Sustav za automatsko upravljanje vozilom u ovisnosti o stanju na cesti

Svetinović, Borna

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

Sveučilišni studij

SUSTAV ZA AUTOMATSKO UPRAVLJANJE VOZILOM U OVISNOSTI O STANJU NA CESTI

Diplomski rad

Borna Svetinović

Osijek, 2023.

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 01.12.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Borna Svetinović
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. Pristupnika, godina upisa:	D-1381, 07.10.2021.
OIB studenta:	12190932428
Mentor:	prof. dr. sc. Marijan Herceg
Sumentor:	,
Sumentor iz tvrtke:	Zvonimir Kaprocki
Predsjednik Povjerenstva:	prof. dr. sc. Mario Vranješ
Član Povjerenstva 1:	prof. dr. sc. Marijan Herceg
Član Povjerenstva 2:	izv. prof. dr. sc. Ratko Grbić
Naslov diplomskog rada:	Sustav za automatsko upravljanje vozilom u ovisnosti o stanju na cesti
Znanstvena grana diplomskog rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	Potpuno autonomna vozila su vozila koja se mogu kretati u potpunosti samostalno bez ljudskog upravljana. Takva vozila, korištenjem različitih senzora, mogu raspoznati objekte u svojoj okolini i na temelju dobivenih informacija poduzeti određenu radnju (kočenje, skretanja, ubrzanje, itd). Jedan od najčešće korištenih senzora za detekciju objekta u okolini vozila je kamera. U okviru ovog diplomskog rada potrebno je izraditi algoritam zasnovan na nekoj od postojećih neuronskih mreža (npr. PilotNet) koji će na temelju slike s kamere na vozilu upravljati vozilom tijekom kretanja po unaprijed definiranom putu od točke A do točke B. Algoritam treba biti treniran i testiran na slikama dobivenim iz simulatora (npr. Carla simulator). Ulaz u neuronsku
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene od strane mentora:	01.12.2023.

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

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 13.12.2023.

Ime i prezime studenta:	Borna Svetinović
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1381, 07.10.2021.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Sustav za automatsko upravljanje vozilom u ovisnosti o stanju na cesti**

izrađen pod vodstvom mentora prof. dr. sc. Marijan Herceg

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

U eri obilježenoj brzim tehnološkim napretkom, automobilska industrija predvodi u razvoju inovacija automobilskih sustava. Razvoj tehnologija poput računalnog vida (engl. *computer vision*), strojnog učenja (engl. *machine learning*), dubokog učenja (engl. *deep learning*) i sl. omogućuje razvoj inovativnih tehnika za unapređivanje automobilskih sustava. Primjer primjene tih tehnologija jest pojava i razvoj sustava za podršku vozaču pri upravljanju vozilom (engl. *Advanced Driver Assistance Systems* - ADAS) i sustava za autonomnu vožnju. Bez obzira na sve tehnološke napretke, broj žrtava u prometnim nesrećama je i dalje velik. Prema podacima Svjetske zdravstvene organizacije, 2016. godine je bilo 1.35 milijuna prometnih žrtava [1]. Napredniji i sigurniji ADAS sustavi i sustavi autonomne vožnje bi mogli znatno smanjiti utjecaj ljudske pogreške na prometne nesreće.

Izgradnja potpuno autonomnih i sigurnih vozila je, za sada, nemoguća, zbog izuzetno puno situacija i okolnosti u kojima se može naći automobil te zbog raznolikih prirodnih uvjeta koji utječu na vidljivost i na stanje ceste. Autonomnim vozilima je potreban veliki broj podataka kako bi mogli uspješno izvršiti određenu radnju. Podaci igraju ključnu ulogu u učenju i donošenju odluka te osiguravaju pouzdanost i učinkovitost tih sustava. Kako bi se dobili potrebni podaci, na automobilima se postavljaju senzori, odnosno oprema kao što su video kamere, LIDAR-i (engl. *LIght Detection And Ranging*), RADAR-i (engl. *RAdio Detection And Ranging*) i GPS (engl. *Global Positioning System*). Prikupljeni podaci šalju se električnoj kontrolnoj jedinici (engl. *electronic control unit*, ECU) koja s dobivenim podacima i implementiranim sustavima može rješavati sigurnosne probleme kao što je detekcija automobila, pješaka i znakova ispred automobila, sprječavanje napuštanja automobila iz prometnih traka i sl. Za realizaciju sustava koji rješavaju navedene sigurnosne probleme potrebna je velika količina podataka. Prikupljanje realnih podataka je vrlo dugotrajan i skup postupak, stoga se za dodatno prikupljanje podataka koriste simulatori. Oni omogućuju efikasno i kontrolirano prikupljanje podataka. CARLA (engl. *CAR Learning to Act*) simulator je jedan od popularnijih simulatora koji omogućava prikupljanje raznolikih podataka od senzora.

Cilj ovog rada je implementirati i testirati autonomni sustav za procjenu brzine i kuta zakreta volana vozila na temelju slike stanja na cesti ispred vozila. Procjenu će odrađivati neuronska mreža (engl. *neural network*) zasnovana na dubokom učenju. Ulazni podaci za neuronsku mrežu su slike iz video kamere koje prikazuju stanje na cesti ispred automobila. Izlazni podaci su kut zakreta

volana u rasponu od -1 do 1 kao i brzina izražena u m/s. U drugom poglavlju ovog rada je objašnjen problem predviđanja kuta zakreta volana i kojim metodama se može riješiti isti problem. U trećem poglavlju opisani su korišteni alati za kreiranje vlastitog sustava, proces prikupljanja i kreiranja skupa podataka te je prikazan skup podataka dobiven iz CARLA simulatora, opis korištenog modela te proces treniranja predloženog modela. U četvrtom poglavlju je objašnjeno testiranje dobivenog sustava u simulatoru te su prikazani rezultati testiranja vozila na do sada neviđenim podacima. Posljednje poglavlje je zaključak cjelokupnog rada.

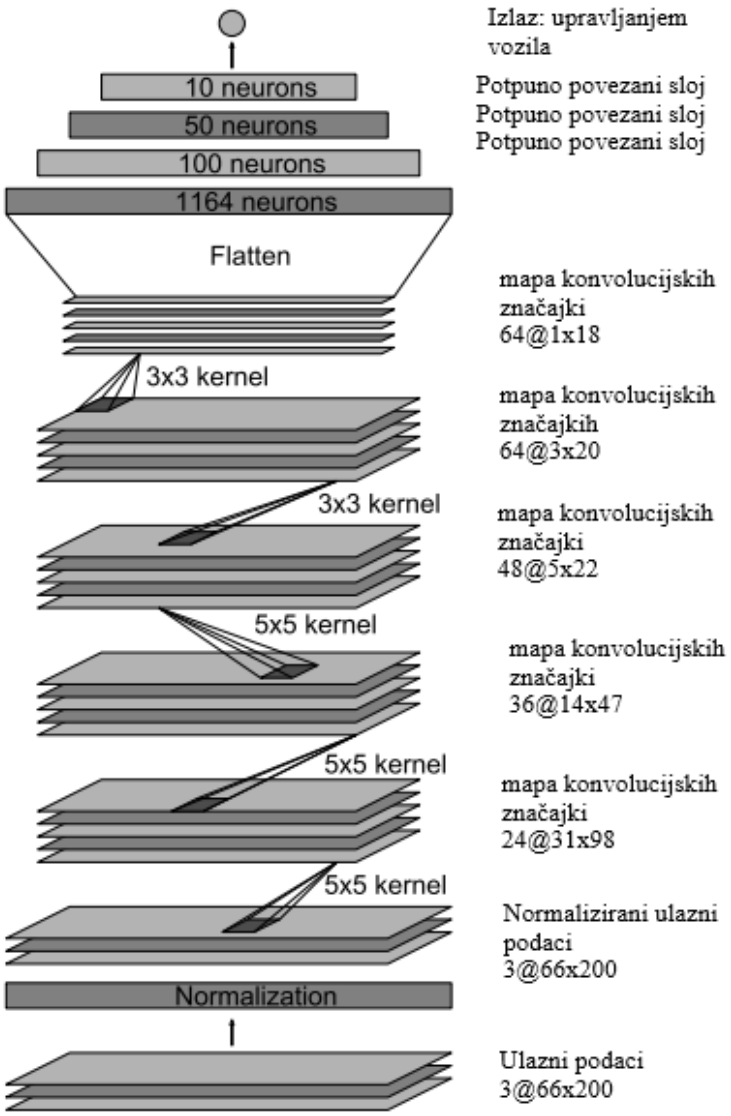
2. PROBLEM AUTOMATSKOG UPRAVLJANJA VOZILOM U OVISNOSTI O STANJU NA CESTI

Potpuno autonomno vozilo podrazumijeva vozilo koje se može kretati prometom bez ljudskog nadzora. Kako bi se postigla autonomnost vozila, ono treba imati implementirane različite sustave koje imitiraju ljudski um i obavljaju određene zadatke poput čovjeka. Postoje različite tehnologije kojima se mogu napraviti takvi sustavi, kao što su računalni vid, duboko učenje, podržano učenje (engl. *reinforcement learning*) i sl. Korištenjem navedenih tehnologija mogu se kreirati algoritmi za predviđanje zakreta volana automobila, kao i adaptivni tempomat (engl. *Adaptive cruise control*). Algoritmi za predviđanje kuta skretanja automobila se dijele na algoritme temeljene na računalnoj viziji i algoritme temeljene na neuronskim mrežama. U ovome poglavlju će se prikazati radovi napravljeni već spomenutom, tehnologijom.

Model ALVINN (engl. *An Autonomous Land Vehicle in an Neural Network*) je jedan od prvih autonomnih sustava koji je objavljen još 1989. godine. On je dizajniran kako bi se mogao koristiti u specifičnom vozilu NAVLAB (engl. *Navigation and autonomous vehicle lab*), koje je napravljeno za testiranje autonomne navigacije. Model je napravljen od tri sloja koji se temelje na povratnom širenju (engl. *Back Propagation*). Sastoji se od jednog ulaza i jednog izlaza. Ulaz u mrežu su podaci dobiveni od kamere i laserskog daljinometra. Ulaz se dijeli na tri dijela: dva dijela simuliraju ljudske mrežnice, a treći predstavlja jednu ćeliju povratne informacije intenziteta svjetline ceste u odnosu na svjetlinu prostora pored ceste. Za treniranje mreže korišteni su umjetno napravljeni prikazi ceste. Model je treniran na 1200 umjetno napravljenih slika cesta kojima je naknadno promijenjena svjetlina ceste na slici. Nakon 40 epoha treniranja model je uspješno mogao procijeniti smjer kretanja vozila u 90 % slučajeva [2].

Konvolucijske neuronske mreže (engl. *Convolutional Neural Network*, CNN) se uglavnom koriste za obradu prostornih informacija, za izdvajanje određenih značajki (engl. *features*) te kao univerzalni aproksimatori nelinearnih funkcija. Prije pojave dubokog učenja, koristili su se sustavi računalnog vida na temelju ručno izrađenih značajki. Neki od tih sustava su: *Viola&Jones* sustav, sustavi lokalnih binarnih uzoraka i sustavi histogramsko usmjerenih gradijenata (engl. *histogram of oriented gradients*, *Hog*). Za razliku od prijašnje spomenutih sustava, CNN algoritmi su sposobni automatski naučiti prikaz značajke prostora u trening skupu podataka [3]. Jedan od najznačajnijih CNN algoritama za autonomnu vožnju je *Nvidia*-in s kraja na kraj (engl. *end to end*) algoritam *PilotNET*. Podaci za treniranje modela su prikupljeni vožnjom automobila na autocesti

kroz nekoliko nezavisnih američkih država. Predloženi model sastoji se od jednog normalizacijskog sloja, pet konvolucijskih slojeva te se dobivene značajke predaju u tri potpuno povezana sloja. Na slici 2.1. prikazana je arhitektura *PilotNET*-a. Prikazani model sadrži oko 250



Slika 2.1. Prilaz arhitekture *PilotNET* mreže [4]

tisuća parametara. Prije treniranja, postojeće slike su augmentirane sa zrcaljenjem, ali i pomicanjem perspektive slika na cestu kako bi se vozilo znalo oporaviti od pogreške. Vrijednosti kuta zakreta volana koje su dolazile sa slikama su također postavljene na adekvatnu vrijednost. Evaluacija dobivenog modela je odrađena u dva koraka, prvo u simulaciji pa na realnim cestama. Razina autonomije je računata po formuli:

$$autonomija = \left(1 - \frac{(broj\ intervencija) \times 6\ [s]}{proteklo\ vrijeme\ [s]}\right) \times 100. \qquad (2\text{-}1)$$

U navedenoj formuli, *broj intervencija* označava broj ljudskih intervencija kako se ne bi dogodila pogreška. Šest sekundi označava prosječno vrijeme trajanja ljudske intervencije. *Proteklo vrijeme* je ukupno vrijeme izvođenja testa u sekundama. Rezultati evaluacije u simulatoru su pokazali 90 % autonomije, dok je test u realnim uvjetima pokazao da model ima 98 % autonomije [4]. U idućem radu [5] predložen je CNN model koji bi se implementirao u predloženog robota. Robot bi se testirao u eksperimentalnom okruženju kako bi se utvrdila razina autonomnosti. Nakon toga robot je testiran u realnim uvjetima. Predloženi model sastoji se od normalizacijskog sloja, četiri konvolucijska sloja, tri potpuno povezana sloja te se na kraju nalazi izlaz koji predstavlja kut zakreta volana vozila. Skup podataka, na kojoj je treniran navedeni model, je preuzet od *Udacityjevog gitHub* profila. Dodane su augmentacije kuta gledanja i pomicanje slika kako bi se robot znao oporaviti od pogreške. Izdvojeno je 80 % slika u trening skup i 20 % slika u validacijski skup. Nakon treniranja, model je izračunao gubitak (engl. *loss*) od 0.00038 nakon 30 epoha. Model je testiran u CARLA simulatoru te je dobivena razina autonomnosti od 86.43 % prema formuli (2 – 1).

