Analiza i primjena metoda umjetne inteligencije u upravljanju vozilom u videoigri Grand Theft Auto V

Široki, Dario

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Dario Široki

ANALIZA I PRIMJENA METODA UMJETNE INTELIGENCIJE U UPRAVLJANJU VOZILOM U VIDEOIGRI GRAND THEFT AUTO V

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dario Široki

Matični broj: 0016135842

Studij: Informacijski sustavi

ANALIZA I PRIMJENA METODA UMJETNE INTELIGENCIJE U UPRAVLJANJU VOZILOM U VIDEOIGRI GRAND THEFT AUTO V

ZAVRŠNI RAD

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2021.

Dario Široki

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Tema ovog rada je upravljanje vozilom u videoigri Grand Theft Auto V. Nekoliko modela temeljenih na najnovijim otkrićima na području umjetne inteligencije bit će povezani u skladnu cjelinu koja će činiti sustav. Taj sustav će na temelju procjena dobivenih od modela umjetne inteligencije moći upravljati automobilom u virtualnom okruženju videoigre. Cilj je dokazati kako je područje umjetne inteligencije dovoljno napredovalo da je uz pomoć relativno malo programskog koda i znanja moguće postići impresivne rezultate.

Ključne riječi: umjetna inteligencija; neuronske mreže; konvolucijske neuronske mreže; autonomna vozila; računalne igre; python; keras; tensorflow;

Sadržaj

1. Uvod

Ideja za temu nastala je iz serijala YouTube korisnika Sentdex [1]. Serijal je nastao 2017. godine i koristio je drugačije modele umjetne inteligencije od onih koji su korišteni u ovom radu, no serijal može i dalje poslužiti kao dobar izvor ideja i savjeta. Rad detaljno opisuje arhitekturu sustava za autonomnu vožnju, prvo pokrivajući teoretski dio, a zatim i kako sve sklopiti u skladnu cjelinu koja čini sustav koji je spreman za autonomnu vožnju virtualnim svijetom u videoigri Grand Theft Auto V.

Primjena umjetne inteligencije za rješavanje problema autonomne vožnje jedna je od njenih primjena u čiji se razvoj ulaže najviše. [2] Znanstvenici i privatne tvrtke usavršavaju svoje sustave već dugi niz godina, no dobivanje dopuštenja za javnu primjenu je izazovno. Na prometnicama je moguć velik broj scenarija i nešto trivijalno poput prepoznavanja znaka stop je zapravo zaista težak zadatak kada se uzme u obzir da znak može biti oštećen, oksidiran, slova izguljena, vegetacija ga može prekrivati itd. Nakon ovog objašnjenja, stvari počinju zvučati doista obeshrabrujuće. Baš zato je ovo toliko izazovan zadatak za koji još ne postoji savršeno rješenje, a pitanje je hoće li ikada i postojati.

Ovaj rad tek grebe površinu znanosti koja se krije iza područja autonomne vožnje. Umjesto kompleksnog sustava za koji je potrebno mnogo vremena, novaca i znanja da bi se implementirao, koristi se kraći put korištenjem modela čiji je cilj oponašanje vozača. Model na temelju onoga što vidi od pravog vozača može naučiti kako ga oponašati i time kako se ponašati u prometu. Sustavi koji se koriste u produkciji uglavnom imaju mogućnost prepoznavanja objekata, koriste razne senzore i algoritme, istovremeno dobivaju povratne informacije od više modela koji imaju različite namjene i na temelju toga donose odluke. Kraćim putem, cilj je da model samostalno nauči rješavati što više problema bez da mu se eksplicitno zadaju ti zadaci. Takvi problemi uključuju prepoznavanje prometnih traka, prometnih znakova, zadržavanje automobila u traci, upravljanje brzinom i sve ostalo što svaki vozač treba znati.

2. Metode i tehnike rada

Kao alat za istraživanje korištene su internet tražilice Google[1], Google Scholar[2], YouTube[3] i Github[4].

Praktični dio rada implementiran je u programskom jeziku Python[5] (verzija 3.9.5). Uz njega je korišten niz biblioteka: Tensorflow[6], Keras[7], H5PY[8], Numpy[9], PYXInput[10], Matplotlib[11] i OpenCV[12]. Biblioteke su opisane kroz rad u trenutcima kada se prvi puta spominju.

Za treniranje modela umjetne inteligencije korišten je servis Google Colaboratory. Colaboratory, ili Colab skraćeno, je Googleov proizvod namijenjen istraživačima. Omogućuje razvojnim programerima pisanje i izvršavanje Python koda kroz web preglednik. Okruženje je spojeno na Googleove udaljene poslužitelje koji imaju snažne grafičke kartice i tenzorske procesorske jedinice koje moderne biblioteke za duboko učenje koriste kako bi ubrzale izvršavanje svojih funkcija. Colab je dostupan u besplatnoj verziji i nekoliko plaćenih verzija. Za potrebe ovog rada korištena je besplatna verzija. Sve ove značajke čine Colab odličnom platformom za istraživački rad poput ovog. [3]

Za izradu samog rada korišten je servis Overleaf [13] gdje je ovaj dokument stvoren u LaTeX programskom jeziku.

Prvo je provedeno duboko istraživanje i informiranje o temi i proveden je praktični dio rada. Nakon što je praktični dio rada bio završen, izrađen je ovaj dokument.

[1] https://www.google.com/
[2] https://scholar.google.com/
[3] https://www.youtube.com/
[4] https://www.github.com/
[5] https://www.python.org/
[6] https://www.tensorflow.org/
[7] https://keras.io/
[8] https://docs.h5py.org
[9] https://numpy.org/
[10] https://github.com/bayangan1991/PYXInput
[11] https://matplotlib.org/
[12] https://opencv.org/
[13] https://www.overleaf.com/

3. Neuronske mreže

3.1. Uvod

Ideja za umjetne neuronske mreže nastala je još pedesetih godina prošlog stoljeća. [4] U posljednje vrijeme došlo je do golemog napretka na tom polju zbog razvoja tehnologije i znanosti. Zbog digitalizacije društva došlo je do napretka u količini podataka koja je dostupna, računalne komponente su napredovale što dozvoljava kompleksniju arhitekturu mreža, organizacije počinju shvaćati da mogu stvarati dodatnu vrijednost uz pomoć neuronskih mreža, interes za ovom djelatnošću naglo raste i nova otkrića su sve češća.

Neuronske mreže imaju široku primjenu, neke od njih su modeliranje složenih procesa, regresija, predikcija trendova, raspoznavanje uzoraka i klasifikacijski problemi. [5] Glavni ciljevi istraživanja umjetnih neuronskih mreža usmjereni su na razvijanje struktura novih mreža koje prate funkcioniranje ljudskog mozga ili barem približno oponašaju njegove funkcije u svrhu rješavanja praktičnih problema. Ukoliko računalo posjeduje sposobnost donošenja zaključaka na temelju određenih činjenica, smatra se da je ponašanje računala inteligentno. Operacije klasificiranja i uočavanja objekata iz svakodnevnog života čovjek obavlja intuitivno, dok one računalu predstavljaju teškoću. Neuronske mreže opisuje visok stupanj tolerancije grešaka pa je tako čak i pri analizi nejasnih i nepotpunih podataka moguće doći do zadovoljavajućeg rješenja. [6]

Pojam „neuronske mreže“ je dvoznačan. On se odnosi na umjetnu ili na biološku neuronsku mrežu koja je građena od bioloških neurona povezanih u periferni ili središnji živčani sustav. Neuron je osnovna jedinica živčanog sustava te se ujedno smatra i najsloženijom jedinicom ljudskog organizma. [6] Građen je od tijela stanice te mnoštva dendrita i aksona. Dendriti su kraći produžeci te njihova uloga je prijenos živčanih impulsa s osjetilnih organa do tijela stanice. Aksone sačinjavaju tanke cjevčice čiji je jedan kraj povezan s tijelom neurona, a drugi kraj se dijeli na niz grana te njihova uloga je predaja signala koje tijelo proizvodi. Razmak između završetka aksona prethodnog neurona i dendrita je sinapsa. Upravo zbog paralelnog rada neurona ljudski mozak ima mogućnost izrazito visoke brzine i sposobnosti realiziranja više raznovrsnih zadataka u isto vrijeme. Signali prelaze s jedne stanice na drugu putem sinapse, električnim ili kemijskim putem.

