Jorma Sahakangas

Koneoppiminen

Peliä oppiva ohjelma

Opinnäytetyö
Kevät 2019
SeAMK Tekniikka
Tietotekniikan Tutkinto-ohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikka

Tutkinto-ohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Sulautetut järjestelmät

Tekijä: Jorma Sahakangas

Työn nimi: Koneoppiminen Peliä oppiva ohjelma

Ohjaaja: Petteri Mäkelä

Vuosi: 2019 Sivumäärä: 36

Opinnäytetyön aiheena oli tutustua koneoppimiseen ja luoda ohjelma, joka oppii pelaamaan peliä.

Työn teoriaosuudessa käydään läpi mitä tekoäly on ja syvennytään koneoppimiseen ja sen toimintaan. Teoriaosuudessa käydään läpi myös työssä käytettyjä OpenAI-yhteisön julkaisemia tekniikoita Gym Retro ja PPO. Teoriaosuus sisältää myös käytettyjen ohjelmointikielien teorian. Työn sovellusosuudessa käytiin läpi ohjelman ja opetuksen toiminta.

Lopputuloksena saavutettiin tavoitteiden mukainen ohjelma, joka oppi pelaamaan Super Mario Bros -peliä.

Avainsanat: koneoppiminen, vahvistusoppiminen, OpenAI, Gym Retro

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Embedded Systems

Author: Jorma Sahakangas

Title of thesis: Machine Learning Program learning to play a game

Supervisor: Petteri Mäkelä

Year: 2019 Number of pages: 36

The topic of the thesis was to study artificial intelligent and machine learning. The second goal of the thesis was to create a program that learns to play a game.

The theoretical part of the thesis explained what artificial intelligence is and it focused on machine learning and its functions. The theoretical part also studied OpenAI:s Gym Retro and PPO technologies which are used in the program. In the theoretical part also theory about the programming languages used in the pro-gram was introduced. The practical part concentrated on the implementation and testing of the application, and it was explained how the program and learning works. The end result of the thesis was a program that leant to play the Super Mario Bros game and it was working as intended.

Keywords: machine learning, reinforcement learning, OpenAI, Gym Retro

SISÄLTÖ

Kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

Agentti	Koneoppimisessa luodaan agentti, joka pystyy oppimaan uutta itsenäisesti.
Avoin lähdekoodi	Ohjelma, jossa lähdekoodiin voi tutustua ja muokata omien tarpeidensa mukaan.
Gym Retro	OpenAI-yhteisön julkaisema koneoppimisympäristö.
json	JavaScript Object Notation. Avoimen standardin tiedosto-muoto tiedonvälitykseen.
Kehys	Videon tai pelin näkymä koostuu kehyksistä.
Linux	Avoimeen lähdekoodiin perustuva käyttöjärjestelmä, joka perustuu Linux-ytimeen.
Lua	Kevyt ja pienikokoinen komentokieli.
Malli	Oppimisen jälkeen luotu tiedosto, jota käytetään halutun vastauksen ennustamiseen.
Neuroverkko	Datan käsittelyyn, matematiikkaan tai laskentaan perus-tuva malli, joka perustuu yhdistävään laskentaan.
Ohjaamaton oppiminen	Koneoppimisen muoto, jossa ohjelma hakee rakennetta annetusta datasta.
Ohjattu oppiminen	Koneoppimisen muoto, jossa ohjelmalle annetaan opetus-dataa sekä oikea ratkaisu.
PPO	Proximal Policy Optimization, vahvistusoppimisluokka.
Python	Monipuolinen, tulkattava ja oliopohjainen ohjelmointikieli.
Renderöinti	Kuvan luominen ohjelman avulla.

Syväoppiminen	Koneoppimisen muoto, jossa hyödynnetään usean kerroksen neuroverkkoja.
Vahvistusoppiminen	Koneoppimisen muoto, jossa ohjelma oppii tekemällä toimintoja.
Valmiussääntö	Kertoo ohjelmalle, koska oppiminen aloitetaan uudestaan aloitusvaiheesta.

1 JOHDANTO

1.1 Työn tausta

Tässä opinnäytetyössä keskitytään koneoppimiseen. Opinnäytetyössä käytetään OpenAI-yhteisön julkaisemaa Gym Retro -ympäristöä ja PPO-vahvistusoppimisluokkaa. Oppimisen kohteeksi on valittu pelin pelaaminen, koska se on yksinkertainen tapa visualisoida ongelman ratkaisua ja oppimisen etenemistä.

Opittavaksi peliksi on valittu Nintendo Entertainment Systemin konsolille julkaistu Super Mario Bros -peli. Super Mario Bros -peli on valittu opittavaksi, koska se on yksi myydyimmistä peleistä ja useat ihmiset tietävät entuudestaan, mitä pelissä tehdään.

1.2 Työn tavoite

Työn tavoitteena on luoda ohjelma, joka pystyy oppimaan pelaamaan Super Mario Bros -peliä. Tavoitteena on löytää sopiva vahvistusoppimisluokka, jolla pelin oppiminen on mahdollista. Opinnäytetyössä käydään läpi eri koneoppimistapoja ja mihin kyseisiä tapoja käytetään. Työssä käydään myös läpi teoria tekoälystä, sekä käytettyjen ympäristöjen ja ohjelmointikielien teoria ja käyttökohteet.

Tämän työn avulla lukijan pitäisi pystyä ymmärtämään, mitä koneoppiminen on, ja mihin sitä tarvitaan. Työn avulla pitäisi myös olla selvää, mitkä asiat vahvistusoppimisessa ovat haastavimpia.

1.3 Työn rakenne

Luvussa 2 käydään läpi mitä tekoäly on. Luvussa kerrotaan myös, mitä koneoppiminen on ja mihin sitä käytetään. Luvussa käsitellään erilaisia koneoppimisen tyyppejä ja tapoja.

Luvussa 3 tutustutaan OpenAI-yhteisön julkaisemiin ympäristöihin, joita käytetään työssä. Luku sisältää käytetyt ohjelmointikielet, lisäksi kerrotaan, miksi Python on suosittu kieli koneoppimisessa.

Luvussa 4 käydään läpi työn vaiheet, sekä ohjelman toiminta. Luvussa kerrotaan myös tarkemmin, miten ohjelma oppi. Luku 5 sisältää työn yhteenvedon ja pohdinnan

2 TEKOÄLY JA KONEOPPIMINEN

Tässä luvussa käydään läpi, mitä tekoäly on, ja kerrotaan koneoppimisen teoriaa.

2.1 Tekoäly

Tekoäly simuloi ihmisten älykkyyttä, joka prosessoidaan tietokoneilla. Prosessit sisältävät oppimista, merkityksen löytämistä, päättelyä ja itseoikaisua. (Copeland 2018.) Tekoälyn voi luokitella kahteen kategoriaan, jotka ovat heikko ja vahva tekoäly. Heikko tekoäly, jota kutsutaan myös nimellä kapea tekoäly, on suunniteltu ja koulutettu tiettyyn tehtävään. Virtuaaliset henkilökohtaiset avustajat, kuten Applen Siri, luokitellaan heikkoon tekoälyyn. Vahvaan tekoälyyn sisältyy yleisluonteiset ihmisten tiedolliset kyvyt. Kun vahvalle tekoälylle esitetään tuntematon tilanne, se pystyy löytämään ratkaisun siihen ilman ihmisen apua. (Rouse 2018a.)

2.1.1 Tekoälyn kategoriat

Michiganin yliopiston apulaisprofessori Arend Hintze luokittelee tekoälyn neljään kategoriaan. Kategorioissa on sekä olemassa olevia tekoälyn tekniikoita että aistivia tekoälytekniikoita, jotka ovat vasta teoreettisia. (Rouse 2018a.)

Tyyppi 1 on reaktiiviset koneet. Esimerkkinä on Deep Blue, joka on shakkia pelaava ohjelma. (Rouse 2018a.) Deep Blue voitti ottelun shakin maailmanmestaria Garry Kasparovia vastaan vuonna 1997. Otteluita oli kuusi, joista Kasparov voitti ensimmäisen ja hävisi toisen. Seuraavat kolme ottelua päättyivät tasapeliin, jonka jälkeen Deep Blue -ohjelma voitti viimeisen pelin. (Anderson 2017.) Deep Blue pystyy tunnistamaan laudalla olevat pelinappulat ja niiden paikat. Ohjelmalla ei ole muistia, joten se ei pysty oppimaan aikaisemmin pelatuista peleistä tai siirroista. Ohjelma analysoi strategian omien ja vastustajan pelinappuloiden sijainnin mukaan. Ohjelma analysoi vastustajalle strategian ja valitsee sen perusteella parhaan siirron. Deep Blue on suunniteltu vain tähän tarkoitukseen ja sen käyttäminen muihin tarkoituksiin ei ole helposti mahdollista. (Rouse 2018a.)

Tyyppi 2 on rajoitettu muisti. Nämä tekoälyohjelmat pystyvät käyttämään aikaisempia kokemuksia tuleviin ratkaisuihin. Tietyt päätöksentekotehtävät itsestään ajavissa autoissa on suunniteltu näin. Ohjelman tekemiä havaintoja, kuten kaistan vaihtoja, ei säilytetä pysyvästi. (Rouse 2018a.)

Tyyppi 3 on mielen teoria. Tämä psykologian termi viittaa siihen, että toisilla on omat uskomuksensa, toiveensa ja aikomuksensa, jotka vaikuttavat heidän päätöksiinsä, mitä he tekevät. Tämän tyyppistä tekoälyä ei vielä ole olemassa. (Rouse 2018a.)