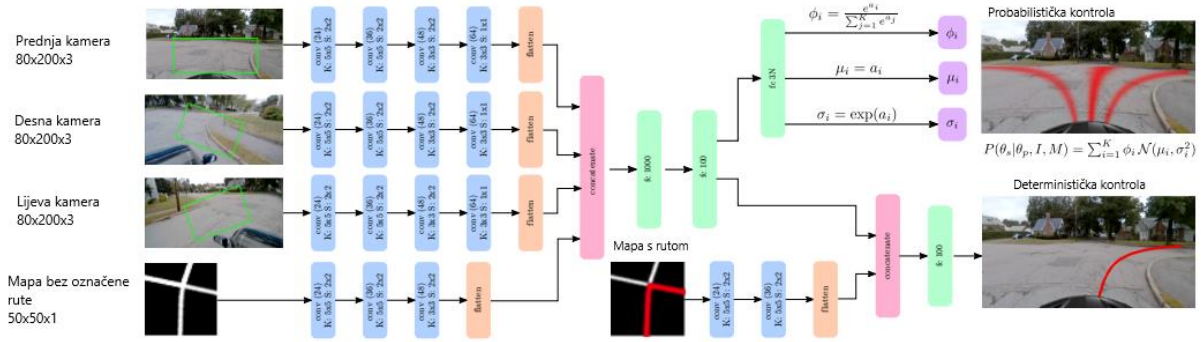
Rekurentna neuronska mreža (engl. *Recurrent Neural Network*, RNN) je metoda dubokog učenja koja je posebno dobra u obradi podataka prikazanih u vremenskim sekvencama kao što su videozapisi. Ona je različita od ostalih metoda dubokog učenja jer sadrži vremenski ovisnu povratnu petlju u svojim memorijskim ćelijama. Glavni izazov treniranja RNN mreža je nestajanje gradijenta koji se javlja tijekom treniranja mreže. Zbog tog problema obični RNN algoritmi se ne koriste za treniranje dugotrajnih ovisnosti u dugačkoj podatkovnoj sekvenci. Jedna od najkorištenijih RNN-ova jest LSTM (engl. *Long-Short-Term memory*) mreža. To je nelinearni funkcijski aproksimator za prepoznavanje vremenskih ovisnosti u podatkovnom nizu. LSTM rješava problem nestajanja gradijenta tako da uključuje tri ulaza koji kontroliraju ulaz i izlaz iz mreže te memorijska stanja [3]. U radu [6] predložen je C-LSTM (engl. *Convolutional LSTM*) model za rješavanje problema kuta zakreta volana vozila. C-LSTM se sastoji od dva modela dubokog učenja, a to su CNN i LSTM. Model CNN izvlači značajke iz svake slike u predanome nizu te iste predaje u LSTM koji uči vremensku ovisnosti između predanih slika te se kut zakreta volana izračunava pomoću klasifikacijskog sloja s podacima dobivenih iz LSTM-a. U fazi treniranja navedeni model se malo proširuje kako bi se uključili: realni podaci skretanja,

mehanizam filtriranja, koder i na kraju algoritam temeljen na povratnom širenju. Model je treniran na *comma.ai* skupu podatka. Rezultati treniranja su uspoređeni s drugim metodama dubokog učenja treniranim na istom skupu podataka. Predloženi model je računao gubitke po funkciji RMSE (engl. *Root Mean Square Error*) koja je prikazana formulom:

$$RMSE = \sqrt{\frac{1}{|D|}\sum_{i=1}^{|D|}(G_i - P_i)^2}. \tag{2-2}$$

U prethodnoj formuli G_i i P_i su stvarne i predviđene vrijednosti za sliku *i*, a *D* predstavlja ukupan broj slika. Ostvareni gubitak modela jest 16 stupnjeva pogreške zakret volana. Predloženi model je dobio 35 % bolje rezultate u odnosu na druge modele prikazane u radu.

U idućem radu autora Amini et al. [7] prikazan je problem odlučivanja kretanja automobila na temelju predane rute. Autori rada predlažu varijacijski model koji na temelju slika stanja ispred automobila i predložene rute određuje buduću kretnju automobila. Na slici 2.2. prikazan je predloženi model. Model je treniran na skupovima podataka dobivenim iz vožnje vozila u realnim

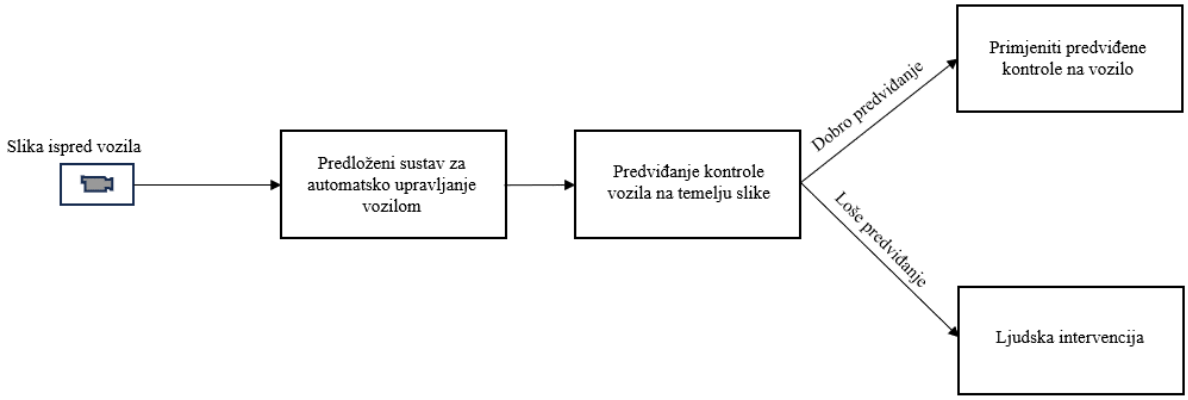


Slika 2.2. Prijedlog arhitekture modela za kontrlou vozila [7]

situacijama. Svaka slika ispred automobila pridružena je slici karte bez prikazane rute i slici karte s prikazanom rutom automobila. Testom se pokazalo, preko matrice konfuzije, kako mreža može povezati sliku ispred vozila i sliku karte bez prikazane rute [7].

3. VLASTITI SUSTAV ZA AUTOMATSKO UPRAVLJANJE VOZILOM U OVISNOSTI O STANJU NA CESTI

U ovome poglavlju opisan je cjelokupni proces kreiranja vlastitog rješenja sustava za automatsko upravljanje vozilom. Također, bit će navedeni i opisani korišteni alati pomoću kojih je kreiran vlastiti sustav i, za njega, potreban skup podataka. Sustav će raditi na temelju neuronskih mreža s jednim ulazom u mrežu te će imati dva izlaza iz mreže. Ulaz u mrežu će biti slika stanja na cesti ispred automobila, dok su izlazne vrijednosti kut zakreta volana u rasponu od -1 do 1 kao i brzina automobila u rasponu od 0 do 8 m/s. U rasponu zakreta volana negativne vrijednosti predstavljaju zakret volana u lijevo, a pozitivne vrijednosti zakret volana u desno. Unutar sustava, izlaze će određivati dvije neuronske mreže gdje je jedna zadužena za predviđanje brzine kretanja vozila, a druga za predviđanje kuta zakreta volana. Ako su vrijednosti predviđanja loše, vozač mora intervenirati i preuzeti vožnju kako ne bi došlo do sudara. Na slici 3.1. prikazan je dijagram tijeka koncepta rješenja.



Slika 3.1. Dijagram toka predloženog koncepta sustava.

Kako bi se realizirao navedeni koncept, potrebno je prvo napraviti skup podataka kojim će biti istrenirane navedene neuronske mreže. Skup podataka mora sadržavati veliku količinu podataka i podaci moraju prikazivati stanje ispred vozila u što više različitih situacija. Nakon što je pripremljen skup podataka, odabran je algoritam dubokog učenja s kojom će se realizirati predloženi sustav. Dva najpopularnija algoritma dubokog učenja za vozila su CNN i LSTM algoritmi. U ovom radu je izabran CNN model. Kada je model odabran, potrebno je podatke obraditi i izbaciti nebitne podatke. Potrebno je postaviti prihvatljive parametre treniranja kako bi se model mogao istrenirati s napravljenim podacima. Na kraju, dobiveni sustav se treba evaluirati kako bi se provjerila valjanost modela.

3.1. Podešavanje i instalacija potrebnih alata

Vlastito rješenje za sustav automatskog upravljanja vozilom, u ovisnosti o stanju ceste, je realizirano u programskom jeziku *python* pomoću programskog alata *Visual Studio Code* na operacijskom sustavu *Microsoft Windows 10*. Skup podatka je napravljen u CARLA simulatoru pomoću programskog jezika *python* i programskog alata *Visual Studio Code*. Napravljen skup podataka je strukturiran u programu za proračunske tablice *Microsoft Excel* u obliku *csv* (engl. *Comma Separated Value*) datoteke, radi olakšanog pristupa podacima. Za učitavanje podataka i treniranje neuronske mreže korišteno je razvojno okruženje *Anaconda Navigator*, programski alat *Visual Studio Code* te *NumPy, Pandas, TensorFlow, Keras, OpenCV* biblioteke.

3.1.1. *NumPy* biblioteka

NumPy je biblioteka otvorenog koda koja je temeljna *Python* biblioteka za numeričko i znanstveno računalstvo. Ova biblioteka nudi potporu za stvaranje velikih i višedimenzionalnih nizova, matrica i polja kao i razne alate za napredne i brze matematičke operacije nad skupovima podataka. Neke od operacija na koje se može primijeniti ova biblioteka su: matematičke, logičke, manipulacija oblikom, sortiranje, Furijerova transformacija, itd. *NumPy* biblioteka također omogućava učinkovitu manipulaciju podataka različitih formata te se može koristiti u analizi podataka, strojnom učenju i obradi slike. Za izradu ovoga diplomskog rada korištena je *NumPy* verzija 1.24.3 [8].

3.1.2. *Pandas* biblioteka

Pandas je *Python* biblioteka koja pruža brze i fleksibilne podatkovne strukture osmišljene kako bi rad s relacijskim ili označenim podacima bio jednostavan i intuitivan. *Pandas* je izgrađen na temelju *NumPy* biblioteke i namijenjen je dobroj integraciji unutar znanstvenog računalnog okruženja s mnogim drugim bibliotekama. Nudi dvije primarne strukture podataka: *DataFrame* i *Series*. S navedenim strukturama podataka može se obraditi većina tipičnih slučajeva upotrebe u statistici, financijama i mnogim drugim poljima. Za izradu ovoga diplomskog rada korištena je *Pandas* verzija 1.5.3 [9].

3.1.3. *TensorFlow* biblioteka

TensorFlow je popularna biblioteka otvorenog koda za strojno učenje koju je razvio *Google*. Ona se može koristiti u velikom broju programskih jezika, uključujući *Python, JavaScript, C++* i *Java*. Fleksibilnost ove biblioteke omogućuje niz primjena u različitim sektorima.

TensorFlow se primarno koristi za definiranje i kreiranje modela strojnog učenja. Posebno se koristi za modele dubokog učenja, treniranje definiranog, ili već predtreniranoga modela te za evaluaciju i predikciju rezultata s definiranim modelima. Ona nudi izvođenje operacija na komponentama računala. Za izradu ovoga diplomskog rada korištena je *TensorFlow-GPU* (engl. *Graphics Processing Unit*) biblioteka verzije 2.6.0. jer ona omogućuje treniranje modela preko GPU-a, što znatno ubrzava proces treniranja [10].

3.1.4. *Keras* biblioteka

Keras je biblioteka otvorenog koda koja pruža *Python* sučelje za kreiranje i upravljanje umjetnim neuronskim mrežama. Ona je dizajnirana da pruži jednostavno sučelje visoke razine za izgradnju, treniranje, evaluaciju i implementaciju neuronskih mreža. *Keras* je poznat po svojoj jednostavnosti, fleksibilnosti i kompatibilnosti s drugim popularnim okvirima dubokog učenja *TensorFlow* i *Theano* [11]. Za izradu ovoga diplomskog rada korištena je *Keras*-GPU verzija 2.6.0.

3.1.5. *OpenCV* biblioteka

OpenCV (engl. *Open Source Computer Vision*) je biblioteka otvorenog koda koja služi za računalni vid i obradu slika. Biblioteka je napisana u programskom jeziku *C++* ali je napravljena za korištenje u *Python* programskom jeziku. Nudi veliku količinu alata i funkcija koji služe za učitavanje, manipulaciju, spremanje slika i videozapisa. U biblioteci se nalaze vrlo korisni algoritmi za prepoznavanje raznih objekata, prepoznavanje lica i praćenje objekata u videozapisu. Za izradu ovoga diplomskog rada korištena je *OpenCV* verzija 4.7.0.72 [12].

3.1.6. *Unreal Engine*

Unreal Engine vrlo je popularan te je jedan od najraširenijih alata za stvaranje videoigara, a vrlo često se naziva samo UE. Alat je razvila svjetski poznata kompanija *Epic Games*. Vodeći je *engine* u industriji videoigara. Dizajniran je za stvaranje interaktivnog, visokokvalitetnog 3D sadržaja, što podrazumijeva video igrice, simulacije kao i arhitektonske vizualizacije. Posebno je popularan zbog svoje dostupnosti programerima početnicima kao i velikim kompanijama. U ovom radu UE je nužan za instaliranje CARLA simulatora. Za izradu ovoga diplomskog rada korištena je verzija UE-a 4.26 [13].

