Tampereen yliopisto

Matti Valkeinen

LASKENTAPALVELINTEN HYÖDYNTÄMINEN NEUROVERKKOJEN OPETUKSESSA

Tekniikan ja luonnontieteiden tiedekunta
Kandidaatin työ
Tammikuu 2020

TIIVISTELMÄ

Matti Valkeinen: Laskentapalvelinten hyödyntäminen neuroverkkojen opetuksessa
Kandidaatin työ
Tampereen yliopisto
Tekniikan kandidaatin tutkinto
Tammikuu 2020

Tämän kandidaatin työn tarkoitus on selvittää, miten laskentapalvelimia voitaisiin hyödyntää neuroverkkojen opetuksessa. Suuret laskentaoperaatiot saattavat vaatia hyvinkin paljon resursseja, joita laskentapalvelimella on normaalia tietokonetta enemmän. Neuroverkko, joka tässä työssä tehtiin, oli konvoluutioneuroverkko, jolla pyrittiin parantamaan satelliittien rataennustuksia. Näitä satelliittirataennustuksia käytetään esimerkiksi paikannuksessa. Työn neuroverkko ja data pohjautuvat vahvasti Pihlajasalon diplomityöhön [1].

Työssä tutkittiin, miten konvoluutioneuroverkon kerrosten muuttaminen vaikuttaa tuloksiin ja ajankäyttöön. Työssä muutettiin konvoluutioverkon filttereiden määrää, jotta saavutettaisiin optimaalein tulos työssä olevalle datalle. Tuloksena löydettiin optimaalinen määrä filttereitä ja huomattiin, että filttereiden lisääminen tuottaa lineaarisesti kasvavan ajan muutoksen neuroverkon opetuksessa.

Avainsanat: konvoluutioneuroverkot, satelliittien rataennustus, laskentapalvelin, TCSC

ALKUSANAT

Kiitos paljon matematiikan laitokselle aiheesta. Kiitos Jaakko Pihlajasalolle ja Simo Ali-Löytylle ohjauksesta työssä. Kiitos myös sukulaisilleni ja erityisesti siskolleni Teijalle motivoivasta piikittelystä.

Tampereella, 29. tammikuuta 2020

Matti Valkeinen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

$A \circ B$	Hadamardin tulo matriiseille A ja B
A^{T}	transpoosi matriisille A
Δx	muuttujan x virhe
$\mathrm{tr}(A)$	matriisin A jälki
$\mathbb{N}$	luonnolliset luvut
∇_θ	gradientti muuttujan θ suhteen
$\sum_i$	summa yli indeksin i
$o, b, n, m, d, K, O, P, S...$	skalaareja
$v, w...$	vektoreita
$\mathbb{R}$	reaaliluvut
$f(x), g(x), J(\theta)...$	funktioita
$A, B, C, I, W, F...$	matriiseja
CNN	Konvoluutioneuroverkko
FC	Fully connected
GPS	Global positioning system
ReLU	Rectified linear unit
SGDM	Stochastic gradient descent with momentum
SISRE	Signal-in-space ranging errors
TCSC	Tampereen tieteellisen laskennan keskus

1 JOHDANTO

Tässä kandidaatin työssä selvitetään, miten neuroverkot toimivat ja miten niitä voidaan luoda ja opettaa ulkoisilla palvelimilla. Hyödyntämällä ulkoisia palvelimia saavutetaan massiiviset erot laskentatehossa verrattuna esimerkiksi kannettavaan tietokoneeseen tai jopa hyvin varusteltuun pöytätietokoneeseen. Työ pohjautuu vahvasti Pihlajasalon diplomityöhön [1], jossa luotiin neuroverkko parantamaan satelliittiratojen ennusteita. Näitä ennusteita hyödynnetään esimerkiksi GPS-laitteiden paikannuksissa. Paikannukseen tarvitaan useampi satelliitti ja niiden sijainnit. Satelliittien tarkka sijainti on tietoa, joka ei ole välittömästi saatavilla, joten nopeassa paikannuksessa on turvauduttava ennusteisiin. Virheet ennusteissa näkyvät siis myös lopullisessa GPS-laitteen paikannuksessa.

Neuroverkkoja voidaan kuvata mustina laatikkoina, joille annetaan syöte ja ne tuottavat lopputuloksen tietämättä enempää, mitä laatikon sisällä tapahtuu. Neuroverkot toimivatkin työssä ratkottavan ongelman ratkaisemiseen erinomaisesti, sillä satelliittiratojen data on monimutkaista ja ainoastaan neuroverkon tuottamalla lopputuloksella on merkitystä. Neuroverkkojen opetus saattaa vaatia paljon dataa ja laskentaresursseja, minkä vuoksi työssä kokeillaan neuroverkkojen opettamista ulkoisilla palvelimilla.

Työn alussa esitellään työn kannalta oleellinen teoria. Teoriassa esitellään tarkemmin neuroverkon komponentteja ja niiden funktiota neuroverkon kokonaiskuvassa sekä opetukseen ja testaukseen liittyvää teoriaa. Tämän jälkeen esitellään taustaa, miten satelliittien rataennustuksia on pyritty parantamaan sekä esitellään työssä käytetty neuroverkko ja sen tuottamat tulokset. Lopussa esitellään työn yhteenveto, jossa esitellään vielä lyhyesti tulokset ja mietitään, mitä työssä olisi voinut tehdä toisin.

2 TEOREETTINEN TAUSTA

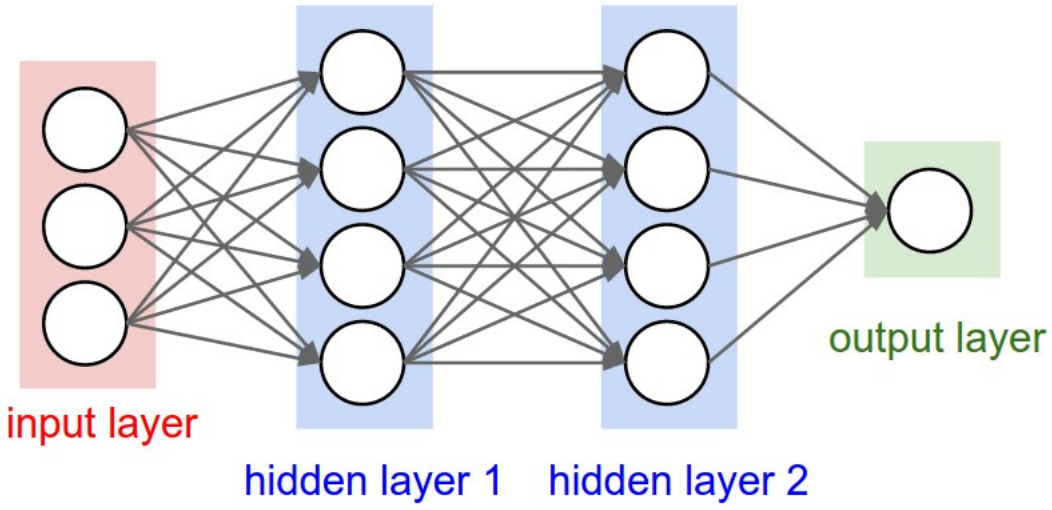
Tässä luvussa esitellään työn taustalla oleva teoria.

