SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2363

IZBJEGAVANJE PREPREKA DUBOKIM POTPORNIM UČENJEM

Pavao Jerebić

Zagreb, veljača 2021.

ZAHVALA

Zahvaljujem se mentoru prof. dr. sc. Domagoju Jakoboviću na razumijevanju, savjetovanju i pomoći, kako pri odabiru teme tako i tijekom same razrade ovog diplomskog rada.

Veliku zahvalu zaslužuje i moja obitelj zbog velike podrške tijekom cijelog mog obrazovanja.

Sadržaj

Uvod

Upravljanje autonomnim vozilima je izazov koji je jako popularan u zadnjih nekoliko godina. Rješavanje tog problema bi uvelike utjecalo na transport ljudi i dobara [1]. Jedna od prepreka u ostvarivanju tog cilja je upravljanje vozilom u nepoznatim situacijama. Zbog uvjeta robusnosti na različita okruženja potrebno je razmotriti mogućnost korištenja neuronskih mreža za ovaj zadatak. Raznim metodama moguće je postići bolju robusnost modela, što je za ovaj problem jako bitno. Stvaranjem simulacija u kojima možemo učiti model bez posljedica u stvarnom svijetu, omogućujemo učenje modela za ovakve zadatke. Razvojem računalnih mogućnosti brzina i uvjerljivost simulacija raste. Zbog toga možemo učiti modele kojima treba jako puno podataka i jako puno vremena. Kombiniranjem više vrsta strojnog učenja, kao što su potporno učenje (eng. *Reinforcment Learning*) i duboko učenje(eng. *Deep Learning*), dolazimo do jako složenih modela koji bi mogli biti u stanju dati točna rješenja na problem izbjegavanja prepreka.

Najvažnija informacija kod izbjegavanja prepreka je njihova udaljenost. Ta informacija se može dobiti iz modula poput RGB-D kamere, radara ili lidara. Budući da su takve komponente puno skuplje od obične kamere, potrebno je istražiti mogućnost predviđanja dubine iz slike dobivene jednom kamerom. Problem je osjetljivost kamere na svjetlost i kontrast, ali u kontroliranim uvjetima, na primjer skladišta, ne bi trebalo biti značajnih problema.

Prvi dio rada opisuje metode strojnog učenja korištene dalje u radu. Predstavljeno je potporno učenje i duboko učenje. Dani su razni primjeri neuronskih mreža. U drugom dijelu je dan model okruženja i ponašanja drona. Prikazano je sučelje programskog sustava Unity te je opisana priprema okruženja za učenje. Treći dio se bavi arhitekturom neuronske mreže za predviđanje dubine iz slike i predstavljeni su njeni rezultati. Algoritmi izbjegavanja prepreka se opisuju u četvrtom poglavlju. Predstavljeni su neki heuristički algoritmi. Opisan je model dubokog potpornog učenja, arhitektura neuronske mreže i proces učenja. U petom poglavlju se nalaze rezultati algoritama za izbjegavanje prepreka. Prikazani su grafovi na kojima su rezultati za svaki algoritam i usporedba algoritama za svaku od scena za testiranje. U zaključku se nalazi kratak osvrt na rezultate i moguća daljnja poboljšanja.

1. Metode strojnog učenja

1.1. Potporno učenje

Potporno učenje je vrsta strojnog učenja kod kojeg agent uči kako provoditi akcije u zadanom okruženju kako bi maksimizirao ukupnu nagradu. Razlika od klasičnih problema strojnog učenja je ta da nisu jasno definirani parovi ulaz/izlaz, već ih agent sam pronalazi kroz faze istraživanja (eng. *exploration*) i iskorištavanja (eng. *exploitation*). Okruženje se najčešće može predstaviti kao Markovljev proces odlučivanja gdje su definirana stanja i akcije uz pomoć kojih se prelazi iz jednog stanja u drugo.

Markovljev proces odlučivanja (MDP) se sastoji od: skupa stanja S, skupa akcija A, funkcije prijelaza *T(s,a,s')* koja opisuje vjerojatnost da pomoću akcije *a* dođemo iz stanja *s* u *s'*. Funkcija nagrade *R(s,a,s')* opisuje nagradu koju agent dobiva odabranom akcijom. Svaki MDP ima i početno stanje, a može imati i konačna stanja. Također može imati i faktor propadanja γ koji smanjuje utjecaj nagrada u kasnijim koracima.

Cilj algoritama potpornog učenja je odabrati optimalnu politiku (eng. *policy*). Politiku označavamo izrazom $\pi(s)$. Ta funkcija vraća optimalnu akciju za stanje *s*. Vrijednost nekog stanja, to jest, očekivanu vrijednost nagrade označavamo izrazom *V(s)*. Funkcija kojom opisujemo očekivanu vrijednost akcije a u stanju s je dana u izrazu *Q(s,a).*

Jedna od osnovnih metoda izračunavanja optimalne politike je iteriranje vrijednosti [2] gdje ažuriramo funkciju vrijednosti izrazom :

$$V_{k+1}(s) \leftarrow max_a \sum_{s\prime} T(s,a,s') * [R(s,a,s') + \gamma V_k(s')] \quad (1)$$

Optimalnu politiku biramo maksimiranjem očekivanja vrijednosti sljedećeg stanja korištenjem trenutne akcije, odnosno:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s\prime} T(s,a,s')V(s') \quad (2)$$

1.1.1. Q-učenje

Druga metoda je Q-učenje. U toj metodi u hodu ažuriramo Q-vrijednosti prolaskom kroz MDP krećući od početnog stanja. Time smanjujemo prostor pretraživanja tako da ne prolazimo kroz sva moguća stanja. Q-vrijednosti ažuriramo izrazom (3), gdje je α stopa učenja. Politika za takav tip učenja je jednaka izrazu (4).

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \quad (3)$$

$$\pi^*(s) = \operatorname*{argmax}_a \sum_{s'} Q(s,a) \quad (4)$$

Obavljanjem akcija dolazimo u nova stanja dok konačno ne dođemo u neko od konačnih stanja. Takav niz koraka, od početnog do konačnog stanja se zove epizoda. Učenje Q-vrijednosti se onda ponavlja kroz više epizoda. Algoritam Q-učenja je dan u pseudokodu 1.1.

```
Algoritam Q-učenje
Za svaku epizodu:
  s <- početno stanje epizode
  Za svaki korak epizode:
    Izračunaj akciju a po izrazu (4)
    izvedi akciju a, očitaj nagradu r, novo stanje s'
    Ažuriraj Q(s,a) po izrazu (3)
    s <- s'
Dok s nije završno stanje
```

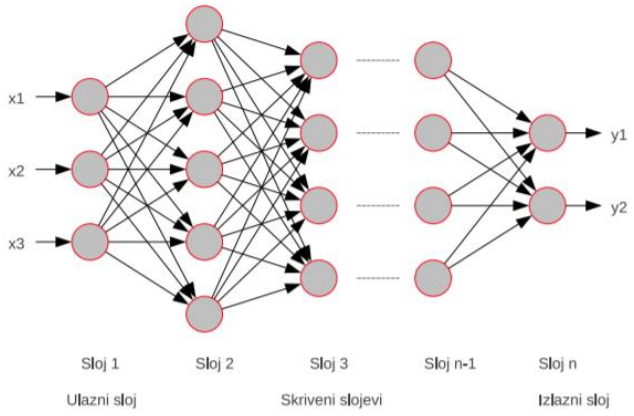
Pseudokod 1.1 Algoritam Q-učenja

1.2. Duboko učenje

Unaprijedna neuronska mreža je osnovni duboki model čiji je cilj aproksimacija funkcije $y = f(x)$ za dane parove x, y.

Osnovni građevni element ovog modela je neuron. Neuron je struktura koja se sastoji od težina $\boldsymbol{w}$ i b i aktivacijske funkcije f. Neuron kao ulaz prima vektor $\boldsymbol{x}$, a na ulazu daje skalar čija je vrijednost $y = f(\boldsymbol{w}^T \boldsymbol{x} + b)$.

Neuronska mreža se sastoji više slojeva, a svaki sloj sadrži više neurona. Svaka mreža počinje od ulaznog sloja neurona i može sadržavati skrivenih slojeva te završava izlaznim slojem. Primjer takve mreže vidimo na slici 1.1. Aktivacijske funkcije u ovim modelima su najčešće sigmoida, tangens hiperbolni ili zglobnica.



Slika 1.1 Unaprijedna neuronska mreža

Kao i kod svakog problema strojnog učenja, potrebno je odabrati funkciju gubitka koja će se optimirati. Kod neuronskih mreža najčešće se koriste algoritmi gradijentnog spusta, a za funkciju gubitka zbroj kvadratnih odstupanja. Izračunava se parcijalna derivacija funkcije gubitka L za svaki sloj mreže i težine se ažuriraju izrazom (5).

$$w^{k+1} = w^k - \alpha \frac{\partial}{\partial w^k} L(w^k) \tag{5}$$

Oznaka težine k se odnosi na korak algoritma gradijentnog spusta. Parametar α određuje stopu učenja. Ako je stopa prevelika onda dolazi do divergencije, a ako je premala onda sporo konvergira u optimalno stanje.

Dodavanjem momenta, odnosno prethodnu promjenu težina, možemo ubrzati konvergenciju, a možda i izbjeći lokalni optimum povećavanjem. Taj proces ažuriranja je definiran u izrazima (6) i (7), gdje je μ stopa momenta.

$$\Delta w^k = \frac{\partial}{\partial w^k} L(w^k) \tag{6}$$

$$w^{k+1} = w^k - \alpha \frac{\partial}{\partial w^k} L(w^k) - \mu \Delta w^{k-1} \tag{7}$$

ADAM (eng. *ADAptive Moment estimation*) algoritam je nadogradnja na prijašnje algoritme. ADAM računa eksponencijalni pomični prosjek gradijenta i kvadratnog gradijenta. Time se korigira magnituda gradijenta i dodaje robusnost na početni odabir hiperparametara. Algoritam je dan u pseudokodu 1.2.