Drugo značenje neuronskih mreža podrazumijeva umjetne neuronske mreže. One su građene od međusobno povezanih umjetnih neurona. Svrha umjetne neuronske mreže je razumijevanje bioloških neuronskih mreža te rješavanje problema na području umjetne inteligencije. [6] Da bi se razvila primjerena strategija analize podataka koriste se strukturom ljudskog mozga, stoga umjetni neuron oponaša osnovne funkcije biološkog neurona. I dalje postoje mnogi fenomeni živčanog sustava koji nisu uspješno modelirani umjetnim neuronskim mrežama, a postoje i karakteristike umjetnih neuronskih mreža koje se ne slažu s prirodnim neuronskim mrežama. Moć analize nalazi se u snazi veza između pojedinih neurona, točnije u težinama do kojih se dolazi postupkom prilagodbe koje se razvija učenjem iz skupa podataka. [6]

3.2. Osnove umjetnih neuronskih mreža

Postoji mnogo vrsta neuronskih mreža, no sve dijele dvije zajedničke karakteristike [7], a to su čvorovi (umjetni neuroni) i konekcije između njih (umjetne sinapse). Čvorovi se mogu opisati kao jedinice koje izvršavaju neku funkciju. Ta funkcija može biti jednostavna poput sumiranja ulaza, a može biti i kompleksnija (jedan čvor može u sebi sadržavati cijelu mrežu). Interakcija između čvorova određuje ponašanje cijele mreže kao cjeline. Veze određuju tokove podataka između čvorova, a koje mogu biti jednosmjerne ili dvosmjerne. [7]

Ovakvim modelom, kompleksnost pravih neurona i mozga visoko je apstrahirana. Svaki čvor dobiva ulazne podatke putem toka podataka i zatim računa funkciju koja mu je dodijeljena. Ulazni podaci su pomnoženi težinom, koja bi se dala opisati kao jačina signala kada bi se povukla paralela sa funkcioniranjem mozga. Nakon što čvor izračuna jednadžbu, rezultat te jednadžbe prolazi kroz aktivacijsku funkciju. [7] Aktivacijska funkcija određuje kako se rezultat koji je čvor izračunao prenosi dalje u mrežu. Vizualni prikaz umjetnog neurona prikazan je na slici 1, a jednostavna neuronska mreža na slici 2. Slijedi matematički zapis funkcije koju izvršava opisani neuron. Varijabla w je težina, a x je ulazni podatak:

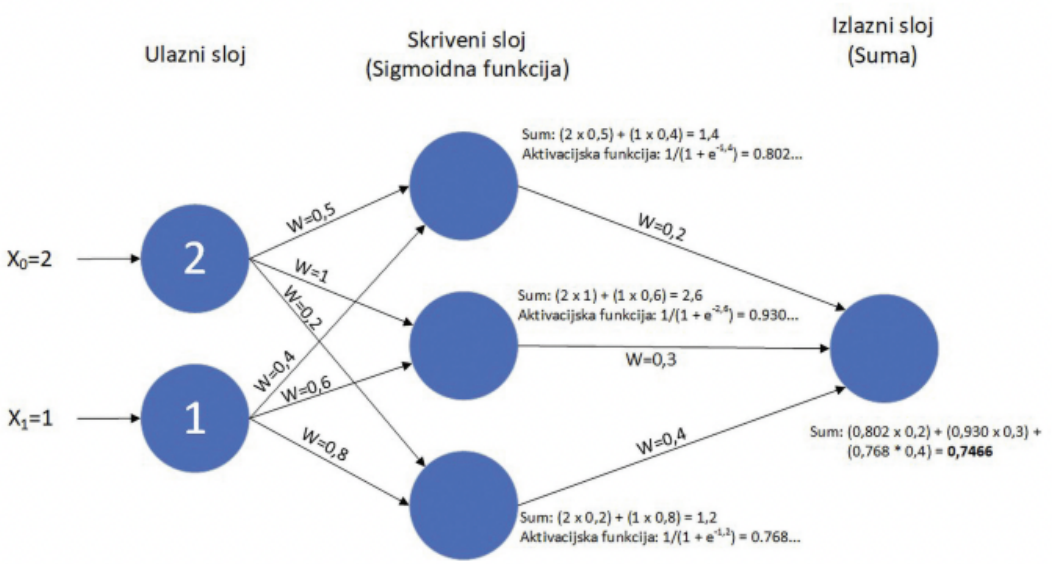
$$f(x) = w * x$$

Aktivacijske funkcije čvorova u neuronskim mrežama koriste se kako bi mreža bila nelinearna. [4] Ukoliko se ne bi koristile aktivacijske funkcije, neuronska mreža bi mogla imati samo jedan sloj i zamijeniti se logističkom regresijom. Pri konstruiranju neuronske mreže, bitno je odabrati aktivacijsku funkciju koja će rezultirati najpreciznijim modelom umjetne inteligencije. [4]

Slika 1: Umjetni neuron; preuzeto iz [7] i prevedeno

Ovisno o težini kojom je pomnožen ulazni podatak čvora, ovisi izlazna vrijednost funkcije koju neuron računa. Modificiranjem težine kojom je ulazni podatak pomnožen, moguće je prilagoditi mrežu da daje željene vrijednosti s obzirom na ulazne vrijednosti. S obzirom na to da mreža može imati milijune čvorova, ovo nije jednostavan zadatak. Kako bi se pronašle barem približno željene težine za svaki čvor, potrebno je proći kroz proces treniranja ili učenja neuronske mreže.

Postoji mnogo vrsta neuronskih mreža, pa tako i podešavanja njenih težina, no najčešći i najpoznatiji način [7] na koji se podešavaju vrijednosti težina neuronskih mreža je algoritam unazadne propagacije *(engl. backpropagation algorithm)*. Neuroni u mreži su posloženi u slo-



Slika 2: Umjetna neuronska mreža; preuzeto iz [4]

jeve, kao što je vidljivo na slici 2. Mreže propagiraju podatke iz jednog sloja prema sljedećemu prema naprijed, a greške se zatim propagiraju unazad. Mreže su uglavnom građene od više slojeva. Algoritam unazadne propagacije uz pomoć primjera ulaza koji dalju željene izlaze podešava težine kako bi one davale što bolje rezultate. [7]

Unazadna propagacija je algoritam koji je zaslužan za sposobnost neuronske mreže da uči. Model umjetne inteligencije u procesu treniranja prolazi kroz ručno zadan broj koraka. Prilikom svakog koraka, algoritam unazadne propagacije podešava težine neuronske mreže. Funkcija koja računa razliku između željenih rezultata i dobivenih rezultata naziva se funkcija gubitka. [8] Algoritam unazadne propagacije uzima gradijent te funkcije. Da bi se težine podesile na temelju izračuna gradijenta, koristi se jedan od optimizacijskih algoritama. [9] Neki od optimizacijskih algoritama su gradijentni pad, stohastički gradijentni pad, Momentum, Adagrad, Nesterov, Adadelta, Adam, Nadam, RMSprop, Adamax i sl. [10]

4. Konvolucijske neuronske mreže

U prvom potpoglavlju nalazi se uvod u način na koji funkcioniraju konvolucijske neuronske mreže, osnovne ideje iza konvolucijskih slojeva i kratka povijest. U drugom potpoglavlju sustav naziva PilotNet detaljno je opisan uz arhitekturu i otkrića Nvidijinog tima znanstvenika kako bi se pružio uvid u konvolucijsku neuronsku mrežu koja se koristi u praktičnom dijelu rada. U trećem poglavlju opisan je sustav za detekciju objekata u relanom vremenu naziva YOLOv3 koji također koristi konvolucijsku neuronsku mrežu.

4.1. Osnove konvolucijskih neuronskih mreža

Konvolucijske neuronske mreže revolucionizirale su prepoznavanje uzoraka uz pomoć neuronskih mreža. Prije nego što su konvolucijske mreže postale široko prihvaćene, većina zadataka koja je uključivala prepoznavanje uzoraka je bila izvršena ručnim izdvajanjem oblika koji su se zatim koristili kao ulazi u neuronske mreže koje se koriste za klasifikaciju. Proboj koji su učinile konvolucijske mreže je taj da su one omogućile automatsko prepoznavanje uzoraka iz podataka koji se koriste za treniranje. Iako je ovaj oblik konvolucijskih mreža u upotrebi već preko 20 godina, tek u posljednjih desetak godina je došlo do eksplozije u primjeni istih zbog javno dostupnih setova podataka i optimizacije algoritama neuronskih mreža za korištenje na grafičkim karticama, što znatno povećava brzinu treniranja i inferenciranja modela umjetne inteligencije. [11]

Arhitektura konvolucijskih neuronskih mreža je nešto drugačija od tradicionalnih neuronskih mreža. One su namijenjene prepoznavanju detalja u slikama. [12] Zbog velikog broja ulaza u mrežu što bi učinilo računanje funkcije gubitka vrlo skupom operacijom i zbog mogućnosti prepoznavanja uzoraka u slikama imaju jednu specifičnost koja ih razlikuje od klasičnih neuronskih mreža. [12] Ta specifičnost su konvolucijski slojevi.

