Konvolutionaaliset neuroverkot

Teemu Sarapisto

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 7. 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	7. toukokuuta 2018	20

Tiivistelmä — Referat — Abstract

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. Joissakin koneoppimisen sovelluskohteissa, kuten kuvien sisällön tunnistamisessa, keinotekoiset neuroverkot ovat osoittautuneet toistaiseksi ylivoimaisesti parhaiten pärjääviksi ratkaisuiksi. Tässä kirjallisuuskatsaukseen perustuvassa tutkielmassa käsitellään ensin kaikille neuroverkoille yhteisiä piirteitä, kuten yksittäisten neuronien ja neuroverkojen rakennetta. Seuraavaksi esittelemme tärkeimmät neuroverkkojen harjoittamisen menetelmät: gradienttimenetelmän ja takaisinvirtausalgoritmin. Lopuksi käymme läpi miten konvolutionaaliset neuroverkot eroavat muista neuroverkoista, ja modernin niitä hyödyntävän sovelluksen.

ACM Computing Classification System (CCS):
Computing methodologies → Machine learning approaches → Neural networks

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 [5], jolloin kybernetiikan tutkimuksen myötä McCulloch ja Pitts kehittivät mukaansa nimetyn McCulloch-Pitts neuronin, tarkoituksenaan luoda matemaattinen malli, jolla kuvailla biologista aivoissa tapahtuvaa oppimista [15]. 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 [19].

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 [20].

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 [7]. 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 [10]. 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 [5].

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 [5].

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 [1]. Kuvien sisällön tunnistamisen lisäksi syväoppiminen on osoittautunut erittäin hyödylliseksi useissa aikaisemmin haasteellisiksi osoittautuneissa sovelluskohteissa, kuten puheentunnistuksessa [2], liikennemerkkien luokittelussa [22], sekä jalankulkijoiden tunnistamisessa [24].

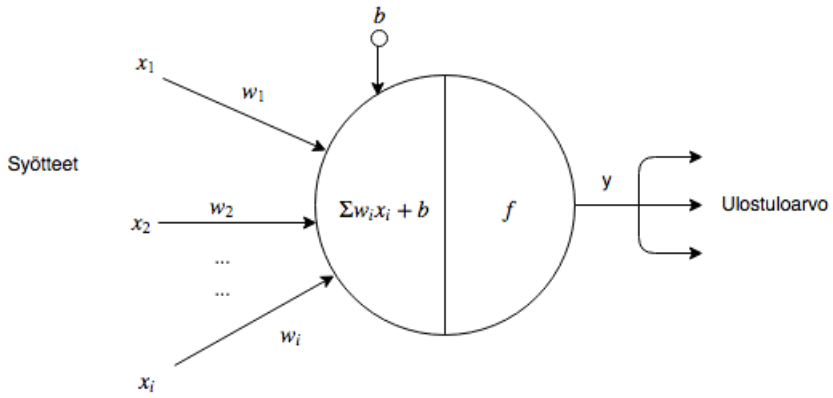
Tässä tutkielmassa aloitetaan käymällä läpi yleisiä keinotekoisten neuroverkkojen piirteitä, kuten yksittäisten neuronien ja eteenpäinsyöttävien neuroverkkojen rakennetta. Seuraavaksi käydään läpi kuinka neuroverkkoja harjoitetaan harjoitusaineiston avulla takaisinvirtausalgoritmia hyödyntäen. Lopuksi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

2 Neuroverkkojen rakenne

Tässä kappaleessa käydään läpi Michael A. Nielsenin kirjaan [17] pohjautuen perusteet keinotekoisten neuroneiden ja neuroverkkojen rakenteesta.

2.1 Keinotekoinen neuroni

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.



Kuva 1: Keinotekoisen neuronin rakenne

Ensimmäisessä vaiheessa syötteet kerrotaan painotusarvollaan ja litistetään yhdeksi arvoksi summaamalla. Summaan lisätään lopuksi taipumusvakio (bias) b. Tästä saadaan Kuvan 1 neuronin vasemman puoliskon kaava $\sum w_i x_i + b$.

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

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

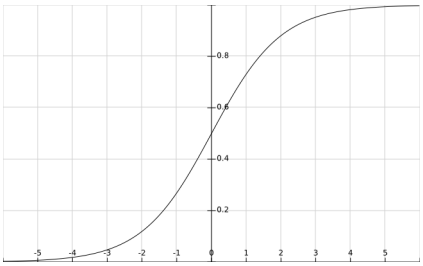
Aktivaatiofunktioina käytetään usein derivoituvia funktioita, sillä tämä helpottaa myöhemmin kappaleessa 3.1 esiteltävän gradienttimenetelmän käyttöä. Eräs suosittu aktivaatiofunktio on sigmoidinen funktio

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

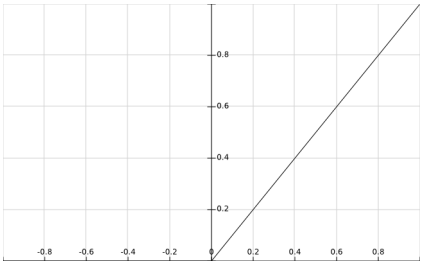
Nykyään kuitenkin suosituin aktivaatiofunktio on Rectified Linear Unit (ReLU)

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

koska sitä käytettäessä on todettu oppimisen olevan monikerroksisissa neuroverkkoarkkitehtuureissa huomattavasti nopeampaa verrattuna sigmoidiseen funktioon [12].



