Panu Nummelin

Puhujariippuvainen puhekomentojentunnistus neuroverkoilla

Tietotekniikan pro gradu -tutkielma

4. kesäkuuta 2021

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Panu Nummelin

Yhteystiedot: panu.p.p.nummelin@student.jyu.fi

Ohjaajat: Ilkka Pölönen ja Jussi Hakanen

Työn nimi: Puhujariippuvainen puhekomentojentunnistus neuroverkoilla

Title in English: Speaker-dependent speech command recognition using neural networks

Työ: Pro gradu -tutkielma

Opintosuunta: Teknis-matemaattinen mallintaminen ja päätösanalytiikka

Sivumäärä: 54+0

Tiivistelmä: Tässä tutkimuksessa etsittiin puhekomennontunnistusmallia, joka voitaisiin kouluttaa pienellä määrällä äänitteitä tunnistamaan muutamia ennalta määrättyjä tietyn henkilön komentoja. Kolmea puhujariippuvaisella datalla koulutettua neuroverkkomallia vertailtiin muun muassa tunnistustarkkuuden ja tunnistusnopeuden suhteen. Tutkimuksessa parhaitenkin suoriutunut malli todettiin liian epäluotettavaksi käytännön käyttöön. Mahdollisiksi tavoiksi parantaa mallia esitettiin muuksi kuin komennoiksi luokiteltavan äänidatan ottaminen osaksi koulutusta ja data-augmentaatio.

Avainsanat: puheentunnistus, avainsanan tunnistus, neuroverkko

Abstract: In this study a speech command recognition model that could be trained with small amount of data to recognize a few predefined commands spoken by a specific person was sought. Three neural network models trained with speaker-dependent data were compared by their recognition accuracy and inference speed among other metrics. Even the best performing model of the study was deemed to be unsuitable for practical application. Integrating non-command speech data into the training process and data-augmentation were brought up as possible ways to improve the model's performance.

Keywords: speech recognition, keyword recognition, neural network

Termiluettelo

CNN	Konvoluutioneuroverkko (convolutional neural network) on konvoluutio-operaatioita sisältävä neuroverkko.
GAP	Globaalien keskiarvojen yhtymä (global average pooling) on operaatio, jossa lasketaan sen syötteen tietyn ulottuvuuden mukaiset keskiarvot.
HMM	Markovin piilomalli (hidden Markov model) on tilastollinen malli, jossa pyritään päättelemään piilotilojen arvot havaittujen arvojen perusteella.
Mel-spektrogrammi	Mel-spektrogrammi on kuvaus Mel-taajuuteen muunnetusta äänisignaalista.
Mel-taajuusasteikko	Mel-taajuusasteikko on ihmisen kuuloon perustuva asteikko äänitaajuuksille.
MFCC	Mel-taajuuden cepstral-kertoimet (Mel-frequency cepstral coefficients) on epäkorreloitu kuvaus Mel-spektrogrammista.
MLP	Monikerroksinen perseptroniverkko (multilayer perceptron) on kolmesta tai useammasta neuronikerroksesta koostuva neuroverkko.
NN	Neuroverkko (neural network) on pienistä laskentayksiköistä (neuroneista) rakentuva matemaattinen malli.
MP	Maksimien yhtymä (max pooling) on operaatio, jossa syötteen suorakulmaisilta alueilta poimitaan suurimmat arvot.
ReLU	Rectified linear unit on aktivaatiofunktio, joka nollaa negatiiviset syötteet ja säilyttää nollaa suuremmat syötteet ennallaan.
Softmax	Softmax on aktivaatiofunktio, joka skaalaa syötevektorinsa arvot välille $[0, 1]$, niin että alkioiden summa on 1.

Kuviot

Taulukot

Sisällys

1 Johdanto

Suomalainen yritys pyrkii vähentämään hoitajien kotikäyntejä vanhusten ja kuntoutettavien koteihin kehittämällä robotin, joka hoitaisi osan hoitajien suorittamista toimenpiteistä. Robotti voisi muun muassa muistuttaa vanhusta lääkkeiden otosta, peseytymisestä ja syömisestä sekä vaaratilanteiden, kuten kaatumisten, sattuessa se voisi hälyttää apua. Robotti myös helpottaisi vanhusten yhteydenpitoa hoitajiin ja omaisiin tarjoamalla mahdollisuuden videopuheluiden soittamiseen.

Robotin käyttämisestä halutaan tehdä mahdollisimman luontevaa myös vähemmän teknologiaa tunteville ihmisille, mikä aiotaan osaltaan saavuttaa tekemällä robotista puheohjattava. Yleispätevään puheentunnistukseen vaadittavan datamäärän kerääminen on työlästä ja kallista, joten olisi toivottavaa löytää puheentunnistusmalli, joka olisi koulutettavissa pienellä datamäärällä, jonka yksittäinen henkilö saisi pienellä vaivalla kerättyä, tunnistamaan muutamia datan luoneen henkilön suomenkielisiä komentoja. Robotti toimii akkuvirralla ja laskentatehoa on saatavilla rajallisesti, joten puheentunnistusratkaisun ajamisen tulisi kuluttaa mahdollisimman vähän resursseja. Kyky tunnistaa käheän kurkun, tunnetilan, unisuuden tai muun vastaavan seurauksena muuttunut puheääni on toivottua. Sopivaa ratkaisua olisi mahdollista hyödyntää muissakin sovelluskohteissa, joissa yhden henkilön olisi hyödyllistä pystyä ohjaamaan tietoteknistä laitetta muutamia puhekomentoja käyttäen. Tässä tutkimuksessa etsittiin edellä kuvatut kriteerit mahdollisimman hyvin täyttävää puheentunnistumallia. Kirjallisuushaun perusteella päädyttiin keskittymään neuroverkkopohjaisiin ratkaisuihin. Sopivaa mallia etsittiin vertailemalla kirjallisuudesta poimittuja malleja seuraavien kysymysten osalta:

- Kuinka tarkasti mallit tunnistavat puheen?
- Kuinka nopeasti mallit tunnistavat puheen?
- Kuinka laskennallisesti vaativia mallit ovat?
- Paljonko muistia mallit vaativat?
- Paljonko koulutusdataa mallit vaativat?

Luvussa 2 on esitelty aihealueen aiempaa tutkimusta. Luvussa 3 on esitelty neuroverkkojen toimintaperiaatteet ja puhekomentojen tunnistuksessa tarpeelliset piirteenirrotusmetodit. Kolme kirjallisuushaun pohjalta vertailtaviksi valittua neuroverkkoa ja muut tutkimusasetelman yksityiskohdat on esitetty luvussa 4. Vertailusta saadut tulokset on esitelty luvussa 5 ja tulosten pohjalta tehdyt päätelmät ja pohdinnat on esitetty luvussa 6.

2 Aiempi tutkimus

Automaattisessa puheentunnistuksessa pyritään tunnistamaan jatkuvaa puhetta luokittelemalla sanoja ja niiden eri muotoja tai sanojen osia, joista tunnistettavat sanat rakentuvat. Automaattisen puheentunnistuksen mallit rakentuvat usein erillisistä äänne- ja kielimalleista. Äännemalli tulkitsee sille syötetyn äänidatan sarjaksi kirjaimia ja sanojen välejä, jotka kielimalli pyrkii muuttamaan kielellisesti oikeellisempaan muotoon. (Anusuya ja Katti 2010, 182.) Nykyään yleistyvät myös neuroverkkoihin perustuvat ratkaisut, joissa ei ole erillisiä äänne- ja kielimalleja (Nassif ym. 2019, 19158). Automaattisen puheentunnistuksen mallit vaativat paljon laskentatehoa ja siten niiden ajaminen paikallisesti vähävirtaisissa laitteissa ei ole käytännöllistä. Vaihtoehtona on rajoittaa tunnistus muutamiin ennalta määrättyihin komentoihin, jolloin malleista voidaan tehdä yksinkertaisempia ja siten myös vähentää niiden laskennallista vaativuutta. (Sørensen, Epp ja May 2020, 1.) Tästä käytetään nimityksiä puhekomentojentunnistus (speech command recognition) tai puhuttujen avainsanojen tunnistus (spoken keyword spotting/recognition) (Majumdar ja Ginsburg 2020; Sørensen, Epp ja May 2020).

Puheentunnistus voidaan edelleen jakaa puhujariippuvaiseen ja puhujariippumattomaan tunnistukseen (Anusuya ja Katti 2010, 183). Tässä tutkimuksessa puhujariippuvaisella komennontunnistuksella tarkoitetaan yhden henkilön puheeseen mukautettua puhekomentojentunnistusta. Muiden puhujien puheen tunnistumiseen samalla mallilla ei oteta kantaa. Osassa muuta kirjallisuutta puhujariippuvaisella tunnistuksella saatetaan tarkoittaa tietyn puhujan tunnistusta samalla, kun tunnistetaan puhetta (Zhao ja Zhu 2017). Puhuja riippumattomalla tunnistuksella tarkoitetaan yleistä komentojen tunnistusta, jonka tavoitteena on tunnistaa kenen tahansa lausumat komennot. Puhujariippumattomassa tunnistuksessa pyritään tunnistamaan myös sellaisten henkilöiden puhe, joiden puhenäytteitä ei esiinny koulutusdatassa. Yleisiä puheentunnistukseen käytettäviä mallinnustapoja ovat neuroverkot (neural network, NN), Markovin piilomallit (hidden Markov model, HMM), tukivektorikoneet (support vector machine, SVM) ja dynamic time warping (DTW) (Anusuya ja Katti 2010, 184-187).

Puhekomennontunnistuksen kirjallisuutta haettiin ACM Digital Librarystä, IEEE Xploresta, SpringerLinkistä ja ScienceDirectistä hakulausekkeella *"keyword spot*" OR "keyword*

recog" OR "isolated word recog*" OR "speech command"* keskittyen vuoden 2019 ja uudempiin julkaisuihin. ScienceDirectin hausta jätettiin *-merkit pois, sillä niitä ei tuettu. Tuloksista valikoitiin sellaisia tutkimuksia, jotka keskittyivät puhuttujen avainsanojen tunnistukseen esimerkiksi esittelemällä mallin, jolla avainsanoja voidaan tunnistaa. Mallit, jotka käyttivät automaattista puheentunnistusta osanaan, jätettiin vähemmälle huomiolle niiden suurempien suorituskykyvaatimusten takia. Lisäksi puhujariippuvaiseen komennontunnistukseen keskittyvää kirjallisuutta haettiin Google Scholarista yhdistelemällä hakusanoja *speaker dependent*, *single speaker*, *speech recognition*, *keyword spot**, *keyword recog** ja *speech command*. Lähteitä etsittiin myös löydettyjen julkaisujen lähdeluetteloiden avulla ja hakemalla tutkimuksia, joissa viitataan löydettyihin lähteisiin.

2.1 Puhujariippuvainen puhekomentojentunnistus

Myers, Rabiner ja Rosenberg (1980), Fontaine ja Bourlard (1997) sekä Lopez-Meyer ym. (2015) tutkivat DTW-pohjaisia ratkaisuja puhujariippuvaisessa tapauksessa. Lisäksi Fontaine ja Bourlard (1997) sekä Lopez-Meyer ym. (2015) vertasivat DTW-ratkaisujaan neuroverkkoihin pohjautuviin malleihin. Lopez-Meyer ym. (2015) käyttivät neuroverkkorakenteenaan monikerroksista perseptroniverkkoa (multilayer perceptron, MLP), jollaista tutkivat myös Medhi ja Talukdar (2015). Toisin kuin Lopez-Meyer ym. (2015), joiden data oli englanninkielistä, Medhi ja Talukdar (2015) käyttivät assaminkielistä dataa.

Myers, Rabiner ja Rosenberg (1980) käyttivät kahden puhujan puhujariippuvaista dataa, joka koostui 39 eri sanasta, joista jokaisesta oli seitsemän näytettä. Jokaista yksittäisen puhujan sanaa kohden käytettiin kahta näytettä DTW:n koulutukseen ja viittä testaukseen. Lisäksi malli opetettiin tunnistamaan 54 eri teknologia-aiheista sanaa, joiden koulutusnäytteet saatiin klusterointianalyysin avulla 100 puhujan aineistosta, jossa jokainen puhuja oli toistanut jokaisen sanan kerran. Kaikki opetetut sanat olivat englanninkielisiä. Koulutukseen käytettiin kaksi näytettä jokaista sanaa kohden. Sanoja testattiin neljän puhujan toistamilla vastaavilla sanoilla. Parhaimmillaan tutkimuksessa päädyttiin keskimäärin n. 95 % tunnistustarkkuuteen kaikkien sanojen osalta.