Konvolucijski sloj primjenjuje filter na ulazne podatke (tj. sliku). [12] Filter je matrica određene veličine. Intenzitet svakog piksela u filteru podešen je algoritmom unazadne propagacije tijekom procesa učenja. Pri propagaciji unaprijed, filter je konvoluiran s ulaznim podacima. Tijekom konvoluiranja filtera s ulaznim podacima, filter se pomiče za određeni korak koji je ručno zadan. Sumi vrijednosti u matrici filtera je zatim pribrojena težina i ta suma je upisana u aktivacijsku mapu. Zahvaljujući aktivacijskoj mapi, mreža ima sposobnost prepoznavanja uzoraka u slici. Nad vrijednostima unutar aktivacijske mape je zatim primjenjena odabrana aktivacijska funkcija. Zatim se nad tom mapom primijeni novi filter odabrane veličine i opet se provodi postupak konvolucije. U novu matricu se upisuju maksimalne ili prosječne vrijednosti filtera. [12] Ovaj zadnji korak naziva se metoda sažimanja maksimalnom vrijednosti (ili prosječnom vrijednosti). [12] Vrijednosti dobivene u ovoj matrici se zatim koriste kao ulazi u neuronsku mrežu. Primjer jedne konvolucijske mreže koja je opisana u sljedećem poglavlju je vidljiv na slici 3.

4.2. PilotNet

PilotNet je naziv za konvolucijsku neuronsku mrežu koju je 2016. godine prestavio Nvidijin tim stručnjaka u svom znanstvenom radu "End to End Learning for Self-Driving Cars". [11] U njihovom radu opisana je arhitektura neuronske mreže i model umjetne inteligencije koji je istreniran koristeći tu arhitekturu. Njihov model je istreniran tako da su pikseli slike koja dolazi iz kamere montirane na prednju stranu automobila i komande za upravljanje automobila korišteni kao ulazne vrijednosti u neuronsku mrežu. Ovaj pristup se dokazao vrlo dobrim i uz relativno malo podataka dobiveni su dobri rezultati. Mreža ima sposobnost prepoznavanja obilježja poput ceste, vozila u prometu i ostalih obilježja koja su bitna za uspješno kretanje kroz promet. Mreža nije eksplicitno istrenirana kako bi prepoznavala takva obilježja već sama u procesu treniranja prepoznaje uzorke i zaključuje kako neki uzorci utječu na kut skretanja. [11] Autori argumentiraju da tradicionalni sustav u kojem je više dijelova povezano u skladnu cjelinu vjerojatno ne može funkcionirati toliko dobro koliko može sustav koji se sastoji od jedne cjeline. Razlog tog argumenta je taj da se ljudski odrađeno povezivanje dijelova sustava temelji na ručno izrađenim pravilima, dok sustav koji je jedna cjelina (jedna konvolucijska neuronska mreža) može povezivati te cjeline na temelju podataka na kojima je istreniran, a na taj način postoji mogućnost da se ostvaruju kvalitetnije postavljena interna pravila nego što bi se ona mogla postaviti ručno. Pitanja koja su postavljena na početku ovog istraživanja su [13]:

- Može li ovakav pristup biti skaliran do razine produkcijskog sustava?
- Sadrže li signali korišteni za trening koji su dobiveni uparivanjem ljudskih komanda upravljanja sa slikama ceste ispred vozila dovoljno informacija kako bi model naučio samostalno izvršavati zadatak vožnje?
- Može li ovakav pristup s kraja na kraj *(engl. end-to-end)* parirati i postati bolji od sustava koji koriste tradicionalnu arhitekturu?

Težine mreže su istrenirane na način kako bi se smanjila srednja kvadratna pogreška između komande upravljanja između čovjeka i one koju je predvidjela mreža. Arhitektura konvolucijske mreže prikazana je na slici 3. Sastoji se od devet slojeva, uključujući sloj normalizacije, pet konvolucijskih slojeva, i tri potpuno povezana sloja. Za potrebe Nvidijinog rada, slika je prebačena u YUV prostor boja prije nego je proslijeđena mreži.

Prvi sloj mreže odrađuje normalizaciju slike. Normalizacija omogućuje arhitekturi da bude fleksibilna kako bi mogla prihvaćati razne podatke. Konvolucijski slojevi su dizajnirani kako bi odradili izdvajanje oblika (prepoznavanje uzoraka) i odabrani su empirijskim pristupom kroz niz eksperimenata s razno postavljenim konfiguracijama slojeva. Prva tri konvolucijska sloja koriste filter veličine 5x5 i 2x2 veličinu koraka, a sljedeća dva sloja koriste filter veličine 2x2 i korak veličine 1x1. Ovakva arhitektura ima oko 27 milijuna konekcija i 250 000 parametara. [11]

Podaci uz pomoć kojih je mreža istrenirana prikupljeni su na raznim cestama i u raznim vremenskim uvjetima. Kako bi se model umjetne inteligencije znao vratiti iz situacija u kojima je

vozilo sišlo s putanje, na setu slika je izvršena augmentacija podataka. Dio slika i komanda za skretanje je modificiran kako bi se simulirali uvjeti u kojima se vozilo nalazi izvan dobre putanje i želi se vratiti na pravilnu putanju. Broj augmentiranih podataka je pažljivo odabran kako ne bi došlo do neželjenih posljedica. [11]

Testiranje modela odrađeno je u simulatoru i na cesti u realnim uvjetima. Kako bi se dobila metrika kvalitete, Nvidijin tim je koristio jednostavnu formulu za izračun:

$$autonomija = (1 - \frac{intervencije * 6}{vrijeme}) * 100$$

Varijabla intervencije je broj intervencija u razdoblju testiranja, a varijabla vrijeme označava duljinu testiranja u sekundama. Broj šest, konstanta, označava vrijeme u sekundama koje je potrebno vozaču da vrati vozilo na pravi smjer kako bi se oporavilo od greške. [11] Ova formula se koristi i u ovome radu kako bi se procjenile performanse modela.

Ovim radom Nvidijina istraživačka skupina potvrdila je kako je konvolucijskom mrežom moguće doći do sustava koji će u potpunosti sam naučiti slijediti cestu bez ručne dekompozicije problema na planiranje putanje, detekciju objekata i ostale dijelove tradicionalnog sustava za autonomnu vožnju. Testiranjem u raznim uvjetima došli su do razine autonomije od 98% koristeći prethodno navedenu formulu. [11]

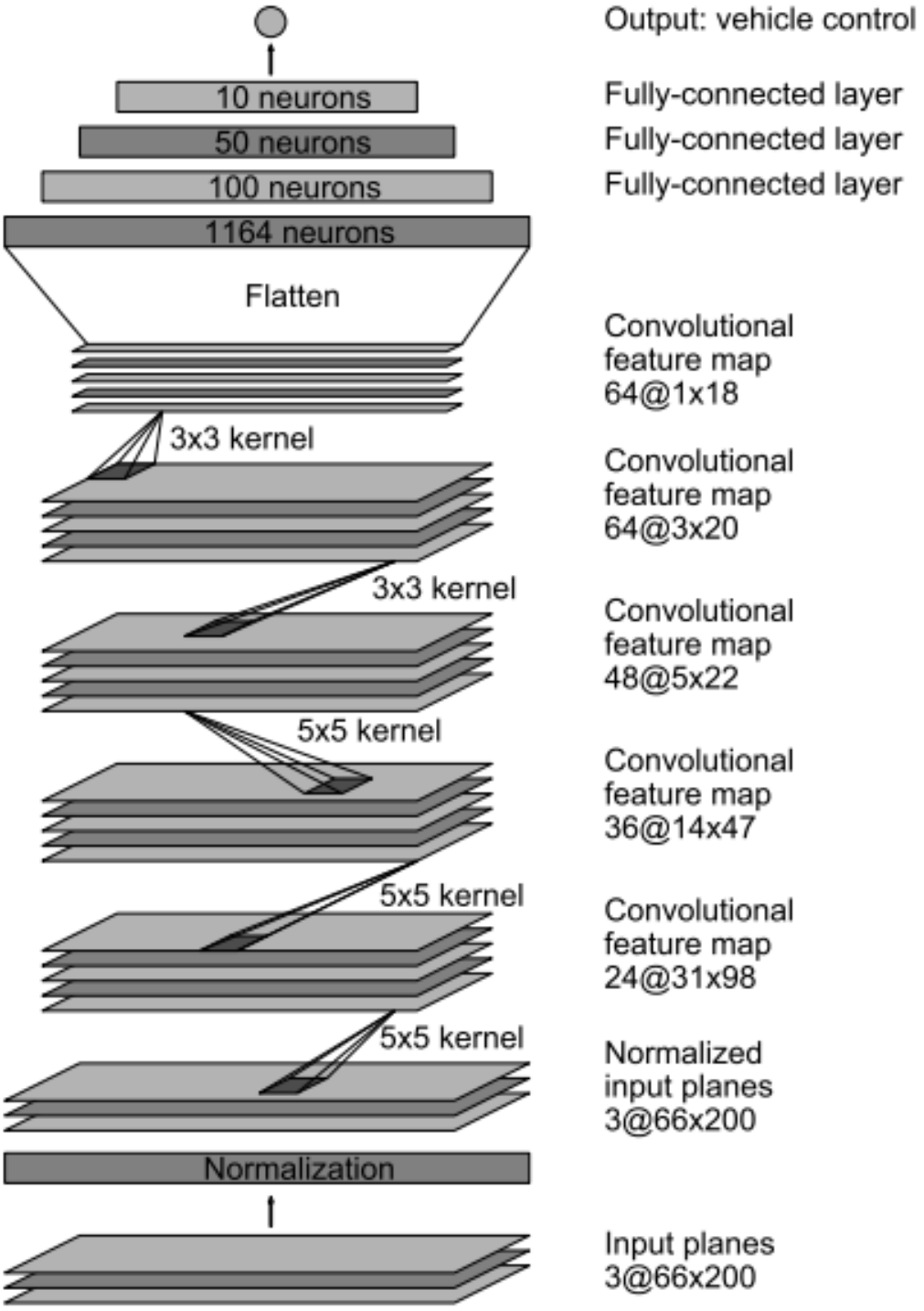
Nakon ovog znanstvenog rada, Nvidijin istraživački tim je 2020. godine objavio još jedan znanstveni rad u kojem su opisani eksperimenti i nastavak rada na PilotNet-u. Nvidijin tim se proširio i aktivno radio na poboljšavanju sustava. Sustav je i dalje daleko od sustava koji se u današnjici koriste u produkciji. [13]

