KANDIDAATINTYÖ

Neuroverkkojen FPGA-toteutus

Ville Paananen

Ohjaaja: Jukka Lahti

ELEKTRONIIKAN JA TIETOLIIKENNETEKNIIKAN TUTKINTO-OHJELMA

2018

Paananen V.[1] (2018) Neuroverkkojen FPGA-toteutus. Oulun yliopisto, Elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Kandidaatintyö, 21 s

TIIVISTELMÄ

Tässä työssä esitellään kaksi erilaista neuroverkkoa, monikerrosperceptron ja konvolutionaalinen neuroverkko, ja tutkitaan niiden toteutettavuutta käyttämällä FPGA-piirirakennetta. Työssä kuvataan neuroverkkojen taustaa ja esitellään niiden toimintaa ja suunnittelua ohjaavia parametreja. Lisäksi tutkitaan FPGA-piirien eduja ja haasteita. Työ tehtiin kirjallisuuskatsauksena käyttämällä lähteinä ajankohtaisten neuroverkkotutkimusten tuloksia.

Avainsanat: FPGA, CNN, konvoluutio, neuroverkko

Paananen V.[1] (2018) FPGA implementation of Neural Networks. University of Oulu, Degree Program in Electrical Engineering, Bachelor's Thesis, 21 p.

ABSTRACT

This work presents two different neural networks, multi-layer-perceptron and convolutional neural network and their FPGA-implementation is researched. The work describes the background for neural networks and presents their operation and the parameters that guide their design. The benefits and challenges of FPGA-circuits are also researched. The work was done as a literature review using the the contemporary neural network research.

Key words: FPGA, CNN, convolution, neural network

SISÄLLYS

LYHENTEIDEN JA MERKKIEN SELITYKSET

ASIC	Application-Specific Integrated Circuit
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FPGA	Field Programmable Gate Array
GFLOPS	Giga Floating-Point Operations Per Second
GMAC/s	Giga Multiply-Accumulate Operations Per Second
GPU	Graphics Processing Unit
HDL	Hardware Description Language
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
XOR	Exclusive-OR

1. JOHDANTO

Tutkimuksessa selvitetään miten neuroverkkoja voidaan toteuttaa käyttämällä FPGA-piirejä. FPGA (Field Programmable Gate Array) on digitaalilogiikkaa hyödyntävä piirityyppi, joka perustuu loogisten porttien massiiviseen rinnakkaisuuteen. Neuroverkot ovat yksi tekoälyn muoto, joka pohjautuu ihmisaivojen neuronien biologiseen toimintaan. On kuitenkin huomattava, että neuroverkkojen malli on erittäin yksinkertaistettu neuronien toiminnasta, joten keinotekoisen älykkyyden sijaan on järkevämpää puhua ongelmaratkaisukyvystä. Kyseessä ei siis ole tapa tehdä "digitaaliset ihmisaivot", vaan mallintaa yksinkertaistetusti sen toimintaa ja tuottaa älykäs laite, joka kykenee parempaan ongelmanratkaisuun.

Työssä esitellään kaksi erilaista neuroverkkoa: monikerrosperceptron [1] ja konvolutionaalinen neuroverkko [2], joiden ominaisuudet esitellään ja selvitetään miten ne soveltuvat FPGA-piirillä toteutettavaksi. Tutkimus toteutetaan kartoittamalla kirjallisuudessa esitettyjä tutkimustuloksia neuroverkkojen FPGA-toteutuksen eduista ja haasteista ja jäsentämällä niistä järkevä kokonaisuus, joka toimii hyvänä esittelynä aiheeseen entuudestaan perehtymättömälle.

Optimaalisen tekoälyn ja ongelmanratkaisykyvyn saavuttamiseksi tulee tehdä paljon tutkimusta käytettävästä laitteistosta, jolle neuroverkot toteutetaan. Oleellinen tekijä neuroverkkojen optimaalisuudessa on rautatason tehokas hyödyntäminen. Neuroverkkotutkimusta on tehty hyvin paljon ja erilaisia verkkoja on kehitetty lukuisia. Sen vuoksi onkin tärkeä tutkia mitä haasteita neuroverkkojen toteutettavuus voi asettaa.

Lukijalle valaistaan miksi FPGA:n ominaisuudet ovat erittäin hyvin soveltuvia neuroverkkojen toteutukseen verrattuna esimerkiksi CPU- tai GPU-toteutukseen, mutta kuitenkin samalla tuoden esille FPGA-piirin rajoitukset. Perinteiset, yhtä tapausta varten suunnitellut ASIC-piirit (Application Specific Integrated Circuit), sisältävät rajoitetun määrän kytkentöjä eri piirielementtien välillä ja ne ovat huonosti uudelleenkonfiguroitavissa, mutta voivat olla tehokkaampia kuin FPGA-toteutukset. Neuroverkkojen painoarvovektorit vaativat nopeaa kirjoitus- ja lukunopeutta, joten rinnakkainen toiminta mahdollistaa nopeamman oppimisen. FPGA-piirien rajoituksena toimii haasteet suurten liukulukujen esittämisessä verkon tarkkuuden ja oppimisnopeuden osalta. Tämä vaatii siten kompromisseja FPGA-piirin käytetyn pinta-alan ja numeerisen tarkkuuden välillä.

2. TEORIA

2.1. Neuroverkot

Neuroverkot ovat yksi tekoälyn muoto, jonka kehitys alkoi 1960-luvulla, mutta matemaattista pohjaa luotiin jo 1940-luvulla. Vuosituhannen loppupuoliskolla tehtiin paljon tutkimusta neuroverkoista, mutta elektroniikan tehokkuuden ja valmistustiheyden rajoitukset olivat esteenä kehityksen jatkumiselle. Teknologian kehittyessä tehokkaampien prosessoreiden myötä 2010-luvulla neuroverkot nousivat takaisin suosioon ja tekoälystä tuli varteenotettava työkalu vaativissa ongelmanratkaisuissa. [1]

Neuroverkot saavat nimensä ihmisen aivojen neuronirakenteesta, ja siten neuroverkon pohjalla on yksinkertaistettu malli aivojen toiminnasta neuronitasolla. Neuroverkkojen tarkoituksena on mallintaa neuronien oppimista ja luoda verkko, joka pystyy muistamaan sille annetut syötteet ja luokittelemaan uutta dataa sen perusteella.

