Kuvantunnistus neuroverkoilla ja analogisen laskennan mahdollisuudet sen tehostamisessa

Turun yliopisto
Tietotekniikan laitos
LuK-tutkielma
Tietojenkäsittelytiede
Maaliskuu 2026
Olli Piili

TURUN YLIOPISTO
Tietotekniikan laitos

Olli Piili: Kuvantunnistus neuroverkoilla ja analogisen laskennan mahdollisuudet sen tehostamisessa

LuK-tutkielma, 26 s.
Tietojenkäsittelytiede
Maaliskuu 2026

Tekoäly on tietokoneille laskennan, muistin käytön ja energiankulutuksen kannalta erittäin vaativaa. Kuvantunnistuksessa, tekoälyn haarassa, hyödynnettävät neuro- ja konvoluutioverkot vaativat kehityksen myötä jatkuvasti enemmän laskentatehoa ja sitä myötä myös kuluttavat enemmän energiaa. Digitaalisen tietokoneen tueksi on kehitetty analogisia teknologioita tehostamaan tekoälyn vaatimia laskennan ja muistin tarpeita.

Tämä kandidaatintutkielma toteutettiin kirjallisuuskatsauksena. Tutkielman lähteinä toimivat pääasiallisesti vertaisarvioidut tieteelliset artikkelit sekä alalla toimivien yritysten verkkosivut ja julkaisut. Tutkielmassa käsitellään digitaalisen ja analogisen laskennan eroavaisuuksia, esitellään tekoälyyn liittyviä tekniikoita, jotka ovat kuvantunnistuksen toiminnan ymmärtämisen kannalta olennaisia ja perehdytään syvällisemmin yhteen kuvantunnistuksen saralla ansioituneeseen kuvantunnistusohjelmaan. Lopuksi esitellään teknologia, joka hyödyntää analogista laskentaa kuvantunnistuksen tehostamiseen ja pohditaan sen tulevaisuutta.

Konvoluutioverkot, joilla pystyy suorittamaan huomattavan tarkkaa kuvantunnistusta, vaatii miljoonien parametrien tarkkaa säätämistä, joka vie valtavan määrän energiaa ja laskentatehoa. Analogisin menetelmin pystytään vähentämään energiankulutusta ja tehostamaan laskentaa, mutta teknologia ei ole vielä valmis kuluttajamarkkinoille ja vaatii vielä valtavan määrän uutta tutkimusta.

Asiasanat: neuroni, konvoluutioneuroverkko, tekoäly, Analog In Memory Computing, VGGNet-16

Sisällys

Kuvat

Taulukot

1 Johdanto

Tekoälyn räjähdysmäisen suosion kasvun sekä kuvantunnistuksessa käytettävien neuro- ja konvoluutioverkkojen kehityksen seurauksena perinteisen digitaalisen tietokoneen arkkitehtuurin on vaikea pysyä kasvavien laskentatarpeiden kehityksen mukana. Digitaalisten tietokoneiden kehitys perustuu transistorien jatkuvaan pienentämiseen, mutta transistorien fyysiset rajat alkavat tulla vastaan. Kun komponenttien pienentäminen ei enää mahdollista merkittävää suorituskyvyn kasvua, laskentatehoa joudutaan lisäämään kasvattamalla prosessorien kokoa. Koon kasvattaminen puolestaan lisää energiankulutusta sekä resurssien tarvetta. Tämän vuoksi on ajankohtaista tutkia vaihtoehtoisia menetelmiä.

Tämä kandidaatintutkielma käsittelee analogisen laskennan hyödyntämistä kuvantunnistuksessa. Tutkielman alussa tutustutaan digitaalisen- ja analogisen tietokoneiden eroihin sekä tekoälyn ja kuvantunnistuksen periaatteisiin digitaalisissa järjestelmissä. Tarkoituksena on tutustua ensin jo vakiintuneisiin prosesseihin tehdä kuvantunnistusta, jonka jälkeen tutkia analogisen implementoinnin hyötyjä, että haittoja.

Tämän kandidaatintutkielman tavoitteena on perehtyä siihen, miten kuvantunnistusta suoritetaan digitaalisissa järjestelmissä, mitkä ovat sen vahvuudet ja heikkoudet ja selvittää miten analoginen laskenta voi tehostaa jo käytössä olevia vakiintuneita digitaalisia kuvantunnistusjärjestelmiä. Näiden tavoitteiden perusteella tutkimuskysymyksiksi muodostuivat:

TK1 Miten analogista laskentaa voidaan hyödyntää kuvan tunnistuksessa?

TK2 Mitkä ovat analogisen laskennan hyödyt ja haitat kuvan tunnistuksessa?

Tietokoneiden laskentatehovaatimuksien, että virrankulutuksen kasvaessa sekä digitaalisten tietokoneiden käyttämien transistorien pienentyessä lähelle fysikaalisia rajoja on tärkeää löytää uusia tapoja tehostaa laskentaa. Analogisten tietokoneiden hyödyntämistä tarkasteltaessa tutkitaan, miten analogista laskentaa voidaan hyödyntää tehostamaan jo nykyisin olemassa olevia digitaalisia järjestelmiä. Tämän tutkielman tavoitteena ei ole esittää täysin toimivaa analogiseen laskentaan perustuvaa kuvantunnistusjärjestelmää, vaan esitellä erilaisia analogisia lähestymistapoja, joilla voidaan toteuttaa jo vakiintuneita digitaalisia kuvantunnistusteknologioita.

Tämä kandidaatintutkielma suoritettiin kirjallisuuskatsauksena, tavoitteena tarkastella neuroverkkoja ja koneellista kuvantunnistusta ilmiönä, sekä tutkia analogisen laskennan mahdollisuuksia näiden menetelmien tehostamiseen. Tutkielman kirjallisuuslähteinä käytettiin Google Scholar- hakukonetta sekä Turun yliopiston digitaalista Volter-tietokantaa. Tutkimukseen käytetyt lähteet koostuivat tieteellisistä vertaisarvioiduista artikkeleista, joihin löytyy lukuoikeus Volterin kautta. Osa artikkeleista on kaikille avoimia. Eri analogisten mikrosirujen ominaisuuksiin tutustuttiin myös valmistajien verkkosivujen kautta.

Tutkielman alussa esitellään ensin yleisesti tietokoneiden toimintaa sekä tarkastellaan analogisen ja digitaalisen tietokoneiden eroja. Tämän jälkeen avataan tekoälyn ja kuvantunnistuksen perusperiaatteita sekä tutustutaan neuroverkkojen toimintaan ja kouluttamiseen. Kolmannessa luvussa kuvataan digitaalinen tapa toteuttaa kuvantunnistusta, jonka jälkeen neljännessä luvussa esitellään analogisia ratkaisuja digitaalisiin ongelmiin sekä arvioidaan näiden ratkaisujen mahdollisia haittapuolia.

2 Kuvantunnistuksen teknologioita

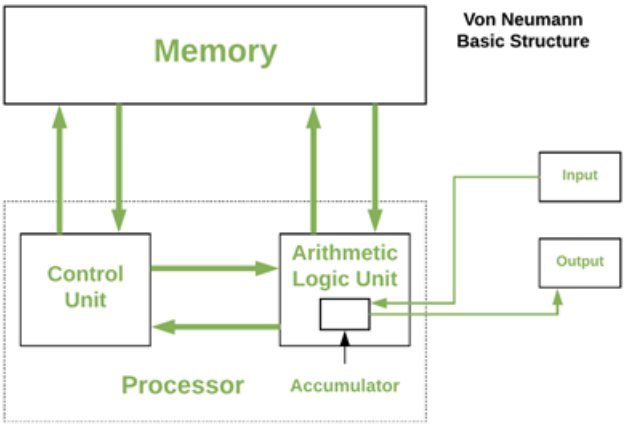
Tässä luvussa käsitellään lyhyesti tutkimukseen liittyviä käsitteitä, teknologioita sekä niiden eroavaisuuksia. Erilaisten tietokoneiden toimintaan sekä tekoälyn ja kuvantunnistuksen menetelmiin perehdytään vain yleisellä tasolla, eli käsitteisiin ei syvennytä kuin tarvittavan ymmärrettävyyden tasolle.