Tyyppi 4 on itsetietoisuus. Tässä kategoriassa tekoälyllä on itsetuntemus ja tietoisuus. Itsetietoisuuden omaavat koneet ymmärtävät nykyisen tilanteensa ja pystyvät tekemään johtopäätöksiä siitä, mitä muut tuntevat. Tämän tyyppistä tekoälyä ei vielä ole olemassa. (Rouse 2018a.)

2.1.2 Missä tekoälyä käytetään

Tekoälyä käytetään monilla eri aloilla, kuten esimerkiksi automaatiossa, jonka ansiosta järjestelmät tai prosessit toimivat automaattisesti. Se voidaan ohjelmoida suorittamaan suuren määrän toistuvia tehtäviä, joita ihmiset normaalisti suorittavat. Tekoälyä käytetään myös konenäössä, joka antaa ohjelmille mahdollisuuden nähdä. Tekoälyä käytetään myös puheen tunnistamiseen ja moniin muihin tarkoituksiin. Tekoälyn yksi osa-alue on koneoppiminen, johon perehdytään seuraavaksi tarkemmin. (Rouse 2018a.)

2.2 Koneoppiminen

Koneoppiminen on tekoälyn osa-alue. Koneoppiminen antaa ohjelmille mahdollisuuden ennustaa haluttuja tuloksia tarkemmin ja paremmin, ilman ohjelman erillistä ohjelmointia. Yksinkertaisesti koneoppimisessa rakennetaan algoritmi, johon annetaan dataa. Algoritmi käyttää annettuun dataan tilastoanalyysiä, jonka perusteella se ennustaa ja antaa vastauksen. Kun algoritmiin annetaan lisää dataa, se antaa uuden ennusteen. Koneoppimisen prosessi on samantapainen, kuin tiedonlouhinta

ja ennustava mallinnus. Näissä molemmissa etsitään datasta rakennetta ja säädetään ohjelman toimintoa sen mukaisesti. Monelle koneoppiminen on tullut vastaan nettikaupoissa, joissa on annettu mainoksia ostoksien perusteella. Tämän mahdollistaa suosittelujärjestelmät, jotka käyttävät reaaliajassa koneoppimista mainosten henkilökohtaistamiseen. Muita yleisiä koneoppimisen käyttökohteita ovat petoksien havaitseminen, roskapostisuodatus, verkon tietoturvariskien havaitseminen, ennustettava huolto ja uutissyötteen rakentaminen. (Rouse 2018b.) Koneoppiminen voidaan jakaa kategorioihin: ohjattu-, ohjaamaton- ja vahvistusoppiminen (Itewiki [Viitattu 8.1.2019]).

2.2.1 Ohjattu oppiminen

Ohjatussa oppimisessa ohjelma saa opetusdataa sekä oikean ratkaisun, esimerkiksi, jos halutaan löytää ja tunnistaa valokuvan sisältävä liikennemerkki. Ohjelma oppii tunnistamaan, mikä liikennemerkki on kuvassa, jonka jälkeen se luo mallin. Ideana on opettaa ohjelma tunnistamaan oikea vastaus opetusdatasta. Jotta tämä toimisi, jonkun on toimittava ohjaajana, joka kertoo opetusesimerkkien oikeat vastaukset. Tästä tulee nimitys ohjattu oppiminen. Oppimisen jälkeen ohjelma luo mallin, joka osaa ratkaista oikean vastauksen myös uudesta datasta. Ohjelma luo paremmin toimivan mallin, mitä enemmän opetusdataa ja aikaa on oppimiseen käytetty. (Reaktor [Viitattu 5.1.2019].)

Esimerkkinä on ruoan tunnistaminen kuvasta. Alussa kerrotaan, kuinka tämä voisi tapahtua ilman koneoppimista. Ohjelman tarkoitus on etsiä kuvasta aikaisemmin luokiteltuja muotoja ja värejä. Jos ohjelma löytää kuvasta ruoan, se kategorioidaan ennestään tehtyyn kategoriaan. Ohjelma voi olla myös hyvin yksinkertainen, sen voi kouluttaa tunnistamaan, onko kuvassa esimerkiksi pizza vai ei. Kun ohjelmalle annetaan uusi kuva, yleisesti se antaa ennusteen, kuinka todennäköistä on, että kuvassa on pizza. Esimerkiksi kuvan ennuste voi olla pizzan mahdollisuus kuvassa 0,6 ja ei pizzaa kuvassa 0,4. Tulokseen pitää antaa kynnys, milloin ennuste kategorioi tuloksen pizzaksi. Asetetaan kynnykseksi 0,5. Aikaisempi ennuste luokitellaan pizzaksi, koska tulos on suurempi kuin asetettu kynnys. (Bedford 2016.)

Ohjelman tarkkuus voidaan laskea sekaannusmatriisista (confusion matrix). Ohjelmalle voi esimerkiksi antaa 100 kuvaa ja luoda taulukon. (Bedford 2016.)

Taulukko 1. Sekaannusmatriisi kuvien tunnistamiselle (Perustuu. Bedford 2016).

		Ennustettu kategoria	
		Pizza	**Ei Pizza**
Oikea Kategoria	**Pizza**	**True Positive**	**False Negative**
	Ei Pizza	**False Positive**	**True Negative**

Taulukossa 1 rivit kuvaavat oikeita kategorioita ja sarakkeet kuvaavat ennusteita.

Taulukossa yksi on kategorioinnit seuraavat:

- True Positive: Kuvat, jotka on oikein luokiteltu pizzaksi
- False Negative: Kuvat, jotka on väärin luokiteltu ei pizzaksi
- False Positive: Kuvat, joissa ei ole pizzaa, on luokiteltu väärin pizzaksi
- True Negative: Kuvat, joissa ei ole pizzaa, on luokiteltu oikein ei pizzaksi

Taulukon arvoista voidaan laskea kaksi arvoa.

$$Precision = \frac{\text{True Positive}}{\text{True Positive+False Positive}} \quad (1)$$

$$Recall = \frac{\text{True Positive}}{\text{True Positive+False Negative}} \quad (2)$$

Kaavassa 1 Precision mittaa kaikista pizzaksi luokitelluista kuvista, mikä osio sisältää oikeasti pizzan. Arvo kertoo paljonko kuvia, joissa ei ole pizzaa, on luokiteltu sisältävän pizzan. Kaavassa 2 Recall mittaa pizzan sisältävistä kuvista, mikä osio oli oikein ennustettu sisältävän pizzan. Tämä arvo kertoo, kuinka tarkasti ohjelma osaa oikein luokitella pizzan sisältävät kuvat. (Bedford 2016.)

Ilman koneoppimista ruoan tunnistamista tulisi liian työlästä, koska esimerkiksi banaanin tunnistamiseen voisi käyttää sen muotoa, kokoa ja väriä. Tunnistamiseen pitäisi lisätä myös raaka vihreä banaani ja ylikypsä ruskea. Vielä vaikeammaksi tunnistamisesta tulee, kun kyseessä on valmistettu ruoka, joka sisältää montaa raaka-ainetta. Tämän takia ohjelmaan kannattaa lisätä ohjattua oppimista. (Bedford 2016.)

Ohjelman voi kouluttaa valmiiksi luokitetuilla kuvilla. Esimerkissä on käytetty valmista tietokantaa, joka sisältää 1 000 ruokakategoriaa, ja jokainen kategoria sisältää 101 kuvaa. Kuvat on jaettu osajoukkoon koulutusta ja testausta varten. Koulutuskuvia näytetään oppimisalgoritmille oikeiden vastauksien kanssa. Testausta varten jätetyt kuvat kertovat koulutuksen jälkeen, kuinka hyvin ohjelma on oppinut. Koulutuksessa on tärkeää, ettei testausta varten jätettyjä kuvia ole käytetty opetuksessa. Tämä vaikuttaisi tulokseen, kuinka hyvin ohjelma ennustaa oikeita vastauksia. Testausta varten jätettyjä kuvia on hyvä olla 15 – 20 % koulutuskuvien määrästä. Ohjatun oppimisen ansiosta eri ruoista ei tarvitse itse kertoa ohjelmalle, mitä sen pitäisi etsiä. Koulutuksen jälkeen ohjelma osaa löytää ennalta nähdystä kuvasta ruoan. (Bedford 2016.)

2.2.2 Ohjaamaton oppiminen

Ohjaamattomassa oppimisessa ohjelma hakee rakennetta annetusta datasta, josta ei tiedetä ennakolta mitään. Ohjaamattomassa oppimisessa ei haeta oikeita ratkaisuja, joten ohjelma ei ohjatun oppimisen tapaan luo mallia. Tyypillisesti ohjaamattomassa oppimisessa ohjelman tavoite on löytää datasta jonkinlaista rakennetta. Tämä tarkoittaa esimerkiksi visualisointia, jossa samankaltaiset löydetyt esimerkit sijoitetaan kuvaan lähelle toisiaan, ja erilaiset esimerkit kauas toisistaan. Ohjaamaton oppiminen voi myös tarkoittaa ryvästämistä, jossa on tavoitteena muodostaa ryhmiä datasta. (Reaktor [Viitattu 5.1.2019].)

Ohjatussa oppimisessa visualisointia voisi käyttää esimerkiksi kaupan ostosdatan analysointiin. Tämä voisi luoda kuvan, jossa samoja tuotteita ostavat asiakkaat ovat lähempänä toisiaan, tai saman verran rahaa käyttävät ovat lähempänä toisiaan. (Reaktor [Viitattu 5.1.2019].) Visualisoinnin tarkoitus on löytää mielenkiintoisia suhteita saadusta datasta (Khatun 2018).