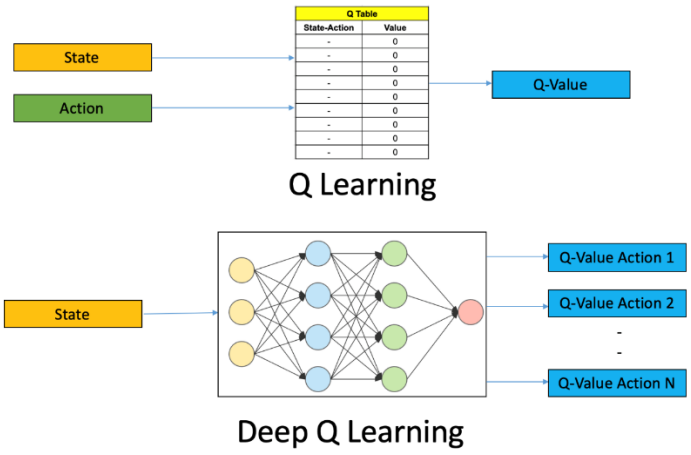
```
Algoritam ADAM
α, β1, β2, ε
m0 <- 0
v0 <- 0
t <- 0
θ0 <- inicijalizacija parametara
ponavljaj dok θ ne konvergira:
  t <- t + 1
  gt <- ∇θ ft(θt-1)
  mt <- β1*mt-1 + (1-β1)*gt
  vt <- β2*vt-1 + (1-β2)*gt^2
  mt' <- mt / (1-β1^t)
  vt' <- vt / (1-β2^t)
  θt <- θt-1 - α*mt' / (√(vt') + ε)
kraj ponavljanja
```

Pseudokod 1.2 Algoritam ADAM

Prema radu [3] preporučeni parametri su $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$.

1.2.1. Duboko Q-učenje

Razlika običnog i dubokog Q-učenja (eng. *Deep Q-learning*) je u tome što ne pamtimo tablicu svih parova stanja i akcija s pripadajućim vrijednostima, nego za to koristimo neuronsku mrežu. Na ulaz neuronske mreže postavljamo trenutno stanje, a na izlazima predviđenu Q-vrijednost za svaku od mogućih akcija. Tako naša duboka neuronska mreža implicitno pamti u sebi klasičnu tablicu Q-vrijednosti. Usporedbu možemo vidjeti na slici 1.2.



Slika 1.2 Usporedba klasičnog i dubokog Q-učenja [4]

1.2.2. Konvolucijske neuronske mreže

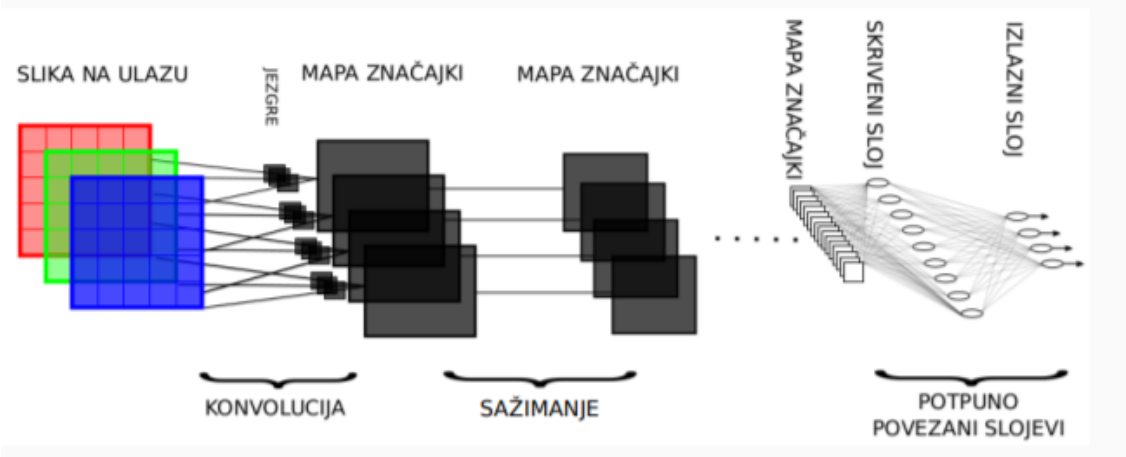
Konvolucijski modeli su modeli specijalizirani za podatke s topologijom rešetke, npr. slike, vremenski sljedovi. Konvolucijski modeli se, uz potpuno povezane slojeve, sastoje i od konvolucijskog sloja. Svaki konvolucijski sloj se sastoji od više filtara. Svaki filtar provodi konvoluciju ulaznog tenzora s jezgrom. Jezgra filtra je tenzor koji sadrži težinske faktore kojima se množe vrijednosti ulaznog tenzora. Tu operaciju možemo prikazati kao:

$$s(t) = (x * w)(t) = \sum_{\tau=\tau_{min}}^{\tau_{max}} x(\tau)w(t+\tau), \tag{8}$$

gdje je x ulazni tenzor, w jezgra filtra i s izlaz konvolucijskog filtra.

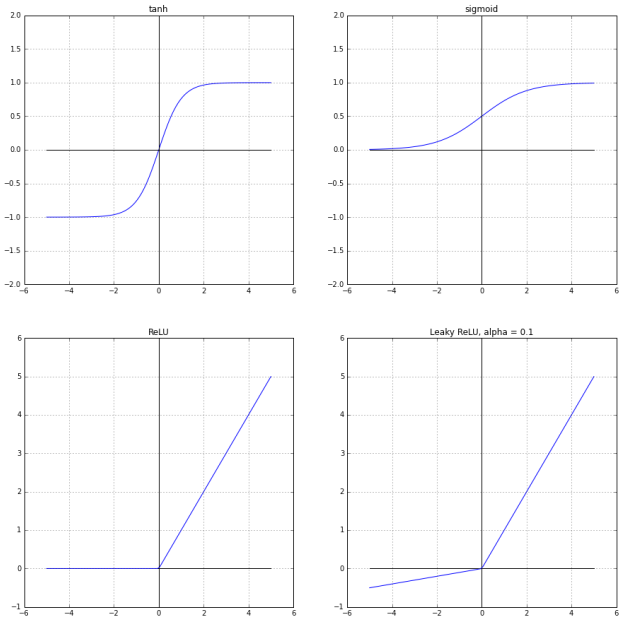
Prednost konvolucijskih modela nad unaprijednim modelima je da mogu bolje aproksimirati podatke koji imaju hijerarhije. Kad pokušavamo naučiti model prepoznavati ljude na slici, konvolucijski model može u jednom sloju naučiti prepoznavati crte, u sljedećem sloju dijelove tijela, u sljedećem udove i glavu te u zadnjem cijelog čovjeka.

Najčešće se konvolucijski model sastoji od nekoliko konvolucijskih slojeva nakon kojih slijede potpuno povezani dio koji uči ispravno klasificirati podatke dobivene iz izlaza konvolucijskih filtera. Na slici 1.3 vidimo klasičan način korištenja konvolucijskih slojeva u neuronskim mrežama.



Slika 1.3 Konvolucijska neuronska mreža

Nakon konvolucijskog sloja često slijedi aktivacijski sloj. U tom sloju primjenjujemo neku aktivacijsku funkciju na izlaz prethodnog sloja. Najčešće funkcije možemo vidjeti na slici 1.4. Svaka od funkcija ima svoje prednosti i mane. Ove funkcije su nelinearne pa možemo riješiti i takve vrste problema. Problem sigmoidalne funkcije je da je teška za računanje, nije centrirana oko nule i kad je ulaz jako velik ili jako malen gradijent je blizu nule. Tangens hiperbolni također ima taj problem, ali je centriran oko nule. Funkcije ReLU i Leaky ReLU nisu teške za računanje i problem nestajanja gradijenta je riješen za jako velike i male brojeve. Kod ReLU funkcije nema gradijenta ako je vrijednost manja od nule, ali to rješava Leaky ReLU varijanta jer ima neku manju vrijednost za ulaz manji od nule pa rješava i taj problem. Izrazi ovih funkcija su dani u jednadžbama (9).



Slika 1.4 Aktivacijske funkcije

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

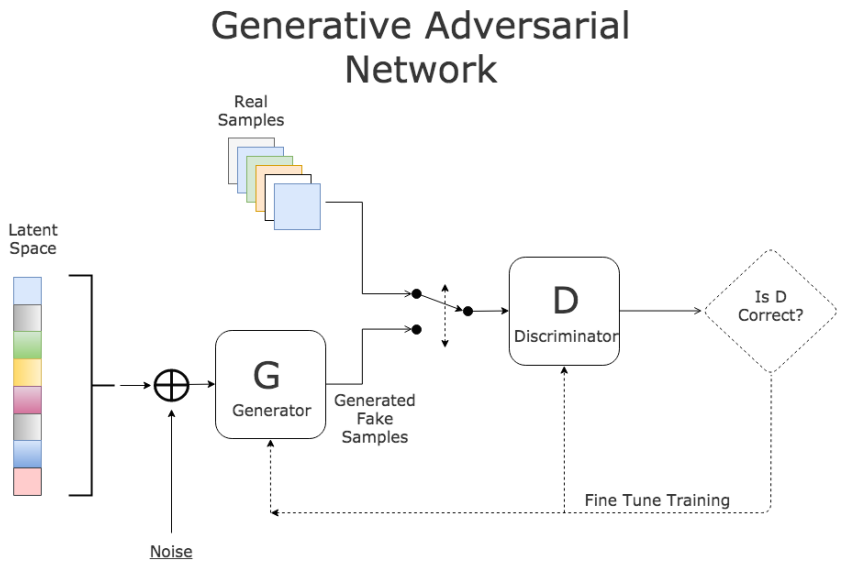
$$sigmoid(x) = \frac{1}{1 - e^{-x}}$$

$$ReLU(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{9}$$

$$LeakyReLU_{\alpha}(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

1.2.3. Generativne suparničke mreže

Generativne suparničke mreže (eng. *Generative Adverserial Network – GAN*) su neuronske mreže koje služe za generiranje podataka. Ovaj model se razvio zbog restrikcija dotadašnjih generativnih modela. GAN se sastoji od dva glavna dijela, generatora i diskriminatora, što je vidljivo na slici 1.5 Generator generira što uvjerljiviji izlaz, a diskriminator pokušava razaznati je li izlaz generatora stvarna ili lažna slika. Ta dva modela se mogu zamisliti kao dva igrača u igri. Što je generator uvjerljiviji to diskriminator mora biti bolji, i obrnuto. Učenje ta dva modela se može zamisliti kao traženje Nashove ravnoteže, što je ujedno i optimum ovakvog modela.



