LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

NES SPĒĻU SPĒLĒŠANA, IZMANTOJOT IMITĀCIJU MĀCĪŠANOS

BAKALAURA DARBS

Autors: **Eduards Jubels**
Studenta apliecības Nr.: ej11054
Darba vadītājs: docents Dr. dat. Agris Šostaks

RĪGA 2019

ANOTĀCIJA

Mākslīgā intelekta pielietojumi kļūst aizvien būtiskāki un izplatītāki. Pētījumā apskatīta imitāciju mācīšānās metode, kas var paātrināt mācīšanās procesu problēmu risināšanai, ja ir pieejamas eksperta demonstrācijas.

Par pētījuma risināmo problēmu izvēlēta NES spēļu spēlēšana, izmantojot vispārīgu algoritmu, kas nav atkarīgs no katras spēles noteikumiem. Darbā realizēti divi imitāciju mācīšanās algoritmi, izmantojot pārraudzīto mācīšanos un stimulēto mācīšanos. Abi piegājieni beidzās neveiksmīgi.

Atslēgvārdi: mašīnmācīšanās, dziļā mašīnmācīšanās, mākslīgie neironu tīkli, imitāciju mācīšanās, uzvedības klonēšana, stimulētā mācīšanās

ABSTRACT

NES GAME PLAYING VIA IMITATION LEARNING

Artificial intelligence uses are becoming more and more significant and widespread. In this paper we look at the method of imitation learning, which can speed up the learning process for problem solving, given that there is access to expert demonstrations.

The problem to be solved in this paper was NES game playing using an algorithm that is not dependent on the rules of individual games. The author implemented two imitation learning algorithms using supervised learning and reinforcement learning. Both attempts ended without positive results.

Keywords: machine learning, deep learning, artificial neural networks, imitation learning, behavioral cloning, reinforcement learning

SATURS

APZĪMĒJUMU SARAKSTS

NES – 1983. gadā izlaista spēļu konsole “Nintendo Entertainment System”. [1]

Ņemot vērā to, ka daudziem terminiem vai nu nav tulkojuma latviešu valodā, vai arī tulkojums nav plaši pazīstams, vēršu uzmanību šajā darbā lietotiem tulkojumiem.

policy – rīcības funkcijas

reward function – stimula funkcija

behavioral cloning – uzvedības klonēšana

activation function – aktivācijas funkcija

supervised learning – pārraudzītā mācīšanās

IEVADS

Mākslīgais intelekts ir kļuvis par vienu no aktuālākajām problēmām datorikā. Tā pielietojumu klāsts kļūst arvien plašāks un būtiskāks. Imitāciju mācīšanās ir viens no līdzekļiem, ar kuru mācīšanās procesu var krietni paātrināt, ja ir pieejami eksperti no kuriem mācīties. [2] Imitāciju mācīšanās ir mācīšanas, kuras ietvaros aģents cenšas atkārtot demonstratora uzvedību kādā uzdevumā.

Darba mērķis ir, pirmkārt, iemācīt mākslīgam aģentam spēlēt "Super Mario Bros" un "Dr. Mario" spēles, dodot cilvēka spēlētāja (demonstratora) piemērus un, otrkārt, izvērtēt imitāciju mācīšanās efektivitāti un problemātiku.

1. NES DATU PĀRSKATS

Par eksperimentu pamatu izvēlējos spēles “Super Mario Bros” un “Dr. Mario”. Apskatīsim NES spēļu konsoles saistošos aspektus.

1.1. Attēls [1]

NES krāsu palete kopā satur 52 atšķirīgas krāsas, taču vienlaicīgi visas krāsas ekrānā nevar attēlot. NES izmanto divas 13 atšķirīgu krāsu paletes – vienu fona attēlam, otru spraitiem (grafiskiem objektiem), taču abas paletes satur fona jeb caurspīdības krāsu, tāpēc kopējais atšķirīgu krāsu skaits vienlaicīgi ekrānā ir ne vairāk kā 25.

NES spēļu konsoles PPU (*Picture Processing Unit*) ģenerē kopējo attēlu 256x240 pikseļu lielumā. Fona attēls sākotnēji ir sadalīts 960 (32x30) kvadrātos 8x8 pikseļu lielumā, taču pastāv iespēja to bīdīt horizontāli vai vertikāli. Spraiti ir 8x8 un 8x16 pikseļu lielumā un var atrasties jebkur ekrānā, tai skaitā daļēji aiz fona attēla.

1.2. Pults

Oriģinālajai NES pultij ir 8 taustiņi – *A, B, Start, Select, left (pa kreisi), right (pa labi), up (uz augšu), down (uz leju).* Taustiņi *Start* un *Select* spēlēšanai praktiski nav nepieciešami, tos izmanto, piemēram, lai uzsāktu spēli, tāpēc tos varam neņemt vērā. Atliek 6 taustiņi, ar kuriem varam veidot taustiņu kombinācijas. Taustiņus *uz augšu* un *uz leju*, kā arī *pa kreisi* un *pa labi* vienlaicīgi nav iespējams nospiest pults uzbūves dēļ, tādēļ šīs kombinācijas uzskatīsim par nelegālām. Līdz ar to kopā sanāk 36 iespējamas taustiņu kombinācijas (sk. 1.1. tabulu).

1.1. tabula

Pieļaujamās taustiņu kombinācijas

Nr.p.k.	Taustiņi	Nr.p.k.	Taustiņi	Nr.p.k.	Taustiņi
0.	visi atlaisti	12.	down	24.	up, right
1.	A	13.	down, A	25.	up, right, A
2.	B	14.	down, B	26.	up, right, B
3.	A, B	15.	down, A, B	27.	up, right, A, B
4.	up	16.	left	28.	down, left
5.	up, A	17.	left, A	29.	down, left, A
6.	up, B	18.	left, B	30.	down, left B
7.	up, A, B	19.	left, A, B	31.	down, left, A, B
8.	right	20.	up, left	32.	down, right
9.	right, A	21.	up, left, A	33.	down, right, A
10.	right, B	22.	up, left, B	34.	down, right, B
11.	right, A, B	23.	up, left, A, B	35.	down, right, A, B

1.3. Vides apstākļi

Visi eksperimenti tika veikti, izmantojot pašmodificētu nes-py [3] emulatoru, kurš integrēts OpenAI Gym [4] ietvarā. "Super Mario Bros" videi izmantoju pašmodificētu gym-super-mario-bros [5] pakotni. "Dr. Mario" vidi izstrādāju pats, analoģiski "Super Mario Bros" videi.

