Учење со поттикнување во средина на Atari игри

Мила Куч, 171074

Факултет за информатички науки и компјутерско инженерство, Скопје
Проектна задача по предметот Интелигентни Информациски Системи

Септември, 2021 година

Абстракт - Примената на вештачката интелигенција и интелигентни системи во видео игри не е нешто ново и датира уште од самите почетоци на овие технологии. Најчестата употреба на интелигентните системи во игрите е да се генерираат реактивни, адаптивни и интелигентни однесувања за ликови кои не се играни од играч (NPC) слични со човечката интелигенција. Во ова истражување ќе биде разработено учењето со поттикнување во средина на Atari игри со помош на интелигентен DQN агент, споредба на резултатите и заклучоци.

I. Вовед

Вештачката интелигенција во видео игрите е долготрајна област на истражување, која ги проучува интелигентните технологии кои се користат да се достигне перформанс на ниво слично на човечкото при играње. Генерално, ги проучува комплексните интеракции на агентите и средините на играње. Различни игри нудат интересни и комплексни проблеми кои агентите можат да ги решат, и тоа ги прави видео игрите совршена средина во истражувањата за вештачка интелигенција. Овие виртуелни околини се безбедни и можат да се контролираат. [1]

Дополнително, видео игрите нудат бесконечно снабдување со корисни податоци за алгоритмите со машинско учење. Вештачката интелигенција исто така помага и да го усовршиме начинот на кој ги играме видео игрите, да ги разбереме подобро, за тие да имаат уште подобар дизајн. [2]

Учењето со поттикнување (Reinforcement Learning) е тип на машинско учење кој се стреми да ја земе во предвид максималната награда при правење на одлуки. [3] Основните компоненти на учењето со поттикнување се агентот и средината - агентот прима фидбек во форма на обзервирање и награда по секоја акција. За да се генерира подобра политика, ќе продолжи да комуницира со средината и да ја подобрува способноста за правење одлуки додека политиката не конвергира.
Длабокото учење со поттикнување направи голем подем во видео игрите како Atari, ViZDoom, StarCraft, Dota2 итн. Конкретно за Atari игрите, тие се игри поддржани од козолата Atari 2600, и спаѓаат во класичните игри кои го имаат запечатено своето место во историјата.

Atari 2600 е домашна конзола за видео игри од Atari, Inc. [4]. Пуштена во употреба на 11 Септември 1977 година, таа е сочинета од хардвер базиран на микропроцесори и игри кои се зачувани на ROM кертриџ отколку да се физички вградени во самата конзола. Atari 2600 доаѓа со два џојстик контролери, споен пар paddle контролори, и еден кертриџ за игра: првично Combat, а подоцна и Pac-Man.

Во денешно време популарни се Atari емулаторите, кои дозволуваат старите Atari игри да бидат играни на модерните компјутери. Некои од нив се достапни и како JavaScript.

II. Историја и слични истражувања

Во последните години, учењето со поттикнување стана популарно заради Alpha Go, програма која може да победи човечки експерт во Go. [5] Во 2017, на настанот Future of Go Summit, оваа програма го шокираше светот со победа на сите три игри против Ke Jie, најдобриот играч во Go на светот. Но, истражувањето во областа на учењето со поттикнување почна уште порано од тоа.

Според Sutton, раната историја на учењето со поттикнување може да се подели на два главни дела. Еден од нив е оптималната контрола. Како метод на справување со овој проблем, во 1954 година теоријата за Динамичко Програмирање беше предложена од Белман[6]. Во теоријата, тој го предложил концептот на функционална равенка, која во денешно време се нарекува Белманова равенка.

Иако динамичкото програмирање беше еден од најефективните пристапи кога станува збор за проблеми со оптимална контрола во тоа време, пречка беа високите компутациски барања кои не се лесни да се решат. По три години, Белман го изгради моделот наречен Markov Decision Processes (MDPs) за да изгради дискретен детерминистички процес. [7] Детерминистичкиот систем и концептот од функција на вредност опишана во

Белмановата равенка се основите на модерното учење со поттикнување.

