Upravljanje zrakoplovom putem neuronske mreže i genetskog algoritma

Premuš, Bojan

VELEUČILIŠTE U RIJECI

Bojan Premuš

UPRAVLJANJE ZRAKOPLOVOM PUTEM NEURONSKE MREŽE I GENETIČKOG ALGORITMA

(završni rad)

Rijeka, 2019.

VELEUČILIŠTE U RIJECI

Poslovni odjel
Stručni studij Informatika

UPRAVLJANJE ZRAKOPLOVOM PUTEM NEURONSKE MREŽE I GENETIČKOG ALGORITMA

(završni rad)

MENTOR

Vlatka Davidović, viši predavač

STUDENT

Bojan Premuš

MBS: 2422000087/16

Rijeka, prosinac 2019.

VELEUČILIŠTE U RIJECI **Prilog 1.**

Poslovni odjel

Rijeka, 1. listopada 2019.

ZADATAK
za završni rad

Pristupnik BOJAN PREMUŠ **MBS: 2422000087/16**

Studentu stručnog studija Informatika izdaje se zadatak završni rad – tema završnog rada pod nazivom:

Upravljanje zrakoplovom putem neuronske mreže i genetskog algoritma

Sadržaj zadatka: U završnom radu implementirati neuronsku mrežu u primjer upravljanja zrakoplovom u kojem će se učenje upravljanja izvoditi putem genetskog algoritma. Opisati osnovne koncepte neuronske mreže i *feed forward* algoritma. U implementacijskom dijelu opisati algoritam za treniranje upravljanja zrakoplovom. Na temelju naučenih podataka prikazati simulaciju upravljanja zrakoplovom.

Preporuka _____

_____

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

Zadano: 1. listopada 2019. **Predati do: 15.veljače 2020.**

Mentor: **Pročelnica odjela:**

Vlatka Davidović, v.pred. **mr.sc.Anita Stilin, v.pred.**

Zadatak primio dana: _____

Bojan Premuš

(Ime i prezime studenta)

Dostavlja se:
- **mentoru**
- **pristupniku**

Prilog 5.

I Z J A V A

Izjavljujem da sam završni rad pod naslovom

UPRAVLJANJE ZRAKOPLOVOM PUTEM NEURONSKE MREŽE I GENETIČKOG ALGORITMA

izradio samostalno pod

nadzorom i uz stručnu pomoć mentora Vlatke Davidović, viši predavač .

Ime i prezime

Bojan Premuš

(potpis studenta)

Sažetak

Ovaj rad je namijenjen svima koji se žele upoznati ili implementirati u svojim projektima neuronske mreže i/ili genetičke algoritme. Kroz bazični primjer predlažu se početna struktura, polazne točke, faze razvoja te prepreke i ograničenja u implementaciji zadanog sustava.

Analiziran je sustav upravljanja zrakoplovom, dekomponirani su ključni podsustavi i identificirana potreba za implementacijom neuronske mreže te genetičkog algoritma u svrhu treniranja zadanih mreža.

Zadnji korak ovog rada je implementacija uspješno trenirane neuronske mreže u sličan sustav kao dokaz koncepta prilagodljivosti neuronske mreže.

Ključne riječi: Neuronska mreža, Genetički algoritam, AI, Autopilot

SADRŽAJ

1. UVOD 1
2. NEURONSKE MREŽE 2
3. *FEED-FORWARD* ALGORITAM 6
4. SIMULATOR UPRAVLJANJA ZRAKOPLOVOM 10
5. GENETSKI ALGORITAM ZA TRENIRANJE UPRAVLJANJA ZRAKOPLOVOM 18
6. IMPLEMENTACIJA SIMULATORA 23
6.1. *Neural Network* 24
6.2. *AIController* 27
6.3. *Training manager* 28
7. TRENIRANJE UPRAVLJANJA ZRAKOPLOVOM 30
7.1. Bodovanje položaja zrakoplova 30
7.2. Bodovanje visine leta 31
7.3. Bodovanje brzine leta 31
7.4. Bodovanje prolaska kroz točku 32
8. TESTIRANJE I SIMULACIJA UČENJA 33
9. ZAKLJUČAK 39
LITERATURA 41
POPIS SLIKA 42

1. UVOD

Pravilnom primjenom neuronske mreže računalo izvršava zadatke poput klasifikacije rukopisa, govora i slika, prevođenja stranog jezika sa korekcijom gramatike te simulira upravljanje i odlučivanje nalik ljudskom mozgu. Vjerovalo se da računalo nikad neće moći obavljati takve zadatke ili bar ne sa efikasnošću ljudskog mozga, no razvojem i povezanošću računala stvoreno je plodno tlo za evoluciju na području računalnog učenja. Neuronske mreže otvaraju mogućnost implementacije sustava gdje imamo poznate ulaze i očekivane ili željene izlaze, ali ne znamo kako implementirati algoritme za obradu ili implementaciju istih jer su postupci prezahtjevni i često neisplativi. Dostupnost velike količine digitalnih klasificiranih podataka putem interneta i znatno napredovanje u brzini obrade podataka olakšava proces učenja ili ovisno o kontekstu bolje bi bilo reći treniranja neuronskih mreža. (Ilić, 1999) Paralelnom izvedbom neutralnih mreža ubrzava se proces treniranja, odnosno izvođenjem većeg broja jednakih zadataka i istih tipova mreža na većoj populaciji. Također unutar svake jedinke populacije moguće je paralelno izvršavanje više neuronskih mreža koje zajedno rješavaju jedan zadatak. Svaka mreža je zadužena za jedan segment, ali ocjenjuju se performanse cjelokupnog sustava i rada svih segmenata zajedno.

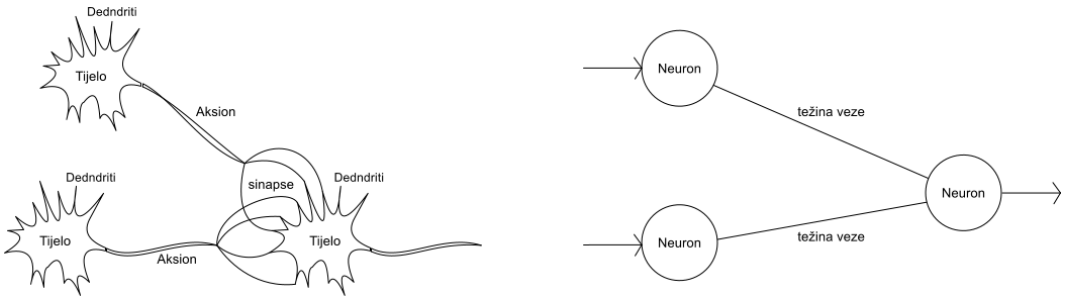
Tema ovog rada je prikaz upravljanja zrakoplovom paralelnim izvođenjem neuronskih mreža tako da funkcije upravljanja zrakoplovom podijelimo na sličan način kako bi ih posada zrakoplova izvodila[1] (eng *Multi crew coordination*). Mreže se treniraju na modelu sa simplificiranom fizikom upravljanja te po završetku treninga, trenirana mreža se testira na simulatoru sa profesionalnim fizičkim modelom upravljanja.

[1] MCC, Multi crew coordination je dozvola koju piloti zrakoplova stječu gdje se boduje sposobnost upravljanja zrakoplova koordinacijom više članova posade, a rezultat ovisi o uspjehu svih članova.

2. NEURONSKE MREŽE

Neuronska mreža (engl. *Neural network*) je vrsta računalne obrade podataka modelirane na temelju rada živčanog sustava i ljudskog mozga, još je nazivamo i umjetna neuronska mreža (engl. *Aritifical neural network*). Obrada podataka vrši se kroz slojeve temeljnih jedinica, odnosno neurona i aktivacijom njihovih međusobnih veza. (automatika.rs,2019) Prvi model neuronske mreže razvili su neuropsiholog Warren McCulloch i matematičar Walter Pitts 1943. godine (Ilić, 1999), jednostavnim modelom simulacije ljudskog mozga sačinjenog od električkih vodljivih krugova koji su spajanjem i prekidanjem pojedinih segmenata simulirale "inteligentno" ponašanje (Strachnyi,2019). Na temelju njihovog rada postavljeni su temelji Hebbianove logike prema kojoj Massachusetts Institute of Technology (MIT) već 1954. godine razvija prvu računalnu neuronsku mrežu. (Strachnyi,2019)

Slika 1. Usporedba biološkog i umjetnog neurona



Izvor: autor

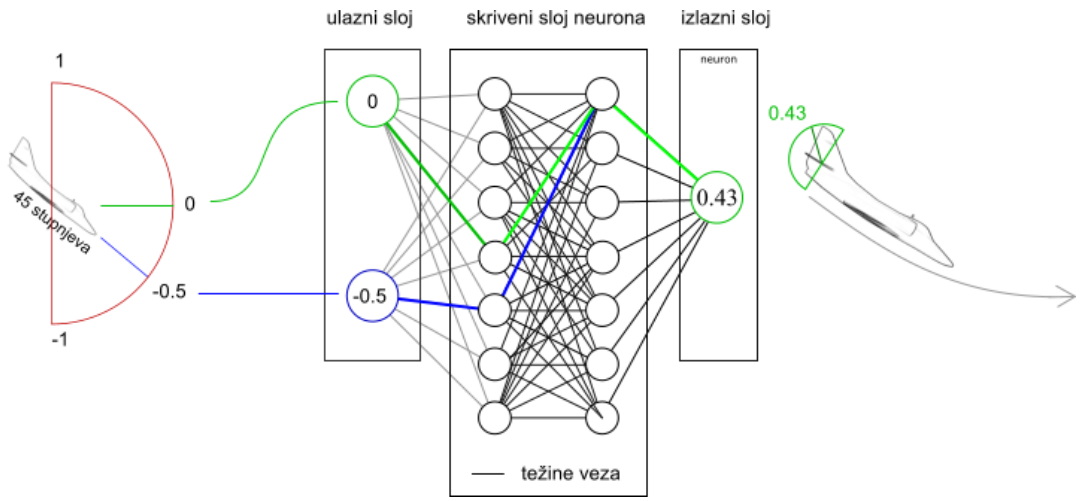
U ljudskom mozgu postoji oko tisuću različitih vrsta neurona od kojih je svaki dalje povezan sa oko sto drugih neurona čineći veliku neuronsku mrežu te aktivacijom pojedinih neurona prema precizno definiranom redoslijedu mozak šalje informacije koje prolaze transformaciju u željene radnje. Neuron je sastavljen od tijela stanice koja sadrži informaciju u obliku električnog (oko 70mV kada se nalazi u

neutralnom stanju) potencijala. Neuroni su dalje povezani sinapsama (i dendritima) kojima se potencijal šalje, zatim sumira i utječe na potencijal stanice primatelja. (Bašić, Čupić, Šnajder, 2019)

Ukoliko potencijal prođe određeni prag, neuron prelazi u aktivno stanje i generira novi akcijski potencijal koji putem članaka (aksiona) neurona pokreće elektro-kemijsku reakciju i inicira postupak prema drugim vezanim neuronima. Signali na taj način putuju neuronskom mrežom do svoje destinacije gdje se interpretiraju. (Bašić, Čupić, Šnajder, 2019)

Umjetna neuronska mreža preuzima model ljudskog mozga i električni potencijal neurona logički prikazuje u obliku realnog broja, najčešće u intervalu [0,1] ili [-1,1] radi jednostavnosti obrade. Neuron je vezan sa drugim neuronima putem veza. Svaka veza sadrži dodatni atribut odnosno težinu veze (engl. *Weights*). Informacija se šalje na način da neuron sumira težine svih dolaznih veza pomoću aktivacijskih funkcija i inicira sljedeće neurone koji ponavljaju postupak i kaskadno pokreću putovanje signala.