1.4. Izvēlētās spēles

"Super Mario Bros" spēle ir lineāra tipa platformeris. Spēle ir sadalīta vairākos līmeņos, kur mērķis ir aiziet līdz līmeņa labējam galam. Spēlētāja kustību traucē dažādi pretinieki ar dažādām īpašībām, kā arī bedres un platformas. Spēlētājs zaudē, pieskaroties jebkuram pretiniekam vai iekrītot kādā bedrē.

"Dr. Mario" ir līdzīga spēlei "Tetris". Burkā krīt kapsula, kas vienmēr ir vienā formā – ar divām komponentēm (kauliņiem). Katrs kauliņš var būt kādā no trim krāsām, kas var būt vienādas vai atšķirīgas. Spēles pamatmehānisms ir iespēja notīrīt 4 kauliņus, saliekot tos

vertikālā vai horizontālā rindā. Lai uzvarētu, šādā veidā jāiznīcina visi burkā esošie “vīrusi”, kas ir iekrāsoti kādā no trim krāsām.

2. UZVEDĪBAS KLONĒŠANA

Kā pirmo eksperimentu izvēlējos veikt uzvedības klonēšanu. Uzvedības klonēšana ir vienkāršākā imitāciju mācīšanās pieeja. Uzvedības klonēšana ir imitāciju mācīšanās problēmas redukcija līdz pārraudzītās mācīšanas problēmai. [6]

2.1. Uzvedības klonēšanas definīcija

Uzvedības klonēšana (*behavioral cloning*) ir imitāciju mācīšanās metode, kas reducē imitācijas problēmu līdz pārraudzītās mācīšanās (*supervised learning*) uzdevumam, kur ieejā ir stāvokļi un izejā ir darbības; stāvokļu—darbību pārus sagādā demonstrators. [6] [7]

2.2. Interpretācija un datu sagatavošana

Šīs pieejas būtība ir izveidot 36 klašu klasifikatoru, kur katra klase atbilst vienai no pieļaujamām taustiņu kombinācijām, kas klasificē spēles kadrus. No tā uzreiz redzams, ka aģents vienmēr, nonākot stāvoklī s, rīkosies identiski. Tādēļ ieejā izvēlējos padot 8 kadru sekvenci kā vienu piemēru, lai iekodētu kustību, izejā sagaidot optimālo taustiņu kombināciju, ko izpildīt nonākot šādā situācijā.

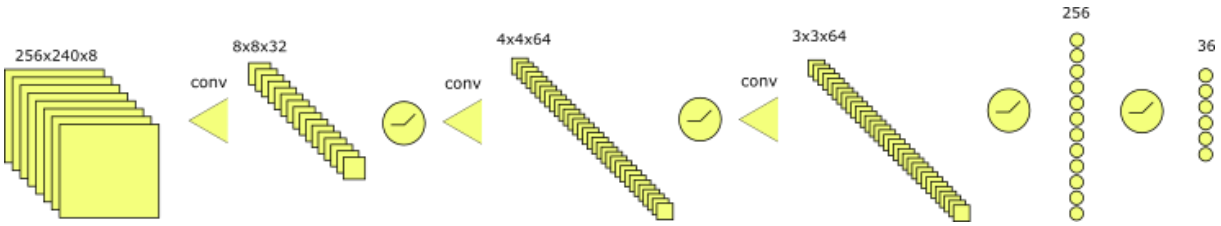
Pirms modeļa trenēšanas visus attēlus pārveidoju no RGB uz melnbaltu, katram pikselim pielietojot krāsu komponenšu svēršanas formulu (sk. 2.1. att.), kur Y – rezultējošais pikseļa spožums, R – sarkanās, G – zaļās un B – zilās krāsu komponenšu vērtības.

$$Y = 0.2125\,R + 0.7154\,G + 0.0721\,B$$

2.1. att. **Spožuma funkcija RGB attēlam** [8]

2.3. Neironu tīkla struktūra

Neirona tīklam izvēlējos secīgu slāņu struktūru (sk. 2.2. att.). Mākslīgais neironu tīkls pamatā ir konvolūciju tīkls. Lai gan konvolūciju tīkli pazīstami jau kopš 1989. gada, [9] tie tiek plaši izmantoti attēlu atpazīšanā tikai kopš 2012. gada. [10] Konvolūciju tīklos tiek izmantotas konvolūcijas, kas ir specializētas lineāras operācijas. [11]



2.2. att. **Mākslīgā neironu tīkla struktūra**

Pirmais slānis ir divdimensionāls konvolūciju slānis, kam ieejā padod astoņus 256x240 pikseļu melnbaltus kadrus. Filtra izmērs ir 8x8 pikseļi, solis (*stride*) ir 4 pikseļi abos virzienos. Konvolūcijas procesā viedojas 4 reizes mazāks 64x60 pikseļu attēls. Slāņa izejā ir 32 filtri, kam pielietota ReLU aktivācijas funkcija (sk. 2.3. att.), kas uzlabo objektu atpazīšanas spēju. [12]

$$f(x) = \max(0, x)$$

2.3. att. **ReLU aktivācijas funkcija**

Otrais slānis ir konvolūciju slānis ar 64 filtriem izejā, šoreiz ar 4x4 filtru un 2 pikseļu soli, kas tālāk sadala attēlu vēl 2 reizes abos virzienos, iegūstot 32x30 lielu attēlu, kam pielietota ReLU aktivācijas funkcija. Tad seko trešais konvolūciju slānis ar 3x3 filtru un 1 pikseļa soli, izejā dodot 64 filtrus, kam pielietota ReLU aktivācijas funkcija. Nākamais ir pilnsaistes slānis (*dense*) ar 256 mezgliem, kam pielietota ReLU aktivācijas funkcija. Pēc šī pilnsaistes slāņa pielietota atmešanas (*dropout*) metode, ar kuras palīdzību katram mezglam piešķiram 30% varbūtību tikt atmestam. Visbeidzot ir vēl viens pilnsaistes slānis, kas izejā dod 36 vērtības, kuras attiecīgi interpretēsim kā pults taustiņu kombinācijas, un rezultātam pielieto *softmax* aktivācijas funkciju, kas izsaka rezultātu par varbūtību sadalījumu (sk. 2.4. att.).

$$softmax(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

2.4. att. **Softmax funkcija** [11]

2.4. Trenēšana

Neironu tīkls tika implementēts, izmantojot Keras [13] ietvaru. Modelis tika trenēts pa 32 piemēru partijām, ierobežotās videokartes atmiņas dēļ. Katrai spēlei modelis tika trenēts atsevišķi, izmantojot datus no attiecīgās spēles.

