Konvolutionaaliset neuroverkot

Teemu Sarapisto

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 4. toukokuuta 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

Konvolutionaaliset neuroverkot

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma	4. toukokuuta 2018	15

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

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

Avainsanat — Nyckelord — Keywords
neuroverkot, neuroverkkojen harjoittaminen, konvolutionaaliset neuroverkot, kuvien luokittelu

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Syväoppimisen historia ulottuu 1940-luvulle asti [2], jolloin kybernetiikan tutkimuksen myötä McCulloch ja Pitts kehittivät mukaansa nimetyn McCulloch-Pitts neuronin, tarkoituksenaan luoda matemaattinen malli, jolla kuvailla biologista aivoissa tapahtuvaa oppimista [8]. Heidän kehittelemällään lineaarisella mallilla oli mahdollista tunnistaa kahden syötekategorian välillä kategoriat määrittelevien painotuksien 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 [11].

Kiinnostus kybernetiikkaan hiipui 1960-luvun aikana, jonka jälkeen merkittävää kehitystä tapahtui seuraavan kerran vasta 80-90-luvulla konnektionismin tuodessa neuroverkkomallit takaisin suosioon. Yksi tärkeimmistä näihin aikoihin tapahtuneista kehityksistä syväoppimisen kannalta oli, kun takaisinvirtausalgoritmin (backpropagation) keksittiin 1986 soveltuvan monikerroksisten neuroverkkojen tehokkaaseen harjoittamiseen [12].

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 [4]. 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 konvolutionaalinen neuroverkko [6]. 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 [2].

Nykyään ihmisille monimutkaisissakin tehtävissä pärjäävät oppimisalgoritmit ovat pääasiassa samoja kuin jo 80-luvulla keksityt. Jonkin verran muutoksia silloisiin algoritmeihin on kuitenkin tehty, erityisesti syvien verkkorakenteiden harjoittamista helpottavina yksinkertaistuksina. Suurimmat syyt syväoppimisen tärkeäksi muuttumiselle vasta viime vuosina on yhteiskunnan digitalisoitumisen myötä merkittävästi kasvanut helposti saatavilla olevan luokitellun datan määrä ja valtavasti kasvanut laskentakapasiteetti, jotka olivat edellytyksiä algoritmien mielekkäälle käyttämiselle [2].

Esimerkkinä tarvittavan harjoitusdatan määrästä konvolutionaalisten neuroverkkojen harjoittamiseen kuvien luokittelua varten toimii yleensä tuhansia ellei jopa miljoonia kuvien sisällön perusteella etukäteen luokiteltuja

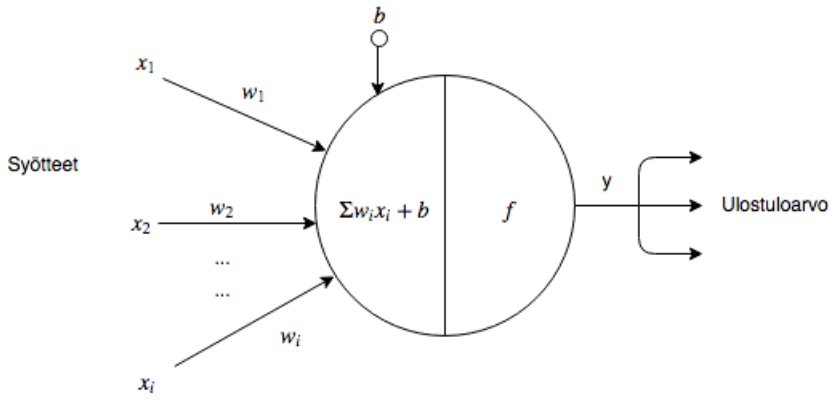
kuvia. Esimerkiksi ILSVRC-kilpailun harjoitusdatana käytössä oleva ImageNet sisältää yli 14 miljoonaa luokiteltua kuvaa [9]. Kuvien sisällön tunnistamisen lisäksi syväoppiminen on osoittautunut erittäin hyödylliseksi useissa aikaisemmin haasteellisiksi osoittautuneissa sovelluskohteissa, kuten puheentunnistuksessa [1], liikennemerkkien luokittelussa [13], sekä jalankulkijoiden tunnistamisessa [15].

Tässä tutkielmassa aloitetaan käymällä läpi yleisiä keinotekoisten neuroverkkojen piirteitä, kuten yksittäisten neuronien ja eteenpäinsyöttävien neuroverkkojen rakennetta. Seuraavaksi käydään läpi kuinka neuroverkkoja harjoitetaan harjoitusaineiston avulla takaisinvirtausalgoritmia hyödyntäen. Lopuksi tutkitaan miten konvolutionaaliset neuroverkot eroavat muista neuroverkoista ja mitä sovelluksia niille löytyy.

2 Neuroverkkojen rakenne

Tässä kappaleessa käydään läpi Michael A. Nielsen, Neural Networks and Deep Learning [10] kirjaan pohjautuen perusteet keinotekoisten neuroneiden ja neuroverkkojen rakenteesta.