2.1 Konvoluutioneuroverkot

Konvoluutioneuroverkot ovat neuroverkkoja, jotka koostuvat neuroneista, joilla on opittavia painoja ja vakioita. Jokainen neuroni saa jonkin syötteen, suorittaa pistetulon ja seuraa valinnaisesti tulosta ei-lineaarisesti. [2] Yksittäisen neuronin vaste o voidaan määritellä seuraavasti:

$$o_j = \sum_{i=1} g(w_{ij}x_i + b_j), \tag{2.1}$$

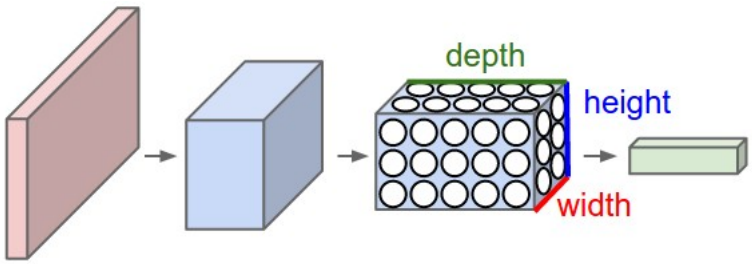
jossa o_j on järjestysnumerolla j oleva neuronin vaste, x_i on järjestysnumeroa i vastaava syöte neuronille, w_{ij} on neuronin syötettä vastaava paino, b_j on neuronin vakio ja g on jokin edellä mainituille muuttujille suoritettu funktio, kuten esimerkiksi konvoluutio.



Kuva 2.1. *Kuva tavallisesta kolmikerroksisesta neuroverkosta. Syöte-kerrosta ei lasketa mukaan laskettaessa kerrosten määrää. Haettu [2].*

Kuvassa 2.1 on yksinkertainen neuroverkko, jonka syötteen koko on kolme alkiota. Nämä kolme alkiota kulkevat kerroksen läpi, josta saadaan kaavan (2.1) mukainen vastine, joka yhdistetään piilokerroksessa (hidden layer) neuroniin. Neuronien vastineita käytetään

seuraavien kerroksien syötteinä. Viimeinen kerros neuroverkoissa yhdistää edellisen kerroksen tulokset haluttuun muotoon.



Kuva 2.2. *Kuva kolmikerroksisesta konvoluutioneuroverkosta. Syötteenä kuva, jolla on kolme dimensiota: korkeus, leveys ja syvyys. Syvyys vastaa kuvassa esimerkiksi RGB-värikanavia. Haettu [2].*

Konvoluutioneuroverkon syötteinä käytetään kuvia, jotka voidaan merkitä $I \in \mathbb{R}^{m \times n \times d}$, jossa m on korkeus, n leveys ja d syvyys. Idealtaan konvoluutioneuroverkko ei kuitenkaan poikkea normaalista neuroverkosta: syötteet kulkevat läpi kerroksen, josta ne saavat vastineet, jotka yhdistetään piilokerroksessa. Konvoluutioneuroverkon piilokerroksessa neuronit asetetaan kolmiulotteiseksi aktivointikerrokseksi $W_i \in \mathbb{R}^{m_i \times n_i \times d_i}$. Neuronien määrää on myös mahdollista rajata ja kasvattaa erilaisilla kerroksilla.

Neuroverkkojen käyttö soveltuu hyvin ongelmiin, joissa opetettava data on epätarkkaa, kompleksista sensoridataa, kuten kameran ja mikrofonin syöte. Neuroverkot ovat hyviä työkaluja tilanteissa, joissa ihmisen ei ole tarpeellista ymmärtää, miksi neuroverkko on painottanut tiettyjen neuroneiden vastineiden merkityksiin, mikä mahdollistaa niin kutsuttujen piilopiirteiden löytämisen. [3]

2.2 Verkkokerrokset

Konvoluutioverkoilla on käytössä monenlaisia kerroksia, jotka soveltuvat eri tarkoituksiin. Tässä työssä oleellisimmat ovat konvoluutiokerros, ReLu-kerros (Rectified Linear Unit) ja FC-kerros (Fully Connected). Näiden lisäksi työssä käytetään kuvan syöttökerrosta (Image input layer), jonka tarkoituksena on ilmoittaa verkolle sille tulevan syötteen koko ja normalisoida se.

Konvoluutiokerros on konvoluutioverkkojen tärkein kerros. Konvoluutiokerrokselle annetaan neljä (4) hyperparametriä: filttereiden määrä K, filttereiden avaruudellinen alue F_s, filtterin askellusväli kuvalle S ja 0-tasoituksen määrä P. Käytettäessä konvoluutiokerrosta voidaan kerroksen tuloksen O_s leveyden tai korkeuden koko laskea kaavalla

$$O_s = \frac{I_s - F_s + 2P}{S} + 1, \tag{2.2}$$

jossa I_s on syötteen vastaavan dimension koko, F_s filtterin vastaavan dimension koko ja P syötteen reunoille lisättävien nollien määrä. [2] Näitä parametreja hienosäätämällä voi

askelluksen S kokoa muuttaa suuremmaksi ja täten pienentää ulostulon kokoa. On kuitenkin huomioitava, että ulostulon täytyy olla kokonaisluku. [2] Tuloksen leveys ja korkeus voivat olla yhtä suuret tai pienemmät kuin alkuperäisen syötteen leveys ja korkeus.

Konvoluutiokerroksessa lasketaan 2D-konvoluutio kuvan kaikille värikanaville. Konvoluutiossa käytettävän filtterin dimensiot voivat olla pienempiä tai yhtäsuuria kuin syötteenä olevan kuvan. Konvoluution filtteri B kulkee kulkee kuvan A läpi valittujen askellusvälin S ja tasoituksen P mukaisesti. Konvoluutio voidaan määritellä Hadamardin tulojen summilla, jotka jaetaan matriisin alkioiden lukumäärällä. Hadamardin tulo tunnetaan myös Schurin tulona. Hadamardin tulo alkioittain on määritelty samankokoisille matriiseille A ja B seuraavasti

$$(A \circ B)_{(i,j)} = A_{(i,j)} B_{(i,j)}, \tag{2.3}$$

jossa i ja j ulottuvat matriisien kokoon. [4]

Hadamardin tulomatriisin elementtien summa on määritelty

$$\operatorname{tr}(AB) = 1_{1\times m}(A^{\mathrm{T}}B)1_{n\times 1}, \tag{2.4}$$

jossa m ja n ovat matriisien dimensioiden koot ja $1_{1\times m}$ ja $1_{n\times 1}$ ovat vektoreita, joiden kaikki alkiot ovat 1. [4]

