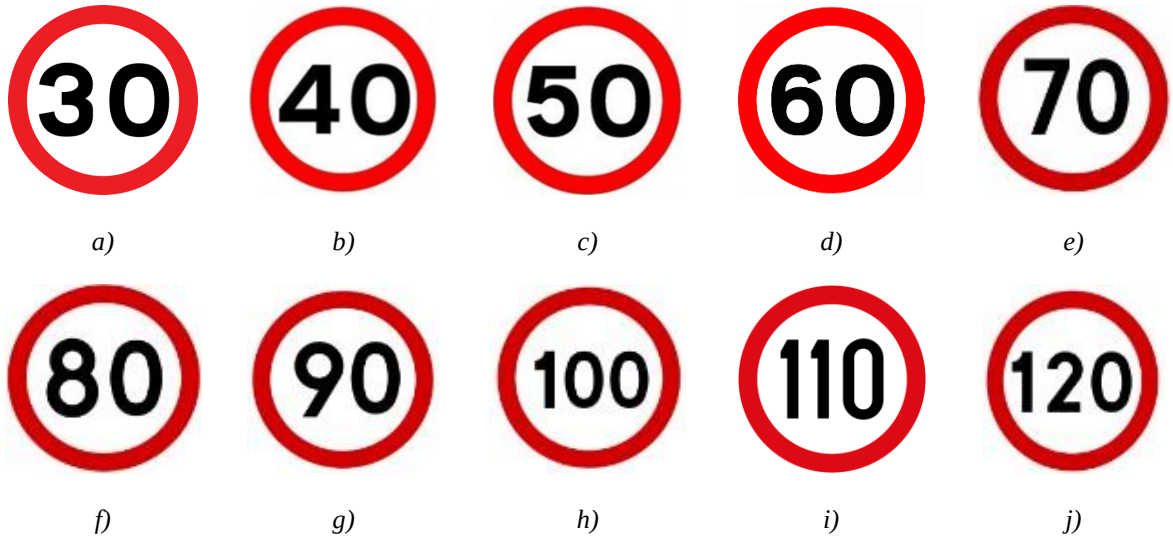
boje svakog elementa slike može izračunati pomoću modela vjerojatnosti boje koji je predložen u[19]. Izračun vjerojatnosti boja se izvodi prije izvođenja predloženog rješenja i u projekt će se uvesti tablice (engl. *LookUp Table*) veličine 256 × 256 × 256 × N, pri čemu je *N* broj boja prometnog znaka. Nakon ovog koraka, ulazna slika pretvara se u sliku u skali sive boje kod koje vrijednost svakog elementa slike odgovaraju vjerojatnosti boje pri čemu visok intenzitet u skali sive boje predstavlja prisutnost boje prometnog znaka. Dakle, područja na slici gdje je intenzitet svjetline izražen, na slici u skali sive boje, predstavljaju moguće kandidate prometnih znakova za ograničenje brzine. Da bi se odredili znakovi ograničenja brzine među tim crvenim područjima, koristi se svojstvo znaka ograničenja brzine koji ima 2 - 3 crne znamenke na bijeloj pozadini unutar crvenog kruga. Sljedeća faza je faza segmentacije znamenki i prepoznavanja znamenki. Za otkrivanje sadržaja unutar crvenog kruga primjenjuje se metoda adaptivnog praga. Umjesto da se koristi vrijednost globalnog praga, ova tehnika razdvaja sliku na male regije i izračunava vrijednosti lokalnog praga. Zatim se izdvajaju konture binarne slike pomoću vrijednosti lokalnog praga. Kako bi se osiguralo da su objekti segmentirani pomoću adaptivnog praga znamenke, konture objekata moraju biti provjerene kako bi se utvrdilo zadovoljavaju li zahtjeve svojstva znamenki ili ne. Kandidati koji prođu fazu eliminacije bit će definirani kao znamenke. Značajke ovih slika izdvojit će se i klasificirati sa SVM klasifikatorom od 10 klasa, koje odgovaraju brojevima od 0 do 9. Svaka se znamenka prepoznaje i sprema u slijed s lijeva na desno u memoriju. Tada će se znamenke kombinirati kako bi postale broj koji odgovara vrijednosti podataka u otkrivenom prometnom znaku ograničenja brzine. Slike na kojima je izvršeno testiranje su snimljene u različitim uvjetima okoline kao što su: dnevno svjetlo, slabo osvjetljenje i noć. Za prikupljanje slika korištena je kameru od 12 megapiksela te je rezolucija svake slike 5184 × 3456 elemenata slike. Ukupan broj slika u testnom skupu je 300, od kojih je 250 slika snimljeno u normalnom osvjetljenju i 50 slika u slabom osvjetljenju. Jedna od karakteristika ovih slika je njihova kompliciranа pozadina koja sadrži mnogo različitih vrsta vozila (motocikl, automobil i bicikl), oglasne ploče i oglase. Točnost sustava iznosi 98.4%.

3. PREDLOŽENO RJEŠENJE ZA PREPOZNAVANJE PROMETNIH ZNAKOVA OGRANIČENJA MAKSIMALNE BRZINE KRETANJA ZA PRIMJENU U NAPREDNIM ADAS ALGORITMIMA

U ovom poglavlju objašnjeno je rješenje izrađeno u sklopu ovog diplomskog rada, odnosno algoritam koji rješava prethodno predstavljene probleme detekcije i prepoznavanja prometnih znakova ograničenja brzine. Iako u prometu postoje razni tipovi znakova, u ovom je radu fokus stavljen samo na prometne znakove ograničenja brzine. Rješenje se sastoji od dva dijela. Prvi dio rješenja ima za zadatak detektirati prometne znakove ograničenja brzine dok drugi dio rješenja ima za zadatak ispravno prepoznati znakove ograničenja brzine odnosno klasificirati ih. Prije same izrade rješenja postavljeni su zahtjevi na samo rješenje. Iako postoji više različitih znakova ograničenja brzine, u ovom diplomskom radu zadatak je bio ispravno prepoznati sljedeće prometne znakove ograničenja brzine:

- Znak ograničenja brzine 30 km/h (slika 3.1. (a))
- Znak ograničenja brzine 40 km/h (slika 3.1. (b))
- Znak ograničenja brzine 50 km/h (slika 3.1. (c))
- Znak ograničenja brzine 60 km/h (slika 3.1. (d))
- Znak ograničenja brzine 70 km/h (slika 3.1. (e))
- Znak ograničenja brzine 80 km/h (slika 3.1. (f))
- Znak ograničenja brzine 90 km/h (slika 3.1. (g))
- Znak ograničenja brzine 100 km/h (slika 3.1. (h))
- Znak ograničenja brzine 110 km/h (slika 3.1. (i))
- Znak ograničenja brzine 120 km/h (slika 3.1. (j))

Slika 3.1. prikazuje prometne znakove koje rješenje treba prepoznati. U sljedećim potpoglavljima opisano je razvojno okruženje Vision SDK te alati i metode koji su korišteni prilikom izrade programskog rješenja, kao i samo programsko rješenje.

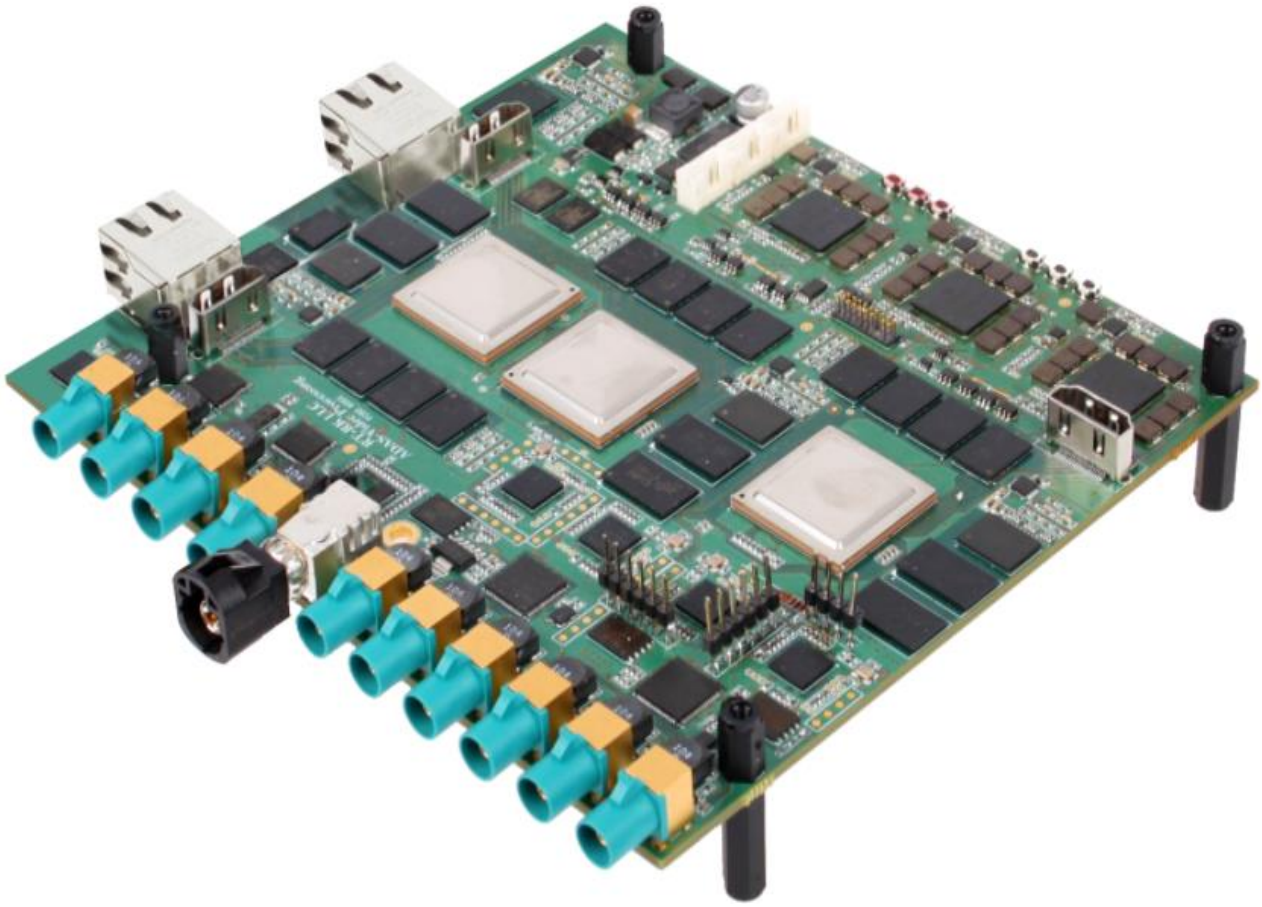


Sl. 3.1. *Prometni znakovi koje algoritam treba prepoznavati (a) znak ograničenja brzine 30 km/h, (b) znak ograničenja brzine 40 km/h, (c) znak ograničenja brzine 50 km/h, (d) znak ograničenja brzine 60 km/h, (e) znak ograničenja brzine 70 km/h, (f) znak ograničenja brzine 80 km/h, (g) znak ograničenja brzine 90 km/h, (h) znak ograničenja brzine 100 km/h, (i) znak ograničenja brzine 110 km/h, (j) znak ograničenja brzine 120 km/.,*

3.1. Razvojno okruženje Vision SDK i ADAS ploča

Kako bi ADAS obavljao određene funkcije u vožnji bez potrebe za intervencijom vozača, potrebno mu je ugraditi umjetnu inteligenciju, tj. razviti algoritme koji će u stvarnom vremenu obrađivati signale koji dolaze s različitih senzora postavljenih u vozilu i implementirati ih u ADAS[20]. Pritom su na raspolaganju različiti senzori (dinamički senzori, ultrazvučni senzori, RADAR, LIDAR, video kamere) i ostalo sklopovlje u automobilu koje omogućava stvarno-vremensku obradu signala iz okoline. Središnji dio ADAS-a je ADAS ploča, na koju se dovode signali s različitih senzora, a koji se na njoj onda obrađuju u stvarnom vremenu te se na osnovu obrađenih signala dolazi do zaključka koju radnju obaviti pri samoj vožnji. Konkretna ADAS razvojna ploča koja se koristila u ovom diplomskom radu je ALPHA razvojna ploča prikazana na slici 3.2.[21]. Dizajnirana je tako da pruža podršku za osnovne i napredne sustave upozoravanja, aktivne sustave za upravljanje, polu-autonomne operacije. Skup ciljanih aplikacija obuhvaća funkcionalnosti vezane za sustav kamera na vozilu: pogled odostraga (engl. *rear view*), prepoznavanje prometnih znakova (engl. *traffic sign recognition*), pogled oko automobila (engl. *surround view*), pomoć pri parkiranju i u prometu (engl. *parking and*

traffic assistance), praćenje vozača unutar kabine (engl. *in-cabin driver monitoring*), automatska vožnja na autocesti (engl. *highway lane self-driving*) itd. ADAS razvojna ploča koja se koristi sastoji se od tri sustava na čipu (engl. *System on a Chip*, SoC) u sklopu kojih se nalaze procesori na kojima se izvršavaju zadani algoritmi.