2.1 Keinotekoinen neuroni



Kuva 1: Keinotekoisen neuronin rakenne

Keinotekoiset neuronit ottavat vastaan yhden tai useampia syötteitä x_i ... x_i, joista kullakin on jokin painotusarvo w_i. Neuroneilla on yksi ulostuloarvo y, joka muodostetaan sen syötteistä kahdessa vaiheessa.

Ensimmäisessä vaiheessa syötteet kerrotaan painotusarvollaan ja litistetään yhdeksi arvoksi summaamalla. Summaan lisätään lopuksi taipumusvakio (bias) b.

Toisessa vaiheessa summa syötetään aktivaatiofunktiolle f, jonka ulostuloarvo toimii koko neuronin yksittäisenä ulostuloarvona. Vaikka neuroneilla

on vain yksi ulostuloarvo, tämä arvo voi toimia usean muun neuronin syötteenä.

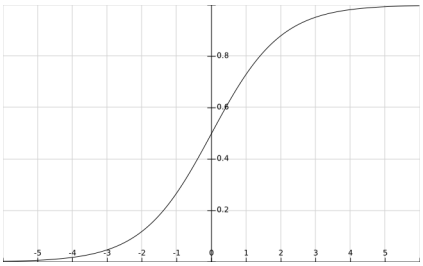
Aktivaatiofunktioina käytetään usein derivoituvia funktioita, sillä tämä helpottaa myöhemmin kappaleessa 3.1 esiteltävän gradienttimenetelmän käyttöä. Eräs suosittu aktivaatiofunktio on sigmoidinen funktio

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{1}$$

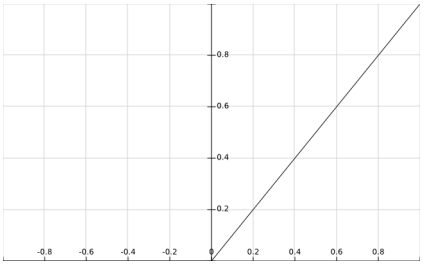
Nykyään kuitenkin suosituin aktivaatiofunktio on Rectified Linear Unit (ReLU)

$$f(x) = max(0, x), \tag{2}$$

koska sitä käytettäessä on todettu oppimisen olevan monikerroksisissa neuroverkkoarkkitehtuureissa huomattavasti nopeampaa verrattuna sigmoidiseen funktioon [7].



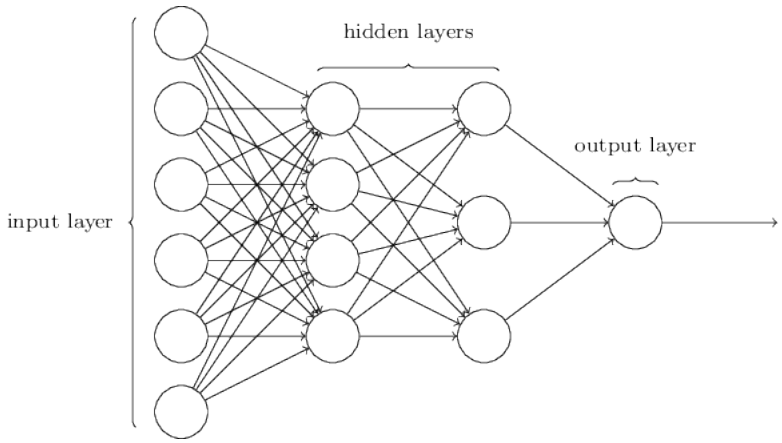
Kuva 2: Sigmoidinen funktio



Kuva 3: Rectified Linear Unit (ReLU)

2.2 Keinotekoisten neuroverkkojen rakenne

Yksinkertaisimman verkkorakenteen omaavat eteenpäinsyöttävät neuroverkot muodostetaan kerroksittain niin, että kunkin verkon kerroksen neuronien syötteet ovat niitä edeltävän tason neuroneiden ulostuloarvoja. Yleisimmissä täysin yhdistetyissä (fully connected) neuroverkkokerroksissa jokainen kerroksen neuroni on yhdistetty jokaiseen sitä edeltävän kerroksen neuroniin. Myös



Kuva 4: Tyypillinen neuroverkon rakenne, jossa kaksi piilokerrosta [10]

muita tapoja yhdistää kerrokset käytetään, kuten esimerkiksi kappaleessa 4.1 esiteltävissä konvoluutiokerroksissa.

Neuroverkoissa on aina ainakin syöte- ja ulostulokerros, sekä vaihteleva määrä niiden välissä olevia piilokerroksia. Eteenpäinsyöttävissä neuroverkoissa ei ole silmukoita, eli informaatio kulkee niissä aina vain yhteen suuntaan. Silmukoita sisältäviä neuroverkkorakenteita hyödynnetään esimerkiksi takaisinkytkeytyvissä neuroverkoissa [TODOcite], mutta ne eivät kuulu tämän tutkielman aihepiiriin.

