RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts

Ēvalds Urtāns
Doktora studiju programmas "Datorsistēmas" doktorants

FUNKCIJU FORMĒŠANA DZIĻAJĀ MAŠĪNMĀCĪŠANĀ

Promocijas darba kopsavilkums

Zinātniskais vadītājs
profesors *Dr. sc. ing.*
Agris Ņikitenko

RTU Izdevniecība
Rīga 2021

Urāns Ē. Funkciju formēšana dziļajā mašīnmācīšanā. Promocijas darba kopsavilkums. — Rīga: RTU Izdevniecība, 2021. — 76. lpp.

Iespiests saskaņā ar promocijas padomes "RTU P-xxx" 2021. gada xx. mēneša lēmumu, protokols Nr. ? .

ISBN xxx-xxxx-xx-xxx-x (pdf)

PROMOCIJAS DARBS IZVIRZĪTS ZINĀTNES DOKTORA GRĀDA IEGŪŠANAI RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ

Promocijas darbs zinātnes doktora (*Ph. D.*) grāda iegūšanai tiek publiski aizstāvēts 2021. gada 1. decembrī Rīgas Tehniskās universitātes Datorzinātnes un informācijas tehnoloģijas fakultātē (attālināti).

OFICIĀLIE RECENZENTI

Dr. William Sayers,
University of Gloucestershire,
Lielbritānija,

Dr. Houxiang Zhang,
Norwegian University of Science and Technology,
Norvēģija

APSTIPRINĀJUMS

Apstiprinu, ka esmu izstrādājis šo promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā universitātē zinātnes doktora (*Ph. D.*) grāda iegūšanai. Promocijas darbs zinātniskā grāda iegūšanai nav iesniegts nevienā citā universitātē.

Ēvalds Urtāns …… (paraksts)
Datums: ……

Promocijas darbs ir uzrakstīts angļu valodā, tajā ir astoņas nodaļas, 27 attēli, 14 tabulu, pieci pielikumi, kopā 165 lappuses, ieskaitot pielikumus. Literatūras sarakstā ir 136 nosaukumi.

ANOTĀCIJA

Arvien lielāku zinātnieku interesi iegūst kļūdas funkciju konstruēšana dziļās mašīnmācīšanās uzdevumiem, jo empīriski iegūtās kļūdas funkcijas ieguvušas labākus rezultātus nekā standarta kļūdas funkcijas, kas tika iegūtas analītiski no matemātikas teorijas.

Šis darbs apraksta kļūdas funkciju un ar to saistītu metožu ietekmi uz dziļo stimulēto mašīnmācīšanos un dziļo metriku mašīnmācīšanos. Jaunizveidotā *MDQN* kļūdas funkcija pārspēja *DDQN* kļūdas funkciju *PLE* datorspēļu vidēs, un jaunizveidotā eksponenciālās trijotnes kļūdas funkcija pārspēja standarta trijotnes kļūdas funkciju sejas reidentifikācijas uzdevumā, izmantojot *VGGFace2* datu kopu. Tā sasniedz 85,7% precizitāti, izmantojot nulles-šāviena metodi. Šajā darbā iekļauts arī jaunizveidots *UNet-RNN-Skip* modelis, kas uzlabo veiktspēju vērtību funkcijai ceļu plānošanas uzdevumos. Tas iegūst tādu pašu rezultātu, kā vērību iterāciju algoritms 99,8 % gadījumu, un var tikt apmācīts uz 32x32 izmēra kartēm, bet lietots arī uz lielākām kartēm, piemēram 256x256. Jaunizveidotās kļūdas funkcijas un metodes tiek veiksmīgi lietotas vairākos komerciālos produktos sejas un balss reidentifikācijā, audiosignāla attīrīšanā un hromatogrāfijā.

Saturs

SAĪSINĀJUMI

A3C — asinhronie aktiera — kritiķa aģenti *(Asynchronous Actor-Critic Agents*)
ACER — aktiera — kritiķa aģenti ar pieredzes atmiņu (*Actor—Critic with Experience Replay*)
AI — mākslīgais intelekts (*Artificial Intelligence*)
ASR — automātiskā runas atpazīšana (*Automatic Speech Recognition*)
ConvNet, CNN — konvolūciju neironu tīkli (*Convolutional Neural Network*)
CPU — centrālais procesors (*Central Processing Unit*)
DQN — dziļais Q-tīkls (*Deep Q-Network*)
DDPG — dziļais deterministiskais politikas gradients (*Deep Deterministic Policy Gradient*)
DDQN — dubultais dziļais Q-tīkls (*Double Deep Q-Network*)
DNN — dziļais neironu tīkls (*Deep Neural Network*)
DML — dziļā metriku apmācība (*Deep Metric Learning*)
GPU — grafiskais procesors (*Graphical Processing Unit*)
GRU — periodiskā šūna ar vārtiem (*Gated Recurrent Unit*)
HPC — augstas veiktspējas skaitļošanas klasteris (*High Performance Cluster*)
HPLC — augstas veiktspējas šķidrā hromatogrāfija (*High Performance Liquid Chromatography*)
IMAPLA — svērta svarīguma aktiera-skolnieka arhitektūra (*Importance Weighted Actor-Learner Architecture*)
LSTM — ilgtermiņa-īstermiņa atmiņas šūna (*Long Short-Term Memory*)
MDQN — vairāku dziļo Q-modeļu kļūdas funkcija (*Multi Deep Q-Network*)
MERLIN — atmiņas, stimulētās mašīnmācīšanās un secinājumu tīkls (*Memory, RL, and Inference Network*)
ML — mašīnmācīšanās (*Machine Learning*)
PLE — PyGame apmācību vides (*PyGame Learning Environment*)
PPO — tuvinātā politikas optimizācija (*Proximal Policy Optimization*)
RL — stimulētā apmācība (*Reinforcement Learning*)
RNN — periodiskie neironu tīkli (*Recurrent Neural Network*)
ResNet — pārpalikuma konvolūciju neironu tīkli (*Residual Convolutional Neural Network*)
SLAM — vienlaicīga lokalizācija un kartēšana (*Simultaneous Localization and Mapping*)
SLR — sistemātisks literatūras apskats (*Systematic Literature Review*)

TD — atšķirība laikā (*Temporal Difference*)
UNet — U-formas konvolūciju neironu tīkli (*U-haped Convolutional Neural Network*)
VI — vērtību iterāciju algoritms (*Value Iteration Algorithm*)
VIN — vērtību iterāciju tīkls (*Value Iteration Network*)

1. VISPĀRĒJAIS DARBA APRAKSTS

1.1. Tēmas aktualitāte

Pēdējā desmitgadē dziļā mašīnmācīšanās ir pārspējusi klasiskās mašīnmācīšanās metodes, lai iegūtu tuvinātus risinājumus sarežģītām matemātiskām funkcijām daudzdimensiju datu kopām [115], [48], [118], [126]. Dziļās mašīnmācīšanās modeļi nereti tiek izmantoti ar nepārraudzīto vai daļēji pārraudzīto apmācību, lai iegūtu matemātisku funkciju tuvinājumus, kas apraksta sakarības datu kopā [69], [83], [19], [22]. Šādi modeļi tiek apmācīti, izmantojot kļūdas funkcijas, lai iegūtu latentas reprezentācijas ievades datiem, kas raksturo to īpašības, sakarības un loģiku. Pirms dziļās mašīnmācīšanās datu īpašību inženierija un ekspertu sistēmas sakarību atpazīšanai bija svarīgas pētījumu daļas, taču šodien kļūdas funkcijas un modeļu arhitektūras ir kļuvušas par aktuāliem pētāmiem objektiem [57].

Ar dziļajām mašīnmācīšanās metodēm ir iespējams iegūt augstāku precizitāti attēlu klasifikācijas uzdevumos [107], [45], dabīgās valodas modelēšanas uzdevumos [10], [95], runas atpazīšanas uzdevumos [78], [85], laika rindu uzdevumos [9], [72] un citos uzdevumos, kur ievades dati satur daudz dimensiju.

Dziļās mašīnmācīšanās metodes sasniedz arī augstākos rezultātus stimulētās mašīnmācīšanās uzdevumos datorspēļu vidēs un robotikā, kur vadības signālus senāk spēja dot tikai cilvēks [35], [103], [21], [76]. Modernās dziļās mašīnmācīšanāš metodes lēmumu analīzei spēj nodrošināt klasiskajām statistiskas metodēm līdzvērtīgu "caurspīdīgumu" [63], [93].

Promocijas darba mērķis ir izstrādāt jaunas kļūdas funkcijas, modeļus, lai iegūtu tuvinājumus jau zināmām funkcijām, balstoties uz empīriskiem pētījumiem. Vēsturiski lielākā daļa kļūdu funkciju ir iegūtas no statistikas, informācijas vai varbūtību teorijām, bet salīdzinoši nesen empīriski atrastas kļūdas funkcijas var tās aizstāt un iegūt labākus rezultātus [61], [89].

Lai sasniegtu mērķi, darbā tika pētītas vairākas aktuālas tēmas:

1. kļūdu funkciju formēšana dziļajā stimulētajā mašīnmācīšanā un dziļajā metriku mašīnmācīšanā (kļūdas funkcijas dziļajā stimulētajā mašīnmācīšanā tika pārbaudītas datorspēļu vidēs, lai ātrāk sasniegtu augstākus rezultātus, kas tiek novērtēti ar skaitlisku balvas vērtību, un balvas vērtība nereti tiek aprēķināta kā piešķirto punktu vērtība jeb rezultāta

vērtība, bet kļūdas funkcijas dziļajā metriku mašīnmācīšanā tika pārbaudītas sejas reidentifikācijas un balss reidentifikācijas uzdevumos);

2. modeļa arhitektūras un apmācību procesa izstrāde, lai tuvināti atrisinātu vērtību funkciju un paātrinātu vērtību iterāciju algoritma darbību;

3. eksperimenāla analīze jaunajām kļūdas funkcijām un metodēm, izmantojot akadēmiskās, sintētiskās un privātās datu kopas un vides, lai pārbaudītu to lietojumu praktiskos risinājumos.

Jaunās kļūdas funkcijas, kas tika izstrādātas šajā darbā, dziļajā metriku mašīnmācīšanā sasniedz 85,7 % precizitāti seju reidentifikācijas uzdevumos, savukārt standarta kļūdas funkcijas šājā pašā uzdevumā sasniedz 78,6 % precizitāti. Šī precizitāte tika sasniegta, izmantojot nulles šāviena mašīnmācīšanu [23], [116], kur sejas reidentifikācijas uzdevuma gadījumā tās sejas, kas tika iekļautas apmācību datu kopā, netika iekļautas testēšanas datu kopā, un tā vietā precizitāte tiek rēķināta, atrodot vislīdzīgākās sejas starp reģistrācijas seju paraugiem un neklasificētiem testa seju paraugiem. Šīs pašas kļūdas funkcijas tika lietotas arī balss reidentifikācijas uzdevumam, kur tika sasniegta 88,4 % precizitāte uz privātas datu kopas, kas tiek izmantota komerciālā produktā. Arī jaunās kļūdas funkcijas, kas tika izstrādātas dziļajai stimulētajā mašīnmācīšanā, sasniedza augstākus rezultātus *PyGame Learning Environment* vidē. Šīs pašas kļūdas funkcijas ir veiksmīgi tikušas lietotas arī komerciālā produktā analītiskajā organiskajā ķīmijā, lai veiktu šķīdinātāja koncentrācijas gradienta optimizāciju, kas nepieciešama, lai atdalītu savienojumus paraugos. Visbeidzot, tika izstrādāti arī jauni dziļās mašīnmācīšanās modeļi, lai uzlabotu veiktspēju jau labi zināmai vērtību funkcijai, ko izmanto ceļu plānošanas uzdevumos mobilajiem robotiem. Jaunais modelis ir spējīgs iegūt vērtību funkcijai tuvinātu rezultātu, izmantojot paralēlo skaitļošanu. Modelis iegūst vērtību karti vairākas reizes ātrāk par standarta metodi, izmantojot vērību iterāciju algoritmu.

1.2. Darba mērķis un uzdevumi

Promocijas darba mērķis ir uzlabot dziļās mašīnmācīšanās veiktspēju un apmācību rezultātus dažādiem lietojumiem, izmantojot jaunizveidotās kļūdas funkcijas un modeļu arhitektūras dziļo metriku mašīnmācīšanā un stimulētajā mašīnmācīšanā.

Jaunizveidotajām kļūdas funkcijām, kas aprakstītas šajā darbā, jāspēj ātrāk konverģēt apmācības laikā, jāiegūst labākus rezultātus praktiskiem lietojumiem un jābūt lietojamām dažādos uzdevumos, sākot ar klasifikāciju sejas reidentifikācijai, analītiskajai ķīmijai un arī aģentu vadībai dažādās sarežģītās vidēs, piemēram, 3D datorspēlēs.

Konverģence dziļajā mašīnmācīšanā tiek sasniegta, kad izpildās 1. vienādojums, kur relatīvā starpība starp vidējo vērtību kļūdas funkcijai $\mathcal{L}$, ņemot vērā ievades datus x un patiesos datus y, pašreizējā epohā un iepriekšējā epohā ir mazāka par δ.

$$|\frac{\mathcal{L}(f_\theta(x), y)_i}{\mathcal{L}(f_\theta(x), y)_{i-1}}| < \delta \tag{1}$$

Atkarībā no datu kopas tīrības δ var būt robežās no 0,1-0,001. Kļūdas funkcijas vērtībām nepieciešams samazināties, balstoties uz datu kopu vai apmācību vidi. Parasti kļūdas funkcijām ir globālais minimums ar vērtību nulle, izņemot stimulētajā mašīnmācīšanā, kur nereti nav iespējams noteikt patiesos stāvokļus, kas iegūst augstāko balvas vērtību vidē.

Uzdevumi, kas tika veikti promocijas darbā:

1. veikt literatūras analīzi eksistējošām kļūdas funkcijām, kas ir līdzīgas *DQN* [68] un tripleta kļūdas funkcijām [23];

2. izstrādāt jaunas kļūdas funkcijas, kas ir līdzīgas *DQN* un tripleta kļūdas funkcijām, izmantojot uz nulles šāviena apmācību;

3. izveidot jaunu modeli, lai iemācītos tuvināt vērtības funkciju, kas tiek izmantota vērtību iterāciju (*VI*) algoritmā [88];

4. analizēt rezultātus jaunizveidotajai *DQN* funkcijai dažādās datorspēļu vidēs un analītiskās ķīmijas uzdevumos;

5. analizēt rezultātus jaunizveidotās tripleta kļūdas funkcijas sejas reidentifikācijas uzdevumu veikšanā;

6. analizēt rezultātus jaunizveidotajam modelim, lai iegūtu vērtību funkcijas tuvinājumu, un salīdzināt tos ar pilnā vērtību iterāciju algoritma rezultātiem;

7. publicēt pētījumu rezultātus zinātniskās publikācijās un pievienot tās promocijas darbam.

1.3. Aizstāvamās tēzes

Promocijas darbā aizstāvamās tēzes:

1. jaunizveidotā *MDQN* kļūdas funkcija Q-vērtību stimulētās mašīnmācīšanās uzdevumos pārspēj *DQN* kļūdas funkcijas;

2. jaunizveidotā eksponenciālā tripletu kļūdas funkcija dziļo metriku apmācības uzdevumos pārspēj klasisko tripletu kļūdas funkciju;

3. jaunizveidotais *UNet-RNN-Skip* modelis uzlabo vērtību funkcijas veiktspēju, izmantojot to, lai atrisinātu vērtību iterācijas algoritma uzdevumu;

4. jaunizveidotās *MDQN* un tripletu kļūdas funkcijas var izmantot praktiskos risinājumos seju reidentifikācijai, runātāju reidentifikācijai, audio trokšņu noņemšanai un hromatogrāfijas uzdevumiem analītiskajā ķīmijā.

1.4. Pētījuma metodika

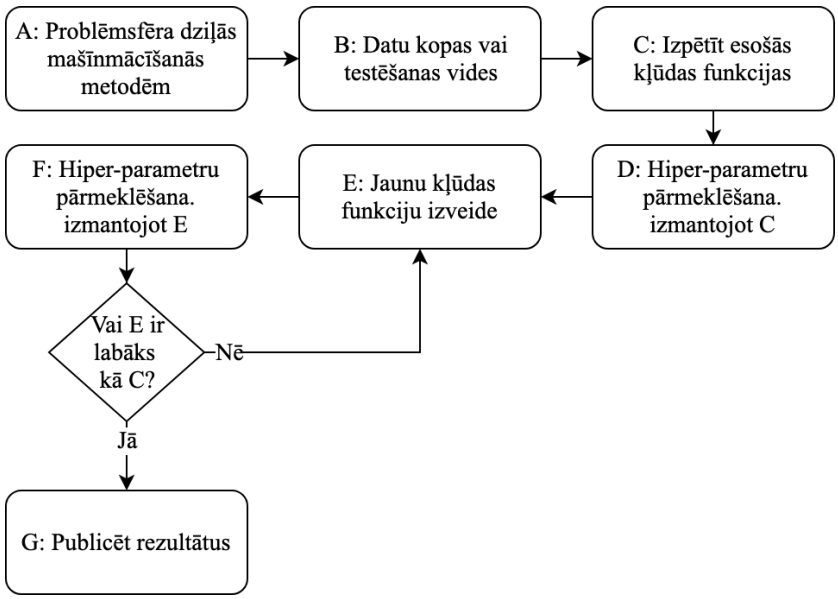
Promocijas darba mērķi mērķi tika sasniegti, izmantojot analītiskos un eksperimentālos pētījumus, kas publicēti zinātniskās publikācijās (1.6. nodaļa).

Promocijas darbā aprakstītajos pētījumos izmantotas eksperimentālās un analītiskās pētījumu metodes.

Kvantitatīvās un kvalitatīvās pētījumu metodes izmantotas, lai analizētu zinātnisko literatūru, jau eksistējošas un arī jaunas metodes.

Pētījumos primāri izveidotas jaunas kļūdas funkcijas un metodes, lai uzlabotu *DML* (*Deep Metric Learning*) un *RL* (*Reinforcement Learning*) modeļu veiktspēju un rezultātus.

Jaunizveidoto kļūdas funkciju izstrādes process shematiski parādīts 1. attēlā.



1. att. Vienkāršota darbību shēma jaunu kļūdas funkciju izveidei dziļajā mašīnmācīšanā.

Funkciju izveides process sākas ar problēmsfēras (A) izvēli, kas ir atbilstoša dziļajai mašīnmācīšanai. Šajā darbā problēmsfēras ietver *DML* balstītus latentos vektorus sejas un balss reidentifikācijas uzdevumiem; *RL*, izmantojot Q-vērtību funkcijas aģentu politikas apmācībai datorspēļu vidēs; hromatogrāfijas šķīdinātāju gradientu atrašana, lai labāk sadalītu parauga vielas pa sastāvdaļām analītiskajā ķīmijā, izmantojot *RL*.

Pēc tam nepieciešams atrast piemērotas datu kopas vai testēšanas vides (B). Šajā pētījumā tika izmantotas seju reidentifikācijas un attēlu klasifikācijas datu kopas un arī datorspēļu vides *RL* aģentu testēšanai.

Tad, izmantojot zinātnisko literatūru un jaunākās publikācijas, tika analizētas jau eksistējošas kļūdas funkcijas (C). Tālāk šīs funkcijas tika realizētas dziļās mašīnmācīšanās satvarā *PyTorch* [79] un tika pētītas to īpašības, ņemot vērā to parametrus.

Jau esošās kļūdas funkcijas tika testētas uz izvēlētajām datu kopām vai vidēm, pārmeklējot labākās hiperparameteru kombinācijas (D). Hiperparametri šajā gadījumā ir modeļa parametri, ko modelis apmācības laikā pats nemācās. Katrs apmācību mēģinājums tiek izpildīts, līdz tas sasniedz kļūdas konverģenci pret apmācību un testēšanas kļūdas funkciju skaitliskajām vērtībām. Konverģence dziļajā mašīnmācīšanā tiek sasniegta, kad kļūdas

funkcijas skaitliskā vērtība starp epohām vairs ievērojami nemainās. Gan apmācību, gan testēšanas kļūdas funkcijas rezultāta izmaiņa, piemēram, paliek zem 0,1 % robežām, kā arī testēšanas kļūdas funckijas rezultāts nedrīkst sākt pieaugt. Ja testēšanas kļūdas funkcijas rezultāts sāk pieaugt, tad tas var nozīmēt, ka ir notikusi pār-apmācīšanās. Epoha dziļajā mašīnmācīšanā ir viena iterācijā, kuras laikā tiek apmācīti modeļa parametri ar kļūdas funkciju, izmantojot visus apmācību un testēšanas datu paraugus. Katrai kļūdas funkcijai ir nepieciešams veikt pilnās pārlases pārmeklēšanu hiperparametriem, jo vieni un tie paši hiperparametri dažādu formu kļūdas funkcijām neiegūs labākos rezultātus vienlīdzīgi. Bez hiperparametru pārmeklēšanas salīdzināšanas process nebūtu objektīvs.