Sl. 3.2. *Prikaz ALPHA ploče[17]*

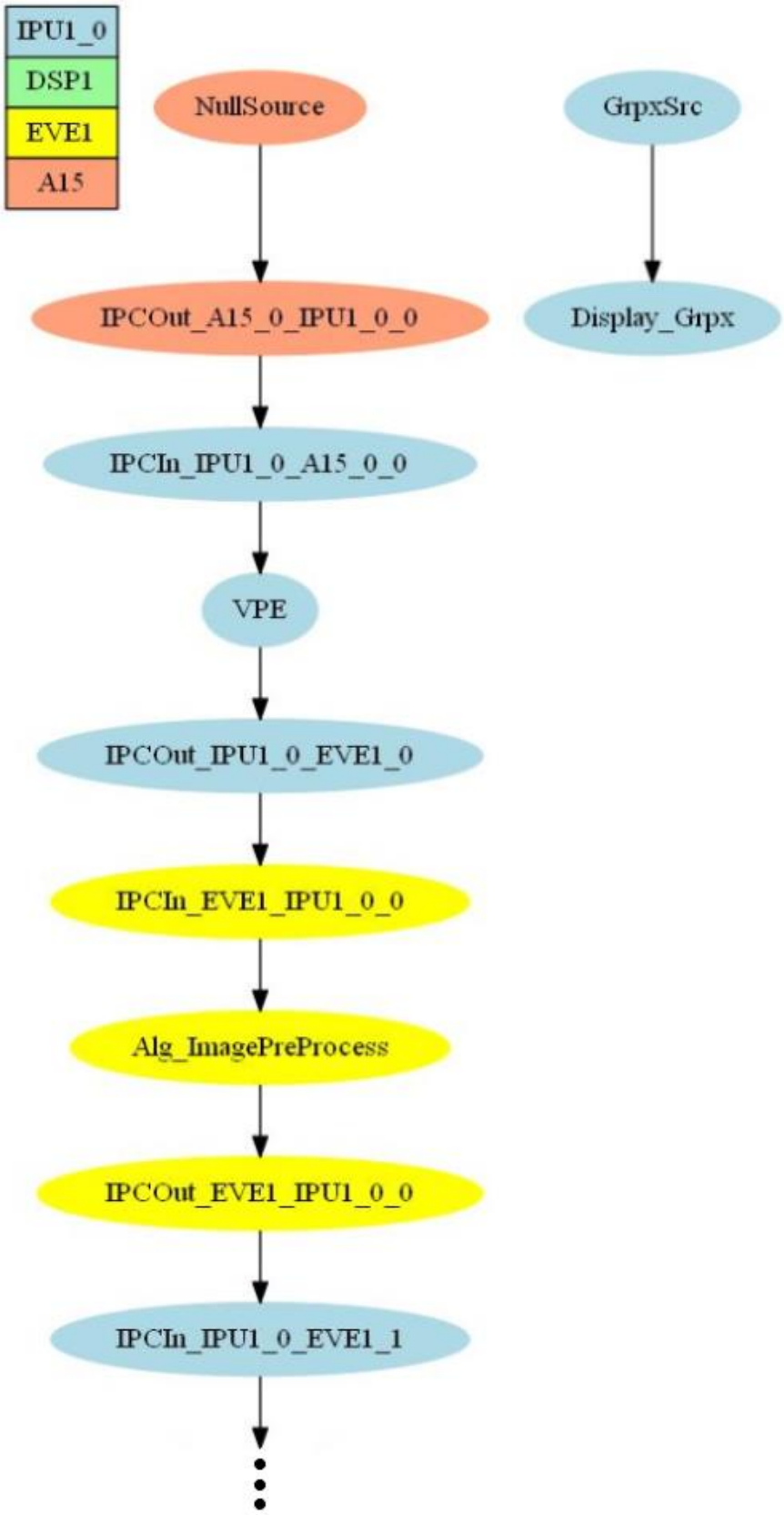
Za potrebe programiranja obitelji ADAS SoC-ova, tvrtka Texas Instruments razvila je više-procesorsku softversku razvojnu platformu, nazvanu Vision SDK. VisionSDK je softverski okvir koji omogućava korisnicima stvaranje različitih tokova podataka slučajeva upotrebe (engl. *Use Case*), uključujući preuzimanje/snimanje videa, pred-obradu videa, algoritme za analizu videa i prikaz videa. Omogućava podršku da se obrada različitih tokova podataka izvršava na različitim procesorskim jedinicama (engl. *Central Processing Unit*, CPU) i pomoću različitih sklopovskih akceleratora, s ciljem učinkovitog korištenja različitih SoC-ova. VisionSDK dolazi s pokaznim primjerima tokova podataka koji demonstriraju korištenje raznih procesora i hardverskih akceleratora u SoC-ovima, a

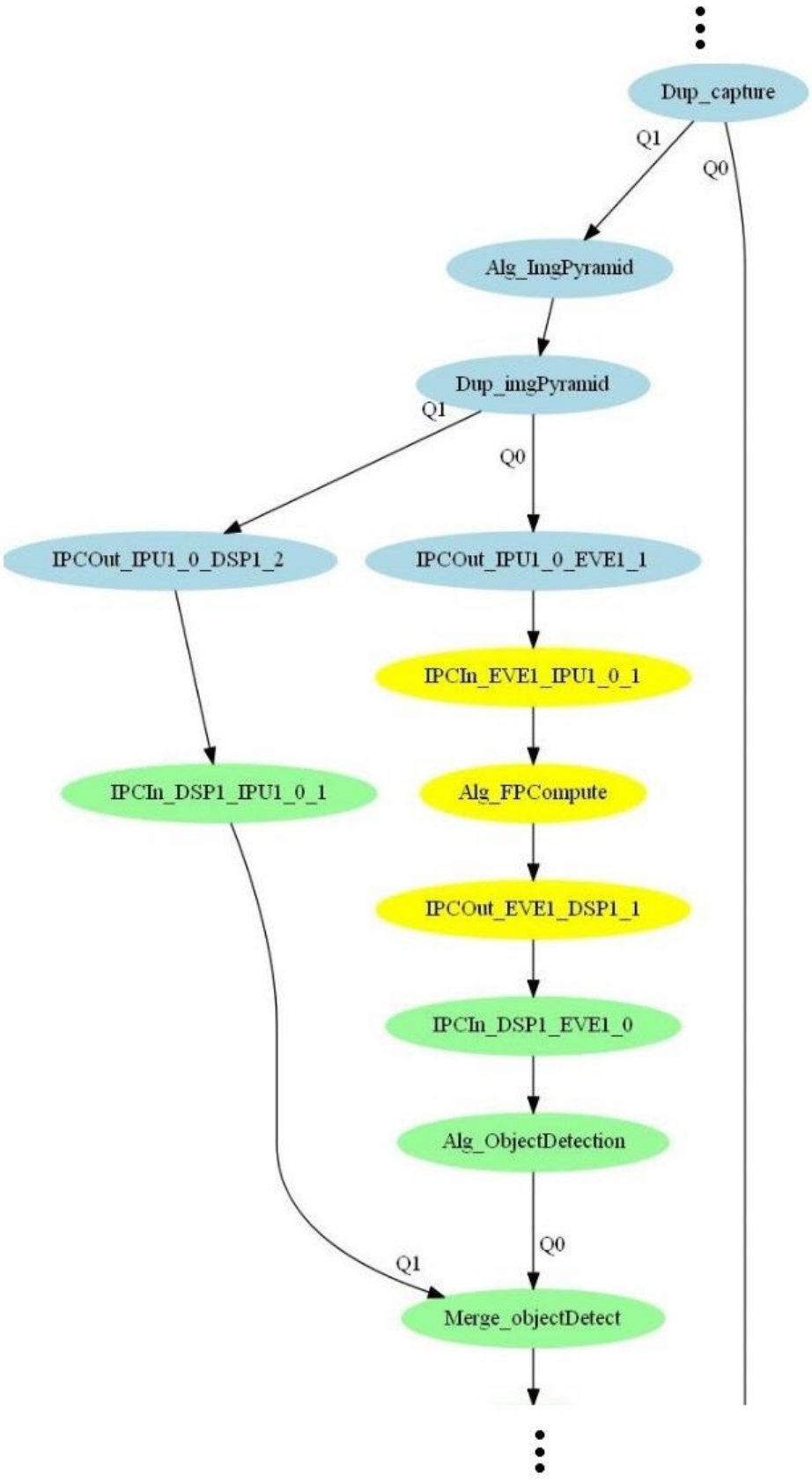
koji pokazuju korisnicima kako koristiti pojedine SoC pod-sustave. Sam VisionSDK zasnovan je na okviru (engl. f*ramework*) zvanom „Links and Chains“ („Veze i Lanci“), a korisničko programsko sučelje (engl. *Application Programming Interface,* API) ovog okvira zove se „Link API“. Paket za instalaciju SDK dolazi sa svim alatima i komponentama potrebnim za kompilaciju aplikacija (alati za kodiranje, BIOS (engl. *Basic Input/Output System*), kodeci, stogovi umrežavanja, jezgre algoritama,...).

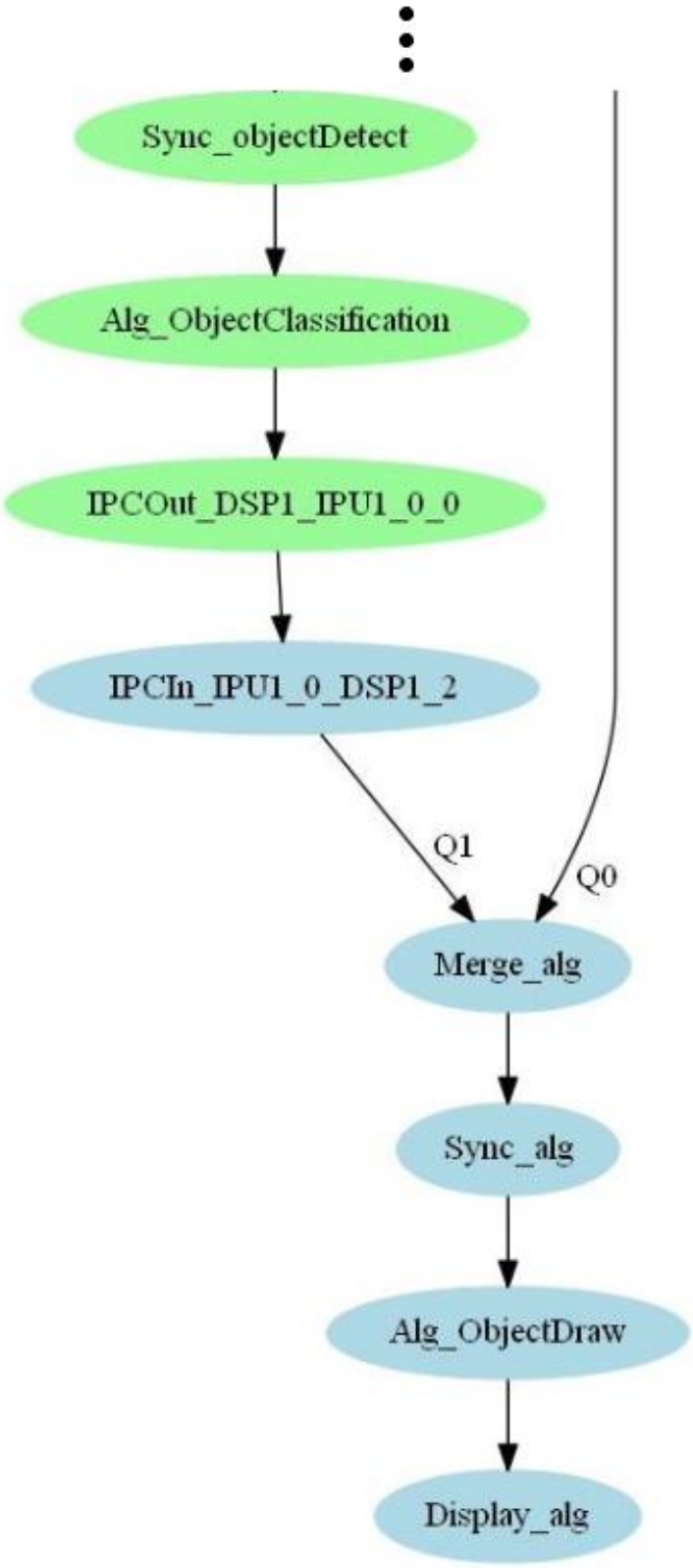
Tijekom izgradnje programskog rješenja stvaraju se dvije datoteke naziva *AppImage* i *MLO,* koje je potrebno pohraniti na microSD karticu koja se zatim umetne u pripadni utor na ADAS ALPHA ploči kako bi se programsko rješenje moglo pokrenuti na razvojnoj ploči.