Vuosikymmenien aktiivisen tutkimuksen myötä erilaisia neuroverkkoja on kehitetty paljon ja niiden uniikkien ominaisuuksien myötä kullakin on tietty ongelma-avaruus, jossa ne toimivat parhaiten. Näitä neuroverkkoja ovat esimerkiksi tässä työssä esiteltävät perinteinen monikerros-perceptron (*multilayer perceptron,* MLP) ja uudempi kuvantunnistamisessa käytetty konvolutionaalinen neuroverkko (*convolutional neural network,* CNN). Muita neuroverkkoja on esimerkiksi puheentunnistamisessa ja kielen kääntämisessä tehokkaaksi havaittu LSTM, eli *long-short term memory* -verkko tai takaisinkytkentää hyödyntävä RNN, eli *recurrent neural network*.

2.1.1. Perceptron

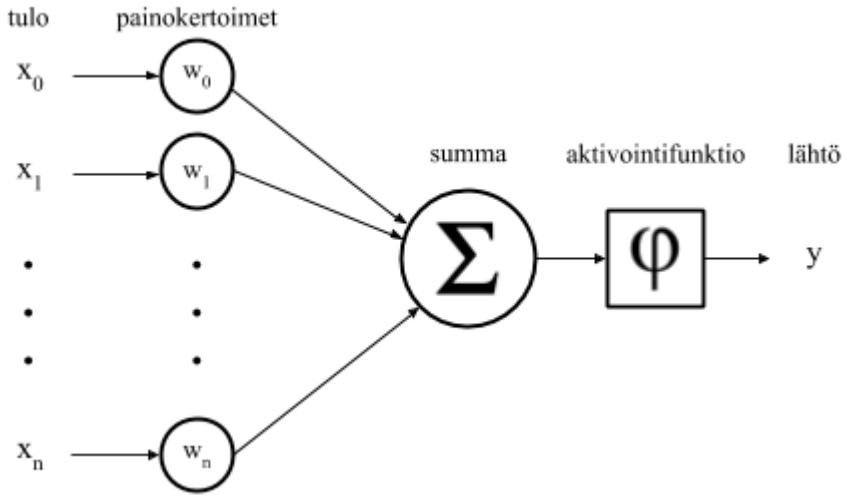
Ihmisaivojen hermoimpulsseja välittävää neuronia mallintamiseen kehitettiin perceptron, joka kuvaa neuronin syttymistä tiettyjen tulosignaalien ja painoarvojen funktiona. Perceptronin mallin esitteli Frank Rosenblatt vuonna 1962 kirjassaan "Principles of Neurodynamics". Perceptroneista rakennettua neuroverkkoa usein kuvataan virheellisesti ihmisaivojen malliksi ja ajatellaan, että riittävän suuressa mittakaavassa toteutettuna se voisi tuottaa tietoisuutta samaan tapaan kuin ihmisaivoissa. Tätä kuvaa hyvin seuraava lainaus:

> *"He describes these perceptrons as simplified networks in which certain properties of real nervous systems are exaggerated whilst others are ignored. He stated that they are not intended to serve as detailed copies of any real nervous system; in other words, he realised at this early stage that he was dealing with a basic model. This fact is often lost in the popular press as the idea of computer "brains", based on these techniques, grabs the imagination. We are not attempting to build computer brain, nor are we trying to mimic parts of real brains - rather we are aiming to discover the properties of models that take their behaviour from extremely simplified versions of natural neural systems, usually on a massively reduced scale as well. Whereas the brain has at least 10^{10} neurons, each connected to 10^{4} others, we*

are concerned here with maybe a few hundred neurons at most, connected to a few thousand input lines". [1 s. 44-45]

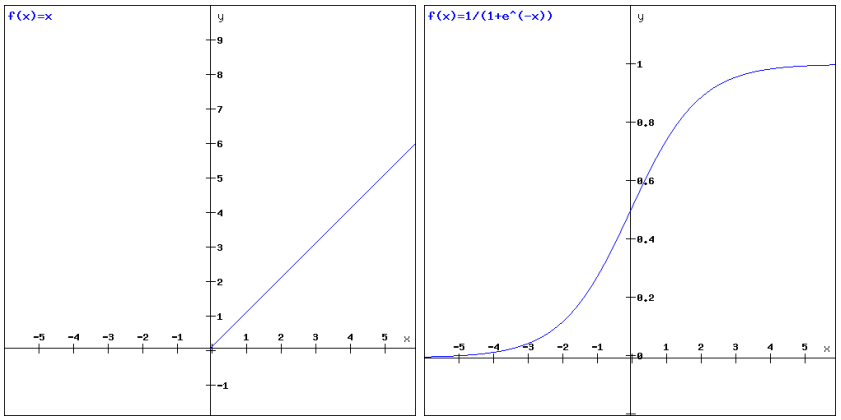
On siis huomattava, että perceptronin malli jättää aina jotain pois, joten on siis on hyvin epätodennäköistä, että sitä hyödyntämällä voitaisiin toteuttaa ihmiseen verrattavaa älykkyyttä. Siitä huolimatta on todistetusti huomattu, että perceptroneja hyödyntämällä on saatu hyvin tehokkaita neuroverkkoja, jotka pystyvät ratkaisemaan tietokoneille aiemmin mahdottomia ongelmia.

Perceptronin rakenne on alla olevan kuvan 1 mukainen, jossa on tulovektori $\hat{x}$, painovektori $\hat{w}$, summausfunktio $\sum$ ja aktivointifunktio φ. Toiminta on seuraavanlainen: jokainen tulovektorin elementti $\hat{x}$ kerrotaan vastaavan painoarvovektorin elementin $\hat{w}$ kanssa, ja summataan yhteen. Tämä painotettu summa syötetään aktivointifunktioon, joka antaa lähtöön arvon 0 tai 1, kuvastaen neuronin syttymistä.



Kuva 1. N-tuloisen perceptronin malli.