3.1.7. CARLA simulator

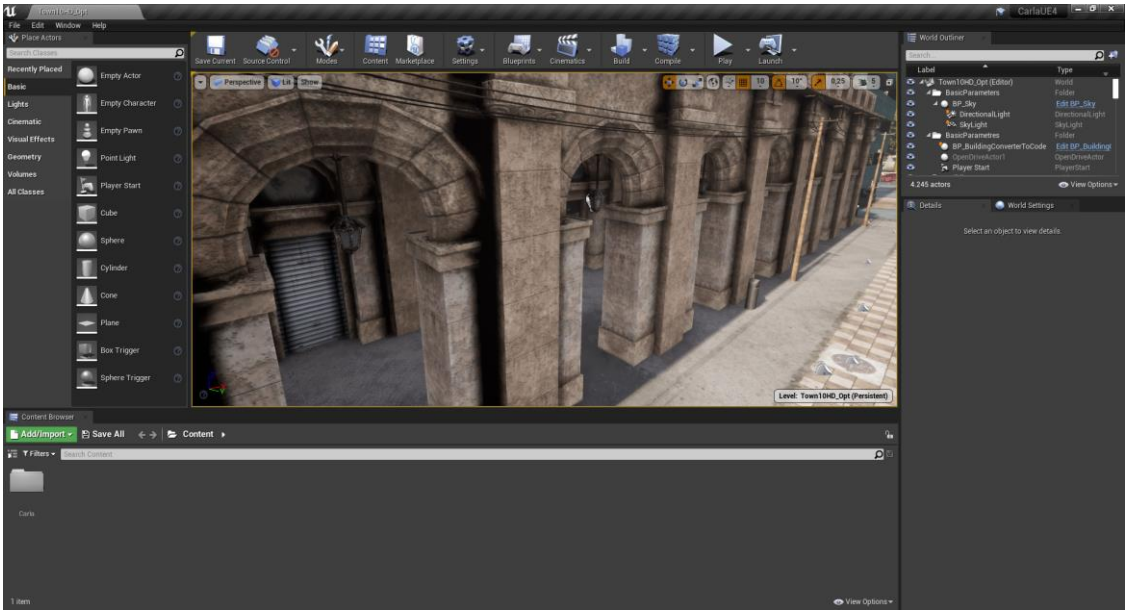
CARLA je simulator autonomne vožnje otvorenog koda. Simulator je napravljen od nule kako bi mogao služiti kao API (engl. *Application Programming Interface*) koji se može koristiti za rješavanje velikog raspona problema vezanih s autonomnom vožnjom. Glavni cilj simulatora je približiti istraživanje i razvoj autonomne vožnje prosječnom čovjeku. Korisnik s lakoćom može koristiti i prilagođivati sadržaj simulatora. CARLA simulator se temelji na alatu *Unreal Engine* za izvođenje simulacije, a koristi *OpenDRIVE* standard za definiranje cesta i urbanih sredina. Pomoću CARLA API-ja korisnik može kontrolirati simulaciju preko programskih jezika *Python* ili *C++*. CARLA simulator funkcionira na principu klijent-server arhitekture. Server je odgovoran za sve što je vezano sa simulacijom, kao što je: čitanje podataka sa senzora, izračunavanje fizike ponašanja objekata, ažuriranje stanja i svih aktera unutar simulacije. Simulacije najbolje performanse postižu kada se izvode na jednom GPU-u čija je jedina namjena pokretanje i održavanje servera. Klijentska strana zadužena je za kreiranje modula s kojim se upravljaju akteri u trenutnoj simulaciji. Akteri u CARLA-i su elementi koji izvode akcije unutar simulacije, a mogu utjecati na druge aktere. Korisnik može upravljati vremenskim uvjetima na mapi te može mijenjati gradove u kojima će se izvoditi simulacija. Sve navedeno korisnik može postići korištenjem CARLA API-ja. CARLA ima mnogo različitih značajki i elemenata koji koegzistiraju unutar nje. Neke od tih elemenata su:

- Voditelj prometa (engl. *Traffic manager*) – ugrađeni sustav koji preuzima kontrolu vozila. Sustav se koristi za kreiranje velikih i kompleksnih urbanih scenarija, pri tome da akteri imaju realistična ponašanja.
- Senzori (engl. *Sensors*) – vozila se oslanjaju na informacije o okolini dobivene od senzora. U CARLA simulatoru oni su posebna vrsta aktera koji se mogu spojiti s vozilom te se podaci mogu dohvatiti i pohraniti. Simulator nudi veliku količinu postavki senzora koji se mogu mijenjati kako bi se promijenilo ponašanje senzora. Senzori koji se mogu koristiti u simulatoru su različite vrste kamera i detektora, RADAR, LIDAR, akcelerometar, GNSS (engl. *Global Navigation Satellite System*), IMU (engl. *Inertial Measurement Unit*) te RSS (engl. *Remote Sensing System*) sigurnosni senzor.
- Snimač (engl. *Recorder*) – koristi se za rekonstrukciju simulacije korak po korak za svakog aktera unutar simulacije. Omogućuje vidljivost svakog trenutka tijekom simulacije i savršen je alat za praćenje simulacije.

- Otvorena sredstva (engl. *Open assets*) – CARLA koristi razne mape s prikazanim različitim urbanim okruženjima. Korisnik može mijenjati vremenske uvjete i pristupiti knjižnici nacrta s različitim akterima. Ti se akteri mogu promijeniti, a mogu se kreirati i potpuno novi.
- ROS (engl. *Robot Operating System*) most i *Autoware* implementacija – integracija simulatora u druga okruženja za učenje.
- Pokretač scenarija (engl. *Scenario runner*) – CARLA nudi niz ruta koje opisuju različite situacije za ponavljanje scenarija za što lakše učenje vozila.

CARLA simulator se može instalirati na operacijskom sustavu *Windows* ili *Linux* te kao *Docker* verzija. Svaka se verzija može instalirati kao paket verzija ili kao *Unreal Engine* urednik. Paket verzija ne sadrži alate za modificiranje sadržaja i postavke za razvoj. Za izradu i validaciju ovoga rada instalirana je *Unreal Engine* urednik verzije 0.9.13. Navedena verzija simulatora instalirana je radi potrebe stvaranja skupa podataka. Instalacija *Windows* verzije simulatora se odvija u dva velika koraka. Prvi korak se odnosi na provjeru sustavnih zahtjeva računala i instalaciju potrebnog softvera koji su nužni za uspješnu instalaciju simulatora. *Unreal Engine* urednik CARLA simulatora je vrlo zahtjevan te je za instalaciju potreban 64 bitni *Windows* operacijski sustav, 165 gigabajta slobodnog prostora na tvrdom diku, GPU s minimalno 6 gigabajta video memorije, dva TCP porta i dobra internetska konekcija. Potreban softver za instalaciju se dijeli na manje instalacije, veće instalacije i *python* alate. Potrebne su i manje softverske instalacije kao što su *CMake, Git, Make* (isključivo verzije 3.81, inače su moguće poteškoće tijekom instalacije), *7Zip*, *Python3* x64. Putanje svih navedenih programa potrebno je postaviti u varijable okruženja. P*ython* alat *pip3* obavezno mora biti verzije veće od 20.3. Veće instalacije podrazumijevaju instalaciju programskog alata *Visual Studio 2019* te još dodatnih komponenti putem ugrađenog instalacijskog programa *Visual Studio Installera* i instalaciju posebne verzije urednika *Unreal Engina*, s adaptacijama koje podržavaju simulator. Tu verziju urednika može se preuzeti s *git* stranice simulatora. Nakon instaliranih svih potrebnih alata može se preuzeti simulator preko *gita* i započeti drugi korak postupka. U drugom koraku potrebno je pokrenuti skriptu *Update.bat* za preuzimanje svog potrebnog sadržaja i postaviti varijablu okruženja *UE4_ROOT*. Sljedeći je korak sastaviti klijentsku i serversku stranu s funkcijama upisanim u x64

Visual Studio konzolu, *make pythonAPI* i funkcijom *make launch*. S funkcijom *make launch* sastavlja se i pokreće server te se prikazuje uređivač (Slika 3.2) [14].



Slika 3.2. Izgled otvorene *Unreal Engine* verzije simulatora

Tijekom instaliranja došlo je do greške pa se nisu mogle učitavati korisnikove *python* skripte u simulator. Bilo je potrebno provjeriti jesu li se stvorile *.egg* datoteke u *carla/PythonAPI/carla/dist* mapi. U ovome se slučaju datoteke nisu stvorile. Kako bi se riješio taj problem potrebno je bilo obrisati cijelu *carla/Build* mapu te u x64 terminalnu upisati naredbu *make osm2odr*. Nakon toga potrebno je bilo na internetu pronaći *xerces-c-3.2.3* datoteku, preuzeti ju i spremiti u *carla/Build* mapu. Na kraju je bilo potrebno ponovno upisati naredbu *make PythonAPI*. Server se opet pokreće naredbom *make launch* te se otvara urednik i korisnik može koristiti skripte.

3.2. Kreiranje i opis skupa podataka

Za potrebe vlastitog rješenja kreiran je skup podataka koji sadrži tri podatka, jedan ulaz i dva izlaza. Ulaz mora biti slika iz kamere koja gleda u smjeru kretanja vozila. Dva izlaza su brzina kojom se vozilo kreće i kut zakreta volana vozila. Postoje razni javno dostupni skupovi podatka s kojim se mogu istrenirati modeli za predviđanje kuta zakreta volana. Neki od tih skupova su *Sully Chen* [15] i *comma2k19* skup podataka [16]. Oni sadrže slike prikupljene tijekom vožnje na cestama Kalifornije te svaka slika sadrži svoju oznaku zakreta volana.

Za treniranje ovoga sustava potrebno je bilo napraviti novi skup podataka. Skup podataka je kreiran pomoću CARLA simulatora Za dohvaćanje i spremanje slika koristi se ključni senzor,

RGB kamera, koju je moguće postaviti na bilo koju *x*, *y* i *z* koordinatu u učitanoj simulaciji te ju je moguće okretati oko svake navedene osi. Funkcija *listen()* pokreće „slušanje“ na svako osvježavanje podatka u kameri te pokreće korisnikovu funkciju. Pokrenuta funkcija se odvija u zasebnoj niti (engl. *thread*) i odvija se autonomno od ostatka programa. Ubačena su u simulaciju i dodatna vozila kako bi se mogao simulirati promet.

Nužno je postaviti alat na sinkronu simulaciju kako bi izvođenje simulacija postalo deterministično. Prije pokretanja simulacije potrebno je postaviti funkciju svih vozila *set_autopilot()*. Funkcija *set_autopilot()* postavlja vozilu autonomnu kontrolu te vozilo slijedi sva pravila vožnje.

Neophodno je snimiti slike koje prikazuju stanje ispred vozila. Kamera se može postaviti na vozilo, što znači da će se kamera automatski prilagođavati i pratiti kretanje odabranog vozila. Metodom *listen()* se pokreće napisana korisnikova funkcija za spremanje slika.

Posljednji postupak je prikupljanje podatka o kutu zakreta volana i brzini vozila u svakom trenutku. Brzinom se u simulatoru može na više načina upravljati. Može se upravljati sustavom papučice za ubrzavanje i papučice za kočenje, također se može upravljati putem direktnog podešavanja brzine vozila. Za izradu skupa podataka se prikupljala trenutna brzina automobila izraženim u m/s. Prvo je brzina ograničena na 30 km/h na svim cestama u mapi. Nakon toga se pomoću funkcije *get_velocity()* dobiva vektor brzine vozila. Kako bi se dobila ukupna brzina vozila, računamo ju izrazom:

$$brzina = \sqrt{{b_x}^2 + {b_y}^2 + {b_z}^2}, \tag{3-1}$$

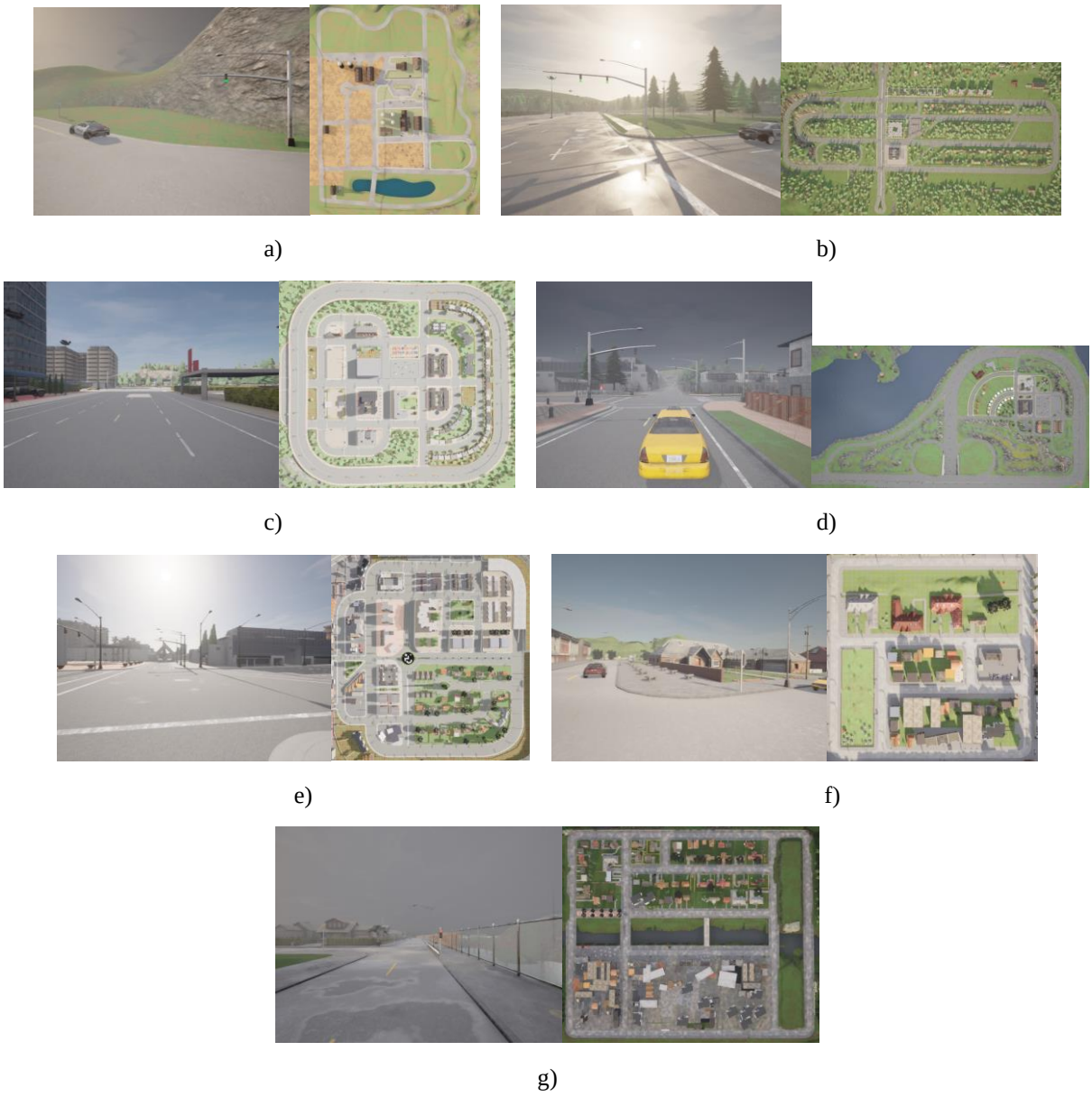
gdje su b_x, b_y, b_z vrijednosti brzine u određenom smjeru koordinatnog sustava, a *brzina* je ukupna brzina vozila izražena u m/s. Brzina se vozilu, također, može postaviti i funkcijom *set_target_velocity()* koja prima vektor brzine. Kako bi se dobio vektor brzine potrebno je dobiti klasu *transform* vozila. Unutar *transform* klase postoji metoda *get_foward_vector()* koja vraća jedinični vektor koji pokazuje smjer prednjeg dijela vozila. Množenjem jediničnog vektora vozila i skalarom brzine dobiva se vektor brzine koji se može predati funkciji *set_target_velocity()*. Kut zakreta volana se može dobiti preko *VehicleControl* klase vozila. U navedenoj klasi postoji atribut *steer* koji prikazuje kut zakreta volana i može iznositi od -1 za skretanje lijevo do 1 za skretanje desno. *VehicleControl* klasa se može dobiti od vozila metodom *get_control()*. Atribut *Steer* se

mora postaviti na željenu vrijednost u *VehicelCorntrol* klasi. Navedena klasa se može postaviti metodom vozila *apply_control()*. Realizirane su sve potrebne komponente za pravljenje skupa podataka.

