Petri Monola

Generatiivisen kilpailevan verkon soveltaminen Conwayn Game of Life -soluautomaattiin

Tietotekniikan pro gradu -tutkielma

12. kesäkuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Petri Monola

Yhteystiedot: petri.k.monola@student.jyu.fi

Ohjaajat: Tommi Kärkkäinen ja Joonas Hämäläinen

Työn nimi: Generatiivisen kilpailevan verkon soveltaminen Conwayn Game of Life -soluautomaattiin

Title in English: Using Generative Adversarial Networks to play Conway's Game of Life

Työ: Pro gradu -tutkielma

Opintosuunta: Tietoliikennetekniikka

Sivumäärä: 43+0

Tiivistelmä: Generatiiviset kilpailevat verkot ovat yksi tuoreimmista koneoppimisen kehityksistä ja sillä on saatu lupaavia tuloksia kuvien generoinnissa. Tässä tutkielmassa tarkastellaan GAN-menetelmää ja sen ominaisuuksia sekä siihen pohjautuvan Pix2pix-menetelmän toimivuutta pelatessa Conwayn Game of Life -peliä. Pix2pix-menetelmä pystyy generoimaan tarkasti pelin sukupolvia, noin 2,49% virheellä.

Avainsanat: Generatiiviset kilpailevat verkot, GAN, neuroverkko, puoliohjattu oppiminen, kuvanluokittelu, koneoppiminen, soluautomaatti, Game of life, DCGAN, syväoppiminen

Abstract: In recent years Generative Adversarial Networks have shown interesting developement in generating realistic images. In this thesis we study GAN and it's capabilities while also applying it in a new setting using Pix2pix to play Conway's Game of life. The network is able to generate accurate generations to the game with about 2.49% error.

Keywords: Generative Adversarial Networks, GAN, neural network, adversarial learning, semi-supervised learning, image classification, machine learning, Cellular automaton, Game of Life, DCGAN, deep learning

Termiluettelo

Activation function	Aktivaatiofunktio
Back Propagation	Vastavirta-algoritmi
Cellular automaton	Soluautomaatti
CNN	Convolutional Neural Network
DCGAN	Deep Convlutional Generative Adversarial Network
Deep learning	Syväoppiminen
Error function	Virhefunktio
Feedforward neural network	Eteenpäinkytketty neuroverkko
GAN	Generative Adversarial Network
Label	Leima
Layer	Kerros
Multilayer Perceptron, MLP	Monikerroksinen perseptroni
Neuron	Neuroni
Learning system	Oppiva järjestelmä
Preprocessing	Esikäsittely
Recurrent neural network	Takaisinkytketty neuroverkko
Reinforcement learning	Vahvistava oppiminen
ReLU	Rectified Linear Unit
Script	Skripti
Semi-supervised learning	Puoliohjattu oppiminen
Supervised learning	Ohjattu oppiminen
Unsupervised learning	Ohjaamaton oppiminen

Kuviot

Taulukot

Sisältö

1 Johdanto

Koneoppiminen on olennainen osa tämän päivän järjestelmiä. Tulevaisuudessa niiden käyttö tulee lisääntymään, koska niillä voidaan ratkoa entistä monimutkaisempia ongelmia, jotka ovat liian vaikeita tai hitaita ihmiselle ratkaistavaksi. Jatkuvasti kehittyvä tekoäly luo uusia mahdollisuuksia ja sitä kautta myös haasteita, siksi koneoppimisen tutkiminen on kannattavaa. Koneoppimisella on pystytty luomaan järjestelmiä, jotka kykenevät entistä paremmin luokittelemaan erilaista tietoa, muun muassa kuvia, ääntä, anomalioita, liikennettä ja haittaohjelmia (Goodfellow, Bengio ja Courville 2016, 96-100). Koneoppimista hyödyntävät menetelmät ovat jo kehittyneet tietyillä teknologian osa-alueilla sellaiselle tasolle, että muut menetelmät eivät kykene ratkomaan ongelmia yhtä tehokkaasti (Goodfellow, Bengio ja Courville 2016, 96). Tulevaisuudessa on todennäköistä, että vain oppivat järjestelmät keskenään voivat kilpailla toisiaan vastaan, sillä muut menetelmät eivät pärjää kamppailussa.

Koneoppimisen yksi tuoreimmista menetelmistä on GAN-menetelmä. Generatiiviset kilpailevat verkot eli GAN *(engl. Generative Adversarial Networks)* koostuvat generaattorista ja diskriminaattorista, jotka kilpailevat keskenään generaattorin yrittäessä huijata diskriminaattoria ja diskriminaattorin yrittäessä tunnistaa generoidut kuvat aidoista kuvista. GAN-menetelmä on Goodfellow ym. (2014) kehittämä menetelmä ja sen tarkoituksena on luoda mahdollisimman aidonnäköisiä kuvia. Sitä on sovellettu onnistuneesti muun muassa kuvan- ja videonkäsittelyn maailmassa, missä menetelmällä on kyetty muodostamaan parempia suodattimia kun muilla menetelmillä (Gopan ja Kumar 2018; Li ym. 2018; Zhao, Weng ja Liu 2017; Vondrick, Pirsiavash ja Torralba 2016). Lisäksi GAN-menetelmällä on kyetty ohittamaan hyökkäyksentunnistusjärjestelmä *(engl. Intrusion Detection System)* mallintamalla aitoa verkkoliikennettä ja toimimaan haitallisesti järjestelmän sisällä (Rigaki ja Garcia 2018).

GAN-menetelmä pystyy omalla lähestymistavallaan luomaan uutta aidonnäköistä dataa opetusdatan perusteella, jolloin sitä voidaan hyödyntää kohteissa, joissa opetusdataa on niukasti saatavilla. Tässä tutkielmassa tutkitaan GAN-menetelmän soveltuvuutta generoidessa Conwayn Game of Life -soluautomaatin sukupolvia. Generatiivisella mallilla pyritään luomaan aidonnäköisiä kuvia soluautomaatista, jonka jälkeen analysoidaan luotujen verkkojen suorituskykyä vertaamalla tuloksia soluautomaatin todellisiin iteraatioihin.

2 Tutkimuskysymykset ja motivaatio

Perinteisellä ohjatulla oppimisella sekä moderneilla syvillä neuroverkoilla pystytään saavuttamaan tarkkoja tuloksia luokitteluongelmissa, mutta ongelmaksi saattaa muodostua datan puute tietyillä alueilla. GAN-menetelmän avulla voidaan opettaa neuroverkkoja, jotka pystyvät luomaan otteita puuttuvan datan alueista. Esimerkiksi puoliohjatussa oppimisessa datasta puuttuu osa opetusdatan leimoista, mutta GAN-menetelmällä opetetut neuroverkot pystyvät kehittämään luokitteluaan ilman leimoja (Goodfellow 2017). GAN-menetelmää voidaan siis hyödyntää täydentämään puuttuvaa tietoa sekä tukemaan muita koneoppimisen menetelmiä.

Kargaard ym. (2018) ovat esitelleet tutkimuksessa, että GAN menetelmällä voidaan tunnistaa uusia Zero-Day-haittaohjelmia tehokkaasti. Toisaalta (Rigaki ja Garcia 2018) ovat tutkineet kuinka GAN-menetelmällä voi jäljitellä oikeaa liikennettä siten, että IPS ei tunnista tai estä haitallista liikennettä. Näiden tuloksien perusteella voidaan olettaa, että on mahdollista muodostaa verkkohyökkäys, joka läpäisee tarkimmatkin turvakeinot ja GAN-menetelmä voi osoittautua olennaiseksi työkaluksi turvallisuusjärjestelmien kehittämisessä.

Tutkimusten käyttämistä eri menetelmistä ja niiden tuloksista huomataan, että ei ole olemassa yhdenlaista kokoonpanoa, jonka suorituskyky olisi ylitse muiden. Siksi on tärkeää testata johdonmukaisesti erilaisia kokoonpanoja erilaisiin ongelmiin ja raportoida tulokset. Näistä tuloksista voidaan taas koota uutta tietoa GAN-menetelmän käyttäytymisestä ja soveltuvuudesta.

Päätarkoitus tällä tutkielmalla on koota tietoa koneoppimisesta ja GAN-menetelmästä sekä sen soveltuvuudesta generoimaan uudenlaista tietoa. Tutkimuksessa pyritään muodostamaan kokonaiskuvaa siitä, että mihin GAN-menetelmää voi soveltaa ja mikä sen suorituskyky on. GAN-menetelmä on tuore ja sen tunteminen vaatii lisää tutkimusmateriaalia, siksi tutkielman tutkimusosuudessa tutkitaan GAN-menetelmän soveltuvuutta soluautomaatissa Pix2pix-menetelmällä ja pyritään selvittämään, onko Pix2pix-menetelmällä opetettu generaattori riittävän hyvä iteroimaan soluautomaattia.

3 Koneoppiminen

Koneoppiminen voidaan määritellä ohjelmana, joka pyrkii suorittamaan tehtävän opitun kokemuksen avulla ja sen kyky suorittaa tehtävä paranee kokemuksen kasvaessa (Goodfellow, Bengio ja Courville 2016, 96). Monikerroksiset perseptronit *(engl. Multilayer Perceptron, MLP)* ovat käytetty koneoppimismenetelmä ja niistä voidaan käyttää vain nimitystä neuroverkot.

Tehtävät joihin liittyy yleistämistä, todennäköisyyksiä ja tilastoja ovat sopivia kohteita käyttää koneoppimista (Alpaydin 2010, 1-4). Koneoppiminen on olennainen tapa ratkaista erilaisia luokitteluongelmia, erityisesti koneoppimista on käytetty kuvantunnistuksessa. Kuvantunnistus on ihmiselle luontainen taito, ja sitä on vaikea selittää kuinka sen teemme, siksi pyrimme muodostamaan tapoja, joilla pystymme mallintamaan omaa toimintaamme (Alpaydin 2010, 1-4). Kuvantunnistuksessa tarkoituksena on tunnistaa kuvasta piirteitä, jotka todennäköisimmin kuuluvat jollekin tietylle luokalle. Parhaimpiin menetelmiin kuvantunnistuksessa kuuluu muun muassa konvoluutioneuroverkot.

Konvoluutioneuroverkkojen verkkojen idea perustuu neurobiologisiin havaintoihin, jotka löydettiin Hubel ja Wiesel (1962) toimesta tutkiessaan kissojen visuaalista aivokuorta. Konvoluutioneuroverkot ovat esimerkki syvistä neuroverkoista, joissa on useita kerroksia syötekerroksen ja ulostulokerroksen välissä. Syvien neuroverkkojen koneoppimisalgoritmeista käytetään myös nimitystä syväoppimisalgoritmit *(engl. deep learning algorithms)*. Neuroverkkoihin tutustutaan tarkemmin luvuissa 3.2 ja 3.2.2.

3.1 Koneoppimisen menetelmät

Koneoppimisen menetelmät voidaan jakaa eri tyyppeihin rakenteen, oppimistavan tai tehtävän mukaan. Vaikka ohjatun- ja ohjaamattoman oppimisen välillä ei ole formaalia ja tarkkaa määritelmää, ne voidaan jakaa sen perusteella, onko datan merkkaamisessa käytetty ihmistä (Goodfellow, Bengio ja Courville 2016, 142). Vaikka on olemassa eri tapoja lajitella koneoppimisen menetelmiä, on tässä tutkielmassa valittu lajitteluksi ohjattu oppiminen, ohjaamaton oppiminen ja vahvistava oppiminen. Tässä luvussa tarkastellaan näiden menetelmien omi-

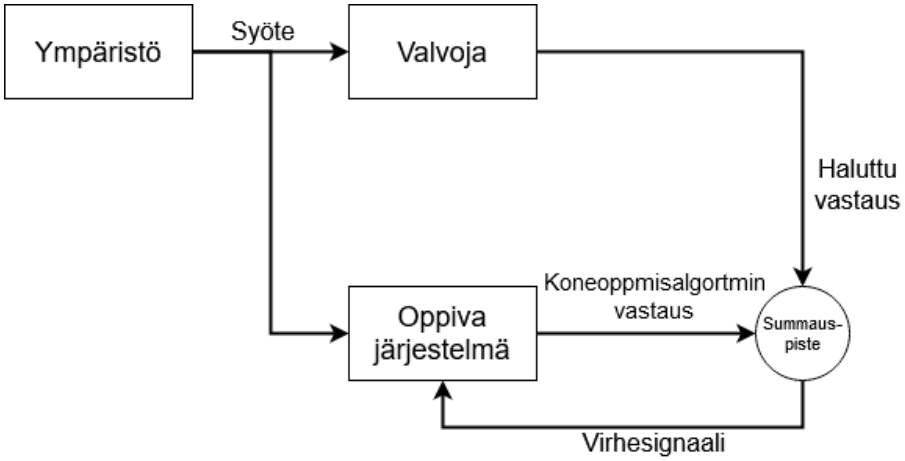
naisuuksia, tehtäviä ja suorituskykyä.