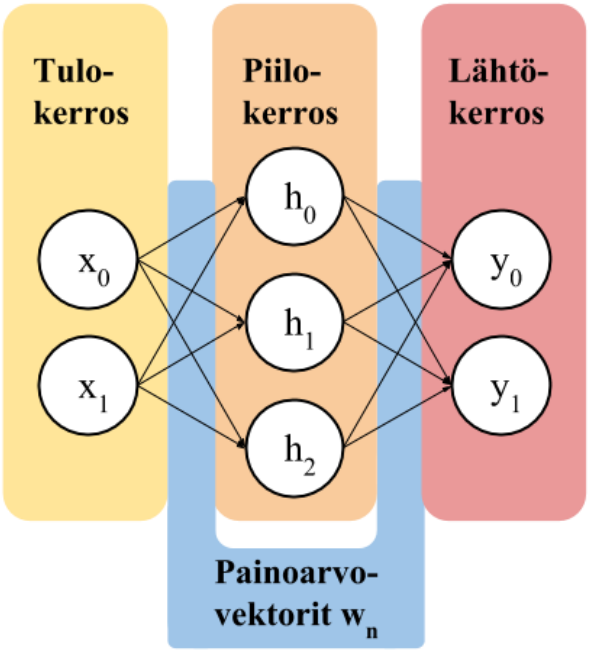
Aktivointifunktion φ tarkoituksena on normalisoida arvot tietylle alueelle ja epälinearisoida ne siten, että lähdössä voidaan havaita pienet tulojen vaihtelut. Jos aktivointifunktio lähtö saisi pelkästään arvon 0 tai 1, olisi mahdotonta opettaa neuroverkkoa. Lisäksi ilman epälineaarisuutta koko verkon voisi yksinkertaistaa yhdeksi funktioksi [1, s. 66]. Aktivointifunktio voi olla alla olevan kuvan 2 mukainen yleisesti käytetty yksinkertainen tasasuuntausfunktio tai sigmoid. Eri aktivointifunktioiden käyttö vaikuttaa erityisesti neuroverkon oppimisnopeuteen.



Kuva 2. Neuroverkoissa usein käytetyt aktivointifunktiot: tasasuuntaaja (Rectified Linear Unit, ReLu), vasen ja sigmoid, oikea.

2.1.2. Monikerros-perceptron

Kytkemällä monituloiset perceptronit rinnakkain ja kerroksittain saadaan yleisin neuroverkkojen malli, eli monikerros-perceptron (*Multilayer Perceptron*, MLP), kuten kuvassa 3, alla. Kuvassa jokainen solmu sisältää aiemmin esitellyn perceptronin, mukaan lukien aktivointifunktion. Tälläinen kytkentä kasvattaa neuroverkon ongelmanratkaisukykyä huomattavasti. On huomattu, että jo kolmella kerroksella saadaan tehokas neuroverkko [1]. Ensimmäistä kerrosta nimetään tulokerrokseksi (*input layer*), viimeistä lähtökerrokseksi (*output layer*), ja niiden välissä olevat kerrokset ovat piilotettuja kerroksia (*hidden layers*), joka viittaa siihen että sen solmujen tilat eivät näy suoraan tulossa tai lähdössä.



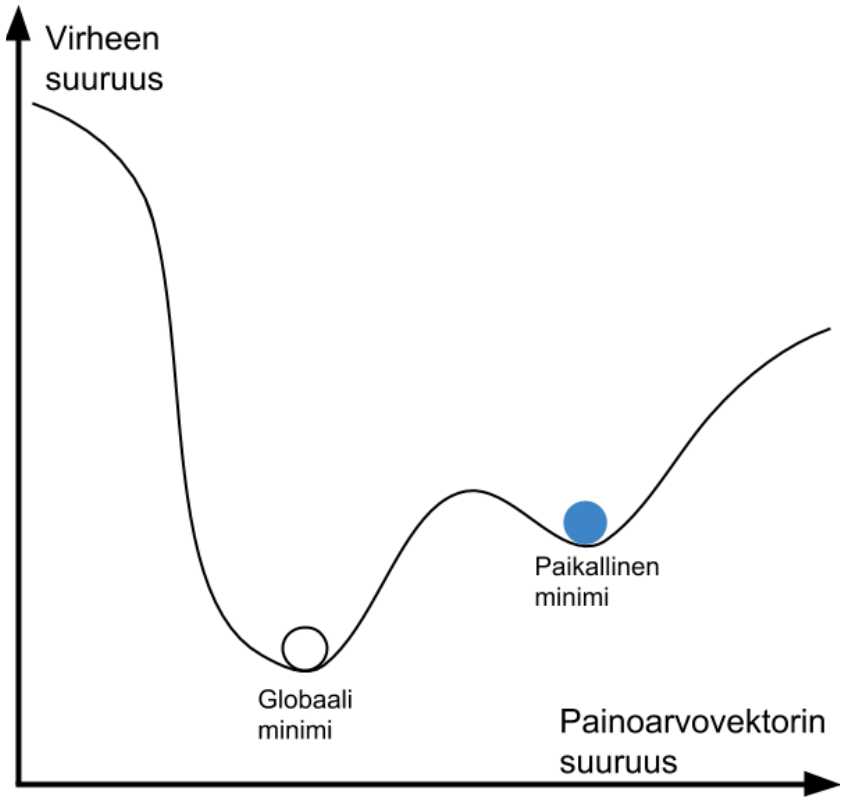
Kuva 3. MLP-verkon eri kerrokset ja niiden väliset painoarvovektorien kytkennät.

Kerroksen jokaisen solmun lähtö on siis kytketty jokaiseen seuraavan kerroksen solmun tuloon. Nämä kerrosten välissä olevat kytkennät ovat painoarvokertoimia

(*weights*), joiden optimaalisten arvojen määrittäminen on oleellista kun neuroverkkoja opetetaan. Koska jokainen lähtö on kytketty jokaiseen seuraavan kerroksen tuloon, huomataan, että painoarvojen määrä kasvaa eksponentiaalisesti kun kerroksessa lisätään rinnakkaisia perceptroneita. Tämä eksponentiaalisuus näkyy suoraan realisoituvuudessa: sarjassa toimivassa piirissä painoarvovektorien arvojen päivittäminen kestää kauemmin kuin samanaikaisesti toimivassa rinnakkaisessa piirissä.

2.1.3. Opettaminen ja optimointi

Kuten kaikki neuroverkot, myös MLP tulee opettaa, jotta pystytään ratkaisemaan ongelmia suurella tarkkuudella ja luotettavuudella. Tämä tapahtuu löytämällä oikeat arvot painovektoreille. Verkon optimointi voi olla haastavaa ja oppimista seurataan tehtyjen virheiden ja virheiden muutosgradienttien avulla, mikä auttaa löytämään neuroverkkosysteemin globaalit minimit. Haasteena neuroverkon opetuksessa on tilanteet, jossa verkko löytää yhden paikallisen minimin, eikä lähde kehittymään eteenpäin, kuten alla olevassa kuvassa 4.