Kuva 2: Sigmoidinen funktio

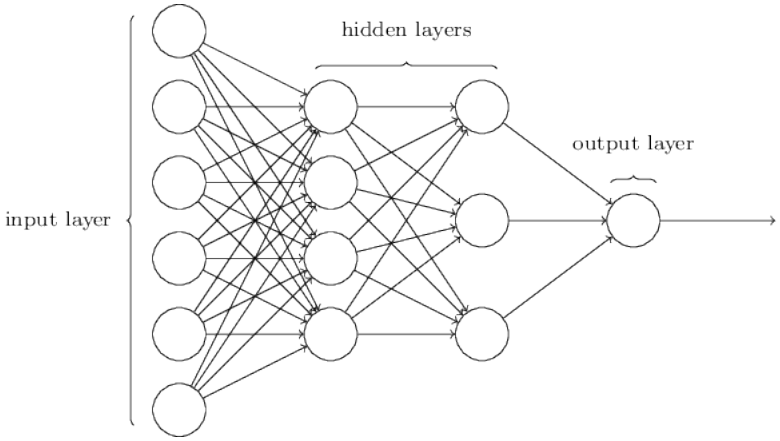


Kuva 3: Rectified Linear Unit (ReLU)

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

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



Kuva 4: Nelikerroksinen neuroverkko, jossa on kaksi piilokerrosta [17]

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, syötekerroksesta mahdollisten piilokerrosten kautta kohti ulostulokerrosta, jonka neuroneiden ulostuloarvot ovat neuroverkon laskennan lopputulos. Silmukoita sisältäviä takaisinkytkeytyviä neuroverkkoja on myös olemassa, mutta ne eivät kuulu tämän tutkielman aihepiiriin [8].

Kuvassa 4 vasemmanpuoleisimpana nähdään syötekerros. Esimerkiksi haluttaessa syöttää 64x64 kuva neuroverkolle voidaan syötekerroksena käyttää 64x64 neuronin kerrosta, johon kuvan pikselien väriarvot koodataan.

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 [9]. 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 [12]. Ohjatussa oppimisessa harjoitusaineisto on luokiteltu jollakin perusteella ja harjoittamis-

vaiheessa pyritään saamaan opetettava malli löytämään näille ennakkoon määritellyille luokille yhteisiä ja harjoitusaineiston ulkopuolisiin syötteisiin yleistettävissä olevia tekijöitä. Harjoitusaineisto on luokiteltu yleensä kahteen osaan, joista ensimmäistä käytetään itse mallin harjoittamiseen, ja toista osaa harjoittamisen jälkeen mallin toimivuuden mittaamiseen.

Tyypillinen neuroverkkojen harjoittamiseen käytetty harjoitusaineisto on suuri joukko pareiksi järjestettyjä yksiköitä $\{(\mathbf{x_i}, \mathbf{z_i})\}_{i=1}^{N}$, jossa $\mathbf{x_i}$ on jokin syötevektori ja $\mathbf{z_i}$ toivottu ulostulovektori tälle syötteelle [12]. Esimerkiksi käsin kirjoitettujen numeroiden kuvista tunnistamista varten tehdyssä harjoitusaineistossa $\mathbf{x_i}$ voi olla vektori kuvan pikselien värit määritteleviä arvoja ja $\mathbf{z_i}$ vektori arvoja, jossa vain kuvasta löytyvää numeroa vastaava arvo on asetettu nollasta poikkeavaksi.

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.

Hyvin yleisesti käytetään neliöllistä virhefunktiota

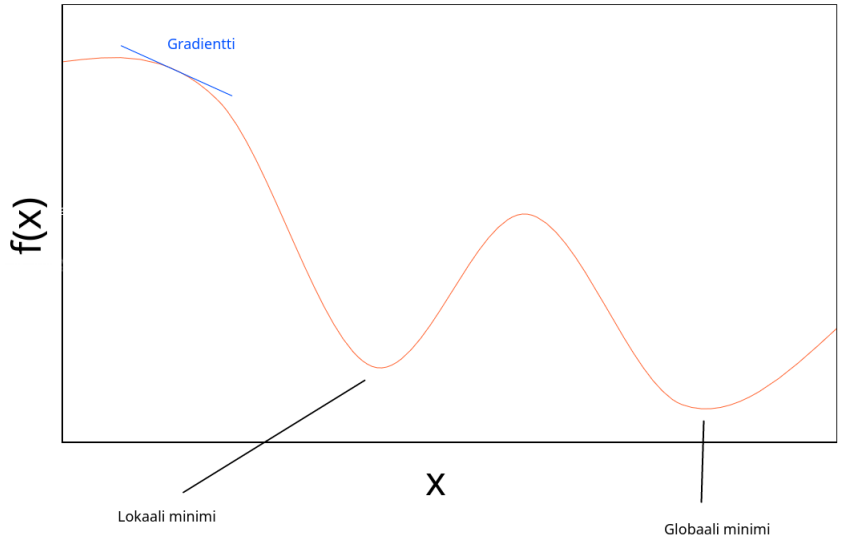
$$E(x) = \frac{1}{2} \sum_{i=1}^{N} \|y(\mathbf{x_i}) - \mathbf{z_i}\|^2, \tag{3}$$

jossa $y(\mathbf{x_i})$ on neuroverkon tuottama tulos jollakin harjoitusaineiston yksikön syötteellä $\mathbf{x_i}$, $\mathbf{z_i}$ on harjoitusaineiston perusteella syötteelle $\mathbf{x_i}$ toivottu ulostuloarvo, ja N syötteiden määrä [5]. Koska $y(\mathbf{x_i})$ ja $\mathbf{z_i}$ ovat yleensä vektoreita, merkinnällä $\|v\|$ tarkoitetaan erotuksesta muodostuvan vektorin euklidista normia.

Harjoitusvaiheessa ainoat neuroverkossa muutettavissa olevat parametrit ovat sen neuroneiden syötteiden painotusarvot ja taipumusvakiot. Seuraavaksi esitellään gradienttimenetelmä ja takaisinvirtausalgoritmi, joiden avulla voidaan selvittää, kuinka suuri vaikutus kullakin painotusarvolla ja taipumusvakiolla on koko verkon virheeseen ja muuttaa niitä suuntaan, joka pienentää virhettä.