4.3. YOLOv3

YOLO je akronim za „You only look once“. YOLOv3 je treća iteracija originalnog YOLO modela predstavljenog u istraživačkom radu 2015. godine [14]. YOLO arhitektura je specifična po tome što za razliku od većine drugih modernih pristupa detekciji objekata slika prolazi kroz mrežu samo jednom kako bi se objekti detektirali. [15]

Ovaj sustav koristi konvolucijsku neuronsku mrežu za detekciju objekata. Ima sposobnost detektiranja više objekata na jednoj slici, njihovih klasa i lokacija na slici. Ulazna slika se dijeli na ćelije *(engl. grid cells)*. Nakon što je podijeljena na ćelije, te ćelije su zadužene za detekciju objekata koji se nalaze u njima. Za svaku ćeliju se zatim predviđa koji objekt bi se u njoj mogao nalaziti (ako se uopće nalazi ondje) i predviđaju se granični okviri *(engl. bounding box)*. Svakom graničnom okviru je pridružena vjerojatnost da se neka klasa nalazi unutar okvira i najveće vjerojatnosti se uzimaju u obzir. Granični okviri također imaju svoj centar (izražen u x i y koordinatama sa slike), širinu i visinu okvira. Ovo je vrlo pojednostavljen opis rada YOLO sustava. [16]

YOLOv3 se sastoji od 53 konvolucijska sloja (koja se također nazivaju i Darknet-53) i 53 potpuno povezana sloja. Detekcije se odvijaju na slojevima 82, 94 i 106. Svaki od tih slojeva



Slika 3: Arhitektura Nvidijine PilotNet arhitekture; preuzeto iz [11]

izvršava poduzorkovanje *(engl. undersampling)* nad ulaznim podacima. Uzme li se za primjer slika rezolucije 416x416 kao primjer, na 82. sloju ulazna slika će se smanjiti na 13x13, na 94. sloju na 16x16, a na 106. sloju na 32x32 piksela. Kako bi se simuliralo poduzorkovanje, korištena je veličina koraka 32 na 82. sloju, korak veličine 16 na 94. sloju i korak veličine 8 na 106. sloju. 82. sloj je zadužen za detektiranje velikih objekata, 94. sloj za detektiranje objekata srednje veličine, a 106. sloj za detektiranje malih objekata. Bitno je napomenuti da,

zbog korištenja koraka veličine 32, ulazna slika mora biti u rezoluciji kojoj su visina i širina slike brojevi djeljivi sa 32. [16]

Na internetu su javno dostupni modeli bazirani na YOLOv3 arhitekturi. U ovom radu su korišteni modeli istrenirani na COCO setu podataka[1]. COCO je set podataka koji je 2014. godine objavio Microsoft. Sadrži 2.5 milijuna označenih objekata u 328 tisuća slika. [17] Modeli su dostupni na web stranicama autora YOLO arhitekture, Josepha Redmona [2]. Konkretno, korištena je *tiny* verzija v3 modela zbog ograničenih resursa na testnom računalu.

[1] https://cocodataset.org
[2] https://pjreddie.com/darknet/yolo/

5. Razvoj autonomnog sustava

Za razvoj autonomnog sustava potrebno je proći kroz nekoliko faza razvoja koje su opisane u sljedećim poglavljima. Faze je nužno provesti redoslijedom kojim su opisane jer se u svakoj sljedećoj fazi koriste izlazne vrijednosti prethodne faze. Na kraju faze prikupljanja podataka rezultat su sirovi podaci koje je potrebno obraditi u sljedećoj fazi. Obrađeni podaci se zatim balansiraju i u fazi treniranja koriste kako bi se težine modela umjetne inteligencije podesile za izvršavanje željenog zadatka. Na kraju se dobiveni model koristi kao ulaz u algoritam za vožnju.

5.1. Prikupljanje podataka

Prikupljanje podataka obavljeno je igranjem videoigre Grand Theft Auto V u trajanju od otprilike petnaest sati. Potrebno je napomenuti da je vrijeme u igri bilo postavljeno na sunčano i čisto nebo kako bi se dobili bolji rezultati tijekom treniranja mreže. Podaci se prikupljaju vožnjom automobila kroz virtualni svijet videoigre dok skripta napisana u programskom jeziku Python prikuplja podatke u pozadini. Program se odvija u beskonačnoj petlji. U svakoj iteraciji petlje uzima se snimak zaslona i stanje igrače palice. Za pristup stanju igrače palice korištena je biblioteka Inputs. Iz stanja igrače palice prikupljaju se samo jačina kojom su pritisnuti gas i kočnica te kut pod kojim vozilo skreće. Duljina procesa prikupljanja podataka ovisi o procesorskoj snazi računala. Na testnom računalu u koje je ugrađen Intel Core i3 8100 procesor moguće je napraviti u prosjeku pet ili šest snimaka zaslona u sekundi. To će rezultirati duljim procesom prikupljanja podataka ukoliko se kao cilj postavi određeni broj uzoraka koji se želi prikupiti. Za potrebe ovog rada prikupljeno je 225 000 uzoraka. Uzorci su prikupljeni u rezoluciji od 480x270 piksela i RGB prostoru boja. Iako je u ovom radu model treniran koristeći slike u rezoluciji od 160x120 piksela i korišten je sivi prostor boja, nije naodmet spremiti podatke u višoj kvaliteti kako bi se u slučaju daljnjeg istraživanja ove teme pokušao istrenirati model sa višom rezolucijom ulaznih slika ili koristeći RGB prostor boja umjesto sivog.

5.2. Priprema podataka

Kako bi proces učenja prošao što lakše, podatke je prethodno potrebno pripremiti. Loš set podataka rezultira time da neuronska mreža ne može naučiti ono što je zadano. Najbitniji korak u pripremi podataka je balansiranje seta podataka kako bi sadržavao otprilike podjednak broj svih klasa koje se nalaze u setu podataka. Klase podataka su sve moguće vrijednosti koje se pojavljuju u setu podataka kao primjer izlazne vrijednosti. U ovom konkretnom slučaju, set podataka se balansira na temelju kuta skretanja automobila čije su moguće vrijednosti prirodni brojevi u rasponu od -32 768 do 32 767 što znači da set podataka sadrži 65 536 klasa podataka. Više o balansiranju podataka opisano je u poglavlju o balansiranju podataka.

5.2.1. Pridruživanje i obrada podataka

Podaci su u fazi prikupljanja prikupljani u niz datoteka u NPY formatu. Nova datoteka je kreirana svaki put nakon što je prikupljeno 500 uzoraka. S obzirom na to da računalo uzima pet do šest uzoraka po sekundi, nova datoteka nastane otprilike svakih minutu i pola. NPY je format koji koristi biblioteka Numpy [18] za spremanje podataka.

