



Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Kauri Lehtinen

Kuvan äänentaajuuteen yhdistävä neuroverkkosovellus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

9.11.2018

Tekijä Otsikko	Kauri Lehtinen Kuvan äänentaajuuteen yhdistävä neuroverkkosovellus
Sivumäärä Aika	34 sivua 9.11.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Mediatekniikka
Ohjaaja	Tutkijaopettaja Aarne Klemetti
<p>Insinööriyönä kehitettiin internetselaimessa toimiva sovellus, joka koneoppimista hyödyntäen assosioi kuvan tiettyyn äänentaajuuteen. Kuva- ja äänilähteenä sovellukselle toimivat kaikki kamerat ja mikrofoni, joihin internetselaimella on pääsy. Käyttäjä tuottaa haluamallaan tavalla erilaisia äänentaajuuksia, joihin sovellus yhdistää kuvan. Näin koneoppimismalli oppii yhdistämään tietyn asetelman kuvassa tiettyyn äänentaajuuteen.</p> <p>Sovelluksessa käytettiin TensorFlow.js-Javascript-kirjastoa, joka mahdollistaa koneoppimismallien koulutuksen ja käytön internetselaimessa. Esikoulutettua MobileNet-mallia käytettiin esimerkkiaineiston esiprosessoinnissa, ja varsinainen ennustaminen toteutettiin omassa mallissaan. Äänen prosessoinnissa käytettiin Web Audio API -Javascript-rajapintaa. Insinööriyöraportissa selvitettiin koneoppisen teoriaa, sovelluksen kehityksessä käytettyjä menetelmiä ja sovelluksen toiminta.</p> <p>Aineiston keräämisessä ilmenneen ongelman vuoksi sovellus jäi soveltuvuusselvitystasolle, minkä vuoksi suunniteltua koneoppimismallin optimointia ei voitu järkevästi toteuttaa. Sovellus kuitenkin osoittautui toimivan pienilläkin aineistomäärillä, kun koulutettavat luokat erosivat toisistaan riittävästi. Liian pienet eroavaisuudet kuvissa eri luokkien välillä olivat pienellä aineistomäärillä liian samankaltaisia, eikä malli kyennyt tuottamaan luotettavia ennusteita.</p> <p>Riittävällä aineistomäärällä koulutettua optimoitua koneoppimismallia voitaisiin käyttää käyttöliittymänä esimerkiksi yksinkertaisissa web-peleissä tai soittamisen opetuksessa. Sovellus voisi verrata käyttäjän äänentuottamistekniikkaa hyvinä pidettyihin tekniikoihin ja ohjeistaa käyttäjää oikeaan suuntaan toiminnassa.</p>	
Avainsanat	neuroverkko, TensorFlow.js, tietokonenäkö

Author Title Number of Pages Date	Kauri Lehtinen Neural network application connecting the image to the audio frequency 34 pages 9 November 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Media Technology
Instructor	Aarne Klemetti, Researching Lecturer
<p>The goal of this final year project was to develop an application which exploits machine learning. The application runs fully in the browser and associates audio frequencies to images. User produces audio frequencies as they wish and machine learning model learns to connect certain audio frequencies to corresponding images. Applications can process video- and audio streams from any video or audio source that is available for the browser.</p> <p>TensorFlow.js javascript-library was used to teach and use machine learning models in the browser and audio processing was managed with Web Audio API javascript interface. Pre-trained MobileNet model was used for data preprocessing and predictions were produced in its own model. In thesis part machine learning theory, used methods and application principles were covered.</p> <p>Because of the problem detected while gathering data the application got stuck to proof of concept level. Due to that planned optimization of the model could not been implement in a meaningful matter. The application though proved to work with small amount of data if the classes to be taught differed sufficiently.</p> <p>If taught with sufficient amount of training data the model could be used as an interface for web-based games or to teach how to play musical instruments. The application could compare users sound producing technique to good practices and thus guide user to preferable direction.</p>	
Keywords	Machine learning, TensorFlow.js, Computer vision

Sisällys

Lyhenteet

1	Johdanto	1
2	Koneoppiminen	2
2.1	Terminologia	4
2.2	Oppimistavat	5
2.3	Aineisto	8
2.4	Koneoppimistekniikat	12
2.4.1	Lineaarinen regressio	12
2.4.2	Logistinen regressio	17
2.5	Mallin yleistäminen	18
2.6	Mallin sääntely	20
2.7	Neuroverkot	20
2.8	Konvolutionaaliset neuroverkot	21
3	Neuroverkkosovelluksessa käytetyt tekniikat	23
3.1	TensorFlow.js-kirjasto	23
3.2	WebGL-rajapinta	25
3.3	MobileNets-neuroverkot	27
3.4	Web Audio API -rajapinta	27
4	Neuroverkkosovelluksen rakenne ja toiminta	28
4.1	Sovelluksen rakenne	28
4.2	Alkuvalmistelut	28
4.3	Aineiston keräys	30
4.4	Mallin koulutus	31
4.5	Mallin avulla ennustaminen	33
4.6	Sovelluksen haasteet	34
5	Yhteenveto	34
	Lähteet	36

Lyhenteet

WebGL	Web Graphics Library. Javascript-rajapinta, joka mahdollistaa näytönohjaimen hyödyntämisen internetselaimessa.
OpenGL	Open Graphics Library. Ohjelmointirajapinta tietokonegrafiikan tuottamiseen.
GPGPU	General-purpose computing on graphics processing unit. Tekniikka, jonka avulla näytönohjaimessa suoritetaan laskentaa.

1 Johdanto

Koneoppimisen arvo on huomattu useilla, paljon aineistoa käsittelevillä teollisuuden aloilla. Koneoppimisen avulla yritykset ja organisaatiot pystyvät toimimaan tehokkaammin. Rahoituselämässä pankit ja yritykset käyttävät koneoppimista tunnistamaan tärkeitä tietoja aineistossa ja välttämään petoksia. Valtion virastot käyttävät koneoppimista toimintojen tehokkuuden lisäämiseksi ja rahan säästämiseksi. Terveystieteiden alalla käytetään koneoppimista arviomaan potilaan terveydentilaa erilaisten laitteiden ja sensoreiden avulla reaaliajassa. Koneoppimiseen törmää nykyisin lähes kaikilla kuviteltavissa olevilla teollisuuden aloilla.

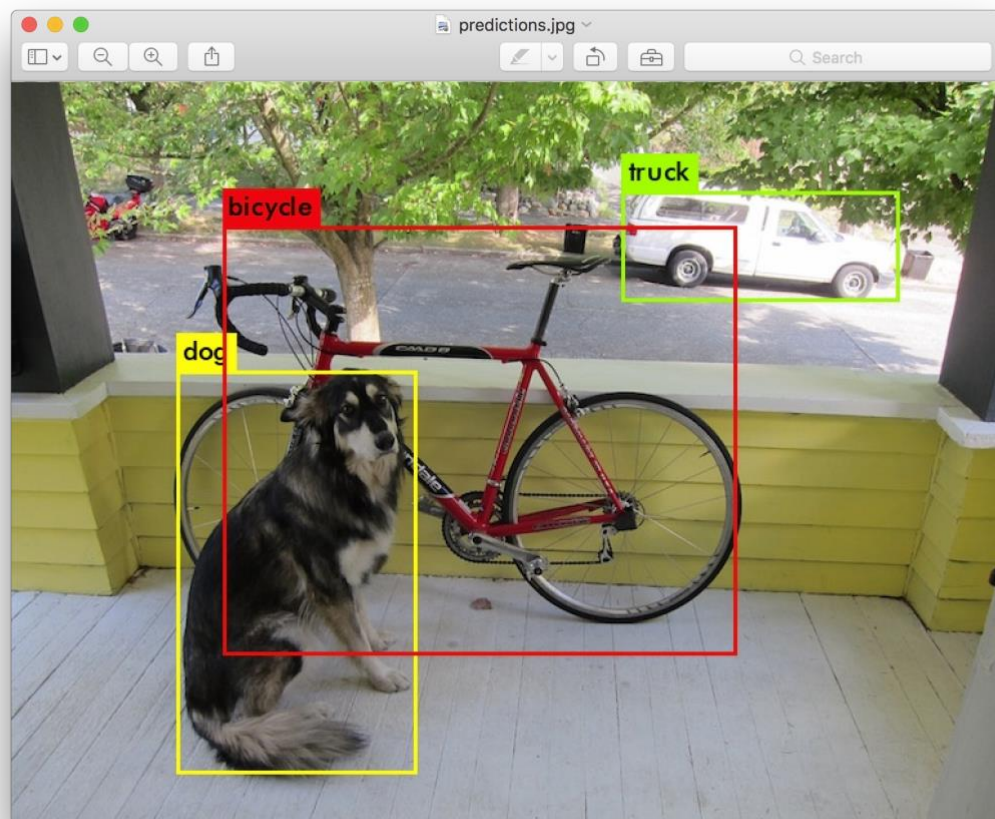
Koneoppimisen avulla pystyy ratkaisemaan ongelmia, joiden ratkaiseminen käsin olisi tarpeettoman työlästä, ellei jopa mahdotonta. Ihminen tunnistaa kasvot ja ymmärtää puhetta alitajuisesti, mutta jos yrittää ilman koneoppimista kehittää ohjelman, joka tekee saman, on tehtävä hyvin haastava tai mahdoton. Tällaisista tehtävistä koneoppiminen suoriutuu erinomaisesti.

Koneoppiminen muuttaa tapaa, jolla ajattelemme ongelmia. Kun tehdään havaintoja epävarmasta maailmasta, painopiste siirtyy matematiikasta luonnontieteeseen. Logiikan sijaan tehdään kokeita ja käytetään tilastoja tavoitteen saavuttamiseksi. Ihmiseltä jäävät helposti monet sellaiset säännönmukaisuudet aineistossa huomaamatta, jotka koneoppimisen avulla löytää helposti.

Insinöörityönä kehitetään koneoppimista hyödyntävä web-sovellus, joka ennustaa käyttäjän tuottaman äänentaajuuden videosyötteeseen perustuen. Sovelluksen kehityksessä käytetään TensorFlow.js-nimistä ohjelmointikehystä, joka mahdollistaa koneoppimismallien rakentamisen ja käytön internetselaimessa. Insinöörityöraportti koostuu koneoppimisen perusteista, sovelluksessa käytettävistä tekniikoista ja sovelluksen toiminnan kuvaamisesta.

2 Koneoppiminen

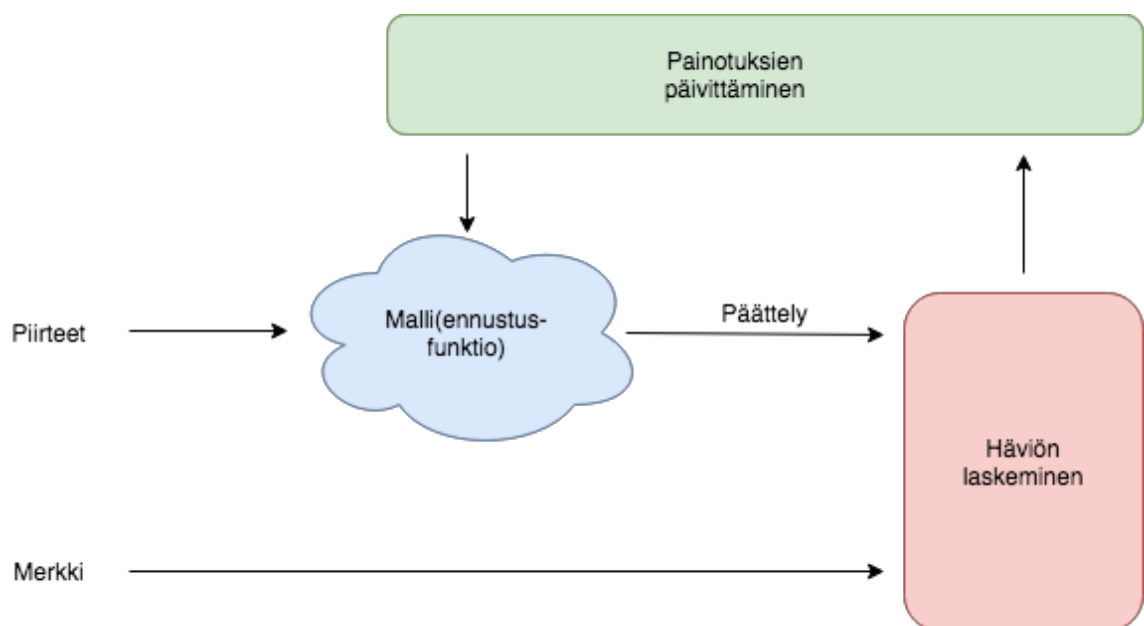
Koneoppiminen on järjestelmä tai ohjelma, jolla voi rakentaa ennustavan mallin syöteaineistosta. Ohjelma käyttää mallia tekemään hyödyllisiä ennustuksia uudesta aineistosta, joka on samasta lähteestä kuin mallin koulutukseen käytetty aineisto, mutta jota ei ole käytetty mallin koulutuksessa. Mallin palauttavat ennustukset ovat syötteen pohjalta laskettuja todennäköisyyksiä. Esimerkiksi kuvista eri objekteja tunnistava koneoppimismalli voi palauttaa todennäköisyyden, että kuvassa on koira 99 prosentin varmuudella. Kuvassa 1 on YOLO-algoritmin (You Only Look Once) palauttamien arvojen pohjalta tuotettu kuva, johon on merkitty kuvasta löydetyt objektit. Koneoppiminen viittaa myös tieteenalaan, joka tutkii koneoppimisohjelmia ja järjestelmiä. (1.)



Kuva 1. YOLO-algoritmin palauttamien arvojen pohjalta tuotettu kuva (2).

