Haaga-Helia
ammattikorkeakoulu Oy

Käsinkirjoitetun numeron tunnistus TensorFlow:n JavaScript rajapinnalla koneoppimista soveltaen

Jaakko Jokinen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2019

Haaga-Helia
ammattikorkeakoulu Oy

Tiivistelmä

12.04.2019

Tekijä(t)
Jokinen Jaakko

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Käsinkirjoitetun numeron tunnistus TensorFlow JavaScript rajapinnalla.

Sivu- ja liitesivu-määrä
7

Tavoitteena oli tehdä sovellus, joka pystyisi valokuvasta tunnistamaan käsinkirjoitetut numerot ja muuttamaan ne digitaaliseen muotoon ja tallentamaan tietokantaan, muun historiallisen havaintodatan jatkoksi.
Työssä sukelletaan melko syvälle koneoppimisen menetelmiin. Erilaisia tekniikoita ja algoritmeja, jotka ovat pakollisia työn onnistumisen kannalta, joudutaan käymään pinnallisesti läpi, sillä muuten työ paisuisi liikaa sen vaatimuksiin nähden. Tämä työ on ennen kaikkea soveltuvuusselvitys teknologian mahdollisuuksista asetetun ongelman ratkaisussa.
Tämä on toiminnallinen opinnäytetyö, jossa tietoperusta käydään läpi niiltä osin, kun empirian toteuttaminen onnistuneesti on pakollista. Tämä tarkoittaa neuroverkkojen perusteiden ja yleisimmin käytettyjen tekniikoiden läpikäyntiä.

Opinnäytetyössä käytän Googlen TensorFlow kirjastoa ja Googlen julkaisemaa lähdekoodia pohjana. Tämän JavaScriptilla kirjoitetun sovelluksen lähdekoodia olen muuttanut niiltä osin, mitä työni tulosten validointi on sitä vaatinut. Käytännössä tämä tarkoittaa sitä, että olen pyrkinyt tuottamaan opetetulle neuroverkolle aineistoa, jota se ei ole ennen käsitellyt.
Opinnäytetyö on aloitettu syyskuussa 2018. Sitä on edistetty joulukuussa 2018, tammikuussa 2019 ja se saatettiin loppuun huhtikuussa 2019.
Työssäni pääsin osaan tavoitteista, mutta ensisijainen tavoite jäi saavuttamatta. Opin neuroverkkojen perusteet ja ymmärrän koneoppimisen mahdollisuuksia paremmin. Ymmärrän myös, ettei näillä resursseilla voida tehdä tavoiteltua sovellusta. Sovelluskehitykseen pitäisi käyttää enemmän aikaa ja olisi hyvä saada ympärille tiimi, jolla on jo kokemusta koneoppimisesta ja koneoppimista käyttävistä sovelluksista.

Asiasanat

koneoppiminen, neuroverkot, ohjelmointi, ohjattu oppiminen, javascript, python, tensorflow

Haaga-Helia
ammattikorkeakoulu Oy

Sisällys

Lyhenteet

ANN	Artificial Neural Network – Keinotekoinen neuroverkko
ReLU	Rectified Linear Unit
FCNN	Fully Connected Neural Network – Täysin kytketty neuroverkko
CNN	Convolutional Neural Network - Konvoluutioneuroverkko
SDNN	Space Displacement Neural Network
Tfjs	TensorFlow.js

1 Johdanto

Opinnäytetyön aihe on koneoppiminen ja sovelluskehitys TensorFlow.js rajapinnalla. TensorFlow.js on keväällä 2018 julkaistu JavaScript versio koneoppimisessa ehkäpä eniten käytetystä TensorFlow Python kirjastosta.

Tavoitteenani on luoda web-sovellus, joka pystyy digitoimaan käsinkirjoitettuja numeroita ja tallentamaan ne tietokantaan desimaaliluku -tietotyyppisenä datana. Käsinkirjoitetut numerot luetaan valokuvasta, joka syötetään html-canvas elementtiin ja siitä edelleen luokittelija-algoritmille. Tätä ennen algoritmi on opetettu tunnistamaan käsinkirjoitettuja numeroita MNIST-tietokannan 60 000 näytteellä käsinkirjoitetuista numeroista.
MNIST-tietokanta on koostettu Yhdysvaltain väestönlaskentaviraston ja high school-opiskelijoiden tuottamista käsinkirjoitetuista numeroista. Tämä aineisto on saatavilla helposti ainakin koneälypioneerin, Yann Lecunin kotisivuilta. (LeCun 2019)

Tämän opinnäyte sai alkunsa eräänä perjantai-iltapäivänä, kun mietimme työkaverini kanssa, minkälaisia sovelluksia voisimme käytännössä rakentaa TensorFlow:n JavaScriptiin pohjautuvalla kirjastolla.

Käsinkirjoitettuja numeroita sisältäviä lomakkeita löytynee varmasti useasta Suomen valtion viraston tai laitoksen arkistosta. Idea projektille tuli tarpeesta digitoida suuria määriä numeraalisia havaintoarvoja sisältäviä lomakkeita. Digitointia on epämielekästä ja epätehokasta tehdä lukemalla lukuja paperista ja syöttämällä niitä järjestelmään manuaalisesti.

2 Koneoppiminen

Tekoäly on tiedon soveltamiskykyä vaativan tehtävän suorittamisen simulointia. Koneoppiminen voidaan perustellusti nähdä keinoälyn ytimessä, muiden keinoälysovellusten, kuten konenäön ja robotiikan mahdollistajana (OCDevel 2019).
Menemättä sen syvemmälle koneen määrittelemisessä, sanottakoon, että kone ei ole luonnostaan älykäs. Ihminen on luonut koneen auttamaan asioissa, joita on vaikea suorittaa toistuvasti samalla tavalla. Kone on hyvä toistuvassa ja mekaanisessa työssä, jossa tulisi helposti inhimillisiä virheitä. Perinteisesti kone ei ole kyennyt oppimaan tai käyttämään älyllisesti dataa, jota se tavalla tai toisella käsittelee (Mohammed & Muhammad & Eihab 2017).

Koneoppimista voidaan luonnehtia tietokoneen kyvyksi oppia kokemuksista ja yleistää opittua ratkaisua yli oppimisessa käytetyn opetusaineiston, muihinkin ongelmiin (Akhter 2017).

Mohammed ym. (2017) mukaan Koneoppiminen jakautuu neljään paradigmaan.

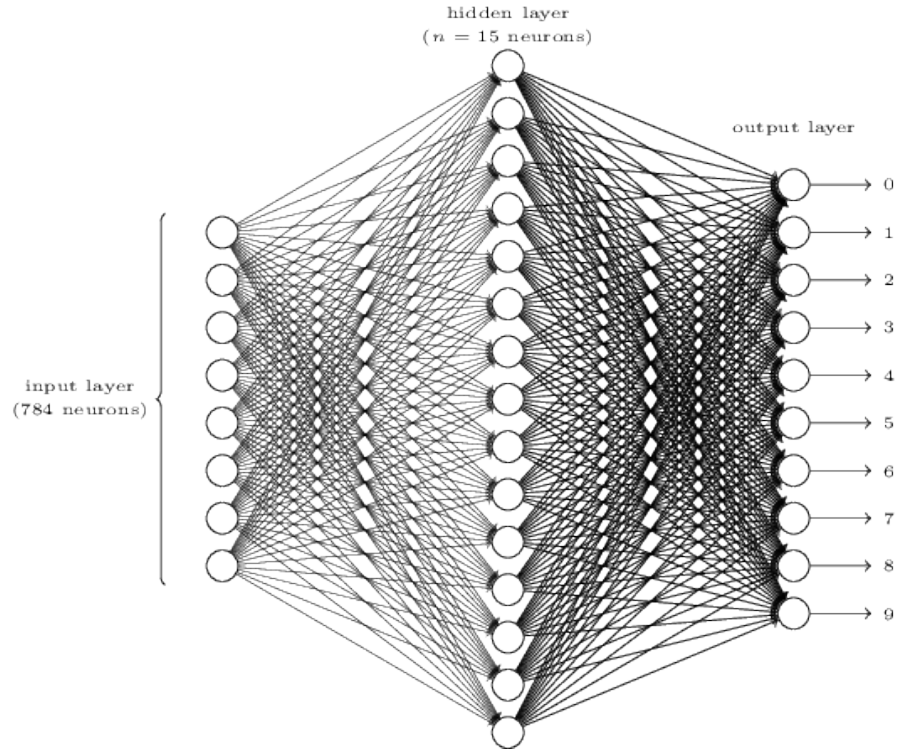
- Ohjattu oppiminen, jossa käytetään luokiteltua opetusaineistoa.
- Ohjaamaton oppiminen, jossa opetusaineisto on luokittelematonta.
- Osittain ohjattu oppiminen, jossa käytetään sekoitusta luokitellusta ja luokittelemattomasta aineistosta.
- Vahvistusoppiminen, jossa dataa ei käytetä opetusaineistona.