Kopumā "Super Mario Bros" modelis tika trenēts ar 65 tūkstošiem kadru no 37 izspēlēm. "Dr. Mario" tika trenēts ar aptuveni 20 tūkstošiem kadru no 25 izspēlēm.

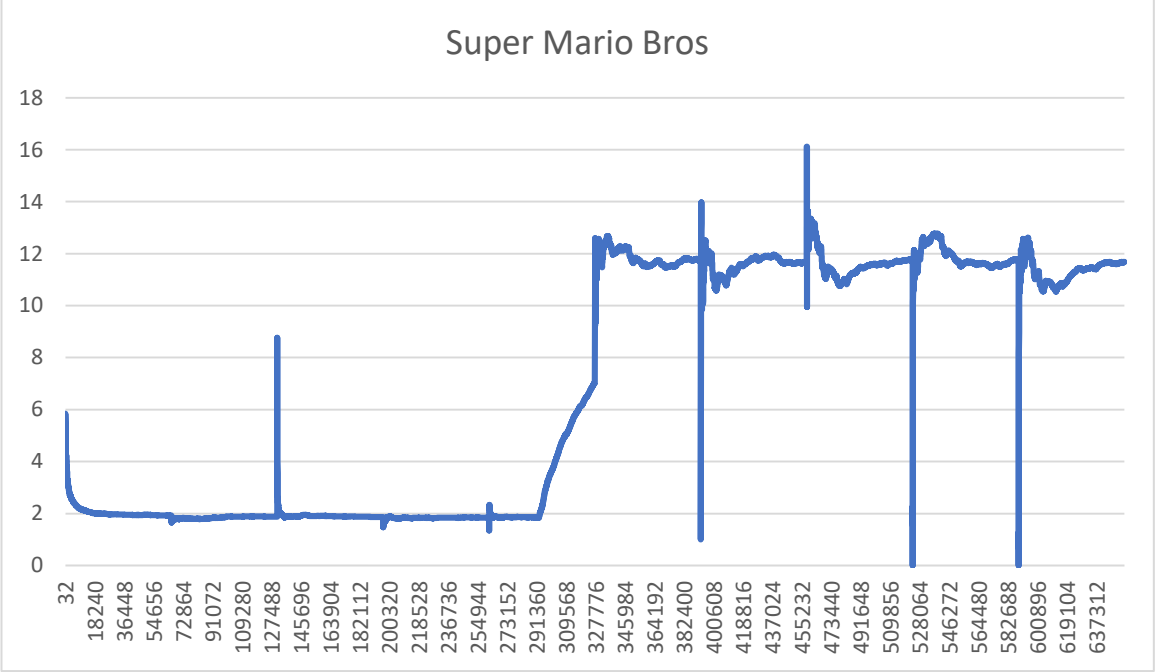
2.5. Rezultāti

"Super Mario Bros" klasifikatora precizitāte sasniedza 27%. Aģents iet pa labi līdz pirmajam pretiniekam, necenšas lēkt un tālāk netiek. Šis uzskatāms par ļoti vāju sniegumu.

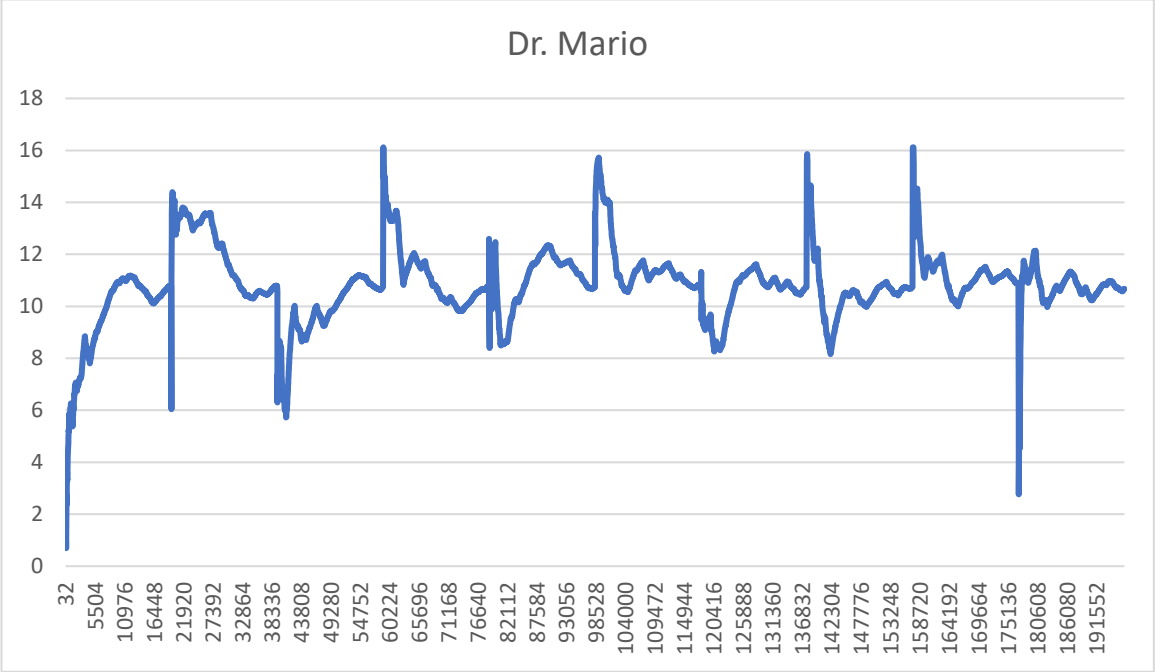
"Dr. Mario" klasifikatora precizitāte sasniedza 41%, taču faktiskā izspēlē aģents izpilda tikai vienu no 36 iespējām – krist uz leju. Šeit, atšķirībā no "Super Mario Bros", kam bija tikai viena, rezultējošā varbūtību sadalījumā bija 5 vērtības virs 1%, taču tās nemainījās pietiekami kadru no kadra, lai mainītu darbību ar maksimālo varbūtību. Šis arī uzskatāms par ļoti vāju sniegumu.

Šaubos, ka šī problēma ir atrisināma ar pārraudzītās mācīšanās rīkiem. Dotie treniņpiemēri neiekļauj lielu daļu stāvokļu kopas un pārsvarā ir doti tikai pozitīvi piemēri. Aģents, nonākot kādā nebijušā stāvoklī, nezina, kā rīkoties, jo nav uz to trenēts.