Slika 1.5 Generativna suparnička mreža [5]

Generator na ulaz prima šum *z* iz uniformne distribucije i generira izlaz $\tilde{x}$, iz distribucije p_z. Diskriminator prima na ulaz stvarne podatke iz zadane distribucije p_{data}. Ako je izlaz iz

generatora, uz parametre θ_G , jednak $G(z|\theta_G)$, a izlaz iz diskriminatora, uz parametre θ_D, jednak $D(x|\theta_D)$, onda diskriminator minimizira izraz (10), a generator izraz (11).

$$L_D(\theta_D, \theta_G) = -E_{x\sim p_{data}}\left[\log\left(D(x|\theta_D)\right)\right] - E_z[\log\left(1 - D(G(z|\theta_G)|\theta_D)\right)] \quad (10)$$

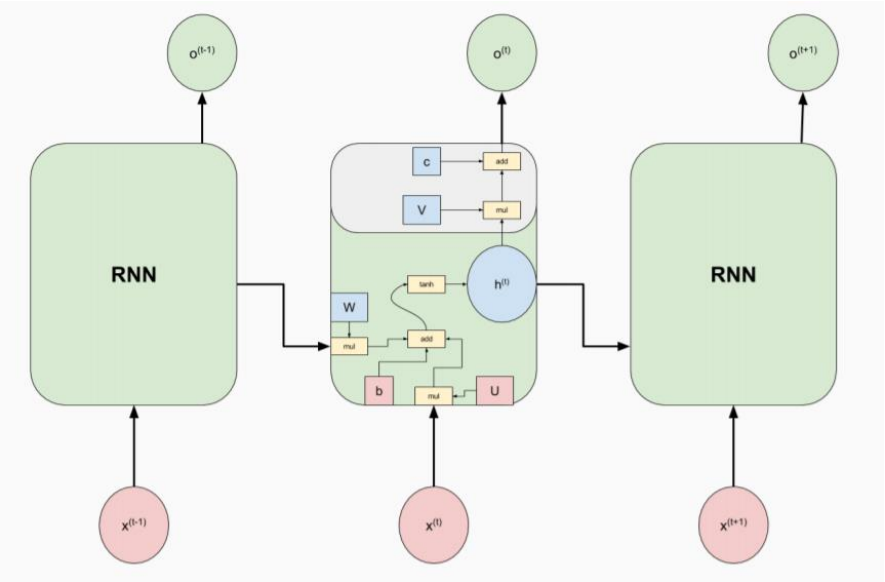
$$L_G(\theta_D, \theta_G) = -L_D(\theta_D, \theta_G) \quad (11)$$

Iz toga vidimo da je učenje ova dva modela minimax igra, tj. generator minimizira maksimum diskriminatora.

1.2.4. Povratne neuronske mreže

Povratne neuronske mreže (eng. *recurrent neural network*, RNN) su podvrsta umjetnih neuronskih mreže gdje ulaz nije samo trenutni primjer već i primjeri koji su mu prethodili. RNN-ovi koriste unutarnje stanje, memoriju, kako bi obradili niz ulaza. Zbog toga su ove mreže korisne u prepoznavanju rukopisa ili zvuka.

Svaki neuron u povratnoj neuronskoj mreži se zove povratna ćelija. Svaka ćelija se može interpretirati kao jedan sloj neuronske mreže u kojoj je prijenosna funkcija tangens hiperbolni. Povratna ćelija je prikazana na slici 1.6.



Slika 1.6 Povratna ćelija

Izlaz ćelije možemo prikazati izrazom:

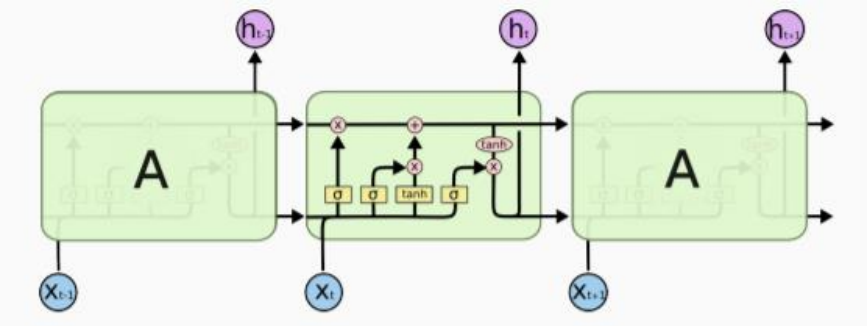
$$h^{(t)} = \tanh\left(Wh^{(t-1)} + Ux^t + b\right). \quad (12)$$

Jedna od popularnih varijanti povratnih ćelija je LSTM (eng. *Long short-term memory*). LSTM uvodi zaboravljanje dijela informacija iz prošlog stanja. Na slici 1.7 vidimo prikaz LSTM ćelije. Ćelija se sastoji od tri propusnice. Propusnica računa koliki će se dio ulazne informacije zaboraviti, odnosno propustiti na izlaz. Propusnica f je propusnica zaboravljanja, g je propusnica novog ulaza, a s je izlaz ćelije. Propusnice definiramo izrazima:

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f) \tag{13}$$

$$g^{(t)} = \sigma(W_g h^{(t-1)} + U_g x^{(t)} + b_g) \tag{14}$$

$$s^{(t)} = f^{(t)} s^{(t-1)} + g^{(t)} \sigma(W_h h^{(t-1)} + U x^{(t)} + b_s) \tag{15}$$

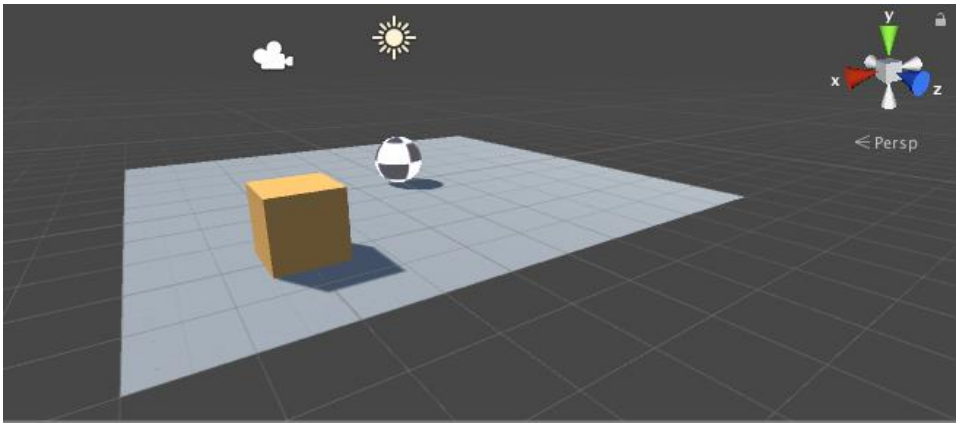


Slika 1.7 LSTM ćelija

2. Programski sustav Unity

Unity je program koji omogućava izradu 3d i 2d video igara te izradu simulacija. U Unity engine-u je vrlo jednostavno moguće stvaranje 3d okruženja, simulacije fizike i definiranje ponašanja agenata u sustavu.

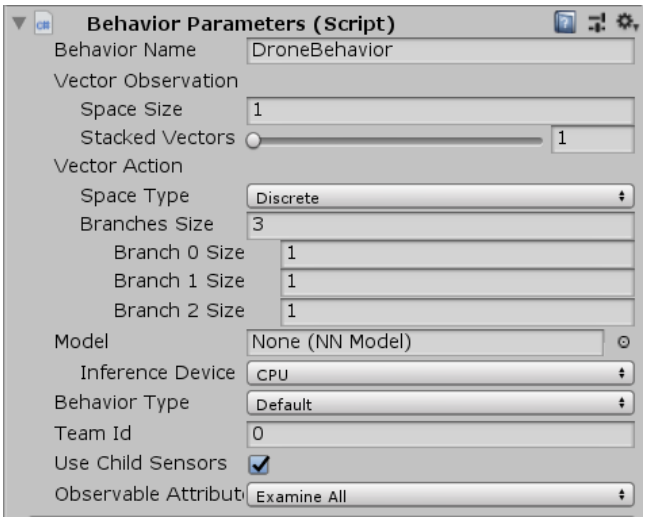
Stvaranjem nove scene stvaramo novo okruženje. Unutar te scene dodajemo elemente našeg svijeta poput tla, zidova, cilja i agenta. Pisanjem skripti u programskom jeziku C# opisujemo ponašanje našeg agenta ovisno o akcijama koje igrač odabere. Jedna jednostavna scena je prikazana na slici 2.1.



Slika 2.1 Scena u Unity-u

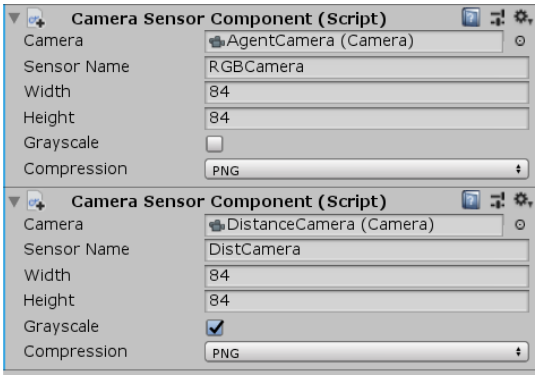
Dodavanjem Unity ML-Agents Toolkit (ML-Agents) plugin-a omogućavamo lakše učenje inteligentnih agenata. Plugin se sastoji od dva dijela. Prvi dio je kod koji omogućava komunikaciju između izvršne datoteke stvorene u Unity-u i Python skripti u kojima se obavlja učenje. Drugi dio se sastoji od Python skripti u kojima su implementirani neki algoritmi strojnog učenja.

Kako bi omogućili komunikaciju potrebno je na agenta dodati Behavior Parameters objekt. Primjer tog objekta je na slici 2.2. U tom objektu se definira koliko postoji akcija i koliko agent ima senzora.



Slika 2.2 Behavior Parameters objekt

Također možemo dodati i senzore za kamere. Njih dodajemo tako da na agenta dodamo objekt Camera Sensor prikazan na slici 2.3.



Slika 2.3 Senzori za kamere

Kroz izvođenje, agent prikuplja opažanja koja su definirana i prosljeđuje Python skripti koja na temelju tih opažanja uči svoje modele i šalje koju akciju treba poduzeti. Agent prima tu informaciju i odradi ju.