2.1 Tietokone

Yleiskielessä puhuttaessa tietokoneella tarkoitetaan digitaalista tietokonetta. Kuitenkin ennen transistorien keksimistä **analogiset tietokoneet** olivat tehokkaimpia tietokoneita maailmassa. Analogisten tietokoneiden toiminta perustuu fyysisten ilmiöiden, kuten sähkövirtojen tai mekaanisten liikkeiden, mallintamiseen ja käyttöön laskennassa. Ne käyttävät laskennassaan jatkuvia signaaleja, kuten sähkövirtaa. Sähkövirran jännitetaso voi kuvastaa muuttujaa ja yhdistelemällä sähkövirtoja tietyillä tavoilla voidaan suorittaa matemaattisia laskentoja. Tämä tekee analogisista tietokoneista hyvin tehokkaita. Varsinaista perinteistä laskentaa ei tarvitse suorittaa vaan vastaus saadaan reaaliajassa lukemalla yhdistyneiden sähkövirtojen arvo. Samasta syystä analogisen tietokoneen laskelmat eivät ole täydellisen tarkkoja eikä toistettavia. Koska syötteenä voidaan saada vain jatkuvia ja muuttuvia arvoja reaalimaailmasta tuo se mukanaan myös likimääräisyyttä palautteeseen. Vaikka digitaaliset tietokoneet ovat vallanneet suurimman osan tietokoneiden maailmasta, on

analogisilla tietokoneilla edelleen erityisiä sovelluksia. joissa niiden jatkuvan arvon käsittelytapa on ylivertainen. [1]

Digitaaliset tietokoneet kattavat suurimman osan maailman tietokoneista. Niihin kuuluvat kaikki henkilökohtaiset tietokoneet ja älypuhelimet. Toisin kuin analogiset tietokoneet, ne käsittelevät dataa binäärisessä muodossa diskreetteinä arvoina, eli nollina ja ykkösinä. Tämä mahdollistaa, että samalla syötteellä saa aina saman vastauksen ja näin myös tarkan laskennan. Nykytietokoneet perustuvat Von Neumannin tietokonearkkitehtuuriin, joka on nimetty yhden kehittäjistä John Von Neumannin mukaan. Kuvassa 2.1 esitellään arkkitehtuurin toimintaperiaatetta.



Kuva 2.1: Yksinkertaistettu kaavio Von Neumannin arkkitehtuurista. [2]

Von Neumannin arkkitehtuurin perusidea on, että tietokoneen käyttämät ohjelmat ja niiden data varastoidaan samanlaisiin muistipaikkoihin, joita voidaan lukea ja kirjoittaa uudestaan tarpeen mukaan. Tämä mahdollistaa yleiskäyttöisten tietokoneiden rakentamisen, jotka voivat suorittaa monenlaisia tehtäviä ohjelmiston vaihtamisen avulla. Tietokoneessa on keskusyksikkö (engl. *Central Processing Unit*, CPU), joka suorittaa ohjelmakoodin, hallinnoi tiedonsiirtoa ja muistin käyttöä. Muistiyksikkö koostuu kahdesta pääosasta, välimuistista sekä pitkäaikaisesta muistista. Välimuistiin tallennetaan suoritettavat ohjelmat ja väliaikaisesti käytössä

oleva data, kun taas pitkäaikaiseen muistiin tallennetaan peruskäyttöön tarvittavat ohjelmistot ja pitkäaikainen data.[2]

2.2 Tekoäly

Tekoäly voidaan jakaa kahteen pääluokkaan: heikko tekoäly ja vahva tekoäly. Heikko tekoäly on rajoitetumpi ja erikoistuneempi versio tekoälystä. Se pohjautuu enemmän ohjelmoituihin sääntöihin ja malleihin, eikä se kykene oppimaan ja sopeutumaan ympäristöönsä samalla tavalla kuin vahva tekoäly. Heikko tekoäly on hyödyllinen monissa sovelluksissa, kuten kieliprosessoinnissa, kuvantunnistuksessa ja pelien pelaamisessa, mutta sen toiminta rajoittuu ennalta määriteltyihin tehtäviin. Heikko tekoäly ei voi ymmärtää syvällisesti tietoa tai syy-seuraussuhteita, vaan se toimii ennalta opetettujen ohjeiden mukaisesti. Vahva tekoäly sen sijaan voisi päätellä uusia asioita itsenäisesti ja soveltaa oppimaansa uusiin tilanteisiin. Vahva tekoäly on siis teoreettinen tavoite, joka ylittäisi ihmisen älykkyyden. [3]

Kuvantunnistus on yleinen termi, jolla kuvataan useampaa eri konenäön osa-aluetta, jotka kaikki liittyvät asioiden tunnistamiseen digitaalisista kuvista. Kuvantunnistus voidaan jakaa kolmeen luokkaan; kuvan luokitteluun (engl. *image classification*), esineiden etsimiseen (engl. *object localization*) ja esineiden tunnistamiseen (engl. *object detection*). Kuvien luokittelussa pyritään ennustamaan mitä kuva esittää. Syötteenä annetaan kuva sisältäen yhden tunnistettavan kohteen ja palautteena saadaan ennuste mitä kuva sisältää. Esineiden etsimisessä pyritään etsimään tunnistettavia esineitä valokuvasta. Syötteenä annetaan valokuva ja mahdollinen etsittävä esine ja palautteena saadaan yhdet tai useammat koordinaatit mistä esine kuvasta löytyy. Esineiden tunnistamisessa yhdistetään kaksi edellistä ja pyritään löytämään ja luokittelemaan kaikki kuvasta löytyneet esineet ja palauttamaan niiden koordinaatit sekä ennusteet mitä löydetyt esineet ovat. Esineiden tunnistamisen alaluokka, esineiden segmentointi (engl. *object segmentation*) pyrkii löytämään esi-

neet kuvasta pikselin tarkkuudella. Kuvassa 2.2 esitellään havainnollistetaan nämä kolme kuvantunnistuksen osa-aluetta.[4]

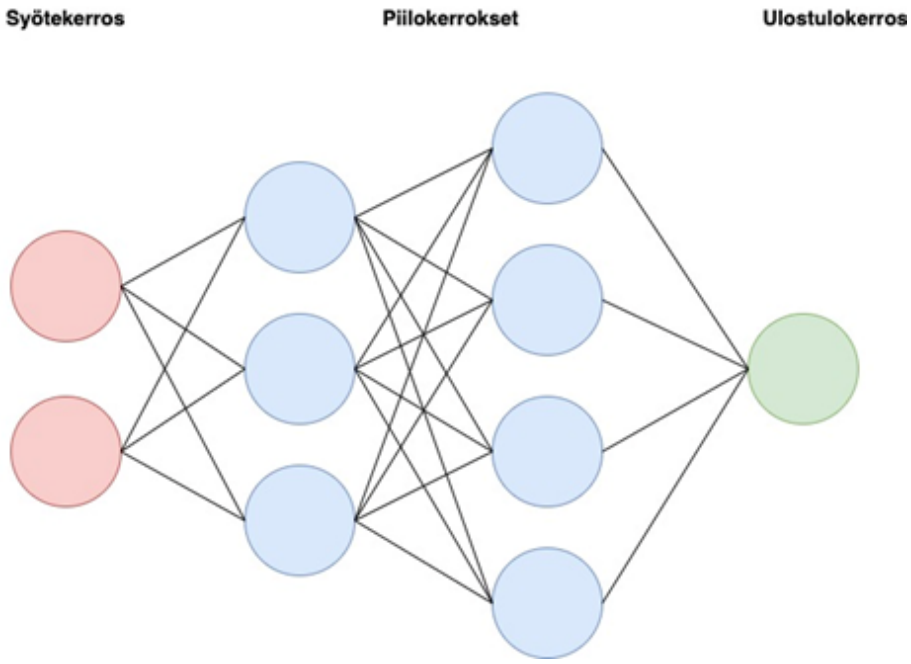
Kuva 2.2: Kuvan tunnistuksen kolme luokkaa; kuvan luokittelu, esineiden etsiminen ja esineiden tunnistaminen.