Fontainen ja Bourlardin (1997) tutkimuksen kohdealueena oli puhelinnumeron valinta pu-

hekomennon avulla. Tutkimuksen malleissa ideana oli, että tunnistettavat opetuskomennot muunnetaan yleisiä äänteitä luokittelevalla mallilla sarjoiksi äänteitä kuvaavia luokkatunnuksia, jotka tallennetaan. Kun malliin syötetään äänidataa, josta komentoja halutaan etsiä, ääni muunnetaan sarjoiksi luokkatunnuksia opetusdatan tavoin, jonka jälkeen saatua luokkatunnussarjaa verrataan tallennettuihin sarjoihin, joista lähin vastaava palautetaan tunnistuksen tuloksena. Toteutustapa teki uusien komentojen opettamisesta helppoa. DTW-mallissa äänteet luokiteltiin k:n keskiarvon klusterointimenetelmällä ja luokkasarjoja vertailtiin DTW:tä käyttäen. HMM-NN-mallissa äänteet luokiteltiin neuroverkon avulla ja luokkasarjoja vertailtiin euklidista etäisyyttä käyttäen. Luokittelumallit koulutettiin käyttäen TIMIT-tietoaineistoa, joka sisältää englanninkielistä puhedataa useilta puhujilta, ja tunnistettavat komennot opetettiin BDSONS-tietoaineistoa käyttäen. BDSONS-tietoaineisto sisältää ranskankieliset numerot nollasta yhdeksään 23 eri puhujan lausumana. Jokaiselta puhujalta oli 40 näytettä jokaista numeroa kohden. Opetukseen käytettiin jokaista puhujaa kohden jokaisesta numerosta kaksi näytettä. Loppuja näytteitä käytettiin testaukseen. DTW-mallit saavuttivat 97,3-98,1 % tunnistustarkkuuden riippuen mallinäytteiden pakkauksen tasosta yhden puhujan tapauksessa. Vastaavasti HMM-NN-mallit saavuttivat 98,6-99,2 % tarkkuuden.

Lopez-Meyer ym. (2015) käyttivät data-aineistoa, joka koostui 1210 äänitteestä, jotka jakautuivat tasaisesti 11 eri avainsanan välille. Puhujia oli 11, joista jokainen lausui kunkin avainsanan 10 kertaa. Näytteitä käytettiin sellaisenaan ja niistä luotiin melullisia variantteja lisäämällä niihin autossa, puistossa tai pubissa äänitettyä ääntä 15 dB hiljaisempana kuin meluttoman äänitteen melutaso. Sekä NN:stä että DTW:stä toteutettiin puhujariippuvainen ja riippumaton versio. Puhujariippuvaisen DTW-mallin koulutukseen käytettiin kolmea yhden puhujan näytettä jokaista avainsanaa kohden ja NN-mallin koulutukseen käytettiin seitsemää yhden puhujan näytettä jokaista avainsanaa kohden. Loppuja näytteitä käytettiin testaukseen. Melusta huolimatta NN-mallit suoriutuivat hyvin tarjoten yli 95 % tarkkuuden joka tilanteessa. Puhuja riippuvainen DTW-malli tarjosi yli 96 % tarkkuuden pubimelua lukuun ottamatta, jolloin se saavutti vain 88 % tarkkuuden, mutta puhujariippumaton DTW-malli suoriutui merkittävästi huonommin tunnistaen parhaimmillaankin ilman melua vain noin 68 % testidatasta oikein.

Medhi ja Talukdar (2015) käyttivät 40 000 näytteestä koostuvaa data-aineistoa, joka sisälsi

20:n eri henkilön lausumana 100 eri sanaa, joista jokainen oli lausuttu 20 kertaa henkilöä kohden. Ratkaisu saavutti noin 99 % tunnistustarkkuuden puhujariippuvaisessa tapauksessa, kun MLP oli koulutettu yksittäisen puhujan datalla tunnistamaan viisi eri komentoa. Koulutukseen käytettiin 12 näytettä ja testaukseen käytettiin 8 näytettä jokaista sanaa kohden. Puhujariippumattomassa tapauksessa malli opetettiin tunnistamaan viisi sanaa käyttäen koulutukseen 10 puhujan kaikkia näytteitä kyseisten sanojen osalta. Testaukseen käytettiin loppujen 10 puhujan kaikkia näytteitä samojen sanojen osalta. Puhujarippumattomassa tapauksessa tunnistustarkkuudeksi saatiin noin 93 %.

2.2 Puhujariippumaton puhekomentojentunnistus

Puhekomentojentunnistuksen tutkimus on viime vuosina keskittynyt puhujariippumattomaan tunnistukseen. Käytettyjä neuroverkkomalleja ovat muun muassa konvoluutioneuroverkot (convolutional neural network, CNN), takaisinkytkeytyvät neuroverkot (recurrent neural network, RNN) ja aikaviiveneuroverkot (time delay neural network, TDNN). Malleista on pyritty tekemään vähän laskentatehoa vaativia, sillä puhekomennontunnistimia halutaan käyttää päätelaitteissa, joissa ei ole paljon laskentatehoa tai jotka toimivat akkuvirralla.

Syvyyskohtaisesti eroteltavia konvoluutioita (depthwise separable convolution, DSC) käyttivät Gu ym. (2020), Lu, Shan ja Xu (2019), Majumdar ja Ginsburg (2020), Mittermaier ym. (2020), Anh ym. (2021) sekä Sørensen, Epp ja May (2020). Tavallisia konvoluutioita käyttivät Chen ym. (2019). Yang ym. (2020) ja Wei ym. (2020) tutkivat CNN-RNN hybridimalleja. TDNN-RNN hybridimallia kehittivät Chai ym. (2019). Pervaiz ym. (2020) vertailivat RNN, DNN ja CNN mallien suorituskykyä melulla augmentoidun datan tapauksessa. Malleista pyrittiin tekemään laskennallisesti kevyitä käyttämällä vähäparametrisia rakenteita (Majumdar ja Ginsburg 2020; Mittermaier ym. 2020) tai korvaamalla osa neuroverkon laskutoimituksista laskennallisesti kevyemmillä approksimaatioilla (Lu, Shan ja Xu 2019; Sørensen, Epp ja May 2020). Mallien melusietoisuutta pyrittiin parantamaan lisäämällä koulutusdataan melua (Majumdar ja Ginsburg 2020; Sørensen, Epp ja May 2020) tai lisäämällä malliin äänen esiprosessointiin tarkoitettuja koulutettavia kerroksia (Gu ym. 2020; Mittermaier ym. 2020; Chen ym. 2019). Kaikki paitsi Chai ym. (2019) käyttivät Google Speech Commands -data-aineistoa neuroverkkojensa kouluttamiseen.

Konvoluutiot ovat laskennallisesti tehokas tapa käsitellä ruudukkomaista dataa, joten ne soveltuvat hyvin lyhyiden äänten luokitteluun (Goodfellow, Bengio ja Courville 2016, 326). Syvyyskohtaisesti eroteltavat konvoluutiot vaativat perinteisiä konvoluutioita vähemmän laskutoimituksia, kuitenkin tarjoten lähes vastaavia tuloksia kuin tavalliset konvoluutiot puhekomentojentunnistuksessa, mikä tekee niistä suosittuja rakennuspalikoita laskennallisesti tehokkaita malleja toteutettaessa (Sørensen, Epp ja May 2020, 1-2). RNN ja sen variaatiot ovat laskennallisesti vaativampia kuin konvoluutiot, mutta pystyvät huomioimaan paremmin ajalliset piirteet datassa (Yang ym. 2020, 81469). Käytetyistä neuroverkkorakenteista huolimatta parhaat mallit saavuttivat noin 96-98 % tunnistustarkkuuden puhujariippumattomassa tunnistuksessa meluttomalla datalla. Meluisan datan tapauksessa tunnistustarkkuus voi olla useita prosenttiyksikköjä heikompi riippuen käytetystä melusta.

Jos sopivaa koulutusdataa ei ole paljon saatavilla, on mahdollista hyödyntää muulla datalla koulutettua neuroverkkoa käyttämällä osaa sen koulutetuista kerroksista. Valmiiksi koulutettu neuroverkko on oppinut erottelemaan datasta joitain piirteitä, jotka voidaan syöttää toiseen neuroverkkoon, joka koulutetaan uuden käyttökohteen datalla. Vaihtoehtoisesti valmiiksi koulutetusta neuroverkosta voidaan jäädyttää uudelleenkäytettävät kerrokset, jonka jälkeen loput kerrokset uudelleenkoulutetaan uuden kohteen datalla. Tällaista oppimisen siirtoa (transfer learning) puhekomentojentunnistukseen sovelsivat Karunanayake, Thayasivam ja Ranathunga (2019) käyttämällä englantia tunnistavaa mallia pohjana sinhalan- ja tamilinkielisten puhekomentojen tunnistusta varten. Uudelleenkäytettävän mallin ei tarvitse olla koulutettu juuri puheentunnistukseen, kuten McMahan ja Rao (2017) osoittivat hyödyntämällä ympäristön ääniä tunnistavan mallin piirteitä Google Speech Commands -dataaineiston tunnistustarkkuuden parantamiseksi.

Neuroverkot ovat kehittyneet nopeasti viime vuosina, mutta puhekomentojentunnistuksen tutkimus on keskittynyt puhujariippumattomaan tunnistukseen, joten tässä tutkimuksessa tarkastellaan viimeaikaisten neuroverkkorakenteiden soveltuvuutta puhujariippuvaiseen tunnistukseen. Vertailtaviksi malleiksi valittiin Majumdarin ja Ginsburgin (2020) kehittämä MatchboxNet sekä Anhin ym. (2021) kehittämä LIS-Net niiden Google Speech Commands -dataaineistolla useidenkin komentojen tapauksessa saavuttamien korkeiden tunnistustarkkuuksien takia. Vertailuun otettiin mukaan myös Lopezin ym. (2015) käyttämä neuroverkko tut-

kimusaiheen vastaavuuden perusteella. Mallien valinta tehtiin Google Speech Commands -data-aineistolla saatujen tulosten perusteella, sillä sitä oli käytetty monissa viimeaikaisissa puhekomentojentunnistusmalleja käsittelevissä tutkimuksissa.

3 Puhekomentojentunnistus neuroverkoilla

Tässä luvussa esitellään neuroverkkojen teoriaa yleisesti keskittyen vertailtaviksi valittujen neuroverkkorakenteiden kannalta keskeisiin periaatteisiin. Lisäksi esitellään tutkimuksessa käytetyt piirteenirrotusmetodit.

3.1 Neuroverkko

Neuroverkkojen ajatuksena on vastata kysymykseen pilkkomalla kysymys joukoksi pienempiä kysymyksiä, joiden vastaukset yhdistämällä saadaan vastaus alkuperäiseen kysymykseen. Neuroneista koostuvat ihmisten aivot toimivat saman tapaisesti, mistä neuroverkot saavatkin nimensä. Yksinkertaistettuna neuroverkot koostuvat toisiinsa yhdistetyistä laskentayksiköistä, neuroneista, jotka ottavat vastaan syötteitä ja antavat niiden pohjalta jonkin ulostulon, joka voidaan välittää toisille neuroneille. Neuroneita voidaan kasata kerroksiin, jolloin yksittäinen kerros saa syötteensä alemmalta kerrokselta ja välittää ulostulonsa ylemmälle kerrokselle. (Nielsen 2015, luku 1.) Tyypillinen neuroni k kerroksella l, jolla on N syötettä, on muotoa

$$a_k^{(l)} = f(\sum_{j=1}^{N} w_{kj}^{(l)} a_j^{(l-1)} + b_k^{(l)}), \tag{3.1}$$

jossa $a_k^{(l)}$ on kerroksen l neuronin k ulostulo, f on aktivaatiofunktio, $w_{kj}^{(l)}$ on paino syötteelle $a_j^{(l-1)}$ ja $b_k^{(l)}$ on harha (bias) (Nielsen 2015, luku 2). Neuroni laskee yhteen saamansa syötteet painottaen kutakin syötettä neuronille määritettyjen painojen mukaan, lisää summaan harhan ja syöttää sen aktivaatiofunktiolle, jonka tulos on neuronin ulostulo. Neuronin toimintaa voidaan mukauttaa muuttamalla painoja, harhaa ja aktivaatiofunktiota.

Neuroverkkojen painoille ja harhoille voidaan etsiä optimaaliset arvot tietyn ongelman ratkaisemiseksi määrittämällä kustannusfunktio, joka suhteuttaa neuroverkon ulostulon koulutusdatan oikeisiin arvoihin, ja optimoimalla se pienin askelin päivittäen askelten välissä neuroverkon painojen ja harhojen arvot. Optimointiin voidaan käyttää esimerkiksi gradienttimenetelmää (gradient descent), jonka periaatteena on liikkua pieni askel optimoitavan funktion negatiivisen gradientin suuntaan, jolloin funktion arvo laskee. Muita gradienttipohjaisia op-

timointialgoritmeja ovat muun muassa Adam ja NovoGrad (Kingma ja Ba 2017; Ginsburg ym. 2020). Gradienttimenetelmä ja muut vastaavat menetelmät vaativat gradientin laskemista ja neuroverkkojen tapauksessa se tapahtuu tehokkaasti vastavirta-algoritmia käyttäen.