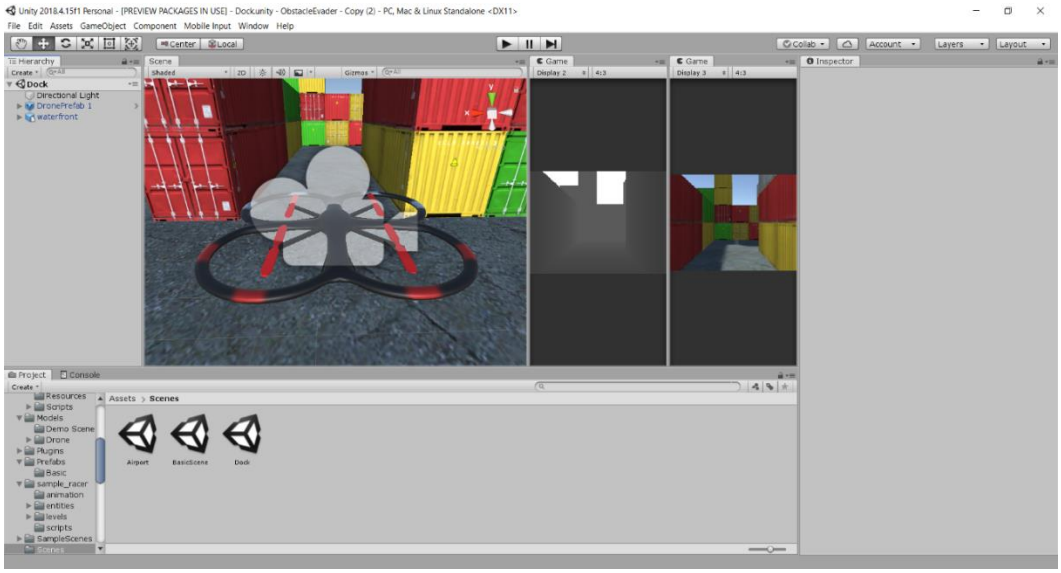
Modeli se mogu na kraju učenja izvesti u Unity. Kad se postavka Behavior type postavi na Inference only tada agent šalje prikupljene podatke na ulaz modela i koristi izlaz modela kao akciju koju treba poduzeti.

2.1. Okruženje

U okviru ovog rada su u programu Unity napravljene tri scene. Te scene su: pristanište, aerodrom i testna I. Zamišljeno je da se na tim scenama nalaze prepreke pokraj kojih bi dron trebao prolaziti. Scene pristanište i aerodrom bi trebale imati realne teksture dok je testna

scena samo skup jednoličnih objekata koji služe za testiranje nepoznatog okruženja. Scena pristanište služi kao scena s puno svjetlosti i šarenih objekata. Scena aerodrom je noćna scena s malo svjetla i par objekata.

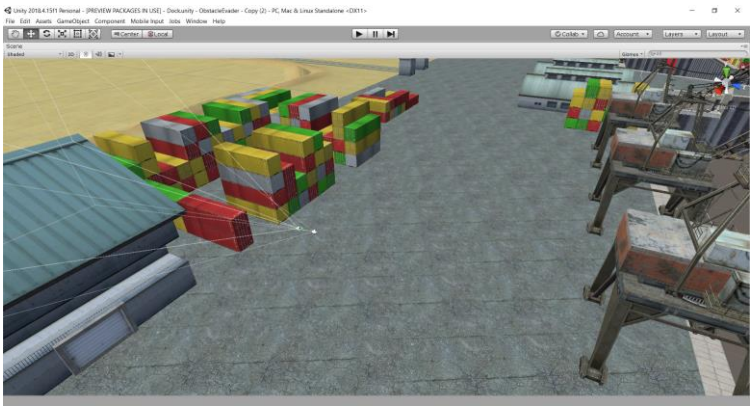
U svakoj od tih scena se nalazi dron kojim bi trebalo upravljati i dovesti ga na odredište. Dron možemo vidjeti na slici 2.4.



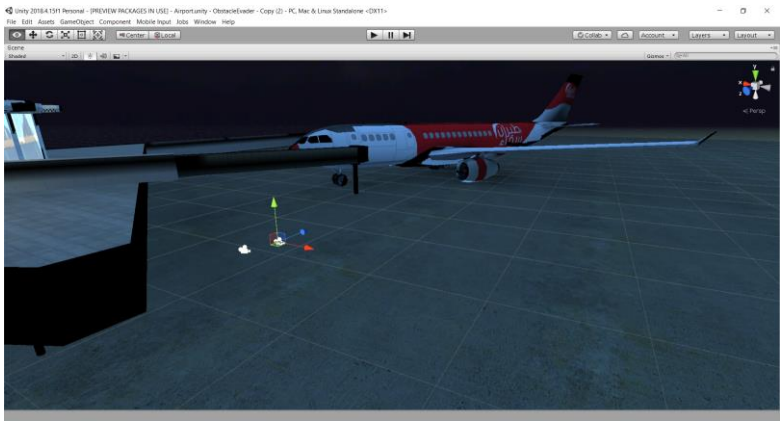
Slika 2.4 Dron

Vrste ciljeva se dijele na jednostavne i teške. Za dolazak do jednostavnih dovoljno je samo okrenuti se prema cilju i nastaviti ravno. Za dolazak do teških ciljeva potrebno je zaobići neke prepreke.

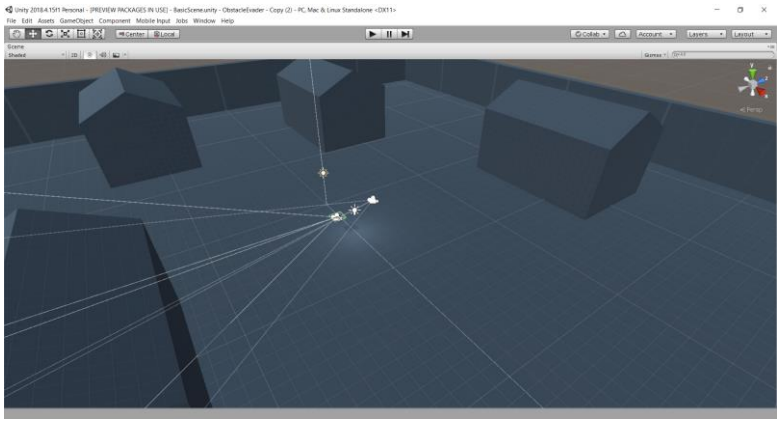
Okruženje aerodrom se sastoji od 7 jednostavnih ciljeva, ali jako udaljenih od početne pozicije. Na pristaništu ima 6 jednostavnih ciljeva i 4 teška cilja. Testno okruženje ima 10 jednostavnih i 2 teška cilja. Prikazi ove tri scene su na slikama 2.5, 2.6 i 2.7. Usporedba okruženja je dana u tablici 2.1.



Slika 2.5 Pristanište



Slika 2.6 Aerodrom



Slika 2.7 Testna scena

Ime scene	Teksture	Svjetlina	Broj jednostavnih ciljeva	Broj teških ciljeva	Svrha
Pristanište	Realne	Svijetlo	6	4	Učenje teških ciljeva
Aerodrom	Realne	Mračno	7	0	Učenje dalekih ciljeva
Test	Jednolične	Svijetlo	10	2	Nepoznato okruženje

Tablica 2.1 Usporedba scena za učenje

2.2. Model drona

Ponašanje drona je određeno akcijama. Više akcija čine jednu epizodu. Epizoda završava kad dron ili dođe do cilja ili pogodi neku od prepreka. Kako ne bi trajala beskonačno, epizoda automatski završava nakon 400 akcija. Akcije su rotacije lijevo i desno u y osi i

kretanje prema naprijed. Dron se može gibati u jednoj ravnini i nema akcija za mijenjanje visine drona. Jedna akcija označava jedan korak simulacije i ona se zahtjeva od drona svakih pet sličica (bez ubrzanja simulacija se izvodi u 60 sličica po sekundi). Dron na svojoj prednjoj strani ima jednu kameru čiji se izlazi prosljeđuju modelu za odabir akcije. Kamera proizvodi dva izlaza. Jedan izlaz je RGB slika, a drugi je slika čiji pikseli predstavljaju dubinu promatrane scene. Slike dubine su vizualizirane crno-bijelo, gdje bijelo označava nešto što je jako daleko, a crno jako blizu. Slika pokriva polje vidljivosti od 60 stupnjeva.

Senzori koje dron šalje algoritmu za upravljanje su: dvije slike dobivene s kamere i kut otklona cilja u stupnjevima u rasponu od -180 do +180.

Kako bi se moglo omogućiti potporno učenje, dron dobiva kazne i nagrade ovisno o svom ponašanju. Ako je dron uspješno došao do cilja, epizoda završava i agent dobiva nagradih 10 bodova. Ako je dron udario u neku od prepreka epizoda završava i agent dobiva kaznu od -5 bodova. Moguća funkcija nagrade je postepena nagrada za svaki odrađeni dio puta do cilja, na primjer nagradnih 1 bod za svaku desetinu puta. Također je moguće dodati jako malu kaznu za svaki korak agenta. Time se može postići da agent brže dolazi cilja.

