Konvoluutioneuroverkot

Teemu Sarapisto

Aine
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 5. maaliskuuta 2018

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Teemu Sarapisto

Työn nimi — Arbetets titel — Title

Konvoluutioneuroverkot

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Aine	5. maaliskuuta 2018	8

Tiivistelmä — Referat — Abstract

Viimeisen hieman yli kymmenen vuoden aikana voidaan sanoa keinotekoisten neuroverkkojen ja syväoppimisen tehneen läpimurron. Syväoppimisen voidaan katsoa syntyneen jo 40-luvulla, mutta laajamittaiseen sovelluskäyttöön se on tullut vasta viime vuosina, kun sekä riittävä määrä luokiteltua dataa, että riittävästi prosessointitehoa on tullut helposti saataville. Myös algoritmipuolella tapahtuneet edistykset ovat edesauttaneet läpimurtoa. Aikaisemmin koneoppimisen alalla haasteelliseksi osoittautuneissa sovelluskohteissa kuten kuvien sekä puheen sisällön tunnistamisessa keinotekoiset neuroverkot ovat osoittautuneet toistaiseksi ylivoimaisesti parhaiten toimiviksi ratkaisuiksi.

Neuroverkkojen opetuksessa tärkeimpiä menetelmiä ovat gradienttimenetelmä ja takaisinvirtausalgoritmi

Konvoluutioneuroverkot ovat eräänlaisia neuroverkkoja jotka soveltuvat hyvin kuvien ja muiden paikallisuudesta hyötyvien syötteiden käsittelyyn

Avainsanat — Nyckelord — Keywords
avainsana 1, avainsana 2, avainsana 3

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Syväoppimisen historia ulottuu 1940-luvulle asti, jolloin kybernetiikan tutkimuksen myötä McCulloch ja Pitts kehittivät mukaansa nimetyn McCulloch-Pitts neuronin, tarkoituksenaan luoda matemaattinen malli jolla kuvailla biologista aivoissa tapahtuvaa oppimista. Heidän kehittelemällään lineaarisella mallilla oli mahdollista tunnistaa kahden syötekategorian välillä kummasta on kyse kategoriat määrittelevien painotuksien (weights) avulla, ihmisen joutuessa määrittelemään nämä painot.

Vasta 1950-luvulla kehitettiin ensimmäinen malli joka pystyi oppimaan syötekategorioita kuvaavat painotukset niistä annettujen esimerkkien perusteella, niin kutsuttu perseptroni.

Kiinnostuksen kybernetiikkaan hiivuttua 1960-luvun aikana, seuraavan kerran merkittävää kehitystä tapahtui 80-90-luvulla konnektionismin tuodessa neuroverkkomallit takaisin suosioon. Yksi tärkeimmistä näihin aikoihin tapahtuneista kehityksistä syväoppimisen kannalta oli, kun takaisinvirtausalgoritmin (backpropagation) keksittiin 1986 mahdollistavan monikerroksisten neuroverkkojen harjoittamisen verrattain tehokkaasti.

90-luvun puolivälin jälkeen syväoppiminen eli jälleen hiljaiseloa vuoteen 2006 asti, jonka jälkeen se on ollut jatkuvasti pinnalla tähän päivään asti. Geoffrey Hinton osoitti tällöin syvien uskomusverkkojen (deep belief network) olevan harjoitettavissa tehokkaasti tasoittain ja muut tutkimusryhmät yleistivät tämän harjoitustavan muille syville keinotekoisille neuroverkoille. Näiden tutkimuksien myötä syväoppiminen terminä alkoi yleistyä, termin käytön tarkoituksena korostaa aikaisempaa syvempien verkkojen harjoitettavissa olemista.

Lopullinen syväoppimisen läpimurto tapahtui vuonna 2012 kun suurimman kuvista objektien tunnistamisen kilpailun, ImageNet Large Scale Visual Recognition Challengen (ILSVRC), voitti ensimmäistä kertaa syvä (konvoluutio)neuroverkko. Voitto tapahtui myös huomattavalla erolla toisen sijan saavuttaneeseen sekä aikaisempien vuosien voittajiin. Tämän jälkeen kilpailun on joka vuosi voittanut syvä konvoluutioverkko, ja nykyään neuroverkot pärjäävät kyseisessä (varsin rajoitetussa) tunnistamistehtävässä ihmistä paremmin.