2.3 Neuroverkot

Perinteinen neuroverkko Neuroverkot (engl. *Artificial Neural Network*, ANN) ovat monimutkaisia laskennallisia järjestelmiä, jotka perustuvat ihmisen aivojen toimintaan. Neuroverkkojen suosio on kasvanut niiden kyvyn vuoksi oppia monimutkaisia tehtäviä kuten käännöksiä, pelien pelaamista ja kuvantunnistusta. Yksinkertaistettuna neuroverkko on suuri joukko toisiinsa kytkettyjä funktioita, joissa aikaisemman funktion lopputulos vaikuttaa seuraaviin funktioihin.[3]

Neuroverkko koostuu useista toisiinsa kytketyistä kerroksesta ja kerrokset rakentuvat yksittäisistä neuroneista (neuron) jotka aktivoituvat, kun ne saavat riittävän syötteen. Syötekerroksessa (engl. *input layer*) neuroverkolle annetaan sen käsiteltävä informaatio ja sen neuronien määrä vaihtelee sen mukaan, miten montaa ominaisuutta syötteestä tutkitaan. Neuronit ovat yhteyksissä toisiinsa liitoksilla, joilla on yksilölliset painokertoimet. Neuroni laskee yhteen liitokset painotetuksi summaksi. Summa viedään neuronin aktivaatiofunktioon, joka määrittää minkälaisen signaalin se lähettää eteenpäin seuraaville neuroneille. Syöte etenee samalla tavalla neuroverkon piilokerroksien (engl. *hidden layer*) ja ulostulokerroksen (engl. *output layer*)

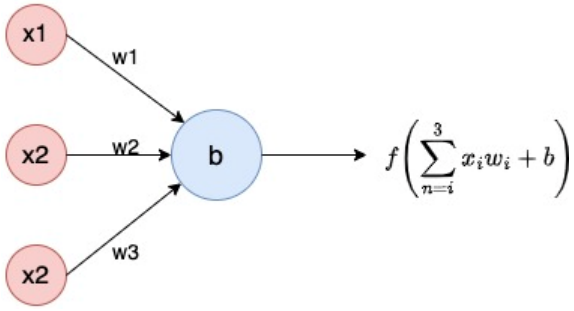
läpi. Piilokerroksia voi olla jopa tuhansia ja useamman piilokerroksen neuroverkkoja kutsutaan syviksi neuroverkoiksi (engl. *Deep Neural Network*, DNN). Kuvassa 2.3 esitellään yksinkertainen neuroververkko jossa kaikki neuronit ovat yhteydessä toisiinsa. [5]



Kuva 2.3: Yksinkertainen neuroverkko jossa kaikki kerroksien neuronit ovat toisiinsa kytkettyinä. Tällaista rakennetta kutsutaan täysin kytketyksi kerrokseksi (engl. *Fully Connected Layer*).

Aktivaatiofunktio määrittää minkälaista informaatiota neuroni lähettää eteenpäin. Neuroni ottaa syötteitä toisilta neuroneilta tai syötekerroksessa ulkoisesta lähteestä. Jokaisella syötteellä on paino (engl. *weight*) mikä määrittää syötteen tärkeyden verrattuna muihin syötteisiin. Syötteistä lasketaan painotettu summa ja viedään aktivaatiofunktioon. Kuvassa 2.4 esitellään havainnointi neuronin toiminnasta.

Neuroni ottaa syötteenä numerolliset arvot x1, x2 ja x3 joilla jokaisella on painot w1..w3. Lisäksi neuronilla on itsellään arvo b (engl. *bias*), eli vakiotermi, jonka tarkoitus on antaa neuronille kyky muokata itseään. Kasvattamalla tai pienentämällä arvoa b saadaan neuroni aktivoitumaan pienemmillä syötteillä tai vastaavasti vaikeuttamaan aktivoitumista. Arvot lasketaan yhteen ja viedään aktivaatiofunktioon.



Kuva 2.4: Neuroni lisää omaan arvoonsa b syötteiden painotetun summan ja vie sen aktivaatiofunktioon f.

Aktivaatiofunktion tehtävä on tuoda epälineaarisuutta verkkoon. Epälineaarisuuden lisääminen verkkoon on tärkeää koska reaalimaailman informaatio on usein epälineaarista. Aktivointifunktioita on erilaisia ja niiden tarkoitus on pitää neuronin palautuksien arvot tietyissä rajoissa. Esimerkkejä yleisesti käytetyistä aktivaatiofunktioista on esitetty kuvassa 2.5.

Sigmoid-aktivaatiofunktio ottaa syötteenä luvun ja palauttaa arvon väliltä 0 ja 1.

$$\sigma(x) = 1/(1 + exp(-x))$$

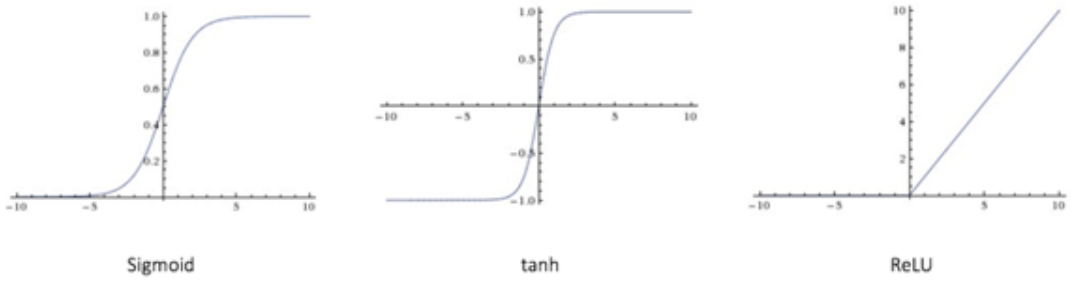
Tanh-aktivaatiofunktio ottaa syötteenä luvun ja palauttaa arvon väliltä -1 ja 1.

$$tanh(x) = 2\sigma(2x) - 1$$

ReLU-aktivaatiofunktio ottaa syötteenä luvun ja palauttaa negatiiviset luvut nollana.

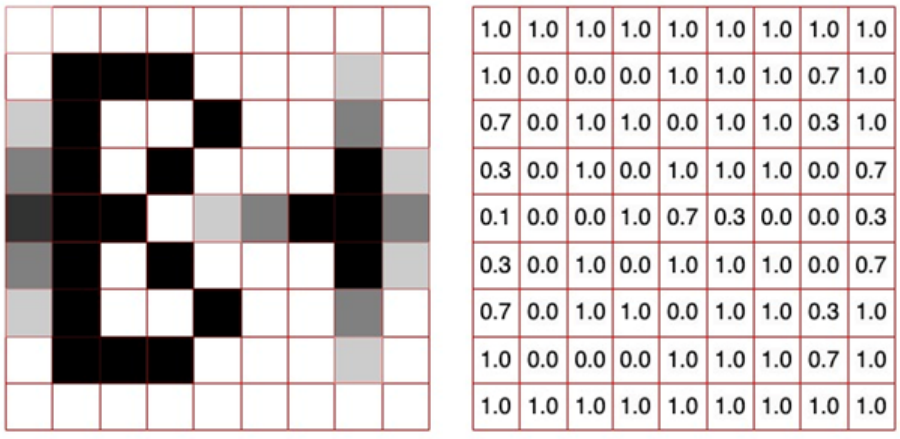
$$f(x) = max(0, x)$$

Konvoluutioneuroverkot (engl. *convolutional neural network*, CNN) ovat yksi neuroverkkojen tyyppi, joka on erikoistunut käsittelemään dataa, jota voidaan esittää ruudukkona, kuten kuvia tai videoita. Konvoluutioneuroverkkoja hyödynnetään siis yleisesti kuvantunnistuksessa ja sen toiminta perustuu kuvan eri piirteiden poimimiseen ja niiden analysointiin. Yksinkertaisten piirteiden kuten reunojen, viivojen



Kuva 2.5: Aktivaatiofunktiot Sigmoid, tanh ja ReLU.

tai yksinkertaisten muotojen etsintä tapahtuu monessa kerroksessa, eli kuvasta tehdään monta erilaista versiota (engl. *feature map*). Yksinkertaisia muotoja yhdistelemällä verkko oppii tunnistamaan kuvasta isompia kokonaisuuksia kuten eläimiä tai kasveja. Tietokone näkee digitaaliset kuvat pitkänä joukkona arvoja, jotka on järjestetty ruudukkomaisesti. Jokainen arvo vastaa pikseliä ruudulla ja ilmaisee pikselin kirkkauden ja värin. Kuvassa 2.6 pikselin kirkkaudet on yksinkertaisuuden vuoksi esitetty arvoilla 0.0–1.0, mutta värikuvien kohdalla asteikko 0–255 on yleisesti käytössä. [6]



