Konvolutionaaliset neuroverkot

Teemu Sarapisto

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 2. 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	2. toukokuuta 2018	14

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, 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 [6]. 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 [10].

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 [11].

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 [3]. 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 [5]. 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 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. Selkeästi suurin syy syväoppimisen vasta äskettäin tärkeäksi muuttumiseen on kuitenkin yhteiskunnan digitalisoitumisen myötä merkittävästi kasvanut helposti saatavilla olevan luokitellun datan määrä, sekä valtavasti kasvanut laskentakapasiteetti, jotka olivat edellytyksiä algoritmien mielekkäälle käyttämiselle [2].

Esimerkkinä tarvittavan harjoitusdatan määrästä konvolutionaalisten neuroverkkojen harjoittamiseen kuvien luokittelua varten toimii yleensä tu-

hansia ellei jopa miljoonia kuvien sisällön perusteella etukäteen luokiteltuja kuvia. Esimerkiksi ILSVRC-kilpailun harjoitusdatana käytössä oleva ImageNet sisältää yli 14 miljoonaa luokiteltua kuvaa [7]. Kuvien sisällön tunnistamisen lisäksi syväoppiminen on osoittautunut erittäin hyödylliseksi useissa aikaisemmin haasteellisiksi osoittautuneissa sovelluskohteissa, kuten puheentunnistuksessa [1], liikennemerkkien luokittelussa [12], sekä jalankulkijoiden tunnistamisessa [14].

Ensin käydään läpi yleisiä keinotekoisten neuroverkkojen piirteitä, kuten yksittäisten neuronien ja eteenpäinsyöttävien neuroverkkojen rakennetta. Tämän jälkeen selitetään kuinka neuroverkkoja harjoitetaan harjoitusaineiston avulla takaisinvirtausalgoritmia ja gradienttimenetelmää hyödyntäen. Lopuksi käydään läpi miten konvolutionaaliset neuroverkot eroavat muista neuroverkoista ja mitä sovelluksia niille löytyy.

2 Neuroverkkojen rakenne

2.1 Keinotekoinen neuroni

TODO: Pitäisikö tässä tai johdannon lopussa mainita, että rakenneosiot ovat Neural Networks and Deep Learning kirjasta [8] peräisin ja ei lisätä citeä joka lauseen loppuun?

Biologisista vaikuttimistaan huolimatta keinotekoiset neuronit ovat käytännössä (TODO: selitä pelkästään tekstimuodossa re: palaute)

$$x_1, x_2, ..., x_n \mapsto \Sigma w_i x_i \mapsto f(\Sigma w_i x_i + b)$$

kaltaisia matemaattisia funktioita. Ne ottavat 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, tarkoituksena säätää neuronin ulostuloarvoja painotuksista riippumatta [8].

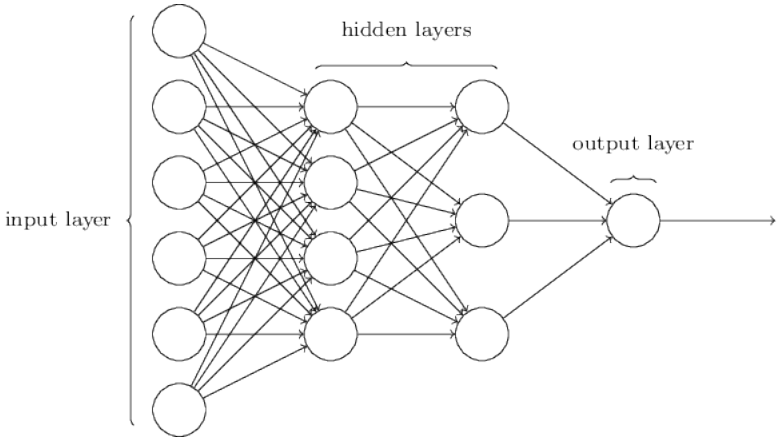
50-luvulla kehitetyn perseptronin, joka oli ensimmäinen tällainen neuroni, syötteet ja ulostuloarvot ovat binäärisiä ja aktivaatiofunktio muotoa

$$f(\Sigma w_i x_i + b) \begin{cases} 0 \text{ jos } \Sigma x_i w_i + b \leq 0 \\ 1 \text{ jos } \Sigma x_i w_i + b > 0. \end{cases} \tag{1}$$

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



Kuva 1: Tyypillinen neuroverkon rakenne, jossa kaksi piilokerrosta [8]

Yksinkertaisimman verkkorakenteen omaavat eteenpäinsyöttävät neuroverkot muodostetaan tasoittain, jossa jokaisen verkon tason neuronit saavat 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. Neuroverkon laskennan lopputuloksena toimii verkon viimeisen tason neuroneiden ulostuloarvot.

