SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

DDPG agent u HalfCheetah okruženju

Seminarski rad

Raspoznavanje uzoraka i strojno učenje

Denis Lazor

Osijek, 2020

SADRŽAJ

1 UVOD

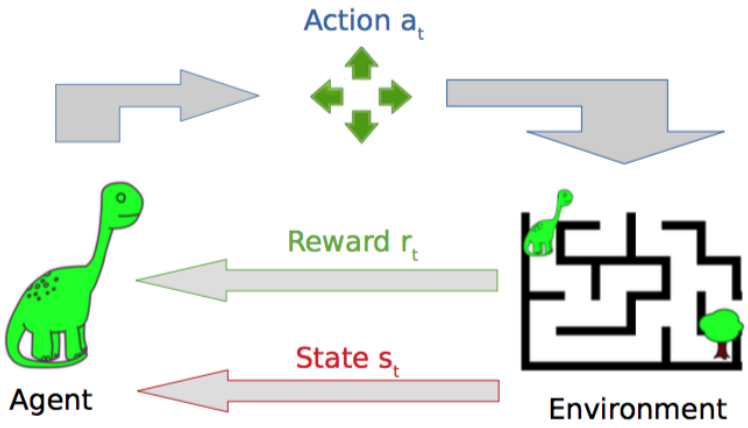
Prošlih godina ostvario se značajan napredak u sposobnostima umjetne inteligencije (*eng. AI*) i metoda podržanog učenja. Moderni agenti temeljeni na algoritmima podržanog učenja mogu riješiti probleme za kojima bi čovjeku trebale godine u samo nekoliko sati. Imaju generalnu primjenu, to jest mogu rješavati veliki raspon zadataka sa minimalnim preinakama..

Glavna ideja ovoga rada je naučiti agenta da mijenja u vremenu vrijednosti momenta sila motora zglobova robotskog geparda kako bi se on što prirodnije i brže kretao prema naprijed. Za tu svrhu koristit će se *Deep Determenistic Policy Gradient* algoritam. Učenje će biti provedeno na 5 različitih agenata koristeći isti algoritam.

2 PODRŽANO UČENJE

Kod podržanog učenja ne zna se unaprijed točan izlazan podatak *Y* za neki ulazni podatak *X*. Izlazni podatci i vrijednosti kriterijske funkcije se spremaju tijekom rada agenta. Tek kada agent dobije nagradu može znati jesu li njegove odluke dobre.

Kvaliteta pojedinih odluka u podržanom učenju se predstavlja nagradom. Akcije koje donose veću nagradu će imati veću mogućnost odabira u budućim situacijama.



Slika 1 Podržano učenje

Na slici 1 je prikazan opći postupak podržanog učenja. Radi se o petlji u kojoj agent poduzima neke akcije i time djeluje na okolinu i dovodi je u novo stanje. Kao odgovor od okoline dobiva informaciju o novom stanju okoline i nagradi. Pomoću tih informacija agent može odrediti koje su akcije bolje ili lošije i s obzirom na to donositi bolje odluke u budućnosti.

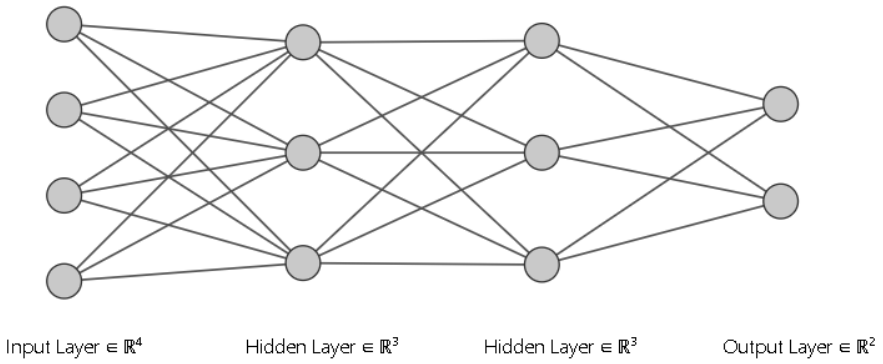
Cilj podržanog učenja je podesiti agenta tako da on poduzima akcije koje će mu vratiti maksimalnu vrijednost kumulativne nagrade za određen broj koraka u budućnost. Način kojim se to postiže je ovisan o algoritmu koji se koristi.