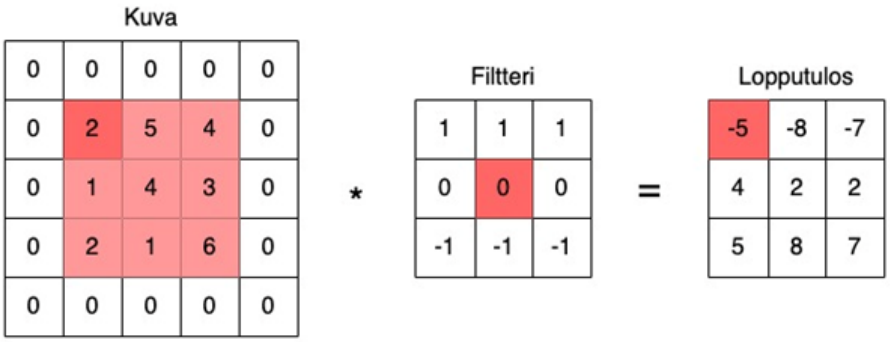
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.7	1.0
0.7	0.0	1.0	1.0	0.0	1.0	1.0	0.3	1.0
0.3	0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.7
0.1	0.0	0.0	1.0	0.7	0.3	0.0	0.0	0.3
0.3	0.0	1.0	0.0	1.0	1.0	1.0	0.0	0.7
0.7	0.0	1.0	1.0	0.0	1.0	1.0	0.3	1.0
1.0	0.0	0.0	0.0	1.0	1.0	1.0	0.7	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Kuva 2.6: Yksinkertainen kuva pikselitaulukkona esitettynä jossa värit voivat saada arvon välillä 0-1

Mustavalkoiset kuvat pystytään esittämään yhdellä kaksiulotteisella taulukolla. Värillisen digitaalikuvan esittämiseen tarvitaan kolme erillistä taulukkoa. Kaikki eri-

laiset värit pystytään esittämään yhdistelemällä kolmea pääväriä: punaista, vihreää ja sinistä. Digitaaliset värikuvat ovat siis kolme pikselitaulukkoa päällekkäin, joissa jokainen edustaa yhtä väriä. [6]

Konvoluutioneuroverkon nimi tulee sen keskeisestä toimintaperiaatteesta, eli vähintään sen yhdessä kerroksessa suoritetaan konvoluutio-operaatioita. Konvoluutiossa alkuperäisestä kuvasta luodaan erilaisten suodattimien kanssa uusia kuvia. Jokainen suodatin etsii kuvasta erilaisia piirteitä. Kuvan jokaiselle pikselille lasketaan uusi arvo sen naapuripikselien, suodattimen painokertoimien ja pikselin omaan arvoon perustuen. Suodatinta kuljetetaan kuvan päällä ja kerrotaan jokaisen pikselin ja sen ympäröimien pikselien arvot suodattimen arvojen kanssa ja summataan luvut yhteen. Näin kuvan jokaiselle pikselille saadaan uusi arvo. Uudet pikselit muodostavat kuvan, joka sisältää suodattimen kuvaavia piirteitä. Kuvassa 2.7 esitetään yksinkertainen esimerkki konvoluutiosta, jossa suodattimella etsitään vaakasuoria reunoja. [7]



Kuva 2.7: Konvoluution periaate

Kuvan pikselit operoidaan suodattimella. Suodatin asetetaan yksitellen jokaisen pikselin päälle. Suodattimen alle jäävät arvot kerrotaan suodattimen painokertoimilla ja summataan lopputuloksen uudeksi pikseliksi. Kuvassa 4 esitetyn lopputuloksen numero -5 saadaan kuvan tummanpunaista pikseliä käsittelemällä. Suodattimen tummanpunainen arvo 0 asetetaan kuvan tummanpunaisen pikselin päälle ja suoritetaan alle jäävät yhdeksän kertolaskua ja summataan ne yhteen: 0 * 1 + 0 * 1 +

0 * 1 + 0 * 0 + 2 * 0 + 5 * 0 + 0 * -1 + 1 * -1 + 4 * -1 = -5. Pikselin arvosta 2 tulee siis -5. Seuraavaksi siirrytään yksi pikseli oikealle ja suoritetaan samat toimenpiteet ja lopputuloksen seuraavan pikselin arvoksi saadaan -8. Näin käsitellään rivin loput pikselit, jonka jälkeen siirrytään yksi rivi alaspäin. Näin käsitellään kuva kokonaisuudessaan. [7]

Jotta neuroverkot toimivat halutulla tavalla, tulee verkkoa kouluttaa. Verkko oppii suorittamaan tehtäviä analysoimalla esimerkkejä halutusta lopputuloksesta. Usein koulutusaineisto on tehty ennakkoon, esimerkiksi kuvantunnistuksen tapauksessa koulutusaineisto voisi olla tuhansien kuvien joukko ennalta luokiteltuja kuvia. Verkolle syötetään kuvia esimerkiksi eläimistä tai kasveista ja verkko oppii yhdistämään tietynlaiset kuviot tiettyihin luokkiin. Tätä kutsutaan ohjatuksi oppimiseksi. Erilaisia koulutustapoja ja -algoritmeja on lukuisia ja tilanteeseen tulisi valita sopiva koulutustapa.

Yksi yleisimmistä koulutusalgoritmeista on vastavirta-algoritmi (engl. *backpropagation*). Vastavirta-algoritmin perusajatus on käyttää differentiaalilaskentaa neuroverkon painojen optimoinnissa. Algoritmin nimi johtuu sen toimintaperiaatteesta. Algoritmi aloittaa virheen etsimisen ja uusien painojen laskemisen neuroverkon ulostulokerroksesta ja etenee verkkoa käänteisessä järjestyksessä takaisin syötekerrokseen. Vastavirta-algoritmia käytetään yhdessä koulutusaineiston kanssa. Neuroverkon palautetta verrataan koulutusaineiston ratkaisuun ja takaisin alkuun iteroimalla painoja muokataan siten että palaute olisi mahdollisimman lähellä ratkaisua. [5], [8]

3 Kuvantunnistus VGGNet-16 -konvoluutioverkolla

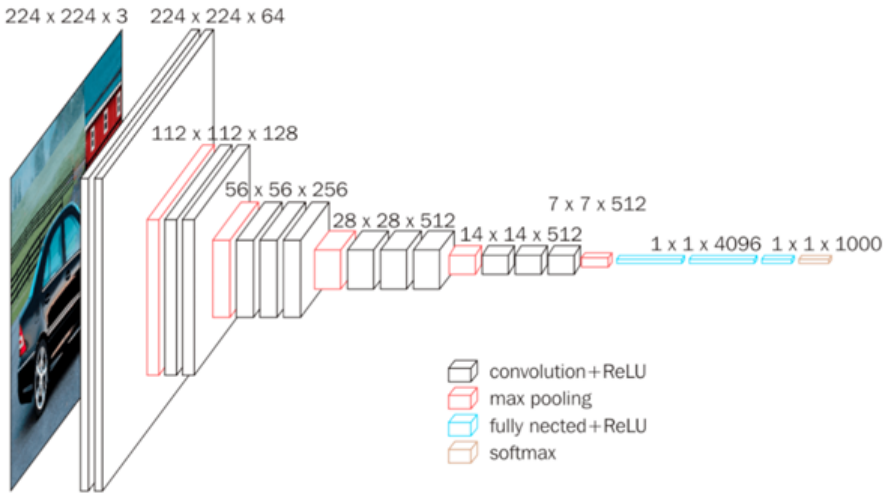
Tässä luvussa esitellään kuvantunnistuksen vaiheet. Työssä esitelty tapa ei ole ainoa, eikä välttämättä ajankohtaisesti paras ja tehokkain tapa tehdä kuvantunnistusta. Kuinka monta ja minkälaisia kerroksia verkko sisältää, on aina tapauskohtaista ja tulee ottaa huomioon mallin suunnittelussa. Esimerkkinä on käytetty VGGNet-16 -arkkitehtuuria mutta vaihtoehtoisia, vielä kehittyneempiä malleja löytyy. Esitellyssä versiossa on 16 kerrosta, mutta VGGNet:stä on myös syvempi kerroksisia versioita. Tässä työssä VGGNet:llä tarkoitetaan sen 16 kerroksista versiota. [9]

Verkon nimi, VGGNet, tulee Oxfordin yliopiston tiede ja insinööriosastoon kuuluvalta ryhmältä nimeltään Visual Geometry Group. Sen alkuperäinen tarkoitus oli tutkia miten konvoluutioverkkojen syvyys vaikuttaa tehokkuuteen kuvantunnistustehtävissä. Verkko sijoittui ensimmäiselle sijalle esineiden etsimisessä ja toiselle sijalle kuvan luokittelussa vuonna 2014 järjestetyssä ImageNet ILSVRC-2014 kuvantunnistuskilpailussa. ImageNet on valtava kuvien kirjasto, jossa on jopa 14 miljoonaa käsin tunnistettua ja kategorisoitua valokuvaa. Kirjaston materiaalit ovat vapaassa käytössä ja ovat avaintekijänä konenäön ja tekoälyn kehityksen vauhdittamisessa. Kilpailun tavoitteena on kehittää neuroverkko, joka antaa mahdollisimman pienen top-5 luokitteluvirhearvon. Top-5 luokitteluvirhearvolla tarkoitetaan kuinka

isolla prosentilla verkon viiden parhaan arvauksen joukossa ei ole oikeaa vastausta. VGGNet saavutti top-5 luokitteluvirhearvon 7.5 %. [9], [10]