Koneoppiminen auttaa sitä hyödyntävää ohjelmaa suorittamaan tehtäviä ilman täsmällistä ohjelmointia tai sääntöjä. Jos on esimerkiksi kirjoittanut käsin ohjelman (laatinut kaikki säännöt), joka korjaa kielioppivirheet englannin kielestä, ja haluaa laajentaa ohjelman muille kielille, joutuu aloittamaan jokaisen kielen kohdalla käytännössä alusta. Jos ohjelma pohjautuu koneoppimiseen, tarvitsee optimaalisessa tilanteessa vain hankkia aineistoa halutusta uudesta kielestä ja syöttää se samaan malliin, jolla ohjelman englanninkielinen versio laadittiin. Tuloksena saadaan uusi kieliversio ohjelmasta.

Tavanomainen koneoppimisskenaario koostuu aineistosta, mallista ja koneoppimisalgoritmista. Malli on infrastruktuuri, jota koneoppimisalgoritmi muokkaa. Kun malliin syötetään aineistoa, optimoi koneoppimisalgoritmi mallin sisäisiä painotuksia kohti mahdollisimman pientä häviötä. Mallista muodostuu aineiston ja koneoppimisalgoritmin yhteistyöllä funktio. Kuvassa 2 on kaavio, joka kuvaa koneoppimisprosessia.



Kuva 2. Koneoppimisprosessi (3).

Koneoppimismallin koulutuksen peruseriaate on pyrkiä minimoimaan häviö (loss). Häviö on ennustetun arvon ja oikean arvon erotus. Huonossa ennusteessa on suuri häviö, ja jos mallin ennuste on täydellinen, häviö on 0. Koulutuksen tavoitteena on löytää

mallille koneoppimisalgoritmin avulla sopivat painotukset (weights) ja aloituspiste (bias), joiden tuottama häviö on mahdollisimman pieni läpi koko esimerkkiaineiston. (4.)

2.1 Terminologia

Koneoppimisen terminologiassa merkki (label) on asia, jota yritetään päätellä. Merkki voi olla esimerkiksi eläinlaji tai lämpötila. Piirre (feature) puolestaan on aineistossa oleva piirre. Niitä voi yksi tai useampia. Esimerkiksi kuvassa, joka esittää koira, voi olla piirteenä pelkkä koira tai sen lisäksi koiran rotu, ikä ja karvan väri. Piirteitä käytetään mallin koulutuksessa syötemuuttujina (input variable), jotka tuottavat ennustavan mallin. Ohjatussa oppimisessä häviö lasketaan vertaamalla ennustetta merkkiin.

Aineistoesimerkki on yksi rivi aineistossa. Merkitty esimerkki sisältää sekä merkin (esimerkiksi kuvan kohdalla tiedon siitä, mitä kuvassa on) että piirteet. Merkitsemätön esimerkki sisältää piirteet, mutta ei merkkiä (kuvan tapauksessa tietoa siitä, mitä kuvassa on). Kun malli on koulutettu merkityillä esimerkeillä, sitä käytetään tuottamaan ennusteita merkitsemättömien esimerkkien pohjalta (koneoppimismalli antaa jonkin todennäköisyyden siitä, mitä kuvassa on). Kuvan 3 taulukossa on viisi merkittyä esimerkkiä.

housingMedianAge (feature)	totalRooms (feature)	totalBedrooms (feature)	medianHouseValue (label)
15	5612	1283	66900
19	7650	1901	80100
17	720	174	85700
14	1501	337	73400
20	1454	326	65500

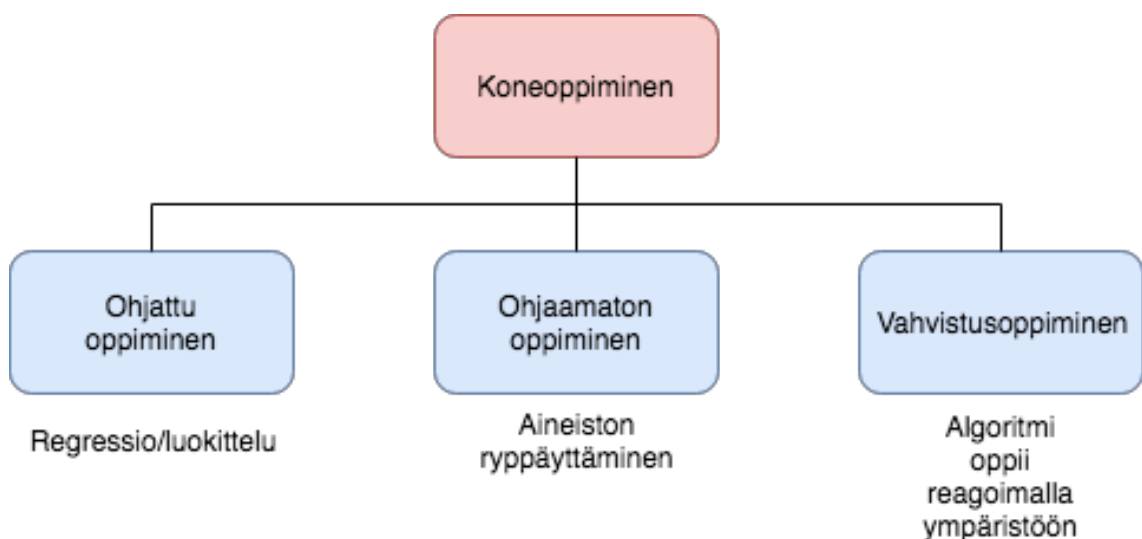
Kuva 3. Taulukossa on viisi riviä merkittyä aineistoa, joista kukin vastaa yhtä aineistoesimerkkiä. Merkitsemätön versio taulukon aineistosta on muuten sama, mutta oikeassa reunassa oleva merkkisarake puuttuu. (5.)

Malli (model) yhdistää piirteet merkkiin. Esimerkiksi malli, joka tunnistaa koiran kuvasta, osaa etsiä kuvasta piirteitä, jotka ovat koiramaisia. Niitä voivat olla esimerkiksi tumma kirsu kuonon päässä tai hännän kapea muoto. Mallia koulutettaessa se oppii asteittain esimerkkien avulla merkin ja piirteiden suhteen. Valmista mallia käytetään päättämään merkitsemättömän aineiston merkki. (5.)

Regressiomalli ennustaa jatkuvia arvoja, kuten asunnon hintaa tai klikkauksen todennäköisyyttä, kun taas luokittelumalli ennustaa tarkkoja arvoja. Regressiomallin tuottamat arvot ovat tyypillisesti liukulukuja, kun taas luokittelumallin tuottamat arvot erillisiä (esimerkiksi kissa, koira, lammas). (5.)

2.2 Oppimistavat

Eri oppimistavat jaotellaan niille annettavan aineiston perusteella. Kolme yleisintä oppimistapaa ovat ohjattu oppiminen, vahvistusoppiminen ja ohjaamaton oppiminen. Kuvassa 4 on kaavio oppimistavoista.



Kuva 4. Eri koneoppimisen oppimistavat (6).

Näiden tapojen lisäksi on siirto-oppiminen, jota insinööritöyönä tehdyssä web-sovelluksessa on käytetty ohjatun oppimisen ohella. Koska insinööritöyössä on käytetty

siirto-oppimisen lisäksi ohjattua oppimista, keskitytään tässä luvussa jatkossa ohjattuun oppimiseen muiden oppimistapojen esittelyn jälkeen.

Ohjattu oppiminen

Ohjatussa oppimisessa koneoppimismalli koulutetaan tunnetulla aineistolla. Aineisto koostuu syöte-tulospareista eli aineisto on merkittyä. Mallille syötetään esimerkkejä, joiden pohjalta se koneoppimisalgoritmin avulla optimoi mallille painotukset. Mallin koulutus on iteratiivista, ja mallia optimoidaan mahdollisimman pieneen häviöön yrityksen ja erehdyksen kautta. Iteratiivinen lähestymistapa on vallitseva koneoppimismallin koulutuksessa, koska se mukautuu hyvin isoihin aineistomääriin (3).

Koulutuksen aikana mallille syötetään kerralla aineistosta yksi tai useampia esimerkkejä. Malli tekee esimerkkien pohjalta ennusteen senhetkisillä painotuksillaan ja ennustetta verrataan oikeaan tulokseen. Jos ennuste on väärä, algoritmi laskee painotuksille uudet arvot iteraatiossa olevien esimerkkien tuottaman häviön perusteella. Uudet painotukset ohjaavat mallia kohti pienempää häviötä. Mallia iteroidaan, kunnes häviö saadaan minimoitua, eli se ei enää muutu tai muutos on hyvin pientä. Tällaista mallia kutsutaan lähentyneeksi (3).

Vahvistusoppiminen

Sekä ohjatussa että vahvistusoppimisessa verrataan syötettä tulokseen. Vahvistusoppimisessa on kuitenkin mukana myös palkinto-funktio, joka antaa agentille palautetta toiminnoista. Agentin tavoitteena on maksimoida odotettu kumulatiivinen palkintojen määrä ympäristössään. (6.)

Vahvistusoppimisessa agentti (esimerkiksi robotti) oppii käyttäytymään ympäristössään suorittamalla toimintoja, joista se saa palkintoja tai rangaistuksia. Palkintona voi olla esimerkiksi robotin tapauksessa pystyssä pysyminen ja rangaistuksena kaatuminen (7).

Vahvistusoppiminen on tavoitteellista oppimista. Tavoite voi olla esimerkiksi kuoppaisen labyrintin keskellä olevan aarteen löytäminen. Jos olemassa oleva koulutusaineisto puuttuu, agentti oppii kokemuksesta. Se kerää yrityksen ja erehdyksen kautta koulutusesimerkkejä suorittaessaan tehtäväänsä.

Vahvistusoppiminen voidaan esittää silmukkana. Aluksi agentti saa tilan S0 ympäristöltään. S0-tilaan perustuen agentti suorittaa toiminnon A0, joka voi olla esimerkiksi liike johonkin mahdolliseen suuntaan. Silmukka päättyy, kun toiminnon jälkeen ympäristö siirtyy tilaan S1 ja antaa agentille jonkin palkinnon R1 (esimerkiksi jos robotti on pysynyt pystyssä liikkeen jälkeen) tai rangaistuksen P1 (robotti on kaatunut). (8.)

Ohjaamaton oppiminen

Ohjaamattoman oppimisen tavoite on mallintaa aineiston alla piilevää rakennetta tai jakaumaa ja tällä tavoin oppia aineistosta. Koulutuksessa käytetään tyypillisesti merkitsemätöntä aineistoa, joten syötemuuttujille ei ole vastaavaa ulostulomuuttujaa (9). Ohjatussa oppimisessa ei ole ”oikeita vastauksia”, toisin kuin ohjatussa oppimisessa.

Yleinen ohjaamattoman oppimisen käytätapa on aineiston jaottelu samankaltaisten aineistoesimerkkien ryppäiksi. Jos esimerkiksi ohjelman tarkoitus on ehdottaa käyttäjälle uutisartikkeleita, algoritmi etsii samankaltaisia artikkeleita, kuin mitä käyttäjä on aiemmin lukenut. Ohjaamattomassa oppimisessa algoritmien annetaan itsenäisesti etsiä rakenteita aineistosta.

Siirto-oppiminen

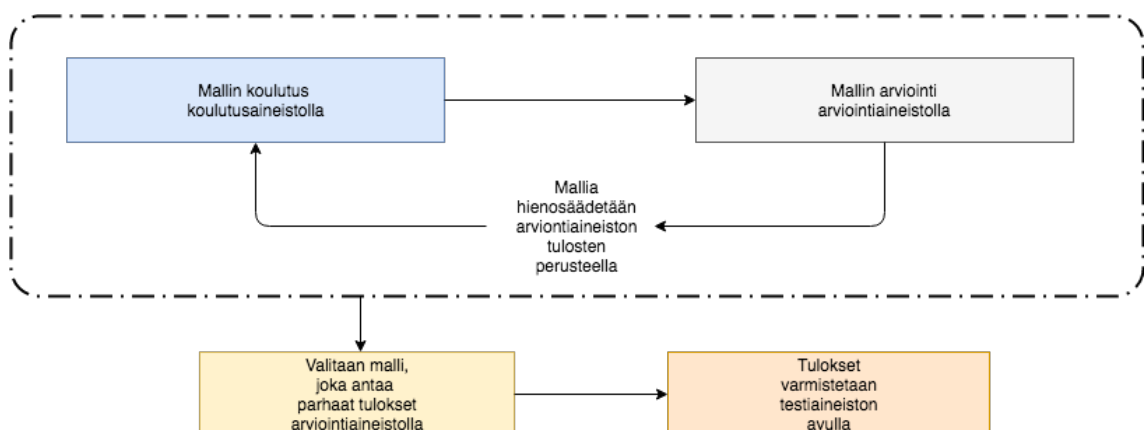
Siirto-oppimisessa johonkin tiettyyn tehtävään koulutettua koneoppimismallia A käytetään aloituspisteenä mallille B. Malli B käyttää syötteenään mallin A ulostuloja suorittaessaan omaa erillistä, mutta A:n tehtävään liittyvää tehtävää. Siirto-oppiminen on suosittua, koska se mahdollistaa koneoppimismallin koulutuksen suhteellisen pienellä aineistolla. Tämä on hyödyllistä, koska useimmista tosielämän ongelmista ei ole saatavilla miljoonia merkittäviä aineistoesimerkkejä, joilla pystyy rakentamaan tarpeeksi toimivan koneoppimismallin. (10.)

Kun pystytään hyödyntämään olemassa olevaa koneoppimismallia, säästetään myös runsaasti aikaa. Esimerkiksi kielen prosessointimallin kehittäminen vaatii massiivista laskentaa ja on aikaa vievää, joten valmiin mallin käyttäminen on suuri etu.