Koneoppimista pyritään hyödyntämään useiden eri ongelmien ratkaisemiseen, joita ovat esimerkiksi luokittelu, regressio, transkriptointi, käännös, strukturointi, anomaliantunnistus, syntetisointi, kohinanpoisto ja tiheysestimointi (Goodfellow, Bengio ja Courville 2016, 96-100).

Kuten tämän kappaleen johdannossa mainittiin, on olemassa eri tapoja luokitella koneoppimisen menetelmiä, muun muassa tehtävän mukaan, esimerkiksi tiedonlouhinnassa voidaan käyttää sekä ohjattua oppimista että ohjaamatonta oppimista. Koneoppimisalgoritmeissa voidaan myös yhdistää koneoppimisen menetelmiä, jolloin saadaan esimerkiksi puoliohjattu oppiminen *(engl. semi-supervised learning)*. Puoliohjatussa oppimisessa hyödynnetään sekä ohjattua oppimista että ohjaamatonta oppimista.

3.1.1 Ohjattu oppiminen

Ohjatussa oppimisessa tavoitteena on oppia yhdistämään syöte annettuun ulostuloon, jotka ovat ennalta määritettyjä (Alpaydin 2010, 11-13). Ohjattu oppiminen toimii hyvin luokittelemaan silloin, kun dataan on merkattu leimat, muissa tapauksissa joudutaan käyttämään joko muita menetelmiä tai yhdistelmiä muista menetelmistä. Ohjatun oppimisen menetelmissä verkon oppimista seurataan virhefunktion avulla, jonka tehtävä on mitata kuinka kaukana verkon vastaus on halutusta vastauksesta (Haykin ym. 2009, 65). Verkon käyttäytymistä muokataan yhdessä opetusvektorin ja virhesignaalin kanssa. Kuvio 1 on yksinkertainen malli ohjatusta oppimisesta, missä syötteenä on yksittäinen havainto ympäristöstä. Havainnolle määritetään haluttu vastaus valvojan toimesta ja oppivan järjestelmän vastaus, jonka jälkeen summauspisteessä lasketaan koneppimisalgoritmin virhesignaali, jonka avulla oppiva järjestelmä oppii. Neuroverkkojen oppimista käsitellään tarkemmin luvussa 3.2.2.



Kuvio 1. Malli ohjatusta oppimisesta

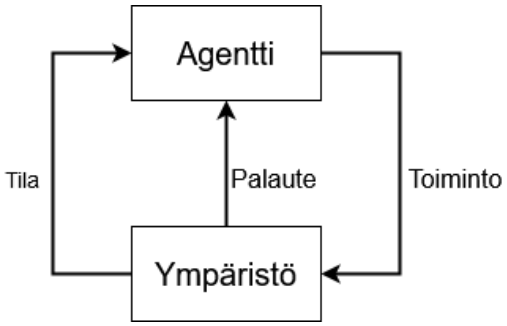
3.1.2 Ohjaamaton oppiminen

Ohjaamattomassa oppimisessa ei ole valvojaa, kuten ohjatussa oppimisessa, joka määrittäisi oikeat tulokset vaan tarkoituksena on löytää syötteen säännöllisyyksiä (Alpaydin 2010, 1-4). Ohjaamattoman oppimisen algoritmit tekevät tätä eri tavoin: esimerkiksi klusteroinnissa datajoukko jaetaan klustereihin, jotka koostuvat samanlaisista näytteistä. Ohjaamatonta oppimista käytetään silloin kun opetusdataan ei ole käytännöllistä merkata leimoja tai jos ohjattu oppiminen ei ole tarpeellista, esimerkiksi autoenkoodaajan tapauksessa. Syväoppimisen yhteydessä tarkoituksena on oppia koko todennäköisyysjakauma, joka muodostaa datajoukko (Goodfellow, Bengio ja Courville 2016, 102-103). GAN-menetelmässä voidaan luokitella ohjaamattomaksi oppimiseksi, koska siinä opetetut neuroverkot saavat palautteen toisiltaan, mutta toisaalta GAN-menetelmässä voidaan hyödyntää myös ohjattua oppimista, kuten esimerkiksi Pix2pix-menetelmässä, jolloin se voidaan luokitella puoliohjatun oppimisen menetelmäksi.

3.1.3 Vahvistava oppiminen

Vahvistavassa oppimisessa *(engl. reinforcement learning)* koneoppimisalgoritmi on vuorovaikutuksessa ympäristön kanssa ja kerää sieltä palautetta toiminnastaan (Goodfellow, Bengio ja Courville 2016, 103). Agentit ovat tyypillinen esimerkki vahvistavan oppimisen algoritmista, missä agentti tekee toiminnon, joka vaikuttaa ympäristöön, minkä jälkeen agentti

tulkitsee ympäristön tilan muutosta ja korjaa toimintojaan. Agentin toimintaa voidaan mallintaa Markovin valintaprosessilla, jossa agentti toimii diskreetin ajan ympäristössä, missä jokaisessa ympäristön tilassa agentilla on äärellinen määrä valintoja, jotka vaikuttavat ympäristön tilaan (Haykin ym. 2009, 657). Kuviossa 2 on yksinkertainen malli agentin toiminnasta. Agentit voivat käyttää esimerkiksi kuvantunnistusta tai puheentunnistusta kerätessään tietoa ympäristöstä (Mnih ym. 2013). Vahvistavalla oppimisella on pystytty muun muassa pelaamaan vanhoja Atarin pelejä (Mnih ym. 2013).



Kuvio 2. Agentin vuorovaikutus ympäristön kanssa

3.2 Neuroverkot

Neuroverkko on laskennallinen järjestelmä, joka pyrkii mallintamaan biologista neuroverkkoa eli aivoja. Aivot ovat monimutkainen tiedonkäsittelijä, joka on epälineaarinen ja kykenee rinnakkaislaskentaan (Haykin ym. 2009, 31-32). Ihminen kykenee aivojensa avulla tehokkaasti ratkaisemaan useita erilaisia ongelmia ja oppimaan uusia asioita. Rutiininomaisesti ihminen tekee muun muassa kuvantunnistusta *(engl. perceptual recognition)*, johon nopeimmillakin tietokoneilla menee huomattavasti enemmän aikaa (Haykin ym. 2009, 31-32). Aivojen kyky prosessoida tietoa on niin tehokas, että sitä on pyritty mallintamaan neuroverkkojen muodossa.

Neuroverkot pyrkivät tekemään päätöksiä niiden oppimansa mukaan. Neuroverkot opetetaan suurella määrällä dataa ja tyypillisesti ohjatussa oppimisessa dataan merkataan, että mitkä havainnot *(engl. samples)* kuuluvat mihinkin luokkaan. Tällaiset neuroverkot ovat tyypiltään luokittelijoita. Tämänlaiset neuroverkot ovat erityisen suosittuja kuvantunnistuksessa niiden tarkkuuden ja tehokkuuden vuoksi.

3.2.1 Rakenne

Neuroni

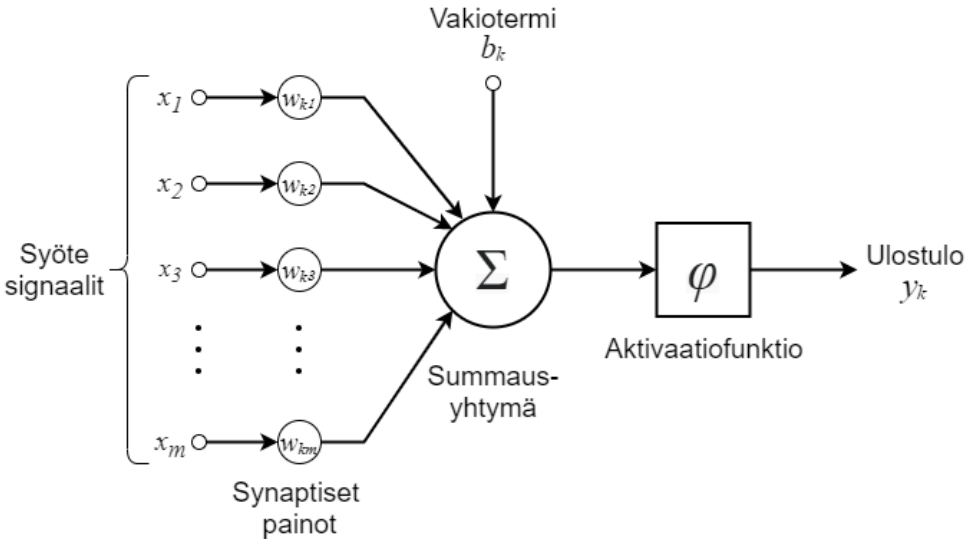
Neuroni on tärkein ja olennaisin osa neuroverkkoa. Neuronin ulostulon laskennassa tarvitaan seuraavia kaavoja:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j,$$

missä $x_1, x_2, \ldots x_m$ ovat syötteet ja u_k synaptisten painoarvojen w_k ja syötteiden x lineaarikombinaatio.

$$y_k = \varphi(u_k + b_k),$$

missä y_k on neuronin ulostulo, φ aktivointifunktio ja b_k vakiotermi (Haykin ym. 2009, 41). Vakiotermin b_k ja neuronin ulostulon u_k summasta käytetään merkintää v_k. Aktivaatiofunktio käsitellään tarkemmin kappaleessa 3.2.2. Kuvio 3 havainnollistaa neuronin rakennetta, syötteitä ja ulostuloa.



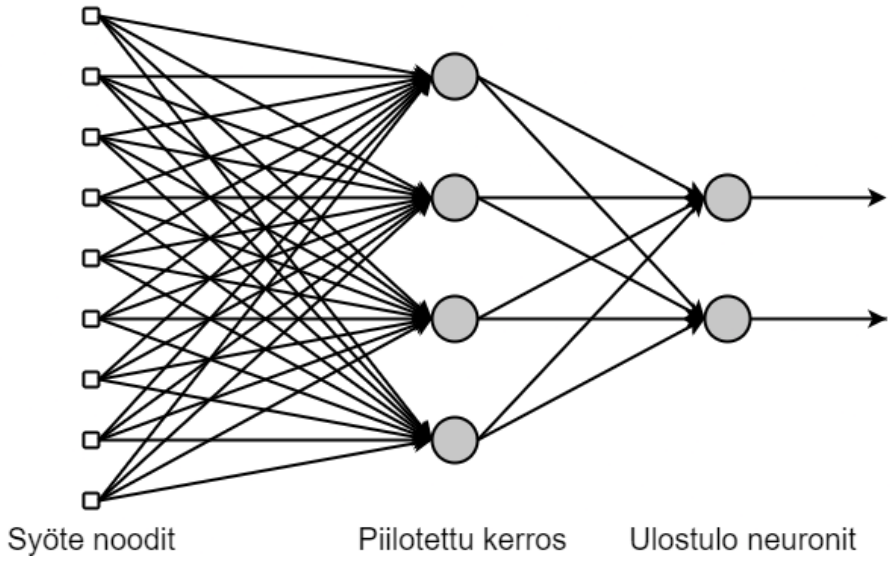
Kuvio 3. Neuroni

Neuroneista voidaan rakentaa verkko yhdistämällä neuroneiden ulostulo toisten neuronien

syötteeseen. Neuroverkkorakenteet voidaan luokitella kahteen arkkitehtuuriluokkaan: eteenpäinkytketyt verkot ja takaisinkytketyt verkot.