3.2. Korišteno programsko rješenje za detekciju prometnih znakova

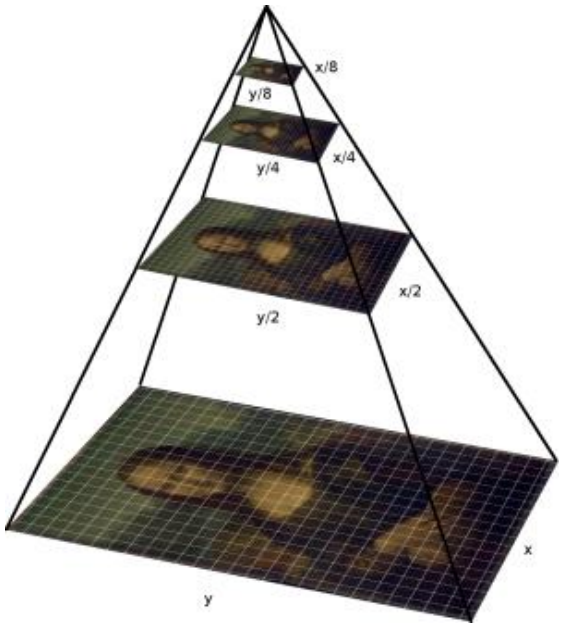
Za detekciju prometnih znakova na danoj sceni korišten je *Object Detection* algoritam koji dolazi s Vision SDK razvojnim okruženjem. Kao i svi ostali algoritmi koji dolaze s Vison SDK okruženjem, *Object Detection* nije javno dostupan algoritam, odnosno nije moguć uvid u samu srž algoritma. Međutim, poznato je da je zasnovan na strojnom učenju, dok sami detalji nisu dostupni. Dani algoritam sastoji se od nekoliko faza, a dijagram toka prikazan je na slici 3.3. Prva faza je *ImagePreProcess* koja radi osnovnu predobradu slike. Predobrada je uobičajeni naziv za operacije sa slikama na najnižoj razini apstrakcije - i ulaz i izlaz su slike intenziteta svjetline. Cilj prethodne obrade je poboljšanje slikovnih podataka koji suzbijaju neželjena izobličenja ili poboljšavaju neke značajke slike važne za daljnju obradu. Zatim se ulazna slika dijeli na dva kanala, pri čemu se u jednom kanalu čuva originalna ulazna slika, a u drugom kanalu se odvija daljnja obrada slike. Sljedeći algoritam koji se primjenjuje u kanalu za daljnju obradu slike je *Image Pyramid.* Piramida slike, formalno nazvana "piramidalna reprezentacija slike" je tehnika obrade slike i signala za predstavljanje jedne slike pomoću skupa kaskadnih slika. Slikovna piramida pruža mnoga korisna svojstva za mnoge aplikacije, poput smanjenja šuma, analize slike, poboljšanja slike itd. Piramida slike je zapravo prikaz slike nizom slika različitih rezolucija. Primjer piramide slike prikazan je na slici 3.4.[22], pri čemu *x* i *y* predstavljaju početnu rezoluciju. Jedna od primjena piramidalne reprezentacije slike koja se koristi upravo za algoritam *Object Detection* je podudaranje uzoraka: da bi se pronašao određeni ciljani uzorak koji se može pojaviti na bilo kojoj skali unutar slike, potrebna je piramidalne reprezentacije slike.


IPU1_0
DSP1
EVE1
A15
NullSource
GrpxSrc
IPCOut_A15_0_IPU1_0_0
Display_Grpx
IPCIn_IPU1_0_A15_0_0
VPE
IPCOut_IPU1_0_EVE1_0
IPCIn_EVE1_IPU1_0_0
Alg_ImagePreProcess
IPCOut_EVE1_IPU1_0_0
IPCIn_IPU1_0_EVE1_1


Dup_capture
Q1
Q0
Alg_ImgPyramid
Dup_imgPyramid
Q1
Q0
IPCOut_IPU1_0_DSP1_2
IPCOut_IPU1_0_EVE1_1
IPCIn_EVE1_IPU1_0_1
IPCIn_DSP1_IPU1_0_1
Alg_FPCompute
IPCOut_EVE1_DSP1_1
IPCIn_DSP1_EVE1_0
Alg_ObjectDetection
Q0
Q1
Merge_objectDetect



Sl. 3.3. *Dijagram toka programskog rješenja detekcije prometnih znakova*



Sl. 3.4. *Primjer piramide slike[20]*

Daljnja obrada podrazumijeva fazu detekcije. Algoritam je napravljen tako da može tražiti pješake, prometne znakove i automobile, što se odabire pri pokretanju algoritma. Algoritam radi na način da traži zadane objekte na danoj sceni. Informacije o pronađenom objektu, kao što su širina i visina ocrtanog graničnog okvira i koordinate gornjeg lijevog ugla ocrtanog graničnog okvira, spremaju se u strukturu. Dakle izlaz ovog algoritma je struktura s informacijama o detektiranom objektu koja se dalje prosljeđuje algoritmu za klasifikaciju. Algoritam za klasifikaciju prima prethodno spomenutu strukturu koja sadrži pronađeni objekt, te klasificira pronađeni objekt. Informacije o klasificiranom objektu se spremaju u novu strukturu koja se prosljeđuje algoritmu za iscrtavanje okvira oko pronađenog i klasificiranog objekta. Zadnji korak *Object Detection* algoritma je iscrtavanje graničnog okvira oko pronađenog i klasificiranog objekta te u slučaju prepoznavanja prometnih znakova, iscrtavanje referentnog obrasca pronađenog znaka. Za ovaj korak potrebno je spojiti dva kanala, kanal s originalnom ulaznom slikom i kanal u kojem se odvijala obrada slike. Iz kanala u kojem se odvijala obrada slike prikupljene su informacije o poziciji, veličini i klasi pronađenog objekta te se navedene informacije iscrtavaju na originalnoj ulaznoj slici, pri čemu navedeni zadatak izvršava algoritam *Object Draw*. Krajnji rezultat ovog algoritma je ocrtani granični okvir oko pronađenog objekta te u slučaju prepoznavanja prometnih znakova, iscrtavanje pronađenog znaka, kao što se može vidjeti na slici 3.5.

Sl. 3.5. *Rezultat detekcije i klasifikacije Object detection algoritma*

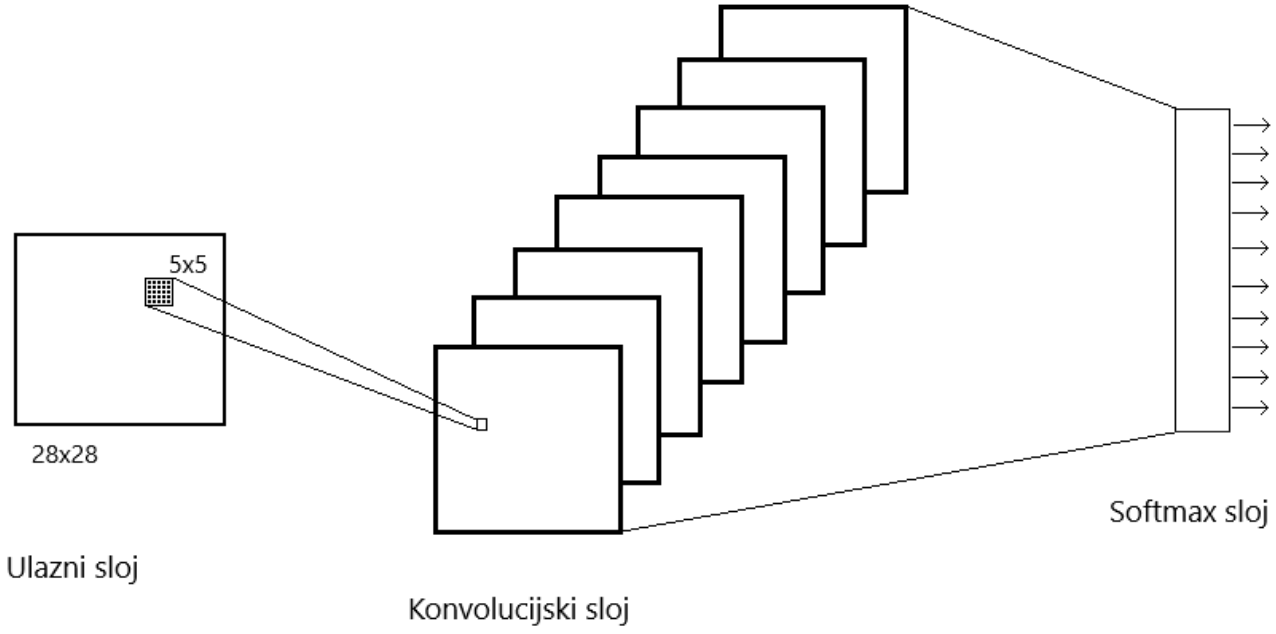
3.3. Kreirano programsko rješenje za klasifikaciju prometnih znakova

Nakon što je prethodno objašnjen algoritam *Object detection* detektirao prometni znak, potrebno je klasificirati pronađeni prometni znak. Za klasifikaciju detektiranih prometnih znakova kreirana je CNN. Ulazni podatak CNN je slika u skali sive boje rezolucije 28x28 elemenata slike, odnosno vektor veličine 784 elemenata. Slika 3.6. prikazuje ulazni podatak CNN.

Sl. 3.6. *Ulazni podatak u CNN*

Za potrebe izrade diplomskog rada cilj je bio implementirati CNN na zadanu ADAS razvojnu ploču, što znači da je prilikom kreiranja mreže bilo potrebno obratiti pozornost na procesorske mogućnosti zadanog sklopovlja. Stoga umjesto velikog broja konvolucijskih slojeva, željena CNN sadrži samo

jedan konvolucijski sloj s 8 filtara veličine 5x5 elemenata slike i *softmax* sloj. Struktura korištene CNN prikazana je na slici 3.7.