2.3 Aineisto

Ohjatussa koneoppimisessa aineisto jaetaan yleensä aluksi kahteen osaan, koulutusaineistoon ja testiaineistoon. Koulutusaineiston avulla malli koulutetaan, minkä jälkeen sen suoriutumiskykyä arvioidaan testiaineiston avulla. Mitä enemmän aineistoa on, sitä paremman mallin sen avulla pystyy rakentamaan. Liian pieni koulutusaineiston määrä heikentää mallin ennustuskkyä, ja toisaalta, jos testiainestoa on liian vähän, heikentyy mallin toimivuuden arvioiminen.

Jos mahdollista, kannattaa aineisto jakaa vielä kolmanteen osaan, jota kutsutaan arviointiaineistoksi. Tällöin koulutusaineistolla koulutetun mallin toimivuutta arvioidaan arviointiaineistolla. Arviointiaineiston tulosten pohjalta mallia hienosäädetään ja arvioidaan sitten uudestaan arviointiaineiston avulla niin kauan, kuin on tarpeen. Tässä on vaarana, että malli ylisopeutuu (overfits) arviointiaineiston erityispiirteisiin, eli arviointiaineisto ikään kuin ”kuluu”. Tämän vuoksi mallia arvioidaan vielä käyttämättömän testiaineiston avulla. Testiaineiston tuloksia verrataan arviointiaineiston tuloksiin, ja näin pystytään arvioimaan mallin luotettavuutta. (11.) Kuvassa 5 on kaavio työnkulusta koulutuksessa, jossa aineisto on jaettu koulutus-, testi- ja arviointiaineistoon.



Kuva 5. Koulutuksen työnkulku, jossa aineisto on jaettu kolmeen osaan (11).

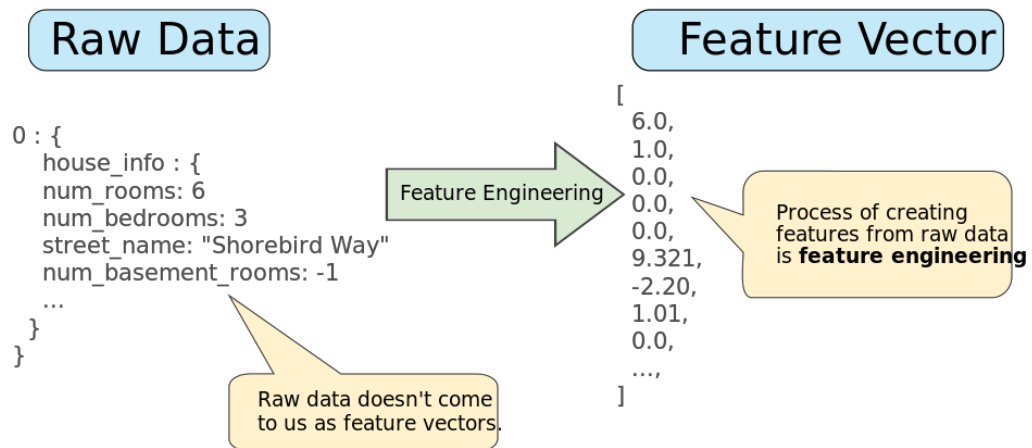
On tärkeää, että aineisto on ennen osiin jakamista sekoitettu. Näin varmistetaan, että kaikki aineistot edustavat kokonaisvaltaista kuvaa aineistosta eikä esimerkiksi ilmene tilannetta, jossa testiaineisto sisältää aineistoa vain talvelta ja koulutusaineisto vain kesältä. Lisäksi aineiston lähteen on syytä olla tasainen, jolloin se ei muutu ajan kuluessa. Aineiston lähteen tulee myös olla sama koulutus-, arviointi- ja testiaineiston kohdalla.

Mallia ei tule myöskään kouluttaa testiaineistolla tai arviointiaineistolla. Jos ennustukset ovat liian hyviä, on syytä tarkistaa, millä aineistolla malli on koulutettu (12). Koko testauksen idea menee, jos ennustuksia tehdään mallin jo tuntemilla aineistoesimerkeillä.

Aineiston esiprosessointi

Ennen kuin aineisto on koulutukseen soveltuvaa, tulee raaka-aineisto käsitellä. Koneoppimismallit tyypillisesti olettavat, että aineistoesimerkit esitetään reaailukuvektoreina. Tämän vuoksi raaka-aineisto muutetaan piirrevektoreiksi, ja tätä toimenpidettä kutsutaan piirteiden laatimiseksi (feature engineering).

Piirrevektori on joukko liukulukuarvoja, jotka vastaavat aineistojoukon arvoja. Numeeristen arvojen sisällyttäminen piirrevektoriin käy luonnollisesti, mutta merkkijonojen kohdalla täytyy hieman soveltaa. Kuvassa 6 on kaavio raaka-aineiston muuttamisesta piirrevektoriksi.



Kuva 6. Raaka-aineiston muuttaminen piirrevektoriksi (13).

Koneoppimismallit eivät pysty oppimaan merkkijonoista. Sen vuoksi merkkijonojen kohdalla käytetään one-hot-koodausta. One-hot-koodauksessa muodostetaan kaikista merkkijonovaihtoehdoista binäärivektori, jonka pituus on mahdollisten merkkijonovaihtoehtojen määrä. One-hot-koodauksessa siis yksi elementti saa arvon 1 ja kaikki muut arvot 0. Esimerkiksi jos vaihtoehdot ovat "kissa", "koira" ja "lammas" ja merkkijono on "koira", annetaan kissalle arvoksi 0, koiralle 1 ja lampaalle 0. Tapauksissa, joissa yksi vaihtoehdoista on oppimistavoitteiden kannalta tärkeämpi kuin muut, voidaan valitulle merkkijonolle antaa arvoksi 1 ja niputtaa kaikki muut vaihtoehdot omaan kategoriaan, jonka edustajat saavat arvokseen 0.

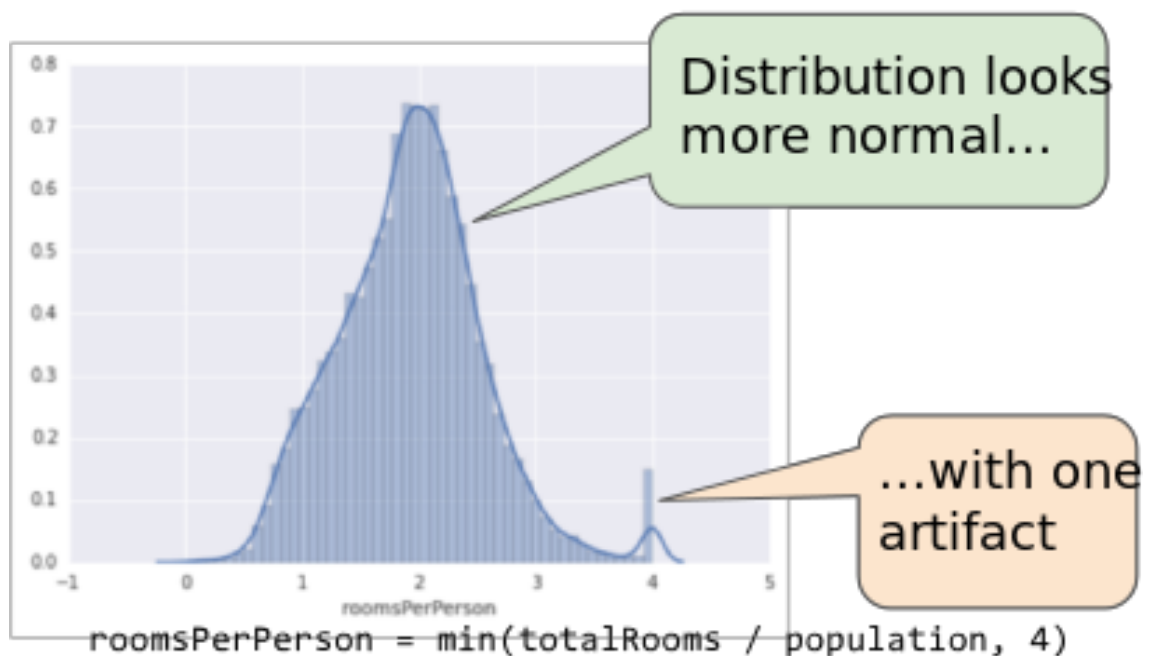
Multi-hot-koodaus on muuten sama kuin one-hot-koodaus, mutta siinä voi usea elementti saada arvokseen 1. Esim. jos ruoka on makeaa ja hapanta, mutta ei suolaista ja karvasta, saavat makea ja hapan arvokseen 1 ja muut perusmaut arvokseen 0. (13.)

Aineiston siivous

Mikäli aineistossa on useampia kuin yksi piirre, kannattaa sen piirteitä mukauttaa. Mukautuksella tarkoitetaan piirteen luonnollisten arvojen muuntamista standardialueelle. Jos piirteen luonnolliset arvot ovat esimerkiksi alueella 100–500, ne skaalataan esimerkiksi välille 0 ja 1 tai -1 ja 1. Skaalaaminen mahdollistaa koneoppimismallin nopeamman suppenemisen ja sopivien painotusten löytymisen sekä estää epänumeeristen (Nan) arvojen syntymisen. Ilman skaalausta malli keskittyy liikaa

piirteisiin, joiden arvoskaala on leveämpi. Kaikkien piirteiden ei tarvitse olla skaalattuna samalle alueelle, mutta mitä lähempänä skaalat ovat toisiaan, sitä tasapuolisemmin malli piirteitä kohtelee. (14.)

Myös suuret poikkeamat piirteiden arvoissa kannattaa siivota pois. Esimerkiksi, jos muutama arvo tuhannesta poikkeaa merkittävästi muista, on mitä todennäköisimmin kyseessä piirteen erikoisilmentymä, joka ei anna realistista kuvaa piirteestä. Jos poikkeamat jättää aineistoon, ne muokkaavat häviötä harhaanjohtavasti ja heikentävät näin mallin suorituskykyä. Yksi tapa tasoittaa poikkeamat on leikata piirre sopivasta kohdasta niin, että kaikki leikattua kohtaa suuremmat tai pienemmät arvot tapauksesta riippuen kootaan leikkauskohtaan. Kuvassa 7 piirre on leikattu arvon 4 kohdalta.



Kuva 7. Kaikki piirteen arvot, jotka ovat yli 4, on niputettu arvoon 4 (14).

Leikkaus toimintatavassa ei siis varsinaisesti poisteta poikkeamien arvoja vaan ne niputetaan leikkauskohtaan. Jos leikkauskohdan arvo on 4, muuttuvat kaikki sitä suuremmat arvot myös 4:ksi tapauksessa, jossa siivotaan suurempia arvoja.

Aineiston tarkistus

Ennen koneoppimismallin koulutusta kannattaa tarkistaa, puuttuuko aineistosta arvoja (esimerkiksi henkilö on unohtanut merkitä autolla ajettua kilometrit), onko aineistossa kaksoisesimerkkejä, onko aineisto väärin merkitty (kissan kuva on merkitty erheellisesti koiraksi) tai onko aineistoon tullut ylimääräisiä numeroita virheellisen näppäilyjen seurauksena (tai esimerkiksi lämpömittari on unohtunut auringonpaisteeseen). Jos havaitut virheet ovat oikaisun jälkeen luotettavia, ne korjataan, muutoin ne poistetaan, jotta ne eivät vääristä koneoppimismallin sopeutumista. Kaksoisesimerkkien ja puuttuvien arvojen etsiminen onnistuu yksinkertaisella ohjelmalla, mutta virheellisten arvojen tai merkintöjen haarukoiminen on vaikeaa. Niiden havaitsemiseen kannattaa käyttää aineiston visualisointia, eritoten histogrammia, josta näkee helposti esimerkiksi suuret virheelliset poikkeamat. (14.)

Myös piirteiden minimi- ja maksimiarvoja, mediaania, keskiarvoa ja keskihajontaa pystyy käyttämään virheellisten arvojen seulomiseksi. Kannattaa myös tarkkailla rajallisten piirteiden yleisimpiä arvoja: vastaako esimerkiksi jonkin eläinlajin määrä jollain maantieteellisellä alueella otaksuttua määrää.

2.4 Koneoppimistekniikat

Ohjatussa oppimisessa käytetään regressio- ja luokittelutekniikkaa koneoppimismallien kehityksessä. Regressiotekniikka tuottaa jatkuvia arvoja, jotka ovat tyypillisesti liukulukuja (15). Tuotetut arvot voivat olla esimerkiksi lämpötilan muutoksia tai asuntojen hintoja. Regressiotekniikkaa käytetään, kun ennuste on oikea numero. Luokittelutekniikka puolestaan tuottaa erillisiä arvoja. Luokittelutekniikan tuottama arvo voi esimerkiksi kertoa, onko kuvassa koira tai onko sähköposti roskapostia. Luokittelutekniikkaa käytetään, jos aineisto voidaan luokitella kahteen tai useampaan ennustevaihtoehtoon.