3.1 Gradienttimenetelmä

Gradientti on derivaatan yleistys useamman kuin yhden muuttujan funktioille, joka kertoo mihin suuntaan funktion arvo kasvaa nopeimmin. Gradientti-



Kuva 5: Gradientti sekä funktion lokaali ja globaali minimi

menetelmällä (gradient descent) tarkoitetaan numeerista menetelmää, jossa kuljetaan iteratiivisesti negatiivisen gradientin suuntaan kunnes gradientti on tarpeeksi pieni [20].

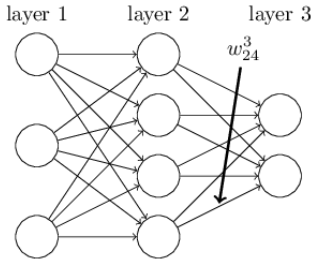
Funktion on oltava derivoituva, jotta sille voidaan muodostaa gradientti, joten myös gradienttimenetelmä edellyttää funktioiden derivoituvuutta. Käytännössä kuitenkin riittää, että aktivaatiofunktiot ovat suurimmilta osin derivoituvia funktioita. Esimerkiksi kappaleessa 2.1 esitelty ReLU $f(x) = max(0, x)$ ei ole derivoituva kohdassa $x = 0$, mutta sovelluksissa tämä voidaan kiertää vain valitsemalla aktivaatiofunktion derivaataksi tapauksessa $x = 0$ jokin kiinteä arvo [5].

Vaikka gradienttimenetelmä soveltuukin vain lokaalien minimien etsintään, sen ja sen muunnelmien on huomattu olevan yleensä riittävän toimivia ratkaisuja neuroverkkojen harjoittamiseen [20][5]. Toistaiseksi käytännön neuroverkkototeutuksissa on todettu olevan tärkeämpää etsiä parametriavaruudesta sellaisia lokaaleita minimeitä, joissa virhefunktio saa pieniä arvoja, kuin etsiä funktion todellista globaalia minimiä [6]. Tämä on kuitenkin vielä aktiivisen tutkimuksen aluetta, josta ei ole varmaa tietoa [5].

Sen sijaan että virhefunktion gradientin keskiarvo laskettaisiin mahdollisesti miljoonille harjoitusaineiston yksiköille jokaisella harjoituskierroksella, yleensä käytetään stokastista gradienttimenetelmää, jossa gradientin keskiarvo muodostetaan vain osasta harjoitusaineistoa [12]. Menetelmää kutsutaan stokastiseksi, koska harjoitusaineiston pienelle osajoukolle laskettu keskiarvo on kohinainen estimaatti todellisesta keskiarvosta.

3.2 Takaisinvirtausalgoritmi

Takaisinvirtausalgoritmilla viitataan kaksivaiheiseen neuroverkkojen opettamiseen kehitettyyn menetelmään [20]. Eteenpäinvirtaukseksi kutsutaan ensimmäistä vaihetta, jossa neuroverkon virhe lasketaan jollekkin harjoitusaineiston osajoukolle aikaisemmin esitellyllä tavalla. Taaksepäinvirtaukseksi, josta algoritmi on saanut nimensä, kutsutaan virheen kuljettamista verkossa ulostulokerroksesta takaisin kohti syötekerrosta virhefunktion osittaisderivaattojen selvittämiseksi verkon painotusarvojen ja taipumusvakioiden suhteen. Kun virhefunktion osittaisderivaatat painotusarvojen ja taipumusvakioiden suhteen tunnetaan, voidaan niistä muodostaa gradientti ja soveltaa gradienttimenetelmää virheen minimoimiseksi.



Kuva 6: Painotuksiin viittaamiseen käytetty notaatio [17].

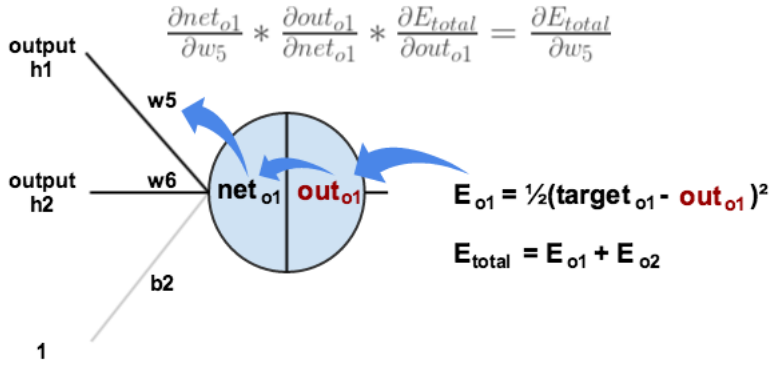
Algoritmin käsittelyn helpottamiseksi käytetään verkon osiin viittaamiseen alan kirjallisuudessa yleisesti käytettyä notaatiota likimain vastaavia merkintätapoja [17][18]. Kerroksessa l olevan i:nnen neuronin ulostuloarvosta käytetään merkintää out_i^l, neuronin painotettujen syötteiden summasta merkintää net_i^l, kerroksessa $l-1$ olevan i:nnen neuronin ja kerroksessa l olevan j:nnen neuronin välillä olevasta painotuksesta merkintää w_{ji}^l, ja koko verkon virheestä merkintää E_{koko}. Käydään seuraavaksi läpi, kuinka virhefunktion derivaatta yksittäisen painotusarvon suhteen selvitetään ensin ulostulokerroksessa, ja tämän jälkeen muissa kerroksissa.

Virhefunktion derivaatta painotuksen suhteen voidaan laskea derivaatan ketjusäännön avulla

$$\frac{\partial E_{koko}}{\partial w_{ji}^l} = \frac{\partial E_{koko}}{\partial out_j^l} \cdot \frac{\partial out_j^l}{\partial net_j^l} \cdot \frac{\partial net_j^l}{\partial w_{ji}^l}$$

Kuvan 7 mukaisesti. Käydään järjestyksessä läpi kuinka yhtälön oikean puolen kolme termiä selvitetään.