Kuvassa 4 vasemmanpuoleisimpana nähdään syötekerros. Esimerkiksi haluttaessa syöttää 64x64 kuva neuroverkolle voidaan syötekerroksena käyttää 64x64 neuronin kerrosta, johon kuvan pikselien väriarvot koodataan. Neuroverkon laskennan lopputuloksena toimii verkon viimeisen kerroksen neuroneiden ulostuloarvot.

Kasvattamalla piilokerroksien sekä kerroksissa olevien neuronien määrää, neuroverkoilla voidaan mallintaa entistä monimutkaisempia funktioita. Vaikka syväoppimista voidaan harjoittaa myös muutoin kuin keinotekoisilla neuroverkoilla, neuroverkkojen tapauksessa termillä viitataan neuroverkkojen piilokerrosten määrään.

3 Neuroverkkojen harjoittaminen

Eteenpäinsyöttävien neuroverkkojen, joissa on yksi piilokerros, on todistettu olevan universaaleja approksimaattoreita [5]. Eteenpäinsyöttävät neuroverkot pystyvät siis approksimoimaan mielivaltaisella tarkkuudella mitä tahansa avaruuden R^n kompakteilla osajoukoilla määriteltyjä jatkuvia funktioita, kunhan tarpeeksi laskentaresursseja on käytettävissä.

Suurin osa neuroverkkojen ja ylipäätään koneoppimisen sovelluksista hyödyntää mallien harjoittamisessa ohjattua oppimista [7]. Ohjatussa oppimises-

sa harjoitusaineisto on luokiteltu jollakin perusteella ja harjoittamisvaiheessa pyritään saamaan opetettava malli löytämään näille ennakkoon määritellyille luokille yhteisiä ja harjoitusaineiston ulkopuolisiin syötteisiin yleistettävissä olevia tekijöitä.

Tyypillinen neuroverkkojen harjoittamiseen käytetty harjoitusaineisto on suuri joukko pareiksi järjestettyjä yksiköitä (x_i, z_i) jossa x_i on jokin syötevektori ja z_i toivottu ulostulovektori tälle syötteelle. Esimerkiksi käsin kirjoitettujen numeroiden kuvista tunnistamista varten tehdyssä harjoitusaineistossa i voi olla vektori kuvan pikselien värit määritteleviä arvoja ja z_i vektori arvoja, jossa vain kuvasta löytyvää numeroa vastaava arvo on asetettu nollasta poikkeavaksi [10].

Laskemalla neuroverkolla harjoitusaineistosta peräisin olevalle syötteelle ulostuloarvo, voidaan saatua ulostuloarvoa verrata harjoitusaineistosta löytyvään toivottuun ulostuloarvoon. Näin neuroverkolle voidaan laskea sen tekemä virhe, ja kun virhe tunnetaan, sitä voidaan pyrkiä pienentämään. Neuroverkkojen harjoittamisen voidaan siis sanoa olevan neuroverkon tekemän virheen minimointia sen approksimoidessa jotakin funktiota.

Neuroverkkojen tekemää virhettä mitataan tyypillisesti virhefunktion (error function) avulla. Virhettä halutaan minimoida koko harjoitusaineiston suhteen, joten minimoitava virhefunktio koostetaan useista harjoitusaineiston yksiköistä saatujen virheiden summasta. Usein harjoitusaineistot kuitenkin koostuvat miljoonista yksiköistä jolloin ongelmaksi muodostuu tehokkuus, joten käytännössä yleensä virhe lasketaan vain osalle harjoitusaineistosta.

Eräs hyvin yleisesti käytetty virhefunktio on neliöllinen virhefunktio

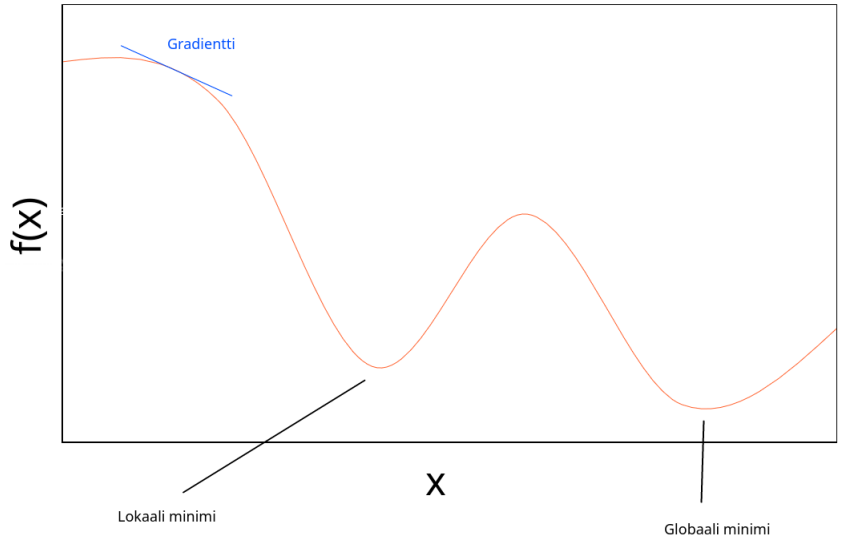
$$E(x) = \frac{1}{2} \sum_{i=1}^{N} \|y(x_i) - z_i\|^2, \tag{3}$$

jossa $y(x_i)$ on neuroverkon tuottama tulos jollakin harjoitusaineiston yksikön syötteellä x_i, z_i on harjoitusaineiston perusteella syötteelle x_i toivottu ulostuloarvo, ja N syötteiden määrä.