Nykyään käytössä olevat ihmisille monimutkaisissakin tehtävissä pärjäävät oppimisalgoritmit ovat pääasiassa samoja kuin jo 80-luvulla käytössä olleet. Jonkin verran muutoksia silloisiin algoritmeihin on tehty syvien verkkorakenteiden harjoittamista helpottavina yksinkertaistuksina, mutta selkeästi suurin syy syväoppimisen tärkeäksi muuttumiseen vasta äskettäin on kuitenkin yhteiskunnan digitalisoitumisen myötä merkittävästi kasvanut helposti saatavilla olevan luokitellun datan määrä, sekä valtavasti kasvanut laskentakapasiteetti, jotka olivat edellytyksiä algoritmien kunnolliselle toiminnalle.

Esimerkkinä tarvittavan harjoitusdatan määrästä konvoluutioneuroverk-

kojen harjoittamiseen kuvien luokittelua varten toimii yleensä tuhansia ellei jopa miljoonia kuvien sisällön perusteella etukäteen luokiteltuja kuvia. Esimerkiksi ILSVRC-kilpailun harjoitusdatana käytössä oleva ImageNet sisältää yli 14 miljoonaa luokiteltua kuvaa. (http://image-net.org/about-stats)

Kuvien sisällön tunnistamisen lisäksi syväoppiminen on osoittautunut erittäin hyödylliseksi useissa haasteellisissa sovelluskohteissa, kuten puheen sisällön, liikennemerkkien luokittelun sekä jalankulkijoiden tunnistamisessa.

2 Neuroverkkojen rakenne

2.1 Keinotekoinen neuroni

TODO: selkiytä kaavahommelia

Biologisista vaikuttimistaan huolimatta keinotekoiset neuronit ovat käytännössä Kaavan 1 muotoisia matemaattisia funktioita.

$$\Sigma w_i x_i \mapsto f(\Sigma w_i x_i + b) \tag{1}$$

Yleisessä muodossaan neuroni ottaa vastaan yhden tai useampia syötteitä x_1, x_2, ..., x_n, joista kullekin on asetettu jokin painoarvo w_i. Syötteiden ja painotuksien tulojen summa $\Sigma x_i w_i$ annetaan parametrina aktivaatiofunktiolle f ja tämän funktion arvo toimii neuronin lopullisena ulostuloarvona.

Toisinaan käytetään myös taipumusvakiota (bias) b, joka lisätään syötteiden ja painotuksien tulojen summaan.

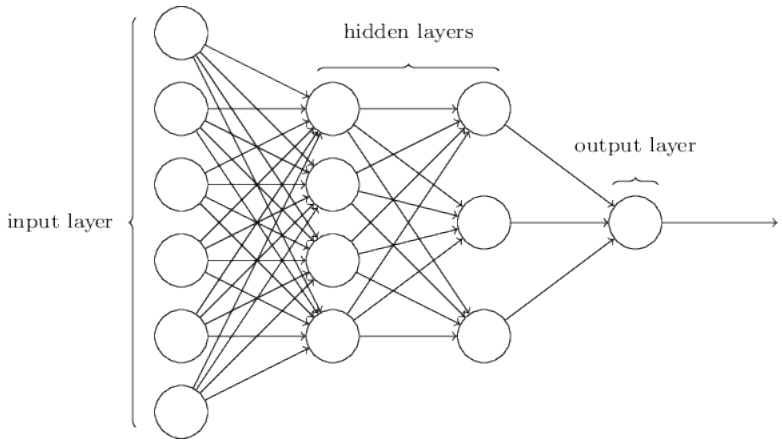
Ensimmäinen tällainen neuroni, perseptroni, kehitettiin 50-luvulla. Sen syötteet ja ulostuloarvot ovat binäärisiä ja aktivaatiofunktiona toimii Kaavan 2 mukainen funktio.

$$ulostuloarvo \begin{cases} 0 \; jos \; \Sigma x_i w_i + b \leq 0 \\ 1 \; jos \; \Sigma x_i w_i + b > 0 \end{cases} \tag{2}$$