Ryvästäminen eristää tai jakaa datan lukuisiin ryhmiin. Ryhmitys tapahtuu niin, että samoissa ryhmissä olevien data on enemmän samanlaista kuin muissa ryhmissä. Yksinkertaisesti tavoite on jakaa saatu data samojen piirteiden omaavien kanssa. (Khatun 2018.) Ohjatussa oppimisessa ryvästämistä voidaan käyttää myös kaupan

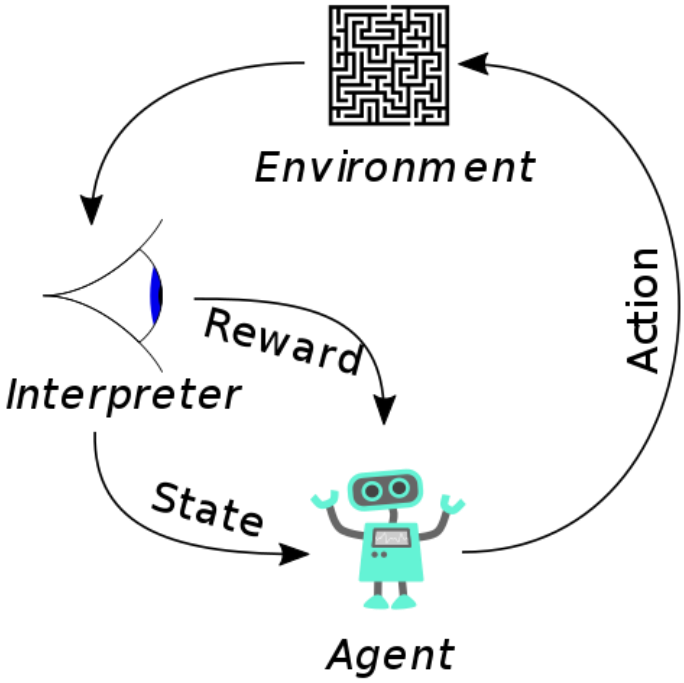
ostosdatan analysointiin. Ryvästäminen voi jaotella asiakkaat esimerkiksi matala-budjettisiin, jotka ostavat vain valmisruokaa, tai vain kasvistuotteita ostaviin, jotka ostavat aina samat tuotteet. Ryvästäminen löytää vain ryhmiä, mutta ei osaa nimetä ryhmiä. (Reaktor [Viitattu 5.1.2019].)

2.2.3 Vahvistusoppiminen

Vahvistusoppimisessa luodaan agentti, joka oppii ympäristössä tekemällä toimintoja, joista agentti saa heti palautetta (Simonini 2018a). Jos agentilla ei ole aikaisempaa koulutusta, niin sen pitää kokeilla eri asioita ja oppia siitä. Agentti kerää tietoa asioista, joista on tullut palkinnoksi pisteitä, ja mistä pisteitä on otettu pois. Agentin päämäärä on kerätä mahdollisimman paljon pisteitä. (Maini 2017.)

Esimerkkinä on agentti, joka pelaa Super Mario -peliä, jonka tavoitteena on liikkua tason oikeaan reunaan. Agentti vastaanottaa ympäristöltä tilan, joka tässä esimerkissä olisi pelin ensimmäinen kehys. Jos agentti siirtää pelihahmoa oikealle, niin ympäristö siirtyy uuteen tilaan. Ympäristö antaa agentille palkinnoksi pisteen, koska hahmo on lähempänä päämääräänsä. Agentin päämääränä on kerätä mahdollisimman paljon pisteitä. Epäonnistumisen jälkeen agentti yrittää uudestaan. (Simonini 2018a)

Kuvio 1 kuvaa agentin toimintaa vahvistusoppimisessa. Kuviossa agentti (Agent) saa tilan (State), jonka jälkeen se tekee toiminnon (Action) ympäristössä (Environment), jonka aikana tulkitsija (Interpreter) seuraa, onko agentti ansainnut palkinnon (Reward) ja lähettää agentille taas uuden tilan.



Kuvio 1. Vahvistusoppimisen diagrammi (Reinforcement learning diagram 2017).

2.2.4 Neuroverkko ja syväoppiminen

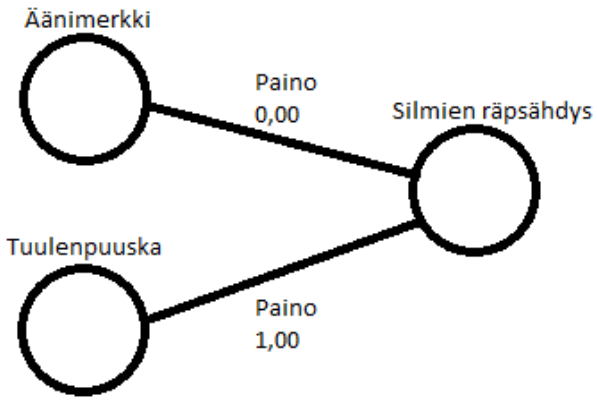
Ohjelman tekemä neuroverkko koostuu yleensä suuresta määrästä yksinkertaisia yksiköitä eli neuroneja. Nämä lähettävät ja vastaanottavat viestejä toisiltaan. Neuronit ovat yksinkertaisia tiedonkäsittelijöitä. Ne koostuvat solukeskuksesta ja johdoista. Yleensä yksittäiset neuronit eivät tee muuta kuin tarkkailevat signaaleja, joita muut neuronit lähettävät niille johtoja pitkin. Neuroneiden yhteyksiin kuuluu muuttuvia parametrejä, joita kutsutaan painoiksi tai painokertoimiksi. Yksittäinen neuroni pystyy vain hyvin yksinkertaisiin toimintoihin, mutta suuri joukko neuroneita kytkettynä toisiinsa pystyy monimutkaisiin toimintoihin. (Reaktor [Viitattu 10.1.2019].)

Syväoppimisella tarkoitetaan tietynlaista koneoppimista, jossa yksinkertaisista prosessointiyksiköistä koostuvia kerroksia yhdistetään verkoksi. Tässä verkossa ohjelman tieto kulkee kerroksien läpi vuoron perään. (Reaktor [Viitattu 10.1.2019].)

Seuraavassa on esimerkki neuroverkon rakentamisesta. Kuvataan eläimen käytöstä, kun sen kasvoihin tulee tuulenpuuska. Joka kerta tuulenpuuskan jälkeen eläin räpsäyttää silmiään. Tästä voidaan luoda neuroverkkomalli, jossa ensimmäisessä neuronissa tuulenpuuska antaa arvon yksi, ja ilman tuulenpuuskaa arvo on nolla.

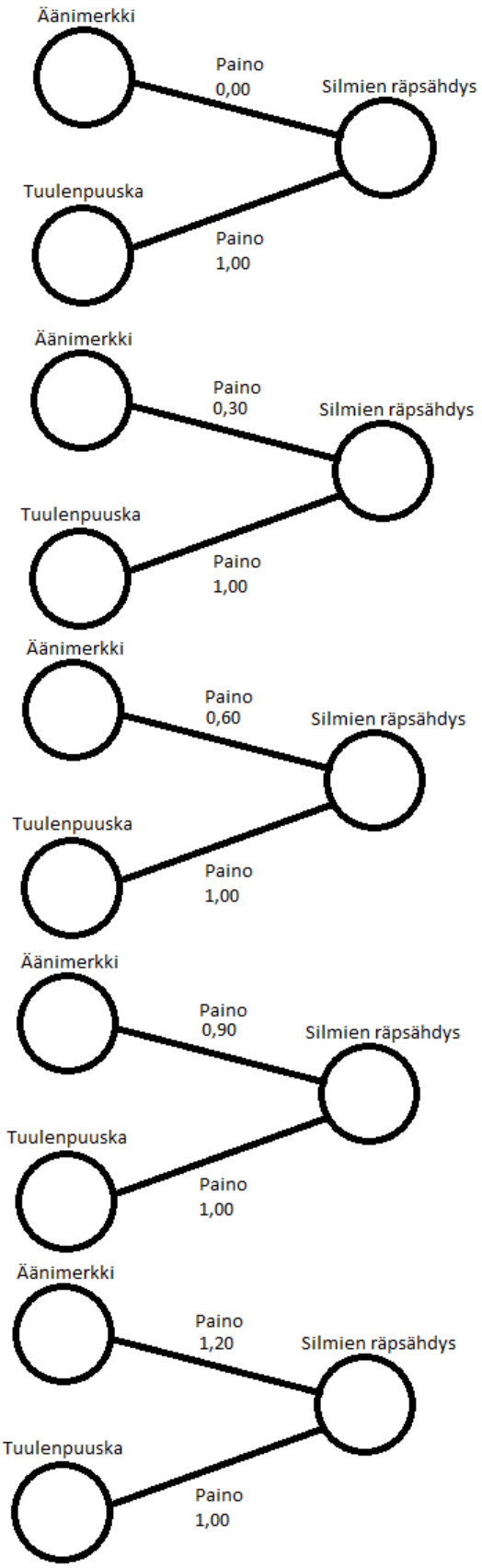
Eläimen silmien räpsähdys päätetään toisessa neuronissa. Jos neuroni vastaanottaa arvon yksi tai isomman, eläin räpsäyttää silmiään. Jos se vastaanottaa alemman arvon kuin yksi, se ei tee mitään. Neuroneiden välisellä johdolla on painoarvo, joka tässä esimerkissä on yksi. Kun ensimmäinen neuroni havaitsee tuulenpuuskan, niin arvo on yksi. Tämä arvo kerrotaan painolla, joka esimerkissä on yksi. Tulokseksi tulee yksi ja lopputuloksena on silmien räpsähdys. (Leon 2017.)