Sl. 3.7. *Shematski prikaz korištene CNN za klasifikaciju prometnih znakova ograničenja brzine*

CNN je kreirana u dva programska jezika, programskom jeziku Python za potrebe treniranja CNN na računalu i u programskom jeziku C za implementaciju CNN na ADAS razvojnu ploču. CNN je za potrebe treniranja kreirana kao Python skripta, a njeno izvršavanje je ostvareno putem Anaconda terminala koji dolazi s Anaconda distribucijom Pythona. Uz programski jezik Python korišten je TensorFlow[23], javno dostupni razvojni okvir za Python, koji omogućuje numeričke proračune visokih performansi. Prilikom realizacije mreže, nisu korištene varijable već *tensori*. Tensor predstavlja n-dimenzionalnu matricu pa samim time i podatak. Tensor dimenzije 0 naziva se skalar i predstavlja jedan broj, *tensor* dimenzije 1 naziva se vektor, a dimenzije 2 matrica[24]. Ulazni podatak se pohranjuje u *tensor* dimenzije 1, veličine 784 elemenata, izlazni podatak se sprema u *tensor* dimenzije 1, veličine 10 elemenata, pri čemu svaki element izlaznog vektora predstavlja vjerojatnost da se na slici nalazi odgovarajući prometni znak. Zbroj svih elemenata izlaznog vektora jednak je 1. Nadalje, bilo je potrebno definirati *tensore* koje odgovaraju vrijednostima težina W_{ij} i *biasima* b_i koji predstavljaju parametre filtara. Broj težina odgovarat će umnošku broja elemenata slike i broja prometnih znakova koje je potrebno klasificirati, dok će broj *biasa* odgovarati broju prometnih

znakova. Kako bi se izgradio konvolucijski sloj, odnosno realizirala operacija 2D konvolucije, u okviru TensorFlow-a koristi se funkcija *conv2d* čiji je prototip dan na slici 3.8.

```
tf.nn.conv2d(input, filters, strides, padding, data_format='NHWC', dilations=None, name=None)
```

Sl. 3.8. *Prototip funkcije conv2d korištene za izgradnju konvolucijskog sloja*

Parametar input je ulazni *tensor*. Parametar *filters* je konvolucijski filtar koji će se koristiti za konvoluciju oblika [visina filtra, širina filtra, ulazni kanali, izlazni kanali]. Visina i širina filtra za ovu mrežu je 5. Broj ulaznih kanala postavlja se na broj slojeva prethodnog sloja (u ovom slučaju 1) dok se broj izlaznih postavlja na dubinu sloja filtra koji definiramo (u ovom slučaju 8). *Strides* definira broj koraka pri pomicanju filtra po ulaznoj slici i oblik mu je [1, pomak, pomak, 1], za potrebe ove mreže *stride* je podešen na [1,1,1,1]. Parametar *padding* se postavlja na vrijednost „VALID“ što znači da filter ne smije izaći van iz originalne slike odnosno da neće moći ići od ruba do ruba, i izgubit će po dva elementa slike u svakom smjeru (ukupno četiri). Ostale parametre nije potrebno koristiti. Na rezultat konvolucijske funkcije pribraja se vrijednost *biasa* b_1 te se ukupan rezultat predaje aktivacijskoj funkciji ReLu. Rezultatu aktivacijske funkcije prvo je potrebno promijeniti veličinu *tensora,* budući da su izgubljena po dva elementa u svakom smjeru na veličinu 24x24x8 elemenata. Zatim se primjenjuje TensorFlow funkcija *matmul,* odnosno množi se *tensor* rezultata aktivacijske funkcije s *tensorom* težina drugog konvolucijskog sloja te se dobivenom umnošku pridodaju vrijednosti *biasa* b_2. Dobivena vrijednost se predaje *softmax* sloju čiji rezultat predstavlja ishod klasifikacije. Kompletan kod za programsko rješenje CNN nalazi se u datotetci cnn.py koja je dana kao elektronički prilog P.3.1 na DVD-u priloženom uz ovaj rad.

CNN kreirana za ADAS razvojnu ploču je napisana u C programskom jeziku pri čemu su filteri ostvareni pomoću *for* petlji. S obzirom da je filter veličine 5x5 elemenata slike, kao što je prethodno spomenuto, izgubit će se po dva elementa slike u svakom smjeru, stoga je veličina polja Y1 u koje se pohranjuje rezultat konvolucijske funkcije 24x24x8 elemenata (8 zbog 8 slojeva filtra). Korišteno je ukupno pet ugniježđenih *for* petlji za izračun konvolucijske funkcije. Vrijednost iteratora prve dvije petlje kreću od nule te idu do 24, treća for petlja kreće od 0 i ide do 8, zbog 8 slojeva filtra, a zadnje dvije for petlje idu od 0 do 5 zbog veličine filtera. Varijabla *sum* predstavlja rezultat konvolucijske funkcije te se u nju prvo upisuje vrijednost *biasa* b_{1k} , a zatim rezultat konvolucijske funkcije. Na slici 3.9. nalazi se prethodno opisane linije koda kojima je ostvarena konvolucija funkcija.

```
1. for (i = 0; i < 24; i++) {
2.   for (j = 0; j < 24; j++) {
3.     for (k = 0; k < 8; k++) {
4.       float sum = *(b1Algo+k);
5.       for (m = 0; m < 5; m++){
6.         for (n = 0; n < 5; n++) {
7.           sum += ((float) *(inputImage + i*28 + j + m*28 + n) / 255.0) * *(W1A
   lgo +k*5*5 + m + n*5);
8.         }
9.       }
10.      if (sum < 0)
11.        Y1[cnt] = 0;
12.      else
13.        Y1[cnt] = sum;
14.     cnt++;
15.           }
16.  }
17.}
```

Sl. 3.9. *Linije koda CNN kreirane za ADAS razvojnu ploču*

Prvi dio linije 7 odnosi se na kretanje po ulaznoj slici, te se ta vrijednost dijeli s 255 kako bi se vrijednosti normirale na iznose od 0 do 1. Zatim se množi s vrijednostima filtra koje se nalaze u polju *W1Algo*, što predstavlja parametre filtra. Nakon toga rezultat se sprema u polje Y1. Nadalje, potrebno je izračunati sljedeći sloj koji ima 10 izlaznih vrijednosti. Izlaz ovog sloja pohranjuje se u polje Y2 koje je veličine 10 i na svaki od tih 10 elemenata unutar polja, prvo se pohranjuje vrijednost *biasa* $b2_j$ a zatim se pribrajaju vrijednosti prošlog sloja pomnoženi s parametrima *W2Algo*. Sljedeći korak je *softmax* proračun u kojem se numerički izlaz zadnjeg linearnog sloja klasifikacije pretvara u vjerojatnosti, uzimajući eksponente svakog izlaza, a zatim normalizirajući svaki broj zbrojem tih eksponenata, tako da zbroj izlaznog vektora daje zbroj jednak jedan. Zadnji korak je pronaći najveću vjerojatnost, odnosno indeks elementa s najvećom vjerojatnošću predstavlja klasu koja je rezultat klasifikacije.

3.3.1. Skup podataka za treniranje, validaciju i testiranje neuronske mreže.

U sklopu izrade diplomskog rada prikupljeno je 7623 slike za treniranje, validaciju i testiranje odnosno kreiran je skup potrebnih podataka. Slike sadržane u spomenutom skupu podataka su prikupljene s raznih javno dostupnih izvora na Internetu te je velika količina slika prikupljena vlastitim snimanjem znakova na lokalnim prometnicama. Primjeri slika prikupljenih vlastitim snimanjem i s javno dostupnih izvora Interneta dani su na slici 3.8. Kada su slike prikupljene, bilo je potrebno izdvojiti znakove iz cjelokupne scene, kako bi slike bile prikladne za učenje neuronske mreže.

a)

b)

Sl. 3.8. *Slike prikupljene za skup podataka za treniranje i validaciju neuronske mreže(a) slika prikupljena vlastitim snimanje (b) slika prikupljena s javno dostupnog izvora na Internetu*

Izdvajanje znakova je napravljeno pomoću dva javno dostupna Python alata. Prvi je *OpenLabeling*[25], koji je omogućio označavanje i ocrtavanje okvira oko željenog znaka što je prikazano na slici 3.10. Zatim je korišten *PascalVOC-to-Images*[26] alat kako bi izdvojio označene okvire, što je prikazano na slici 3.11.

a)

b)

Sl. 3.10. *Slike nakon primjene alata OpenLabeling (a) slika prikupljena vlastitim snimanje(b) slika prikupljena s javno dostupnog izvora na Internetu*

a) *b)* *c)*

Sl. 3.11. *Slike nakon primjene alata PascalVOC-to-Images (a) desni znak vlastite slike (b) lijevi znak vlastite slike (c) znak slike s javno dostupnih izvora Interneta*

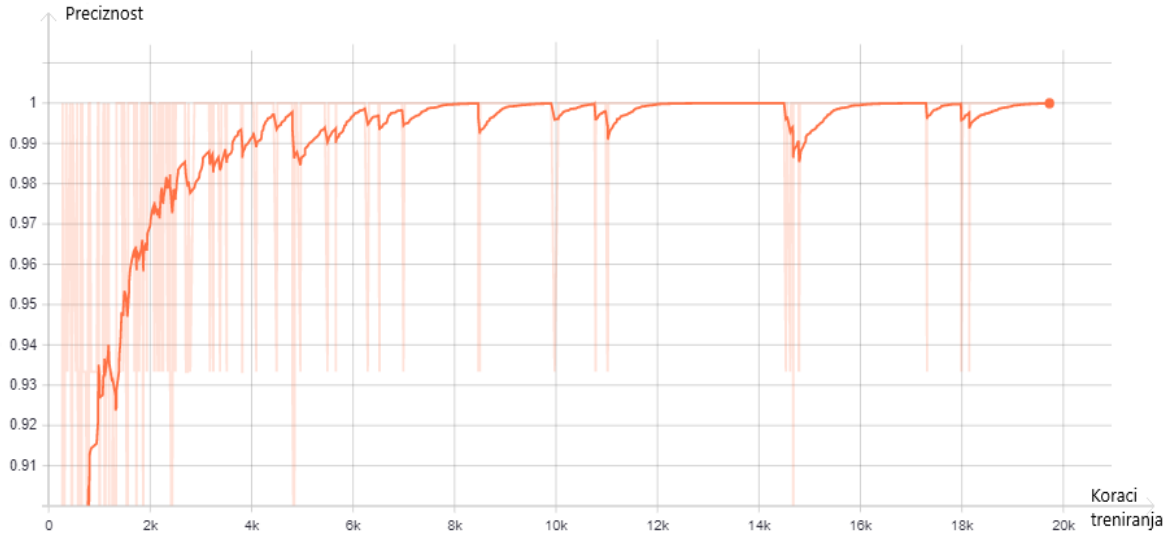
Nadalje, korišteni skup podataka je rađen po uzoru na MNIST[27] skup podataka te je korišten *JPG-and-PNG-to-MNIST*[28], također Python alat, kako bi podaci bili u istom formatu kao i MNIST skup podataka, odnosno veličine 28x28 elemenata slike i u skali sive boje. Ukupno je prikupljeno 7623 slika (3251 vlastita slika i 4372 slika s javno dostupnih izvora Interneta), a točan broj slika po određenoj klasi i raspodjela tih slika na trening skup, validacijski skup te testni skup prikazani su u tablici 3.1. Ukupan broj slika za svaku klasu je podijeljen približno u omjeru 80:10:10 odnosno 10% prikupljenih slika za pojedinu klasu je uzeto za validacijski skup (elektornički prilog P.3.2.) i 10% za testni skup (elektornički prilog P.3.3.), a ostatak je dodiljen skupu za treniranje (elektornički prilog P.3.1.).