Vastavirta-algoritmissa käytetään yhtälöitä

$$\boldsymbol{\delta}^{(L)} = \nabla_a C \odot f'(\boldsymbol{z}^{(L)}), \tag{3.2}$$

$$\boldsymbol{\delta}^{(l)} = ((\boldsymbol{w}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)}) \odot f'(\boldsymbol{z}^{(l)}), \tag{3.3}$$

$$\frac{\partial C}{\partial b_j^{(l)}} = \delta_j^{(l)}, \tag{3.4}$$

$$\frac{\partial C}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \delta_j^{(l)}, \qquad l = 1,2,...,L-1 \qquad \text{ja} \tag{3.5}$$

$$\boldsymbol{z}^{(l)} = \boldsymbol{w}^{(l)} \boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}, \tag{3.6}$$

joissa $\boldsymbol{\delta}^{(l)}$ on kerroksen l virhe, L on kerrosten lukumäärä, $\nabla_a C$ on kustannusfunktion C gradientti neuroverkon viimeisten ulostulojen suhteen ja $\odot$ on alkiokohtainen tulo (Nielsen 2015, luku 2). Muut merkinnät vastaavat yhtälön 3.1 merkintöjä. Yhtälö 3.6 kuvaa kerroksen l neuroneiden painotettua ja harhautettua syötettä. Vastavirta-algoritmin suoritus alkaa syöttämällä koulutusnäyte neuroverkon läpi. Samalla lasketaan kunkin kerroksen painotettu ja harhautettu syöte 3.6 sekä ulostulo $\boldsymbol{a}$. Seuraavaksi neuroverkko käydään läpi vastakkaiseen suuntaan liikkuen laskemalla aluksi viimeisen kerroksen virhe käyttäen yhtälöä 3.2. Kun viimeisen kerroksen virhe on tiedossa, voidaan siirtyä yksi kerros neuroverkon alkuun päin ja laskea saavutetun kerroksen virhe yhtälön 3.3 avulla. Tätä toistetaan, kunnes kaikkien kerrosten virheet on laskettu. Lopuksi kustannusfunktion gradientit harhojen ja painojen suhteen saadaan yhtälöiden 3.4 ja 3.5 mukaisesti. Laskettuja gradientteja voidaan käyttää gradienttipohjaisissa optimointimenetelmissä neuroverkon painojen ja harhojen päivittämiseen. Tiivistetysti vastavirta-algoritmi on seuraavanlainen:

1. Aseta syötekerroksen ulostulon $\boldsymbol{a}^{(1)}$ arvoksi neuroverkon syöte.
2. Laske kunkin kerroksen $l = 2,3,...,L$ painotettu ja harhautettu syöte 3.6 sekä ulostulo $\boldsymbol{a}^{(l)}$.
3. Laske viimeisen ulostulon virhe 3.2.
4. Laske aiempien kerrosten $l = L-1, L-2, ..., 2$ virheet 3.3.
5. Laske harhojen 3.4 ja painojen 3.5 gradientit.

3.1.1 Aktivaatiofunktio

Aktivaatiofunktion ulostulo kuvaa neuronin aktiivisuuden tasoa. Ulostulon voidaan myös ajatella kuvaavan kyseisen neuronin kuvaaman piirteen voimakkuutta. Käyttämällä epälineaarisia aktivaatiofunktioita, voidaan neuroverkoilla mallintaa epälineaarisiakin ilmiöitä (Goodfellow, Bengio ja Courville 2016, 165). Yleisesti käytettyjä aktivaatiofunktioita ovat muun muassa sigmoid, hyperbolinen tangentti (tanh), rectified linear unit (ReLU) ja softmax. Näistä tarkemmin esitellään ReLU ja softmax, jotka esiintyvät tutkimuksen neuroverkoissa.

ReLU-funktion määritelmä on

$$f(x) = max(0,x)\,. \tag{3.7}$$

ReLU on epälineaarinen funktio, joka palauttaa syötteensä sellaisenaan, jos syöte on positiivinen. Muussa tapauksessa se palauttaa nollan. Sigmoid- ja tanh-aktivaatiofunktioihin verrattuna ReLU on laskennallisesti halvempi ja se auttaa välttämään katoavan gradientin. ReLU:n heikkouksia ovat sen rajattomuus, joka voi johtaa numeerisiin ongelmiin, sekä mahdollisuus skaalata painoja ja harhoja muuttamatta neuroverkon toimintaa. (Glorot, Bordes ja Bengio 2011, 318-319.)

Softmax-funktion määritelmä on

$$\sigma(\boldsymbol{x})_i = \frac{\mathrm{e}^{x_i}}{\sum_{j=1}^{N_x} \mathrm{e}^{x_j}}\,, \tag{3.8}$$

jossa $\boldsymbol{x}$ on N_x:n alkion mittainen vektori. Softmax skaalaa syötteenä saamansa vektorin alkiot välille $[0,1]$ niin, että alkioiden summa on 1. Softmax-funktiolla kuvataan syötteen todennäköisyysjakauma N_x:n luokan osalta. (Goodfellow, Bengio ja Courville 2016, 180-181.) Jokaisessa tämän tutkimuksen neuroverkossa viimeisenä aktivaatiofunktiona on softmax todennäköisimmän puhekomennon luokan selvittämiseksi.

3.1.2 Kustannusfunktio

Kustannusfunktio kuvaa, kuinka hyvin neuroverkko arvioi syötteensä oikein. Tämän tutkimuksen neuroverkoissa käytetään ristientropia-kustannusfunktiota (cross-entropy). Ristientropia-

funktion määritelmä on

$$C = -\frac{1}{N_x}\sum_{i=1}^{N_x}\sum_{j=1}^{N_j}[y_j \log a_j^{(L)} + (1-y_j)\log(1-a_j^{(L)})], \tag{3.9}$$

jossa N_x on koulutusnäytteiden määrä, N_j on ulostuloneuroneiden määrä, y on toivottu ulostulo ja *log* on luonnollinen logaritmi (Nielsen 2015, luku 3). Ristientropia palauttaa nollan, kun $y_j = a_j^{(L)}$ kaikilla $j = 1,...,N_j$, eli neuroverkon arvioidessa kaikki koulutusnäytteet oikein on kustannus pienin mahdollinen. Muuten ristientropia palauttaa positiivisia lukuja, jotka ovat sitä suurempia mitä kauempana syötteet ovat toivotusta arvosta.

3.1.3 Katoava gradientti

Katoavalla gradientilla tarkoitetaan tilannetta, jossa kustannusfunktion osittaisderivaatat lähestyvät nollaa (Pascanu, Mikolov ja Bengio 2013, 2). Ongelma korostuu syvissä neuroverkoissa, sillä vastavirta-algoritmin mukaisesti syvemmän kerroksen gradientti lasketaan edellisen kerroksen gradienttien perusteella, jolloin pienentyminen on eksponentiaalista gradientin arvojen ollessa alle yhden. Neroverkko ei opi käytännössä lainkaan, jos gradientti muuttuu liian pieneksi.

Katoavan gradientin ongelmaa voidaan yrittää välttää käyttämällä ReLU aktivaatiofunktiota 3.7, jäännösoikoteitä (residual shortcut) tai eränormalisointia (batch normalization). Jäännösoikotiellä ohitetaan muutama neuroverkon kerros summaamalla aiempi syöte ohitettavien kerrosten ulostuloon (He ym. 2015, 2). Oikotien avulla kustannusfunktion derivaatta ei pääse kutistumaan yhtä nopeasti kuin ilman oikotietä. Eränormalisointikerros pyrkii muuntamaan syötteensä niin, että kerroksen ulostulojen keskiarvo on 0 ja keskihajonta 1 (Ioffe ja Szegedy 2015, 3). Eränormalisointikerrokset sisältävät datasta riippuvia parametreja, joten ne tulee kouluttaa osana muuta neuroverkkoa. Eränormalisointi heikentää gradientin riippuvuutta neuroverkon parametreistä ja niiden alkuarvoista, mikä auttaa gradienttia kantautumaan neuroverkon läpi (Ioffe ja Szegedy 2015, 2).

3.1.4 Monikerroksinen perseptroniverkko

Monikerroksinen perseptroniverkko (multiplayer perceptron) koostuu kolmesta tai useammasta kerroksesta, jotka rakentuvat aiemmin kuvatun mukaisista neuroneista 3.1. Ensim-

mäistä kerrosta kutsutaan syötekerrokseksi, viimeistä kerrosta ulostulokerrokseksi ja väliin jääviä kerroksi kutsutaan piilokerroksiksi (hidden layer). Monikerroksisissa perseptroniverkoissa esiintyvistä kerroksista, joissa jokainen neuroni on yhdistetty jokaiseen edeltävän tai seuraavan tason neuroniin, käytetään nimityksiä tiheä kerros (dense layer) ja täysin yhdistetty kerros (fully connected layer).

3.1.5 Konvoluutioneuroverkko

Konvoluutioneuroverkot käyttävät konvoluutioita vähintään yhdellä kerroksellaan. Konvoluutio on kahdelle funktiolle suoritettava lineaarinen operaatio, joka kuvaa, miten toinen funktio muokkaa toista. Konvoluutiot ovat hyödyllisiä käsiteltäessä ruudukkomaista dataa, kuten kaksiulotteisia kuvia tai aikasarjadataa, jonka voidaan ajatella koostuvan tasaisin väliajoin otetuista yksiulotteisista näytteistä. (Goodfellow, Bengio ja Courville 2016, 326-327.) Puhe voidaan kuvata edellä mainitun kaltaisena aikasarjadatana.

Tietokoneella käsiteltävä data on yleensä epäjatkuvaa tai se on järkevää muuntaa epäjatkuvaksi laskennallisista syistä, joten konvoluutio esitellään epäjatkuvien tilanteiden näkökulmasta. Epäjatkuvan konvoluution, joka ottaa vastaan yksiulotteisen syötteen I ja käyttää yksiulotteista ydintä K, määritelmä on:

$$S(i) = (I * K)(i) = \sum_{m=1}^{N_m} I(m)K(i-m)\,. \tag{3.10}$$

Jos syöte I ja ydin K ovat kaksiulotteisia, voidaan konvoluution määritelmä esittää yhtä pätevästi kahdella eri tavalla:

$$S(i,j) = (I * K)(i,j) = \sum_{m=1}^{N_m}\sum_{n=1}^{N_n} I(m,n)K(i-m,j-n)\,, \tag{3.11}$$

$$S(i,j) = (K * I)(i,j) = \sum_{m=1}^{N_m}\sum_{n=1}^{N_n} I(i-m,j-n)K(m,n)\,, \tag{3.12}$$

joissa N_m ja N_n ovat vakioita, jotka määrittävät ytimen koon. (Goodfellow, Bengio ja Courville 2016, 328.) Edeltävissä yhtälöissä konvoluution tulos lasketaan yhdelle pisteelle, mutta käytännössä konvoluutio halutaan usein laskea kaikkien syötteen, esimerkiksi kuvan, pisteiden suhteen, jolloin I on vain se osa kuvasta, jota kulloinkin ytimellä K käsitellään. Konvoluutio esitellään tarkemmin kaksiulotteisesta näkökulmasta, mutta samat periaatteet pätevät

myös yksiulotteisissa tapauksissa. Konvoluutiossa tarkastellaan vain osaa syötteestä kerrallaan, mikä mahdollistaa pienien ominaisuuksien löytämisen tehokkaasti. Kuitenkin suoritettaessa useita konvoluutioita peräkkäin ydintä suurempi joukko alkuperäisen syötteen datapisteistä vaikuttaa epäsuorasti yksittäiseen myöhemmän konvoluution datapisteeseen olettaen, että käytettävä ydin on alkuperäistä syötettä pienempi. Epäsuora vaikutus myöhempiin konvoluutioihin mahdollistaa monimutkaisempienkin yhteisvaikutusten mallintamisen. (Goodfellow, Bengio ja Courville 2016, 330-332.)

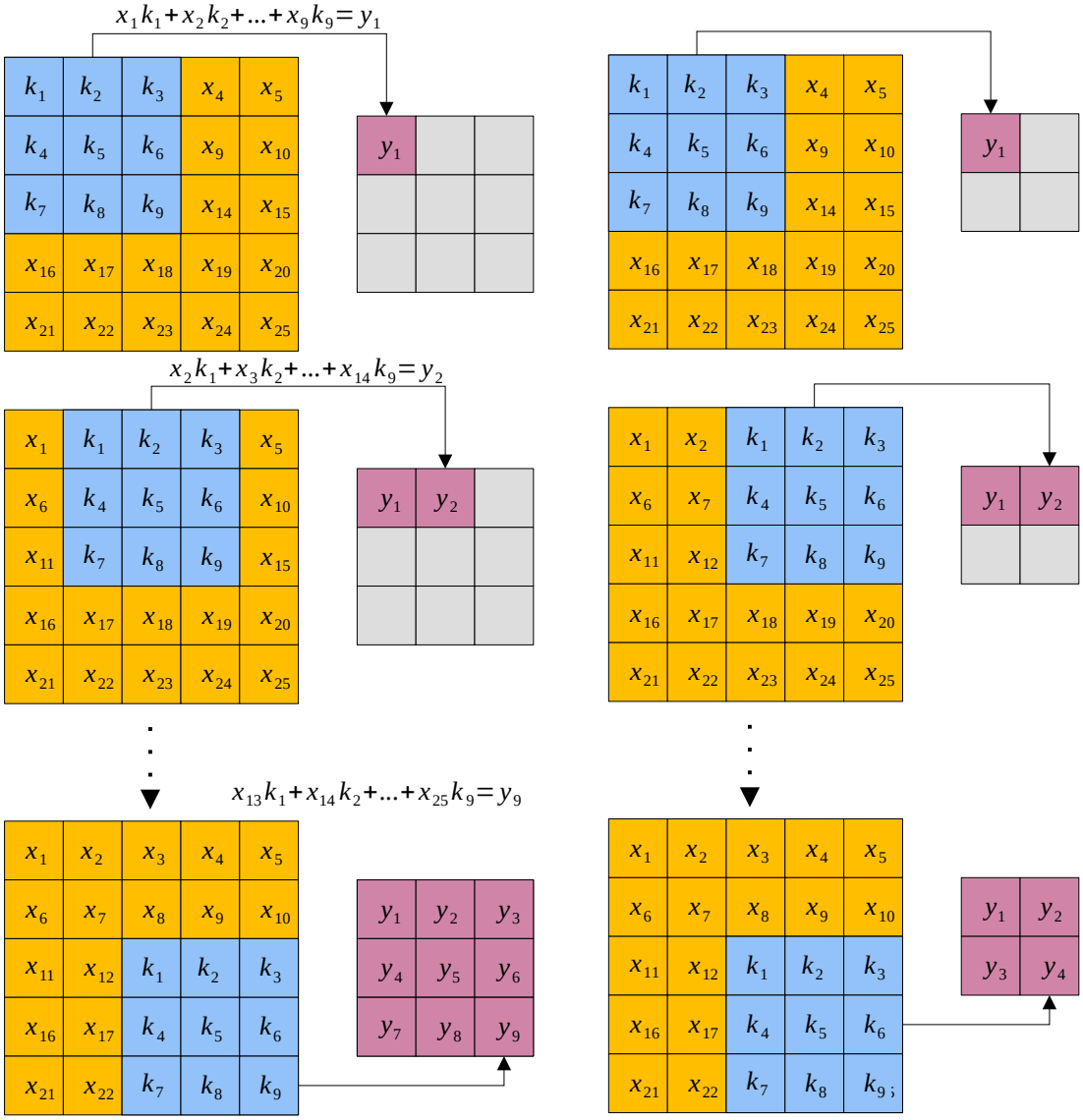
Konvoluutio voidaan suorittaa jokaisen syötteen pisteen suhteen tai jättää osa pisteistä välistä laskennan helpottamiseksi (Goodfellow, Bengio ja Courville 2016, 343). Askeleella (stride) määritetään, kuinka monta syötteen pistettä jää niiden pisteiden, joiden suhteen konvoluutiot lasketaan, välille. Askelpituudet voidaan määrittää ulottuvuuskohtaisesti. Esimerkiksi olkoot pisteet (i, j), $i \in 1,2,...,N_i$ ja $j \in 1,2,...,N_j$, kuvan P pisteitä, joiden suhteen voidaan P:stä poimia alue I, jonka oikean alanurkan piste vastaa P:n pistettä, jonka suhteen I poimittiin. Muuten I:n mitat ovat samat kuin konvoluution S ytimen K. Suoritettaessa konvoluutio S (3.11) kuvan P alueille askelpituuden ollessa $a \in \mathbb{N}$ kummankin ulottuvuuden suhteen lasketaan konvoluutio S vain P:n pisteiden (i_a, j_a), $i_a \in 1, 1+a, 1+2a, ..., N_i$ ja $j_a \in 1, 1+a, 1+2a, ..., N_j$ suhteen pomituille alueille. Käytettäessä suurempia askelia pienenee askellettavaa ulottuvuutta vastaava ulottuvuus ulostulossa suhteessa käytettyyn askelpituuteen, koska muunnokset lasketaan vain osalle syötteen pisteistä. Askelpituuden vaikutusta on havainnollistettu kuviossa 1.