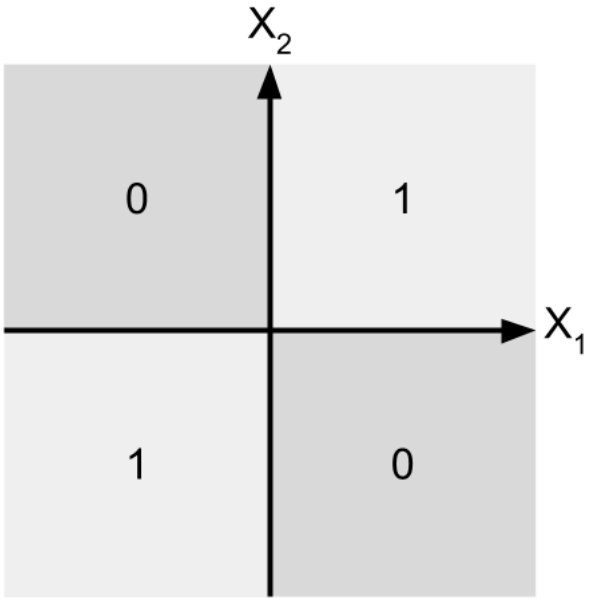


Kuva 4. Painoarvovektorien määrittämisessä ja virheiden minimoimisessa saattaa tulla tilanne, jossa verkko löytää paikallisen minimin globaalin minimin sijaan. Voidaan kuvitella, että pallo asettuu paikallisen minimin kohdalla olevaan kuoppaan, jolloin vaaditaan ylimääräistä energiaa, jotta saavutettaisiin globaali minimi.

Neuroverkon opettaminen tapahtuu syöttämällä sille ensin suuri määrä opetusdataa, jonka jälkeen sen tarkkuutta testatataan testidatalla, joka poikkeaa opetusdatasta. Usein käytetään niin sanottua valvottua oppimista, jossa syötettävän datan oikeat luokitukset tunnetaan. Siten voidaan arvioida päätyykö neuroverkko oikeaan luokitukseen vai ei. Erilaisia malleja opettamiselle on useita, mutta usein käytetään positiivista vahvistusta, jossa oikeissa luokituksissa vahvistetaan niitä painoarvokertoimia, jotka johtivat siihen luokitukseen. Vastaavasti virheellisissä luokituksissa heikennetään relevantteja painoarvokertoimia.

2.1.4. Ongelmanratkaisu

Perinteinen ensimmäinen testi tekoälyn testaamisessa on XOR-ongelma sen yksinkertaisuuden, mutta haastavuuden vuoksi. XOR-ongelmassa pyritään luokittelemaan lähtö tietyillä kriteereillä. Kuvassa 5, alla, on esitetty XOR-funktio, joka saa arvon 0 tai 1 riippuen tulojen X_1 ja X_2 arvoista. Nähdään, että funktioavaruutta ei voida jakaa yhdellä viivalla, jotta saataisiin lineearisesti erotetut luokat. Funktion lähdön luokitteluun vaaditaan kaksi viivaa, johon monikerroksinen neuroverkko pystyy.



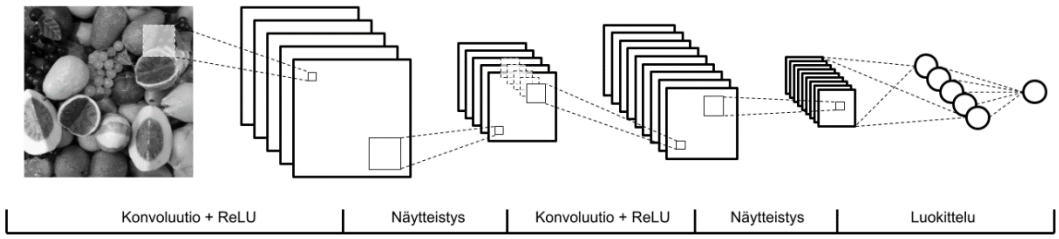
Kuva 5. XOR-funktion lähtö, tulojen X_1 ja X_2 perusteella.

Neuroverkkojen ongelmanratkaisussa on oleellisessa osassa ongelmien luokittelu. Esimerkiksi kuvantunnistamisessa hyödynnetään erilaisten visuaalisten elementtien luokittelua, kuten viivat, muodot, kontrastit ja värit. Kun neuroverkko havaitsee tietyt luokat, pystyy se opetuksen perusteella tarkkaan luokitteluun. Siten neuroverkon ei tarvitse opetella jokaista korkean tason elementtiä erikseen, vaan yksinkertaisempia lohkoja, joista ne muodostuvat.

2.1.5. Konvolutionaalinen neuroverkko

Konvolutionaalinen neuroverkko (*Convolutional Neural Network*, CNN) on yksi tekoälyn syväoppimisen neuroverkkomuoto, jossa hyödynnetään perinteisen monikerros-perceptronin luokitteluominaisuuksia ja konvoluutio-operaation mahdollistamaa ominaisuuksien erottelua ja tunnistamista. Siinä missä MLP on saanut inspiraationsa aivoneuronien toiminnasta, on CNN luotu mallintamaan eläinten näköhermojen toimintaa. CNN on osoittautunut erittäin tehokkaaksi kuvantunnistuksessa, joka on ollut perinteisesti haastava ongelma tietokoneille. Kuvantunnistuksessa on paljon sovelluskohteita, mikä on ajanut kehitystä eteenpäin myös CNN:n kohdalla.

CNN:ssä käytetään useita erilaisia kerroksia muokkaamaan ja erottelemaan tulovektorista erilaisia ominaisuuksia. Näitä kerroksia ovat konvoluutio-, aktivointi-, näytteistys- ja luokittelukerrokset (*convolution*, *activation*, *subsampling* ja *categorisation/fully connected* vastaavasti), kuten kuvassa 6, alla.



Kuva 6. Konvolutionaalisen neuroverkon eri osat.

Konvoluutiokerros (*convolutional layer*) tekee suurimman osan laskennasta hyödyntäen konvoluutio-operaatiota ja kernelvektoria, jolle on asetettu tietyt painoarvot. Konvoluutio on operaatio, jossa liu'utetaan konvoluutiokerneliä tulovektorin yli, tuottaen niin sanotun ominaisuusvektorin (*feature map*). Matemaattisesti konvoluutiossa tulovektorin elementit kerrotaan kernelin vastaavilla arvoilla ja summataan yhteen, jolloin saadaan lähtövektorin arvo. Tämän jälkeen kerneliä siirretään eteenpäin ja suoritetaan uusi konvoluutio-operaatio lähtövektorin seuraavan arvon laskemiseksi. Tätä jatketaan kunnes saadaan laskettua lähtövektorin kaikki arvot eli konvoluutiokerneli on liu'utettu koko tulovektorin yli. Konvoluutiokerroksessa jokainen kernel tuottaa yhden ominaisuusvektorin, joten useaa kerneliä käyttämällä saadaan erotettua erilaisia ominaisuuksia tulovektorista.