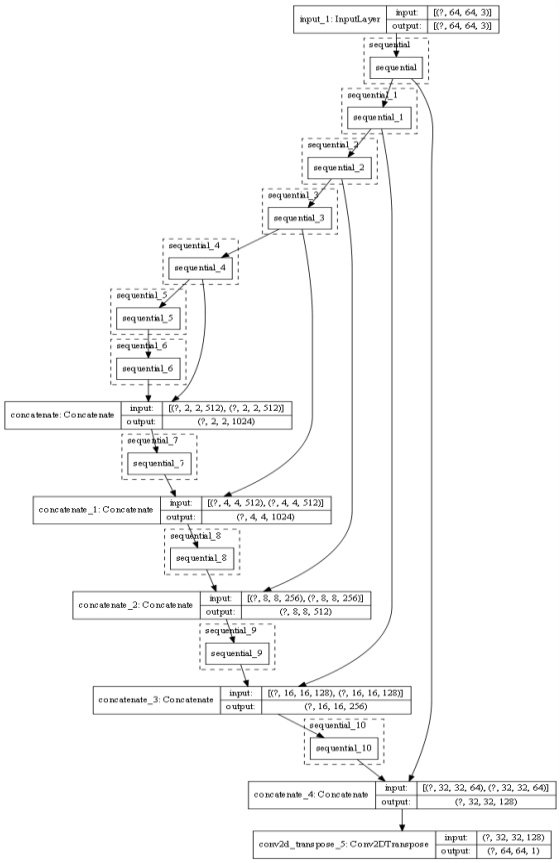
3. Procjenjivanje dubine scene iz slike

3.1. Model procjenjivanja dubine iz slike

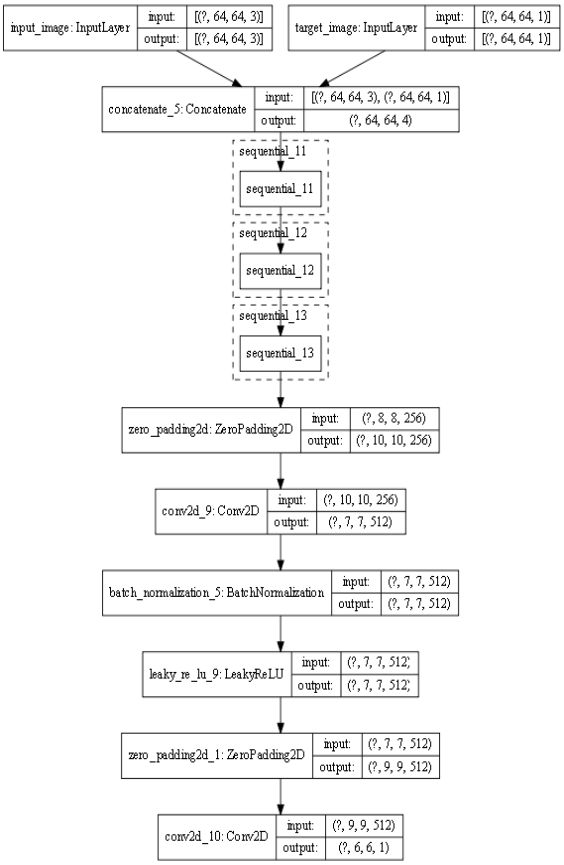
Model koji je korišten za dobivanje udaljenosti je uvjetovana suparnička mreža (*cGAN*), konkretno Pix2Pix prema radu [6]. Kod uvjetovanih suparničkih mreža obje komponente suparničke mreže na ulaz dobivaju nešto što će ih uvjetovati, na primjer, oznake ili ulazne slike. U ovom slučaju je na ulaze generatora i diskriminatora postavljena još i ulazna slika. Generator je U-Net mreža koja prima sliku dimenzije 64x64x3, a izlaz je slika dimenzije 64x64x1. U-Net struktura je specifična po svom U obliku gdje se jednak broj koraka provodi konvolucija i dekonvolucija [7]. Također se svaki izlaz nekog konvolucijskog sloja konkatenira s ulazom u simetrični dekonvolucijski sloj. Šum koji se klasično daje generatoru je dobiven nasumičnim gašenjem neurona (eng. *Dropout*) prilikom dekonvolucije. Generator je prikazan na slici 3.1.

Diskriminator je PatchGAN koji prima dvije slike dimenzija 64x64x3 i 64x64x1 a daje izlaz dimenzije 6x6x1. Za razliku od klasičnih klasifikatora koji na izlazu daju kategoriju ulaza, PatchGAN pokušava provjeriti kategoriju dijela slike. Tako se daje veća važnost lokalnim dijelovima slike nego kompletnoj slici. Prikaz diskriminatora je na slici 3.2.

Podaci za učenje su generirani kroz simulaciju. Ručnom vožnjom drona su prikupljeni podaci o RGB slici i odgovarajućoj dubini. Postoje dva skupa podataka. Jedan je sa slikama iz svih scena, dok je drugi bez testne scene. Za učenje modela je korišten algoritam ADAM.



Slika 3.1 Generator



Slika 3.2 Diskriminator

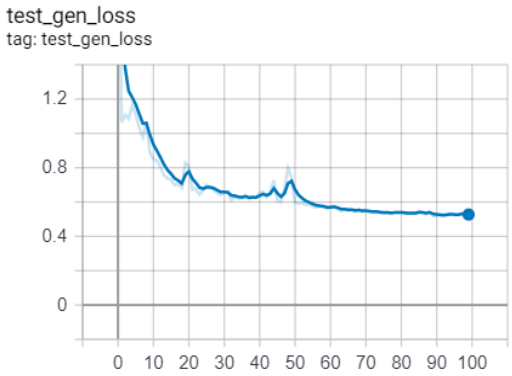
3.2. Rezultati

Kod učenja se podaci dijele na podatke za učenje i podatke za evaluaciju. Učenje se izvodi u 100 epoha. U svakoj epohi se model uči tako da podijeli podatke na manje grupe i na svakoj odradi jedan korak ADAM algoritma. Nakon što se sve grupe odrade, model se evaluira na podacima za evaluaciju. Funkcija gubitka prema kojoj se evaluira je binarna unakrsna entropija i ona je zadana izrazom:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\big(p(y_i)\big) + (1-y_i)\log\big(1-p(y_i)\big), \tag{16}$$

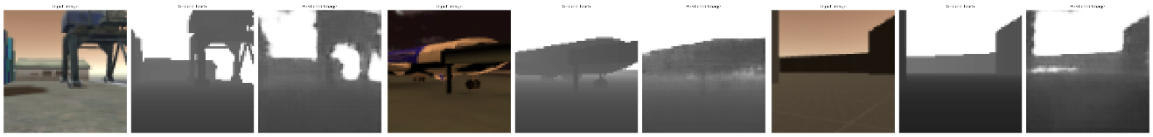
gdje je N broj točaka, a y_i je izlaz modela.

Rezultat evaluacije modela na skupu podataka za evaluaciju nakon svake epohe učenja je prikazana na slici 3.3.



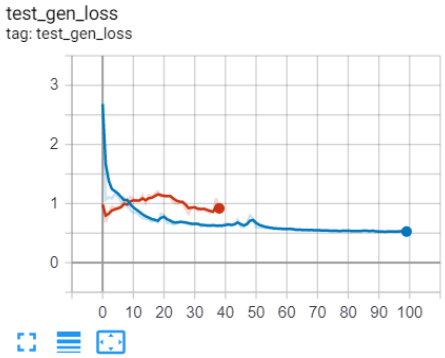
Slika 3.3 Funkcija gubitka kroz učenje na cijelom skupu podataka

Na slici 3.4 vidimo usporedbu RGB slike, prave dubine i dubine predviđene opisanim modelom. Slike su redom iz scena pristanište, aerodrom i testna scena. Vidi se da je model dobro predvidio dubinu. Dobro se vidi relativna udaljenost. Pod, koji je blizu, je tamnije obojen od objekata koji su dalje.



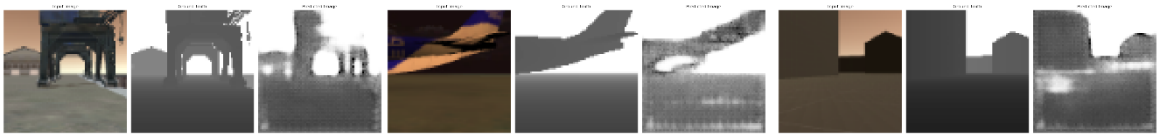
Slika 3.4 Usporedba slike u boji, prave dubine i predviđene dubine

Prilikom učenja na skupovima pristanište i aerodrom i evaluaciji na testnoj sceni, nakon 40 epoha dobiveni su rezultati prikazani na slici 3.5. Plavom bojom je obojen rezultat evaluacije na testnom skupu tijekom učenja sa svim podacima. Narančastom bojom je obojen rezultat evaluacije modela tijekom učenja na skupu bez testne scene. Odabrano je 40 epoha kako se model ne bi prenaučio na skupu za učenje. Nakon te točke bi evaluacijom na testnoj sceni model davao lošije rezultate.



Slika 3.5 Usporedba funkcija gubitka modela za predviđanje dubine

Na slici 3.6 vidimo kako je puno zrnatiji izlaz modela, ali još uvijek daje korisnu informaciju o dubini. Zanimljivo je kako uspijeva predvidjeti dubinu i u nepoznatom okruženju.



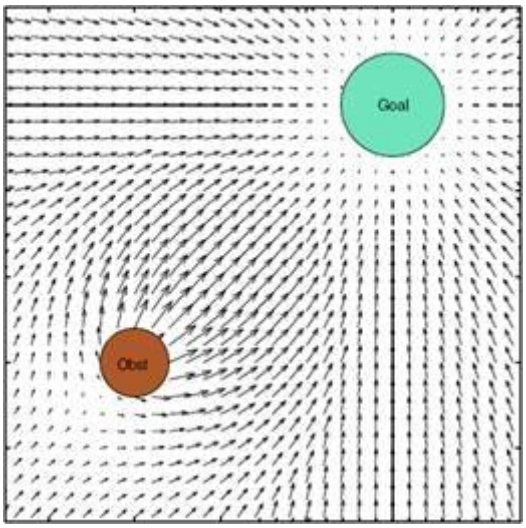
Slika 3.6 Usporedba slike u boji, prave dubine i predviđene dubine za manji skup za učenje

4. Izbjegavanje prepreka

Za usporedbu postupaka izbjegavanja prepreka napisano je više politika odabira akcija. Nasumična politika odabire akcije iz uniformne razdiobe. Pohlepna politika u kojoj dron samo prati kut prema cilju. Metoda polja potencijala [8] u kojoj se uzima u obzir udaljenosti od prepreka i generiraju sile koje odguruju drona, dok ga cilj privlači. Također su implementirane dvije verzije dubokog potpornog učenja, jedna klasična i jedna s povratnom neuronskom mrežom, prema radu [9].

4.1. Metoda polja potencijala

U ovoj metodi se koriste predviđene udaljenosti u svrhu izbjegavanja prepreka. Princip algoritma je temeljen na potencijalima polja i da svaka prepreka pruža silu od prepreke prema vozilu, proporcionalno blizini. Cilj također djeluje na vozilo, ali iz smjera vozila prema cilju. Rezultat sume tih sila određuje smjer u kojem se vozilo treba gibati. Vizualizaciju odbojnih i privlačnih sila možemo vidjeti na slici 4.1, gdje je tirkiznom bojom obojen cilj, a smeđom prepreka.