Agenta predstavlja model koji pokušavamo optimizirati. U ovom radu agenta će predstavljati dvije neuronske mreže čije težine želimo optimizirati minimizirajući kriterijsku funkciju. Više o neuronskim mreža će biti riječi u 2.1. poglavlju.

Postoje mnogi algoritmi podržanog učenja ovisno o primjeni. U ovom radu će biti korišten DDPG algoritam o kojem će biti više riječi u 3. poglavlju.

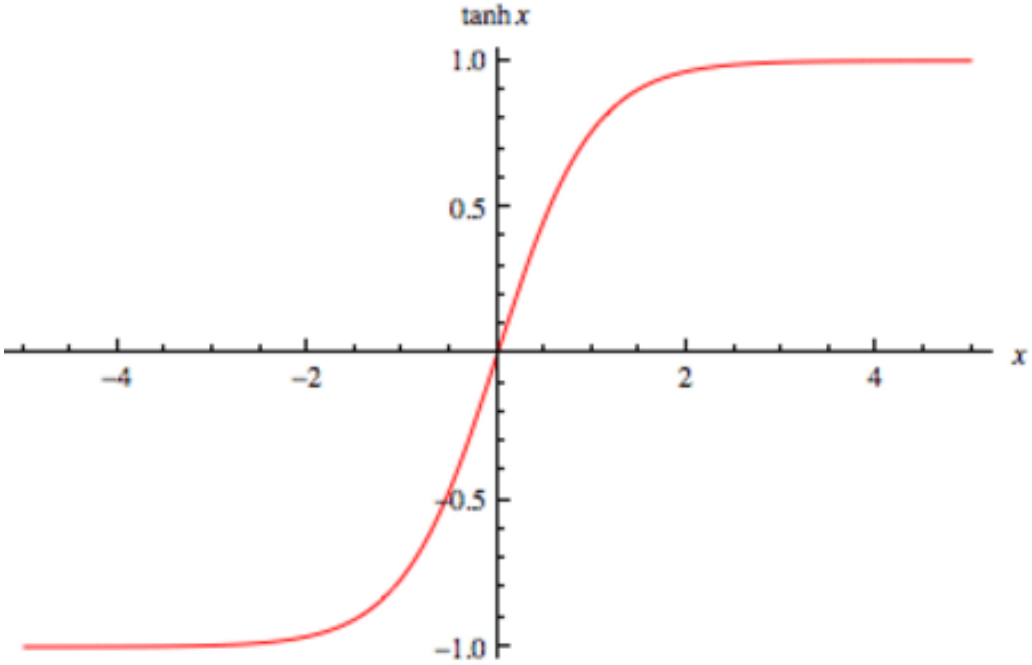
2.1 Neuronske mreže

Neuronske mreže, ili umjetne neuronske mreže, algoritmi su strojnog učenja modelirani po uzoru na biološke mreže neurona unutar ljudskog mozga. Današnje neuronske mreže su većinom sve po definiciji duboke neuronske mreže, ili neuronske mreže s dva ili više skrivenih slojeva. Skriveni slojevi (*eng. hidden layer*) su svi slojevi neuronske mreže između ulaznog (*eng. input layer*) i izlaznog sloja (*eng. output layer*). Shema jednostavne potpuno povezane neuronske mreže nalazi se na slici 1.



