Detekcija prometnih znakova ograničenja brzine iz slike kamere na vozilu

Živković, Anastazija

Undergraduate thesis / Završni 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:200:114930

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-06**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

DETEKCIJA PROMETNIH ZNAKOVA OGRANIČENJA BRZINE IZ SLIKE KAMERE NA VOZILU

Završni rad

Anastazija Živković

Osijek, 2020.

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

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 07.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Anastazija Živković
Studij, smjer:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4441, 24.09.2019.
OIB studenta:	95786924263
Mentor:	Prof.dr.sc. Snježana Rimac-Drlje
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Detekcija prometnih znakova ograničenja brzine iz slike kamere na vozilu
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	07.09.2020.
Datum potvrde ocjene Odbora:	11.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2020.

Ime i prezime studenta:	Anastazija Živković
Studij:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4441, 24.09.2019.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom: **Detekcija prometnih znakova ograničenja brzine iz slike kamere na vozilu**

izrađen pod vodstvom mentora Prof.dr.sc. Snježana Rimac-Drlje

i sumentora

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

Potpis studenta:

SADRŽAJ

1. UVOD

Znakovi ograničenja brzine su postavljeni kako bi davali vozačima informaciju koja je maksimalna dozvoljena brzina vožnje. Upravo iz tog razloga ovi prometni znakovi nalaze se posvuda i veoma su česti jer omogućavaju održavanje sigurnosti u prometu. Zato je jasno kako prepoznavanje znakova ograničenja brzine predstavlja veoma važnu komponentu u razvoju autonomnih vozila, osobito onih sustava koji će iz slike kamere na vozilu upozoriti vozača o ograničenju brzine u različitim okolnim uvjetima. Zbog tako bitne uloge, jako je važno osigurati da sprovedene metode koje se koriste u svrhu rješavanja ovog problema imaju visoku točnost, unatoč nesavršenim uvjetima.

Iako su znakovi ograničenja brzine definirani bojom i oblikom, ipak predstavljaju popriličan izazov prilikom detekcije i prepoznavanja prometnog znaka. Dva su načina detekcije navedenih objekata: pomoću oblika i pomoću boje. Dakako da postoje i algoritmi koji koriste obje značajke, ali takav način bitno usporava rad samog algoritma. Budući da je boja važan atribut svakog znaka ograničenja brzine, jasno je da izostanak boje prilikom analize prometnog znaka može predstavljati veliki problem. Nedostatak informacije o boji javlja se na fotografijama onda kad je prometni znak snimljen za vrijeme loših vremenskih uvjeta kao što su magla, kiša, snijeg i sl. Veliku razliku čini i činjenica u koje doba dana je fotografija snimljena, budući da za fotografije u boji ista boja može izgledati drugačije u različitim uvjetima osvjetljenja, osobito ako je u pitanju sjena ili refleksija, dok se pak za fotografije snimljene po noći ne može dobiti nikakva korisna informacija o boji [1]. Ako boja jest vidljiva, često s vremenom postaje blijeda i neprepoznatljiva pod utjecajem izloženosti sunčevom svjetlu te reakciji boje sa zrakom [2]. Također, za potrebe ovog rada treba naglasiti i kako se boja znakova ograničenja brzine često razlikuje po zemljama pa je to još jedan problem jer bi izrađeni sustav ipak trebao moći raditi posvuda bez obzira u kojoj se zemlji vozilo nalazi.

Dva glavna koraka prilikom prepoznavanja nekog prometnog znaka su segmentacija i prepoznavanje. Budući da se svaki znak ograničenja brzine sastoji od kružnog vijenca ispunjenog bojom te bijele pozadine sa crnim znamenkama, koje se u većini slučajeva nalaze veoma blizu središta objekta, tako teče i proces pronalaženja objekata koji predstavljaju kandidate za traženi prometni znak, a kako je potrebno prepoznati znak ograničenja brzine iz kamere na vozilu, u stvarnosti taj znak nije savršena kružnica nego ima oblik elipse. Upravo zbog takvog karakterističnog oblika je moguće prepoznati traženi objekt čak i onda kada informacija o boji nije

dostupna, odnosno kada ista nije iskoristiva. Često se zato problemu detekcije i prepoznavanja znaka ograničenja brzine pristupa tako što se kao glavna karakteristika nekog prometnog znaka koristi njegov oblik. Za prepoznavanje znaka postoji mnogo metoda, od kojih valja spomenuti metodu potpornih vektora, konvolucijske mreže (engl. *ConNets*), metodu K-najbližih susjeda i neuronske mreže.

Temelj detekcije i prepoznavanja znaka ograničenja brzine u ovom radu biti će prepoznavanje oblika samog znaka te zatim izdvajanje značajki unutar tog oblika, tj. kružnice. Te informacije će biti proslijeđene algoritmu koji će na osnovu tih značajki izvršiti prepoznavanje brojeva koji se nalaze unutar samog znaka.

Ovaj je rad strukturno podijeljen na šest dijelova. U prvom poglavlju predstavljena je tema rada i opisan zadatak rada te su najavljene metode koje će se kasnije pojasniti. Zatim je, u drugom dijelu, dan teorijski pregled nekoliko metoda navedenih u dosadašnjoj znanstvenoj literaturi, većinom baziranih na detekciju samo na temelju oblika, budući da je takav način puno brži od detekcije na temelju boje. U trećem je dijelu pojašnjeno par najkorištenijih algoritama koji služe za detekciju i prepoznavanje znaka ograničenja brzine. Budući da se detekcija može vršiti na osnovu oblika i na osnovu boje prometnog znaka, u četvrtom dijelu rada ukratko je predstavljeno kako vršiti detekciju kada se želi iskoristiti samo informacija o boji. Na kraju je, u petom dijelu, predstavljeno programsko rješenje izrađeno za rješavanje problematike detekcije i prepoznavanja prometnog znaka ograničenja brzine, kao i analizirani rezultati tog rješenja.

U programskom rješenju ovog zadatka naglasak će se staviti na strojno učenje iako su u današnjici u modernim vozilima ovakvi sustavi većinom izvedeni dubokim učenjem (eng. *deep learning*), odnosno neuronskim mrežama. Strojno učenje predstavljeno u ovom radu zasniva se na korištenju odgovarajućih algoritama i funkcija biblioteka implementiranih u biblioteke kao što je primjerice OpenCV za Python programski jezik, koji je korišten za potrebe rada. Python je često korišten za izradu ovakvih sustava, osobito jer sadrži brojne ugrađene biblioteke s funkcijama korisnim za izradu ovakvih i sličnih rješenja.

1.1. Zadatak završnog rada

Razvoj algoritama za obradu slike omogućio je primjenu kamera u automobilima za prepoznavanje objekata tijekom vožnje. Jedna od čestih funkcija koja je dostupna u novijim vozilima je prepoznavanje znaka ograničenja brzine. U završnom radu je potrebno napraviti pregled metoda za detekciju prometnih znakova te detaljno opisati jednu izabranu metodu pogodnu za detekciju znakova ograničenja brzine. U praktičnom dijelu rada treba napraviti program za detekciju znakova ograničenja brzine i analizirati njegovu učinkovitost.

2. PREGLED METODA ZA DETEKCIJU ZNAKOVA OGRANIČENJA BRZINE

U znanstvenoj literaturi objavljen je veliki broj metoda za prepoznavanje prometnih znakova, a među njima su i metode za prepoznavanje znaka ograničenja brzine. U [1] autori su predstavili metodu za prepoznavanje znaka ograničenja brzine na temelju oblika koja prepoznavanje provodi na monokromatskoj slici, a funkcionira u stvarnom vremenu. Ona koristi AdaBoost algoritam i Houghovu transformaciju za kružnice za detekciju te metodu potpornih vektora (engl. *Support Vector Machine*, SVM) za identifikaciju znaka. Kako se svaki znak ograničenja brzine sastoji od obojanog kružnog vijenca i bijele pozadine sa crnim znamenkama, ako se pronađe bijela pozadina, moguće je prepoznati i znamenke koje se nalaze na toj površini. Zato se koristi još i OTSU algoritam kojim se automatski pronalazi binarni prag za neko elipsasto područje [1]. Ovaj algoritam je otporan na promjene osvjetljenja i osigurava veliku točnost [3]. Nakon toga primjenjuje se Canny detektor za detekciju rubova te se postavlja bijela elipsasta maska kako bi se ujedinile sve komponente objekta i predstavljale jednu cjelinu, tj. kako bi se odredila kontura samog objekta. Budući da je potrebno prepoznati znak ograničenja brzine iz kamere na vozilu, u stvarnosti taj znak nije savršena kružnica nego predstavlja elipsu te se zato još radi i korak u kojem se vrši aproksimacija pronađenih objekata s elipsom. Najtočnijom se aproksimacijom smatra ona koja je izvedena iz konture najveće komponente prethodno pronađene Canny detektorom, a ona služi kako bi se pronašli koeficijenti elipse jer se iz njih može izvući informacija o veličini elipse te informacija o kutu. Kada je traženi objekt detektiran, treba ga identificirati, odnosno pronaći i prepoznati sadržaj koji se nalazi unutar samog objekta kako bi bilo jasno o kojem točno znaku ograničenja brzine je riječ. Zato se koristi filter za izoštravanje (engl. *Sharpening filter*) kako bi se smanjio utjecaj osvjetljenja te povećala točnost segmentiranog numeričkog dijela slike. Nakon toga se koristi metoda potpornih vektora koja omogućava prepoznavanje znamenki na traženom prometnom znaku. Iako ova metoda koristi brojne algoritme i filtre, ipak bi još trebalo unaprijediti filtre koji bi smanjili mogućnost lažne detekcije, tj. detekcije koja će proglasiti objekte prometnim znacima iako oni to nisu.

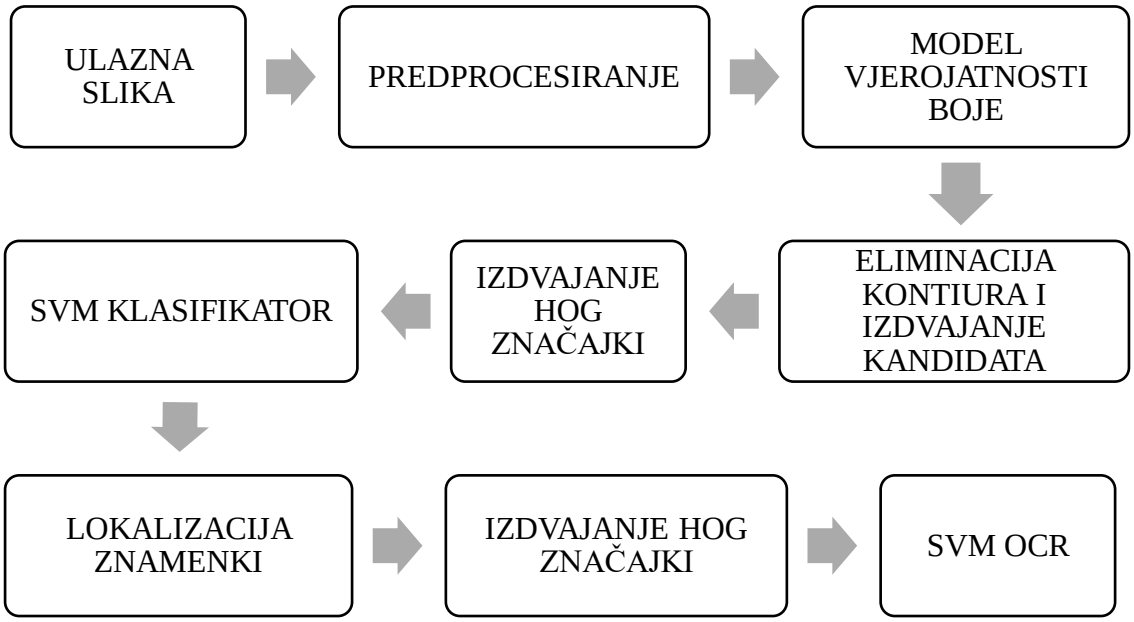
Još jedna metoda koja se preporučuje za korištenje onda kada informacija o boji nije dostupna predložena je u [6] , a koristi klizeći prozor koji detektira prometni znak u slici razloženoj na različite rezolucije (engl. *multiscale image*) te HOG (engl. *Histogram of Oriented Gradients*) uz SVM metodu, gdje se koriste još i grubi i fini detektori (engl. *coarse and fine detectors*) [4]. Za ovu metodu koriste se dva seta fotografija od kojih se jedan sastoji od novih prometnih znakova, a drugi od starih

prometnih znakova. Nakon toga se korištene fotografije pretvaraju u HSV (engl. *Hue Saturation Value*) sustav boja. Odgovarajuće vrijednosti HSV-a opisuju se polarnim koordinatama gdje je nijansa (engl. *hue*) predstavljena kutom koji započinje iz pozitivnog smjera x-osi i raste u smjeru kazaljke na satu, dok je zasićenje (engl. *saturation*) predstavljeno udaljenošću od početka polarnih koordinata do pozicije boje. Ovim načinom jasno je pokazano kako se novi prometni znaci nalaze u području s bojom, dok se set fotografija sa starim prometnim znakovima nalazi u području žute boje ili je, još gore, potpuno izgubio informaciju o boji. Sljedeći korak je višerezolucijska slika gdje se klizećim prozorom tijekom procesa sličnog konvoluciji pronalazi traženi prometni znak koji se može nalaziti na bilo kojoj rezoluciji analizirane slike. Za svaku poziciju klizećeg prozora koristi se HOG koji se nakon toga prosljeđuje SVM-u koji je prethodno treniran setom sličnih uzoraka [4]. HOG je tehnika koju su predstavili Dalal i Triggs u [5], a koja koristi rubove objekta kako bi se kreirao set značajki kojima bi se opisao traženi objekt [4]. Te značajke prosljeđuju se SVM klasifikatorima koji su podijeljeni u grube i fine detektore kojima se jasno izdvaja područje traženog objekta.

Prilikom prepoznavanja znamenki unutar znakova ograničenja brzine, razlikuju se dva pristupa: holistički pristup i pristup temeljen na znamenkama. Prepoznavanje znaka u prvoj metodi se vrši pomoću metode potpornih vektora i sličnih metoda kao što su, primjerice, KNN (engl. *k-nearest neighbor*), odnosno metoda K-najbližih susjeda ili neuronska mreža. Za razliku od holističkog pristupa, u pristupu temeljenom na znamenkama su izdvojene znamenke na prometnom znaku koje se potom koriste za klasifikaciju i prepoznavanje [6].

Općenito, drugi pristup će osigurati veću točnost u odnosu na prvi, holistički, pristup, ali koristi složeniji algoritam. Moguće ga je koristiti zajedno s metodom potpornih vektora [7] gdje se koristi monokromatska slika te crna maska za segmentiranje znamenki i AND - NOT operacija kako bi se izdvojile znamenke. Još jedna mogućnost koju ovakav pristup pruža je korištenje HOG - SVM tehnike [8] kako bi se unaprijedio proces prepoznavanja objekta, a ovo je relativno dobro rješenje budući da je HOG metoda veoma otporna na uvjete u kojima se objekt nalazi. Najprije se uklanjanju svi krivi kandidati, a na preostalim pravim kandidatima se izdvajaju potrebne informacije. Cilj je napraviti sustav koji bi mogao savladati izazove poput prepoznavanja prometnog znaka u okolini punoj gradske pozadine i vozila, odnosno sustav koji bi bio otporan na razne nepovoljne uvjete. Predprocesiranje ulazne slike radi se kako bi se naglasila njena kvaliteta, a nakon toga se ista konvertira u monokromatsku sliku. Ovaj korak se prema [9] radi uz naglašavanje kontrasta, korištenje medijan filtra, segmentacije znamenki te normalizacije izdvojenog znaka koja se vrši uz metodu K-najbližih susjeda pomoću koje se traži sličnost između uzoraka, odnosno traži se uzorak

između dvije klase koji se potom klasificira. Zatim, tijekom eliminiranja kontura odbacuju se sve one konture koje su prevelike ili premale nakon čega se za sve preostale kandidate računa HOG čije se značajke prosljeđuju SVM klasifikatoru koji odlučuje predstavlja li neka površina traženi objekt. Svaki od ovih koraka predložene HOG – SVM metode predstavljen je na slici 2.1. koja se nalazi u nastavku.