Harjoitusvaiheessa ainoat neuroverkossa muutettavissa olevat parametrit ovat sen neuroneiden syötteiden painotusarvot sekä neuroneiden taipumusvakiot. Seuraavaksi esitellään gradienttimenetelmä ja takaisinvirtausalgoritmi, joiden avulla voidaan selvittää, kuinka suuri vaikutus kullakin verkon painolla on koko verkon tuottamaan virheeseen ja pienentää tätä virhettä gradienttimenetelmän mukaisesti.

3.1 Gradienttimenetelmä

Gradientti on derivaatan yleistys useamman kuin yhden muuttujan funktioille, joka kertoo mihin suuntaan funktion arvo kasvaa nopeimmin. Gradienttimenetelmällä (gradient descent) tarkoitetaan numeerista menetelmää, jossa kuljetaan iteratiivisesti negatiivisen gradientin suuntaan kunnes gradientti on tarpeeksi pieni.



Kuva 5: Gradientti sekä funktion lokaali ja globaali minimi

Funktion on oltava derivoituva, jotta sille voidaan muodostaa gradientti, joten myös gradienttimenetelmä edellyttää funktioiden derivoituvuutta. Käytännössä kuitenkin riittää, että aktivaatiofunktiot ovat suurimmilta osin derivoituvia funktioita. Esimerkiksi kappaleessa 2.1 esitelty ReLU $f(x) = max(0, x)$ ei ole derivoituva kohdassa $x = 0$, mutta sovelluksissa tämä voidaan kiertää vain valitsemalla aktivaatiofunktion derivaataksi tapauksessa $x = 0$ jokin kiinteä arvo.

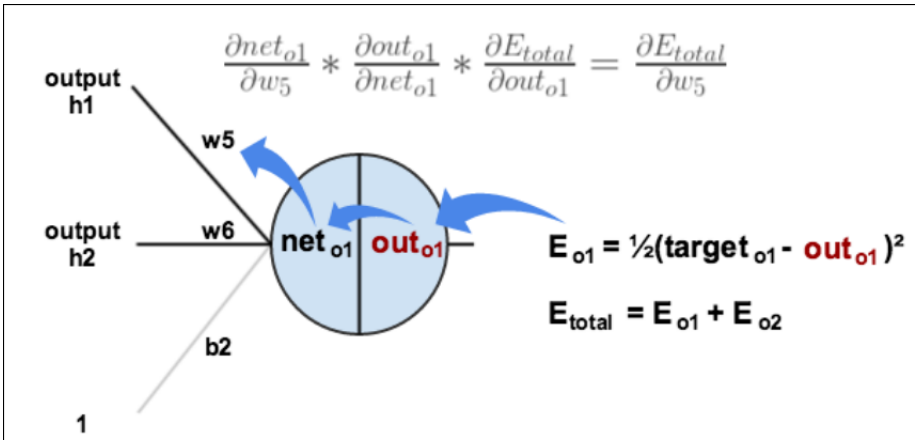
Vaikka gradienttimenetelmä soveltuukin vain lokaalien minimien etsintään, sen ja sen muunnelmien on huomattu olevan yleensä riittävän toimiva ratkaisu neuroverkkojen harjoittamiseen [12][2]. Toistaiseksi käytännön neuroverkkototeutuksissa on todettu olevan tärkeämpää etsiä parametriavaruudesta sellaisia lokaaleita minimeitä, joissa virhefunktio saa pieniä arvoja, kuin etsiä funktion todellista globaalia minimiä [3]. Tämä on kuitenkin vielä aktiivisen tutkimuksen aluetta, josta ei ole varmaa tietoa [2].

3.2 Takaisinvirtausalgoritmi

Takaisinvirtausalgoritmi on ohjatun oppimisen algoritmi neuroverkkojen harjoittamiseen gradienttimenetelmää hyödyntämällä. Takaisinvirtausalgoritmissa lasketaan ensin virhefunktiolle osittaisderivaatta kunkin verkon painon suhteen. Osittaisderivaatoista muodostetaan virhefunktiolle gradientti ja sovelletaan siihen gradienttimenetelmää. Virhefunktion osittaisderivaatan selvittämiseksi yksittäisen neuronien n_i ja n_j välillä olevan verkon painon w_{ij} suhteen tarvitaan ensin välituloksena virhefunktion osittaisderivaatta neuronin n_j kohdalla.

TODO miten selvitetään virheen määrä 1) ulostulokerroksen neuronin kohdalla 2) piilokerroksen neuronin kohdalla.