Kako bi se olakšalo daljnje rukovanje podacima, sve datoteke NPY formata objedinjene su u jednu datoteku HDF5 (Hierarchical Data Format 5) formata. Za lakše rukovanje formatom korištena je H5PY biblioteka. [19] Glavni razlog korištenja ovog formata je to što omogućava jednostavno čitanje uzoraka koji se nalaze u datoteci sa diska umjesto da se cijela datoteka učita u radnu memoriju računala. Ovo je sporiji pristup, no pojednostavljuje rad s podacima u slučaju kada je datoteka koja sadrži set podataka prevelika da bi se učitala u memoriju računala. Osim toga, nudi rad s tipovima podataka (što inače nije dostupno u Pythonu), spremanje podataka u hijerarhijsku strukturu, spremanje metapodataka, spremanje podataka u komadima, kompresiju podataka i još mnogo toga. Format se koristi u akademskim istraživanjima, vladinim, vojnim i nacionalnim organizacijama, raznim industrijama (naftna, silikonska, financijska, aeronautička...), znanstvenim poljima (astronomiji, medicini, genomiki, fizici...). [19]

Prije nego što su podaci spremljeni u novu datoteku HDF5 formata, svaka snimka zaslona smanjena je na rezoluciju 160x120 piksela i prebačena je u sive tonove *(engl. grayscale)* kako bi neuronska mreža kasnije mogla lakše učiti iz tih podataka. Što je veći set ulaznih podataka, mreža će teže pronaći odgovarajuće težine, stoga je smanjenje rezolucije i prebacivanje iz RGB prostora boja u prostor u kojem se koriste samo tonovi sive boje logičan korak. Kako bi se opisala boja jednog piksela, RGB format koristi tri boje u rasponu od 0 do 255, dok prostor sivih boja koristi samo intenzitet svjetla u rasponu od 0 do 255. Osim snimaka zaslona, normalizirane su i vrijednosti koje predstavljaju stanje igrače palice (gas, kočnica i kut skretanja). Gas i kočnica su izraženi kao cijeli brojevi u rasponu od 0 do 255. Ukoliko je došlo do kočenja, kao izlazna vrijednost kretanja unaprijed ili unazad razina kočenja je normalizirana kako bi bila decimalni broj u rasponu od -1 do 0. Ukoliko je došlo do ubrzanja, sila ubrzanja je normalizirana i izražena kao vrijednost u rasponu od 0 do 1. Kut skretanja je u procesu prikupljanja podataka prikupljen kao vrijednost između -32 768 do 32767, a u procesu obrade podataka je normaliziran da bude u rasponu od -1 do 1.

Na internetu je moguće pronaći setove podataka koji su slični onome prikupljenome u ovom radu. Jedan takav set podataka [1] je nadodan na set podataka koji je stvoren nakon prethodno opisanog postupka. Ovaj set također koristi slike rezolucije 160x120 u sivim tonovima.

5.2.2. Balansiranje podataka

Kako se tijekom procesa treniranja težine ne bi podesile tako da je model pristran vožnji unaprijed, lijevo ili desno, potrebno je balansirati podatke. Kut skretanja normaliziran je da bude decimalan broj u rasponu od -1 do 1 u procesu obrade podataka. Kako postoji velik broj klasa, podatke nije moguće balansirati prema klasama. Umjesto toga kao klase se definiraju

[1] https://github.com/Alzaib/Autonomous-Self-Driving-Car-GTA-5

kategorije. U rasponu od -1 do 1 definirano je 26 kategorija s jednakom veličinom intervala i u njih su dodani uzorci iz seta podataka koji pripadaju intervalima kategorija. Broj uzoraka prema kategorijama u obrađenom setu podataka prikazan je na slici 4. Nakon što je iz svake kategorije nasumično odabrano 3000 uzoraka (ili manje ako ih je manje dostupno) dobiven je histogram skretanja koji je vidljiv na slici 5.

Na slici 5 je vidljivo da postoji nešto više skretanja u lijevo što bi moglo rezultirati malom pristranošću da model skreće lijevo kada ne bi trebao. Proces treniranja i rezultati su pokazali da to nije slučaj, no u slučaju da se to dogodilo, postoje tehnike koje rješavaju problem nebalansiranog seta podataka kao što su augmentacija podataka ili još jedno brisanje uzoraka iz klase u kojoj se nalazi previše uzoraka. Postoje i tehnike kojima se proces učenja može prilagoditi kako bi se težine što bolje podesile čak i kada se koristi lošije balansiran set podataka. Jedna od takvih tehnika je dodjeljivanje veće težine klasama koje su manje zastupljene.

5.3. Treniranje modela

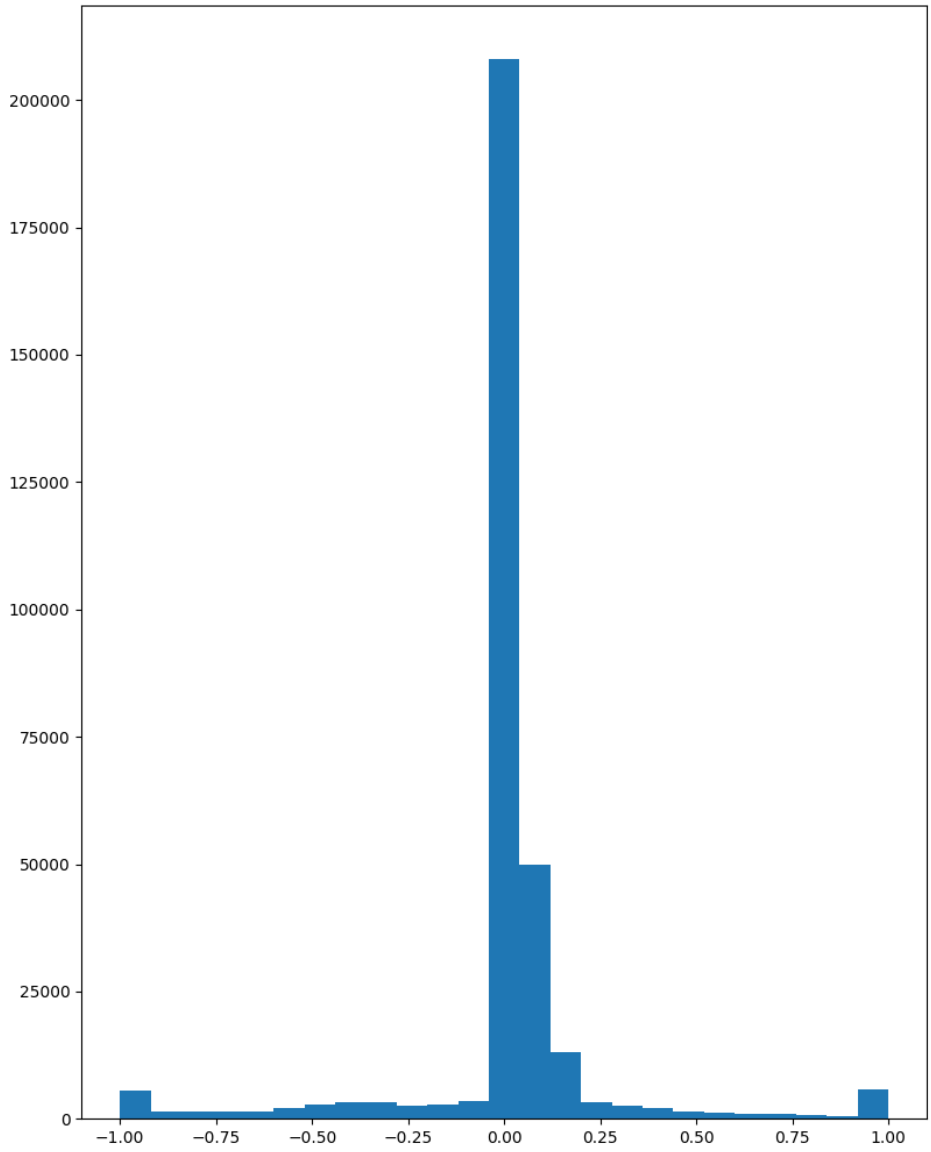
Za treniranje modela korištena je biblioteka Keras koja djeluje kao sučelje za biblioteku Tensorflow. Kako bi se smanjilo vrijeme treniranja modela, korištena je besplatna platforma Google Colab na kojoj je dostupno interaktivno okružje u Pythonu. Platforma nudi besplatan pristup grafičkim karticama, a s obzirom na to da su moderne biblioteke optimizirane kako bi iskoristile snagu grafičkih kartica za paralelizaciju algoritama, pristup grafičkoj kartici je gotovo neophodan jer smanjuje vrijeme koje je potrebno za trening. Testno računalo također ima dobru grafičku karticu, no Google Colab nudi pristup i boljem hardveru.