Друга добро позната приказна од денешницата на полето на учење со поттикнување е TD-gammon, програма која ја игра играта табла (backgammon).[8] Оваа програма целосно се потпира на алгоритам за учење со поттикнување и самостојно играње, и достигнува надчовечко ниво на игра. TD-gammon алгоритмот не содржи модели но е сличен на Q-учењето, каде се апроксимира функцијата на вредност преку повеќеслоен перцептрон со еден скриен слој.

III. Опис на проблемот

Генерално, вештачката интелигенција во видео игрите ја инволвира перцепцијата и правењето на одлуки во средината на игра. Со овие компоненти, има некои неизбежни предизвици и со тоа предложени решенија.

Првиот предизвик е тоа што просторот на состојби во игра е многу голем, особено во стратешки игри. Со порастот на репрезентациското учење, целиот систем успешно моделира голем простор на состојби со невронските мрежи со длабоко учење.

Вториот предизвик е учење на соодветни политики кои помагаат при правењето на одлуки во динамични непознати околини, што е тешко. За овој проблем, data-driven методи како учење со поттикнување се можни решенија.

Третиот предизвик е тоа што најголемиот дел од вештачката интелигенција на игри се развиени во специјална виртуелна околина. Да се пренесе способноста на вештачката интелигенција низ различни земји е клучен предизвик, а со тоа доаѓа и потребата за погенерален систем за учење.

Во овој проект ќе бидат опфатени неколку Atari 2600 игри кои се имплементирани во Arcade Learning Environment (ALE)[9]. Целта на овој проект е да се направи единствен агент кој е способен успешно да научи да игра што повеќе игри. Моделот не е приспособен за секоја игра посебно и не му се дадени никакви информации специфични за играта, или визуелни карактеристики - учи само од видео инпутот, наградата и терминалните сигнали, и множеството на можни акции - исто како што би можел и човечки играч. Дополнително, мрежната архитектура и сите хиперпараметри кои се користат при играње се константни низ сите игри..

Така, конкретните чекори во овој проект би биле:

1. Да се изберат соодветни средини на игрите врз кои ќе се изврши симулацијата
2. Градење на универзален интелигентен DQN агент за сите игри
3. Симулации и резултати
4. Споредба на перформансот и заклучоци

IV. Одбирање на средина

OpenAI Gym [10] е алатка која нуди широк опсег на симулирани средини (Atari игри, друштвени игри, 2D и 3D физички симулации), со цел тренирање на агенти, нивна споредба, или за развивање на нови алгоритми за машинско учење (учење со поттикнување).
OpenAI Gym поддржува 118 средини на Atari игри [11], со по неколку различни верзии на истите.

Atari средините се поразлични од игрите како на пр. cart-poll. Атари игрите типично ги користат нивните 2D дисплеи како состојба во средината. Gym ги репрезентира Atari игрите како 3D(висина по ширина по боја) состојба базиран на екранот, или векторот кој ја репрезентира состојбата на компјутерскиот RAM на играта. За да се претпроцесираат Atari игрите за подобра компутационална ефикасност, генерално треба да се скокнат и некои рамки, да се намали резолуцијата и да се отстрани информацијата за боја.

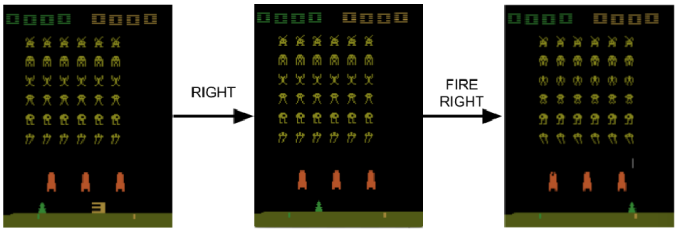
За овој проект ќе се опфатат вкупно 3 од најпознатите игри, а за истите следуваат повеќе информации.

A. Space invaders

Space Invaders[12] е аркадна игра со пукање од 1978 година, развиена во Јапонија, а била првата игра со пукање со фиксиран стрелец, и понатаму станува урнек за ваквиот тип на игри. Целта на играта е да се победи секој бран на вонземјани кои се спуштаат, и со тоа да се заработат што е можно повеќе поени.