Kuviossa 2 samaan neuroverkkoon on lisätty uusi neuroni, joka kuvaa äänimerkkiä. Jos neuroni havaitsee äänimerkin, se saa arvon yksi ja ilman äänimerkkiä arvon nolla. Ainoana erona aikaisempaan neuroniin on näiden välinen paino, joka tässä tapauksessa on nolla. Tämä tarkoittaa sitä, että vaikka neuroni saisi äänimerkin eläin ei räpsäytä silmiä. (Leon 2017.) Neuroverkkoa voi kuvailla ennen koulutusta kuviolla 2.



Kuvio 2. Kouluttamaton neuroverkko (Perustuu. Leon 2017).

Neuroverkon koulutusta voi kuvailla yksikertaisesti painojen muutoksena neuroneiden välillä ajan kuluessa. Tämän neuroverkon voi kouluttaa siten, että eläin räpsäyttää silmiään, myös äänimerkin saadessaan. Koulutus tapahtuisi toistamalla samaa asiaa, kunnes äänimerkki-neuronin ja silmien räpsähdys -neuronin välinen paino kasvaisi tarpeeksi suureksi. Oppimisvaiheessa annetaan ensin äänimerkki, jonka jälkeen tulee tuulenpuuska. Eläin räpsäyttää silmiä vasta tuulenpuuska-neuronin signaalista. Äänimerkin ja räpsähdyksen välinen paino kasvaisi tämän tapahtuman jälkeen asetetulla arvolla. Arvoa voidaan kasvattaa 0,3. Kuvion 3 mukaan tämä tarkoittaisi sitä, että silmien räpsähdys tapahtuisi neljännellä kerralla, jolloin paino olisi kasvanut arvoon 1,2, joka kerrotaan äänimerkin arvolla yksi. Silmien räpsähdyksen kynnyksenä on yksi, joten neuroverkko on oppinut uutta. (Leon 2017.)



Kuvio 3. Neuroverkon koulutusvaihe (Perustuu. Leon 2017).

3 SOVELLUSYMPÄRISTÖ JA TARVITTAVAT TEKNOLOGIAT

Tässä luvussa kerrotaan käytetyistä välineistä ja tekniikoista, joita on hyödynnetty työssä. Tässä luvussa kerrotaan OpenAI-yhteisön luomista koneoppimistyökaluista ja käytetyistä ohjelmointikielistä.

3.1 OpenAI

OpenAI on vuonna 2015 perustettu voittoa tavoittelematon yhteisö, joka kehittää tekoälyä ja koneoppimista. OpenAI-yhteisön tavoite on edistää digitaalista älyä tavalla, joka hyödyntäisi ihmiskuntaa kokonaisuutena. (Brockman & Sutskever 2015.)

3.1.1 Gym Retro

Gym Retro on OpenAI-yhteisön julkaisema ympäristö, joka on tarkoitettu vahvistusoppimisen tutkimukseen pelien kanssa. Retro-ympäristössä on valmiiksi yli 1 000 peliä ja OpenAI on julkaissut työkalut, joilla pelejä on lisätty. OpenAI-yhteisön aikaisemmat vahvistusoppimistutkimukset ovat enimmäkseen keskittyneet mallien optimointiin, jotka ratkaisivat yksinkertaisia tehtäviä. Gym Retrolla on kyky yleistää pelejä, joissa on samantapainen konsepti, mutta eri ulkonäkö. (Pfau, Nichol, Hesse, Schiavo & Klimov 2018.)

Gym Retro on tuettu Windows-, MacOS- ja Linux-käyttöjärjestelmissä. Ohjelmointikielinä ovat Python-kielen versiot 3.5, 3,6 ja 3,7. (OpenAI 2018a.)

Vuonna 2018 OpenAI järjesti Retro-kilpailun. Kilpailijat loivat vahvistusoppimisohjelmillaan agentin Sonic The Hedgehog -peliin. OpenAI-yhteisön tiimi testasi kahdessa salaisessa testissä kilpailijoiden agentteja. Kilpailun tarkoituksena oli nähdä, kuinka hyvin eri algoritmeilla luodut agentit pystyvät ohjaamaan pelihahmoa tasoilla, joita agentti ei ollut ennen pelannut. Koneoppimiselle asetettiin myös aikaraja, joka oli noin 18 tuntia. (Hesse, Schulman, Pfau, Nichol, Klimov & Schiavo 2018a.) Parhaan tuloksen kilpailun aikana sai kuuden hengen tiimi Dharmaraja 4692 pisteellä, teoreettisesta enimmäismäärästä 10 000. Kilpailu osoitti, että parhaat tulokset tuotettiin

koneoppimisella, eikä kilpailuun räätälöidyllä ohjelmalla. Tämä näyttää sen, että tätä ongelmaa ei pysty ratkaisemaan huijaamalla. Kilpailuun osallistui 923 tiimiä, joista 229 antoi ratkaisunsa openAi-yhteisölle. (Hesse, Schulman, Pfau, Nichol, Klimov & Schiavo 2018b.)

3.1.2 PPO (Proximal Policy Optimization)

PPO on OpenAI-yhteisön julkaisema vahvistusoppimisluokka. PPO on OpenAI-yhteisön käytössä oleva oletusvahvistusoppimis-algoritmi, sen helppokäyttöisyyden ja hyvän suorituskyvyn takia. (Sculman, Klimov, Wolski, Dhariwal & Radford 2017.)

PPO-algoritmin tavoite on löytää isoin mahdollinen edistysaskel käyttäen saatavalla olevaa dataa ja samalla välttää luomasta liian suurta askelta, ettei se vahingossa aiheuttaa suorituskykyongelmia (Achiam 2018). PPO-algoritmin keskeinen ajatus on välttää liian suurta menettelytapaa (Simonini 2018b). Menettelytapaa voi kuvailla vahvistusoppimisagentin strategiana, miten agentti ansaitsee seuraavan pisteen (StackOverflow 2017).

PPO-luokkaa on käytetty OpenAI Five -nimisessä projektissa. Tässä projektissa OpenAI-yhteisö koulutti ohjelman oppimaan Dota 2 -peliä (OpenAI 2018b). Dota 2 on hyvin monimutkainen peli. Pelissä on yli sata hahmoa, joilla jokaisella on omat kykynsä ja attribuuttinsa. Dota 2 on moninpeli, jossa viiden pelaajan joukkue taistelee toisiaan vastaan. Lyhyesti sanottuna molempien joukkueiden tavoite on tuhota vastustajien päärakennus, jonka jälkeen rakennuksen rikkonut joukkue on voittanut. (Gies 2016.) Opetteluun OpenAI Five käyttää PPO-luokkaa. Ohjelma pystyy päivän aikana pelaamaan 180 vuoden edestä pelejä itseään vastaan. Ohjelmaan käynnissä pitoon käytettiin 256 grafiikkaprosessoria ja 128 000 prosessorin ydintä. (OpenAI 2018b.) OpenAI Five pelasi tiimiä vastaan, jonka pelaajat on luokiteltu paremmaksi kuin 99,95 prosenttia muista. Ottelu oli paras kolmesta, joista OpenAI Five voitti 2 ensimmäistä. (OpenAI 2018c.)

3.2 Ohjelmointikielet

Tässä työssä on käytetty Python- ja Lua-ohjelmointikieltä. Koneoppimisen yhteydessä käytetään usein Python-kieltä.

3.2.1 Python

Python on tulkattava, interaktiivinen ja oliopohjainen ohjelmointikieli. Python Software Foundation on itsenäinen voittoa tavoittelematon yhteisö, joka omistaa tekijänoikeuden Python versioon 2.1 ja siitä uudempiin. Yhteisön tavoite on edistää avoimen lähdekoodin teknologiaa, joka liittyy Python-ohjelmointikieleen ja mainostaa kielen käyttöä. Python on laajennettavissa C- ja C++-ohjelmointikielillä. Python toimii myös laajennuksena kielille, jotka tarvitsevat ohjelmoitavan rajapinnan. Python kielen mukana tulee valmiiksi myös laaja valikoima ohjelmointikirjastoja. (Python [Viitattu 2.2.2019].)

Pythonia on käytetty esimerkiksi useissa Linux-käyttöjärjestelmissä, kuten Red Hat. Pythonia on käytetty myös järjestelmänhallintaan. Python-kieltä käyttää sisäisesti muun muassa Google, Yahoo ja Lucasfilm Ltd. (Python [Viitattu 2.2.2019].)

Python versioita on 2 ja 3. Viimeisin versio on 2.7.15, joka julkaistiin 01.05.2018 ja 3.7.2, joka julkaistiin 24.12.2018. (Python [Viitattu 3.2.2019].) Python-version 2.7 elämän päättymispäivä (The End Of Life date) on vuonna 2020 (Python [Viitattu 19.2.2019]). Python 2.7 oli vuonna 2017 63,7 % käytetympi kuin Python 3. Python 2 on vieläkin tuettu ja käytössä, koska se on ollut vuosia suosittu ohjelmointikieli. Jotkut Python 2 -versiolle luodut ohjelmointikirjastot eivät toimi uudemman version kanssa ja versiolle 3 tehdyt kirjastot eivät sovi vanhempaan. Osa isoista yrityksistä on myös siirtynyt Python 2 -versiosta versioon 3, kuten Instagram ja Facebook. (Bradford 2018.)