Simulator nudi osam različitih gradova u kojima korisnik može kreirati svoje simulacije. Cilj je napraviti skup podataka za svaku mapu. Simulator u postavkama sadrži atribut *fixed_delta_seconds* koji određuje koliko će se često događati otkucaji simulatora na principu: „Broj otkucaja = 1/ *fixed_delta_seconds*“. Broj otkucaja jest ukupan broj izračuna simulacije u sekundi. Što je atribut manji to će se odvijati više otkucaja u simulatoru te će se simulacija opteretiti i bit će sporija. Sporija simulacija omogućava da se napisanim kodom preuzme više podataka tako što će se podaci na kameri sporije osvježavati. Prebrzo osvježavanje podataka na kameri rezultira gomilanjem slika koje se trebaju spremiti na disk. Preuzete dimenzije slika su 1024x768 te je postavljeno da kamera uzima novi podatak svakih 0.45 otkucaja (engl. *tick*) u simulatoru. Kako bi se usporila simulacija može se povećati brzina otkucaja (engl. *tick rate*) simulatora. Podaci su spremljeni u *csv* datoteke u formatu: *putanja_slika_ispred,brzina,kut_zakreta_volana*. Iz simulatora je prvo preuzeto 21.590 slika sa svojim oznakama za prvi skup podataka. Podaci su podijeljene na trening, testni i validacijski skup. U validacijskom skupu izdvojeno je 2422 (11.2 %) podataka, u testnom 2284 (10.5 %) podataka, a ostatak 16,884 (78.2 %) podataka je u trening skupu. Na slici 3.3. prikazane su slike ispred vozila, brzina i kuta zakreta volana vozila u jednom trenutku tijekom vožnje te slika grada 10.

Slika 3.3. Prikaz stanja ispred vozila u gradu10, dok je brzina: 7.93, a kut zakreta volana: -0.25

Svaki grad u simulatoru je drugačiji. Na slikama 3.4. prikazani su slike stanja ceste ispred vozila, brzina i kut zakreta volana vozila u jednom trenutku tijekom vožnje na različitim gradovima u simulatoru.



a) b) c) d) e) f) g)

Slika 3.4. Prikaz različitih gradova i primjer jednog podatka za svaki grad sa oznakama a) grad7, brzina: 3.43 , kut zakreta volana:-0.38, b) grad6, brzina: 7.98 , kut zakreta volana: 0.61, c) grad5, brzina: 8.03, kut zakreta volana: 0, d) grad4, brzina: 0 , kut zakreta volana: 0, e) grad3, brzina: 4.59 , kut zakreta volana: 0.11, f) grad2, brzina: 7.02, kut zakreta volana: 0.8, g) grad1, brzina: 3.97 , kut zakreta volana: 0.01

Za potrebe treniranja neuronske mreže, preuzet je skup podataka *Sully Chen* i napravljen je dodatni trening skup podataka s velikom količinom slika kamere postavljene ispred vozila u CARLA simulatoru. Oba navedena dodatna skupa namijenjena su za treniranje mreže koja predviđa kut zakreta volana. Napravljeni dodatni trening skup podataka sastoji se od 82,777 podataka koji su preuzeti iz svih gradova u simulatoru. U *csv* datoteci prikazano je ime slike sa

kutom zakreta volana. Slike su slične kao slike ispred vozila prikazane na slici 3.3. i 3.4. *Sully Chen* skup podataka sastoji se od dva dijela koji prikazuje dvije različite vožnje. Prvi dio se sastoji od 45405, a drugi dio se sastoji od 63824 slika. Slike su spremljene u JPEG formatu dimenzije 455x256. Slike i podatke o zakretu volana spaja *csv* datoteka koja ima navedena imena slika i pripadajući zakret volana. Zakret volana prikazan je u stupnjevima u rasponu od 540° do -540° gdje pozitivne vrijednosti prikazuju skretanje udesno, a negativne vrijednosti prikazuju skretanje ulijevo. Potrebno je pretvoriti raspon u stupnjevima u raspon od -1 do 1. Na slici 3.5. je prikazan primjer slike iz *Sully Chen* skupa podataka.

Slika 3.5. Primjer slike iz *Sully Chen* skupa podataka i pripadajuća vrijednost zakreta volana od 0°

3.3. Opis vlastitog sustava za automatsko upravljanje vozilom u ovisnosti o stanju na cesti

CNN model korišten je za realizaciju predviđanja kuta zakreta volana na temelju pogleda ispred vozila te predviđanja brzine, sličan *Nvidinom* modelu *PilotNET* prikazan na slici 2.1. Glavna razlika između predloženog modela i *PilotNETa* je u tome što su između konvolucijskih slojeva dodani slojevi normalizacije serije (engl. *BatchNormalization layer*), a na kraju konvolucijskih slojeva dodan je sloj gubitka (engl. *Dropout layer*) sa stopom od 50 %, kako bi se spriječilo pretreniravanje (engl. *overfitting*) modela. Ulaz u model je četverodimenzionalno polje. Prva oznaka polja označava veličinu serije (engl. *batch size*) koja ulazi u model. Kada se napravi pregled modela, uvijek se oznaka prikaže kao *None* (slika 3.9), jer ona nije poznata unaprijed i određuje se u procesu treniranja. Druga i treća ulazna veličina označavaju dimenzije slike koje se prosljeđuju modelu što je u ovome slučaju 66x200. Posljednja veličina označava dubinu slike, to jest koliko kanala sadrži slika. Na primjer, koriste se tri kanala ako je slika u RGB ili YUV formatu, ili jedan kanal ako je slika u sivim nijansama (engl. *grayscale*). Model sadrži sveukupno 252,907

parametara od kojih 344 ne mijenjaju parametre tijekom treniranja. Veličine značajki slika, koje model obrađuje, prikazane su na slici 2.1. Bilo je potrebno napraviti dvije različite mreže istog sastava, pri čemu jedna predviđa kut zakreta volana, a druga brzinu pa ih je potrebno spojiti u jedan model. Treniranjem brzine i kuta zakreta volana s istim modelom nije dalo dobre rezultate. U prilogu P.3.1. prikazan je kod kojim se kreira model prikazan na slici 3.6.

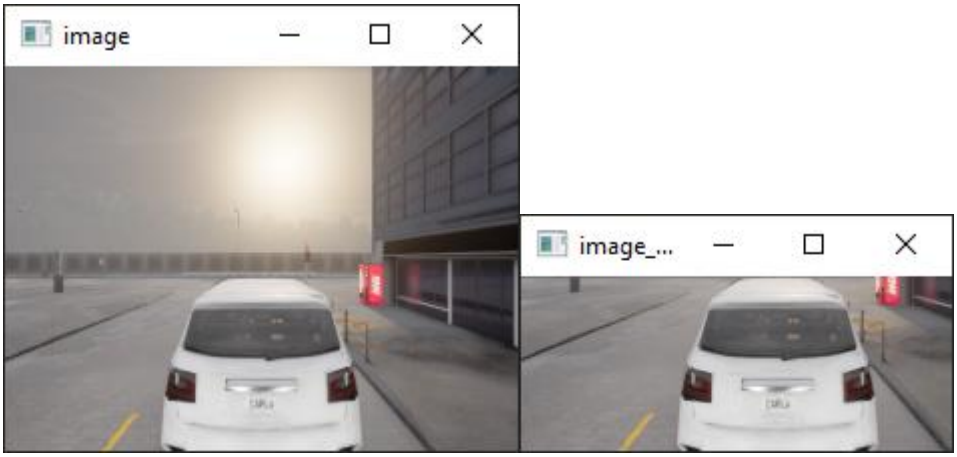
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 66, 200, 3)]      0

conv2d (Conv2D)              (None, 31, 98, 24)        1824

batch_normalization (BatchN  (None, 31, 98, 24)        96
ormalization)

conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636

batch_normalization_1 (Batc  (None, 14, 47, 36)        144
hNormalization)

conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248

batch_normalization_2 (Batc  (None, 5, 22, 48)         192
hNormalization)

conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712

batch_normalization_3 (Batc  (None, 3, 20, 64)         256
hNormalization)