Konvoluutiossa dilaatiolla määritetään ytimen tarkastelemien pisteiden etäisyys toisistaan (Yu ja Koltun 2016, 2-3). Otettaessa dilaatio $d \in \mathbb{N}$ huomioon yhtälö 3.11 voidaan kirjoittaa muotoon

$$S(i,j) = (I * K)(i,j) = \sum_{m=1}^{N_m} \sum_{n=1}^{N_n} [\\ I(1+((m-1)d), 1+((n-1)d)) K(i-1-((m-1)d)), j-1-((n-1)d))] . \tag{3.13}$$

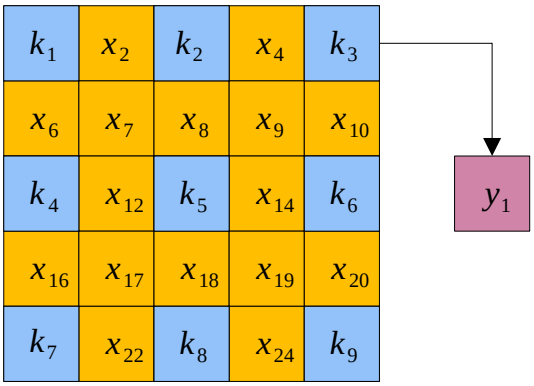
Dilaatiota on havainnollistettu kuviossa 2.

Syötteen reunoja voidaan täydentää (padding) nollilla, jotta ulostulon koko ei kutistuisi syötteeseen nähden (Goodfellow, Bengio ja Courville 2016, 343). Ilman nollatäydennystä ydin käy läpi sellaiset kohdat, joissa kaikki ytimen pisteet peittävät jonkin syötteen pisteen, jol-



(a) Kaksiulotteinen konvoluutio (b) Kaksiulotteinen konvoluutio askelpituudella 2

Kuvio 1: Tapauksessa 1a kaksiulotteinen konvoluutio suoritetaan $5 \times 5 \times 1$ syötteelle käyttäen $3 \times 3 \times 1$ ydintä. Tuloksena on $3 \times 3 \times 1$ ulostulo. Syötteen arvot on merkitty muuttujalla x, ytimen arvot muuttujalla k ja ulostulon arvot muuttujalla y. Askelpituus ja dilaatio ovat 1. Tapaus 1b on kuten 1a, mutta askelpituutena on 2, jolloin ulostulon mitat ovat $2 \times 2 \times 1$.

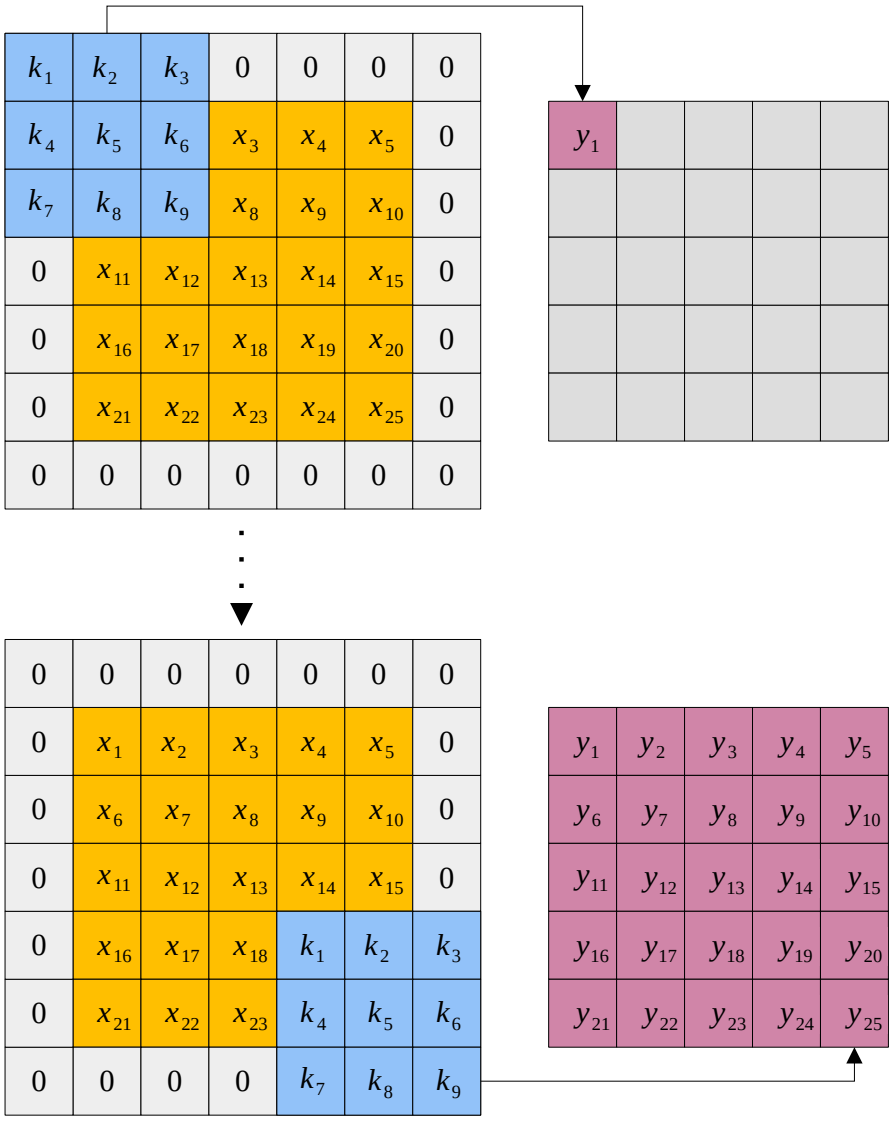


Kuvio 2: Tapaus on vastaava kuin kuviossa 1a, mutta dilaationa on 2, jolloin ulostulon mitat ovat $1 \times 1 \times 1$.

loin ulostulon ydintä vastaavat ulottuvuudet kutistuvat $N_x - 1$ verran, jossa N_x on ytimen syötteen ulottuvuutta x vastaavan ytimen ulottuvuuden pituus. Yleensä nollia lisätään sen verran, että ydintä vastaavat ulottuvuudet syötteen ja ulostulon osalta pysyvät yhtä suurina. Nollatäydennystä on havainnollistettu kuviossa 3.

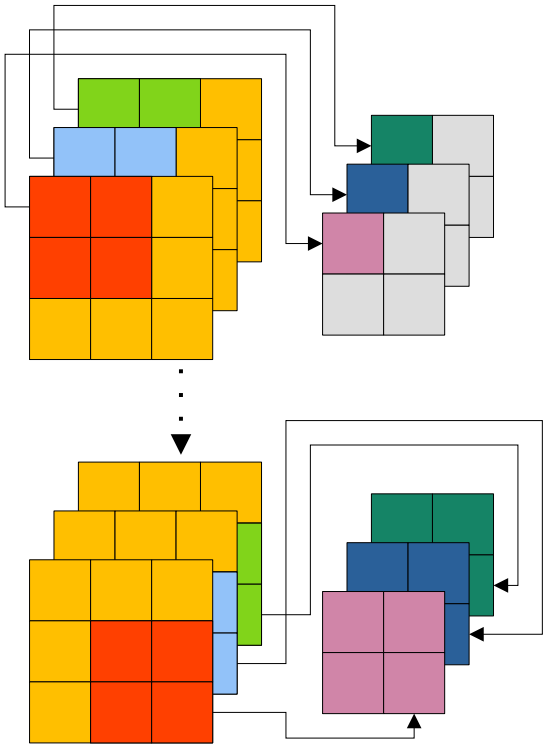
Yksittäisellä neuroverkon konvoluutiokerroksella voi olla useampi ydin, jolloin kaikkien ytimien tulokset välitetään seuraavalle kerrokselle. Lisäksi kaksiulotteiselle konvoluutiolle voidaan syöttää useampiulotteista dataa, mutta tällöin konvoluution ytimet tulee laajentaa kattamaan ylimääräiset ulottuvuudet. Esimerkiksi syötettäessä $3 \times 3 \times 3$ kuva kaksiulotteiselle konvoluutiolle, joka sisältää neljä $2 \times 2 \times 3$ ydintä ja täyttää syötteen reunat nollilla, niin että ulostulon ensimmäiset kaksi ulottuvuutta ovat samat kuin syötteen, on lopputuloksena $3 \times 3 \times 4$ kuva. Kaksiulotteisen konvoluution tapauksessa ydin liikkuu vain kahden ulottuvuuden mukaan, mutta on mahdollista toteuttaa myös korkeamman ulottuvuuden konvoluutioita. Tämän tutkimuksen neuroverkoissa esiintyy yksi- ja kaksiulotteisia konvoluutioita.

Konvoluution erikoistapauksia ovat syvyyskohtainen konvoluutio (depthwise convolution) ja pistekohtainen konvoluutio (pointwise convolution). Syvyyskohtaisessa konvoluutiossa käytetään eri ydintä jokaista syötteen kanavaa kohden. Esimerkiksi edellisen esimerkin tapauksessa, jos käytössä olisikin syvyyskohtainen konvoluutio, olisi $2 \times 2 \times 1$ muotoisia ytimiä kolme, joista kukin käy $3 \times 3 \times 1$ kuvan läpi, ja lopputuloksena olisi $3 \times 3 \times 3$ kuva. Edellä esitetty tapaus on kuvattu kuviossa 4. Pistekohtaisesta konvoluutiosta puhutaan, kun syöt-

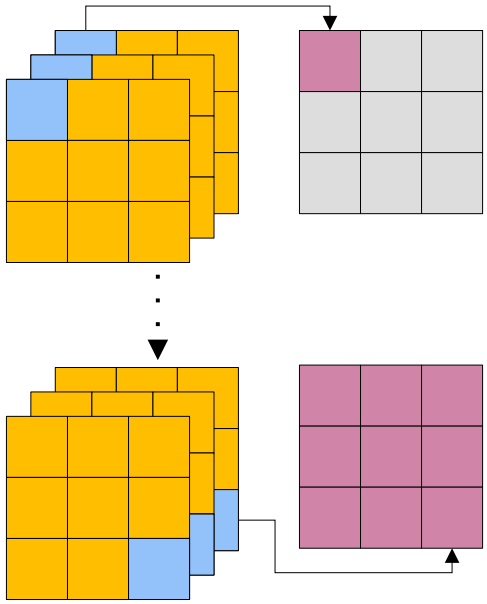


Kuvio 3: Tapaus on vastaava kuin kuviossa 1a, mutta syötteen reunoja on täydennetty nollilla, niin että ulostulo säilyttää täydentämättömän syötteen mitat.

teessä on yksi ulottuvuus enemmän kuin konvoluutiossa ja käytössä on ydin, jonka konvoluution kattamien ulottuvuuksien pituus on 1 ja muuten vastaavat kuin syötteessä. Edellisen esimerkin syötteen tapauksessa ydin olisi muodoltaan $1 \times 1 \times 3$ pistekohtaista konvoluutiota käytettäessä. Pistekohtaista konvoluutiota on havainnollistettu kuviossa 5.