Tīkla trenēšana noritēja ļoti nestabili un konverģences pazīmes nav novērojamas (sk. 2.5. un 2.6. att.).



2.5. att. **”Super Mario Bros” kļūdas vērtība pirmajām 10 epohām**



2.6. att. **”Dr. Mario” kļūdas vērtība pirmajām 10 epohām**

3. PĀREJA UZ STIMULĒTO MĀCĪŠANĀS

Nākamā eksperimenta pamatā ir stimulētā mācīšanās (*reinforcement learning*).

3.1. Markova lēmumu procesa definīcija

Markova lēmumu process (MDP) ir vide, kas definējama kā kortežs $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, [14] kur

- $\mathcal{S}$ ir galīga stāvokļu, kuriem piemīt Markova īpašība, kopa,
- $\mathcal{A}$ ir galīga darbību kopa,
- $\mathcal{P}$ ir pāreju matrica $\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$, kas apzīmē varbūtību nonākt stāvoklī s', veicot darbību a stāvoklī s,
- $\mathcal{R}$ ir stimula funkcija $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$, kas apzīmē iegūto stimulu, izpildot darbību a stāvoklī s,
- γ ir stimula dilšanas koeficients, kas samazina gaidāmā stimula vērtību, atkarībā no tā, cik tālu nākotnē stimuls ir sagaidāms.

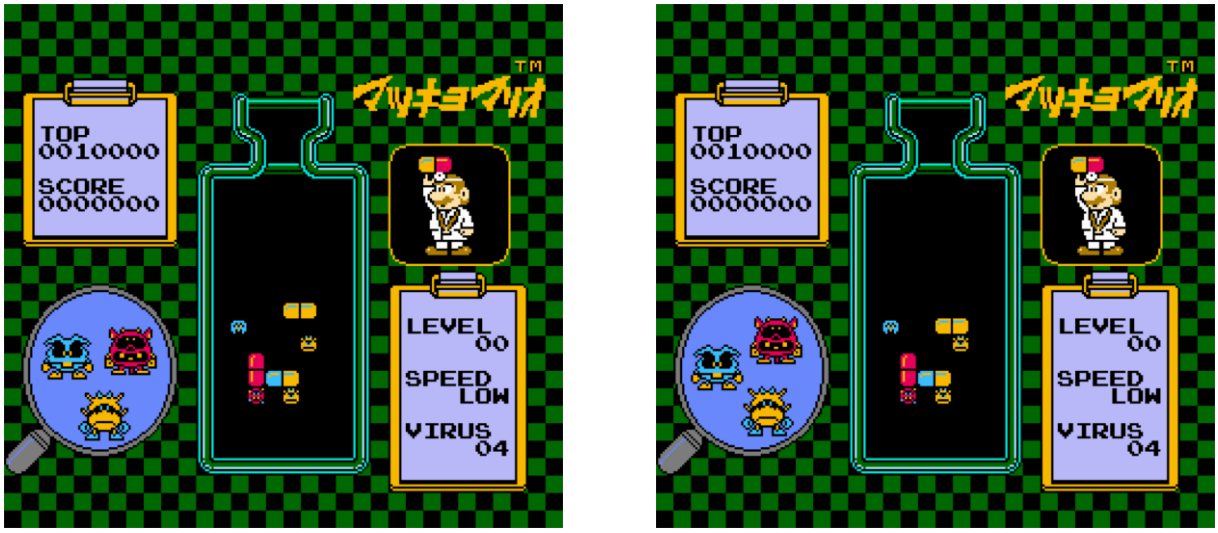
MDP pamatā ir mijiedarbība starp vidi un aģentu. Par aģentu dēvē mācekli un lēmumu pieņēmēju. Par vidi dēvē visu pārējo, t.i., visu, kas ir ārpus aģents. Aģents pastāvīgi mijiedarbojas ar vidi – aģents izvēlās darbības un vide attiecīgi reaģē uz šīm darbībām. Vide dod stimulu katrā laika solī, ko aģents cenšās maksimizēt. Katrā diskrētā laika solī t aģents saņem stāvokļa reprezentāciju $S_t \in \mathcal{S}$ un balstoties uz to, izvēlās kādu darbību $A_t \in \mathcal{A}(s)$. Laika solī $t + 1$ aģents saņem stimulu $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ un nonāk stāvoklī S_{t+1}. [15]

3.2. Markova lēmumu procesa adaptācija

Interpretēsim MDP mūsu gadījumā.

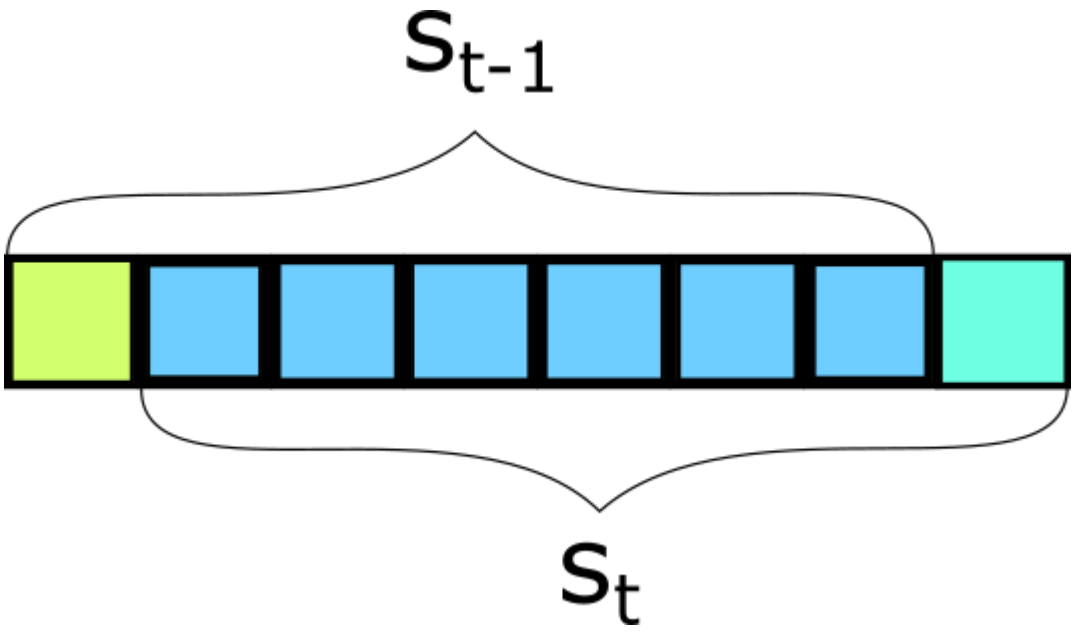
Par stāvokļu kopas $\mathcal{S}$ elementiem noteiksim spēles iespējamos stāvokļus. Intuitīvais piegājiens būtu definēt stāvokli kā kārtējo spēles kadru (ekrānā redzamo pikseļu matricu), taču šādi stāvoklim nepiemītu Markova īpašība. Markova īpašība nosaka to, ka varbūtība nonākt kādā no nākamajiem stāvokļiem nav atkarīga no pagātnes stāvokļiem, bet tikai no pašreizējā stāvokļa. Redzot tikai vienu kadru, ne vienmēr ir iespējams pateikt, kādā stāvoklī aģents nonāks nākamajā laika solī. Piemēram, ja kādam objektam piemīt ātrums, no viena kadra to bieži vien