Играчот може да го поместува ласерскиот топ на хоризонтална оска на долниот дел на екранот, и со тоа да пука нагоре. Вонземјаните почнуваат како пет реда од единаесет единки кои се движат лево и десно на екранот, и се поместуваат надолу секој пат кога ќе го допрат левиот или десниот раб на екранот. Играчот има три животи, а играта веднаш завршува ако вонземјаните го допрат долниот дел од екранот. Дополнително, вонземјаните се обидуваат да го уништат топот на играчот преку пукање на проектили, а играчот е делумно заштитен од статични бункери кои се полека уништувани од вонземјаните.

Како што се уништуваат вонземјаните, нивото движење се забрзува, а уништувањето на сите вонземјани носи уште еден бран кој започнува подолу. Исто така, доколку играчот уништи специјални мистериозни бродови, добива бнус поени. Пример за состојбите и акциите на оваа игра може да се види на Сл.1



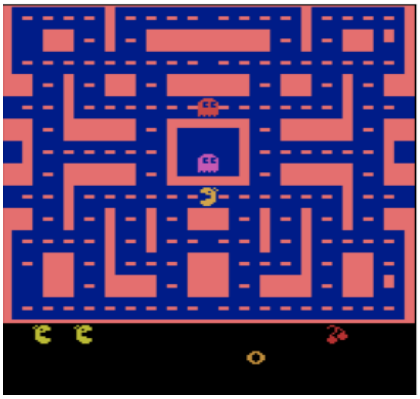
Сл.1 Space Invaders, каде секоја слика е состојба, а секоја стрелка измеѓу состојбите покажува некоја акција.

Во склоп на Gym, 'Space-Invaders-v0' средината е RGB слика од екранот, што е низа со форма на (210, 160, 3). Секоја акција е константно изведувана за траење од k рамки, каде k е униформен примерок од \{2, 3, 4\}{2,3,4}.[13]
Играта е симулирана од Arcade Learning Environment [ALE] кој го користи Stella[14] Atari емулаторот.

Б. Ms. Pacman

Ms. Pac-Man [15] е видео игра во форма на лавиринт развиена и објавена од Atari, Inc во 1982 година, како адаптација на аркадната игра со исто име која излегла две години порано. Играчот го контролира ликот Ms.Pac-Man, слична на оригиналниот Pac-Man, која се обидува да консумира одредени објекти на екранот, додека ги одбегнува четирите духови кои ја бркаат. Доколку играчот изеде објекти кои се во ќошињата на екранот, тоа ги прави духовите сини и бегаат, што му дава на играчот бонус поени.

Ms. Pac-Man може да се движи во склоп на лавиринтот, и тоа во 4 насоки - лево, десно, горе и долу.
Секоја игра започнува со четири животи, но ако еден од духовите ја допре Ms.Pac-Man, таа губи живот, а кога сите животи се изгубени, играта завршува. Пример за изгледот на играта може да се најде на Сл.2.


Сл.2 Изглед на играта Ms. Pacman во Atari средина

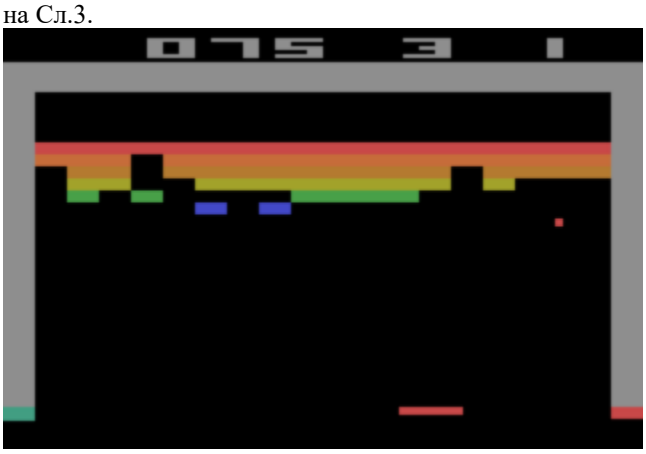
Во склоп на Gym, 'MsPacman-v0' средината ги има истите карактеристики како и 'Space-Invaders-v0' околината.