3.1 Arkkitehtuuri

VGGNet on pohjimmiltaan hyvin tavanomainen konvoluutioneuroverkko. Sille tyypillistä on sen käyttämä erittäin pienet (3x3) konvoluutiofiltterit, mutta verkon syvyys johti merkittävään tehokkuuden kasvuun aikaisempiin teknologiohin verrattuna. Julkaisuhetkellään VGGNet:in 16 kerroksinen verkko oli alansa syvimpiä. VGGNet sisältää yhteensä kolmetoista konvoluutiokerrosta, kolme täysin kytkettyjä kerrosta sekä näiden lisäksi viisi koontikerroksia (engl. *pooling layer*) joita ei tyypillisesti lasketa mukaan verkon syvyyteen. Kuvassa 3.1 esitellään VGGNet:in arkkitehtuuri. Käsitteet max pooling, fully connected ja softmax esitellään syvällisemmin myöhemmin. [9], [11], [12]

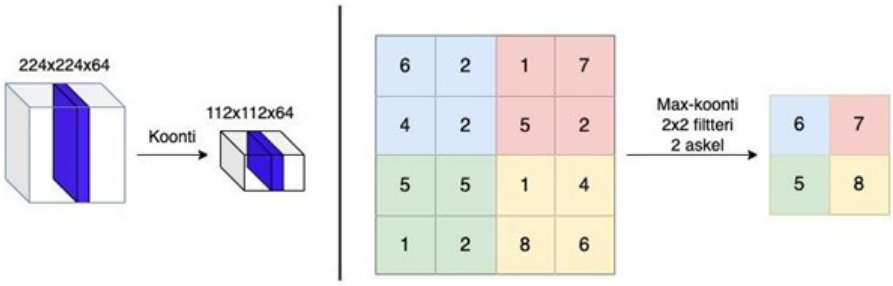


Kuva 3.1: VGGNet:in arkkitehtuuri. [11]

VGGNet ottaa syötteeksi 224x224 pikselisen värikuvan. Jokaisessa konvoluutiokerroksessa suoritetaan kahdesta neljään konvoluutio-operaatiota 3x3 kokoisella suodattimella (engl. *kernel*). Suodattimen liikkumis askel (engl. *stride*) vaihtelee yhden ja kahden pikselin välillä. Erilaisten suodattimien määrä kasvaa mitä syvemmällä

verkossa operoidaan, päättyen jopa 512 erilaiseen suodattimeen yhdessä kerroksessa. Näin myös uusia piirrekuvia (engl. *feature map*) luodaan vastaava määrä jokaisessa kerroksessa.

Verkossa suoritetaan Max-koontia (engl. *max pooling*) viidessä eri kerroksessa. Koontikerroksien tarkoituksena on vähentää kerroksien tilankäyttöä ja pienentää laskentavaatimuksia. Koontikerroksessa jokainen piirrekuva pienennetään käyttäen Max-operaatiota 2x2 kokoisella suodattimella. Suodatinta kuljetetaan samaan tapaan kuin luvussa 3 esitelty konvoluutiosuodatinta mutta jokaisen pikselin päällä käydään vain kerran eli askel on 2. Suodatin asetetaan neljän ensimmäisen pikselin päälle, joista arvoltaan suurin viedään tulokseksi. Kuvassa 3.2 havainnollistetaan koontikerroksen toimintaa. [12], [13]



Kuva 3.2: Koontikerrokset tiivistävät piirrekartat käyttäen Max-operaatioita.

Konvoluutio- ja koontikerroksien jälkeen syöte etenee täysin kytkettyihin kerroksiin, joiden tehtävänä on laskea prosessoitujen piirrekuvien arvojen perusteella mitä alkuperäinen kuva sisältää. Piirrekuvien informaatio muutetaan vektorimuotoon perinteistä neuroverkkoarkkitehtuuria varten. Viimeisen koontikerroksen jälkeen piirrekuvien dimensio on $7\times7\times512$, mikä muunnetaan vektorimuotoon. Tämä 25 088 elementin pituinen vektori toimii syötteenä täysin kytketyille kerroksille, joissa suoritetaan kuvan varsinainen luokittelu. Täysin kytketyssä kerroksessa jokainen neuroni on yhteydessä kaikkiin edellisen kerroksen neuroneihin, jolloin jokainen neuroni saa syötteekseen koko vektorin. Neuroni laskee painotetun summan syötevektorin

arvoista sekä lisää vakio-termin, minkä jälkeen suoritetaan aktivaatiofunktio ReLU. Näin yksittäisen neuronin laskeva funktio voidaan esittää:

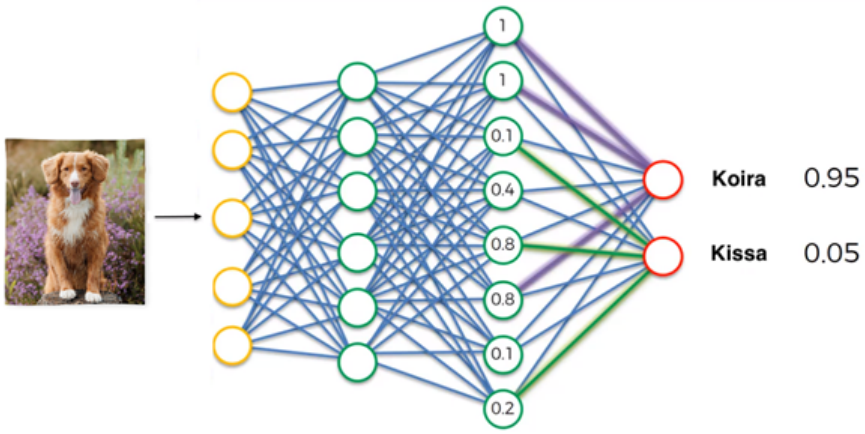
$$y = \max(0, w_1 x_1 + w_2 x_2 + \cdots + w_{25088} x_{25088} + b)$$

Seuraavissa kerroksissa jokainen neuroni saa syötteekseen edellisen kerroksen 4096 tulosta ja laskevat niistä painotetun summan samalla periaatteella. Viimeinen näistä kerroksista vastaa luvussa 2 esiteltyä ulostulokerrosta ja antaa palautteena ennustuksen mitä kuvassa esiintyy. Ulostulokerroksessa on tuhat neuronia, joista jokainen kuvastaa yhtä luokkaa eli VGGNet pystyy siis tunnistamaan kuvista tuhat erilaista esinettä tai asiaa. Taulukossa 3.1 esitellään VGGNet numeroina.

Taulukko 3.1: VGGNet:in arkkitehtuuri

	Layer	Feature Map	Size	Kernel size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 x Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 x Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 x Convolution	256	56 x 56 x 128	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 x Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 x Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Jokaisessa kerroksessa käytetään ReLU-aktivaatiofunktiota paitsi ulostulokerroksessa missä suoritetaan Softmax-funktio joka antaa tuloksena ennustukset kuvan sisällöstä. Kuvassa 3.3 on havainnollistettu Softmax-funktion toimintaa yksinkertaisessa luokitteluesimerkissä. Funktio antaa tuloksen muodossa, jossa ennustuksen oikeellisuuden todennäköisyys ilmoitetaan arvoilla nollan ja ykkösen väliltä. Softmax-funktio myös varmistaa, että ennustuksien summa on yhteensä tasan 1.0 jotta ennustukset ovat helposti vertailukelpoisia, mikä auttaa verkon koulutusfunktioiden toiminnassa. [9], [11]–[13]



Kuva 3.3: Softmax-funktio varmistaa, että ennustukset ovat vertailukelpoisia

Kuvassa 3.3 verkko luokittelee syötekuvan kahteen luokkaan, koiraan ja kissaan. Softmax-funktio muuntaa verkon ulostulon todennäköisyyksiksi, jolloin tässä esimerkissä verkko ennustaa kuvan esittävän koiraa todennäköisyydellä 0.95 ja kissaa todennäköisyydellä 0.05.