3.2.2 Lua

Lua on tehokas ja kevyt komentokieli. Luassa on avoin lähdekoodi ja se on MIT-lisenssin alla. Lisenssin mukaan Luaa voi käyttää mihin tahansa tarkoitukseen, se on myös ilmainen kaupallisiin tarkoituksiin. Luaa käytetään muun muassa Adobe Photoshop Lightroom -ohjelmassa. Kieltä voidaan käyttää myös sulautetuissa järjestelmissä. Luan käyttö myös peleissä on yleistä, esimerkiksi suositussa pelissä World of Warcraft ja suomalaisen yrityksen luomassa Angry birds -pelissä. Luaa käytetään muiden ohjelmointikielien laajentamiseen, joihin kuuluu C, C++, C# ja Java sekä monia muita. Lua sopii myös muiden komentokielien laajennukseen kuten Perl ja Ruby. (Lua 2018.)

Lua-kieli on siirrettävä ohjelmointikieli sen 800 kilotavun lähdetiedostojen koon takia. Pienen koon takia Lua on ideaali kieli sulatettuihin järjestelmiin, niiden rajoitteellisten resurssien takia. Kieli on kirjoitettu ANSI C -kielellä, joka tarkoittaa sen toimimista lähes kaikilla laitteilla, joissa on C-ohjelmointikielen kääntäjä. Tämä tarkoittaa, että Luan käyttö on mahdollista teollisessa internetissä. (Radcliffe [Viitattu 19.2.2019].)

3.2.3 Ohjelmointikielet koneoppimisessa

Kysyttäessä yli 2 000 Data scientist -asiantuntijalta ja koneoppimiskehittäjältä ohjelmoinnissa käytettyä ohjelmointikieltä vastausten mukaan 57 % käyttää Python-ohjelmointikieltä. Suosion takana on muun muassa Python-kielelle kehitetyt koneoppimiskirjastot kuten TensorFlow ja OpenAI. Koneoppimisessa ja tekoälyssä ei kuitenkaan ole parasta ohjelmointikieltä. Ohjelmointikielen valinta riippuu kohteesta. Muita suosittuja ohjelmointikieliä koneoppimisessa ovat C/C++, Java, R ja JavaScript (Voskoglou 2017.)

Koneoppimisessa yhtenä tärkeänä tehtävänä on erottaa, käsitellä, määrittää, siistiä, järjestää ja ymmärtää dataa, jotta älykkäiden algoritmien kehittäminen on mahdollista. Monimutkaiset matemaattiset yhtälöt on helppo käsitellä tuomalla ulkoisia moduuleja ja kirjastoja. Yksi syy Python-kielen suosioon on sen helppo luettavuus ja kirjastojen laajuus. Kirjastoja on muun muassa kuvienkäsittelyyn, tekstinkäsittelyyn,

ääneen, koneoppimisongelmiin ja moniin muihin koneoppimista helpottaviin töihin. Kirjastot ovat helppokäyttöisiä ja ne on helppo tuoda ohjelmaan. Kirjastojen käyttö on ilmaista GNU (General Public License) -lisenssin ansiosta. (Patel 2019.)

4 PELIÄ OPPIVA OHJELMA

Tässä luvussa kerrotaan tarkemmin, mitä tietoja ohjelma tarvitsee oppiakseen. Tässä luvussa kerrotaan myös, miten ohjelma ja ohjelman opetus toimii.

4.1 Pelistä tarvittavat tiedot

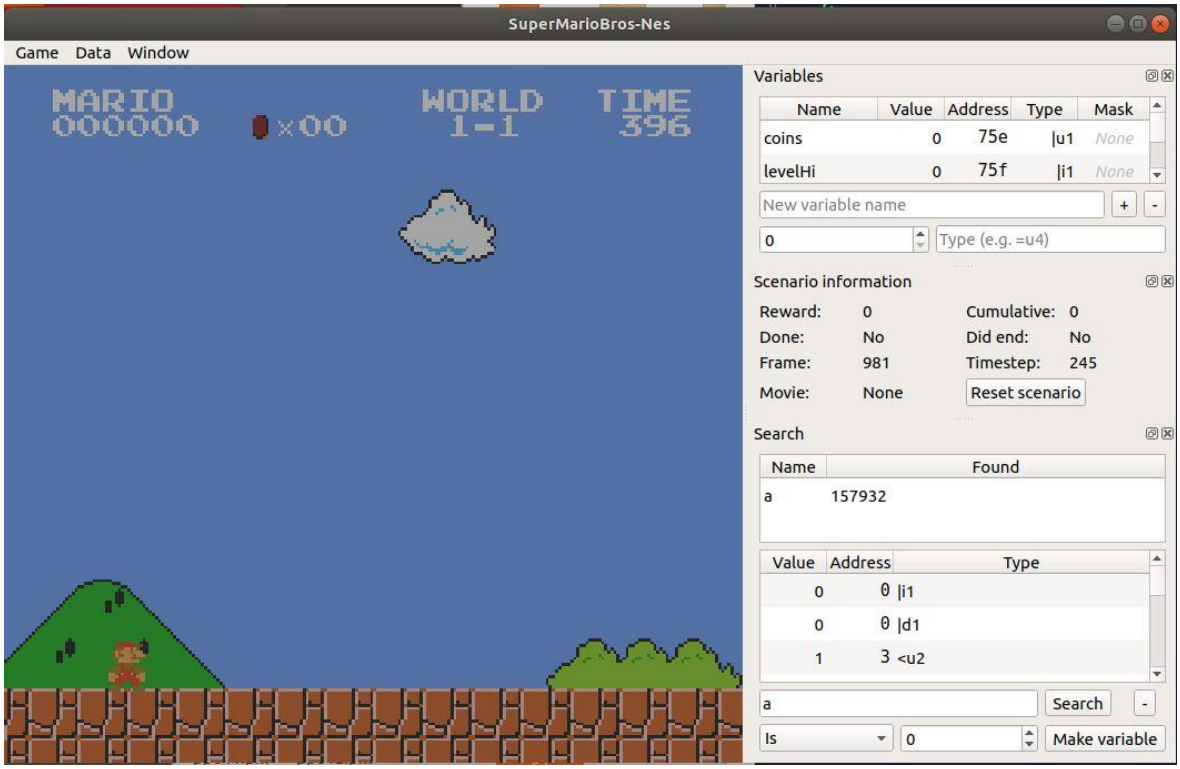
Gym Retro -ympäristössä olevissa peleissä on valmiiksi haettu pelin käyttömuistiosoitteita. Suureen osaan pelejä on haettava lisää muistiosoitteita, jotta ohjelman oppiminen olisi mahdollisimman tehokasta. Gym Retro -ympäristössä on työkalu Retro UI, jolla pystyy hakemaan haluttua muistipaikkaa. Muistiosoitteita haetaan karsimalla niin paljon, että jäljelle jää vain yksi. Retro UI -työkalulla pystytään luomaan pelille aloituskohdat, mistä peli alkaa.

Opittavaksi peliksi on valittu Super Mario Bros. Peliin on valmiiksi muistiosoitteita, kuten pelihahmon sijainti, pistemäärä ja kolikoiden määrä. Käyttömuistiosoitteet luetaan suoraa pelin käyttömuistista, tietyistä muistiosoitteista. Näillä muistiosoitteilla ohjelman voi jo laittaa oppimaan peliä, mutta paremman ja nopeamman oppimisen saavuttamisen ohjelma tarvitsee lisää tietoa pelistä.

Ohjelman tavoite on läpäistä pelin taso. Super Mario Bros -pelin kaikki tasot alkavat vasemmalta ja maali on oikealla. Ohjelma voi hyödyntää jo valmiiksi etsittyä muistiosoitteesta, joka kertoo pelihahmon X-osoitteen eli missä pelihahmo on vaakatasossa. Pelissä on kaksi muistiosoitetta, joita käytetään sijainnin määrittämiseen. Ensimmäinen kertoo hahmon siirtymisen arvoon 225 asti ja alkaa taas nollasta, jonka jälkeen toinen muistiosoite kasvaa yhdellä. Tiedostossa ei ole valmiiksi muistiosoitetta, joka kertoo pelihahmon juoksunopeuden. Juoksunopeuden kertovan muistiosoitteen voi etsiä kuviossa 4 esitetyllä Gym Retron mukana tulleessa Retro UI -ohjelmalla. Ohjelma etsii ja näyttää pelin kaikki muistiosoitteet ja niiden arvon. Retro UI antaa mahdollisuuden etsiä karsimalla muistiosoitteita annetuilla ehdoilla.

Ehtoja ovat:

- **Is**-ehdolla etsitään muistiosoitetta annetulla arvolla
- **Increased by** -ehdolla etsitään muistiosoitetta, jonka arvo on noussut tietyllä arvolla
- **Decreased by** -ehdolla etsitään muistiosoitetta, jonka arvo on laskenut tietyllä arvolla
- **Increased**-ehto jättää kaikki arvot, joiden arvo on kasvanut
- **Decreased**-ehto jättää kaikki arvot, joiden arvo on laskenut
- **Changed**-ehto jättää kaikki arvot, joiden arvo on muuttunut
- **Unchanged**-ehto jättää kaikki arvot, joiden arvo ei ole muuttunut



Kuvio 4. Retro UI.