Pēc tam notiek radošais process, izstrādājot jaunu kļūdas funkciju, kura laikā tiek ņemta vērā matemātikas teorija un citu funkciju īpašības. Tiek analizētas funkciju formas un vēlamās īpašības, balstoties uz ievades parametriem un kļūdas gradientiem. Šo procesu var arī automatizēt, izmantojot funkciju konstruēšanas algoritmus un optimizācijas metodes, piemēram, Beijesa optimizāciju, Monte-Carlo optimizāciju vai ģenētisko algoritmu optimizāciju. Praksē kļūdas funkciju atrašana, izmantojot optimizācijas metodes, ir nepraktiska, jo hiperparametru pilnās pārlases pārmeklēšana katrai kļūdas funkcijai prasa ļoti daudz laika, taču šādā gadījumā būtu nepieciešams pārmeklēt hiperparametrus arī pašai funkciju konstruēšanas metodei. Hiperparameteru pārmeklēšana tika veikta uz RTU *HPC* (*High-Performance Cluster*), kur tā prasīja vairākas nedēļas un pat mēnešus katrai datu kopai, kļūdas funkcijai, apmācību metodei un modeļa arhitektūrai.

Pēc tam, kā jau minēts, tika veikta hiperparametru pārmeklēšana (F) jaunajai kļūdas funkcijai tādā pašā veidā, kā jau eksistējošām kļūdas funkcijām. Ja jaunā kļūdas funkcija ieguva labākus rezultātus nekā eksistējošās kļūdas funkcijas, tad rezultāti tika publicēti zinātniskajā literatūrā (G), taču, ja jaunā kļūdas funkcija nebija labāka, tad bija nepieciešams atgriezties kļūdas funkcijas izstrādes fāzē (E).

Katrai jaunajai kļūdas funkcijas izpētei tika izmantota šāda metodoģija:

1. Līdzīgu kļūdu funkciju, modeļu un apmācību procedūru analīze.

2. Implementācija un testēšana jaunām kandidātu kļūdas funkcijām, modeļiem vai apmācību procedūrām.

3. Jauno un jau esošo kļūdas funkciju, modeļu vai apmācību procedūru

testēšana uz atskaites datu kopām vai vidēm *RL* gadījumā. Katrai datu kopai vai videi tika veikta pilnās pārlases hiperparametru pārmeklēšana.

4. Jauno kļūdas funkciju, modeļu vai apmācību procedūru daļu ietekmes analīze.

RL kļūdas funkcijām tika veikts apskats ar visām mūsdienīgajām kļūdas funkcijām [109]. *DQN* (*Deep Q-Network*) tipa kļūdas funcijas tika testētas vismaz četrās dažādās *PLE* (*PyGame Learning Environment*) datorspēļu vidēs [104], kā arī tika testētas hromatogrāfijas uzdevumam analītiskajā ķīmijā [24].

Vēlāk arī jauna eksponenciālā trijotnes kļūdas funkcija [110] tika izstrādāta un testēta nulles šāviena reidentifikācijas uzdevumiem, izmantojot datu kopas, piemēram, *VGGFace2* [13], *EMNIST* [15] un *CIFAR10* [47]. Datu kopas tika pārkārtotas tā, lai klases, kas ir iekļautas testa sadalījumā, netiktu iekļautas arī apmācību sadalījumā, lai panāktu, ka šie modeļi strādā ar nulles šāviena metodoloģiju [71].

Visbeidzot, VI (vērtību iterācijas) funkcijas modelis tika izveidots, izmantojot *UNet-RNN-Skip* [111] modeli. Tas tika salīdzināts ar dažādiem citiem modeļiem, piemēram, *UNet* [87] modeļu varianti, lai uzlabotu VI algoritma veiktspēju. Tika radīta arī jauna sintētiska datu kopa, lai pārbaudītu *VI* darbību ceļu atrašanai kartē mobilajiem robotiem, un tā var tikt izmantota arī citiem *RL* tipa modeļu uzdevumiem.

Jaunradītās implementācijas ir publiski pieejamas tīmeklī atvērtā tipa programmatūras formātā. To saites ir iekļautas publikācijās. Tās tika implementētas, izmantojot *PyTorch* matemātiskās progamatūras satvaru [79].*PyTorch* tika izvēlēts, jo tas spēj veidot dinamisku funkciju grafu, ko var mainīt apmācību laikā, kā arī ir iespējams ērti atkļūdot un mainīt funkciju gradientus. Katrai no metodēm un kļūdas funkcijām tika veikta hiperparametru pārmeklēšana, izmantojot augstas veiktspējas datorus ar *NvidiaV100* video kartēm RTU *HPC* (*High Performance Cluster*).

1.5. Zinātniskā novitāte un autora ieguldījums

Promocijas darbā iekļautas šādas jaunas kļūdas funkcijas: *MDQN* kļūdas funkcija, ko izmanto *RL* un kas detalizēti aprakstīta 3.3. nodaļā; eksponenciālā trijotnes kļūdas funkcija, ko izmanto *DML*, un tā detalizēti aprakstīta 4.3. nodaļā. Vēl tika radītas jaunas latento vektoru normalizācijas funkcijas *Unit-Range* un *Unit-Bounce*, kas arī aprakstītas 4.3. nodaļā. Līdzīgi tika radīts jauns *UNet-RNN-Skip* modelis, lai uzlabotu vērtību funkcijas veiktspēju, lai izvēlētos aģenta politiku 2D vidē mobilā robota ceļu plānošanai, un tas aprakstīts 2.4. nodaļā. Visbeidzot tika radīts jauns sintētiskas datu kopas ģenerators *OccupancyMapGenerator* un iepriekš sagatavota datu kopa, lai testētu *UNet-RNN-Skip* modeli, un tas tika aprakstīts 2.4. nodaļā.

Promocijas darba autors ir norādīts kā pirmais autors visās publikācijās, kas pievienotas promocijas darba pielikumā, izņemot publikācijā analītiskajā ķīmijā [24], kur promocijas darba autors bija galvenais autors saistībā ar *RL* metodoloģiju uzdevuma risināšanu, izmantojot dziļo mašīnmācīšanos, savukārt pārējie autori veica pētījumu saistībā ar analītisko ķīmiju.

Zinātniskie darbi, kas pievienoti promocijas arbam, publicēti trīs zinātniskajās konferencēs un vienā zinātniskajā monogrāfijā. Darbi ir ieguvuši vairākus apbalvojumus:

1. labākais pētījums konferencē *ICCDA* 2020 (ASV);
2. treša vieta doktorantu pētījumu konkursā *ResearchSlam* 2018 (Latvija).

1.6. Darba struktūra un apjoms

Promocijas darbā kopsavilkumā ir 76. lapas. Tas ir iedalīts septiņās galvenajās nodaļās un veidots kā publikāciju kopa ar paplašinātiem skaidrojumiem, jo lielākā daļa pētījumu ir publicēti zinātniskajās konferencēs, kur ir ierobežots pieļaujamo lapu skaits vienam darbam.

Promocijas darba kopsavilkuma struktūra izklāstīta tekstā tālāk.

1. nodaļa. Ievads, pētījumu pamatojums, pētījumu aktualitāte un pētījumu mērķi.

2. nodaļa. apraksta *VI* algoritma problemātiku, dod teorētisku ievadu esošajās dziļās mašīnmācīšanās modeļu pieejām, piemēram, *ConvNet*, *UNet*, *RNN* un arī apraksta jaunizveidoto *UNet-RNN-Skip* modeli. *UNet-RNN-Skip* tika apmācīts, lai atdarinātu vērtības funkciju un samazinātu iterāciju skaitu, lai iegūtu optimālu politiku. Jaunizveidotais modelis tika izmantots, lai iemācītos jau eksistējošu neparalelizējamu funkciju tā, lai tā strādātu ātrāk un būtu paralelizējama.

3. nodaļa. apraksta Q-vērtības funkcijā balstītu *RL* kļūdas funkcijas un metodes, testa vides un pieejas rezultātu analīzei, kā arī jaunizveidoto *MDQL* kļūdas funkciju, kas tika testēta un analizēta *PLE* vidēs un hromatogrāfijā analītiskās ķīmijas uzdevumā.

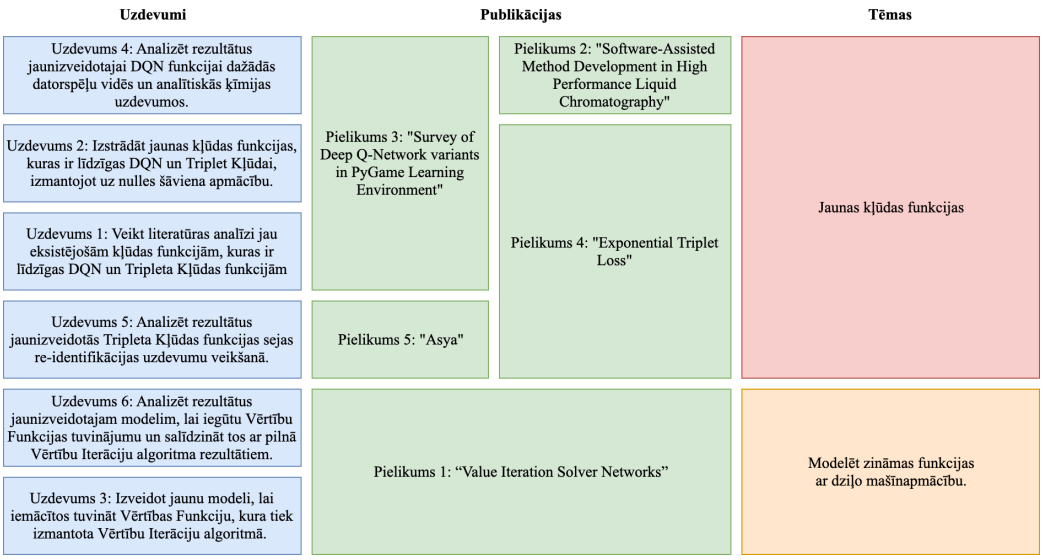
4. nodaļa. skaidro dziļo metriku apmācību, izmantojot nulles šāviena modeļus, kas spēj iegūt latenta vektorus, kas satur semantisko informāciju par ievades datiem. Šie modeļi tika apmācīti ar trijotnes kļūdas funkciju un jaunizveidoto eksponenciālo trijotnes kļūdas funkciju. Tās tika testētas uz vairākām datu kopām reidentifikācijas uzdevuma veikšanai.

5. nodaļa. apraksta eksperimentālos rezultātus jaunizveidotajām kļūdas funkcijām un modeļiem, kas aprakstīti iepriekšējās nodaļās, kā arī apraksta praktiskos lietojumus šīm kļūdas funkcijām un modeļiem.

6. nodaļa. apraksta tālāko pētījumu tēmas, kas tika atrastas, pētot jaunās kļūdas funkcijas un modeļus.

7. nodaļa. apraksta galvenos secinājumus un zinātnisko pienesumu pētījumiem, kas aprakstīti šajā promocijas darbā.

Pilnajā promocijas darbā papildus *DQN* apskatam iekļauts arī zinātniskās literatūras apskats, kurā analīzētas plašāk izmantotākās *DML* kļūdas funkcijas. Pilnajā darbā ir iekļauti pielikumi ar zinātniskajām publikācijām, uz kurām ir balstīts šis darbs. Sasaiste starp promocijas darba mērķiem, tēmām un publikācijām redzama 2. attēlā.



2. att. Darba struktūra un publikāciju saistība ar pētāmajām tēmām un uzdevumiem. Pielikumi pieejami promocijas darba pilnā tekstā angļu valodā.

Publikācijas, kas iekļautas promocijas darbā, un to galvenie jaunievedumi ir aprakstīti tālāk tekstā.

1. **"Value Iteration Solver Networks", International Conference on Intelligent Autonomous Systems, 2020, IEEE, Evalds Urtans, Valters Vecins.** Tika izveidots jauns UNet-RNN-Skip modelis, lai uzlabotu VI algoritma veiktspēju, kā arī tika radīts OccupancyMapGenerator sintētiskas datu kopas ģenerators, lai testētu karšu plānotāju modeļus.

2. **"Software-Assisted Method Development in High Performance Liquid Chromatography"; ISBN: 978-1-78634-545-5, Sep. 2018, Sergey V. Galushko, Irina Shishkina, Evalds Urtans, Oksana Rotkaja.** Tika izveidots jauns RL balstīts modelis, lai secīgi atrastu labākos šķīdinātāju gradientus priekš HPLC.

3. **"Survey of Deep Q-Network variants in PyGame Learning Environment", International Conference on Deep Learning technologies, 2018, ACM, Evalds Urtans, Agris Nikitenko** Tika izveidota jauna RL apmācīšanas metode un jauna MDQN kļūdas funkcija.

4. **"Exponential Triplet Loss", International Conference on Compute and Data Analysis, 2020, IEEE/ACM, Evalds Urtans, Valters Vecins, Agris Nikitenko** Tika izveidota jauna dziļo metriku apmācības kļūdas funkcija ar nosaukumu Eksponenciālā Trijotnes kļūdas funkcija.

5. **"asya: Mindful verbal communication using deep learning", Cornell University, Computing Research Repository, 2020, Evalds Urtans, Austris Tabaks** Tika izveidota jauna sistēma, izmantojot Eksponenciālo Trijotnes kļūdas funkciju, lai veiktu balss reidentifikācijas uzdevumu.

2. FUNKCIJU VEIKTSPĒJAS UZLABOŠANA DZIĻĀ MAŠĪNMĀCĪŠANĀ

Šī nodaļa iepazīstina ar jaunu dziļās mašīnmācīšanās paņēmienu, lai optimizētu klasiskās funkcijas veiktspēju, kas jau ir daudz pētīta un tiek plaši izmantota, bet ir ļoti lēna. Tika pētīta vērtību funkcija, ko izmanto vērtību iterāciju (*VI*) algoritms. Tā paredzēta, lai atrastu īsāko ceļu aizņemtības režģa kartēs no jebkuras pozīcijas tajā līdz mērķa pozīcijai.

Tās veiktspēja sarūk eksponenciāli, palielinot kartes izmēru, kā arī tā nav izpildāma paralēli. Savukārt jaunais iteratīvais modelis, kas tiek piedāvāts šajā darbā, var iegūt līdzvērtīgus rezultātus kā *VI* algoritms daudz ātrāk, jo to var izpildīt paralēli. Ievērojami labākus rezultātus var iegūt, izmantojot lielāka izmēra kartes.

Vērtību funkcijas un vērtību iterācijas algoritma problēmsfēra ir aprakstīta 2.1. nodaļā. Dziļās mašīnmācīšanās metodes, ko var izmantot, lai modelētu vērtību funkciju, ir aprakstītas 2.2. un 2.3. nodaļā. Visbeidzot, jaunizveidotā metode ir aprakstīta 2.4. nodaļā, tās rezultāti — 5.1. nodaļā.

2.1. Vērtību iterāciju algoritms

Vērtību iterāciju algoritms (*VI*) tiek lietots klasiskās stimulētās mašīnmācīšanās uzdevumos, lai atrastu optimālu politiku vidē, kuras stāvokļi ir pilnībā novērojami. Tas var ņemt vērā arī stāvokļu pārejas modeli, kur šīs pārejas var būt nedroši nosakāmas [88]. VI nereti tiek izmantots, lai atrastu optimālu ceļu kartēs ar diskrētiem stāvokļiem. Ceļu meklēšanas uzdevumā tiek formalizēta un diskretizēta dabiska vide un tās šķēršļi. Nereti šos datus iegūst no sensoriem, kas ir uzstādīti uz mobilas robota platformas. Sensori parasti var būt *LIDAR* sensori attāluma noteikšanai, izmantojot gaismu, ultraskaņas attāluma sensori, *IMU* akselometra un žiroskopa sensori pārvietojuma noteikšanai, magnetometrs, *GPS* u. c. sensori. Diskretizācija kartēm parasti tiek veikta, izmantojot aizņemtības režģa reprezentāciju. *VI* ir iteratīvs algoritms, kas atkārtoti lieto vērtību funkciju 3. vienādojumā visām pozīcijām kartē, lai atrastu kumulatīvo pozīcijas vērtību, kā parādīts 3. attēlā.

Tālāk, vērtību gradients starp šīm pozīcijām norāda uz optimālo ceļu un stāvokļu pāreju politiku. Optimāla ceļa politika nodrošina, ka aģents var atrast ceļu no jebkura stāvokļa diskretizētā kartē līdz pozitīvam terminālajam

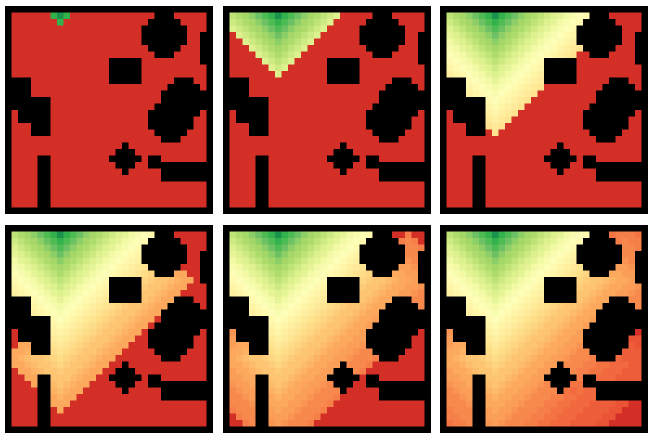
stāvoklim. Katram stāvoklis s tiek aprakstīts ar režģa pozīcijām. Katrā stāvoklī ir iespējams veikt darbības a, piemēram, pārvietoties augšup, lejup, pa kreisi, pa labi vai palikt nekustīgam iepriekšējā stāvoklī. Atkarībā no vides darbības var būt vairāk vai citādas.

Darbība a ir jāizvēlas tā, lai maksimizētu summēto balvas vērtību šai darbībai R_a. Pēc tam tā tiek reizināta ar pārvietojuma modeļa varbūtību P_a un tiek pieskaitītas blakus esošo stāvokļu vērtības $V(s')$, kas tiek reizinātas ar atlaides koeficientu γ, kā parādīts 3. vienādojumā.

Vērtību funkcija tiek izsaukta iteratīvi visām kartes pozīcijām līdz vērtības konverģē starp iterācijām. Iterāciju skaits pieaug eksponenciāli, palielinoties kartes izmēram. Konverģence *VI* algoritmam notiek, kad izpildās 2. vienādojums, kur δ ir vidējā skaitliskā starpība starp iterācijām, kurai var būt robežas no 0,1 — 0,001 atkarībā no vides, stāvokļu novērojumu tīrības un pieļaujamās kļūdas.

$$|\frac{\overline{V(s)_i}}{\overline{V(s)_{i-1}}}| < \delta \tag{2}$$

$$V_{i+1}(s) := \max_a \{\sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V_i(s'))\} \tag{3}$$



3. att. Secīgu *VI* algoritma iterāciju vizualizācija. Iterācijas ir attēlotas secīgi, sākot no augšējā kreisā stūra līdz augšējam labajam stūrim, tad no apakšējā kreisā stūra līdz apakšējam labajam stūrim. Kad ir sasniegta konverģence, tiek iegūta optimāla politika tā, lai no katra stāvokļa kartē var iegūt augstāko kumulatīvo balvas vērtību, nonākot pozitīvā terminālajā stāvoklī. Zaļā krāsā ir augstākās pozīciju vērtības (tuvāk pozitīvajam terminālajam stāvoklim), sarkanā krāsā zemāko vērtību pozīcijas.

Pamatojums tam, lai modelētu vērtību funkciju, izmantojot dziļo mašīnmācīšanos, ir tajā, ka šādi ir iespējams paātrināt šo algoritmu, to paralēlizējot un samazinot tā secīgo operāciju skaitu (1. tabula). Eksistē arī citi populāri algoritmi, kas lieto heiristikas, lai samazinātu secīgo operāciju skaitu, piemēram, *Dijkstra* vai $A*$ algoritmi [88], bet, tos lietojot, samazinās arī rezultāta precizitāte.

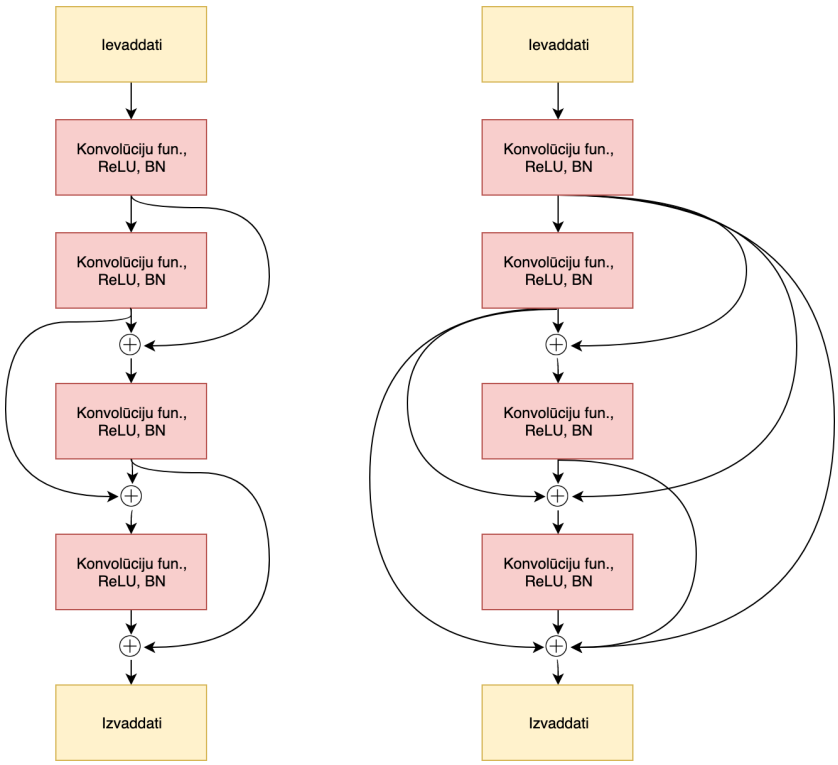
1. tabula.