Slika 2.1. Model sustava tijekom korištenja HOG i SVM metode

3. PREGLED ALGORITAMA KORISNIH ZA DETEKCIJU I PREPOZNAVANJE ZNAKA OGRANIČENJA BRZINE

3.1. AdaBoost algoritam

AdaBoost (eng. *adaptive boosting*) algoritam strojnog učenja su predložili Viola i Jones u [10], a inače je to jedan od najčešće korištenih algoritama za razvoj detektora za detekciju objekta uz korištenje monokromatske slike. Bitno je naglasiti kako velik utjecaj na uspjeh ovog algoritma imaju trening slike. Prema [1], detektor konstruiran AdaBoost algoritmom otporan je na promjene osvjetljenja i šum ako korišteni set trening slika sadrži uzorke slikane u različitim uvjetima. Međutim, ipak se mogu javiti poneke greške kada detektor pogrešno prepozna neki drugi objekt kao traženi znak. Zato se, pored ovog algoritma, preporučuje napraviti još jedan korak u kojem se koristi Houghova transformacija za kružnice. Na taj način povećava se učinkovitost i točnost detekcije prometnog znaka. AdaBoost algoritam radi na način da odabire samo korisne značajke za traženi objekt ignorirajući sve ostale nepotrebne značajke. Sa svakom od tih značajki povezan je jedan od slabih klasifikatora. Klasifikator je slab ako pruža ispravan izlaz za više od 50% uzoraka [11]. Izlazni rezultat može biti 0 ili 1, odnosno binaran je. Svi ti slabi klasifikatori linearnom kombinacijom grade jedan jaki klasifikator. Ova linearna kombinacija prema [11] glasi:

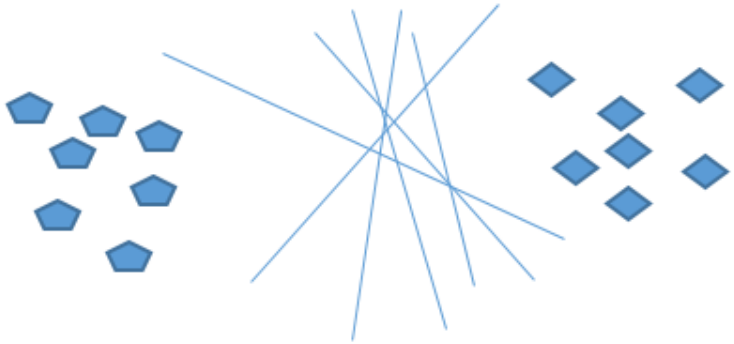
$$H(x) = \sum_{n=1}^{N} a_{\mathrm{n}}\, h_{\mathrm{n}}\,(x) \tag{3-1}$$

Povećanjem primjera za učenje, postotak pogreške jakog klasifikatora se smanjuje jer svaki klasifikator nastoji bolje klasificirati pogrešno klasificirane uzorke prethodnog klasifikatora. Rezultat toga je velika točnost, a to i je temeljna ideja AdaBoost algoritma.

3.2. Metoda potpornih vektora

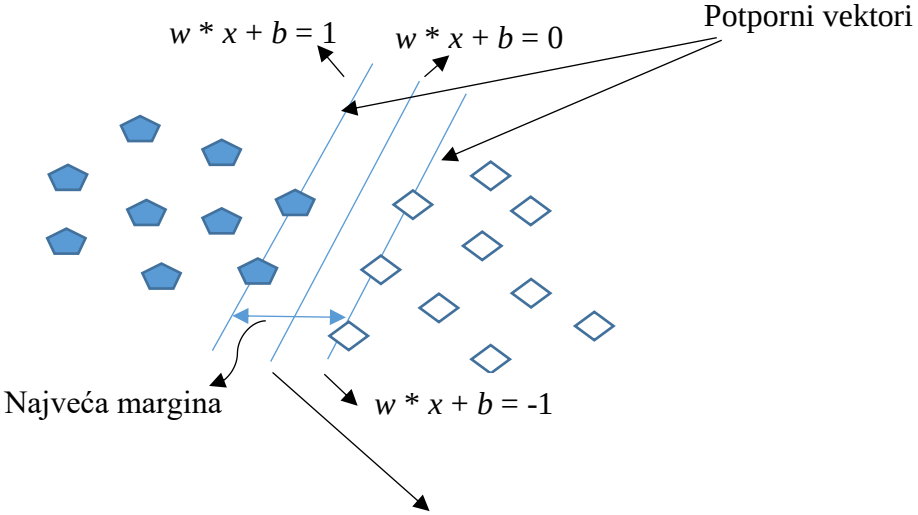
Dva glavna koraka prilikom prepoznavanja nekog prometnog znaka jesu segmentacija i prepoznavanje. Upravo u ovom drugom koraku često se koristi SVM, odnosno metoda potpornih vektora. Metodu potpornih vektora, koja se još naziva i metoda kernel (jezgrenih) funkcija, predložili su Vapnik i suradnici 1992. godine. Kako je navedeno u [4], točnost ove metode je iznad 98,6%, a njen temelj je korištenje trening slika na kojima se nalaze traženi objekti u kadrovima različitih vremenskih uvjeta, različitog osvjetljenja i sl. na način da se izdvajaju značajke boja i oblik traženog prometnog znaka. Ova metoda spada u nadgledane metode klasifikacije u koje spadaju i neuronske mreže te Bayesov klasifikator. Stoga, prema [12] ova metoda se temelji na skupu uzoraka koji je već

ranije poznat, tj. taj skup predstavlja trening slike. SVM pruža izvrsne rezultate onda kada se treba prepoznati neki objekt koji ima više značajki, odnosno kada treba upravljati velikim brojem podataka. Cilj ove metode je pronaći hiperravninu koja razdvaja dvije klase, odnosno svi podaci jedne klase moraju biti s iste strane ravnine. Ovakva klasifikacija uzoraka provodi se u prostoru značajki (eng. *feature space*) budući da takvo razdvajanje uzoraka nije moguće u ulaznom prostoru (eng. *input space*). Prema Slici 3.1. vidljivo je kako postoji beskonačno mnogo takvih hiperravnina koje razdvajaju dvije klase.



Slika 3.1. Hiperravnine između dvije klase podataka

Od svih tih hiperravnina, postavlja se pitanje po kojem načelu odabrati samo jednu. Prema [13] pronađena hiperravnina mora imati najveću marginu razdvajanja klasa. Na slici 3.2. vidi se kako je margina udaljenost između „kritičnih“ točaka najbliže plohi razdvajanja (eng. *decision boundary*). Prema tome, metoda potpornih vektora maksimizira marginu, odnosno navedenu udaljenost, oko plohe razdvajanja. Pomoću potpornih vektora (eng. s*upport vectors*) definira se funkcija odluke.



Normala na hiperravninu razdvajanja

Slika 3.2. Prikaz hiperravnine s potpornim vektorima i maksimalnom marginom

Hiperravnina razdvajanja prikazana na slici 3.2 naziva se granica odluke (eng. *decision boundary*) i označava jednadžbom navedenom u [12]:

$$w * x + b = 0 \tag{3-2}$$

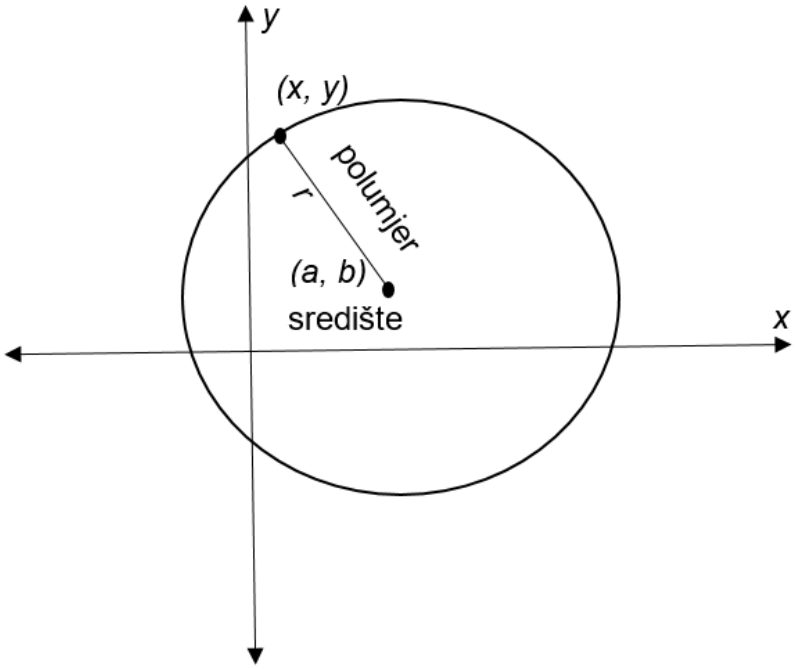
gdje je *w* vektor normale na ravninu razdvajanja, *x* ulazni vektor i *b* pomak.

3.3. Houghova transformacija

Houghova transformacija robusna je tehnika detekcije oblika. Predložio ju je Paul Hough 1962. godine u [14]. Koristi se kako bi se povećala točnost detekcije prometnog znaka, odnosno kako bi se smanjio broj prepoznatih objekata koji zapravo nisu znak ograničenja brzine. Ova metoda je prema [15] složena za izvedbu i nije primjenjiva u stvarnom vremenu, ali uz kombinaciju s HOG metodom daje veoma visok postotak točnosti [9]. Prema [16] Houghova kružna transformacija može se još opisati i kao transformacija središnje točke kruga iz *x-y* ravnine u parametarski prostor pomoću jednadžbe kruga, opisane u [17], koja glasi:

$$(x - a)^2 + (y - b)^2 = r^2 \tag{3-3}$$

gdje veličine *a* i *b* označavaju središte kruga (*a, b*), a *r* je polumjer istog kruga, kao što je vidljivo na u nastavku prikazanoj slici 3.3. Houghova transformacija slijedi nakon detekcije rubova jer se na taj način lakše određuje kružnica prometnog znaka koji se traži.

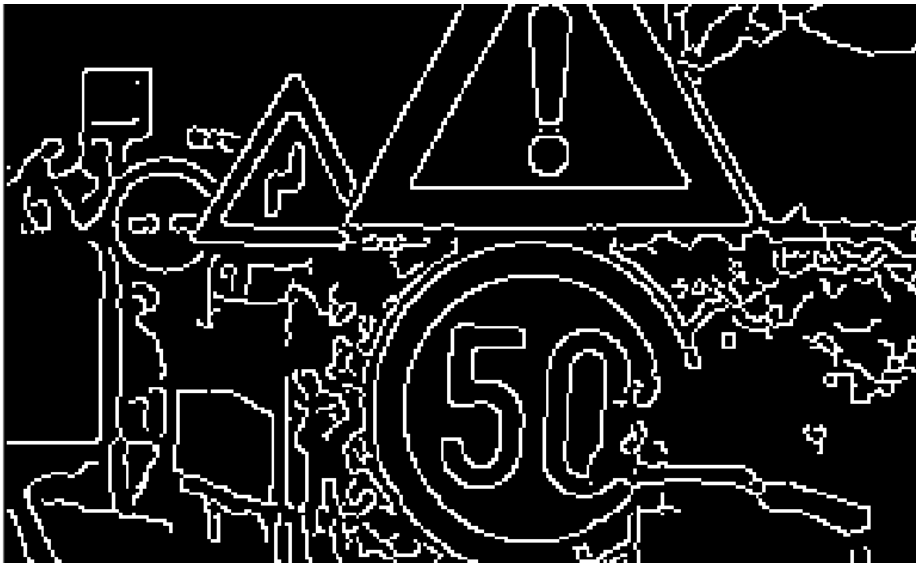


Slika 3.3. Houghova transformacija

3.4. Canny detektor

Uloga Canny detektora je detektiranje rubnih točaka slike kandidata jer takva separacija uvelike olakšava čitav proces detektiranja nekog objekta. Naime, u slučaju znaka ograničenja brzine, na ovaj način se Canny detektorom mogu uspješno odvojiti bitne komponente, kao što su kružnica, bijela pozadina i sadržaj na toj pozadini. Na slici 3.4 prikazana je obična monokromatska fotografija, a na slici 3.5 prikazana je ista fotogafija nakon korištenja Canny detektora. Vidljivo je kako ovaj postupak izdvaja rubove na monokromatskoj slici, a to znatno olakšava postupak pronalaska željenog objekta.

Slika 3.4. Monokromatska fotografija



Slika 3.5. Monokromatska fotografija nakon korištenja Canny detektora

4. DETEKCIJA I PREPOZNAVANJE OBJEKTA SAMO NA OSNOVU INFORMACIJE O BOJI

Često se za detekciju i prepoznavanje znaka ograničenja brzine koristi boja kao veoma važan atribut svakog prometnog znaka. Budući da kadar kamere može biti snimljen u različitim uvjetima, potrebno je ukloniti šum i poboljšati filtre kako uvjeti u kojima je objekt snimljen ne bi imali utjecaj na točnost same detekcije prometnog znaka. Jedna od najvažnijih metoda koja se koristi za prepoznavanje znaka na temelju boje jest ROI (engl. *Region of Interest*). Međutim, pitanje je što napraviti kada boja postane neupotrebljiva, odnosno kada ona izblijedi i postane jedna od nijansi žute boje ili se pak informacija o boji samog objekta potpuno izgubi. To je bitan problem budući da postoji mnogo toga u okolini traženog objekta što može negativno utjecati na prikaz boje. Tada je potrebno prilagoditi granicu segmentacije na temelju boje, a ako je informacija o boji potpuno izgubljena, onda ne preostaje ništa drugo negoli vršiti detekciju na temelju drugog atributa objekta kao što je primjerice njegov oblik [4]. RGB i HSV dva su najčešća modela boje koji se koriste, a od ostalih je važno spomenuti Lab i YcrCb modele boje. RGB je jednostavan te osjetljiv na uvjete osvjetljenja, dok je HSV neosjetljiv na uvjete osvjetljenja, ali je složeniji za izvedbu. [6]. Tri osnovne boje RGB-a su crvena, zelena i plava; tim bojama se opisuje svaki element slike. Svaka od navedenih boja posjeduje intenzitet svjetlosti od 256 vrijednosti, tj. od 0 do 255. Slika 4.1 prikazuje RGB model boje, dok su na slici 4.2, slici 4.3 i slici 4.4 redom prikazane R, G i B komponente RGB modela boje. Očito je kako svaka pojedina slika prikazuje ili crvenu ili zelenu ili plavu boju, odnosno ako se izdvoji samo jedna komponenta, slika se prikazuje u samo jednoj od ove tri boje.