Tablica 3.1. *Skup prikupljenih podataka za treniranje i validaciju neuronske mreže za prepoznavanje 10 tipova prometnih znakova ograničenja brzine*

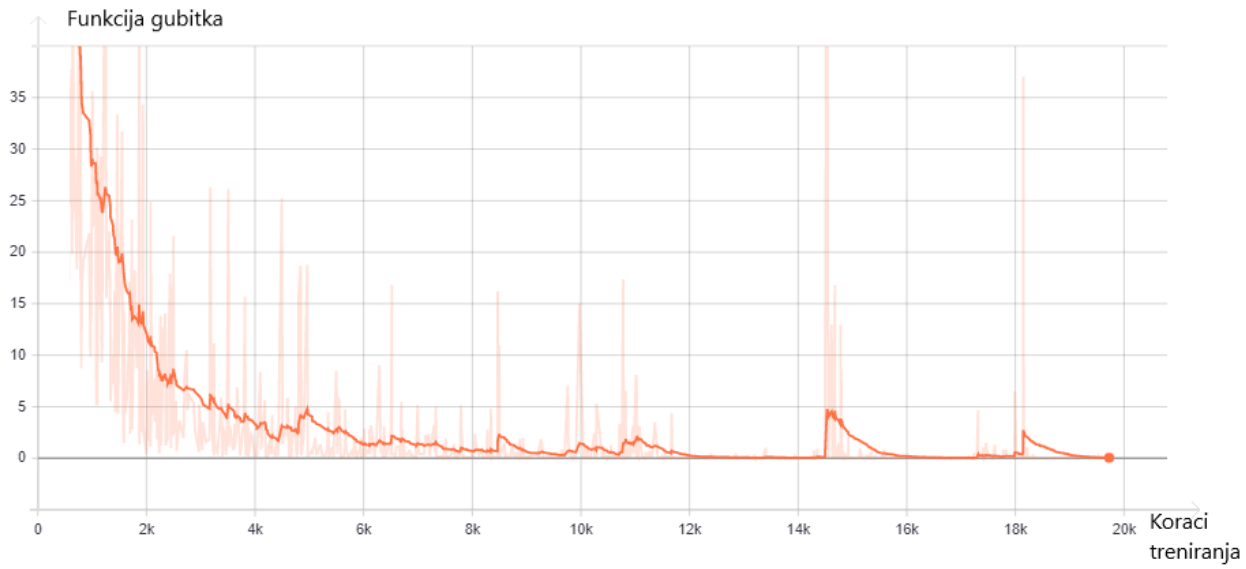
Klasa	**Trening skup**	**Validacijski skup**	**Testni skup**	**Ukupno**
Klasa 0: znak ograničenja brzine 30 km/h	1226	140	140	1506
Klasa 1: znak ograničenja brzine 40 km/h	1210	135	136	1481
Klasa 2: znak ograničenja brzine 50 km/h	758	94	94	946
Klasa 3: znak ograničenja brzine 60 km/h	863	100	100	1063
Klasa 4: znak ograničenja brzine 70 km/h	734	91	91	916
Klasa 5: znak ograničenja brzine 80 km/h	609	75	75	759
Klasa 6: znak ograničenja brzine 90 km/h	153	18	18	189
Klasa 7: znak ograničenja brzine 100 km/h	384	48	48	480
Klasa 8: znak ograničenja brzine 110 km/h	41	10	10	61
Klasa 9: znak ograničenja brzine 120 km/h	181	21	20	222
Ukupno	6159	732	732	7623

3.3.2. Učenje neuronske mreže za klasifikaciju prometnih znakova ograničenja brzine

Procesom učenja je potrebno dobiti odgovarajuće težine i *bias*-e odnosno parametre filtara konvolucijskog sloja. Uz definiranu strukturu neuronske mreže, procjena parametara mreže koji su se mijenjali u procesu treniranja provodi se minimizacijom odgovarajuće kriterijske funkcije. Parametri koji su se mijenjali u procesu treniranja kako bi se mreža istrenirala su stopa učenja (engl. *learning rate*), *batch size* i broj epoha. Za praćenje postupka učenja koristio se alat TensorBoard[29]. Učenje neuronske mreže odvija se gradijentnom metodom (engl. *gradient descent*), pri čemu se u svakoj iteraciji učenja trenutne vrijednosti parametara mreže mijenjaju na temelju gradijente kriterijske funkcije po svakom parametru mreže. Pri tome se u svakoj iteraciji odabire mali dio podataka iz skupa za učenje (engl. *batch size*) pa se postupak naziva *mini-batch stochastic gradient descent*. Samo izračunavanje gradijenata kriterijske funkcije s obzirom na parametre mreže radi se pomoću *backpropagation* algoritma. Jedan takav iterativni prolazak kroz cijeli skup za učenje naziva se epoha. Stopa učenja je hiperparametar koji kontrolira koliko treba (kojom „brzinom“) promijeniti model kao odgovor na procijenjenu pogrešku i to svaki put kada se ažuriraju težine modela. Sami odabir stope učenja je izazovan zadatak jer premala vrijednost može rezultirati dugim procesom učenja koji bi se mogao „zaglaviti“ u jednom trenutku, dok prevelika vrijednost može rezultirati prebrzim, a prilično neuspješnim učenjem optimalnog skupa težina ili nestabilnim trening procesom [30]. Parametri koju su postavljeni prilikom treniranja su sljedeći: *batch size* = 15, stopa učenja = 0.001 i broj epoha = 48. Sami proces treniranja CNN praćen je pomoću TensorBoard-a, a rezultati su prikazani na slikama 3.12.i 3.13.

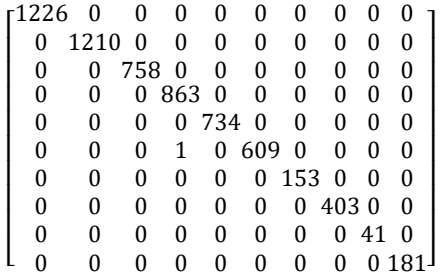


Sl. 3.12. *Promjena preciznosti predviđanja neuronske mreže po koracima treniranja*

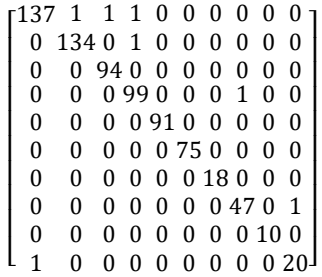


Sl. 3.13. *Promjena gubitka (pogreške) prilikom predviđanja tijekom treniranja neuronske mreže*

Preciznost modela se definira prema formuli 2.3. gdje TP predstavlja ispravno klasificirane prometne znakove, a FP predstavlja neispravno klasificirane prometne znakove (npr. prometni znak ograničenja brzine 30 km/h je klasificiran kao prometni znak ograničenja brzine 60 km/h). Slika 3.12. prikazuje preciznost predviđanja modela. Pri tome su na x osi prikazani koraci treniranja, a na y osi preciznost, odnosno da je 100% znakova u trening skupu ispravno prepoznato (TP). Korak treninga je jedno ažuriranje gradijenta. U jednom koraku obrađuje se broj primjera koji je zadan varijablom *batch size*, dok se epoha sastoji od jednog punog prolaska kroz trening podatke. Slika 3.13 prikazuje promjenu gubitka mreže tijekom procesa učenja, pri čemu su na x osi prikazani koraci treniranja, a na y osi funkcija gubitka. Matrica zabune (engl. *confusion matrix*) prikazana na slici 3.14., pokazuje koliko je znakova prepoznato ispravno u trening skupu, te koliko je znakova zamijenjeno s nekim drugim znakom, i s kojim konkretno. Matrica zabune za trening skup podataka pokazuje izvanredne rezultate, tj. preciznost predviđanja od 100%. Slika 3.15. prikazuje matricu zabune za validacijski skup podataka, a dobivena je nakon završetka treninga i odabira konačnih parametara CNN. Rezultati za validacijski skup podataka su na prvi pogled malo lošiji, no i dalje prihvatljivi. Prema formuli 2.3., preciznost za validacijski skup je 99,04%, što je također vrlo dobar rezultat. S obzirom na ove podatke vjerojatno je za očekivati visoke rezultate i na testnim skupovima, no o tome više u 4. poglavlju.



Sl. 3.14. *Matrica zabune za trening skup podataka*



Sl. 3.15. *Matrica zabune za validacijski skup podataka*

Tablica 3.2. prikazuje funkciju gubitka po epohama. Broj epoha je odabran empirijski, odnosno početni broj epoha je postavljen na 10, a zatim je povećavan kako bi se dobio što manji gubitak. Epoha 48 je pokazala najmanji gubitak te su parametri iz te epohe odabrani kao konačni parametri za CNN. Nakon 48. epohe gubitak se počeo malo povećavati te je zaključeno da će 48 biti dovoljan broj epoha te je gubitak zadovoljavajućeg iznosa.

Tablica 3.2. *Gubitak po epohama u procesu treniranja CNN*

Epoha	Gubitak	Epoha	Gubitak	Epoha	Gubitak
1	93,471	17	1,733	33	0,092
2	31,164	18	0,986	34	0,059
3	19,300	19	0,727	35	0,076
4	13,354	20	1,127	36	3,476
5	9,934	21	0,978	37	0,825
6	7,843	22	0,399	38	0,094
7	5,779	23	0,852	39	0,055
8	4,763	24	0,314	40	0,043
9	4,323	25	0,483	41	0,038
10	3,120	26	0,310	42	0,041
11	2,790	27	1,517	43	0,609
12	2,349	28	0,613	44	1,217
13	1,835	29	0,170	45	0,899
14	1,990	30	0,115	46	0,153
15	1,479	31	0,089	47	0,034
16	1,445	32	0,079	48	0,027

3.4. Spajanje različitih dijelova rješenja u konačno rješenje za prepoznavanje prometnih znakova za ograničenje brzine kretanja vozila

Kada *Object Detection* algoritam detektira prometni znak, potrebno ga je klasificirati pomoću kreirane CNN. Kako bi detektirani prometni znak bio adekvatan za klasifikaciju pomoću neuronske mreže, potrebno je pronađeni prometni znak izdvojiti iz cjelokupne scene, izdvojenom dijelu prikaza promijeniti rezoluciju na 28x28 elemenata slike te dobiti sliku u skali sive boje. Slika 3.16. prikazuje cjelokupnu scenu i detektirani znak na sceni. Prikaz na kojem se izvršava *Object Detection* algoritam je rezolucije 1280x720 elemenata slike, a format ulaznog zapisa je YUV420_SP, što znači da u zapisu slike prvo ide 1280x720 elemenata Y komponente, a zatim redom U i V komponente. Dakle, iz prvih 1280x720 elemenata slike potrebno je izdvojiti traženi prometni znak, čime će se zadovoljiti potrebni uvjet slike u skali sive boje. Za potrebe izdvajanja detektiranog prometnog znaka napisana je funkcija *cutAndGray* koja to omogućava. Funkcija kao ulazne parametre prima pokazivač na trenutnu sliku prikaza i širinu slike ovisno o ulaznom formatu slike (širina zapisa slike nije ista s obzirom na format boja YUYV i sl.), funkcija ne vraća ništa ali rezultat funkcije je izdvojen zadani dio slike. Funkcija se poziva nakon što je prometni znak detektiran i predaju joj se prethodno spomenuti parametri. Funkcija *cutAndGray* ja napisana unutar algoritma *Object Draw*, budući da su tamo dostupne

Sl. 3.16. *Prikaz scene i detektiranog prometnog znaka nakon implementacije Object Detection algoritma na ALPHA ploči*

varijable koje daju informacije o visini i širini ocrtanog graničnog okvira i početnim koordinatama gornjeg lijevog ugla ocrtanog graničnog okvira na cjelokupnom prikazu. Glavna logika funkcije je da prepiše vrijednosti koje se nalaze unutar ocrtanog graničnog okvira na novo mjesto u memoriji, a to je ostvareno pomoću dviju *for* petlji. Prva *for* petlja je podešena da kreće od nule i ide do visine ocrtanog graničnog okvira, a druga for petlja kreće od x koordinate gornjeg lijevog ugla ocrtanog graničnog okvira i ide do x koordinate koja je za iznos širine ocrtanog graničnog okvira veća od x koordinate gornjeg lijevog ugla ocrtanog graničnog okvira, čime je dobiven traženi sadržaj unutar ocrtanog graničnog okvira. Zatim je pozvana funkcija *fileWrite* koja izdvojeni sadržaj sprema na microSD karticu u *bin* formatu. Na slici 3.17. prikazan je kod koji izdvaja prometni znak iz cjelokupne scene.

```
1.  void cutAndGray(UInt8 *inPtr[],UInt32 inPitch[])
2.  {
3.    Vps_printf("Start of cutAndGray");
4.
5.    UInt16 rowIdx;
6.    UInt16 colIdx;
7.
8.    UInt8 *inputPtr;
9.    UInt8 *outputPtr;
10.
11.   outputPtr = (UInt8*)malloc(widthOfBB*heightOfBB);
12.   int i = 0;
13.
14.   inputPtr  = inPtr[0]; //Index 0 - Pointer to Y data in case of YUV420SP
15.   inputPtr += (inPitch[0]) *startY;
16.
17.   for(rowIdx = 0; rowIdx < heightOfBB ; rowIdx++)
18.   {
19.     for(colIdx =startX ; colIdx < startX + widthOfBB ; colIdx++)
20.     {
21.       *(outputPtr+i)=*(inputPtr+colIdx);
22.       i++;
23.     }
24.
25.     inputPtr += inPitch[0];
26.
27.   }
28.
29.   fileWrite(outputPtr, widthOfBB*heightOfBB);
30.
31.   rescaleImageFun(widthOfBB);
32.
33.   Vps_printf("End of cutAndGray");
34. }
```