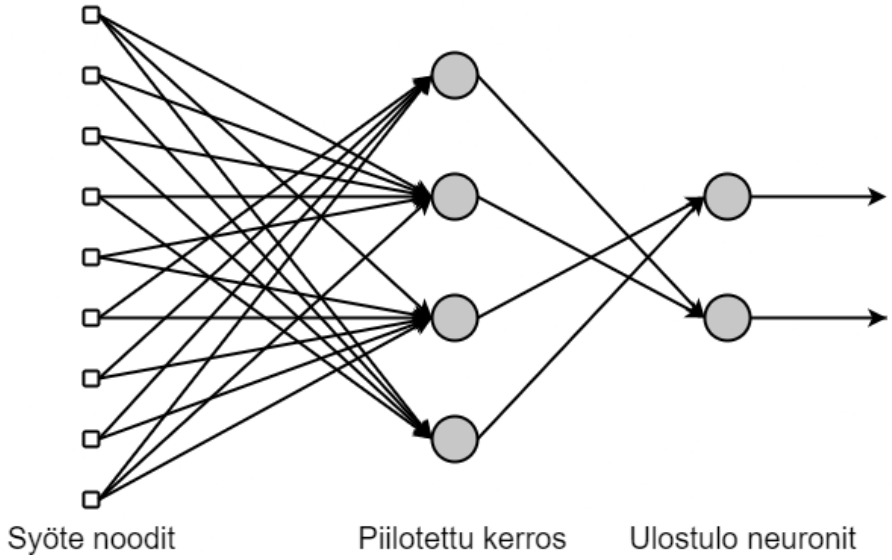
Eteenpäinkytketty verkko

Neuroverkossa voi olla useita piilotettuja kerroksia ulostulo- ja sisääntulokerroksien välissä. Yksinkertaisimmilaan neuroverkossa sisääntulokerros ja ulostulokerros ovat suoraan kytketty yhteen. Verkon kerroksissa neuronit ovat *rivissä* ja verkko pyrkii tallentamaan oppimansa neuroneiden välisten painojen arvoihin.



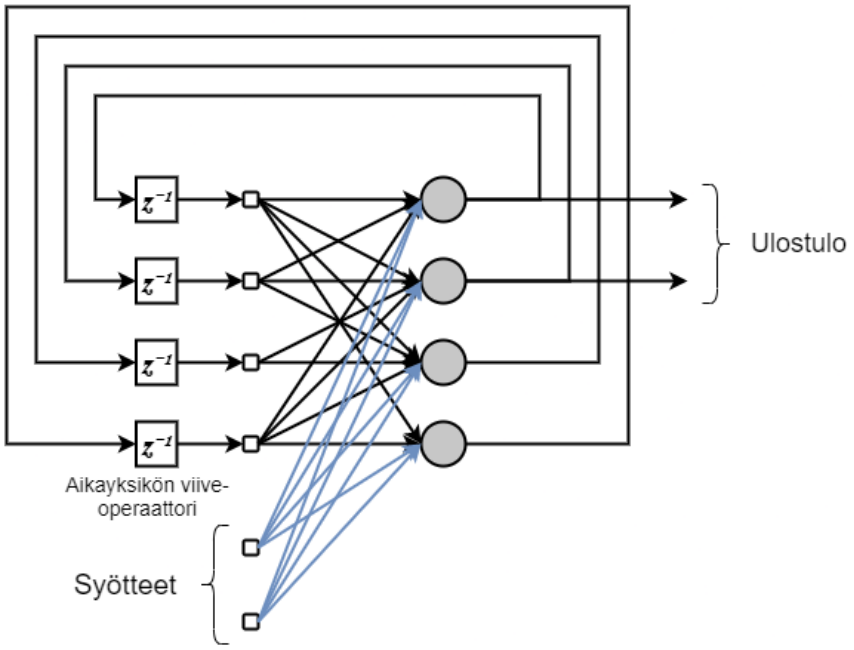
Kuvio 4. Monitasoinen täysin kytketty eteenpäinkytketty verkko

Eteenpäinkytketyn verkon *(engl. feedforward neural network)* kerrokset voi olla joko täysin kytkettyjä kerroksia *(engl. fully connected layer)* kuten kuviossa 4 tai vain osittain kytkettyjä kuten kuviossa 5.



Kuvio 5. Monitasoinen osittain kytketty eteenpäinkytketty verkko

Takaisinkytketty neuroverkko



Kuvio 6. Takaisinkytketty neuroverkko

Takaisinkytketty neuroverkko eroaa myötäkytketystä neuroverkosta siten, että siinä on ainakin yksi palautesilmukka *(engl. feedback loop)* (Haykin ym. 2009, 53). Kuten myötäkytketyssä neuroverkossa, siinä voi olla piilotettuja neuronikerroksia. Takaisinkytketyssä neuroverkossa neuronit eivät saa syötteitä edelliseltä kerrokselta vaan niille syötetään saman kerroksen neuronien ulostulo. Kuvio 6 havainnollistaa takaisinkytketyn verkon rakennetta.

3.2.2 Oppiminen

Verkko tallentaa oppimansa neuroneihin ja niiden välisiin painoarvoihin. Opetuksen aikana neuroverkon tekemät päätökset muuttavat näitä arvoja väärien ja oikeiden valintojen perusteella. Nämä painoarvot kuvastavat biologisten neurovekkojen synapsien voimakkuutta. Jokaisella opetuskierroksella lasketaan virhesignaali, jonka avulla lasketaan neuronien uudet painoarvot. Opetuksessa toistetaan kierroksia kunnes oppiminen tasaantuu ja tulokset eivät muutu. Kaikilla neuroverkkomenetelmillä ja aktivointifunktioilla ei saada samoja tuloksia, siksi useita eri menetelmiä, aktivaatiofunktioita ja rakenteita sovelletaan ongelmiin (Good-

fellow, Bengio ja Courville 2016, 96-100).

Aktivaatiofunktio

Aktivaatiofunktion tarkoituksena on määrittää ulostulon arvo. Aktivaatiofunktio voi olla muotoa

$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases},$$

jolloin se toimii kynnysfunktiona (Haykin ym. 2009, 43). Useimmiten aktivaatiofunktiona käytetään sigmoidifunktiota, joka on muotoa

$$\varphi(v) = \frac{1}{1 + exp(-av)},$$

missä a on sigmoidifunktion jyrkkyys (Haykin ym. 2009, 44). Sigmoidifunktio muodostaa S-kirjaimen muotoisen graafin, joka muodostaa pehmeämmän käyttäytymisen (Haykin ym. 2009, 13-14). Lisäksi sigmoidifunktio on derivoituva toisin kuin kynnysfunktio. Aktivaationfunktion derivoituvuus on olennainen vaatimus vastavirta-algoritmin *(engl. backpropagation algorithm)* toiminnassa. Muita käytettyjä aktivaatiofunktioita ovat muun muassa ReLU- ja Tanh-funktiot.

Virhefunktio

Neuroverkon opettamisessa on olennaista määrittää kohdefunktio *(engl. objective function)*, jota kutsutaan tyypillisesti virhefunktioksi *(engl. loss function, cost function)* silloin kuin sitä halutaan minimoida (Goodfellow, Bengio ja Courville 2016, 79-80). Virhefunktiota siis käytetään arvioimaan neuroverkon tekemää virhettä. Tyypillisesti virhefunktioksi muodostuu keskineliösumma *(engl. mean squared error, MSE)* (Goodfellow, Bengio ja Courville 2016, 173).

Olkoon $y_j(n)$ ulostuloneuronin j tuottama ulostulo syötteelle n. Tällöin virhe $e(n)$ lasketaan neuronin j antaman tuloksen $y_j(n)$ ja halutun tuloksen $d_j(n)$ erotuksena (Haykin ym. 2009, 157):

$$e_j(n) = d_j(n) - y_j(n).$$

Koko verkon hetkellinen virhe saadaan kaavalla

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n),$$

missä joukko C sisältää kaikki ulostulokerroksen neuronit (Haykin ym. 2009, 157). Syötteen koostuessa N:stä määrästä syötteitä, tällöin verkon virhefunktio saadaan keskineliösummasta (Haykin ym. 2009, 65,157)

$$E_{av}(N) = \frac{1}{N} \sum_{n=1}^{N} E(n).$$

Käyttämällä keskineliösummaa huomataan, että virhe kasvaa kun euklidinen etäisyys ulostulon ja halutun tuloksen välillä kasvaa (Goodfellow, Bengio ja Courville 2016, 105). Jotta saadaan toimiva koneoppimisalgoritmi, tarvitaan menetelmä, joka muuttaa neuroverkon painoarvoja siten, että keskineliösumma saadaan pienemmäksi (Goodfellow, Bengio ja Courville 2016, 105). Keskineliösumma on lisäksi derivoituva, mikä mahdollistaa gradienttimenetelmän *(engl. gradient descent)* käytön.

Gradienttimenetelmä ja vastavirta-algoritmi

Gradienttimenetelmää käytetään virhefunktion minimiarvon löytämiseen. Se löytää funktion lokaalin minimiarvon ja kun sitä toistetaan eri alkuarvoista lähtien, voidaan löytää parempia minimiarvoja. Gradienttimenetelmä on olennainen osa vastavirta-algoritmin toimintaa, mikä on tyypillinen tapa korjata neuroverkon painoarvoja. Gradienttimenetelmässä pyritään minimoimaan funktion arvoa liikkumalla negatiivisen gradientin suuntaan (Goodfellow, Bengio ja Courville 2016, 82). Vastavirta-algoritmi määrittäää arvion tehtävän muutoksen suunnasta muuttamalla neuroneiden painoarvoja (Haykin ym. 2009, 167).

Neuroverkon virhefunktio sisältää useita syötteitä, jolloin virhefunktiota voidaan ajatella

usean muuttujan funktiona. Useamman muuttujan funktioista täytyy selvittää osittaisderivaatat, jotta voidaan laskea funktion gradientti (Goodfellow, Bengio ja Courville 2016, 82).

Ulostulokerroksen neuronin j lokaali gradientti $\delta_j(n)$ saadaan kaavalla:

$$\delta_j(n) = \frac{\partial E(n)}{\partial v_j(n)} = e_j(n)\varphi'_j(v_j(n)),$$

missä lokaali gradientti määritellään virhesignaalin e_jn ja aktivaatiofunktion $\varphi_{\mathrm{j}}(\mathrm{v_j(n)})$ derivaatan tulona (Haykin ym. 2009, 161). Ulostuloneuronille voidaan siis laskea lokaali gradientti suoraan virhesignaalin avulla (Haykin ym. 2009, 164).

Piilotetulle neuronille j lokaali gradientti saadaan kaavalla:

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_k \delta_k(n) w_k j(n),$$

missä joukkoon k kuuluvat kaikki ne neuronit, joihin neuroni j on yhteydessä (Haykin ym. 2009, 162). Piilotetulla neuronilla ei ole tiettyä haluttua vastausta, jolloin virhesignaalia ei voida laskea suoraan, siksi gradientti täytyy määrittää rekursiivisesti menemällä takaperin aiempien, siihen yhdistettyjen, neuroneiden gradientteja (Haykin ym. 2009, 164).

Neuronien lokaalin gradientin avulla voidaan määrittää tarvittavat painoarvojen muutokset. Muuttamalla neuroneiden painoarvoja, voidaan optimoida verkon tuloksia. Gradienttimenetelmä ei kuitenkaan takaa virhefunktion globaalin minimiarvon löytämistä, toisin sanoen löydetty minimiarvo ei välttämättä ole paras mahdollinen. Painoarvonmuutos neuronin i ja j välillä voidaan laskea seuraavalla kaavalla (Haykin ym. 2009, 161):

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n),$$

missä η on opetusparametri, $\delta_{\mathrm{j}}(\mathrm{n})$ neuronin j lokaali gradientti ja y_i neuronin i ulostulo. Vastavirta-algoritmin opetusparametria voidaan säätää suuremmaksi, jos halutaan nopeuttaa oppimista. Sen asettaminen liian suureksi saattaa tehdä neuroverkosta epävakaan (Haykin

ym. 2009, 167). Lisäämällä algoritmiin momentti-termi $\alpha\Delta w_{ji}(n-1)$ saadaan

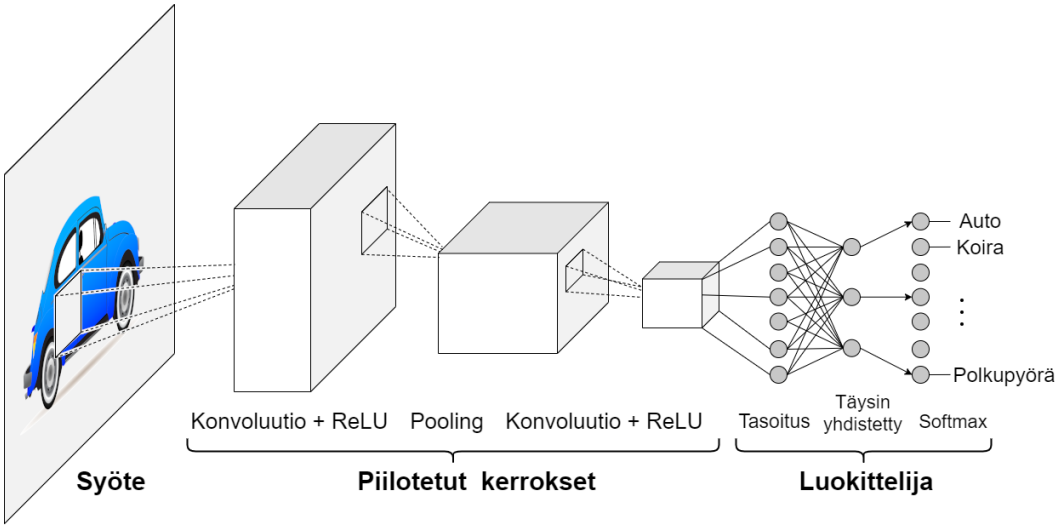
$$\Delta w_{ji}(n) = \alpha\Delta w_{ji}(n-1) + \eta\delta_j(n)y_i(n).$$

Tämän avulla voidaan nopeuttaa oppimista ilman, että verkosta tulee epävakaa (Haykin ym. 2009, 167-168). Vastavirta-algoritmi voidaan myös esittää painomatriisien avulla, toisin kuin miten Haykin ym. (2009) on esittänyt tarkastellen yksittäisiä painoja (Kärkkäinen ja Heikkola 2004).