Slika 4.1. RGB prikaz slike

Slika 4.2. Slika koja sadrži samo R komponentu RGB prikaza

Slika 4.3. Slika koja sadrži samo G komponentu RGB prikaza

Slika 4.4. Slika koja sadrži samo B komponentu RGB prikaza

U nastavku, na slici 4.5, prikazana je Y komponenta YCrCb prikaza te je vidljivo kako slika izgleda identično kao monokromatski prikaz iste fotografije. Y predstavlja intenzitet svjetlosti, Cr predstavlja razliku crvene boje i svjetlosti (R-Y), a Cb razliku plave boje i svjetline (B-Y). Za razliku od RGB modela boje, YcbCr model boje, rastavljen na zasebne komponente, na slikama ne sadrži boje kao što je to slučaj kod RGB-a, nego je sve u monokromatskim tonovima. Prikazom R, G i B

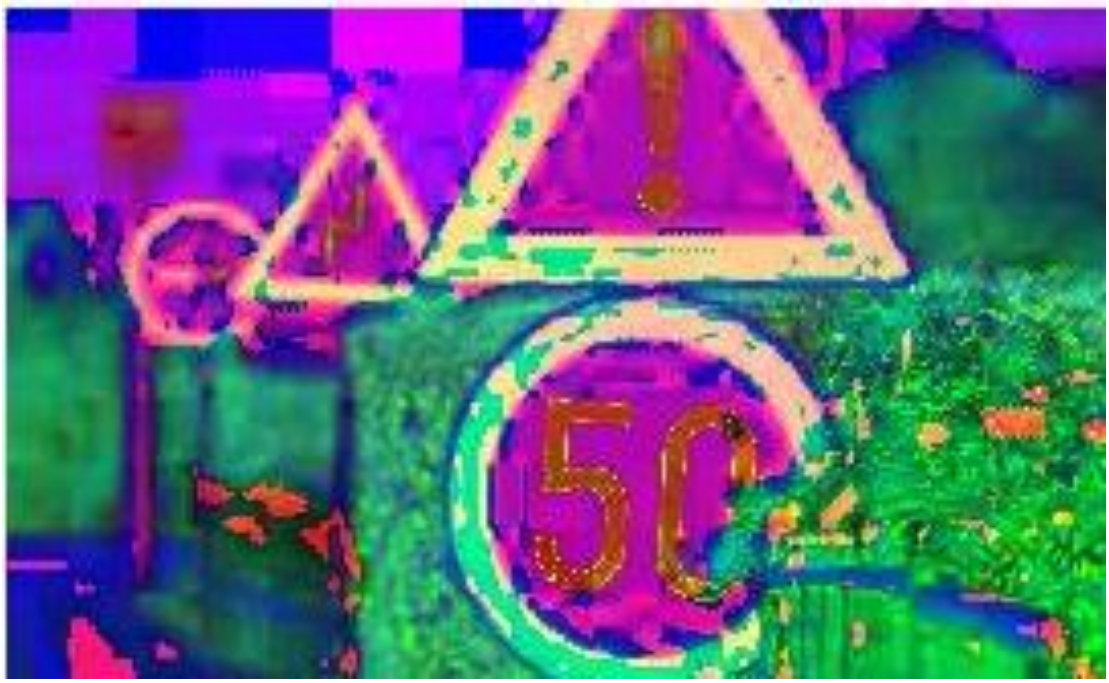
komponente, kao i Y komponente, konture su puno vidljivije negoli prikazom samo Cb ili Cr komponenti koje su vidljive na slikama 4.6 i 4.7. Također, za potrebe ovog rada, vidljivo je kako je izdvajanje Cr komponente korisno jer se na taj način istaknu konture traženog prometnog znaka, odnosno konture kružnog vijenca koji je inače crvene boje.

Slika 4.5. Slika koja sadrži samo Y komponentu YCrCb prikaza

Slika 4.6. Slika koja sadrži samo Cr komponentu YCrCb prikaza

Slika 4.7. Slika koja sadrži samo Cb komponentu YCrCb prikaza

Slika 4.8. prikazuje sliku na kojoj je korišten HSV model boje. Na slici 4.9. također je korišten HSV model boje, ali uz korištenje vrijednosti u rasponu od (0, 25, 25) do (10, 255, 255), tj. uz mijenjanje H vrijednosti od 0 do 10 kako bi se prikazala samo crvena boja na slici.



Slika 4.8. Slika prikazana u HSV-u

Slika 4.9. Slika prikazana u HSV-u ograničena na crvenu boju

4.1. Modeli boje

Kako bi se riješio problem osvjetljenja i smanjilo vrijeme izvršavanja programa, u [18] je predstavljen model vjerojatnosti boje koji računa vjerojatnost svakog elementa slike u odnosu na bilo koju boju prometnog znaka koji se tražio. Veoma važno kod ovog modela je korištenje različitih uzoraka s različitim uvjetima. Kako bi se ostvarila dodatna otpornost na nepovoljne uvjete, preporučeno je koristiti Ohta model boje gdje će se pojaviti N+1 matrica za N setova boja i pozadina. Kako bi se prilagodilo, tj. smanjilo, vrijeme koje je potrebno za konverziju s RGB modela boje na Ohta model boje, koristi se LUT koji je u obliku 256 X 256 X 256 X N elemenata ranije izračunate matrice [6]. Još jedan model boje koji se pokazao veoma učinkovitim u prepoznavanju znaka ograničenja brzine jest Lab model boje, prikazan na slici 4.10, pomoću kojeg se, prema [9], može dizajnirati filter koji bi služio prepoznavanju crvenih elemenata slike na kružnom vijencu prometnog znaka brzine koji se traži, a čija se točnost može povećati koristeći AdaBoost algoritam.

Slika 4.10. Slika prikazana u Lab-u

Još jedan od načina izdvajanja područja boje sa slike, a koji nije segmentacija boje, jest detekcija prometnog znaka koja se, prema [16], provodi uz pomoć formule:

$$S = R - g \tag{4-1}$$

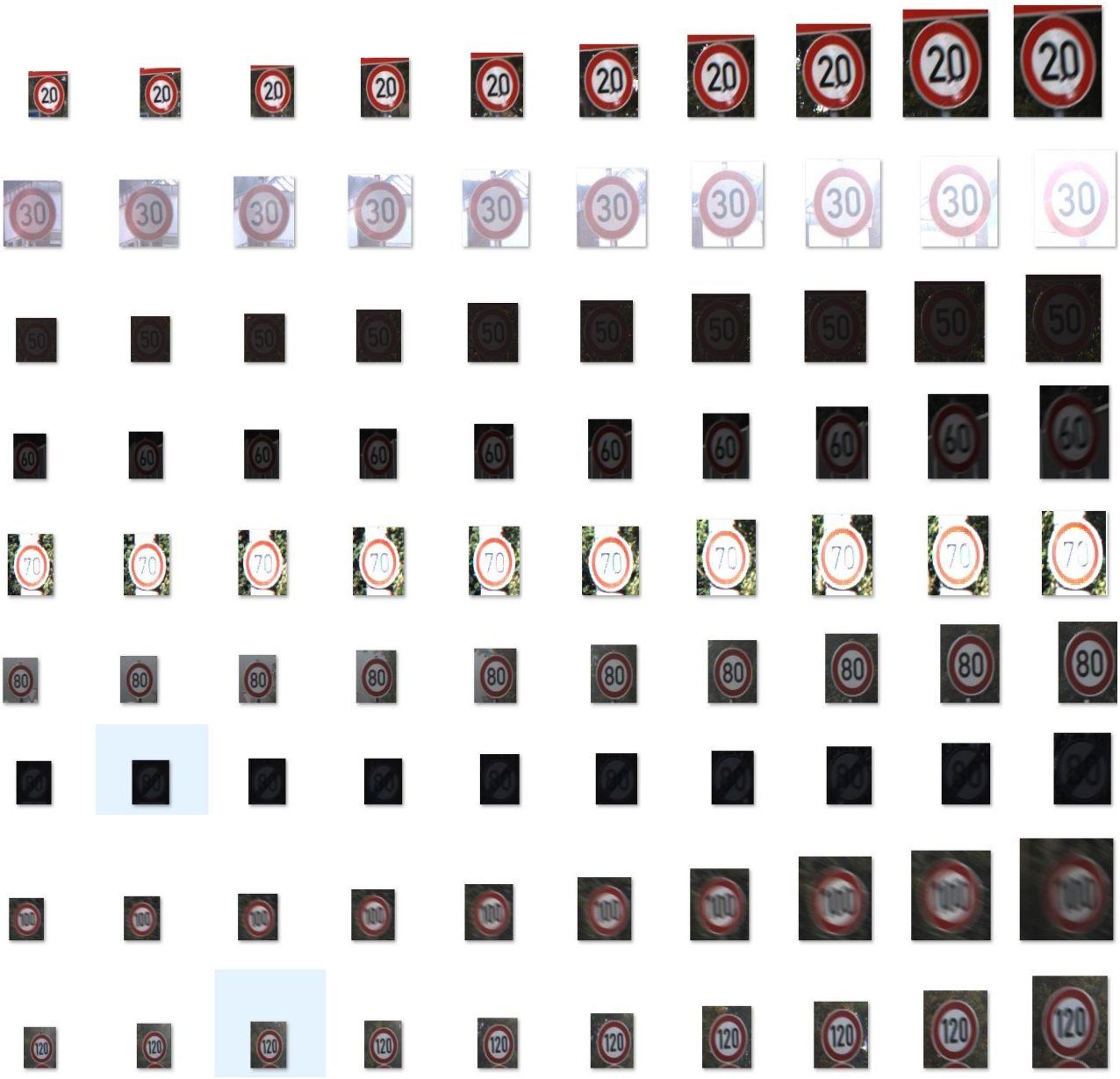
gdje *S* predstavlja segmentiranu sliku. *R*, *G* i *B* su komponente RGB modela boja, a *g* (engl. *gray*) je monokromatska slika iste scene. S veličina je jednaka elementima slike koji preostanu kada se elementi slike monokromatskog kanala oduzmu od njima odgovarajućih elemenata slike kanala crvene boje. Na ovaj način broj kandidata za traženi znak se smanjuje, kao i broj neželjenih objekata. Ostaju sačuvani samo kandidati sa sličnim karakteristikama što smanjuje proces identifikacije traženog objekta.

5. PROGRAMSKO RJEŠENJE ZA PREPOZNAVANJE ZNAKA OGRANIČENJA BRZINE

Programsko rješenje za zadatak ovog završnog rada izrađeno je prema algoritmu predloženom u [19] korištenjem Python programskog jezika. Kako je Python programski jezik koji pruža mogućnost korištenja mnoštva implementiranih biblioteka i funkcija dostupnih svima, tako su i za potrebe ovog rada korištene brojne biblioteke od kojih je za ovaj rad najbitnija OpenCV (eng. *Open Source Computer Vision Library*) bibilioteka. Ona sadrži ogroman broj algoritama, od kojih je velik broj korišten za izradu danog programskog rješenja. Nadalje, programsko rješenje izrađeno je na temelju konvolucijskih neuronskih mreža (eng. *Convolutional Deep Networks*, CNN ili ConvNets), koje su najčešći tip dubokih neuronskih mreža korištenih za potrebe računalnog vida. Korišten je Keras koji je API visoke razine i dolazi s TensorFlow-om, a odlikuje se brzinom izvođenja programskog rješenja s naglaskom na programska rješenja strojnog dubokog učenja.

5.1. Baza podataka za trening i testiranje

Budući da se model najprije treba istrenirati, to se obavlja na bazi podataka. Od nekoliko naširoko korištenih baza podataka koje služe za prepoznavanje i detekciju znakova odabran je Njemački standard prepoznavanja prometnih znakova GTSRB (eng. *German Traffic Sign Recognition Benchmark*). Ova baza inače sadrži 43 klase, sa sveukupno 39209 fotografija seta za treniranje i 12630 fotografija seta za testiranje. Za potrebe rada smanjen je broj kategorija baze podataka, a samim time i broj fotografija, kako bi se treniranje ograničilo samo na fotografije koje prikazuju znakove ograničenja brzine. Čitava je baza fotografija podijeljena na set za treniranje i set za testiranje koji sadrže fotografije koje se međusobno razlikuju. Na ovaj način prepravljeni dio baze podataka koji služi za treniranje sadrži 9 klasa, s ukupno 270 fotografija koje prikazuju znakove ograničenja brzine, snimljenih u različitim uvjetima, različitom osvjetljenju i u različitom okruženju na raznim udaljenostima. Treniranje s takvim fotografijama povećava točnost programskog rješenja. Svaka klasa služi kao predstavnik za svaku pojedinu vrstu prometnog znaka ograničenja brzine. To su redom znakovi koji služe za pokazivanje 20 km/h, 30 km/h, 50 km/h, 60 km/h, 70 km/h, 80 km/h, 100 km/h i 120 km/h te prestanak ograničenja brzine 80 km/h, kao što se može vidjeti na slici 5.1. Na testnom dijelu provjerava se učinkovitost, odnosno točnost, algoritma. Testni dio sadržava 12630 fotografija na kojima su prikazani znakovi ograničenja brzine, ali i ostali prometni znakovi. Korisnik aplikacije prilikom učitavanja slike treba odabrati jednu od fotografija iz testnog dijela koja prikazuje prometni znak ograničenja brzine.