Aktivointikerros (*activation layer*) epälinearisoi vektorin, jotta saadaan paremmin eroteltua ominaisuudet toisistaan, mikä edesauttaa oppimista. Konvoluutioneuroverkkojen aktivointikerroksissa käytetään usein yksinkertaista tasasuuntaus-funktiota (r*ectified linear unit,* ReLU), joka tuottaa arvon 0, jos lähtö on negatiivinen, mutta muuten tulo pysyy muuttumattomana.

Näytteistyskerros (*subsampling/max pooling layer*) pienentää kerroksen dimensioita jakamalla alue pienempiin ei-päällekäisiin alueisiin valitsemalla kunkin alueen maksimiarvo. Tämä helpottaa laskemista ilman suuria kompromissejä oppimisessa.

Luokittelukerros *(categorisation/fully-connected layer*) on samanlainen kuin monikerros-perceptron -verkko ja siten jokainen tulo on kytketty edellisen kerroksen lähtöön. Tämän myötä saadaan muunnettua 2-ulotteiset ominaisuudet 1-ulotteiseksi

vektoriksi, jossa kutakin luokiteltavaa kategoriaa vastaa yksi arvo. Suurimman arvon saanut luokitellaan aiemman opetuksen perusteella. [2]

Tyypillisesti CNN opetetaan valvotusti käyttämällä suurta määrää opetusdataa ja, kuten MLP-verkossa, tutkimalla virheiden suuruuksia kun tiedetään mitä verkon olisi pitänyt tuottaa. Opetuksen aikana CNN muokkaa konvoluutiokerneleiden arvoja virheiden minimoimiseksi. CNN:n toimintaa voidaan lopuksi arvioida syöttämällä opetusdatasta eroavaa testidataa, jolloin voidaan nähdä miten verkko kykenee tulkitsemaan ja luokittelemaan uutta dataa.

2.2. FPGA-piirit

Field Programmable Gate Array, eli FPGA, on digitaalinen piirirakenne, jolla pystytään toteuttamaan uudelleenohjelmoitavia loogisia piirejä. FPGA on siten tehokas työkalu, jolla voidaan tuottaa piirien prototyyppejä suhteellisen vaivattomasti. FPGA:n toiminta perustuu hyvin suureen määrään loogisia portteja, jotka ovat aseteltu matriisiin. Nämä loogiset portit ovat kytketty muihin portteihin, mahdollistaen suuren rinnakkaisuuden. Tämän lisäksi FPGA:t usein sisältävät erilaisia loogisia lohkoja esimerkiksi muistia ja aritmeettisia operaatioita varten.

Koska FPGA:t sisältävät paljon lohkoja ja niiden välisiä kytköksiä, piirien suunnittelu on huomattavan tärkeää jotta saadaan pinta-alaltaan mahdollisimman tiivis lopputulos. Noin 80-90 % FPGA-piirien pinta-alasta menee reititykseen, joten kytkennät ovat optimoitava tarkasti [4]. Lukujen esityksessä käytetään loogisia kiikkuja, joten lukujen esittäminen vaativat paljon FPGA-piiriltä. Tämän myötä FPGA-suunnittelijan tulee tehdä kompromissi käytettävän pinta-alan ja lukutarkkuuden välillä. FPGA-piirit suunnitellaan käyttämällä erilaisia laitteistonkuvauskieliä (*hardware description language*, HDL), jotka toimivat rekisteritasolla, esimerkiksi VHDL ja Verilog, tai systeemitasolla, kuten SystemVerilog tai SystemC.

FPGA-piirien yhtenä vaihtoehtona voidaan nähdä tavanomaiset tietokonesuorittimet (CPU, *Central Processing Unit*). Ne ovat edullisia ja helposti saatavissa, mutta niiden haittapuolena on rajoitettu käskykanta ja sarjaoperaatio, mikä on laskennallisesti tehotonta neuroverkkojen tapauksessa. Rinnakkaisuutta vaativa laskenta sopii siis hyvin FPGA:lla toteutettavaksi.

Toinen vaihtoehto FPGA-piireille on tietokoneiden näytönohjaimet, eli GPU:t (*Graphics Processing Unit*). Kuten FPGA:t, ne sisältävät paljon rinnakkaisuutta, mutta haittana niillä on erittäin suuri tehonkulutus ja suuri pinta-ala, mikä johdosta se soveltuu huonosti sulautettuihin toteutuksiin. Tämän lisäksi GPU:n toimintaperiaate, jossa dataa käsitellään nipuissa, aiheuttaa paljon viivettä neuroverkkototeutuksessa, mikä ei ole suotuisaa reaaliaikaisessa kuvantunnistuksessa [2].

Yhtä sovellusta varten toteutetut ASIC-piirit (*Application Specific Integrated Circuit)* sopivat hyvin kun tiedetään tarkalleen millainen neuroverkko ollaan rakentamassa laitteistotasolla. FPGA:n avulla voidaan tutkia neuroverkon optimaalisuutta pienellä mittakaavalla ja lopulta toteuttaa se massatuotannossa ASIC-piirillä.

3. NEUROVERKKOJEN FPGA-TOTEUTUS

Johtuen konvoluutioverkkojen vaatimuksista kaistanleveyden, muistin ja nopeuden suhteen, se vaatii myös sille tarkasti määritellyn laitteistotason toteutuksen. Tämän myötä yleiskäyttöiset prosessorit eivät ole sopivia CNN:n toteutukseen. Konvoluutioneuroverkkoja on toteutettu muun muassa FPGA-, GPU- (*Graphics Processing Unit*) ja ASIC-pohjaisina. Viimeisimpien tutkimusten myötä on todettu, että FPGA:n tehokkuus tulee ohittamaan GPU:n, lisäten sen kannattavuutta [5]. GPU:lla on huono hyötysuhde ja se sopii huonosti sulautettuun toteutukseen. FPGA:n houkuttelevuutta lisää sen hyvä suorituskyky, suuri tehohyötysuhde, nopea kehitys ja uudelleenrekonfiguroinnin mahdollisuus [6].