Pelihahmon nopeuden kertovan muistiosoitteen karsiminen Retro UI -ohjelmalla tapahtuu seuraavasti: Ohjelmaan ladataan peli, josta ohjelma etsii kaikki muistiosoitteet. Pelihahmon nopeus arvo ei vaihdu, kun pelihahmo seisoo paikallaan, joten kaikki vaihtuvat arvot karsitaan changed-ehdolla. Ohjelmassa voidaan myös liikuttaa pelihahmoa, jolloin pelihahmon nopeus kasvaa ja karsia increased-ehdolla kaikki muistiosoitteet, joiden arvo ei ole kasvanut. Eri ehtoja käytetään, kunnes jäljellä on vain yksi muistipaikka, joka edustaa etsittyä arvoa. Joidenkin etsittävien

muistiosoitteiden kohdalla kaikkien arvojen karsiminen voi olla haastavaa, jolloin ohjelmassa voidaan verrata reaaliajassa päivittyviä jäljellä olevia muistiosoitteiden arvoja etsittävään arvoon. Kun arvo on löydetty, muistiosoite muutetaan Retro UI -ohjelman näyttämästä heksadesimaalimuodosta kuvion 5 json-muodossa olevaan listaan desimaalimuodossa.

```
{
  "info": {
    "coins": {
      "address": 1886,
      "type": "|u1"
    },
    "levelHi": {
      "address": 1887,
      "type": "|i1"
    },
    "levelLo": {
      "address": 1884,
      "type": "|i1"
    },
    "lives": {
      "address": 1882,
      "type": "|i1"
    },
    "score": {
      "address": 2013,
      "type": ">n6"
    },
    "scrolling": {
      "address": 1912,
      "type": "|i1"
    },
    "time": {
      "address": 2040,
      "type": ">n3"
    },
    "xscrollHi": {
      "address": 1818,
      "type": "|u1"
    },
    "xscrollLo": {
      "address": 1820,
      "type": "|u1"
    },
    "vSpeed": {
      "address": 87,
      "type": "|i1"
    },
    "live": {
      "address": 110,
      "type": "|u1"
    }
  }
}
```


Kuvio 5. Käyttömuistiosoitelista.

4.2 Oppimisen säännöt ja palkinnot

Vahvistusoppimisohjelmalle on annettava sääntöjä. Gym Retro -ympäristössä on valmiina säännöt. Monessa pelissä valmiina olevat säännöt eivät sisällä montaa sääntöä. Säännöt ovat json-tiedostomuodossa, joten monimutkaisten sääntöjen antaminen on vaikeaa. Säännöt voidaan antaa ohjelmalle myös Lua-komentokielellä. Tämä mahdollistaa monimutkaisempien sääntöjen annon.

Vahvistusoppimisessa säännöissä määritellään valmius. Valmius määrittää, koska ohjelma on valmis aloittamaan pelin alkutilasta. Valmiussääntönä voi olla esimerkiksi pelihahmon kuolema tai sen saama vahinko. Jos sääntönä on vahingon saaminen, ohjelma oppii täysin välttämään vahingon saamista. Jos sääntönä on pelihahmon kuolema, ohjelma välttää pelihahmon kuolemaa, mutta ei vältä vahingon saamista. Sääntöjä voi myös olla myös useampia. Yhtenä valmiussääntönä on hyvä olla hahmon paikallaan seisominen, jolloin ohjelma tietää aloittaa pelin alusta ja koettaa jotain muuta. Tämä nopeuttaa oppimisprosessia, koska paikallaan odotusta vältetään.

Vahvistusoppimisessa ohjelmalle annetut palkintosäännöt määräävät oppimisen tavoitteen. Tavoitteena pelissä voi olla esimerkiksi maaliin pääsy mahdollisimman nopeasti tai pisteiden keräys. Sääntöjä hienosäätämällä vaikutetaan, kuinka nopeasti ja hyvin ohjelma oppii. Ohjelmalle voidaan antaa negatiivisia palkintoja. Negatiivisen palkinnon avulla ohjelma myös oppii välttämään haluttuja asioita. Esimerkiksi Super Mario Bros -pelille voidaan antaa pisteitä aina, kun pelihahmoa liikutetaan oikealle päin, eli tason loppua kohden. Tällä säännöllä ohjelma oppii pelaamaan tason loppuun asti. Ohjelmalle voidaan myös antaa pisteitä sen nopeudesta, jonka ansiosta ohjelma oppii pelaamaan tason mahdollisimman nopeasti. Super Mario Bros -pelissä on muistiosoite pelihahmon nopeudelle, näin voidaan antaa ohjelmalle pisteitä sen nopeudesta.

4.3 Ohjelman toiminta

Ohjelmointikielenä on käytetty Python-versiota 3.6. Ohjelma lataa ympäristön Gym Retron avulla (kuvio 6). Ympäristö sisältää pelattavan pelin, tason mistä peli aloitetaan ja säännöt. Säännöt haetaan json-muodossa olevasta tiedostosta. Koulutussääntötiedosto voi myös käynnistää Lua-komentokielellä tehdyn tiedoston, johon on mahdollista tehdä monimutkaisempia sääntöjä.

```
env = retro.make('SuperMarioBros-Nes', state='Level1-1', scenario='training')
```

Kuvio 6. Ympäristön luonti.

Ohjelma pystyy luomaan monta ympäristöä ja oppimaan niissä samaan aikaan. Tämä nopeuttaa oppimisen kestoa. Ympäristöjen määrän rajaa tietokoneen prosessori. Jos prosessorissa on esimerkiksi neljä ydintä, voidaan ympäristöjä luoda neljä.

Ohjelma luo mallin, johon määritellään opetusalgoritmi (kuvio 7). Opetusalgoritmiin myös määritellään tapa, miten ohjelma katsoo pelissä näkyvää kuvaa. Oppimista varten luodaan myös loki TensorBoardilla, joka ottaa talteen oppimisen kehityksen ja sen keston.

```
model = PPO2(CnnPolicy, env, verbose=1, tensorboard_log="/home/ubu/Desktop/ohjelma/mario_tensorboard/")
```

Kuvio 7. Mallin luonti.

Ohjelma aloittaa mallin opetuksen, jonka kesto määritetään askeleilla (kuvio 8). Askeleet kertovat montako toimintoa ohjelma tekee luodussa peliympäristössä. Opetuksen jälkeen ohjelma tallentaa opetetun mallin (kuvio 8). Kun malli on luotu, voidaan se ladata ohjelmaan ja aloittaa pelaaminen. Tässä vaiheessa voidaan myös vaihtaa tiedosto, joka sisältää säännöt. Kuviossa 9 on esitetty pelin renderöinti. Pelaamisessa ohjelman tekemä malli ennustaa ympäristön perusteella näppäinpainallukset, jotka syötetään ympäristöön. Tämän jälkeen ohjelma renderöi ympäristön näytettäväksi kuvaksi ja ennustaa taas seuraavat näppäinpainallukset.

```
model.learn(total_timesteps=1000000)
model.save("mariobros-ppo12")
```

Kuvio 8. Mallin koulutus ja tallennus.

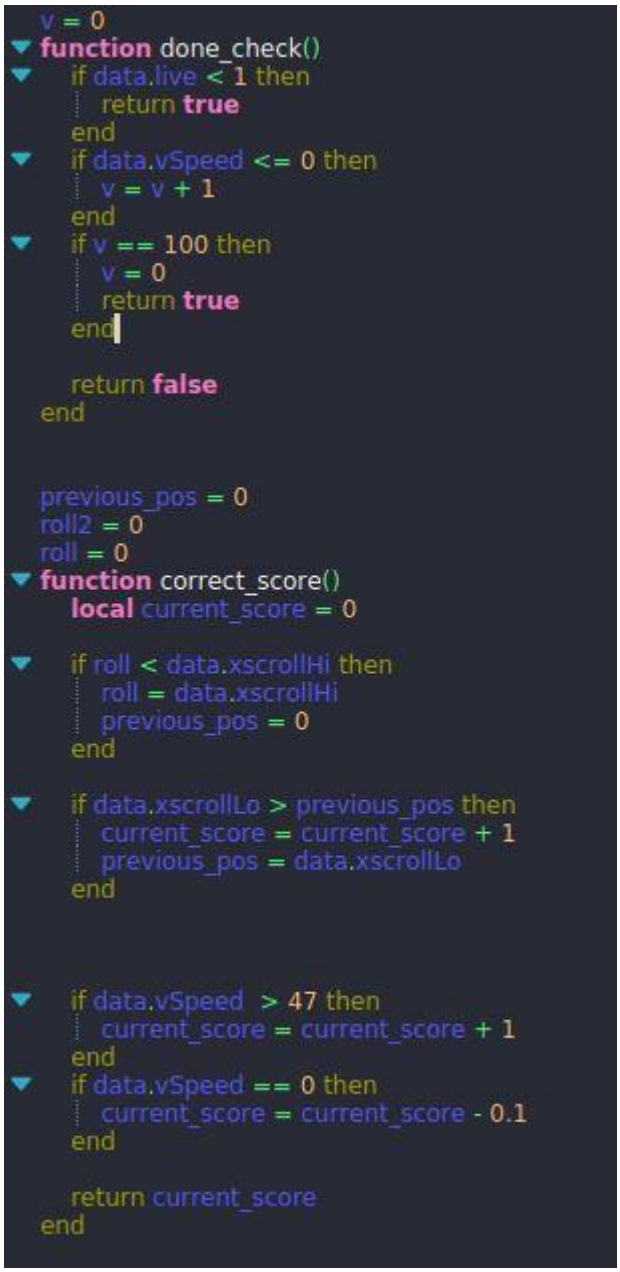
```
obs = env.reset()
while not done:
    action, _states = model.predict(obs)
    obs, rew, dones, info = env.step(action)
    env.render()
```

Kuvio 9. Pelin pelaaminen ja renderöinti.