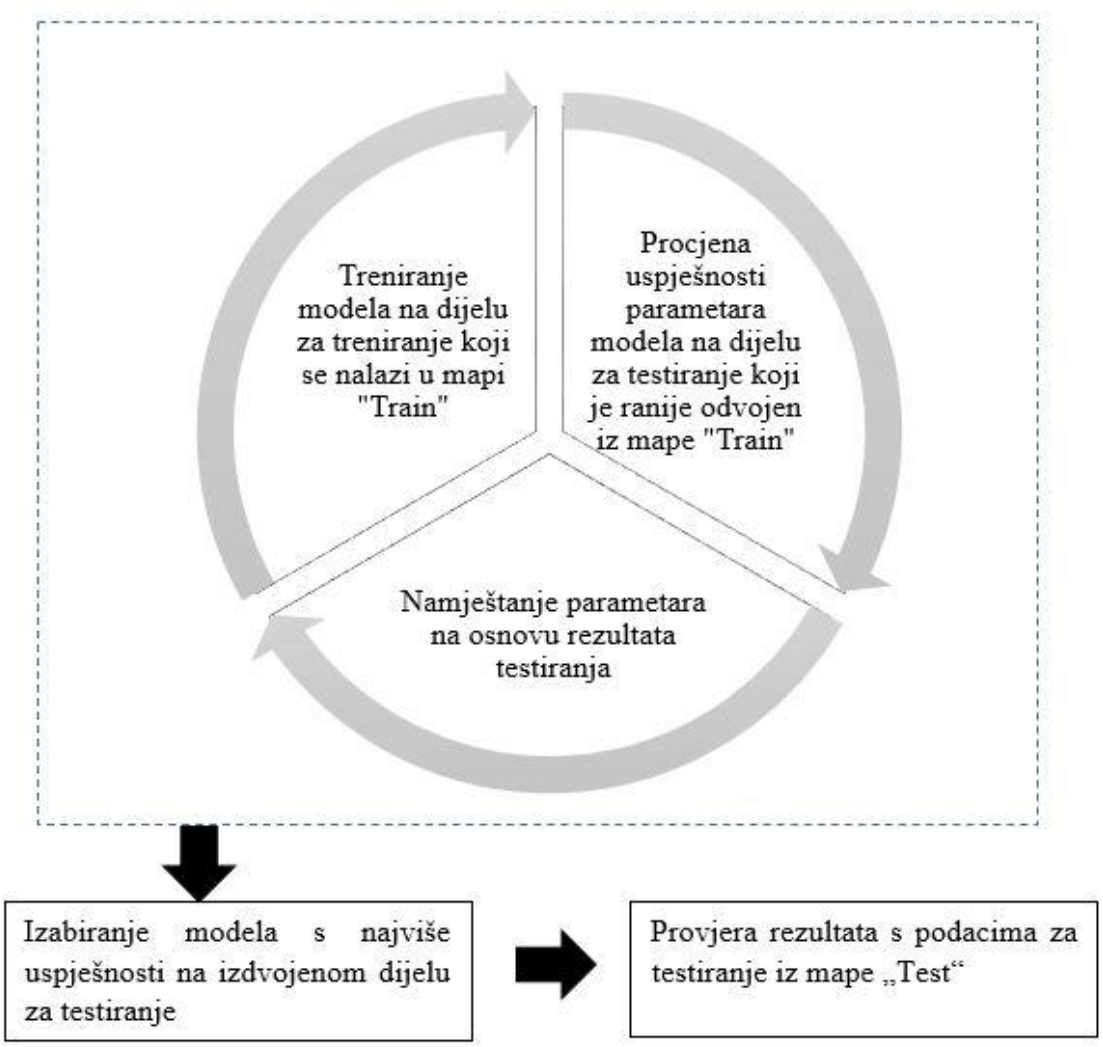


Slika 5.1. Prikaz fotografija svake pojedine klase u dijelu za treniranje

5.2. Dijagram toka

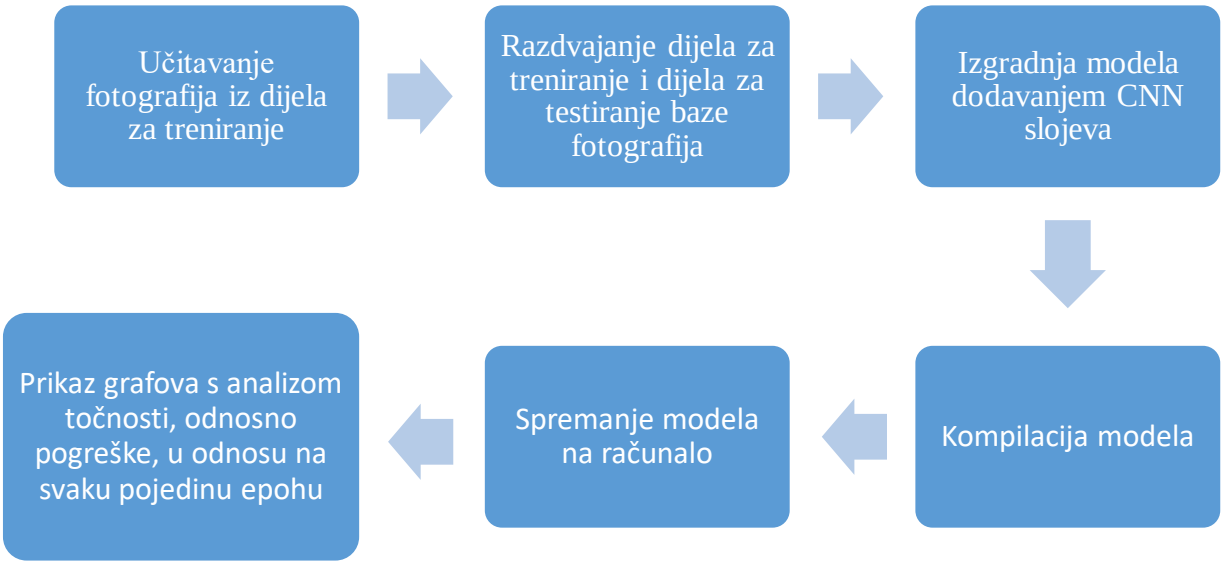
Programsko rješenje izvedeno je pomoću ranije spomenutih neuronskih mreža i Kerasa. Kao što prikazuje slika 5.3 najprije se vrši učitavanje fotografija znakova ograničenja brzine koje se nalaze u dijelu baze podataka za treniranje. Potom se mapa za treniranje dijeli na dva dijela od kojih jedan dio služi za treniranje modela, a drugi dio služi za testiranje uspješnosti treniranja modela. Na temelju tog testiranja u postupku izgradnje modela odlučuje se koji će parametri biti korišteni. Kada ovog postupka testiranja ne bi bilo, odnosno kada bi se parametri izabirali samo na osnovu treniranja na

dijelu za treniranje, vjerojatno bi uspješnost treniranja bila stopostotna, ali bi zato prepoznavanje znaka bilo puno manje točnosti. Treba naglasiti kako ovo testiranje nije jednako testiranju koje se vrši iz mape ”Test” baze podataka jer fotografije te mape služe za provjeru uspješnosti izvođenja programa, dok dio za testiranje iz mape ”Train” služi za namještanje parametara model u postupku izgradnje modela. Ova je razlika prikazana na slici 5.2.



Slika 5.2. Razlika između testnog dijela izdvojenog iz mape za treniranje i testnog dijela iz mape za testiranje

Potom se izgrađuje model na kojeg se dodaju slojevi konvolucijske neuronske mreže. Vrši se kompilacija modela te se isti sprema na računalo. Nakon ovog koraka prikazuju se grafovi koji pokazuju kolika je točnost, odnosno pogreška, modela prema zadanom broju epoha. Dijagram toka program aprikazan je na slici 5.3.



Slika 5.3. Dijagram toka korištenog algoritma

5.3. Neuronske mreže i dijelovi konvolucijskih neuronskih mreža

Sve neuronske mreže predstavljaju modele dubokog učenja (eng. *deep learning*), a u osnovi su građene na temelju proučavanja stvarnih neuronskih mreža živih bića. Svaka neuronska mreža sastoji se od mnoštva slojeva (eng. *layers*) čija se struktura sastoji od neurona.(eng. *neurons*). Neuroni između slojeva međusobno komuniciraju kada za to postoji potreba, odnosno kada se dogodi neka specifična situacija koja zahtijeva navedeno. Konvolucijske mreže sadrže brojne slojeve, a struktura tih slojeva dijeli se na konvolucijski sloj, nelinearni sloj, udruženi (eng. *pooling*) sloj i potpuno povezani sloj [20].

Najjednostavniji model Kerasa je sekvencijalni model (eng. s*equential*). Ovaj model se sastoji od slojeva neuronske mreže. Model sadrži epohe koje su pokazatelji koliko se puta dio za testiranje koristi u određenom modelu. Dijelovi za treniranje i za testiranje parametara modela bivaju razdvojeni iz iste mape ”Train” koja služi za testiranje kako model ne bi zapamtio jedan dio i naučeno testirao na istome. Zato dio za treniranje sadrži jedan skup fotografija, a dio za testiranje skup potpuno drugih fotografija, prikazujući prometne znakove. Ovakav način osigurao je da se nijedna fotografija, koja se nalazi u jednom dijelu, ne nađe u drugom dijelu, odnosno ako postoji neka fotografija u dijelu za treniranje, zasigurno je da se ista ne pojavljuje u dijelu za testiranje, i obratno.

Već je objašnjeno na slici 5.2 po čemu se razlikuju dio za testiranje, dio za treniranje izdvojen iz mape za testiranje i testni dio.

Duboko učenje naziva se upravo tako jer računalo tijekom pokretanja modela uči kako bi se naposljetku ostvarili optimalni rezultati. To znači da neuron u svakom sljedećem sloju primjenjuje znanje stečeno ranije, na prethodnim slojevima, i na taj način izbjegava greške. Ovakvo učenje predstavlja se funkcijom, a Keras pruža različite funkcije za odabir. Jedna od tih funkcija je softmax funkcija. Ona pokazuje kolika je vjerojatnost distribucije neke diskretne varijable s *n* mogućih vrijednosti. Softmax funkcija najčešće se koristi za davanje nekih izlaznih vrijednosti klasifikatora, kako bi se pokazala vjerojatnost distribucije *n* različitih klasa. Ako se ne koristi za izlazne vrijednosti nego u modelu, ova funkcija sudjeluje u odlučivanju modela između *n* različitih opcija za postavljanje neke varijable. Prema [21], ako je ta varijabla binarna:

$$\hat{y} = P(y = 1 \mid x) \tag{5-1}$$

Budući da se ovaj broj nalazi između 0 i 1, a jer se logaritmom tog broja mora lako upravljati sukladno optimizaciji temeljenoj na gradijentu, traži se *z* izrazom:

$$z = \log P(y = 1 \mid x) \tag{5-2}$$

Diskretna varijabla $\hat{y}_i$ koja ima *n* vrijednosti treba biti između 0 i 1, ali također treba i čitava suma vektora biti 1 kako bi vjerojatnost distribucije bila vjerodostojna. Nadalje, linearni sloj predviđa nenormalizirane log vjerojatnosti:

$$z = W^T h + b \tag{5-3}$$

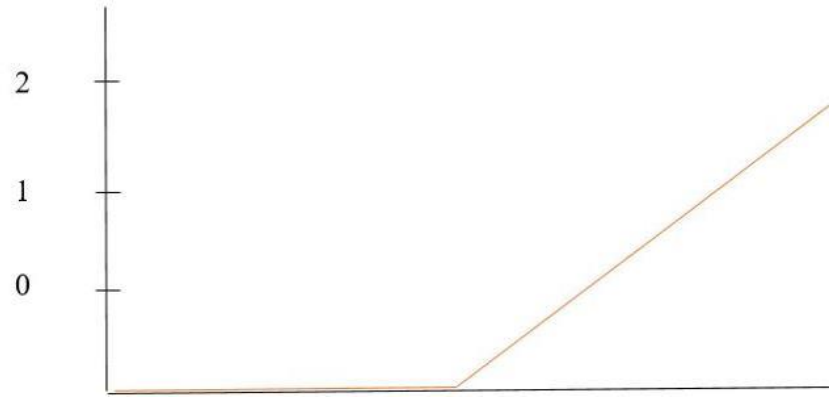
gdje je

$$z_i = \log P(y = 1 \mid x) \tag{5-4}$$

Kako bi se održala željena vrijednost $\hat{y}_i$, potencira se i normalizira *z* pomoću softmax funkcije:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\Sigma_j \exp(z_j)} \tag{5-5}$$

Prirodno je da bi najbolje rezultate trebala davati funkcija koja uči postepeno. Zato je u radu, pored aktivacijske funkcije softmax, korištena aktivacijska funkcija ReLU (eng. *rectified linear unit*). Ova funkcija predstavljena na slici 5.3 se, prema [22], predstavlja kao $f(x) = \max(0, x)$. Naime, za sve negativne vrijednosti, iznos funkcije je 0, a pri pozitivnim vrijednostima funkcija raste linearno te pri tim vrijednostima derivacija ove funkcije iznosi 1, što je uzrok tome da se gradijent ne mijenja te se zato neuronska mreža ne mora baviti problemom gradijenta koji bi inače mogao biti uzrok prestanku postupka treniranja.



Slika 5.3. Prikaz ReLU funkcije

5.4. Odabir parametara

Budući da su grafovi točnosti i pogreške predstavljeni na osnovu izvedenih epoha, mijenjan je broj epoha kako bi se utvrdio njihov optimalni broj uz koji bi programsko rješenje imalo najveću razinu točnosti. Analizom tablica koje se nalaze na slici 5.4, slici 5.7 i slici 5.10, čiji su podaci sakupljeni iz podataka ispisanih pri pokretanju programa, optimalne i najstabilnije rezultate daje broj od 20 epoha. Tablice su dobivene na način da je za svaki od navedenog broja epoha, dakle za 10, 15 i 20 epoha, zasebno pokrenut postupak treniranja. Za svaki broj epoha predstavljena su po dva grafa koji ilustriraju podatke navedene u tablicama, a koji bivaju ispisani nakon kompilacije modela. Ovi grafovi su u nastavku predstavljeni slikama 6.2 i 6.3 za broj od 10 epoha, slikama 6.5 i 6.6 za broj od 15 epoha te slikama 6.8 i 6.9 za broj od 20 epoha. Na grafovima je vidljivo kako se visoka točnost od 95,19% može ostvariti već s 10 epoha, međutim, najveća točnost je postignuta s 20 epoha, s postotkom točnosti od 99,05%.

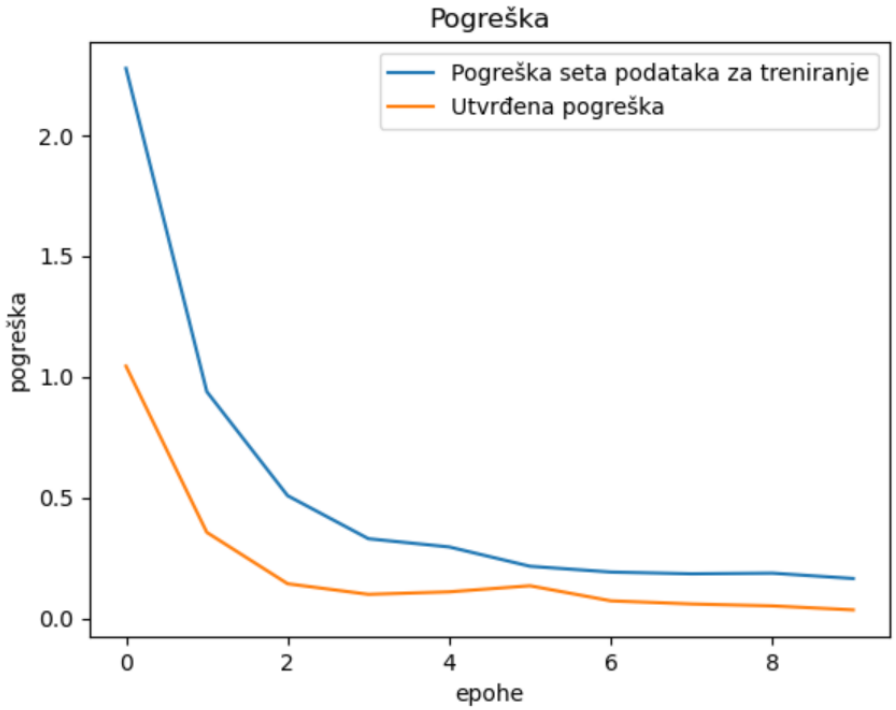
Pogreška prikazana u dolje navedenim tablicama ne predstavlja vjerojatnost te zato u nekim slučajevima ima vrijednost veću od 1. Ona predstavlja vrijednost dobivenu metodom najmanjih

kvadrata ili pak vrijednost logaritamske funkcije koja govori kolika je vrijednost logaritma vjerojatnosti da će rezultat treniranja biti pogrešan, odnosno da će prometni znak biti pogrešno prepoznat. Na grafovima koji su predstavljeni na slikama u nastavku, pogreška seta podataka za treniranje pokazuje rezultate dobivene izračunom pogreške na podacima za treniranje, dok je utvrđena pogreška rezultat izračuna pogreške na podacima ranije izdvojenim iz podataka za treniranje koji služe za testiranje parametara modela. Nadalje, što je niža vjerojatnost ove vrijednosti pogreške, treniranje je uspješnije, tj. treniranje je veće točnosti. U postupku traženja ove vrijednosti pogreške koristi se logaritamska funkcija kako bi izvođenje matematičkih operacija množenja parametara na računalu bilo uspješno. Dok se ova vrijednost ne počne smanjivati, primjenjuju se razne optimizacijske metode kojima se namještaju optimalni parametri koji tvore model. Nakon svake optimizacije, cilj je da vrijednost pogreške bude manja od prethodne iteracije.

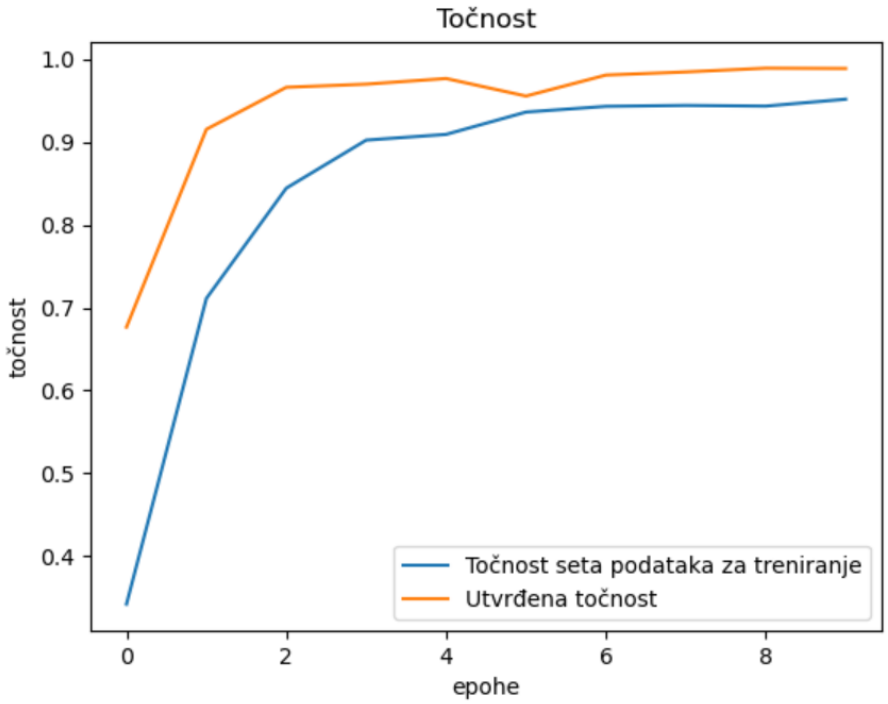
Točnost, za razliku od pogreške, predstavlja vjerojatnost te stoga ima vrijednosti manje ili jednake broju 1. Računa se nakon utvrđivanja parametara te stoga ne utječe na mijenjanje, odnosno optimizaciju, parametara kao što je to slučaj kod vrijednosti pogreške. Računa se uspješnost prepoznavanja traženog prometnog znaka na testnim slikama te se dobivene vrijednosti uspoređuju s podacima koji opisuju svaku testnu sliku i u kojima se nalazi informacija o tome koji se znak treba prepoznati. Kao i kod pogreške, točnost i utvrđena točnost razlikuju se u podacima na kojima se obavlja usporedba. Kod točnosti su ti podaci oni koji služe za testiranje, dok su podaci korišteni za dobivanje utvrđene točnosti podaci koji su ranije izdvojeni iz podataka za treniranje kako bi se pomoću njih testirali parametri modela. Cilj je dobiti obrnutu vrijednost od vrijednosti pogreške. Dakle, ako se s većim brojem iteracija, tj. epoha, smanjuje pogreška, a povećava točnost, to znači da je učenje mreže uspješno provedeno. Takav pad vrijednosti pogreške te povećanje vrijednosti točnosti pokazuju i tablice prikazane u nastavku.