Täten 2D-konvoluutio samankokoiselle kuvalle A ja B voidaan määritellä

$$c = b + \frac{1}{dn}\sum_{i=1}^{d} \operatorname{tr}(A_i B_i), \tag{2.5}$$

jossa c on lopullisen konvoluutiomatriisin C alkio, b on filtterille asetettu painatus, d on kuvan värikanavien määrä, n on kuvassa olevien alkioiden (pikseli) määrä yhdellä värikanavalla ja A_i ja B_i ovat kuvan ja filtterin matriisit värikanavalla i. Lopullisen konvoluutiomatriisin C dimensioihin vaikuttavat kaavan (2.2) mukaisesti kuvan I_s ja filtterin F_s lisäksi tasoitus P ja askellusväli S.

$$
\overset{I_s(:,:,1)}{\begin{bmatrix} -1 & 1 & -1 & 1 & -1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}} \qquad \overset{F_s(:,:,1)}{\begin{bmatrix} -1 & 0 & -1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}}
$$

$$
\overset{I_s(:,:,2)}{\begin{bmatrix} 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 \end{bmatrix}} \quad * \quad \overset{F_s(:,:,2)}{\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}} \quad = \overset{C}{\begin{bmatrix} 8 & 1 & 3 \\ 2 & 1 & -1 \end{bmatrix}}
$$

$$
\overset{I_s(:,:,3)}{\begin{bmatrix} -1 & 0 & -1 & 0 & -1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}} \qquad \overset{F_s(:,:,3)}{\begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}}
$$

Kuva 2.3. *Esimerkki konvoluutiokerroksen laskemasta kuvan $I_s \in \mathbb{R}^{4\times5\times3}$ ja filtterin $F_s \in \mathbb{R}^{3\times3\times3}$ konvoluutiosta. Esimerkiksi vasemman yläkulman arvo tulosmatriisista C voidaan laskea asettamalla filtteri samalle kohtaa kuvaa ja laskemalla arvo kaavalla* (2.5)*.*

Kuvassa 2.3 on esimerkki kuvan konvoluutiosta. Vasemmalla olevat matriisit kuvastavat kuvaa $I_s = 4 \times 2$ ja sen eri värikanavia. Keskellä kuvaa on kuvan yli vietävä filtteri $F_s = 3 \times 3$, jolla on jokaiselle värikanavalle oma filtteri. Esimerkissä on asetettuna 0-tasoitus $P = 0$, filtterin askellus $S = 1$ sekä painatus $b = 0,003$. Filtteri mahtuu kuvaan 6 erilaiseen asentoon ja niinpä ulostulon koko $O_s = 2 \times 3$. Jokaiselle mahdolliselle asennolle lasketaan Hadamardin tulon summa (2.4). Positiiviset arvot tulosmatriisissa C viittaavat filtterin löytämiin piirteisiin. Negatiiviset arvot puolestaan kuvaavat tilannetta, joissa filtterin ja kuvan välille ei löytynyt yhteneväisyyksiä.

$$
0,003 + \frac{1}{3 \cdot 9} \begin{bmatrix} 8 & 1 & 3 \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 0,2993 & 0,0400 & 0,1141 \\ 0,0771 & 0,0400 & -0,0340 \end{bmatrix} \tag{2.6}
$$

Kuvan 2.3 esimerkkiä on jatkettu kohdassa (2.6). Hadamardin tulon summien laskemisen jälkeen jaetaan nämä alkiot termillä $d \cdot n$, jossa dimensioiden määrä $d = 3$ ja filtterin alkioiden määrä $n = 9$. Lopuksi lisätään painokerroin $b = 0,003$ tulosmatriisin alkioille.

ReLU-kerroksen tarkoitus konvoluutioverkoissa on ylläpitää matemaattista terveyttä opitulla arvoilla estämällä lukuja jäämästä nollan läheisyyteen ja estämällä lukuja karkaamasta äärettömään [5]. ReLU-kerroksen syöte ja tulos ovat saman kokoisia. ReLU-kerroksessa

syötteenä olevan matriisin alkioille suoritetaan funktio

$$f(x) = \max(0, x), \tag{2.7}$$

jossa x on syötteen alkio. ReLU-kerroksen tuloksena on siis matriisi tai vektori, jonka positiiviset arvot ovat säilyneet ja negatiivisten arvojen tilalle on asetettu 0.

$$f_{(2.7)}\left(\begin{bmatrix} 0,2993 & 0,0400 & 0,1141 \\ 0,0771 & 0,0400 & -0,0340 \end{bmatrix}\right) = \left(\begin{bmatrix} 0,2993 & 0,0400 & 0,1141 \\ 0,0771 & 0,0400 & 0 \end{bmatrix}\right)$$

Kuva 2.4. *Esimerkki ReLU-kerroksen käytöstä kuvan 2.3 tulokselle.*

Konvoluutioverkoissa on mahdollista käyttää myös datamäärää vähentävää, yhdistävää kerrosta (pool layer). Kyseisiä kerroksia käytetään tilanteissa, joissa halutaan pienentää laskettavan aineiston määrää. Suosituimpana yhdistämisen keinona käytetään max poolia (suurimman arvon säilyttävä), jossa valitulta alueelta säilytetään suurin arvo. Eräs toinen tapa pienentää datan kokoa on average pool (keskiarvon säilyttävä), mutta sen käyttöä on vähennetty ajan saatossa max poolin parempien tuloksien vuoksi. [2] Yhdistävien kerroksien käytöstä on myös eriäviä mielipiteitä, joiden mukaan dataa yhdistävien kerroksien käyttöä tulisi välttää ja sen sijaan käyttää useita konvoluutiokerroksia [6].

$$\max \text{pool}_{1x3}\left(\begin{bmatrix} 0,2993 & 0,0400 & 0,1141 \\ 0,0771 & 0,0400 & -0,0340 \end{bmatrix}\right) = \left(\begin{bmatrix} 0,2993 \\ 0,0771 \end{bmatrix}\right)$$

Kuva 2.5. *Esimerkki max poolin käytöstä kuvan 2.4 tulokselle* 1×3 *kokoisilla alueilla.*

FC-kerroksessa kaikki edeltävän kerroksen neuronit kytketään tulosvektoriin, josta muodostuu neuroverkon lopullinen tulos. Tulosvektori v_{FC} muodostuu seuraavasti

$$v_{FC} = W v_{input} + b^T, \tag{2.8}$$

jossa $W \in \mathbb{R}^{m \times n}$ on neuroverkon painomatriisi, $v_{input} \in \mathbb{R}^n$ on edellisen kerroksen kytketyistä tuloksista muodostunut vektori ja b on tulosten vakiotermit. [7]