В. Breakout

Breakout [16] е аркадна игра развиена и објавена од Atari, Inc., во 1976 година, а црпи инспирација од аркадната игра Pong.

Breakout започнува со осум редови на цигли, секој ред со различна боја и соодветен број на поени. Користејќи едно топче, играчот треба да погоди што е можно повеќе цигли, со помош на ѕидовите и палката на долната страна на екранот, и со тоа да ги елиминира.

Ако палката не го погоди топчето и тоа падне, се губи еден потег, а играчот има вкупно три потега за да се отстранат два екрана на цигли. Дополнително, палката се смалува при последниот ред, а топчето се забрзува на специфични интервали.Пример за изгледот на играта може да се најде на Сл.3.


Сл.3 Изглед на играта Breakout во Atari средина

Во склоп на Gym, 'Breakout-v4' средината ги има истите карактеристики како и 'Space-Invaders-v0' околината.

V. Интелигентен агент

При градењето на интелигентен агент, есенцијално е да се разбере Q-учењето. Тоа претставува основна техника на која се базирани алгоритмите за учење со поттикнување. [17]

Q-учењето работи на начин на градење на табела која сугерира акција за секоја можна состојба. Овој пристап има неколку проблеми - прво, средината обично е составена од непрекината низа од броеви, која резултира во бесконечен број на состојби. Спротивно од тоа, Q-учењето примарно прима дискретни акции, како што е притискање на џојстик горе или долу. Така, како метод за справување со овие непрекинати состојби се користи т.н. binning - ставање на нумеричките вредности во опсег.

Ова учење е основата на интелигентниот DQN агент кој ќе се користи во овој проект.

А.DQN Агент

DQN агентот може да биде искористен во каква било средина која има дискретен простор на акции. Во суштина, DQN агентот е Q-мрежа, невронска мрежа која може да предвиди Q-вредности (очекувани вредности на излез) за сите акции, при обзервација од средината.[18]
Агентите од ваков тип можат да креираат политика за големи простори на состојба, без да треба да ја репрезентираат секоја комбинација на состојба и акција. Невронските мрежи можат да генерализираат и да учат кои се заедничките фактори на нешто.

DQN мрежата ја мапира состојбата во невроните на инпут, и Q-вредноста на акцијата во невроните на излез. DQN ефективно станува функција која прима состојба и сугерира акција со тоа што ќе ја врати очекуваната награда за сите можни акции.

Во овој проект, од библиотеката Tensorflow, се инстанцира DQN агент преку tf_agents.agents.dqn.dqn_agent.[19] Потребни се неколку компоненти, како:

- QNetwork
- time_step_spec
- action_spec
- optimizer
- loss function
- integer step counter.

За Q-мрежата, во овој проект е искористена класата AtariCategoricalQNetwork, од примерите на TF-Agents, како wrapper класа за регуларната Q-network класа. Ова ни обезбедува дека вредноста на пикселите од Atari екранот се поделени со 255, за да се нормализираат вредностите на пикселите помеѓу 0 и 1. Ова би и помогнало на самата невронска мрежа. Другите компоненти ќе бидат објаснети подолу.

Б.Политика

Политиката го дефинира начинот на кој агентот се однесува во средината. Типично, целта на учењето со поттикнување е да се тренира моделот во позадина се додека политиката не го продуцира посакуваниот резултат.

Агентите се сочинети од две политики:

- agent.policy — Главната политика која се користи за евалуација и распоредување
- agent.collect_policy — Втора политика која се користи за собирање на податоци

Политиката може да се креира и независно од агентите; На пример, може да се користи tf_agents.policies.random_tf_policy да се креира политика која по случаен избор ќе избере акција за секој временски чекор.