Yhtälön oikean puolen ensimmäinen termi ulostulokerroksen tapauksessa saadaan ottamalla Kaavan 3 mukaisesta virhefunktiosta osittaisderivaatta out_j^L suhteen, jossa L on ulostulokerros, jolloin summan muut termit katoavat ja jäljelle jää



Kuva 7: Virhefunktion derivaatta syötteen painotuksen suhteen [14]

$$\frac{\partial E_{koko}}{\partial out_j^L} = 2 \cdot \frac{1}{2}(out_j^L - z_j)^{2-1} = out_j^L - z_j.$$

Toinen termi on käsiteltävän neuronin aktivaatiofunktion derivaatta

$$\frac{\partial out_j^l}{\partial net_j^l} = f'.$$

Kolmannen termin kohdalla huomataan otettaessa neuronin summausfunktiosta osittaisderivaatta jonkin sen painotuksen suhteen, muut termit ovat vakioita, joiden derivaatta on 0, w_{ji}^l derivaatta itsensä suhteen on 1, jolloin jäljelle jää

$$\frac{\partial net_j^l}{\partial w_{ji}^l} = 1 \cdot out_i^{l-1} = out_i^{l-1}.$$

Muiden kerroksien kohdalla jälkimmäiset kaksi termiä kolmesta voidaan muodostaa samalla tavalla, mutta ensimmäisen termin laskeminen on monimutkaisempaa, sillä on otettava huomioon, että muissa kuin ulostulokerroksessa neuronien ulostuloarvot vaikuttavat niitä seuraavien kerrosten ulostuloarvoihin ja siten virheisiin. Tällöin

$$\frac{\partial E_{koko}}{\partial out_j^l} = \sum_{k=1}^{N} \frac{\partial E_k^{l+1}}{\partial out_j^l},$$

jossa E_k^{l+1} viittaa niistä $l+1$ kerroksen neuroneista, jotka saavat yhtenä syötteistään arvon out_j^l, koostuvan osajoukon k:nnen neuronin ulostuloarvon osuuteen koko verkon virheestä. Koska algoritmia suoritetaan ulostulokerroksesta syötekerrokseen päin, summan termien selvittämisessä voidaan

hyödyntää aikaisemmissa kerroksissa laskettuja välituloksia ja ketjusääntöä. Saadaan

$$\frac{\partial E_k^{l+1}}{\partial out_j^l} = \frac{\partial E_k^{l+1}}{\partial net_k^{l+1}} \cdot \frac{\partial net_k^{l+1}}{\partial out_j^l},$$

joista yksinkertaisempi jälkimmäinen termi

$$\frac{\partial net_k^{l+1}}{\partial out_j^l} = w_{kj}^{l+1},$$

ja ensimmäinen termi

$$\frac{\partial E_k^{l+1}}{\partial net_k^{l+1}} = \frac{\partial E_k^{l+1}}{\partial out_k^{l+1}} \cdot \frac{\partial out_k^{l+1}}{\partial net_k^{l+1}},$$

jonka ensimmäinen termi on jonkin syötteenään out_j^l ottavan neuronin n ulostuloarvon out_k^{l+1} virheen vaikutus kokonaisvirheeseen, joka on valmiiksi laskettu sitä neuronia käsiteltäessä, ja toinen termi neuronin n aktivaatiofunktion f derivaatta f'. Nähtiin siis syötekerrosta lähempien kerrosten virheiden laskentaan tarvittavan ensin ulostulokerrosta lähempien kerrosten neuroneiden virheet, joten virheen voidaan ajatella virtaavan ulostulokerroksesta kohti syötekerrosta.

3.3 Oppimistahti ja aloituspainot

Kun

$$\frac{\partial E_{koko}}{\partial w_{ji}^l}$$

tiedetään, virhettä pyritään pienentämään muuttamalla painotusta jonkin oppimistahdin (learning rate) $-\eta$ mukaisesti. Virhefunktiolle laskettu gradientti itsessään kertoo vain suunnan infinitesimaalisella alueella johon siirtymällä virhefunktio kasvaa eniten, mutta ei optimaalista askeleen kokoa. Negatiivisen oppimistahdin avulla siirrytään gradientin osoittamaa virhefunktion suurimman kasvun suuntaa vastakkaiseen suuntaan

$$w_{ji}^l = w_{ji}^l - \eta \cdot \frac{\partial E_{koko}}{\partial w_{ji}^l},$$

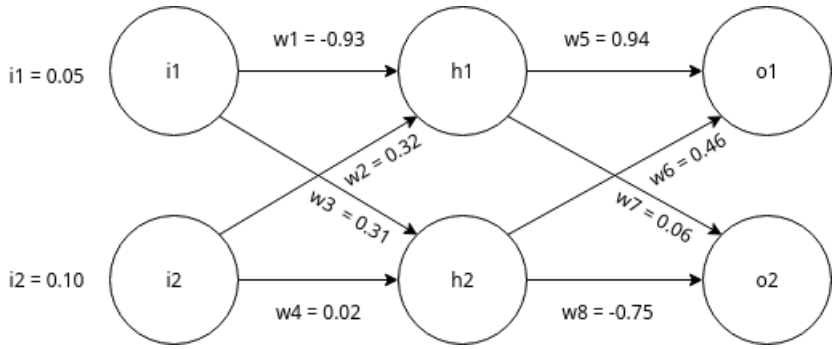
joka pienentää neuroverkon virhettä, mikäli oppimistahti on riittävän pieni. Oppimistahdin valintaan ja muuttamiseen harjoittamisen aikana löytyy lukuisia heuristisia menetelmiä, joista yleisimpänä jonkin pienen arvon valitseminen aluksi, ja sen pienentäminen oppimisen alkaessa hidastumaan harjoittamisen aikana [5][10].

Verkon aloituspainojen valinnassa olennaisinta on, että kaikki painotukset eivät ole samoja, koska silloin virhefunktion gradientti on näille painotuksille aina sama, eivätkä neuronit pysty erikoistumaan [5]. Yleisin tapa painojen alustamiseen on valita niille satunnaisia pieniä arvoja jostakin satunnaisjakaumasta. Taipumusvakioiden aloitusarvot valitaan usein heuristisesti.