Yksittäisen neuronin tasolla neuronien oppiminen tapahtuu syötteiden painotuksien ja taipumusarvon muuttumisen kautta. Perseptroneja käytettäessä törmätään kuitenkin usein ongelmaan, jossa yksi pieni muutos painotuksissa tai taipumusarvossa johtaa ulostuloarvon vaihtumiseen, joka saattaa aiheuttaa suuria muutoksia ulostuloarvoissa myös koko neuroverkon tasolla. Usein halutaan hienovaraisempia muutoksia ja tällöin käytetään neuroneita joiden syöte- ja paluuarvot voivat olla myös mitä vain reaalilukuja nollan ja yhden väliltä. Esimerkiksi yksi tällainen laajalti käytössä oleva neuroni on sigmoidinen neuroni, jonka aktivaatiofunktiona toimii sigmoidinen funktio.

2.2 Keinotekoisten neuroverkkojen rakenne

Yksinkertaisimman verkkorakenteen omaavat eteenpäinsyöttävät neuroverkot muodostetaan tasoittain, jossa jokaisen verkon tason neuronit saavat



Kuva 1: http://neuralnetworksanddeeplearning.com/chap1.html

syötteenään niitä edeltävän tason neuroneiden ulostuloarvot. Poikkeuksena ensimmäinen taso (kuvassa vasemmanpuoleisimpana), joihin verkon syöte koodataan. Esimerkiksi haluttaessa syöttää 64x64 kuva neuroverkolle, voidaan syötekerroksena käyttää 64x64 neuronin kerrosta, johon kuvan pikselien väriarvot koodataan.

Vaikka syväoppimista voidaan harjoittaa myös muutoin kuin keinotekoisilla neuroverkoilla, neuroverkkojen tapauksessa termillä viitataan neuroverkkojen piilokerroksiin ja niiden määrään. Kasvattamalla neuroverkkotasojen sekä tasoissa olevien neuronien määrää neuroverkoilla voidaan mallintaa entistä monimutkaisempia funktioita.

3 Neuroverkkojen harjoittaminen

Neuroverkkojen on todistettu olevan universaaleja approksimaattoreita, eli voidaan taata, että mille tahansa funktiolle on löydettävissä neuroverkko, joka pystyy mallintamaan funktiota halutulla tarkkuudella, kunhan laskentaresursseja on riittävästi käytettävissä [3]. Neuroverkkojen harjoittamisen voidaan siis sanoa olevan neuroverkon tekemän virheen minimointia sen approksimoidessa jotakin funktiota.

Tätä neuroverkon tekemän virheen määrää voidaan mitata virhefunktion (error function) avulla. Usein käytetään neliöllistä virhefunktiota:

$$C(x) = \frac{1}{2}\sum_{i=1}^{N} \|y(x_i) - a(x_i)\|^2$$

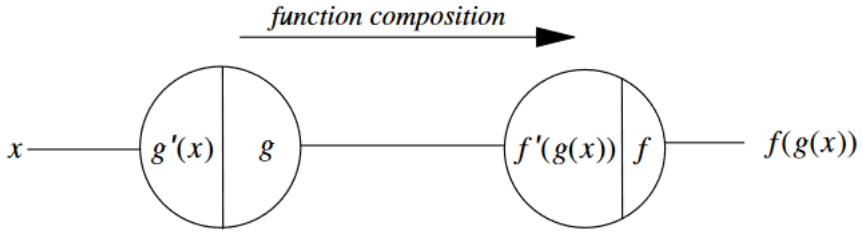
jossa $y(x_i)$ on neuroverkon tulos syötteellä x_i, $a(x_i)$ on harjoitusdatan i:s yksikkö, ja N syötteiden määrä.

3.1 Gradienttimenetelmä ja takaisinvirtausalgoritmi

Takaisinvirtausalgoritmilla etsitään virhefunktion minimiä säätämällä painotuksia ja taipumusvakioita käyttäen gradienttimenetelmää (gradient descent).