2.4.1 Lineaarinen regressio

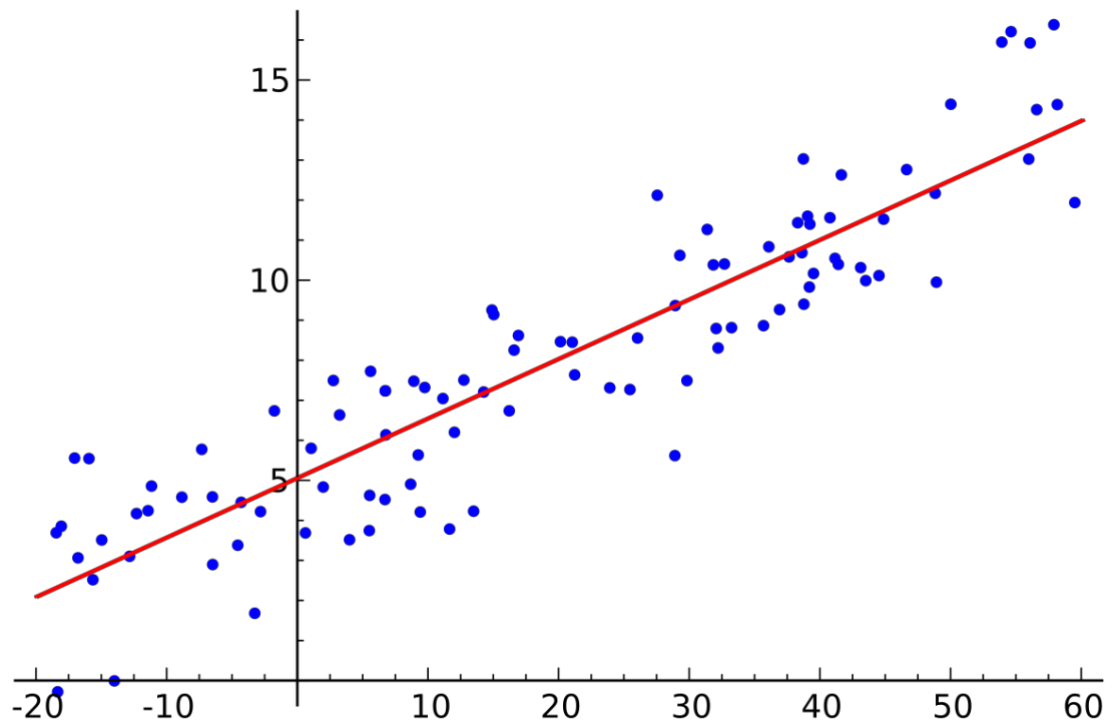
Lineaarinen regressio on regressioalgoritmi, joka tuottaa jatkuvan arvon syötepiirteiden lineaarisesta yhdistelmästä (16). Jos lineaarisen regression syöteaineistossa on vain yksi piirre, sitä kutsutaan yksinkertaiseksi lineaariseksi regressioksi. Jos

syöteaineistossa on enemmän kuin yksi piirre, sitä kutsutaan moninkertaiseksi lineaariseksi regressioksi. Yksinkertaisen lineaarisen regression avulla on hyvä havainnollistaa ohjattua oppimista.

Yksinkertainen lineaarinen regressio voidaan kuvata kaksiulotteisessa koordinaatistossa pisteiden ja suoran avulla. Yksinkertainen lineaarinen regressio muokkaa suoran yhtälöä sopeutumaan aineistoesimerkkeihin mahdollisimman hyvin. Sopeutumisessa käytetään häviön neliötä (Squared Loss tai L2 Loss), joka summaa oikeiden arvojen ja ennusteiden erotuksien muodostamat häviöt ja korottaa ne toiseen potenssiin virheen korostamiseksi. Häviön neliö -funktion kaava:

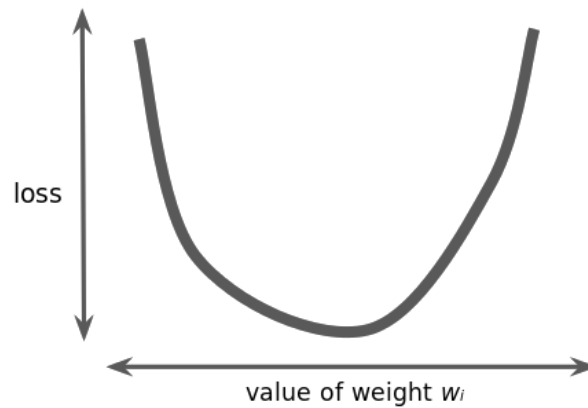
$$L2\ Loss = \sum_{i=1}^n (Y_{true} - y_{predicted})^2$$

Yksinkertaisen lineaarisen regression muodostama suoran yhtälö on muotoa $y' = b + w_1x_1$, jossa y' on yhtälön tuottama ennuste, b (bias) määrittää suoran leikkauskohdan y -akselilla, w_1 on piirteen painotus ja x_1 piirteen arvo. Lineaarisen regression tavoitteena on löytää b :lle ja w_1 :lle arvot, jotka sopeuttavat suoran aineistoon mahdollisimman hyvin. Kuvassa 8 on lineaarisen regression tuottama suora kuvattu kaksiulotteisessa koordinaatistossa.



Kuva 8. Punainen suora edustaa lineaarista regressiomallia ja siniset pisteet merkittyjä aineistoesimerkkejä (17).

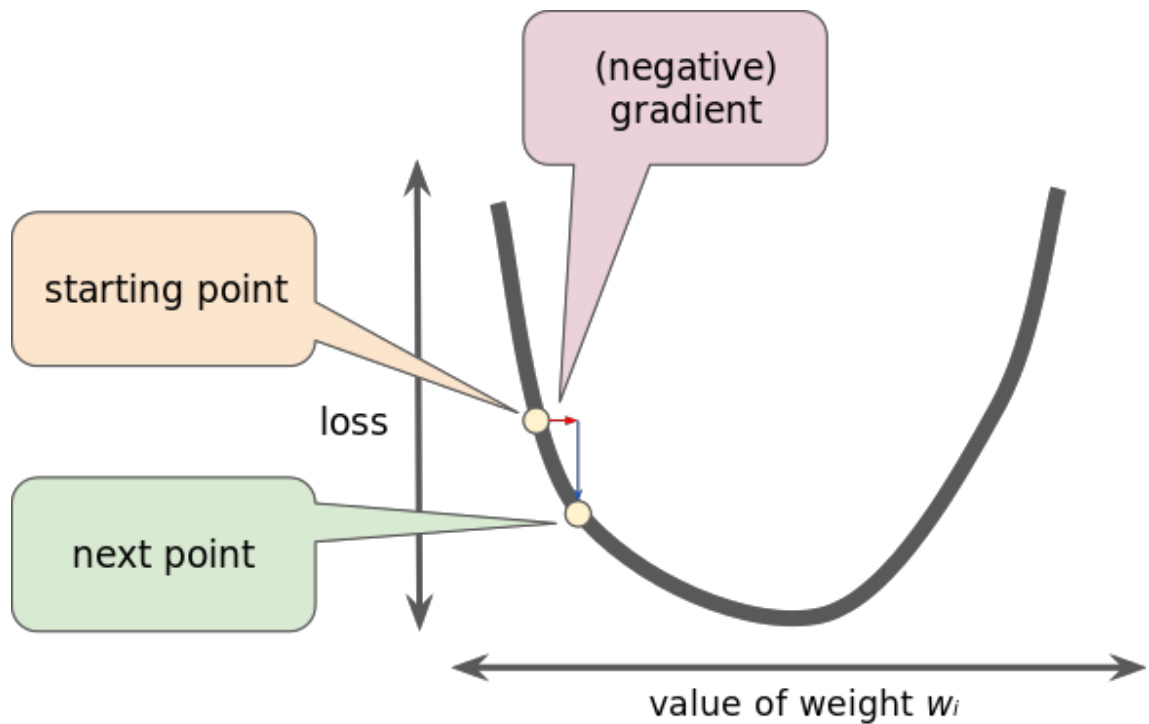
Jos kuvataan yksinkertaisen lineaarisen regression tuottaman häviön ja w_1 :n suhdetta koordinaatistossa, on kuvio aina kupera. Kuvassa 9 on w_1 :n ja häviön tuottama kuvio koordinaatistossa. Kuperilla ongelmilla on vain yksi minimi, jossa kuvion kaltevuus on tasan 0. Koulutuksen tavoitteena on selvittää, mikä w_1 :n arvo on mahdollisimman lähellä minimiä. (18.)



Kuva 9. Häviökäyrä. X-akseli kuvaa painotuksen arvoa ja y-akseli häviömäärää. (18.)

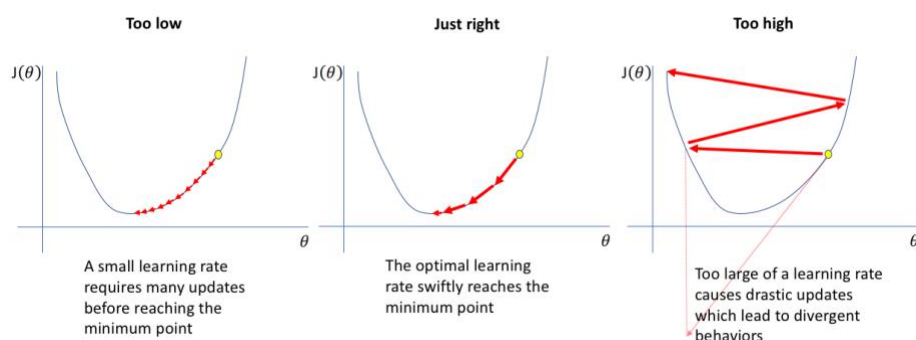
Yksi tapa löytää minimi on laskea häviö kaikilla mahdollisilla w_1 :n arvoilla. Tämä on kuitenkin varsin tehoton toimintatapa. Optimaalisten arvojen löytämiseen b :lle ja w_1 :lle voidaan käyttää esimerkiksi Gradient Descent -nimistä algoritmia. Aluksi b :lle ja w_1 :lle voidaan antaa mielivaltaiset arvot ja lasketaan niiden avulla y' . Saatua ennustetta y' verrataan oikeaan arvoon, ja näin saadaan laskettua häviön neliö. Tämän jälkeen algoritmi laskee uudet arvot b :lle ja w_1 :lle häviön neliön pohjalta siten, että seuraavalla kierroksella häviö olisi pienempi.

Gradient Descent -algoritmi laskee mielivaltaisen aloituskohdan kaltevuuden häviökäyrällä, eli kohdan derivaatan, joka kertoo algoritmille, missä suunnassa minimi sijaitsee. Tämän perusteella Gradient Descent -algoritmi ottaa askeleen uuteen kohtaan häviökäyrällä. Askel otetaan negatiivisen kaltevuuden suuntaan, missä minimi kuperan häviökäyrän tapauksessa sijaitsee. Algoritmi muuttaa b :n ja w_1 :n arvot vastaamaan uuden kohdan arvoja, joiden avulla lasketaan häviö seuraavalla kierroksella. Kuvassa 10 on havainnollistettu askel minimin suuntaan.



Kuva 10. Uuden pisteen painotuksen avulla lasketaan uusi häviö (18).

Askeleen kokoa määrittää skalaari, jota kutsutaan oppimisnopeudeksi (learning rate). Oppimisnopeus on hyperparametri, jota voidaan säätää koulutuksen tehostamiseksi. Jos oppimisnopeus on liian pieni, koulutuksen nopeus hidastuu, koska minimin saavuttaminen vaatii useita iteraatioita. Jos taas oppimisnopeus on liian suuri, pisteiden välisten askeleiden koko kasvaa jokaisella iteraatiolla ja algoritmi alkaa kiivetä häviökäyrää ylöspäin puolelta toiselle, löytämättä koskaan minimiä (19). Kuvassa 11 on havainnollistettu liian pieni, sopiva ja liian suuri oppimisnopeus.



Kuva 11. Eri oppimisnopeuksien käyttäytyminen (20).

Gradient descent -algoritmi kertoo kaltevuuden suuruuden oppimisnopeudella seuraavan kohdan määrittämiseksi. Jos kaltevuuden suuruus on esimerkiksi 2,5 ja oppimisnopeuden suuruus 0,01, seuraava tutkittava piste sijaitsee 0,025 yksikön päässä edellisestä.

2.4.2 Logistinen regressio

Luokittelutekniikassa koneoppimismalli ennustaa luokan annettujen piirteiden avulla, joita voi olla yksi tai useampia. Luokittelua kahteen luokkaan kutsutaan binääriseksi luokitteluksi, ja useampaan kuin kahteen luokkaan luokittelua kutsutaan moniluokkaiseksi luokitteluksi (21). Yksi luokittelualgoritmeista on logistinen regressio, joka tuottaa todennäköisyyden jokaiselle mahdolliselle eri luokalle luokitteluongelmassa.

Logistisessa regressiossa todennäköisyys saadaan soveltamalla sigmoid-funktiota lineaariseen ennusteeseen, joka voi olla arvoltaan mitä tahansa negatiivisesta äärettömyydestä positiiviseen äärettömyyteen (22). Sigmoid-funktio muuttaa logistisen regression tuottamat arvot todennäköisyyksiksi. Sen palauttavat arvot ovat 0:n ja 1:n välillä. Sigmoid-funktion kaava:

$$y = \frac{1}{1 + e^{-\sigma}}$$

Kaavassa σ vastaa kaikkia mallin piirteitä kerrottuna painotuksillaan ja summattuna:

$$\sigma = b + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

Logistisesta regressiomallista saatua todennäköisyyttä voi käyttää sellaisenaan, tai se voidaan muuttaa binääriseksi kynnysarvojen avulla. Jos mallista esimerkiksi saadaan ennuste, että sähköposti on 76 %:n todennäköisyydellä roskapostia, voidaan tulosta käyttää sellaisenaan. Jos taas käytetään kynnysarvoa ja kynnysarvoksi asetetaan 75 % (tai 0,75), muunnetaan kynnysarvon ylittävä 76 %:n todennäköisyys binääriseksi, eli se saa arvon 1, joka tulkitaan roskapostiksi.

Logistisessa regressiossa käytetään häviöfunktiota, joka on nimeltään Log Loss -funktio:

$$\text{Log loss} = \sum_{(x,y) \in D} -y \log(y') - (1 - y) \log(1 - y')$$

Kaavassa $(x,y) \in D$ on aineisto, joka koostuu merkatuista esimerkeistä, jotka ovat (x,y) -pareja. Y on merkki merkityssä esimerkissä, jonka arvo on joko 0 tai 1. Y' on ennustettu arvo, jonka arvo jotain 0:n ja 1:n väliltä. (23.)