2.3 Neuroverkkojen opetus ja testaus

Neuroverkon opetuksessa neuroverkko parantelee sen käyttämiä painoja ja vakiotermejä vastavirta (backpropagation) -algoritmilla. Neuroverkot pohjustetaan satunnaisilla muuttujilla, jolloin neuroverkko laskee tulokset kuvalle, arvioi sen virhettä kustannusfunktion

avulla ja korjaa muuttujia kustannusfunktion tuloksen mukaan. Ajan säästämiseksi on myös mahdollista alustaa neuroverkko ennalta määritellyillä painoilla ja vakiotermeillä, jolloin mahdollisia korjauksia tapahtuu vähemmän, mikäli on tiedossa neuroverkolle sopivat muuttujien arvot. Tässä työssä käytetään neuroverkolla regressiotasoa, jonka kustannusfunktio on *half-mean-square-error* (HMSE), eli puolittainen jäännösvarianssi. HMSE määritellään

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \frac{(t_i - y_i)^2}{n}, \tag{2.9}$$

jossa θ on kaikki neuroverkon painot, n on ennusteiden määrä, t_i on tiedetty tulos ja y_i on vastaavan ennusteen tulos. Vastavirta-algoritmissa painojen ja vakiotermien korjaus tapahtuu stokastisen gradientin momentin menetelmällä (Stochastic gradient descent with momentum, SGDM), joka on määritelty

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta), \tag{2.10}$$

jossa v_t on momenttivektori, v_{t-1} iteraatiota edeltänyt momenttivektori, γ momentin taso kerroin, η oppimisen tasokerroin ja $J(\theta)$ on gradientin kustannusfunktio, kuten esimerkiksi kaavassa (2.9) määritelty. [8] Iteraatioita jatketaan määrättyyn rajaan asti, joka voi olla esimerkiksi iteraatioiden määrä, aika tai riittävän tarkkuuden saavuttaminen.

Neuroverkon opettamisen jälkeen on syytä testata tuloksia kuvilla, joita ei ole käytetty opetuksessa. Testauksessa pyritään selvittämään, ettei neuroverkko ole ylisovittanut dataa, jolloin neuroverkon mallista on tullut liian lineaarinen. [9] Neuroverkon tuottamia tuloksia testattaville kuville verrataan aitoihin tiedettyihin tuloksiin ja sen pohjalta lasketaan neuroverkolle tarkkuus.

3 SATELLIITTIEN RATAENNUSTEIDEN LASKEMINEN NEUROVERKOILLA

Tässä luvussa esitellään lyhyt pohjustus siihen, miten neuroverkkoja päädyttiin käyttämään satelliittiratojen ennusteiden parannuksessa. Lisäksi esitellään tämän työn käyttämä konvoluutioneuroverkko ja sillä saadut tulokset.

3.1 Taustaa

Tämän kandidaatin työn data ja idea ovat lähtöisin Pihlajasalon diplomityöstä [1], jossa tutkittiin, miten spektrianalyysin ja syväoppimisen avulla voitaisiin parantaa satelliittiratojen ennusteita. Osalla satelliittiradoista satelliitit tekevät säännöllisiä liikkeitä pysyäkseen radalla. Pihlajasalon työssä pyrittiin ennustamaan ajanhetket, jolloin nämä liikkeet tehdään paikannuksen parantamiseksi, hyvin tuloksin.

Paikannusalgoritmien tutkimusryhmä, johon Pihlajasalokin kuului, oli aiemmin tutkinut muita tapoja parantaa satelliittiratojen ennustuksen laatua ja tarkkuutta. Tutkimusryhmä muodosti satelliittiin vaikuttavista voimista rataennustusmallin, joka hyödyntää vain voimia, joilla on suurimmat vaikutukset satelliitin liikkumiseen. Näistä suurimpia voimia ovat muun muassa Maan, Kuun ja Auringon gravitaatiovuorovaikutukset ja Auringon säteilypaine. Muita huomioituja voimia ovat muun muassa maankuoren vuorovaihtelut, suhteellisuusteoreettiset korjaukset ja muiden planeettojen gravitaatiovuorovaikutukset. Malli paransi ennustuksia hiukan. [10]

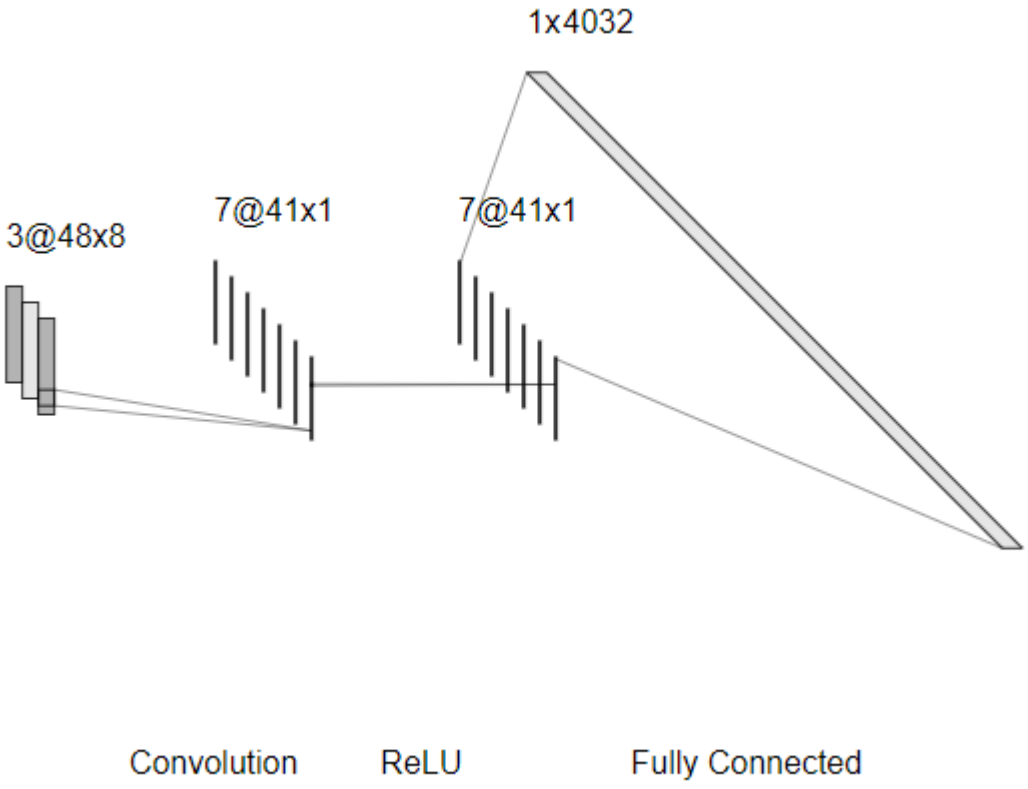
Muitakin datapainotteisia lähestymistapoja parantaa satelliittiratoja on tutkittu aiemmin mainitun syväoppimisen lisäksi. Piileviä satelliittiratoihin vaikuttavia voimia voidaan arvioida ennusteilla tai tarkalla datalla ja näitä voimia voidaan käyttää satelliittirataennustusten parantamiseen. Näillä on onnistuttu parantamaan ennustuksia. [11, 12]

3.2 Neuroverkon käyttö ennustuksessa

Opetettava data oli annettu valmiiksi kuvamuodossa, jossa yhden kuvan koko oli $48 \times 8 \times 3$, jossa dimensiot vastaavat järjestyksessä orbitaalin jaksoa, ennustuksen kestoa (päivissä) ja kuvan RGB-tasoja, jotka koostuvat radiaali-, tangentiaali- ja normaalivirheistä vastaavassa järjestyksessä. Lisäksi datalle oli annettu kuvaa vastaavat virheet seuraavalta kahdelta viikolta. Nämä virheet olivat jaettuna kolmeen vektoriin, jotka muodostivat matriisin $R \in \mathbb{R}^{3\times 1344}$.