2.1 Koneoppimisen sovelluksia

Koneoppimisen sovellukset ovat tehokkaita ratkaisemaan kompleksisia ongelmia, joita olisi tehotonta tai mahdotonta ratkaista deterministisellä, asteittain suoritettavalla laskennalla. Tyypillisiä koneoppimisen sovelluksia ovat roskapostin havaitseminen, osakekurssien muutosten ennustaminen, sää ennusteet ja konenäön eri sovellukset. Näissä sovelluksissa käytetään eri koneoppimisen algoritmeja, mutta lähes kaikille niistä yhteistä on aikaisemmasta esimerkistä, eli luokitellusta datasta oppiminen. Koneoppimisen alueella tässä on kyse ohjatusta oppimisesta, joka on koneoppimisen sovellusalueista käytetyin ja johon tämä opinnäytetyö enimmälti keskittyy (Akhter 2017).

3 Keinotekoinen neuroverkko

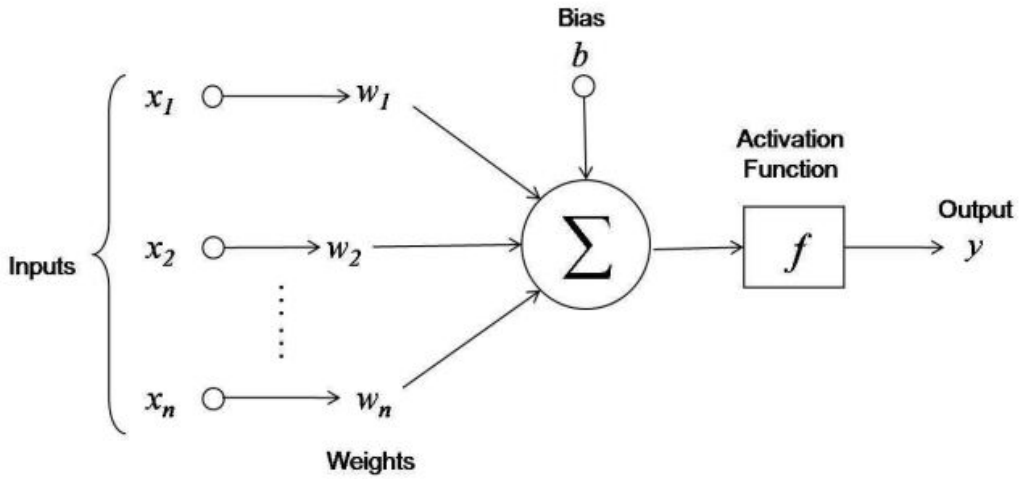
Keinotekoinen neuroverkko (Artificial Neural Network - ANN) on malli, joka on syntynyt biologisen hermoverkon inspiroimana (Mohammed ym. 2017).
Se on tavallaan representaatio ihmisen aivoista. Sen rakenne ja tapa käsitellä sen läpi kulkevaa dataa muistuttaa ihmisen tai eläimen biologista keskushermostoa.



Kuva 1. Täysin kytketty neuroverkko numeroiden 0-9 luokitteluun. (Neural Networks and Deep Learning, Chapter 1)

3.1 Neuroni

Neuroverkko koostuu yksittäisistä neuroneista eli aktivointifunktioista, jotka ovat arkkitehtuurin määrittelemällä tavalla yhteydessä toisiinsa. Nämä yhteydet muuttuvat jatkuvasti neuroverkon oppimisen seurauksena. Tässä opinnäytteessä esiintyvät neuronityypit perseptroni, sigmoidi, tanh ja ReLu. On olemassa muitakin neuronityyppejä, ja niiden vertailukehittäminen parantaa neuroverkkojen suorituskykyä ja tarkkuutta (Ramachandran & Barret & Quoc 2017).



Kuva 1. Neuronin mekanismi. (Medium 2019)

Neuroni voidaan ajatella matemaattisena funktiona, mikä se pohjimmiltaan onkin. Se voidaan myös kirjoittaa millä tahansa ohjelmointikielellä, jolla voi kirjoittaa matemaattisia funktioita (O'Reilly Media 2017a).

3.2 Perseptroni

Yksittäinen perseptroni-neuroni on yksinkertaisin keinotekoinen neuroverkko. Sen avulla voidaan suhteellisen helposti ymmärtää neuroverkon perusmekanismi. Kun neuroverkkoa opetetaan, perseptroni voi saada useita syötteitä, joihin perustuen se tuottaa ulostuloarvon. Perseptronin sisään tulevilla syötteillä on ennalta määritellyt painoarvot, mikä käytännössä tarkoittaa sitä, että jollakin syötteellä on enemmän vaikutusta ulostuloarvoon, jollakin vähemmän. Perseptroni on luonteeltaan binäärinen. Esimerkki sen toiminnasta on tarkkailla, palautuuko sen ulostulonarvona luku 0 vai 1. Perseptroniin johtaa kolme syötettä, joista jokaisella on oma painoarvonsa (englanniksi weight). Perseptroniin on määritelty raja-arvo (bias), jonka ylittyminen pistää perseptronin tuottamaan ulostulona arvon 1. Jos raja-arvo ei ylity, perseptroni palauttaa 0 (O'Reilly Media 2017a).

3.3 Sigmoidi

Toinen yleinen neuronityyppi on sigmoidi funktio, jonka avulla voidaan hienovaraisesti säätää neuroverkon laskentaa. Sigmoidin ulostuloarvona on liukuluku lukujen 0 ja 1 väliltä.

Nykyään perseptroni ja sigmoidi funktiot ovat vanhentuneet pois käytöstä, mutta niiden avulla voi hyvin opetella neuroverkkojen alkeita ja havainnollistaa uusien aktivointifunktioiden ylivertaisuutta. Perseptroni on liian epätarkka yksinkertaisen binäärisen rakenteensa takia ja sigmoidi on hidas verrattuna tanh- eli hyperboliseen tangent funktioon (O'Reilly Media 2017a).

3.4 *Tanh* ja ReLU

Nykypäivänä käytettävät aktivointifunktiot eivät muuta neuronin toiminnan perusmekanismia. Ainoastaan niiden tuottama ulostuloarvo muuttuu tarkemmaksi, joka mahdollistaa neuroverkon hienovaraisemman laskennan, joka puolestaan mahdollistaa paremmat tulokset oppimisessa.
Tanh funktion ulostuloarvo on liukuluku lukujen -1, 0, 1 väliltä (O'Reilly Media 2017a).
ReLU tuottaa jonkun luvun 0:sta ylöspäin. Aktivointifunktioista nimenomaan ReLU on osoittautunut tehokkaaksi ja yleisesti käytetyimmäksi. (Krizhevsky & Sutskever & Hinton 2012).

4 Oppiminen

Seuraavassa on kuvailtu neuroverkon oppimisen kannalta oleellisia tekniikoita.

4.1 Ristientropia

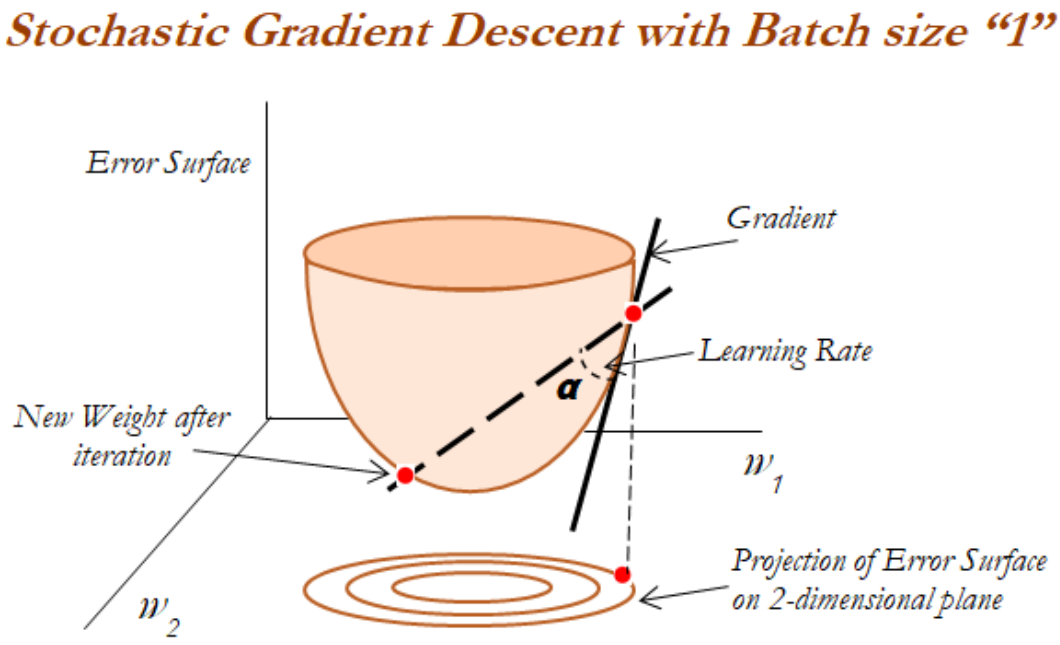
Ristientropia-funktiolla voidaan laskea erotus neuronin tuottaman arvon ja odotetun arvon välillä. JavaScript esimerkissä funktio saa parametrin y, joka on funktion odotettu paluu-arvo, ja parametrin a, joka on ollut funktion toteutunut paluuarvo, kun opetusaineisto on ajettu neuroverkon läpi. Ristoentropian johdannaisella lasketaan arvo, joka edustaa koko neuroverkon epätarkkuutta (O'Reilly Media 2017a).

```
function crossEntropy(y, a) {
    return -1 * (y * Math.log(a) + (1 - y) * Math.log(1 - a))
}


crossEntropy(0, 0.3) //0.35667494393873245
```