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

Eteenpäinsyöttävien neuroverkkojen joissa on yksi piilokerros jossa on tarpeeksi neuroneita on todistettu pystyvän mallintamaan mitä tahansa avaruuden R^n kompakteilla osajoukoilla määriteltyjä jatkuvia funktioita, kunhan tarpeeksi laskentaresursseja on käytettävissä [4]. Neuroverkkojen harjoittamisen voidaan 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

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

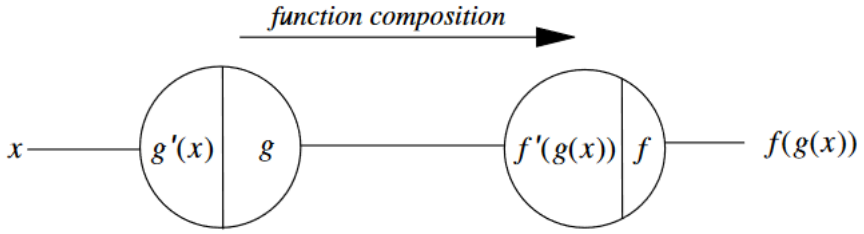
jossa $y(x_i)$ on neuroverkon tulos syötteellä x_i, z_i on jonkin harjoitusaineiston yksikön i:s jäsen, ja N syötteiden määrä.

3.1 Gradienttimenetelmä ja takaisinvirtausalgoritmi

Gradienttimenetelmä (gradient descent) on numeerinen menetelmä joka toimii minkä tahansa derivoituvan funktion lokaalien minimien etsintään. Käytännössä kuitenkin yleensä riittää, että funktiot ovat suurimmilta osin derivoituvia. Neuroverkkojen yhteydessä gradienttimenetelmä toimii valittaessa neuronien aktivaatiofunktioksi pääosin 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 lokaalia minimiä.

Takaisinvirtausalgoritmilla voidaan laskea virhefunktiolle osittaisderivaatta kunkin verkon painon suhteen. Osittaisderivaatoista voidaan muodostaa virhefunktiolle gradientti ja siten soveltaa 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.

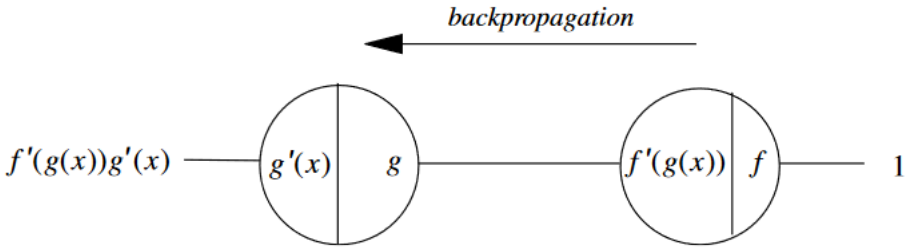
Neuroverkon voidaan ajatella olevan yhdistetty funktio sen neuroneiden funktioista, joten derivoimisessa voidaan soveltaa ketjusääntöä. Jotta virhefunktion arvot saadaan osaksi verkkoa, lisätään ulostulokerroksen jälkeen yksi näennäinen neuroni, joka ottaa ulostulokerroksen neuroneiden ulostulot syötteenään, ja laskee niiden perusteella virhefunktion arvon. Nyt virhefunktion derivaattojen laskentaan voidaan hyödyntää seuraavaksi esitettävää verkkomuotoista derivaatan ketjusäännön toteutusta.



Kuva 2: Ketjusääntö ja eteenpäinvirtaus kahden solmun verkossa [9]

Ketjusäännön mukainen derivointi suoritetaan kahdessa osassa, käyden verkko ensin läpi normaaliin suuntaan syötteistä ulostuloarvojen kautta virhefunktion arvoon, ja sen jälkeen takaperin. Takaisinvirtausalgoritmi on saanut nimensä tästä taaksepäin kulkemisesta, jossa virhefunktion arvojen voidaan ajatella virtaavan taaksepäin verkon loppupäästä.

Eteenpäinsyöttövaiheessa jokaisen solmun kohdalla tallennetaan solmun toteuttaman funktion derivaatta. Kuvassa 2 tämä tallennettava arvo näkyy palloina esitettyjen solmujen vasemmassa puoliskossa.