Data oli jaettu kahteen osaan, jossa toinen oli opetusta ja toinen testausta varten. Tämän tarkoitus on estää mahdollinen ylioppiminen, jolloin neuroverkko oppii liikaa annetusta datasta. Kuvien suhde opetettavan ja testattavan datan välillä oli kahden suhde yhteen $(2 : 1)$.

Työssä käytetty neuroverkko toteutettiin MATLAB-ohjelmistolla ja täten sen rakenne poikkeaa hiukan esimerkiksi pythonin avoimella Keras-kirjastolla toteutetusta neuroverkosta. Neuroverkolle oppimisen tasokertoimeksi kokeiltiin eri arvoja ja lopulliseksi arvoksi valittiin $\eta = 0,001$. Valitsemalla tasokertoimeksi liian ison luvun olivat tulokset heikkoja ja valitsemalla liian pienen luvun neuroverkon opetukseen kulunut aika oli epämiellyttävän pitkä ja tulokset eivät juurikaan parantuneet. Neuroverkon optimoijaksi valittiin SGDM (2.10).



Kuva 3.1. *Työssä käytetyn konvoluutioneuroverkon kaavio. Konvoluutiokerroksessa* $K = 7$.

Koulutus tapahtui yksinkertaisella neuroverkolla. Verkon ensimmäinen kerros oli kuvan syöttö -kerros, joka määritti syötteen koon ja suoritti normalisoinnin datalle. Tämän jälkeen tuli konvoluutiokerros. Testeissä testattiin konvoluutiokerroksessa eri määriä ulostuleville piirteille, toisin sanoen käytössä olevien filttereiden määrälle K ja sitä, miten ne vaikuttivat koulutuksen kestoon ja tuloksiin. Konvoluutiokerroksen konvoluutiossa käyttämän filtterin koko oli 8×8, askellus $S = 1$ ja reunan tasoitus $P = 0$. Käyttämällä kaavaa (2.2) saadaan konvoluutiokerroksen ulostulon kooksi vektori, jonka pituus on 41 jokaiselle käytetylle filtterille K.

Konvoluutiokerroksen jälkeen data siirtyi ReLU-kerrokseen, jossa suoritetaan kaavan (2.7) mukainen operaatio datalle. ReLU-kerroksesta data siirtyi FC-kerrokseen, josta muodostui tässä työssä käytetyn neuroverkon tulos. FC-kerroksessa määriteltiin ulostulevien parametrien määräksi 4032, joka vastaa orbitaalin virheiden kahden viikon mittaista ajanjaksoa 15 minuutin intervallein.

3.3 TCSC-palvelimen hyödyntäminen

Työssä hyödynnettiin TCSC-palvelinta. Yhteys palvelimeen otettiin tietokoneelta komentorivityökalulla. Palvelimella oli käytössä interaktiivinen käyttöliittymä sekä komentorivityökalut MATLABin ajoa varten. Laskentaa ajettaessa oli mahdollista määrittää käytettävien ydinten määrä, jolla MATLAB pääasiassa suorittaa laskutoimitukset. [13]

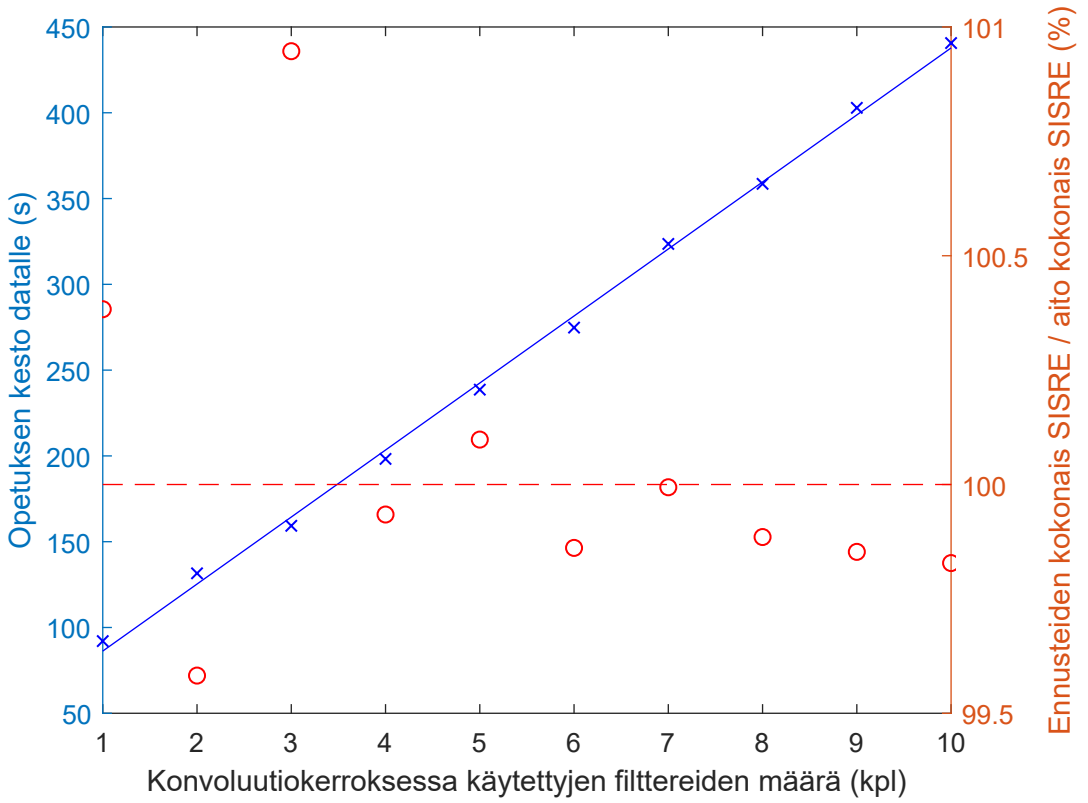
Työhön tuotettu MATLAB-koodi suoritettiin TCSC-palvelimen Narvi-klusterissa, johon määriteltiin Slurm-ohjelmiston avulla työ, joka ajoi laskutoimitukset, kun klusterilla oli tarpeeksi tilaa muilta ajoilta. Laskutoimitukset oli myös mahdollista siirtää rinnakkaisiksi operaatioiksi *Parallel Computing*-kirjastolla, mutta se ei ollut tämän työn kannalta tarpeellista, sillä opetettavan datan määrä oli pidetty maltillisena työn mielekkyyttä ajatellen, eikä opetuksen kesto yksittäisellä ytimelläkään ollut kovin pitkä.