4.2 SGD

SGD (Stochastic Gradient Descent) on tekniikka, jonka avulla voidaan löytää optimaaliset painot ja raja-arvot, joilla saavutetaan pienin mahdollinen virhemarginaali. Tässä laskennassa käytetään ristientropian derivaattaa. Opetusaineisto jaetaan osiin sen sijaan että se ajettaisiin kerralla kokonaan. Tällä tavalla neuroverkko ehtii korjaamaan itseään jokaisen eräajon jälkeen (O'Reilly Media 2017a).



Kuva 2. SDG menetelmä koneoppimisessa. (O'Reilly Media, Inc 2017b)

4.3 Vastavirta-algoritmi

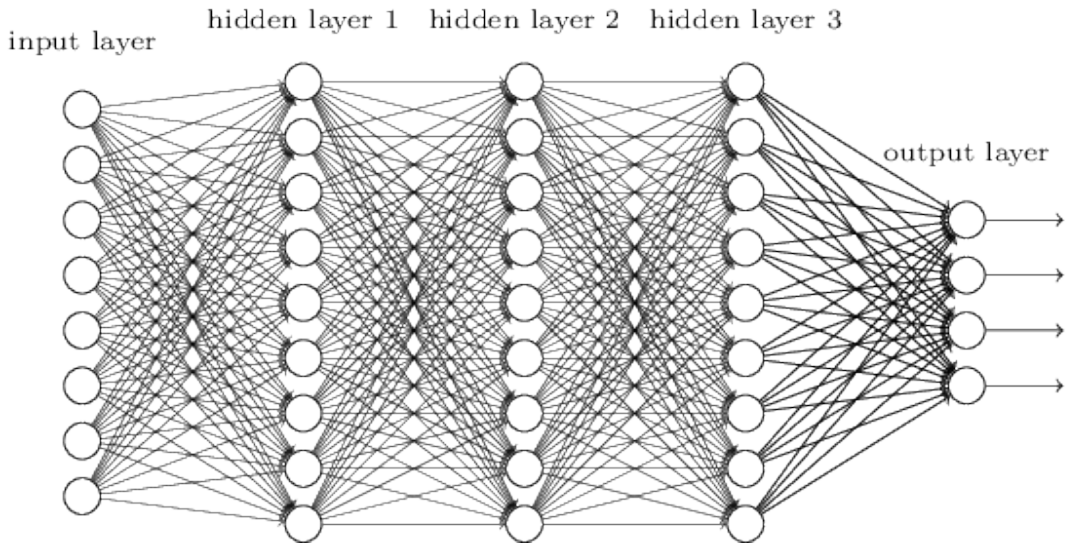
Kun neuroverkko luokittelee aineiston väärin, voidaan käyttää vastavirta-algoritmia. Vastavirta laskee uudelleen gradientin, kunnes ollaan riittävän lähellä pienintä virhemarginaalia. Vastavirta tekniikan ideaa voi sanoa yksinkertaiseksi, mutta siihen liittyvä matematiikkaa ei. Korkealla tasolla kuvailtuna algoritmi käyttää differentiaalilaskentaa läpikäydessään neuroverkon ulostulokerroksesta sisääntulokerrokseen, eli lopusta alkuun (Nilsson 2010 s.508).

4.4 Ylisovitus

Ylisovitettu malli pystyy luokittelemaan ainoastaan täysin samankaltaista aineistoa, millä sitä on opetettu. Se ei pysty luokittelemaan oikein, jos sille syötetään ennen näkemätöntä aineistoa. Yksi ratkaisu tähän ongelmaan on lisätä alkuperäiseen opetusaineistoon uusia ominaisuuksia. Esimerkiksi kuvaan voidaan lisätä kohinaa tai sitä voidaan liikutella kuvapinnalla eri kohtiin, jolloin malli ymmärtää moniulotteisemmin käsittelemäänsä aineistoa (O'Reilly Media 2017a).

4.5 Syväoppiminen

Kun neuroverkkoarkkitehtuurissa on sisääntulo- ja ulostulokerroksen välissä vähintään 3 piilokerrosta, voidaan puhua syväoppimisesta. Tästä eteenpäin ei ole enää mahdollista tietää tarkasti, mitä neuroverkon sisällä tapahtuu, koska neuroverkossa liikkuvia painoja eli parametreja saattaa olla miljoonia (O'Reilly Media 2017a).



Kuva 3. Syväoppiva keinotekoinen neuroverkko. (Neural Networks and Deep Learning, Chapter 6)

5 Neuroverkon kerrokset

Seuraavassa on kuvailtu tekniikoita ja funktioita, joita yhdistelemällä voidaan muodostaa neuroverkko kuvantunnistamiseen.

5.1 Täysin kytketty kerros

Täysin kytketty kerros (Fully Connected) on kytketty kaikkiin sitä edeltäviin neuroneihin. FCNN parantaa siihen yhdistettyjen neuroverkkojen tarkkuutta, mistä johtuen sitä yleensä käytetään viimeisenä piilokerroksena juuri ennen ulostulokerrosta (O'Reilly Media 2017a).

5.2 Softmax-kerros

Softmax-funktiolla lasketaan ulostulokerroksen neuroneihin päätyvistä sisääntuloista yksi, joka todennäköisimmin edustaa sitä luokkaa, mitä kyseisen ulostulokerroksen neuroni edustaa. Jokainen ulostulokerroksen neuroni edustaa yhtä mahdollista luokkaa, johon aineisto halutaan luokitella (Bridle 1990).

5.3 Maxpool-kerros

Kuva jaetaan matriisiin, joista valitaan kerralla tarkasteltavaksi pienempi osa, vaikkapa 2 x 2 pikselin alue. Tätä tarkasteltavaa aluetta kutsutaan kerneliksi. Kernelin alueella on siis 4 pikseliä, joista voidaan valita tummin tai vaalein tai mikä vaan pikseli jollain perusteella. Valittu pikseli edustaa koko 4:n pikselin joukkoa. Sen jälkeen kernel liikkuu seuraavan 2 x 2 pikselialueelle ja valitsee siitä yhden ja niin edelleen. Kernel aloittaa kuvapinnan vasemmasta yläkulmasta. (Ludwig 2004).

5.4 Konvoluutiokerros

Konvoluutiokerroksen neuronit tunnistavat erilaisia kaavamaisia rakenteita, kuten kulmia, kaaria tai viivoja. Konvoluutiokerroksen eri neuronit etsivät läpikäytävästä aineistosta eri rakenteita. Kun neuroni havaitsee aineistosta kaavan, jota se etsii, se *aktivoituu* ja tuottaa sen mukaisen ulostuloarvon (LeCun & Boser & Denker & Henderson & Howard & Hubbard & Jackel 1989).

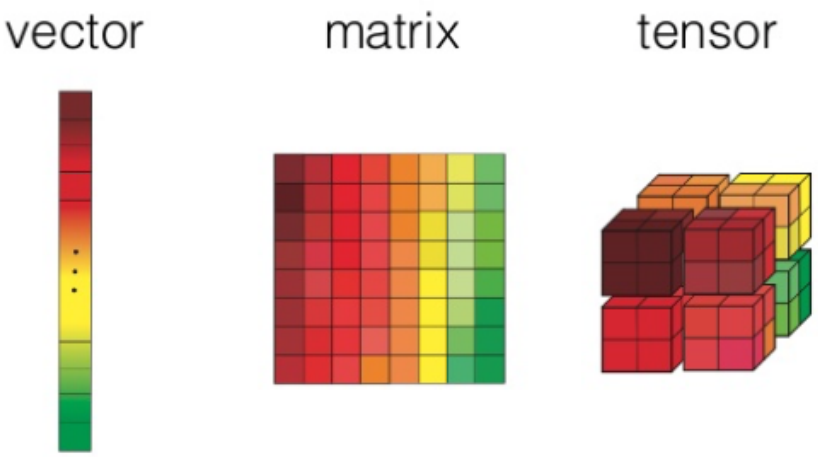
6 TensorFlow

TensorFlow on Google Brain -tiimin kehittämä kirjasto, joka kattaa laajan valikoiman koneoppimisessa käytettäviä funktioita ja tietorakenteita. Kaikki tässä opinnäytteessä esitellyt funktiot ja neuroverkkoarkkitehtuurit on toteutettavissa TensorFlow -rajapinnalla.