3.2 Koulutus ja resurssien käyttö

VGGNet-16 ei ole nykypäivän standardeinkaan pieni verkko. VGGNet sisältää noin 138 miljoonaa säädettävää parametria, jotka on varastoitava muistiin. Jokainen vaikuttaa osaltaan mallin ennustuksen lopputulokseen, joten niitä on hienovaraisesti säädettävä, kunnes verkko tuottaa halutun lopputuloksen. Verkon kouluttaminen kesti tauotta noin kahdesta kolmeen viikkoon käyttäen neljää, aikansa yhtä tehokkaimmista, Nvidia Titan Black -näytönohjainta samanaikaisesti. Koulutusmateriaalina käytettiin kuvia, jotka oli aikaisemmin luokiteltu tuhanteen erilaiseen luokkaan. Koulutukseen käytettiin 1.3 miljoonaa uniikkia kuvaa, validointiin 50 tuhatta kuvaa ja verkon testaukseen vielä 100 tuhatta kuvaa. Taulukossa 3.2 esitellään VGGNetin muistin käyttöä. [14]

Taulukossa esitetään VGGNet-16-verkon kerroskohtaiset ulostulokoot, muistinkäyttö sekä parametrien eli painojen määrät. Verkko muodostuu konvoluutioker-

Taulukko 3.2: VGGNet-16 -verkon kerrosten ulostulokoot, muistinkäyttö ja painojen määrä.

Kerros	Ulostulon koko	Muisti	Painot
Input	$224 \times 224 \times 3$	150k	0
Conv3-64	$224 \times 224 \times 64$	3.2M	1 728
Conv3-64	$224 \times 224 \times 64$	3.2M	36 864
Pool2	$112 \times 112 \times 64$	800k	0
Conv3-128	$112 \times 112 \times 128$	1.6M	73 728
Conv3-128	$112 \times 112 \times 128$	1.6M	147 456
Pool2	$56 \times 56 \times 128$	400k	0
Conv3-256	$56 \times 56 \times 256$	800k	294 912
Conv3-256	$56 \times 56 \times 256$	800k	589 824
Conv3-256	$56 \times 56 \times 256$	800k	589 824
Pool2	$28 \times 28 \times 256$	200k	0
Conv3-512	$28 \times 28 \times 512$	400k	1 179 648
Conv3-512	$28 \times 28 \times 512$	400k	2 359 296
Conv3-512	$28 \times 28 \times 512$	400k	2 359 296
Pool2	$14 \times 14 \times 512$	100k	0
Conv3-512	$14 \times 14 \times 512$	100k	2 359 296
Conv3-512	$14 \times 14 \times 512$	100k	2 359 296
Conv3-512	$14 \times 14 \times 512$	100k	2 359 296
Pool2	$7 \times 7 \times 512$	25k	0
FC	4096	4096	102 760 448
FC	4096	4096	16 777 216
FC	1000	1000	4 096 000
Kokonaismuisti		24M×4 tavua ≈93MB/kuva (eteenpäin; ≈ ×2 taaksepäin)	—
Kokonaisparametrit		—	138M

roksista ja täysin kytketyistä kerroksista, joista jälkimmäiset oppivat suodattimien avulla kuvasta paikallisia piirteitä ja jälkimmäiset vastaavat lopullisesta luokittelusta. Taulukon perusteella verkon muistitarve on noin 93 MB kuvaa kohden eteenpäinlaskennan aikana, jolloin syöte etenee kerroksittain verkon läpi ja tuottaa ennusteen kuvan sisällöstä. Koulutusvaiheessa tämän lisäksi suoritetaan vastavirta-algoritmiin kuuluva taaksepäinlaskenta, jossa virhesignaalin perusteella lasketaan painoille uudet arvot. Tällöin kerrosten välisiä aktivaatioita ja muita välituloksia on säilytettävä muistissa, minkä vuoksi kokonaismuistitarve kasvaa tyypillisesti noin kaksinkertaiseksi verrattuna pelkkään eteenpäinlaskentaan.[12]

4 Muistissa laskeminen

Neuroverkot ovat tietokoneelle hyvin raskaita operaatioita. Varsinkin verkkojen koulutus on erityisen raskasta ja vaatii miljoonia iterointeja. Jotta pystytään operoimaan isompia neuroverkkoja ja nopeuttamaan sen vaatimaa laskentaa on pystyttävä kasvattamaan laskentatehoa ja lisäämään sekä nopeuttamaan muistia. Tietokoneiden nopeuttamisen kannalta on havaittavissa kolme suurempaa pullonkaulaa: laskentatehon kasvattaminen, virrankulutus ja muistin kaistanleveys.

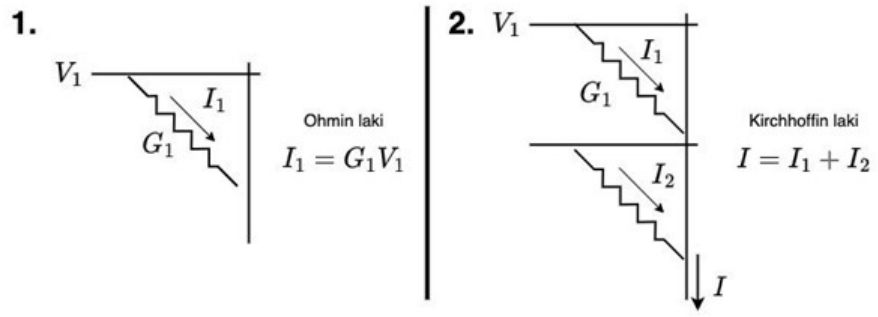
Tietokoneiden laskentateho on kasvanut viimeisen viidenkymmenen vuoden ajan suhteellisen tasaisesti. Laskentatehon kasvu on seurannut hyvin tarkasti Mooren lakia. Ennustuksen mukaan transistorien tiheys kuluttajahintaisissa prosessoreissa tuplaantuu noin joka toinen vuosi. Transistorien määrä vaikuttaa määräävästi prosessorin laskentatehoon. Nyt ennustuksen toteutuminen alkaa hidastua, kun transistorien koko alkaa lähentelemään fysikaalisia rajoja. Tämän jälkeen laskentatehoa on helpointa nostaa kasvattamalla prosessorien kokoa, mutta samalla kasvaa myöskin prosessorin virrankulutus. [15]

Virrankulutus on toinen erittäin rajoittava tekijä. Kehittyneet prosessorit käyttävät noin yhden pico-joulen (pJ) energiaa yhdessä laskenta operaatiossa ja ison neuroverkon kouluttamiseen tarvitaan satoja miljardeja operaatioita pelkästään laskemiseen. Laskettujen operaatioiden muistiin kirjoittamiseen ja sieltä hakemiseen kuluu energiaa vielä enemmän kuin itse laskemiseen. [16], [17]

Kolmas merkittävä pullonkaula on muistin kaistanleveys. Neuroverkot eivät ole pelkästään laskennallisesti raskaita ohjelmia vaan myös muistin käytöllisesti. Perinteisten digitaalisten tietokoneiden käyttämän, luvussa 2 esitelty, Von Neumann-arkkitehtuurin on vaikea vastata muistillisesti raskaiden ohjelmien vaatimuksiin ja tätä kutsutaan Von Neumannin pullonkaulaksi (engl. *Von Neumann bottleneck*). Arkkitehtuuri luo kolme pääongelmaa: (1) Datan liikuttelu muistin ja prosessorin välillä luo huomattavan viiveen hidastaen koko tietokoneen toimintaa; (2) Muistin kaistanleveys ei riitä. Tiedon haku muistista vie enemmän aikaa kuin itse tiedon prosessointi; (3) Muistin käyttö on erittäin energiaa kuluttavaa. Tiedon liikutteluun prosessorin ja muistipaikkojen välillä voi viedä jopa sata kertaa enemmän energiaa kuin saman tiedon prosessointiin kuluu. [18]

Aikaisemmin kappaleessa esiteltyjen pullonkaulojen voittamiseksi on kehitteillä teknologioita, joissa hyödynnetään analogista laskemista. Analoginen muistissa laskeminen (engl. *Analog In-Memory Computing*, AIMC) on uusi kehitteillä oleva tapa laskea neuroverkoissa tehtävää matriisilaskentaa. Matriisilaskentaa käytetään neuroverkoissa neuronien painojen kuin konvoluutio-operaatioidenkin laskentaan. Sen sijaan että laskenta ja siihen tarvittavan datan säilytys tapahtuisi eri paikoissa, rakennetaankin uusi siru, joka suorittaa molempia. [19]

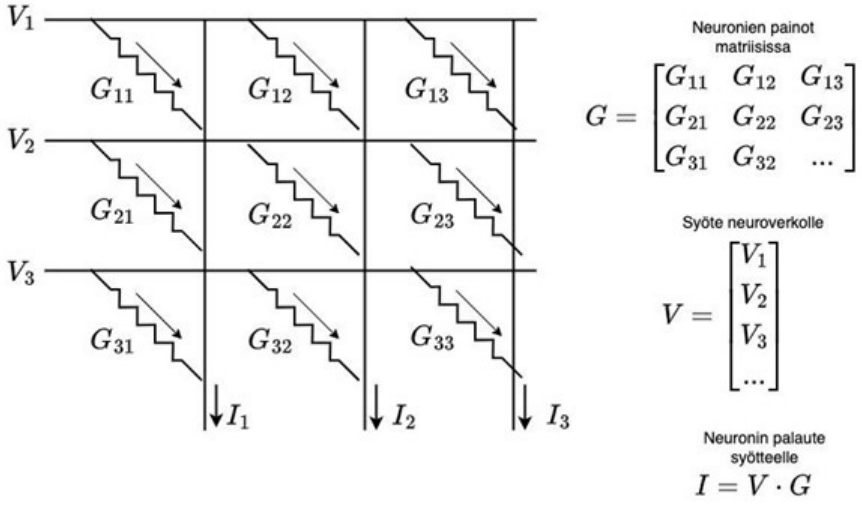
Analogisesti voi summata ja kertoa lukuja yhteen käyttämällä Ohmin- ja Kirchhoffin piirilakeja, jotka esitellään kuvassa 4.1.