Slika 4.1 Metoda polja potencijala [8]

Iz RGB slike dimenzije 84x84 piksela, generira se mapa dubina. Iz te slike dubina se uzima 15 redaka piksela koji se nalaze otprilike na istoj visini kao dron. Ti podaci se koriste za izračunavanje sila i konačnog smjera kretanja. Ako je kut od željenog smjera veći od 10 ili manji od -10 stupnjeva onda dron skreće u prema željenom smjeru, a inače ide ravno. Ovaj

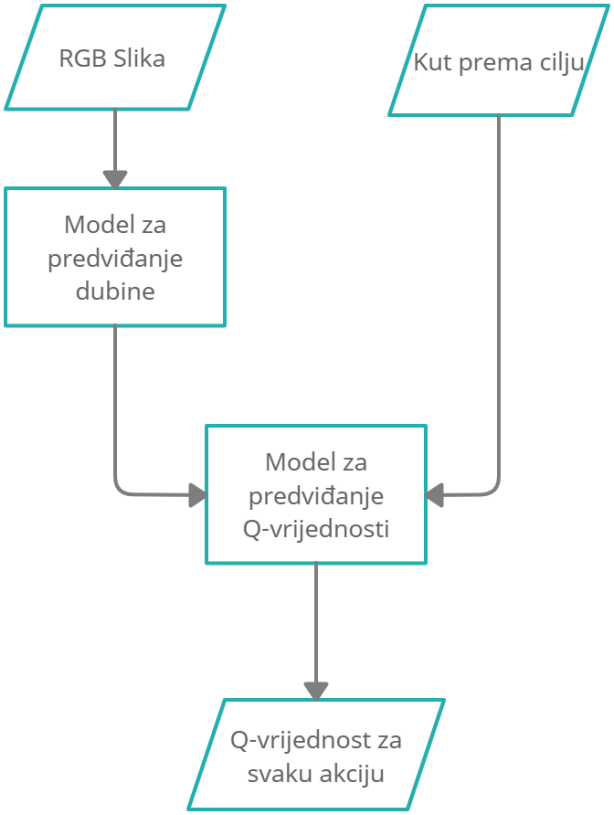
algoritam je opisan u pseudokodu 4.1. Parametri α i β su postavljeni na vrijednosti 84 i 1, tim redoslijedom.

```
Algoritam polja potencijala
α, β
dok epizoda nije gotova
  kut_prema_cilju <- dohvati iz simulacije
  traka_udaljenosti <- dohvati iz simulacije
  sila <- vektor(α, kut_prema_cilju)
  za svaki stupac_udaljenosti, kut iz traka_udaljenosti
    sila <- sila – β*vektor(prosjek(udaljenosti), kut)
  kraj za svaki petlje
  kut_sile <- kut(sila)
  ako je kut_sile < -10:
    skreni desno
  inače ako je kut_sile > 10:
    skreni lijevo
  inače
    nastavi ravno
kraj dok je petlje
```

Pseudokod 4.1 Algoritam polja potencijala

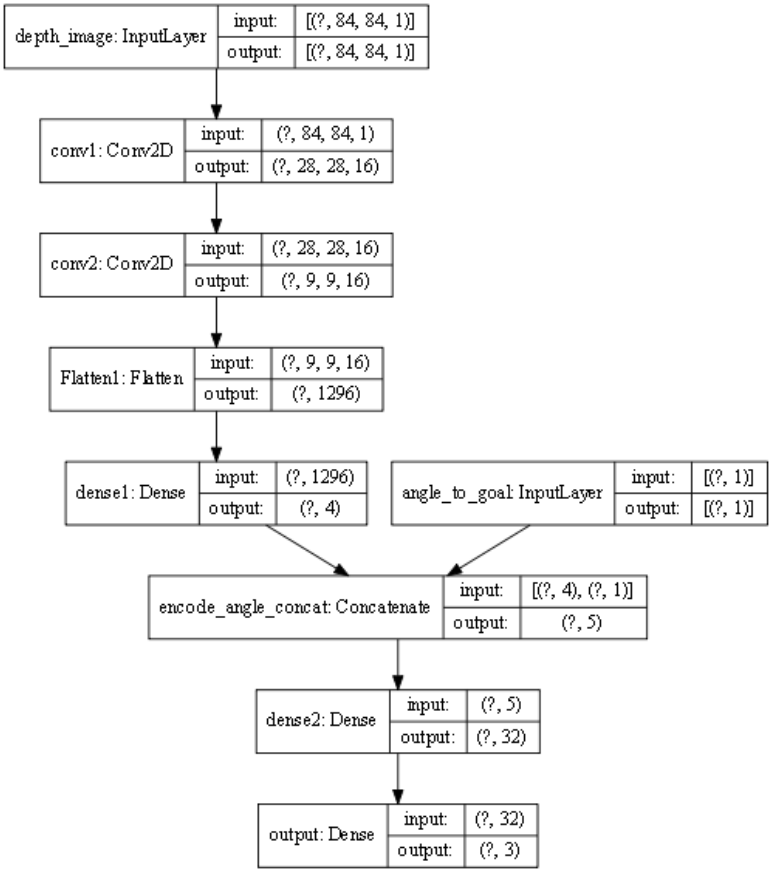
4.2. Duboko konvolucijsko potporno učenje

Model za upravljanje dronom se sastoji od više dijelova. Prvi dio je model za predviđanje dubine. Korišten je model predstavljen u poglavlju 3. Tom modelu se daje na ulaz RGB slika s drona. Predviđene dubine se zajedno s kutom prema cilju daju na ulaz modelu za duboko potporno učenje. To možemo vidjeti na slici 4.2.



Slika 4.2 Model predviđanja Q-vrijednosti

Mreža za učenje Q-vrijednosti se sastoji od više slojeva. Predane dubine se prvo sažimaju s nekoliko konvolucijskih slojeva. Svaki konvolucijski sloj se sastoji od 16 filtara, veličine 3 i pomaka 3. Zatim se poravnava izlaz iz zadnjeg konvolucijskog sloja i dodatno sažima jednim slojem potpuno povezanih neurona. Na izlaz tog sloja se dodaje kut prema cilju. Te vrijednosti se šalju na ulaz potpuno povezanog sloja veličine 32 i aktivacijske funkcije tangens hiperbolni. Zadnji sloj je potpuno povezani sloj veličine 3 za svaku od akcija. Aktivacijska funkcija zadnjeg sloja je linearna funkcija. Grafički prikaz ovog modela je dan na slici 4.3.

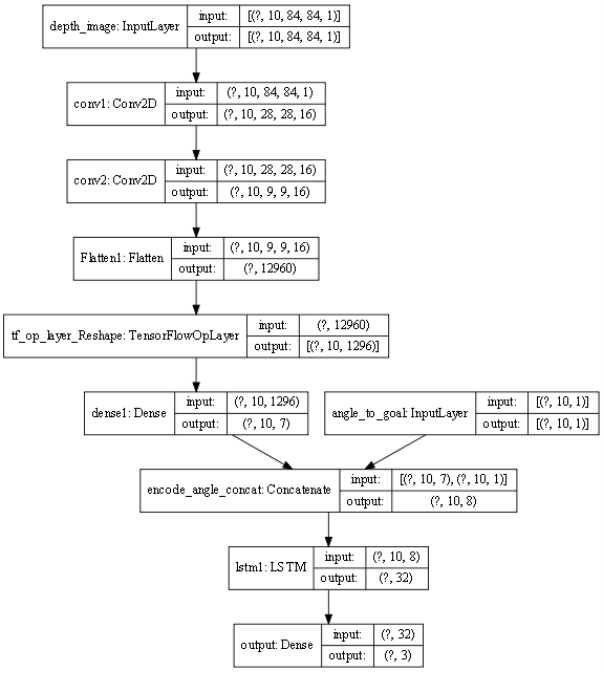


Slika 4.3 Model duboke Q-mreže

Kod prvog pokretanja koristi se nasumičan odabir akcije i skuplja se 10000 koraka kroz više epizoda simulacije. Tako bi agent dobio veći početni skup za učenje i istražio što veći prostor stanja. Pamtila bi se nagrada za zadani ulaz i promatranja sa senzora. Na nagradu je primijenjen faktor propadanja od 0.96. Kad bi se skupile epizode za učenje, one bi se podijelile u 20 manjih grupa radi ograničene radne memorije na grafičkoj kartici. Učenje na manjim grupama se ponavljalo tri puta. Algoritam za učenje je ADAM. Nakon prvog učenja, agent bi poduzimao akcije sljedećih 4000 koraka i ponovio bi se isti proces učenja. Akcije koje je agent radio su u 20% slučajeva bile nasumične u svrhu eksploracije. U 30% slučajeva su bile uzorkovane tako da je Q-vrijednost za svaku akciju bila umanjena za najmanju i pridodano je 1 svakoj Q-vrijednosti kako bi i najlošija akcija imala nekakvu vjerojatnost. Te vrijednosti su se skalirale da suma daje 1. Zatim bi se koristila ta vrijednost kao vjerojatnost odabira te akcije. U ostalih 50% slučajeva koristila se akcija s maksimalnom Q-vrijednosti.

4.3. Duboko povratno potporno učenje

Za razliku od klasičnog dubokog potpornog učenja, u ovom slučaju se pamti zadnjih 10 koraka simulacije. Umjesto predzadnjeg sloja potpuno povezanih neurona nalazi se LSTM ćelija. Ovaj model je dodan kako bi se provjerilo koliko prethodne akcije pomažu u odabiru nove. Motivacija je pamćenje trenutnog manevra zaobilaženja prepreke. Izgled ovog modela se vidi na slici 4.4.