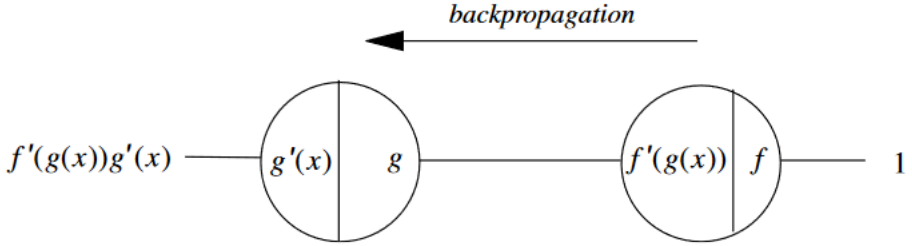
Gradienttimenetelmä on numeerinen menetelmä joka toimii minkä tahansa derivoituvan funktion lokaalien minimien etsintään. Neuroverkkojen tapauksessa gradienttimenetelmän käytön edellytyksenä on, että neuronien aktivaatiofunktiot ovat derivoituvia funktioita, jolloin myös koko neuroverkon tuottamat arvot ja siten virhefunktio ovat derivoituvia. Gradientti kertoo mihin suuntaan funktion arvo laskee nopeimmin, joten kuljettaessa iteratiivisesti tähän suuntaan kunnes gradientti on tarpeeksi pieni, päädytään lähelle paikallista minimiä.

Takaisinvirtausalgoritmi on tiivistetysti tapa derivoida neuroverkon muodostamaa yhdistelmäfunktiota derivaatan ketjusäännön mukaisesti. Tämä voidaan tehdä esimerkiksi kahden neuroniyksikön tapauksessa seuraavasti:



Kuva 2: http://page.mi.fu-berlin.de/rojas/neural/ kappale 7.2.2

Eteenpäinsyöttövaiheessa normaalin eteenpäinsyötön lisäksi jokaisen yksikön kohdalla tallennetaan neuroniin sen (TODO: ei ihan näin yksinkertaista, ulostuloarvo on vakio joka derivoitaessa katoaisi, mutta jotakin suhteessa sen painoihin) ulostuloarvon derivaatta. Kuvassa 3.1 tämä tallennettava arvo näkyy palloina esitettyjen neuroniyksiköiden vasemmassa puoliskossa.



Kuva 3: http://page.mi.fu-berlin.de/rojas/neural/ kappale 7.2.2

Takaisinvirtausvaiheessa verkkoa kuljetaan algoritmin nimen mukaisesti takaperin aikaisempaan nähden. Takaisinvirtaus aloitetaan syöttämällä verkkoon numero yksi, kuljettaen tätä arvoa mukana neuronilta toiselle, ja uuteen

neuroniyksikköön saavuttaessa kertomalla mukana kulkeva arvo neuroniin tallennetulla, kuvassa 3.1 neuroniyksikön vasemmalla puoliskolla näkyvällä arvolla. Näin lopulta esimerkin kahden neuroniyksikön läpi kulkemisen jälkeen $f(g(x))$ muotoinen funktio on saatu sen derivoituun muotoon $f'(g(x))g'(x)$ joka vastaa ketjusäännön avulla derivoitua tulosta. Tämä metodi yleistyy myös monimutkaisemmille verkkorakenteille, ja on helposti hajautettavissa jakamalla verkko osaverkkoihin ja yhdistämällä osaverkot myöhemmin.

TODO:

- Mitä backpropagation algoritmin esimerkissä oikeastaan tallennetaan yksikköön eteenpäinsyöttövaiheessa jotta derivoidessa ei jää vain 0?

- Verkkoon voidaan ajatella lisättävän backpropagationia varten ulostulokerroksen jälkeen ulostulot yhdistävä virhefunktion arvon kertova "neuroni", se auttoi jossain

- Joku summarum hommasta

Virhefunktion arvon laskeminen jokaiselle syötteelle ja keskiarvon ottaminen ja tämän perusteella takaisinvirtausalgoritmin suorittaminen olisi erittäin raskasta kun syötteitä voi olla miljoonia kappaleita. Tämän takia käytetään yleensä stokastista gradienttimenetelmää, jossa virhefunktion keskiarvo lasketaan kaikkien syötteiden sijaan joukolle satunnaisesti valittuja syötteitä.