Kuvio 4: Syvyyskohtainen konvoluutio, jossa kolme erillistä $2 \times 2 \times 1$ ydintä kukin käy läpi yhden kanavan $3 \times 3 \times 3$ syötteestä, jolloin lopputuloksena on $2 \times 2 \times 3$ ulostulo



Kuvio 5: Pistekohtainen konvoluutio, jossa $1 \times 1 \times 3$ ydin käy läpi $3 \times 3 \times 3$ syötteen muodostaen $3 \times 3 \times 1$ ulostulon

3.1.6 Yhtymä

Yhtymällä (pool) tarkoitetaan operaatiota, jossa syötettä yksinkertaistetaan jakamalla syöte pistejoukkoihin ja korvaamalla muodostetut joukot niistä lasketuilla tilastollisilla tunnusluvuilla. Yhtymän ulostulossa on siis vähemmän pisteitä kuin syötteessä, joten yhtymiä voidaan käyttää laskennan helpottamiseksi niitä seuraavilla kerroksilla. Yhtymillä voidaan myös

tehdä neuroverkosta riippumaton pienistä syötteen muunnoksista, jolloin neuroverkko yleistyy paremmin muulle datalle. (Goodfellow, Bengio ja Courville 2016, 335-339.) Tämän tutkimuksen neuroverkoissa käytetään globaalien keskiarvojen yhtymää (global average pooling, GAP) ja maksimien yhtymää (max pooling, MP).

Globaalien keskiarvojen yhtymä on suunniteltu korvaamaan konvoluutioneuroverkon viimeinen tiheä kerros. Globaalien keskiarvojen yhtymässä ulostulo muodostuu syötteen jokaisesta kanavasta otetusta keskiarvosta. Jos kanavia on yhtä monta kuin tunnistettavia luokkia, voidaan globaalien keskiarvojen yhtymää käyttää tiheän kerroksen korvikkeena. (Lin, Chen ja Yan 2014, 4.) Maksimien yhtymässä ulostulo muodostuu syötteen suorakulmaisilta alueilta poimituista suurimmista arvoista (Goodfellow, Bengio ja Courville 2016, 335).

3.1.7 Pudotus

Pudotuksessa (dropout) satunnainen osa neuroverkon kerroksen neuroneita jätetään huomioimatta. Pudotettavat neuronit arvotaan uudelleen kunkin koulutusnäytteen kohdalla. Vain pudottamattomien neuroneiden koulutettavat parametrit päivitetään osana vastavirta-algoritmia. Pudotuksen tavoitteena on parantaa neuroverkon yleistyvyyttä välttämällä ylioppiminen. (Srivastava ym. 2014, 1930-1931.)

3.2 Piirteenirrotus

Piirteenirrotuksella (feature extraction) pyritään tutkittava data muuntamaan muotoon, joka sisältää vain datan käyttötarkoituksen kannalta olennaiset piirteet datasta. Tarkoituksena on luoda yksinkertaistettu kuvaus datasta, joka säilyttää datan ilmaisuvoiman ollen samalla laskennallisesti helpommin käsiteltävissä. Puheentunnistuksessa yleisimmin käytetty piirteenirrotusmenetelmä on mel-taajuuden cepstral-kertoimien laskeminen (mel-frequency cepstral coefficients, MFCC) syöteäänisignaalista. Muita tapoja ovat muun muassa lineaarinen erotteluanalyysi (linear discriminant analysis), linear predictive coding ja perceptual linear predictive. (Nassif ym. 2019, 19155.) Tässä tutkimuksessa käytetään neuroverkkojen syötteinä MFCC-piirteitä ja mel-spektrogrammeja, joiden laskeminen on osa MFCC-muunnosprosessia.

Analoginen ääni muunnetaan digitaaliseksi mittaamalla analogisen signaalin amplitudi säännöllisin väliajoin ja tallentamalla havaittu amplitudi kokonaislukuna. Tietyn taajuuksinen signaali voidaan muuntaa häviöttömästi digitaaliseksi käyttämällä analogisen signaalin taajuuteen verrattuna kaksinkertaista näytteenottotaajuutta, mistä käytetään nimitystä Nyquistin teoreema (Jurafsky ja Martin 2020, 552). Näytteenottotaajuus tulee valita niin, että äänisignaalista saadaan talteen käyttötarkoituksen kannalta tärkeät taajuudet. Puheentunnistuksen tapauksessa liian suuri näytteenottotaajuus lisää laskennallista vaativuutta tunnistustarkkuuden pysyessä ennallaan. Vastaavasti liian pieni näytteenottotaajuus ei kata kaikkia puheen kannalta oleellisia taajuuksia, jolloin tunnistustarkkuus kärsii. Ihmisten puheen taajuus on yleensä alle 10 000 Hz, joten puheentunnistuksessa käytetään yleisesti 16 000 Hz näytteenottotaajuutta, sillä se kattaa tarpeeksi hyvin puheen merkittävimmät piirteet (Jurafsky ja Martin 2020, 552). Amplitudi kuvataan yleensä 16 bittisten kokonaislukujen avulla. Käytettyjen kokonaislukujen skaala määrää sen, kuinka pieniä amplitudieroja pystytään kuvaamaan (Jurafsky ja Martin 2020, 552).

Äänen esittämiseksi ihmisen kannalta merkityksellisemmässä muodossa käytetään mel-skaalaa, joka kuvaa äänitaajuudet ihmisten havaitsemien taajuuserojen suhteen. MFCC-piirteet ovat epäkorreloitu kuvaus mel-skaalaan sovitetusta äänisignaalista. (Bäckström ym. 2019, luku 2.f.) MFCC-piirteiden laskeminen äänisignaalista sisältää seuraavat vaiheet:

1. Signaalin jakaminen lyhyihin ikkunoihin.
2. Ikkunoiden magnitudispektrien laskeminen diskreetillä Fourier-muunnoksella (discrete Fourier transform, DFT).
3. Mel-spektrogrammien muodostus muuntamalla magnitudispektrit mel-skaalaan.
4. Mel-taajuuksisten spektrien muuntaminen logaritmiskaalaan.
5. Mel-taajuuksisten spektrien cepstral-kertoimien laskeminen diskreetillä kosinimuunnoksella (discrete cosine transform, DCT).

Ensimmäisessä vaiheessa äänisignaali jaetaan limittäisiin ikkunoihin. Ikkunan tulee olla tarpeeksi pitkä, jotta sen rajaamasta signaalin osasta laskettava magnitudispektri on tarkka, ja samalla riittävän lyhyt ajallisten ominaisuuksien havaitsemiseksi. (Rao ja K.E. 2017, 85.) Puheentunnistuksessa käytetään tyypillisesti 20-30 millisekunnin mittaisia ikkunoita ja 10-15 millisekunnin siirtymiä ikkunoiden välillä. Limittäisyydellä tavoitellaan sitä, että kukin

äänisignaalin puheäänne olisi jonkin ikkunan keskellä (Rao ja K.E. 2017, 85-86). Jokaista ikkunan arvoa painotetaan sopivaa ikkunafunktiota käyttäen, millä pyritään vähentämään epäjatkuvuuksia peräkkäisten ikkunoiden välillä (Jurafsky ja Martin 2020, 553). Yleisesti käytettyjä ikkunafunktioita ovat Hamming- ja Hann-funktiot. Tässä tutkimuksessa käytetään Hamming-funktiota, jolloin L:n näytteen pituisen ikkunan yksittäinen arvo x ajanhetkellä n lasketaan yhtälöillä

$$x_n = w(n)s_n, \tag{3.14}$$

$$w(n) = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{L}), & 0 \leq n \leq L-1 \\ 0, & muulloin, \end{cases} \tag{3.15}$$

joissa w on Hamming-funktio ja s_n on signaalin arvo ajanhetkellä n (Jurafsky ja Martin 2020, 553).

Toisessa vaiheessa selvitetään, kuinka paljon energiaa äänisignaali sisältää kullakin taajuudella. Tähän käytetään DFT:tä, jonka määritelmä on

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn}, \qquad 0 \leq k \leq N-1, \tag{3.16}$$

jossa j on imaginaariyksikkö. DFT:llä lasketaan kunkin N:n näytteen pituisen ikkunan jokaiselle näytteelle kompleksiluku $X(k)$, joka kuvaa ikkunan näytteen x_k magnitudia ja vaihetta alkuperäisessä signaalissa. (Jurafsky ja Martin 2020, 554.)

Kolmannessa vaiheessa DFT:llä saadut magnitudispektrit muunnetaan ihmisten havaitsemia äänitaajuuseroja kuvaavaan mel-skaalaan. Tuloksena saadaan mel-spektrogrammiesitykset alkuperäisistä äänisignaaleista. Ihmiset havaitsevat matalien äänitaajuuksien muutokset korkeita äänitaajuuksia paremmin, joten mel-skaala harventuu taajuuksien kasvaessa (Jurafsky ja Martin 2020, 554). DFT:stä saatu magnitudispektri muunnetaan mel-skaalaan käyttäen kolmiomaisia suodattimia, jotka keräävät niiden esittämiä mel-taajuuksia vastaavien taajuksien energiat yhteen. Mel-muunnoksen $s(m)$ määritelmä on

$$s(m) = \sum_{k=0}^{N-1} (|X(k)|^2 H_m(k)) \qquad 0 \leq m \leq M-1\,, \tag{3.17}$$

jossa M on mel-suodattimien määrä. (Rao ja K.E. 2017, 86-87.) Painokerroin $H_m(k)$ määrittää sen, kuinka suuri vaikutus magnitudilla $X(k)$ on mel-taajuuskaistaan m ja sen määritelmä

on

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \frac{2(k-f(m-1))}{f(m)-f(m-1)}, & f(m-1) \leq k \leq f(m) \\ \frac{2(f(m+1)-k)}{f(m+1)-f(m)}, & f(m) < k \leq f(m+1) \\ 0, & k > f(m+1), \end{cases} \tag{3.18}$$

jossa f on $M+2$ mittainen lista tasaisesti poimittuja mel-taajuuksia vastaavia fyysisiä taajuuksia. Taajuuslista f lasketaan seuraavan yhtälön mukaisesti:

$$f(m) = (\frac{N}{F_s}) f_{mel}^{-1}(f_{mel}(f_l) + m(\frac{f_{mel}(f_h) - f_{mel}(f_l)}{M+1})), \tag{3.19}$$

jossa F_s on alkuperäisen signaalin näytteenottotaajuus, f_l on alin huomioitava taajuus ja f_h ylin huomioitava taajuus. (Huang ym. 2001, 314-315.) Fyysinen hertseinä kuvattu taajuus y skaalataan mel-asteikolle ja takaisin seuraavien yhtälöiden mukaisesti:

$$f_{mel}(y) = l = 2595 \log_{10}(1 + \frac{y}{700}), \tag{3.20}$$

$$f_{mel}^{-1}(l) = y = 700(10^{\frac{l}{2595}} - 1) \tag{3.21}$$

(Rao ja K.E. 2017, 86).

Neljännessä vaiheessa jokaisesta edellisen vaiheen ulostulosta otetaan luonnollinen logaritmi, sillä ihmisen kuulon herkkyys äänisignaalin magnitudiin on logaritminen, kuten se on taajuudenkin tapauksessa. Logaritmi tekee piirteistä vähemmän riippuvaisia pienistä muutoksista syötesignaalissa. (Jurafsky ja Martin 2020, 555.)

Viimeisessä vaiheessa log-mel-spektrien korrelaatiota toisiinsa nähden vähennetään DCT:llä. Tämän operaation tuloksena saadaan MFCC-piirteet. DCT $c(n)$ suoritetaan log-mel-magnitudille $s(m)$ seuraavasti:

$$c(n) = \sum_{m=0}^{M-1} \log_{10}(s(m)) \cos(\frac{\pi n(m-0.5)}{M}), \qquad n = 0, 1, 2, ..., C-1, \tag{3.22}$$

jossa C on laskettavien MFCC:iden lukumäärä. Ensimmäinen MFCC, $n = 0$, jätetään yleensä pois, sillä se kuvaa signaalin keskimääräistä logaritmista energiaa, joka sisältää vain vähän puhujakohtaista tietoa. (Rao ja K.E. 2017, 87.)

MFCC-piirteiden ajallisia muutoksia voidaan kuvata tarkemmin laskemalla MFCC:iden ensimmäinen ja toinen derivaatta. Ensimmäisellä derivaatalla kuvataan puheen tahtia ja toisella derivaatalla kuvataan puheen tahdin muutosnopeutta (Rao ja K.E. 2017, 88). Derivaatta $d_t(m)$ lasketaan m:nnelle MFCC:lle $c_t(m)$ ikkunassa t seuraavan yhtälön mukaisesti:

$$d_t(m) = \frac{\sum_{n=1}^{N} n(c_{t+n}(m) - c_{t-n}(m))}{2\sum_{n=1}^{N} n^2} \qquad m = 1,2,3,...,C. \tag{3.23}$$

Derivaatat lasketaan tyypillisesti ajallisesti kahden edeltävän ja kahden seuraavan MFCC-piirteen perusteella, jolloin N on 2. (Lyons 2013.) Toiset derivaatat saadaan saman yhtälön mukaisesti, kun syötteenä käytetään ensimmäisiä derivaattoja.