Epoha	Pogreška	Točnost	Utvrđena pogreška	Utvrđena točnost
1/10	2.2778	0.3418	1.0442	0.6765
2/10	0.9385	0.7112	0.3556	0.9155
3/10	0.5076	0.8444	0.1427	0.9663
4/10	0.3293	0.9025	0.0988	0.9701
5/10	0.2952	0.9094	0.1091	0.9769
6/10	0.2151	0.9364	0.1343	0.9557
7/10	0.1912	0.9433	0.0719	0.9811
8/10	0.1839	0.9444	0.0589	0.9848
9/10	0.1866	0.9436	0.0509	0.9894
10/10	0.1643	0.9519	0.0344	0.9890

Slika 5.4. Tablica koja prikazuje vrijednosti različitih parametara tijekom 10 epoha



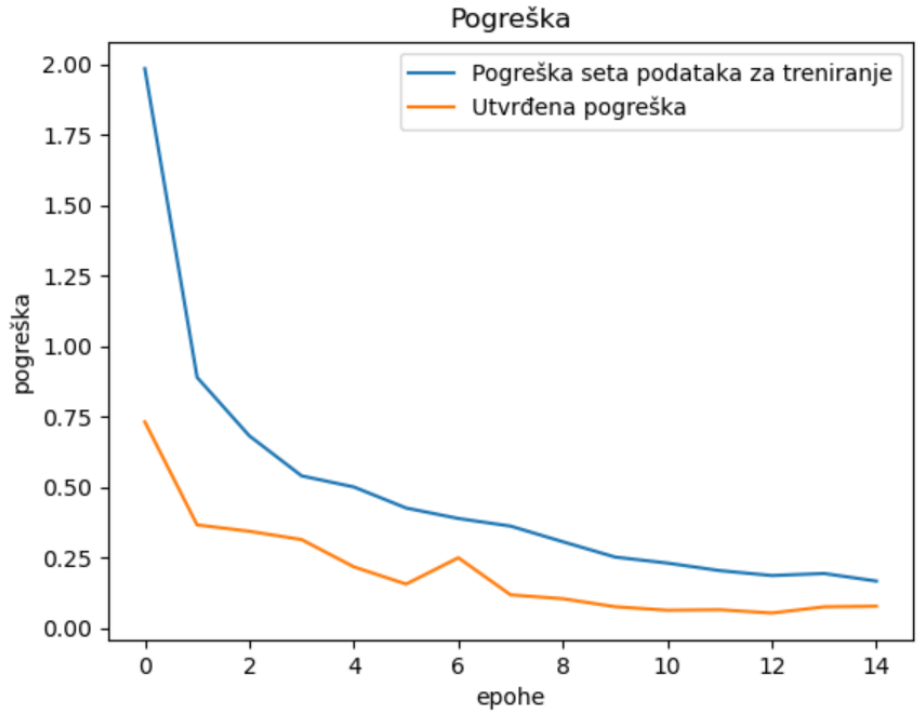
Slika 5.5. Graf pogreške dijela za treniranje i potvrđene pogreške tijekom 10 epoha



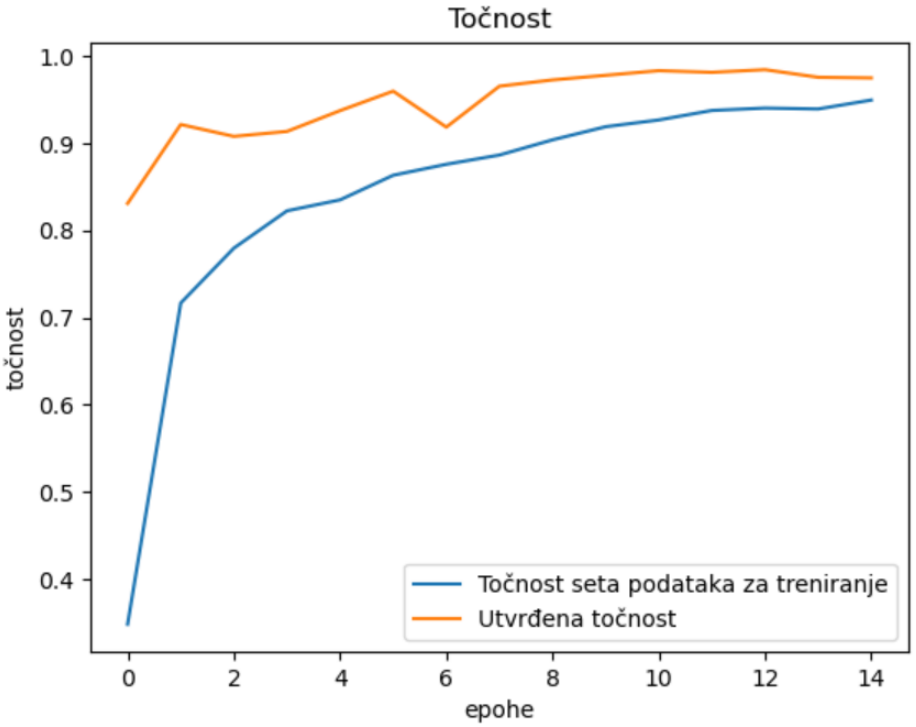
Slika 5.6. Graf točnosti dijela za treniranje i potvrđene točnosti tijekom 10 epoha

Epoha	Pogreška	Točnost	Utvrđena pogreška	Utvrđena točnost
1/15	1.9858	0.3483	0.7326	0.0785
2/15	0.8905	0.7168	0.3670	0.9216
3/15	0.6830	0.7797	0.3445	0.9080
4/15	0.5411	0.8225	0.3150	0.9136
5/15	0.5016	0.8351	0.2182	0.9375
6/15	0.4270	0.8634	0.1571	0.9598
7/15	0.3898	0.8759	0.2505	0.9186
8/15	0.3627	0.8866	0.1188	0.9655
9/15	0.3073	0.9040	0.1052	0.9727
10/15	0.2529	0.9190	0.0767	0.9780
11/15	0.2316	0.9268	0.0643	0.9833
12/15	0.2050	0.9377	0.0643	0.9814
13/15	0.1874	0.9404	0.0544	0.9845
14/15	0.1946	0.9394	0.0763	0.9758
15/15	0.1678	0.9496	0.0785	0.9750

Slika 5.7. Tablica koja prikazuje vrijednosti različitih parametara tijekom 15 epoha



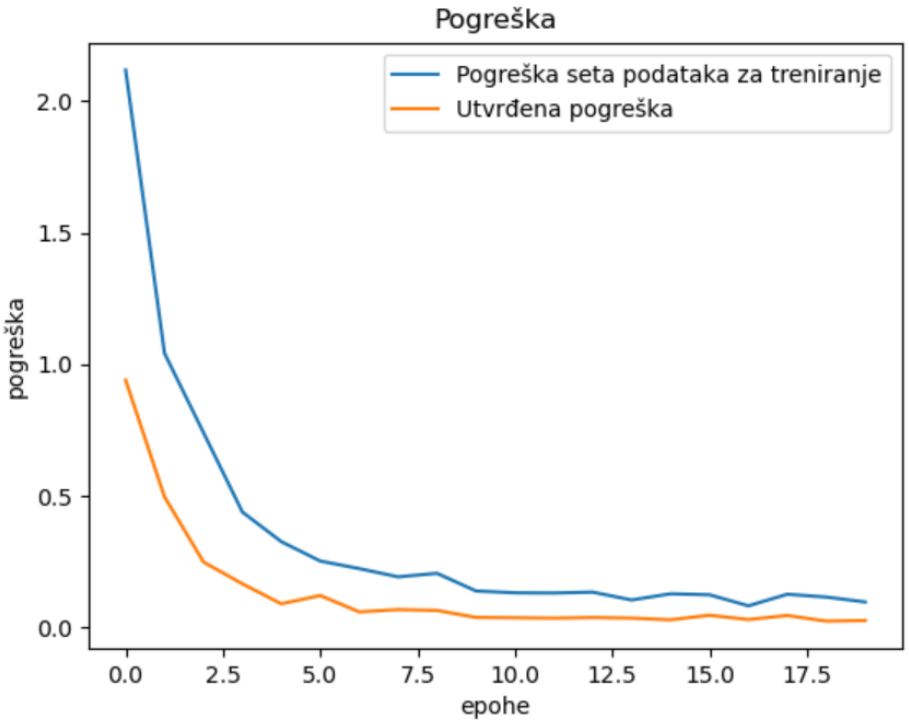
Slika 5.8. Graf pogreške dijela za treniranje i potvrđene pogreške tijekom 15 epoha



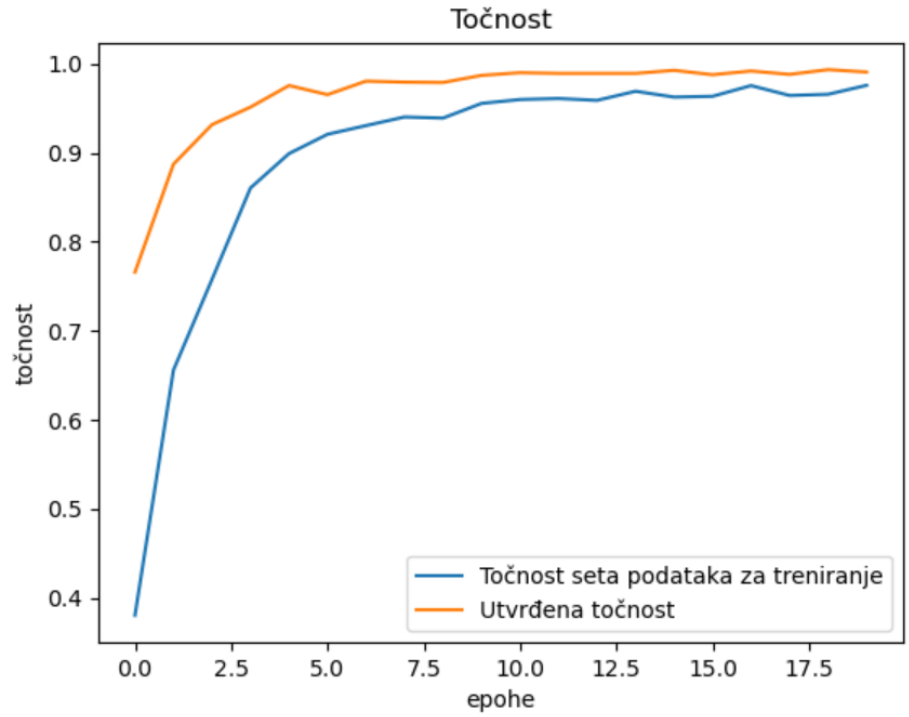
Slika 5.9. Graf točnosti dijela za treniranje i potvrđene točnosti tijekom 15 epoha

Epoha	Pogreška	Točnost	Utvrđena pogreška	Utvrđena točnost
1/20	2.1175	0.3808	0.9399	0.7659
2/20	1.0411	0.6561	0.4959	0.8871
3/20	0.7404	0.7578	0.2489	0.9314
4/20	0.4386	0.8604	0.1659	0.9511
5/20	0.3262	0.8990	0.0900	0.9754
6/20	0.2524	0.9206	0.1216	0.9652
7/20	0.2238	0.9304	0.0595	0.9803
8/20	0.1925	0.9401	0.0681	0.9792
9/20	0.2058	0.9388	0.0649	0.9788
10/20	0.1388	0.9553	0.0385	0.9867
11/20	0.1318	0.9597	0.0371	0.9898
12/20	0.1311	0.9608	0.0355	0.9890
13/20	0.1343	0.9587	0.0380	0.9890
14/20	0.1047	0.9689	0.0357	0.9890
15/20	0.1281	0.9624	0.0293	0.9924
16/20	0.1244	0.9633	0.0467	0.9875
17/20	0.0824	0.9753	0.0303	0.9917
18/20	0.1264	0.9642	0.0457	0.9879
19/20	0.1157	0.9655	0.0243	0.9932
20/20	0.0971	0.9756	0.0266	0.9905

Slika 5.10. Tablica koja prikazuje vrijednosti različitih parametara tijekom 20 epoha



Slika 5.11. Graf pogreške dijela za treniranje i potvrđene pogreške tijekom 20 epoha



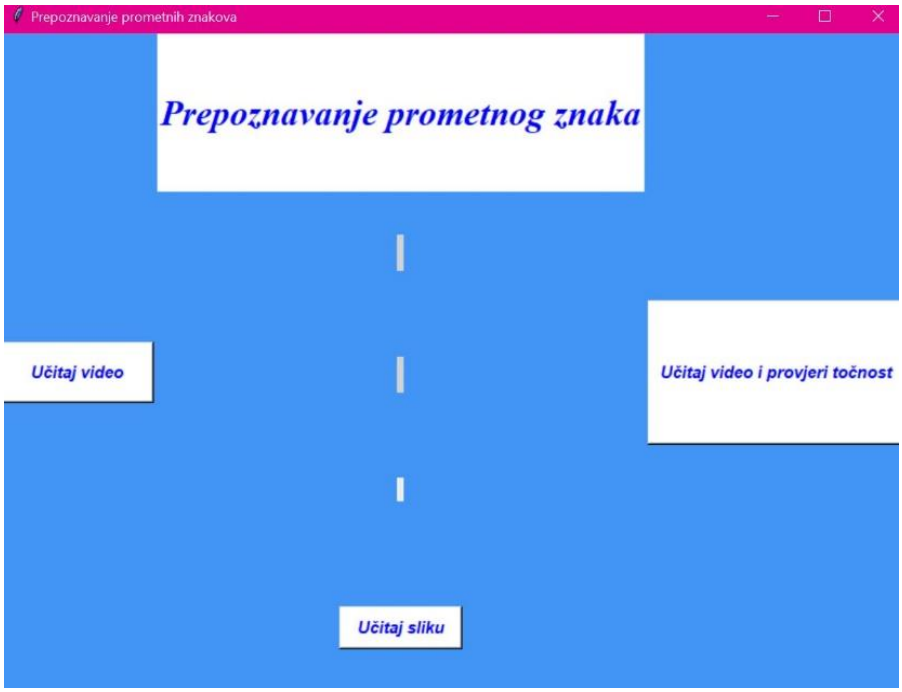
Slika 5.12. Graf pogreške dijela za treniranje i potvrđene pogreške tijekom 20 epoha

6. REZULTATI PROGRAMSKOG RJEŠENJA

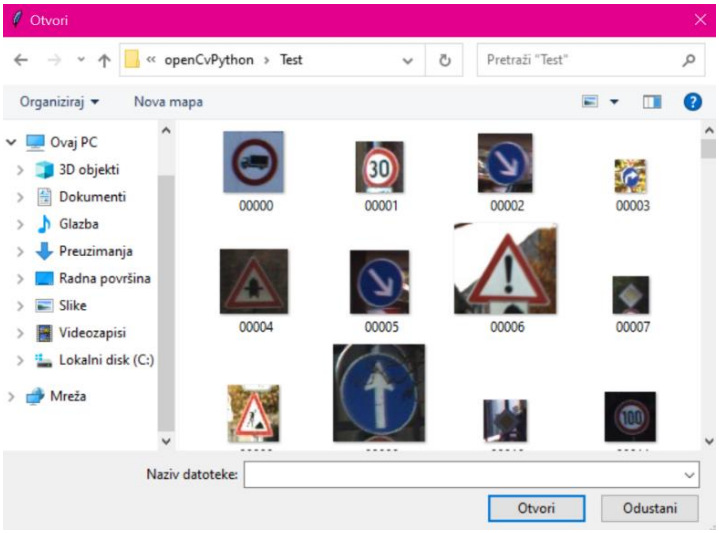
Sučelje za programsko rješenje detekcije znakova ograničenja brzine izrađeno je uz pomoć Pythonovog standardnog Tkinter korisničkog sučelja, a prilagođeno je ne samo za učitavanje fotografija nego i za učitavanje videozapisa. Razlog tome leži u namjeri da se programsko rješenje predstavljeno u ovom radu analizira na što vjerodostojniji način. Dakle, analiza učinkovitosti ovog programskog rješenja koje služi za prepoznavanje znakova ograničenja brzine na ovaj način nije ograničeno samo na analizu fotografija iz dijela za testiranje, nego i na analizu fotografija koje predstavljaju okvire videozapisa. Moguće je učitati bilo koji videozapis, ali za analizu su korišteni vlastiti videozapisi snimljeni za potrebe ovog rada.