Jaunās *VI* funkcijas modeļa *VSIN* (*Value Itereation Solver Network*) salīdzinājums ar klasiskajām funkcijām.

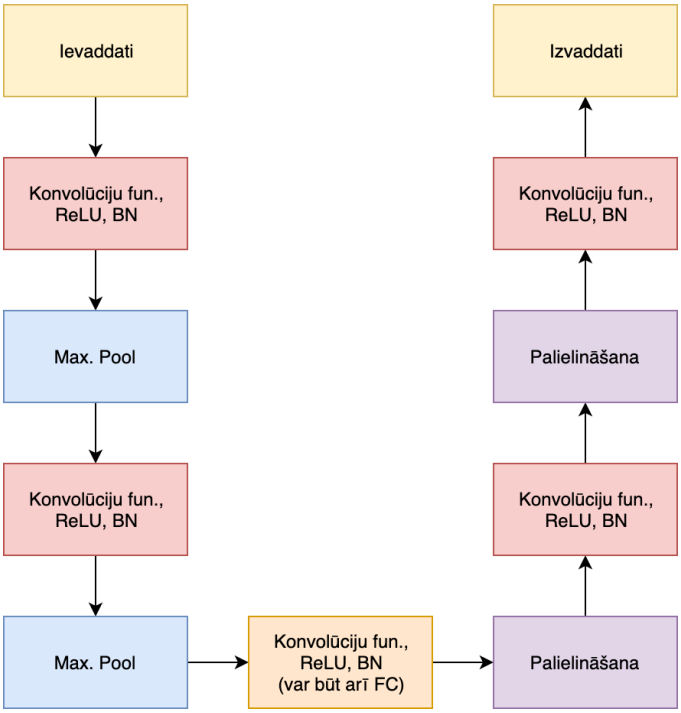
Metode	**Sarežģītība**	**Kvalitāte**
VI (vērtību funkcija)	$O(m \cdot n^2)$	Optimāla
Dijkstra	$O(n^2)$	Laba
A*	$O(h \cdot n)$	Vidēja
VISN (jaunievestā metode)	$O(h \cdot n)$	Laba

2.2. ConvNet un UNet modeļi

Lai palielinātu *VI* algoritma veiktspēju, šī funkcija var tikt modelēta, izmantojot ConvNet (konvolūciju tīkla) modeli [56], [48], [97], [102], [31], [38], kas tiek apmācīts, lai prognozētu skaitliskās vērtības $V(s)$ funkcijai. Pats modelis var tik apmācīts vērtību funkcijas rezultāta tuvinājuma aprēķināšanai. *ConvNet* modelis strādā kā filtrs visai aizņemtības režģa kartei un prognozē skaitliskās vērtības *VI* algoritma rezultātiem bez vajadzības pēc iteratīva procesa. *ConvNet* modeļi var tikt paralelizēti uz mūsdienīgiem *GPU*, lietojot dziļās mašīnmācīšanās satvarus. Šajā pētījumā tika izmantoti *ResNet* [31] un *DenseNet* [38] tipa *ConvNet* modeļi kā iezīmju kodētāji vērību funkcijai. ResNet un DenseNet modeļi, kas tika izmantoti kā iekodētāji, shematiski redzami 4. attēlā. Modeļa iezīmju kodētāks kondensē augstu dimensiju ievades datus uz zemu dimensiju latentajiem vektoriem, ko tālāk var izmantot dziļākās modeļa daļās. Šajā pētījumā kā dekodera modelis tika izmantotas transponētas konvolūciju funkcijas [74]. Dekodera daļa no modeļa daļa rekonstruē zemu dimensiju latentos vektorus par augstas dimensijas rezultātiem, kuri tiek uzklāti kā filtrs ievades datiem. Uzklāšana var notikt, izmantojot vienkāršu aritmētisku saskaitīšanu, reizināšanu vai aizvietošanu. Autokodētaja modelis, kas satur dekoderu redzams 5. attēlā.



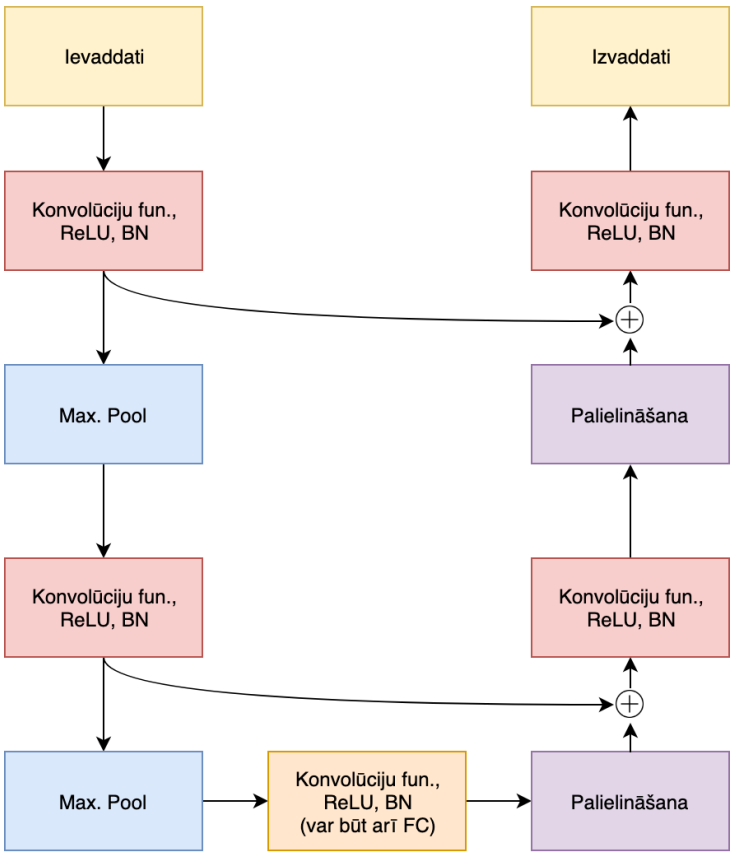
4. att. Iekodētāju modeļu salīdzinājums (*ResNet* — kreisajā pusē, *DenseNet* — labajā pusē). Plus zīme norāda uz aritmētisko saskaitīšanas operāciju.



5. att. Autokodētāja modeļa piemērs. Funkciju grafa vidū (neobligāti) var tikt ievietota lineārās transformācijas funkcija *FC* (*Fully Connected*), ja modelim ir pieļaujami fiksēta izmēra ievades un izvades dati.

VI algoritma darbība ir līdzīga daudzkārtējam filtrēšanas uzdevumam, jo pati ievades datu struktūra netiek izmainīta, bet katrā iterācijā tiek pamainītas tikai nelielas detaļas.

Filtrēšanas un stila pārneses un pat segmentēšanas uzdevumiem nereti tiek izmantots *UNet* tipa modelis [87], jo tas satur lēcienu savienojumus starp funkcijām, kas uztur detalizējumu oriģinālajiem datiem dažādas iekodētāja dimensijās atkarībā no modeļa dziļuma. Šajā pētījumā *UNet* modeļi tika trenēti, lai iegūtu *VI* algoritma rezultātus ar vairākām iterācijām, kā arī vienā piegājienā.



6. att. *UNet* modeļa piemērs, kurā redzams, kā tajā tiek izmantoti lēciena savienojumi, kur datu īpašības tiek pieskatītas no iekodētāja puses uz dekodētāja pusi funkciju grafā.

Oriģinālais *UNet* modelis (6. attēls) satur funkciju grafu, tā vienādojumi doti zemāk 4. vienādojumā. Vienādojumos ievades dati x tiek padoti caur vairākām iekodētāja funkcijām, kur *Conv* funkcija ir lineāra 2D konvolūciju funkcija ar kodola izmēru 3x3, soli 1 un aplodu 1, un kuras rezultāts būs tāda paša izmēra kā ievades dati. Šī funkcija nemaina aizņemtības režģa izmērus. Savukārt *DeConv* ir transponēta 2D konvolūciju funkcija kodola izmēru 4x4, soli 2 un aplodu 1, kas dos divas reizes lielāku izvades karti nekā ievades datu karti. Līdzīgi arī *MaxPool* funkcijas rezultāts ar šiem pašiem parametriem dos divas reizes mazāku izvades karti nekā ievades karti, tikai katrā no kodola apgabaliem tiks izvēlēta maksimālā vērtība no vienas no režģa šūnām. Lēciena savienojumi (10. un 13. vienādojums) izmanto apvienošanas operā-

ciju, savukārt segmentācijas uzdevumiem bieži tiek lietota arī saskaitīšanas operācija, kā tas ir *ResNet* modeļos [102]. Gala rezultāta skaitliskās vērtības y tiek ierobežots ar sigmoida funkciju σ, kur katra vērtība tiek mērogota pret minimālo maksimālo kumulatīvo balvas vērtību kartē.

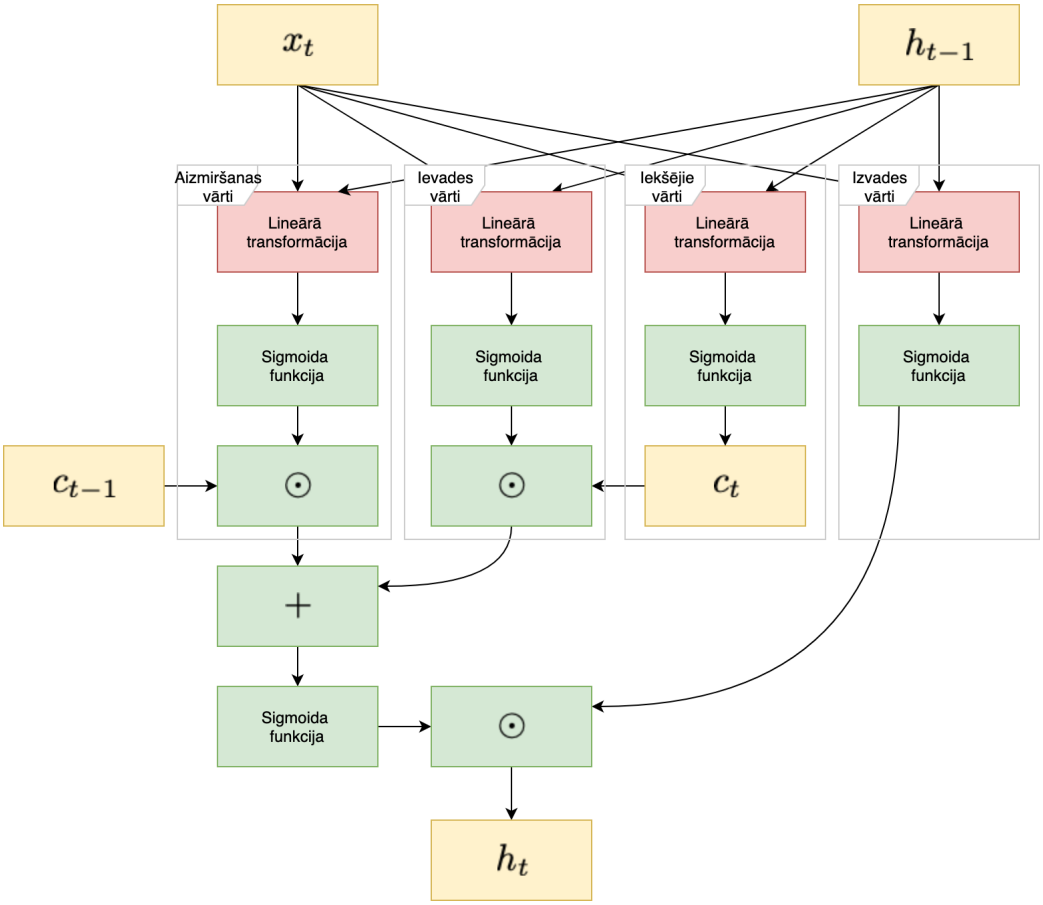
$$o_1 = ReLU(Conv(x)) \tag{4}$$
$$o_2 = MaxPool(o_1) \tag{5}$$
$$o_3 = ReLU(Conv(o_2)) \tag{6}$$
$$o_4 = MaxPool(o_3) \tag{7}$$
$$o_5 = ReLU(Conv(o_4)) \tag{8}$$
$$o_6 = DeConv(o_5) \tag{9}$$
$$o_7 = (o_6, o_3) \tag{10}$$
$$o_8 = Dropout(ReLU(Conv(o_7))) \tag{11}$$
$$o_9 = DeConv(o_8) \tag{12}$$
$$o_10 = (o_9, o_1) \tag{13}$$
$$o_11 = ReLU(Conv(o_10)) \tag{14}$$
$$y = \sigma(o_11) \tag{15}$$

2.3. RNN modeļi

Lai modelētu iteratīvus procesus un laika rindu tipa datus, nereti tiek izmantoti *RNN* (rekurento neironu tīklu) modeļi. *VI* algoritma gadījumā šie modeļi tiek lietoti, lai samazinātu filtra uzdevuma sarežģītību, kad ir nepieciešams ar vienu *ConvNet* iterāciju prognozēt $V(s)$ vērtības, neko nezinot par iepriekšējām vērtībām. Ar *RNN* tipa modeli vērtības tiek prognozētas secīgi ar iteratīviem soļiem līdzīgi tam, kā to dara *VI* algoritms, bet RNN modelis nodrošina konverģenci ar daudz mazāku iterāciju skaitu.

Šajā pētījumā tika izmantoti populāri *RNN* modeļi, kā *LSTM* (ilgtermiņa — īstermiņa atmiņas šūna) [36] un *GRU* (rekurentā šūna ar vārtiem) [28].

Lai uzlabotu veiktspēju un konverģences ātrumu, tika lietotas parametru inicializācijas stratēģijas. Piemēram, nobīdes parametri aizmiršanas vārtiem *LSTM* modelī tika inicializēti ar skaitlisko vērtību 1, lai mazāk informācijas tiktu aizmirstas apmācību sākumā. Līdzīgi arī parametru nobīdes vektori *GRU* modelim tika inicializēti ar skaitlisko vērtību -1, lai panāktu to pašu efektu [41], [54], [25], [106].



7. att. *LSTM* modeļa funkciju grafs.

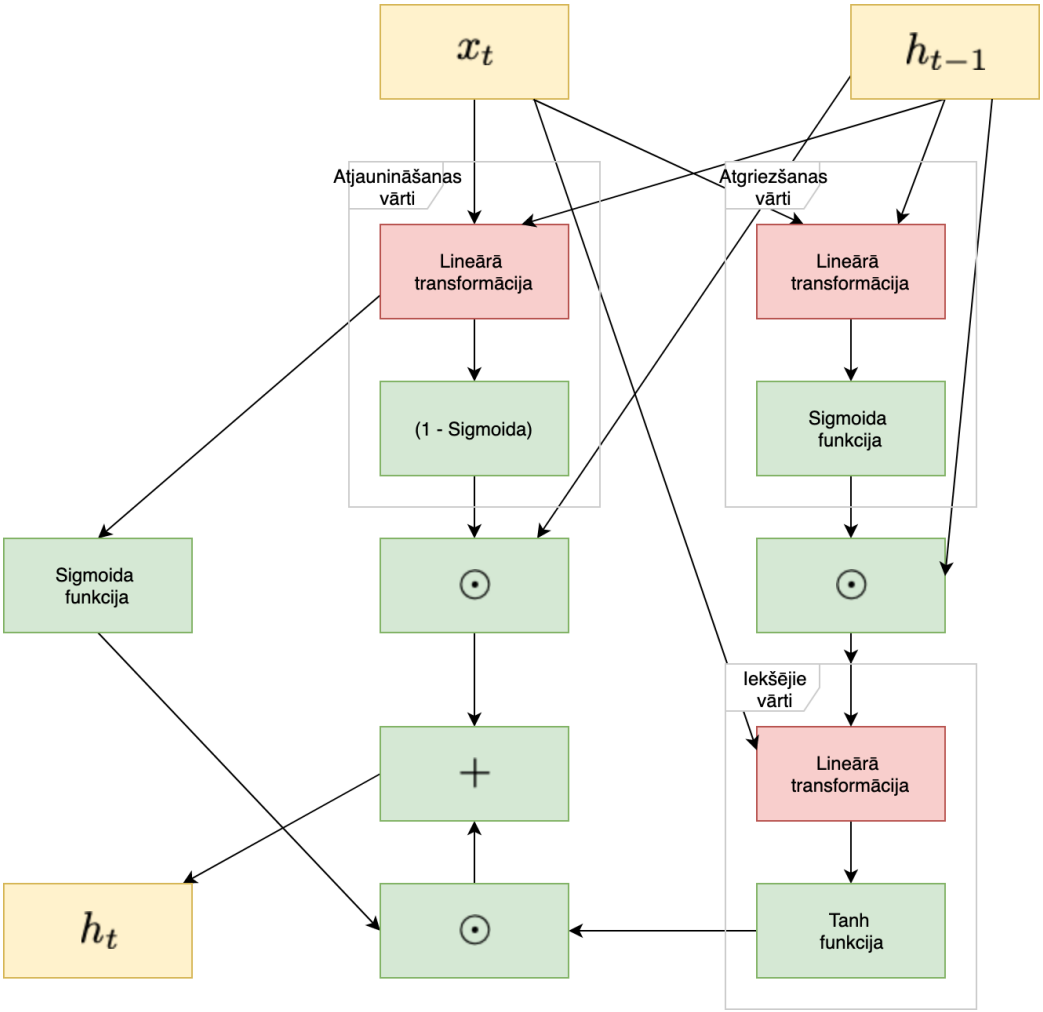
LSTM modeļa grafs ir redzams 16. vienādojumā un shematiski parādīts 7. attēlā. Apmācāmie svari un nobīdes ir apzīmēti ar W, U un b, sigmoida funkcijā σ balstītie vārti ir f_t aizmiršanas vārti, i_t ievades vārti un o_t datu izvades vārti. Iekšējā stāvokļa vektors, kas nav apmācāms ir c_t un neapmācāms slēptā stāvokļa vektors h_t, kuri apmācību sākumā tiek uzstādīti ar nulles skaitliskām vērtībām, bet ar katru sekvences laika soļi jeb iterāciju tiek izmainīti ar informāciju no iepriekšējiem laika soļiem. Katrā laika solī *LSTM* modeļa rezultāti sakrīt ar pašreizējo h_t vērtību. Pēc jaunākajiem pētījumiem b_f jāinicializē ar vērtību $b_f = 1$ [106], tāpat jāpievieno *RNN* caurkrišana (*Dropout*) vai *ZoneOut* regularizācija added [49], [94], h_0 un c_0 var tikt apmācīti arī jau no pirmās iterācijas [75]. Slāņa Normalizācijas funkcija jālieto pēc katras lineārās transformācijas *LSTM* modelī [3], un, visbeidzot,

jāpievieno lēciena savienojumi starp *LSTM* modeļu slāņiem, tos summējot $h_t = \sum_{k=i}^{layers} h_t^k$ līdzīgā veidā kā to dara *ResNet* modelī, lai vienmērīgi izplatītos kļūdas gradients dziļos mākslīgos neironu tīklos [102], [38], [106].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{16}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{17}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{18}$$
$$\tilde{c}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \tag{19}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{20}$$
$$h_t = o_t \odot \sigma_h(c_t) \tag{21}$$

GRU modelis, kā redzams 22. vienādojumā un 8. attēlā ir *LSTM* modeļa vienkāršots variants ar mazāk apmācāmiem parametriem. Tas satur tikai z_t atjaunināšanas vārtus un r_t atgriezšanas vārtus. Tāpat tas lieto ne tikai sigmoida funkcijas, bet arī tanh funkcijas ϕ līdzīgi kā tradicionālie vienkāršie *RNN*. Lai iegūtu augstāko veiktspēju ar šiem modeļiem, nepieciešams lietot visas tās pašas izmaiņas, kādas aprakstītas saistībā ar *LSTM* modeļiem augstāk, izņemot to, ka r_t vārtu nobīdi jāinicializē ar skaitlisko vērtību $b_r = -1$ [106].

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{22}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{23}$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{24}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{25}$$



8. att. *GRU* modeļa funkciju grafs.

2.4. UNet-RNN-Skip modelis

Daļu promocijas darba veido jaunizveidotais modelis, kas apvieno *UNet* un *RNN* modeļus vienā. Jaunajam modelis *UNet-RNN-Skip* ir vairākas tādu modeļu īpašības, ko izmanto citām vajadzībām, lai nodrošinātu jaunā modeļa spēju sasniegt ātrāku konverģenci VI algoritma rezultāta iegūšanai. Tas satur *UNet*, *LSTM* un lēciena savienojumus līdzīgi kā *ResNet*, taču tie tiek izmantoti laika rindu un segmentācijas uzdevumiem.