Muiden ulostulokerroksen neuroneiden syötteiden painotukset voidaan laskea vastaavasti. Piilokerroksen painojen aiheuttaman virheen pienentäminen tehdään muuten vastaavasti, mutta laskettaessa kokonaisvirheen muutosta piilokerroksen neuroneiden ulostuloarvojen suhteen, on otettava huomioon piilokerroksen neuroneiden ulostuloarvojen vaikutus useiden ulostulokerroksen neuroneiden tekemään virheeseen.



Kuva 6: TODO piirrä oma kuva, atm: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Yksittäisen painotuksen vaikutus koko verkon virheeseen voidaan laskea ketjusäännön avulla

$$\frac{\delta E_{koko}}{\delta w_5} = \frac{\delta E_{koko}}{\delta ulos_{o1}} * \frac{\delta ulos_{o1}}{\delta summa_{o1}} * \frac{\delta summa_{o1}}{\delta w_5}.$$

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, ja ajetaan takaisinvirtausalgoritmi tämän tuloksen perusteella.

3.3 Oppimistahti ja aloituspainot

alkup. backprop paperissa on symmetry breakingista ... Virhettä korjataan yleensä jonkin oppimistahdin (learning rate) mukaisesti, jota usein muutetaan neuroverkon harjoittamisen aikana.

3.4 Ylisovitus ja sen ratkaiseminen

Suuri haaste neuroverkkojen harjoittamisessa on ylisovitus (overfitting), jossa neuroverkon harjoittamisen jälkeen neuroverkko saa harjoitusaineistosta valituille syötteille pieniä arvoja virhefunktiosta, mutta uuden aineiston kanssa virhefunktio tuottaa suuria arvoja. Tällöin neuroverkon oppima malli vastaa harjoitusaineistoa liian tarkkaan, eikä pysty yleistämään harjoitusaineistosta löytyviä ominaisuuksia aineiston ulkopuolisille syötteille. Ratkaisuksi ylisovitukseen on kehitetty lukuisia menetelmiä.

Neuroniyksikköjen pudotus (dropout) on menetelmä, jossa harjoitusvaiheessa yksittäisiä neuroneita poistetaan satunnaisesti käytöstä, jolloin yksittäiset neuronit naapureineen eivät erikoistu tiettyihin aineiston ominaisuuksiin liian tarkasti [14].

Regularisaation selitys, yleisesti kaikessa koneoppimisessa oleellista

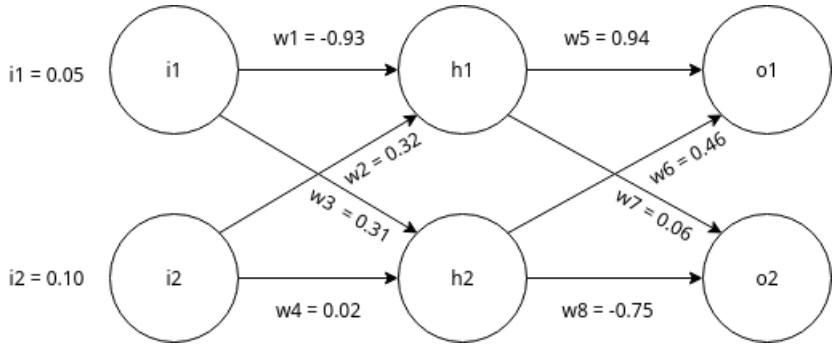
L2 ja muut regularisaatiot

L2 regularisaatiolla saadaan verkko suosimaan pieniä painotusarvoja.

TODO:Harjoitusaineiston laajentaminen kuvien kääntäminen, siirtäminen, osakuvat, skaalaus, noisen lisääminen ...? Harjoitusaineiston laajentaminen muussa kuin kuvien tunnistamisessa? Puheentunnistusmateriaaliin noisea ja kolinaa taustalle etc?

3.5 Esimerkki harjoittamisesta

Lasketaan ensin neuroverkon antama tulos esimerkkisyötteelle satunnaisesti alustetuin painotuksin ja selvitetään sen jälkeen takaisinvirtausalgoritmin avulla yhdelle verkon painotukselle uusi arvo, joka vähentää verkon tekemää virhettä esimerkin harjoitusaineiston yksiköllä. Esimerkin neuroverkko on eteenpäinsyöttävä, siinä ei ole taipumusvakioita, ja siinä on kolme kerrosta: syötekerros, täysin yhdistetty piilokerros, sekä täysin yhdistetty ulostulokerros. Kussakin kerroksessa on 2 neuronia. Oppimistahtina käytetään mielivaltaisesti valittua arvoa $\eta = 0.5$. Aktivaatiofunktiona käytetään lausekkeessa 1 esiteltyä sigmoidista funktiota $\sigma(x)$.