nevar noteikt, kā arī, ja kustīgs objekts nav vizuāli atšķirams no nekustīgiem objektiem, tad nevar viennozīmīgi pateikt, vai objekts vispār ir kustībā (sk. 3.1. att.).



3.1. att. **Kustības noteikšanai nepieciešami vairāki kadri**

Šī iemesla dēļ par stāvokli izvēlējos pēdējo 8 kadru kopu $\phi(s_t) = \{s_{t-7}, s_{t-6}, s_{t-5}, s_{t-4}, s_{t-3}, s_{t-2}, s_{t-1}, s_t\}$. Šādā veidā redzams, ka katrā nākamajā stāvoklī ir 7 kadri no iepriekšējā un 1 jauns kadrs (sk. 3.2. att.).



3.2. att. **Stāvokļa ilustrācija**

Kopā $\mathcal{A}$ ietilpst visas 36 pieļaujamās taustiņu kombinācijas.

Pāreju matricu $\mathcal{P}$ nosaka katras spēles individuālie noteikumi, kas izriet no iekšējās, uzprogrammētās spēles loģikas.

Stimula funkcijas $\mathcal{R}$ izvēle ir ļoti būtiska, jo tieši tā ietver mācīšanās mērķi.

- "Super Mario Bros" par stimula funkciju izvēlējos doto $r = v + c + d$, [5] kur
  - r – stimuls
  - v – par katru pozīciju, ko aģents pavirzījies uz labo pusi +1
  - c – par katru laika soli – stimuls -1.
  - d – par nāvi -15 stimuls.
- "Dr. Mario" par stimulu noteicu +1 par katru nobeigto vīrusu un -1 par nāvi. Šīs vērtības tieši nolasu no emulatora operatīvās atmiņas.

Stimula dilšanas koeficientam izvēlos vērtību $\gamma = 0.99$. Šādi tiek noteikts, ka nākotnes stimuls ir nedaudz mazāk vērtīgs nekā tūlītējais.

3.3. DQN algoritms

Pamatā izvēlējos adaptēt DQN [16] algoritmu, pēc nepieciešamības pielāgojot to. DQN ir stimulētās mācīšanās algoritms, kas pielieto dziļās mašīnmācīšanās metodes. Algoritma procesā tiek uztrenēts Q-tīkla aģents, kas, saņemot ieejā stāvokli, izdod gaidāmo stimulu katrai darbībai. Algoritma mērķis ir tuvināti atrast optimālo rīcības funkciju π^*, izmantojot Q-funkciju (sk. 3.3. att.).

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \mid s_t = s, a_t = a, \pi\right]$$

3.3. att. **Optimālā Q-funkcija**

Dots parametru vektors θ. Rīcības funkcija π ir funkcija, kas, saņemot vides stāvokli s, nosaka veicamo darbību a (sk. 3.4. att.). Tā apzīmē varbūtību veikt darbību a stāvoklī s.

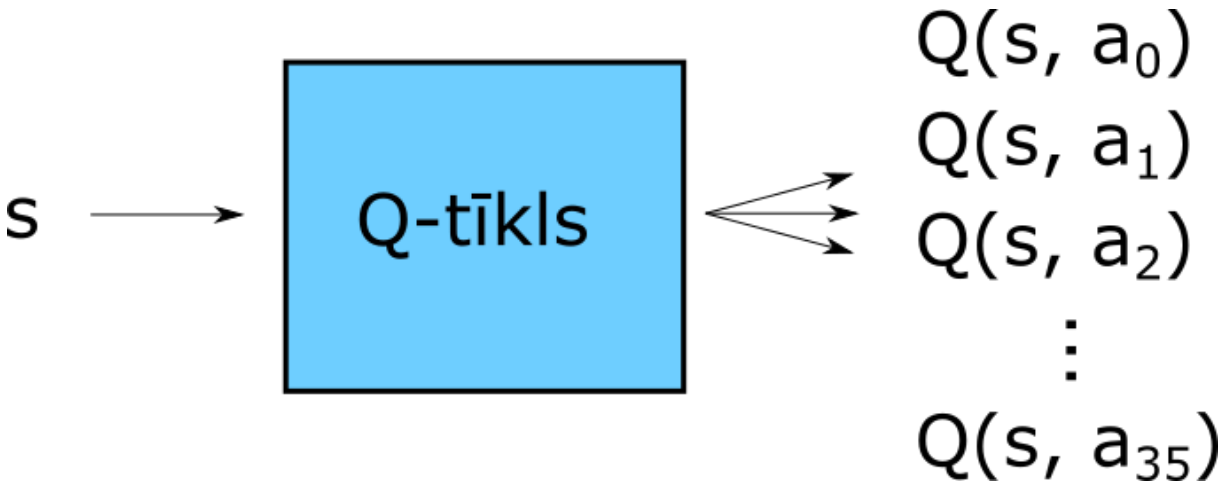
$$\pi(a \mid s, \theta)$$

3.4. att. **Rīcības funkcija** [15]

Q-funkija apzīmē "kvalitāti" jeb gaidāmo stimulu summu, izpildot darbību a stāvoklī s. Zinot Q-funkciju, lai rīkotos optimāli, atliek katrā stāvoklī rēķināt Q-vērtības visām darbībām un izvēlēties to darbību, kurai Q-vērtība šajā stāvoklī ir visaugstākā.