Neuroverkon opetuksen aikana toistetaan laskenta ja vastavirta askelia kunnes ollaan saavutettu lopetusehto *(engl. stopping criterion)*. Lopetusehdoksi valitaan tyypillisesti jokin riittävän pieni keskineliösumman arvo, joka lasketaan aina jokaisen opetuskierroksen *(engl. epoch)* lopuksi (Haykin ym. 2009, 167-168). Vaikka vastavirta-algoritmi on käytetty menetelmä, sen huono puoli on se, että se vaatii suuria määriä iteraatioita lähestyäkseen haluttua tulosta (Specht 1991).

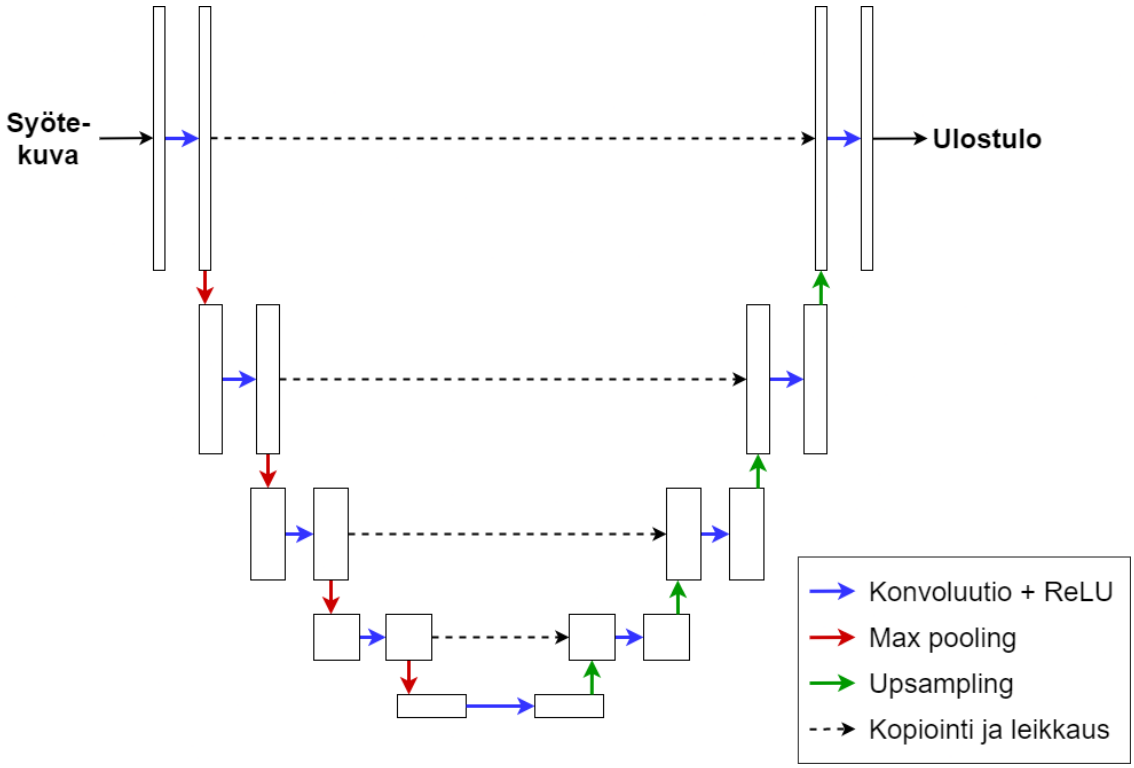
3.3 Konvoluutioneuroverkot (CNN)

Konvoluutioneuroverkko on monikerroksinen perseptroni ja se on kehitetty prosessoimaan taulukkomaista tietoa (Goodfellow, Bengio ja Courville 2016, 321). Se on siksi erityisen hyvä tunnistamaan kaksiulotteisia muotoja, joissa on suuria eroavaisuuksia niiden asennossa, koossa tai niissä on muita vääristymiä (Haykin ym. 2009, 231). Toisin sanoen neuroverkolle syötetyssä kuvassa oleva objekti voidaan venyttää tai kääntää, ja verkko pystyy silti tunnistamaan objektin. Konvoluutioneuroverkon kerroksilla suoritetaan konvoluutio-operaatioita, jotka pyrkivät vahvistamaan kuvasta löytyviä piirteitä (Goodfellow, Bengio ja Courville 2016, 322-323). Näiden kerroksien jälkeen verkko voidaan ohjata luokittelemaan lopullinen kuva. Luokitteleva konvoluutioneuroverkko sisältää vielä täysin yhdistettyjä kerroksia sekä softmax-kerroksen, joka tekee lopullisen luokittelun. Softmax-kerros muuntaa neuroverkon ulostulon todennäköisyysjakauman muotoon n:lle luokalle (Goodfellow, Bengio ja Courville 2016, 179). Konvoluutiokerroksien tehtävänä on yksinkertaistaa syötettä, jotta sen luokittelu olisi mahdollisimman helppoa.



Kuvio 7. Konvoluutioneuroverkko, CNN

Konvoluutioneuroverkon kerrokset koostuu konvoluutio-operaatiosta, ReLU:sta *(engl. Rectified Linear Unit)* ja pooling-funktiosta (Goodfellow, Bengio ja Courville 2016, 331). Kuvio 7 havainnollistaa konvoluutioneuroverkon rakennetta, missä näitä kerroksia on laitettu peräkkäin sekä ohjattu verkko tekemään luokittelua. Konvoluutioneuroverkko on mahdollista myös ohjata ohittamaan tarpeettomia kerroksia, jolloin saadaan U-kirjaimen muotoinen verkko *Unet*. Tätä on havainnollistettu kuviossa 8.



Kuvio 8. Unet-arkkitehtuuri, käytetty arkkitehtuuri konvoluutioneuroverkkojen yhteydessä (Ronneberger, Fischer ja Brox 2015)

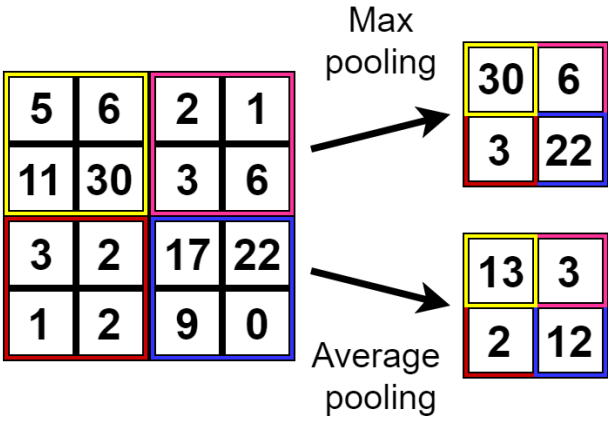
Konvoluutio

Konvoluutio-operaation tehtävä on muodostaa painotettu keskiarvo operaatio käsiteltävästä datasta (Goodfellow, Bengio ja Courville 2016, 322). Konvoluutioneuroverkkojen yhteydessä käsiteltävä data on neuroverkon syötteet. Konvoluutio-operaatio tapahtuu yhdistämällä data ja määritetty ydin *(engl. kernel)*, jota kutsutaan tyypillisesti piirrekartaksi *(engl. feature map)* (Goodfellow, Bengio ja Courville 2016, 322-323). Kuvanmuotoisen datan tapauksessa piirrekartat voidaan mieltää suodattimina *(engl. filter)*, jotka pyrkivät korostamaan olennaisia piirteitä ja pienentämään epäolennaisia piirteitä. Syvemmällä neuroverkossa olevat konvoluutiokerrokset pyrkivät korostamaan tarkempia ja monimutkaisempia piirteitä. Suodattimella voi erottaa esimerkiksi objektin reunat kuten kuviossa 9.

Kuvio 9. Suodatin, joka havaitsee reunoja.

Pooling

Pooling-kerroksilla pyritään pienentämään syötteen kokoa tiivistämällä olennaiset kohdat syötteestä. Syöte jaetaan alueisiin, joista määritellään joko keskiarvo tai suurin arvo. Tällöin saadaan max pooling ja average pooling -menetelmät. Pooling-menetelmällä korvataan neuroverkon ulostulo yhdistämällä muita lähistön ulostuloja tietyssä verkon kohdassa (Goodfellow, Bengio ja Courville 2016, 330). Kuvio 10 havainnollistaa pooling-kerroksen toimintaa missä ruudukon arvot tiivistetään joko keskiarvolla tai maksimilla.



Kuvio 10. Esimerkki max pooling ja average pooling

Rectified Linear Unit (ReLU)

ReLU on konvoluutioneuroverkoissa käytetty aktivaatiofunktio, joka on muotoa

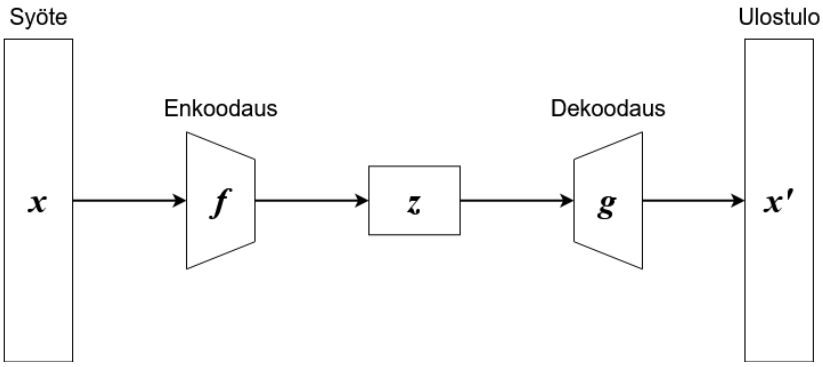
$$\varphi(x) = max\{0, x\}$$

(Goodfellow, Bengio ja Courville 2016, 187).

ReLUa käytetään, koska sen gradientit ovat suuria, mutta myös vakaita (Goodfellow, Bengio ja Courville 2016, 187). Yksi ReLU:n huono puoli on, että se ei pysty opettamaan gradienttimenetelmän avulla, silloin kun aktivointi on nollassa (Goodfellow, Bengio ja Courville 2016, 187). Siksi on kehitetty variaatioita ReLU:sta, jotta sillä saataisiin gradientti kaikkialla. Käytettyjä variaatioita ovat muun muassa Leaky ReLU, missä x kerrotaan luvulla $0,01$, kun $x \leq 0$, sekä PReLU, missä Leaky ReLU:n luku $0,01$ korvataan opeteltavalla parametrilla (Goodfellow, Bengio ja Courville 2016, 187).

3.4 Autoenkoodaaja

Autoenkoodaaja *(engl. autoencoder)* on neuroverkko, joka pyrkii kopioimaan syötteen ulostuloksi ensin enkoodaamalla syötteen, ja sen jälkeen dekoodaamalla uudelleenrakentemaan syötteen (Goodfellow, Bengio ja Courville 2016, 493). Olkoon f enkoodausfunktio ja g dekoodausfunktio, syötteelle x enkoodauksen tulos on $z = f(x)$ ja ulostulo saadaan funktiolla $g(f(x))$. Minimoitava virhefunktio voidaan määritellä yksinkertaisesti $E(x, g(f(x)))$, missä E:ksi voidaan valita esimerkiksi keskineliösumma (Goodfellow, Bengio ja Courville 2016, 494). Autoenkoodaaja voidaan luokitella ohjaamattomaksi oppimiseksi.



Kuvio 11. Yksinkertainen malli autoenkoodaajasta, missä $x' \approx x$.