Kuva 7: Esimerkin neuroverkon syötteet ja painotukset

Lasketaan ensin verkon tuottamat ulostuloarvot syötteillä $i_1 = 0,05$ ja $i_2 = 0,10$ sekä väliltä $[-1, 1]$ arvotuilla painotuksilla $w_1 ... w_8$. Piilokerroksen neuroneiden syötteiden summaus tehdään kappaleessa 2.1 esitetyn kaavan $\Sigma x_i w_i$ mukaisesti

$$h_1^{in} = w_1 \cdot i_1 + w_2 \cdot i_2 \approx -0.0145$$
$$h_2^{in} = w_3 \cdot i_1 + w_4 \cdot i_2 \approx 0.0175$$

Syöttämällä nämä arvot sigmoidiseen aktivaatiofunktioon saadaan

$$h_1^{out} = \frac{1}{1 + e^{-h_1^{in}}} \approx 0.496$$

$$h_2^{out} = \frac{1}{1 + e^{-h_2^{in}}} \approx 0.504.$$

Toistaen vastaavat vaiheet ulostulokerrokselle käyttäen piilokerroksen ulostuloarvoja syötteenä saadaan:

$$o_1^{in} = w_5 \cdot h_1 + w_6 \cdot h_2 \approx 0.628$$

$$o_2^{in} = w_7 \cdot h_1 + w_8 \cdot h_2 \approx -0.348$$

$$o_1 out = \frac{1}{1 + e^{-o_1^{in}}} \approx 0.652$$
$$o_2 out = \frac{1}{1 + e^{-o_2^{in}}} \approx 0.414.$$

Kun harjoitusaineistoksi valitaan arvot $z_1^{out} = 0.01$ ja $z_2^{out} = 0.99$ saadaan kaavan 3 virhefunktiosta

$$\frac{1}{2} \cdot ((o_1^{out} - z_1^{out})^2 + (o_2^{out} - z_2^{out})^2) \approx 0.372.$$

Lasketaan painon w_5 vaikutus virheeseen. E_{koko} koostuu ulostulokerroksen neuroneiden tekemien virheiden summista, joten kun se derivoidaan tietyn neuronin ulostulon suhteen, muiden neuroneiden virheiden termit putoavat pois. Jäljelle jää

$$\frac{\delta E_{koko}}{\delta o_1^{out}} = 2 \cdot \frac{1}{2}(o_1^{out} - z_1^{out})^{2-1} \cdot 1 = o_1^{out} - z_1^{out} = 0.651$$

Aktivaatiofunktion $\sigma(x)$ derivaatta on $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$. Koska

$$o_1^{out} = \frac{1}{1 + e^{-o_1^{summa}}}$$

niin

$$\frac{\delta o_1^{out}}{\delta o_1^{summa}} = o_1^{out} \cdot (1 - o_1^{out}) \approx 0.227.$$

$$o_1^{summa} = w5 \cdot h_1^{out} + w_6 \cdot h_2^{out}$$

joten viimeinen tarvittava osa

$$\frac{\delta o_1^{summa}}{\delta w_5} = 1 \cdot h_1^{out} + w_5 \cdot 0 + 0 = out_{h1}$$

joten

$$\frac{\delta E_{koko}}{\delta w_5} = 0.651 \cdot 0.227 \cdot 0.496 \approx 0.073.$$

Korjaamme nyt w_5 virhettä asettamalla sen uudeksi arvoksi

$$w_5^+ = w_5 - \eta \cdot \frac{\delta E_{koko}}{\delta w_5} = 0.9035.$$

4 Konvolutionaalisten neuroverkkojen rakenne

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

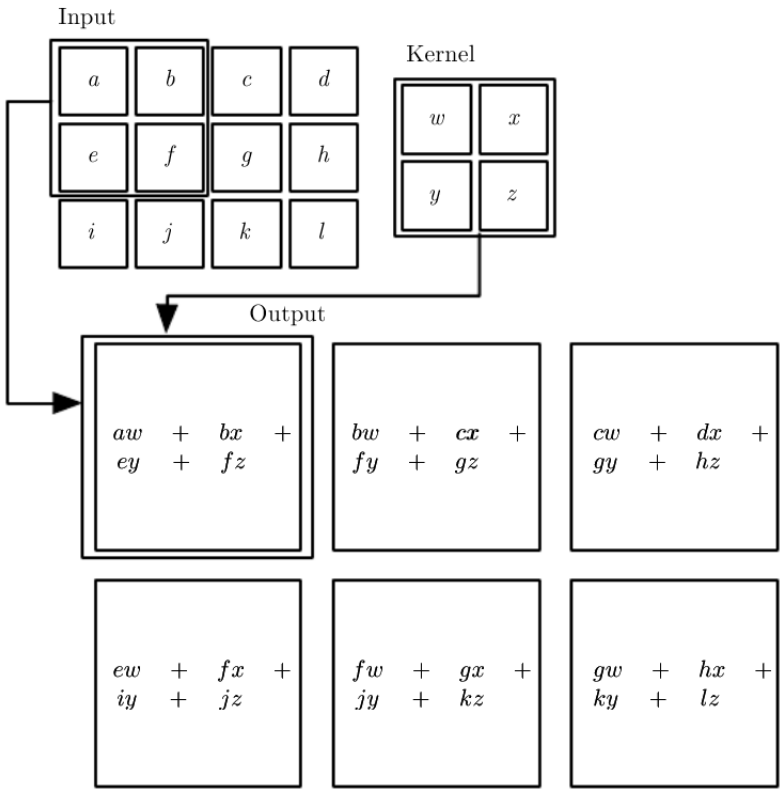
4.1 Konvoluutiokerrokset

Konvoluutioverkkojen nimi juontaa juurensa matemaattiseen konvoluutioon, 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).