Lai atrastu Q-funkciju, būtībā ir jāatrod visas iespējamās Q-vērtības, taču mūsu gadījumā stāvokļu kopa ir pārāk liela, lai Q-funkciju realizētu kā tabulu, kur glabātos attiecīgās Q-vērtības katram stāvokļa-darbības pārim, tāpēc tiek izmantots mākslīgs neironu tīkls, kas tuvināti aprēķina Q-vērtības.

Par tīkla struktūru izvēlējos to pašu, ko iepriekšējā eksperimentā, ar vienīgo atšķirību – nepielietojot softmax aktivācijas funkciju rezultātam. Šis Q-tīkls tuvināti aprēķina Q-vērtības visām darbībām, saņemot ieejā tikai stāvokli s (sk. 3.5. att.).



3.5. att. **Q-tīkls**

Q-tīkla parametru θ trenēšanai tiek dots stāvoklis s kā ieeja un par mērķi noteikts tūlītējais stimuls, ko iegūst katrā no stāvokļiem s', attiecīgi izpildot katru no iespējamām darbībām A stāvoklī s, plus stimuls, kas tiks saņemts, izpildot optimālo darbību a' stāvoklī s', pareizināts ar stimula dilšanas koeficientu γ (sk. 3.6. att.).

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

3.6. att. **Mācīšanās mērķis**

Par cik mums nav zināma Q-funkcijas vērtība, mācīšanās mērķis nav tieši izrēķināms. Savukārt, ja Q-funkcija būtu zināma, tad uzdevums jau būtu atrisināts. Tādēļ Q-funkciju rēķina iteratīvi. Sākotnēji Q-tīkla parametrus inicializē nejauši.

Mijiedarbojoties ar vidi, aģents saņem stimulu. Šī procesa trajektoriju glabājam atmiņā un varam izmantot to kā treniņa datus. Spēlējot spēli, katrā laika solī glabājam atmiņā kortežu (s, a, r, s'), kur

- s – iepriekšējais stāvoklis, kurā aģents atradās,
- a – darbība, kuru aģents izpildijā stāvoklī s,
- r – stimuls, ko aģents saņēma, izpildot darbību a stāvoklī s,
- s' – stāvoklis, kurā aģents nonācis, izpildot darbību a stāvoklī s.

Izmantojot Q-tīklu ar pašreizējiem parametriem, izrēķinām Q-vērtības stāvoklim s'. No šīm vērtībām izvēlamies lielāko un atceramies. Tad, par mācīšanās mērķi stāvoklim s varam noteikt r, kam pieskaitīta stāvokļa s' lielākā Q-vērtība, pareizināta ar stimula dilšanas koeficientu γ. Visbeidzot, aprēķina pašreizējo tuvinājumu Q-vērtībai stāvoklī s un trenē Q-tīklu ar šo vērtību starpības (attāluma) kvadrātu (sk. 3.7. att.). Q-tīkla trenēšanas procesā tiek mazināta šī kļūdas vērtība.

$$\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)^2$$

3.7. att. **Kļūda**

3.4. Treniņa dati

Treniņa datus varam ģenerēt trijos veidos, proti,

- izvēloties darbības nejauši,
- izvēloties darbības, balstoties uz pašreizējo Q-funkcijas tuvinājumu,
- no cilvēka demonstratora (imitāciju mācīšanās).

Mūsu darba fokuss ir uz pēdējo metodi. Imitāciju mācīšanās mērķis ir uztrenēt aģenta rīcības funkciju, izmantojot demonstrācijas. [17]

Ar π^A apzīmēsim aģenta (mācekļa) rīcības funkciju, savukārt ar π^E – eksperta (demonstratora) rīcības funkciju. Mūsu mērķis ir atrast tādu π^A, kam $Q(\pi^A) \geq Q(\pi^E)$. [18] Citiem vārdiem – atrast rīcības funkciju, kas ir tikpat laba vai labāka nekā eksperta rīcības funkcija.

Tomēr šajā gadījumā tikai ar cilvēka eksperta piemēriem nepietiek. Eksperta piemēri ir noderīgi kā papildus dati, kas rāda kvalitatīvas trajektorijas. Lai algoritms strādātu, ir nepieciešams trenēt Q-tīklu ar pēc iespējas vairāk stāvokļiem. Ja aģents mijiedarbojoties ar vidi nonāk stāvoklī, ar kuru Q-tīkls nav bijis trenēts, aprēķinātie Q-vērtību tuvinājumi būs krietni neprecīzāki nekā tiem stāvokļiem, ar kuriem Q-tīkls ir trenēts. Taču nav iespējams apskatīt visus stāvokļus, par cik mūsu stāvokļu kopa ir pārak liela. Tāpēc ir būtiski, ka Q-tīkls spēj ģeneralizēt stāvokļa reprezentāciju. Empīriski rezultāti liecina, ka konvolūciju tīkli labi spēj atpazīt objektus attēlos. [10] [19]

DQN algoritms lieto parametru ε, kas ir varbūtība izvēlēties nejaušas rīcības piemēru, savukārt $1 - \varepsilon$ ir varbūtība izvēlēties rīcības piemēru, balstoties uz pašreizējo Q-funkcijas tuvinājumu. Savā algoritma adaptācijā ieviesu vēl papildu parametru φ (notācija patvaļīga), kas apzīmē varbūtību izvēlēties eksperta dotu piemēru. Līdz ar to, jaunā varbūtība izvēlēties rīcības piemēru, balstoties uz pašreizējo Q-funkcijas tuvinājumu, ir $1 - \varepsilon - \varphi$. Pastāv invariants $\varepsilon + \varphi \leq 1$. Šīs vērtības var nebūt konstantas, piemēram, var samazināt parametru vērtības, atkarībā no uztrenēto piemēru skaita.