Slika 2. Simplificirani prikaz obrade neuronskom mrežom



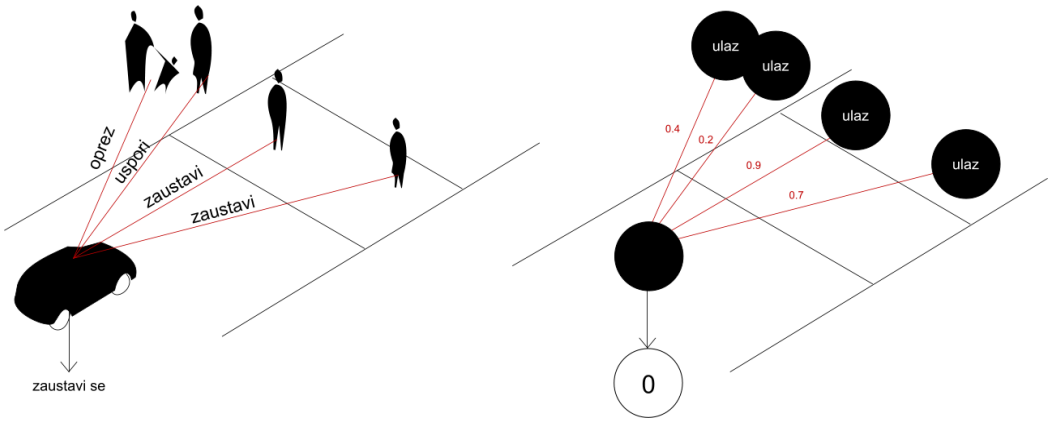
Izvor: autor

Spomenute aktivacijske funkcije koriste se kako bi dobivene rezultate „stisnuli“ unutar neurona te smanjili velike oscilacije među neuronima i samim time smanjili utjecaj pojedinih neurona unutar

mreže. Sam tip funkcije ovisit će o zadatku koji neuronska mreža obavlja, na primjer ukoliko želimo da naša mreža daje točan odgovor na određeno pitanje dovoljan je interval od 0 do 1, odnosno od netočnog ukoliko je manje od 0.5 ili točnog ukoliko je veće od 0.5. Ukoliko se radi o primjeru zrakoplova zanimaju nas i negativne vrijednosti radi otklona komandi zrakoplova.

Opravdano je za očekivati da neuronska mreža može donijeti odluku na temelju numeričkih ulaza i valjane aktivacije pojedinih neurona. Ukoliko se radi o ljudskom mozgu, u primjeru vozača automobila koji se približava prijelazu za pješake mozak dobiva mnoštvo informacija o sudionicima u prometu, njihovom položaju u odnosu na vozilo, udaljenosti, smjeru kretanja te dobi sudionika. Na temelju dobivenih informacija i stečenog iskustva[2] vozač povezivanjem svih elemenata donosi odluku da je potrebno zaustaviti vozilo. Neuronskoj mreži je upravo to iskustvo potrebno kako bi mogla donijeti ispravnu odluku.

Slika 3. Donošenje odluka neuronskom mrežom



Izvor: autor

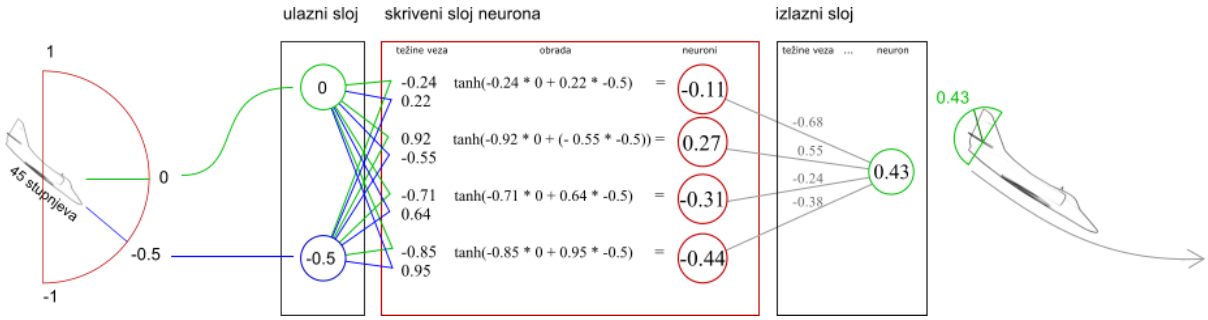
Jedno od glavnih svojstva neuronske mreže i sličnost sa ljudskim mozgom je upravo njena mogućnost učenja odnosno treniranja. Učenje se vrši uz pomoć dodatnih podataka (engl. Training data) gdje izlaze neuronske mreže uspoređujemo sa očekivanim podacima te mrežu optimiziramo kako bi dala

[2] Iskustvo stečeno u auto školi i upravljanju vozila, ali također i životno iskustvo u prepoznavanju ponašanja ljudi

točniji rezultat. Primjer učenja se može vidjeti kroz neuronske mreže za prepoznavanje slika, gdje za svaku ulaznu sliku postoje i dodatni podaci o obliku opisa (eng. Tag) što mreža koristi za usporedbu datog izlaza. Također osim učenja na temelju očekivanih podataka, mrežu je moguće trenirati i kontrolirano voditi njezinu evoluciju putem genetskih algoritama u očekivanju da se mreža sama optimizira za najbolji rezultat.

Neuronska mreža nema mogućnost dinamičkog kreiranja novih neurona stoga trening moramo postići sa konačnim brojem definiranih neurona i njihovih međusobnih veza. Ukoliko je potrebno mrežu naučiti stvari koje nisu bile definirane početnim opsegom mreže, povećati ćemo broj neurona i ponoviti fazu treniranja.

Slika 4. Prikaz donošenja odluke



Izvor: autor

U umjetnoj neuronskoj mreži iako se obrađuju svi neuroni, aktivacijom nazivamo kad se vrijednost pojedinih neurona ističu među ostalim i utječu na konačni rezultat. U primjeru zrakoplova udaljenost od zemlje može imati vrlo veliku težinu, odnosno važnost u letu i utjecati na odluku da zrakoplov ne udari u tlo, no također varijacijom ulaza gdje zrakoplov leti vrlo visoko ili želi letjeti prema dolje, aktivacija drugih neurona će nadjačati odluku da usmjeri zrakoplov u horizontalni let.

3. *FEED-FORWARD* ALGORITAM

Jednosmjerne neuronske mreže, engl. *Feed Forward (FF)*, odnosno *Deep Feed Forward* neuronske mreže sadrže osnovu koja se proteže i kroz ostale modele neuronskih mreža. Kako samo ime govori impulsi se šalju u jednom smjeru i povratak impulsa nije moguć[3]. Biolozi pretpostavljaju da takva situacija i kod biološke neuronske mreže. Ova informacija je bitna prilikom treniranja neuronske mreže, no da bi bilo moguće razumjeti treniranje prvo je potrebno shvatiti kako mreža radi.

FF Mreža se sastoji od nizova neurona podijeljenih u tri glavne skupine, a to su ulazni, izlazni i skriveni nizovi. Veličinu nizova sami definiramo, zato što imamo poznate ulaze i očekivane izlaze te na temelju tih varijabli predvidjeti ćemo koliko skrivenih nizova i koliko neurona bi svaki skriveni niz trebao imati. Iako postoje različite formule po kojima se može definirati potreban broj neurona, količina neurona i nizova rezultirat će performansama [4]same mreže i rezultatu koji želimo postići tako da će se ta definicija u konačnici svesti na subjektivni odabir. (Gori, 2018)

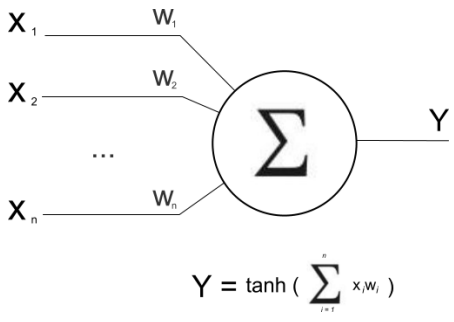
Neuroni niza su dalje povezani sa svim neuronima u sljedećem nizu i te veze nazivamo još težine veza (engl. *Weights*). Svaki ulaz u neuron biti će obrađen i proslijeđen sljedećem nizu. Koliko će se njegova vrijednost promijeniti ovisit će o težini veze. Iznimka su *bias* [5]neuroni koji su česta pojava u manjim neuronskim mrežama te služe za korekciju rezultata aktivacijske funkcije.

Kako smo napomenuli da je mreža jednosmjerna, nakon što dobijemo rezultat iz izlaza neuronske mreže, nismo u mogućnosti napraviti povratni prolaz kroz mrežu (engl. *Back-Propagation*). Povratni prolaz bi dao mogućnost da ukoliko znamo koji bi bio očekivani rezultat mrežu postepeno optimiziramo od trenutno dobivenog rezultata prema početku, odnosno ulazu mreže kroz skrivene nizove kako bi dala bolji, odnosno točniji rezultat koji mi očekujemo. Iako takve mreže su mnogo naprednije, za upravljanje zrakoplova odabrali smo ipak FF neuronsku mrežu i u daljnjem tekstu ćemo vidjeti koje su prednosti iste.

[3] Povratak impulsa je moguć na *Back-Propagation* tipu neuronske mreže
[4] Neuronska mreža se mora moći izvršiti više puta unutar jedne sekunde, zrakoplov ne može čekati na obradu podataka.
[5] U primjeru upravljanja zrakoplova *bias* neuroni nisu potrebni.

Aktivacija neurona FF mreže izvršava se kroz ulaze neurona prethodnog sloja na način da se za svaki pojedini ulaz množi njegova vrijednost i težina veze između prethodnog sloja i neurona te dobivene vrijednosti sumiraju (Bašić, Čupić, Šnajder, 2019). Dobivena suma se zatim obrađuje aktivacijskom funkcijom kako bi se rezultat „stisnuo“[6] unutar intervala -1 i 1. Izbor aktivacijske funkcije ovisit će o potrebama neuronske mreže.

Slika 5. Aktivacijska funkcija u neuronu

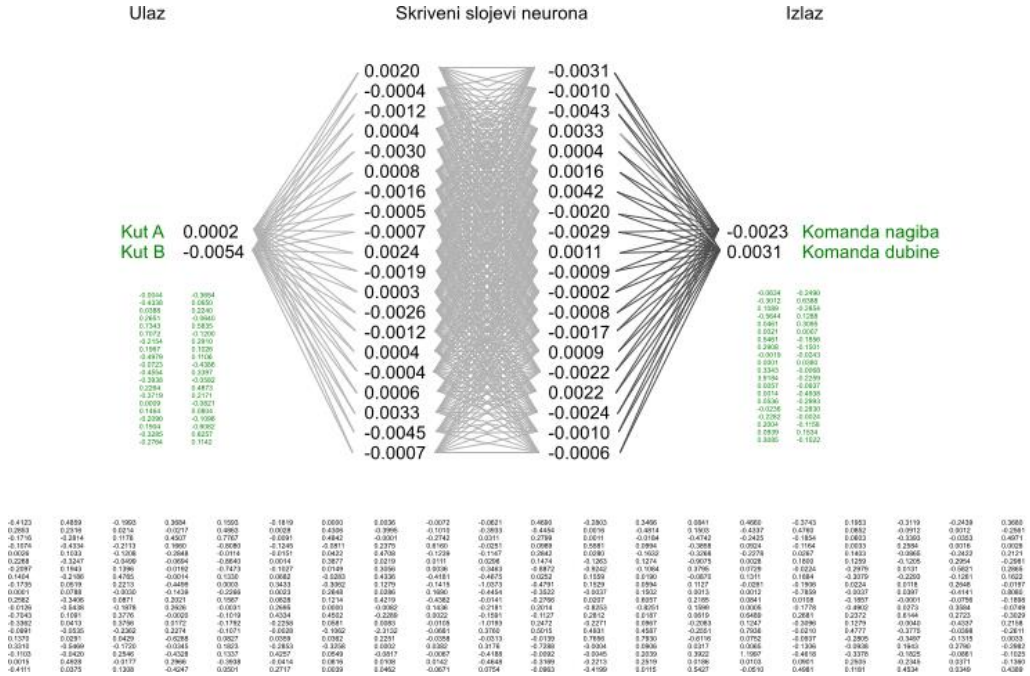


Izvor: autor

U primjeru zrakoplova koristili smo hiperboličku tangens aktivaciju iz razloga što želimo da vrlo velike, odnosno vrlo male vrijednosti budu 1 odnosno -1 ili maksimalni otkloni upravljačkih površina i da imamo precizni raspon pomaka unutar dopuštenih otklona sa nulom u stanju mirovanja.