Slika 2 Shema jednostavne potpuno povezane neuronske mreže

Struktura neuronske mreže s slike 2 je (4 – 3 – 3 – 2), gdje su prvi i četvrti slojevi ulazni i izlazni slojevi, a drugi i treći slojevi su skriveni slojevi. Slojevi se sastoje od neurona (krugovi na slici 1), a svaki neuron je nositelj jednog realnog broja. Ova neuronske mreža je potpuno povezana jer je svaki neuron iz sloja *k* povezan s svakim neuronom iz slojeva *k-1* i *k+1*. Za slojeve neurona mogu se vezati funkcije poput aktivacijske funkcije, funkcije za opadanje neurona (*eng. dropout*), *Softmax* funkcija te mnoge druge. Aktivacijska funkcija se primjenjuje na sve neurone sloja te služi za uvođenje nelinearnosti u neuronsku mrežu, jer uvođenjem linearnosti neuronska mreža može estimirati nelinearne funkcije. Dodatno, kodomene aktivacijske funkcije se često prostiru od 0 do 1, pa služe za ograničavanje vrijednosti neurona. Često korištena aktivacijska funkcija je Relu ili Rectified linear unit kojoj je kodomena od 0 do $+\infty$.



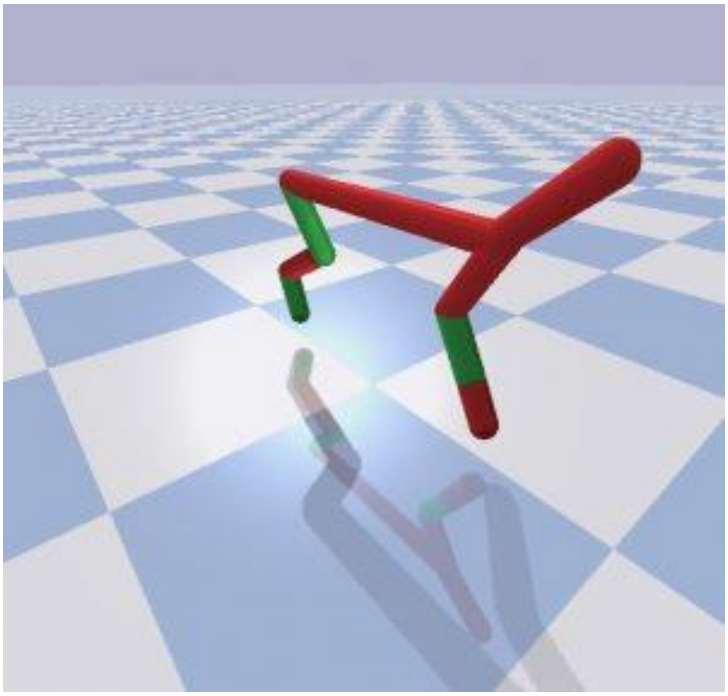
Slika 3 Tamh aktivacijska funkcija

Na slici 3 je prikazana aktivacijska funkcija Tanh koja će biti korištena u ovom radu uz Relu i Linear aktivacijsku funkciju. Ova aktivacijska funkcija nam je posebno važna jer vrijednosti neurona skalira između vrijednosti -1 i 1 jer akcije koje mreža treba dati kao rezultat moraju biti u tom istom intervalu.

3 RJEŠENJE PROBLEMA

3.1. HalfCheetah okruženje

OpenAI Gym je alat koji sadrži gotova okruženja za agente podržanog učenja. Za potrebe ovog rada preuzeto je HalfCheetah kontinuirano okruženje. HalfCheetah okruženje sastoji se od ravne podloge i dvonogog robotsko geparda prikazanog na slici 4.



Slika 4 – HalfCheetah okruženje

Svaka noga geparda sadrži po 3 zgloba što je ukupno 6, tako da je moguće izvršiti 6 akcija nad ovom okolinom. Te akcije predstavljaju momente sila motora tih 6 zglobova.

Samo okruženje kao odgovor na akciju daje 26 veličina koje predstavljaju zapažanja okoline. Ta zapažanja predstavljaju 6 kuteva zglobova, 6 kutnih brzina zglobova, položaj i brzina članaka te kontakti zglobova i članaka sa podlogom. Pri kontaktu zglobova sa podlogom agentu se pridodaje negativna nagrada kako ne bi naučio trčati na koljenima ili drugim zglobovima već samo na vrhovima 2 članaka koji predstavljaju stopala.