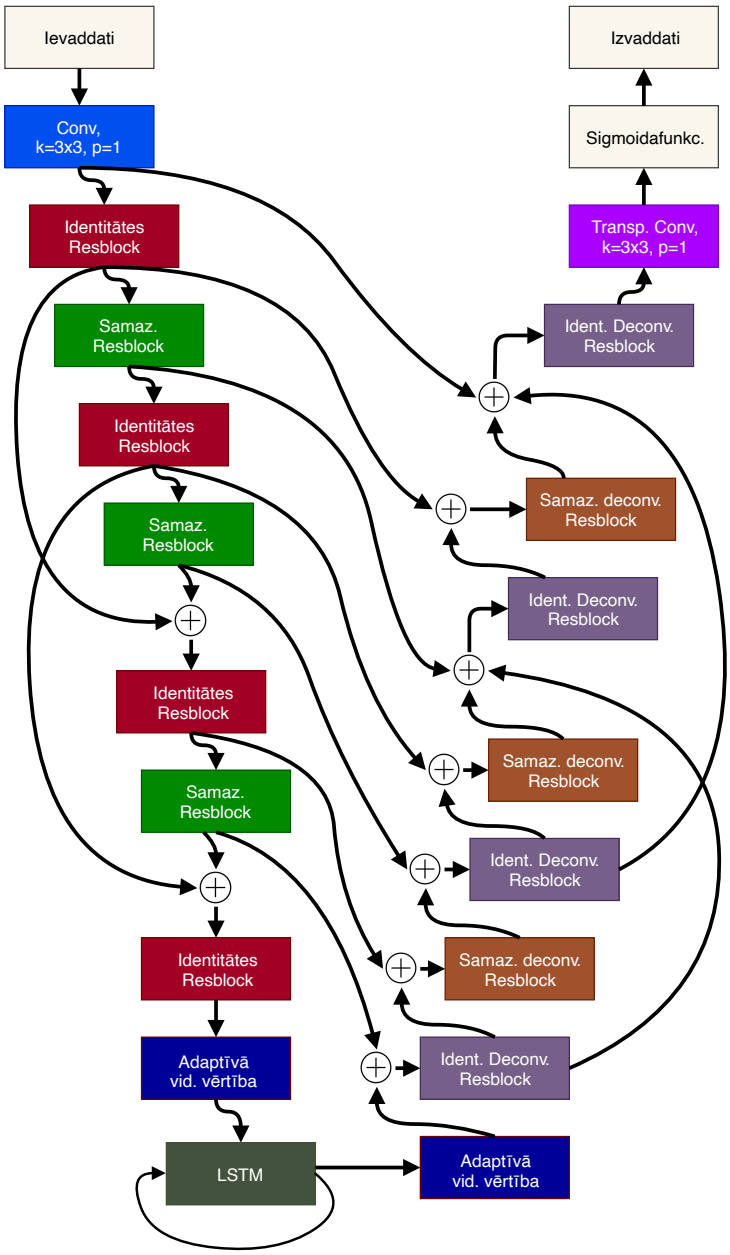
UNet-RNN-Skip modelis tika radīts, lai iegūtu tuvinātas skaitliskās vēr-

tības funkcijai, ko izmanto VI algoritmā, bet tā, lai to varētu izpildīt paralēli pilnīgi visām aizņemtības režģa šūnām. Savukārt VI algoritmam nepieciešams šīs vērtības aprēķināt secīgos soļos, apskatot katru šūnu aizņemtības režģī. Jaunais modelis tika radīts arī tāpēc, lai aprēķinātu ar vienu iterāciju tuvinātu skaitlisko vērtību, ko VI algoritms aprēķina ar vairākām iterācijām.

Vēl viens *UNet-RNN-Skip* modeļa ieguvums ir tas, ka to var apmācīt uz daudz mazāka izmēra kartēm, piemēram, 32x32, un vēlāk produkcijā to var izmantot uz lielām kartēm, piemēram, 256x256 bez nepieciešamības to apmācīt papildus, jo modelis ir iemācījies vērtību funkciju, nevis sakarības pašā kartē.

Visas kartes tika ģenerētas, izmantojot jaunu *OccupancyMapGenerator* algoritmu, kas arī bija daļa no šī pētījuma un ir aprakstīts 5.1. nodaļā. Turpat aprakstīti arī eksperimentālie rezultāti un modeļa īpašības.

UNet-RNN-Skip modelī svarīga detaļa ir *LSTM* novietojums "U formas" pašā apakšā funkciju grafā, bet tajā pašā laikā modelis izmanto arī *UNet* lēciena savienojumu funkcijas no kodētāja uz dekodētāja daļām (līdzīgi kā *ResNet* [31], *UNet++* [136] un *UNet 3+* [39] modeļos). Pretēji parastajam *UNet*, kas izmanto īpašību apvienošanu, paplašinot matricu izmērus (6. attēls), jaunais modelis izmanto aritmētisko saskaitīšanu lēciena funkcijās. *ResNet* [31] funkciju bloki tika izveidoti divos dažādos veidos (9. attēls).



9. att. *UNet-RNN-Skip* modelis. Ar krāsām iezīmēti dažādi funkciju bloki, kas izmantoti šajā modelī.

Vispirms tika izveidots identitātes *ResBlock* funkcija, kā redzams 26. vien-

ādojumā.

$$y_{c\times w\times h} = ReLU(Conv3x3(x_{c\times w\times h})) \tag{26}$$
$$y'_{c\times w\times h} = BatchNorm(y_{c\times w\times h}) \tag{27}$$
$$z_{c\times w\times h} = ReLU(Conv3x3(y'_{c\times w\times h})) \tag{28}$$
$$z'_{c\times w\times h} = z_{c\times w\times h} + x_{c\times w\times h} \tag{29}$$
$$u_{c\times w\times h} = ReLU(z'_{c\times w\times h}) \tag{30}$$
$$u'_{c\times w\times h} = BatchNorm(u_{c\times w\times h}) \tag{31}$$

Pēc tam tika izveidota *BottleNeck* funkcija, kas samazina iezīmju karšu izmērus ar konvolūciju funkcijas palīdzību, bet tajā pašā laikā arī palielina iezīmju skaitu karšu kanālu dimensijā, kā redzams 32. vienādojumā. *ResBlock* funkcija ir līdzīga pirms-aktivizācijas *ResBlock* funkcijai [32], kur partiju normalizācija tiek izpildīta pirms lineārajām transformācijām. Tādas pašas funkcijas spoguļskatā tika izmantotas arī transponēto konvolūciju funkciju daļā, kas palielina karšu izmērus, taču samazina kanālu iezīmju skaitu.

$$y_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = ReLU(Conv3x3(x_{c\times w\times h})) \tag{32}$$
$$y'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = BatchNorm(y_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}}) \tag{33}$$
$$z_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = ReLU(Conv3x3(y'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}})) \tag{34}$$
$$x'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = Conv1x1(x_{c\times w\times h}) \tag{35}$$
$$z'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = z_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} + x'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} \tag{36}$$
$$u_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = ReLU(z'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}}) \tag{37}$$
$$u'_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}} = BatchNorm(u_{n\cdot c\times \frac{w}{n}\times \frac{h}{n}}) \tag{38}$$

3. FUNKCIJU FORMĒŠANA DZIĻAJĀ STIMULĒTAJĀ MAŠĪNMĀCĪŠANĀ

Šī nodaļa iepazīstina ar jaunu kļūdas funkciju un apmācību procedūru dziļajai Q-vērtību funkcijas apmācībai. Jaunizveidotā *MDQN* kļūdas funkcija izmanto dinamisku funkcijas grafu, kas maina kļūdas funkcijas formu apmācības procesa laikā. Tika veikta arī citu līdzīgu funkciju analīze. Jaunizveidotās procedūras iekļauj arī metodi, kas ļauj vizualizēt stāvokļu vērtības un politiku, lai "melnās kastes modeļus" padarītu par interpretējamiem "baltās kastes modeļiem".

Dziļās stimulētās mašīnmācīšanās problēmsfēra aprakstīta 4.1. nodaļā Jau pazīstamas *DQN* kļūdas funkcijas un to paveidi aprakstīti 3.2. nodaļā. Visbeidzot, jaunizveidotā kļūdas funkcija aprakstīta 3.3. nodaļā, un tās rezultāti doti 5.2. nodaļā.

3.1. Q-Vērtību funkcija un politikas gradientu funkcijas

Stimulētās mašīnmācīšanās algoritmi jau bija labi izpētīti pat pirms dziļās mašīnmācīšanās metodēm, taču līdz ar jaunajām metodēm to kapacitāte ir ievērojami palielinājusies [68]. Līdzīgi, kā iepriekšējās nodaļās aprakstītais VI algoritms, arī šie modeļi aprēķina labāko politiku π balstoties uz stāvokļu kumulatīvajām vērtībām r_t un reizēm izmantojot tikai pašu stāvokli s_t.

Stimulētajā mašīnmācīšanā eksistē trīs galvenās metožu kopas.

1. Q-vērtību funkcijā balstītās metodes.
2. Politikas gradienta metodes.
3. Aktiera-kritiķa metodes (Q-vērtību funkcijas un politikas gradienta metožu apvienojums).

Šīs metodes savā starpā atšķiras galvenokārt ar kļūdas funkcijas formu, kas ir viena no svarīgākajām sastāvdaļām, lai šīs metodes spētu strādāt.

Politikas gradienta metodes strādā, pamatojoties uz politikas gradienta teorēmu (39. vienādojums), kur kļūdas gradients politikas funkcijai π ir novērtējuma vērtība iepriekšējo stāvokļu trajektorijai τ, pašreizējam stāvoklim s_t un katrai pieļaujamai darbībai šajā stāvoklī a_t, reizinot ar kumulatīvo balvas vērtību R (40. vienādojums) trajektorijai τ. Šī gradienta funkcija

ir nestabila un grūti konverģējama, taču nesenie pētījumi to ir padarījuši daudz stabilāku un praktiski izmantojamu, kā piemēram kļūdas funkcijas TRPO [92], PPO [91], MERLIN [125], IMPALA [21], u.c. kļūdas funkcijas.

$$\nabla \pi_\theta(a_t|s_t) = \nabla log \pi_\theta(a_t|s_t) R(\tau) \tag{39}$$

Kumulatīvā balvas vērtība R (40. vienādojums) ir summēta balvas vērtība stāvokļu trajektoriju balvu vērtības, kur tās katrā laika solī t tiek reizinātas ar atlaides vērtību γ. Atlaides vērtība γ nosaka, cik ļoti nozīmīgas ir ātri iegūstamas balvas pret balvām, ko var iegūt epizodes beigās. Jo mazāka ir šī skaitliskā vērtība, jo vairāk modelis liek uzsvaru uz ātri iegūstamām balvām. Dažās vidēs, kur aģents var viegli nokļūt negatīvā terminālā stāvoklī, šī vērtība būtu jāuzstāda zemāka. Parasti šo vērtību uzstāda kā hiperparametru vērtību robežās 0.9 — 0.99,jo nozīmīgākie notikumi datorspēļu vidēs parasti notiek pēc ilgāka laika un pie tiem jānonāk ar mērķtiecīgu plānošanu.

$$R = \sum_{t=0}^{n} \gamma^t r_t \tag{40}$$

Otrā metožu kopa sakņojas Bellmana vienādojumā (41. vienādojums), kas tuvina R vērtību katrā stāvoklī s_t pie darbības a_t, to modelējot ar Q-vērtību funkcijas modeli. Secīgi apskatot darbības a_t un izvēloties to, kas sniedz augstāko Q-vērību, ir iespējams noteikt politiku π.

$$Q_\pi(s_t, a_t) = r_t + \max_{a'} Q_\pi(s_{t+1}, a') \tag{41}$$

Trešā metožu kopa ir aktiera-kritiķa metodes, kas sakņojas abās iepriekšējās metodēs. Tās galvenokārt apvieno abas iepriekšējās kļūdas funkcijas, izmantojot jauktu kļūdas funkciju ar vairākām daļām, starp kurām ir koeficienti. Ir izstrādāti daudzi veiksmīgi šādu metožu varianti, piemēram, *DDPG* [60], *A3C* [67], *GPU A3C* [4], *ACER* (*Actor-Critic with Experience Replay*) [124] u.c. metodes.

Lai arī šķiet, ka labākos rezultātus vienmēr var iegūt, izmantojot kombinētās kļūdas funkcijas, tomēr labākos rezultātus nereti izdodas iegūt tieši ar katru kļūdas funkciju atsevišķi, taču tas ir atkarīgs arī no vides īpatnībām.

3.2. *DQN* kļūdas funkcija

Viens no veiksmīgākajiem dziļās stimulētās mašīnmācīšanās modeļiem ir *DQN* (*Deep Q-Network*). Pēc sākotnējiem un veiksmīgajiem *DQN* [68]

rezultātiem tika radīta virkne citu Q-vērtības funkcijā sakņotu modeļu uzlabojumi [2].

Sākotnējais *DQN* tika testēts, izmantojot *Atari* spēles, kurām ir daudzdimensiju stāvokļu reprezentācijas - operatīvās atmiņas apgabala matrica vai spēles attēla pikseļu matrica katram spēles stāvoklim (10. attēls). Vēl cita bieži izmantota stimulētās mašīnmācīšanās vide ir *OpenAI Gym* [6], kurā tiek izmantotas maza skaita dimensijas stāvokļu reprezentācijai, piemēram, MoonLander vidē tiek dota informācija par kosmosa kuģa atrašanās vietu, ātrumu un leņķi attiecībā pret mērķa pozīciju uz Mēness, kur tam ir jānolaižas. Šajā pētījumā tika izmantotas *PLE* (*PyGame Learning Environment*) vides, kas satur gan daudzdimensiju vides, gan vides ar mazāku skaitu dimensiju, un ir iespējams pat izmainīt vides uzvedību, kamēr modelis uz tās tiek apmācīts, lai būtu iespējams dziļāk izpētīt modeļa īpašības.

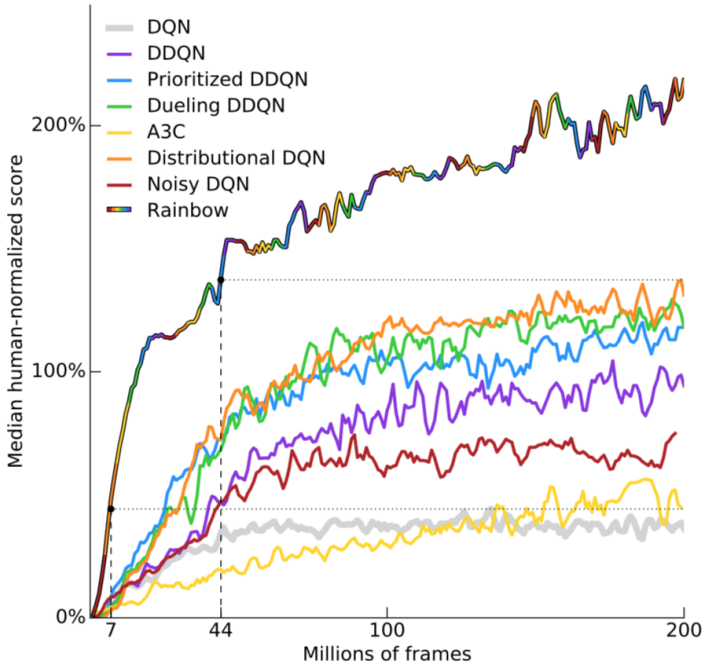
Pēdējos gados *DQN* modelim ir veiktas dažas mazas, bet nozīmīgas izmaiņas, kas ir devušas ievērojamus uzlabojumus tā veiktspējā. Viens no svarīgākajiem uzlabojumiem ir prioritārā pieredzes atmiņa [90], kas nodrošina to, ka apmācībai tiek izvēlētas stāvokļu pārejas ar lielāko temporālo kļūdu. Vēl viens nozīmīgs uzlabojums ir dueļa *DQN* [123], kas ar modeļa arhitektūras palīdzību tam liek iemācīties atpazīt ātri iegūstamās balvas un ilgtermiņa balvas, kas nereti ir daudz nozīmīgākas. Nākamais uzlabojums ir *DDQN* (dubultais *DQN*) [29], [30], kas nodrošina stabilāku *DQN* kļūdas funkcijas konverģenci (42. vienādojums), izmantojot jaunu kļūdas *DDQN* kļūdas funkciju (43. vienādojums) *DDQN* kļūdas funkcijas gadījumā apmācības laikā Q_{target} modeļa svari tiek kopēti un sasaldēti no Q_Θ ik pēc katra, iepriekš definēta skaita laika soļu.

$$\mathcal{L}_{dqn} = \begin{cases} (r_t + \gamma \max_{a'} Q_\Theta(s_{t+1}, a') - Q_\Theta(s_t, a_t))^2 & if\ t < t_{last} \\ (r_t - Q_\Theta(s_t, a_t))^2 & if\ t = t_{last} \end{cases} \tag{42}$$

$$\mathcal{L}_{ddqn} = \begin{cases} (r_t + \gamma \max_{a'} Q_{target}(s_{t+1}, a') - Q_\Theta(s_t, a_t))^2 & if\ t < t_{last} \\ (r_t - Q_\Theta(s_t, a_t))^2 & if\ t = t_{last} \end{cases} \tag{43}$$

10. att. *Atari* spēles, kas tika izmantotas, lai pārbaudītu *DQN* modeli [68].

Visbeidzot varavīksnes-*DQN* modelis parāda, ka apvienojot visas nelielās izmaiņas kopā vienā modelī, tās pārspēj katru no izmaiņām atsevišķi [35], kā redzams 11. attēlā, kurā dots salīdzinājums normalizētiem *Atari* spēļu rezultātiem, izmantojot dažādas kļūdas funkcijas un metodes atkarībā no miljoniem kadru, kas bija nepieciešami, lai apmācītu aģentu. Varavīksnes-*DQN* modelis izmanto *DDQN* kļūdas funkciju kā arī iepriekš minētās un vēl citas izmaiņas.



11. att. Varavīksnes-*DQN* modeļa un tā daļu salīdzinājums, izmantojot normalizētu Atari balvas vērtību [35].

3.3. MDQN kļūdas funkcija

Promocijas darbā aprakstīta jauna *MDQN* (*Multi Deep Q-Network*) kļūdas funkcija, kas ir dinamiska un maina savas īpašības apmācības laikā. Līdzīgi kā *DDQN*, tā satur mērķa modeļus, kuri tiek atjaunināti ik pēc noteikta skaita laika soļu, taču atšķirībā no *DDQN* kļūdas funkcijā tiek izmantota vairāk nekā viena kopija *DQN* svaru. Divas, trīs vai vairāk kopijas svaru var izmantot, lai samazinātu neseno un nejaušu notikumu ietekmi uz apmācības procesu. Šīs svaru kopijas tiek mainītas savā starpā balstoties uz dažādiem soļu skaitiem, tādējādi liekot modelim atcerēties dažādus notikumus katrā no *DQN* modeļu kopijām, lai pilnvērtīgi veidotu galveno *DQN* modeli. *MDQN* kļūdas funkcija pseidokodā ir aprakstīta 1. vienādojumā.

Atkarībā no *MDQN* varianta vairāki *DQN* modeļu svari $Q_1, Q_2, ..., Q_n$ tiek inicializēti apmācības sākumā. Piemēram, *MDQN-3* satur trīs paralē-

lus Q modeļus, pēc tam sākotnējie Q_a un Q_b tiek nejauši izvēlēti no *DQN* modeļu kopas. Kadru skaitītāji $c_a, c_b, ..., c_n$ tiek uzstādīti ar skaitlisko vērtību nulle. Hiperparametri $threshold_a, threshold_b, ..., threshold_n$, kas nosaka mērķa modeļa maiņu, tiek atrasti, izmantojot parametru režģa pārmeklēšanu, taču tos var arī uzstādīt kā apmācāmus parametrus. MDQN izmanto arī stāvokļu pāreju kopas a_t, s_t, s_{t+1}, R_t, kas ir līdzīgas tām, kas ir redzētas atmiņā, lai apmācītu Q-vērtību funkciju, izmantojot patiesās kumulatīvās balvas vērtības. Šīs pārejas var viegli salīdzināt atmiņā, kad tiek izmantotas vides ar mazu skaitu dimensiju, kā tas ir *PLE* (*PyGame Learning Environment*) [104], kā arī tās var tik atrastas arī daudzdimensiju (pikseļu matricu) vidēs, izmantojot dziļo metriku apmācību, kas aprakstīta nākamajā nodaļā.

1. algoritms. *MDQN* kļūdas funkcija

procedure TRAIN
    $Q_1, Q_2, ..., Q_n$
    $threshold_a, threshold_b, ..., threshold_n$
    $c_a, c_b, ..., c_n = 0$
    $Q_a = Sample(Q_1, Q_2, ..., Q_n)$
    $Q_b = Sample(Q_1, Q_2, ..., Q_n)$
    while $Training = True$ **do**
        for **do**$\{a_t, s_t, s_{t+1}\}$ sample from $ReplayBuffer$
            $\forall n, c_n = c_n + 1$
            if $\forall n, c_n > threshold_n$ **then**
                $Q_b = Q_a$
                $Q_a = Q_n$
                $c_n = 0$
            if $\{a_t, s_t, s_{t+1}\}$ similar exist in $ReplayBuffer$ **then**
                $Q_a(a_t,s_t) \leftarrow \sum_{t=0}^{t+1} \gamma^t R_t$
            else
                if $s_t \neq$ terminal state **then**
                    $Q_a(a_t,s_t) \leftarrow R_t + \gamma \max_a Q_b(a, s_{t+1})$
                else
                    $Q_a(a_t,s_t) \leftarrow R_t$
        while $s_t \neq$ terminal state **do**
            $a_t \leftarrow max_a average(\{Q_a(a, s_t), Q_b(a, s_t)\})$
            ...
            store $\{a_t, s_t, s_{t+1}, r_t\}$ in $ReplayBuffer$

indentfirst

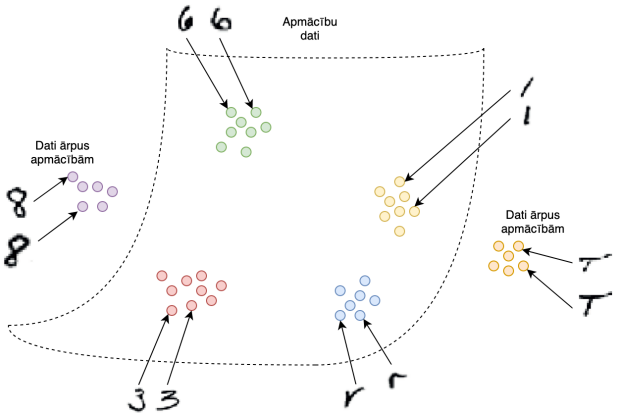
4. FUNKCIJU FORMĒŠANA DZIĻAJĀ METRIKU MAŠĪNMĀCĪŠANĀ

Šī nodaļa iepazīstina ar jaunizveidoto eksponenciālo trijotnes kļūdas funkciju un jaunām apmācības procedūrām dziļo metriku apmācības uzdevumiem. Tajā tiek iepazīstināts arī ar jaunām normalizācijas funkcijām latentajiem vektoriem, kas nodrošina leņķisko attālumu īpašības, izmantojot Eiklīda attālumus.