6.1 Tensori

”Tensorit ovat TensorFlow.js:n ydintietorakenteita. Ne pitävät sisällään vektorit ja matriisit sekä niiden potentiaalisesti korkeammat ulottuvuudet” (TensorFlow 2019).

TensorFlow rajapinnassa on 0 – 6 tensorityyppiä, joiden järjestysnumero vastaa tensorin ulottuvuuksien lukumäärää. Tämän lopputyön empiirisessä osuudessa käytetään 2-ulotteisia tensoreita, mikä on helppo ajatella ja visualisoida 2-ulotteisena matriisina (TensorFlow 2019).



Kuva 5. Vektori, matriisi ja tensori. (SlideShare 2019)

6.2 JavaScript API

TensorFlow on alun perin Python -ohjelmointikielelle kehitetty kirjasto. Keväällä 2018 Google julkaisi kirjaston myös JavaScriptille.
Empiirisessä osassa käytän tfjs -rajapinnasta seuraavia funktioita. Nämä funktiot perustuvat neuroverkon luomisen ja käsittelemisen tekniikkoihin, mitä esittelin kohdissa 3-4.

Funktiokutsu	**Selite**
tf.tensor2d()	luo kaksiulotteisen tensorin
tf.sequential()	luo instanssin tf.Sequential luokasta
tf.layers.conv2d()	luo layers-rajapinnan kautta 2-ulotteisen konvoluutiokerroksen
tf.layers.maxPooling2d()	luo maxpool-kerroksen
tf.layers.flatten()	muuttaa moniulotteisen tensorin yksiulotteiseksi
tf.layers.dense()	luo täysin kytketyn kerroksen
tf.train.adam()	luo adam optimointi algoritmin, joka voidaan liittää malliin.

Taulukko 1. Empiirisen osuuden TensorFlow funktiot. (TensorFlow 2019)

7 Empiirinen osa

7.1 Tavoitteet

Tämän opinnäytteen tavoite on tehdä soveltuvuusselvitys teknologisesta sovelluksesta, joka pystyy koneoppimisen keinoin luokittelemaan digitaaliseen muotoon käännettyjä, käsinkirjoitettuja numeroita. Tavoitteena on saada kehitettyä proof-of-concept mobiili- tai progressiivisesta verkkosovelluksesta, jota voisi ajaa android-puhelimella, ja joka pystyisi puhelimella otetun valokuvan perusteella luokittelemaan kuvassa näkyviä numeroita digitaalisiksi kokonaisluvuiksi.
Tavoitteena on löytää optimaaliset konfiguraatiot neuroverkkomallille, jotta se pystyy tarkasti ja nopeasti tunnistamaan käsinkirjoitetun numeron.

7.2 Suunnitelma

Opinnäytetyön perustana on JavaScript:lla toteutettu TensorFlow -sovellus, joka käyttää MNIST-tietokannan dataa neuroverkon opetusaineistona. TensorFlow:n JavaScript -pohjainen versio mahdollistaa koneoppimisen mallien kehittämisen ja opetusaineistojen prosessoimisen asiakaspään resursseilla. Aikaisemmin tämä on ollut mahdollista lähinnä palvelinpään resursseilla ja yleisesti käytetty kieli on ollut Python. Näin ollen TensorFlow.js:n ilmestyttyä koneoppimisen perusteiden opetteleminen muuttui vaikeasta ja vastenmielisestä vähintäänkin saavutettavaksi. Tätä opinnäytetyötä ei siis olisi tehty millään muulla teknologisella ratkaisulla.
Toimiva sovellus, joka pystyy käsittelemään ja luokittelemaan MNIST-tietokannan aineistoa, on julkaistu TensorFlow:n www-sivuilla. Suunnitelmissa on muokata sovelluksen lähdekoodia siten, että MNIST-tietokannan aineiston sijaan sille syötetään itse tuotettua ja esikäsiteltyä testiaineistoa. Sovelluksen luoma neuroverkko edelleen koulutetaan MNIST-tietokannan aineistolla, mutta varsinainen validaatio suoritetaan valokuvasta tuotetulla png-kuvalla.

7.3 Alkuvalmistelut

Sovellus vaatii toimiakseen ainoastaan selaimen. Mielellään Chrome selaimen, koska esimerkiksi Safarilla oppiminen tuntui toivottoman hitaalta. Sovelluksen pohjana käytän lähdekoodia TensorFlow:n sivuilta löytyvältä pikakurssilta ”Recognize handwritten digits with CNNs”. Kyseessä on klassinen koneoppimisen avulla ratkaistava ongelma, josta löytyy myös runsaasti esimerkkejä internetistä, mutta ei JavaScript kielelle.
Ennen testiaineiston validointia lähdekoodia tulee muuttaa siten, ettei siinä käytetä alkuperäisen sovelluksen testidataa. Tämä vaatii datan esikäsittelyä muotoon, jossa se voidaan syöttää neuroverkolle. Haluan kuitenkin säilyttää neuroverkon opetusaineiston, ettei

koko sovellusta tarvitsisi koodata alusta alkaen. Testidataksi olen tuottanut itse käsinkirjoitettuja numeroita, jotka yritän saada tunnistettua valmiiksi koulutetun mallin avulla.

```
b_thesis/
├── data.js
├── index.html
├── script.js
└── styles.css
```

Sovelluksen rakenne tree-ohjelmalla havainnollistettuna.

7.4 Neuroverkon opettaminen JavaScript kielellä

Sovellus koostuu kahdesta JavaScript-tiedostosta ja kahdesta kirjastosta, jotka ovat TensorFlow ja tfjs-vis. JavaScript sovelluksen asynkronisen luonteen vuoksi sovelluksen toiminta on hankala dokumentoida komento – tulos menetelmällä. Ongelma on ratkaistu käyttämällä JavaScriptin async – await syntaksia, voidaan varmistua siitä, että edellisen funktion komputaatio on valmistunut, ennen kun seuraava aloitetaan.

```
async function run() {
    const data = new MnistData();
    await data.load();
    await showExamples(data);

    const model = getModel();
    tfvis.show.modelSummary({ name: 'Model Architecture' }, model);

    await train(model, data);
    await model.save('localstorage://my-model-1');

    await showAccuracy(model, data);
    await showConfusion(model, data);
}
```

Sovelluksen ajon aikana suoritetaan ylläolevat funktiot synkronisoidusti, eli siinä järjestyksessä missä ne run() funktion sisällä esitetään. MnistData-luokka on vastuussa MNIST aineiston muuttamisesta tensoreiksi. Näitä tensoreita käytetään script.js tiedostossa mallin (koodissa model) opettamiseen. Malli luodaan getModel() funktion sisällä, jossa siihen lisätään kerrokset, jotka yhdessä muodostavat neuroverkon.

```
function getModel() {
    // Sequential-mallissa edellisen kerroksen ulostuloarvot
    // ovat seuraavan kerroksen sisääntuloarvoja.
    // Malli on yksinkertainen pino erilaisia kerroksia.
    const model = tf.sequential();

    const IMAGE_WIDTH = 28;
    const IMAGE_HEIGHT = 28;
    const IMAGE_CHANNELS = 1;

    //  Neuroverkon ensimmäinen kerros on konvoluutio
    //  jolle annetaan parametrit. Aktivointifunktiona
    //  ReLU.
    model.add(tf.layers.conv2d({
        inputShape: [IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS],
        kernelSize: 5,
        filters: 8,
        strides: 1,
        activation: 'relu',
        kernelInitializer: 'varianceScaling'
    }));

    // Maxpool-kerros laskee 2x2 alueen pikseleistä korkeimman
    // ja asettaa sen edustamaan koko 4 pikselin joukkoa.
    model.add(tf.layers.maxPooling2d({ poolSize: [2, 2], strides: [2, 2] }));

    // Luodaan toinen konvoluutiokerros suuremmalla määrällä suotimia.
    model.add(tf.layers.conv2d({
        kernelSize: 5,
        filters: 16,
        strides: 1,
        activation: 'relu',
        kernelInitializer: 'varianceScaling'
    }));
    model.add(tf.layers.maxPooling2d({ poolSize: [2, 2], strides: [2, 2] }));

    // Kaksiulotteinen ulostulodata muutetaan yksiulotteiseksi vektoriksi.
    model.add(tf.layers.flatten());

    // Viimeinen kerros on täysin kytketty kerros ja siinä on 10 neuronia,
```

```
    // yksi jokaista luokaa kohti (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
    const NUM_OUTPUT_CLASSES = 10;
    model.add(tf.layers.dense({
        units: NUM_OUTPUT_CLASSES,
        kernelInitializer: 'varianceScaling',
        activation: 'softmax'
    }));

    // Optimointiin käytetään adam() funktiota, joka asetetaan
    // kääntämisen yhteydessä mallille parametriksi. Samalla
    // virhefunktio.
    const optimizer = tf.train.adam();
    model.compile({
        optimizer: optimizer,
        loss: 'categoricalCrossentropy',
        metrics: ['accuracy'],
    });

    return model;
}
```