[6] Rezultati neurona će uvijek biti između 1 i -1 čak i ako je dobivena suma 10 ili -10, zadatak treninga je optimizirati mrežu i koristiti ispravno aktivacije.

Slika 6. Prikaz izračuna mreže za upravljanje zrakoplovom (skriveni slojevi)



Izvor: autor

Važno je dobiti sliku o količini obrađenih podataka kako bi lakše planirali potrebne resurse. U primjeru neuronske mreže upravljanja zrakoplova za jednu od tri aktivne mreže jednog zrakoplova prikazane su vrijednosti za pojedini neuron nakon obrade i lista težina veza u stupcima.

FF Neuronska mreža bez potrebnog treninga dati će nam nasumične rezultate, koji naravno mogu biti i vrlo dobri te nerijetko je da čak u prvoj generaciji, odnosno pokušaju treninga dobijemo zadovoljavajući rezultat pogotovo ako imamo veliki broj uzoraka na kojem treniramo.

No kako bi FF mrežu bilo moguće kvalitetno trenirati potreban je genetski algoritam koji će preuzeti ulogu trenera, baviti se evolucijom naše mreže, mutirati i rangirati najbolju mrežu. Kao kod prirodne selekcije zadatak je izdvojiti najbolje neuronske mreže i raditi daljnju evoluciju od njih te eliminirati slabije neuronske mreže kako bi ostali samo najbolji neuroni za daljnju reprodukciju. Na

temelju izlaza neuronske mreže genetski algoritam daje ocjenu mreži u obliku pozitivnog ili negativnog broja te će kreirati novu neuronsku mrežu na temelju prethodne sa evolucijom neurona. Malom izmjenom vrijednosti težina veza neurona dobivenu ocjenu nove mreže uspoređivati će sa prethodnom te od bolje rangirane mreže izvršavati će se kopija i evolucija. (Upadhyay,2019)

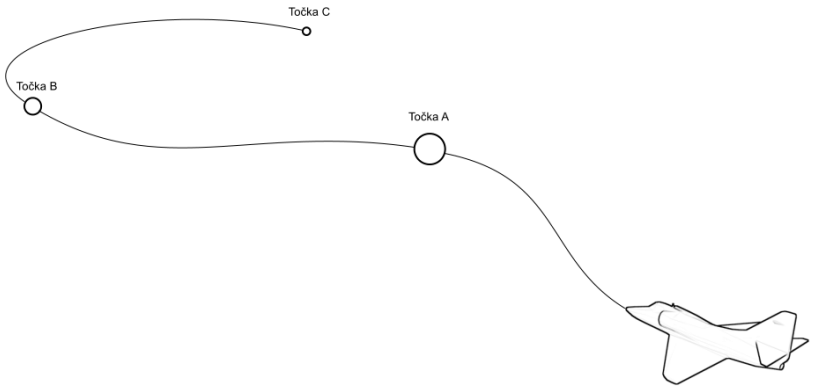
Proces se ponavlja dok ne dobijemo zadovoljavajući rezultat, a rangiranje kao preduvjet traži dobro poznavanje tematike koju neuronska mreža mora riješiti. Pozitivne i negativne radnje moraju biti jasno definirane te omjeri u bodovanju. Algoritam će iskoristiti sve moguće propuste naše aplikacije u dobivanju najveće pozitivne vrijednosti. Korektivne mjere su potrebne bilo da se radi o korekciji grešaka u samoj aplikaciji ili korekciji negativnog bodovanja kako bi uzeli u obzir nova „otkrića[7]“ genetskog algoritma. Algoritam također može ući u stanje lokalnog minimuma, odnosno postići minimalno zadovoljavajući rezultat bez mogućnosti većeg napretka dok izlazak traži i veće mutacije u vrijednosti neuronske mreže što može također nazadovati kvalitetu mreže u nadi da bi postigli bolji rezultat. U daljnjem tekstu vidjet ćemo na primjeru zrakoplova način treninga.

[7] Mreža izvršava zadatak na optimalan način, ukoliko postoje propusti u kodu, mogućnost obilazne radnje ili ne izvršavanja zadataka kao prihvatljiv rezultat, mreža će to iskoristiti.

4. SIMULATOR UPRAVLJANJA ZRAKOPLOVOM

Zadatak mreže je upravljanje zrakoplovom kroz niz unaprijed određenih točaka, odnosno koordinata na trodimenzionalnoj virtualnoj stazi po kojoj se zrakoplov kreće u što kraćem vremenskom roku poštujući Newtonovu [8]fiziku. Implantacija u simplificiranom simulatoru letenja dozvoljava nam korištenje više instanci istog treninga te povećava šansu za bržom i uspješnijom evolucijom. Cilj je da neuronska mreža može upravljati različitim tipovima zrakoplova, odnosno da „nauči“ logiku upravljanja zrakoplovom te rezultat testiramo na drugom sofisticiranijem tipu simulatora.

Slika 7. Koncept simulatora



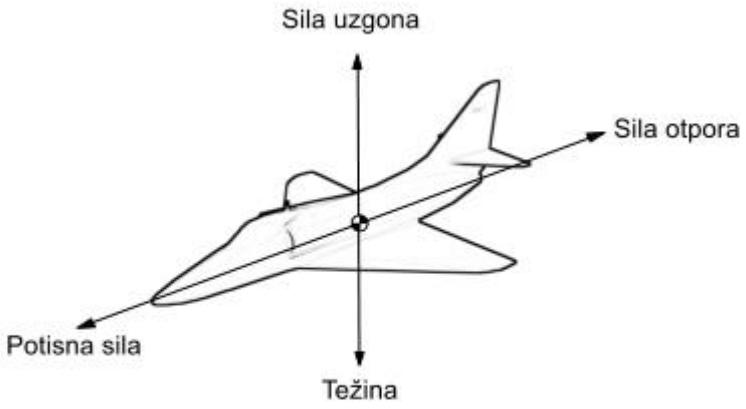
Izvor: autor

Simulator na kojem treniramo neuronske mreže razvijen je pomoću *Unity engine software*-a i C# programskog jezika, a računamo sile poput uzgona, otpora, gravitacije i potiska te dodatne aerodinamične sile koje postižu stabilnost leta. Aerodinamične sile se povećavaju ili međusobno neutraliziraju

[8] Sile koje djeluju na zrakoplov moraju biti unutar sila koje ljudsko tijelo može izdržati, idealno je da to bude u rasponu od -1 do 2G

povećanjem brzine, a radi jednostavnosti određene sile poput gubitka uzgona su simplificirane. Zadatak treninga je upravljanje zrakoplovom unutar normalnog režima leta.

Slika 8. Sile koje djeluju na zrakoplov



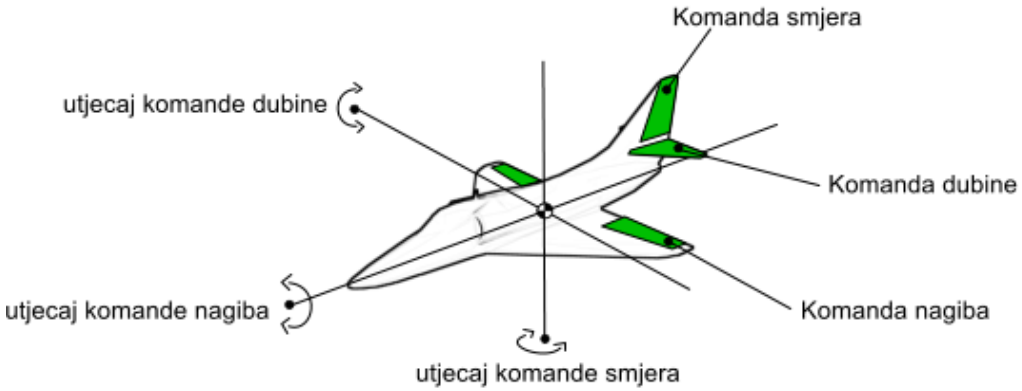
Izvor: autor

Upravljanje zrakoplovom moguće je postići i klasičnim sustavom poput autopilota koji bi prema striktno određenim pravilima usmjeravao zrakoplov prema zadanoj točki. Problem autopilota kao i u pravom zrakoplovu je ukoliko ulazi izlaze iz režima dozvoljenih ulaza, odnosno ako se zrakoplov nalazi u bilo kojem nedozvoljenom položaju autopilot se automatski isključuje i prepušta upravljanje pilotu. Korištenjem neuronske mreže želimo da upravljanje zrakoplovom bude nalik upravljanju pilota te ukoliko postoji mogućnost da mreža i dalje evoluira.

Različite vrste zrakoplova ponašaju se drugačije u skladu sa aerodinamičnim svojstvima, ali svi imaju identičnu logiku upravljanja gdje pilot pomacima komandi zrakoplova dovodi zrakoplov u određenu poziciju.

Opravdano je očekivati da pilot koji je treniran na jednom tipu zrakoplova sposoban je upravljati, odnosno preciznije održavati visinu i zadani pravac bilo kojeg tipa zrakoplova podešenog u režimu krstarenja[9] samo pomoću osnovnih komandi, odnosno palice / volana za upravljanje (engl. *Flight stick, yoke*), pedala (engl. *Rudder*) i poluge / poluga gasa (engl. *Flight throttle*). Slijetanja i polijetanja zrakoplova traže dodatno poznavanje zrakoplovnih sistema i procedura za pojedini tip, stoga njih nećemo obrađivati ovom temom.

Slika 9. Utjecaj komandi zrakoplova



Izvor: autor

Poziciju, položaj i brzinu kretanja zrakoplova u prostoru dobivamo putem žiroskopskih i pito-statičkih instrumenata. Podatke ćemo simulirati i prikazati kao digitalne vrijednosti te proslijediti neuronskoj mreži na obradu. Izlazi neuronske mreže će direktno rezultirati pomacima upravljačkih

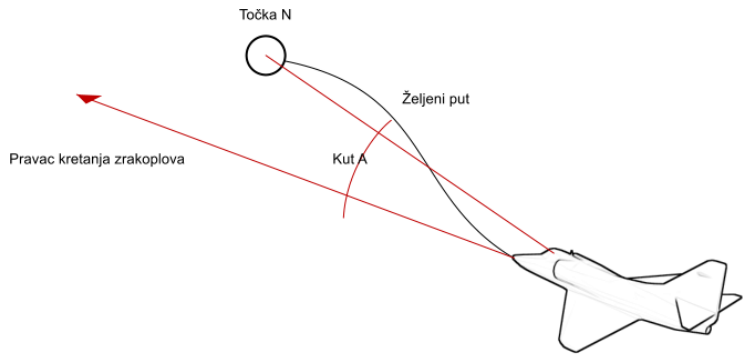
[9] Površine zrakoplova su optimizirane za vodoravni let (engl. *level flight*) te postavke motora za optimalnu potrošnju u odnosu na udaljenost

površina zrakoplova. Proces se odvija u realnom vremenu sa otprilike 60 kalkulacija u sekundi. Dodatno proces je podijeljen u tri pod-procesa, odnosno tri neuronske mreže. Razlog je optimizacija zadataka, pošto i veći zrakoplovi imaju raspodjelu uloga upravljanja na pilota i kopilota.