3.4 Tulokset

Tuloksien oikeellisuutta arvioitiin vertailemalla alkuperäisiä vastineita neuroverkolla ennustettuihin vastineisiin. Tämän lisäksi verrattiin aidoista ja ennustetuista virheistä laskettuja SISRE (signal in space ranging errors) -lukuja, jotka kertovat, montako metriä virheet vastaavat. SISRE lasketaan kaavalla

$$\mathrm{SISRE} = \sqrt{w_R \Delta R^2 + w_{T,N}^2 (\Delta T^2 + \Delta N^2)}, \tag{3.1}$$

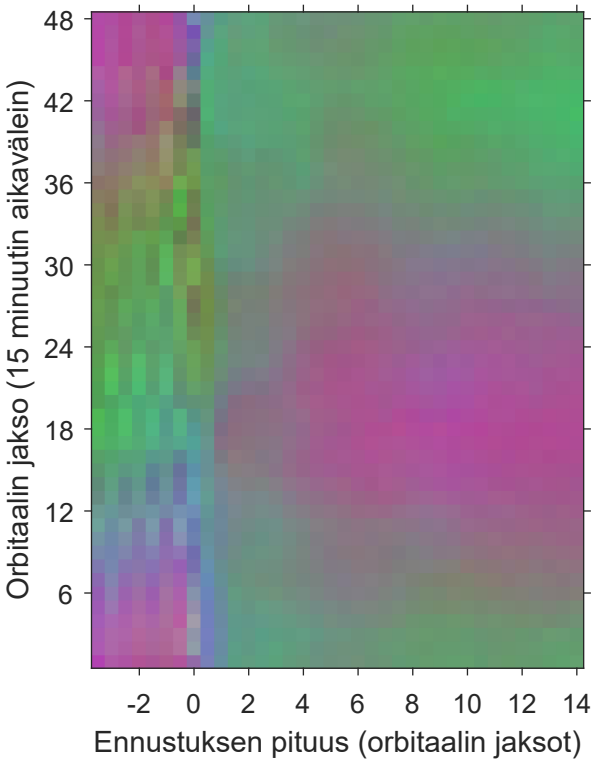
jossa ΔR, ΔT ja ΔN ovat radiaalin, tangentiaalin ja normaalin suuntaiset virhekomponentit. w_R ja $w_{T,N}$ ovat painokertoimia. SISRE-painokertoimet GPS-mallille ovat $w_R = 0,98$ ja $w_{T,N}^2 = \frac{1}{49}$. [14]



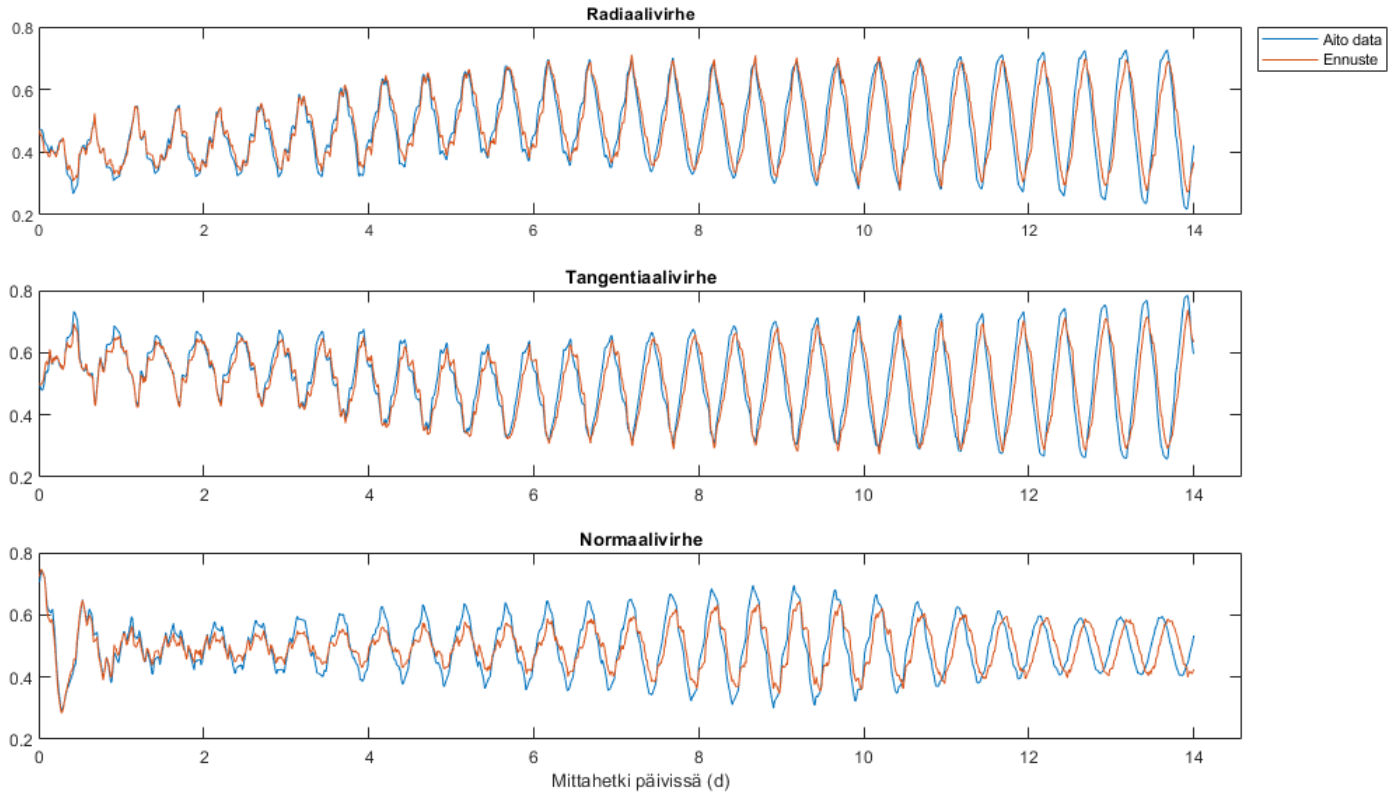
Kuva 3.2. *Työssä käytetyn yksinkertaisen neuroverkon opetuksen kesto ja koko datan yhteenlasketun SISREn suhde aidon ja ennustetun datan välillä eri filttereiden määrälle K konvoluutiokerroksessa.*

Opetettaessa neuroverkkoa suoritettiin opetus konvoluutiokerroksen arvoille $K \in \mathbb{N}^{\leq 10}$. Kuvassa 3.2 nähdään opetuksen kesto sekunteina eri määrille filttereitä K vasemmanpuoleisella y-akselilla. Ajoista muodostunut kuvaaja tuotti halutun tuloksen, sillä konvoluution laskeminen on lineaarinen operaatio, joka näkyy myös ajoissa lineaarisena kasvuna, kun konvoluutiokerrokselle kasvatetaan filttereiden määrää.