6.1. Mogućnosti aplikacije za detekciju prometnih znakova ograničenja brzine

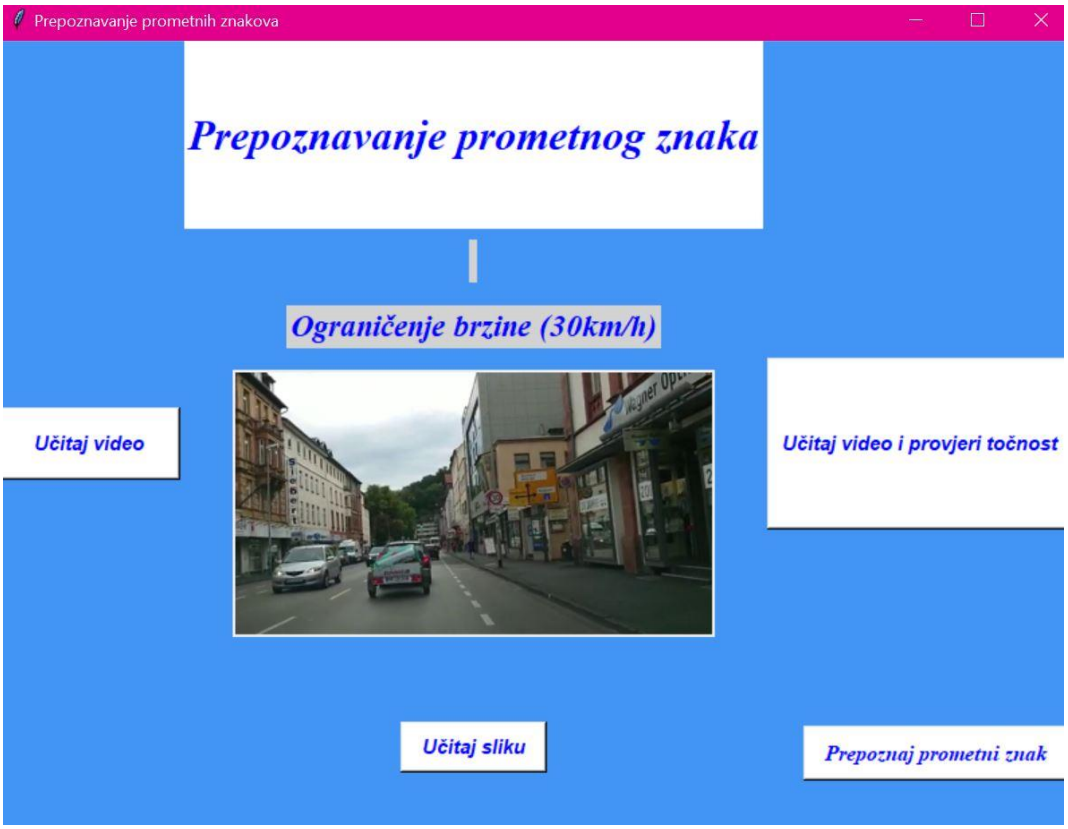
Na sučelju aplikacije za detekciju prometnih znakova korisnik može odabrati tri opcije „Učitaj sliku“, „Učitaj video“ te „Učitaj video i provjeri točnost“ kao što prikazuje slika 6.1. Svaka od tih opcija potrebna je kako bi korisnik mogao iskoristiti sve funkcionalnosti izgrađenog programskog rješenja. Kako bi pristupio fotografijama, korisnik treba odabrati opciju “Učitaj sliku”. Time se pruža mogućnost pristupu željenoj slici koja služi za testiranje dane aplikacije te se otvara prozor prikazan na slici 6.2. Isti prozor otvara se i kada korisnik klikne na opciju “Učitaj video”. Nakon odabira željene fotografije u danom prozoru, prikaz fotografije pojavljuje se na korisničkom sučelju, a osim toga se pojavljuje i dodatna opcija “Prepoznaj prometni znak”. Klikom na tu opciju vrši se prepoznavanje znaka ograničenja brzine pomoću prethodno treniranog algoritma te se, kao što prikazuje slika 6.3, rezultat ispisuje iznad fotografije. Pored ovih opcija, korisniku je pružena mogućnost korištenja i opcije “Učitaj video i provjeri točnost”. Ova opcija radi sve što i opcija “Učitaj video”, ali pored učitavanja videozapisa i spremanja okvira u zasebnu mapu “frames”, program učitava okvire izabranog jednog od vlastitih videozapisa i za svaki okvir provjerava koji je prometni znak na njemu prikazan. Nakon te provjere, u korisničkom sučelju bivaju ispisani postotci točno, odnosno pogrešno, prepoznatih znakova, kako je i prikazano na slici 6.4.



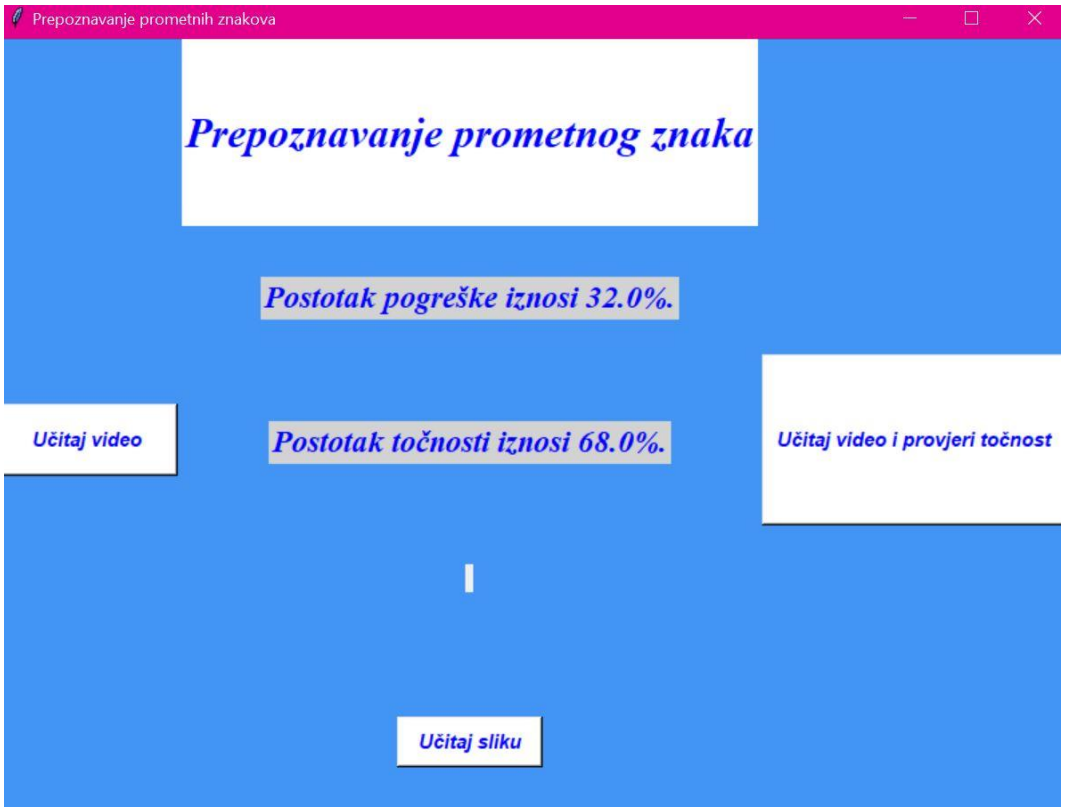
Slika 6.1. Korisničko sučelje s danim opcijama za učitavanje slike/videozapisa



Slika 6.2. Prozor koji se pojavljuje nakon odabira opcije učitavanja i koji služi za odabir željene datoteke



Slika 6.3. Korisničko sučelje nakon učitavanja fotografije i klika na opciju „Prepoznaj sliku“



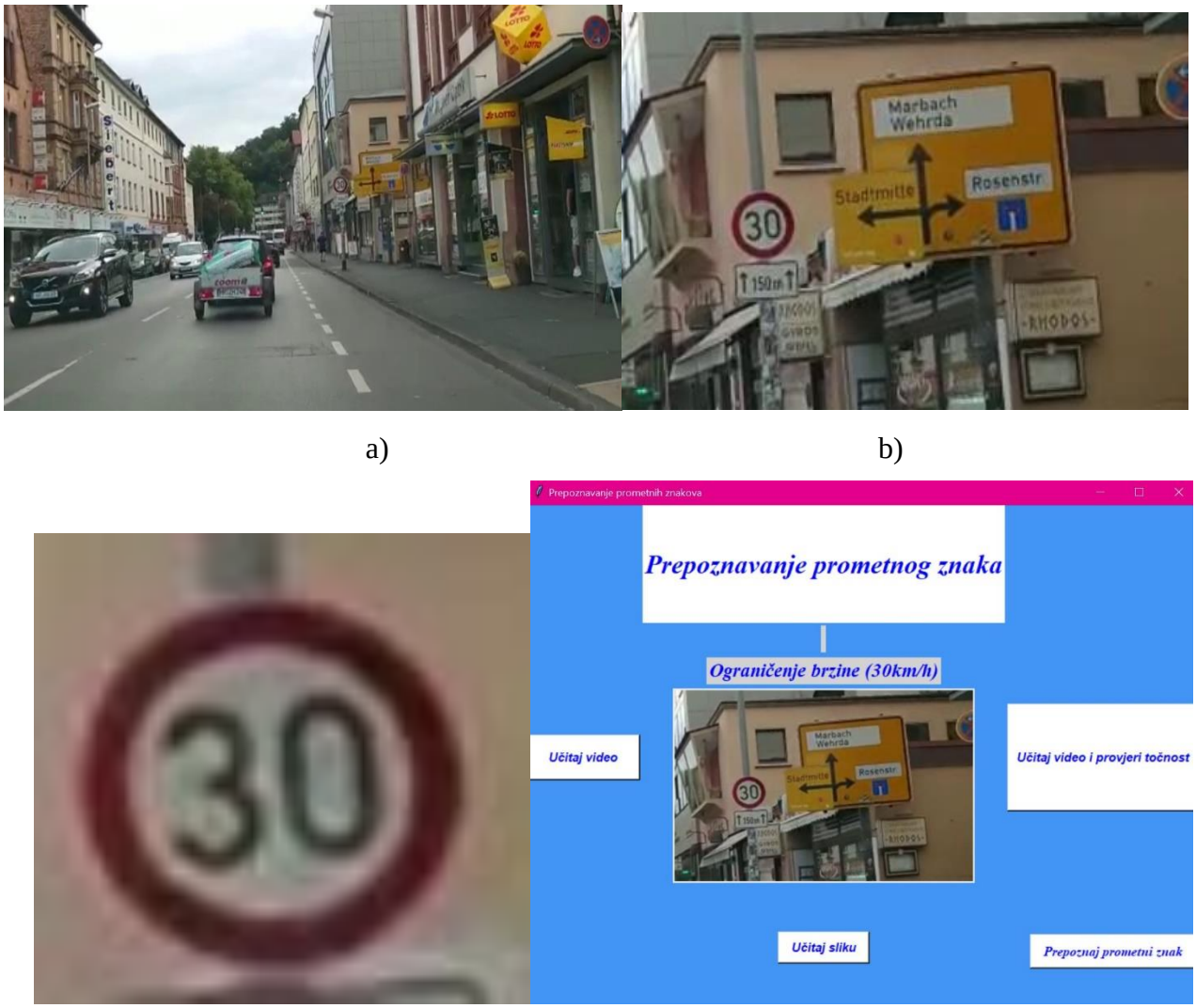
Slika 6.4. Korisničko sučelje nakon učitavanja videa klikom na opciju „Učitaj video i provjeri točnost“ i učitavanja niza okvira te ispisa postotka točno, odnosno pogrešno, prepoznatih znakova

6.2. Usporedba rezultata detekcije prometnog znaka na fotografiji i iz okvira videozapisa

Kao što je ranije rečeno, za potrebe rada snimljena su dva videozapisa, na gradskim ulicama i na autocesti. Pritiskom opcije “Učitaj video” i odabirom željenog videa, u istom se trenutku (ako ranije nije stvorena), stvara mapa “frames” u koju se spremaju okviri videozapisa. Ista se mogućnost pruža i klikom na opciju ”Učitaj video i provjeri točnost” koja, pored učitavanja videozapisa, pruža i mogućnost provjere postotka točno prepoznatih znakova ograničenja brzine. Zatim, klikom na opciju “Učitaj sliku”, korisnik može odabrati bilo koju fotografiju iz mape “frames”, odnosno bilo koji okvir ranije otvorenog videozapisa.

Uočeno je kako se pri učitavanju izdvojenih okvira videozapisa smanjuje točnost algoritma budući da okviri videozapisa prikazuju ne samo znak ograničenja brzine već i čitavu okolinu koja se nalazi u blizini prometnog znaka. Zato su u mapu gdje se nalaze datoteke baze podataka dodane datoteke u kojima je izdvojena nekolicina fotografija od ukupno 3310 izdvojenih okvira oba videozapisa. Te fotografije su izrezane tako da prikazuju samo znakove ograničenja brzine koji se nalaze u videozapisima. Fotografije su poredane od najveće prema najmanjoj udaljenosti od promatranog znaka ograničenja brzine. Neke su fotografije izrezane na način da prikazuju samo prometni znak, a neke da prikazuju znak i čitavu okolinu oko tog znaka ili pak samo dio te okoline.