MNIST-tietokannan käsinkirjoitetut numerot on tiivistetty yhdeksi kuvaksi (image sprite). Tämä noin 10 megatavun kokoinen kuva ei ole sellaisenaan ihmisluettava, joten ennen opetusta ja validaatiota se pitää esikäsitellä array-tietotyyppiseen muotoon. Siinä muodossa siitä voidaan piirtää html-canvas elementille ihmisluettavia numeroita ja sen jälkeen syöttää neuroverkolle luokittelua varten.

```
    const BATCH_SIZE = 512;
    const TRAIN_DATA_SIZE = 5000;
    const TEST_DATA_SIZE = 1000;

    const [trainXs, trainYs] = tf.tidy(() => {
        const d = data.nextTrainBatch(TRAIN_DATA_SIZE);
        return [
            d.xs.reshape([TRAIN_DATA_SIZE, 28, 28, 1]),
            d.labels
        ];
    });
```

Opetusaineistona käytetään 5000 kuvan otantaa, ja ne syötetään neuroverkkoon 512 kappaleen erissä.

```
const [validX, validY] = tf.tidy(() => {
    const d = data.getSingleValditationImage(1);
    return [
        d.xs.reshape([1, 28, 28, 1]),
        d.labels
    ];
});
```

Kun opetusaineisto on ajettu neuroverkon läpi, suoritetaan validaatio, eli testataan, pystyykö neuroverkko luokittelemaan numeron oikein. Tässä esimerkissä MnistData-luokasta kutsutaan funktiota, joka palauttaa testikuvan ja sen luokan, joka tallennetaan muuttujaan *label*. Tämä on array tyyppinen muuttuja, jossa on numero 1 merkitsemässä luokittelun odotettua tulosta. Testikuvassa on numero 7, joten numero 1 on indeksissä 7. Itse testikuvan lähdekoodi on liitteessä 1, josta pystyy hahmottamaan numeron 7. Kuvaliite 2. sisältää esikäsitellyn kuvan, joka ei ole oikeastaan enää ihmisluettava ja joka voidaan muuttaa tensoriksi tf.tensor2d() funktiolla.

```
async getSingleValditationImage(batchSize) {
    const image = [];
    const label = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0];

    const xs = tf.tensor2d(image, [1, IMAGE_SIZE]);
    const labels = tf.tensor2d(label, [1, NUM_CLASSES]);
    return {xs, labels};
}
```

7.5 Validaatio ja lopputulokset

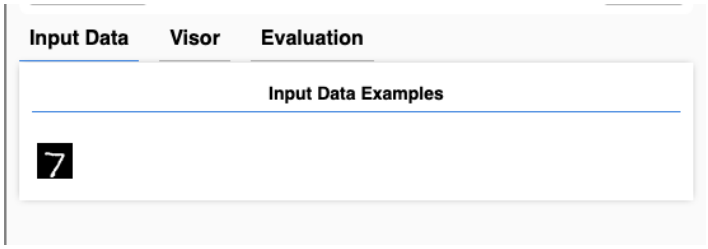
Neuroverkko pystyy luokittelemaan yhden numeron (liitteet 1 ja 2) oikein lähes aina, kun opetusaineistoa käytetään riittävästi. Muuttamalla arvoja muuttujissa BATCH_SIZE ja TRAIN_DATA_SIZE voidaan vaikuttaa luokittelun lopputulokseen.

```
const BATCH_SIZE = 512;
const TRAIN_DATA_SIZE = 5000;
```

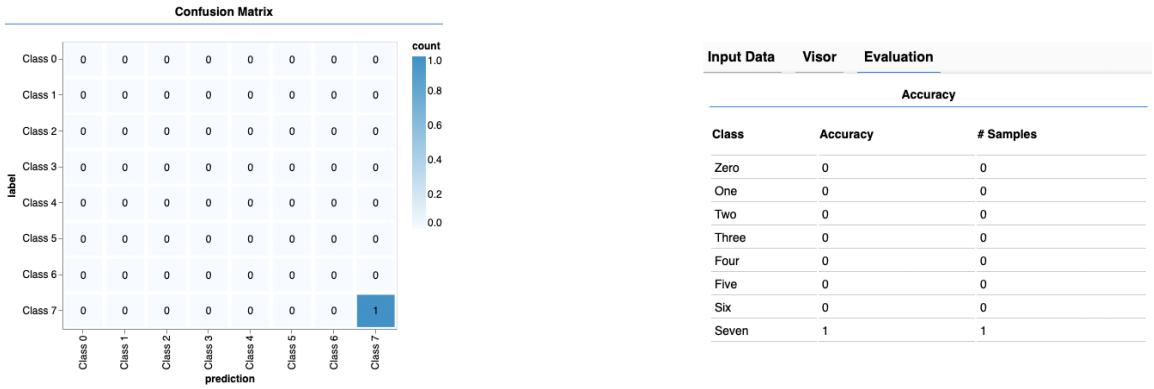
Neuroverkkoa opetetaan 5000 kuvalla, 512 kuvan erissä.

```
return model.fit(trainXs, trainYs, {
    batchSize: BATCH_SIZE,
    //validationData: [testXs, testYs],
    validationData: [validX, validY],
    epochs: 1,
    shuffle: true,
    callbacks: fitCallbacks
});
```

Näillä parametreilla luokittelu onnistui jokaisella ajolla oikein. Kun parametreja muutettiin



Ruutukaappaus validaatiossa käytetystä numerosta piirrettynä html-canvas elementille.

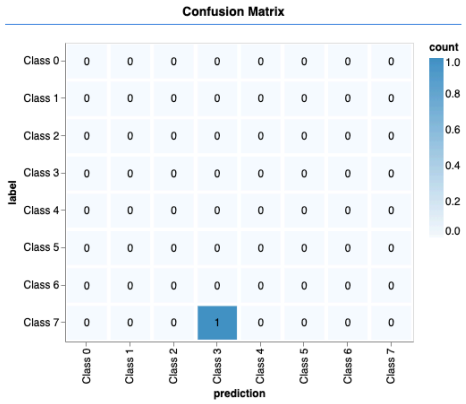


Kuva 6. Onnistunut luokittelu, sovellus luokittelee numeron 7 oikein. (koostettu kuvakaappauksista)

Kun parametrit asetetaan vaikkapa seuraavasti:

```
const BATCH_SIZE = 512;
const TRAIN_DATA_SIZE = 2000;
```

Luokittelu onnistuu huomattavasti harvemmin.



Accuracy

Class	Accuracy	# Samples
Zero	0	0
One	0	0
Two	0	0
Three	0	0
Four	0	0
Five	0	0
Six	0	0
Seven	0	1

Kuva 7. Epäonnistunut luokittelu, jossa luokittelun tulos on numero 3. (koostettu kuvakaappauksista)

7.6 Jatkokehitettävää

Sovellus pystyi tunnistamaan valmiiksi esikäsitellyn aineiston oikein. En kuitenkaan saanut toteutettua alkuperäistä ideaa, eli esikäsittelemään itsetuotettua aineistoa muotoon, jolla neuroverkon luokittelua voisi testata. Tämän lisäksi pitäisi miettiä, mitä muita esikäsittelyvaiheita tarvitaan. Kuva ainakin pitää muuttaa mustavalkoiseksi ja normalisoida 28x28 pikselin kokoiseksi.
Seuraavaksi pitäisi ratkaista segmentoinnin ongelma, eli löytää tapa numerosarjojen luokittelemiseen. Tähän löytyi muutama erilainen lähestymistapa, joista ensimmäinen on numerosarjoja tunnistava SDNN (Space Displacement Neural Network).
SDNN arkkitehtuuri saattaa olla toteutettavissa TensorFlow kirjaston avulla. Toinen vaihtoehto segmentoinnille on tunnistaa numerot kuvapinnalta, erotella ne ja luokitella yksitellen kuten tässä opinnäytteessä on tehty.
Jatkokehitykseen voisi laittaa myös perehtymisen ml5.js kirjastoon, joka on rakennettu TensorFlow.js:n päälle. Kirjaston tavoite on tehdä koneoppimisesta helpommin lähestyttävää.