3.2 Ongelmia

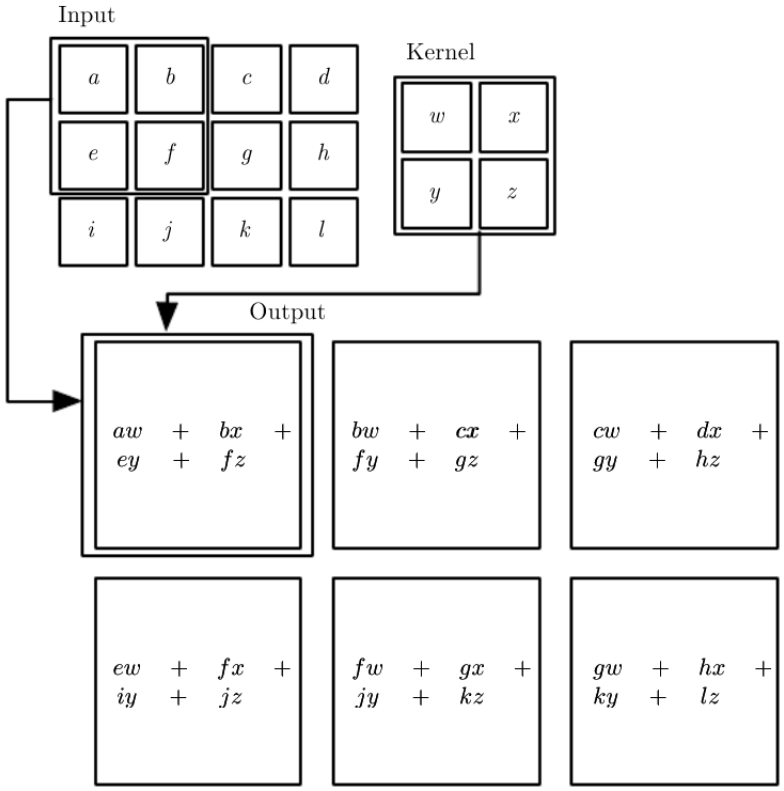
Suuri haaste neuroverkkojen harjoittamisessa on ylisovitus (overfitting) jossa neuroverkon virhefunktion arvo on harjoitusdatalla saatu erittäin pieneksi, mutta uuden datan kanssa virhefunktio antaa suuria arvoja. Tällöin neuroverkon oppima malli vastaa harjoitusdataa liian tarkkaan, eikä enää suoriudu yleisestä tapauksesta toivotulla tavalla. Ylisovitusta korjaamaan on kehitetty metodeja kuten esimerkiksi neuroniyksikköjen pudotus (dropout) jossa harjoitusvaiheessa yksittäisiä neuroneita poistetaan satunnaisesti käytöstä, joka estää yksittäisiä neuroneita naapureineen erikoistumatta tiettyyn datan ominaisuuteen liian tarkasti.

Neuroverkkojen harjoittamisessa olennaisessa osassa on myös verkon alkuperäisten painotuksien valitseminen sopivalla tavalla. Joissakin tapauksissa tämä tehdään vain valitsemalla painotuksille satunnaiset alkuarvot.

4 Konvoluutioneuroverkkojen rakenne

Konvoluutioneuroverkkojen rakenne eroaa tavanomaisista täysin yhdistetyistä neuroverkoista konvoluutio- ja kokoamiskerroksien (pooling layer) olemassaolon, kerrosten mahdollisen rinnakkaisuuden, sekä ominaisuuskarttojen jaettujen painojen kautta.

4.1 Konvoluutiokerrokset



Kuva 4: http://www.deeplearningbook.org/contents/convnets.html

Konvoluutioverkkojen nimi juontaa juurensa matemaattiseen operaattoriin konvoluutio, sillä kaava jolla konvoluutiokerrosten neuroneiden syötteet $a_{x,y}$ painotuksineen $w_{x,y}$ voidaan esittää matemaattisesti esimerkiksi 5x5 kokoiselle paikalliselle vastaanottavalle kentälle (local receptive field) muodossa:

$$\sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m}a_{j+l,k+m}$$

joka on käytännössä diskreetti konvoluutio, jossa painotukset ovat konvoluution ydin (kernel).

Intuitiivisemmin, konvoluution voidaan ajatella olevan liukuva ikkuna, joka rajoittaa neuronit saamaan kuvan 4.1 mukaisesti syötteenään vain osan syötekerroksensa ulostuloista, täysin yhdistetyistä neuroverkkokerroksista poiketen. Tämä rajoittuneisuus mahdollistaa neuroneiden erikoistumisen johonkin osa-alueeseen syötteissään, joka on erityisen hyödyllistä haluttaessa käyttää neuroverkkoja syötteiden tutkintaan joissa ilmenee paikallisuutta.