Kuvassa 3.2 on oikeanpuoleiselle y-akselille merkitty konvoluutiokerroksen filttereiden määrän vaikutus aitojen ja ennustettujen virheiden SISREn suhteelle. SISRE on laskettu jokaiselle ajanhetkelle kaavalla (3.1). Lopullinen prosentuaalinen suuruus on saatu jakamalla ennustetun datan SISREn summat aidon datan SISREn summalla. Tuloksissa filttereiden määrän K ollessa 7 saavutettiin parhaat tulokset, saavuttaen arvon $99,9940\%$, joka on lähimpänä aitoa virhettä.



Kuva 3.3. *Yksittäisen orbitaalin ennuste. Negatiiviset arvot ennustuksen pituudessa vastaavat alkuperäisiä arvoja ja positiiviset ovat neuroverkon ennustamia arvoja kahdelle viikolle. Jokainen sarake vastaa 12 tunnin jaksoa.*



Kuva 3.4. *Yksittäisen orbitaalin virheet eriteltynä. Sininen kuvaaja vastaa aitoa dataa ja oranssi kuvaaja vastaa neuroverkon ennustamaa virhettä.*

Työssä käytetyn neuroverkon tulokset voidaan palauttaa takaisin kuvamuotoon, kuten yllä olevassa kuvassa 3.3. Negatiiviset arvot ovat alkuperäinen kuva ja positiiviset arvot vastaavat orbitaalille ennustettuja virheitä kahden viikon päähän. Yksi kolumni vastaa yhtä 12 tunnin jaksoa orbitaalille. Kuvassa 3.4 on eroteltu saman orbitaalin ennusteen virheet ja verrattu niitä alkuperäiseen dataan. Kuvasta on nähtävissä, miten neuroverkko ennustaa hyvin radiaali- ja tangentiaalivirheet, mutta normaalivirhe ei ole yhtä tarkka. Suurin osa tuloksista oli vastaavanlaisia, eli niissä normaalivirhe ei ollut yhtä tarkka kuin kaksi muuta virhettä. Normaalivirheen tulos ei kuitenkaan ole yhtä merkittävä tekijä laskettaessa SISREä (3.1) verrattuna radiaalivirheeseen.

4 YHTEENVETO

Työssä pyrittiin luomaan neuroverkko, jolla voitaisiin parantaa satelliittiratojen ennustuksia. Käytössä oli opetusta sekä testausta varten olevaa dataa, joka koostui yksittäisen radan radiaali-, tangentiaali- ja normaalivirheistä sekä vastineista, joita radan virheet ovat olleet alkuperiodin jälkeen. Lopputuloksena saatiin yksinkertainen konvoluutioneuroverkko, jolla voidaan ennustaa satelliittiratojen virheitä kahden viikon päähän. Konvoluutioverkolla tutkittiin yhden konvoluutiokerroksen filttereiden määrän vaikutusta tuloksiin ja kestoon. Tutkimuksessa verkolla saavutettiin $99,9940\%$ tarkkuus testiaineiston virheistä laskettuun SISREen nähden filttereiden määrän ollessa 7. Työssä käytettiin TCSC-palvelinta tulosten laskemisessa.

Työssä olisi ollut mahdollista tutkia monimutkaisempien verkkojen tehokkuutta ja ajallista kulutusta TCSC-palvelimella esimerkiksi lisäämällä useamman konvoluutiokerroksen neuroverkkoon. Tällä ei kuitenkaan olisi tuloksiin ollut suurta merkitystä, sillä yksinkertaisellakin neuroverkolla saavutettiin riittävän tarkka tulos. Testiaineiston määrän kasvatuksella tai datojen sekoituksella olisi työhön saanut myös datan validoinnin aspektia, jossa neuroverkkojen opetuksen ja validoinnin tulokset olisivat muuttuneet eri datakombinaatioilla. Myös uuden opetusdatan luominen muuntelemalla olemassa olevaa dataa, esimerkiksi kohinan avulla, olisi ollut mielenkiintoista tulosten kannalta.

LÄHTEET

[1] J. Pihlajasalo. Improvement of Satellite Orbit Prediction Accuracy and Quality With Deep Learning and Spectral Analysis. "http://urn.fi/URN:NBN:fi:tty-201710262067"[Online; haettu 05.01.2019]. Tutkielma. TTY, 2017.

[2] A. Karpathy. *Convolutional Neural Networks (CNNs / ConvNets)*. "https://cs231n.github.io/convolutional-networks/". [Online; haettu 05.01.2019].

[3] T. M. Mitchell. *Machine learning*. New York: McGraw-Hill, 1997.

[4] D. S. Bernstein. *Matrix mathematics*. eng. 2. ed. Princeton (N.J.): Princeton University Press, cop. 2009. ISBN: 978-0-691-13287-7. URL: http://www.loc.gov/catdir/toc/ecip0826/2008036257.html.

[5] B. Rohrer. *How do Convolutional Neural Networks work?* "https://brohrer.github.io/how_convolutional_neural_networks_work.html". [Online; haettu 05.01.2019]. 2016.

[6] J. T. Springenberg, A. Dosovitskiy, T. Brox ja M. Riedmiller. Striving for Simplicity: The All Convolutional Net (2014). [Online; haettu 06.01.2019].

[7] L. A. dos Santos. *Artificial Inteligence*. "https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/". [Online; haettu 20.01.2019].

[8] S. Ruder. *An overview of gradient descent optimization algorithms*. "https://arxiv.org/pdf/1609.04747.pdf". [Online; haettu 20.01.2019]. Kesäkuu 2017.

[9] T. Hastie, R. Tibshirani ja J. Friedman. *The elements of statistical learning: Data mining, inference, and prediction. 2nd ed., corrected at 11th printing*. English. New York: Springer, 2016. ISBN: 0172-7397.

[10] A. Pukkila. GNSS satelliitin radan ennustamisen virhetermien analysointi. "http://URN.fi/URN:NBN:fi:tty-201505081267"[Online; haettu 24.03.2019]. Tutkielma. TTY, 2015.

[11] S. Rautalin. Data-driven force models in GNSS satellite orbit prediction. "http://URN.fi/URN:NBN:fi:tty-201703211195"[Online; haettu 24.03.2019]. Tutkielma. TTY, 2017.

[12] J. Hartikainen, M. Seppänen ja S. Särkkä. *State-Space Inference for Non-Linear Latent Force Models with Application to Satellite Orbit Prediction*. "https://icml.cc/2012/papers/477.pdf". [Online; haettu 24.03.2019].