3.2. Biblioteke

Korištene su sljedeće biblioteka:

Tensorflow-rocm

- ➔ TensorFlow je biblioteka otvorenog koda za strojno i duboko učenje. Razvio ga je Google Brain tim koji se bavi dubokim učenjem i umjetnom inteligencijom.

Keras

- ➔ Keras je Python biblioteka otvorenog koda koja u svojoj pozadini može koristiti TensorFlow, CNTK, Theano ili MXNet. U izradi našeg rješenja, koristit ćemo TensorFlow u pozadini.

Keras-rl

- ➔ Sadrži gotove funkcije i klase napisane u Keras-u za rad s podržanim učenjem

Pybullet

- ➔ Python biblioteka za simulaciju, robotiku i duboko učenje bazirana na *Bullet Physics SDK-u*

3.3. DDPG

Učenje agenta je izvedeno koristeći Deep Deterministic Policy Gradient algoritam. DDPG je namijenjen izrazito za rješavanje kontinuiranih problema. U algoritmu se koriste dvije neuronske mreže jer je on zapravo mješavina algoritama koji se svode na traženje tzv. Optimalne politike i algoritama koji se svode na traženje optimalne vrijednosti Q-funkcije.

DDPG spada u Actor-Critic metode. Što znači da jedna neuronska mreže predlaže akcije s obzirom na zapažanja okoline(Actor) dok druga mreža procjenjuje koliko su te akcije zapravo dobre ili loše(Critic). Actor kao ulaz uzima zapažanja iz okoline, a kao izlaz daje akcije. Ctiric uzima akcije i zapažanja i daje Q vrijednost koja predstavlja ukupnu nagradu za buduće akcije ako se iz trenutnog stanja poduzme određena akcija i do se nakon toga prati trenutna politika. Za učenje tih mreža koristi se 'minibatch', nasumično odabrani skupovi mogućih akcija i stanja iz buffer-a.

DDPG je off-policy algoritam je ne koristi istu politiku i za odabir akcija i za optimizaciju same politike

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Slika 5 DDPG algoritam

Slika 5. prikazuje DDPG algoritam.

Prvi korak je nasumično inicijalizirati težine Actor i Critic mreže te Target Actor i Target Critic mreže. Target mreže su zapravo kopije originalnih mreža koje predstavljaju ranije verzije mreža. One se ažuriraju samo u određenim koracima. Daju stabilnost u procesu učenja i daju mogućnost da se istražuje prostor akcija bez posljedica gubljenja dobrih postavki mreža. Zatim se inicijalizira spremnika koji će sadržavati određen broj stanja, akcija i mogućih prijelaza u nekom trenutku. Prijelaz predstavlja trenutno stanje, akciju i nagradu te sljedeće stanje ako je poduzeta neka akcija.

Slijedi petlja kroz željeni broj epizoda. Prvo se inicijalizira određeni šum kako bi se potaklo istraživanje prostora akcija te se dohvaća početni skup stanja okoline.

Svaka epizoda se sastoji od određenog broja koraka. Jedan korak predstavlja poduzimanje akcije koristeći trenutnu verziju agenta, promatranje nagrade i sljedećeg stanja. Zatim spremanje tog prijelaza u spremnik. Iz spremnika se zatim uzima predodređen broj prijelaza i računa se optimalna vrijednost kumulativne nagrade koristeći trenutnu nagradu i optimalnu verziju agenta (Target mreže). Taj se rezultat uspoređuje sa rezultatima trenutne verzije agenta i računa se srednja kvadratna greška za ažuriranje Critic mreže. Korištenjem unazadne propagacije na Critic mreži dobivaju se potrebne derivacije za izračun gradijenata potrebnih za optimiziranje Actor mreže koristeći neki od optimizatora. U ovom slučaju je to Adam optimizator. Te na kraju slijedi ažuriranje Target mreža s unaprijed određenom stopom ažuriranje. To se sve ponavlja određen broj koraka kroz sve epizode.