7.7 Pohdintaa

Opinnäytetyön tavoitteena oli oppia ymmärtämään, miten koneoppiminen käytännössä toimii, minkälaisia sovelluksia sen avulla voidaan toteuttaa, onko ne toteutettavissa ilman syvällistä osaamista ja saada toteutettua kuvanluokittelija.
Sovelluksen toteutus jäi kesken eli siltä osin tavoitteisiin ei päästy. Kuvan muuttaminen JavaScriptilla array-tyyppiseksi dataksi, jonka voisi syöttää neuroverkkoon, ei onnistunut. Uskoisin pystyväni ratkaisemaan ongelman, jos kehitykseen voisi käyttää enemmän aikaa, nyt se loppui kesken.

Sen sijaan onnistuin demystifioimaan neuroverkkojen toimintaa ja opin TensorFlow:n rajapinnasta oleellisia tekniikoita. Tähän vaikutti ratkaisevasti tietoperustan kirjoittaminen ennen empiirisen vaiheen toteutusta. Kokeilin empiirisen osuuden toteuttamista jo ennen tietoperustaan perehtymistä, enkä ymmärtänyt sovelluksen sisäisestä logiikasta juuri mitään. Koneoppiminen vaatii perehtymistä vielä tässä vaiheessa, ennen kun ml5.js kaltaiset kirjastot yleistyvät ja tarjoavat kehitysympäristön, joka abstrahoi kompleksisen teknologian kehittäjiltä.

Lähteet

Akhter, S. 2017. Using machine learning to predict potential online gambling addicts. Aalto University. Espoo.

Bridle, J. S. 1990. Training Stochastic Model Recognition Algorithms as Networks can lead to Maximum Mutual Information Estimation of Parameters. Royal Signals and Radar Establishment. Great Malvern. Luettavissa: https://papers.nips.cc/paper/195-training-stochastic-model-recognition-algorithms-as-networks-can-lead-to-maximum-mutual-information-estimation-of-parameters.pdf. Luettu: 02.04.2019.

Krizhevsky, A & Sutskever, I & Hinton, G. 2012. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS 2012) 25, 1097–1105. MIT Press.

Krohn, J. 2017. Deep Learning with TensorFlow: Applications of Deep Neural Networks to Machine Learning Tasks. Addison-Wesley Professional 2017.

LeCun, Y & Boser, B & Denker, J & Henderson, D & Howard, R & Hubbard, W & Jackel, L. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. AT&T Bell Laboratories. Holmdel. Luettavissa: http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf. Luettu: 02.04.2019.

Ludwig, J. 2004. Image Convolution. Portland State University.

Medium 2019. Getting started with Neural Network for regression and Tensorflow. Luettavissa: https://medium.com/@rajatgupta310198/getting-started-with-neural-network-for-regression-and-tensorflow-58ad3bd75223. Luettu: 12.04.2019.

Mitchell, T. M. Machine Learning, 1 ed. McGraw-Hill, Inc., New York, NY, USA, 1997.

Mohammed, M & Muhammad, B & Eihab, B. 2017. Machine Learning Algorithms and Applications. Taylor & Francis Group. Florida USA.

Nilsson, N. 2009. The Quest for Artificial Intelligence, A History of Ideas and Achievements. Cambridge University Press.

Neural Networks and Deep Learning. Luettavissa: http://neuralnetworksanddeeplearning.com/images/tikz12.png. Luettu: 12.04.2019.

OCDevel 2019. Kuunneltavissa: http://ocdevel.com/mlg/2. Kuunneltu: 31.03.2019

O'Reilly Media, Inc 2017a. Safari Books. Intranet. Deep Learning with TensorFlow: Applications of Deep Neural Networks to Machine Learning Tasks. Luettu: 02.04.2019.

O'Reilly Media, Inc 2017b. Safari Books. Intranet. Statistics for Machine Learning. Luettu: 12.04.2019.

Ramachandran, P & Barret, Z & Quoc, V. 2017. Searching for Activation Functions. Google Brain. Mountain View.

SlideShare 2019. A brief survey of tensors. Luettavissa: https://www.slideshare.net/BertonEarnshaw/a-brief-survey-of-tensors. Luettu: 12.04.2019.

TensorFlow 2019. API Reference. Luettavissa: https://js.tensorflow.org/api/1.0.0/. Luettu: 01.04.2019.

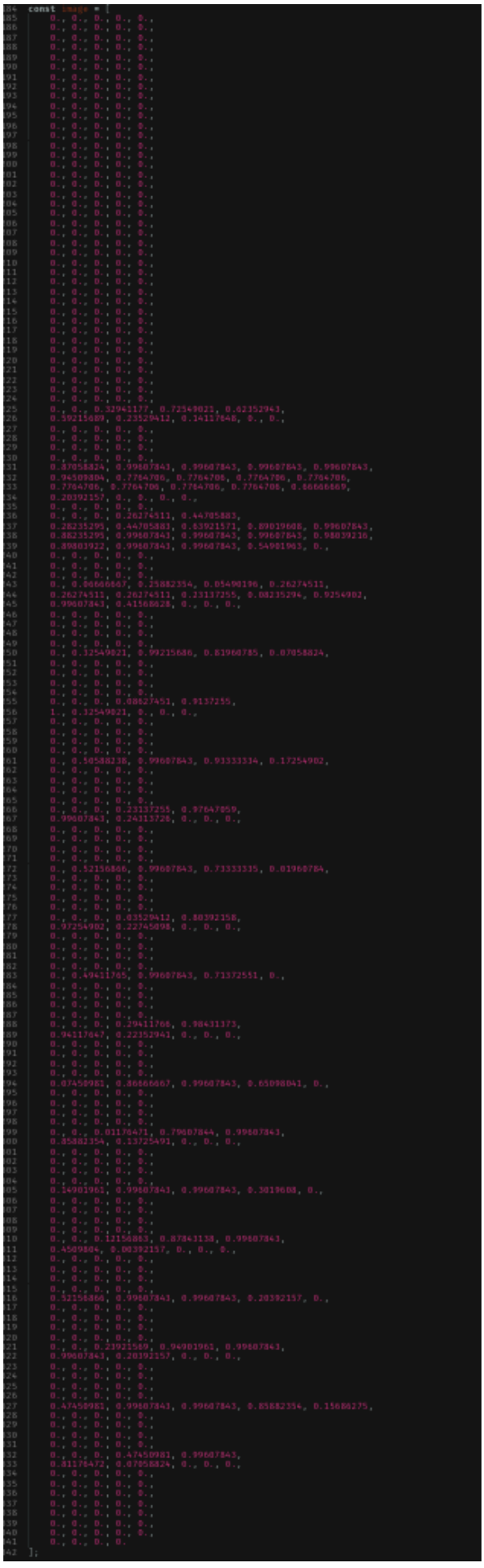
Yann Lecun 2019. The MNIST DATABASE. Luettavissa: http://yann.lecun.com/exdb/mnist/. Luettu: 12.04.2019.

Liitteet

Liite 1. Numero 7 array-tietotyyppinä

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  84, 185, 159, 151,  60,  36,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0, 222, 254, 254, 254, 254, 241, 198,
        198, 198, 198, 198, 198, 198, 198, 170,  52,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  67, 114,  72, 114, 163, 227, 254,
        225, 254, 254, 254, 250, 229, 254, 254, 140,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  17,  66,
         14,  67,  67,  67,  59,  21, 236, 254, 106,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,  83, 253, 209,  18,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,  22, 233, 255,  83,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0, 129, 254, 238,  44,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,  59, 249, 254,  62,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0, 133, 254, 187,   5,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   9, 205, 248,  58,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0, 126, 254, 182,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  75, 251, 240,  57,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         19, 221, 254, 166,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
        203, 254, 219,  35,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  38,
        254, 254,  77,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  31, 224,
        254, 115,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 133, 254,
        254,  52,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  61, 242, 254,
        254,  52,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 121, 254, 254,
        219,  40,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 121, 254, 207,
         18,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
```

Liite 2. Numero 7 esikäsiteltynä



Liite 3. Lähdekoodi

Sovelluksen lähdekoodi löytyy osoitteesta playground.kaleks.io. Sovellus on siellä myös käytettävissä. Lähdekoodi on myös liitteinä tiedosto kerrallaan.

Liite 4. JavaScript tiedosto, joka vastaa MNIST datasta ja validaatiodatasta.