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 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 8: 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

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

Syöttämällä nämä arvot sigmoidiseen aktivaatiofunktioon saadaan

$$h_1^{out} = \sigma(h_1^{in}) \approx 0.496$$

$$h_2^{out} = \sigma(h_2^{in}) \approx 0.504.$$

Toistaen vastaavat vaiheet ulostulokerrokselle käyttäen piilokerroksen ulostuloarvoja syötteenä saadaan:

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

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

$$o_1^{out} = \sigma(o_1^{net}) \approx 0.652$$
$$o_2^{out} = \sigma(o_2^{net}) \approx 0.414.$$

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

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

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

$$o_1^{out} = \sigma(o_1^{net})$$

niin

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

$$o_1^{net} = w_5 \cdot h_1^{out} + w_6 \cdot h_2^{out}$$

joten viimeinen tarvittava osa

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

joten

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

Korjaamme nyt w_5 virhettä asettamalla sen uudeksi arvoksi

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

3.5 Ylisovitus ja sen ratkaiseminen

Suuri haaste neuroverkkojen harjoittamisessa, ja koneoppimisessa ylipäätään, on ylisovitus (overfitting), jossa neuroverkon harjoittamisen jälkeen neuroverkko saa harjoitusaineistosta valituille syötteille pieniä arvoja virhefunktiosta, mutta uuden aineiston kanssa virhefunktio tuottaa suuria arvoja [17]. Tällöin neuroverkon oppima approksimaatio vastaa harjoitusaineistoa liian tarkkaan, eikä pysty yleistämään harjoitusaineistosta löytyviä ominaisuuksia aineiston ulkopuolisille syötteille.

Ratkaisuksi ylisovitukseen on kehitetty useita menetelmiä. Neuroniyksikköjen pudotus (dropout) on menetelmä, jossa harjoitusvaiheessa yksittäisiä neuroneita poistetaan satunnaisesti käytöstä, jolloin yksittäiset neuronit naapureineen eivät erikoistu tiettyihin aineiston ominaisuuksiin liian tarkasti [23]. L2 regularisaatiossa neuroverkon virheeseen lisätään arvo $\lambda \sum_{i=1}^{N}$ jossa λ on regularisaatioparametri, jolla regularisaation voimakkuutta voidaan säätää, ja N on painojen määrä. Tällä lisäyksellä neuroverkko saadaan priorisoimaan pieniä painotusarvoja [16].

Ylisovitusta voidaan vähentää myös laajentamalla harjoitusaineistoa [10]. Esimerkiksi kuvien tunnistuksessa harjoitusaineiston kokoa on kasvatettu kääntämällä ja siirtämällä kuvia, ottamalla kuvista osakuvia ja skaalaamalla niitä suuremmaksi ja pienemmäksi, sekä lisäämällä kuviin satunnaista kohinaa.

Ylisovittamisen välttäminen on sitä vaikeampaa, mitä enemmän verkossa on opittavia parametreja, mistä johtuen sen välttämiseksi kehitetyt menetelmät ovat muuttuneet tärkeämmiksi modernien neuroverkkojen kasvaessa yhä suuremmiksi ja syvemmiksi [25]. Seuraavaksi esiteltävät konvolutionaaliset neuroverkot vähentävät tarvittavien yhteyksien ja painotuksien määrää, näin osaltaan helpottaen ylisovitusta.

4 Konvolutionaalisten neuroverkkojen rakenne

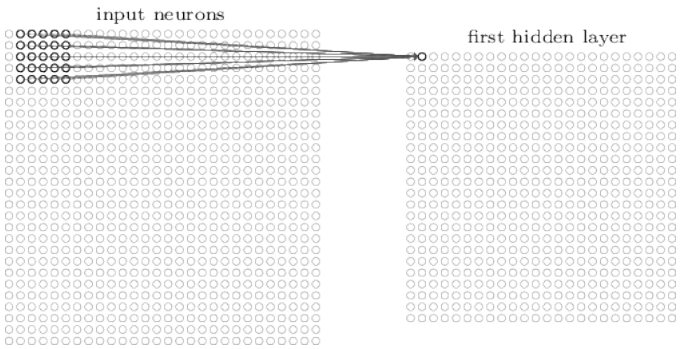
Konvolutionaalisten neuroverkkojen rakenne eroaa tavanomaisista täysin yhdistetyistä neuroverkoista konvoluutio- ja kokoamiskerroksien (pooling layer), kerrosten rinnakkaisuuden, sekä jaetuista painotuksista muodostettujen piirrekarttojen kautta [13]. Näiden ominaisuuksien avulla pystytään vähentämään harjoitettavien parametrien määrää ja ottamaan verkkojen rakenteessa huomioon sovelluskohteiden luontaisia ominaisuuksia [25]. Pienempien parametrimäärien mahdollistamat syvemmät neuroverkot pystyvät hyödyntämään rakenteessaan monissa luonnollisissa yhteyksissä ilmenevää ominaisuutta, jossa korkeamman abstraktiotason käsitteet voidaan muodostaa yhdistelmästä matalamman tason käsitteitä [12]. Esimerkiksi kuvissa lähekkäiset viivat muodostavat muotoja, muodot muodostavat osia, ja osista muodostuu objekteja. Syvät, kuvien tunnistamiseen harjoitetut konvolutionaaliset neuroverkot vastaavatkin suorituskyvyltään ja toimimismalleiltaan

yllättävän läheisesti joitakin kädellisten nisäkkäiden visuaalista prosessointia suorittavia aivolohkoja [3].

Konvolutionaalisia neuroverkkoja harjoitetaan samoilla gradienttimenetelmään perustuvilla opetusmenetelmillä, kuin muitakin neuroverkkoja [17]. Esimerkiksi aikaisemmin esitelty takaisinvirtausalgoritmi soveltuu myös konvolutionaalisten neuroverkkojen harjoittamiseen pienin muutoksin.