Log loss -funktio mittaa mallin suoriutumiskykyä, ja koulutuksen aikana mallin tavoite on minimoida häviöfunktion tuottama arvo. Log Loss -funktion tuottama häviö kasvaa, jos ennustettu todennäköisyys poikkeaa merkistä. Jos ennuste on esimerkiksi 0,09 ja oikea arvo 1, on ennuste huono ja se tuottaa suuren häviön.

2.5 Mallin yleistäminen

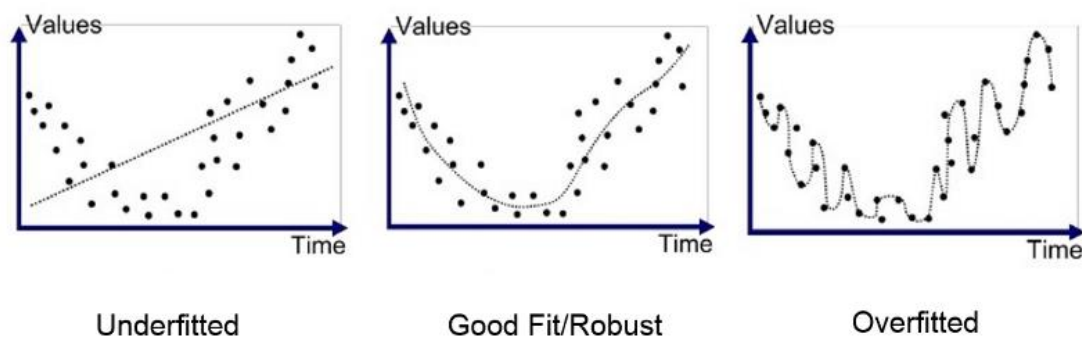
Koneoppimisen tavoitteena on kouluttaa malli, joka sopeutuu hyvin uuteen, ennennäkemättömään aineistoon. Yksi koneoppimisen haasteista on mallin ylisopeutuminen. Ylisopeutunut malli ennustaa koulutusaineiston täydellisesti, mutta kun mallilla tehdään ennustuksia uudella aineistolla, ovat ennusteet huonoja, koska malli on liikaa sopeutunut koulutusaineiston erityispiirteisiin. Ylisopeutuminen on seurausta mallin liiallisesta yrityksestä ottaa huomioon aineiston "kohinaa" (noise) koulutuksessa. Kohinalla tarkoitetaan aineiston edustajia, jotka eivät todella edusta aineiston ominaisuuksia, vaan sattumanvaraista mahdollisuutta (24).

Kuvitellaan esimerkiksi oppilas, joka ei osaa sanaakaan englantia ja jolle teini-ikäinen alkaa opettaa englantia. Oppilas (joka tässä tapauksessa voidaan mieltää kouluttamattomaksi malliksi) oppii englantia hyvin, mutta opittu englanti on slangipainotteista teini-ikäisen opettajan vuoksi (koulutusaineiston erityispiirre tässä tapauksessa), eikä henkilö opi kunnolla puhumaan englantia muulla tapaa. Tällöin oppilas on ylisopeutunut teini-ikäisen opettajan aineistoon.

Ideaalitilanteessa malli on mahdollisimman yksinkertainen ja ennustaa hyvin uudet esimerkit. Ylisopeutunut malli on tarpeettoman monimutkainen ja liian joustava. Vaikka

malli näyttäisi ennustavan aluksi hyvin, ei ilman mallin luotettavuuden arviointia kannata luottaa siihen, että mallin ennusteet olisivat hyviä myös tulevaisuudessa. (25.)

Jos koneoppimismallin ennusteet ovat huonoja, voi syynä olla myös alisopeutuminen. Alisopeutunut malli on liian joustamaton aineiston suhteen. Se ei ole tarpeeksi monimutkainen eikä sen vuoksi pysty yhdistämään syötettä ja merkkiä toisiinsa (26). Kuvassa 12 on alisopeutunut, sopeutunut ja ylisopeutunut malli koordinaatistossa kuvattuna.



Kuva 12. Koordinaatistoissa pisteet edustavat aineistoesimerkkejä ja viiva koneoppimismallia (27).

Mallin yli- tai alisopeutumista voi tarkkailla vertaamalla ennusteen häviötä koulutus- ja arviointiaineistossa. Ylisopeutumista voi ehkäistä vähentämällä mallin joustavuutta. Yksi keino on aikainen pysäytys, jossa koneoppimismallin koulutus keskeytetään, ennen kuin se on lähentynyt (28). Alisopeutumista voi ehkäistä lisäämällä aineiston kokoa (jos mahdollista) ja varmistamalla, että koneoppimismalli on riittävän monimutkainen. Monimutkaisuutta pystyy lisäämään lisäämällä aineiston piirteitä koulutuksessa (26).

Mallin yleistämisessä on myös tärkeää, että aineistoesimerkit ovat mielivaltaisessa järjestyksessä. Kun aineisto on sekoitettu, aineistoesimerkit ja niiden järjestys eivät vaikuta toisiinsa. Lisäksi aineiston lähteen on syytä olla tasainen, eli se ei muutu ajan kuluessa. Aineiston lähteen sisältö saattaa kuitenkin muuttua ajan mukaan; esimerkiksi kuluttajakäyttäytyminen muuttuu vuodenajan mukaan. (25.)

2.6 Mallin sääntely

Mallin sääntelyn avulla yritetään välttää mallin ylisopeutuminen. Aikaisen pysäytyksen lisäksi on muita sääntelystrategioita, jotka pyrkivät rankaisemaan mallin monimutkaisuudesta eli muovaavat mallia joustamattommaksi. Yksi tunnetuimmista sääntelystrategioista on L2-sääntely.

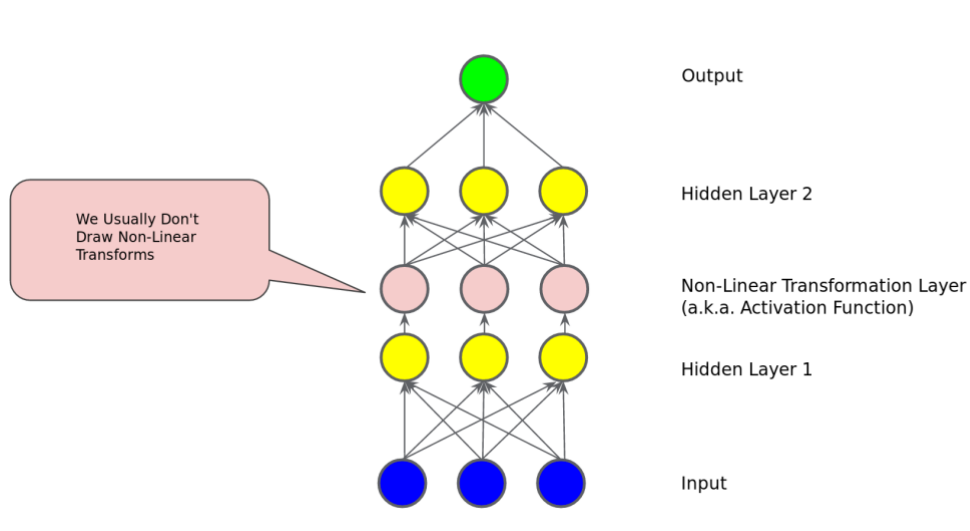
L2-sääntely pyrkii minimoimaan häviön ja mallin monimutkaisuuden yhdistelmän. Tätä kutsutaan rakenteelliseksi riskien minimoinniksi. L2-sääntely lisää häviöfunktioon rangaistustermiä, joka on yhtä suuri kuin mallin painotusten neliöiden summa (29). Täten L2-sääntely pakottaa painotukset pieniksi. Mitä suurempi rangaistustermi on, sitä pienempiä painotukset ovat. Yksinkertaistettuna kaavana esitetty L2-sääntelyn rakenne:

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \lambda \text{ complexity}(\text{Model}))$$

Kaavassa λ on skalaari, jolla voidaan säätää rangaistustermiä vaikutusta. Kun λ kasvaa, kasvaa myös rangaistustermiä vaikutus ja painotukset lähestyvät nollaa. Tällöin mallista tulee yhä yksinkertaisempi ja alisopeutumisen riski kasvaa. Kun taas λ pienenee, rangaistustermiä vaikutus pienenee ja mallin ylisopeutumisen riski kasvaa. Ideaalia λ :n arvoa ei voi määrittää, sillä se on aina aineistosta riippuvainen. (30.)

2.7 Neuroverkot

Piirteiden risteytys -tekniikalla pystytään ratkaisemaan yksinkertaiset epälineaariset ongelmat, mutta kun ongelmat monimutkaistuvat, täytyy käyttää neuroverkkoa. Neuroverkot pystyvät oppimaan epälineaarisuuksia itsestään, ilman manuaalista spesifiointia. Esimerkiksi kuvat, ääni ja video ovat monimutkaista aineistoa, jonka prosessointiin neuroverkot soveltuvat hyvin. Epälineaaristen ongelmien ratkaisemiseksi neuroverkot rakennetaan niin, että ne koostuvat piilotetuista tasoista ja aktivointifunktioista. Piilotetut tasot koostuvat neuroneista. Kuvassa 13 on yksinkertaisen neuroverkon rakenne.



Kuva 13. Kolmetasoisien neuroverkon rakenne (31).

Kuvassa 13 aineistoesimerkin piirteet tulevat neuroverkkoon sisääntulo-tason (input) kautta, jossa piirteet kerrotaan painotuksilla. Painotuksilla kerrotut piirteiden arvot jatkavat seuraavaksi piilotettuun tasoon, jossa ne syötetään aktivointifunktioon. Aktivointifunktion tuottamat arvot kerrotaan painotuksilla, ja ne etenevät toiseen piilotettuun tasoon. Lopuksi arvot etenevät ulostulo-tasoon, joka tuottaa ennusteen.

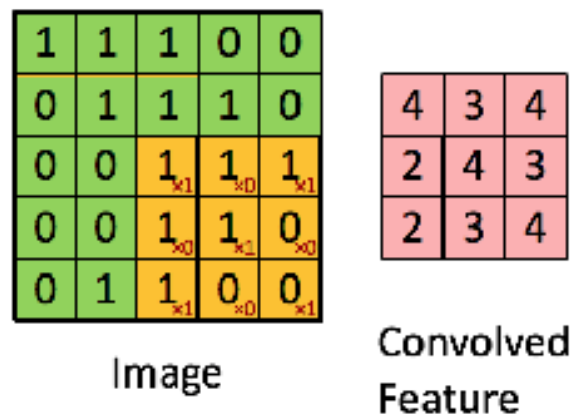
Ohjatussa oppimisessa neuroverkko ottaa sisäänsä aineiston ja vastaavat halutut merkit ja optimoi sisäiset painotuksensa niiden pohjalta. Neuroverkko laskee ennusteen aineistoesimerkille senhetkisten painotustensa mukaan, vertaa ennustetta todelliseen merkkiin ja muuttaa painotuksia siihen suuntaan, jossa ennuste on mahdollisimman lähellä todellista merkkiä. Näin neuroverkko sopeutuu aineistoon kierros kierrokselta. (32.)

2.8 Konvolutionaaliset neuroverkot

Konvolutionaalisia neuroverkkoja käytetään pääasiassa sellaisten algoritmien koulutuksessa, jotka oppivat tunnistamaan objekteja kuvista (33). Tietokone "näkee" kuvat eri tavalla, kuin ihmiset. Tietokoneelle kuva on kaksiulotteinen taulukko numeroita, jotka vastaavat pikseleitä. Värikuvissa näitä kaksiulotteisia tasoja on kolme, yksi jokaiselle RGB-väriavaruuden värille. Yksi numeroarvo taulukossa vastaa pikselin värin

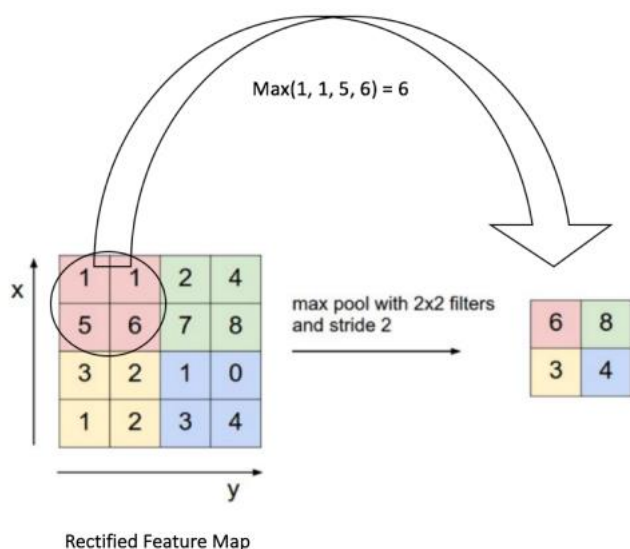
intensiteettiä, ja värin määrittää taulukon RGB-arvo. Algoritmeille täytyy näyttää miljoonia kuvia, ennen kuin ne pystyvät tekemään ennusteita uuden aineiston pohjalta (34). Värilliset kuvat syötetään konvoluutionaliseen neuroverkkoon kolmiulotteisina objekteina, tensoreina. Tensorissa on jokaiselle RGB-väriavaruuden värille oma kaksiulotteinen taso, joka leveydeltään ja korkeudeltaan vastaa kuvan pikselien määrää.

Matematiikassa konvoluutio on integraali, joka mittaa, kuinka paljon kaksi funktiota peittää toisensa, kun toinen ohittaa toisen (33). Konvoluutionalissa neuroverkoissa nämä funktiot ovat matriiseja, jotka kulkevat pikselitaulukoiden yli. Matriiseja kutsutaan suodattimiksi, ja ne etsivät kuvasta erilaisia muotoja. Kun suodatin kulkee pikselitaulukon yli, se tuottaa uuden tensorin tason. Taso koostuu arvoista, jotka kertovat, kuinka hyvin suodattimen etsimä muoto toteutui tarkkailualueella. Kuvassa 14 on vihreä matriisi, joka vastaa kuvaa ja sen pikseleitä. Keltainen matriisi kuvamatriisin sisällä on suodatin, joka kulkee kuvamatriisin yli. Oikealla on punaisessa matriisissa konvoluution tulos, piirrekartta.