```
/* eslint-disable no-unused-vars */
/* eslint-disable no-undef */

/**
 * @license
 * Copyright 2018 Google LLC. All Rights Reserved.
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
==========================================================================
=========
 */

const IMAGE_SIZE = 784;
const NUM_CLASSES = 10;
const NUM_DATASET_ELEMENTS = 65000;

const TRAIN_TEST_RATIO = 5 / 6;

const NUM_TRAIN_ELEMENTS = Math.floor(TRAIN_TEST_RATIO * NUM_DA-
TASET_ELEMENTS);
const NUM_TEST_ELEMENTS = NUM_DATASET_ELEMENTS - NUM_TRAIN_ELE-
MENTS;

const MNIST_IMAGES_SPRITE_PATH =
    'https://storage.googleapis.com/learnjs-data/model-builder/mnist_images.png';
const MNIST_LABELS_PATH =
    'https://storage.googleapis.com/learnjs-data/model-builder/mnist_labels_uint8';
```

```
/**
 * A class that fetches the sprited MNIST dataset and returns shuffled batches.
 *
 * NOTE: This will get much easier. For now, we do data fetching and
 * manipulation manually.
 */
export class MnistData {
  constructor() {
    this.shuffledTrainIndex = 0;
    this.shuffledTestIndex = 0;
  }

  async load() {
    // Make a request for the MNIST sprited image.
    const img = new Image();
    const canvas = document.createElement('canvas');
    const ctx = canvas.getContext('2d');
    const imgRequest = new Promise((resolve, reject) => {
      img.crossOrigin = '';
      img.onload = () => {
        img.width = img.naturalWidth;
        img.height = img.naturalHeight;

        const datasetBytesBuffer =
          new ArrayBuffer(NUM_DATASET_ELEMENTS * IMAGE_SIZE * 4);

        const chunkSize = 5000;
        canvas.width = img.width;
        canvas.height = chunkSize;

        for (let i = 0; i < NUM_DATASET_ELEMENTS / chunkSize; i++) {
          const datasetBytesView = new Float32Array(
            datasetBytesBuffer, i * IMAGE_SIZE * chunkSize * 4,
            IMAGE_SIZE * chunkSize);
          ctx.drawImage(
            img, 0, i * chunkSize, img.width, chunkSize, 0, 0, img.width,
            chunkSize);
```

```
            const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);

            for (let j = 0; j < imageData.data.length / 4; j++) {
              // All channels hold an equal value since the image is grayscale, so
              // just read the red channel.
              datasetBytesView[j] = imageData.data[j * 4] / 255;
            }
          }
          this.datasetImages = new Float32Array(datasetBytesBuffer);

          resolve();
        };
        img.src = MNIST_IMAGES_SPRITE_PATH;
      });

      const labelsRequest = fetch(MNIST_LABELS_PATH);
      const [imgResponse, labelsResponse] =
          await Promise.all([imgRequest, labelsRequest]);

      this.datasetLabels = new Uint8Array(await labelsResponse.arrayBuffer());

      // Create shuffled indices into the train/test set for when we select a
      // random dataset element for training / validation.
      this.trainIndices = tf.util.createShuffledIndices(NUM_TRAIN_ELEMENTS);
      this.testIndices = tf.util.createShuffledIndices(NUM_TEST_ELEMENTS);

      // Slice the the images and labels into train and test sets.
      this.trainImages =
          this.datasetImages.slice(0, IMAGE_SIZE * NUM_TRAIN_ELEMENTS);
      this.testImages = this.datasetImages.slice(IMAGE_SIZE * NUM_TRAIN_ELE-
MENTS);
      this.trainLabels =
          this.datasetLabels.slice(0, NUM_CLASSES * NUM_TRAIN_ELEMENTS);
      this.testLabels =
          this.datasetLabels.slice(NUM_CLASSES * NUM_TRAIN_ELEMENTS);
    }

    nextTrainBatch(batchSize) {
      return this.nextBatch(
```

```
      batchSize, [this.trainImages, this.trainLabels], () => {
        this.shuffledTrainIndex =
          (this.shuffledTrainIndex + 1) % this.trainIndices.length;
        return this.trainIndices[this.shuffledTrainIndex];
      });
  }

  nextTestBatch(batchSize) {
    return this.nextBatch(batchSize, [this.testImages, this.testLabels], () => {
      this.shuffledTestIndex =
        (this.shuffledTestIndex + 1) % this.testIndices.length;
      return this.testIndices[this.shuffledTestIndex];
    });
  }

  nextBatch(batchSize, data, index) {
    const batchImagesArray = new Float32Array(batchSize * IMAGE_SIZE);
    const batchLabelsArray = new Uint8Array(batchSize * NUM_CLASSES);

    for (let i = 0; i < batchSize; i++) {
      const idx = index();

      const image =
        data[0].slice(idx * IMAGE_SIZE, idx * IMAGE_SIZE + IMAGE_SIZE);
      batchImagesArray.set(image, i * IMAGE_SIZE);

      const label =
        data[1].slice(idx * NUM_CLASSES, idx * NUM_CLASSES + NUM_CLASSES);
      batchLabelsArray.set(label, i * NUM_CLASSES);
    }

    const xs = tf.tensor2d(batchImagesArray, [batchSize, IMAGE_SIZE]);
    const labels = tf.tensor2d(batchLabelsArray, [batchSize, NUM_CLASSES]);

    return { xs, labels };
  }

  getSingleValditationImage(batchSize) {
    /*
```

```
    const img = new Image();
    const canvas = document.createElement('canvas');
    const ctx = canvas.getContext('2d');
    let validationImg = null;
    const imgRequest = new Promise((resolve, reject) => {
      img.crossOrigin = '';
      img.onload = () => {
        img.width = img.naturalWidth;
        img.height = img.naturalHeight;

        const datasetBytesBuffer = new ArrayBuffer(IMAGE_SIZE * 4);

        canvas.width = img.width;
        canvas.height = img.height;

        const datasetBytesView = new Float32Array(datasetBytesBuffer, IMAGE_SIZE
* 4);

        ctx.drawImage(img, 0, 0);

        const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);

        for (let j = 0; j < imageData.data.length / 4; j++) {
          // All channels hold an equal value since the image is grayscale, so
          // just read the red channel.
          datasetBytesView[j] = imageData.data[j * 4] / 255;
        }
        validationImg = new Float32Array(datasetBytesBuffer);
        const imageArray = new Float32Array(batchSize * IMAGE_SIZE);
        const labelArray = new Uint8Array(batchSize * NUM_CLASSES);

        //imageArray.set(validationImg);
        //labelArray.set(samples.sample8.label);

        const xs = tf.tensor2d(validationImg, [batchSize, IMAGE_SIZE]);
        const labels = tf.tensor2d(samples.sample8.label, [batchSize, NUM_CLAS-
SES]);

        return {xs, labels};
```

```
    };
    img.src = samples.sample8.url;
});
*/

const image = [
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0.,
```

0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0.32941177, 0.72549021, 0.62352943,
0.59215689, 0.23529412, 0.14117648, 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0.87058824, 0.99607843, 0.99607843, 0.99607843, 0.99607843,
0.94509804, 0.7764706, 0.7764706, 0.7764706, 0.7764706,
0.7764706, 0.7764706, 0.7764706, 0.66666669,
0.20392157, 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0.26274511, 0.44705883,
0.28235295, 0.44705883, 0.63921571, 0.89019608, 0.99607843,
0.88235295, 0.99607843, 0.99607843, 0.99607843, 0.98039216,
0.89803922, 0.99607843, 0.99607843, 0.54901963, 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0.06666667, 0.25882354, 0.05490196, 0.26274511,
0.26274511, 0.26274511, 0.23137255, 0.08235294, 0.9254902,
0.99607843, 0.41568628, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0.32549021, 0.99215686, 0.81960785, 0.07058824,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0.08627451, 0.9137255,

1., 0.32549021, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0.50588238, 0.99607843, 0.93333334, 0.17254902,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0.23137255, 0.97647059,
0.99607843, 0.24313726, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0.52156866, 0.99607843, 0.73333335, 0.01960784,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0.03529412, 0.80392158,
0.97254902, 0.22745098, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0.49411765, 0.99607843, 0.71372551, 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0.29411766, 0.98431373,
0.94117647, 0.22352941, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0.07450981, 0.86666667, 0.99607843, 0.65098041, 0.,

0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0.01176471, 0.79607844, 0.99607843,
0.85882354, 0.13725491, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0.14901961, 0.99607843, 0.99607843, 0.3019608, 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0.12156863, 0.87843138, 0.99607843,
0.4509804, 0.00392157, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0.52156866, 0.99607843, 0.99607843, 0.20392157, 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0.23921569, 0.94901961, 0.99607843,
0.99607843, 0.20392157, 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0.47450981, 0.99607843, 0.99607843, 0.85882354, 0.15686275,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0., 0.,
0., 0., 0., 0.47450981, 0.99607843,
0.81176472, 0.07058824, 0., 0., 0.,

```
            0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0.,
            0., 0., 0., 0., 0.,
            0., 0., 0., 0.
        ];

        const label = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0];

        const xs = tf.tensor2d(image, [1, IMAGE_SIZE]);
        const labels = tf.tensor2d(label, [1, NUM_CLASSES]);
        return {xs, labels};
    }
}

const samples = {
    'sample1': {
        'url': 'http://playground.kaleks.io/sample1.png',
        'label': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
    },
    'sample4': {
        'url': 'http://playground.kaleks.io/sample4.png',
        'label': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
    },
    'sample5': {
        'url': 'http://playground.kaleks.io/sample5.png',
        'label': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
    },
    'sample8': {
        'url': 'http://playground.kaleks.io/mnist_8.png',
        'label': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
    },
}

const VALIDATION_DATA = samples.sample5.url;
const VALIDATION_LABEL = samples.sample5.label;
```

Liite 5. JavaScript tiedosto, joka luo neuroverkon ja vastaa sen opettamisesta.