Sl. 3.17. *Kod cutAndGray funkcije za izdvajanje detektiranog prometnog znaka*

Nadalje, nakon što je pronađeni znak izdvojen, potrebno je promijeniti rezoluciju slike tog izdvojenog znaka na 28x28 elemenata slike. Traženo je ostvareno pomoću funkcije *rescaleImageFun* koja je napisana unutar istog algoritma. Funkcija *rescaleImageFun* kao ulazni parametar prima samo jednu jedinstvenu informaciju o širini i visini ocrtanog graničnog okvira (varijabla *sizeOfInput*), budući da su širina i visina ocrtanog graničnog okvira međusobno jednake. Funkcija nema povratnu vrijednost, a rezultat funkcije se prosljeđuje funkciji koja predstavlja CNN. Slika kojoj je potrebno promijeniti rezoluciju se učitava sa microSD kartice. Ideja promjene rezolucije slike je da se uzima svaki *n*-ti element kako bi se slika bilo koje veličine smanjila na traženu dimenziju 28x28 elemenata slike. Traženo je ostvareno pomoću dvije *while* petlje, jedna za kretanje po širini slike, a druga za kretanje po visini slike. Obje petlje imaju iteratore koji su tipa podatka *float* i u svakoj iteraciji se povećavaju za *sizeOfInput* / 28. Pri prepisivanju vrijednosti originalne slike u vrijednosti nove slike, iteratori se zaokružuju na cijeli broj pomoću operatora *floor*. Na slici 3.18. dan je programski kod *rescale* funkcije.

```
1.  void rescaleImageFun(UInt32 sizeOfInput){
2.
3.    UInt16 i = 0,j = 0, k = 0, x = 0;
4.
5.    UInt8 *imageForRescale;
6.    imageForRescale = (UInt8*)malloc(sizeOfInput*sizeOfInput);
7.    fileRead(imageForRescale, sizeOfInput*sizeOfInput);
8.
9.    UInt8 *resultImage;
10.   resultImage = (UInt8*)malloc(28*28);
11.
12.   float helpVar2;
13.   helpVar2 = (float)sizeOfInput;
14.
15.   float wCnt = 0.0;
16.   float hCnt = 0.0;
17.
18.   while ( hCnt < sizeOfInput && j<28){
19.       j++;
20.       while ( wCnt < sizeOfInput && i<28){
21.           i++;
22.           *(resultImage + x ) = imageForRescale[(int)floor(hCnt)][(int)floor(wCnt)];
23.            x++;
24.            wCnt+=helpVar2/28.0;
25.        }
26.       wCnt=0.0;
27.       i=0;
28.       hCnt+=helpVar2/28.0;
29.    }
30.
31.   fileWriteSecond(resultImage, 28*28);
32.
33.   inference();
34. }
```

Sl. 3.18. *Kod rescale funkcije za promjenu rezolucije slike izdvojenog znaka*

Slika 3.19. prikazuje sliku prometnog znaka izdvojenog iz cjelokupne scene pomoću funkcije *cutAndGray*, dok slika 3.19. prikazuje rezultat *rescaleImageFun* odnosno sliku prometnog znaka smanjene rezolucije na 28x28 elemenata slike.

Sl. 3.20. *Slika prometnog znaka izdvojena funkcijom cutAndGray*

Sl. 3.19. *Slika prometnog znaka promijenjene rezolucije funkcijom rescale*

Nakon što je slika ispravno izdvojena i nakon što je promijenjena rezolucija slike, slika se šalje funkciji *inference* koja implementira neuronsku mrežu. Spomenuta funkcija, također je napisana kao dio *Object Draw* algoritma. Parametri neuronske mreže koji su dobiveni treniranjem na računalu, izvezeni su kao tekstualna datoteka i zapisani kao globalne varijable unutar algoritma, kako bi bili dostupni neuronskoj mreži.

3.5. Način pokretanja kreiranog rješenja na ADAS razvojnoj ploči

Za pokretanje vlastitog rješenja na ADAS razvojnoj ploči potrebno je slijediti sljedeće korake:

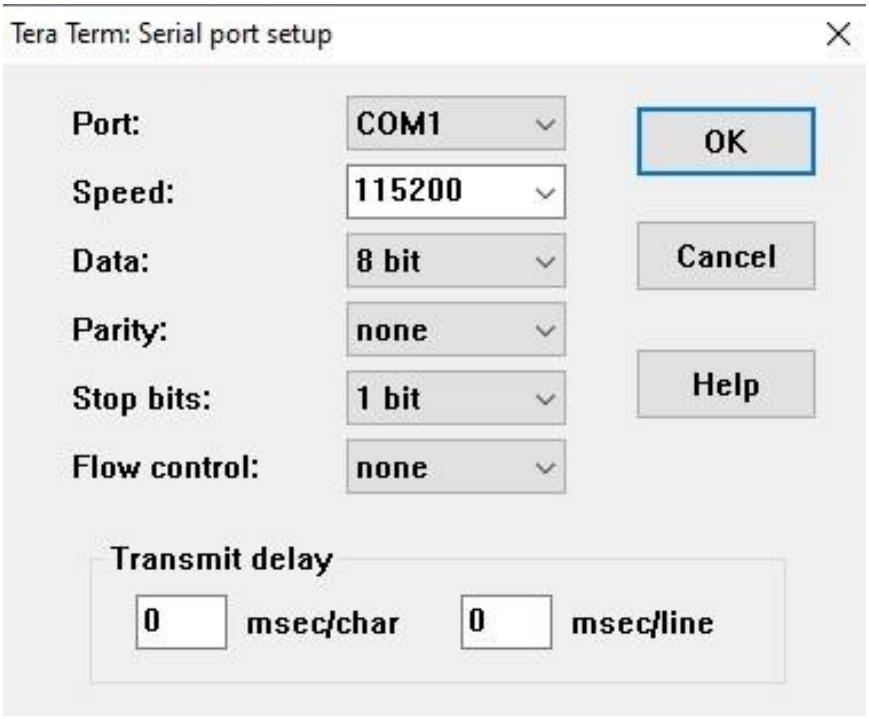
- Pomoću Windows terminala (engl. *Windows Command Prompt*) potrebno je pozicionirati se u mapu s putanjom: C:\VISION_SDK_02_12_01_00\vision_sdk
- Zatim treba izgraditi niže razine VisionSDK pomoću naredbe: *gmake -s -j depend*
- Kako bi se izgradio *bootloader* potrebna je naredba: *gmake -s -j sbl_sd*
- Kako bi se izgradile VisionSDK dodaci i pokazni primjeri, potrebna je naredba: *gmake -s –j*

- I naposljetku, kako bi se izgradila *image* datoteka, potrebna je naredba: *gmake -s appimage*

Programska podrška se na SoC-u pokreće putem microSD kartice. Karticu je potrebno spojiti s računalom i u korijenski direktorij treba prebaciti sljedeće dvije datoteke:

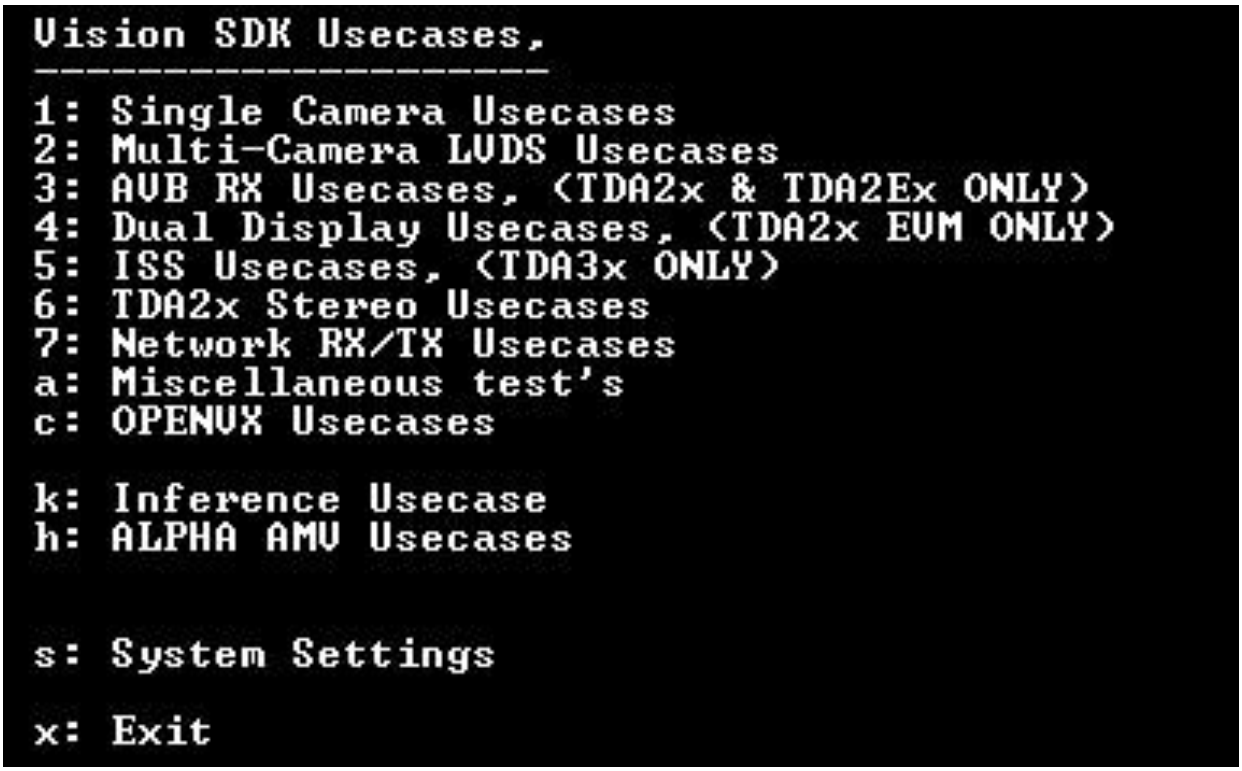
- C:\VISION_SDK_02_12_01_00\vision_sdk\build\scripts\sd_tda2xx-evm\MLO
- C:\VISION_SDK_02_12_01_00\vision_sdk\binaries\tda2xx_evm_bios_all\vision_sdk\bin\tda2xx-evm\sbl_boot\AppImage

te nakon toga karticu umetnuti u odgovarajući utor na ADAS razvojnoj ploči. Korisničkom sučelju aplikacije pristupa se preko serijskog izlaza na ploči i UART-USB pretvarača. Za komunikaciju s ADAS razvojnom pločom se koristi TeraTerm [31] program, a potrebne postavke programa su prikazane na slici 3.21. Naposljetku, microSD karticu treba ubaciti u odgovarajući utor na ADAS razvojnoj ploči i uključiti napajanje ploče.



Sl. 3.21. *Postavke TeraTerm-a*

Ako je sve ispravno napravljeno, na serijskom terminalu (TeraTerm) će se prikazati popis mogućnosti, što je prikazano na slici 3.22. Za pokretanje rješenja za detekciju i klasifikaciju prometnih znakova potrebno je pritisnuti *h*, *6* a zatim *2*.