Piirteisiin voidaan lisätä myös ikkunoiden kokonaisenergioita kuvaava vektori (Lopez-Meyer ym. 2015, 275). Ikkunan t kokonaisenergia E lasketaan seuraavan yhtälön mukaisesti:

$$E = \sum_{n=1}^{N} x_t(n)^2, \tag{3.24}$$

jossa $x_t(n)$ on ikkunan t n:s näyte (Muda, Begam ja Elamvazuthi 2010, 140). Kokonaisenergia lasketaan ennen MFCC-piirreirrotuksen toista vaihetta.

4 Tutkimusasetelma

Kolme neuroverkkoihin perustuvaa puheentunnistusmallia valittiin vertailtavaksi tunnistustarkkuuden, tunnistusnopeuden, koulutusajan sekä parametrien ja liukulukulaskuoperaatioiden määrän suhteen. Puheentunnistusmalleille muodostettiin useita eri konfiguraatioita vaihtelemalla mallien rakennetta hallinnoivia parametreja. Koulutukseen käytettiin kolmelta eri puhujalta kerättyjä ääninäytteitä. Piirteenirrotusmetodeina käytettiin mel-spektrogrammeja ja MFCC:itä.

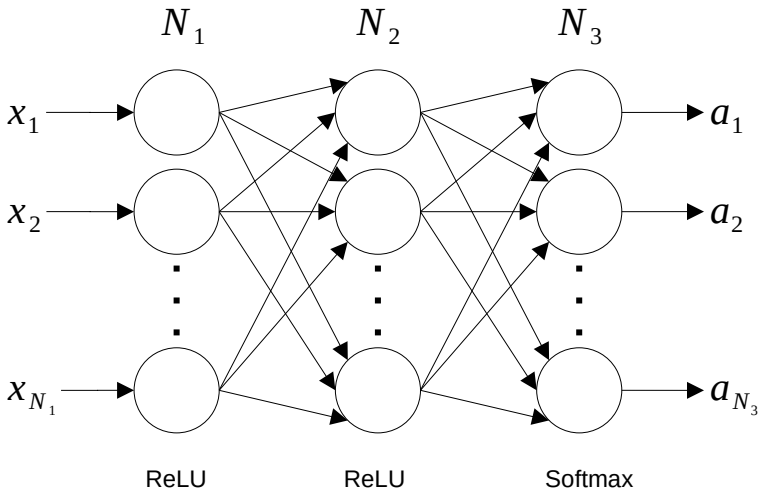
4.1 Tutkittavat neuroverkot

Vertailtavat kolme neuroverkkoa toteutettiin Lopezin ym. (2015), Majumdarin ja Ginsburgin (2020) sekä Anhin ym. (2021) esittelemien neuroverkkorakenteiden perusteella. Neuroverkot koodattiin TensorFlow 2.4.1 -ohjelmointikirjaston Python rajapintaa käyttäen.[1]

Lopezin ym. (2015) tutkimukseen perustuva neuroverkko on kolmikerroksinen MLP. MLP:n syöte muutetaan yksiulotteiseksi vektoriksi, jonka pituus määrää MLP:n syötekerroksen N_1 neuroneiden määrän. Ulostulokerros N_3 sisältää yhtä monta neuronia kuin on äänikomentoluokkia ja sen aktivaatiofunktiona käytetään softmaxia. MLP:n rakennetta säädeltiin muuttamalla piilokerroksen N_2 neuronimäärää. MLP:n rakenne on kuvattu kuviossa 6.

Toinen tutkittava neuroverkko vastaa rakenteeltaan Majumdarin ja Ginsburgin (2020) tutkimaa MatchboxNetiksi nimettyä neuroverkkoa. Neurvoverkko koostuu B peräkkäisestä palikasta, jotka rakentuvat R alipalikasta. Yksittäinen alipalikka suorittaa yksiulotteisen syvyyseroteltavan konvoluution, eränormalisoinnin, ReLU-aktivoinnin ja pudotuksen. Jokainen palikka sisältää jäännösoikotien, jossa palikan syötteelle suoritetaan pistekohtainen konvoluutio ja eränormalisointi, minkä tulos lisätään palikan viimeisen ReLU-aktivaation syötteeseen. MatchboxNetissä on yksi alipalikka ennen palikoita ja kaksi niiden jälkeen. Viimeisinä kerroksina on pistekohtainen konvoluutio, yksiulotteinen GAP ja softmax-aktivaatio. MatchboxNetin rakennetta on havainnollistettu kuviossa 7. MatchboxNetin rakennetta säädeltiin muuntelemalla palikoiden määrää B, niiden sisältämien alipalikoiden määrää R ja konvoluu-

1. Koodit ovat saatavilla osoitteessa: https://github.com/pappnu/sd-scr.



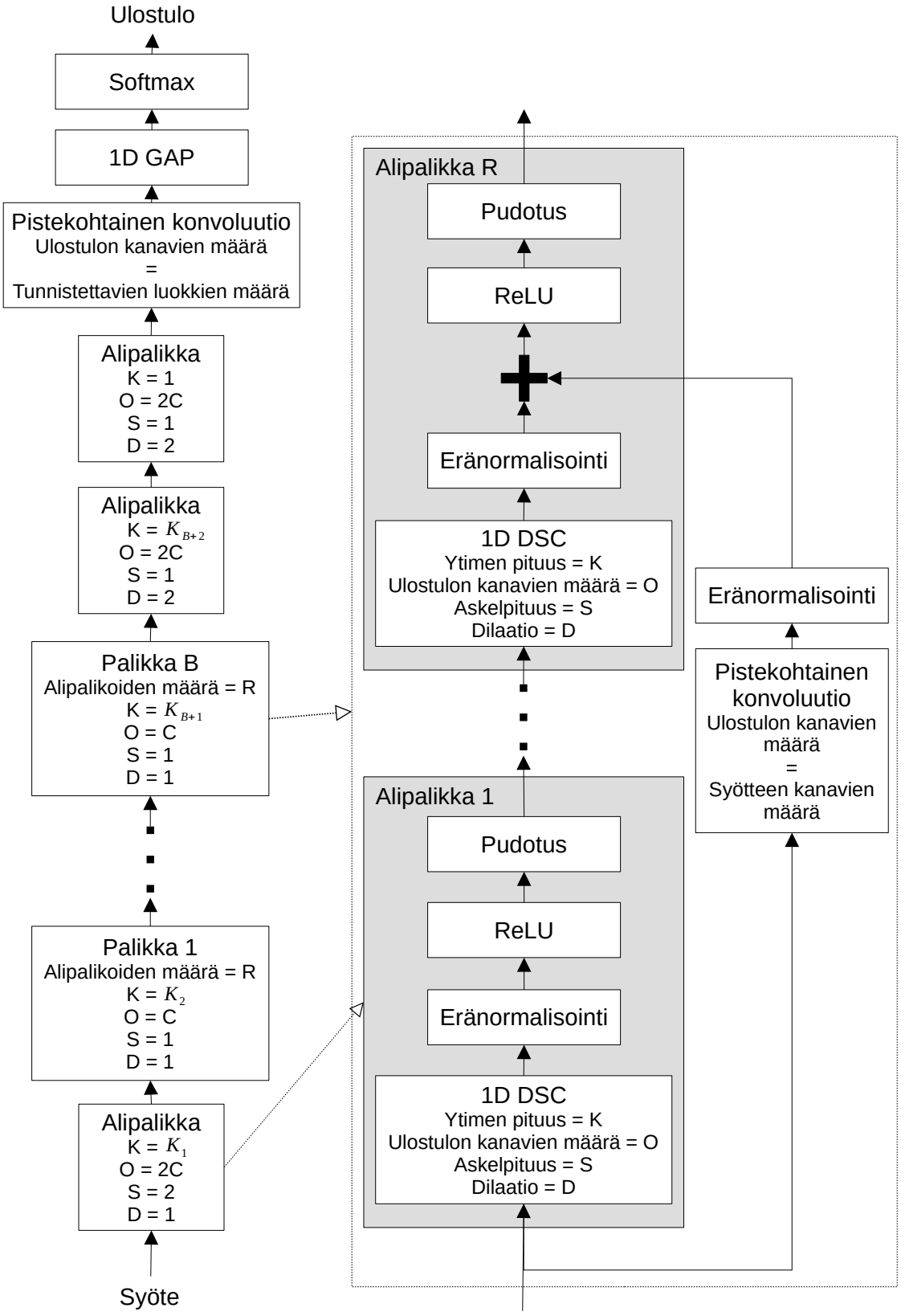
Kuvio 6: Monikerroksisen perseptroniverkon rakenne

tioiden ulostulojen kanavien määrää C, joka vaikuttaa konvoluutioiden ydinten määrään.

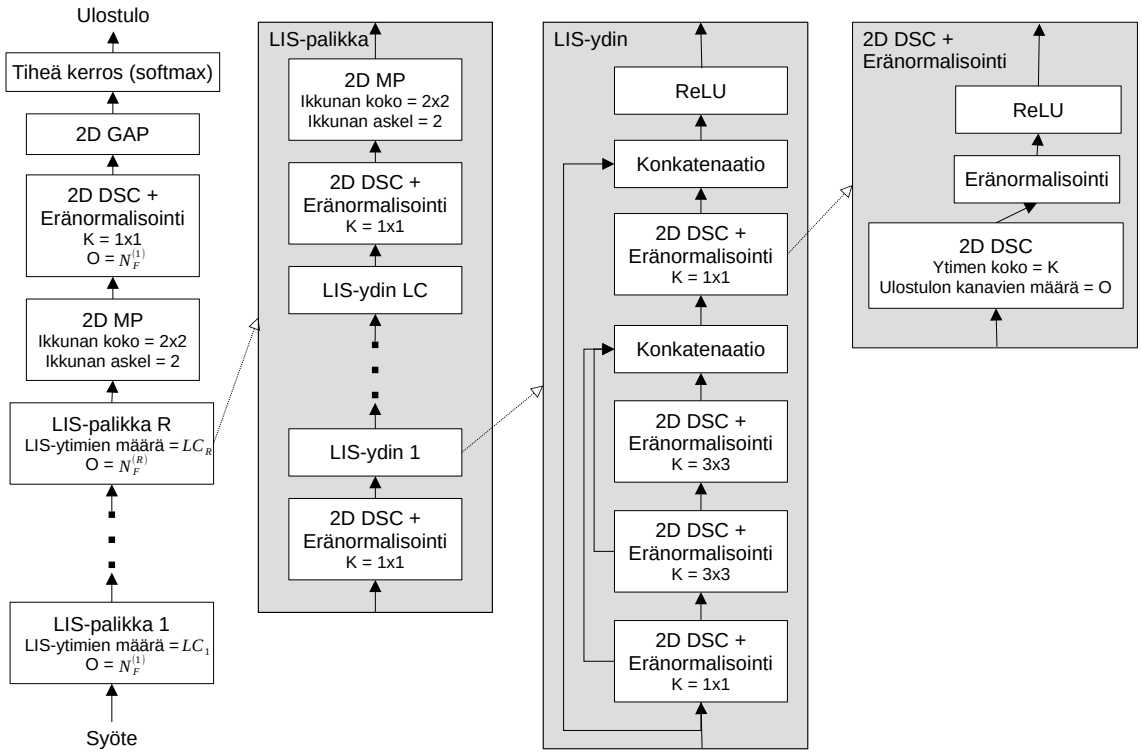
Kolmas tutkittava neuroverkko toteutettiin Anhin ym. (2021) tutkimuksen pohjalta. LIS-Netiksi nimetty neuroverkko rakentuu LIS-palikoista, jotka koostuvat LIS-ytimistä. LIS-ytimessä syötteelle suoritetaan pistekohtainen konvoluutio ja kahdesti peräkkäin 3×3 ytiminen konvoluutio. Konvoluutioiden tulokset ketjutetaan ja kokoelmalle suoritetaan pistekohtainen konvoluutio, jonka ulostulo ketjutetaan LIS-ytimen syötteeseen, mistä muodostuu jäännösoikotie. Jokaisen konvoluution jälkeen suoritetaan eränormalisointi ja ReLU-aktivaatio. LIS-palikoiden välillä käytetään maksimien yhtymää laskutoimitusten vähentämiseksi. Viimeisenä kerroksena on täysin yhdistetty softmax-aktivaatiokerros, jota edeltää globaalien keskiarvojen yhtymä. Kaikki LIS-Netin konvoluutiot ja yhtymät ovat kaksiulotteisia. LIS-Netin rakennetta säädeltiin muuntelemalla LIS-palikoiden määrää R, niiden sisäisten LIS-ytimien määrää LC ja konvoluutioiden ulostulojen kanavien määrää N_F. LIS-verkon rakenne on esitetty kuviossa 8.

4.2 Data-aineisto

Datana käytettiin kolmen nuoren miehen lausumia äänitteitä sanoista apua, ei, kello, kyllä, lääkkeet, muistutus, ohjelma, soita, säätiedote ja viesti. Komennoiksi valittiin sanoja, joille



Kuvio 7: MatchboxNetin rakenne



Kuvio 8: LIS-Net koostuu *R* LIS-palikasta. Parametri ***LC*** määrää LIS-palikan ytimien määrän ja $\boldsymbol{N}_F$ määrää konvoluutioiden ydinten määrän.

voisi olla käyttöä vanhustenhoitoon suunnatussa robotissa. Mukaan otettiin lyhyitä ja pitkiä sanoja, jotta nähtäisiin, aiheuttaako komentojen pituuden vaihtelu ongelmia puheentunnistukselle. Kustakin sanasta on kutakin henkilöä kohden 30 noin sekunnin mittaista näytettä. Lisäksi jokaista henkilöä kohden on noin viiden minuutin edestä muuta sekalaista puhetta ja taustahälyä. Kukin puhuja äänitti näytteet kotonaan valitsemiaan laitteita käyttäen ja pyrki lausumaan sanat monin eri tavoin, esimerkiksi muuttamalla puhenopeutta ja äänenvoimakkuutta, millä pyrittiin simuloimaan vaihtelevia tilanteita, joissa puheentunnistuksen tulisi toimia. Äänitteet muunnettiin vastaavaan muotoon kuin Lopezin ym. (2015, 275) tutkimuksessa ja Googlen Speech Commands -data-aineistossa (Warden 2018, 6), jota käytettiin MatchboxNetin ja LIS-Netin tutkimusjulkaisuissa, eli yksikanavaisiksi 16 000 Hz näytteenottotaajuutta ja 16 bittiä näytettä kohden käyttäviksi WAVE-tiedostoiksi (Waveform audio file format). Äänitteistä leikattiin sanaa edeltävä hiljaisuus, jotta yli sekunnin mittaisista komennoista menetettäisiin mahdollisimman pieni osuus piirteenirrotusvaiheessa. Sekalainen puhe ja taustahäly pilkottiin sekunnin mittaisiksi pätkiksi.