Iako je jedan pilot dovoljan za upravljanje zrakoplovom pojedini zadaci se dijele, primjerice dok pilot upravlja zrakoplovom prilikom slijetanja kopilot je zadužen za korekcije gasa, kontrolu zakrilaca i pred-krilca, zračnih kočnica i stajnog trapa te komunikaciju. Prva mreža ima zadatak upravljanja zrakoplova prema zadanoj točki u prostoru.

Kao ulaz uzima poziciju zrakoplova u prostoru u odnosu na zadanu točku i vrši pomake upravljačkih površina (nagib zrakoplova te kut penjanja / spuštanja)

Slika 10. Kut A između osi kretanja zrakoplova i pravca prema željenoj destinaciji

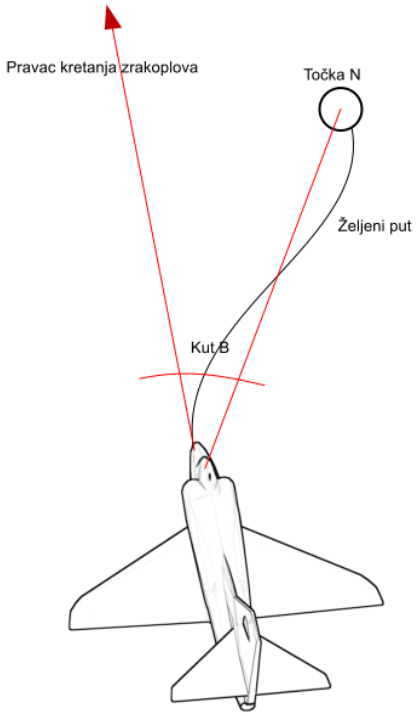


Izvor: autor

Mjerimo kut u pravcu kretanja zrakoplova te pravcu prema točki u prostoru, kut projiciramo u trodimenzionalni koordinatni sustav gdje Y os pokazuje vertikalno gore, odnosno (0,1,0) te dobivamo dva kuta u odnosu na zadanu točku. Prvi kut A, je kut u odnosu na visinu točke između zrakoplova i zadane točke. Kut se zatim pretvara u interval između -1 i 1 gdje nula reprezentira horizontalni let zrakoplova. Pozicija zrakoplova u odnosu na zemlju nije bitna, dopušteni su svi položaji zrakoplova.

Radi lakšeg razumijevanja moguće je zamisliti kretanje slično kretanju svemirskog broda van orbite zemlje. Stabilnost ćemo postići aerodinamičnim svojstvima [10]samog zrakoplova.

Slika 11. Kut B između osi kretanja zrakoplova i pravca prema željenoj destinaciji



Izvor: autor

Drugi dobiveni kut je kut u odnosu na polegnuti pravac na ravninu XZ u smjeru kretanja zrakoplova. Dobiveni rezultat nam govori pod kojim kutom se točka nalazi desno ili lijevo od zrakoplova te također dobiveni rezultat pretvaramo u interval između -1 i 1.

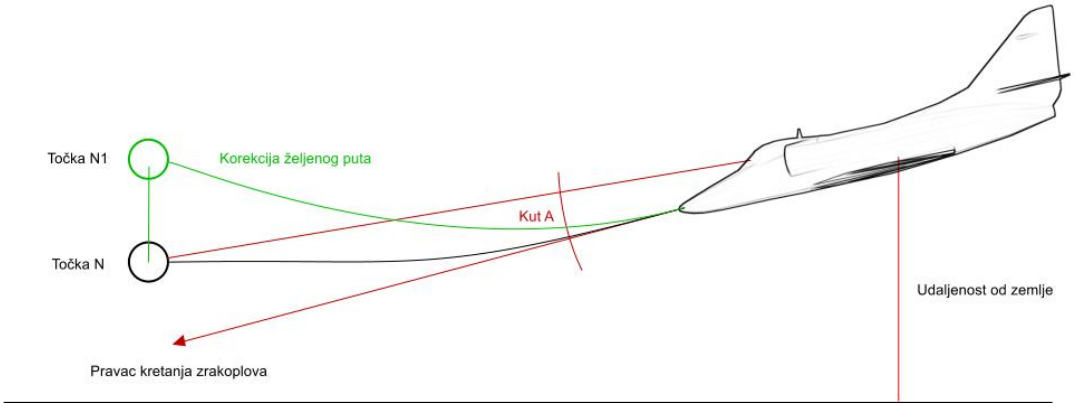
Problem koji nastaje u upravljanju zrakoplova kod horizontalne promjene pravca, odnosno skretanja lijevo ili desno postiže se rotacijom zrakoplova oko uzdužne osi (nagibom zrakoplova) u kombinaciji sa rotacijom oko poprečne osi (za razliku od automobila gdje se skretanje vrši oko vertikalne osi) što zahtjeva sinkroniziranu [11] koordinaciju obje komande kako bi zrakoplov zadržao visinu te

[10] Dizajn krila u obliku slova V gledano u pravcu uzdužne osi ima zadatak stabilizirati let zrakoplova.
[11] U zaokretu zrakoplov ima tendenciju poniranja.

postepeno skrenuo prema zadanoj točki. Zrakoplov može također letjeti invertno gdje dolazi do zamjene komandi tako da sila na zrakoplov i dalje bude pozitivna. Druga mreža ima zadatak održavanja zrakoplova na zadanoj visini od zemlje. Kao ulaz uzimamo visinu od zemlje i vertikalnu brzinu u odnosnu na zemlju, a kao izlaz vršimo korekciju visine točke prema kojoj prva neuronska mreža upravlja.

Dobivene vrijednosti se preračunavaju u intervale od -1 do 1 te također definirana je i maksimalna visina na kojoj udaljenost od zemlje nije moguće izmjeriti[12].

Slika 62. Korekcija puta zrakoplova u odnosu na udaljenost od zemlje



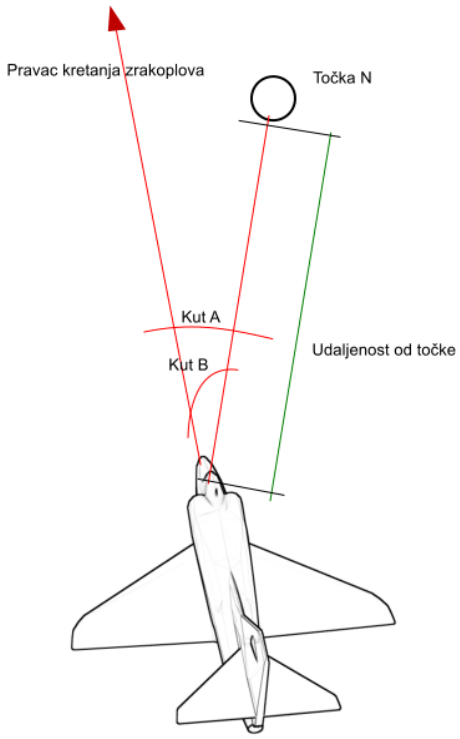
Izvor: autor

Zrakoplov mora imati mogućnost niskog leta, no prilikom poniranja potrebno je dovoljno rano reagirati kako bi zrakoplov mogao prijeći u horizontalni let. Prilikom poniranja zrakoplov akumulira dodatnu brzinu radi utjecaja gravitacije.

Treća mreža ima zadatak održavanja optimalne brzine. Mreža uzima poziciju zrakoplova u odnosu na točku prema kojoj se kreće i udaljenost do točke te kao izlaz vršimo pomak ručice gasa zrakoplova. Udaljenost od točke je bitna kako bi mreža mogla pravovremeno usporiti ukoliko je potrebno izvesti određeni zaokret prema točki.

[12] Pravi zrakoplovi koriste radarski visinomjer koji ima maksimalni domet.

Slika 13. Korekcija brzine zrakoplova ovisno o položaju i udaljenosti od tražene točke



Izvor: autor

Brzina zrakoplova direktno utječe na sile u zaokretu zrakoplova te povećava ili smanjuje radijus prolaska oko točke. Problemi koji se javljaju su da mreža ne može jednostavno predvidjeti odluke prve i druge mreže te utjecaje brzine može najviše štetiti objema mrežama iako su performanse prve dvije mreže su zadovoljavajuće.

Vrednovanje svih mreža ovisi o tome hoće li zrakoplov u zadanom vremenu stići do sljedeće točke u prostoru i najbolja mreža je ona koja u najkraćem vremenskom periodu prođe najviše točaka, odnosno najoptimalnije leti. Bodovanje se vrši kad zrakoplov leti prema točki te prođe zadanu točku unutar dozvoljenog vremena te se dodatno kažnjavaju negativne [13]sile na zrakoplovu i naravno udarac u zemlju. Kada bi željeli mrežu implementirati u pravi zrakoplov dodatne restrikcije poput maksimalnih

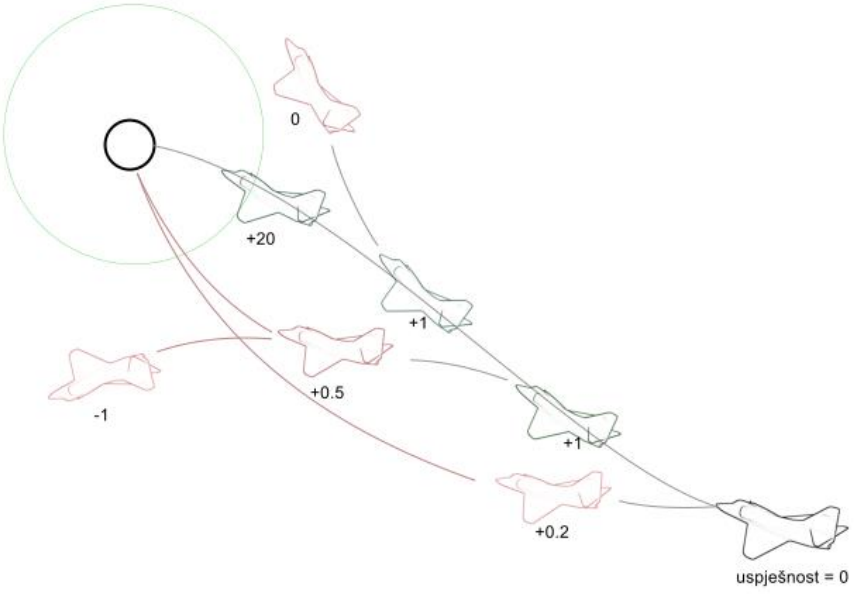
[13] Zrakoplovi su konstruirani da većinom mogu podnijeti veće pozitivne sile od negativnih.

pozitivnih i negativnih sila, maksimalni nagib i kut zrakoplova te brzine pomaka komandi bile bi potrebne, no za potrebe našeg treninga minimalni set je zadovoljen.