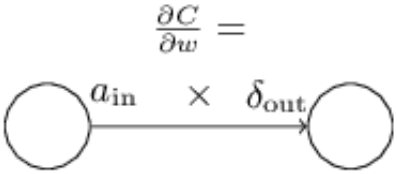
Kuva 3: Ketjusääntö ja takaisinvirtaus kahden solmun verkossa [9]

Takaisinvirtausvaiheessa verkkoa kuljetaan takaperin aikaisempaan nähden. Takaisinvirtaus aloitetaan syöttämällä verkkoon numero yksi, kuljettaen tätä arvoa mukana solmulta toiselle, ja uuteen solmuun saavuttaessa kertomalla mukana kulkeva arvo solmuun tallennetulla, kuvassa 3 solmun vasemmalla puoliskolla näkyvällä arvolla. Näin lopulta esimerkin kahden solmun läpi kulkemisen jälkeen yhdistetty funktio $f(g(x))$ on saatu muotoon $f'(g(x))g'(x)$, joka on funktion $f(g(x))$ derivaatta.

Kuvaamalla virhefunktion derivaattaa neuronin n_j kohdalla merkinnällä δ_j voidaan virhefunktion derivaatta suhteessa painoon w_{ij} esittää muodossa

$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j,$$

jossa o_i on neuronin n_i ulostuloarvo.



Kuva 4: Virhefunktion derivaatta yksittäisen painon suhteen [8]. TODO: piirrä oma kuva omilla merkinnöillä

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.

3.2 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. Ylisovitusta korjaamaan on kehitetty lukuisia menetelmiä, kuten esimerkiksi neuroniyksikköjen pudotus (dropout), jossa harjoitusvaiheessa yksittäisiä neuroneita poistetaan satunnaisesti käytöstä, jolloin yksittäiset neuronit naapureineen eivät erikoistu tiettyihin aineiston ominaisuuksiin liian tarkasti [13].

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?

3.3 Aktivaatiofunktion, aloituspainojen, hyperparametrien, yms. valitseminen?

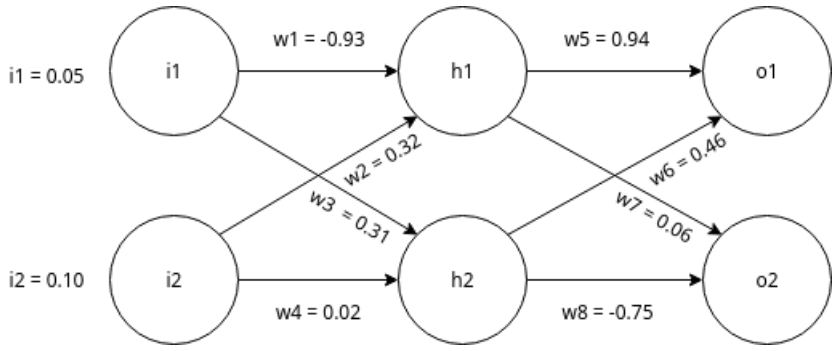
oppimistahti ja sen muuttuminen, regularisaatioparametri jne valitseminen [8] ...

3.4 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 taipumusarvoja, ja siinä on kolme kerrosta: syötekerros, täysin yhdistetty piilokerros, sekä täysin yhdistetty ulostulokerros. Kussakin kerroksessa on 2 neuronia. Aktivaatiofunktiona käytetään sigmoidista funktiota

$$\frac{1}{1+e^{-x}}$$

TODO: jossain voisi käsitellä miksi sigmoidinen, ja olla kuva sigmoidisesta funktioista?.



Kuva 5: 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

$$h_1 syöte = w_1 * i_1 + w_2 * i_2$$

$$h_1 syöte = -0,93 * 0,05 + 0,32 * 0,10 = -0,0145$$

$$h_2 syöte = w_3 * i_1 + w_4 * i_2$$

$$h_2 syöte = 0,31 * 0,05 + 0,02 * 0,10 = 0,0175$$

Syöttämällä nämä arvot sigmoidiseen aktivaatiofunktioon saadaan

$$h_1 ulos = \frac{1}{1 + e^{-(-0,0145)}} \approx 0,496$$

$$h_2 ulos = \frac{1}{1 + e^{-0,0175}} \approx 0,504.$$

Toistaen vastaavat vaiheet ulostulokerrokselle käyttäen piilokerroksen ulostuloarvoja syötteenä saadaan:

$$o_1 syöte = w_5 * h_1 + w_6 * h_2$$

$$o_1 syöte = 0,94 * 0,496 + 0,32 * 0,504 \approx 0,628$$

$$o_2 syöte = w_7 * h_1 + w_8 * h_2$$

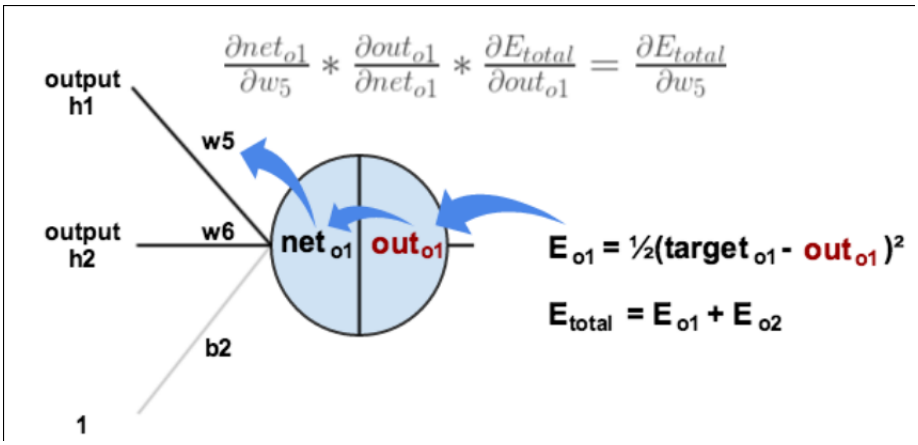
$$o_2 syöte = 0,06 * 0,496 + (-0,75) * 0,504 \approx -0,348$$

$$o_1 ulos = \frac{1}{1 + e^{-0.627}} \approx 0,652$$

$$o_2 ulos = \frac{1}{1 + e^{-(-0.348)}} \approx 0,414.$$

Kun harjoitusaineistoksi valitaan arvot $o_{harjoitus1} = 0.01$ ja $o_{harjoitus2} = 0.99$ saadaan kaavan 2 mukaisesta virhefunktiosta arvoksi

$$0,5 * ((0,652 - 0,01)^2 + (0,414 - 0,99)^2) = 0,372.$$



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

Taaksepäinkulkeutumisvaiheessa painon vaikutus koko verkon virheeseen voidaan laskea ketjusäännön avulla

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

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

$$\frac{\delta E_{koko}}{\delta ulos_{o1}} = 2 * \frac{1}{2}(ulos_{o1} - harjoitus_{o1})^{2-1} * 1 = ulos_{o1} - harjoitus_{o1} = 0.652 - 0.01 = 0.651$$

Sigmoidisen aktivaatiofunktion $f(x) = \frac{1}{1+e^{-x}}$ derivaatta on $f'(x) = f(x)(1 - f(x))$. Koska $ulos_{o1} = \frac{1}{1+e^{-net_{o1}}}$ niin

$$\frac{\delta ulos_{o1}}{\delta summa_{o1}} = ulos_{o1}(1 - ulos_{o1}) = 0.652(1 - 0.652) \approx 0.227.$$

$summa_{o1} = w5 * out_{h1} + w_6 * out_{h2}$ joten viimeinen tarvittava osa

$$\frac{\delta summa_{o1}}{\delta w_5} = 1 * out_{h1} + w_5 * 0 + 0 = out_{h1}$$

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

Virhettä korjataan yleensä jonkin oppimistahdin (learning rate) mukaisesti, jota usein muutetaan neuroverkon harjoittamisen aikana. Tässä esimerkissä käytetään mielivaltaisesti valittua $\eta = 0.5$ oppimistahtia. Korjaamme nyt w_5 virhettä asettamalla sen uudeksi arvoksi

$$w_5^+ = w_5 - \eta * \frac{\delta E_{koko}}{\delta w_5} = 0.94 - 0.5 * 0.073 = 0.9035.$$

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.

TODO: Pitäisikö laskea neuroverkolle uusi ulostulo kun yksi verkon paino muutettu? Vai laskea kaikille painoille muutokset ja sen jälkeen ulostulo?

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 ReLU

...