Piirteenirrotusvaiheessa kaikki yli sekunnin mittaiset näytteet leikattiin sekunnin mittaisiksi ja alle sekunnin mittaiset näytteet täydennettiin nollilla, toisin sanoen äänettömyydellä, sekunnin mittaisiksi. Kunkin tutkittavan neuroverkon kohdalla käytettiin vastaavassa tutkimusjulkaisussa esiteltyjä piirteenirrotusparametreja. Lopez-Meyer ym. (2015) sekä Majumdar ja Ginsburg (2020) eivät maininneet mel-taajuuksien laskemiseen käyttämiään taajuusvälejä, joten kyseisten neuroverkkojen kohdalla käytettiin ihmiselle tyypillisesti kuultavissa olevia taajuuksia.

MLP:n syötteinä käytettiin MFCC-piirteitä, niiden ensimmäisiä ja toisia derivaattoja sekä ikkunoiden kokonaisenergioita (Lopez-Meyer ym. 2015, 275). Muodostetuissa piirteissä oli 12 ensimmäistä MFCC-piirrettä, niille 12 derivaattaa ja 12 toisen asteen derivaattaa sekä ikkunoiden kokonaisenergiat, jolloin muodostui 50×37 matriiseja. MatchboxNetin syötteinä käytettiin MFCC-piirteitä (Majumdar ja Ginsburg 2020, 2). Piirteiden ensimmäistä ulottuvuutta, eli aikaulottuvuutta, täydennettiin symmetrisesti nollilla, niin että lopputuloksena saatiin 128×64 matriiseja. LIS-verkon syötteinä käytettiin mel-spektrogrammeja (Anh ym. 2021, 7). Muodostettujen mel-spektrogrammien mitat olivat 125×80. Piirteiden laskemiseen käytetyt parametrit on esitetty kunkin mallin osalta taulukossa 1.

Malli	**Ikkunan pituus (ms)**	**Ikkunan askel (ms)**	**Pienin taajuus (Hz)**	**Suurin taajuus (Hz)**	**Mel-taajuudet**	**Poimitut mel-taajuudet**
MLP	20 (320)	20 (320)	20	8000	40	12 ensimmäistä
Matchbox-Net	25 (400)	10 (160)	20	8000	64	Kaikki (64)
LIS-Net	64 (1024)	8 (128)	40	8000	80	Kaikki (80)

Taulukko 1: Piirteenirrotuksessa käytetyt parametrit. Ikkunan pituuksien ja askeleiden kohdalla on ilmoitettu suluissa, montako taajuusnäytettä kyseiseen aikaikkunaan mahtuu.

4.3 Metriikat

Neuroverkoista mitattiin niiden tunnistustarkkuus testidatan osalta, testierän tunnistukseen kulunut aika, koulutukseen kulunut aika, parametrien eli painojen määrä ja liukulukulaskuoperaatioiden määrä (floating point operations, FLOs), joka kertoo kuinka monta liukulukulaskuoperaatiota täytyy suorittaa neuroverkon läpikäymiseksi kerran. Tunnistustarkkuus *ACC*, eli oikein tunnistettujen äänitteiden osuus kaikista testinäytteistä, laskettiin kaavalla

$$ACC = \frac{N_1}{N_2}, \tag{4.1}$$

jossa N_1 on oikein tunnistettujen testiäänitteiden määrä ja N_2 kaikkien testiäänitteiden määrä. Yhden 32 testinäytteen erän (batch) tunnistukseen kuluva aika mitattiin mallikohtaisesti ajoittamalla Pythonin timeit-moduulin timeit-metodilla TensorFlowin tf.keras.model.predict-metodin suorittamiseen kuluva aika kahden eri testinäyte-erän osalta 20 toiston verran. Koulutusaika mitattiin ajoittamalla Pythonin time-moduulin perf_counter-metodilla TensorFlowin tf.keras.Model.fit-metodin mukaisen koulutuksen alusta viimeisen epochin loppuun kulunut aika. Parametrien määrä laskettiin TensorFlowin tf.keras.backend.count_params-metodia käyttäen. Liukulukulaskuoperaatioiden määrän laskemiseen käytettiin keras-flops-paketin versiota 0.1.2 (Tokusumi 2020).

Mallien kykyä tunnistaa vain opetettuja sanoja tutkittiin asettamalla softmax-arvoille rajaarvo, jota varmempi mallin tuli olla ennusteesta, jotta se hyväksyttiin. Sekä hylättyjen oikei-

den ennusteiden osuus kaikista oikeista ennusteista että väärien hälytysten, eli väärin komennoiksi tunnistettujen äänitteiden, jotka eivät sisällä koulutettuja komentoja, osuus kaikista ei komennoiksi luokitelluista äänitteistä mitattiin. Osuudet laskettiin samalla periaatteella kuin tunnistustarkkuus 4.1.

4.4 Koulutuksen hyperparametrit

Kutakin mallia koulutettiin 190 epochin verran, jonka jälkeen aktivoitiin esilopetusehto, joka edellytti, että validaatiotappion tulisi laskea aiemmin sillä koulutuskerralla havaittua alinta validaatiotappiota pienemmäksi 20 epochin aikana tai muuten koulutusta ei jatkettaisi. Jos uusi validaatiotappiominimi saavutettiin 20 epochin aikana, annettiin koulutukselle taas 20 epochia aikaa saavuttaa uusi minimi. Tätä jatkettiin kunnes uutta minimiä ei enää saavutettu 20 epochin sisällä tai yhteensä 270 epochia tuli täyteen mukaan lukien esilopetusehtoa edeltäneet 190 epochia. Esilopetusehdolla pyrittiin välttymään turhalta kouluttamiselta, jossa malli ylioppii koulutusdatansa. Koulutuksen lopuksi mallien painot palautettiin niiksi, joilla kukin malli saavutti validaatiominiminsä. Koulutusnäytteet syötettiin koulutusalgoritmille 32 näytteen erissä. Validaatio- ja testiäänitteitä oli kumpiakin aina viisi jokaista komentoa kohden. Koulutusnäytteiden määrää vaihdeltiin, joten se on mainittu erikseen tulosten yhteydessä.

Lopez-Meyer ym. (2015) eivät maininneet, mitä optimointialgoritmia he käyttivät, joten MLP:n optimointiin valittiin yleisesti käytetty Adam. Adamia käytettiin myös LIS-Netin koulutuksessa. Adamin oppimistahdiksi (learning rate) asetettiin 0.001 ja liikemääräksi (momentum) 0.9. (Anh ym. 2021, 7.) MatchboxNet-mallien kanssa käytettiin NovoGrad-algoritmia, jossa oppimistahti oli 0.001, β_1 oli 0.95, β_2 oli 0.5 ja painojen kuluma (weigth decay) oli 0.001 (Majumdar ja Ginsburg 2020). MatchboxNet-mallien pudotuskerrosten pudotustodennäköisyydeksi asetettiin 20 %.

5 Tulokset

MLP:n, MatchboxNetin ja LIS-Netin eri rakenteita vertailtiin aluksi puhujan 1 datan pohjalta. Koulutukseen käytettiin 20 näytettä sekä validointiin ja testaukseen viisi näytettä jokaista komentoa kohden. Kolme MLP-mallia, 27 MatchboxNet-mallia ja 27 LIS-Net-mallia koulutettiin käyttäen samoja näytteitä kunkin eri mallin koulutukseen ja testaukseen. Vertailuun valittiin enemmän yksinkertaisia rakenteita, sillä Lopez-Meyer ym. (2015), Ginsburg ym. (2020) sekä Anh ym. (2021) kertoivat tutkimuksissaan, että mallien kasvattaminen ei johtanut merkittävästi parempiin tunnistustarkkuuksiin. Tutkimuksissa ei testattu yhtä monipuolisesti yksinkertaisempia rakenteita. Yksinkertaisempien rakenteiden etuna on vähäisempi laskentatehon, muistin ja siten myös energian tarve, jolloin ne soveltuvat paremmin akkukäyttöisiin laitteisiin. MLP-, MatchboxNet- ja LIS-Net-mallien testitulokset on esitetty taulukoissa 2, 3 ja 4.

Kaikki MLP-mallit suoriutuivat huonosti sekä validaatio että testidatan osalta jääden enimmäkseen alle 60 % tunnistustarkkuuden, mikä on merkki koulutusdatan ylioppimisesta. Tulos poikkeaa huomattavasti Lopezin ym. (2015) saavuttamista korkeista tunnistustarkkuuksista. Lopezin ym. keräämä data saattoi sisältää vähemmän vaihtelua yksittäisten komentojen näytteiden välillä, mutta data ei ole vapaasti saatavilla, joten asiaa ei voitu varmistaa. Lopezin ym. käyttämä tallennuslaitteisto oli tarkempi ja äänitystila oli häiriöttömämpi kuin tässä tutkimuksessa käytetyt laitteistot ja äänitystilat. MLP-mallien heikon tunnistustarkkuuden ja MatchboxNettiin verrattuna suuren parametrimäärän vuoksi niitä ei tutkittu tarkemmin.

MatchboxNetit saavuttivat parhaimmillaan 86 % tunnistustarkkuuden testidatan osalta. Validaatio- ja testitarkkuudet olivat enimmäkseen noin 80 % paikkeilla kaikkien paitsi kolme palikkaa sisältävien mallien osalta. Kolme palikkaa sisältävät mallit suoriutuivat kaikista yksinkertaisintakin mallia huonommin jääden paikoin alle 70 % tarkkuuden. Heikommat tulokset eivät johdu suoraan parametrien määrästä, sillä malli, jossa on enemmän parametrejä kuin joissain kolme palikkaa sisältävissä malleissa, saavutti yli 80 % tunnistustarkkuuden.

Eniten ja kolmanneksi eniten parametreja sisältävät LIS-Net-mallit saavuttivat 96 % tunnistustarkkuuden, joka on korkein kaikkien mallien osalta saavutettu tarkkuus. Toiseksi eniten

parametreja sisältävä LIS-Net malli saavutti merkittävästi pienemmän 84 % tunnistustarkkuuden. Kaikki muut LIS-Net-mallit jäivät alle 80 % tunnistustarkkuuden ja erityisesti vähän parametreja sisältäneet mallit suoriutuivat heikosti tunnistaen vain noin 20-45 % testidatasta oikein.

N_2	**FLOs (miljoonina)**	**Parametrit (tuhansina)**	**Validaatio-tarkkuus (%)**	**Testitarkkuus (%)**
40	7,76	3882,9	54,0	18,0
50	7,8	3902,51	60,0	26,0
60	7,84	3922,12	48,0	32,0

Taulukko 2: Piilotason neuronimäärän N_2 vaikutus MLP:n tunnistustarkkuuteen

Sekä MatchboxNettien että LIS-Nettien joukosta valittiin tarkempaan tarkasteluun kolme suurimman testitunnistustarkkuuden saavuttanutta mallia. Valittuihin malleihin viitataan jatkossa lyhenteillä Mbox MatchboxNettien osalta ja LIS LIS-Nettien osalta. Mallit numeroitiin kasvavasti niiden ylhäältä alas järjestyksen mukaan taulukoissa 3 ja 4. Tarkempaan tarkasteluun valitut MatchboxNet- ja LIS-Net-mallit koulutettiin erikseen kunkin puhujan datoilla sekä käyttäen 5, 10, 15 ja 20 koulutusnäytettä kutakin komentoa kohden. Kukin tapaus ajettiin viidesti käyttäen uudelleen arvottua koulutus-, validaatio- ja testinäytejakoa. Jokaisen tapauksen tulokset on esitetty taulukossa 5 eri toistojen keskiarvoina.