4.1 Konvoluutiokerrokset

Konvoluutiokerrosten neuroneiden syötteet $a_{x,y}$ painotuksineen $w_{x,y}$ voidaan esittää matemaattisesti esimerkiksi 5x5 kokoiselle paikalliselle vastaanottavalle kentälle (local receptive field) muodossa

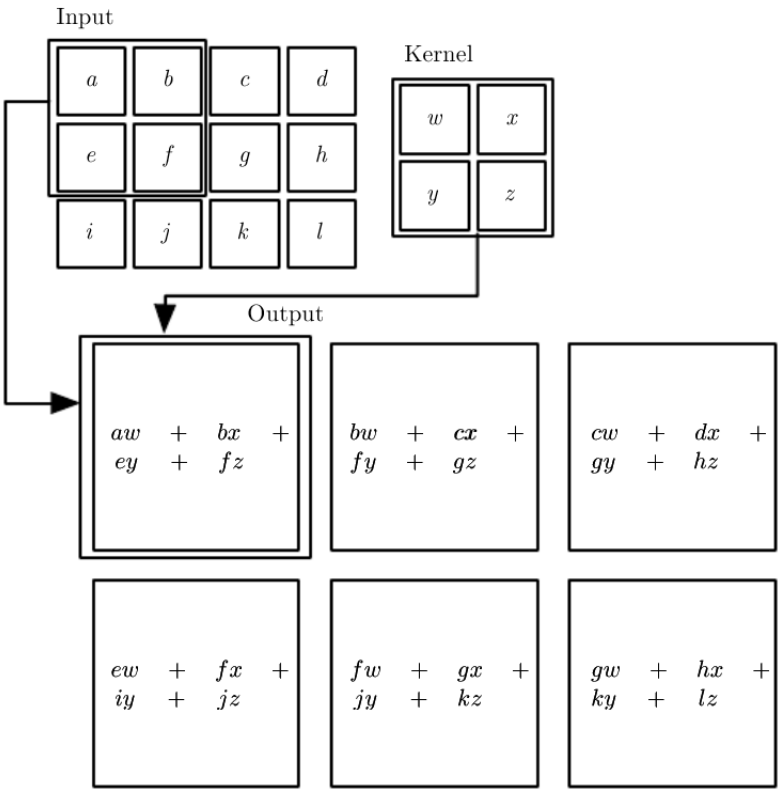


Kuva 9: Paikallinen vastaanottava kenttä [17]

$$\sum_{l=1}^{5}\sum_{m=1}^{5} w_{l,m}a_{j+l,k+m},$$

joka on käytännössä diskreetti konvoluutio, jossa painotukset ovat konvoluution ydin (kernel) [11]. Konvoluution voidaan ajatella olevan liukuva ikkuna, joka rajoittaa neuronit saamaan Kuvan 9 mukaisesti syötteenään vain osan syötekerroksensa ulostuloista, täysin yhdistetyistä kerroksista poiketen.

Konvoluutiokerroksia on yleensä useita rinnakkain ja kunkin kerroksen neuronit jakavat samassa kerroksessa olevien neuroneiden kesken yhteiset painot (shared weights), tarkoituksena kannustaa verkon rakenteella neuroverkko oppimaan näihin kerroksiin syötteiden lokaaleja ominaisuuksia tunnistavia piirrekuvauksia (feature maps) [11]. Rinnakkaisissa kerroksissa samassa kohtaa olevat neuronit saavat saman syötteen, mutta konvoluutio suoritetaan eri ytimen perusteella, eli syötettä verrataan samasta kohtaa useaan eri piirrekuvaukseen. Yksittäinen neuroni konvoluutiokerroksessa siis kertoo, löytyykö sen saamien syötteiden alueelta kerroksen piirrekuvausta vastaavaa ominaisuutta.

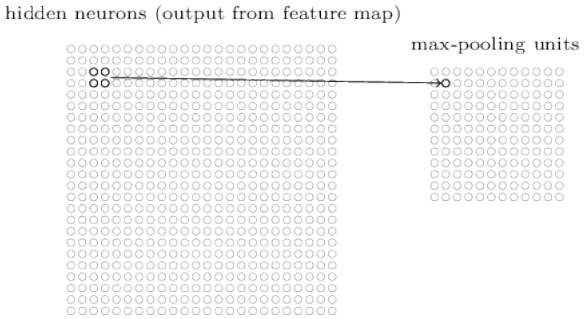


Kuva 10: Konvoluutiokerrosten syötteiden valinta ja ulostulojen muodostuminen: syöte muuttuu ytimen pysyessä samana [5].

Kuvasta 10 ilmenee hyödyllinen konvoluutiokerrosten piirre: mikäli tehdään vain konvoluutioita joissa ydin mahtuu kokonaan syötekuvaan, konvoluutiokerroksella on vähemmän ulostuloja kuin syötteitä, eli ne tiivistävät informaatiota. Kuvan tapauksessa nähdään syötematriisin ollessa 3x4 ja ytimen 2x2 kokoinen, ulostulomatriisin kooksi tulee 2x3. Tällä tavalla mahdollistaa myös sen, että konvolutionaaliset neuroverkoet voivat ottaa vastaan vaihtelevan kokoisia syötteitä.