Podaci su prvo učitani sa platforme Google Drive gdje se nalazi balansirani set podataka. Zatim je nad njima izvršena augmentacija podataka. Svaka slika iz seta podataka je okrenuta i kut skretanja je okrenut kako bi se povećala veličina seta podataka. Podaci su zatim nasumice izmiješani kako bi se poboljšao proces učenja i podijeljeni su u set za treniranje i set za validaciju. Ukupan set podataka sadrži 51 060 uzoraka. Nakon augmentacije taj broj je duplo veći pa raste na 103 220 uzoraka. Nakon podjele na set za treniranje (80% podataka) i set za validaciju (ostalih 20% podataka), set za treniranje sadrži 82576 uzoraka, a set za validaciju 20644 uzoraka.

Nakon što su podaci pripremljeni, stvoren je model koji se temelji na Nvidijinoj PilotNet arhitekturi:

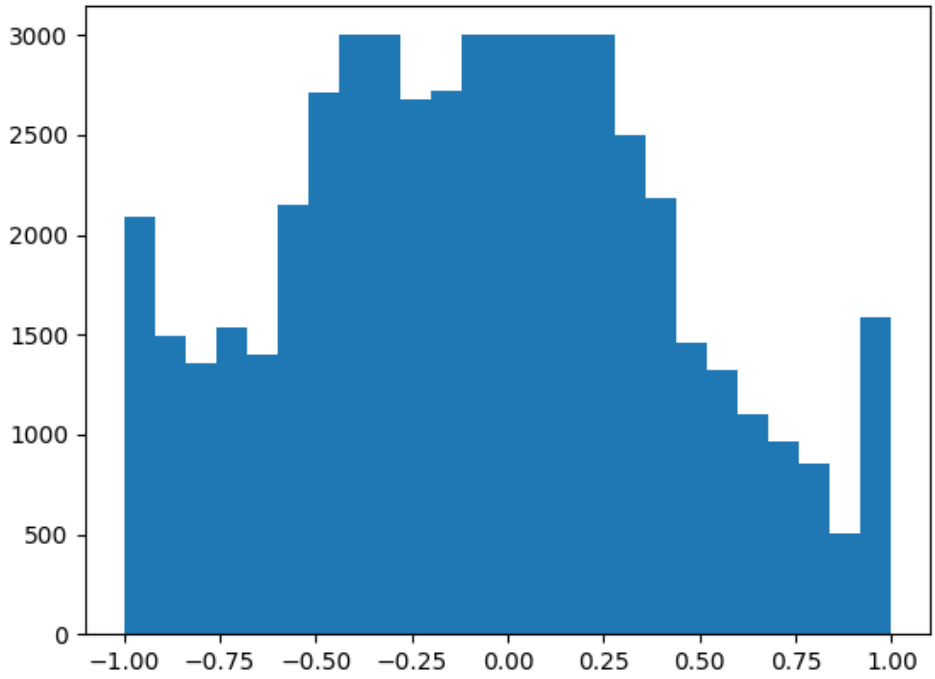
Isječak kôda 1: Keras implementacija Nvidijine PilotNet arhitekture

```
1 model = Sequential()
2 model.add(Conv2D(24, (5, 5), activation='elu', strides=(2, 2), input_shape=(160,
   120, 1)))
3 model.add(Conv2D(36, (5, 5), activation='elu', strides=(2, 2)))
4 model.add(Conv2D(48, (5, 5), activation='elu', strides=(2, 2)))
5 model.add(Conv2D(64, (3, 3), activation='elu'))
6 model.add(Conv2D(64, (3, 3), activation='elu'))
7
8 model.add(Flatten())
```



Slika 4: Histogram nebalansiranog seta podataka; vlastita izrada

```
9  model.add(Dense(100, activation='elu'))
10 model.add(Dense(50, activation='elu'))
11 model.add(Dense(10, activation='elu'))
12 model.add(Dense(2))
13
14 model.compile(loss='mean_squared_error', metrics=["accuracy"],  optimizer=Adam(
    learning_rate=0.001))
```



Slika 5: Histogram balansiranog seta podataka; vlastita izrada

Hiperparametri koji su korišteni pri treniranju modela prikazani su u tablici 1. Ovo je samo jedna od mogućih postavki hiperparametara do koje se došlo nakon nekoliko iteracija treniranja modela kako bi se ustanovilo kako koji hiperparametar utječe na izlazne metrike modela.

Tablica 1: Hiperparametri neuronske mreže; vlastita izrada

Parametar	Vrijednost
Broj epoha	25
Veličina serije podataka za treniranje	300
Veličina serije podataka za validaciju	100
Stopa učenja	0.001

Pokazalo se da mreža ima sklonost pretreniravanju *(engl. overfitting)*. Da bi se to izbjeglo dodana je dinamična augmentacija podataka koja se odvija prilikom učitavanja uzoraka prije nego što ulaze u cjevovod za treniranje. Tehnike koje se koriste za augmentaciju podataka su zamućivanje slike, mijenjanje svjetline slike, zumiranje slike i zakretanje slike. Bitno je naglasiti da se augmentacija podataka odvija samo nad podacima koji se koriste za treniranje.

5.4. Algoritam za vožnju

Slijedi pseudokod algoritma koji upravlja vozilom:

Isječak kôda 2: Pseudokod algoritma za vožnju

```
while True:
      slika_zaslona = zgrabi_sliku_zaslona()
      kut_skretanja, ubrzanje = pilotnet_model.predikcija(slika_zaslona)
      objekti = yolo_model.predikcija(slika_zaslona)

      brzina = dohvati_brzinu()
      if brzina < 40:
        ubrzanje = 1
      else:
         ubrzanje = 0

      if potrebno_kociti(objekti):
        ubrzanje = -1

      upravljaj_automobilom(kut_skretanja, ubrzanje)
```

Stvarna implementacija se pomalo razlikuje, no izvršava zadatke u ovom slijedu. Program se izvršava u beskonačnoj petlji. Na početku svake iteracije uzima se slika zaslona koja se zatim predaje modelu baziranom na PilotNet arhitekturi, a zatim i modelu baziranom na YOLO arhitekturi kako bi se detektirali objekti. Dohvaća se trenutna brzina iz memorije računala i ovisno o trenutnoj brzini, gas se dodaje ili se ne dodaje. Detektirani objekti se predaju funkciji koja na temelju pravokutnika čije granice označavaju mjesto gdje se objekt nalazi na ekranu procjenjuje je li potrebno kočiti ovisno o tome koja je širina pravokutnika. Npr. ako je pravokutnik širine 100 piksela i označava automobil, vrlo vjerojatno je potrebno zakočiti kako ne bi došlo do sudara jer se ispred vozila nalazi automobil. Na kraju se kut upravljanja i procjenjena količina ubrzanja (ili kočenja) predaju funkciji koja simulira igraču palicu i šalje te upute igri.