Dodatno se može ubaciti i prekidna rutina za prekid epizode ukoliko agent dođe u terminalno stanje. Prekidna rutina se češće koristi u diskretnim sustavima.

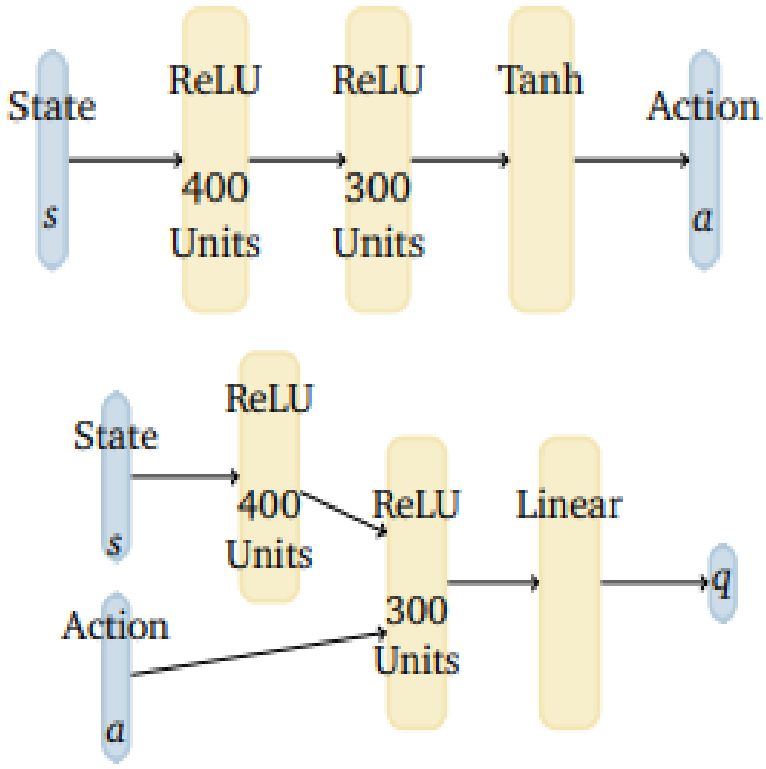
3.4. Struktura neuronske mreže

U svrhu testiranja utjecaja različitih kombinacija broja neurona na rezultat isprobano je 5 različitih kombinacija. Dalje u tekstu prikazana je originalna kombinacija. Ostale 4 kombinacije će imati istu strukturu, ali drugačiji broj neurona u skrivenim slojevima.

Actor mreža se sastoji od ulaza 26 ulaznih neurona pošto kao ulaz prima 26 opažanja iz okoline. Zatim slijede dva skrivena *Dense* sloja sa 400 i 300 neurona s *Relu* aktivacijskom funkcijom, te izlazni sloj sa 6 neurona pošto toliko ima mogućih akcija i *Tanh* aktivacijskom funkcijom.

Critic mreža ima također ulazni sloj od 26 neurona, zatim Dense sloj od 400 neurona i *Relu* aktivacijskom funkcijom nakon čega se slijedi sloj gdje se dodaju akcije kao ulazi i spajaju se u jedan tenzor sa opažanjima okoline. Zatim slijedi još jedan Dense sloj od 300 neurona i *Relu* aktivacijskom funkcijom te izlazni sloj od 1 neurona koji predstavlja Q vrijednost aktiviran linearnom aktivacijskom funkcijom.

Struktura navedenih mreža je prikazana na slici 7.(Actor-gore, Critic-dolje).