```
/* eslint-disable no-unused-vars */
/* eslint-disable no-undef */

import { MnistData } from './og_data.js';

let VALIDATION_DATA = null;

async function showExamples(data) {
  // Create a container in the visor
  const surface =
    tfvis.visor().surface({ name: 'Input Data Examples', tab: 'Input Data' });

  // Get the examples
  //const examples = data.nextTestBatch(20);
  const examples = data.getSingleValditationImage(1);
  const numExamples = examples.xs.shape[0];

  // Create a canvas element to render each example
  for (let i = 0; i < numExamples; i++) {
    const imageTensor = tf.tidy(() => {
      // Reshape the image to 28x28 px
      return examples.xs
        .slice([i, 0], [1, examples.xs.shape[1]])
        .reshape([28, 28, 1]);
    });

    const canvas = document.createElement('canvas');
    canvas.width = 28;
    canvas.height = 28;
    canvas.style = 'margin: 4px;';
    await tf.toPixels(imageTensor, canvas);
    surface.drawArea.appendChild(canvas);

    imageTensor.dispose();
  }
}
```

```
async function run() {
  const data = new MnistData();
  await data.load();
  await showExamples(data);

  const model = getModel();
  tfvis.show.modelSummary({ name: 'Model Architecture' }, model);

  await train(model, data);
  await model.save('localstorage://my-model-1');

  await showAccuracy(model, data);
  await showConfusion(model, data);
}

document.addEventListener('DOMContentLoaded', run);

function getModel() {
  // Sequential-mallissa edellisen kerroksen ulostuloarvot
  // ovat seuraavan kerroksen sisääntuloarvoja.
  // Malli on yksinkertainen pino erilaisia kerroksia.
  const model = tf.sequential();

  const IMAGE_WIDTH = 28;
  const IMAGE_HEIGHT = 28;
  const IMAGE_CHANNELS = 1;

  //  Neuroverkon ensimmäinen kerros konvoluutio
  //  jolle määritellään parametrit. Aktivointifunktiona
  //  käytetään ReLU neuronia.
  model.add(tf.layers.conv2d({
    inputShape: [IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS],
    kernelSize: 5,
    filters: 8,
    strides: 1,
    activation: 'relu',
    kernelInitializer: 'varianceScaling'
  }));
```

```
// The MaxPooling layer acts as a sort of downsampling using max values
// in a region instead of averaging.

// Maxpool-kerros laskee 2x2 alueen pikseleistä korkeimman
// ja asettaa sen edustamaan koko 4 pikselin joukkoa.
model.add(tf.layers.maxPooling2d({ poolSize: [2, 2], strides: [2, 2] }));

// Luodaan toinen maxpooling- ja konvoluutiokerros suuremmalla määrällä suotimia.
model.add(tf.layers.conv2d({
   kernelSize: 5,
   filters: 16,
   strides: 1,
   activation: 'relu',
   kernelInitializer: 'varianceScaling'
}));
model.add(tf.layers.maxPooling2d({ poolSize: [2, 2], strides: [2, 2] }));

// Kaksiulotteinen ulostulodata muutetaan yksiulotteiseksi vektoriksi.
model.add(tf.layers.flatten());

// Viimeinen kerros on täysin kytketty kerros ja siinä on 10 neuronia,
// yksi jokaista luokaa kohti (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
const NUM_OUTPUT_CLASSES = 10;
model.add(tf.layers.dense({
   units: NUM_OUTPUT_CLASSES,
   kernelInitializer: 'varianceScaling',
   activation: 'softmax'
}));


// Choose an optimizer, loss function and accuracy metric,
// then compile and return the model

// Optimointiin käytetään adam() funktiota, joka asetetaan
// kääntämisen yhteydessä mallille parametriksi. Samalla
// virhefunktio.
const optimizer = tf.train.adam();
model.compile({
   optimizer: optimizer,
```

```
        loss: 'categoricalCrossentropy',
        metrics: ['accuracy'],
    });

    return model;
}

async function train(model, data) {
    const metrics = ['loss', 'val_loss', 'acc', 'val_acc'];
    const container = {
        name: 'Model Training', styles: { height: '1000px' }
    };
    const fitCallbacks = tfvis.show.fitCallbacks(container, metrics);

    //const BATCH_SIZE = 512;
    const BATCH_SIZE = 512;
    const TRAIN_DATA_SIZE = 2000;
    const TEST_DATA_SIZE = 1000;

    const [trainXs, trainYs] = tf.tidy(() => {
        const d = data.nextTrainBatch(TRAIN_DATA_SIZE);
        return [
            d.xs.reshape([TRAIN_DATA_SIZE, 28, 28, 1]),
            d.labels
        ];
    });

    const [testXs, testYs] = tf.tidy(() => {
        const d = data.nextTestBatch(TEST_DATA_SIZE);
        return [
            d.xs.reshape([TEST_DATA_SIZE, 28, 28, 1]),
            d.labels
        ];
    });

    const [validX, validY] = tf.tidy(() => {
        const d = data.getSingleValditationImage(1);
        return [
            d.xs.reshape([1, 28, 28, 1]),
```

```
            d.labels
        ];
    });

    // my own validata
    return model.fit(trainXs, trainYs, {
        batchSize: BATCH_SIZE,
        //validationData: [testXs, testYs],
        validationData: [validX, validY],
        epochs: 1,
        shuffle: true,
        callbacks: fitCallbacks
    });
}

const metrics = ['loss', 'val_loss', 'acc', 'val_acc'];

const classNames = ['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine'];

function doPrediction(model, data, testDataSize = 1) {
    const IMAGE_WIDTH = 28;
    const IMAGE_HEIGHT = 28;
    const testData = data.getSingleValditationImage(testDataSize);
    const testxs = testData.xs.reshape([testDataSize, IMAGE_WIDTH, IMAGE_HEIGHT,
1]);
    const labels = testData.labels.argMax([-1]);
    const preds = model.predict(testxs).argMax([-1]);

    testxs.dispose();
    return [preds, labels];
}


async function showAccuracy(model, data) {
    const [preds, labels] = doPrediction(model, data);
    const classAccuracy = await tfvis.metrics.perClassAccuracy(labels, preds);
    const container = { name: 'Accuracy', tab: 'Evaluation' };
    tfvis.show.perClassAccuracy(container, classAccuracy, classNames);
```

```
    labels.dispose();
}

async function showConfusion(model, data) {
    const [preds, labels] = doPrediction(model, data);
    const confusionMatrix = await tfvis.metrics.confusionMatrix(labels, preds);
    const container = { name: 'Confusion Matrix', tab: 'Evaluation' };
    tfvis.render.confusionMatrix(
        container, { values: confusionMatrix }, classNames);

    labels.dispose();
}
```

Liite 6. Sovelluksen index.html tiedosto

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>TensorFlow.js Tutorial</title>
    <link rel="stylesheet" type="text/css" href="./styles.css">
    <!-- Import TensorFlow.js -->
    <script src="https://cdn.jsdelivr.net/npm/@tensor-
flow/tfjs@0.14.1/dist/tf.min.js"></script>
    <!-- Import tfjs-vis -->
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis@1.0.2/dist/tfjs-
vis.umd.min.js"></script>

    <!-- Import the data file -->
    <script src="./og_data.js" type="module"></script>

    <!-- Import the main script file -->
    <script src="og_script.js" type="module"></script>

</head>

<body>
    <div id="my_junk">
        <img src="./numberseven.png">
    </div>
</body>

</html>
```

Liite 7. Sovelluksen tyylitiedosto

```
.css-q2ki5z {
    right: 10% !important;

}

.css-iettse {
    width: 750px !important;
}

body {
    background: #999;
}
```