4 Generative Adversarial Networks

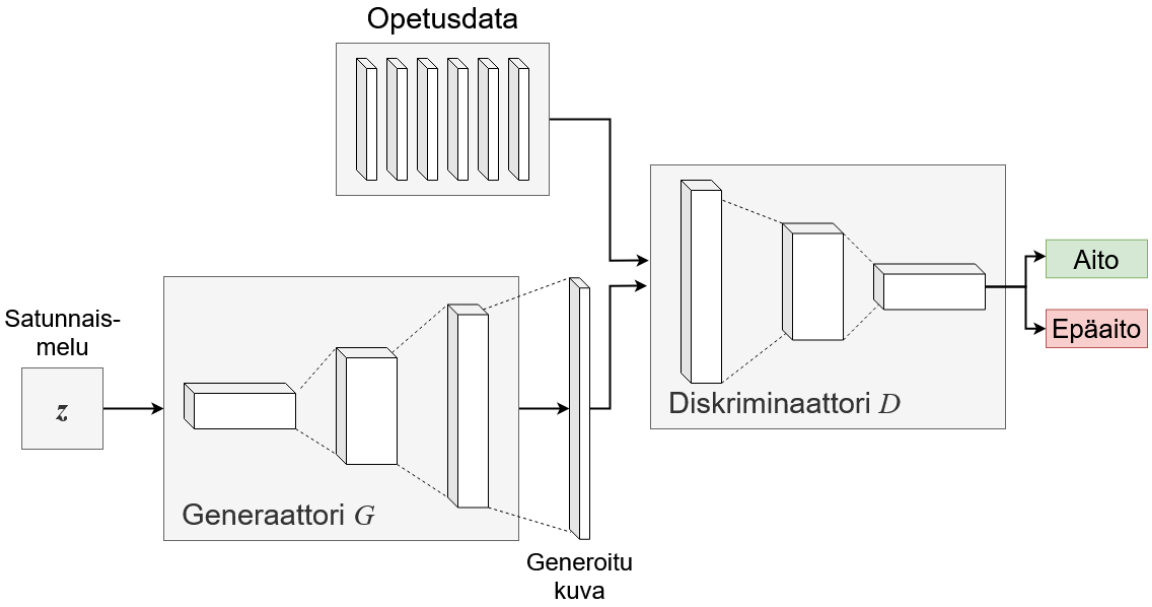
Generative Adversarial Networks -menetelmä eli GAN-menetelmä on Goodfellow ym. (2014) kehittämä menetelmä, jossa neuroverkkoja opetetaan laittamalla kaksi neuroverkkoa kilpailemaan keskenään. Näistä kahdesta verkosta käytetään nimitystä generaattori *G (engl. Generator)* ja diskriminaattori *D (engl. Discriminator)*. Nämä nimet myös kuvastavat niiden tehtävää, eli generaattorin tehtävä on muodostaa ulostulo, joka annetaan syötteeksi diskriminaattorille, jonka tehtävä puolestaan on päättää, onko annettu syöte aito, eli peräisin opetusdatasta, vai onko se epäaito, eli generaattorin luoma.

Vaikka GAN on vielä nuori menetelmä, on siitä tehty useita tutkimuksia: sovelluksia on tehty useaan eri tarkoitukseen, muun muassa videon generointiin, kasvojen generoitiin ja tunnistamiseen (Radford, Metz ja Chintala 2015a; Zhao, Weng ja Liu 2017; Vondrick, Pirsiavash ja Torralba 2016). Lisäksi GAN-menetelmällä on saatu hyviä tuloksia resoluution skaalauksessa *(engl. supersampling)*, sekä interpolaatiossa (Gopan ja Kumar 2018; Li ym. 2018). GAN-menetelmän tutkimista on helpottanut se, että koneoppiminen on jo laajasti tutkittu alue. Im ym. (2016) ovat vertailleet menetelmien suorituskykyä muun muassa MNIST-opetusdataan, jossa on käsinkirjoitettuja numeroita. Myös kuvien värillistäminen on ollut tutkimuskohteena sekä maisemakuvien ja esineiden generointi (Isola ym. 2016; Dosovitskiy, Tobias Springenberg ja Brox 2015; Radford, Metz ja Chintala 2015b; Vondrick, Pirsiavash ja Torralba 2016; Albert ym. 2018).

Goodfellow ym. (2014) mainitsevat puoliohjatun oppimisen sekä ehdollisen mallin *(engl. conditional model)* tutkimuksensa kehityskohteina. Ehdollista mallia ovat tutkineet Isola ym. (2016) kehittämällä Pix2pix-menetelmän, joka hyödyntää DCGAN-toteutusta lisäten siihen riippuvuuden lähdekuvaan. Pix2pix-menetelmään perehdytään tarkemmin luvussa 4.2.2. Muita Goodfellow ym. (2014) mainitsemia GAN-menetelmän kehityskohteita ovat muun muassa sen tehokkuuden parantaminen ja puoliohjattu oppiminen. Tehokkuutta voisi heidän mielestään parantaa, erityisesti opetuksen aikana, esimerkiksi käyttämällä eri menetelmiä koordinoimaan generaattorin ja diskriminaattorin toimintaa.

4.1 Rakenne ja toiminta

GAN-verkko koostuu kahdesta neuroverkosta: diskriminaattorista D ja generaattorista G. Generaattorin tehtävä on luoda mahdollisimman aitoja kuvia annetun syötteen ja palautteen avulla. Palautteen generaattori saa diskriminaattorilta, joka puolestaan pyrkii erottamaan generoidun kuvan aidosta kuvasta. Diskriminaattori voi myös tehdä luokittelua, jos malliin kuuluu useita vaihtoehtoja. Molemmat verkot, sekä diskriminaattori että generaattori ovat aluksi huonoja omissa tehtävissään, mutta useiden kierroksien jälkeen ne alkavat generoida ja tunnistaa systemaattisemmin. Kuviossa 12 generaattori G ottaa syötteeksi satunnaista melua z, mistä se pyrkii generoimaan kuvan, joka syötetään diskriminaattorille D, joka päättää onko kuva aito vai epäaito eli generoitu. Diskriminaattori voi vaihtoehtoisesti saada syötteeksi kuvan opetusdatasta, riippuen siitä missä vaiheessa opetusta ollaan.



Kuvio 12. Generative Adversarial Network

Käytännössä neuroverkot pelaavat minmax-peliä, jossa generaattori pyrkii maksimoimaan diskriminaattorin virheen ja diskriminaattori puolestaan pyrkii minimoimaan omat virheensä:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))],$$

missä $p_z(z)$ on generaattorin syötemelu, $D(x)$ kuvastaa todennäköisyyttä siitä tuleeko x opetusdatasta vai generaattorin jakaumasta p_g (Goodfellow ym. 2014). Menetelmän oppiminen on silloin tasapainossa kun kumpikaan ei pysty enää liikkumaan kohti tavoitetta, toisin sanoen kun generaattorin sekä diskriminaattorin virhe on 50%.

Goodfellow ym. (2014) ehdottavat, että GAN-menetelmä olisi suoraviivaisinta toteuttaa käyttäen monikerroksista perseptronia eli neuroverkkoa. Tässä tutkielmassa sovellettu Pix2pix-menetelmä käyttää monikerroksisia neuroverkkoja sekä generaattorissa, että diskriminaattorissa. Myös DCGAN-menetelmä, joka on jatkokehitystä GAN-menetelmälle, on toteutettu käyttäen monikerroksisia neuroverkkoja ja sen opetuksessa on käytetty vastavirta-algoritmia (Radford, Metz ja Chintala 2015b).

4.2 Menetelmät

Tutkimukset, jotka hyödyntävät GAN-menetelmää, ovat tehneet muutoksia alkuperäiseen Goodfellow ym. (2014) kehittämään menetelmään parantaakseen neuroverkkojen tuloksia tai soveltaakseen niitä erilaiseen opetusdataan. Käytetyissä menetelmissä on hyödynnetty muun muassa takaisinkytkettyjä neuroverkkoja (Im ym. 2016). Mielenkiintoisia tuloksia on saatu DCGAN-menetelmällä, jossa on käytetty konvoluutioneuroverkkoja generaattorissa ja diskriminaattorissa (Radford, Metz ja Chintala 2015b). Pix2pix-menetelmällä on saatu myös tarkkoja ja aidonnäköisiä kuvia käyttäen ehdollista GAN-menetelmää (Isola ym. 2016). Tutkimuksissa on saatu vaihtelevia tuloksia ja usein ongelmana on generoitujen kuvien kuvanlaatu, vaikkakin osassa tuloksista on myös tarkkoja ja aidonnäköisiä kuvia. Tässä luvussa käsiteltäviksi menetelmiksi on valittu vain ne menetelmät, jotka liittyvät olennaisesti tutkielmassa sovellettuun Pix2pix-menetelmään. Itse Pix2pix-menetelmä käsitellään aliluvussa 4.2.2.

4.2.1 DCGAN

Deep Convolutional Generative Adversarial Networks (DCGAN) on Radford, Metz ja Chintala (2015b) kehittämä menetelmä, jossa on yhdistetty konvoluutiokerrokset, syväoppiminen sekä GAN-rakenne. DCGAN-menetelmää voidaan käyttää sekä kuvien generointiin,

että niiden luokitteluun. DCGAN-menetelmällä voidaan generoida kuvia, joissa on yhdistettu useamman kuvan piirteitä, esimerkiksi kasvoihin voidaan yhdistää aurinkolasit (Radford, Metz ja Chintala 2015b). He ovat käyttäneet generaattorin kaikilla kerroksilla ReLU-aktivaatiofunktiota, lukuun ottamatta ulostulokerrosta, joka käyttää Tanh-aktivaatiofunktiota. Lisäksi verkon piilotetut kerrokset eivät ole täysin yhdistettyjä.

4.2.2 Pix2pix

Pix2pix-menetelmä on ehdollinen GAN eli cGAN *(engl. conditional generative adversarial network)*, jonka tarkoituksena on oppia kartoittamaan lähdekuva eli syöte kohdekuvaan eli tulosteeseen. Esimerkiksi kesäisen maisemakuvan muuttaminen talviseksi tai väritön kuva värilliseksi. Pix2pix-menetelmä (Isola ym. 2016) pohjautuu vahvasti DCGAN-menetelmään (Radford, Metz ja Chintala 2015b), mutta lisää menetelmään mahdollisuuden opettaa verkon generaattori riippuvaiseksi lähdekuvasta. Menetelmän vahvuus tulee juuri siitä, että generoinnissa ei ole käytetty pelkästään satunnaismelua vaan se on myös riippuvainen lähdekuvasta.

Menetelmän generaattori on autoenkoodaaja, jolloin siinä on enkoodaus-kerroksia ja dekoodauskerroksia, missä kerroksilla on käytetty konvoluutiota ja de-konvoluutiota. Generaattorin ja diskriminaattorin arkkitehtuurit ovat mukailtu DCGAN-toteutuksen pohjalta, missä jokainen enkoodaus-kerros on muotoa konvoluutio-BatchNorm-ReLu. Generaattorin arkkitehtuuri on mallinnettu U-verkon mukaan. U-verkon hyödyt eivät näy pelkästään GAN-menetelmän yhteydessä, mutta kun verkko opetetaan L1-virheen kanssa, sen suorituskyky on parempi kuin encoder-decoder verkolla (Isola ym. 2016).

Pix2pix-menetelmässä GAN-verkkojen opetus tapahtuu kahdessa osassa: diskriminaattorin opetus ja generaattorin opetus (Hesse 2017). Verkkojen opetus on tehty puoliohjatulla oppimisella, missä diskriminaattori vertaa syötekuvaa ja kohdekuvaa, jonka jälkeen se vertaa syötekuvaa ja generoitua kuvaa tietämättään kumpaa paria verkko vertailee. Tämän jälkeen diskriminaattori tekee arvauksen kuinka realistiselta kuvat näyttävät, jonka jälkeen lasketaan virhe ja korjataan verkon painoarvoja. Generaattorin opetus tapahtuu myös puoliohjatulla oppimisella, missä generoitua kuvaa verrataan kohdekuvaan, jonka jälkeen saadaan myös tulos

diskriminaattorilta. Näiden yhdistelmästä lasketaan virhe, ja korjataan verkon painoarvoja.

Analyysissä on käytetty FCN-pistetytystä, jossa mitataan miten klassisesti ohjatulla oppimisella opetetut verkot pystyvät tunnistamaan generoidut kuvat (Isola ym. 2016). Intuitiivisesti ajatellen, jos kuva on aidon näköinen niin nämä klassiset verkot pystyvät tunnistamaan myös generoidut kuvat.