4.4 Säännöt ja opetus

Ohjelman tavoite on suorittaa Super Mario Bros -pelin taso mahdollisimman nopeasti. Ohjelmalle ei ole annettu tietoa, mitä pelissä pitäisi tehdä. Opetuksen aloitusvaiheessa agentti ohjaa pelihahmoa satunnaisesti, kunnes se tekee toiminnon, josta agentti ansaitsee pisteen. Sääntöinä ensimmäisessä opetuksessa oli yksi piste joka kerta, kun pelihahmo siirtyy vaakatasolla oikealle. Valmiussääntö oli pelihahmon kuolema. 100 000 askeleen opetus näillä säännöillä tuotti tuloksen, jossa pelihahmo eteni hitaasti ja jäi kävelemään seinää vastaan. Agentti ei myöskään osannut liikuttaa pelihahmoa kauas pelatussa kentässä. Tämän ongelma vältetään asettamalla toinen valmiussääntö. Toinen valmiussääntö aloittaa pelin alusta, jos pelihahmo on seisonut paikallaan 100 kehystä. Sääntöihin lisättiin myös rangaistus, jos pelihahmo seisoo paikallaan. Edeltävät säännöt olivat tiedostossa json-muodossa, mutta uudet monimutkaisemmat säännöt oli helpompi toteuttaa Lua-ohjelmointikielellä. Uusilla säännöillä ohjelma pystyi oppimaan nopeammin.

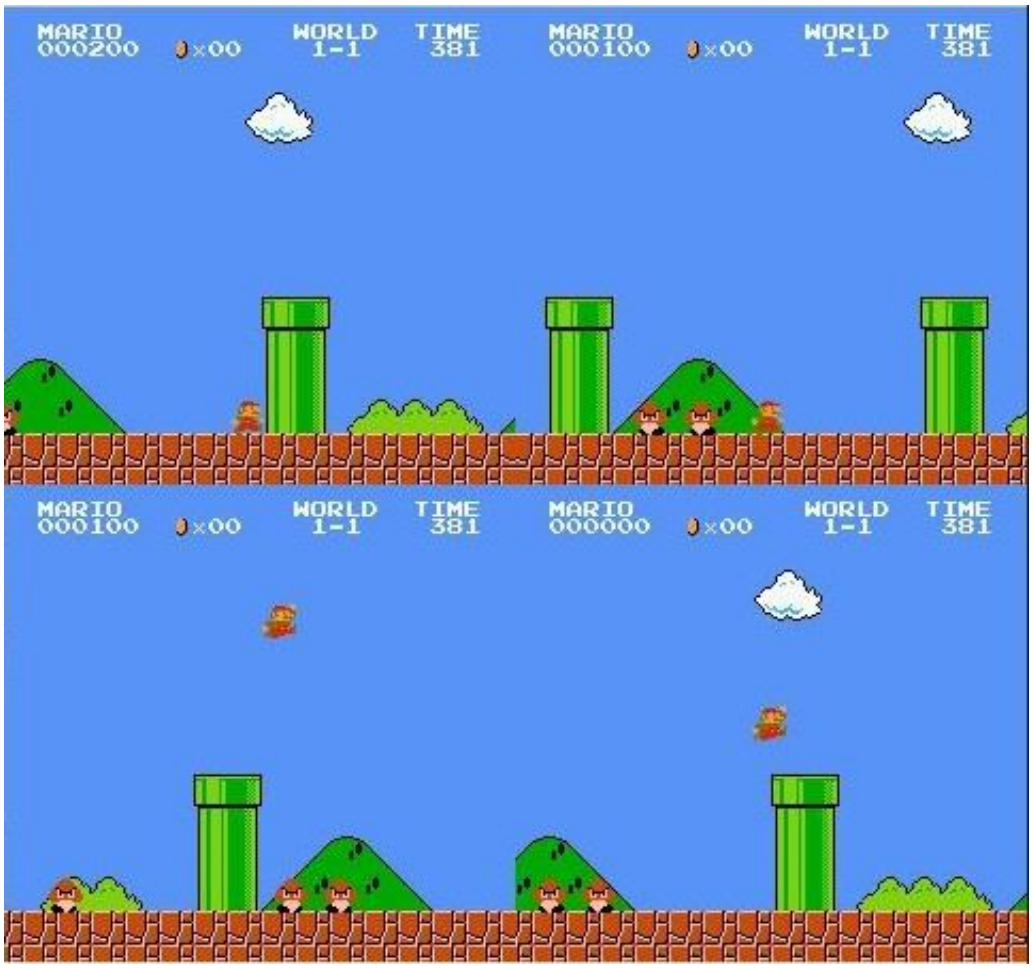
Lopulliset säännöt ja valmiudet on esitetty kuviossa 10. Valmiussääntöjä on kaksi, jotka ovat pelihahmon kuolema ja pelihahmon seisominen 100 kehystä paikallaan. Ohjelma sai säännöistä palkintoja, kun pelihahmo liikkuu oikealle ja silloin kun pelihahmon juoksunopeus on suurin mahdollinen. Rangaistuspisteitä ohjelma saa, kun pelihahmo seisoo paikallaan.



```
v = 0
function done_check()
    if data.live < 1 then
        return true
    end
    if data.vSpeed <= 0 then
        v = v + 1
    end
    if v == 100 then
        v = 0
        return true
    end

    return false
end


previous_pos = 0
roll2 = 0
roll = 0
function correct_score()
    local current_score = 0

    if roll < data.xscrollHi then
        roll = data.xscrollHi
        previous_pos = 0
    end

    if data.xscrollLo > previous_pos then
        current_score = current_score + 1
        previous_pos = data.xscrollLo
    end



    if data.vSpeed  > 47 then
        current_score = current_score + 1
    end
    if data.vSpeed == 0 then
        current_score = current_score - 0.1
    end

    return current_score
end
```

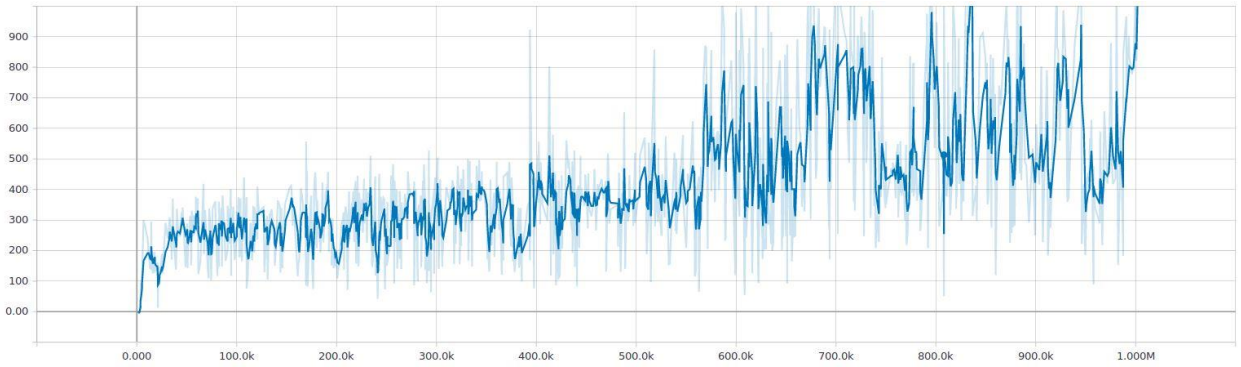
Kuvio 10. Ohjelman valmius ja säännöt.

Kuvion 10 säännöillä ja viidellä miljoonalla askeleella agentti oppi pelaamaan peliä niin hyvin, että se läpäisee tason. Koulutusvaiheessa käytettiin prosessorin kaikkia neljää ydintä, mikä nopeutti oppimista. Ohjelma pystyy oppimaan ja pelaamaan kuvion 11 mukaan. Samanaikaisen oppimisen ja pelaamisen määrää prosessori ja sen ydinten määrä.



Kuvio 11. Useampi peli samanaikaisesti.

Opetusvaiheessa on käytössä Tensorboard, joka esittää kuvion 12 mukaisesti ohjelman saavutetut palkintojen määrän, askelten suhteen. Kuviossa 12 on esitetty pisteiden määrä pystysuoralla parametrillä ja askelten määrä vaakasuoralla parametrillä.



Kuvio 12. Tensorboard-palkintojen määrä askeleittain.

Opetuksen alussa agentin pitää löytää oikea suunta, johon kävellä. Opetus tapahtui ensimmäisessä tasossa. Tasossa toisena asiana ohjelman pitää oppia tunnistamaan ja välttämään viholliset. Pienellä askelmäärällä ohjelma ei oppinut välttämään vihollista, vaan pelihahmo hyppelehti satunnaisesti edetäkseen. Tasossa seuraavana esteenä on seinä, jonka yli pelihahmon pitää oppia hypäämään. Viiden miljoonan askeleen opetuksen jälkeen ohjelma on oppinut välttämään vihollisia, sekä juoksemaan mahdollisimman nopeasti kentän läpi. Tällä oppimismäärällä ohjelma ei läpäise tasoa joka kerta, mutta pystyy läpäisemään sen. Ohjelma on oppinut tunnistamaan viholliset ja niiden yli hyppäämisen. Ohjelma on myös oppinut juoksemisen ja hyppypituuden säätämisen. Hyppypituudella ja nopeudella ohjelma osaa ohjata hahmoa siten, että pelihahmo välttyy esteiltä ja pitää nopeutensa.