4.2 Kokoamiskerrokset

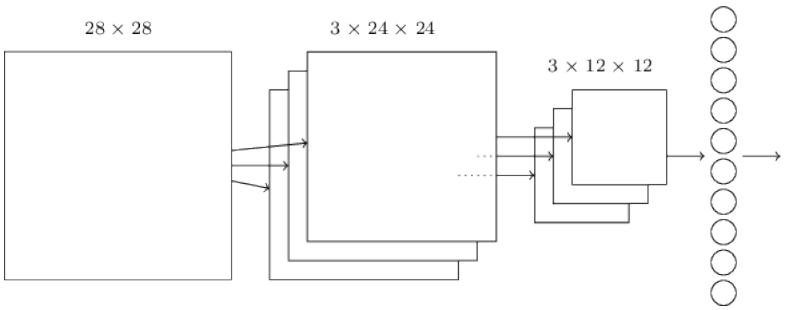
Usein konvolutionaalisissa neuroverkoissa käytetään konvoluutiokerroksien jälkeen kokoamiskerroksia, jotka tekevät yhteenvedon jostakin edeltävästä neuroverkkokerroksen alueesta ilman opittavia parametreja, näin pienentäen verkkoa ja vähentäen opittavien parametrien määrää entisestään [17]. Hyvin yleisesti käytetyn maksimikokoamiskerroksen (max-pooling) neuronit antavat ulostulokseen syöteneuroniensa ulostuloista suurimman [10]. Esimerkiksi Kuvassa 11 nähtävä 2x2 maksimikokoamiskerros antaisi syötteille



Kuva 11: 2x2 maksimikokoamiskerros [17]

$\{0.1, 0.2, 0.3, 0.4\}$ ulostuloarvoksi $max(\{0.1, 0.2, 0.3, 0.4\}) = 0.4$.

4.3 Täysin yhdistetyt kerrokset



Kuva 12: Yksinkertaisen konvolutionaalisen neuroverkon kerrokset: syötekerros, konvoluutiokerrokset, kokoamiskerrokset, täysin yhdistetty kerros [17]

Konvoluutiokerrosten kohdalla konvolutionaaliset neuroverkot haarautuvat rinnakkaisiin kerroksiin, jotka voidaan litistää (flattening) takaisin yhteen sijoittamalla verkon viimeiseksi kerrokseksi täysin yhdistetyn kerroksen [17]. Täysin yhdistetyt kerrokset kokoavat piirrekarttojen keräämän informaation ja tuottavat niistä lopullisen neuroverkon ulostuloarvon. Viimeisessä kerroksessa käytetään usein aktivaatiofunktiona softmax-funktiota, joka tekee ulostulokerroksen neuroneiden ulostuloarvoista suhteellisia toisiinsa nähden ja pakottaa arvojen summaksi arvon 1 [17]. Neuroverkon ulostuloarvovektori voidaan tulkita tällöin todennäköisyysjakaumaksi neuroverkon tunnistamista käsitteistä.

5 Konvolutionaaliset neuroverkot käytännössä

Konvolutionaalisia neuroverkkoja käytetään nykyään puheentunnistuksessa [2], liikennemerkkien luokittelussa [22] ja jalankulkijoiden tunnistamisessa [24] kuvien sisällön tunnistamisen lisäksi. Vaikka konvolutionaaliset neuroverkot keksittiin pääpiirteittäin jo vuonna 1980 [4], vasta LeCunin työn myötä 80-luvun lopussa niille keksittiin käytännön sovelluksia, kuten käsinkirjoitettujen postinumeroiden tunnistaminen, joissa ne pärjäsivät muita ratkaisuja paremmin [13].

Ennen konvolutionaalisia neuroverkkoja kuvien sisällön tunnistaminen perustui pääasiassa ihmisten käsin luomien piirretunnistimien (feature detector) käyttöön, jotka olivat sekä erittäin työläitä tehdä, että herkkiä pienillekkin muutoksille syötteessä [11]. Konvolutionaaliset neuroverkot pystyvät oppimaan nämä piirteet itse, tehden niistä yleistettäviä moniin eri tilanteisiin, ja niiden luontainen kyky tunnistaa samoiksi syötteitä, joissa on vain pieniä sijainnillisia eroja, saa ne suoriutumaan vanhoja menetelmiä paremmin.

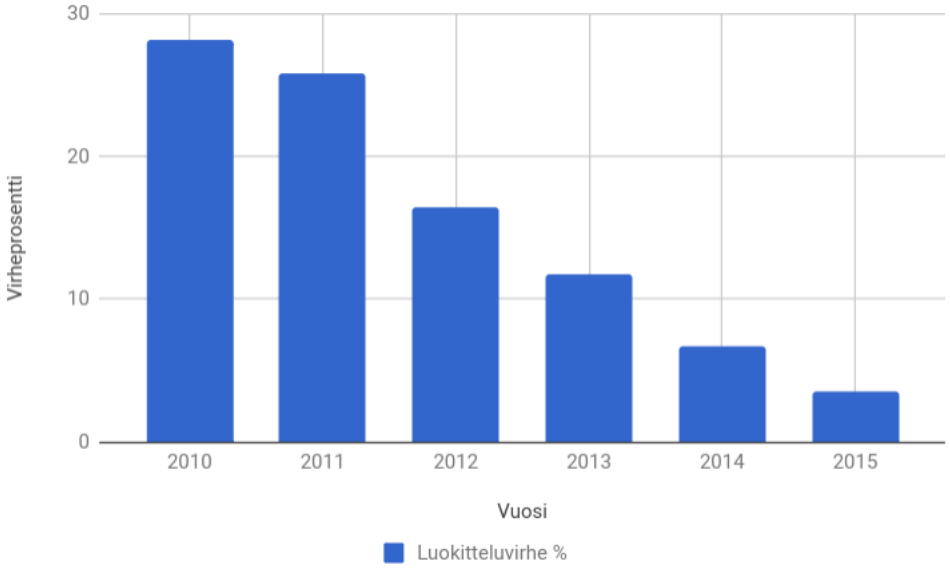
5.1 ImageNet Large Scale Visual Recognition Challenge

ILSVRC-kilpailussa kilpaillaan pienimmän virheprosentin saavuttamisesta 1000 objektikategorian tunnistuksessa [21]. 2012 vuodesta lähtien kilpailuun osallistuvien mallien toiminnan testaamiseen on käytetty suurinpiirtein samaa noin 150000 kuvan aineistoa, harjoitusaineiston koostuessa noin 1,2 miljoonasta kuvasta. Harjoitusaineiston kuvissa on jokaista objektiluokkaa kohden määritelty, löytyykö kyseistä objektia kyseisestä kuvasta.