4.2.3 CycleGAN

CycleGAN on Zhu ym. (2017) kehittämä menetelmä, joka on jatkoa Pix2pix-menetelmälle. Menetelmä mahdollistaa kuvamuunnoksien muodostamisen ilman, että kuvat ovat paritettuja toisin kuin Pix2pix-menetelmässä. Opetukseen on lisätty myös yksi lisäaskel, jossa muunnos tehdään myös takaperin, jolloin syötekuva pyritään muodostamaan uudelleen käyttäen yhdenpitävyyksiä: $x \to G(x) \to F(G(x)) \approx x$, missä x syötekuvajoukon esimerkki, ja $y \to F(y) \to G(F(y)) \approx y$, missä y on on kohdejoukon esimerkki (Zhu ym. 2017). Menetelmässä hyödynnetään ohjattua oppimista kokonaisten joukkojen yhteydessä toisin kuin Pix2pix-menetelmässä, missä jokaiselle syötteelle määritellään myös kohdekuva. Tällöin CycleGAN hyödyntää yksittäisten kuvien kohdalla valvomatonta oppimista, jolloin verkko oppii muuttamaan syötekuvan kohdekuvajoukon kuvaksi, esimerkiksi kesäinen maisemakuva talviseksi kuvaksi. Zhu ym. (2017) toteavat, että menetelmän toimii hyvin värien tai tekstuurien muutoksissa, mutta esimerkiksi rakenteelliset muutokset kuvissa tuottavat vaihtelevia tuloksia. Lisäksi he toteavat, että CycleGAN-menetelmällä saadut tulokset eivät ole kaikilla opetusdatoilla parempia kuin Pix2pix-menetelmä, erityisesti silloin kun valvottu oppiminen on selkeästi parempi valinta.

4.3 Tulosten analyysi ja jatkokehitys

Neuroverkkojen tekemien päätöksien analyysi on olennainen osa niiden kehittämistä ja tehokkuuden arvioimista. Koska neuroverkkojen opetuksesta kerääntyy paljon dataa, voidaan analyysissä soveltaa tilastollisia menetelmiä, joilla voidaan helposti mitata esimerkiksi verkon tekemää virhettä. On myös olemassa vahvistavan oppimisen malleja, joissa verkkoa pyritään palkitsemaan hyvistä päätöksistä virheen minimoinnin sijasta. Virhettä onnistuneesti

minimoimalla päästään hyviin tuloksiin, mutta on huomattu, että neuroverkkojen optimointi on vaikea prosessi (Zhu ym. 2017; Vondrick, Pirsiavash ja Torralba 2016).

Sovelluksen toimivuutta voidaan mitata tekemällä tilastollista analyysiä, mutta se ei pysty mittaamaan GAN-menetelmä tärkeintä ominaisuutta: mahdollisimman aidon ulostulon generoiminen. GAN pyrkii luomaan mahdollisimman aidonnäköistä dataa ja jäljittelemään kuvien yhteyksiä. Vastaavat tutkimukset ovat osoittaneet, että GAN-menetelmän toimivuuden mittaaminen on hankalaa (Albert ym. 2018; Radford, Metz ja Chintala 2015b). Useissa tapauksissa on kyetty muodostamaan dataa, joka vaikuttaa abstraktilta ja satunnaiselta, mutta kuitenkin muistuttaa hämmästyttävän paljon aitoa dataa (Dosovitskiy, Tobias Springenberg ja Brox 2015; Radford, Metz ja Chintala 2015b). GAN pystyy mallintamaan muun muassa maiseman dynamiikkaa, liikettä sekä kontekseja, mutta sen tarkkuus on vaihtelevaa sovelluskohteesta riippuen (Radford, Metz ja Chintala 2015b; Vondrick, Pirsiavash ja Torralba 2016). Muilla menetelmillä pystytään saavuttamaan tilastollisesti tarkkoja tuloksia, mutta ne eivät välttämättä pysty jäljittelemään kuvien dynamiikkaa eikä kontekstia. Ihmisellä on luontainen abstrakti päättelykyky, jolloin ihmisen on helppo tunnistaa, onko videossa liikkuva eläin tai ihminen oikea vai tietokoneella generoitu. Siksi GAN-menetelmän tulosten mittauksessa on käytetty ihmisten mielipidettä ja on huomattu, että ihminen huomaa tarkasti virheet generoiduista kuvista (Isola ym. 2016).

Albert ym. (2018) soveltavat GAN-menetelmää tutkiessaan urbaanien ympäristöjen muotoja ja kuvioita satelliittikuvien avulla. He ovat käyttäneet analyysissään *average radial profile*-kaavaa, jolla kuvataan nimenomaan urbaanien asuinympäristöjen ja kaupunkien jakaumia. He ovat soveltaneet GAN-menetelmän tuottamia kuvia kaavaan ja toteavat, että kuvat mallintavat todellisuutta hyvin. Analyysissä ei puututa yksittäisten pikselien tarkkuuteen vaan kokonaiskuvaan. Lisäksi he toteavat, että on vaikeaa mitata GAN-menetelmällä luotujen kuvien realistisuutta.

Vaikka GAN-menetelmästä on ehditty tekemään useita tutkimuksia, siinä on vielä kehitettävää. Radford, Metz ja Chintala (2015b) toteavat DCGAN-toteutuksen olevan vielä epävakaa ja vaatii jatkotutkimusta. Myös (Zhu ym. 2017) toteavat, että vaikka tulokset ovat positiivisia ja lupaavia, ne eivät ole virheettömiä ja edelleen alttiita epävakauksille. He ehdottavat, että puoliohjatun oppimisen soveltaminen voisi johtaa parempiin tuloksiin.

5 Soluautomaatti

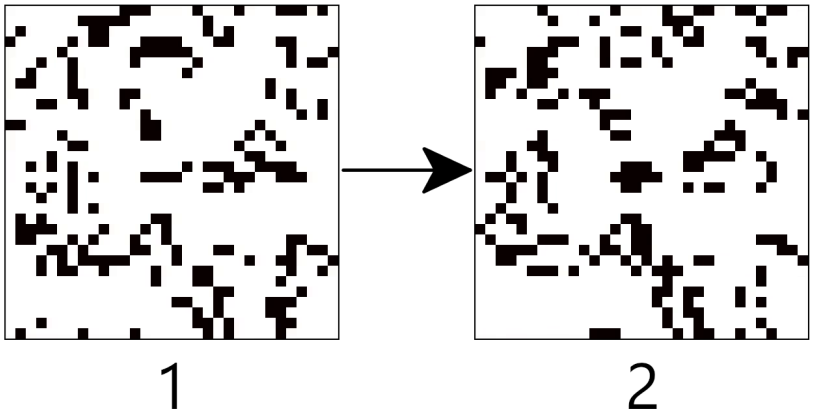
Conway's Game of Life on John Horton Conwayn vuonna 1970 kehittämä soluautomaatti *(engl. cellular automaton)*, jossa pelikentällä olevat solut lisääntyvät ja kuolevat pelin sääntöjen mukaisesti. Pelissä ei ole pelaajaa vaan peli pelaa peliä itse eikä se vaadi muuta kuin lähtötilan, jonka jälkeen peli muodostaa seuraavia sukupolvia sääntöjen mukaisesti. Pelissä on useita mielenkiintoisia ominaisuuksia, esimerkiksi pelissä voi ilmaantua *Glider*-kuvio, joka on eteenpäin liikkuva entiteetti. Lisäksi pelin sisään on kyetty rakentamaan toimiva Turingin-kone, joka kykenee luomaan säännönmukaisia *koneita* sekä muistisoluja (Rendell 2000). Pelin ominaisuuksia on myös analysoitu konvoluutioneuroverkkojen avulla (Gilpin 2019). Yleisesti ottaen soluautomaattia voi käyttää kuvaamaan säännöllistä järjestelmää, esimerkiksi liikennettä (Hämäläinen 2006).

5.1 Game of life -säännöt

Tutkimuksessa käytetty soluautomaatti on Conwayn Game of Life -peli. Pelissä on ruudukko, jossa jokainen solu voi olla joko elossa (1) tai kuollut (0). Visuaalisesti tyypillisesti merkataan elossa olevaa solua tummalla värillä ja kuollutta vaalealla värillä. Jokaisessa iteriaatiossa ruudukon soluille lasketaan uusi tila. Soluautomaatin yhteydessä iteraatiolla ja sukupolvella tarkoitetaan samaa asiaa. Jokaisessa iteraatiossa lasketaan kaikille pelikentän soluille uusi arvo. Solun syntyminen, elossa pysyminen tai kuolema määritellään seuraavien sääntöjen mukaan:

1. Jokainen elossa oleva solu, jolla on joko 2 tai 3 naapuria, pysyy elossa
2. Jokaisesta kuolleesta solusta, jolla on tasan 3 elossa olevaa naapuria, tulee elävä solu
3. Jokainen elävä solu, jolla on yli 3 naapuria, kuolee
4. Jokainen elävä solu, jolla on vain 1 naapuri, kuolee

Pelikentän reunoilla viereiset solut siirretään pelikentän toiselle puolelle, jotta peli pysyy määriteltyjen rajojen sisällä. Lisäksi pelikentän kooksi on valittu 32x32 solua. Kuvio 13 kuvastaa soluautomaatin yhtä iteraatiota, jossa sukupolvesta 1 siirrytään sukupolveen 2.



Kuvio 13. Soluautomaatin iteraatio

6 Sovellus ja tutkimus

Tutkimuksessa pyritään selvittämään GAN-menetelmän suorituskykyä uudenlaisessa ympäristössä. Halutaan selvittää, pystyykö GAN-menetelmällä opetetut verkot generoimaan soluautomaattiin uusia iteraatioita ja kuinka hyvin. Tällä asetelmalla pyritään muodostamaan "piilotettu"säännöstö, jota neuroverkot pyrkivät generoimaan ja tunnistamaan. Neuroverkot eivät tiedä soluautomaatin säännöistä, muuten kuin annetusta kuvadatasta. Tutkimuksessa käytetyn opetusdatan sisältö ja asettelu on yksinkertaisempi kuin esimerkiksi kuvissa luonnosta tai eläimistä: soluautomaatin kuvat on yksivärisiä ja yksinkertaisia suorakulmioita.

Motiivi tutkimuksella on tuottaa uutta dataa GAN-menetelmästä. GAN-menetelmää on sovellettu jo useissa tutkimuksissa, mutta se on vielä suhteellisen uusi menetelmä, joten sen soveltuvuutta erilaisiin ongelmiin ei ole vielä täysin kartoitettu.

Pix2pix-menetelmä osoittaa, että GAN-menetelmän avulla on mahdollista tehdä kartoitusta yhdestä joukosta toiseen useilla eri opetusdatoilla (Isola ym. 2016). Jotta tämä olisi mahdollista, verkon täytyy ymmärtää kuvien välistä yhteyttä. Koska neuroverkot ovat musta laatikko, niiden ymmärtämisen tasosta on vaikeaa tehdä selkeitä johtopäätöksiä, esimerkiksi kykeneekö neuroverkko ymmärtämään kuvien taustalla olevia sääntöjä, tämän tutkielman tapauksessa soluautomaatin sääntöjä.

Tässä tutkimuksessa testataan kuinka hyvin Pix2pix-menetelmällä opetettu verkko osaa pelata Game of Life -peliä. Pix2pix-menetelmällä opetetaan joukko verkkoja, jotka tekevät kuvamuunnoksia pelistä ottamalla sisääntulona iteraation pelistä ja generoimalla seuraavan iteraation. Diskriminaattorin tehtävä on tunnistaa, onko tämä muunnos oikeanlainen muunnos. Verkkojen opetusten jälkeen ajetaan testejä, joissa generoidaan iteraatioita pelistä, minkä jälkeen analysoidaan tulokset. Pyritään selvittämään, onko eri tavoilla opetettujen verkkojen välillä eroja niiden suorituskyvyssä.

6.1 Menetelmä ja sen valinta

Tutkimuksessa haluttiin tutkia GAN-menetelmän soveltuvuutta soluautomaattiin. Tutkimukseen on valittu käytettäväksi menetelmäksi Pix2pix-menetelmä, joka pohjautuu DCGAN-menetelmään, jolloin se soveltuu hyvin hahmontunnistus- ja kuvantunnistustehtävään. Lisäksi Pix2pix-menetelmä hyödyntää syötekuvaa uuden kuvan generoinnissa, mikä vahvistaa luodun kuvan ja syötekuvan välistä yhteyttä.