Esimerkiksi kuvat ovat erittäin paikallistuneita, pikselien etäisyys toisistaan korreloi vahvasti sen kanssa, liittyykö niiden sisältö toisiinsa.

Kuvasta 4.1 ilmenee myös toinen yleinen konvoluutiokerrosten piirre: mikäli tehdään vain konvoluutioita joissa ydin mahtuu kokonaan syötekuvaan, konvoluutiokerroksella on vähemmän ulostuloja kuin sisääntuloja. Kuvan tapauksessa nähdään syötematriisin ollessa 3x4, ja ytimen 2x2 kokoinen, ulostulomatriisin kooksi tulee 2x3. Tällä tavalla konvoluutio mahdollistaa myös sen, että konvoluutioverkot voivat ottaa vastaan vaihtelevan kokoisia syötteitä.

4.2 Ominaisuuskartat ja jaetut painot

Konvoluutiokerroksia on yleensä useita rinnakkain, ja konvoluutiokerrosten neuronit jakavat samassa kerroksessa olevien neuroneiden kesken yhteiset painot (shared weights). Kerroksien yhteiset painot mahdollistavat sen, että kukin kerros oppii tunnistamaan sijainnista riippumattomia (translationally invariant) ominaisuuksia syötteistään, ja yksittäisen neuronin voidaan ajatella kertovan löytyykö sen saamien syötteiden alueelta tätä ominaisuutta. Esimerkiksi kuvien tapauksessa ominaisuuskartta joka löytää kuvasta suoria viivoja saattaa olla hyödyllinen, sillä suoria viivoja voi ilmetä useassa eri paikassa kuvassa, ja toisaalta korkeammalla abstraktiotasolla kissa on edelleen kissa, vaikka sitä olisi siirretty muutamia pikseleitä johonkin suuntaan.

(TODO: voisi muotoilla paremmin, laajemmin, esimerkein?) Painotuksien jakamisen ansiosta konvoluutioverkon parametrien määrä on usein huomattavasti vastaavaa täysin yhdistettyä neuroverkkoa pienempi, mahdollistaen tehokkaamman harjoittamisen.

4.3 Kokoaminen ja tehokkuus

Usein konvoluutioverkoissa käytetään heti konvoluutiokerrosten jälkeen kokoamiskerroksia, jotka tekevät yhteenvedon jostakin edeltävästä neuroverkkokerroksen alueesta. Hyvin yleinen maksimikokoamiskerros (max-pooling) antaa ulostuloksi esimerkiksi 2x2 kokoiselle syötteelle suurimman yksittäisen aktivointiarvon alueen neuroneista.

(TODO pidemmin?:) Sekä kokoamiskerrokset että jaetut painotukset yhdessä vähentävät konvoluutioneuroverkoissa tarvittavien harjoitettavien parametrien määrää ja mahdollistavat niiden tehokkaan käytön suurillekkin syötteille.

4.4 Täysin yhdistetyt kerrokset

Viimeinen kerros konvoluutioneuroverkoista on yleensä tavallinen täysin yhdistetty verkkokerros, joka samalla litistää (flattening) useaan rinnakkaiseen kerrokseen jakautuneen verkon takaisin yhteen.

5 Konvoluutioneuroverkkojen sovellukset

5.1 Kuvien luokittelu

Tuloskerroksessa käytännössä todennäköisyysarvio luokittain

Lähteet

[1] Goodfellow, Ian, Bengio, Yoshua ja Courville, Aaron: *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Hinton, Geoffrey E., Osindero, Simon ja Teh, Yee Whye: *A Fast Learning Algorithm for Deep Belief Nets.* Neural Computation, 18(7):1527–1554, 2006. `https://doi.org/10.1162/neco.2006.18.7.1527`.

[3] Hornik, Kurt, Stinchcombe, Maxwell B. ja White, Halbert: *Multilayer feedforward networks are universal approximators.* Neural Networks, 2(5):359–366, 1989. `https://doi.org/10.1016/0893-6080(89)90020-8`.

[4] Nielsen, Michael A.: *Neural Networks and Deep Learning.* Determination Press, 2015. `http://neuralnetworksanddeeplearning.com/`.

[5] Rojas, Raúl: *Neural Networks - A Systematic Introduction.* Springer, 1996.