Slika 6 Struktura Actor-Critic neuronskih mreža

3.5. Parametri

Parametri neuronskih mreža i agenta su preuzeti iz rada “Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control“[1].

```
agent = DDPGAgent(nb_actions=actions_n[0], actor=actor, critic=critic, batch_size=64, critic_action_input=action_input,
                  memory=memory, nb_steps_warmup_critic=1000, nb_steps_warmup_actor=1000,
                  random_process=random_process, gamma=.99)

agent.compile([Adam(lr=1e-4), Adam(lr=1e-3)], metrics=['mse'])
```

Slika 7 Parametri DDPG agenta

Veličina batch-a (minibatch u ovom slučaju) je parametar koji se mora prilagoditi sposobnostima računala. Isprobani su veličine 32 , 64, 128, 200. Najveće veličine su brzo konvergirale većim nagradama, ali se učestalo trening znao prekinut. Na kraju je postavljena veličina od 64.

Također je postavljen parametar koji pušta neuronske mreže da polako povećavaju stopu učenja to zadane u 1000 koraka kako bi se izbjegao rani *overfitting.* Gama parametar je jako važan u podržanom učenju, a pogotovo u kontinuiranim zadacima jer osigurava konvergiranje vrijednosti budućih nagrada ka nekom broju jer kontinuirani prostor ima neograničen broj mogućih akcija. Isto tako će više na značaju davati trenutnim nagradama , a manje onima u budućnosti.

Na objema mrežama korišten je Adam optimizator.

```
agent.fit(env, env_name=ENV_NAME, nb_steps=500000, action_repetition=5, visualize=False, verbose=1)
```

Slika 8 Parametri fit() funkcije

Agent je učen na 2500 epizoda, pri čemu je je ponavljao iste akcije 5 puta prije nego li je istraživao dalje što ubrzava proces učenja , a slabo utječe na performanse agenta.

[1] https://arxiv.org/pdf/1708.04133.pdf

4 REZULTATI

Učenje je provedeno na 2500 epizoda odnosno 500000 koraka na 5 različitih modela. Na slikama 10,11 i 12. prikazan je proces učenja i testiranja modela.

```
Training for 500000 steps ...
Interval 1 (0 steps performed)
2020-08-29 16:44:12.983195: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library librocblas.so
10000/10000 [==============================] - 148s 15ms/step - reward: -4.7714
50 episodes - episode_reward: -954.289 [-1750.002, 165.911] - loss: 3.217 - mse: 6.433 - mean_q: -23.502

Interval 2 (10000 steps performed)
10000/10000 [==============================] - 178s 18ms/step - reward: -0.9990
50 episodes - episode_reward: -199.803 [-1034.381, 358.908] - loss: 7.349 - mse: 14.698 - mean_q: -26.438

Interval 3 (20000 steps performed)
10000/10000 [==============================] - 170s 17ms/step - reward: -0.2644
50 episodes - episode_reward: -52.880 [-1120.052, 404.198] - loss: 13.913 - mse: 27.826 - mean_q: -18.320

Interval 4 (30000 steps performed)
10000/10000 [==============================] - 168s 17ms/step - reward: -0.1860
50 episodes - episode_reward: -37.194 [-1139.450, 425.160] - loss: 17.623 - mse: 35.246 - mean_q: -7.582
```