5 YHTEENVETO JA POHDINTAA

Opinnäytetyön tavoitteena oli tutustua koneoppimiseen ja kehittää ohjelma, joka oppii pelaamaan peliä ilman erillisiä ohjeita. Työssä on kerrottu mitä tekoäly ja koneoppiminen on ja tarkemmin niiden kategorioista. Opinnäytetyön lopputuloksena on ohjelma, jolla on kyky oppia pelaamaan peliä omalla kokemuksellaan.

Työssä perehdyttiin koneoppimistekniikoihin ja mihin tarkoitukseen eri koneoppimistyylejä käytetään. Työn aikana opinnäytetyön tekijä on opiskellut ja tutustunut koneoppimiseen internetin avulla.

Suurin haaste työssä oli koneoppimiseen tutustuminen ja ohjelman sääntöjen laatiminen. Ohjelmalle annetut säännöt määräävät, mitä ohjelma yrittää suorittaa. Pienikin virhe säännöissä voi pilata oppimisprosessin, koska ohjelma voi löytää virheen ja käyttää sitä hyväksi ei-halutulla tavalla. Esimerkiksi virhe tapahtui json-muodossa olevista säännöistä Lua-ohjelmointikielellä tehtyihin sääntöihin. Virheen tuloksena pelihahmo juoksee suoraan ensimmäistä vihollista päin ja aloittaa tason alusta. Virhe tapahtui muistiosoitteen arvon muuttumisen väärinymmärryksestä. Korjauksen jälkeen lyhyemmällä opetusajalla ohjelma eteni pidemmälle kentässä kuin aikaisemmin. Vahvistusoppimisessa haastavinta oli luoda sopivat säännöt palkinnoille.

Aiheeksi valittiin koneoppiminen, koska sen käyttö on yleistynyt viime vuosina, kun koneoppimisalgoritmit ovat kehittyneet. Nykyisin koneoppimista käytetään yksinkertaisten sekä monimutkaisten ongelmien ratkaisuun.

Opinnäytetyö antoi hyvän ymmärryksen koneoppimisesta ja niiden tekniikoista.

LÄHTEET

Achiam, J. 2018. Proximal Policy Optimization. [www-lähde]. OpenAI. [Viitattu 16.2.2019]. Saatavissa: https://github.com/openai/spinningup/blob/master/docs/algorithms/ppo.rst

Anderson, R. M. 2017. Twenty years on from Deep Blue vs Kasparov: how a chess match started the big data revolution. [www-lähde]. theconversation. [Viitattu 13.2.2019]. Saatavissa: https://theconversation.com/twenty-years-on-from-deep-blue-vs-kasparov-how-a-chess-match-started-the-big-data-revolution-76882

Bedford, S. 2016. Machine Learning & Food Classification. [www-lähde]. Bedford. [Viitattu 15.2.2019]. Saatavissa: https://simonb83.github.io/machine-learning-food-classification.html

Bradford, L. 2018. What Should I Learn as a Beginner: Python 2 or Python 3?. [www-lähde]. Learn to Code With Me. [Viitattu 19.2.2019]. Saatavissa: https://learntocodewith.me/programming/python/python-2-vs-python-3/

Brockman, G., Sutskever, I. & OpenAI. 2015. Introducing OpenAI. [www-lähde]. OpenAI. [Viitattu 3.1.2019]. Saatavissa: https://blog.openai.com/introducing-openai/

Copeland, B.J. 2018. Artificial intelligence. [www-lähde]. Encyclopædia Britannica, Inc. [Viitattu 6.1.2019]. Saatavissa: https://www.britannica.com/technology/artificial-intelligence

Gies, A. 2016. A normal person's guide to watching Dota 2. [www-lähde]. Polygon. [Viitattu 25.1.2019]. Saatavissa: https://www.polygon.com/2016/8/8/12401068/dota-2-watching-guide

Hesse, C., Schulman, J., Pfau, V., Nichol, A., Klimov, O. & Schiavo, L. 2018a. Retro Contest. [www-lähde]. OpenAI. [Viitattu 19.1.2019]. Saatavissa: https://blog.openai.com/retro-contest/

Hesse, C., Schulman, J., Pfau, V., Nichol, A., Klimov, O. & Schiavo, L. 2018b. Retro Contest. [www-lähde]. OpenAI. [Viitattu 16.2.2019]. Saatavissa: https://blog.openai.com/first-retro-contest-retrospective/

Itewiki. Ei päiväystä. Koneoppiminen. [www-lähde]. Ite wiki oy. [Viitattu 8.1.2019] Saatavissa: https://www.itewiki.fi/opas/koneoppiminen/

Khatun, A. 2018. Let's know Supervised and Unsupervised in an easy way. [www-lähde]. Chatbots Magazine. [Viitattu 15.2.2019]. Saatavissa: https://chatbots-magazine.com/lets-know-supervised-and-unsupervised-in-an-easy-way-9168363e06ab

Leon, K. 2017. Making a Simple Neural Network. [www-lähde]. Medium. [Viitattu 16.2.2019]. Saatavissa: https://becominghuman.ai/making-a-simple-neural-network-2ea1de81ec20

Lua. 2018. About. [www-lähde]. PUC-Rio. [Viitattu 29.1.2019] Saatavissa: https://www.lua.org/about.html

Maini, V. 2017. Machine Learning for Humans, Part 5: Reinforcement Learning. [www-lähde]. Medium. [Viitattu 15.1.2019]. Saatavissa: https://medium.com/machine-learning-for-humans/reinforcement-learning-6eacf258b265

OpenAI. 2018a. Gym Retro. [www-lähde]. OpenAI. [Viitattu 10.1.2019]. Saatavissa: https://github.com/openai/retro/blob/master/README.md

OpenAI. 2018b. OpenAI Five. [www-lähde]. OpenAI. [Viitattu 24.1.2019]. Saatavissa: https://blog.openai.com/openai-five/

OpenAI. 2018c. OpenAI Five Benchmark: Results. [www-lähde]. OpenAI. [Viitattu 25.1.2019]. Saatavissa: https://blog.openai.com/openai-five-benchmark-results/

Patel, P. 2018. Why Python is the most popular language used for Machine Learning. [www-lähde]. Udacity India. [Viitattu 11.3.2019]. Saatavissa: https://medium.com/@UdacityINDIA/why-use-python-for-machine-learning-e4b0b4457a77

Pfau, V., Nichol, A., Hesse, C., Schiavo, L. & Klimov, O. 2018. Gym Retro. [www-lähde]. OpenAI. [Viitattu 4.1.2019]. Saatavissa: https://blog.openai.com/gym-retro/

Python. Ei päiväystä. General Python FAQ. [www-lähde]. Python Software Foundation. [Viitattu 2.2.2019]. Saatavissa: https://docs.python.org/3/faq/general.html

Python. Ei päiväystä. PEP 373 -- Python 2.7 Release Schedule. [www-lähde]. Python Software Foundation. [Viitattu 19.2.2019]. Saatavissa: https://www.python.org/dev/peps/pep-0373/

Python. Ei päiväystä. Python releases by version number. [www-lähde]. Python Software Foundation. [Viitattu 3.2.2019]. Saatavissa: https://www.python.org/downloads/

Radcliffe, T. Ei päiväystä. Lua: Not Your Average Scripting Language. [www-lähde]. ActiveState. [Viitattu 19.2.2019]. Saatavissa: https://www.activestate.com/blog/lua-not-your-average-scripting-language/

Reaktor. Ei päiväystä. Koneoppimisen lajit. [www-lähde]. Reaktor Innovations Oy. [Viitattu 5.1.2019]. Saatavissa: https://course.elementsofai.com/fi/4/1

Reaktor. Ei päiväystä. Neuroverkkojen periaatteet. [www-lähde]. Reaktor Innovations Oy. [Viitattu 10.1.2019]. Saatavissa: https://course.elementsofai.com/fi/5/1

Reinforcement learning diagram. 2017. [Kuva]. Wikimedia Commons. [Viitattu 19.3.2019]. Saatavissa: https://commons.wikimedia.org/wiki/File:Reinforcement_learning_diagram.svg

Rouse, M. 2018a. AI (artificial intelligence). [www-lähde]. TechTarget. [Viitattu 13.2.2019]. Saatavissa: https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence

Rouse, M. 2018b. machine learning (ML). [www-lähde]. TechTarget. [Viitattu 14.2.2019]. Saatavissa: https://searchenterpriseai.techtarget.com/definition/machine-learning-ML

Sculman, J., Klimov, O., Wolski, F., Dhariwal, P. & Radford, A. 2017. Proximal Policy Optimization. OpenAI [www-lähde]. OpenAI [Viitattu 12.1.2019]. Saatavissa: https://blog.openai.com/openai-baselines-ppo/

Simonini, T. 2018a. An introduction to Reinforcement Learning. [www-lähde]. FreeCodeCamp. [Viitattu 15.1.2019]. Saatavissa: https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419

Simonini, T. 2018b. Proximal Policy Optimization (PPO) with Sonic the Hedgehog 2 and 3. [www-lähde]. Medium. [Viitattu 16.2.2019]. Saatavissa: https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e

StackOverflow. 2017. What is a policy in reinforcement learning?. [www-lähde]. StackOverflow. [Viitattu 16.2.2019]. Saatavissa: https://stackoverflow.com/questions/46260775/what-is-a-policy-in-reinforcement-learning

Voskoglou, C. 2017. What is the best programming language for Machine Learning?. [www-lähde]. Developer Economics. [Viitattu 11.3.2019]. Saatavissa: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7