conv2d_4 (Conv2D)            (None, 1, 18, 64)         36928

dropout (Dropout)            (None, 1, 18, 64)         0

flatten (Flatten)            (None, 1152)              0

dense (Dense)                (None, 100)               115300

dense_1 (Dense)              (None, 50)                5050

dense_2 (Dense)              (None, 10)                510

dense_3 (Dense)              (None, 1)                 11

=================================================================
Total params: 252,907
Trainable params: 252,563
Non-trainable params: 344
_________________________________________________________________
```

Slika 3.6. Prikaz sastava neuronske mreže

3.3.1. Priprema podataka za treniranje mreže

Trenutni podaci unutar skupa podataka nisu pogodni za treniranje neuronskih mreža. Glavni problem predstavlja prevelika količina nevažnih dijelova slike i neadekvatne dimenzije slika koje su nepogodne za treniranje mreže. Slikama ispred vozila potrebno je smanjiti dimenzije (engl. *resize*) i podrezati ih (engl. *crop*) tako da slike ne sadrže nevažne dijelove, kao što su nebo i priroda te se zadržava samo najvažnije, dio ceste ispred vozila. Slike iz CARLA simulatora smanjene su četiri puta, s 1024x768 na 256x192 piksela. Nakon što je slika smanjena, potrebno je bilo odrediti područje interesa (engl. *Region of interest*, ROI) slike. Regija interesa, u slučaju

smanjene slike, predstavlja dio visine od 105 do 192 elemenata slike, a širine od 0 do 215 elemenata slike. Što se tiče slika iz *Sully Chen* skupa podataka, njihova veličina nije promijenjena, već je samo određeno područje interesa. Prvih 140 elemenata visine slike i posljednjih 10 elemenata slike su uklonjeni, dok su svi elementi slike u širini zadržani. Područje interesa na slikama iz oba skupa podataka prikazano je na slici 3.7.



a)

b)

Slika 3.7. Prikaz područja interesa na slikama iz a) CARLA simulatora, b) *Sully Chen* skup podataka

Kada je određeno područje interesa, sliku je bilo potrebno pretvoriti u YUV format ako se želi trenirati kut zakreta volana vozila. Ako se trenira brzina, format slike se pretvara u *grayscale*. Slika se normalizira na vrijednosti piksela od 0 do 1 kako bi bila pogodnija modelu za učenje. Normalizacija slike je postupak prilagođavanja intenziteta piksela u slici kako bi se postigla određena standardizacija ili distribucija vrijednosti. Na kraju je slika skalirana na veličinu 66x200 piksela. Jedna od slika koja je spremna za treniranje modela prikazana je na slici 3.8.

Slika 3.8. Podatak spreman za treniranje mreže za predviđanje kuta kut zakreta volana

Kada je u pitanju obrada slike za treniranje mreže koja je zadužena za određivanje brzine vozila, na slici se, također, treba odrediti područje interesa. Umjesto pretvaranja slika u YUV format, slike se transformiraju iz BGR formata u *grayscale* format. Ako je slika u *grayscale* formatu, onda je ulazni podatak u mrežu manje veličine (None, 200, 66, 1) jer slike u *grayscaleu* imaju samo jedan kanal dubine. Prikaz slike za treniranje modela za predviđanje brzine prikazan je na slici 3.9.

Slika 3.9. Podatak spreman za treniranje mreže za predviđanje brzine

Augmentacija podataka je tehnika koja se obično koristi u strojnom učenju i računalnom vidu, posebno za treniranje modela dubokog učenja, kako bi se poboljšala njihova izvedba i generalizacija. Uključuje primjenu različitih transformacija na postojeći skup podataka kako bi se stvorile nove, malo izmijenjene verzije podataka uz zadržavanje istih oznaka ili klasifikacija. U ovome slučaju korištene su tri augmentacije na slikama, a to su zrcaljenje slike, mijenjanje svjetline slike i pomicanje kuta gledišta vozila. Zrcaljenje slike služi kako bi model dobio potpuno novi podatak s oznakom koja je suprotna od stare oznake. Pomicanje kuta gledišta omogućava da model dobije podatak koji predstavlja pomak vozila u odnosu na sredinu prometne trake. Pomicanjem gledišta se, također, mora znatno promijeniti i oznaka za taj podatak, a dobiva se formulom:

$$nova\ oznaka = stara\ oznaka\ \pm\ random\left(\frac{2 \times random + 1}{4}\right), \qquad (3\text{-}2)$$

gdje je *random* slučajan broj od -0.6 do 0.6 te on odlučuje i za koliko će se promijeniti kut gledišta izvorne slike, a *stara oznaka* dolazi uz izvorni podatak. Primjer svake primijenjene augmentacije, na istoj slici, prikazan je na slici 3.10.

Slika 3.10. a) Izvorna slika sa oznakom 0.0, b) slika kojoj je promijenjena svjetlina, c) okrenuta slika, d) slika sa promijenjenim kutom gledišta te sa novom oznakom 0.146.

3.4. Treniranje mreže

Treniranje CNN arhitekture, prikazane na slici 3.6, provedeno je u tri faze i to korištenjem skupa podataka prikazanim u poglavlju 3.2. Prva faza uključuje treniranje na *Sully Chen* skupu podataka. U drugoj fazi je primijenjena tehnika prijenosa učenja (engl. *transfer learning*) koristeći dodatni trening skup podataka od 82,777 slika iz CARLA simulatora. Treća faza obuhvaća treniranje na trening skupu od 16,884 uz dodatak augmentacije na kutu gledišta slike, kako bi se vozilo znalo oporaviti od pogreške izlijetanja iz kolničkih linija tijekom vožnje. Budući da je količina podataka u trening skupu velika, bilo je potrebno napraviti generator podataka jer

dostupna radna memorija računala nije bila dovoljna za učitavanje svih podataka skupa odjednom. Generator predaje modelu dio po dio podataka iz skupa. Generator prima sljedeće ulazne vrijednosti: skup podataka i oznaku koja označava je li predani skup podataka dio trening skupa. U slučaju kada je skup podataka označen kao trening skup, podaci se uzimaju iz skupa i slike se augmentiraju prema prethodno opisanom postupku. Ako oznaka ukazuje da se ne radi o trening skupu, podaci se ne augmentiraju. Nakon obrade, slike i oznake pohranjuju se u zasebne *Numpy* nizove veličine jedne serije (engl. *batch*). Kada je serija u potpunosti popunjena, ona se predaje funkciji *fit_generator()* s kojom se započinje treniranje neuronske mreže. Kada generator dođe do kraja skupa podataka, skup se ponovno promiješa (engl. *shuffle*) i nastavlja se treniranje.

Prije nego što se započinje treniranje potrebno je odvojiti *Sully Chen* skup podataka na trening, validacijski i testni skup. Od cijelog skupa 10 % (10,923) podataka je korišteno za validacijski skup, 10 % (10,923) podataka je korišteno za testni skup, a ostalih 80 % (87,385) je korišteno u trening skupu.

Maksimalna količina epoha za treniranje je 200, ali je također postavljeno prijevremeno zaustavljanje (engl. *Early stopping*). Prijevremeno zaustavljanje funkcionira tako da se odabere parametar treniranja i ako se ona ne poboljša unutar proizvoljne količine epoha, zaustavlja se treniranje. Na taj način se sprječava pretreniravanje. Veličina serije je postavljena na 64 podatka jer je ta veličina pokazala najbolje rezultate. Za kriterijsku funkciju odabrana je srednja kvadratna pogreška (engl. *Mean squared error*, MSE). Također, dodana je metrika, srednja apsolutna pogreška (engl. *Mean apsolute error*, MAE) za dodatnu preglednost treniranja. Korišteni optimizator je *Adam* optimizator s brzinom učenja (engl. *learning rate*) od 0.001. Također je korištena *ReduceLROnPlateau()* funkcija koja promatra određeni parametar treniranja i ako se taj parametar ne poboljša unutar proizvoljne količine epoha, smanji se brzina učenja. Tijekom treniranja spremljen je model s najmanjim validacijskim gubitkom. Treniranje se odvijalo korištenjem procesora i grafičke kartice, što je znatno ubrzalo proces treniranja.

Za treniranje CNN modela definiranog na slici 3.6, korišteno je osobno računalo koje sadrži sljedeće komponente:

- *Procesor Intel Core i7 – 6700 CPU*;
- 16 gigabajta radne memorije;
- *NVIDIA GeFroce GTX 1070 Ti GPU*.

Korišteni parametri za treniranje spomenutog modela na *Sully Chen* skupu podataka su:

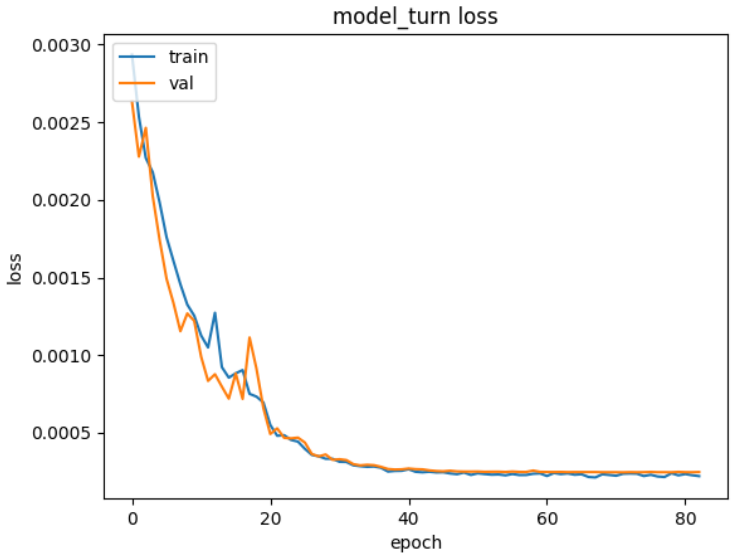
- Veličina serije – 64;
- Broj epoha – 200;
- Adamov optimizator;
- Brzina učenja – 0.001;
- *ModelCheckpoint* – promatra *val_loss* kriterijsku funkciju, koja označava gubitke na validacijskom skupu i sprema parametre modela na najmanju zabilježenu vrijednost;
- *ReduceLROnPlateau* – promatra varijablu *val_loss* na odabranu proizvoljnu količinu epoha, 5;
- *EarlyStopping* - promatra varijablu *val_loss* na odabranu proizvoljnu količinu epoha, 11.

Kao funkcija gubitaka koristila se funkcija MSE. Računala se metrika MAE koja dodatno prikazuje tijek treniranja i ne utječe na izračunavanje gubitaka. One se izračunavaju sljedećim funkcijama:

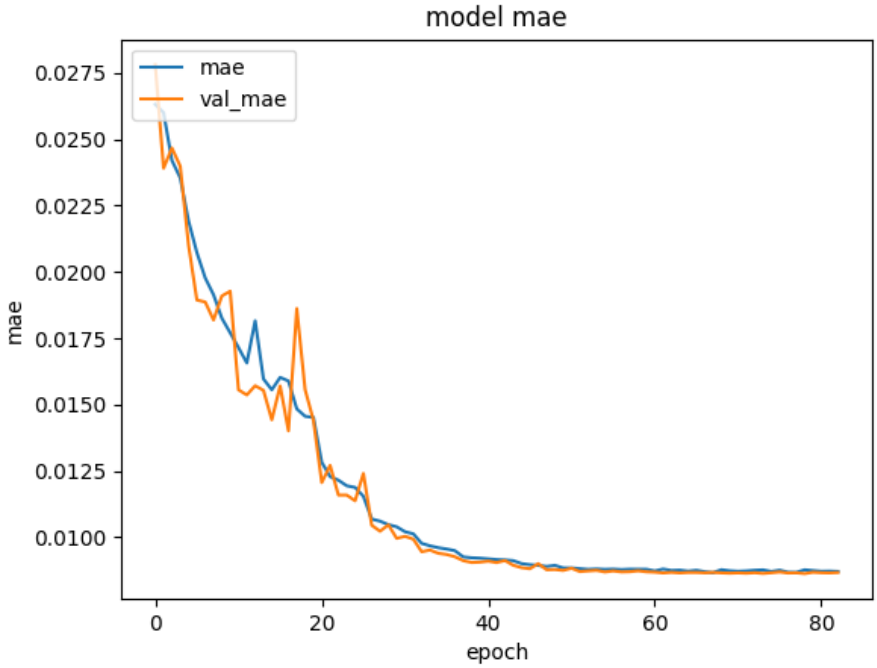
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2, \tag{3-3}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - p_i|, \tag{3-4}$$

gdje je n ukupni broj podataka, y_i je broj koji se nalazi u trenutnoj iteraciji, a p_i je broj koji je predviđen za trenutnu iteraciju. Prikaz krivulje gubitaka i prikaz MAE metrike, trenirane na *Sully Chen* skupu podataka, prikazane su na slici 3.11.



a)

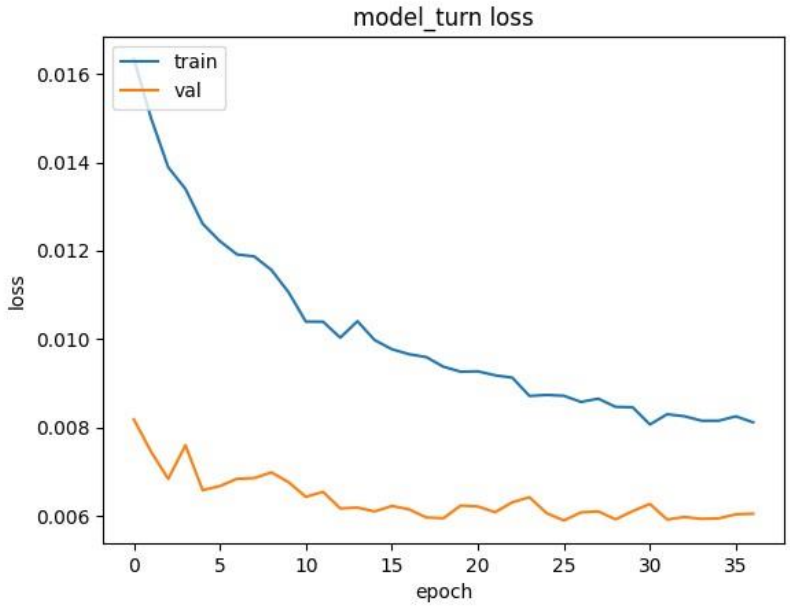


b)

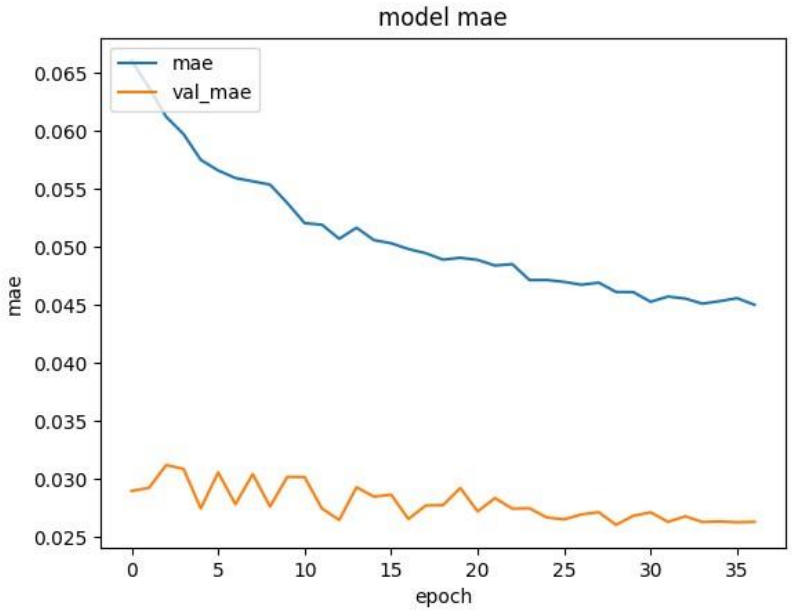
Slika 3.11. Krivulje tijeka treniranja modela na *Sully Chen* skupu podataka a) Krivulja gubitaka na trening i validacijskom skupu, b) krivulja metrike MAE na trening i validacijskom skupu

„*Train“* se odnosi na kriterijsku funkciju gubitaka MSE izračunatu na trening skupu, „*val*“ se odnosi na kriterijsku funkciju MSE izračunatu na validacijskom skupa. „*Mae*“ se odnosi na metriku MAE izračunatu na trening skupu, a „*val_mae*“ se izračunava na validacijskom skupu.

Pad krivulje znači da je model naučio predviđati vrijednosti na temelju unesenih ulaznih podataka predanih iz *Sully Chen* skupa podataka. Parametri su spremljeni na kraju epohe u kojoj je izračunat najmanji validacijski gubitak. Taj model je dobiven oko 73. epohe jer je treniranje zaustavljeno 11 epoha poslije dobivenih najboljih parametara. Nakon treniranja, potrebno je model istrenirati na podacima iz CARLA simulatora. Korištena metoda jest prijenos učenja gdje se učitaju parametri nekog drugog modela kako bi model mogao naučiti rješavati nove probleme na temelju sličnih problema ili podataka. U ovome slučaju je model naučen na jednom skupu podataka, ali on će biti naučen i na drugačijem skupu podataka s istom namjenom. Trening skup se sastoji od 82,777 podataka, a validacijski skup se sastoji od 2,422 podataka. Dodatni skup iz CARLA simulatora je smanjen, zbog velike količine podataka u kojem se vozilo kretalo ravno. Treniranje se odvijalo pomoću istog uređaja i pomoću istih trening parametara navedenih u prijašnjem treniranju. Krivulja gubitaka i krivulja MAE metrike, koje su trenirane na dodatnom skupu podataka iz CARLA simulatora, prikazane su na slici 3.12.



a)

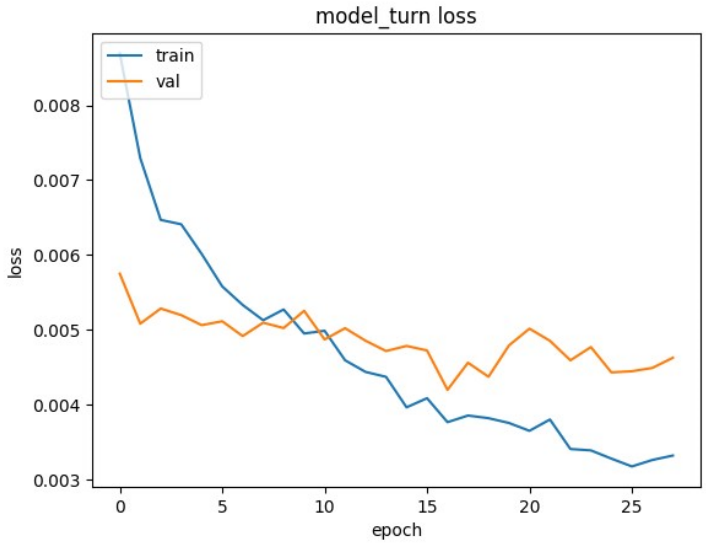


b)

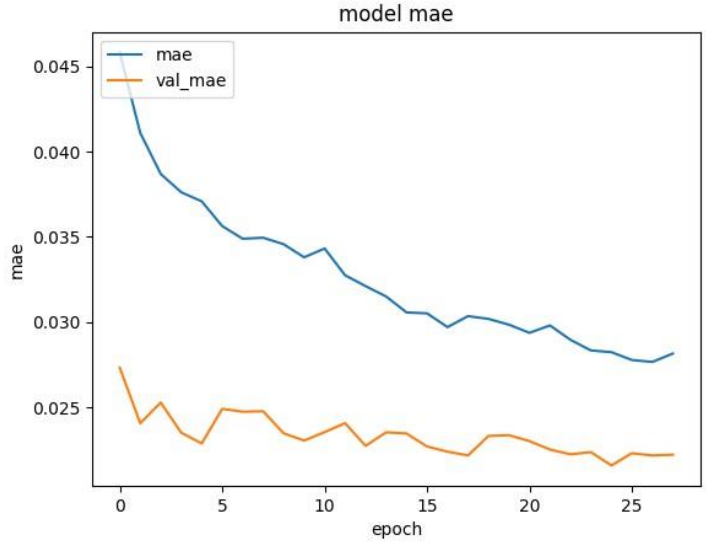
Slika 3.12. Krivulje tijeka treniranja modela na dodatnom skupu iz CARLA simulatora a) Krivulja gubitaka na trening i validacijskom skupu, b) krivulja metrike MAE na trening i validacijskom skupu

Arhitektura je trenirana još jednom sa prvim kreiranim skupom. Implementirana je augmentacija kojom se translatira slika. Time je dobivena slika koja prikazuje trenutak u kojem se vozilo mora oporaviti od pogreške. Trening skup sadrži 16,884 podataka, a validacijski skup sadrži 2,422 podataka. Treniranje se odvijalo pomoću istog uređaja i pomoću istih trening parametara

navedenih u prijašnjem treniranju. Krivulja gubitaka i krivulja MAE metrike, koje su trenirane na slikama u smjeru kretanja vozila, iz prvog skupa podataka, prikazane su na slici 3.13.