Učitavanjem tih fotografija u aplikaciju vidljivo je kako se točnost prepoznavanja znaka ograničenja brzine povećava ako se udaljenost od traženog objekta smanjuje. Na slici 6.5 nalazi se fotografija koja se dobije izdvajanjem okvira videozapisa nakon uporabe opcije “Učitaj video” (Slika 6.5. a), isti taj okvir obrađen kako bi se smanjio utjecaj elemenata okoline na prepoznavanje znaka (Slika 6.5 b), traženi znak ograničenja brzine koji se na tom okviru traži (Slika 6.5. c) te uspješno prepoznavanje znaka nakon uporabe opcije “Prepoznaj sliku” (Slika 6.5. d).



a) b)

c) d)

Slika 6.5. Prepoznavanje znaka ograničenja brzine iz izdvojenog okvira videozapisa: a) Okvir videozapisa, b) Obrađena fotografija okvira, c) Prikazan znak koji se traži, d) Uspješno prepoznavanje prometnog znaka u korisničkom sučelju

6.3. Uspješnost prepoznavanja prometnih znakova ograničenja brzine

Za primjer točnosti prepoznavanja znaka ovisno o udaljenosti tog znaka od kamere automobila, uzet je uzorak od 10 obrađenih okvira videozapisa koji prikazuju znak ograničenja brzine od 100 km/h. Okviri te obrađeni okviri korišteni za ispitivanje točnosti prepoznavanja prometnog znaka ograničenja brzine ovisno o udaljenosti prikazani su na slici 6.6. U tablici prikazanoj na slici 6.7, u kojoj se nalaze podaci svakog od okvira te broj elemenata slike koje zauzima traženi prometni znak na originalnom okviru, vidljivo je kako je na prvoj polovini puta, kada je udaljenost veća, točnost prepoznavanja traženog znaka manja te obratno, što je udaljenost manja, točnost se povećava. Osim

udaljenosti, činjenica je kako i količina zamućenosti obrađenog okvira također utječe na uspješnost prepoznavanja te bi stoga okvire trebalo dodatno izoštriti za povećanu točnost prepoznavanja prometnog znaka.

a) b)

c) d)

e) f)


g) h)


i) j)


k) l)

m) n)

o) p)

q) r)

s) t)

Slika 6.6. Prikaz obrađenih slika (desno) iz izdvojenih okvira videozapisa (lijevo)

Objekt detektiran?	Broj elemenata slike dijela video okvira koji prikazuju prometni znak	Dio puta	Točnost prepoznavanja
NE	32 × 30	1.polovina	0%
NE	25 × 26		
NE	24 × 27		
NE	35 × 32		
NE	39 × 39		
DA	43 × 42	2.polovina	100%
DA	56 × 47		
DA	61 × 51		
DA	73 × 65		
DA	95 × 88		

Slika 6.7. Primjer kako udaljenost prometnog znaka utječe na točnost prepoznavanja

Također, za potrebe analize uspješnosti prepoznavanja prometnog znaka ograničenja brzine, klikom na opciju „Učitaj sliku“ i „Prepoznaj prometni znak“ kroz program je provučeno 100 testnih slika od kojih svaka prikazuje neki od znakova ograničenja brzine. Analizom rezultata prepoznavanja znaka, točnost prepoznavanja iznosi 98%. Slike na kojima je odrađen postupak prepoznavanja prikazuju prometne znakove u raznim vremenskim uvjetima, uvjetima osvjetljenja, zamagljenosti te s različitom vrijednošću informacije o boji. Od 100 slika korištenih za ispitivanje točnosti

prepoznavanja znaka, na slici 6.7 prikazane su fotografije na kojima se prepoznavanje pokazalo neuspješnim. Obe fotografije prikazuju ograničenje brzine 120 km/h. Postoji mogućnost da se ovaj problem riješi na način da se broj slika za treniranje koje prikazuju ograničenje brzine od 120 km/h poveća. Ispitivanje uspješnosti prepoznavanja prometnog znaka ograničenja brzine je također napravljeno i s jednim od vlastitih videozapisa, na uzorku od 200 okvira, pomoću opcije „Učitaj video i provjeri točnost“. Pri tome je postignuta točnost prepoznavanja od 54,7%. Uspoređujući taj rezultat s rezultatima prepoznavanja prometnog znaka ograničenja brzine na testnim slikama može se zaključiti da je točnost prepoznavanja na uzorku testnih slikapuno bolja jer iznosi 98% . Ovakva je vrijednost usporedbe očekivana budući da se u videozapisu uvjeti vidljivosti, zamagljenosti, osvjetljenja, udaljenosti i sl. mijenjaju te prepoznavanje ne može biti uspješno onoliko koliko je uspješno na testnim slikama koje ne prikazuju nikakve elemente okoline već samo prometni znak ograničenja brzine. Radi usporedbe, treba naglasiti kako primjerice prvi okvir korištenog videozapisa na ukupno tek 27x24 elemenata slike ima prikazan traženi znak. Ova se veličina do posljednjeg okvira videozapisa poveća do samo 95x88 elemenata slike, za razliku od testnih slika koje prikazuju isključivo traženi prometni znak. Treba naglasiti i kako je za čitav proces učitavanja videa, prikaza i spremanja okvira, ispisivanja vrijednosti prometnih znakova te, naposljetku, ispisivanja postotka točnosti prepoznavanja u korisničko sučelje, programu trebalo 12 sekundi. Kako su za sve slike korištene jednake vrijednosti parametara, iako se slike razlikuju u različitim segmentima, osobito okviri videozapisa, rezultati se smatraju zadovoljavajućima.

a) b)

Slika 6.7. Slike na kojima je prepoznavanje neuspješno obavljeno od ukupnog uzorka od 100 testnih slika: a) Ograničenje brzine (120 km/h) pogrešno prepoznato kao ograničenje brzine (80 km/h), b) Ograničenje brzine (120 km/h) pogrešno prepoznato kao ograničenje brzine (80 km/h)

7. ZAKLJUČAK

U današnjem svijetu, kada ljudi imaju potrebu korištenja automobilskih sustava, veoma je važna sigurnost vozača, ali i svega što se nađe na vozačevom putu. Zato je potrebno da sustavi korišteni u automobilima budu iznimne točnosti. Što se tiče prepoznavanja prometnog znaka ograničenja brzine, ono se može odraditi na mnogo načina. Mnoge metode korisne za ovaj postupak su otkrivene i naveliko korištene, a u radu je pružen pregled nekih dosad objavljenih metoda. Za detekciju i prepoznavanje prometnih znakova ograničenja brzine mogu se koristiti značajke boje i/ili oblika znaka, a najčešće se koriste postupci detekcije ruba, Houghova transformacija za detekciju kružnice, metoda potpornih vektora te neuronske mreže.

Za ovaj rad zadatak je bio prepoznati znak ograničenja brzine iz slike kamere na vozilu, a za rješavanje su korištene neuronske mreže, tj. konvolucijske mreže koje su podtip neuronskih mreža. Prepoznavanje objekta na slici pomoću neuronske mreže je u današnje vrijeme jedan od najbržih i najtočnijih metoda obrade slike i korišten je u gotovo svakom autonomnom sustavu koji zahtijeva visoku točnost. Aplikacija za prepoznavanje znaka u ovom radu razvijena je korištenjem Python programskog jezika uz OpenCV biblioteku, Keras API i Tensorflow. Korišten je sekvencijalni model Keras-a na kojeg su zatim dodavani slojevi konvolucijske neuronske mreže. Za treniranje i testiranje programskog rješenja korištena je baza fotografija GTSRB, koja se dijeli na dio za treniranje i dio za testiranje, a korišteni su i vlastiti videozapisi.

Pokretanjem aplikacije i njenim analiziranjem uočeno je kako je prepoznavanje uspješno provedeno, ali kako se problem javlja onda kada učitane fotografije nisu kao one iz dijela za treniranje nego kada osim prometnog znaka sadrže i druge elemente okoline. Ti elementi okoline imaju utjecaj na točnost prepoznavanja. Zato se točnost prepoznavanja smanjivala prilikom učitavanja spremljenih okvira videozapisa te je iz tog razloga preporučeno spremljene okvire videozapisa najprije obraditi tako da se izostave svi elementi okoline traženog prometnog znaka. Ako se tako obrađene fotografije učitaju, prepoznavanje traženog objekta će biti uspješno odrađeno. Postignuta točnost prepoznavanja prometnog znaka iz videozapisa od 54,7% nije ni približno dovoljna za praktičnu primjenu, ali uspješnost na tesnim slikama od 98 % ukazuje da sam algoritam dobro funkcionira, ali je potrebna dodatna predobrada okvira videozapisa.

LITERATURA

[1] Y. Huang, Y. Lee, Detection and recognition of speed limit signs, 2010 International Computer Symposium (ICS2010), str. 107-112, Tainan, 2010

[2] J. Miura, T. Kanda, Y. Shirai, An active vision system for real-time traffic sign recognition, in, 2000 IEEE Intelligent Transportation Systems, str. 52-57, Dearborn, MI, USA, 2000

[3] Y. Jiang, S. Zhou, Y. Jiang, J. Gong, G. Xiong, H. Chen, Traffic sign recognition using Ridge Regression and OTSU method, 2011 IEEE Intelligent Vehicles Symposium (IV), str. 613 – 618, Baden-Baden, Germany, 2011

[4] H. Fleyeh, Traffic sign recognition without color information, 2015 Colour and Visual Computing Symposium (CVCS), str. 1-6, Gjovik, 2015

[5] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, str. 886-893, 2005

[6] H. Ngoc Do, M. Vo, H. Quoc Luong, A. Hoang Nguyen, K. Trang and L. T. K. Vu, Speed limit traffic sign detection and recognition based on support vector machines, 2017 International Conference on Advanced Technologies for Communications (ATC), str. 274-278, Quy Nhon, 2017

[7] D. Agudo, A. Sanchez, J. F. Velez, and A. B. Moreno, Real-time railway speed limit sign recognition from video sequences, International Conference on System, Signal and Image Processing-IWWSSIP, str. 1–4, 2016

[8] A. Mammeri, A.Boukerche, J.Feng, and R.Wang, North-american speed limit sign detection and recognition for smart cars, IEEE International Workshop on Performance and Management of Wireless and Mobile Network, str. 154–161, 2013

[9] Y. Wang, M. Shi and T. Wu, A Method of Fast and Robust for Traffic Sign Recognition, 2009 Fifth International Conference on Image and Graphics, str. 891-895, Xi'an, Shanxi, 2009

[10] Sin-Yu Chen; Jun-Wei Hsieh, 2008, Boosted road sign detection and recognition, IEEE International Conference on Machine Learning and Cybernetics, str.3823-3826

[11] M. Vranješ, Digitalna obrada slike i videa za autonomna vozila: Detekcija lica, pješaka i automobila,
(https://loomen.carnet.hr/pluginfile.php/1576711/mod_resource/content/7/06%20Detekcija%20lica%20pje%C5%A1aka%20i%20automobila.pdf), pristup ostvaren 03.09.2020.

[12] Banjanović-Mehmedović L., Izabrana poglavlja inteligentnih sistema: Metode klasifikacije, (http://lejla-bm.com.ba/IPIS/IPIS_2_Metode_klasifikacije.pdf), pristup ostvaren 07.06.2020.

[13] T. Šmuc, Strojno učenje: Metoda potpornih vektora, Zagreb, 2013

[14] Hough, P.V.C., Method and means for recognizing complex patterns, Technical report, U.S. Patent No. 3069654., 1962.

[15] H. Huang and L. Hou, Speed Limit Sign Detection Based on Gaussian Color Model and Template Matching, 2017 International Conference on Vision, Image and Signal Processing (ICVISP), str. 118-1222, Osaka, 2017

[16] R. Biswas, H. Fleyeh and M. Mostakim, Detection and classification of speed limit traffic signs, 2014 World Congress on Computer Applications and Information Systems (WCCAIS), str. 1-6, Hammamet, 2014

[17] S. K. Nair and A. R.P., Recognition of Speed Limit from Traffic Signs Using Naive Bayes Classifier, 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), str. 1-6, Kottayam, India, 2018

[18] Y. Yang and F. Wu, Real-time traffic sign detection via color probability model and integral channel features, Communications in Computer and Information Science, vol. 484, str. 545–554, 2014

[19] Python Project on Traffic Signs Recognition with 95% Accuracy using CNN & Keras, (https://data-flair.training/blogs/python-project-traffic-signs-recognition/), pristup ostvaren 10.07.2020.

[20] S. Albawi, T. A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, International Conference on Engineering and Technology (ICET), str. 1-6, 2017

[21] I.Goodfellow, Y.Bengio, A.Courville, Deep Learning, MIT Press, 2016

[22] A. Gulli, S. Pal, Deep learning with Keras, Packt Publishing, Birmingham – Mumbai, 2017.

SAŽETAK

U okviru ovog završnog rada izrađena je aplikacije za prepoznavanje prometnih znakova ograničenja brzine na učitanoj slici primjenom konvolucijske neuronske mreže. Za razvoj aplikacije korišteni su Keras te OpenCV. Za treniranje i testiranje aplikacije korištene su fotografije iz GTSRB baze podataka te vlastiti videozapis iz kojega su okviri spremljeni u zasebnu mapu. Kod detekcije znakova iz videozapisa pokazalo se kako elementi okoline prometnog znaka imaju znatan utjecaj na točnost njegova prepoznavanja. Također, utvrđeno je kako točnost prepoznavanja uvelike ovisi i o udaljenosti znaka od kamere. Unatoč tome, testiranje korištene metode je pokazalo kako je prepoznavanje znaka u većini slučajeva uspješno obavljeno i da je trening nad slikama koje čine bazu podataka odrađen uspješno.

Ključne riječi: detekcija objekata, Keras, neuronske mreže, OpenCV, prepoznavanje znaka

DETECTION OF SPEED LIMIT TRAFFIC SIGNS FROM ON-VEHICLE CAMERA IMAGE

ABSTRACT

An application for recognition of the speed limit road sign on the loaded photo is made within this bachelor's thesis. Keras and OpenCV are used for the application development. Photos from GTSRB database, as well as own videos from which video frames are extracted and saved into a folder, are utilized for training and testing the application. Detecting the signs has shown that environmental elements have a very strong influence on accuracy of the sign recognizing. Also, it has been established that the accuracy depends on distance between car camera and the traffic sign. Nevertheless, testing of the presented method showed that the sign recognition was well done and that training, performed on images which make the database, was successfully carried out.

Key words: object detection, Keras, neural networks, OpenCV, sign recognition

ŽIVOTOPIS

Anastazija Živković rođena je 3. studenog 1998. u Vinkovcima. Dolazi iz Kostrča kraj Orašja, BiH, a u Tolisi završava Osnovnu školu fra Ilije Starčevića, gdje je proglašena učenicom generacije te nakon završetka osnovne škole pohađa Franjevačku klasičnu gimnaziju u Visokom, koju završava s odličnim uspjehom. Tijekom školovanja sudjelovala je i osvajala visoka mjesta na natjecanjima iz matematike, fizike, stranih jezika i glazbe te na međunarodnim natjecanjima iz klasičnih jezika. Sudjelovala je i na srednjoškolskim razmjenama učenika u Austriji i Njemačkoj. U sklopu srednjoškolskog obrazovanja 2017. godine stječe Njemačku jezičnu diplomu razine C1.

Trenutno je studentica završne godine preddiplomskog studija.