CNN:n toteuttamiseen on olemassa useita erilaisia kehyksiä (*framework*), kuten AlexNet, SqueezeNet, GoogLeNet, Caffe ja VGG-16. Näiden eroina on pääosin kerneleiden koko ja kerrosten määrä. Lisäksi erilaiset toteutukset hyödyntävät rinnakkaisuutta eri tavalla, jolloin niiden tehokkuuden vertaaminen on haastavaa, vaikka käytössä olisi samat yksiköt, kuten GMAC/s. [2]

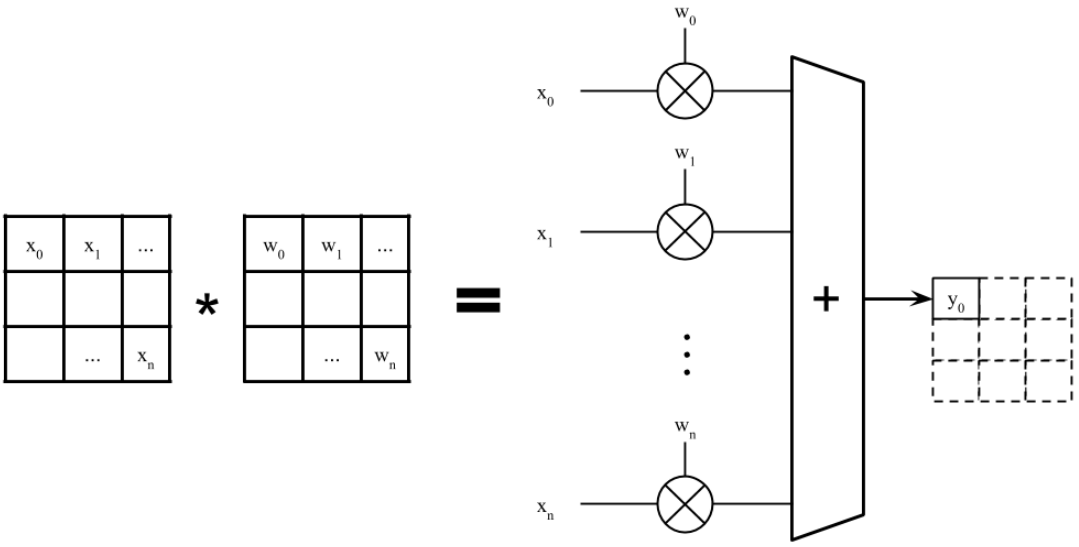
Konvoluutioverkon erilaisten kerrosten toteutusta on kuvattu seuraavissa luvuissa.

3.1. Konvoluutiokerros

Konvoluutiokerrokset ovat CNN:n tärkeimmät kerrokset. Niitä voi olla konvoluutioverkossa useita ja niiden kernelien koot voivat vaihdella. Noin 90 prosenttia konvolutionaalisen neuroverkon laskennasta tapahtuu konvoluutiokerroksissa, joten niiden huolellinen optimointi on tärkeää [3].

Konvoluutio voidaan toteuttaa FPGA-piireissä yksinkertaisesti MAC-operaatiolla (*Multiply-Accumulate operation*), jossa nimensä mukaisesti kerrotaan ja summataan lukuja yhteen. Nykyiset FPGA-piirit ovat hyvin tehokkaita toteuttamaan MAC-operaatioita, esimerkiksi yksinkertaisesti porttitasolla käyttämällä summaimia ja kertoimia, kuten kuvassa 7, alla.

Konvoluutioverkkoa suunnitellessa oleellisia parametrejä on konvoluutiokernelien koko, määrä ja liuku, eli *stride*. Konvoluutiossa on aiemmin käytetty 11*11 kerneleitä, mutta on huomattu, että pienemmät, 3*3 kernelit toteuttavat vastaavan tehokkuuden [2]. Konvoluutiokernelin koko määrittää miten pieniä ominaisuuksia voidaan tulkita tulovektorista. Toinen oleellinen parametri on kerneleiden määrä. N kappaletta kerneleitä tuottaa N kappaletta ominaisuusvektoreita (*feature maps*). Mitä enemmän kerneleitä on, sitä paremmin se pystyy tunnistamaan erilaisia ominaisuuksia tulovektorista. Stride-parametri kuvaa sitä, miten paljon konvoluutiokerneliä liu'utetaan kerrallaan. Toisin sanoen stride määrää kuinka päällekäin eri kernelit ovat. Tyypillisesti käytetään stride-parametrin arvoa 2, jolloin konvoluutiokerneli hyppää jokaisen laskun jälkeen 2 pikseliä eteenpäin. Siten verrattuna stride-parametrin arvoon 1, päällekkäisyys on pienempi ja siten lähdön koko on pienempi.

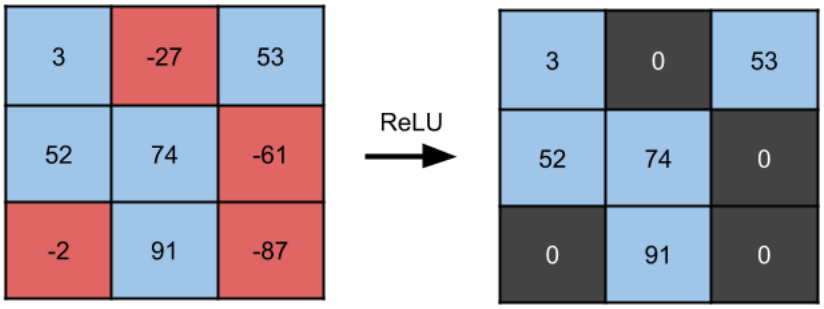


Kuva 7. Konvoluutio-operaation toteuttaminen kertoimella ja summaimella.

3.2. Aktivointikerros

Aktivointikerroksen toteutus on yksinkertainen: käydään tulovektorin kaikki arvot läpi yksi kerrallaan ja annetaan lähtöön aktivointifunktion mukainen arvo, kuten alla olevassa kuvassa 8. Siten aktivointikerroksessa dimensio ei muutu, mutta data muuttuu epälineaariseksi, mikä helpottaa erojen tunnistamista. Useimmiten käytetään ReLU-aktivointifunktiota [2], joka tuottaa lähtöön arvon 0 negatiivisilla tulon arvoilla, mutta muuten tuottaa lähtöön tulon arvon sellaisenaan.

Aktivointikerros voidaan toteuttaa erillisenä moduulina tai osana konvoluutiokerrosta, kuten usein tehdään. Erillisenä moduulina tulisi käydä koko tulovektori läpi, mikä on laskennallisesti kallista jos verkossa on monta parametria. Siksi on helpompi suorittaa aktivointifunktio suoraan konvoluutio-operaation jälkeen. Tällöin pelkästään luvun etumerkkiin perustuva operaatio voidaan toteuttaa yksinkertaisesti ja tehokkaasti porttitasolla.