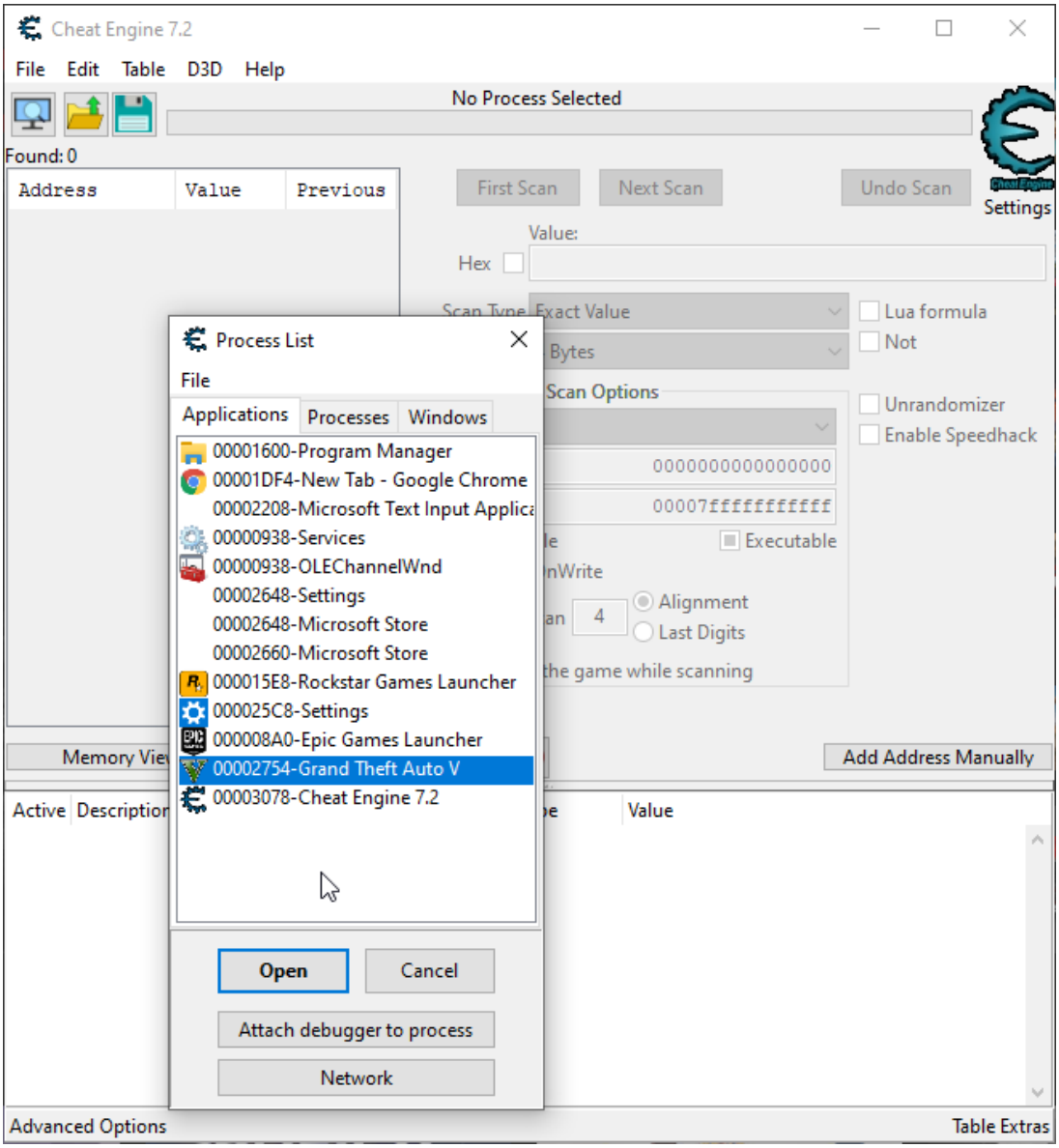
Brzina je ograničena na 40km/h za potrebe testiranja jer se uglavnom desi da vozilo krene voziti prebrzo. Od mreže zapravo nije bilo niti očekivano da nauči kontrolirati gas i kočnicu vrlo dobro jer je takvu informacije nemoguće zaključiti iz slike čak i čovjeku, no svejedno je zanimljiv eksperiment. Kako bi se izračunala brzina vozila unutar igre, potrebno je pronaći memorijske adrese unutar procesa videoigre na kojima se nalaze koordinate igrača u virtualnom prostoru.

Dohvaćanje adresa iz memorije računala odrađeno je uz pomoć besplatnog programa Cheat Engine. Kako bi se dobila ispravna adresa u memoriji, potrebno je u Cheat Engineu prvo odabrati proces u prozoru koji se otvori nakon klika na ikonu ekrana i povećala u gornjem lijevom kutu (slika 6), a zatim kliknuti na "add address manually" i podesiti postavke kao što je prikazano na slici 7. Na dobivenoj memorijskoj adresi nalazi se X koordinata unutar videoigre. Povećanjem memorijske adrese za 4 bajta, dobiva se Y koordinata, a povećanjem adrese Y koordinate za 4 bajta dobiva se adresa Z koordinate. U svakoj iteraciji algoritma za vožnju računa se udaljenost od točke na kojoj se igrač nalazio u prošloj iteraciji prema formuli:

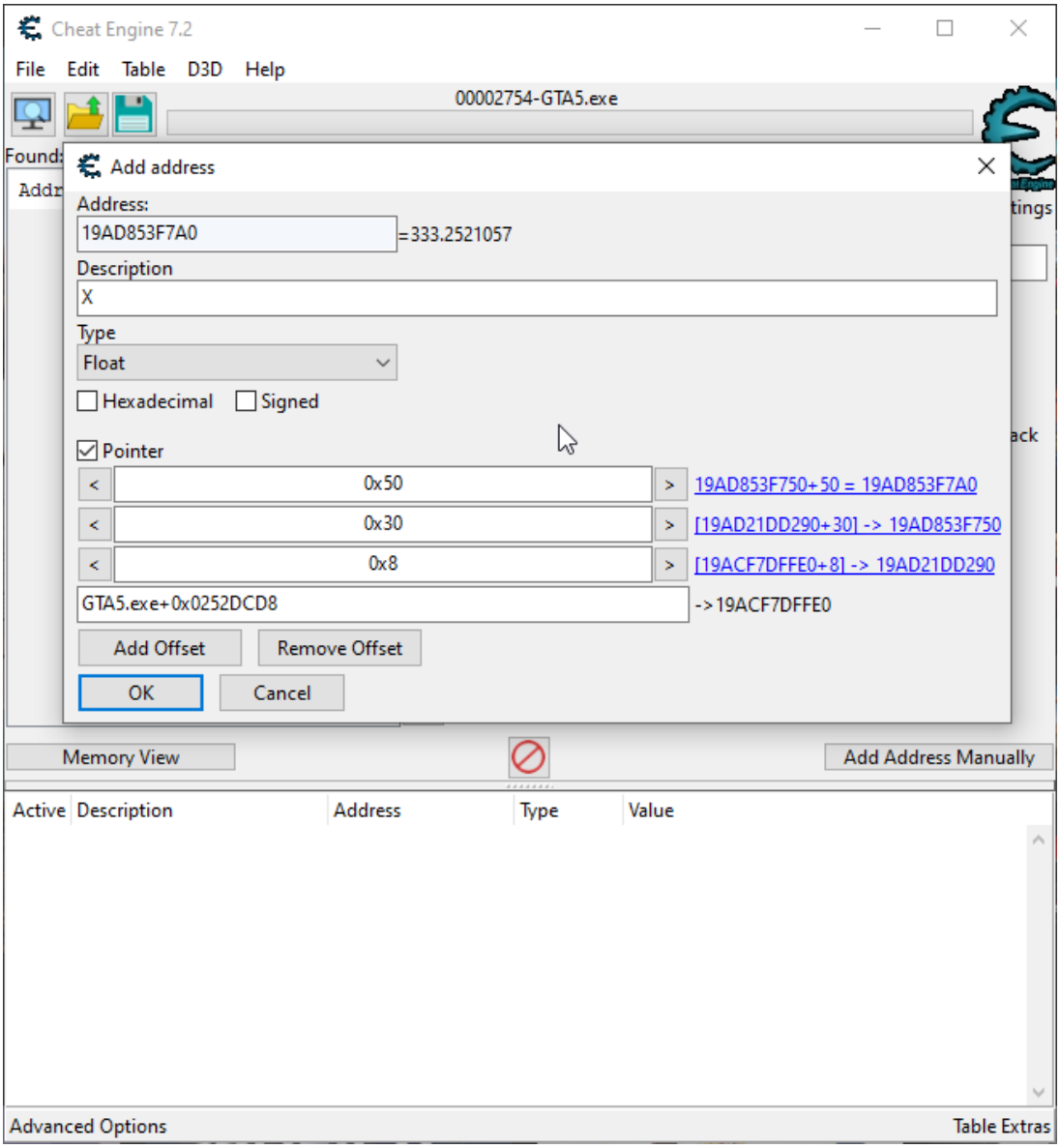
$$brzina = ((x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2) * 5 * 3.6$$

Nakon što se izračuna udaljenost između dvije točke, broj se množi sa pet jer se distanca računa otprilike pet puta po sekundi. Ovo je u praksi bolje zamijeniti sa brojem iteracija u sekundi koje sustav može izvršavati kako bi se došlo do nešto preciznijeg izračuna. Na kraju se sve množi sa 3.6 kako bi se iz metara u sekundi prebacilo u kilometre na sat.

Memorijske adrese se mijenjaju gotovo svakim ažuriranjem videoigre. Verzija korištena u ovom radu je Grand Theft Auto V 1.57.



Slika 6: Odabir procesa videoigre u programu Cheat Engine; vlastita izrada



Slika 7: Pronalaženje memorijske adrese X koordinate u programu Cheat Engine; vlastita izrada

6. Rezultati

Kako je set podataka na temelju kojeg je model istreniran uglavnom sadržavao uzorke sa autoceste, model je testiran na autocesti. Testiranja su obavljena na praznoj i prometnoj autocesti kako bi se procijenilo koliko promet utječe na preciznost modela.

Jedanaest minuta vožnje autocestom uz promet je prouzročilo četiri intervencije, dok je u čak gotovo 22 minute vožnje bez prometa bila potrebna samo jedna intervencija. Intervencije su bile odrađene jedino u slučajevima kada je model bio u opasnosti od kritičnog sudara.