[13] K. Lehtonen. *TCSC - Tampere Center for Scientific Computing*. "https://wiki.eduuni.fi/display/tuttcsc/TCSC". [Online; haettu 05.01.2019].

[14] L. Chen, Q. Zhao, Z. Hu, X. Jiang, C. Geng, M. Ge ja C. Shi. *GNSS global real-time augmentation positioning: Real-time precise satellite clock estimation, prototype system construction and performance analysis*. eng. 2018.

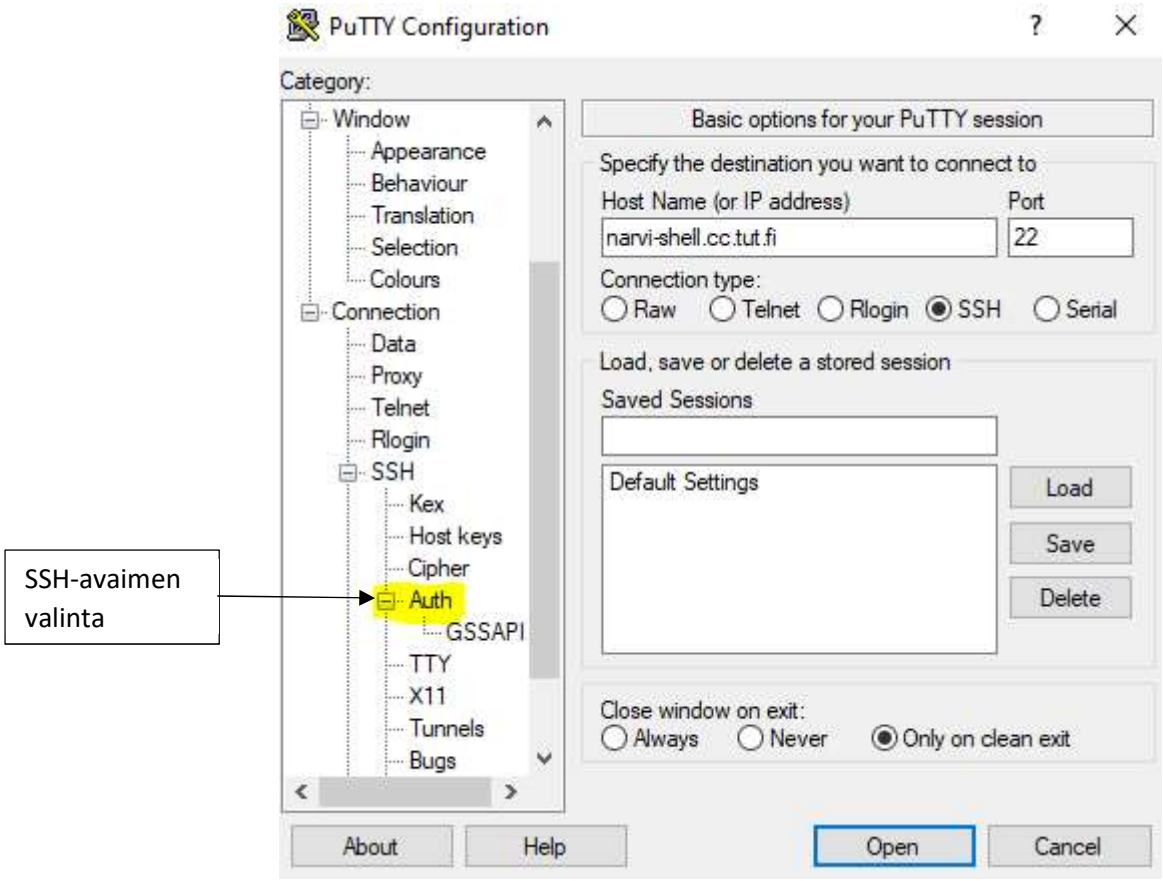
5 LIITTEET

Liite 1 Pikaopas Narvi-klusterin käyttöön

Pikaopas Narvi-klusterin käyttöön

Tämän liitteen tarkoituksena on toimia apuna käytettäessä Narvi-klusteria Windows käyttöjärjestelmällä. Ohje lähtee siitä oletuksesta, että käyttäjällä on tunnukset Narvi-klusteriin, lisätty SSH-avain tililleen ja asennettu PuTTY ohjelmisto.

1. Yhdistä klusteriin (Narvi: narvi-shell.cc.tut.fi) PuTTY työkalulla. Muista valita Auth valikosta tilillesi liitetty SSH avain yhdistäessäsi. Kysyttäessä käyttäjänimeä, kirjoita käyttäjänimesi, jolle tunnus on tehty



Kuva 1 PuTTYlla kirjautuminen

2. (vaihtoehtoinen) Vie koodi, jonka haluat suorittaa klusteriin avaamalla uusi komentotyökaluikkuna ja ajamalla komento:

pscp -i [SSH avaimen sijainti] [vietävän koodin sijainti] [käyttäjänimi]@narvi-shell.cc.tut.fi:[sijainti palvelimella]

   I. Ensimmäisellä kerralla voi joutua lisäämään ympäristömuuttujiin pscp-komennon sijainti. Sen lisäämiseen löytyy ohjeet täältä: http://the.earth.li/~sgtatham/putty/0.58/htmldoc/Chapter5.html#pscp

3. Tee TCSC wikin ohjeiden mukainen .sh tiedosto, joka ajetaan Slurm ohjelmistolla https://wiki.eduuni.fi/display/tuttcsc/Slurm

```
#!/bin/bash
#
# At first let's give job some descriptive name to distinct from other jobs running on cluster
#SBATCH -J Matin-kandi-testi
#
# Let's redirect job's out some other file than default slurm-%jobid-out
#SBATCH --output=slurm.out
#SBATCH --error=slurm.err
#
# We'll want mail notifications from job errors and ending
#SBATCH --mail-user=matti.valkeinen@tuni.fi
#SBATCH --mail-type=END
#
# We'll want to allocate just one cpu from some node for this job
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#
# We'll want to reserve 2GB memory for job (per node), and
# we estimate that job should finish in maximum of 8 hours and 15 minutes.
#SBATCH --time=0:00:20
#SBATCH --mem=2048


# Then actual program, commands e.g.


module load matlab
matlab -r "cd /home/valkeinm; try, run ('/home/valkeinm/testi_narvi.m'); end; quit"
```

Kuva 2 Esimerkki .sh tiedostosta

4. Lisää palvelimella oleva .sh tiedosto Slurm jonoon. Voit viedä .sh tiedoston kohdan 2 avulla lokaalilta koneelta palvelimelle
    I. Jonoon lisääminen tapahtuu komennolla: *sbatch [tiedoston nimi].sh*
5. Mikäli olet .sh tiedostossa määritellyt sähköpostin, niin sähköpostiin tulee ilmoitus ajon loppumisesta ja sen onnistumisesta
    I. Tulokset voidaan siirtää kohdan 2 avulla vaihtaen sijainnit keskenään