Problēmsfēra reidentifikācijas uzdevumam aprakstīta 4.1. nodaļā. Jau zināmās dziļās mašīnmācīšanās kļūdas funkcijas, lai risinātu reidentifikācijas uzdevumu aprakstītas 4.2. nodaļā. Visbeidzot, jaunā kļūdas funkcija un normalizācijas funkcijas aprakstītas 4.3. nodaļā, un to rezultāti doti 5.3. nodaļā.

4.1. Nulles-šāviena mašīnmācīšanās un reidentifikācijas uzdevums

Nulles-šāviena mašīnmācīšanās ir metode [66], [23], [44], [62], kas balstās uz modeli, ko lieto kategorizācijas uzdevumam jaunām, apmācības laikā neredzētām klasēm (12. attēls). Parasti datu kopas apmācībai un testēšanai ir no vienas un tās pašas sfēras, piemēram, seju fotoattēlu, balss ierakstu, auto fotoattēlu, u.c. datu kopas. Ieguvums šādiem modeļiem ir tajā, ka tos nav nepieciešams vēlreiz apmācīt, kad uzdevumā parādās jaunu klašu piemēri, un šo modeļu uzvedību var pielāgot arī pēc apmācības fāzes. Pielāgošanu veic, mainot parametrus klasterizācijas un latento vektoru telpas normalizācijas parametriem pēc apmācības. Vēl viens ieguvums ir tas, ka jaunajām klasēm nepieciešami tikai pāris datu piemēri, nevis tūkstošiem datu piemēru kā parasti tas būtu nepieciešams klasifikācijas modeļiem, kas neizmanto nulles-šāviena mašīnmācīšanās metodes [116]. Nulles-šāviena apmācība ir efektīva, ja ir mazs paraugu skaits, pat ar diviem datu paraugiem var sasniegt augstu klasifikācijas precizitāti [116].



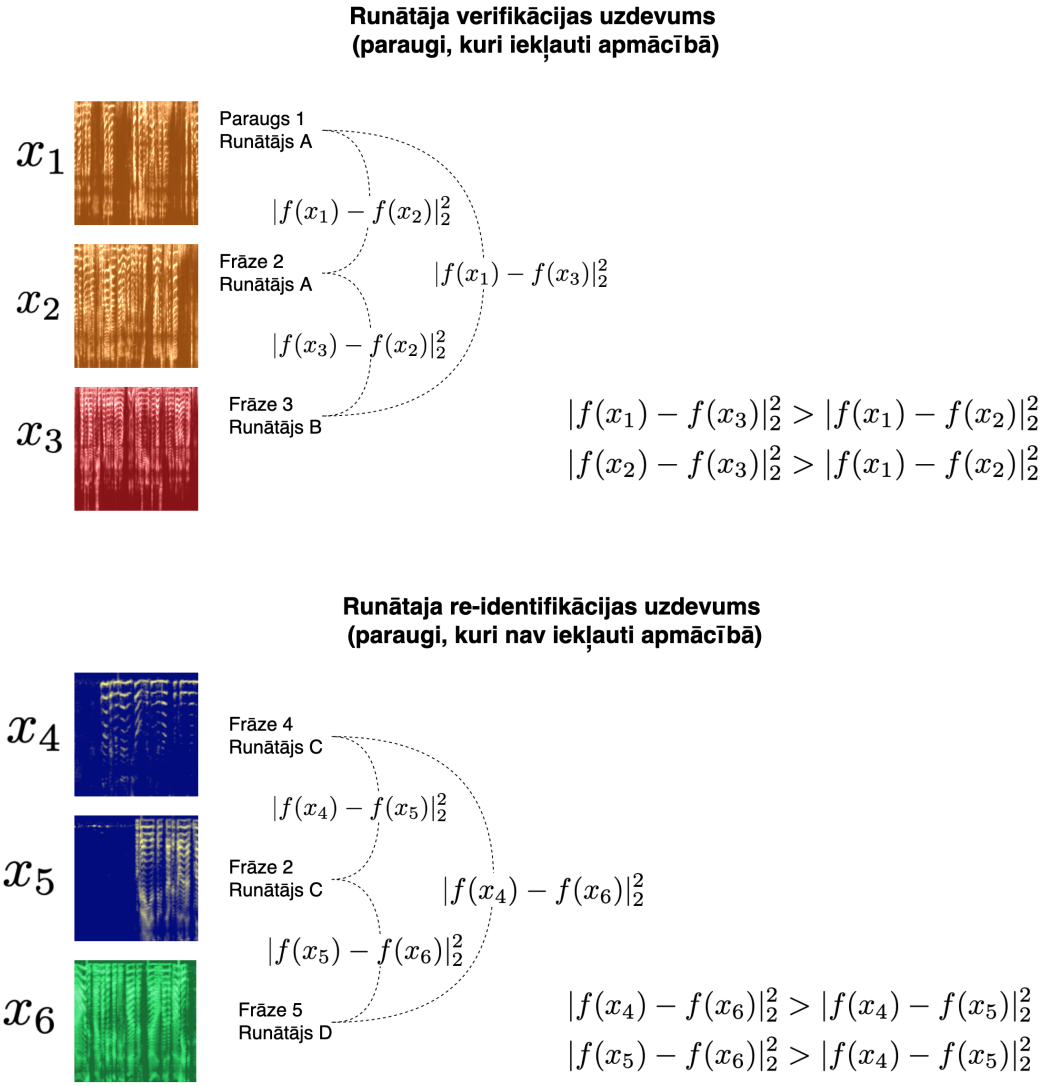
12. att. Nulles-šāviena mašīnmācīšanās klasteru vizualizācija, izmantojot *EMNIST* datu kopu, kur tās sadalītas tā, lai apmācības un testa datu kopās nebūtu iekļautas vienas un tās pašas klases.

Dažreiz zinātniskajā literatūrā nulles-šāviena apmācību sauc arī par viena-šāviena apmācību, lai gan tehniski viena-šāviena apmācība nozīmē, ka viens paraugs no mērķa klasēm ir ticis iekļauts apmācības procedūrā. Tāpat eksistē arī pāris-šāvienu mašīnmācīšanās, kur pāris paraugu no mērķa klasēm tiek iekļauti apmācību kopā. Šādus modeļus parasti papildus apmāca vēlāk, izmantojot pārcelšanas apmācības metodes (*transfer learning*), lietojot šīs pāris jaunās klases. Nulles-šāviena mašīnmācīšanās gadījumā modelis netiek nemaz apmācīts ar jauno klašu paraugiem.

Modelis ir kļuvis par kodētāju, kas spēj daudzu dimensiju datus, piemēram, fotoattēlus, spēj iekodēt pāris dimensiju latentajos vektoros, starp kuriem attālumi jeb distances nodrošina to klasterizāciju pēc semantiskās informācijas. Šādus nulles-šāviena modeļus arī nereti sauc arī par dziļo metriku apmācību (*DML*) vai distanču metriku apmācību.

Šie modeļi bieži tiek lietoti seju reidentifikācijas [23] un citu biometrisko datu verifikācijas sistēmām (13. attēls).

Šādās sistēmās katra persona tiek uzņemta sistēmā, izmantojot vienu vai vairākus paraugus ar sejas fotoattēlu, kas pieder jaunai datu klasei, ko vēlāk modelis spēj atpazīt jebkurā fotoattēlā, kurš netika iekļauts uzņemšanas procedūrā.



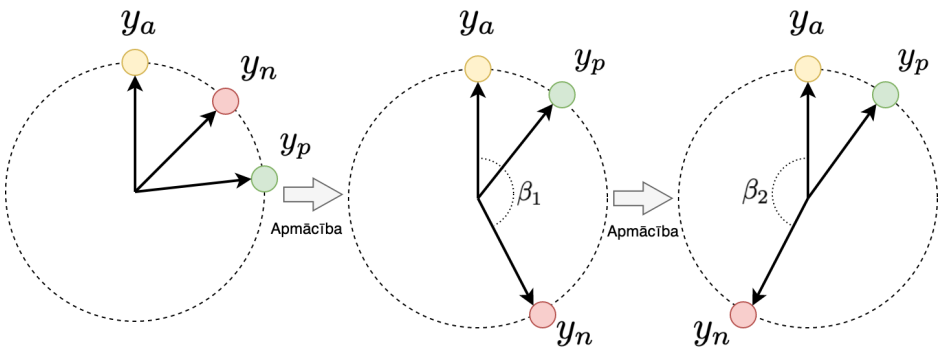
13. att. Atšķirība starp runātāja verifikācijas un runātāja reidentifikācijas uzdevumu (implementēts praktiskā lietojumā - komerciālā lietotnē "asya") Redentifikācijas uzdevums pieder pie nulles-šāviena apmācības, jo modelis f nav redzējis audio paraugus x_4, x_5, x_6 apmācības laikā.

4.2. Trijotnes kļūdas funkcija

Lai modelis no ievades datiem iemācītos dziļas reprezentācijas latentajā vektoru telpā, izmantojot nulles-šāviena apmācību, parasti tiek izmantota kontrastīvā kļūdas funkcija [8] vai trijotnes kļūdas funkcija [23].

Trijotnes kļūdas funkcijas (44. vienādojums) mērķis ir samazināt attālumu latentajiem vektoriem starp vienas klases paraugiem $\|y_a - y_p\|_2^2$ un palielināt attālumu starp dažādu klašu paraugiem $\|y_a - y_n\|_2^2$, kā redzams 14. attēlā.

$$\mathcal{L}_{std} = |\|y_a - y_p\|_2^2 - \|y_a - y_n\|_2^2 + \alpha|_+ \tag{44}$$



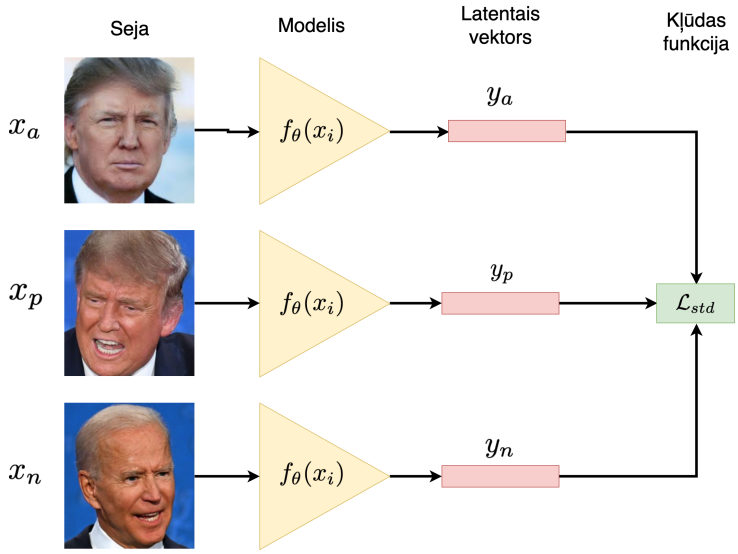
14. att. Apmācības process, izmantojot trijotnes kļūdas funkciju (44. vienādojums) ar kosinusa distanci starp enkura y_a vektoru, tās pašas klases pozitīvo vektoru y_p, un citas klases negatīvo vektoru y_n. Kad ir sasniegts maksimālais attālums starp vektoru pāriem, šis attālums atkal samazinās.

Tajā pašā laikā funkcija ir izveidota tā, lai tā nereducētu visus vienas klases vektorus vienā punktā telpā, bet, lai starp tiem būtu robežas attālums α. Nereti kā attālumu metriku izmanto kosinusa vai Eiklīda distances. Vēlams izmantot kosinusa distanci, jo tai ir cikliska daba, kur paraugu attālums sasniedzot robežvērtību 2 atkal samazinās līdz robežvērtībai 0.

Trijotnes kļūdas funkcijām nepieciešams paraugu meklēšanas algoritms un ierobežojumi, kādus paraugus drīkst lietot ar šo kļūdas funkciju. Piemēram, parasti modelis tiek apmācīts ar pozitīvajiem vienas klases paraugu pāriem, starp kuriem ir lieli attālumi un starp negatīvajiem paraugu pāriem, kuriem ir mazi attālumi. Šādi paraugi, pabeidzot apmācību, nav

vēlami, tāpēc tie dod lielāko ietekmi uz kļūdas funkciju, taču šo paraugus ir īpaši jāmeklē, un tie parasti netiek atlasīti nejaušā veidā.

Modelis $f_\theta(x)$ ir kodētājs, kas reducē dimensiju skaitu no daudzdimensiju ievades datiem x uz mazāka skaita dimensiju latento vektoru y. Latentam vektoram jābūt vismaz 32 dimensijām vai vairāk, kā paskaidrots pielikumā [110]. Parametri θ ir vienādi visiem datu paraugiem x, kā redzams 15. attēlā.



15. att. Piemērs trijotnes kļūdas funkcijai sejas reidentifikācijas uzdevumam. Kodētāja modelim $f_\theta(x_i)$ ir kopēji svari. Dimensiju skaits y_i vektoram ir ievērojami mazāks nkā x_i ievades datiem.

4.3. Eksponenciālā kļūdas funkcija

Promocijas darbā līdz ar citiem jaunievedumiem, ir aprakstīta jaunizveidotā eksponenciālā trijotnes kļūdas funkcija $\mathcal{L}_{exp}$, kuras skaitlisko vērtību telpā kļūdas telpa parādīta 16. attēlā, kura konverģē ātrāk par trijotnes kļūdas funkciju $\mathcal{L}_{std}$, kura aprakstīta iepriekšējā nodaļā.

Eksponenciālajai trijotnes kļūdas funkcijai $\mathcal{L}_{exp}$ ir forma, kas nepieļauj negatīvo pāru attālumus, kuri būtu mazāki par pusi no maksimā-

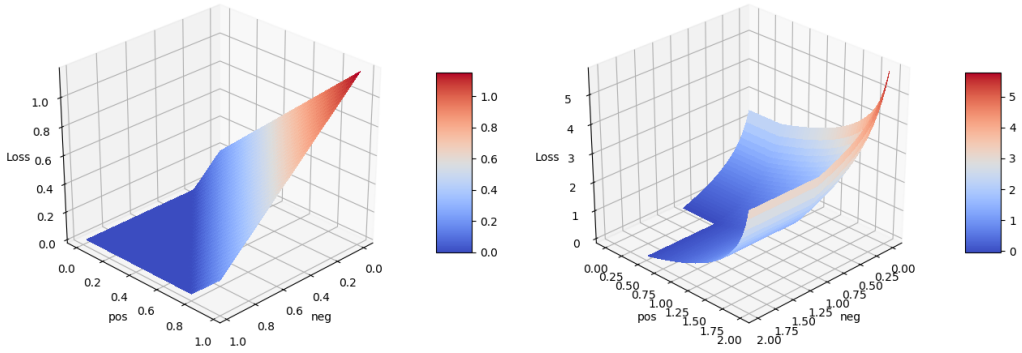
lā attāluma sfēriskajā telpā $\max(f_{emb}(x))$, kas bez mērogošanas parasti ir $\max(f_{emb}(x)) = 2.0$.

Savukārt c_n ir minimālā klasteru distance līdzīgi kā robežvērtība α, ko lieto trijotnes kļūdas funkcijā $\mathcal{L}_{std}$. To aprēķina, dalot maksimālo latento vektoru telpas distanci $\max(f_{emb}(x))$ ar klašu skaitu apmācību apmācību kopā K (45. vienādojums).

$$c_n = \frac{\max(f_{emb}(x))}{K} \tag{45}$$

$$emb_p = \frac{\|y_a - y_p\|_2^2}{\max(f_{emb}(x))} \qquad emb_n = \frac{\|y_a - y_n\|_2^2}{\max(f_{emb}(x))} \tag{46}$$

$$\mathcal{L}_{exp} = -log(1.0 - \frac{|emb_p - c_n|_+}{1 - c_n} + \epsilon) - log(1.0 - \frac{|0.5 - emb_n|_+}{0.5} + \epsilon) \tag{47}$$



16. att. $\mathcal{L}_{std}$ un $\mathcal{L}_{exp}$ kļūdas funkciju salīdzinājums. Pozitīvie vienas klases pāru attālumi norādīti ar $\|y_a - y_p\|_2^2$ (pos), negatīvie dažādu klašu attālumi ar $\|y_a - y_n\|_2^2$ (neg).

Lai vēl vairāk uzlabotu L_{exp} veiktspēju, vairākas citas kļūdas funkcijas no neseniem pētījumiem tika pievienotas kļūdas funkcijai, lai izveidotu kompozīta kļūdas funkciju *Lcomp* (51. vienādojums). Tika pievienotas L2-ierobežotā krusteniskā-entropijas kļūdas funkcija $\mathcal{L}_{class}$ [84] un Centra kļūdas funkcija $\mathcal{L}_{center}$ [128] (49. un 50. vienādojums). L2-ierobežotās krusteniskās-entropijas kļūdas funkcijas $\mathcal{L}_{class}$ gadījumā pirms *Softmax* funkcijas latentais

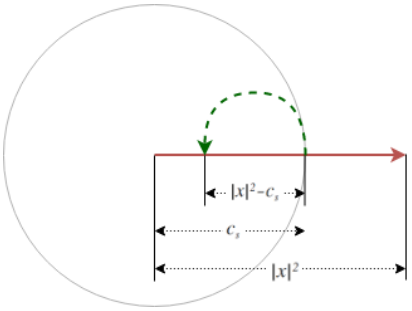
vektors $f(x)$ tiek L2 normalizēts un mērogots ar s. Savukārt $\mathcal{L}_{center}$ gadījumā apmācību laikā klašu latentie vektori tiek akumulēti un tiek aprēķināti c_{y_i} klašu masas centru vektori, kas tiek izmantoti šajā kļūdas funkcijā.

$$\mathcal{L}_{class} = -\sum_{i=1}^{M} y_i log \frac{e^{W_i^T s|f(x_i)|_2^2 + b_i}}{\sum_{j=1}^{C} e^{W_j^T s|f(x_i)|_2^2 + b_j}} \tag{48}$$

$$\mathcal{L}_{center'} = \sum_{i=1}^{M} ||x_i - c_{y_i}||_2^2 \tag{49}$$

$$\mathcal{L}_{center} = \sum_{i=1}^{M} |||x_i - c_{y_i}||_2^2 - \frac{c_n}{2}|_+ \tag{50}$$

$$\mathcal{L}_{comp} = \mathcal{L}_{exp} + C_{center}\mathcal{L}_{center} + C_{class}\mathcal{L}_{class} \tag{51}$$



17. att. *Unit-Bounce* normalizācijas funkcijas ilustrācija latentajiem vektoriem eksponenciālajā trijotnes kļūdas funkcijā.

Vēl viens jaunievedums šajā darbā ir latento vektoru normalizācijas funkcija *Unit-Bounce* (52. un 53. vienādojums). Tai ir līdzīgas īpašības kā kosinusa distancei, bet tā tiek aprēķināta ar Eiklīda distancēm. Kad latentais vektors sasniedz sfēras rādiusu c_s, tas virzās atpakaļ latentās telpas centra virzienā (17. attēls), un tad, kad tas sasniedz otru sfēras malu, tas atkal virzās uz centru pretējā virzienā. Tādējādi tiek panākts, ka visa 3D latentā telpa tiek efektīvi izmantota, nevis kā L2 gadījumā, kur vektori atrodas tikai uz sfēras virsmas.

$$f'_{emb}(x) = \begin{cases} f_{bounce}(x), & \text{if } |x|^2 \geq 1 \\ x, & \text{citos gadījumos} \end{cases} \tag{52}$$

$$f_{bounce}(x) = \begin{cases} |x|^2 - \left\lfloor \frac{|x|^2}{c_s} \right\rfloor - c_s \frac{x}{|x|^2}, & \text{if } \lfloor \frac{|x|}{c_s} \rfloor \bmod 2 = 0 \\ c_s \frac{x}{|x|^2} - |x|^2 - \lfloor \frac{|x|^2}{c_s} \rfloor, & \text{citos gadījumos} \end{cases} \tag{53}$$

5. EKSPERIMENTĀLIE REZULTĀTI UN PRAKTISKIE LIETOJUMI

Šajā nodaļā doti galvenie eksperimentu rezultāti jaunizveidotajām kļūdas funkcijām un metodēm, kas aprakstītas 2., 3. un 4. nodaļā Jaunizveidotās metodes tika pārbaudītas ar dažādām problēmsfērām, sākot ar *VI* algoritma optimizāciju, dziļās mašīnmācīšanos datorspēļu vidēs, dziļās metriku apmācību seju reidentifikācijas uzdevumam, un beidzot ar praktiskiem lietojumiem analītiskajā ķīmijā un cilvēka balss analīzē. Pilni eksperimentālie rezultāti atrodami publikācijās, kas tika pievienotas pilnajam promocijas darba tekstam pielikumu veidā [111], [109], [24], [112], [110].