Kuva 4.1: Tulo ja yhteenlasku hyödyntämällä Ohmin ja Kirchhoffin lakeja.

Tulo saadaan käyttämällä Ohmin lakia. Kuvassa 4.1 osassa 1 on kahden metallijohdon väliin laitettu resistori jonka virranjohtavuus on ohjelmoitu arvoksi G_1.

Johtoon V_1 syötetään virtaa ja Ohmin lain mukaan tulo saadaan laskettua lukemalla sähkövirran voimakkuus I_1. Kirchhoffin lakia käyttämällä saadaan suoritettua yhteenlaskuja. Kuvan 4.1 osassa 2 piiriä on jatkettu toisella resistorilla. Arvojen $I_1 + I_2$ summa saadaan lukemalla johdon päästä tulevan sähkövirran voimakkuus. [19], [20] Kuvassa 4.2 on aikaisemmin esiteltyjä komponentteja käyttämällä rakennettu ruudukkomainen piiri.



Kuva 4.2: Analogisesti pystyy myös suorittamaan matriisilaskentaa.

Resistorit $G_(11)...G_(33...)$ ovat ohjelmoitavissa kuvastamaan neuronien painoja. Jokaiselle riville $V_1...V_(3...$ syötetään virtaa tavalla, joka vastaa neuroverkon saamaa syötettä vektorina esitettynä. Sarakkeiden päästä luettavat sähkövirrat $I_1...I_(3...)$ mallintavat suoraan neuronin painojen matriisilaskennan tulosta, joka voidaan viedä neuronin aktivaatiofunktioon. [19], [20]

Tällä tavoin voidaan rakentaa analoginen prosessori palvelemaan suoraan neuroverkkojen tarpeita. Neuroverkkojen päättelyvaiheen laskentatarpeista jopa 60-90 % on vastaavaa matriisilaskentaa. Laskutoimitus saadaan suoritettua ilman että laskemiseen tarvittavia parametreja tarvitsee odottaa erillisestä muistista. Kaikkien neu-

ronien laskutoimitukset tapahtuvat samanaikaisesti toisin kuin digitaalisesti. Implementaatio on sekä laskenta- että energiatehokas, mutta käytännön toteutukseen liittyy myös useita haasteita. [21], [22]

Analogisissa komponenteissa esiintyy väistämättä kohinaa sekä valmistusprosessin aiheuttamaa varianssia, minkä seurauksena laskennan tulokset eivät ole yhtä tarkkoja ja toistettavia kuin digitaalisissa järjestelmissä. Lisäksi neuroverkkojen käyttö edellyttää usein analogisen ja digitaalisen laskennan yhdistämistä, koska syötteiden ja tulosteiden käsittely sekä ohjauslogiikka toteutetaan tavallisesti digitaalisesti. Tällöin signaalin muunnokset analogisten ja digitaalisten järjestelmien välillä muodostavat oman kustannuksensa sekä energiankulutuksen että viiveen näkökulmasta.[21], [22]

Muistissa laskentaa pidetään lupaavana lähestymistapana erityisesti päättelyvaiheessa, jossa valmiiksi opetettua neuroverkkoa käytetään ennusteiden tuottamiseen. Päättelyssä vaadittu laskenta koostuu suurelta osin matriisi-vektorituloista, jotka soveltuvat hyvin analogisesti toteutettaviksi. Sen sijaan neuroverkon opettaminen on yleensä vaativampaa, koska painoja on päivitettävä toistuvasti. Tämän kehitteillä olevat ratkaisut keskittyvät siihen, että opetus suoritetaan digitaalisesti ja analogista muistissa laskemista hyödynnetään ennusteiden tuottamiseen. [19], [21]–[23]

5 Pohdinta

Analog In-Memory Computing eli muistissa laskeminen, ei kuitenkaan ole vielä valmista teknologiaa. Vaikka onnistuneita toteutuksia on tehty useamman tahon kuten IBM:n[1] tai Mythic AI:n[2] toimesta, ei kuluttajamarkkinoilla ole vielä yhtään valmista tuotetta. Kuten luvussa 2 esitettiin, analogisesti ei pystytä laskemaan täysin diskreeteillä arvoilla. Analogisiin laitteisiin liittyy aina pientä varianssia.

Neuroverkkojen tapauksessa pieni varianssi ennusteissa pystytään hyväksymään, sillä kuvantunnistustehtävissä mallin tuottamat ulostulot muunnetaan todennäköisyyksiksi Softmax-funktion avulla. Funktio normalisoi verkon tuottamat arvot siten, että suurin arvo vastaa ennustettua luokkaa ja muut arvot esittävät suhteellisia todennäköisyyksiä. Näin ollen useammankin prosenttiyksikön varianssi kuvan sisältämän kohteen luokitteluun liittyvässä todennäköisyydessä ei yleensä muuta mallin ennustamaa luokkaa, vaan johtaa käytännössä samaan luokittelupäätökseen. Kuten Kuvan 3.3 esimerkkitapauksessa, neuroverkon ulostulokerroksen tuottamat todennäköisyysarvot eri luokille määrittävät mallin ennustaman luokan. Vaikka suurimman todennäköisyyden arvo vaihtelisi jonkin verran, pysyy se usein edelleen selvästi muita luokkia suurempana.

[1]IBM Hermes, https://research.ibm.com/publications/hermes-core-a-14nm-cmos-and-pcm-based-in-memory-compute-core-using-an-array-of-300pslsb-linearized-cco-based-adcs-and-local-digital-processing–1

[2]Mythic Ai M1076 Analog Matrix Processor, https://mythic.ai/products/m1076-analog-matrix-processor/

Osa laskennassa joudutaan kuitenkin tälläkin implementaatiolla suorittamaan digitaalisessa muodossa. Kaikkia funktioita ei myöskään ole järkevää toteuttaa analogisessa muodossa. Analogisen laskennan aiheuttamaa virhettä pystytään minimoimaan viemällä laskenta välillä takaisin digitaaliseen muotoon [22]. Näin analogisen melun aiheuttamat pyöristysvirheet eivät kumuloidu liikaa ajan myötä. Siirrot digitaalisen ja analogisen välillä ovat kuitenkin laskennan kannalta turhaa työtä mikä hidastaa prosessia kokonaisuudessaan.

Yksi kehittyneimmistä Analog In-Mermory Computing-teknologian implementoinneista on IBM:n kehittämä Hermes -mikrosiru. Mikrosirussa 64 ydintä, jokaisessa 256x256 kokoinen, kuvan 4.2 mukainen, analoginen matriisilaskentakomponentti. Sirulla pystyy siis laskemaan muistissa samaan aikaan noin 4,2 miljoonaa neuroverkon painoarvoa. Näiden lisäksi Hermeksessä on kahdeksan digitaalista prosessoriyksikköä, joiden tarkoitus on prosessoida sirun analogista signaalia. Siru ottaa vastaan ja lähettää eteenpäin digitaalisia signaaleja, eli sirua käytetään digitaalisen tietokoneen rinnalla nopeuttamaan neuroverkoissa tapahtuvaa laskentaa. [22]

Sirulla onnistuttiin ajamaan ResNet-9 -arkkitehtuurista neuroverkkoa. ResNet-9 on verrattain kevyt neuroverkko. Verkossa on kahdeksan konvoluutiokerrosta ja yksi täysin kytketty kerros, jotka sisältävät yhteensä 1 869 122 koulutettavaa parametria. Aineistona käytettiin CIFAR-10 -kuvakirjastoa, joka sisältää hyvin pieniä 32x32 pikselisiä värikuvia. Tällä konfiguraatiolla saavutettiin noin 93 % tarkkuus kuvantunnistuksessa. [22]

Ottaen huomioon, että Hermeksen tutkimusraportti julkaistiin joulukuussa 2022, voidaan teknologian katsoa olevan vielä varhaisessa tutkimusvaiheessa. Tämän perusteella on todennäköistä, ettei vastaavaa teknologiaa nähdä kuluttajamarkkinoilla vielä lähitulevaisuudessa. ResNet-9:n käyttäessä melkein puolet sirun kapasiteetissa olevista koulutettavista parametreista täytyy joko sirujen tai ohjelmistojen vielä huomattavasti kehittyä ennen kuin teknologialla pystytään tuottamaan kuluttajil-

le hyödyllisiä sovelluksia. CIFAR-10:n pienet 32x32 kuvat ovat hyvä aineisto tieteellisen tutkimuksen pohjaksi mutta ei palvele kuluttajalle tarkoitettua toteutusta. Halvimpienkin älypuhelimien kamerat tallentavat kuvia yli kymmenen megapikselin resoluutiolla eli sisältävät yli kymmenen tuhatta pikseliä. Kehityssuunta on kuitenkin mielenkiintoinen historian toistaessa itseään kääntymällä takaisin digitaalisesta analogiseksi.