Koristeći formule koje su korištene u Nvidijinom istraživačkom radu za mjerenje autonomnosti (već opisano u PilotNet poglavlju) [11], dobivene su sljedeće metrike:

Vožnja autocestom bez prometa:

$$autonomija = (1 - \frac{1 * 6}{1300}) * 100 = 99.54\%$$

Vožnja autocestom uz promet:

$$autonomija = (1 - \frac{4 * 6}{660}) * 100 = 96.36\%$$

Vožnja uz promet očekivano smanjuje preciznost modela. Autonomnost od preko 90 posto je sasvim zadovoljavajući početni rezultat na kojem se može još dosta raditi. Tijekom testiranja je uočeno da je model u stanju pratiti cestu i donekle izbjegavati objekte. Nažalost, testno računalo ima premalo snage kako bi istovremeno inferenciralo YOLO model i PilotNet model. Inferencija se odvija dva do tri puta u sekundi korištenjem najslabijeg javno dostupnog YOLO modela [1]. Najslabiji model ne daje zadovoljavajuće rezultate, no jedan izvor[2] na internetu potvrđuje da je ovo problem koji se može riješiti korištenjem boljih modela za što bi bila potrebna snažnija grafička kartica. Korištenjem boljeg modela za detekciju objekata, sustav bi imao bolju sposobnost uočavanja objekata s kojima se ne smije sudariti. S druge strane, uočljivo je da PilotNet model prati oznake na cesti kako bi održao smjer. Najveća poteškoća proizlazi iz toga da model nije svjestan koja cestovna traka je njegova i zbog toga često završi u suprotnoj traci. Sljedeća veća poteškoća je da ponekad ograde koje se nalaze uz cestu vjerojatno prepoznaje kao cestovnu traku što uzrokuje da stalno želi voziti (i tako grebati automobil) uz tu ogradu.

Još jedan problem je to što model nema zadani cilj i prema tome uopće ne zna u kojem smjeru je potrebno ići na raskrižjima. Ponekad uspije odabrati neki smjer, no često se dogodi i da malo skreće u jednu stranu, pa malo skreće u drugu stranu, da bi se na kraju zabio u neki objekt koji se najčešće nalazi između dvije ceste.

Video materijali sustava u vožnji javno su dostupni putem video servisa Youtube[3][4].

[1] https://pjreddie.com/darknet/yolo/
[2] https://www.youtube.com/watch?v=nJKaj59GePk
[3] https://www.youtube.com/watch?v=hFddSH0PRrU
[4] https://www.youtube.com/watch?v=QJRZxt-RscI

7. Zaključak

Ovim radom prikazan je jedan od načina rješavanja problema autonomne vožnje koji odstupa od tradicionalnih modularnih sustava za autonomnu vožnju. I sam Nvidijin istraživački tim u svom posljednjem znanstvenom radu 2020. godine navodi kako ovakav sustav nije spreman za produkciju. Sustavi u produkciji imaju daleko više sigurnosnih mehanizama i pripremljeni su na veći broj različitih scenarija. [13] Svejedno, PilotNet je sustav koji pruža zanimljivu i ambicioznu perspektivu. Namjena mu se može pronaći u sustavu za autonomnu vožnju kao karika koja daje svoj prijedlog za akciju koju bi vozilo trebalo izvršiti, a ljudski stvorena pravila mogu uzeti taj prijedlog u obzir. Kako bi se sustav poboljšao i PilotNet zadržao u njegovoj jezgri, potrebno mu je dodati module čime se odbacuje početna ideja da ovakva neuronska mreža može zamijeniti tradicionalnu modularnu arhitekturu. U ovom radu dodana je detekcija objekata, no bilo bi potrebno dodati gotovo sve module koje ima i tradicionalni sustav kako bi ga se dovelo do razine produkcije.

S obzirom na to da se sustav razvija u okolišu koji je najviše zanimljiv adolescentima (Grand Theft Auto V jedna je od najpopularnijih videoigara na svijetu), ovakav pristup rješavanju problema autonomne vožnje može poslužiti kao primjer učenja kroz igru, gamifikacije. Umjetna inteligencija je znanost iza čije se površine krije mnogo matematike koja može zastrašiti i obeshrabriti ulazak novih kadrova u ovo polje računalnih znanosti, a stručnjaka nedostaje na globalnoj razini. Neki autori navode kako duboko poznavanje matematike i algoritama nije potrebno. Moderne biblioteke visoko apstrahiraju interakciju s neuronskim mrežama. Duboko poznavanje mehanizama je svakako potrebno za istraživački rad, no za praktičnu primjenu je dovoljno imati intuiciju o tome što se krije unutar neuronske mreže.

Buduća istraživanja na temu ovog rada mogu uključiti modifikaciju PilotNet mreže kako bi predviđala putanju automobila umjesto kuta skretanja, dodavanje modula algoritmu za vožnju u vidu detekcije linija na cesti, usmjeravanja vozila u željenim smjerovima putem GPS-a, ponašanja u skladu s prometnim znakovima... Autonomna vožnja je problem na kojem se rješenje uvijek može poboljšati, tako da prostora za daljnje istraživanje nikako ne nedostaje.

Popis literature

[1] H. Kinsley. (2018.). „Python Plays : Grand Theft Auto V," YouTube, adresa: https://www.youtube.com/playlist?list=PLQVvvaa0QuDeETZEOy4VdocT7TOjfSA8a (pogledano 15. 9. 2021.).

[2] C. F. Kerry i J. Karsten. (2017.). „Gauging investment in self - driving cars," Brookings, adresa: https://www.brookings.edu/research/gauging-investment-in-self-driving-cars%20/ (pogledano 15. 9. 2021.).

[3] O. G. Yalçın. (2020.). „4 reasons why you should use google colab for your next project," Medium, adresa: https://towardsdatascience.com/4-reasons-why-you-should-use-google-colab-for-your-next-project-b0c4aaad39ed (pogledano 15. 9. 2021.).

[4] T. Kramberger, B. Nožica, I. Dodig i D. Cafuta, „Pregled tehnologija u neuronskim mrežama," pregledni rad, Tehničko veleučilište u Zagrebu, Zagreb, HR, 2019.

[5] B. D. Bašić, M. Čupić i J. Šnajder. (2008.). „Umjetne neuronske mreže," adresa: http://degiorgi.math.hr/~singer/ui/ui_1415/UI_12_UmjetneNeuronskeMreze[1].pdf (pogledano 18. 9. 2021.).

[6] S. Dumančić, „Neuronske mreže," diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Osijek , HR, 2014.

[7] C. Gershenson, *Artificial neural networks for beginners*, 2003. arXiv: cs/0308031 [cs.NE].

[8] C. Pere. (2020.). „What are loss functions?" Youtube, adresa: https://towardsdatascience.com/what-is-loss-function-1e2605aeb904 (pogledano 15. 9. 2021.).

[9] J. Starmer. (2019.). „Gradient descent, step-by-step," Youtube, adresa: https://www.youtube.com/watch?v=sDv4f4s2SB8 (pogledano 15. 9. 2021.).

[10] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016. arXiv: 1609.04747 [cs.LG].

[11] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao i K. Zieba, *End to end learning for self - driving cars*, 2016. arXiv: 1604.07316 [cs.CV].

[12] J. Starmer. (2021.). „Neural networks part 8: Image classification with convolutional neural networks," Youtube, adresa: https://www.youtube.com/watch?v=HGwBXDKFk9I (pogledano 15. 9. 2021.).

[13] M. Bojarski, C. Chen, J. Daw, A. Değirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, L. Jackel i dr., *The nvidia pilotnet experiments*, 2020. arXiv: 2010.08776 [cs.CV].

[14] J. Redmon, S. K. Divvala, R. B. Girshick i A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, 2015. arXiv: 1506.02640 [cs.CV].

[15] K. Alderliesten. (2020.). „Yolov3 — real-time object detection," Medium, adresa: https://medium.com/analytics-vidhya/yolov3-real-time-object-detection-54e69037b6d0 (pogledano 15. 9. 2021.).

[16] V. Sichkar. (2020.). „Introduction into yolo v3," Youtube, adresa: https://www.youtube.com/watch?v=vRqSO6RsptU (pogledano 15. 9. 2021.).

[17] T. Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick i P. Doll ar, *Microsoft COCO : Common Objects in Context*, 2015. arXiv: 1405.0312 [cs.CV].

[18] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke i T. E. Oliphant. (2020.). „Array programming with NumPy," adresa: https://doi.org/10.1038/s41586-020-2649-2 (pogledano 15. 9. 2021.).

[19] The HDF Group. (2000-2010.). „Hierarchical data format version 5," The HDF Group, adresa: http://www.hdfgroup.org/HDF5 (pogledano 15. 9. 2021.).

Popis slika

Popis tablica

Popis isječaka koda