Slika 10 Prvih 200 epizoda treninga

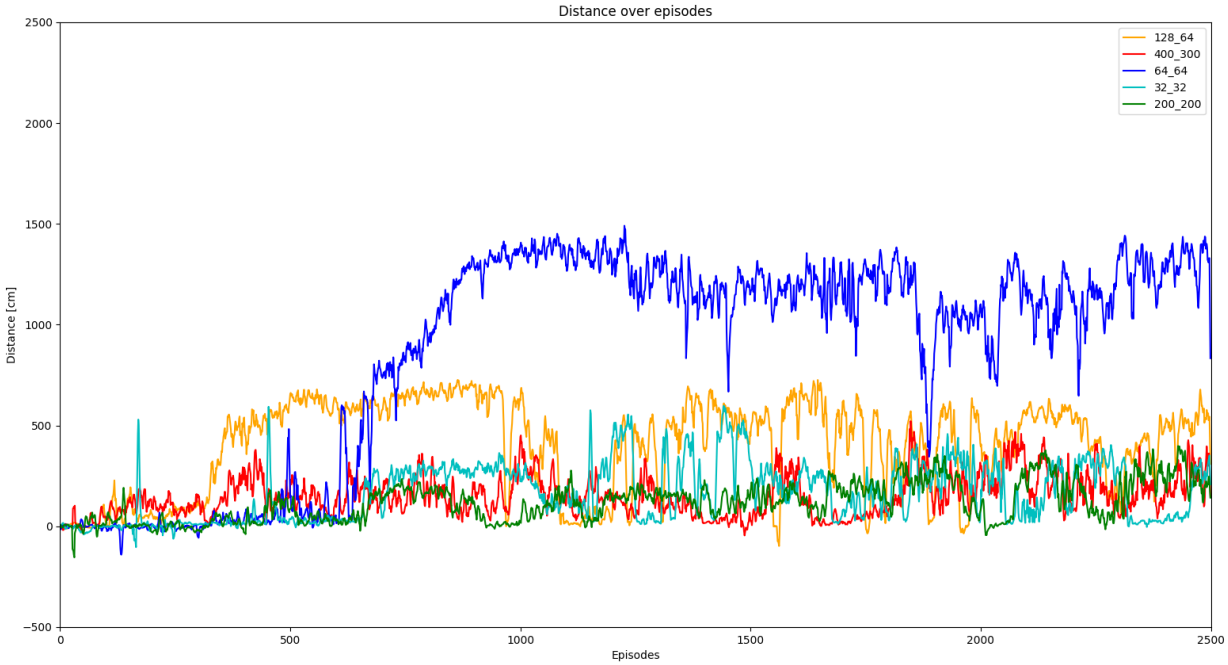
```
Interval 15 (140000 steps performed)
10000/10000 [==============================] - 189s 19ms/step - reward: 5.1726
50 episodes - episode_reward: 1034.514 [673.899, 1283.113] - loss: 247.349 - mse: 494.698 - mean_q: 304.272

Interval 16 (150000 steps performed)
10000/10000 [==============================] - 189s 19ms/step - reward: 5.0950
50 episodes - episode_reward: 1018.995 [7.914, 1222.569] - loss: 263.805 - mse: 527.609 - mean_q: 311.345

Interval 17 (160000 steps performed)
10000/10000 [==============================] - 189s 19ms/step - reward: 5.3641
50 episodes - episode_reward: 1072.815 [679.698, 1229.916] - loss: 259.333 - mse: 518.666 - mean_q: 319.316

Interval 18 (170000 steps performed)
10000/10000 [==============================] - 189s 19ms/step - reward: 5.4025
50 episodes - episode_reward: 1080.501 [729.814, 1306.441] - loss: 277.939 - mse: 555.877 - mean_q: 327.098

Interval 19 (180000 steps performed)
10000/10000 [==============================] - 189s 19ms/step - reward: 5.1317
50 episodes - episode_reward: 1026.341 [741.819, 1233.052] - loss: 296.584 - mse: 593.168 - mean_q: 333.809

Interval 20 (190000 steps performed)
10000/10000 [==============================] - 188s 19ms/step - reward: 5.2666
```

Slika 11 Zadnjih 300 epizoda.

```
Testing for 5 episodes ...
Episode 1: reward: 1129.183, steps: 1000
Episode 2: reward: 1092.632, steps: 1000
Episode 3: reward: 1310.326, steps: 1000
Episode 4: reward: 1086.311, steps: 1000
Episode 5: reward: 1124.010, steps: 1000
```

Slika 12 Testiranje na 5 epizoda

Kao što je već navedeno, učenje je provedeno na 5 modela. Svaki model je imao drugačiji broj neurona i skrivenim slojevima. Koristeći informaciju o trenutnom položaju centra mase robota u prostoru, dobivene iz okoline, postignuća različitih modela su prikazana pomoću prijeđenog puta robota tijekom epizoda. Prijeđeni put je direktno povezan sa nagradom kojom nastojimo maksimizirati. Što veći put prema naprijed robot pređe to je veća nagrada.