Konvoluution voidaan ajatella olevan liukuva ikkuna, joka rajoittaa neuronit saamaan kuvan 8 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.



Kuva 8: Konvoluutiokerrosten syötteiden valinta ja ulostulojen muodostuminen [2]

Kuvasta 8 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

TODO: lisää kuvia yms. siitä mitä ydin on jne.

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 translationaalisesti invariantteja ominaisuuksia syötteistään, ja yksittäisen neuronin voidaan ajatella kertovan löytyykö sen saamien syötteiden alueelta tätä ominaisuutta. Esimerkiksi ku-

vien tapauksessa ominaisuuskartta joka löytää kuvasta suoria viivoja saattaa olla hyödyllinen, sillä suoria viivoja voi ilmetä useassa eri paikassa kuvassa, ja toisaalta korkeammalla abstraktiotasolla kuvassa esiintyvä objekti on edelleen sama objekti, vaikka sitä olisi siirretty muutamia pikseleitä johonkin suuntaan.

4.3 Kokoaminen ja tehokkuus

Usein konvoluutioverkoissa käytetään konvoluutiokerroksien jälkeen kokoamiskerroksia, jotka tekevät yhteenvedon jostakin edeltävästä neuroverkkokerroksen alueesta. Esimerkiksi hyvin yleisen maksimikokoamiskerroksen (max-pooling) neuronit antavat ulostulokseen syöteneuroniensa ulostuloista suurimman.

Yhdessä kokoamiskerrokset ja jaetut painotukset vähentävät merkittävästi konvolutionaalisten neuroverkkojen harjoitettavien parametrien määrää verrattuna neuroverkkoihin, joissa on vain täysin yhdistettyjä kerroksia. Tällä voidaan nopeuttaa harjoittamista ja lisätä suorituskykyä.

4.4 Täysin yhdistetyt kerrokset

Konvoluutiokerrosten kohdalla konvolutionaaliset neuroverkot haarautuvat yleensä rinnakkaisiin kerroksiin. Sijoittamalla verkon viimeiseksi kerrokseksi täysin yhdistetyn kerroksen nämä rinnakkaiset kerrokset voidaan litistää (flattening) takaisin yhteen.

5 Konvolutionaaliset neuroverkot käytännössä

5.1 Kirjastot

Johonkin saisi ehkä pari kappaletta tekstiä (konvolutionaalisten) neuroverkkojen luomista helpottavista kirjastoista yms. kuten TensorFlow.

5.2 Verkon toiminnan visualisointi

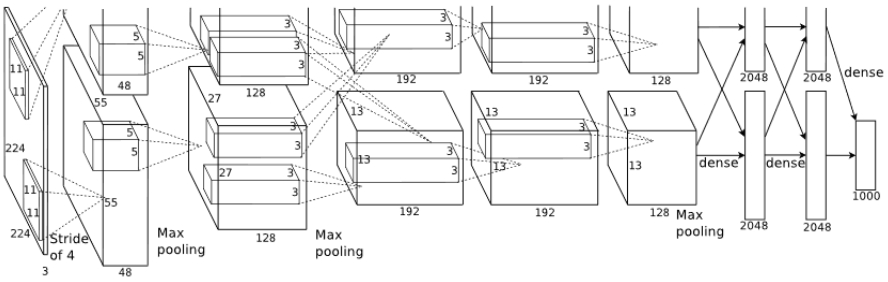
... [16]

5.3 Kuvien luokittelu

Konvoluutioverkkoja on käytetty erityisen onnistuneesti kuvien sisällön luokitteluun. Monille kuvia tunnistaville neuroverkoille yhteinen piirre on, että ulostulokerroksessa on yksi neuroni jokaista tunnistettavaa objektiluokkaa kohden, ja neuronin arvo kertoo kuinka todennäköisesti kyseisen luokan objekti löytyy kuvasta.

TODO: Softmax

TODO: Imagenet kilpailun kuvailu? Pitäisikö olla oma kokonainen section kuvien luokittelulle ja sen subsectionina imagenet ja CNN:ien toiminnan visualisointi. Esimerkkikuvia imagenet aineistosta?