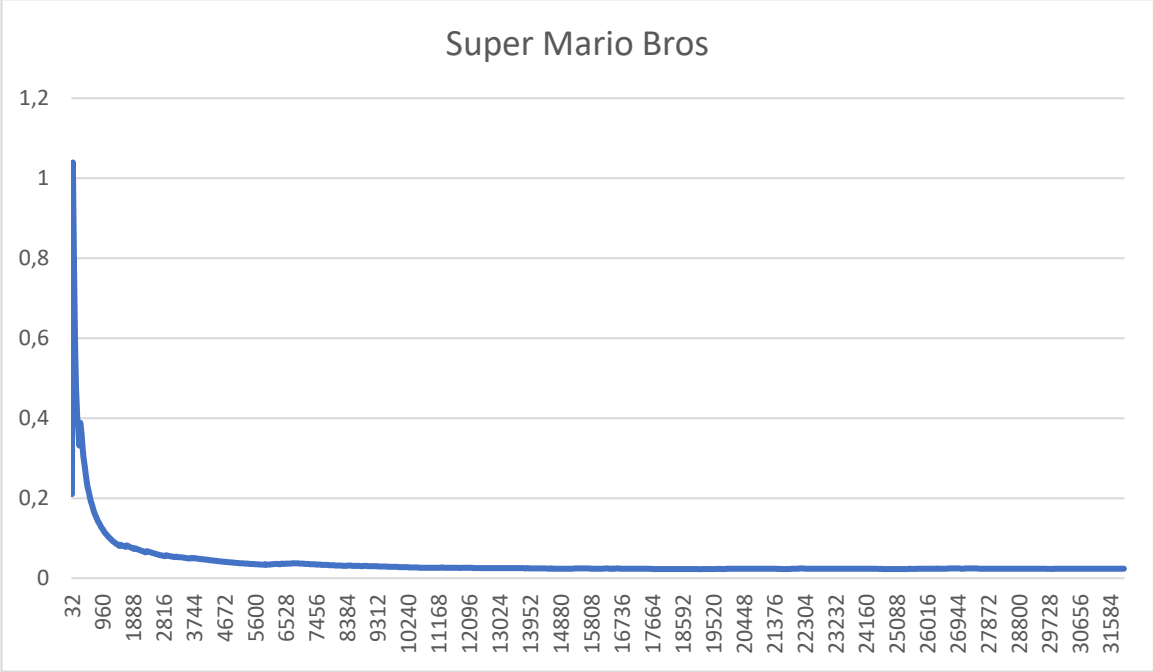
Savā eksperimentā izvēlējos trenēt Q-tīklu ar $\varepsilon = 0.2$ un $\varphi = 0.5$, ko periodiski samazināju par 0.1 līdz $\varphi = 0.2$. Trenēšanu veicu pa 32 piemēru partijām.

3.5. Rezultāti

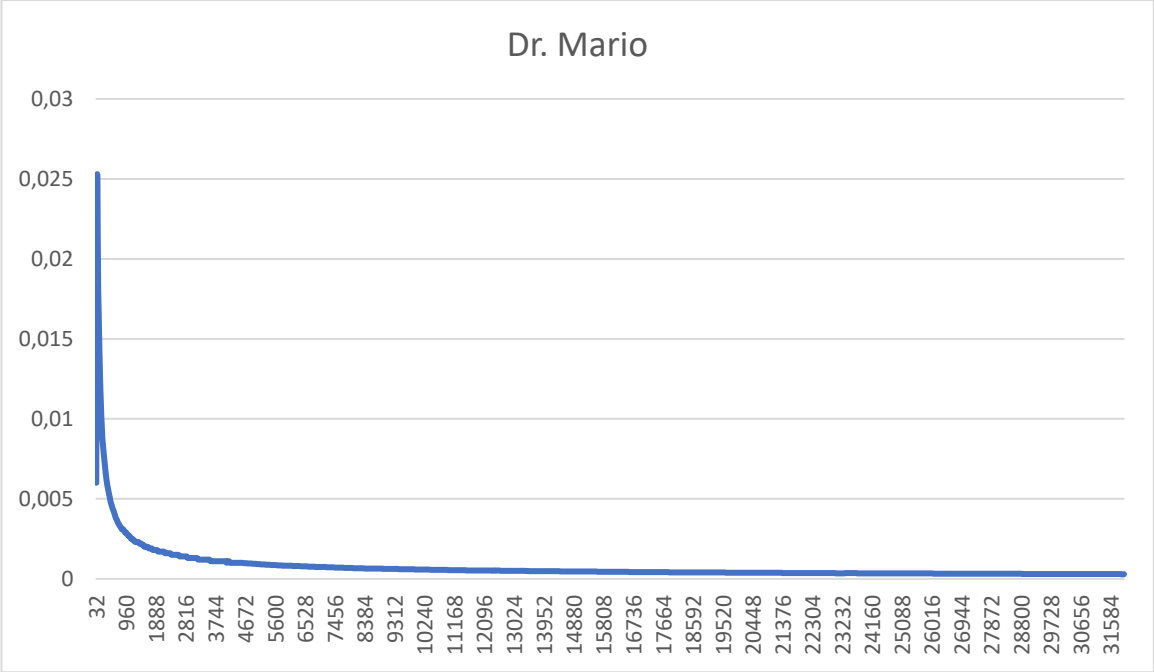
"Super Mario Bros" aģents pēc 10 miljoniem kadru apstrādes negūst jēgpilnus rezultātus. Aģents, neatkarīgi no ievada, izpilda vienu darbību – iet pa labi. Šādi tiek līdz pirmajam pretiniekam un nespēj to apiet.

"Dr. Mario" aģents pēc 10 miljoniem kadru apstrādes nespēj gūt rezultātus. Aģents visās situācijas izpilda vienu un to pašu darbību – uz leju.

Treniņa laikā novērojama tīklu konverģence (sk. 3.8. un 3.9. att.).



3.8. att. **"Super Mario Bros" kļūdas vērtība pirmajiem 32 tūkstošiem kadru**



3.9. att. **"Dr. Mario" kļūdas vērtība pirmajiem 32 tūkstošiem kadru**

REZULTĀTI

Darbā konstatēju, ka imitāciju mācīšanās redukcija uz pārraudzīto mācīšanos šai problēma nav pielietojama. Mākslīgo neironu tīklu trenēšanas procesā netika novērota konverģence.

Savukārt ar stimulētās mācīšanās pieeju Q-tīkli konverģēja, taču iegūtie risinājumi nespēj gūt pat minimālus rezultātus, jo faktiski nespēj atšķirt ieejas datus. Ar šo pieeju potenciāli problēmu varētu risināt, tomēr nepieciešama turpmāka izpēte, lai tiktu skaidrībā ar Q-tīkla nespēju atšķirt ieejas datus.

DISKUSIJA

Diemžēl rezultāti nav apmierinoši. Darba gaitā kļuva skaidrs, ka īstenot praktiski strādājošus dziļās mašīnmācīšanās algoritmus, it īpaši dziļās stimulētās mācīšanās algoritmus, ir krietni sarežģītāk, nekā sākotnēji šķita. Reproducēt iepriekšējus pozitīvus rezultātus ir ļoti sarežģīti. [20] Iespējams, ka man ir radies maldīgs uzskats, lasot rakstus un publikācijas par veiksmīgiem rezultātiem, reti sastopoties ar negatīviem rezultātiem, par cik šādu publikāciju ir mazāk un tās arī netiek tik plaši izplatītas.