Kuva 14. Konvoluutio. Piirrekartan arvot on tuotettu yksi kerrallaan, liu'uttamalla suodatin kuvan yli pikseli kerrallaan (35).

Toinen konvoluutionalissa neuroverkoissa käytetty tekniikka on pooling. Pooling-tekniikalla pienennetään piirrekarttoja. Pooling-tyyppä on erilaisia, kuten max-, average- ja sum-pooling. Esimerkiksi max-pooling valitsee tietyltä alueelta suurimman arvon. Kuvassa 15 on max-pooling havainnollistettuna.



Kuva 15. Max-pooling koostaa piirrekartasta uuden, pienemmän matriisin (35).

Konvolutionaaliset neuroverkot koostuvat piilotetuista tasoista, joissa suoritetaan sarja konvoluutioita ja pooling-operaatioita. Lisäksi niiden lopussa on luokittelija, joka antaa prosessoitujen piirteiden (pikselien) perusteella todennäköisyyden kaikille merkeille, joilla kyseinen verkko on koulutettu (34).

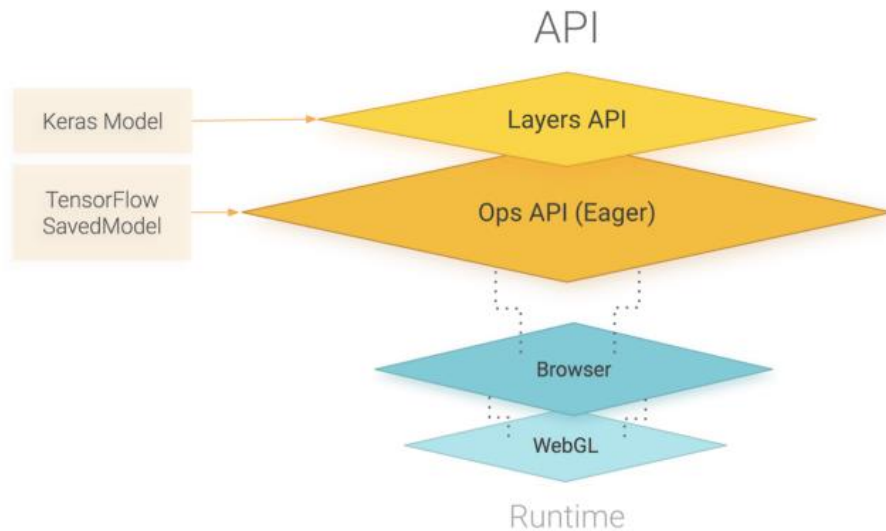
3 Neuroverkkosovelluksessa käytetyt tekniikat

3.1 TensorFlow.js-kirjasto

Tensorflow.js on avoimenlähdekoodin koneoppimis-Javascript-kirjasto, joka käyttää WebGL-rajapintaa (Web Graphics Library) hyväkseen rinnakkaislaskennassa. TensorFlowjs:n avulla pystyy kouluttamaan neuroverkkoja ja suorittamaan valmiita koneoppimismalleja internetselaimessa tai palvelimella Node.js:n avulla. TensorFlow.js:n avulla pystyy myös toteuttamaan koneoppimisen perusvaiheet eli koulutuksen ja ennustamisen.

Tensorflow.js mahdollistaa koneoppimisen liittämisen osaksi verkkosovelluksia. Se sisältää matalan ja korkean tason ohjelmointimahdollisuudet ja on kehittynyt

DeepLearn.js:stä, jota kutsutaan nykyisin TensorFlow.js:n ytimeksi (36). Kuvassa 16 on TensorFlow.js:n rajanpinnan rakenne.



Kuva 16. TensorFlow.js:n rakenne (37).

Internetselaimen päällä on Ops-rajapinta (entinen DeepLearn.js) ja ylimpänä layers-rajapinta, joka sisältää korkean tason työkalut koneoppimismallien luontiin ja koulutukseen internetselaimessa. TensorFlow.js tulee erottaa TensorFlow'sta, joka Googlen vuonna 2011 kehittämä patentoitu kone- ja syväoppimiskirjasto. TensorFlow on kirjoitettu C++-kielellä, ja se tukee Python-, R- ja Java-kieliä.

TensorFlow.js koostuu kolmesta perusyksiköstä. Ensimmäinen on tensori. Se koostuu joukosta arvoja, joista muodostuu yksi- tai moniulotteisia taulukoita. Tensorin muoto määritellään muoto-attribuutilla, joka kertoo konstruktorille, kuinka monta riviä arvojoukosta muodostetaan ja kuinka monta saraketta niissä on. Tensorin luomisen jälkeen sen arvoja ei pysty muuttamaan. Niihin kohdistuvilla operaatioilla muodostetaan uusia tensoreita.

Toinen perusyksikkö on operaatiot. Operaatioiden avulla voi manipuloida muuttumattomia tensoreita. Kun tensoreihin kohdistetaan jokin operaatio, se palauttaa

uuden tensorin operaation tuottamilla arvoilla. Operaatioita ovat esimerkiksi toiseen potenssiin korottaminen, kertominen, vähennyslasku ja yhteenlasku.

Kolmas ja tärkein perusyksikkö ovat yhdessä mallit ja tasot. Mallit on rakennettu yhdestä tai useammasta tasosta. Erilaisiin koneoppimistehtäviin tarvitaan yksilölliset mallit, ja se onnistuu yhdistelemällä erilaisia tasoja. Tasoista rakennetaan neuroverkkomallit, joita koulutuksen jälkeen voidaan käyttää ennustamaan arvoja. (38.)

TensorFlow.js:n etuna on sen helppokäyttöisyys. Asennusta tai ajureita ei tarvita. Koneoppiminen internetselaimessa avaa uusia mahdollisuuksia. Internetselain vapauttaa koneoppimisen käyttöympäristöön, joka tarjoaa runsaasti helposti saatavilla olevaa aineistoa käyttäjältä. Se mahdollistaa interaktiivisen koneoppimisen eri sensorien avulla. Mobiililaitteessa internetselaimelle voi sallia pääsyn lukuisiin sensoreihin, ja näin pystyy hyödyntämään esimerkiksi gyroskooppia ja kiihdytysanturia koneoppimisessa. Käyttäjän tuottaman aineiston ei tarvitse lähteä mihinkään internetselaimesta, mikä on tärkeää käyttäjän yksityisyyden kannalta.

3.2 WebGL-rajapinta

WebGL on Javascript-rajapinta, joka mahdollistaa näytönohjaimen hyödyntämisen internetselaimessa. WebGL:n on standardisoinut Khronos Group, joka on myös OpenGL:n (Open Graphics Library) takana. WebGL perustuu OpenGL ES:ään (OpenGL for Embedded Systems), joka on rajapinta edistyneen 2D- ja 3D-grafiikan prosessointiin sulautetuissa- ja mobiilijärjestelmissä (39).

WebGL piirtää pisteitä, viivoja ja kolmioita perustuen koodisyytteesen (40). Se siis muuttaa vektorit erillisiksi pikseliarvoiksi. WebGL:ää käytetään 2D- ja 3D-grafiikan prosessointiin internetselaimessa, mutta sitä voi hyödyntää myös GPGPU:ssa (General-purpose computing on graphics processing units), joka on tekniikka, jossa näytönohjaimella suoritetaan laskentaa. WebGL ei ole GPGPU:n tarpeisiin mukautettu, joten koneoppimiseen liittyvät operaatiot täytyy muuttaa WebGL:lle sopiviksi.

WebGL-ohjelmat koostuvat kahdesta osasta: prosessorissa ajettavasta, Javascriptilla kirjoitetusta kontrollointikoodista ja varjostinkoodista (shader code), joka suoritetaan

näytönohjaimessa (41). Kontrollointikoodi liittää geometriaa ja matriiseja näytönohjaimen muistiin ja hallitsee kanssakäymisen käyttäjän kanssa. Graafinen koodi on kirjoitettu GLSL-kielellä (Graphic Library Shading Language), joka on C-kielen kaltainen. Lähdekoodi ja muuttujien nimet täytyy antaa WebGL-kontekstille merkkijonoina, koska GLSL:ää ei pysty parsimaan luonnostaan Javascriptin avulla (42).

Varjostimia on WebGL:ssä kahdenlaisia. Kulmapistevarjostin (Vertex shader) ottaa sisäänsä geometriadataa ja muuttaa sitä. Se operoi kulmapistedataa, joka on tallennettu kulmapistepuskuriobjektiin (Vertex Buffer Object (VBO)). Kulmapistevarjostin suoritetaan ennen rasterointiprosessia. Se siis määrittää vektorin kulmapisteiden positiot näytöllä. Toinen varjostin on pikselivarjostin (fragment shader), joka suoritetaan rasterointi-prosessin jälkeen. Se suoritetaan kerran jokaiselle piirretylle pikselille, ja se määrittää pikselin värin. Kulmapiste- ja pikselivarjostimen yhdistelmää kutsutaan varjostinohjelmaksi (40). Kulmapistevarjostin palauttaa aina sijainnin, joka on 4-ulotteinen vektori $[x, y, z, w]$ leikkaavassa koordinaatistossa. Pikselivarjostin puolestaan palauttaa pikselin RGBA-väriarvon.

Rinnakkaislaskentaa voidaan harjoittaa kahdessa eri paikassa WebGL:n avulla, pikselivarjostimessa tai kulmapistevarjostimessa. Kun varjostimia hyödynnetään koneoppimisessa, suoritetaan pääasiallinen laskenta pikselivarjostimissa. Molempia varjostimia kuitenkin tulee käyttää, joten yleensä kulmapistevarjostinta käytetään pelkästään täyttämään näyttöalue (viewport). Kulmapistevarjostin palauttaa `gl_Position`-muuttujan, joka määrittää sijainnin, eikä sitä pysty suoraan lukemaan tai tallentamaan tekstuuriin tai muuhun objektiin, toisin kuin pikselivarjostimen palauttamaa `gl_FragColor`-muuttujaa. GPGPU-tekniikkaa hyödynnettäessä kulmapistevarjostinta käytetään ainoastaan täyttämään näyttöalue ja pikselivarjostimet hoitavat laskemisen. (42.)

WebGL on tällä hetkellä ainut tapa hyödyntää näytönohjaimen isoja rinnakkaislaskentamahdollisuuksia internetselaimessa, ja se nopeuttaa koneoppimisen vaatimaa laskentaa huomattavasti. WebGL:n avulla voi hyödyntää kaikkia näytönohjaimen kykyjä, ja näytönohjaimen ohjeistukset ovat samanlaiset riippumatta, siitä onko lähteenä WebGL tai sovellus, joka käyttää OpenGL:ää suoraan.

3.3 MobileNets-neuroverkot

Monet mobiililaitteissa tehtävät koneoppimistehtävät suoritetaan pilvessä. Kun kuva halutaan luokitella, se lähetetään palvelimelle, joka tekee luokittelun ja lähettää kuvan takaisin käyttäjälle. Mobiililaitteiden laskentateho kasvaa kuitenkin koko ajan, ja koneoppimismallien kompleksisuus laskee, mikä mahdollistaa koneoppimistehtävien suorittamisen tehokkaasti suoraan käyttäjän päätelaitteessa.

MobileNets-neuroverkot ovat joukko Googlen suunnittelemaa tehokkaita, kevyitä konvoluutionaalisia neuroverkkoja TensorFlow'lle. Ne on suunniteltu mobiili- ja sulautettujen visuaalisten sovellusten käyttöön. MobileNets-neuroverkot ovat resurssiystävällisiä ja toimivat nopeasti mobiililaitteissa. Niitä käytetään esimerkiksi kuvien luokitteluun ja tunnistamiseen, ja ne on koulutettu ImageNetin ILSVRC-2012-CLS-aineistolla (43). ImageNet on palvelu, joka tarjoaa merkittyjä kuva-aineistoja tutkijoille (44). Esikoulutettuja MobileNets-neuroverkkoja on useita erilaisia, joista voi valita tarpeisiinsa sopivimman viiveen ja koon perusteella. (45.)

3.4 Web Audio API -rajapinta

Web Audio API on korkeantason Javascript-rajapinta äänen prosessointiin internetselaimessa. Sen avulla luodaan audiokonteksti, jonka sisällä äänen prosessointi toteutetaan. Audiokontekstille osoitetaan prosessoitava äänilähde, jota voidaan prosessoida solmujen (node) avulla. Yhteen linkitetyt solmut muodostavat audioreitityskaavion. Solmujen avulla voidaan säätää esimerkiksi äänenvoimakkuutta. Lopuksi, kun halutut prosessoinnit on suoritettu, ohjataan prosessoitu ääni audiokontekstin ulosmenoon. (46.)

Web Audio API toimii kaikissa suosituimmissa internetselaimissa, paitsi Internet Explorerissa. Sitä käytetään esimerkiksi interaktiivisissa sovelluksissa ja web-pohjaisissa peleissä.

4 Neuroverkkosovelluksen rakenne ja toiminta

Insinöörityönä kehitin internetselaimessa toimivan neuroverkkosovelluksen, jolle annetaan syötteenä kuvaa ja ääntä. Koneoppimisen avulla sovellus oppii yhdistämään kuvan tiettyyn äänentaajuuteen. Kun koneoppimismalli on koulutettu, se ennustaa eri taajuuksia pelkkään kuvasyötteeseen perustuen. Periaatteessa äänilähde voi olla mikä vain, mutta taajuuden hallitsemiseksi syötin sovelluksen äänilähteenä sähkökitaran signaalia. Toteutin sovelluksen toiminnallisen logiikan Javascriptillä hyödyntäen TensorFlow.js-Javascript-kirjastoa. Sovelluksen ulkoasun toteutin HTML- (Hypertext Markup Language) ja CSS-kielen (Cascading Style Sheets) avulla. Sovellus toimii täysin internetselaimessa hyödyntäen WebGL-tekniikkaa laskennan hajauttamiseen.