Pix2pix-menetelmää on jatkokehitetty CycleGAN-menetelmäksi (Zhu ym. 2017), mutta se ei kuitenkaan sovellu tämän tutkimuksen ongelmaan, sillä tutkimusastelman vuoksi on olennaista, että opettaessa neuroverkkoa se oppii yhdistämään syötekuvan seuraavaan pelin iteraatioon. Jos kuvat eivät ole paritettuja niin verkko oppii luomaan vain satunnaisia iteraatioita pelistä. Toisenlaisessa tutkimusasetelmassa se voisi toimia, mutta tässä tutkielmassa on valittu tarkasteltavaksi kuinka hyvin verkko oppii pelaamaan peliä sääntöjen mukaisesti.

6.2 Tutkimusjärjestelyt

Tutkimuksen järjestelyt koostui useista vaiheista. Ensimmäisenä tutustuttiin käytettävän menetelmän lähdekoodiin ja asetuksiin. Tämän jälkeen tehtiin alustavia kokeiluja sekä valmiilla opetus- ja testidatalla, joka oli saatavilla menetelmän kehittäjien toimesta, että itse luodulla materiaalilla. Alustavien kokeilujen jälkeen perehdyttiin opetusdatan muodostamiseen.

Pix2pix-menetelmän lähdekoodi on toteutettu Python-ohjelmointikielellä käyttäen PyTorch-kirjastoa ja kehittäjät tarjoavat lähdemateriaalissa muun muassa skriptejä, joilla voi muodostaa oman opetusdatan, muuttaa opetusparametreja, käynnistää verkkojen opetus ja testata verkkoja. Tutkimuksen opetusdata luotiin oman python-lähdekoodin avulla käyttäen numpy ja matplotlib -kirjastoja. Ubuntu LTS 18.04 käyttöjärjestelmän pakettihallinnasta löytyvää ffmpeg-työkalua käyttäen opetusdata esikäsiteltiin leikkaamalla ja skaalaamalla. Esikäsittelyn jälkeen opetusdata oli valmis muutettavaksi verkoille sopivaan muotoon. Lähdemateriaalin skripti parittaa opetusdatan lähdekuvan ja kohdekuvan, jonka jälkeen opetusdata on valmis verkkojen opetusta varten. Verkkojen opetusta varten täytyi käynnistää lokaali visdom-palvelin, johon tallentuu opetuksen lokitiedot ja testiesimerkit.

Verkkojen opetuksen jälkeen verkoilla suoritettiin useita testejä, joita käsitellään tarkemmin luvussa 6.4. Suurin osa testeistä ajettiin käyttäen lähdemateriaalin skriptejä, mutta testitapaus, jossa yksi opetetuista verkoista generoi soluautomaattia itsenäisesti, vaati oman skriptin kehittämisen. Tässä skriptissä verkon ulostulo syötetään, silmukassa iteroiden, uudestaan verkolle kunnes saadaan riittävä määrä havaintoja tutkittavaksi eli noin 100 generoitua sukupolvea.

6.3 Verkkojen opetus

Tutkimuksessa opetettiin samalla menetelmällä useampi verkko käyttäen erikokoisia datajoukkoja, eri resoluutiota sekä eri opetuskierrosten *(engl. epoch)* määrää. Verkot ovat listattuna taulukossa 1. Verkkojen opetus on suoritettu Intel i7 9700k (@4.7GHz) prosessorilla ja 16Gb DDR4-muistilla.

Verkko	Opetuskierrokset	Opetusdatan koko	Resoluutio	Opetuksen kesto
$V256_{2500e10}$	10 kierrosta	2500 kuvaa	256x256	365 min
$V256_{100e50}$	50 kierrosta	100 kuvaa	256x256	79 min
$V512_{100e50}$	50 kierrosta	100 kuvaa	512x512	120 min
$V512_{100e200}$	200 kierrosta	100 kuvaa	512x512	472 min

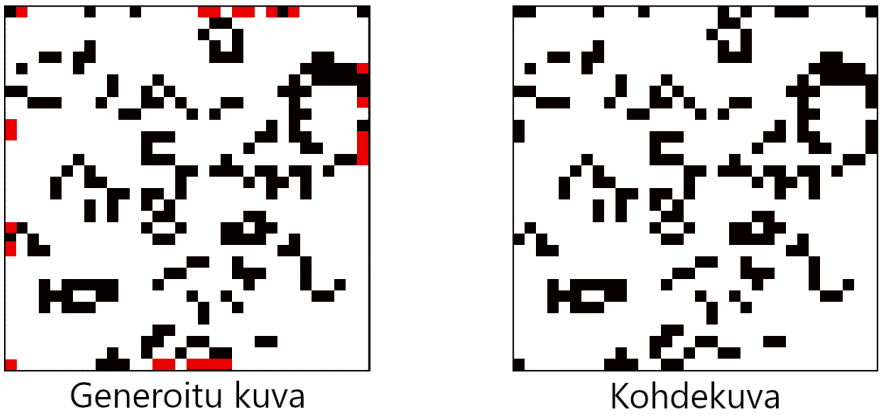
Taulukko 1. Opetetut verkot.

Verkkojen $V256_{100e50}$, $V512_{100e50}$ ja $V512_{100e200}$ opetuksessa käytettiin samaa kuvadataa, jotta saatujen testien tuloksista voidaan vertailla vain resoluution ja opetuskierrosten vaikutusta suorituskykyyn. Verkkojen opetuksessa on käytetty Isola ym. (2016) määrittämiä vakioasetuksia, mutta käytöstä poistettiin esikäsittely *(engl. preprocess)*, jottei kuvissa tapahtuisi leikkausta tai skaalausta, sillä opetuksessa käytettiin useita eri resoluutioita.

6.4 Verkkojen testaus

Verkkojen suorituskykyä testattiin laittamalla verkot generoimaan soluautomaatin sukupolvia samoista syötekuvista ja vertailemalla verkkojen generoimien kuvien laatua. Kuviossa 14 on esimerkki verkon $V512_{100e200}$ generoimasta kuvasta, missä kuvioon on merkattu pu-

naisella värillä puuttuvat solut sekä ylimääräiset solut. Verkolle on annettu syötteenä yksi testidatan kuva, josta verkko on generoinut seuraavan sukupolven. Kuviosta voi huomata, että kuvan reunoilla tapahtuu enemmän virheitä kuin muualla kuvassa. Kuvion generointiin käytetty verkko $V512_{100e200}$ saavutti parhaimmat tulokset kaikista opetetuista verkoista.



Kuvio 14. Verkon $V512_{100e200}$ generoima sukupolvi ja kohdekuva, mihin virheet on merkattu käsin

Opetetut verkot testattiin samalla testidatalla, jossa on 150 kuvaa, joista yksikään ei ole ollut mukana opetusdatassa. Yksittäisen testikuvan virhe lasketaan generoidun kuvan ja kohdekuvan jokaisen pikselin erotuksen summana:

$$E_g = \frac{1}{N^2} \sum_{x=1}^{N} \sum_{y=1}^{N} |G(x,y) - T(x,y)|,$$

missä N^2 on kuvan resoluutio eli pikseleiden kokonaismäärä, $G(x,y)$ generoidun kuvan arvo kohdassa (x,y) ja $T(x,y)$ kohdekuvan arvo kohdassa (x,y). Jokaisen pikselin arvo lasketaan pikselin RGB-värikanavien keskiarvona. Yhden kuvan kokonaisvirhe ilmaistaan prosenttilukuna, joka kertoo kuinka paljon kuvassa on eroa verrattuna kohdekuvaan. Koska verkko on oppinut hyvin luomaan oikean sävyn jokaiselle solulle, pikselin sävystä tuleva virhe on erittäin pieni.

Jokaiselle verkolle laskettiin virheiden keskiarvo testidatalle kaavalla

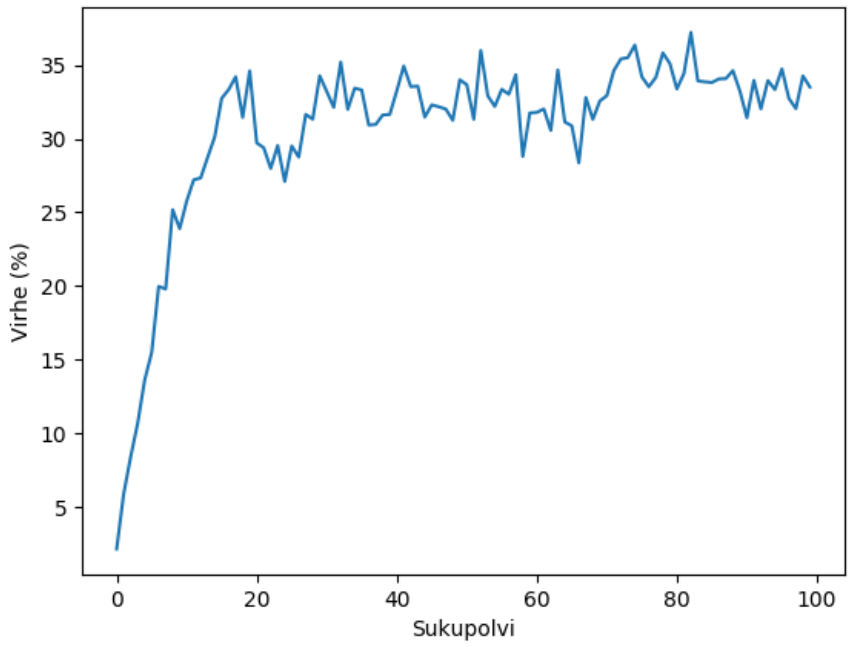
$$E_{av} = \frac{1}{N}\sum_{N} E_g,$$

missä N on testidatan määrä eli 150. Nämä tulokset on esitetty taulukossa 2, missä pienempi virhe tarkoittaa tarkempaa tulosta.

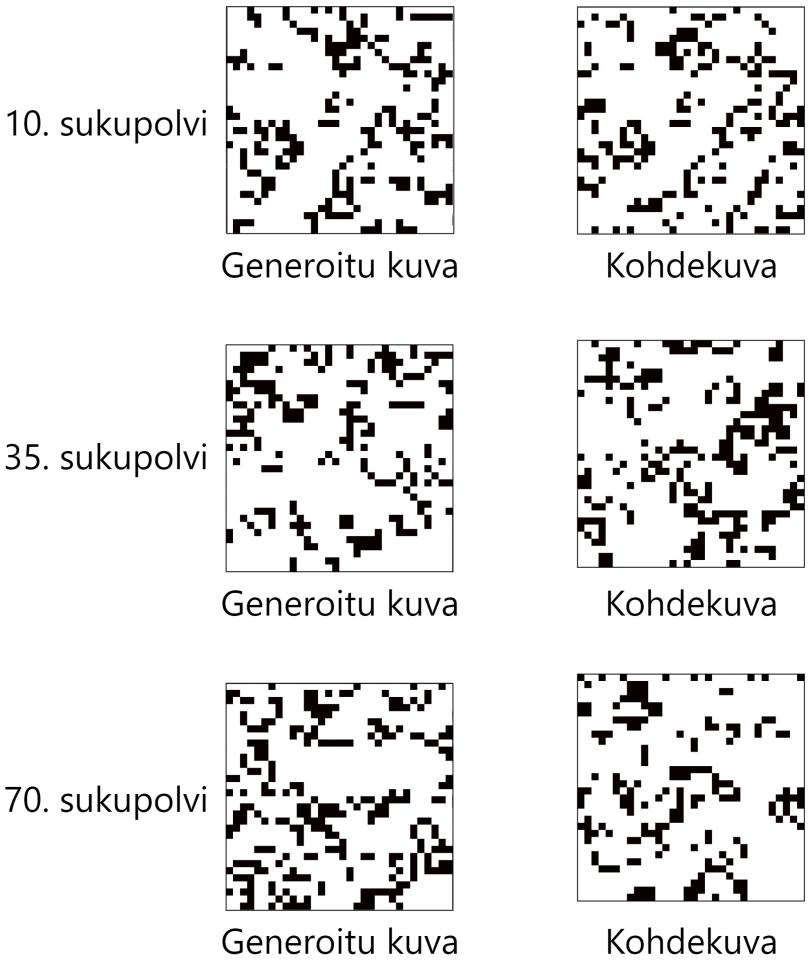
Verkko	Virhe
$V256_{2500e10}$	5.78 %
$V256_{100e50}$	3.24 %
$V512_{100e50}$	2.85 %
$V512_{100e200}$	2.49 %

Taulukko 2. Verkkojen keskivirhe samalla testidatalla.