Sl. 3.22. *Korisničko sučelje - popis mogućnosti nakon pokretanje ADAS razvojne ploče*

4. EVALUACIJA PERFORMANSI PREDLOŽENOG RJEŠENJA ZA PREPOZNAVANJE PROMETNIH ZNAKOVA OGRANIČENJA BRZINE

Ovo poglavlje opisuje način testiranja predloženog rješenja za prepoznavanje prometnih znakova ograničenja brzine. Za ispravno i kvalitetno testiranje, odnosno evaluaciju performansi potrebno je pravilno odabrati skup podataka na kojem će testiranje biti izvršeno. Različiti uvjeti okoline mogu utjecati na uspješnost predloženog rješenja. U sklopu ovog rada korišteni su signali snimljeni za vrijeme sunčanog i oblačnog vremena (bez padalina).

4.1. Opis testnog skupa videozapisa prometnih znakova

U svrhu testiranja prikupljeno je 89 videozapisa (elektronički prilog P.4.1.) koji obuhvaćaju 7 klasa prometnih znakova. Za klase 5, 7 i 9 (znakove ograničenja brzine 80km/h, 100km/h i 120km/h) nije bilo dostupnih znakova za stvaranje videozapisa na lokalnom području na kojem su obavljeni vožnja i snimanje. Svaki videozapis sadrži jedan znak koji je potrebno detektirati i ispravno prepoznati, a tablica 4.1. pokazuje točan broj testnih znakova za svaku klasu. Videozapisi su snimljeni u više navrata na različitim dionicama lokalnih cesta, kako bi testni skup bio raznolik. Videozapisi su snimljeni kamerom mobilnog uređaja uz rezoluciju 1280x720 elemenata slike i brzinu snimanja od 30 okvira u sekundi. Budući da predloženo rješenje radi s formatom boja YUV420_SP, bilo je potrebno videozapise prebaciti u odgovarajući format boja, što je napravljeno na računalu pomoću alata FFmpeg[32].

Tablica 4.1. *Broj testnih znakova iz videozapisa za svaku klasu*

Ime klase	Broj znakova u testnom skupu
Klasa 0: znak ograničenja brzine 30 km/h	9
Klasa 1: znak ograničenja brzine 40 km/h	19
Klasa 2: znak ograničenja brzine 50 km/h	1
Klasa 3: znak ograničenja brzine 60 km/h	10
Klasa 4: znak ograničenja brzine 70 km/h	37
Klasa 5: znak ograničenja brzine 80 km/h	0
Klasa 6: znak ograničenja brzine 90 km/h	9
Klasa 7: znak ograničenja brzine 100 km/h	0
Klasa 8: znak ograničenja brzine 110 km/h	4
Klasa 9: znak ograničenja brzine 120 km/h	0
Ukupno	89

Slika 4.1. prikazuje odabrane okvire iz videozapisa, odnosno primjere scena u kojima je potrebno detektirati i ispravno prepoznati znak.

a)

b)

c)

Sl. 4.1. *Slike sadržane u videozapisima (a) znak ograničenja brzine 70 km/h,(b) znak ograničenja brzine 40 km/h, (c) znak ograničenja brzine 30 km/h*

4.2. Rezultati testiranja i analiza rezultata predloženog rješenja

U ovom potpoglavlju predstavit će se rezultati testiranja na testnom skupu videozapisa pojašnjenom u potpoglavlju 4.1. i na testnom skupu slika koji je pojašnjen u dijelu 3.3.1. Testiranje na skupu podataka videozapisa je izvršeno na ADAS razvojnoj ploči, uz parametre filtara za CNN dobivene treniranjem mreže na računalu, dok je testiranje na skupu slika izvršeno na računalu. Dakle svaki videozapis na kojem je izvršeno testiranje je isporučen s računala na ploču pomoću Ethernet komunikacijske veze, umjesto davanja direktnog ulaza s kamere, te je zatim pokrenut *Object detection* algoritam sa svim dodanim programskim rješenjima. Rezultat klasifikacije je ispisan u TeraTerm terminalu.

Tablica 4.2. prikazuje rezultate testiranja na skupu podataka slika na računalu. S obzirom da se testira klasifikator, a ne detektor, mjera kojom se vrednuje klasifikator je preciznost te je opisana u potpoglavlju 2.3. CNN za svaki izdvojeni znak koji dobije kao izlaz *Object Detectio*n algoritma odredi kojoj klasi pripada te ne postoji 11. klasa koja bi značila da obrađeni znak nije nijedan od traženih znakova. Dakle ne postoje FN i TN rezultati, pa stoga računanje odziva nije relevantno za ovo testiranje.

Tablica 4.2. *Rezultati testiranja skupa podataka slika za svaku klasu i ukupno*

Ime klase	**TP**	**FP**	**Preciznost [%]**
Klasa 0: znak ograničenja brzine 30 km/h	135	5	96,43
Klasa 1: znak ograničenja brzine 40 km/h	135	1	99,26
Klasa 2: znak ograničenja brzine 50 km/h	89	5	94,68
Klasa 3: znak ograničenja brzine 60 km/h	98	2	98
Klasa 4: znak ograničenja brzine 70 km/h	91	0	100
Klasa 5: znak ograničenja brzine 80 km/h	73	2	97,33
Klasa 6: znak ograničenja brzine 90 km/h	17	1	94,44
Klasa 7: znak ograničenja brzine 100 km/h	45	3	93,75
Klasa 8: znak ograničenja brzine 110 km/h	10	0	100
Klasa 9: znak ograničenja brzine 120 km/h	19	1	95
Ukupno	**712**	**20**	**97,28**

Rezultati testiranja skupa podataka slika pokazuju vrlo dobre rezultate. Preciznost sustava je 97,28%, no i dalje ima prostora za napredak te je on potreban s obzirom na namjenu ovog sustava. Nadalje, oni znakovi koji su krivo prepoznati, što se može vidjeti u matrici zabune prikazanoj na slici 4.2. odnosno zamijenjeni s neki drugim znakom, pokazuju određene nelogičnosti. Znak ograničenja brzine 30 je u jednom testnom slučaju prepoznat kao znak ograničenja brzine 60, što i ima smisla, budući da znamenke 3 i 6 imaju neke slične elemente. U nekom drugom testnom slučaju znak 30 prepoznat je kao znak 110 ili kao znak 120, teško je objasniti konkretno zašto je to tako te bi budući rad trebao istražiti ove nelogičnosti. Slični elementi nelogičnosti mogu se uočiti i kod ostalih klasa prometnih znakova ograničenja brzine.

$$\begin{bmatrix} 135 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 135 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 89 & 2 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 1 & 1 & 98 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 73 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 17 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 45 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 19 \end{bmatrix}$$

Sl. 4.2. *Rezultati testiranja skupa podataka slika prikazani pomoću matrice zabune*

Tablica 4.3. prikazuje rezultate testiranja skupa podataka videozapisa na ADAS razvojnoj ploči, pri čemu se klasifikator vrednuje samo prema mjeri preciznosti opisanoj u potpoglavlju 2.3. iz prethodno opisanih razloga.

Tablica 4.3. *Rezultati testiranja skupa podataka videozapisa za svaku klasu i ukupno*

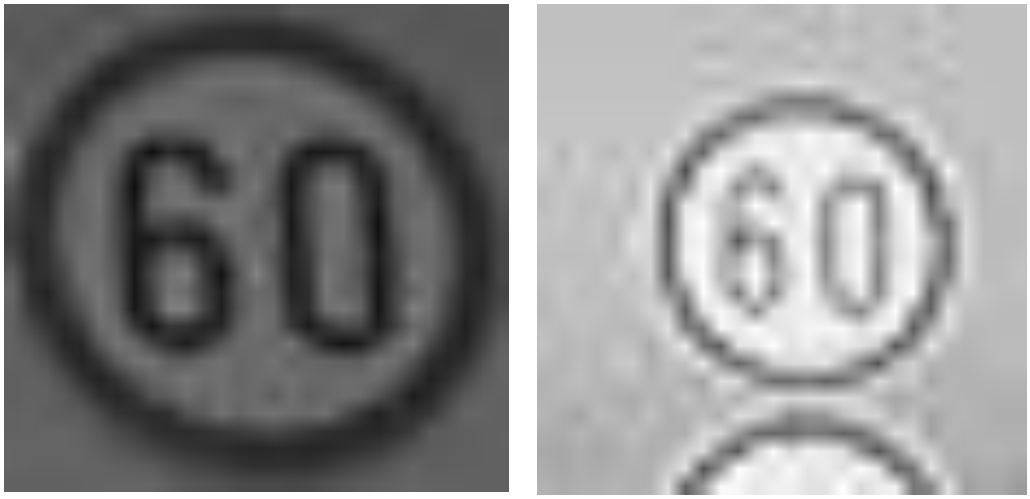
Ime klase	**TP**	**FP**	**Preciznost [%]**
Klasa 0: znak ograničenja brzine 30 km/h	3	6	33,33
Klasa 1: znak ograničenja brzine 40 km/h	7	12	36,84
Klasa 2: znak ograničenja brzine 50 km/h	0	1	0
Klasa 3: znak ograničenja brzine 60 km/h	9	1	90
Klasa 4: znak ograničenja brzine 70 km/h	18	19	48,65
Klasa 5: znak ograničenja brzine 80 km/h	0	0	-
Klasa 6: znak ograničenja brzine 90 km/h	3	6	33,33
Klasa 7: znak ograničenja brzine 100 km/h	0	0	-
Klasa 8: znak ograničenja brzine 110 km/h	0	4	0
Klasa 9: znak ograničenja brzine 120 km/h	0	0	-
Ukupno	**40**	**49**	**44,94**

Rezultati testiranja skupa podataka videozapisa ne prikazuju zadovoljavajući učinak. Preciznost je vrlo niska i iznosi samo 44,94%. Matrica zabune prikazana na slici 4.3. pokazuje rezultate klasifikacije na način da prikaže koliko je prometnih znakova po klasi krivo klasificirano i kako su klasificirani.

$$\begin{bmatrix} 3 & 1 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 7 & 0 & 11 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 5 & 18 & 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 & 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sl. 4.3. *Rezultati testiranja skupa podataka videozapisa prikazani pomoću matrice zabune*

S obzirom da CNN daje razumne rezultate prilikom testiranja na računalu, radi provjere ispravnosti CNN implementirane na ADAS ploči napravljeno je i testiranje izdvojenih slika pomoću ADAS razvojne ploče, ali na računalu. Testiranje izdvojenih slika pomoću ADAS razvojne ploče daje potpuno jednake rezultate klasifikacije kao i testiranje na ADAS razvojnoj ploči, što vodi do zaključka da je CNN dobro implementirana. Razlog zašto su rezultati klasifikacije slika izdvojenih pomoću ADAS razvojne ploče loši bi mogao biti sam proces izdvajanja slike prometnog znaka iz videozapisa te promjena rezolucije slike. Ako se usporede slike 4.4. a) i 4.4. b) mogu se uočiti razlike koje su



a) *b)*

Sl. 4.4. *Usporedba slika: a) slika prometnog znaka iz trening skupa za učenje CNN, b) slika prometnog znaka izdvojena na ADAS razvojnoj ploči*

potencijalno uzrok lošoj preciznosti klasifikacije. Slika prometnog znaka izdvojena na ADAS razvojnoj ploči obuhvaća puno više pozadine nego slike prikupljene za trening, što znači da će se značajke po kojima klasifikator određuje pripadnu klasu razlikovati. Rubovi slika izdvojenih na ADAS razvojnoj ploči su nejasni i nepravilni, odnosno na slikama prikupljenima za trening, znamenke brojeva su jasnije prikazane. Također, slike prikupljene za trening su tamnije nego slike izdvojene na ADAS razvojnoj ploči.