5.1. UNet-RNN-Skip rezultāti

UNet-RNN-Skip modelis tika eksperimentāli testēts *VI* algoritma problēmas risināšanai, lai paātrinātu vērtību funkcijas konverģenci. *VIN* (*Value Iteration Network*) modelis balstīts uz jaunizveidoto *UNet-RNN-Skip*, ko var lietot arī citu problēmu risināšanai. *VIN* modelis ir izmantojams dažādu izmēru aizņemtības režģu kartēm, kā redzams 18. attēlā un 3. tabulā, un to nav nepieciešams papildus apmācīt noteiktam kartes izmēram. *VIN* modeļa konverģences ātrums samazinās lineāri starp dažādu izmēru kartēm, savukārt *VI* algoritma konverģences ātrums samazinās eksponenciāli, palielinoties kartes izmēram. Metrika "mērķa sasniegšanas proporcija" (*success rate*) tika izmantota, lai novērtētu modeļa stabilitāti uz testa datu kopu. Šī metrika norāda uz procentuālu skaitu režģa šūnu, no kurām ar *VIN* palīdzību var nokļūt līdz pozitīvam terminālam stāvoklim kartē. *VI* algoritmam šīs metrikas vērtība vienmēr būs 1, jo pēc konverģences *VI* algoritms ir spējīgs iegūt optimālu ceļa plānojumu no jebkuras režģa šūnas kartē. Pētījumā arī tika noskaidrots arī tas, ka *VI* uzdevumam *UNet* modeļi precizitātē pārspēj kovolūciju *AE* modeļus (2. tabula).

2. tabula.

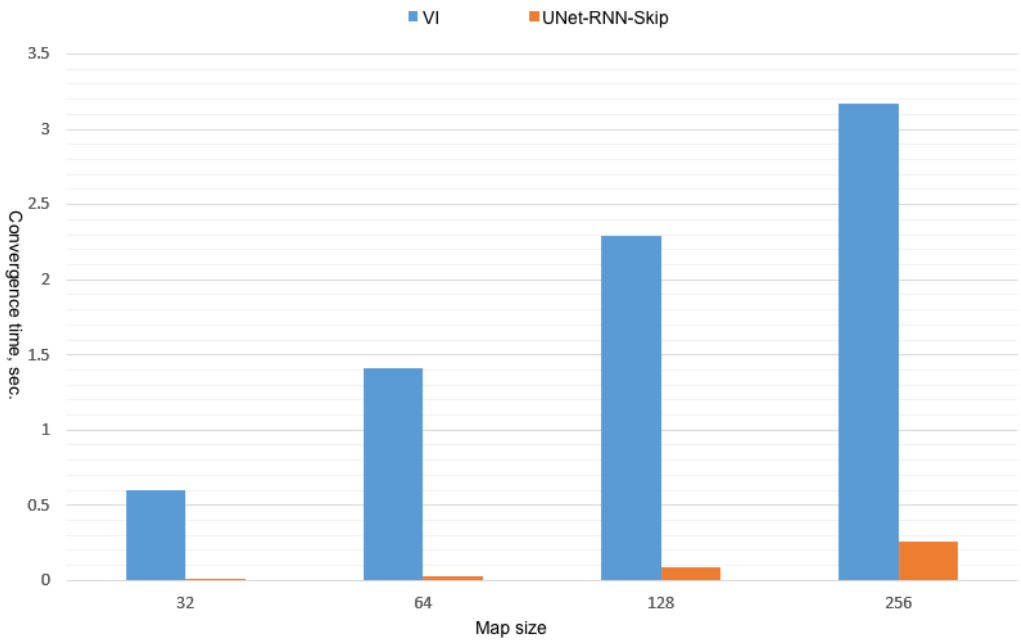
ConvNet un *UNet* modeļu rezultātu salīdzinājums, lai atrisinātu *VI* algoritmu.

Modelis	Kļūda	Mērķa sasnieg. pr.	Epohas laiks (min)
Conv-AE-RNN	8.58E-06	0.598	10.862
UNet-RNN-Skip	3.04E-06	0.998	15.833

3. tabula.

VIN un *VI* metožu konverģences ātruma (sek.) salīdzinājums.

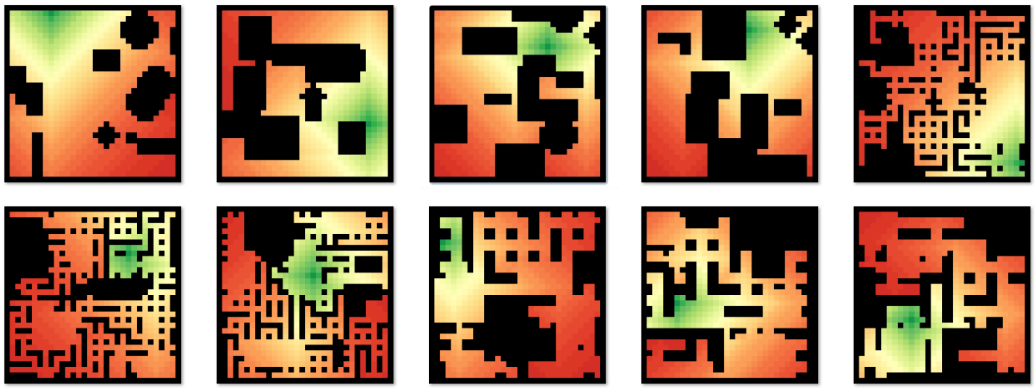
Modelis / Kartes izm.	32	64	128	256
VI	2.95	24.873	195.902	1473.108
VIN	0.031	0.071	0.236	0.833



18. att. *VIN* un *VI* metožu konverģences ātruma (sek.) salīdzinājums.

Līdz ar jaunizveidoto *UNet-RNN-Skip* modeli šajā pētījumā tika izveidots arī sintētiskas datu kopas ģenerators *OccupancyMapGenerator*, kas spēj izveidot aizņemtības režģu kartes ar šķēršļiem, kā redzams 19. attēlā. Ģenerators ir spējīgs izveidot jebkura izmēra 2D aizņemtības režģa karti, izmantojot komandrindas argumentus. Tas izveido kartes *PNG* attēlu formātā un arī izpilda *VI* algoritmu uz šīm kartēm, un šo rezultātu arī saglabā kā PNG attēlu. Ģeneratoram var norādīt šķēršļu daudzumu kartē. Šķēršļi sastāv no nejauši izveidota labirinta, taisnstūriem un apļiem.

Lai pārbaudītu, vai no visām režģa šūnām var nonākt pozitīvā terminālā stāvoklī, tika lietots *Dijkstra* ceļu meklēšanas algoritms [18]. Labirinta izveidošanai tika lie tots rekursīvs atpakaļiešanas algoritms. Pilns karšu ģenerēšanas algoritms dots pseudo kodā (2. algoritms).



19. att. Sintētisko karšu, kas ģenerētas ar *OccupancyMapGenerator* un šūnu vērtībām, piemēri, lai sasniegtu pozitīvu terminālu stāvokli (zaļā krāsā — augstākā balvas vērtība, sarkanā — zemākā balvas vērtība).

2. algoritms. *OccupancyMapGenerator* karšu ģenerēšanas algoritms

procedure GENERATEMAP
    $size$
    $types\_obstacles = \{maze, circles, rectangles\}$
    $max_{coverage}$
    $iterations\_obstacles$
    ϵ_{VI}
    $M_{size \times size} \leftarrow generateZeros(size)$
    if $maze \in types\_obstacles$ **then**
        $M \leftarrow generateMaze(M)$
    if $circles \in types\_obstacles$ or $rectangles \in types\_obstacles$ **then**
        for $iterations\_obstacles$ **do**
            $coverage = \frac{walkable}{size^2}$
            if $coverage < max_{coverage}$ **then**
                $M \leftarrow generateObstacles(M, types\_obstacles)$
            else
                break
    $goal_{x,y} \leftarrow RandomWalkablePosition(M)$
    for $pos_{x,y}$ in M **do**
        if $pos_{x,y} \in walkable$ **then**
            $reachable \leftarrow dijkstra(M, goal_{x,y}, pos_{x,y})$
            if $\neg reachable$ **then**
                $M \leftarrow fillHole(M, pos_{x,y})$
    $M_{vi} \leftarrow valueIteration(M, \epsilon_{VI})$
    $store(M_{vi}, M)$

5.2. MDQN rezultāti

MDQN kļūdas funkcija [109] tika pārbaudīta dažādās datorspēļu vidēs, izmantojot *PLE* (*PyGame Learning Environment*) [104]. Tā tika notestēta spēlēs, piemēram, "*Flappy Bird*", "*Pong*", "*3D Raycast Maze*", "*Doom*" (20. attēls). Šīs spēles satur daudzdimensiju stāvokļus jeb šie stāvokļi ir attēla pikseļu matrica katram spēles kadram, bet dažas spēles, piemēram, "*Pong*" piedāvā arī mazdimensiju spēles stāvokļus, kas ir noderīgi, lai ātri pārbaudītu jaunus modeļus vai kļūdas funkcijas, pirms tās tiek apmācītas uz sarežģītākiem daudzdimensiju stāvokļiem.

Kā redzams 4. tabulā, *MDQN* kļūdas funkcija sasniedz augstākus rezultātus nekā *DDQN* kļūdas funkcija, kas publikācijas izstrādes laikā bija labākā metode Q-funkcijā balstītai stimulētajai mašīnmācīšanai [29]. Visās vidēs un kļūdas funkciju kombinācijās rezultāti tika iegūti ar hiperparametru režģa pārmeklēšanu, lai nodrošinātu jēgpilnu salīdzinājumu starp metodēm, tāpat eksperimenti tika atkārtoti 20 reižu.

20. att. *PLE* spēļu piemēri. Augšējā rindā no kreisās "*Pixel Helicopter*", "*Monster Kong*", bet no labās "*Flappy Bird*". Apakšējā rindā "*Doom*" spēles divi varianti, kas arī tika izmantoti šajā pētījumā (video ar aģentu, kas apmācīts ar *MDQN* kļūdas funkciju pieejams `https://www.youtube.com/watch?v=oqN6rtnv1EI`).

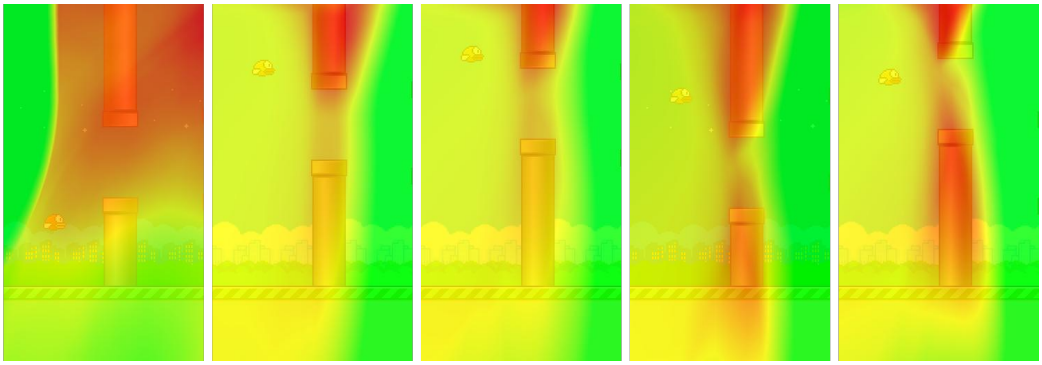
MDQN kļūdas funkcija ieguva labākus rezultātus nekā *DDQN* kļūdas funkcija, un to rezultāti bija stabilāki, atkārtojot eksperimentus vairākas reizes (4. tabula).

4. tabula.

Vidējā balvas vērtība, atkārtojot eksperimentus 20 reižu un izmantojot dažādas kļūdas funkcijas *PLE* vidēs.

Kļūdas funkc.	Vide	Vid. balva
MDQN	Flappy Bird	17.2
DDQN	Flappy Bird	16.9
MDQN	Pong	1.7
DDQN	Pong	1.3
MDQN	3D Raycast Maze	3.9
DDQN	3D Raycast Maze	3.7

Līdz ar *MDQN* kļūdas funkciju, šajā darbā tika izveidota arī jauna metode Q-funkcijas vērtību vizuālai attēlošanai, katram spēles stāvoklim pikseļu matricā. Jaunās Q-vērtību kartes iegūtas, manipulējot ar objektiem pašā datorspēles vidē, tādējādi iegūstot sapratni par aģenta politiku katrā iespējamā spēles stāvoklī. *PLE* vides ir veidotas ar atvērtu kodu, tāpēc ir iespējams manipulēt ar spēlētāja atrašanās vietu spēlē, kas ļauj aprēķināt Q-vērtību katrā pikseļa pozīcijā viena stāvokļa ietvaros (21. attēls).



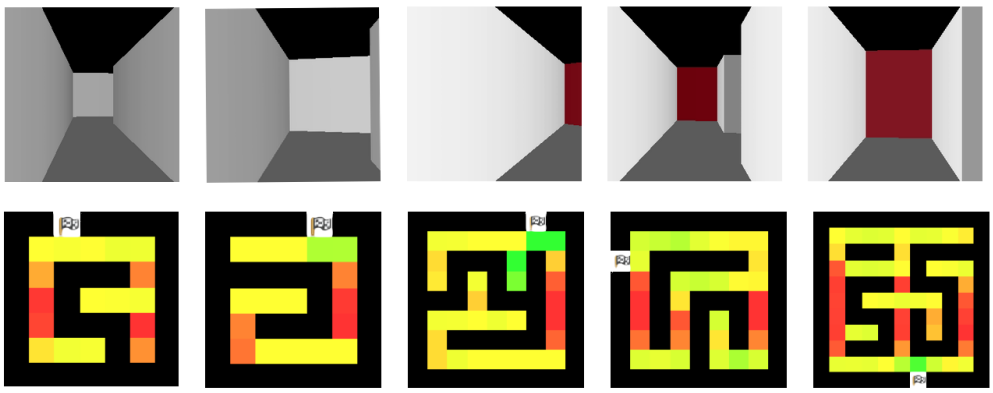
21. att. Q-vērtību karte "*Flappy Bird*" vidē, kur katra pikseļa vērtība nosaka to, kāda būtu stāvokļa Q-vērtība, ja spēlētājs (putns) atrastos šajā pozīcijā. Ar sarkano krāsu apzīmēta zema Q-vērtība, ar zaļu – augsta Q-vērtība. Pa kreisi – Q-vērtību kartes pirms apmācības, vidū – apmācības laikā, pa labi – pēc apmācības.

Pat 3D vidēm, kā "*3D raycast maze*" (3D labirints no pirmās personas skatupunkta) ir iespējams noteikt katra stāvokļa Q-vērtību,pagriežot kameru ap Z asi un aprēķinot vidējo Q-vērtību no katra kadra, kas tika iegūts 360 grādu pozīcijās. Pēc tam vidējās vērtības tiek saglabātas virsskatā (22. attēls). Kā redzams 5. tabulā, *MDQN* kļūdas funkcija ieguva augstāko vidējo balvas vērtību 20 apmācību atkārtojumos un zemāko novirzi, salīdzinot ar *DDQN* un *DQN* kļūdas funkcijām. Ar *MDQN-2* un *MDQN-3* tiek apzīmēts modeļu kopiju skaits – 2 un 3, kas tiek izmantoti *MDQN* kļūdas funkcijā.

5. tabula.

Dažādu kļūdas funkciju rezultāti 20 apmācību atkārtojumos, izmantojot "*3D raycast maze*" vidi.

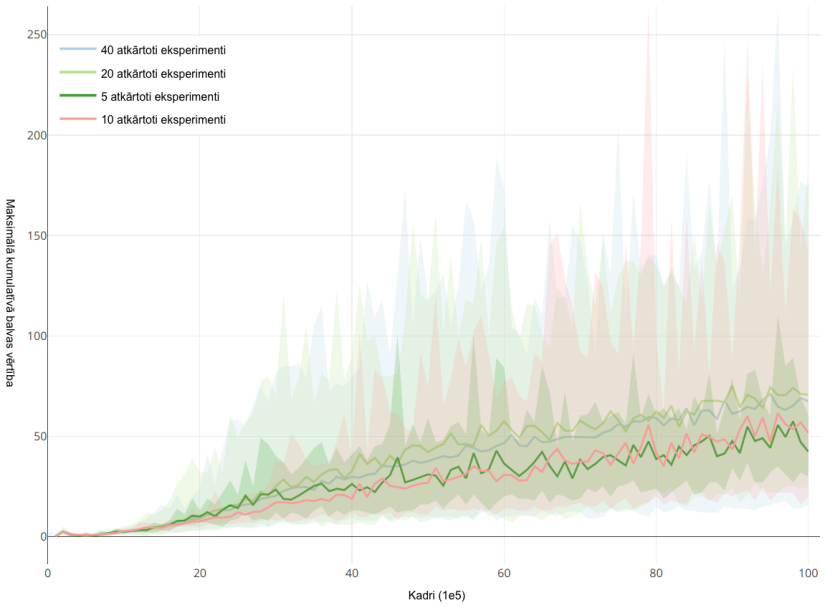
Kļūdas func.	Apmāc. Ātr.	Vid. balva	Std. balvai
MDQN-2	1.00E-05	3.904359232	0.728045918
DQN	1.00E-05	3.88654262	2.124993494
MDQN-2	1.00E-06	3.7166532	0.154117942
DDQN	1.00E-06	3.713829593	1.524318234
DDQN	1.00E-05	3.638360789	1.662039807
DDQN	0.0001	3.267777864	2.889255991
MDQN-3	1.00E-05	3.056116361	2.339890159
DQN	1.00E-06	3.026868771	2.028895348
MDQN-3	1.00E-06	2.770128326	0.714132328
MDQN-2	0.0001	2.545370799	4.120312752
DQN	0.0001	2.24425396	2.153779645
MDQN-3	0.0001	2.174641347	3.541216037



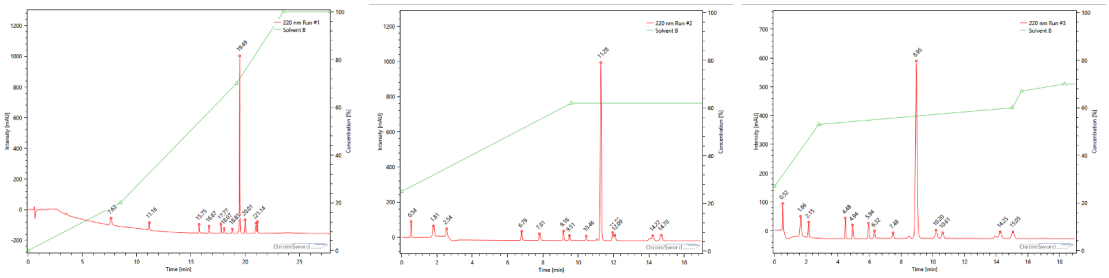
22. att. Augšējā rindā kadri no "*3D raycast maze*" vides. Apakšējā rindā Q-vērtību kartes šai videi no virsskata. Kreisā pusē – pirms apmācības, vidū – apmācības laikā, labajā pusē – pēc apmācības. Apmācību laikā tika pakāpeniski palielināta labirinta sarežģītība. Ar sarkanu krāsu apzīmēta zema Q-vērtība, ar zaļu – augsta Q-vērtība

Visbeidzot, līdz ar *MDQN* kļūdas funkcijas izveidi, tika arī veikta citu dziļo Q-funkciju modeļu analīze un salīdzinājums. Tika secināts, ka šīs metodes ir ļoti jutīgas uz nejaušo skaitļu iestatījumu spēles sākumā, kā arī pašu spēļu nenoteiktību. Kā redzams 23. attēlā, nepieciešams izmantot pietiekamu skaitu atkārtotu eksperimentu jeb apmācību atkārtojumus ar tiem pašiem hiperparametriem, lai noskaidrotu augstāko rezultātu katrā vidē. Tika noskaidrots, ka nepieciešams 20 apmācību atkārtojumi, lai noteiktu precīzu vidējo vērtību apmācību rezultātiem.

MDQN kļūdas funkcija un citas dziļās mašīnmācīšanās metodes tika lietotas arī komerciālā produktā SIA *ChromSword*. Kā redzams 24. attēlā ar trīs secīgiem eksperimentiem, izmantojot šos modeļus, ir iespējams atrast HPLC šķīdinātāja gradientu, lai panāktu labāko pīķu sadalījumu nezināmām vielām. Šos modeļus izmanto analītiskajā ķīmijā un zāļu izstrādē.



23. att. Novirze rezultātu vērtībās, atkārtojot apmācību dažādu skaitu reižu, izmantojot *DQN* kļūdas funkciju "*Flappy Bird*" vidē.



24. att. Secīgi *HPLC* eksperimenti, ko izveido stimulētās mašīnmācīšanās aģents, lai atrastu labāko šķīdinātāja gradientu, lai atdalītu pīķus nezināmām vielām.