Slika 13 Usporedba postignuća modela

Na slici 13. prikazana su dobivena postignuća različitih modela. Negativne vrijednosti predstavljaju pomicanje robota u nazad što pridonosi negativnoj nagradi to jest pomaku. Vidljivo je da određene kombinacije variraju oko iste srednje vrijednosti kroz većinu procesa učenja (zelena i crvena boja na grafu). To je posljedica nedovoljnog istraživanja prostora akcija u 2500 epizoda. Mogući utjecaj je i nesretna početna inicijalizacija agenta koja je nasumična. Dvije kombinacije pokazuju nešto bolje rezultate (narančasta i tirkizna boja). Te dvije kombinacije pokazuju nešto veće istraživanje prostora što je vidljivo u oscilacijama n a grafu. Kombinacija sa 128 i 64 neurona vrlo rano postiže dobre rezultate, ali se do kraja učenja zadržava oko iste srednje vrijednosti. Kao najbolja kombinacija pokazala se ona sa 64 neurona u svim skrivenim slojevima(plava boja). Vidljivo je da postiže najbolje rezultate i ima puno duži rast od ostalih. Taj agent je uspio preći i po 15m u epizodi.

Jedan od glavnih problema u podržanom učenju je omjer istraživanja i iskorištavanja informacija. Veće istraživanje znači više informacija o mogućim akcijama dok više iskorištavanje informacija predstavlja veću mogućnost nalaska optimalnih rješenja u pretraženom prostoru. Potrebno je imati dobar omjer jednog i drugog. Najbolji agent je uspio postići dobre rezultate, ali ne i vrhunske. Naučio je koristiti zadnju nogu što mu je donijelo veliki napredak u prijeđenom putu, ali ga je spriječilo da rano nauči dobro koristiti i prednju nogu.

Broj epizoda je premal da bi mogli izvući sigurne zaključke o kvaliteti modela. Lošiji modeli iz eksperimenta bi mogli puno bolje napredovati kako se broj epizoda povećava jer možda istovremeno pokušavaju naučiti agenta koristiti i prednju i zadnju nogu dok najbolji model možda ostane na trenutnoj vrijednosti prijeđenog puta jer je preveliku važnost dao korištenju zadnje noge pa agent možda više neće naučiti koristiti i prednju. Učenje bi se trebalo provesti dok određeno vrijeme nema nikakvih ili vrlo malih promjena u postignućima svih modela.

5 ZAKLJUČAK

Pojačano učenje je danas vrlo popularno za učenje agenta čija se učinkovitost uspoređuje s ljudskom. Ona je često veća od ljudske. Agenti mogu jako brzo učiti, ali su naučeni samo za specifične zadatke dok se u novim okolnostima ne snalazi. Još smo daleko od nekog inteligentnog generalnog AI-a, ali nas odlično služe za specifične zadatke i zamjenjuju dosta ljudskih poslova. Samo podržano učenje je napravljeno na principu čovjeka koji uči na pokušajima i pogreškama.

DDPG metoda je jako osjetljiva na parametre stoga treba napraviti jako dobru optimizaciju parametara pri korištenju ove metode. U većini radova DDPG je postizao solidan rezultat tek nakon 10000 epizoda.

Nemogućnost korištenja većeg Minibatch-a je također utjecalo na rezultate to jest smanjilo je istraživanje prostora.

Iako je DDPG noviji algoritam i ima jako dobre rezultate na ovom okruženju , stalno se pojavljuju noviji i bolji algoritmi. Podržano učenje je prilično mlado područje i treba očekivati njegov nagli rast.

Jedno od ograničenja je dakako i samo sklopovlje koje mora podržati te zahtjevne algoritme.