5. GENETSKI ALGORITAM ZA TRENIRANJE UPRAVLJANJA ZRAKOPLOVOM

Zadatak genetskog algoritma je rangiranje neuronskih mreža te vršenje selekcije i eliminacije manje uspješnih mreža. Na način sličan prirodnoj selekciji ostaju mreže koje su pokazale najbolji rezultat i njihovom kontinuiranom mutacijom dolazimo do još uspješnijih mreža. Da bi bilo moguće rangirati mrežu kako je prethodno spomenuto, bitno je jasno postaviti ciljeve koje mreža mora izvršiti te korektno nagrađivati i kažnjavati mrežu. Svaka mreža svoju vrijednost prikazuje kroz atribut „sposobnost“ (engl. *Fitness*) te se ta vrijednost uspoređuje sa ostalim mrežama. Najsposobnija mreža se kopira i njena kopija dalje rangira i uspoređuje sa prethodnim mrežama, a najneuspješnija mreža se briše. (Ptrkkim, 2019)

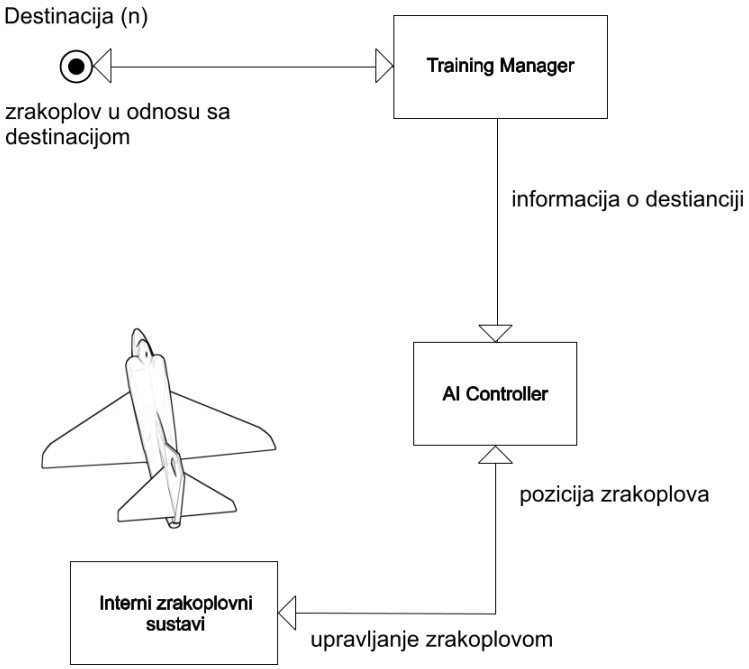
Slika 14. Bodovanje uspješnosti algoritma



Izvor: autor

Bodovanje mreže može biti jedan od najkompleksnijih zadataka, obzirom da mreža nema mogućnost „imitiranja“ drugih objekata kao živo biće, bodovanje potpuno ovisi o jasno postavljenim zadacima i ciljevima. Potrebno je postići ravnotežu između nagrađivanja i kažnjavanja te po potrebi postaviti vrstu mikro nagrađivanja između pojedinih većih točaka za postizanje kontinuiranog napredovanja mreže. Primjerice ukoliko mrežu nagrađujemo za svaku pozitivno izvršenu radnju, mreža će zasigurno nastaviti raditi prvu akciju koja joj donosi pozitivno bodovanje i opstanak u evolucijskom grananju. Sa strane mreže zadatak je pozitivno obavljen iako naša očekivanja nisu ispunjena. Prekomjerno kažnjavanje također nije dobro jer pozitivne radnje brzo budu poništene negativnim bodovanjem i dolazimo do stanja u kojem mreža više ne može ispuniti očekivanja. Iako postoji šansa da mreža i bez treninga nasumičnim postavljanjem da vrlo dobar rezultat, treniranje mreže je svakako sa pogleda na vremenski period najučinkovitiji proces.

Slika 75. Proces treniranja NN zrakoplova

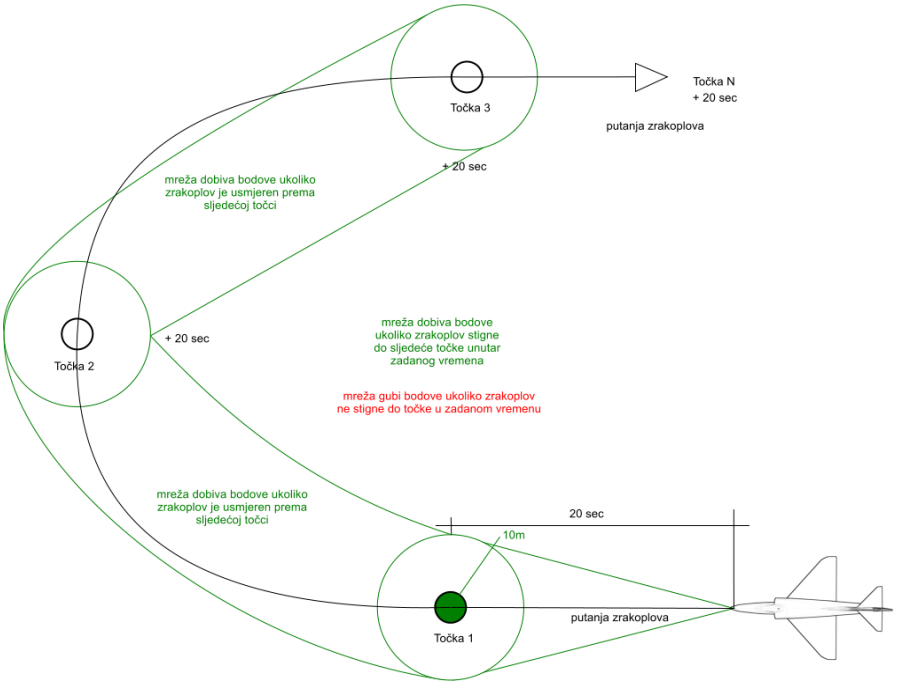


Izvor: autor

Genetski algoritam u našem primjeru sastavni je dio umjetne inteligencije zrakoplova i direktno manipulira neuronskim mrežama zrakoplova. Za uspješno bodovanje genetski algoritam informacije dobiva iz internih i eksternih izvora, odnosno internih sistema zrakoplova te globalne pozicije zrakoplova u odnosu na destinacijsku točku u prostoru. Oba sustava su neophodna za uspješno bodovanje, primjerice zrakoplov može savršeno letjeti, ali u smjeru suprotnom od zadane točke.

Za treniranje pomoću destinacijskih točaka potrebna nam je virtualna staza te *Training manager* čiji zadatak je dodijeliti zrakoplovu sljedeću točku te mjeriti vrijeme prolaza kroz nju. U isto vrijeme nagraditi će trenutnu neuronsku mrežu, odnosno prilikom neuspjeha kazniti je [14] i rastirati trening u svrhu postizanja evolucije.

Slika 86. Putanja zrakoplova i bodovanje

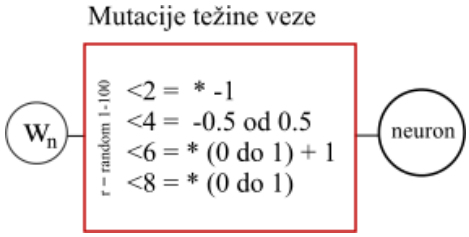


Izvor: autor

[14] Training manager ne vrši samo kažnjavanje ili nagrađivanje i to je zadatak genetskog algoritma, dok zadatak training managera je proslijediti tu informaciju.

Zrakoplov odnosno njegova komponenta *AI manager*[15] prima informacije *Training managera* o točki prema kojoj treba letjeti. *AI mana*ger po primitku točke postaje autonoman i njegov zadatak je koristiti trenutnu neuronsku mrežu, vrednovati je na temelju pozicije zrakoplova u odnosu na nju. U slučaju kada Training manager dodjeli novu točku, nema striktnog prekida rada *AI managera*, za njega je pozicija točke dinamična vrijednost koja se očitava svakih nekoliko milisekundi. Resetiranje dolazi samo na zahtjev *Training managera* ukoliko zrakoplov nije unutar zadanog vremena doletio do točke odnosno zrakoplov je udario u zemlju. U trenutku poziva za resetiranje mreže se sortiraju po „sposobnosti“ (*fitnessu*), od najveće prema najmanjoj, zatim najbolje mreže mutiraju na način da se vrijednosti težina veza pojedinih neurona malo izmjene. Nova mreža postaje aktivna, a njena sposobnost je postavljena na nulu. Na taj način ukoliko mreža ne dostigne sposobnost mreže od koje je nastala, izvorna mreža mutira, a stara kopija će postepeno biti izbrisana. Radi lakšeg praćenja broja mutacija, svakoj novoj mreži dodjeljujemo broj generacije.

Slika 97. Mutacije težine veze pojedinog neurona



Izvor: autor

Osim mutacije neuronske mreže, moguće je postići i promjene križanjem najuspješnijih neuronskih mreža, križanjem pokušavamo optimizirati rad mreže, no veće promjene i skokovi u evoluciji nisu mogući. Korištenjem mutacije, promjene su drastičnije te rezultati nisu nužno bolji svakom mutacijom, ali nakon dovoljno vremena mreža će evoluirati.(The-one, 2017)

Mreža može zapeti na takozvanom lokalnom minimumu gdje mutacijama dobivamo slične rezultate kako bi ubrzali proces i povećali šansu za evolucijom i boljim rezultatom. Najjednostavnije je

[15] AI manager je zadužen za istovremeno upravljanje sa tri neuronske mreže

povećati broj populacije na kojoj se mutacija provodi, odnosno broj zrakoplova. Povećani broj zrakoplova će utjecati i na performanse rada računala.

S obzirom da su početne vrijednosti težina neuronskih mreža nasumično odabrane, moguće da će već prva generacija pokazati dobar rezultat pogotovo ako se radi o velikom broju populacije na kojoj testiramo.

6. IMPLEMENTACIJA SIMULATORA

Za simulaciju upravljanja zrakoplova pomoću neuronske mreže korišten je *Unity Game Engine.* [16] Komponenta zrakoplova sa popratnom fizikom i grafikom te okolina u kojoj se simulacija izvršava radi složenosti pojedinih sustava nisu obuhvaćene ovih radom.

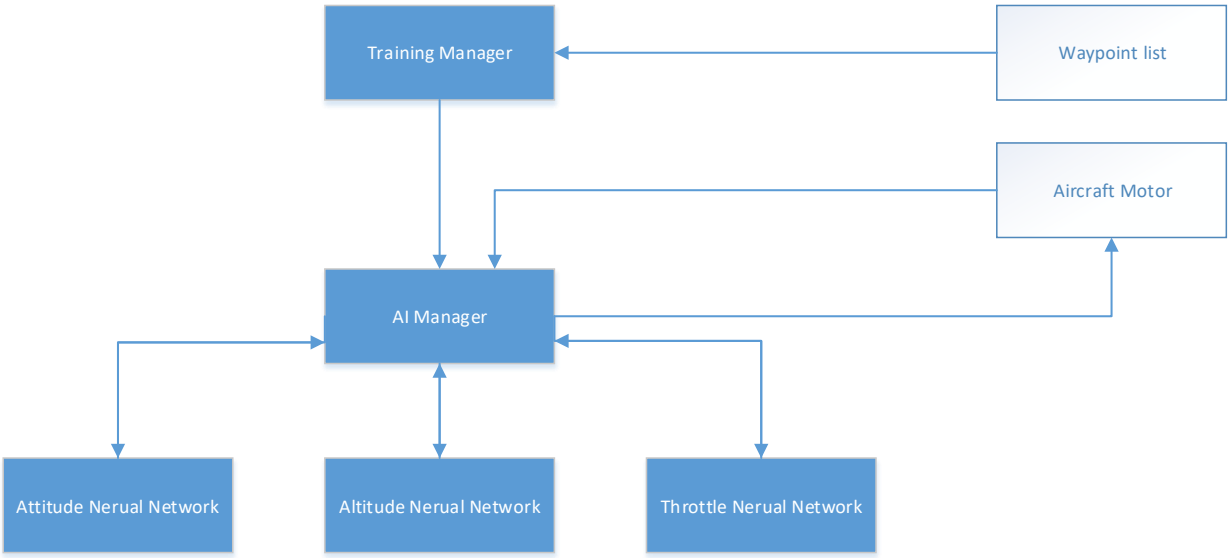
Implementacija neuronskih mreža za upravljanje i sustava treniranja izvršena je u obliku C# komponenti pridruženih simulatoru letenja. Komponente nasljeđuju *MonoBehaviour* klasu, odnosno bazičnu klasu od koje sve *Unity* skripte nasljeđuju.