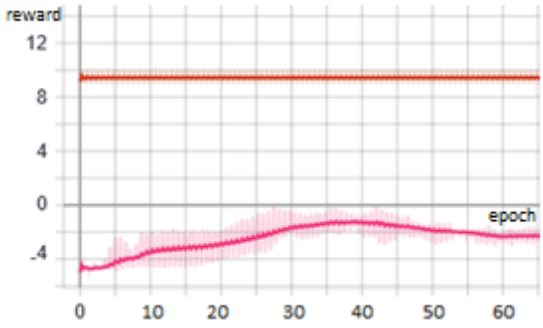


Slika 4.4 Model duboke povratne Q mreže

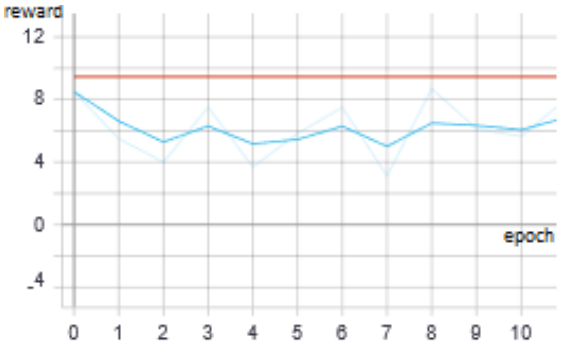
5. Rezultati

Modeli izbjegavanja prepreka su učeni više epoha po zadani broj epizoda. Svakih 5 epoha se spremaju trenutne težine modela. Tijekom učenja, funkcija gubitka oscilira zbog različitih stanja i nagrada, ali nakon što funkcija nagrade miruje onda i funkcija gubitka krene padati. Na grafovima nagrade je na y osi prikazana prosječna nagrada po epizodama, dok je na x osi indeks trenutne epohe. Prosječna nagrada 10 bi značila da je dron svaki put stigao u cilj, dok bi prosječna nagrada -5 značila da je dron svaki put udario u prepreku. Vrijednosti između mogu značiti ili da je dron nekad uspio doći do cilja ili da je simulacija prekinuta i da je dron završio epizodu s nagradom 0.

Kako bi se ubrzalo učenje, dodano je predtreniranje. Skupljen je skup podataka u kojem je ručno vožen dron, zatim je model učen na tom skupu. U tim podacima je svaki put dron uspješno došao do cilja, izbjegavajući i teške i lagane prepreke. To je dosta ubrzalo učenje kao što se vidi kad usporedimo slike 5.1 i 5.2. U ovim grafovima smeđa ravna crta predstavlja početni skup podataka, odnosno ponašanje drona kad ga čovjek kontrolira. Vidimo kako je agent s predtreniranjem odmah skočio na visoke prosječne nagrade, dok je obični polagano rastao.

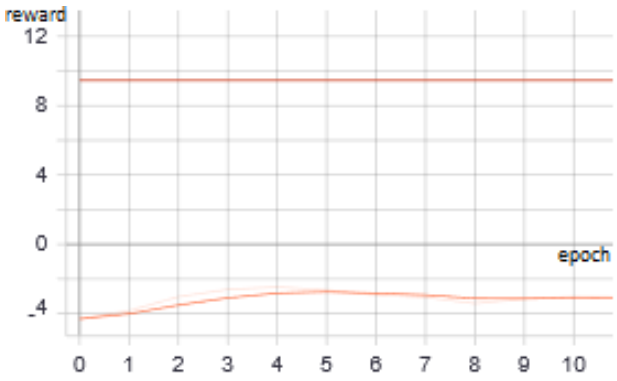


Slika 5.1 Prosječna nagrada tijekom učenja duboke Q-mreže

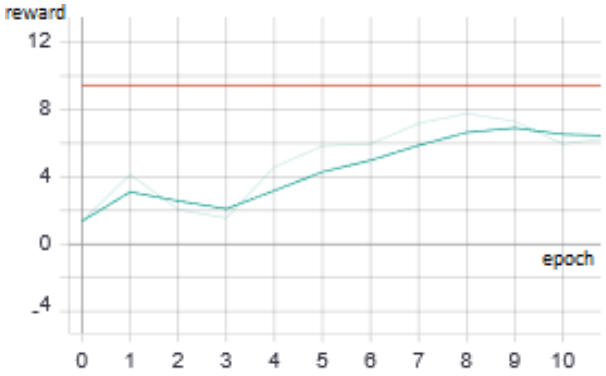


Slika 5.2 Prosječna nagrada tijekom učenja duboke Q-mreže s predtreniranjem

Učenje duboke povratne potporne mreže je trajalo malo duže radi veće složenosti modela. Predtreniranje je također ubrzalo povećanje prosječne nagrade.

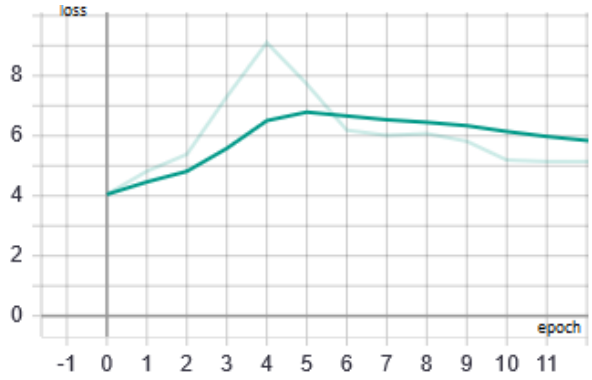


Slika 5.3 Prosječna nagrada tijekom učenja duboke povratne Q-mreže

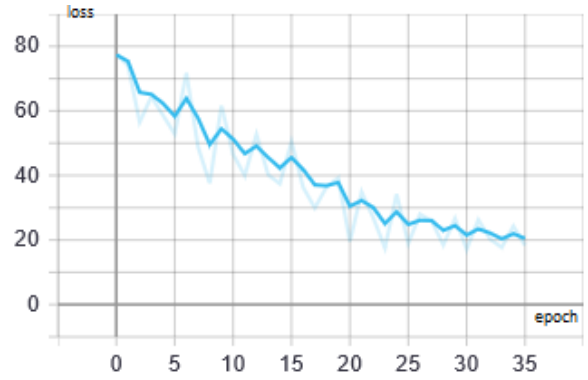


Slika 5.4 Prosječna nagrada tijekom učenja duboke povratne Q-mreže s predtreniranjem

Na slici 5.6 je funkcija gubitka kod učenja duboke Q-mreže. Na y osi je vrijednost funkcije gubitka koja je srednje kvadratno odstupanje. Na x osi je indeks trenutne epohe. Na tom grafu vidimo da vrijednost funkcije gubitka ne konvergira. Pokazalo se teškim naučiti Q-vrijednosti jer se svaki put uči s drugim vrijednostima. Također nema previše veze trenutna slika i kut s veličinom nagrade jer zbog propagacije nagrade za istu poziciju i isti smjer prema cilju za istu akciju može biti dana različita nagrada ovisno o udaljenosti od cilja. Što je agent bliže cilju to je jači utjecaj nagrade za dolaska do cilja. Funkcija gubitka bi padala jedino kad bi nagrade iz trenutne epohe bile slične s prethodnom. To se vidi na slici 5.5.



Slika 5.5 Funkcija gubitka učenja duboke povratne Q-mreže



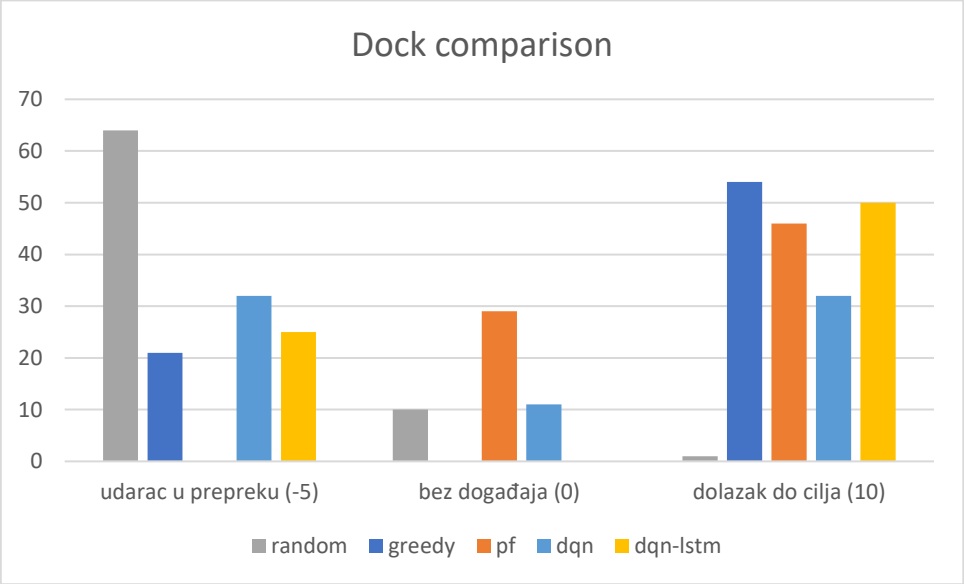
Slika 5.6 Funkcija gubitka kod učenja duboke Q-mreže

Sve politike su pokrenute 75 puta na svakoj sceni. Generirani su histogrami kako bi se pokazala distribucija nagrada koje svaka politika dobiva na svakoj sceni. Svaka epizoda može završiti nagradom -5 za udarac u prepreku, 10 za dolazak do cilja i 0 za prekinutu simulaciji u slučaju prevelikog broja koraka. Stupci histograma predstavljaju broj epizoda koje su završile s nagradom ispod tog stupca. Rezultati na sceni pristanište vide se na slici 5.7. Vidimo da je nasumični agent (*random*) bio jako loš i nije skoro nikad došao do cilja. Pohlepni algoritam (*greedy*) je često udarao u prepreke (28%). Metoda polja potencijala (*pf*) je postigla bolju prosječnu nagradu. To se vidi iz toga da dron nije udarao u prepreke već bi stao ispred prepreke i ne bi nastavljao dalje pa bi simulacija bila prekinuta i bila bi dodijeljena nagrada 0. Politike povratnog dubokog Q-učenja (*dqn-lstm*) su usporedive s pohlepnim algoritmom, dok je klasični DQN model (*dqn*) češće završavao bez da udari ili dođe do cilja.