В.Хиперпараметри

Пред се, треба да се постават неколку параметри за самиот DQN агент; како на пр. број на итерации, максимален број на рамки при епизода, итн. Но, уште поважно, треба да се постават хиперпараметрите за невронската мрежа (која претходно се спомена, AtariCategoricalQNetwork), а тие се fc_layer_params и conv_layer_params

Бидејќи оваа мрежа е конволуциска невронска мрежа, таа ги прима двата параметри кои ја дефинираат. Конволуциските невронски мрежи обично се составени од неколку пара на конволуциски и max-pooling слоеви, кои кулминираат во еден или повеќе целосно поврзани (dense) слоеви.

Поедноставниот од двата параметри е fc_layer_params, кој ја посочува големината на секој од целосно поврзаните слоеви. Секоја торка од листата ја одредува големината на секој од слоевите. Во овој проект, поставен е еден слој од 512 неврони.

Вториот параметар, наречен conv_layer_params, е листа од параметри на конволуциски слој, каде секој елемент е торка со должина 3 и ги посочува филтерите, големината на кернелот и чекорот. Во овој проект, поставени се три конволуциски слоја во форма на: ((32, (8, 8), 4), (64, (4, 4), 2), (64, (3, 3), 1)). Овие параметри се поставени со овие вредности следејќи неколку различни туторијали, за најдобар резултат.
Важно е да се напомене дека Q-мрежата која е дефинирана тука не е агентот, туку е користена од DQN агентот да се имплементира вистинската невронска мрежа. Ваквиот пристап дозволува флексибилност при работа и резултатите.

Г.Оптимизатор

Следно, треба да се дефинира оптимизатор. За овој проект, го искористив AdamOptimizer[20] како класичен избор, но друг популарен избор е RMSPropOptimizer. Адам е алгоритам за оптимизација кој се користи наместо класичната стохастичка постапка со спуштање на градиенти, за да се ажурираат тежините на мрежата базирани врз податоците за тренинг.

Д.Метрики и евалуација

Има многу различни начини да се измери ефективноста на модел кој е трениран со учење со поттикнување. Функцијата на загуба во интерната Q-мрежа не е добра мерка за целосниот DQN алгоритам и неговите резултати. Затоа, треба експлицитно да се наведе функцијата на загуба, а во овој проект е имплементирана како средната вредност на награда која се прима низ неколку епизоди, за да се види дали нејзината вредност е максимизирана.

Ѓ. Replay buffer

DQN работи со тренирање на невронска мрежа за да се предвидат Q-вредностите за секоја можна состојба во средината. На невронската мрежа и требаат податоци за тренирање, за алгоритмот да ги акумулира податоците за тренинг како што течат епизодите. Тоа може да се постигне со т.н. replay buffer, каде се чуваат податоците. Само податоците од најновите епизоди се сочувани, податоците од постарите епизоди се исклучуваат од редицата како што таа прима нови информации. Во овој проект, искористена е класа од Tensorflow со име tf_uniform_replay_buffer.TFUniformReplayBuffer.

E. Random Collection

Тренингот на DQN моделот не може да почне со празен reply buffer; па затоа, треба да има дел од кодот кој ќе генерира иницијални податоци за тренинг преку предефиниран број на чекори.

Ж. Тренирање на DQN

По завршувањето со сите други чекори, DQN агентот е подготвен да се тренира. Процесот може да трае и до неколку часа зависно од бројот на епизоди, па затоа во овој проект го поставив на 3000 за оптимален резултат без премногу чекање. Како што се одвива тренирањето, средната награда треба да се зголеми.

З. Видео

Со трениранот модел, може да се видат резултатите во видео форма т.е. како агентот ја игра играта. Со помош на предефинираните функции embed_mp4 и create_policy_eval_video од туторијалот на Tensorflow[21], може да го следиме агентот и неговите резултати.

VI. Симулации и резултати

Во склоп на овој проект, ќе бидат вклучени два типа на споредби. Примарно, треба да се истакне споредбата на трениранот агент со поттикнато учење на соодветна игра, наспроти агент со политика по случаен избор, за да се увиди дали трениранот агент поспособно ја врши задачата од случаен агент.