Viena no būtiskajām problēmām ir treniņa ilgums. Kā zināt, kad pienācis laiks apstāties? Līdzīgai problēmai, Atari spēļu konsolei, literatūrā minēti 10, 50, 100, 200 miljoni kadru treniņpiemēri. [21] Ar manu aparatūru (viena "NVIDIA GeForce 940M" videokarte) 200 miljonu kadru apstrāde aizņemtu ap 65 diennaktīm katrai spēlei, kas ir stipri par daudz, lai būtu jebkādā ziņā praktiski. Tātad pirmais secinājums – nepieciešams lietot jaudīgāku aparatūru.

SECINĀJUMI

Imitāciju mācīšanās var būt par labu rīku trenēšanas procesa sākumā, taču ar to vien nepietiek. Galu galā aģentam ir jāmācās arī no savām kļūdām. Tomēr imitāciju mācīšanās ir spēcīgs rīks, ja pielietojums ir adekvāts. Nesen veiktajā AlphaStar projektā tika veiksmīgi izmantota imitāciju mācīšanās, uzsākot ar mācīšanos no cilvēku spēlēm. [22]

Viena no problēmām ar stimulētās mācīšanās pieeju ir nepieciešamība pēc efektīvas stimula funkcijas. Noteikt stimula funkciju bieži vien ir sarežģīti, tāpēc ir vērts pētīt imitāciju mācīšanās algoritmus, kuros stimula funkciju nosaka algoritmiski, vadoties pēc eksperta demonstrācijām. [23]

Eksperta demonstrācijas var būt sarežģīti vai pat neiespējami ievākt tadā pašā kontekstā, kā to sagaida aģents. Ja ievāktās demonstrācijas ir dotas no citiem skatu punktiem, tās ir nepieciešams translēt attiecīgajā kontekstā, pirms aģents var mācīties no tām. [24]

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] P. Diskin, "Nintendo Entertainment System Documentation," August 2004. [Tiešsaiste]. Pieejams: http://www.nesdev.com/NESDoc.pdf. [Piekļūts 2019].

[2] C. G. Atkeson un S. Schaal, "Learning tasks from a single demonstration," *Proceedings of International Conference on Robotics and Automation*, 25-2.

[3] C. Kauten, "nes-py," [Tiešsaiste]. Pieejams: https://pypi.org/project/nes-py/. [Piekļūts 2019].

[4] OpenAI, "Gym," [Tiešsaiste]. Pieejams: https://gym.openai.com/. [Piekļūts 2019].

[5] C. Kauten, "gym-super-mario-bros," [Tiešsaiste]. Pieejams: https://pypi.org/project/gym-super-mario-bros/. [Piekļūts 2019].

[6] Y. Yue un H. M. Le, "ICML2018: Imitation Learning," 2018. [Tiešsaiste]. Pieejams: https://sites.google.com/view/icml2018-imitation-learning/. [Piekļūts 2019].

[7] F. Torabi, G. Warnell un P. Stone, "Behavioral Cloning from Observation," *CoRR*, sēj. abs/1805.01954, 2018.

[8] C. Poynton, "Frequently Asked Questions about Color," 1997. [Tiešsaiste]. Pieejams: https://poynton.ca/PDFs/ColorFAQ.pdf. [Piekļūts 2019].

[9] Y. Lecun, "Generalization and network design strategies," in *Connectionism in perspective*, R. Pfeifer, Z. Schreter, F. Fogelman and L. Steels, Eds., Elsevier, 1989.

[10] A. Krizhevsky, I. Sutskever un G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems*, sēj. 25, 1 2012.

[11] I. Goodfellow, Y. Bengio un A. Courville, Deep Learning, MIT Press, 2016.

[12] V. Nair un G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ASV, 2010.

[13] F. Chollet, "Keras: The Python Deep Learning library," 2015. [Tiešsaiste]. Pieejams: https://keras.io/. [Piekļūts 2019].

[14] D. Silver, "Lecture 2: Markov Decision Processes," 2015. [Tiešsaiste]. Pieejams: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Piekļūts 2019].

[15] R. S. Sutton un A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 2018.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik,

I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg un D. Hassabis, "Human-level control through deep reinforcement learning," *Nature,* sēj. 518, p. 529, 2 2015.

[17] A. Hussein, M. M. Gaber, E. Elyan un C. Jayne, "Imitation Learning: A Survey of Learning Methods," *ACM Computing Surveys (CSUR),* sēj. 50, nr. 2, June 2017.

[18] U. Syed un R. E. Schapire, "A Reduction from Apprenticeship Learning to Classification," *Advances in Neural Information Processing Systems 23,* J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel un A. Culotta, Red., Curran Associates, Inc., 2010, pp. 2253-2261.

[19] J. T. Springenberg, A. Dosovitskiy, T. Brox un M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," *arXiv e-prints,* p. arXiv:1412.6806, 12 2014.

[20] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup un D. Meger, *Deep Reinforcement Learning That Matters,* 2018.

[21] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht un M. Bowling, "Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents," *arXiv e-prints,* p. arXiv:1709.06009, 9 2017.

[22] DeepMind, "AlphaStar: Mastering the Real-Time Strategy Game StarCraft II," 24 January 2019. [Tiešsaiste]. Pieejams: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/. [Piekļūts 2019].

[23] A. Y. Ng un S. J. Russell, "Algorithms for Inverse Reinforcement Learning," *Proceedings of the Seventeenth International Conference on Machine Learning,* San Francisco, CA, USA, 2000.

[24] Y. Liu, A. Gupta, P. Abbeel un S. Levine, "Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation," *CoRR,* sēj. abs/1707.03374, 2017.

Bakalaura darbs „NES spēļu spēlēšana, izmantojot imitāciju mācīšanos” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Eduards Jubels _____

Rekomendēju/nerekomendēju darbu aizstāvēšanai *(nederīgo svītro vadītājs)*

Vadītājs: docents Dr. dat. Agris Šostaks _____ __.__.2019.

Recenzents: profesors Dr. dat. Jānis Zuters

Darbs iesniegts Datorikas fakultātē __.__.2019.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.__.2019. prot. Nr. _____

Komisijas sekretārs(-e): _____