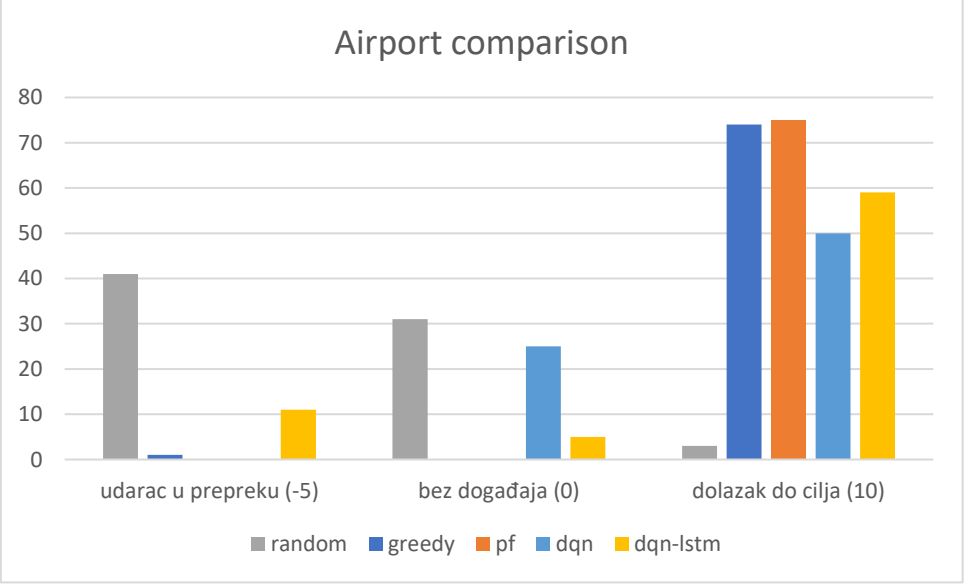
Zbog toga što su ciljevi na sceni aerodrom (slika 5.8) udaljeniji nasumični agent je manje udarao u prepreke, ali bi češće završavala epizoda bez događaja. I pohlepni i algoritam polja potencijala su postigli savršene rezultate na ovoj sceni, tj. svaki put bi došli do cilja bez da

udare u neku prepreku. To je i očekivano jer se sastoji samo od jednostavnih ciljeva. Modeli dubokog učenja su bili lošiji. Nisu često udarali u prepreku, ali ne bi stigli doći do cilja jer imaju dosta nepotrebnih akcija.

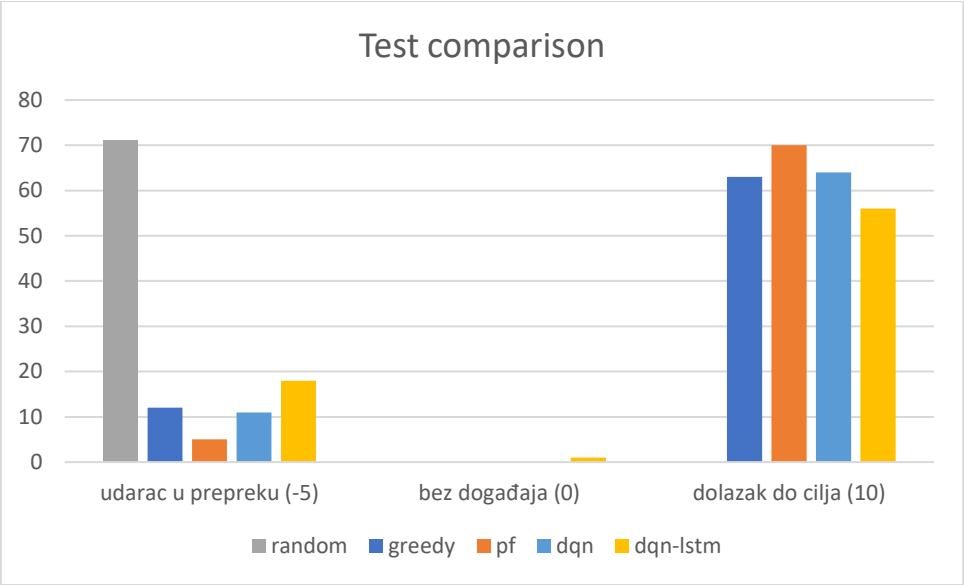
Na testnoj sceni (slika 5.9) nasumični agent je bio jako loš. Svi ostali agenti su usporedivi. Jedino bih istaknuo da algoritam polja potencijala ne bi dolazio do samo jednog težeg cilja dok ostali ne bi došli do dva.



Slika 5.7 Histogram nagrada na pristaništu

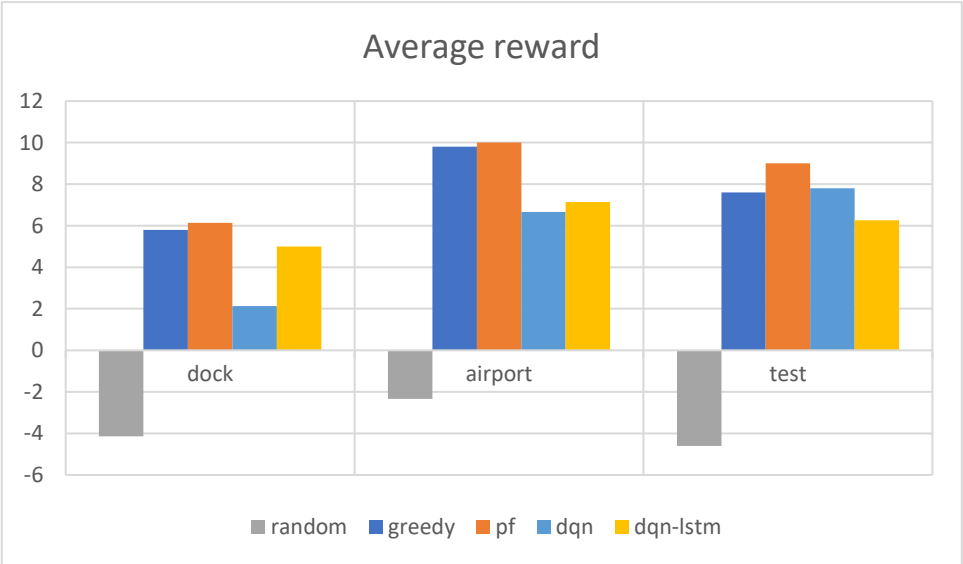


Slika 5.8 Histogram nagrada na aerodromu



Slika 5.9 Histogram nagrada na testnoj sceni

Kad usporedimo prosječnu nagradu (slika 5.10) za svaki algoritam na svakoj sceni vidimo da je kombinacija predviđanja udaljenosti i algoritma polja potencijala bila najbolja na svim scenama. Pohlepni algoritam je bio bolji od modela dubokog učenja samo na sceni aerodrom dok su na ostalima postizali slične rezultate.



Slika 5.10 Prosječna nagrada

Zaključak

Izbjegavanje prepreka je složen problem kojeg nije lako riješiti jednostavnim pristupom. Potrebno je puno podešavanja hiperparametara kako bi klasični algoritmi radili ili jako puno sati učenja složenih modela kako bi postigli zadovoljavajuće rezultate. U ovom radu, najboljim se pokazala kombinacija strojnog učenja s metodom polja potencijala za izbjegavanje prepreka. Korištenjem Pix2Pix arhitekture dobio se jako dobar procjenitelj dubine. Ta komponenta bi mogla biti korisna i za neke druge pristupe i složenije algoritme. Ovi rezultati su pokazali kako bi jedna kamera trebala biti dovoljna za procjenu informacije o udaljenosti objekata u različitim okruženjima. Dodatnim proučavanjem algoritama za izbjegavanje prepreka i različitih algoritama potpornog učenja, na primjer optimizacija bliskih politika (PPO) ili meki djelatnik-kritičar (*eng. soft actor-critic – SAC*), možda bi se moglo moći postići bolje rješenje.

Literatura

[1] Nicholas Hyldmar, Yijun He, Amanda Prorok. A Fleet of Miniature Cars for Experiments in Cooperative Driving. submitted to arXiv, 2019

[2] Lambert, Nathan. Fundamental Iterative Methods of Reinforcement Learning, https://towardsdatascience.com/fundamental-iterative-methods-of-reinforcement-learning-df8ff078652a

[3] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. ICLR 2015

[4] Choudhary, Ankit. A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python. https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/ [Accessed 23 January 2020]

[5] Gharakhanian, Al. GANs: One of the Hottest Topics in Machine Learning, https://www.linkedin.com/pulse/gans-one-hottest-topics-machine-learning-al-gharakhanian/?trk=pulse_spock-articles [Accessed 23 January 2020]

[6] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," 21 November 2016. [Online]. Available: https://arxiv.org/abs/1611.07004. [Accessed 23 January 2020]

[7] Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://arxiv.org/abs/1505.04597 [Accessed 23 January 2020]

[8] Safadi, Hani. Local Path Planning Using Virtual Potential Field. 18 April 2007

[9] Abhik Singla*, Sindhu Padakandla and Shalabh Bhatnagar (2018), Memory-based Deep Reinforcement Learning for Obstacle Avoidance in UAV with Limited Environment Knowledge

Sažetak

Naslov: Izbjegavanje prepreka dubokim potpornim učenjem.

Postavljena je simulacija u programskom sustavu Unity. Uporabljene su tri različite scene za učenje. U sceni se nalazi dron koji se može gibati u dvije dimenzije, s fiksiranom visinom. Naučen je model arhitekture Pix2Pix za mono-okularnu predikciju dubine.. Korišteni su podaci generirani u okviru simulacije. Algoritam izbjegavanja prepreka je Q-učenje. Simulacija daje algoritmu RGB sliku prostora ispred drona. Predviđa se dubina i zajedno sa željenim smjerom daje se kao ulaz u duboku Q-mrežu (DQN). Model se sastoji od potpuno povezanih i konvolucijskih slojeva. Ostvarena je i varijanta s LSTM slojem. Radi usporedbe su korištene i neke heuristike za izbjegavanje prepreka. Najbolji rezultat je postignut kombinacijom predviđene dubine i varijantom algoritma polja potencijala.

Ključne riječi: GAN, Pix2Pix, Računalni vid, Duboko učenje, Monokularno predviđanje dubine, Unity, Potporno učenje, Izbjegavanje prepreka, Duboko potporno učenje, Q-učenje

Abstract

Title: Deep reinforcement learning based obstacle evasion

A simulation is built using Unity. There are three different testing scenes. A drone is placed in the scene. It can only move in two dimensions. The height is fixed. Pix2Pix architecture is used for single camera depth prediction. Training dataset is generated from the simulation. Algorithm used for obstacle evasion is Q-learning. Simulation provides algorithm with RGB image of the area in front of the drone. Predicted depth with target angle is passed as input to a Deep Q-Network (DQN). Model is composed of dense and convolution layers. There is also a DQN version which uses a LSTM layer. Several obstacle avoidance heuristics are developed to better compare results of the DQN. The best result is achieved by using predicted depth with potential field algorithm.

Key words: GAN, Pix2Pix, Computer Vision, Deep learning, Monocular depth prediction, Unity, Reinforcement learning, Obstacle evasion, Deep reinforcement learning, Q-learning