A. Space invaders

Резултатите од симулацијата на играта Space Invaders се прикажани во пет серии со по три животи. Резултатите од трениранот DQN агент се прикажани на Табела 1. Најуспешната серија на првиот агент е последната, петта серија, со финален резултат од 230. Со тек на време, се забележува дека резултатот расте, со 145 во првата серија, а 230 во последната, што значи учењето со поттикнување е успешно во овој случај.

Серија/ Животи	Прва	Втора	Трета	Четврта	Петта
3	5	0	95	90	65
2	75	60	25	15	140
1	70	135	25	60	25
Вкупно	145	195	145	165	230

Табела 1. Резултати од симулација на трениран DQN агент на Space Invaders

Резултатите од симулација на агент со рандом политика се прикажани на Табела 2. Во оваа симулација немаме постепено подобрување на резултатите како што се забележува во претходната, а највисокиот резултат е 165. Со споредба се забележува разликата во резултатите помеѓу двата агента, што значи дека агентот успешно се приспособил кон проблемот за да го надмине агентот со рандом политика.

Серија/ Животи	Прва	Втора	Трета	Четврта	Петта
3	15	75	110	0	35
2	25	50	5	20	10
1	20	35	50	30	35
Вкупно	60	160	165	50	80

Табела 2. Резултати од симулација на агент со рандом политика на Space Invaders

Б.Ms. Pacman

Резултатите од симулацијата на играта Ms.Pacman се прикажани исто така во пет серии со по три животи. Резултатите од трениранот DQN агент се прикажани на Табела 3, а резултатите од симулација на агент со рандом политика се прикажани на Табела 4.

Серија/ Животи	Прва	Втора	Трета	Четврта	Петта
3	160	160	160	160	150
2	680	190	440	780	330
1	310	610	0	350	240
Вкупно	990	800	600	1130	570

Табела 3. Резултати од симулација на трениран DQN агент на Ms.Pacman

Серија/ Животи	Прва	Втора	Трета	Четврта	Петта
3	180	170	190	180	470
2	60	40	50	130	120
1	70	60	50	20	50
Вкупно	310	270	290	330	640

Табела 4. Резултати од симулација на агент со рандом политика на Ms.Pacman

Во овој случај исто така може да воочиме колку подобар перформанс има трениранот агент, со највисок резултат од 1130, наспроти 640 на рандом агентот.

Во овој случај, нема линеарно подобрување на резултатот кај трениранот агент, бидејќи во петтата серија резултатот драстично се спушта. Во таа серија, резултатот од првиот агент (570) е помал отколку тој на вториот (640), што покажува дека некогаш и политиката со случаен избор може да даде подобар резултат доколку се погодат условите.

B. BREAKOUT

Резултатите од симулацијата на играта Breakout не се задоволителни. Истите се прикажани во пет серии со по пет животи, но споредено со другите игри, се значително слаби. Резултатите од трениранот DQN агент се прикажани на Табела 5, а резултатите од симулација на агент со рандом политика се прикажани на Табела 6.

Серија/ Животи	Прва	Втора	Трета	Четврта	Петта
5	0	0	0	1	0
4	0	0	0	0	0
3	0	1	1	0	0
2	0	1	0	0	0
1	1	0	0	0	1
Вкупно	1	2	1	1	1

Табела 5. Резултати од симулација на трениран DQN агент на Breakout

Серија/ Животи	Прва	Втора	Трета	Четврта	Петта
5	0	0	0	0	0
4	0	0	0	0	1
3	0	0	1	1	0
2	0	0	0	0	1
1	0	0	1	0	0
Вкупно	0	0	2	1	2

Табела 5. Резултати од симулација на агент со рандом политика на Breakout

Вториот тип на споредба е, како трениране агенти од сите игри се споредуваат меѓу себе? На која игра перформансот е најдобар, а на која најлош?
Бидејќи сите игри имаат различен систем на бодување, како мерка за одлучување го земав збирот на разликите на трениранот агент и агентот со случајна политика на секоја игра. Конкретните резултати можат да се видат на Табела 7.