5.3. Eksponenciālās trijotnes kļūdas funkcijas rezultāti

Klašu reidentifikācijas uzdevuma rezultāti, izmantojot eksponenciālo trijotnes kļūdas funkciju L_{exp}, apkopoti 6. tabulā, kur standarta trijotnes kļūdas funkcija L_{std}, eksponenciālā trijotnes kļūdas funkcija L_{exp}, centra kļūdas funkcija L_{cen} un klasifikācijas kļūdas funkcija L_{cls}. Visi rezultāti tika iegūti, veicot režģa pārmeklēšanu hiperparametriem, katrai metodei atsevišķi. Tas ir viens no iemesliem, kāpēc arī standarta trijotnes kļūdas funkcija L_{std} ieguva augstus rezultātus daudzās no dotajām datu kopām. Visi klasifikācijas rezultāti tika iegūti, izmantojot nulles-šāviena modeļus un reidentifikācijas uzdevumu, kur pareizā klase tika noteikta pēc latentā vektora attāluma līdz klases masas centram jeb vidējam latentā vektoram no visiem vienas klases paraugiem.

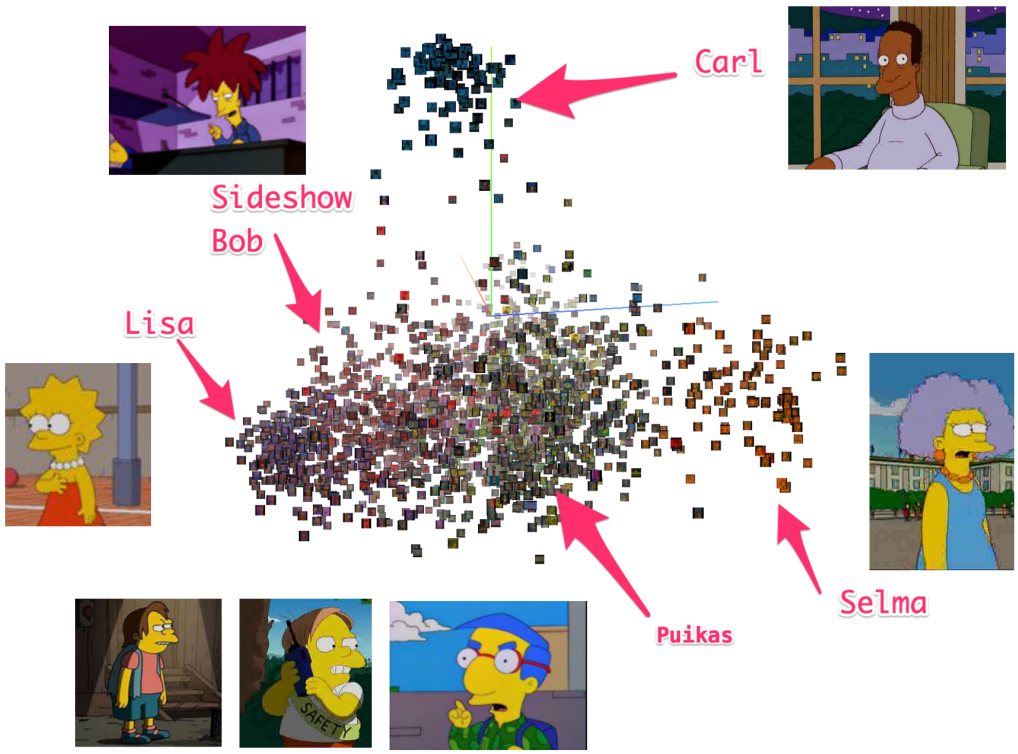
Klases un paraugi, kas tika iekļauti apmācību kopā netika iekļauti testēšanas kopā. Abām L_{std} un L_{exp} tika izmantota vairāku daļu kļūdas funkcija, kas ietvēra arī centra kļūdas funkciju L_{cen} un *L2-Softmax* klasifikācijas kļūdas funkciju L_{cls}. Klasifikācijas kļūdas funkcija L_{cls} aprēķināta tikai apmācības kopai, jo modelim nebija zināms testa kopā iekļauto klašu skaits. Pētījumā seju reidentifikācijas uzdevuma veikšanai tika izvēlēta datu kopa *VGGFace2* [13] ar 9000 klasēm, savukārt modeļi tika arī testēti uz biežāk izmantotām attēlu klasifikācijas datu kopām, piemēram, *MNIST* [55], *Fassion-MNIST* [130], *EMNIST* (*Extended-MNIST*) [15] un *CIFAR10* [47]. Arī šim datu kopām apmācības un testa kopu sadalījums tika pārdalīts tā, lai to klases nepārklātos. Visās datu kopās tika atliktas 20% klases ar to paraugiem testēšanai, savukārt 80% palika apmācībai.

6. tabula.

Nulles-šāviena precizitātes salīdzinājums uz testa datu kopas dažādām kļūdas funkcijām.

Kļūdas funkc. / Prec.	MNIST	FMNIST	EMINST	CIFAR10	Simpsons	VGGFace2
L_{std}	**99.6**	91.4	82.0	56.2	91.0	77.4
$L_{std} + L_{cls}$	**99.6**	92.1	85.0	79.8	91.2	76.3
$L_{std} + L_{cen}$	97.5	71.5	61.7	52.1	90.9	76.4
$L_{std} + L_{cen} + L_{cls}$	97.7	82.0	70.9	62.8	91.2	78.6
L_{exp}	**99.6**	92.7	82.7	85.7	91.5	85.0
$L_{exp} + L_{cls}$	**99.6**	**93.1**	85.2	87.2	90.9	84.1
$L_{exp} + L_{cen}$	**99.6**	**93.1**	85.7	85.3	**92.1**	84.0
$L_{exp} + L_{cen} + L_{cls}$	**99.6**	**93.1**	**86.0**	**87.3**	91.7	**85.7**

Papildus šim pētījumam tika izmantota arī Simpsonu datu kopa, ko var iegūt no *Kaggle* datu zinātnieku sacensību vietnes. Arī uz šīs datu kopas tika apmācīts reidentifikācijas uzdevums. Rezultāti, projicējot latentos vektorus no 32 dimensijām uz 3 dimensijām, izmantojot *PCA* redzami 25. attēlā. Paraugā ir iespējams novērot, ka dažādas multfilmas varoņu vizuālās īpašības tiek veiksmīgi sagrupētas un to savstarpējais attālums atbilst semantiskajai nozīmei.



25. att. *PCA* vizualizācija Simsonu testa dartu kopai, izmantojot modeli, kas apmācīts ar eksponenciālo trijotnes kļūdas funkciju L_{exp}, un šos paraugus nav izmantojis apmācībai.

5.4. Darba praktiskais lietojums

Jaunās kļūdas funkcijas un modeļus, kas aprakstīti šajā promocijas darbā, var izmantot dažādās praktiskās implementācijās, no kurām dažas jau ir testētas komerciālos produktos. Katrai jaunizveidotajai kļūdas funkcijai un modelim praktiskie lietojumi uzskaitīti tālāk tekstā.

Praktiskie lietojumi *UNet-RNN-Skip* aprakstīti tālākajā tekstā.

1. *VI* algoritma optimizācija, lai iegūtu ātrāku vērtību konverģenci un ātrāk veiktu ceļa plānošanas uzdevumu. Šis lietojums tika testēts pētījumā. Modelis var iegūt rezultātus ar par kārtu mazāk iterācijām nekā VI algoritms un ir mērogojamāks uz lielākām kartēm. Lai arī modelis tika apmācīts uz mazākām kartēm, tas vienlīdz labi spēj strādāt uz daudz lielāka izmēra kartēm, saglabājot ieguvumus.

2. *VI* algoritma optimizācija stimulētās mašīnmācīšanās uzdevumiem. Modeli var lietot arī, lai apmācītu politiku lēmumu pieņemšanas uzdevumos, kur, līdzīgi kā ceļa plānošanas uzdevumā, stāvokļi var saturēt informāciju, kas nepieciešama lēmuma pieņemšanai aģentam, bet vērtību gradients darbību grafā norāda uz lēmumu.

3. Stila pārneses uzdevums video datiem [40], [120]. Jauno modeli var izmantot arī, piemēram, lai veiktu melnbaltu filmu krāsošanu, pievienot troksni, lai iegūtu vecmodīgu stilu vai pārveidotu filmas par multfilmām.

4. Stila pārneses uzdevums audio datiem [16], [1]. Jauno modeli var lietot, lai klonētu balsis vai ģenerētu mākslīgas balsis, izmantojot ierakstītu balss paraugu kā ievades datus. Modeli var lietot 1D audio signālam vai 2D spektrogrammām, kur to pēc tam var rekonstruēt ar *Griffin-Lim* algoritmu.

5. Trokšņu noņemšanas uzdevums video datiem [20], [105]. Jauno modeli var izmantot, lai noņemtu troksni un vecu video ierakstu defektus. Modeli var lietot arī, lai uzlabotu video kompresiju, samazinot video detalizētību apgabalos, kuros skatītājs, visticamāk, atšķirību nepamanītu.

6. Trokšņu noņemšanas uzdevums audio datiem [58], [131]. Šis uzdevums veiksmīgi ir ticis implementēts SIA *Asya* komerciālajos risinājumos.

Modelis tiek izmantots, lai reālajā laikā noņemtu fona trokšņus dabīgās vidēs, atstājot tikai balss signālu. Pateicoties šim modelim, SIA *Asya* mobilā lietotne ir spējīga sniegt sarunu analīzi trokšņainās vidēs, piemēram, kafejnīcās vai birojos.

Praktiskie lietojumi *MDQN* kļūdas funkcijai aprakstīti tālākajā tekstā.

1. Q-vērtību funkcijā balstītu modeļu apmācība dažādiem stimulētās mašīnmācīšanās uzdevumiem, piemēram, vides simulācijai [42], pabraucošām mašīnām [134], virtuālajiem asistentiem [50] un robotu vadībai [77]. Promocijas darbā jaunā *MDQN* kļūdas funkcija tika testēta *PLE* datorspēļu kontroles uzdevumos. *MDQN* kļūdas funkcija ir īpaši efektīva mazdimensiju vidēs vai arī, izmantojot dziļo metriku apmācību, lai iekodētu daudzdimensiju stāvokļus mazdimensiju latentajos vektoros.

2. Šķīdinātāju gradientu optimizācija *HPLC*, ko izmanto analītiskajā ķīmijā [26]. Jaunā kļūdas funkcija ir izmēģināta komerciālā risinājuma SIA ChromSword. Modelis ir spējīgs atrast šķīdinātāja gradientu, lai veiktu vielu sastāvdaļu sadalīšanu ar secīgiem eksperimentiem divu stundu laikā, pirms tam šis uzdevums ar manuālu darbu prasīja vairākas dienas.

Praktiskie lietojumi eksponenciālās trijotnes kļūdas funkcijai aprakstīti tālākajā tekstā.

1. Iekodētāja modeļa pirmsapmācība, izmantojot dziļo metriku apmācību dažādiem uzdevumiem, piemēram, stimulētajai mašīnampmācībai, laika rindu apmācībai, klasifikācijai un regresijai [129]. Pirmsapmācība izmantojot eksponenciālo trijotnes kļūdas funkciju var uzlabot apmācības ātrumu pamatuzdevumam. Nosacījums šādas metodes lietojumam ir tajā, ka datu kopai jāsatur klasificētus datus vai arī jāsatur informācija par paraugu līdzību. Šo informāciju iespējams iegūt arī, izmantojot nepārraudzītās apmācības metodes, piemēram, *VAE* (*Variational Auto Encoders*) [19].

2. Reidentifikācijas uzdevums attēliem vai biometriskajiem datiem [23], [5]. Pētījumā jaunā kļūdas funkcija tika veiksmīgi lietota sejas un produktu attēlu reidentifikācijas uzdevumā. Tā veiksmīgi tika izmantota

arī SIA *Asya* komerciālajā risinājumā balss reidentifikācijas uzdevumam. SIA *Asya* veidotā mobilā lietotne reālā laikā izpilda balss reidentifikācijas uzdevumu, izmantojot balss biometriskos datus dabiskās sarunās. To arī ir iespējams izmantot dažādos e-komercijas risinājumos, kur nepieciešams atrast produktus pēc vizuālās līdzības.

6. Tālākie pētījumi

Par pamatu turpmākiem pētījumiem kļūdas funkciju īpašībām un formai var tik izmantoti rezultāti un secinājumi, kas aprakstīti šajā darbā. Pētījumi Q-vērtību funkcijas apmācībā un stimulētajā mašīnmācīšanā galvenokārt ir saistīti ar kļūdas funkciju, kas ir nemainīga neprognozējamā apmācību vidē. Q-vērtību modeļu kļūdas funkcijai varētu pievienot ziņkārības komponenti, lai motivētu vides izpēti bez ārējas balvas vērtības. Balvas vērtību vajadzētu noteikt, izmantojot tikai stāvokļa atšķirību no prognozētā stāvokļa, izmantojot pasaules modeli. Šajā virzienā ir jau veikti pētījumi, taču problēma vēl nav pilnībā atrisināta [12], [11].

Vēl viens pētniecības virziens būtu pievienot stimulētās mašīnmācīšanās modelim vairākas rakstīšanas un lasīšanas galvas, lai tas izmantotu papildu atmiņas tabulas, kurās varētu tikt glabāti redzēto stāvokļu latentie vektori, to secības un balvu vērtības. Šie modeļi tiktu apmācīti saglabāt un izmantot iekodētus faktus par vidi, lai maksimizētu kumulatīvo balvas vērtību pat tad, ja tie nebūs redzējuši šos faktus apmācību procesa laikā, līdzīgi kā strādā nulles šāviena mašīnmācīšanās, kas tika aprakstīta iepriekšējās nodaļās. Modelis spētu iemācīties algoritmu, ar kura palīdzību var atrisināt vidi, nevis tikai sakarības pašā vidē. Lai šādu modeli apmācītu, varētu lietot arī papildu kļūdas funkcijas, kas regulētu rakstīšanas un lasīšanas galvu darbību. Nesen šajā virzienā ir publicēti pētījumi, taču problēma nav atrisināta daudzdimensiju vidēm [125], [81], [51].

Kļūdas funkcijas, lai apmācītu lasīšanas un rakstīšanas galvas ārējās atmiņas modeļiem un ziņķārības modeļiem, ir līdzīgas dziļo metriku apmācības kļūdas funkcijām, piemēram, trijotnes kļūdas funkcija un kontrastējoša kļūdas funkcija. Dziļo metriku apmācība nulles šāviena, viena šāviena un k-šāviena apmācībām arī nav vēl atrisināta problēma. Šo pētījumu varētu papildināt ar *KL* (*Kullback-Leibler*) diverģences funkciju vai kādu citu funkciju, as novērtē varbūtības blīvumu attiecībā pret iepriekš definētiem varbūtīgajiem sadalījumiem. Nereti kombinācijā ar *KL* izmanto normālo sadalījumu ar apmācāmu vidēju un standartnovirzes vērtību. Vēl papildus tam būtu vēlams pievienot arī rekonstrukcijas kļūdas funkciju, ko var iegūt no *VAE* tipa modeļiem vai arī diskriminatora kļūdas funkciju, izmantojot *GAN* tipa modeļus, kas nereti strādā labāk nekā vienkārši autokodētāju modeļi.

Vēl eksponenciālo trijotnes kļūdas funkciju varētu izmantot arī pirmsapmācības uzdevumiem un nepārraudzītai apmācībai, lai bez marķētāju palīdzības paplašinātu datu kopas vai iekodētu daudzdimensiju ievades

datus latentajos vektorus laika rindu vai cita vieda modeļiem.

Nesen zinātniskajā literatūrā publicētas jaunas kļūdas funkcijas, piemēram, *CapsNet* robežu kļūdas funkcija (kapsulu tīkliem) [89] un fokālā kļūdas funkcija [61]. Šīs funkcijas netika iegūtas no informācijas teorijas kā, piemēram, krusta-entropijas kļūdas funkcija, bet gan tās ir iegūtas eksperimentālā ceļā, un rezultātos tās pārspēj klasiskās kļūdas funkcijas. Abām kļūdas funkcijām ir forma kļūdas telpā, kas palīdz tām (atkarībā no parametriem) ātrāk konverģēt. Šie pētījumi liecina par to, ka gan klasifikācijas uzdevumos, gan citos dziļās mašīnmācīšanās uzdevumos, visticamāk, vēl var atrast labākas kļūdas funkcijas, kur to formai varētu būt liela nozīme.

7. Secinājumi

Promocijas darbā tika izveidotas un pārbaudītas jaunas kļūdas funkcijas, modeļu arhitektūras un apmācību algoritmi. Tajā parādīta funkciju formas nozīme dziļajā mašīnmācīšanā.

Šī darba jaunievedumi iekļauj *UNet-RNN-Skip* modeli, *OccupancyMapGenerator* datu kopas ģeneratoru, *MDQN* kļūdas funkciju, Q-vērtību kartes, eksponenciālo trijotnes kļūdas funkciju, *Unit-Range* latentās telpas normalizācijas funkciju, *Unit-Bounce* latentās telpas normalizācijas funkciju un citas metodes, kas ir publicētas zinātniskajā literatūrā un pievienotas pielikumos. Eksperimentālie rezultāti parāda, ka jaunās metodes iegūst labākus rezultātus nekā populārākās dziļās mašīnmācīšanās metodes katrā problēmsfērā, kur tās tika testētas.

Sejas reidentifikācijas uzdevumam un dziļo metriku apmācības uzdevumam izmantojot *VGGFace2* datu kopu, eksponenciālās trijotnes kļūdas funkcija ieguva labākus rezultātus, sasniedzot 85,7 % precizitāti ar nulles šāviena metodi. Eksponenciālā trijotnes kļūdas funkcija konverģē ātrāk nekā citas trijotnes kļūdas funkcijas. *Unit-Range* normalizācijas funkcija un *Unit-Bounce* normalizācijas funkcija nodrošina labāku latentās telpas sadalījumu nekā *L2* normalizācijas funkcija, un jaunizveidotajām normalizācijas funkcijām ir līdzīgas īpašības kā kosinusa distancei, tikai Eiklīda telpā. Promocijas darbam pievienota papildu nodaļa ar padziļinātu literatūras apskatu dziļo metriku apmācībai un to kļūdas funkcijām. Literatūras apskats parāda, ka jaunizveidotās kļūdas funkcijas šajā lietojumā ir aktīva pētījumu tēma, kas vēl nav pilnībā atrisināta un turpina attīstīties vairāk nekā 30 gadu garumā.

Dziļās stimulētās mašīnmācīšanās uzdevumam dažādās datorspēļu

vidēs *MDQN* kļūdas funkcija iegūst lielākas kumulatīvās punktu summas par *DDQN* un *DQN* kļūdas funkcijām. Tā nodrošina arī funkcionalitāti, lai varētu izveidot Q-vērtību kartes, kas ļauj ieskatīties modeļa politikā un lēmumu pieņemšanas procesā, izmantojot vizuālu politikas reprezentāciju katram stāvoklim. Tika noskaidrots arī tas, ka ir nepieciešams veikt vismaz 20 atkārotus eksperimentus ar vieniem un tiem pašiem hiperparametriem *MDQN* un citām kļūdas funkcijām, lai novērstu nejaušu svaru inicializācijas ietekmi vidēs kā *PLE*, kurās liela nozīme ir nejaušībai. *MDQN* kļūdas funkcijas pētījums ietver literatūras apskatu, salīdzinājumus un analīzi saistībā ar Q-vērtību kļūdas funkcijām, kas publicēšanas brīdī uzrādīja labākos rezultātus. Literatūras analīze Q-vērtību kļūdas funkciju gadījumā parāda, ka šī vēl joprojām ir aktīva pētījumu tēma, kas nav vēl līdz galam atrisināta.

Vēl šajā promocijas darbā tika radīts *UNet-RNN-Skip* modelis, kas modelē vērtību funkciju un sniedz par kārtu ātrāku izpildes ātrumu, salīdzinot ar klasisko *VI* algoritmu. Tas iegūst tādu pašu politiku kā VI algoritms 99,8% gadījumu, un var tikt apmācīts uz 32x32 kartēm, bet pēc tam izmantots uz lielākām 256x256 kartēm. Lai iegūtu datu kopas aizņemtības režģa uzdevumiem, var izmantot arī jaunizstrādāto *OccupancyMapGenerator* karšu ģeneratoru. Šīs datu kopas var izmantot *VI* algoritma tipa uzdevumiem, kā arī ar to nesaistītiem uzdevumiem, piemēram, *SLAM* uzdevumiem.

Visas tēzes, kas definētas 1.3. nodaļā, tika pierādītas.

Jaunās *MDQN* un eksponenciālās trijotnes kļūdas funkcijas jau veiksmīgi tiek lietotas komerciālos produktos analītiskās ķīmijas uzdevumu veikšanā SIA *ChromSword* un biometrisko datu reidentifikācijas uzdevumam SIA *Asya*. Vēl jaunās kļūdas funkcijas var izmantot arī dažādiem citiem praktiskiem lietojumiem, kas aprakstīti 5.4. nodaļā. Lietojumi iekļauj nulles šāviena uzdevumus ne tikai biometriskajiem datiem, bet arī, piemēram, līdzīgu produktu atrašanai, izmantojot fotoattēlu. Vēl tās var lietot ceļu plānošanā, analītiskajā ķīmija, runas modeļos, stimulētajā mašīnapmācībā un robotu vadības uzdevumos. Pētījums uzsver jaunu kļūdas funkciju efektivitāti dziļās mašīnmācīšanās uzdevumos, un šīs funkcijas ir nevis izvestas no esošas matemātikas teorijas, bet atrastas, izmantojot empīriskas metodes.