PRILOG

#Glavni program “prometni_znakovi.py”

```
# Kako bi se mogla stvoriti polja (eng. arrays), potrebno je osigurati set numeričkih podatkovnih tipova
import numpy as np

# Biblioteka funkcija koje služe za razvoj računalnog vida
import cv2

# Biblioteka koja služi za strojno učenje
import tensorflow as tf

# Zbirka naredbenih funkcija pomoću kojih
# matlotlib radi jednako kao MATLAB
import matplotlib.pyplot as plt

# Softverska knjižnica koja služi za analizu i manipulaciju podacima
import pandas as pd

# PIL (eng. Python Imaging Library), služi za otvaranje,
# spremanje, i manipulacija fotografija raznih formata
from PIL import Image

# Omogućuje funkcije za interakciju s operacijskim sustavom
import os

# Omogućuje razdvajanje podataka u jednoj liniji
from sklearn.model_selection import train_test_split

# Omogućuje izračunavanje točnosti podseta podataka
from sklearn.metrics import accuracy_score
```

```
# Iz slojeva kerasa uvoze se slojevi potrebni za izgradnju modela
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D

# Sekvencijalni model je početni model Keras-a na kojeg
# se nadovezuju svi ostali slojevi neuronske mreže
from keras.models import Sequential, load_model

# Pretvara polje “labels” podataka u vector koji će biti spreman za daljnju #obradu
from keras.utils import to_categorical

# Stvaranje polja “data”
data = []

# Stvaranje polja “labels”
labels = []

# 9 klasa:
klase = 9

# Trenutna putanja gdje se nalazi mapa
trenutni_put = os.getcwd()

# Za trenutnu putanju otvara svaku pojedinu klasu koja se nalazi
# u mapi za treniranje
for i in range(klase):
    path = os.path.join(trenutni_put, 'train', str(i))
    slike = os.listdir(path)

    # Za trenutnu klasu otvara svaku pojedinu fotografiju koja se nalazi
    # u mapi te klase
    for a in slike:
        try:
```

```
            # Otvara sliku
            image = Image.open(path + '\\' + a)

            # Dimenzije slike pretvara u 30 X 30
            image = image.resize((30, 30))

            # Uvozi podatke slike u numeričko polje (eng. NumPy)
            image = np.array(image)

            # Ista slika dodaje se polju “data” tako da se svaka slika
            # dodaje pojedinačno i sprema na kraj polja povećavajući
            # veličinu polja za 1
            data.append(image)

            # Klasu dodaje polju “labels”
            labels.append(i)
        # Ako se slika ne može učitati, šalje se poruka o pogrešci
        except:
            print("Došlo je do problema tijekom učitavanja.")

data = np.array(data)
labels = np.array(labels)

# Ispisuje definiciju polja i povezanih meta podataka
# za polje “data” i polje “labels”
print(data.shape, labels.shape)

#Razdvajanje seta podataka za treniranje i za testiranje
#iz mape "Train"
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=8)

# Ispisuje definiciju i povezane meta podatke
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
# "One-hot encoding" - pretvaranje u numeričke podatke
y_train = to_categorical(y_train, 9)
y_test = to_categorical(y_test, 9)

# Model:
# Sekvencijalni model:
model = Sequential()

# 5 x 5 konvolucija primjenjuje na X_train.shape, a rezultat su 32 izlazna kanala (32 filtera),
# primjenjuje se aktivacijska funkcija ReLU
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation='relu', input_shape=X_train.shape[1:]));
# Još jedan sloj 5 x 5 konvolucije s 32 izlazna kanala uz ReLU
model.add(Conv2D(filters=32, kernel_size=(5, 5), activation='relu'));

# Max-pooling sloj veličine 2 x 2,
# max-pooling sažima sve neurone s prethodnog sloja u jedan neuron na #sljedećem sloju uzimajući najveću vrijednost
model.add(MaxPool2D(pool_size=(2, 2)));

# Dropout je sloj koji se primjenjuje kako bi se točnost značajno povisila,
# a pogreška znatno smanjila. Dropout nasumično odbacuje pojedine neurone.
model.add(Dropout(rate=0.25));

# 5 x 5 konvolucija, a rezultat su 64 izlazna kanala (64 filtera)
# primjenjuje se aktivacijska funkcija ReLU
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'));

# Još jedan sloj 5 x 5 konvolucije sa 64 izlazna kanala
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'));

# Max-pooling sloj veličine 2 x 2
```

```
model.add(MaxPool2D(pool_size=(2, 2)));

# Dropout sloj s 25% vjerojatnosti, što znači da će
# 1 od 5 neurona biti izbačen
model.add(Dropout(rate=0.25));

# Flatten sloj spajva dimenzije unešenih podataka u dimenzije kanala
model.add(Flatten());

# Sloj neurona u neuronskoj mreži na kojeg se vežu svi neuroni
# prethodnog sloja
model.add(Dense(256, activation='relu'));

# Dropout sloj s 50% vjerojatnosti, što znači da će
# 1 od 2 neurona biti izbačen
model.add(Dropout(rate=0.5));

# U sloju se nalazi 9 neurona u prvom skrivenom (eng. hidden) sloju
model.add(Dense(9, activation='softmax'));

# Kompilacija modela s “Adam” optimizerom
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']);

# 20 epoha
epohe = 20;

# Batch_size definira broj uzoraka koji se uzima za treniranje mreže
# dok se ne istreniraju svi uzorci. Na taj se način štedi memorija.
history = model.fit(X_train, y_train, batch_size=32, epochs=epohe, validation_data=(X_test,
y_test));

# Model se sprema u datoteku “road_signs.h5”
model.save("znakovi_brzine.h5");
```

```
# Stvaranje prvog grafa
plt.figure(0);

# Jedna od dviju linija na prvom grafu s
# naslovom “Točnost seta podataka za treniranje”
plt.plot(history.history['accuracy'], label='Točnost seta podataka za treniranje');

# Druga od dvije linije na prvom grafu s
# naslovom “Utvrđena točnost”
plt.plot(history.history['val_accuracy'], label='Točnost testiranog seta')

# Naslov grafa
plt.title('Točnost')

# Naslov apscise
plt.xlabel('epohe')

# Naslov ordinate
plt.ylabel('točnost')

# Dodavanje legende
plt.legend()

# Prikaz grafa
plt.show()

# Stvaranje drugog grafa
plt.figure(1)

# Jedna od dviju linija na drugom grafu s
# naslovom “Pogreška seta podataka za treniranje”
plt.plot(history.history['loss'], label='Pogreška seta podataka za treniranje')
```

```
# Druga od dvije linije na drugom grafu s
# naslovom “Utvrđena pogreška”
plt.plot(history.history['val_loss'], label='Pogreška testiranog seta')

# Naslov grafa
plt.title('Pogreška')

# Naslov apscise
plt.xlabel('epohe')

# Naslov ordinate
plt.ylabel('pogreška')

# Dodavanje legende
plt.legend()

# Prikaz grafa
plt.show()

# testiranje točnosti na setu podataka za testiranje

# read_csv je važna “pandas” funkcija koja
# čita csv datoteke
# Baza podataka korištena u ovom radu sadrži datoteku “Test.csv” koja sadrži
# podatke o putanjama slika i naslovima klasa
y_test = pd.read_csv('Test.csv')

# Izdvajaju se podaci o naslovima
labels = y_test["ClassId"].values

# Izdvajaju se podaci o putanjama slika
imgs = y_test["Path"].values
```

```
# Stvara se prazno polje
data = []

# Kako bi se predvidio model, potrebno je:
for img in imgs:
    # otvoriti sliku,
    image = Image.open(img)
    # prepraviti veličinu slike,
    image = image.resize((30, 30))
    # uvesti podatke slike u numeričko polje,
    image = np.array(image)
    # sliku dodati polju “data”
    data.append(image)

# Podaci “data” polja uvode se u numeričko polje
X_test = np.array(data)

# Model za predviđanje
pred = model.predict_classes(X_test)

# Točnost seta podataka za testiranje, spremanje modela
# Pokazuje se kolika je točnost
print(accuracy_score(labels, pred))

# Model se sprema pod nazivom “klasifikator_znaka.h5”
model.save("klasifikator_znaka.h5")

#Program za korisničko sučelje “sucelje.py”

#Tkinter je standardno korisničko sučelje Pythona
import tkinter as tk, threading
#omogućava otvaranje datoteka
```

```
from tkinter import filedialog
#uvozi svaki objekt iz Tkinter-a u trenutni imenik
from tkinter import *
# PIL (eng. Python Imaging Library), služi za otvaranje,
# spremanje, i manipulacija fotografija raznih formata
from PIL import ImageTk, Image
# Kako bi se mogla stvoriti polja (eng. arrays), potrebno je osigurati set numeričkih podatkovnih tipova
import numpy
# Biblioteka funkcija koje služe za razvoj računalnog vida
import cv2
# Omogućuje funkcije za interakciju s operacijskim sustavom
import os


#učitavanje modela treniranog za klasificiranje znaka
from keras.models import load_model
#učitavanje modela pod imenom #my_model.h5"
model = load_model('klasifikator_znaka.h5')
#klase svakog pojedinog prometnog znaka

klase = { 1:'Ograničenje brzine (20km/h)',
          2:'Ograničenje brzine (30km/h)',
          3:'Ograničenje brzine (50km/h)',
          4:'Ograničenje brzine (60km/h)',
          5:'Ograničenje brzine (70km/h)',
          6:'Ograničenje brzine (80km/h)',
          7:'Prestanak ograničenja brzine (80km/h)',
          8:'Ograničenje brzine (100km/h)',
          9:'Ograničenje brzine (120km/h)',
          }
#inicijalizacija GUI
#korištenje Tkinter-a za korisničko sučelje
```

```
top=tk.Tk()
#prozor veličine 800 x 600
top.geometry('800x600')
#naslov
top.title('Prepoznavanje prometnih znakova')
#boja pozadine
top.configure(bg='#4295f5')
label=Label(top,bg='#D3D3D3', font=('times',17,'italic bold'))
label1=Label(top,bg='#D3D3D3', font=('times',17,'italic bold'))
sign_image = Label(top)
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    pred = model.predict_classes([image])[0]
    sign = klase[pred+1]
    print(sign)
    label.configure(foreground='blue', text=sign)

def show_classify_button(file_path):
    classify_b=Button(top,text="Prepoznaj prometni znak",command=lambda:
classify(file_path),padx=10,pady=5)
    classify_b.configure(bg='white', fg='blue',font=('Times',12,'italic bold'))
    classify_b.place(relx=0.75,rely=0.85)

def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)
```

```
            sign_image.configure(image=im)
            sign_image.image=im
            label.configure(text='')
            show_classify_button(file_path)
        except:
            pass
    def upload_video():
        # Učitava video:
        myvideo = cv2.VideoCapture('5 - Grad.mp4')

        try:
            file_path = filedialog.askopenfilename()
            uploaded = myvideo.open(file_path)
            uploaded.thumbnail(((top.winfo_width() / 2.25), (top.winfo_height() / 2.25)))
            im = ImageTk.PhotoImage(uploaded)
            sign_image.configure(image=im)
            sign_image.image = im
            label.configure(text='')
            show_classify_button(file_path)

        except:
            pass
        try:
            # stvaranje mape Frames ako ista nije ranije stvorena
            if not os.path.exists("frames"):
                directory = "frames"
                parent = "/Users/User/PycharmProjects/openCvPython"
                path = os.path.join(parent, directory)
                #stvaranje direktorija
                os.makedirs(path)
                #ako direktorij nije stvoren, javlja se obavijest:
        except:
            print('Pojavila se pogreška pri stvaranju mape "frames".')
```

```
 #brojač okvira
   frame_counter = 0

   while (1):

#učitavanje iz okvira
      success, frame = myvideo.read()

      if success:
         # dok video traje, sprema okvire u mapu Frames
         name = './frames/frame' + str(frame_counter) + '.jpg'
         print('Učitava se' + name)
         #svaki frame spremi pod putanju "name" (u datoteku frames)
         cv2.imwrite(name, frame)
         cv2.imshow("Okvir", frame)



         # brojač koji pokazuje koliko je okvira stvoreno; brojač se povećava nakon svakog stvorenog
okvira
         frame_counter += 1

         #što je veći broj unutar "cv2.waitKey", to je videozapis sporiji
         #ako se pritisne tipka "q", zatvara se prozor "Okvir" i završava
         #spremanje okvira
         if cv2.waitKey(10) & 0xFF == ord('q'):
            myvideo.release()
            cv2.destroyAllWindows()
      else:
            myvideo.release()
            cv2.destroyAllWindows()
            break
```

```
def classify_video():
    # Učitava video:
    myvideo = cv2.VideoCapture('5 - Grad.mp4')

    try:
        file_path = filedialog.askopenfilename()
        uploaded = myvideo.open(file_path)
        uploaded.thumbnail(((top.winfo_width() / 2.25), (top.winfo_height() / 2.25)))
        im = ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image = im
        label.configure(text='')
        show_classify_button(file_path)

    except:
        pass
    try:
        # stvaranje mape Frames ako ista nije ranije stvorena
        if not os.path.exists("frames"):
            directory = "frames"
            parent = "/Users/User/PycharmProjects/openCvPython"
            path = os.path.join(parent, directory)
            # stvaranje direktorija
            os.makedirs(path)
            # ako direktorij nije stvoren, javlja se obavijest:
    except:
        print('Pojavila se pogreška pri stvaranju mape "frames".')

    # brojač okvira
    frame_counter = 0

    while (1):
```

```
        # učitavanje iz okvira
        success, frame = myvideo.read()

        if success:
            # dok video traje, sprema okvire u mapu Frames
            name = './frames/frame' + str(frame_counter) + '.jpg'
            print('Učitava se' + name)
            # svaki frame sprema se pod putanju "name" (u datoteku frames)
            cv2.imwrite(name, frame)
            cv2.imshow("Okvir", frame)


        # brojač koji pokazuje koliko je okvira stvoreno; brojač se povećava nakon svakog stvorenog okvira
        frame_counter += 1
        točno = 0
        pogrešno = 0
        if frame_counter == 201:
            myvideo.release()
            cv2.destroyAllWindows()
            trenutni_put = os.getcwd()
            path = os.path.join(trenutni_put, 'frames')
            for a in range(201):

                try:
                    image = Image.open(path + '\\' + 'frame' + str(a) + '.jpg')

                    # Dimenzije slike pretvara u 30 X 30
                    image = image.resize((30, 30))
                    image = numpy.expand_dims(image, axis=0)
                    image = numpy.array(image)
                    pred = model.predict_classes([image])[0]
                    sign = klase[pred + 1]
```

```
                print(sign)

                if file_path == "C:/Users/User/PycharmProjects/openCvPython/7 -
Autocesta.mp4":
                    f1 = "Ograničenje brziine (100km/h)"
                    if f1 == sign:
                        točno += 1
                    else:
                        pogrešno += 1
                else:
                    if file_path == 'C:/Users/User/PycharmProjects/openCvPython/Grad.mp4':
                        f1 = "Ograničenje brzine (30km/h)"
                        if f1 == sign:
                            točno += 1
                        else:
                            pogrešno += 1
                    else:
                        if file_path == 'C:/Users/User/PycharmProjects/openCvPython/fc.mp4':
                            f1 = "Ograničenje brzine (30km/h)"
                            if f1 == sign:
                                točno += 1
                            else:
                                pogrešno += 1
                        else:
                            pass

            except:
                pass
        if a == 200:
            print(str(točno)+" točno prepoznatih prometnih znakova.")
            print(str(pogrešno)+" pogrešno prepoznatih prometnih znakova.")
            postotak_točnih = (točno/(a+1))
            postotak_pogrešnih = (pogrešno/(a+1))
```

```
                b = ("Postotak točnosti iznosi {:.1%}.".format(postotak_točnih))
                c = ("Postotak pogreške iznosi {:.1%}.".format(postotak_pogrešnih))
                label.configure(fg='blue', text = b)
                label1.configure(fg='blue', text=c)


            else:
                pass
        # što je veći broj unutar "cv2.waitKey", to je videozapis sporiji
        # ako se pritisne tipka "q", zatvara se prozor "Okvir" i završava
        # spremanje okvira
        if cv2.waitKey(10) & 0xFF == ord('q'):
            myvideo.release()
            cv2.destroyAllWindows()
    else:
        myvideo.release()
        cv2.destroyAllWindows()
        break


class0=Button(top,text="Učitaj video i provjeri točnost", command=classify_video, padx=5, pady=50)
class0.configure(bg='white', fg='blue',font=('arial',11,'bold italic'))
class0.pack(side=RIGHT,pady=55)
#Stvaranje gumba za opciju #Učitaj video"
upload1=Button(top,text="Učitaj video",command=upload_video,padx=21,pady=13)
upload1.configure(bg='white', fg='blue',font=('arial',11,'bold italic'))
upload1.pack(side=LEFT,pady=55)
#Stvaranje gumba za opciju #Učitaj sliku"
upload=Button(top,text="Učitaj sliku",command=upload_image,padx=10,pady=5)
upload.configure(bg='white', fg='blue',font=('arial',11,'bold italic'))
upload.pack(side=BOTTOM,pady=55)
sign_image.pack(side=BOTTOM,expand=True)
```

```
label.pack(side=BOTTOM,expand=True)
label1.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Prepoznavanje prometnog znaka",pady=50, font=('Times',23,'bold italic'))
heading.configure(bg='white',fg='blue')
heading.pack()
top.mainloop()
```