Zrakoplov je prezentiran kao Instanca *GameObject*-a sa pridruženim komponentama, kao *RigidBody* [17]i *BoxCollider* [18]koja omogućava korištenje 3D fizike *Unity Game Engine*-a te detektirati kolizije sa drugim objektima poput zemlje ili točaka destinacije. Dodatne komponente kao *Aircraft Motor* zadužene su za izračun sila koje djeluju na objekt i simuliraju upravljanje zrakoplovom te očitavanje simuliranih instrumenata zrakoplova potrebnih za rad umjetne inteligencije.

Točke destinacija su također instance game objekta, a sadržene su kao lista referenca *u Nerual manager* komponenti. Zadatak te komponente je distribucija točaka odgovarajućim redom *AI controlleru* te mjerenje prolaznog vremena. Ostale kalkulacije se izvode u *Ai Controller* komponenti. Eksterna informacija je detekcija kolizije sa zemljom koja se bilježi u obliku *Event-a* unutar Neural managera koje se dalje prosljeđuju *AI Controlleru*.

[16] Unity Game Engine je zbirka alata za izradu simulacija i računalnih igara.
[17] Rigid body omogućava pristup simulaciji fizike
[18] Box colider omogućava koliziju objekata odnosno očitavanje kolizije na temelju presjeka kvadara i plohe

Slika 18. Povezanost klasa



Izvor: autor

6.1. *Neural Network*

Neural Network je C# klasa koja implementira *Icomparable interface* koja omogućava pozive sortiranja neuronskih mreža prema vrijednosti uspješnosti mreže (*fitness*-a). Obrada se izvršava u CompareTo metodi.

```
public class NeuralNetwork : IComparable<NeuralNetwork>
```

CompareTo metoda rangira vrijednosti neuronskih mreža na temelju vrijednosti fitness atributa

```
public int CompareTo(NeuralNetwork other)
```

Konstruktori klase koriste se za kreiranje nove neuronske mreže odnosno za „deep copy“ referentne mreže kako bi se mogla izvršiti mutacija mreže. *Layers* predstavlja broj slojeva neuronske mreže te za svaki indeks veličine sloja.

```
public NeuralNetwork(int[] layers)

public NeuralNetwork(NeuralNetwork copyNetwork)
```

Prilikom izvršavanja *deep copy* konstruktora slojevi moraju biti jednaki inicijalnoj mreži. Neuronska mreža će nasumično inicirati početne neurone i njihove težine veza, prije prve upotrebe vrijednosti se mogu učitati iz prijašnjih mreža kako bi se nastavilo treniranje.

Mutacija se izvršava putem Mutate metode.

```
public void Mutate(int evolve = 0)
```

Mutacija će nasumično modificirati neuronsku mrežu, odnosno težine neuronske mreže. Inicijalne težine kreirane su potpuno nasumično pozivom na privatnu metodu InitWeights(), dok Mutate radi samo djelomičnu izmjenu na temelju sljedećih uvjeta, generiramo nasumični broj od 0 do 1 i ako je dobiveni broj:

1. Kod mutacije bez evolucije manji od 2, mijenjamo predznak težine
2. Kod mutacije bez evolucije manji od 4, generiramo novu težinu u intervalu od -0.5 do 0.5
3. Kod mutacije manje od 6, generiramo novu težinu u intervalu od 0 do 1
4. Kod mutacije manje od 8, množimo novu težinu u nasumičnim brojem u intervalu od 0 do

```
//mutate weight value
float randomNumber = UnityEngine.Random.Range(0f, 100f);

if (randomNumber <= 2f && evolve == 0)
{
  //flip sign of weight
    weight *= -1f;
}
else if (randomNumber <= 4f && evolve == 0)
```

```
{
  //pick random weight between -1 and 1
    weight = UnityEngine.Random.Range(-0.5f, 0.5f);
}

else if (randomNumber <= 6f)
{
  //randomly increase by 0% to 100%
    float factor = UnityEngine.Random.Range(0f, 1f) + 1f;
    weight *= factor;
}
else if (randomNumber <= 8f)
{
  //randomly decrease by 0% to 100%
    float factor = UnityEngine.Random.Range(0f, 1f);
    weight *= factor;
}
```

FeedForward metoda sa odgovarajućim inputima, obavlja se obrada putem neuronske mreže.

```
public float[] FeedForward(float[] inputs)
```

Ulazni parametri moraju odgovarati veličini prvog stupca *Layers* polja. Mreža nakon obrade vraća vrijednost u obliku polja veličine zadnjeg stupca *layers* polja.

```
//iterate over all neurons and compute feedforward values
for (int i = 1; i < layers.Length; i++)
{
   for (int j = 0; j < neurons[i].Length; j++)
   {
      float value = 0f;
      for (int k = 0; k < neurons[i - 1].Length; k++)
      {
            //sum off all weights connections of this neuron weight their values in previous layer
         value += weights[i - 1][j][k] * neurons[i - 1][k];
      }

                  neurons[i][j] = (float)Math.Tanh(value); //Hyperbolic tangent activation
…
```

6.2. *AIController*

Klasa *AicraftAI* služi svrhu *AI Controllera* te je zadužena za implementaciju genetskog algoritma, uzima podatke zrakoplova, tumači ih te prosljeđuje neuronskim mrežama.

SetAttitude zadužen je za poziciju zrakoplova u prostoru, očitavanje ulazne točke prosljeđivanje informacija o opravljanju zrakoplovu.

```
public void SetAttitude()
```

Dohvat kutova dobivamo izračunom vektora pozicije točne umanjenog vektorom našom pozicijom kako bi dobili vektor pravca.

```
Vector3 deltaVector = (_target - transform.position).normalized;

float angle_h = AngleDir(transform.forward, deltaVector, transform.up);
float angle_v = AngleDir(transform.forward, deltaVector, transform.right);
```

Trodimenzionalne vektore projiciramo u koordinatni sustav kako bi dobili kut u odnosu na ravninu.

SetAltitude zadužen je za visinu zrakoplova u prostoru, mreža ne radi interakciju komandama zrakoplova već direktno manipulira ulazna očitanja mreže pozicije.

```
public void SetAltitude()
```

Ulazne parametre potrebne za rad mreže računa motor zrakoplova.

SetThrottle zadužen je za brzinu kretanja zrakoplova, direktno utječe na rad motora zrakoplova, a ulazne parametre potrebne za rad mreže računa motor zrakoplova.

```
public void SetThrottle()
```

Evolve je metoda zadužena za evoluciju neuronske mreže, kreiranje novih neuronskih mreža te brisanje neuspjelih mreža. Ovo je glavni dio genetskog algoritma.

```
public void Evolve()
```

Prvi korak je sortiranje mreža po uspješnosti te kreiranje nove neuronske mreže na temelju najuspješnije.

```
attitude_nets.Sort();
NeuralNetwork newNetwork = new NeuralNetwork(AttitudeNetwork);
newNetwork.SetFitness(0);
newNetwork.Mutate();
```

Uspješnost nove mreže postavljano na nulu i pozivamo mutaciju, ukoliko mreža bude manje uspješna od prethodne, u sljedećem pozivu prethodna mreža biti će kopirana.

6.3.*Training manager*

Klasa *NerualManager* služi svrhu *training managera* te je zadužena za kontrolu svakog pojedinog zrakoplova, odnosno njegove *AI Controller* komponente u odnosu na okruženje. Svaki zrakoplov ima svoju instancu klase.

OnGroundCollision detektira udar zrakoplova u zemlju, resetira scenarij treninga za zaduženi zrakoplov i šalje informacije AIControlleru koji zatim kažnjava odgovarajuće mreže i pokreće mutaciju.

```
public void OnGroundCollision(Interactable interactable)
```

Update se izvršava prilikom iscrtavanja svake pojedine slike na ekran. Njegov zadatak je provjera vremena i položaja zrakoplova u odnosu na točku u prostoru. Ukoliko je zrakoplov prošao zadanu točku dodjeljuje mu se nova točka i šalje informacija *AIControlleru* koji vrši nagrađivanje odgovarajućih mreža.

```
public void Update()
```

Update također poziva *reset* metodu ukoliko je vrijeme prolaska isteklo. Reset izvršava se prilikom poziva *Update* ili *OnGrooundCollision* metode. Zadatak je resetirati vrijeme i uvjete testiranja

na početnu točku bez potrebe za ponovnim pokretanjem aplikacije. *Reset* šalje informacije *AiControlleru* koji vrši kažnjavanje odgovarajućih mreža.

```
public void Reset()
```

Nakon resetiranja neuronskih mreža *AIController* pokreće *reset* motora zrakoplova i transformacije na inicijalnu poziciju kako bi se test mogao ponoviti. Taj proces također dovodi do mutacije mreže.

7. TRENIRANJE UPRAVLJANJA ZRAKOPLOVOM

Treniranje je integrirano *u SetAttitude, SetAltitude i SetThrottle* metoda *AIController*-a te dodatne informacije o uspješnosti mreže dolaze od Training managera u vidu prolaska kroz točku, prekoračenja vremena ili udarca u zemlju.

7.1. Bodovanje položaja zrakoplova

Ukoliko zrakoplov leti prema točki unutar 10% dozvoljene tolerancije, nije invertan te sila je pozitivna za svaku jedinicu vremena izvršavanja fizike[19] , vrijednost sposobnosti povećavamo za promil zbroja kutova. Također sposobnost mreže se kažnjava ukoliko je zrakoplov invertan ili je sila djelovanja negativna. Izračun se vrši u prosjeku 60 puta u sekundi. Mreže koje su nakon završenog ciklusa genetičkog algoritma postigle negativne bodove neće dobiti priliku za mutaciju. Iznimka je naravno prva mreža.

```
if (fit_h > 0.1f && fui_v > 0.1f && motor.Inverted == false && gForce == 1)
      {
         // goint into right direction
         AttitudeNetwork.AddFitness((fit_h + fui_v) * 0.01f);
         // DrawLine(transform.position, target.transform.position, Color.red, 0.5f);
      }

      // wa are inverted
      if (motor.Inverted == true)
      {
         AttitudeNetwork.AddFitness(-0.1f);
      }

      // negative force applied
      if (false && gForce == -1)
      {
         AttitudeNetwork.AddFitness(-0.1f);
      }
```

[19] Jedinica koja je potrebna računalu da obradi set fizičkih naredbi i varira ovisno o brzini računala.

7.2. Bodovanje visine leta

U izračunu bodovanja visine koristimo globalni vektor koji je vertikalan sa XZ ravninom zemlje te mjerimo udaljenost od visine tla. Ukoliko zrakoplov udari u zemlju mreža se kažnjava, no ne i odbacuje. Ostavlja se mogućnost napredovanja. Zrakoplov se kažnjava i za preniski let, odnosno ukoliko je margina od prolaska zemlje manja od 5m. Ukoliko visinomjer pređe maksimalnu visinu prikazivat će -1.

```
    // if we hit the ground this one failed
      Vector3 groundPosition = transform.position + (-Vector3.up *  motor.RadarAltitude);

// if we dont hit anything this one failed as well
if (radarAltitude < 5 && radarAltitude != -1)
{
    AltitudeNetwork.AddFitness(-5f);
}
```

Mreža na temelju samo ove logike bi trebala imati bolji rezultat ukoliko leti unutar dozvoljene visine, odnosno više od 5m, no u realnosti ovisit će o radu mreže položaja zrakoplova.

7.3. Bodovanje brzine leta

Izračun brzine leta se kažnjava ukoliko je brzina zrakoplova prekomjerno spuštena, u tom će slučaju performanse mreže položaja zrakoplova biti narušene.