LITERATŪRA

[1] Sercan Ö. Arik u. c. "Neural Voice Cloning with a Few Samples". *ArXiv* abs/1802.06006 (2018).

[2] Kai Arulkumaran u. c. "A Brief Survey of Deep Reinforcement Learning". *ArXiv* abs/1708.05866 (2017).

[3] Jimmy Ba, J. Kiros un Geoffrey E. Hinton. "Layer Normalization". *ArXiv* abs/1607.06450 (2016).

[4] Mohammad Babaeizadeh u. c. "GA3C: GPU-Based A3C for Deep Reinforcement Learning". *CoRR* abs/1611.06256 (2016). URL: `http://arxiv.org/abs/1611.06256`.

[5] H. Bredin. "TristouNet: Triplet Loss for Speaker Turn Embedding". *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017), 5430.—5434. lpp.

[6] Greg Brockman u. c. "OpenAI Gym". *ArXiv* abs/1606.01540 (2016).

[7] J. Bromley u. c. "Signature Verification Using a "Siamese" Time Delay Neural Network". *Int. J. Pattern Recognit. Artif. Intell.* 1993.

[8] Jane Bromley u. c. "Signature Verification Using A "Siamese" Time Delay Neural Network". *IJPRAI* 7.4 (1993), 669.—688. lpp. DOI: `10.1142/S0218001493000339`. URL: `https://doi.org/10.1142/S0218001493000339`.

[9] Edward De Brouwer u. c. "GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series". *ArXiv* abs/1905.12374 (2019).

[10] T. Brown u. c. "Language Models Are Few-Shot Learners". *ArXiv* abs/2005.14165 (2020).

[11] Yuri Burda u. c. "Exploration by Random Network Distillation". *ArXiv* abs/1810.12894 (2019).

[12] Yuri Burda u. c. "Large-Scale Study of Curiosity-Driven Learning". *ArXiv* abs/1808.04355 (2019).

[13] Qiong Cao u. c. "VGGFace2: A Dataset for Recognising Faces across Pose and Age". *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)* (2017), 67.—74. lpp.

[14] Sumit Chopra, Raia Hadsell un Yann LeCun. “Learning a Similarity Metric Discriminatively, with Application to Face Verification”. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. 2005, 539.—546. lpp. DOI: 10.1109/CVPR.2005.202. URL: https://doi.org/10.1109/CVPR.2005.202.

[15] Gregory Cohen u. c. “EMNIST: An Extension of MNIST to Handwritten Letters”. *ArXiv* abs/1702.05373 (2017).

[16] J. Cong u. c. “Data Efficient Voice Cloning from Noisy Samples with Domain Adversarial Training”. *ArXiv* abs/2008.04265 (2020).

[17] Haowen Deng, Tolga Birdal un Slobodan Ilic. “PPFNet: Global Context Aware Local Features for Robust 3D Point Matching”. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 195.—205. lpp.

[18] E. Dijkstra. “A Note on Two Problems in Connexion with Graphs”. *Numerische Mathematik* 1 (1959), 269.—271. lpp.

[19] E. Dupont. “Learning Disentangled Joint Continuous and Discrete Representations”. *NeurIPS*. 2018.

[20] T. Ehret u. c. “Model-Blind Video Denoising via Frame-to-Frame Training”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 11361.—11370. lpp.

[21] Lasse Espeholt u. c. “IMPALA: Scalable Distributed Deep-Rl with Importance Weighted Actor-Learner Architectures”. *ArXiv* abs/1802.01561 (2018).

[22] Chelsea Finn, S. Levine un P. Abbeel. “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization”. *ArXiv* abs/1603.00448 (2016).

[23] Florian Schroff, Dmitry Kalenichenko un James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering”. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2015, 815.—823. lpp. DOI: 10.1109/CVPR.2015.7298682. URL: https://doi.org/10.1109/CVPR.2015.7298682.

[24] S. Galushko u. c. “ChromSword: Software for Method Development in Liquid Chromatography”. 2018.

[25] Yuan Gao un Dorota Glowacka. "Deep Gate Recurrent Neural Network". *ArXiv* abs/1604.02910 (2016).

[26] Tarun Gogineni u. c. "TorsionNet: A Reinforcement Learning Approach to Sequential Conformer Search". *ArXiv* abs/2006.07078 (2020).

[27] Jacob Goldberger u. c. "Neighbourhood Components Analysis". *Advances in Neural Information Processing Systems*. Izdevis L. Saul, Y. Weiss un L. Bottou. 17. sējums. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf.

[28] Klaus Greff u. c. "LSTM: A Search Space Odyssey". *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017. g. okt.), 2222.—2232. lpp. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2016.2582924. arXiv: 1503.04069. URL: http://arxiv.org/abs/1503.04069 (aplūkots 25.04.2019.).

[29] Hado van Hasselt, Arthur Guez un David Silver. "Deep Reinforcement Learning with Double Q-Learning". *CoRR* abs/1509.06461 (2015). URL: http://arxiv.org/abs/1509.06461.

[30] Hado V. Hasselt. "Double Q-Learning". *Advances in Neural Information Processing Systems 23*. Izdevis J. D. Lafferty u. c. Curran Associates, Inc., 2010, 2613.—2621. lpp. URL: http://papers.nips.cc/paper/3964-double-q-learning.pdf.

[31] Kaiming He u. c. "Deep Residual Learning for Image Recognition". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770.—778. lpp.

[32] Kaiming He u. c. "Identity Mappings in Deep Residual Networks". *ArXiv* abs/1603.05027 (2016).

[33] Alexander Hermans, Lucas Beyer un Bastian Leibe. "In Defense of the Triplet Loss for Person Re-Identification". *ArXiv* abs/1703.07737 (2017).

[34] J. Hershey u. c. "Deep Clustering: Discriminative Embeddings for Segmentation and Separation". *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), 31.—35. lpp.

[35] Matteo Hessel u. c. “Rainbow: Combining Improvements in Deep Reinforcement Learning”. *ArXiv* abs/1710.02298 (2018).

[36] S. Hochreiter un J. Schmidhuber. “Long Short-Term Memory”. *Neural Computation* 9 (1997), 1735.—1780. lpp.

[37] C. Huang, Chen Change Loy un X. Tang. “Local Similarity-Aware Deep Feature Embedding”. *NIPS*. 2016.

[38] Gao Huang, Zhuang Liu un Kilian Q. Weinberger. “Densely Connected Convolutional Networks”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 2261.—2269. lpp.

[39] Huimin Huang u. c. “UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation”. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), 1055.—1059. lpp.

[40] Satoshi Iizuka un Edgar Simo-Serra. “DeepRemaster: Temporal Source-Reference Attention Networks for Comprehensive Video Enhancement”. *ArXiv* abs/2009.08692 (2019).

[41] Rafal Józefowicz, Wojciech Zaremba un Ilya Sutskever. “An Empirical Exploration of Recurrent Network Architectures”. *ICML*. 2015.

[42] A. Kiani, Chris Wang un Angela Xu. “Sepsis World Model: A MIMIC-Based OpenAI Gym ”World Model” Simulator for Sepsis Treatment”. *ArXiv* abs/1912.07127 (2019).

[43] B. Kitchenham u. c. “Systematic Literature Reviews in Software Engineering - A Systematic Literature Review”. *Inf. Softw. Technol.* 51 (2009), 7.—15. lpp.

[44] Gregory R. Koch. “Siamese Neural Networks for One-Shot Image Recognition”. 2015.

[45] A. Kolesnikov u. c. “Big Transfer (BiT): General Visual Representation Learning”. *arXiv: Computer Vision and Pattern Recognition* (2019).

[46] Martin Köstinger u. c. “Large Scale Metric Learning from Equivalence Constraints”. *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), 2288.—2295. lpp.

[47] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. 2009.

[48] Alex Krizhevsky, Ilya Sutskever un Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. *Communications of the ACM* 60.6 (2017. g. 24. maijs), 84.—90. lpp. ISSN: 00010782. DOI: 10.1145/3065386. URL: http://dl.acm.org/citation.cfm?doid=3098997.3065386 (aplūkots 25.04.2019.).

[49] David Krueger u. c. “Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations”. *ArXiv* abs/1606.01305 (2017).

[50] Katya Kudashkina, P. Pilarski un R. Sutton. “Document-Editing Assistants and Model-Based Reinforcement Learning as a Path to Conversational AI”. *ArXiv* abs/2008.12095 (2020).

[51] Guillaume Lample u. c. “Large Memory Layers with Product Keys”. *ArXiv* abs/1907.05242 (2019).

[52] M. Law, N. Thome un M. Cord. “Quadruplet-Wise Image Similarity Learning”. *2013 IEEE International Conference on Computer Vision* (2013), 249.—256. lpp.

[53] M. Law, R. Urtasun un R. Zemel. “Deep Spectral Clustering Learning”. *ICML*. 2017.

[54] Quoc V. Le, Navdeep Jaitly un Geoffrey E. Hinton. “A Simple Way to Initialize Recurrent Networks of Rectified Linear Units”. *ArXiv* abs/1504.00941 (2015).

[55] Yann LeCun un Corinna Cortes. “MNIST Handwritten Digit Database”. (2010). URL: http://yann.lecun.com/exdb/mnist/ (aplūkots 14.01.2016.).

[56] Yann LeCun u. c. “Comparison of Learning Algorithms for Handwritten Digit Recognition”. 1995.

[57] H. Lee u. c. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”. *ICML ’09*. 2009.

[58] J. Lee u. c. “Dynamic Noise Embedding: Noise Aware Training and Adaptation for Speech Enhancement”. 2020.

[59] Chao Li u. c. “Deep Speaker: An End-to-End Neural Speaker Embedding System”. *CoRR* abs/1705.02304 (2017). arXiv: 1705.02304. URL: http://arxiv.org/abs/1705.02304.

[60] Timothy P. Lillicrap u. c. “Continuous Control with Deep Reinforcement Learning”. *CoRR* abs/1509.02971 (2015). URL: http://arxiv.org/abs/1509.02971.

[61] Tsung-Yi Lin u. c. “Focal Loss for Dense Object Detection”. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2999.—3007. lpp.

[62] Teng Long u. c. “Zero-Shot Learning via Discriminative Representation Extraction”. *Pattern Recognition Letters* 109 (2018), 27.—34. lpp.

[63] Scott Lundberg un Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. *NIPS*. 2017.

[64] R. Manmatha u. c. “Sampling Matters in Deep Embedding Learning”. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2859.—2867. lpp.

[65] Michael Phi. *Illustrated Guide to LSTM’s and GRU’s: A Step by Step Explanation*. 2020. g. 6. janv. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[66] Erik G. Miller, Nicholas E. Matsakis un Paul A. Viola. “Learning from One Example through Shared Densities on Transforms”. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)* 1 (2000), 464—471 vol.1.

[67] Volodymyr Mnih u. c. “Asynchronous Methods for Deep Reinforcement Learning”. *ICML* 48 (2016), 1928.—1937. lpp. URL: http://arxiv.org/abs/1602.01783.

[68] Volodymyr Mnih u. c. “Playing Atari with Deep Reinforcement Learning”. *CoRR* abs/1312.5602 (2013). URL: http://arxiv.org/abs/1312.5602.

[69] Igor Mordatch. “Concept Learning with Energy-Based Models”. *ICLR*. 2018.

[70] Yair Movshovitz-Attias u. c. “No Fuss Distance Metric Learning Using Proxies”. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 360.—368. lpp.

[71] “N-Shot Learning: Learning More with Less Data”. (). URL: https://blog.floydhub.com/n-shot-learning/.

[72] D. Neil, M. Pfeiffer un Shih-Chii Liu. “Phased LSTM: Accelerating Recurrent Network Training for Long or Event-Based Sequences”. *NIPS*. 2016.

[73] Binh X. Nguyen u. c. “Deep Metric Learning Meets Deep Clustering: An Novel Unsupervised Approach for Feature Embedding”. *ArXiv* abs/2009.04091 (2020).

[74] Hyeonwoo Noh, Seunghoon Hong un Bohyung Han. “Learning Deconvolution Network for Semantic Segmentation”. *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), 1520.—1528. lpp.

[75] “Non-Zero Initial States for Recurrent Neural Networks - R2RT”. (). URL: https://r2rt.com/non-zero-initial-states-for-recurrent-neural-networks.html.

[76] OpenAI u. c. “Solving Rubik’s Cube with a Robot Hand”. *ArXiv* abs/1910.07113 (2019).

[77] OpenAI u. c. “Solving Rubik’s Cube with a Robot Hand”. *ArXiv* abs/1910.07113 (2019).

[78] D. Park u. c. “Improved Noisy Student Training for Automatic Speech Recognition”. *ArXiv* abs/2005.09629 (2020).

[79] Adam Paszke u. c. “Automatic Differentiation in PyTorch”. (2017).

[80] K. Petersen u. c. “Systematic Mapping Studies in Software Engineering”. *EASE*. 2008.

[81] Alexander Pritzel u. c. “Neural Episodic Control”. *ArXiv* abs/1703.01988 (2017).

[82] Qi Qi u. c. “A Simple and Effective Framework for Pairwise Deep Metric Learning”. *Computer Vision — ECCV 2020*. Izdevis Andrea Vedaldi u. c. Cham: Springer International Publishing, 2020, 375.—391. lpp. ISBN: 978-3-030-58583-9.

[83] Hubert Ramsauer u. c. “Hopfield Networks Is All You Need”. *ArXiv* abs/2008.02217 (2020).

[84] Rajeev Ranjan, Carlos D. Castillo un Rama Chellappa. “L2-Constrained Softmax Loss for Discriminative Face Verification”. *CoRR* abs/1703.09507 (2017). arXiv: 1703.09507. URL: http://arxiv.org/abs/1703.09507.

[85] Mirco Ravanelli, Titouan Parcollet un Yoshua Bengio. "The Pytorch-Kaldi Speech Recognition Toolkit". *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), 6465.—6469. lpp.

[86] Oren Rippel u. c. "Metric Learning with Adaptive Density Discrimination". *ICLR* abs/1511.05939 (2016).

[87] Olaf Ronneberger, Philipp Fischer un Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". *MICCAI*. 2015.

[88] Stuart J. Russell un Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2002. g. dec. ISBN: 0-13-790395-2. URL: `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0137903952`.

[89] Sara Sabour, Nicholas Frosst un Geoffrey E. Hinton. "Dynamic Routing between Capsules". *ArXiv* abs/1710.09829 (2017).

[90] Tom Schaul u. c. "Prioritized Experience Replay". *CoRR* abs/1511.05952 (2015). URL: `http://arxiv.org/abs/1511.05952`.

[91] John Schulman u. c. "Proximal Policy Optimization Algorithms". *ArXiv* abs/1707.06347 (2017).

[92] John Schulman u. c. "Trust Region Policy Optimization". *ICML*. 2015.

[93] Ramprasaath R. Selvaraju u. c. "Grad-CAM: Why Did You Say That? Visual Explanations from Deep Networks via Gradient-Based Localization". *CoRR* abs/1610.02391 (2016). URL: `http://arxiv.org/abs/1610.02391`.

[94] Stanislau Semeniuta, Aliaksei Severyn un Erhardt Barth. "Recurrent Dropout without Memory Loss". *COLING*. 2016.

[95] M. Shoeybi u. c. "Megatron-Lm: Training Multi-Billion Parameter Language Models Using Model Parallelism". *ArXiv* abs/1909.08053 (2019).

[96] Ari Silburt u. c. "Lunar Crater Identification via Deep Learning". *Icarus* 317 (2019), 27.—38. lpp.

[97] Karen Simonyan un Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *CoRR* abs/1409.1556 (2015).

[98] Kihyuk Sohn. “Improved Deep Metric Learning with Multi-Class n-Pair Loss Objective”. *NIPS*. 2016.

[99] Hyun Oh Song u. c. “Deep Metric Learning via Facility Location”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 2206.—2214. lpp.

[100] Hyun Oh Song u. c. “Deep Metric Learning via Lifted Structured Feature Embedding”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 4004.—4012. lpp.

[101] Hyun Oh Song u. c. “Learnable Structured Clustering Framework for Deep Metric Learning”. *ArXiv* abs/1612.01213 (2016).

[102] Christian Szegedy, Sergey Ioffe un Vincent Vanhoucke. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”. *AAAI*. 2016.

[103] Haoran Tang u. c. “#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning”. *ArXiv* abs/1611.04717 (2017).

[104] Norman Tasfi. “PyGame Learning Environment”. *GitHub repository* (2016). URL: `https://github.com/ntasfi/PyGame-Learning-Environment`.

[105] Matias Tassano, J. Delon un T. Veit. “FastDVDnet: Towards Real-Time Deep Video Denoising without Flow Estimation”. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), 1351.—1360. lpp.

[106] “Tips for Training Recurrent Neural Networks”. (). URL: `https://danijar.com/tips-for-training-recurrent-neural-networks/`.

[107] Hugo Touvron u. c. “Fixing the Train-Test Resolution Discrepancy”. *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[108] “Triplet Loss and Online Triplet Mining in TensorFlow | Olivier Moindrot Blog”. (). URL: `https://omoindrot.github.io/triplet-loss`.

[109] E. Urtans un Agris Nikitenko. “Survey of Deep Q-Network Variants in PyGame Learning Environment”. *ICDLT ’18*. 2018.

[110] E. Urtans, Agris Nikitenko un Valters Vecins. “Exponential Triplet Loss”. *Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis* (2020).

[111] E. Urtans un Valters Vecins. “Value Iteration Solver Networks”. *2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS)* (2020), 8.—13. lpp.

[112] Evalds Urtans un Ariel Tabaks. *Asya: Mindful Verbal Communication Using Deep Learning*. 2020. arXiv: `2008.08965 [eess.AS]`.

[113] E. Ustinova un V. Lempitsky. “Learning Deep Embeddings with Histogram Loss”. *NIPS*. 2016.

[114] R. Vaillant, C. Monrocq un Y. Le Cun. “Original Approach for the Localisation of Objects in Images”. *IEE Proceedings - Vision, Image and Signal Processing* 141.4 (1994), 245.—250. lpp.

[115] Athanasios Voulodimos u. c. “Deep Learning for Computer Vision: A Brief Review”. *Computational Intelligence and Neuroscience* 2018 (2018).

[116] Chanchin Wang, Xue Zhang un Xipeng Lan. “How to Train Triplet Networks with 100K Identities?”: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (2017), 1907.—1915. lpp.

[117] J. Wang u. c. “Deep Metric Learning with Angular Loss”. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2612.—2620. lpp.

[118] Jinjiang Wang u. c. “Deep Learning for Smart Manufacturing: Methods and Applications”. *Journal of Manufacturing Systems* 48 (2018), 144.—156. lpp.

[119] X. Wang u. c. “Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 5017.—5025. lpp.

[120] Xinrui Wang un Jinze Yu. “Learning to Cartoonize Using White-Box Cartoon Representations”. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), 8087.—8096. lpp.

[121] Xinshao Wang u. c. “Ranked List Loss for Deep Metric Learning”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 5202.—5211. lpp.

[122] Ziyu Wang, Nando de Freitas un Marc Lanctot. “Dueling Network Architectures for Deep Reinforcement Learning”. *CoRR* abs/1511.06581 (2015). URL: http://arxiv.org/abs/1511.06581.

[123] Ziyu Wang, Nando de Freitas un Marc Lanctot. “Dueling Network Architectures for Deep Reinforcement Learning”. *CoRR* abs/1511.06581 (2015). URL: http://arxiv.org/abs/1511.06581.

[124] Ziyu Wang u. c. “Sample Efficient Actor-Critic with Experience Replay”. *ArXiv* abs/1611.01224 (2017).

[125] Greg Wayne u. c. “Unsupervised Predictive Memory in a Goal-Directed Agent”. *ArXiv* abs/1803.10760 (2018).

[126] “We Analyzed 16,625 Papers to Figure out Where AI Is Headed next | MIT Technology Review”. (). URL: https://www.technologyreview.com/2019/01/25/1436/we-analyzed-16625-papers-to-figure-out-where-ai-is-headed-next/.

[127] Kilian Q. Weinberger un L. Saul. “Distance Metric Learning for Large Margin Nearest Neighbor Classification”. *NIPS*. 2005.

[128] Yandong Wen u. c. “A Discriminative Feature Learning Approach for Deep Face Recognition”. *ECCV*. 2016.

[129] Cameron R. Wolfe un Keld T. Lundgaard. “E-Stitchup: Data Augmentation for Pre-Trained Embeddings”. 2019.

[130] Han Xiao, Kashif Rasul un Roland Vollgraf. “Fashion-Mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms”. *ArXiv* abs/1708.07747 (2017).

[131] Yong Xu u. c. “Dynamic Noise Aware Training for Speech Enhancement Based on Deep Neural Networks”. *INTERSPEECH*. 2014.

[132] Dong Yi, Zhen Lei un S. Li. “Deep Metric Learning for Practical Person Re-Identification”. *ArXiv* abs/1407.4979 (2014).

[133] Y. Yuan, Kuiyuan Yang un Chao Zhang. “Hard-Aware Deeply Cascaded Embedding”. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 814.—823. lpp.

[134] Q. Zhang un Tao Du. “Self-Driving Scale Car Trained by Deep Reinforcement Learning”. *ArXiv* abs/1909.03467 (2019).

[135] W. Zheng, S. Gong un T. Xiang. "Reidentification by Relative Distance Comparison". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2013), 653.—668. lpp.

[136] Zongwei Zhou u. c. "UNet++: A Nested u-Net Architecture for Medical Image Segmentation". *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support : 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain, S...* 11045 (2018), 3.—11. lpp.