a)

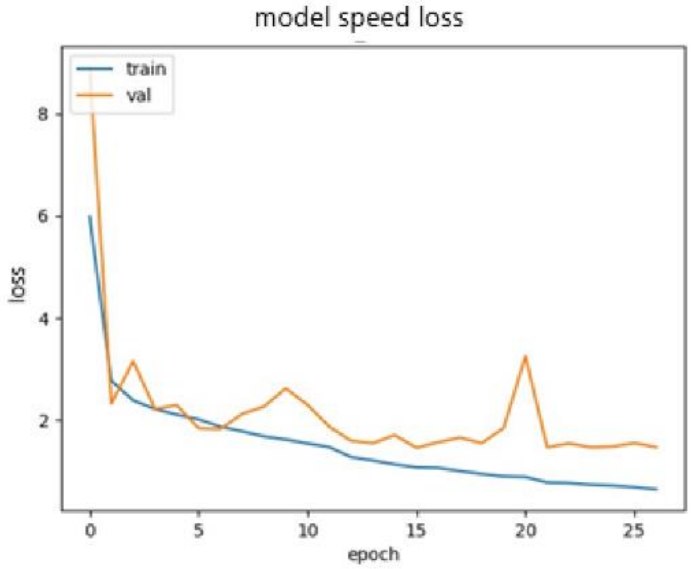


b)

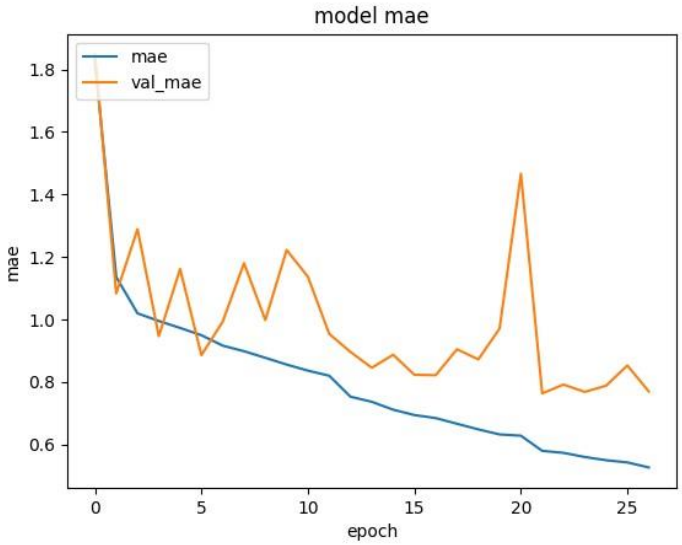
Slika 3.13. Krivulje tijeka treniranja modela na prvom skupu podataka a) krivulja gubitaka na trening i validacijskom skupu, b) krivulja metrike MAE na trening i validacijskom skupu

Istrenirana je neuronska mreža koja obavlja predviđanje brzine vozila u određenom trenutku. Za realizaciju toga, korišten je model prikazan na slici 3.6. U ovom slučaju model, kao što je već prethodno navedeno, prima slike u *grayscale* formatu. Model je treniran na slikama koje prikazuju stanje ceste ispred vozila, oznake tih slika je brzina vozila. Ulaz u model je zato: veličina serije, 66, 200, 1 (1 prikazuje broj kanala u slici). Izabrane su slike u *grayscale* formatu jer su

pokazale bolje rezultate nego slike u YUV ili RGB formatu boja. Trening skup sadrži 16,884 podataka, a validacijski skup sadrži 2,422 podataka. Treniranje se odvijalo pomoću istog uređaja i pomoću istih trening parametara navedenih u prijašnjem treniranju. Prikaz krivulje gubitaka i prikaz MAE metrike je vidljiv na slici 3.14.



a)



b)

Slika 3.14. Krivulje tijeka treniranja modela za predviđanje brzine, trenirane na prvom skupu podataka a) Krivulja gubitaka na trening i validacijskom skupu, b) krivulja metrike MAE na trening i validacijskom skupu

Nakon završetka treniranja razvijeni su modeli za predviđanje brzine i kuta zakreta volana na temelju slike stanja ispred vozila. Model za predviđanje kuta zakreta volana treniran je koristeći

podatke iz situacija gdje se na cesti nalaze pune crte, isprekidane crte u bijeloj ili žutoj boji te raskrižja. Model za predviđanje brzine je treniran sa slikama na kojima se nalazi nekoliko vozila raznih boja i veličina. Vozilo bi se trebalo zaustaviti kada se ispred njega nađe drugo vozilo. Kako bi se procijenila funkcionalnost ovih modela i vidjelo koliko oni dobro rade u stvarnom vremenu, u simulatoru, potrebno je provesti evaluaciju.

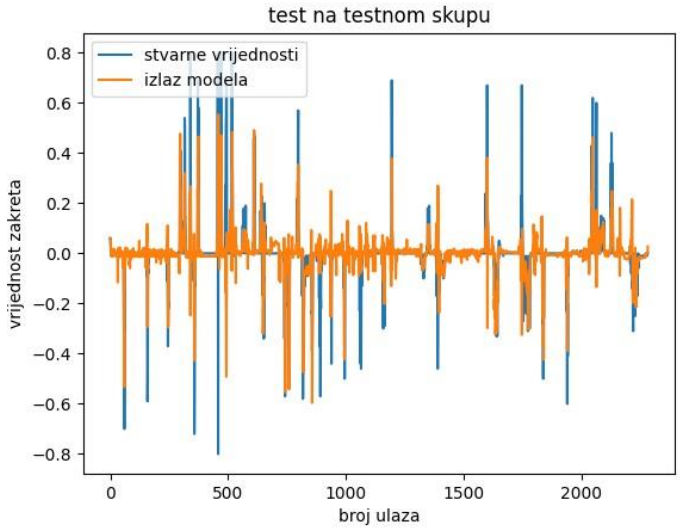
4. TESTIRANJE I EVALUACIJA VLASTITOG SUSTAVA ZA AUTOMATSKO UPRAVLJANJE VOZILOM

U ovome poglavlju objašnjen je proces evaluacije modela na testnom skupu i testom u virtualnom okruženju preko simulatora. Ona je nužan proces u prikazivanju rada modela i razumijevanju njegovih kvaliteta i mana. Evaluacija je izvedena na osobnom računalu s komponentama:

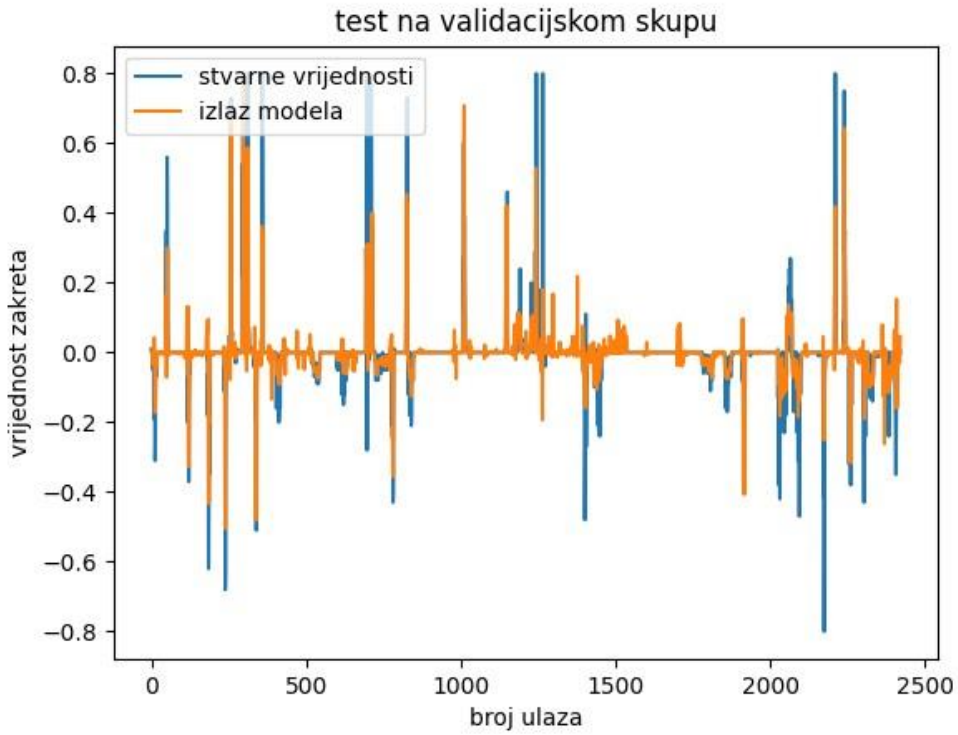
- *Procesor Intel Core i7 – 6700 CPU*;
- 16 gigabajta radne memorije;
- *NVIDIA GeFroce GTX 1070 Ti GPU*;
- *Windows 10, 64 bit* operacijski sustav.

4.1. Rezultati testiranja sustava na testnom skupu

Cilj evaluacije na testnom skupu jest provjeriti funkcionalnost modela. Prvi korak je kreiranje testnog skupa. On je realiziran tijekom kreiranja manjeg skupa podataka i nije sudjelovao u procesu treniranja. Skup sadrži podatke iz vozila na svim mapa. Ukupan broj podataka koji se nalazi u testnom skupu jest 2,284. Također će se napraviti test i na validacijskom skupu. Testiranje je izvedeno pomoću *keras* funkcije *predict*() koja generira izlazna predviđanja za predane ulazne uzorke, koji su u ovome slučaju slike stanja ispred vozila. Uspoređeni su izlazi i originalne vrijednosti iz testnog skupa. Na slici 4.1. i 4.2 prikazane su izlazne vrijednosti zakreta volana modela i originalne vrijednosti skupa.



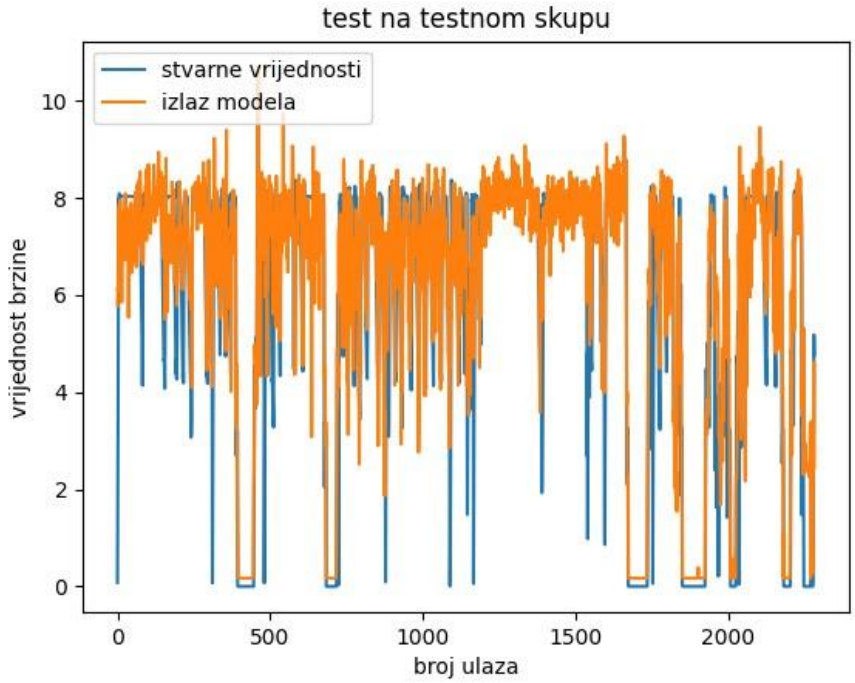
Slika 4.1. Prikaz izlaza modela za predviđanje kuta zakreta volana i stvarnih vrijednosti testnog skupa iz CARLA



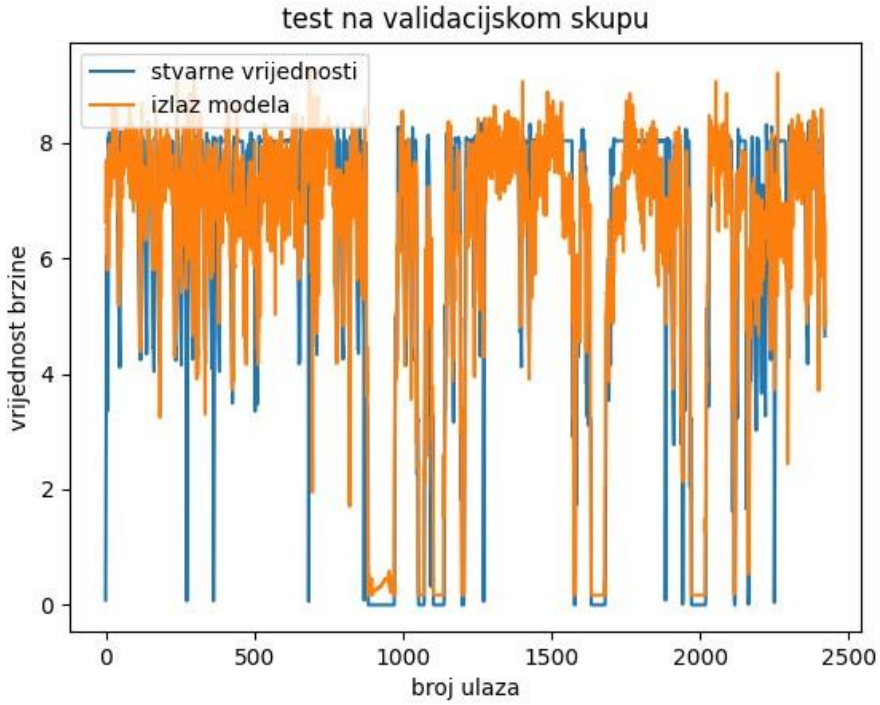
Slika 4.2. Prikaz izlaza modela za predviđanje kuta zakreta volana i stvarnih vrijednosti validacijskog skupa iz CARLA simulatora

Testni i validacijski skup sadrži sve moguće situacije u kojoj se može vozilo nalaziti, kao na primjer situacije na raskrižju, mali, ali i veliki zavoji. Izračunat je MAE na testnom skupu i iznosi 0.0356, na validacijskom skupu iznosi 0.0222. MAE prikazuje za koliko prosječno model griješi u predviđanju. Iz slike se može primijetiti da model dobro predviđa vrijednosti na dani podatak. Mogu se primijetiti pogreške predviđanja prije velikih okreta volana, što je pogreška koju model napravi prije nego što uđe u raskrižje. Navedene pogreške nastaju zbog nedovoljne količine podataka u kojima vozilo ima veliki kut zakreta volana. Također, model ima poteškoća s predviđanjem velikih vrijednosti, ali je smjer zavoja većinom dobar. S ovim testom se može potvrditi da model ima dobru osnovnu funkcionalnost.

Evaluacija na testnom skupu je obavljena i na modelu koji predviđa brzinu vozila. Korišten je isti skupovi podataka kao i u prošloj evaluaciji jer sadrži podatke o brzini vozila i razne situacije koje zahtijevaju različite brzine vozila, kao na primjer: stajanje iza vozila različitih boja i oblika, skretanje na raskrižju, vožnja u pravcu i sl. Na slici 4.3. i 4.4. prikazane su originalne vrijednosti testnog ili validacijskog skupa i izlazne vrijednosti modela za predviđanje brzine.



Slika 4.3. Prikaz izlaza modela za brzinu i stvarnih vrijednosti testnog skupa iz CARLA simulatora



Slika 4.4. Prikaz izlaza modela za brzinu i stvarnih vrijednosti validacijskog skupa iz CARLA simulatora

Izračunat je MAE za testni skup i iznosi 0.776, za validacijski skup iznosi 0.823. Iz rezultata testa može se primijetiti kako se vozilo uspješno zaustavlja kada se ispred njega nalazi drugo vozilo. Osnovna funkcionalnost modela je ostvarena. Iz napravljenih testova ne mogu se uočiti velike mane i pogreške modela, nego se može utvrditi je li ostvarena željena osnovna funkcionalnost. Također u testnom skupu vozilo se uvijek nalazi u sredini prometne trake i uvijek se zaustavi na istoj udaljenosti od vozila. Bilo bi potrebno testirati vozilo u drugačijim, nesavršenim uvjetima. Za dodatnu evaluaciju model je testiran u virtualnom okruženju pomoću CARLA simulatora.

4.2. Rezultati testiranja sustava na simulatoru u realnim uvjetima

U okviru testiranja napravljen je stvarno vremenski test modela u različitim vremenskim uvjetima. Testiranje će se izvesti u virtualnom okruženju pomoću CARLA simulatora. Prvo je potrebno ujediniti već spomenute modele u jedan model kako bi vrijednosti brzine i kut zakreta volana vozila dolazile u istom trenutku. U prilogu P.4.1. prikazan je kod za realiziranje većeg modela. Dobiveni model ima dva ulaza i dva izlaza. Ulaz je ista slika ispred vozila, ali je u jednom slika pretvorena u YUV format boja, a u drugom ulazu u *grayscale* format boja. Izlazi modela su brzina vozila i kut zakreta volana vozila.

Testiranje je provedeno na svim mapama dostupnim s instalacijom CARLA simulatora. Ukupno je izabrana 21 putanja koje sadrže 75 raskrižja. Vozilo se kretalo po vanjskim cestama svake mape te je testirano kako se ponaša na raskrižjima i na kružnim tokovima ako ih mapa sadrži. Kreiran je promet od 25 vozila koji uvijek kruže gradom. Tijekom testiranja računalo se vrijeme trajanja vožnje, ukupni prijeđeni put vozila i broj intervencija. Vrijeme je računato od početka do kraja simulacije, a izraženo u sekundama. Prijeđeni put vozila je suma izračuna udaljenosti između dvije točke u koordinatnom sustavu. Te točke su prijašnja i trenutna točka promatranog vozila. Proces je odrađen tijekom cijelog testiranja i na kraju je dobivena ukupna prijeđena udaljenost. Intervencija se računa svaki put kada korisnik korigira grešku vozila. Greška se smatra svaki trenutak kada vozilo prekrši pravilo vožnje. Kao na primjer: kada vozilo uđe u suprotnu traku, kada se vozilo zabije u drugo vozilo te kada bi se vozilo našlo u situaciji u kojoj nema mogućnosti oporavka od pogreške. Kada bi se greška dogodila, zaustavlja se simulacija i vozilo se vraća na najbližu poziciju poslije greške te se nastavlja testiranje.

Kako bi se modeli bolje testirali, uvedene su i različiti vremenski uvjeti. CARLA simulator nudi razne vremenske parametre koji se mogu mijenjati kako bi se postigli razni vremenski uvjeti.

Najistaknutije promjene su mijenjanje položaja sunca, jačina oborine, količina lokvi na cestama, vlažnost kamere i gustoća magle. Zato su izabrane četiri različita vremenska uvjeta u kojima će se testirati auto. Ti vremenski uvjeti su dan, noć, maglovito vrijeme po danu i kišovito vrijeme po danu. Sunce se može postaviti u rasponu od 90 do -90, gdje raspon označava dio dana (90 je podne, a -90 je ponoć). Gustoća magle, količina oborine, broj lokvi na cesti, vlažnost kamere i oblačnost se mjeri u rasponu od 0 do 100, gdje 100 prikazuje maksimalni intenzitet parametra. Vlažnost kamere se prikazuje kao zamućivanje slike iz RGB kamere. Parametri za navedene vremenske uvjete su slijedeći:

a) Dan:
    - Položaj sunca: 90;
    - Gustoća magle 0;
    - Količina kiše: 0;
    - Broj lokvi na cesti: 0;
    - Vlažnost kamere: 0;
    - Oblačnost: 0;
b) Noć:
    - Položaj sunca: -90;
    - Gustoća magle 0;
    - Količina kiše: 0;
    - Broj lokvi na cesti: 0;
    - Vlažnost kamere: 0;
    - Oblačnost: 0;
c) Kišovito po danu:
    - Položaj sunca: 30;
    - Gustoća magle 0;
    - Količina kiše: 100;
    - Broj lokvi na cesti: 50;
    - Vlažnost kamere: 50;
    - Oblačnost: 50;
d) Maglovito po danu:
    - Položaj sunca: 30;
    - Gustoća magle 50;

- Količina kiše: 0;
- Broj lokvi na cesti: 0;
- Vlažnost kamere: 0;
- Oblačnost: 30.

Svi vremenski uvjeti prikazani su na slici 4.3.

a) b)

c) d)

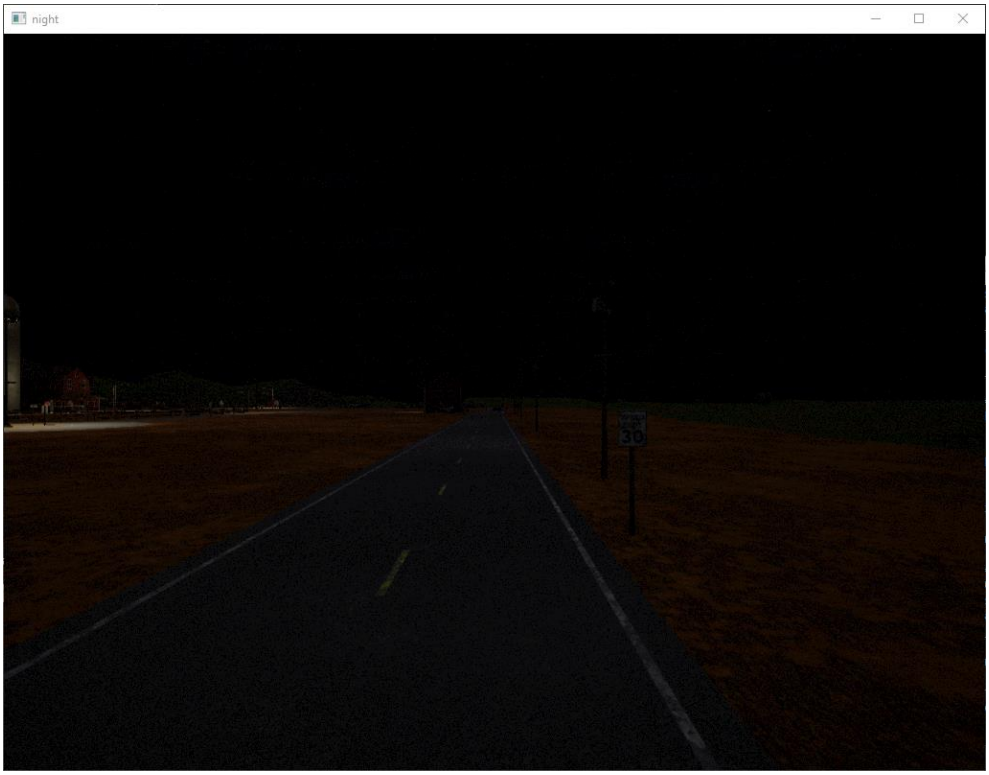
Slika 4.3. Vremenski uvjeti u simulatoru na gradu 10: a) dan, b) noć, c) kišovito, d) maglovito

Postavljena je granična vrijednost od 3.5 m/s za predviđanje brzine. Ako je izlaz modela ispod 3.5 m/s vozilo će stati, ako je izlaz iznad granične vrijednosti, predviđena vrijednost brzine će se primijeniti na vozilo. Za statistiku, izračunat je broj pogrešaka vozila po kilometru i autonomnost. Ukupno vrijeme trajanja testiranja, prijeđeni put i broj grešaka prikazani su na tablici 4.1.

Tablica 4.1. Ukupno vrijeme testiranja, prijeđena udaljenost i broj pogrešaka

Vremenski uvjeti	**Ukupno vrijeme testiranja [s]**	**Ukupna prijeđena udaljenost [m]**	**Broj pogrešaka**
Dan	3,743	20,337	18
Noć	2,187	10,511	205
Kišovito	4,221	20,090	28
Maglovito	3,961	20,264	14

Iz dobivenih rezultata može se primijetiti kako je sustav loše predviđao tijekom noćnih uvjeta vožnje. Sustav je najmanje testiran po noćnim uvjetima jer na gradovima 3, 4, 5, 6 i 7 uopće nije bio funkcionalan te uopće nije mogao autonomno funkcionirati. Razlog tome jest slaba vidljivost kolničkih linija po noći. Također je primijećeno da i prevelika svjetlina može loše utjecati na performanse sustava. Vozilo bi počelo izlijetati s ceste kada bi prolazilo točno ispod upaljene lampe u simulatoru ili kada je odbljesak sunčeve svijetlosti na cesti presvijetao. Na slici 4.4. prikazana je slika po noći te bi pri takvom ulazu u model vozilo izletilo s ceste.



Slika 4.4. Trenutak u kojem vozilo pravi lošu predikciju, zbog slabe osvijetljenosti ceste

Na slici se može primijetiti kako su kolničke linije slabo vidljive, što je i razlog lošeg predviđanja vožnje po noći kada grad nije dobro osvijetljen. Uz vođenje prijašnje spomenutih vrijednosti, također je zabilježeno kako je vozilo pogriješilo. Tijekom testiranja primijećeno je da vozilo može pogriješiti na raskrižju, kada ne poštuje kolničke linije ili kada se zabije u vozilo zbog loše predikcije sustava. Na tablici 4.2. prikazana je broj za svaku vrstu pogreške.

Tablica 4.2. Prikaz vrsta pogrešaka tijekom testiranja