Kuva 9: ILSVRC-2012 kilpailun voittaneen neuroverkon rakenne [6]

Vuonna 2012 ImageNet kuvantunnistuskilpailun voittaneessa Krizhevsky, Sutskever ja Hintonin (myöhemmin KSH) konvolutionaalisessa neuroverkossa on 7 piilokerrosta, joista 5 ensimmäistä ovat konvoluutiokerroksia, ja 2 viimeistä kerrosta täysin yhdistettyjä kerroksia. KSH:ssa on 1000 neuronin ulostulokerros joka vastaa sen tunnistamaa tuhatta erilaista kuvaluokkaa.

ImageNet kuvamateriaalissa on vaihtelevan kokoisia kuvia, mutta KSH:n luomassa neuroverkon syötekerros oli 3x224x224 neuronin kokoinen. KSH:n ratkaisu syötteen sopivaksi saamiseksi oli skaalata lähdekuvat ensin 256x256 pikselin kokoon, ja tämän jälkeen ottaa kuvista 224x224 osakuvia satunnaisista sijainneista lähdekuvasta, näin laajentaen harjoitusdataa ja vähentäen ylisovitusta.

KSH:n verkossa aktivaatiofunktiona toimi viimeaikoina hyväksi havaittu $R(z) = max(0, z)$ muotoinen Rectified Linear Unitiksi (ReLU) kutsuttu funktio [6].

6 Yhteenveto

Kappale siitä, että sitä ei vielä täysin ymmärretä, miksi neuroverkot oppivat niin hyvin. Täs vois mainita [3] :sta siitä abstraktissa mainitusta jutusta et pitkään pelättiin et neuroverkkojen treenaus ois super vaikeeta

Aloitettiin käymällä läpi kaikille neuroverkoille yleisiä piirteitä, kuten yksittäisen neuronin rakennetta ja sitä, miten yksittäisistä neuroneista muodostetaan eteenpäinsyöttävä neuroverkko. Seuraavaksi esiteltiin kuinka neuroverkkoja voidaan harjoittaa harjoitusaineiston perusteella takaisinvirtausalgoritmin ja gradienttimenetelmän avulla. Lopuksi käsiteltiin konvolutionaalisten neuroverkkojen eroavaisuuksia muista neuroverkoista ja esiteltiin käytännön esimerkki konvolutionaalisesta neuroverkosta.

Käsittelemättä jäi miten aktivaatiofunktioita, hyperparametrejä yms. valitaan järkevästi/hyvin (myös koska niistä ei juuri ymmärretä ja hyvin

vaikea tehtävä). Muista neuroverkkorakenteista kuin eteenpäinsyöttävistä konvoluutioverkoista ei kerrottu, esim RNN, muut jotku poolaukset, jne

Lähteet

[1] Ossama Abdel-Hamid, Abdel rahman Mohamed, Hui Jiang ja Gerald Penn: *Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition.* Teoksessa *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, sivut 4277–4280. IEEE, 2012.

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

[3] Ian J. Goodfellow ja Oriol Vinyals: *Qualitatively characterizing neural network optimization problems.* CoRR, abs/1412.6544, 2014. `http://arxiv.org/abs/1412.6544`.

[4] Geoffrey E. Hinton, Simon Osindero ja Yee Whye Teh: *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`.

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

[6] Alex Krizhevsky, Ilya Sutskever ja Geoffrey E. Hinton: *ImageNet Classification with Deep Convolutional Neural Networks.* Teoksessa *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, sivut 1106–1114, 2012. `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks`.

[7] Yann LeCun, Yoshua Bengio ja Geoffrey E. Hinton: *Deep learning.* Nature, 521(7553):436–444, 2015. `https://doi.org/10.1038/nature14539`.

[8] Warren S McCulloch ja Walter Pitts: *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 5(4):115–133, 1943.

[9] TODO nettisivu ei toimi atm tarkista authoriin jotain myöhemmin: *ImageNet*, 2018. `http://image-net.org/about-stats`, vierailtu 2018-05-1.

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

[11] Frank Rosenblatt: *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[12] D. E. Rumelhart, G. E. Hinton ja R. J. Williams: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*. luku Learning Internal Representations by Error Propagation, sivut 318–362. MIT Press, Cambridge, MA, USA, 1986, ISBN 0-262-68053-X. http://dl.acm.org/citation.cfm?id=104279.104293.

[13] Pierre Sermanet ja Yann LeCun: *Traffic sign recognition with multi-scale convolutional networks*. Teoksessa *Neural Networks (IJCNN), The 2011 International Joint Conference on*, sivut 2809–2813. IEEE, 2011.

[14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever ja Ruslan Salakhutdinov: *Dropout: a simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, 15(1):1929–1958, 2014. http://dl.acm.org/citation.cfm?id=2670313.

[15] Mate Szarvas, Akira Yoshizawa, Munetaka Yamamoto ja Jun Ogata: *Pedestrian detection with convolutional neural networks*. Teoksessa *Intelligent vehicles symposium, 2005. Proceedings. IEEE*, sivut 224–229. IEEE, 2005.

[16] Matthew D. Zeiler ja Rob Fergus: *Visualizing and Understanding Convolutional Networks*. CoRR, abs/1311.2901, 2013. http://arxiv.org/abs/1311.2901.