Verkolla $512x512_{100e200}$ ajettiin myös testi, jossa verkolle annettiin vain lähtösukupolvi, minkä jälkeen verkko generoi itse loput sukupolvet, toisin sanoen verkko pelasi peliä itsenäisesti. Kuvio 15 kuvastaa verkon tekemää virhettä pelatessa peliä itse. Kuviossa 16 on esitetty havaintoja generoiduista sukupolvista ja niiden eroista kohdekuviin pelatessa itsenäisesti.



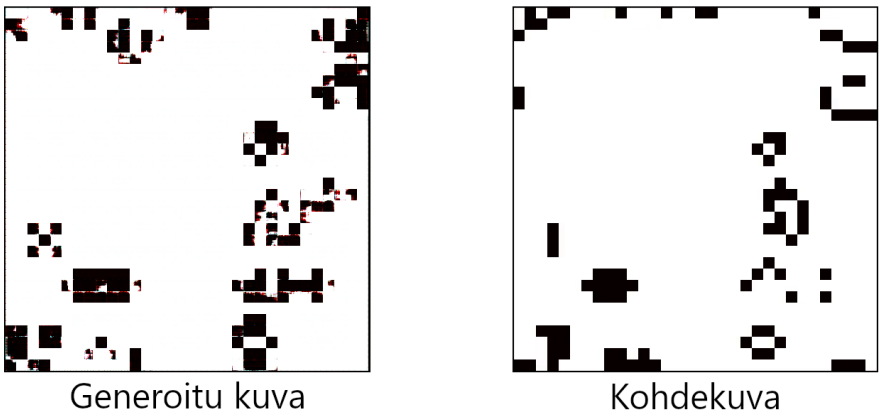
Kuvio 15. Verkon $V512_{100e200}$ virhe pelatessa peliä itsenäisesti



Kuvio 16. Verkon $V512_{100e200}$ generoimat sukupolvet 10, 35 ja 70 pelatessa peliä itsenäisesti

6.5 Tulosten analyysi ja tulkinta

Vaikka verkko $V512_{100e200}$ on tarkka, sen suorituskyvyssä on aukkoja, erityisesti niiden kuvien kohdalla joissa oli vain vähän soluja elossa, verkko teki enemmän virheitä, kuten kuviosta 17 nähdään. Verkon opetuksessa data ei ollut riittävän monipuolista, joten verkko ei kykene generoimaan tarkkoja tuloksia silloin kun pelikentällä on liian vähän soluja. Lisäksi testeistä huomataan, että pelikentän reunoilla tapahtuu suurin osa virheistä, joten voidaan päätellä, että verkolla on vaikeuksia ymmärtää kuinka reunan yli muodostuvat solut siirtyvät kentän toiselle puolelle. Näitä ongelmia voisi parantaa yksinkertaisesti lisäämällä laadukkaampia ja monipuolisempia esimerkkejä opetusdataan, mutta on mahdollista, että reunojen yli muodostuvien solujen havaitseminen on liian vaikea ongelma.



Kuvio 17. Verkon $V512_{100e200}$ generoima sukupolvi ja kohdekuva, virhe 11,19%

Verkon $V512_{100e200}$ pelatessa peliä itsenäisesti sen generoimat kuvasarjat pysyvät alussa, noin 1-25 sukupolvea, tarkkoina, mutta useiden iteraatioiden, noin 30-50 sukupolvea, jälkeen sukupolvet alkavat poiketa toisistaan, kunnes lopulta sadan iteraation jälkeen sukupolven näyttävät täysin erilaisilta. Jokainen verkon tekemä virhe muuttaa pelin suuntaa aina hieman, vaikka verkko pelaa peliä lähes oikein kasautuvat virheet muuttavat pelin suuntaa runsaasti. Jotta verkko pystyisi pelaamaan peliä tarkasti täytyisi jokaisen generoidun kuvan virhe olla erittäin pieni.

Lisäksi tuloksista huomataan, että resoluution ja opetuskierrosten kasvattamisella saadaan parannettua tuloksia huomattavasti. Verraten muihin verkkoihin, suurella datamäärällä opetus kestää huomattavasti pidempään, eikä tulokset ole riittävän hyviä vähäisillä opetuskierroksilla.

7 Johtopäätökset

GAN-menetelmä on ajankohtainen tutkimusaihe ja sillä on saatu mielenkiintoisia tuloksia kuvien generoinnissa. Tämän tutkimuksen kohteena oli Pix2pix-menetelmä ja sen suorituskyky iteroidessa Conwayn Game of Life -soluautomaatin sukupolvia. Tutkimuksesta selvisi, että Pix2pix-menetelmällä on mahdollista opettaa generatiivinen neuroverkko, joka pystyy generoimaan Game of Life -sukupolvia virheellä 2,49%. Verkkojen opetuksessa käytettiin Isola ym. (2016) määrittämiä vakioasetuksia ja verkko oli avoimesti saatavilla GitHub -versionhallinnasta.

GAN-menetelmällä saatujen tuloksien tulkinta on vielä avoin ongelma eli ei olla löydetty selkeästi parasta menetelmää analysoimaan generoitujen kuvien aitoutta, mutta aiemmissa tutkimuksissa kuten (Isola ym. 2016) voidaan käyttää rakenteellista analyysia, mutta se soveltuu parhaiten kuviin, joissa on selkeitä objekteja, kuten eläimiä tai rakennuksia. Tässä tutkielmassa laskettiin generoidun kuvan ja kohdekuvan pikseleiden erot, jolloin virheestä saadaan suoraan puuttuvien solujen tai väärin aseteltujen solujen määrä mitattua. Verkon itse suorittamasta ajosta huomataan, että vaikka se ei pelaa peliä täydellisesti, se pystyy generoimaan seuraavia sukupolvia itse generoimistaan kuvista, ilman että ne muuttuvat erityisen huonolaatuisiksi. Verkon generoidessa sellaista sukupolvea, jossa on vain vähän soluja, tulokset olivat epätarkkoja. Epätarkkuus on esiintynyt myös alkuperäisessä Pix2pix-tutkimuksessa ja on yleisesti esiintyvä ongelma kuvien generoinnissa (Pathak ym. 2016; Dosovitskiy, Tobias Springenberg ja Brox 2015; Im ym. 2016; Isola ym. 2016).

Parannettavaksi jäi verkon heikko suorituskyky pelikentän reunoilla ja silloin kun kentällä oli vain vähän soluja. Jatkotutkimuksena olisi mielenkiintoista testata kuinka pelikentän laajentaminen vaikuttaa suorituskykyyn. Erityisesti mielenkiintoista tässä tutkimuksessa oli kuinka hyvin verkko oppi ymmärtämään soluautomaatin sääntöjä pelkkien kuvien pohjalta. Pix2pix-verkkoa voisi myös soveltaa muihin peleihin tai järjestelmiin, joista pystytään tallentamaan pelin tai järjestelmän tila kuvana ja vertaamaan sitä seuraavaan tilaan. Olisi mielenkiintoista selvittää kuinka tarkkaan monimutkaisia järjestelmiä GAN-menetelmällä voisi mallintaa.

Lähteet

Albert, A., E. Strano, J. Kaur ja M. González. 2018. “Modeling Urbanization Patterns with Generative Adversarial Networks”. Teoksessa *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium,* 2095–2098. Heinäkuu. doi:10.1109/IGARSS.2018.8518032.

Alpaydin, Ethem. 2010. *Introduction to Machine Learning.* 2nd. The MIT Press. ISBN: 026201243X, 9780262012430.

Dosovitskiy, Alexey, Jost Tobias Springenberg ja Thomas Brox. 2015. “Learning to Generate Chairs With Convolutional Neural Networks”. Teoksessa *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Kesäkuu.

Gilpin, William. 2019. “Cellular automata as convolutional neural networks”. *Physical Review E* 100, numero 3 (syyskuu). ISSN: 2470-0053. doi:10.1103/physreve.100.032402. http://dx.doi.org/10.1103/PhysRevE.100.032402.

Goodfellow, Ian J. 2017. “NIPS 2016 Tutorial: Generative Adversarial Networks”. *CoRR* abs/1701.00160. arXiv: 1701.00160. http://arxiv.org/abs/1701.00160.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville ja Yoshua Bengio. 2014. *Generative Adversarial Networks.* arXiv: 1406.2661 [stat.ML].

Goodfellow, Ian, Yoshua Bengio ja Aaron Courville. 2016. *Deep Learning.* The MIT Press. ISBN: 0262035618, 9780262035613.

Gopan, K., ja G. S. Kumar. 2018. “Video Super Resolution with Generative Adversarial Network”. Teoksessa *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI),* 1489–1493. Toukokuu. doi:10.1109/ICOEI.2018.8553719.

Haykin, Simon S, ym. 2009. *Neural networks and learning machines/Simon Haykin.* New York: Prentice Hall,

Hesse, Christopher. 2017. “Image-to-Image Translation in Tensorflow”. https://affinelayer.com/pix2pix/.

Hubel, D., ja T. Wiesel. 1962. “Receptive fields, binocular interaction, and functional architecture in the cat’s visual cortex”. *Journal of Physiology* 160:106–154.

Hämäläinen, Arto. 2006. “Studies of traffic situations using cellular automata” [kielellä en]. G4 Monografiaväitöskirja. http://urn.fi/urn:nbn:fi:tkk-008287.

Im, Daniel Jiwoong, Chris Dongjoo Kim, Hui Jiang ja Roland Memisevic. 2016. *Generating images with recurrent adversarial networks.* arXiv: 1602.05110 [cs.LG].

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou ja Alexei A. Efros. 2016. “Image-to-Image Translation with Conditional Adversarial Networks”. *CoRR* abs/1611.07004. arXiv: 1611.07004. http://arxiv.org/abs/1611.07004.

Kargaard, J., T. Drange, A. Kor, H. Twafik ja E. Butterfield. 2018. “Defending IT systems against intelligent malware”. Teoksessa *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT),* 411–417. Toukokuu. doi:10.1109/DESSERT.2018.8409169.

Kärkkäinen, Tommi, ja Erkki Heikkola. 2004. “Robust formulations for training multilayer perceptrons”. *Neural Computation* 16 (4): 837–862.

Li, C., D. Gu, X. Ma, K. Yang, S. Liu ja F. Jiang. 2018. “Video Frame Interpolation Based on Multi-scale Convolutional Network and Adversarial Training”. Teoksessa *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC),* 553–560. Kesäkuu. doi:10.1109/DSC.2018.00089.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra ja Martin Riedmiller. 2013. *Playing Atari with Deep Reinforcement Learning.* arXiv: 1312.5602 [cs.LG].

Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell ja Alexei A. Efros. 2016. *Context Encoders: Feature Learning by Inpainting.* arXiv: 1604.07379 [cs.CV].

Radford, Alec, Luke Metz ja Soumith Chintala. 2015a. “Unsupervised representation learning with deep convolutional generative adversarial networks”. *arXiv preprint arXiv:1511.06434.*

Radford, Alec, Luke Metz ja Soumith Chintala. 2015b. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* Cite arxiv:1511.06434Comment: Under review as a conference paper at ICLR 2016. http://arxiv.org/abs/1511.06434.

Rendell, P. 2000. *A Turing Machine In Conway's Game Life.* https://www.ics.uci.edu/~welling/teaching/271fall09/Turing-Machine-Life.pdf.

Rigaki, M., ja S. Garcia. 2018. "Bringing a GAN to a Knife-Fight: Adapting Malware Communication to Avoid Detection". Teoksessa *2018 IEEE Security and Privacy Workshops (SPW),* 70–75. Toukokuu. doi:10.1109/SPW.2018.00019.

Ronneberger, Olaf, Philipp Fischer ja Thomas Brox. 2015. "U-Net: Convolutional Networks for Biomedical Image Segmentation". *CoRR* abs/1505.04597. arXiv: 1505.04597. http://arxiv.org/abs/1505.04597.

Specht, Donald F. 1991. "A general regression neural network". *IEEE transactions on neural networks* 2 (6): 568–576.

Vondrick, Carl, Hamed Pirsiavash ja Antonio Torralba. 2016. "Generating Videos with Scene Dynamics". Teoksessa *Advances in Neural Information Processing Systems 29,* toimittanut D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon ja R. Garnett, 613–621. Curran Associates, Inc. http://papers.nips.cc/paper/6194-generating-videos-with-scene-dynamics.pdf.

Zhao, D., J. Weng ja Y. Liu. 2017. "Generating traffic scene with deep convolutional generative adversarial networks". Teoksessa *2017 Chinese Automation Congress (CAC),* 6612–6617. Lokakuu. doi:10.1109/CAC.2017.8243968.

Zhu, Jun-Yan, Taesung Park, Phillip Isola ja Alexei A Efros. 2017. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". Teoksessa *Computer Vision (ICCV), 2017 IEEE International Conference on.*