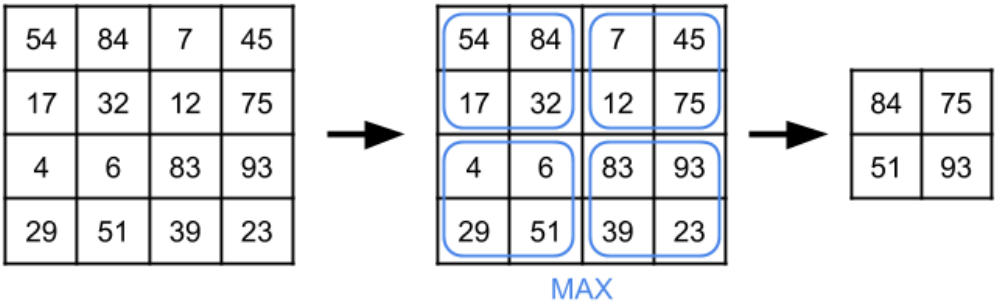


Kuva 8. Aktivointikerros, jossa on käytetty ReLU-aktivointifunktiota 3*3 vektorille.

3.3. Näytteistyskerros

Näytteistyskerros (*max pooling layer*) pienentää vektorin kokoa jakamalla se pienempiin ei-päällekäisiin alueisiin ja valitsemalla jokaisesta alueesta suurin arvo, kuten alla olevassa kuvassa 9. Siten saadaan yksinkertaistettua laskentaa ilman suuria menetyksiä tarkkuudessa. Vaihtoehtoisesti voidaan myös keskiarvoistaa jokaisen alueen arvot.

FPGA:lla tämä voidaan toteuttaa yksinkertaisesti moduulina, joka käy läpi kunkin alueen läpi ja asettaa lähtöön niiden suurimmat arvot. Alueiden koko on määritettävissä ulkoisilla parametreilla, useimmiten alueen koko on 2*2, mikä puolittaa tulovektorin koon lähtöön. Koska alueet eivät ole päällekäin, voidaan koko operaatio suorittaa rinnakkain, mikä hyödyntää hyvin FPGA:n rinnakkaisprosessointia. Tämä on laskennallisesti kevyt operaatio, joten sen optimoinnilla ei saada niin suurta etua kuin esimerkiksi konvoluutiokerroksessa.

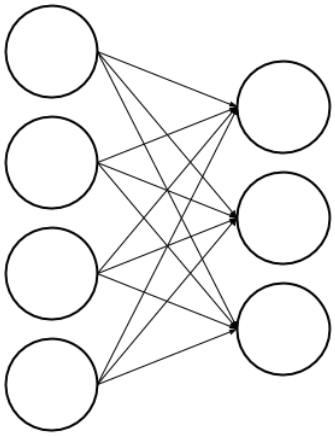


Kuva 9. Näytteistyskerroksen *max pooling*-operaatio.

3.4. Luokittelukerros

CNN-verkon viimeisenä kerroksena on usein luokittelukerros (*categorisation/fully connected layer*), kuva 10, jonka avulla pystytään luokittelemaan tulovektori edellisen kerroksen *feature map* -vektorien perusteella. Luokittelukerroksen tehokkuus riippuu siitä miten hyvin se on opetettu. Vaikkakin luokittelukerroksesta käytetään eri nimeä, on se pääpiirteiltään sama kuin kappaleessa 2.1.2 esitetty MLP-verkko.

Luokittelukerroksessa suurin pullonkaula on paljon muistia ja nopeaa kaistaa vaativat painoarvovektorit, ja ongelma kasvaa eksponentiaalisesti kun luokittelukerroksessa käytetään suurta määrää solmuja. Yksi mahdollisuus on jakaa luokittelukerroksen painoarvot konvoluutiokerroksen kanssa. Toinen vaihtoehto on käyttää yhtä jaettua summainta koko luokittelukerroksessa.



Kuva 10. Luokittelukerros.

3.5. Koko verkon toteutus

Konvolutionaalinen neuroverkko vaatii paljon laskentatehoa ja muistia, joten nykyteknologialla sen toteuttaminen vaatii yksinkertaistamista, jolloin opetusaika ja tarkkuus kärsivät. Tämä näkyy erityisesti tilanteissa, joissa halutaan toteuttaa konvolutionaalinen neuroverkko sulautetusti.

Konvoluutioverkossa on monta osaa, joten sitä voidaan optimoida monella tapaa, mikä tekee myös optimaalisen verkon löytämisestä haastavaa. Yleisimmät optimointitavat liittyvät konvoluutiokerroksen tai luokittelukerroksen optimointiin, mutta nykyään tutkimus on keskittynyt enemmän konvoluutiokerrokseen [2]. Ensimmäisessä ongelmana on suuri MAC-operaatioiden määrä ja jälkimmäisessä suuri vaadittava muistin määrä. Lisäksi kaistanleveyden tulee olla sopiva prosessointitehoon nähden.

Konvoluutioverkon tehokkuutta mitattaessa voidaan käyttää useita erilaisia tunnuslukuja riippuen toteutuksesta. Koska konvoluutiokerroksen oleellinen tekijä on MAC-operaatio ja siten verkon tehokkuutta usein mitataan yksiköllä GMAC/s, eli miljardi MAC-operaatiota sekunnissa. Toinen käytettävä yksikkö on GFLOPS, eli miljardi liukulukuoperaatiota sekunnissa. Lisäksi käytetään myös laajempaa tunnuslukua GOP/s, eli miljardia operaatiota sekunnissa. Erityisesti sulautetuissa sovelluksissa vaaditaan suurta tehokkuutta pinta-alaa kohden, jolloin voidaan käyttää yksikköä GOPS/Slice, eli miljardia operaatiota sekunnissa pinta-ala-yksikköä kohden. GOPS/Slice soveltuu siten erilaisten FPGA-toteutusten vertaamiseen, sillä se ei ole riippuvainen käytetystä laitteistosta [6]. Oleellisesti FPGA-piirin energiatehokkuuteen vaikuttavat käytetty lukutarkkuus ja muistin kaistanleveys. Lisäksi konvoluutioverkon tulee hyödyntää käytettävää rautaa mahdollisimman tehokkaasti [2].

Verkon tehokkuutta tarkastellessa on oleellista myös mihin sovellukseen FPGA-CNN toteutetaan; sulautetussa järjestelmässä on tarkemmat kriteerit tehonkulutukselle ja pinta-alalle kuin pelkällä FPGA-kehityslaudalla suunnitellulla neuroverkolla. Neuroverkot ovat tehokkaita ratkaisemaan tietynlaisia ongelmia, joten nykyään on suosittua toteuttaa neuroverkko sulautettuna piirinä, niin sanottuna kiihdyttimenä (*accelerator*). Tosielämän käyttökohteita sulautetuille neuroverkoille

löytyy paljon, esimerkiksi itseohjautuvien autojen kameroista ja iPhone-puhelimien kasvojentunnistimista.