5. ZAKLJUČAK

Kako bi se povećala sigurnost vožnje i općenito sigurnost prometa, automobilska industrija je dizajnirala elektronički sustav (ADAS) u vozilu koji pomaže vozaču u svakodnevnoj vožnji te razvoj takvih sustava sve više napreduje. Ovaj se diplomski rad bavi problematikom detekcije i ispravnog prepoznavanja prometnih znakova, s naglaskom upravo na znakove ograničenja brzine kretanja vozila. Iako postoji mnoštvo rješenja koje pokušavaju riješiti ovaj problem, mali broj njih koji su javno dostupni, je implementiran na realnu ugradbenu platformu za rad u stvarnom vremenu, kao što je ADAS razvojna ploča. Za ovu se namjenu obrađuje signal s kamere postavljene na prednjoj strani vozila. Programsko rješenje za detekciju prometnih znakova razvijeno je u C programskom jeziku za Vision SDK razvojno okruženje i konkretnu ADAS razvojnu ploču. Za detekciju prometnih znakova na danoj sceni korišten je *Object Detection* algoritam koji dolazi s Vision SDK razvojnim okruženjem, dok je za ispravno prepoznavanje prometnih znakova ograničenja brzine kretanja vozila korištena CNN. CNN je za potrebe treniranja napisana u programskom jeziku Python i istrenirana na računalu, te sadrži samo jedan konvolucijski sloj s 8 filtara veličine 5x5 elemenata slike i *softmax* sloj. Ove karakteristike CNN su odabrane zbog nedostatne procesorske moći zadanog sklopovlja. U sklopu ovog diplomskog rada prikupljen je skup podataka za treniranje, validaciju i testiranje CNN, te sadrži 7623 slike. Rezultati testiranja pokazali su određeni potencijal stvorenog rješenja, ali performanse svakako trebaju biti unaprijeđene. Preciznost CNN pokrenute na ADAS razvojnoj ploči iznosi svega 44,94%. S obzirom da su rezultati testiranja CNN izvršeni na računalu vrlo visoki (preciznost od 97,28%) te da testiranje na računalu slika dobivenih iz videozapisa daje jednake rezultate kao ono na ADAS razvojnoj ploči, da se zaključiti da CNN radi ispravno i pretpostaviti da postoji neki drugi razlog niske preciznosti testiranja na ADAS razvojnoj ploči. Pretpostavlja se da je problem nastao prilikom izdvajanja slike iz videozapisa i promjene rezolucije slike te bi daljnji rad svakako trebao ustanoviti da li je dana pretpostavka ispravna i zašto je problem nastao.

LITERATURA

[1] S. De Keersmaecker, "2019 road safety statistics: what is behind the figures?," *European Commission - European Commission*. https://ec.europa.eu/commission/presscorner/detail/en/qanda_20_1004 (accessed Aug. 28, 2020).

[2] S. Rimac Drlje and M. Vranješ, "Digitalna obrada slike i videa za autonomna vozila." [Online]. Available: https://loomen.carnet.hr/pluginfile.php/1569739/mod_resource/content/1/1.%20Uvos.pdf.

[3] U. Z. Abdul Hamid *et al.*, "Autonomous emergency braking system with potential field risk assessment for frontal collision mitigation," prosinac 2017, doi: 10.1109/SPC.2017.8313024.

[4] I. Mihajlovic, "Everything You Ever Wanted To Know About Computer Vision. Here's A Look Why It's So Awesome.," *Medium*, Feb. 09, 2020. https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e (accessed Sep. 04, 2020).

[5] F. Dimaano, "What is Machine Learning?," *Medium*, Oct. 24, 2019. https://towardsdatascience.com/what-is-machine-learning-885aa35db58b (accessed Sep. 04, 2020).

[6] "Pravilnik o prometnim znakovima, signalizaciji i opremi na cestama." https://narodne-novine.nn.hr/clanci/sluzbeni/2019_09_92_1823.html (accessed Sep. 07, 2020).

[7] J. Frankenfield, "How Artificial Intelligence Works," *Investopedia*. https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp (accessed Aug. 28, 2020).

[8] Simplilearn, *Deep Learning In 5 Minutes | What Is Deep Learning? | Deep Learning Explained*. 2019.

[9] Simplilearn, *Neural Network In 5 Minutes | What Is A Neural Network? | How Neural Networks Work*. 2019.

[10] V. H. Phung, E. J. Rhee, "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets," *ResearchGate*. https://www.researchgate.net/publication/336805909_A_High-Accuracy_Model_Average_Ensemble_of_Convolutional_Neural_Networks_for_Classification_of_Cloud_Image_Patches_on_Small_Datasets (accessed Sep. 07, 2020).

[11] M. Vranješ, "Strojno učenje u sustavima autonomnih i umreženih vozila: Konvolucijske neuronske mreže." FERIT.

[12] B. D. Bašić, J. Šnajder, "Vrednovanje klasifikatora," p. 35.

[13] A. Akula, R. Devi, "Automatic Speed-Limit Sign Detection and Recognition for Advanced Driver Assistance Systems," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 9S2, srpanj 2019, doi: 10.35940/ijitee.I1001.0789S219.

[14] R. Biswas, H. Fleyeh, M. Mostakim, "Detection and classification of speed limit traffic signs," listopad 2014, doi: 10.1109/WCCAIS.2014.6916605.

[15] W. S. Cleveland, S. J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *J. Am. Stat. Assoc.*, vol. 83, no. 403, Jan. 1986, doi: 10.1080/01621459.1988.10478639.

[16] S. Miyata, "Recognition of Speed Limits on Speed-Limit Signs by Using Machine Learning," *J. Imaging*, vol. 3, srpanj 2017, doi: 10.3390/jimaging3030025.

[17] A. W. Mir, H. Ahmed, A. A. Shah, "Automated speed limit identification for efficient driving system," in *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*, Mar. 2017, pp. 299–303, doi: 10.1109/C-CODE.2017.7918946.

[18] H. Ngoc Do, M.-T. Vo, H. Quoc Luong, A. Hoang Nguyen, K. Trang, L. T. K. Vu, “Speed limit traffic sign detection and recognition based on support vector machines,” in *2017 International Conference on Advanced Technologies for Communications (ATC)*, Oct. 2017, pp. 274–278, doi: 10.1109/ATC.2017.8167633.
[19] Y. Yang, H. Luo, H. Xu, F. Wu, “Towards Real-Time Traffic Sign Detection and Classification,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 2022–2031, Jul. 2016, doi: 10.1109/TITS.2015.2482461.
[20] S. Rimas Drlje, M. Vranješ, “Upoznavanje s ADAS razvojnom pločom.” [Online]. Available: https://loomen.carnet.hr/pluginfile.php/1583196/mod_resource/content/1/Vjezba_1.pdf.
[21] “RT-RK - Automotive.” https://www.rt-rk.com/services/automotive (accessed Sep. 02, 2020).
[22] A. Rosebrock, “Image Pyramids with Python and OpenCV,” *PyImageSearch*, Mar. 16, 2015. https://www.pyimagesearch.com/2015/03/16/image-pyramids-with-python-and-opencv/ (accessed Sep. 09, 2020).
[23] “TensorFlow.” https://www.tensorflow.org/ (accessed Sep. 09, 2020).
[24] R. Grbić, M. Vranješ, D. Vajak, “Strojno učenje u sustavima autonomnih i umreženih vozila, Laboratorijske vježbe: Uvod u TensorFlow.” .
[25] Cartucho, *Cartucho/OpenLabeling*. 2020.
[26] G. C. da Silva, *giovannicimolin/PascalVOC-to-Images*. 2020.
[27] “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.” http://yann.lecun.com/exdb/mnist/ (accessed Sep. 08, 2020).
[28] G. S. Kielian, *gskielian/JPG-PNG-to-MNIST-NN-Format*. 2020.
[29] “TensorBoard,” *TensorFlow*. https://www.tensorflow.org/tensorboard (accessed Sep. 09, 2020).
[30] J. Brownlee, “Understand the Impact of Learning Rate on Neural Network Performance,” *Machine Learning Mastery*, Jan. 24, 2019. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/ (accessed Aug. 28, 2020).
[31] “Tera Term - Terminal Emulator for Windows.” https://ttssh2.osdn.jp/ (accessed Sep. 10, 2020).
[32] “FFmpeg.” https://ffmpeg.org/ (accessed Sep. 13, 2020).

SAŽETAK

Zadatak ovog diplomskog rada bio je razviti rješenje za prepoznavanje prometnih znakova ograničenja brzine. Kompletno rješenje bilo je potrebno razviti u programskom jeziku C, izvršiti njegovu optimizaciju i implementirati ga na zadanu ADAS (engl. *Advanced driver-assistance systems*) razvojnu ploču. Kao uvod u rad, dan je pregled postojećih rješenja navedenog problema te su pojašnjene metode i načini koji realiziraju slične zadatke. Detekcija prometnih znakova ograničenja brzine ostvarena je pomoću algoritma koji dolazi uz Vision SDK razvojno okruženje, dok je ispravno prepoznavanje prometnih znakova ostvareno pomoću konvolucijske neuronske mreže (engl. *Convolutional Neural Network,* CNN ili ConvNet). S obzirom da je CNN bilo potrebno implementirati na zadanu ADAS razvojnu ploču, prilikom kreiranja mreže bilo je potrebno obratiti pozornost na procesorske mogućnosti zadanog sklopovlja. Stoga, umjesto velikog broja konvolucijskih slojeva, željena CNN sadrži samo jedan konvolucijski sloj s 8 filtara veličine 5x5 elemenata slike i *softmax* sloj. CNN je kreirana u dva programska jezika, programskom jeziku Python za potrebe treniranja CNN na računalu i u programskom jeziku C za implementaciju CNN na ADAS razvojnu ploču. Rezultati testiranja pokazuju određenu preciznost, međutim ona nije zadovoljavajuća na svim korištenim skupovima testnih podataka. Pretpostavlja se da je rezultat klasifikacije prilično nizak zbog problema nastalog prilikom izdvajanja slike iz videozapisa i promjene rezolucije slike te bi daljnji rad svakako trebao ustanoviti da li je dana pretpostavka ispravna i zašto je problem nastao.

Ključne riječi: ADAS, duboko učenje, klasifikacija prometnih znakova, konvolucijska neuronska mreža

RECOGNITION OF MAXIMAL SPEED LIMIT TRAFFIC SIGNS FOR USE IN ADVANCED ADAS ALGORITHMS

ABSTRACT

The main goal of this thesis was to develope a solution for for recognizing speed limit traffic signs. The complete solution had to be developed in the C programming language, optimized and implemented on the default ADAS (Advanced driver-assist systems) development board. As an introduction to the paper, an overview of the existing solutions to the mentioned problems is given, which are explained methods and ways of realizing similar tasks. Speed limit traffic sign detection is achieved using an algorithm that comes with the Vision SDK development environment, while correct traffic sign recognition is achieved using Convolutional Neural Network (CNN or ConvNet). Since CNN needed to be implemented on a given ADAS development board, when creating the network it was necessary to pay attention to the processing capabilities of the given assembly. Therefore, instead of a large number of convolutional layers, the desired CNN contains only one convolutional layer with 8 filters size of 5x5 pixels and a softmax layer. CNN was created in two programming languages, the Python programming language for CNN training on a computer and the C programming language for implementing CNN on the ADAS development board. The test results show some accuracy, however, it is not satisfactory on all test datasets used. It is assumed that the result of the classifications is quite low due to the problem that arose when extracting images from the video and changing image resolutions. Further work should certainly determine whether the given assumption is corrected and why the problem occurred.

Keywords: ADAS, deep learning, traffic sign classification, convolutional neural network

ŽIVOTOPIS

Barbara Strišković je rođena 31. prosinca 1996. godine u Osijeku. Odrasla je u Donjem Miholjcu gdje je pohađala Osnovnu školu Augusta Harambašića. Nakon završetka osnovne škole upisuje Srednju školu Donji Miholjac, smjer Opća gimnazija. U Osijeku, 2015. godine upisuje preddiplomski studij elektrotehnike na tadašnjem Elektrotehničkom Fakultetu, te na drugoj godini studija odabire smjer Komunikacije i informatika. 2018. godine stječe akademski naziv sveučilišna prvostupnica (lat. baccalaureus) inženjerka elektrotehnike na temu završnog rada „Deskriptor ključnih točaka inspiriran ljudskim vizualnim sustavom“. Iste godine upisuje diplomski sveučilišni studij elektrotehnike, smjer Komunikacije i informatika, izborni blok Mrežne tehnologije.

PRILOZI

P.3.1. Datoteka cnn.py (elektronički prilog)

P.3.2. Baza slika korištena za treniranje CNN (elektronički prilog)

P.3.3. Baza slika korištena za validaciju CNN (elektronički prilog)

P.3.4. Baza slika korištena za testiranje CNN (elektronički prilog)

P.4.1. Baza videozapisa korištena za testiranje CNN (elektronički prilog)