Nagrađuje se let većom brzinom radi prolaska virtualnom stazom u što kraćem vremenu. Brži let također generira veći uzgon na krilima.

```
if (output[0] < 0.5f)
{
    ThrottleNetwork.AddFitness(-0.01f);
}
else
    ThrottleNetwork.AddFitness(0.01f);
```

7.4. Bodovanje prolaska kroz točku

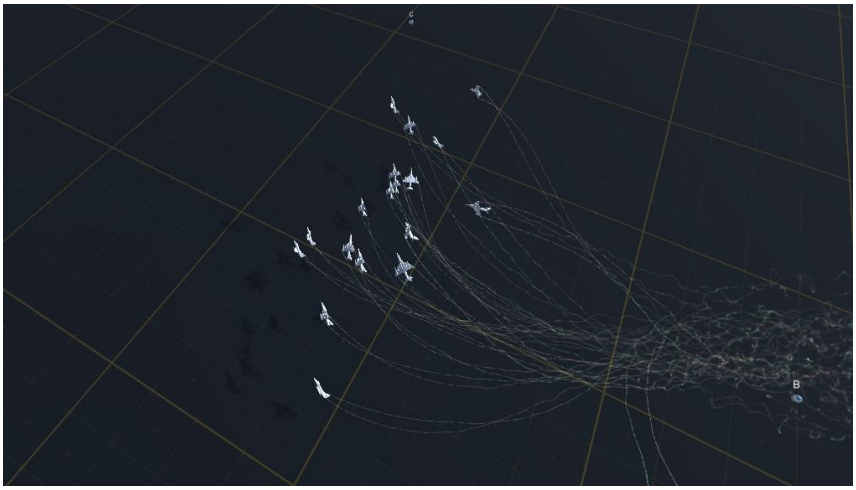
Bodovanje prolaska kroz točku nagrađuje globalno *AI controller* koji pojedinačno nagrađuje sve mreže sa vrijednošću sposobnosti povećanom za preostalo vrijeme umanjeno od ukupnog dozvoljenog vremena. Bodovanje ima najveći utjecaj na rad mreže jer time vrlo brzo odbacujemo mreže koje nisu sposobne izvršiti primarni zadatak.

```
float distance = Vector3.Distance(transform.position, nextWaypoint.transform.position);

    if (distance <= offset)
    {
       index++;
       if (index >= Waypoints.Count)
          index = 0;

       nextWaypoint = Waypoints[index];

       best_time += timer;

       // Aircraft brain
       brain.target = nextWaypoint.transform;
       brain.SetWaypointHit (_TimePerWaypoint - timer);
       _totalWaypointCount++;

       timer = 0;

       if (_totalWaypointCount == Waypoints.Count)
       {
          BestLapPtime = best_time;
          brain.SetBestTime(BestLapPtime);
          _totalWaypointCount = 0;
          best_time = 0;
       }
    }

    // set time left
    brain.SetWPTime(_TimePerWaypoint - timer);


    // time up for this agent
    if (timer > _TimePerWaypoint)
       Reset();
```

8. TESTIRANJE I SIMULACIJA UČENJA

U procesu učenja zrakoplovi se kreću sa razmakom od 0.2 sekunde sa iste početne točke te se kamera fokusira na zrakoplov sa najviše prolaznih točki. Zrakoplov inicijalno ima potpuno nasumično odabrane težine neuronskih mreža, a vrijednosti sposobnosti tih mreža su jednake nuli, što bi značilo da mreža „ne zna“ upravljati zrakoplovom niti zna što je njena svrha. Mreža će „pokušati“ pomicati kontrole u odnosu na primljene ulaze i čekati potvrdu genetskog algoritma nakon kolizije sa zemljom ili isteka vremena. Svaka sljedeća generacija imat će veću šansu za prolazak do sljedeće točke. Obzirom na samostalno učenje svakog zrakoplova, nakon određenog vremena veći broj zrakoplova ponašat će se poput roja ili jata.

Slika 19. Put zrakoplova između točaka B i C



Izvor: autor

U prosjeku potrebno je 18 generacija prije nego zrakoplovi uspiju završiti zadanu stazu.

Slika 20. Udarac u zemlju

Izvor: autor

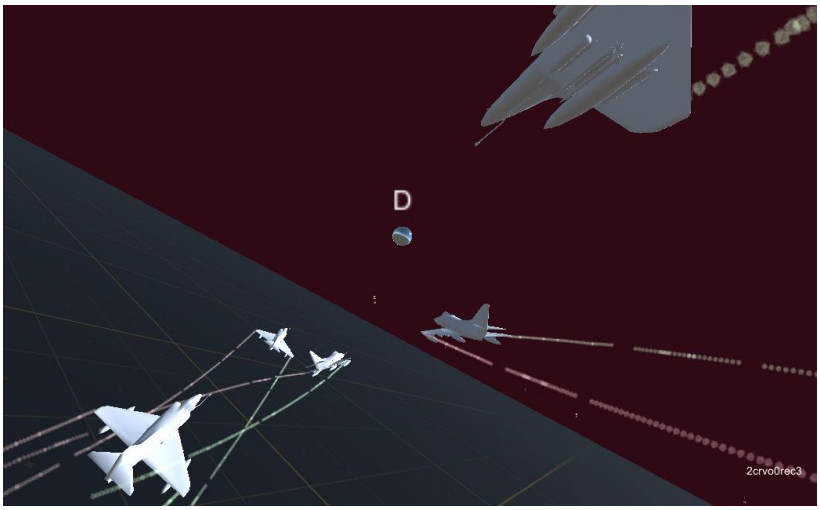
U primjeru na slici 20 vidimo da se razmak pojedinih zrakoplova smanjio što je znak pojave neuronskih mreža sa boljim performansama koje imaju i bolje prolazno vrijeme, odnosno brzinu prolaska kroz zadane točke. Također zrakoplovi koji nisu uspjeli dostići točku ili su udarili u zemlju resetiraju se na početnu poziciju. Kolizija sa ostalim zrakoplovima je isključena kako međusobne kolizije ne bi utjecale na performanse mreže upravljanja. Pozicije i putanje ostalih zrakoplova morale bi biti pridodane ulazima upravljačke mreže kako bi mreža te podatke mogla obraditi u svom treningu.

Slika 21. Upravljanje zrakoplovima

Izvor: autor

Osim brzine prolaska svaki zrakoplov prolazi na temelju vlastitog treninga najbolji put prema drugoj točki. Varijacije u vertikalnim i horizontalnim prolascima zadanih točka vidljive su obzirom da se radi o trodimenzionalnom prostoru.

Slika 22. Vertikalni prikaz prolaska točke u prostoru

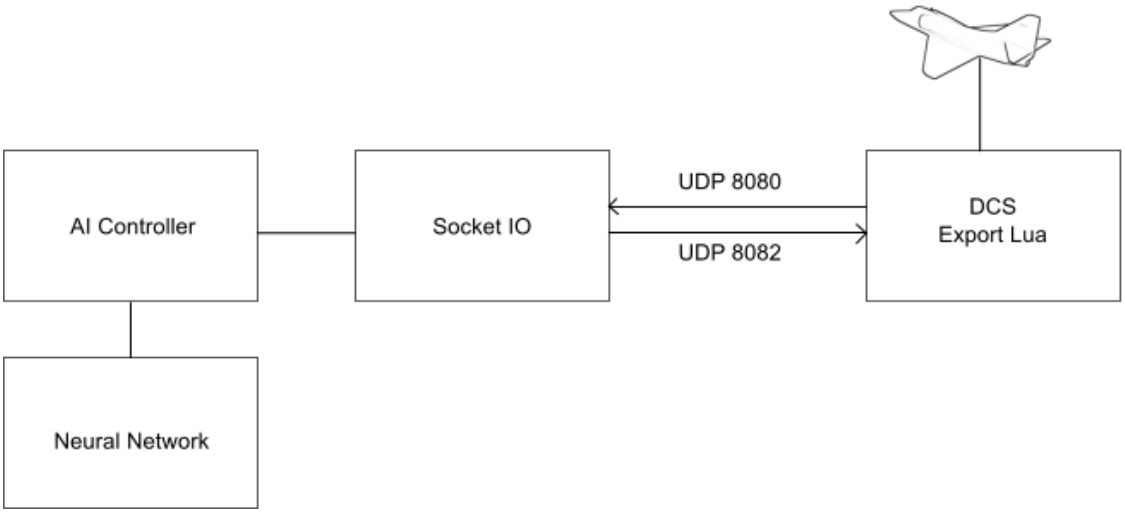


Izvor: autor

Po završetku treniranja, težine mreže se izvoze u tekstualnu datoteku i koriste za nastavak testiranja bilo da se radi o simplificiranom simulatoru ili o testiranju unutar DCS simulatora.

Ulazni i izlazni parametri iz DCS simulatora leta su pristupani pomoću DCS API i lua programskog jezika. Korišten je UDP protokol za komunikaciju između C# AI modula i DCS simulatora. Komunikacija se odvija putem unutar zasebne niti (engl. *Thread*) te koriste se dva port-a, odnosno 8080 za slušanje dolaznih podataka iz DCS simulatora i 8082 za slanje podataka DCS simulatoru.

Slika 23. Prikaz protokola za spajanje



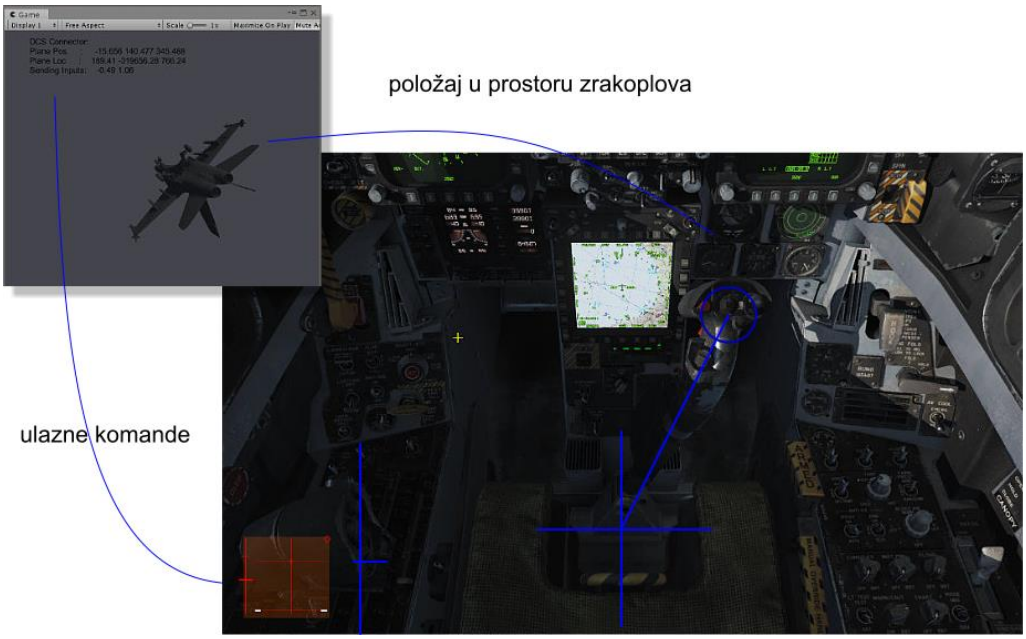
Izvor: autor

Obrada neuronske mreže i upravljanje je postignuto simplificiranijom verzijom AI kontrolera. Metode za evoluciju i bodovanje nisu potrebne već je dovoljna samo interpretacija mreže.