Vremenski uvjeti	**Pogreške na raskrižju**	**Pogreška nepoštivanja kolničkih linija**	**Broj zabijanja u vozila/broj uspješnih stajanja iza vozila**	**Ukupan broj pogrešaka**
Dan	14	4	0/11	18
Noć	57	145	3/11	205
Kišovito	16	8	4/11	28
Maglovito	11	3	0/11	14

Vođenje statistike o vrsti pogreške vozila je nužna jer su pogreške na raskrižjima očekivane. Sustav nema točno definirano ponašanje kada uđe u raskrižje. Može se reći da se sustav ponaša nasumično na raskrižjima. Međutim sustav jest treniran da uspješno skrene na raskrižju i tijekom testiranja po danu primijećeno je kako vozilo vrlo često uspije dobro skrenuti na raskrižju ako skrene čim uđe u raskrižje. Problem skretanja na raskrižju se može riješiti kreiranjem kompleksnijeg sustava koji dobiva podatak sa iscrtanom putanjom vozila i predviđa pravilni smjer kretanja vozila u raskrižju. Takav primjer je prikazan radu autora Amini et al. [7]. U testiranju pri maglovitom vremenu, primijećeno je da se sustav najbolje drži kolničkih linija. Kišoviti vremenski uvjeti pokazali su najlošije performanse modela za predviđanje brzine, gdje je pogriješio četiri puta. Također, sustav je dosta griješio na skretanjima od 90° na svim vremenskim prilikama. Uočeno je 19 pogreška nepoštivanja kolničkih linija koje su se upravo dogodile na tim skretanjima. Izračunate su autonomnost i broj pogrešaka po kilometru sa i bez pogrešaka na raskrižjima kako bi se prikazala kvaliteta svakog sustava. Na tablici 4.3 prikazane su autonomnosti i broj pogrešaka po kilometru. Autonomnost je računata po izrazu (2-1).

Tablica 4.3. Prikaz vrsta pogrešaka tijekom testiranja

Vremenski uvjeti	**Broj pogrešaka po kilometru**	**Autonomnost**	**Broj pogrešaka po kilometru bez grešaka na raskrižju**
Dan	0.88	96.9 %	0.19

Noć	19.5	43.76 %	5.70
Kišovito	1.39	96 %	1
Maglovito	0.69	97.9 %	0.15

Izračunate metrike nam potvrđuju testiranje i prikazuju kvalitetu sustava u svim vremenskim uvjetima. No mora se uzeti u obzir da se vozilo nije kretalo uvijek istom brzinom te autonomnost nije najbolji pokazatelj kvalitete sustava. Zato je, uz autonomnost sustava, izračunat broj pogrešaka po kilometru. Prema izračunatim metrikama sustav ima daleko najlošije rezultate tijekom noći. Nije ga bilo moguće testirati na većini gradova jer uopće nije bio funkcionalan zbog slabe vidljivosti kolničkih linija. Sustav najbolje radi pri sunčanim uvjetima. Također se isto može reći i za maglovito razdoblje, ako se zanemare pogreške na raskrižjima. Sustav je ukupno testiran 14,112 sekundi te je testno vozilo prešlo ukupno 71,202 metara i pogriješilo 265 puta. Ukupna autonomnost sustava iznosi 88.7 %. Ukupni broj pogrešaka po kilometru iznosi 3.72 greška/kilometar, a ako se isključe pogreške na raskrižju, taj broj pogrešaka iznosi 2.34 greška/kilometar. Sustav se uspješno kreće unutar kolničkih linija osim tijekom noći i uspješno se zaustavlja kada se ispred vozila nalazi drugo vozilo.

5. ZAKLJUČAK

Cilj ovog diplomskog rada jest napraviti sustav koji na temelju pregleda stanja ispred vozila može uspješno predvidjeti kut zakreta volana i brzinu kojom će se kretati vozilo. Za realizaciju ovog rješenja, temelj je model *PilotNET* koji je stvorila tvrtka *Nvidia*. Za postizanje cilja bilo je potrebno stvoriti kompletno novi skup podataka koji sadrži promet. Za svaki ulazni podatak skupa, postoje dva izlazna podatka. Prvi skup podataka je napravljen uz pomoć CARLA simulatora te sadrži slike na 8 različitih gradova te ukupno 21,590 podataka. Također je napravljen dodatni skup podataka iz simulatora i preuzet *Sully Chen* skup podataka. Metodom regresijskog učenja istrenirani su dva različita modela gdje jedan predviđa kut zakreta volana na temelju stanja ispred vozila i model koji predviđa brzinu na temelju iste slike. Kao ulaz modela za predviđanje kuta zakreta volana koristi sliku u YUV formatu, a model za predviđanje brzine uzima sliku u *grayscale* formatu. Model za predviđanje kuta zakreta volana treniran je na tri faze. Prva faza je istrenirana na *Sully Chen* skupu podataka, nakon toga je istrenirana s velikim skupom podataka iz CARLA simulatora i na kraju je istrenirana sa manjim kreiranim trening skupom podataka uz dodane augmentacije za oporavljanja od pogreške. Model za predviđanje brzine treniran je sa manjim trening skupom. Evaluacija dobivenih modela je provedena pomoću testnog skupa, koji nije sudjelovao u treniranju, validacijskog skupa i u virtualnom okruženju pomoću simulatora na svim gradovima pri različitim vremenskim uvjetima. Evaluacija na testnom skupu koristi se kako bi se vidjela osnovna funkcionalnost modela. Test preko simulatora pokazuje mane modela i sve eventualne pogreške koje se događaju. Tijekom testa mjerilo se ukupno vrijeme testiranja i ukupna prijeđena udaljenost vozila te su izračunati autonomnost sustava i broj pogrešaka po kilometru. Oba dobivena modela objedinjena su u jedan sustav koji izbacuje vrijednosti u isto vrijeme. Sustav je ukupno postigao autonomnost od 88.7 % i pokazao 3.72 greška/kilometar. Ne uzimajući u obzir greške na raskrižjima, model je postigao 2.34 greška/kilometar. Greške na raskrižjima su očekivane jer sustav nema na odnosu čega donijeti odluku u kojem smjeru će skrenuti na raskrižju te je ponašanje sustava na raskrižju nasumično. Kako bi se izbjegao taj problem i usavršio sustav, potrebno je napraviti kompleksniji sustav koji promatra položaj vozila na karti i njegovo okruženje. Sustav se dobro ponaša kada se nalazi unutar kolničkih linija i uspješno se zaustavlja kada se ispred njega nalazi vozilo.

LITERATURA

[1] World Health Organization, „SDG Target 3.6 Halve the number of global deaths and injuries from road traffic accidents“. Pristupljeno: 21. rujan 2023. [Na internetu]. Dostupno na: https://www.who.int/data/gho/data/themes/topics/sdg-target-3_6-road-traffic-injuries

[2] D. Pomerleau, „ALVINN: An Autonomous Land Vehicle In a Neural Network“, *Proc. NeurIPS Neural Inf. Process. Syst.*, str. 305–313, pros. 1989.

[3] S. Grigorescu, B. Trasnea, T. Cocias, i M. Gigel, „A survey of deep learning techniques for autonomous driving“, *J. Field Robot.*, sv. 37, izd. 3, str. 362–386, lis. 2019, doi: 10.1002/rob.21918.

[4] M. Bojarski *i ostali*, „End to End Learning for Self-Driving Cars“. arXiv, 25. travanj 2016. Pristupljeno: 27. rujan 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1604.07316

[5] A. Agnihotri, P. Saraf, i K. R. Bapnad, „A Convolutional Neural Network Approach Towards Self-Driving Cars“, u *2019 IEEE 16th India Council International Conference (INDICON)*, Rajkot, India: IEEE, pros. 2019, str. 1–4. doi: 10.1109/INDICON47234.2019.9030307.

[6] H. M. Eraqi, M. N. Moustafa, i J. Honer, „End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies“. arXiv, 22. studeni 2017. Pristupljeno: 02. listopad 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1710.03804

[7] A. Amini, G. Rosman, S. Karaman, i D. Rus, „Variational End-to-End Navigation and Localization“, u *2019 International Conference on Robotics and Automation (ICRA)*, svi. 2019, str. 8958–8964. doi: 10.1109/ICRA.2019.8793579.

[8] „NumPy“. Pristupljeno: 21. rujan 2023. [Na internetu]. Dostupno na: https://numpy.org/

[9] „Pandas“. Pristupljeno: 22. rujan 2023. [Na internetu]. Dostupno na: https://pandas.pydata.org/docs/getting_started/overview.html

[10] „Tensoflow“. Pristupljeno: 22. rujan 2023. [Na internetu]. Dostupno na: https://www.tensorflow.org/

[11] „Keras: Deep Learning for humans“. Pristupljeno: 22. rujan 2023. [Na internetu]. Dostupno na: https://keras.io/

[12] „OpenCV - Open Computer Vision Library“. Pristupljeno: 22. rujan 2023. [Na internetu]. Dostupno na: https://opencv.org/

[13] „Unreal Engine: The most powerful real-time 3D creation tool“. Pristupljeno: 22. rujan 2023. [Na internetu]. Dostupno na: https://www.unrealengine.com/en-US/unreal-engine-5

[14] „CARLA Simulator“. Pristupljeno: 26. rujan 2023. [Na internetu]. Dostupno na: https://carla.readthedocs.io/en/0.9.14/

[15] S. Chen, “driving-datasets,” 2018. Pristupljeno: 20. studeni. 2023. , [Na internetu]. Dostupno na: https://github.com/SullyChen/ driving - datasets

[16] H. Schafer, E. Santana, A. Haden, and R. Biasini, “A commute in data: The comma2k19 dataset,” 2018.

SAŽETAK

U ovom diplomskom radu napravljen je sustav za automatsko upravljanje vozilom u ovisnosti o stanju na cesti. Kao osnova sustava korištena je *Nvidia* neuronska mreža zvana *PilotNET*. Zadatak sustava je da mora na odnosu podatka o stanju ceste, predvidjeti brzinu vozila i kut zakreta volana. Potrebno je bilo napraviti potpuno novi skup podataka koji će sadržavati potrebne podatke. Nakon treniranja, razvijene su dvije neuronske mreže koje, u odnosu na pregled stanja na cesti, predviđaju brzinu i kut skretanja. Neuronske mreže su objedinjene u jedan model kako bi se sinkronizirano estimirale vrijednosti zakreta volana i brzine vozila. Evaluacija je provedena u dva koraka. Prvi korak evaluacije je testiranje predviđanja sustava na temelju podataka testnog skupa koji nije korišten u procesu treniranja. Ovim testom dobiven je pregled osnovne funkcionalnosti sustava. Sustav je testiran u virtualnom okruženju preko CARLA simulatora. Ovim testom prikazane su mane i pogreške koje sustav radi u stvarnom vremenskom okruženju. Sustav je postigao autonomnost od 88.7 % i 3.72 grešaka po kilometru.

Ključne riječi: autonomna vožnja, simulator, skup podataka, strojno učenje, neuronska mreža, PilotNET

AUTOMATIC VEHICLE CONTROL SYSTEM DEPENDING ON ROAD CONDITIONS

ABSTRACT

In this thesis, a system was created for automatic vehicle control depending on the road conditions. An *Nvidia* neural network called *PilotNET* was used as the basis of the system. The task of the system is to predict the speed of the vehicle and the angle of rotation of the steering wheel based on the data on the road condition. It was necessary to create a completely new data set that would contain the necessary data. After training, two neural networks were developed which, in relation to the road condition overview, predict the speed and the turning angle. Neural networks are combined into one model in order to synchronously estimate the values of steering wheel rotation and vehicle speed. The evaluation was carried out in two steps. The first step of the evaluation is to test the predictions of the system based on the data of the test set that was not used in the training process. This test provided an overview of the basic functionality of the system. The system was tested in a virtual environment using the CARLA simulator. This test shows the flaws and errors that the system makes in a real time environment. The system achieved an autonomy of 88.7 % and 3.72 errors per kilometer.

Keywords: autonomous driving, simulator, dataset, machine learning, neural network, PilotNET

ŽIVOTOPIS

Borna Svetinović rođen je 11. srpnja 1999. godine u Osijeku. Osnovnoškolsko obrazovanje je završio u osnovnoj školi Vijenac 2014. godine u Osijeku. Nakon osnovne škole upisao je III. gimnaziju Osijek koju je završio i maturirao 2018. godine. Po završetku srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Godine 2021. završava preddiplomski studij te upisuje sveučilišni diplomski studij Elektrotehnika, smjer Komunikacije i informatika, izborni blok Mrežne tehnologije. Na prvoj godini diplomskog studija postaje stipendist tvrtke TTTechAuto.

PRILOZI

Prilog P.3.1. *Python* programski kod za definiranje modela neuronske mreže