Sovellus koostuu yhdestä html-tiedostosta ja viidestä Javascript-tiedostoista. Neljä viidestä Javascript-tiedostosta on moduuleja, jotka ladataan päätiedostoon (app.js) import-lauseen avulla.

4.1 Sovelluksen rakenne

Sovelluksen toiminnassa on kolme vaihetta. Ensimmäisenä on aineiston keräysvaihe, jossa sovellus assosioi web-kamerasta kaapattuja kuvia luokkiin. Luokkia on 47, ja ne vastaavat standardin e-vireisen sähkökitaran (22-nauhainen otelauta) mahdollisia nuottien taajuuksia, aina matalimmasta (E2, 82.40689 Hz) korkeimpaan (D6, 1174.659 Hz). Kun aineistoa on kerätty tarpeeksi, seuraa koulutusvaihe, jossa neuroverkko koulutetaan tunnistamaan luokka kuvasyötteestä. Koulutuksen jälkeen sovellus on valmis viimeiseen vaiheeseen, jossa kuvasyötettä ennustetaan koulutetun mallin avulla.

4.2 Alkuvalmistelut

Kun sovellus käynnistetään, ladataan aluksi TensorFlow.js-kirjasto index.html-tiedoston head-elementissä CDN:n (Content Delivery Network) avulla. CDN on hajautettu sisällönjakeluverkko, jonka avulla voi liittää staattista sisältöä verkkosivulle. App.js-tiedostossa luodaan Webcam-luokan instanssi osoittamalla html-tiedostossa luotu video-elementti luokan konstruktorille. Luokka sijaitsee erillisessä moduulissa ja sisältää

funktiot video-syötteen alustamiseen, kuvakaappaukseen videosta, kuvan rajaamiseen ja videon koon säätämiseen. Myös Dataset-luokasta luodaan uusi instanssi, jonka konstruktorille annetaan parametrina ennustettavien luokkien lukumäärä.

Kun app.js-tiedosto on alustettu, suoritetaan sen sisällä oleva init-funktio. Init-funktiossa ladataan aluksi web-kameran tuottama kuva ja laitteelle osoitettu äänilähde muuttujaan, Webcam-luokan setup-funktion avulla. Setup-funktiossa tarkistetaan aluksi, löytyykö web-kameran syöte. Jos syöte ei sisällä Media Stream API:n navigator.mediaDevices-objektiin, kokeillaan, löytyykö syöte navigator.webkitmediaDevices-, navigator.mozmediaDevices- tai navigator.msmediaDevices-objekteista, jotka ovat objektin spesifioituja versioita eri internetselaimille. Jos syöte löytyy, se tallennetaan navigator.mediaDevices-objektiin, ja jos syötettä ei löydy, navigator.mediaDevides-objektia ei luoda. Media Stream API on Javascript-rajapinta, joka mahdollistaa videosyötteen prosessoinnin internetselaimessa.

Kun kuvalähde on saatu tallennettua muuttujaan, ladataan MobileNet-malli (mobilenet_v1_0.25_224). Ladatusta MobileNet-mallista tallennetaan muuttujaan mallin taso (layer), jonka nimi on "conv_pw_13_relu". Tämä taso palauttaa Relu-funktion (Rectified Linear Unit) muokkaaman sisäiset aktivoinnit. Relu-funktio palauttaa arvon 0 saadessaan negatiivisen syötteen, ja positiiviset arvot se palauttaa sellaisenaan (47). Tason avulla luodaan TensorFlow.js-malli, jonka sisääntulotasoksi osoitetaan ladatun MobileNet-mallin sisääntulotasot ja ulosmenotasoksi "conv_pw_13_relu"-tason ulosmenot. Näin on luotu TensorFlow.js-malli, joka palauttaa sisäiset aktivoinnit, joita käytetään myöhemmin luotavan varsinaisen mallin syötteenä, eli kun sovelluksella tuotetaan ennusteita, aineisto prosessoidaan kahden mallin avulla.

Seuraavaksi tf.tidy-funktion sisällä "lämmitetään" juuri luotu malli antamalla yksi web-kamerakuva ennustettavaksi mallin predict-funktiolle. Tämä lataa painotukset valmiiksi näytönohjaimelle ja alustaa WebGL-ohjelmat, mikä nopeuttaa aineiston keräämistä myöhemmin. Tf.tidy-funktion sisällä luodut tensorit poistetaan automaattisesti näytönohjaimen muistista, kun funktio päättyy. Lopuksi asetetaan vielä "Collect Data" -nimiseen html-nappi-elementtiin klikkauksen käsittelijä, joka aktivoituessaan käynnistää aineiston keräämisen ja muuttaa napin "Stop Collecting Data" -napiksi, jota painettaessa aineiston kerääminen keskeytyy.

4.3 Aineiston keräys

Aineiston keräys alkaa 'Collect Data' -nappia painamalla. Tämä aktivoi collectData-funktion, joka luo audiokontekstin. Audiokontekstin lähdesolmulle osoitetaan äänilähteeksi web-kamerasta tuleva ääni. Tämän jälkeen alustetaan analysointisolmu, jota käytetään äänentaajuuden analysoinnissa. Taajuuden analysointia varten luodaan myös Javascript-solmu, joka mahdollistaa äänen käsittelyn ja analysoinnin Javascriptin avulla. Javascript-solmun onaudioprocess-ominaisuudelle osoitetaan funktio, jossa tapahtuu taajuuden analysointi sitä mukaa, kuin analysoitavia näytteitä tulee.

Ääninäytteen taajuuden analysointi aloitetaan luomalla Float32Array-taulukko, jonka kooksi asetetaan 1024. Taulukkoon tallennetaan analysointisolmun getFloatTimeDomainData-ominaisuuden aineisto, josta koostuu ääninäytteen aallonmuoto. Seuraavaksi lasketaan näytteen taajuus autokorrelointi-funktion avulla, jolle annetaan parametrina audiokonteksti ja vasta luotu taulukko. Jos autokorrelointi-funktio tunnistaa taajuuden, se palauttaa taajuuden arvon sellaisenaan, muutoin se palauttaa arvon -1. Jos taajuus ei ole suurempi kuin 1 175 Hz (korkein kitarasta kieltä venyttämättä saatava taajuus) tai arvo ei ole -1, etsitään lähimpänä saatua taajuutta vastaava nuotti kovakoodatusta nuotti-objektista ja nuotti tallennetaan muuttujaan.

Autokorrelointi-funktiosta saatu nuotti tallennetaan myös lastPitch-muuttujaan, jotta seuraavalla kierroksella voidaan verrata uutta nuottia edellisen kierroksen nuottiin. Kun sähkökitarasta soitetaan sävel, kestää hetken, ennen kuin äänentaajuus tasoittuu. Vertailun avulla vähennetään aineistoesimerkkien virheellistä luokittelua.

Kun kohdalle tulee nuotti, joka on sama kuin edellisen kierroksen nuotti, luodaan Webcam-luokan capture-funktion avulla sen hetkisestä kuvasta tensori. Capture-funktiossa tf.fromPixels-funktiolle annetaan kuva web-kamerasta, ja se tuottaa pikseleistä tensorin. Seuraavaksi kuvatensori rajataan cropImage-funktiossa neliön muotoiseksi MobileNetin tarpeisiin. Lopuksi tensoriin lisätään uusi ulottuvuus, jotta se vastaa MobileNetin vaatimia pakattuja syötteitä (batched inputs), ja tensori normalisoidaan 1:n ja -1:n väliselle alueelle.

Seuraavaksi capture-funktiossa valmiiksi muodostettu tensori annetaan parametrina alussa katkaistusta MobileNet-mallista luodun mallin predict-funktiolle. Predict-funktion palauttavat sisäiset aktivoinnit ja niitä vastaava nuotti annetaan Dataset-luokan addExample-funktiolle. AddExample-funktio muodostaa nuotista tensorin one-hot-koodaustekniikalla ja tallentaa sen ja kuvasta saadun tensorin erillisiin muuttujiin. Sisäiset aktivoinnit edustavat aineiston piirrettä ja nuotti merkkiä (label). Kun addExample-funktioon syötetään uutta aineistoa, se ketjuttaa uuden tensorin vanhan perään. Näin syntyy lopulta kaksi tensoria, jotka sisältävät kaikki kerättyjen kuvien sisäiset aktivoinnit ja niitä vastaavat merkit.

Kun käyttäjä klikkaa "Stop Collecting Data" -nappia, muuttuu collectingData-muuttujan arvoksi "false" ja äänen prosessointi loppuu. Prosessointi lopetetaan irrottamalla äänilähde analysointisolmusta, analysointisolmu Javascript-solmusta ja Javascript-solmu audiokontekstin äänipäätteestä.

4.4 Mallin koulutus

Kun aineistoa on kerätty tarpeeksi, on vuorossa mallin koulutus. Koulutus aktivoidaan painamalla "Train"-nappia, joka käynnistää train-funktion. Ensin funktio tarkistaa, ettei Dataset-luokka ole tyhjä, minkä jälkeen luodaan varsinainen koulutettava malli ja sen tasot.

Mallin luominen

Malli luodaan TensorFlowjs:n tf.sequential-funktion avulla. Tf.sequential luo mallin, jossa yhden tason tuottamat syötteet tai arvot ovat seuraavan tason sisääntulosyötteitä. Malli on siis joukko toisiinsa liittyneitä tasoja, eikä se sisällä haaroja tai tasojen ohituksia (48). Malliin lisätään kolme tasoa, joista ensimmäinen tasoittaa syötteen vektoriksi, jotta sitä voidaan käyttää myöhemmin luotavissa tiheätasoissa (dense layers). Vaikka tämä ensimmäinen taso on teknisesti taso, se ainoastaan uudelleenkäsittelee malliin tulevan syötteen eikä osallistu varsinaiseen mallin koulutukseen.

Ensimmäinen varsinainen koulutustaso on tiheätaso. Tiheätaso on taso, jossa kaikki tason neuronit on täysin yhdistetty (fully connected) edellisen tason neuroneihin. Toisin

sanoen jokainen edellisen tason neuronin lähettää painotuksensa jokaiseen tiheäntason neuronin.

Mallin tasojen konfigurointi

Ensimmäisen tason neuronien määrää käyttäjä voi vaihtaa "hidden units" -painikkeen avulla. Valittu määrä annetaan mallin konstruktorille argumenttina "units: ui.getDenseUnits()", jossa ui.getDenseUnits() on käyttäjän valitsema neuronien määrä. Toisena argumenttina annetaan "activation: 'relu'", joka määrittää tason aktivointi funktion. Kolmantena argumenttina tasolle annetaan "kernelInitializer: 'varianceScaling'" ja neljäntenä "useBias: true", joka kertoo tasolle, että aloituspistettä käytetään. Esimekkikoodissa 1 on ensimmäisen tason konfigurointiobjekti.

```
tf.layers.dense({
    units: ui.getDenseUnits(),
    activation: 'relu',
    kernelInitializer: 'varianceScaling',
    useBias: true
});
```

Esimerkkikoodi 1. Mallin ensimmäisen tason konfigurointiobjekti.

Toinen taso on myös tiheäntaso. Sen aktivointi-funktio on "Softmax", joka on tuottaa todennäköisyydet jokaiselle mahdolliselle luokalle. Aktivointi-funktion antamien todennäköisyyksien summa on 1. Tason neuronien määrä vastaa koulutettavien luokkien määrää, ja näin jokaiselle luokalle saadaan todennäköisyys. Kernelinitializer on sama kuin ensimmäisessä tasossa, ja aloituspistettä ei käytetä.

Mallin konfigurointi

Kun malliin on lisätty konfiguroidut tasot, on vuorossa mallin konfigurointi. Mallin konfiguroinnissa määritellään mallin optimointi- ja häviö-funktio. Optimointi-funktioksi osoitetaan "Adam" ja häviö-funktioksi "categoricalCrossentropy". CategoricalCrossentropy-funktiota käytetään kategoriseen luokitteluun. Se laskee virheen ennustetun luokkien todennäköisyyden jakauman ja todellisen merkin (label) välillä.

Seuraavaksi pakkauksen koko (batch size) lasketaan käyttäjän antaman syötteen pohjalta. Käyttäjä valitsee arvon "batch size" -vetovalikosta, joka toimii kertoimena yhtälössä, jolla pakkauksen koko päätellään. Käyttäjän valitsemalla arvolla kerrotaan `dataset.xs.shape[0]` eli kuvatensorin ensimmäisen ulottuvuuden koko. Näin pakkauskoosta saadaan dynaaminen.

Lopuksi malli koulutetaan fit-funktiolla, jolle annetaan parametrina kuvatensori ja merkkitensori sekä objekti. Objekti sisältää pakkauskoon, käyttäjän syötteestä päätellyn arvon (epochs), joka määrittää, kuinka monta kertaa koko aineisto iteroidaan läpi, ja callback-funktion, joka tulostaa häviön.

4.5 Mallin avulla ennustaminen

Kun malli on koulutettu, se on valmis ennustamaan web-kameran syötettä. Mallin käyttö aktivoidaan painamalla "predict"-nappia. Tämä käynnistää asynkronisen predict-funktion. Aluksi predict-funktiossa luodaan oskillaattori-solmu audiokontekstia varten, joka tuottaa ennustuksen tuottaman taajuuden ja ohjaa sen audiokontekstin ulosmenoon (destination). Seuraavaksi luodaan while-silmukka, jossa web-kameran syötteen pohjalta ennustaminen tapahtuu.