Slika 24. Prikaz očitavanja komandi unutar DCS simulatora

Izvor: autor

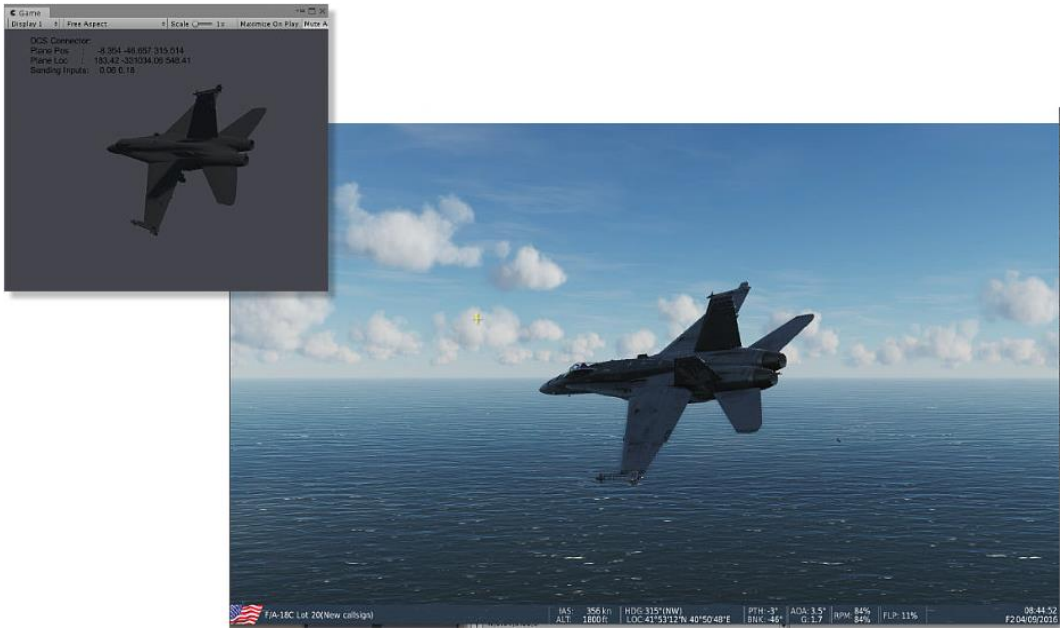
Slika 25. Pomak komandi zrakoplova



Izvor: autor

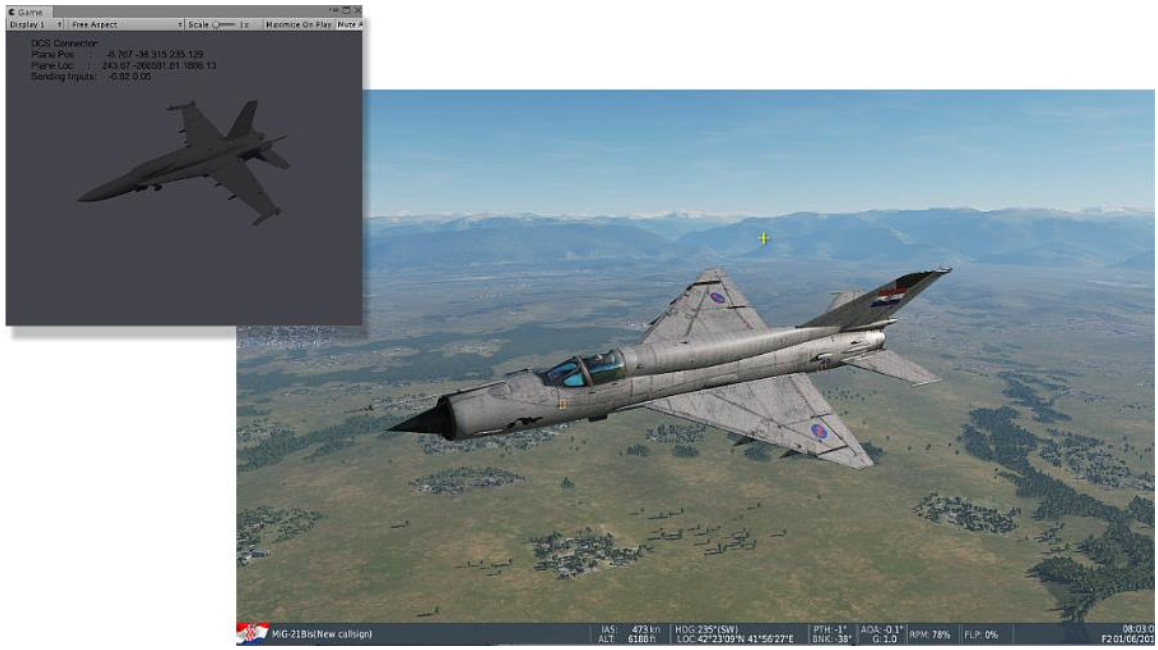
Pomaci komandi zrakoplova i očitavane pozicije te položaja zrakoplova odgovaraju podacima primljenim unutar aplikacije za upravljanje. Iako svaki zrakoplov reagira drugačije na ulazne parametre komandi, neuronska mreža upravljanja dosta dobro se uklopila i uspjela je nekoliko različitih tipova zrakoplova uspješno provesti kroz set predefinirani navigacijskih točki.

Slika 26. Eksterni pogled na zrakoplov FA 18



Izvor: autor

Slika 27. Eksterni pogled na zrakoplov Mig 21



Izvor: autor

9. ZAKLJUČAK

Pravilna implementacija neuronske mreže pruža mogućnost zaključivanja naizgled jednako ljudskom mozgu. Iako trenutna snaga računala nije dovoljno velika za simulaciju jednakih osobina ljudskog mozga, neuronske mreže za razliku od mozga svoj zadatak mogu izvršavati repetitivno bez zamora što je izuzetna prednost. Neuronske mreže su duboko integrirane u različite aplikacije ili servise koje svakodnevno koristimo i njihov rezultat se bazira na postotku neke vrijednosti, što bi značilo da neuronska mreža će prije ili kasnije producirati greške. Razlog tome je različitost ulaznih podataka i varijacija u treniranju mreže.

U primjeru upravljanja zrakoplova ili sustava gdje se dovodi u pitanje sigurnost ljudskih života i imovine, neuronske mreže mogu biti previše nepredvidljive. Sustavi poput autopilota su potpuno predvidljivi i pilot u svakom trenutku može očekivati i kontrolirati njihovu radnju. Problemi nastaju kad parametri autopilota izlaze van granica definiranih za ispravan rad sustava, tada se takav sustav automatski isključuje i prepušta radnju pilotu. Općenito u trenucima kad autopilot ne može upravljati zrakoplovom, upravljanje je otežano i za samog pilota odnosno posadu zrakoplova, bilo da se radi o lošim vremenskim prilikama ili mehaničkom kvaru na zrakoplovu i na žalost odluke određenih pilota kroz povijest nisu bile dovoljno dobre što je rezultiralo zrakoplovnom nesrećom. Određene zrakoplovne nesreće su se mogle izbjeći da je postojao sustav koji bi sugerirao pilota na temelju ulaznih parametara i sugerirao odluke na temelju prošlih iskustava, primjerice ukoliko je došlo do zaleđivanja određenih površina zrakoplova, na temelju usporedbe očitavanja instrumenata sustav bi izolirao neispravne instrumente u realnom vremenu.

Početno treniranje neuronske mreže može se izvršiti u sigurnom okruženju poput različitih simulatora te već postavljenu mrežu koristiti kao polaznu mrežu za nastavak kompleksnijeg treniranja. Takav primjer učenja već primjenjuje Tesla u automobilskoj industriji.

Osim industrije i transporta, neuronske mreže su vrlo dobro sredstvo za učenje i testiranje, primjerice mreža će pokušati koristiti sve propuste u kodu kao što je bilo invertno upravljanje zrakoplova u cilju postizanja boljeg rezultata, a ne kažnjavanje takve akcije je bio upravo propust prilikom izrade aplikacije gdje je jedna od prvih generacija neuronske mreže rezultirala ispravljanjem grešaka na

inicijalnoj aplikaciji upravljanja zrakoplova iako je ista aplikacija bila testirana od strane čovjeka. Određeni slični zadaci mogu se rješavati umjetnom inteligencijom baziranom na neuronskim mrežama gdje će se potpuno ista instanca neuronske mreže prilagoditi objektu za koji je zadužena. Uzmimo za primjer simulaciju prometa u gradu gdje ista mreža za upravljanje vozila će se drugačije ponašati na temelju drugačijih ulaza i samog tipa vozila. Ukoliko nastavimo proces učenja mreža će otkriti nove prednosti i limite upravljanog vozila.

Implementacija neuronske mreže je jednostavna, pogotovo uz veliki broj dostupnog koda putem interneta sa različitim tipovima neuronskih mreža i podacima za treniranje. Ono što je najveća prepreka svake neuronske mreže je uspješno treniranje iste i okruženje u kojem će se to treniranje izvoditi.

LITERATURA

1. Automatika, uvod u neuralne mreže https://www.automatika.rs/baza-znanja/neuralne-mreze/uvod-u-neuralne-mreze.html (10. 8.2019.)
2. Bašić Dalbelo, B., Čupić, M., Šnajder, J., Umjetne neuronske mreže, Fakultet elektrotehnike i računarstva Zagreb, svibanj 2008., https://www.fer.hr/_download/repository/UmjetneNeuronskeMreze.pdf (10. 8. 2019.)
3. Genetic Algorithm Tutorial -How to Code a Genetic Algorithm, GitHub, 1. 8. 2017., https://github.com/ptrkkim/Genetic-Algo-Tech-Talk (15. 7. 2019.)
4. Gori, M., Machine learning, Feedforward Neural Network, 2018., https://www.sciencedirect.com/topics/computer-science/feedforward-neural-network (15. 7. 2019.)
5. Ilić, V., Neuronske mreže, Solair, studenti 1999., http://solair.eunet.rs/~ilicv/neuro.html (10. 6. 2019.)
6. Jiaconda, A Concise History of Neural Networks, Towards Dana Science, 7. 3. 2019., https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec (12. 6.2019.)
7. Strachnyi, K., Brief History of Neural Networks, 23. 1. 2019., https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec (12. 7. 2019.)
8. The One, Tutorial On Programming An Evolving Neural Network In C# w/ Unity3D, 29. 5. 2017., https://www.youtube.com/watch?v=Yq0SfuiOVYE , (10. 6. 2019.)
9. Upadhyay, Y., Introduction to FeedForward Neural Networks, Towards Dana Science, https://towardsdatascience.com/feed-forward-neural-networks-c503faa46620 (7. 3. 2019.)

POPIS SLIKA

Slika 1. Usporedba biološkog i umjetnog neurona 2
Slika 2. Simplificirani prikaz obrade neuronskom mrežom 3
Slika 3. Donošenje odluka neuronskom mrežom 4
Slika 4. Prikaz donošenja odluke 5
Slika 5. Aktivacijska funkcija u neuronu 7
Slika 6. Prikaz izračuna mreže za upravljanje zrakoplovom (skriveni slojevi) 8
Slika 7. Koncept simulatora 10
Slika 8. Sile koje djeluju na zrakoplov 11
Slika 9. Utjecaj komandi zrakoplova 12
Slika 10. Kut A između osi kretanja zrakoplova i pravca prema željenoj destinaciji 13
Slika 11. Kut B između osi kretanja zrakoplova i pravca prema željenoj destinaciji 14
Slika 12. Korekcija puta zrakoplova u odnosu na udaljenost od zemlje 15
Slika 13. Korekcija brzine zrakoplova ovisno o položaju i udaljenosti od tražene točke 16
Slika 14. Bodovanje uspješnosti algoritma 18
Slika 15. Proces treniranja NN zrakoplova 19
Slika 16. Putanja zrakoplova i bodovanje 20
Slika 17. Mutacije težine veze pojedinog neurona 21
Slika 18. Povezanost klasa 24
Slika 19. Put zrakoplova između točaka B i C 33
Slika 20. Udarac u zemlju 34
Slika 21. Upravljanje zrakoplovima 34
Slika 22. Vertikalni prikaz prolaska točke u prostoru 35
Slika 23. Prikaz protokola za spajanje 36
Slika 24. Prikaz očitavanja komandi unutar DCS simulatora 36
Slika 25. Pomak komandi zrakoplova 37
Slika 26. Eksterni pogled na zrakoplov FA 18 38
Slika 27. Eksterni pogled na zrakoplov Mig 21 38