```
def create_model_PN():
    visible = Input(shape=INPUT_SHAPE)
    conv1 = Conv2D(24, kernel_size=(5,5), strides=(2,2),
activation='relu')(visible)
    BN1 = BatchNormalization()(conv1)
    conv2 = Conv2D(36, kernel_size=(5,5), strides=(2,2),
activation='relu')(BN1)
    BN2 = BatchNormalization()(conv2)
    conv3 = Conv2D(48, kernel_size=(5,5), strides=(2,2),
activation='relu')(BN2)
    BN3 = BatchNormalization()(conv3)
    conv4 = Conv2D(64, kernel_size=(3,3), strides=(1,1),
activation='relu')(BN3)
    BN4 = BatchNormalization()(conv4)
    conv5 = Conv2D(64, kernel_size=(3,3), strides=(1,1),
activation='relu')(BN4)
    drop = Dropout(0.5)(conv5)
    flat = Flatten()(drop)
    hidden1 = Dense(100, activation='relu')(flat)
    hidden2 = Dense(50, activation='relu')(hidden1)
    hidden3 = Dense(10, activation='relu')(hidden2)
    output = Dense(1)(hidden3)

    model = Model(inputs=(visible), outputs=(output))

    print(model.summary())
    return model
```

Prilog P.4.1. *Python* programski kod za definiranje modela neuronske mreže za predviđanje brzine i kuta skretanja

```
def create_model_final(model_PN_speed,model_PN_turn):
    visibleS = Input(shape=INPUT_SHAPE_speed)
    model_speed = model_PN_speed(visibleS)

    visibleY = Input(shape=INPUT_SHAPE)
    model_turn = model_PN_turn(visibleY)

    model = Model(inputs=(visibleS,visibleY),
outputs=(model_speed,model_turn))
    print(model.summary())
    return model
```