CNN on tehokas työkalu kun sitä käytetään sille soveltuvissa ongelmissa ja asianmukaisella rautatason toteutuksella. Vaadittava laskentateho riippuu paljon tulossa olevan kuvan dimensioista. Esimerkiksi kuvantunnistuksessa riippuu onko tulossa 3-kanavainen RGB-kuva, 16-bittinen harmaasävykuva vai yksinkertainen mustavalkokuva. RGB-kuvassa voidaan hyödyntää 3-ulotteista konvoluutiokerneliä, mutta tällöin laskenta on huomattavasti vaativampaa kuin 2-ulotteisessa kuvassa.

Konvoluutioneuroverkon suunnittelijan vastuulla on siis tiedostaa hyvin selvästi se ongelma-avaruus, jossa neuroverkko tulee toimimaan, sekä käytetyn laitteiston mahdollisuudet ja rajoitukset. FPGA:n ominaisuuksia hyödyntämällä voi suunnittelija tehdä nopeita muutoksia ja optimoida neuroverkko mahdollisimman hyvin.

4. POHDINTA

Konvolutionaalinen neuroverkko on haastava toteuttaa FPGA-piireillä ilman kompromissejä tarkkuuden suhteen. FPGA on rinnakkaistoiminnallaan hyvin soveltuva konvolutionaalisen neuroverkon toteutukseen, mutta yksittäisten kerrosten erilaiset vaatimukset tekevät optimoinnista haastavaa. Oleellisimmat tekijät ovat FPGA:n tehokkuus pinta-alaa kohti, muistin kaistanleveys ja nopeus, MAC-lohkojen määrä ja nopeus, sekä käytettyjen lukuformaattien pituus. Konvolutionaalisen neuroverkon toteutus vaatii myös sopivien hyperparametrien valitsemista, jolloin suunnittelijan vastuulla on löytää optimaalinen verkon rakenne lukuisista vaihtoehdoista. Toisaalta tekoälyn saralla tehdään paljon tutkimusta parhaiden käytäntöjen löytämiseksi, joten hyväksi todettuja neuroverkkoja muodostuu tutkimuksen myötä.

MLP-verkon toteuttaminen on triviaali konvolutionaaliseen neuroverkkoon nähden ja olennaisesti konvolutionaalisessa neuroverkossa on MLP-verkko luokittelukerroksessa. Siten konvolutionaaliverkko on huomattavasti haastavampi toteuttaa. Tämä myös kuvastaa hyvin tekoälyn saralla tapahtunutta kehitystä vuosikymmenien aikana.

Nanovalmistusteknologioiden kehittyessä pystytään valmistamaan entistä tiiviimpiä piirejä kasvattaen tehokkuutta tilavuutta kohden, mutta FPGA-piirien reititys vie paljon tilaa, mikä on konkreettinen haaste piirinvalmistuksessa. Erityisesti sulautetuissa toteutuksissa, kuten matkapuhelimien tekoälykiihdyttimissä, vaaditaan pientä pinta-alaa.

Tässä työssä esiteltiin kahden erilaisen neuroverkon toimintaperiaatteet ja selvitettiin toteutusmahdollisuuksia FPGA:lla. On kuitenkin huomattava, että lisäkirjallisuuteen on hyvä tutustua, mikäli haluaa toteuttaa työssä esiteltyjä neuroverkkoja käyttämällä FPGA-piiriä.

Neuroverkkojen toteutuksessa haastavaa on niille luonteenomainen todennäköisyyksiin perustuva toiminta. Lähdön määrittäminen verkon painoarvojen perusteella on vaikeaa, joten verkon käytännön toimintaa voidaan arvioida vasta lopuksi. Virheiden minimointi on oleellisessa osassa ja verkon tarkkuus mitataan todennäköisyyksinä, joten satunnaisuus tekee käytännön toteutuksen haastavaksi. Siten neuroverkkojen toteuttaminen on nykyisellään haastavaa sovelluskohteissa, joissa vaaditaan ehdotonta tarkkuutta. On kuitenkin huomattava, että sovelluskohteissa, kuten kuvantunnistus, joissa neuroverkot kilpailevat ihmisen kanssa, se on usein nopeampi reagoimaan. Suunnittelijan tulee olla tietoinen näistä tekoälyn rajoituksista ja käyttää huolellisuutta neuroverkkojen implementoinnissa.

5. YHTEENVETO

Tässä työssä esiteltiin yleisellä tasolla MLP- ja konvolutionaalisen neuroverkon perusperiaatteet ja toteutusmahdollisuudet. FPGA-piirin rinnakkaisuus on selkeä etu kun halutaan toteuttaa massiivista rinnakkaisuutta hyödyntävää neuroverkkoa ja FPGA:n uudelleenohjelmoitavuus mahdollistaa yksinkertaisen prototypioinnin neuroverkkoa optimoidessa.

Konvolutionaalisen neuroverkon haasteet laskentatehon ja muistin osalta ovat suurin haaste FPGA-toteutukselle. 90 prosenttia laskennasta tapahtuu CNN:n konvoluutiokerroksissa kun taas luokittelukerros vaatii suurta muistin määrää ja kaistanleveyttä. FPGA-piireissä 90 prosenttia pinta-alasta menee loogisten lohkojen väliseen reititykseen, joten tehokkaan FPGA-piirin suunnittelu vaatii kompromisseja pinta-alan, sekä rinnakkaisuuden ja tehokkuuden välillä.

Neuroverkkojen sulautettu toteutus tekoälykiihdyttimenä esimerkiksi matkapuhelimissa vaatii huolellista optimointia tehonkulutuksen, ja laskentatehon suhteen. Konvolutionaalisen neuroverkon ongelmanratkaisukyky on havaittu erittäin tehokkaaksi kuvantunnistuksessa, joka on ollut perinteisesti haastava laskennallinen ongelma.

6. LÄHTEET

[1] Beale, R., & Jackson, T. (1998). Neural computing: An introduction (Reprinted ed.). Bristol [u.a.]: Inst. of Physics Publ.

[2] Guo, K. (2018). Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 37*(1), pp. 35-47.

[3] Yuteng Zhou, Shrutika Redkar, & Xinming Huang. (Jan 1, 2017). Deep learning binary neural network on an FPGA. Paper presented at the 281.

[4] Farooq U., Marrakchi Z., Mehrez H. (2012) FPGA Architectures: An Overview. In: Tree-based Heterogeneous FPGA Architectures. Springer, New York, NY

[5] Nurvitadhi, E., Subhaschandra, S., Boudoukh, G., Venkatesh, G., Sim, J., Marr, D., … Moss, D. (2017). Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '17. ACM Press. https://doi.org/10.1145/3020078.3021740

[6] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. (2015). Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA 15*. doi:10.1145/2684746.2689060