6 Yhteenveto

Tietokoneiden historia on ollut pidempään analoginen kuin digitaalinen. Kuitenkin viimeisen viidenkymmenen vuoden aikana digitaaliset tietokoneet ovat dominoineet kuluttajamarkkinoita. Digitaalisen tietokoneiden valtava kehitys on mahdollistanut myös erittäin monimutkaisten tietokoneohjelmien kehityksen. Tietokoneille on rakennettu tekoälyä, jolla pystyy suorittamaan tiettyjä tehtäviä jopa ihmistä tarkemmin. Kuvantunnistus on tekoälyn haara, jonka tarkoituksena on oppia kuvien merkitystä ja tunnistamaan esineitä kuvista. Kuvantunnistusta suoritetaan konvoluutioneuroverkoilla, joissa pienillä suodattimilla kuvasta etsitään merkityksellisiä piirteitä, joiden perusteella ohjelma ennustaa kuvassa esiintyviä esineitä. Tässä tutkielmassa tarkasteltiin kahta tutkimuskysymystä:

TK1 Miten analogista laskentaa voidaan hyödyntää kuvan tunnistuksessa?

TK2 Mitkä ovat analogisen laskennan hyödyt ja haitat kuvan tunnistuksessa?

Kuvantunnistus on laskennallisesti ja muistillisesti erittäin kuormittavaa ja digitaalisen tietokoneen kehityksen on vaikea pysyä ohjelmistollisen kehityksen rinnalla. Tästä syystä on kehitetty analogista laskentaa digitaalisen laskennan tueksi.

Kirjallisuuden perusteella analogista laskentaa voidaan hyödyntää neuroverkkojen matriisilaskennan tehostamisessa toteuttamalla laskentaa suoraan muistin sisällä. Tällöin voidaan vähentää tiedon siirtämistä prosessorin ja muistin välillä. Analogista laskentaa hyödyntämällä pystytään vähentämään digitaalisen tietokoneen energiankulutusta sekä tehostamaan laskentaa ja muistinkäyttöä. Rakentamalla laskeva

järjestelmä muistin sisään voidaan minimoida tarve liikutella tietoa muistin ja prosessorin välillä mikä on sekä hidasta että energiaa vievää.

Haittoina puolestaan ovat toteutuksen monimutkaisuus, laskennan epätarkkuus sekä teknologian keskeneräisyys. Toimivia teknologian implementaatiota on julkaistu, mutta vaativat vielä runsaasti kehitystä ennen kuin teknologiaa voidaan nähdä kuluttajamarkkinoilla.

Lähdeluettelo

[1] B. J. Copeland. ”The Modern History of Computing”, viitattu 11. maaliskuuta 2026. url: https://plato.stanford.edu/archives/win2020/entries/computing-history.

[2] GeeksforGeeks. ”Computer Organization | Von Neumann Architecture”, viitattu 23. maaliskuuta 2026. url: https://www.geeksforgeeks.org/computer-organization-von-neumann-architecture/.

[3] H. Tuominen, P. Neittaanmäki ja E. Niinimäki, *Tekoälyn perusteita ja sovelluksia*. 2019, ISBN: 978-951-39-7796-2. viitattu 23. maaliskuuta 2026. url: https://jyx.jyu.fi/handle/123456789/64975.

[4] MachineLearningMastery. ”A Gentle Introduction to Object Recognition With Deep Learning”, viitattu 23. maaliskuuta 2026. url: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

[5] S. Haykin, *Neural Networks and Learning Machines*, 3. painos. Pearson, 2009.

[6] Towards Data Science. ”Convolutional Neural Networks Explained”, viitattu 23. maaliskuuta 2026. url: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[7] K. O'Shea ja R. Nash. "An Introduction to Convolutional Neural Networks". arXiv: 1511.08458 [cs.NE], viitattu 5. maaliskuuta 2026. url: https://arxiv.org/abs/1511.08458.

[8] Brilliant. "Backpropagation", viitattu 23. maaliskuuta 2026. url: https://brilliant.org/wiki/backpropagation/.

[9] K. Simonyan ja A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015. arXiv: 1409.1556 [cs.CV]. url: https://arxiv.org/abs/1409.1556.

[10] ImageNet. "ImageNet", viitattu 23. maaliskuuta 2026. url: https://www.image-net.org/.

[11] Kaggle. "VGGNet-16 Architecture: A Complete Guide", viitattu 23. maaliskuuta 2026. url: https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide/notebook.

[12] Stanford University. "CS231n Convolutional Neural Networks for Visual Recognition", viitattu 23. maaliskuuta 2026. url: https://cs231n.github.io/.

[13] Medium. "VGG-Net Architecture Explained", viitattu 23. maaliskuuta 2026. url: https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f.

[14] K. Shirahata, Y. Tomita ja A. Ike, "Memory reduction method for deep neural network training", teoksessa *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2016, s. 1–6. DOI: 10.1109/mlsp.2016.7738869.

[15] Intel Corporation. "Moore's Law", viitattu 5. maaliskuuta 2026. url: https://www.intel.com/content/www/us/en/newsroom/resources/moores-law.html.

[16] P. Bowen, G. Regev, N. Regev, B. Pedroni, E. Hanson ja Y. Chen. ”Analog, In-memory Compute Architectures for Artificial Intelligence”, viitattu 5. maaliskuuta 2026. url: https://arxiv.org/abs/2302.06417.

[17] M. Horowitz, ”1.1 Computing’s energy problem (and what we can do about it)”, teoksessa *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, IEEE, 2014, s. 10–14. DOI: 10.1109/isscc.2014.6757323.

[18] X. Zou, S. Xu, X. Chen, L. Yan ja Y. Han, ”Breaking the von Neumann bottleneck: architecture-level processing-in-memory technology”, *Science China Information Sciences*, vol. 64, nro 6, 2021, ISSN: 1869-1919. DOI: 10.1007/s11432-020-3227-1.

[19] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby ja G. W. Burr, ”Recent progress in analog memory-based accelerators for deep learning”, *Journal of Physics D: Applied Physics*, vol. 51, nro 28, s. 283 001, 2018, ISSN: 1361-6463. DOI: 10.1088/1361-6463/aac8a5.

[20] M. Le Gallo-Bourdeau. ”The hardware behind analog AI”, IBM, viitattu 23. maaliskuuta 2026. url: https://research.ibm.com/blog/the-hardware-behind-analog-ai.

[21] M. Le Gallo, C. Lammie, J. Büchel, F. Carta, O. Fagbohungbe, C. Mackin, H. Tsai, V. Narayanan, A. Sebastian, K. El Maghraoui ja M. J. Rasch, ”Using the IBM analog in-memory hardware acceleration kit for neural network training and inference”, *APL Machine Learning*, vol. 1, nro 4, 2023, ISSN: 2770-9019. DOI: 10.1063/5.0168089.

[22] M. Le Gallo et al., ”A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference”, *Nature Electronics*,

vol. 6, nro 9, s. 680–693, 2023, ISSN: 2520-1131. DOI: 10.1038/s41928-023-01010-1.

[23] W. Haensch, A. Raghunathan, K. Roy, B. Chakrabarti, C. M. Phatak, C. Wang ja S. Guha, "Compute in-Memory with Non-Volatile Elements for Neural Networks: A Review from a Co-Design Perspective", *Advanced Materials*, vol. 35, nro 37, 2023, ISSN: 1521-4095. DOI: 10.1002/adma.202204944.