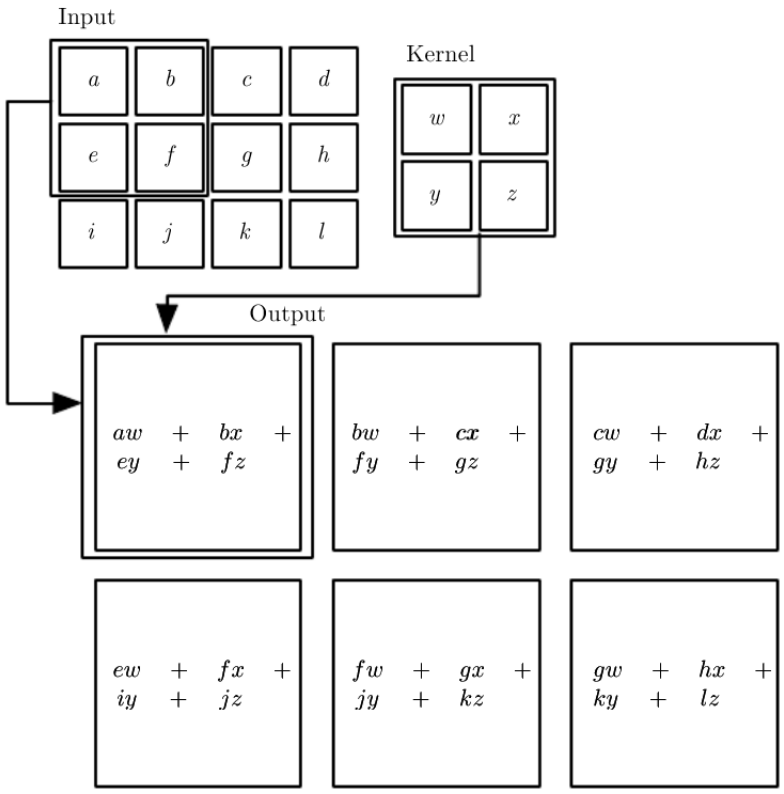
4.2 Konvoluutiokerrokset

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 7 mukaisesti syötteenään vain osan syötekerroksensa ulostuloista, täysin yhdistetyistä neuroverkkokerroksista poiketen. Tämä rajoittuneisuus mahdollistaa neuroneiden erikoistumisen johonkin osa-alueeseen



Kuva 7: Konvoluutiokerrosten syötteiden valinta ja ulostulojen muodostuminen [2]

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 7 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.

4.3 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 kuvien tapauksessa ominaisuuskartta joka löytää kuvasta suoria viivoja saattaa olla hyödyllinen, sillä suoria viivoja voi ilmetä useassa eri paikassa kuvassa, ja toisaalta korkeammalla abstraktiotasolla kuvassa esiintyvät objektit ovat edelleen sama objekti, vaikka niitä olisi siirretty muutamia pikseleitä johonkin suuntaan.

4.4 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.

4.5 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

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.

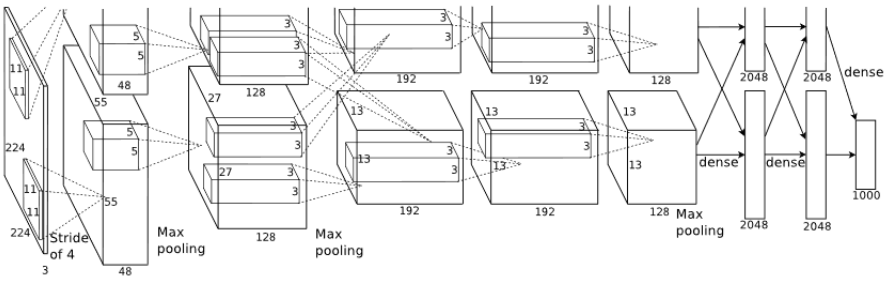
5.1 Verkon toiminnan visualisointi

... [15]

5.2 Kuvien luokittelu

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

visualisointi. Esimerkkikuvia imagenet aineistosta?



Kuva 8: ILSVRC-2012 kilpailun voittaneen neuroverkon rakenne [5]

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 [5].

5.3 Jokin muu sovellus kuin kuvien luokittelu?

...

5.4 Deep dream

Samanlainen toteutustapa kuin takaisinvirtausalgoritmissa, paitsi painotusarvot pysyvät kiinteinä ja muutetaankin syötettä.

5.5 Kirjastot

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

6 Yhteenveto

Kappale siitä, että sitä ei vielä täysin ymmärretä, miksi neuroverkot oppivat niin hyvin.

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 (ehkä) miten aktivaatiofunktioita ja muita juttuja valitaan järkevästi/hyvin yms.

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] 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`.

[4] 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`.

[5] 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`.

[6] 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.

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

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

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

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

[11] 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`.

[12] 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.

[13] 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`.

[14] 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.

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