Резултатите покажуваат дека најголема разлика има агентот при средината Ms.Pacman со вредност од 2250. Генерално, во секоја серија, најуспешни резултати дава агентот на Ms.Pacman, што значи дека овој алгоритам најуспешно се приспособил на таа средина, а очигледно е дека Breakout ги дава најслабите резултати.

Серија/ Животи	Прва	Втора	Трета	Четврта	Петта	Вкупно
Space Invaders	85	35	-20	115	150	365
Ms.Pacman	680	530	310	800	-70	2250
Breakout	1	2	-1	0	-1	1

Табела 7. Резултати од збирот на разликите на трениранот агент и агентот со случајна политика на секоја игра.

VII. ЗАКЛУЧОК

Кодот е поставен на Google Colab [22]. Вештачката интелигенција во игрите е технологија која уште се усовршува, а учењето со поттикнување ветува понатамошен развој на ова поле. Понекогаш, градењето на интелигентни системи во средина на симулација е корисно заради неисцрпниот извор на податоци и контролирана околина, и може да придонесе кон развивање на вештачката интелигенција и во други полиња во иднина.

РЕФЕРЕНЦИ:

[1]Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, Dongbin Zhao:A Survey of Deep Reinforcement Learning in Video Games. CoRR abs/1912.10944 (2019)

[2]N. Y. Georgios and T. Julian, Artificial Intelligence and Games. New York: Springer, 2018.

[3]Yue Zheng: Reinforcement Learning and Video Games. CoRR abs/1909.04751 (2019)

[4]”Atari 2600”, Wikipedia, https://en.wikipedia.org/wiki/Atari_2600. Accessed 30 September 2021.

[5]D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot,L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815, 2017.

[6]R. Bellman et al. The theory of dynamic programming. Bulletin of the American Mathematical Society, 60(6):503–515, 1954.

[7]R. Bellman. A markov decision process. journal of mathematical mechanics. 1957.

[8]Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3):58–68, 1995.

[9] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, 2013.

[10]”OpenAI Gym Docs”, OpenAI Inc., https://gym.openai.com/docs/. Accessed 30 September 2021.

[11]”Atari - Reach high scores in Atari 2600 games.”, OpenAI Inc., https://gym.openai.com/envs/#atari. Accessed 30 September 2021.

[12]”Space Invaders”, Wikipedia, https://en.wikipedia.org/wiki/Space_Invaders. Accessed 30 September 2021.

[13]”SpaceInvaders-v0.”, OpenAI Inc., https://gym.openai.com/envs/SpaceInvaders-v0/. Accessed 30 September 2021.

[14]”Stella - Atari 2600 Video Computer System (VCS) emulator”, GitHub Inc., https://stella-emu.github.io/. Accessed 30 September 2021

[15]”Ms.Pac-Man”, Wikipedia, https://en.wikipedia.org/wiki/Ms._Pac-Man. Accessed 30 September 2021

[16]”Breakout (video game)”, Wikipedia, https://en.wikipedia.org/wiki/Breakout_(video_game) . Accessed 30 September 2021

[17]Jeff Heaton, ”T81-558: Applications of Deep Neural Networks”, GitHub Inc.,https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_12_02_qlearningreinforcement.ipynb. Accessed 30 September 2021

[18]Jeff Heaton, ”T81-558: Applications of Deep Neural Networks”,https://github.com/jeffheaton/t81_558_deep_learning/blob/master/t81_558_class_12_03_keras_reinforce.ipynb. Accessed 30 September 2021

[19]”Train a Deep Q Network with TF-Agents”,Tensorflow.,https://www.tensorflow.org/agents/tutorials/1_dqn_tutorial. Accessed 30 September 2021

[20]”tf.compat.v1.train.AdamOptimizer”,Tensorflow.,https://www.tensorflow.org/api_docs/python/tf/compat/v1/train/AdamOptimizer. Accessed 30 September 2021

[21]”DQN C51/Rainbow”, Tensorflow.,https://tensorflow.google.cn/agents/tutorials/9_c51_tutorial?hl=zh-cn. Accessed 30 September 2021

[22]https://colab.research.google.com/drive/1khydbRfvmmIAfJe2ULKF-HKaAmIljFeG?usp=sharing