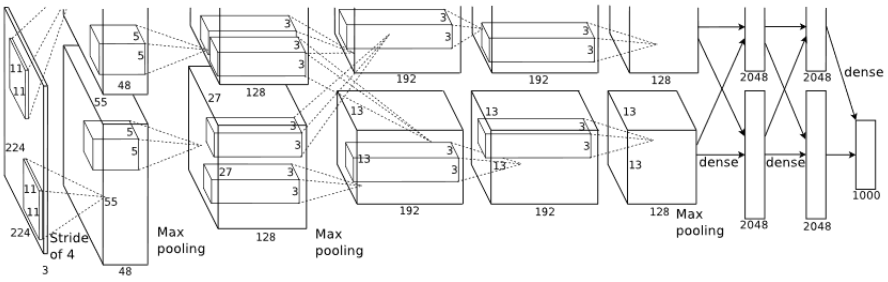
Kilpailua pidetään erittäin merkittävänä konenäön, ja erityisesti konvolutionaalisten neuroverkkojen kehitykselle, sillä ennen vuoden 2012 ILSVRC-kilpailua konvolutionaaliset neuroverkot eivät saaneet juurikaan huomiota konenäön tutkimuksessa [12]. Seuraavaksi esiteltävä vuonna 2012 kilpailun voittanut Krizhevsky, Sutskever ja Hintonin (myöhemmin KSH) konvolutionaalinen neuroverkko sai suuren yleisön huomion voittamalla kilpailun ylivoimaisesti 16.4 tunnistusvirheprosentilla toisen sijan 27.0 tulokseen nähden. Vuodesta 2012 lähtien kilpailun on joka vuosi voittanut konvolutionaalinen neuroverkko.

5.2 Esimerkki modernista konvolutionaalisesta neuroverkosta

KSH neuroverkossa on 7 piilokerrosta, joista 5 ensimmäistä ovat konvoluutiokerroksia, ja 2 viimeistä kerrosta täysin yhdistettyjä kerroksia. ImageNet kuvamateriaalissa on vaihtelevan kokoisia kuvia, mutta KSH 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, näin



Kuva 13: Kuvien luokitteluvirheen lasku ILSVRC kilpailussa vuosittain [21]



Kuva 14: ILSVRC-2012 kilpailun voittaneen neuroverkon rakenne [10]

laajentaen harjoitusdataa ja vähentäen ylisovitusta. Ylisovituksen vähentämiseksi käytettiin myös neuroniyksiköiden pudotusmenetelmää. Piilokerrosten aktivaatiofunktiona verkossa käytettiin aikaisemmin esiteltyä ReLUa, ja ulostulokerroksessa softmaxia. Oppimistahdiksi valittiin ensin 0.01, ja sitä pienennettiin jakamalla sen arvo kymmenellä aina kun oppimisen huomattiin merkittävästi hidastuneen.

KSH harjoittamisessa hyödynnettiin näytönohjaimia. Verkko jaettiin kahteen osaan, sillä se ei olisi mahtunut yhden silloisen kuluttajakäyttöön suunnatun näytönohjaimen näyttömuistiin.

6 Yhteenveto

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

Käsittelemättä jäi millä perusteilla hyperparametrit kuten aktivaatiofunktiot, aloituspainot ja oppimistahti valitaan. Tämä on toisaalta erittäin vaikea tehtävä ja alan kirjallisuudessa kyseiset parametrit valitaankin yleensä heuristisin menetelmin.

Neuroverkot ovat erittäin aktiivisen tutkimuksen aluetta, sillä monilta osin ei ole vielä täysin selvää, miksi ne toimivat niin hyvin. Toisaalta miljoonien parametrien optimointi on lähes mahdoton tehtävä ratkaista täydellisesti.

Lähteet

[1] *ImageNet*, 2018. `http://image-net.org/about-stats`, vierailtu 2018-05-1.

[2] 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.

[3] Charles F. Cadieu, Ha Hong, Daniel Yamins, Nicolas Pinto, Diego Ardila, Ethan A. Solomon, Najib J. Majaj ja James J. DiCarlo: *Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition.* CoRR, abs/1406.3284, 2014. `http://arxiv.org/abs/1406.3284`.

[4] Kunihiko Fukushima: *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.* Biological Cybernetics, 36(4):193–202, Apr 1980, ISSN 1432-0770.

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

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

[7] Geoffrey E. Hinton, Simon Osindero ja Yee Whye Teh: *A Fast Learning Algorithm for Deep Belief Nets.* Neural Computation, 18(7):1527–1554, 2006.

[8] Sepp Hochreiter ja Jürgen Schmidhuber: *Long Short-Term Memory.* Neural Computation, 9(8):1735–1780, 1997. https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Kurt Hornik, Maxwell B. Stinchcombe ja Halbert White: *Multilayer feedforward networks are universal approximators.* Neural Networks, 2(5):359–366, 1989.

[10] 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.

[11] Y. Lecun, L. Bottou, Y. Bengio ja P. Haffner: *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86(11):2278–2324, Nov 1998, ISSN 0018-9219.

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

[13] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard ja Lawrence D. Jackel: *Backpropagation Applied to Handwritten Zip Code Recognition.* Neural Computation, 1(4):541–551, 1989. https://doi.org/10.1162/neco.1989.1.4.541.

[14] Matt Mazur: *A step by step backpropagation example*, 2015. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/, vierailtu 2018-05-07.

[15] 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.

[16] Andrew Y. Ng: *Feature selection, L 1 vs. L 2 regularization, and rotational invariance.* Teoksessa *Proceedings of the twenty-first international conference on Machine learning*, sivu 78. ACM, 2004.

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

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

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

[20] 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.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg ja Fei-Fei Li: *ImageNet Large Scale Visual Recognition Challenge.* International Journal of Computer Vision, 115(3):211–252, 2015.

[22] 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.

[23] 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.

[24] 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.

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke ja Andrew Rabinovich: *Going Deeper with Convolutions.* CoRR, abs/1409.4842, 2014. `http://arxiv.org/abs/1409.4842`.