Aluksi while-silmukassa suoritetaan predictedClass-funktio. Funktio suoritetaan synkronisesti asynkronisen funktion sisällä await-etuliitteen määräämänä. PredictedClass-funktiossa suoritetaan ensin Webcam-luokan capture-funktio, joka palauttaa kuvakaappauksesta muodostetun tensorin. Seuraavaksi tensori viedään katkaistun MobileNet-mallin läpi, josta saadaan sisäiset aktivoinnit varsinaista mallia varten. Lopuksi sisäiset aktivoinnit ajetaan varsinaisen mallin läpi ja näin saadaan kuvasta ennustus. Ennusteesta kaivetaan nuotin indeksi, jolla on suurin todennäköisyys, ja se palautetaan funktiosta. Funktiota kutsuttaessa on siihen ketjutettu data-funktio, joka asynkronisesti lataa ennustuksen arvot luettavaan muotoon predictedClass-funktiosta saadusta tensorista.

Kun ennuste on saatu, poistetaan predictedClass-funktion varaamat muistipaikat TensorFlow.js:n dispose-funktion avulla. Predict-funktion lopussa saadun ennusteen sisältämä taajuus asetetaan oskillaattori-solmun taajuudeksi ja HTML-elementtiin

tulostetaan sisällöksi kaikki ennusteen tiedot (nuotti, taajuus ja merkki (label)). Lopuksi kutsutaan synkronisesti `await-etuliitteellä` `tf.nextFrame`-funktiota, joka ottaa käsittelyyn seuraavan kuvan web-kamerasyötteestä.

4.6 Sovelluksen haasteet

Sovelluksessa ilmeni ongelma, jota en saanut selvitettyä kehitykseen varatun ajan puitteissa. Kun kuvista muodostettuja sisäisiä aktivointeja on ketjutettu yhteen noin 300-400, sovellus kaatuu. Virheilmoitus internetselaimen konsolissa kertoo, että pikselivarjostinta ei pystytty laatimaan, eikä WebGL pystynyt parsimaan riviä pikselivarjostimessa, joka virheen on tuottanut. Tämän vuoksi ei riittävän aineistomäärän kerääminen onnistunut eikä mallin suunniteltua optimointia voinut järkevästi toteuttaa.

Toinen haaste liittyi sähkökitaran tuottamaan ääneen. Kun kitarasta soittaa sävelen, soi sävelestä samalla useita eri harmonioita. Tämän vuoksi autokorrelointi-funktiolla on vaikeuksia löytää aluksi oikea harmonia ja sävelen virheluokittelun riski kasvaa. Huomasin, että kun soitti sävelen sähkökitarasta, kesti aluksi vajaan sekunnin, ennen kuin taajuus tasoittui oikeaksi. Lisäsin koodiin säännön, joka kerää aineistoa vasta, kun taajuus on tasoittunut eli kun se on ollut sama esimerkiksi kolmen ääninäytteen ajan.

5 Yhteenveto

Insinööriyöraportissa perehdyttiin koneoppimisen perusteisiin ohjatun oppimisen näkökulmasta. Koneoppiminen hyvin monitasoinen tekniikka, ja sen kaikkien osa-alueiden läpikäyminen on valtava tehtävä, minkä vuoksi insinööriyöhön otettiin mukaan tärkeimmät peruskäsitteet ja tekniikat, jotka antavat riittävän pohjan insinööriyönä tehdyn neuroverkkosovelluksen toiminnan ymmärtämiseksi.

Aineiston keräämisessä ilmenneen ongelman vuoksi suunniteltu koneoppimismallin optimointi jäi toteuttamatta insinööriyönä toteutetussa neuroverkkosovelluksessa. Sovellus osoittautui kuitenkin toimivaksi pienilläkin aineistomäärillä koulutettuna. Malli oppi ennustamaan nuotteja luotettavasti yhdeltä kieleltä, kun opetettujen nuottien välissä oli vähintään yksi nauha. Jos koulutettavia nuotteja oli tiheämmässä, mallin tarkkuus

alkoi kärsiä. Malli ei oppinut luotettavasti tunnistamaan eri kieliltä soitettuja nuotteja, ellei koulutettavien kielten välimatka ollut riittävä.

Mallin tarkkuutta pystyy parantamaan isommalla koulutusaineistolla. Toinen tapa lisätä tarkkuutta on koulutettavien kuvien esiprosessointi. Esiprosessoinnissa voitaisiin rajata kuvasta muotojen tunnistusalgoritmien avulla vain olennainen osa eli sähkökitaran tapauksessa otelauta ja sormet. Ylimääräinen kuva-alue aineistossa on ”kohinaa” mallille, ja se häiritsee mallin koulutusta.

Sovelluksen avulla voidaan assosoida minkä tahansa äänilähteen tuottamat taajuudet kuviin. Sovellusta jatkokehittämällä voidaan luoda esimerkiksi keholla ohjattavia pelejä tai sovellus, jonka avulla voi opetella soittamaan eri soittimia. Kitaran soiton opetussovellus voisi esimerkiksi opastaa oppilasta havaitessaan oppilaan sormituksissa virheitä tai optimoitavaa. Teoriassa, jos mallin kouluttaisi tarpeeksi suurella ja laaja-alaisella aineistolla, voisi mallia käyttää jollain tasolla videon ääniraidan restaurointiin tapauksessa, jossa videokuvan ääniraita olisi vahingoittunut.

Lähteet

- 1 Machine Learning. 2018. Verkkoaineisto. Machine Learning Glossary. Google. <https://developers.google.com/machine-learning/glossary/#machine_learning>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 2 Yolo:Real-Time Object Detection. 2018. Verkkoaineisto. Yolo. <<https://pjreddie.com/darknet/yolo/>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 3 Reducing Loss: An Iterative Approach. 2018. Verkkoaineisto. Google. <<https://developers.google.com/machine-learning/crash-course/reducing-loss/an-iterative-approach>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 4 Descending into ML: Training and Loss. 2018. Verkkoaineisto. Google. <<https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 5 Framing: Key ML Terminology. 2018. Verkkoaineisto. Google. <<https://developers.google.com/machine-learning/crash-course/framing/ml-terminology>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 6 Shaikh, Faizan. 2017. Simple Beginner's guide to Reinforcement Learning & its implementation. Verkkoaineisto. Analytics Vidhya. <<https://www.analyticsvidhya.com/blog/2017/01/introduction-to-reinforcement-learning-implementation/>>. 19.1.2017. Luettu 1.11.2018.
- 7 Hu, Jungling. 2016. Reinforcement learning explained. Verkkoaineisto. O'Reilly. <<https://www.oreilly.com/ideas/reinforcement-learning-explained>>. 8.12.2016. Luettu 1.11.2018.
- 8 Simonini, Thomas. 2018. An Introduction to Reinforcement Learning. Verkkoaineisto. FreeCodeCamp. <<https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419>>. 31.3.2018. Luettu 1.11.2018.
- 9 Unsupervised Machine Learning. 2018. Verkkoaineisto. Machine Learning Glossary. Google. <https://developers.google.com/machine-learning/glossary/#unsupervised_machine_learning>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 10 Donges, Niklas. 2018. Transfer Learning. Verkkoaineisto. Towards Data Science. <<https://towardsdatascience.com/transfer-learning-946518f95666>>. 23.4.2018. Luettu 1.11.2018.

- 11 Validation: Another Partition. 2018. Verkkoaineisto. Google.
<<https://developers.google.com/machine-learning/crash-course/validation/another-partition>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 12 Training and Test Sets: Splitting Data. 2018. Verkkoaineisto. Google.
<<https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 13 Representation: Feature Engineering. 2018. Verkkoaineisto. Google.
<<https://developers.google.com/machine-learning/crash-course/representation/feature-engineering>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 14 Representation: Cleaning Data. 2018. Verkkoaineisto. Google.
<<https://developers.google.com/machine-learning/crash-course/representation/cleaning-data>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 15 Regression Model. 2018. Verkkoaineisto. Machine Learning Glossary. Google.
<https://developers.google.com/machine-learning/glossary/#regression_model>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 16 Linear Regression. 2018. Verkkoaineisto. Machine Learning Glossary. Google.
<https://developers.google.com/machine-learning/glossary/#linear_regression>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 17 Gandhi, Rohith. 2018. Introduction to machine Learning Algorithms: Linear Regression. Verkkoaineisto. Towards Data Science.
<<https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a>>. 27.5.2018. Luettu 1.11.2018.
- 18 Reducing Loss: Gradient Descent. 2018. Verkkoaineisto. Google.
<<https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 19 Reducing Loss: Learning Rate. 2018. Verkkoaineisto. Google.
<<https://developers.google.com/machine-learning/crash-course/reducing-loss/learning-rate>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 20 Jordan, Jeremy. 2018. Setting the learning rate of your neural network. Verkkoaineisto. Jeremy Jordan. <<https://www.jeremyjordan.me/nn-learning-rate/>>. 1.3.2018. Luettu 1.11.2018.
- 21 Brownlee Jason. 2017. Difference Between Classification and Regression in Machine Learning. Verkkoaineisto. Machine Learning Mastery.
<<https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>>. 11.12.2017. Luettu 1.11.2018.

- 22 Logistic Regression. 2018. Verkkoaineisto. Machine Learning Glossary. Google. <https://developers.google.com/machine-learning/glossary/#logistic_regression>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 23 Logistic Regression: Model Training. 2018. Verkkoaineisto, Google. <<https://developers.google.com/machine-learning/crash-course/logistic-regression/model-training>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 24 Gupta, Prashant. 2017. Regularization in Machine Learning. Towards Data Science. <<https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a>>. 15.11.2017. Luettu 1.11.2018.
- 25 Generalization: Peril of Overfitting. 2018. Verkkoaineisto. Google. <<https://developers.google.com/machine-learning/crash-course/generalization/peril-of-overfitting>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 26 Underfitting. Verkkoaineisto. DataRobot. <<https://www.datarobot.com/wiki/underfitting/>>. Luettu 1.11.2018.
- 27 Bhande, Anup. 2018. What is underfitting and overfitting in machine learning and how to deal with it. Verkkoaineisto. Medium. <<https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76>>. 11.3.2018. Luettu 1.11.2018.
- 28 Nautiyal, Dewang. Underfitting and Overfitting in Machine Learning. Verkkoaineisto. GeeksForGeeks. <<https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>>. Luettu 1.11.2018.
- 29 Nagpal, Anuja. 2017. L1 and L2 Regularization Methods. Verkkoaineisto. Towards Data Science. <<https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>>. 13.10.2017. Luettu 1.11.2018.
- 30 Regularization for Simplicity: Lambda. 2018. Verkkoaineisto. Google. <<https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/lambda>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 31 Introduction to Neural Networks: Anatomy. 2018. Verkkoaineisto. Google. <<https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/anatomy>>. Päivitetty 1.10.2018. Luettu 1.11.2018.
- 32 Moawad, Assad. 2018. Neural networks and back-propagation explained in a simple way. Verkkoaineisto. Medium. <<https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e>>. 1.2.2018. Luettu 1.11.2018.

- 33 A Beginner's Guide to Convolutional Neural Networks (CNNs). Verkkoaineisto. Skymind. <<https://skymind.ai/wiki/convolutional-network>>. Luettu 1.11.2018.
- 34 Cornelisse, Daphne. 2018. An intuitive guide to Convolutional Neural Networks. Verkkoaineisto. Medium. <<https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050>>. 24.4.2018. Luettu 1.11.2018.
- 35 Karn, Ujjwal. 2016. An Intuitive Explanation of Convolutional Neural Networks. Verkkoaineisto. The Data Science Blog. <<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>>. 11.8.2016. Luettu 1.11.2018.
- 36 Frequently Asked Questions. Verkkoaineisto. TensorFlow.js. <<https://js.tensorflow.org/faq/>>. Luettu 1.11.2018.
- 37 Gordon, Josh & Robinson, Sara. 2018. Introducing TensorFlow.js: Machine Learning in Javascript. Verkkoaineisto. Medium. <<https://medium.com/tensorflow/introducing-tensorflow-js-machine-learning-in-javascript-bf3eab376db>>. 30.3.2018. Luettu 1.11.2018.
- 38 Core Concepts in TensorFlow.js. Verkkoaineisto. TensorFlow.js. <<https://js.tensorflow.org/tutorials/core-concepts.html>>. Luettu 1.11.2018.
- 39 OpenGL ES Overview. Verkkoaineisto. Khronos group. <<https://www.khronos.org/opengles/>>. Luettu 1.11.2018.
- 40 WebGL Fundamentals. Verkkoaineisto. WebGL Fundamentals. <<https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html>>. Luettu 1.11.2018.
- 41 Getting started with WebGL. 2018. Verkkoaineisto. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial/Getting_started_with_WebGL>. Päivitetty 3.7.2018. Luettu 1.11.2018.
- 42 Nakano, Reiichiro; Körner, Christoph; Sasaki, Kai & Bourry, Xavier. 2018. Deep Learning in the Browser. E-kirja. Bleeding Edge Press.
- 43 MobileNet_v1. Verkkoaineisto. Github. <https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md>. Luettu 1.11.2018.
- 44 About ImageNet. Verkkoaineisto. ImageNet. <<http://www.image-net.org/about-overview>>. Luettu 1.11.2018.

- 45 Howard, Andrew & Zhu, Menglong. 2017. MobileNets: Open-Source Models for Efficient On-Device Vision. Verkkoaineisto. Google.
<<https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html>>. 14.6.2017. Luettu 1.11.2018.

- 46 Web audio concepts and usage. 2018. Verkkoaineisto. Mozilla.
<https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API>. Päivitetty 26.10.2018. Luettu 1.11.2018.

- 47 Rectified Linear Units (ReLU) in Deep Learning. Verkkoaineisto. Kaggle.
<<https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning>>. Luettu 1.11.2018.

- 48 Tf.sequential. Verkkoaineisto. TensorFlow.js.
<<https://js.tensorflow.org/api/latest/index.html#sequential>>. Luettu 1.11.2018.