Puhujan 1 testidatan tunnisti tarkimmin 90,4 % tarkkuudella LIS 2 -malli koulutettuna 20 koulutusnäytteellä kutakin komentoa kohden. Puhujan 2 datan tunnisti tarkimmin 75,6 % tarkkuudella MBox 3 -malli käytettäessä 20 koulutusnäytettä. Puhujan 3 testidatan tunnisti tarkimmin MBox 1 -malli 96 % tarkkuudella käytettäessä 15 koulutusnäytettä. Keskimäärin kaikkien puhujien testinäytteet tunnisti parhaiten MBox 2 -malli 85,6 % tarkkuudella 15 koulutusnäytettä käytettäessä. MatchboxNet-malleihin verrattuna LIS-Net-mallit pärjäsivät hyvin vain puhujan 1 tapauksessa ja siinäkin tapauksessa huomattavasti huonommin kuin taulukon 4 tulokset antoivat ymmärtää. MatchboxNettien kohdalla 15 koulutusnäytteen käyttäminen antoi keskimäärin parhaat tulokset. Lähelle 15 näytteen tasoa päästiin kymmenelläkin koulutusnäytteellä, jota seurasi 20 näytteen tapaukset. LIS-Nettien tapauksessa 15 näytettä antoi keskimäärin parhaat ja 20 näytettä toiseksi parhaat tulokset. Kymmenen

näytteen käyttäminen johti jopa puolet huonompaan tunnistustarkkuuteen. Viiden näytteen käyttäminen johti huonompiin tuloksiin kaikkien mallien osalta, mutta LIS-Net-mallit kärsivät enemmän pudoten 10 % tunnistustarkkuuteen. Eri puhujien välillä tunnistustarkkuudet vaihtelivat huomattavasti. MatchboxNetit saavuttivat yli 90 % tunnistustarkkuuden puhujan 3 testinäytteiden osalta, noin 85 % tarkkuuden puhujan 1 kohdalla ja vain noin 70 % tarkkuuden puhujan 2 tapauksessa.

LIS-Nettien MatchboxNetteihin verrattuna 50-100 kertainen parametrimäärä ja 300-700 kertainen liukulukulaskuoperaatioiden määrä näkyi pidempinä koulutus- ja testiaikoina. Yhden LIS-Net mallin koulutukseen kului noin 10-15 minuuttia käytettäessä 15 koulutusnäytettä komentoa kohden, kun vastaava aika MatchboxNeteillä oli noin minuutti. Testinäyte-erien luokittelu LIS-Net-malleilla vei noin 3,2-3,5 sekuntia ja MatchboxNet-malleilla noin 1-1,3 sekuntia. Testiajoissa ei ollut suuria eroja mallityyppien eri rakenteiden välillä, vaikka esimerkiksi LIS 3 -mallissa on noin kaksinkertainen määrä parametreja ja noin 30 % enemmän liukulukulaskuoperaatiota LIS 1 -malliin verrattuna.

MBox 2 ja LIS 3 -malleja 15 koulutusnäytteellä koulutettuna tarkasteltiin komentokohtaisten luokitteluvirheiden ja väärien hälytysten näkökulmasta. Kuviosta 9 nähdään, että luokitteluvirheitä oli MBox 2 -mallin sekä puhujien 1 ja 2 osalta erityisesti apua- ja viestikomentojen kohdalla. Puhujan 2 kohdalla monet muutkin komennot tunnistuivat väärin, kuten taulukon 5 tuloksista saattoi olettaa. Kuvion 10 perusteella LIS 3 -malli tunnisti soitakomennon huonosti kaikkien puhujien osalta. Apua-komentojakaan LIS 3 -malli ei tunnistanut täysin oikein yhdenkään puhujan tapauksessa. Puhujan 2 kohdalla apua-komennot luokittuivat enemmän kello- ja ohjelma-komennoiksi kuin apua-komennoiksi. Saman sanan osalta eri puhujien tulokset saattoivat kaikki poiketa toisistaan. Esimerkiksi MBox 2 -mallin kohdalla ohjelma-äänitteet tunnistuivat täysin oikein puhujan 3 tapauksessa, eikä yksikään muu äänite tunnistunut ohjelma-komennoksi, mutta puhujan 1 tapauksessa joillain koulutuskerroilla osa ohjelma-äänitteistä tunnistui väärin ja jotkin kello-äänitteet tunnistuivat ohjelma-komennoiksi. Puhujan 2 tapauksessa keskimäärin alle puolet ohjelma-äänitteistä tunnistui oikein.

Väärien hälytysten ja hylättyjen oikeiden ennusteiden osuudet mitattiin MBox 2 -mallin tapauksessa raja-arvoilla 0,9999; 0,9995; 0,995; 0,99; 0,98; 0,97; 0,96; 0,95; 0,94 ja 0,93 sekä

LIS 3 -mallin tapauksessa raja-arvoilla 0,99; 0,95; 0,9; 0,85; 0,8; 0,75; 0,7; 0,65; 0,6; 0,55; 0,5; 0,45; ja 0,4 kunkin puhujan osalta. Testaukseen käytettiin kunkin puhujan omaa sekalaista puhetta ja taustamelua. Tulokset on esitetty kuviossa 11. Kunkin puhujan ja mallin kohdalla joudutaan tinkimään huomattavasti oikeiden tunnistusten määrästä, jos väärät hälytykset halutaan minimoida. Esimerkiksi oikeista avainsanoista noin puolet tai enemmän jäisi havaitsematta, jos väärät hälytykset haluttaisiin alle 3 %:in. LIS 3 -mallin softmax-ulostulojen arvot olivat jakautuneet tasaisemmin eri luokkien välille kuin MBox 2 -mallissa, jossa yksittäinen luokka saavutti yleensä selkeästi suurimman painoarvon, joten vastaavan väärien hälytysten suhteen hylättyihin oikeisiin ennusteisiin saavuttamiseksi mallien välillä tuli LIS 3 -mallin tapauksessa käyttää pienempiä raja-arvoja. Käytettäessä samaa raja-arvoa eri puhujien välillä puhujan 1 väärien hälytysten osuus oli pienin, puhujan 2 osuus oli toisiksi pienin ja puhujan 3 osuus oli suurin.

B	***R***	***C***	**FLOs (miljoonina)**	**Parametrit (tuhansina)**	**Validaatio-tarkkuus (%)**	**Testitarkkuus (%)**
1	1	48	4,38	35,13	80,0	84,0
1	1	64	7,12	56,84	86,0	**86,0**
1	1	80	10,51	83,67	76,0	82,0
1	2	48	4,76	38,25	84,0	78,0
1	2	64	7,76	62,03	78,0	78,0
1	2	80	11,48	91,43	86,0	84,0
1	3	48	5,15	41,37	80,0	80,0
1	3	64	8,41	67,21	78,0	82,0
1	3	80	12,45	99,19	86,0	**86,0**
2	1	48	5,08	40,84	78,0	76,0
2	1	64	8,32	66,51	76,0	76,0
2	1	80	12,34	98,31	82,0	**86,0**
2	2	48	5,86	47,18	84,0	70,0
2	2	64	9,62	77,0	80,0	84,0
2	2	80	14,3	113,99	80,0	78,0
2	3	48	6,64	53,51	84,0	76,0
2	3	64	10,93	87,5	88,0	74,0
2	3	80	16,26	129,67	82,0	84,0
3	1	48	5,8	46,65	80,0	74,0
3	1	64	9,54	76,3	68,0	76,0
3	1	80	14,19	113,11	84,0	78,0
3	2	48	6,99	56,3	78,0	60,0
3	2	64	11,52	92,23	74,0	64,0
3	2	80	17,16	136,87	82,0	72,0
3	3	48	8,18	65,95	58,0	60,0
3	3	64	13,5	108,17	68,0	62,0
3	3	80	20,12	160,63	82,0	66,0

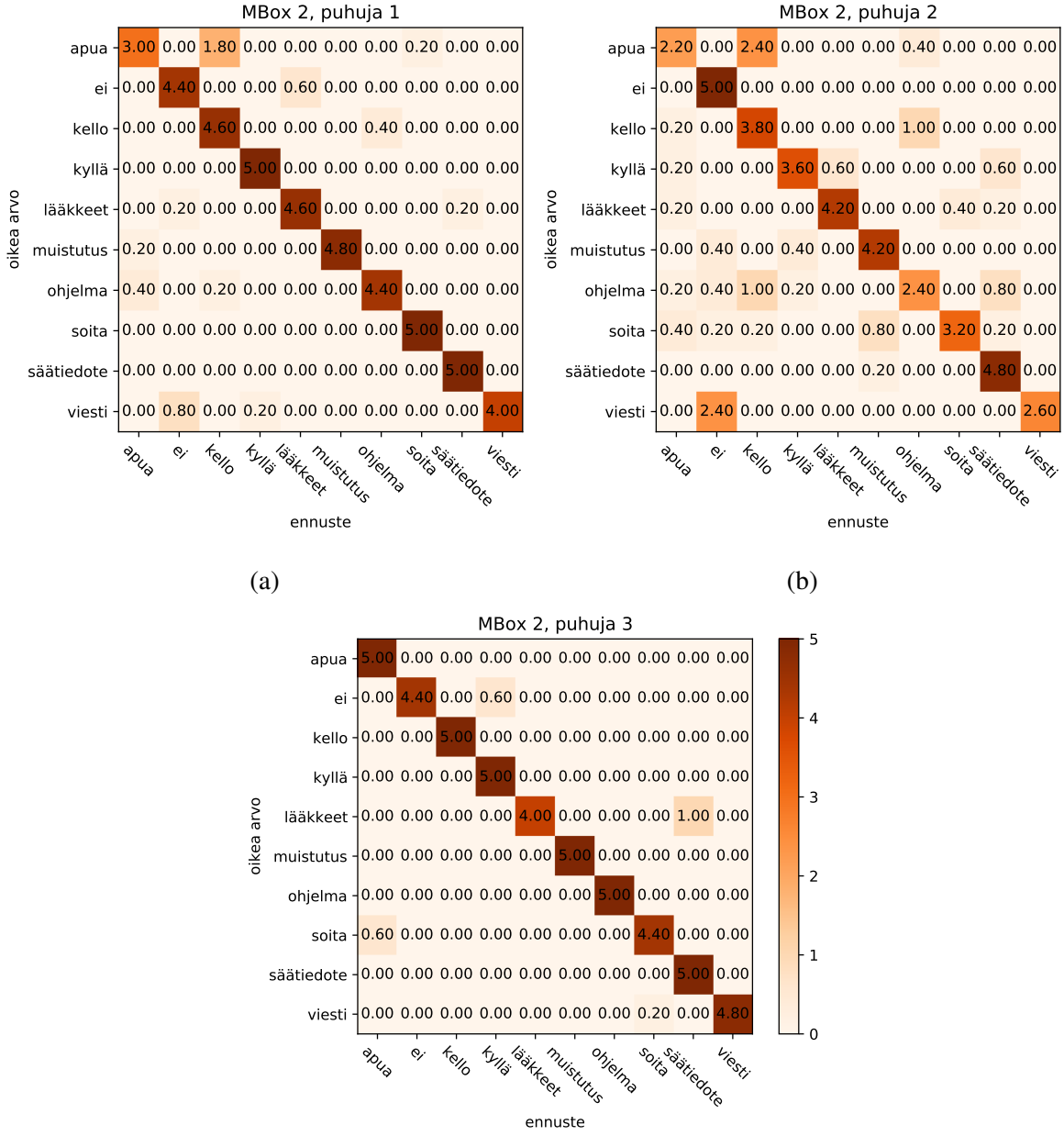
Taulukko 3: Palikoiden *B*, alipalikoiden *R* ja konvoluutioiden ulostulokanavien määrän *C* vaikutus MatchboxNetin tunnistustarkkuuteen

LC	N_F	**FLOs (miljoonina)**	**Parametrit (tuhansina)**	**Validaatio-tarkkuus (%)**	**Testitarkkuus (%)**
[1, 2]	[32, 64]	547,96	85,58	36,0	28,0
[1, 1]	[32, 64]	373,13	50,25	36,0	26,0
[2, 2]	[32, 64]	731,0	95,05	34,0	16,0
[1, 2]	[48, 96]	1200,76	186,73	30,0	10,0
[1, 1]	[48, 96]	814,66	109,16	34,0	16,0
[2, 2]	[48, 96]	1598,2	207,08	26,0	22,0
[1, 2]	[64, 128]	2106,13	326,8	24,0	18,0
[1, 1]	[64, 128]	1426,17	190,6	50,0	24,0
[2, 2]	[64, 128]	2799,89	362,12	30,0	22,0
[1, 2, 3]	[32, 64, 128]	1097,93	522,06	32,0	22,0
[1, 1, 1]	[32, 64, 128]	556,59	197,84	50,0	28,0
[2, 2, 2]	[32, 64, 128]	1087,15	378,83	34,0	16,0
[1, 2, 3]	[48, 96, 192]	2426,56	1156,33	38,0	24,0
[1, 1, 1]	[48, 96, 192]	1223,29	436,52	36,0	26,0
[2, 2, 2]	[48, 96, 192]	2391,7	837,03	30,0	24,0
[1, 2, 3]	[64, 128, 256]	4274,97	2039,43	50,0	46,0
[1, 1, 1]	[64, 128, 256]	2148,95	768,39	40,0	36,0
[2, 2, 2]	[64, 128, 256]	4203,59	1474,44	40,0	28,0
[1, 2, 3, 4]	[32, 64, 128, 256]	1852,7	2896,72	84,0	76,0
[1, 1, 1, 1]	[32, 64, 128, 256]	737,49	771,53	62,0	34,0
[2, 2, 2, 2]	[32, 64, 128, 256]	1438,29	1487,05	76,0	62,0
[1, 2, 3, 4]	[48, 96, 192, 384]	4116,99	6466,28	94,0	**84,0**
[1, 1, 1, 1]	[48, 96, 192, 384]	1628,25	1717,93	54,0	44,0
[2, 2, 2, 2]	[48, 96, 192, 384]	3177,84	3313,45	74,0	70,0
[1, 2, 3, 4]	[64, 128, 256, 512]	7273,23	11450,0	96,0	**96,0**
[1, 1, 1, 1]	[64, 128, 256, 512]	2867,01	3038,09	86,0	66,0
[2, 2, 2, 2]	[64, 128, 256, 512]	5597,67	5861,77	94,0	**96,0**

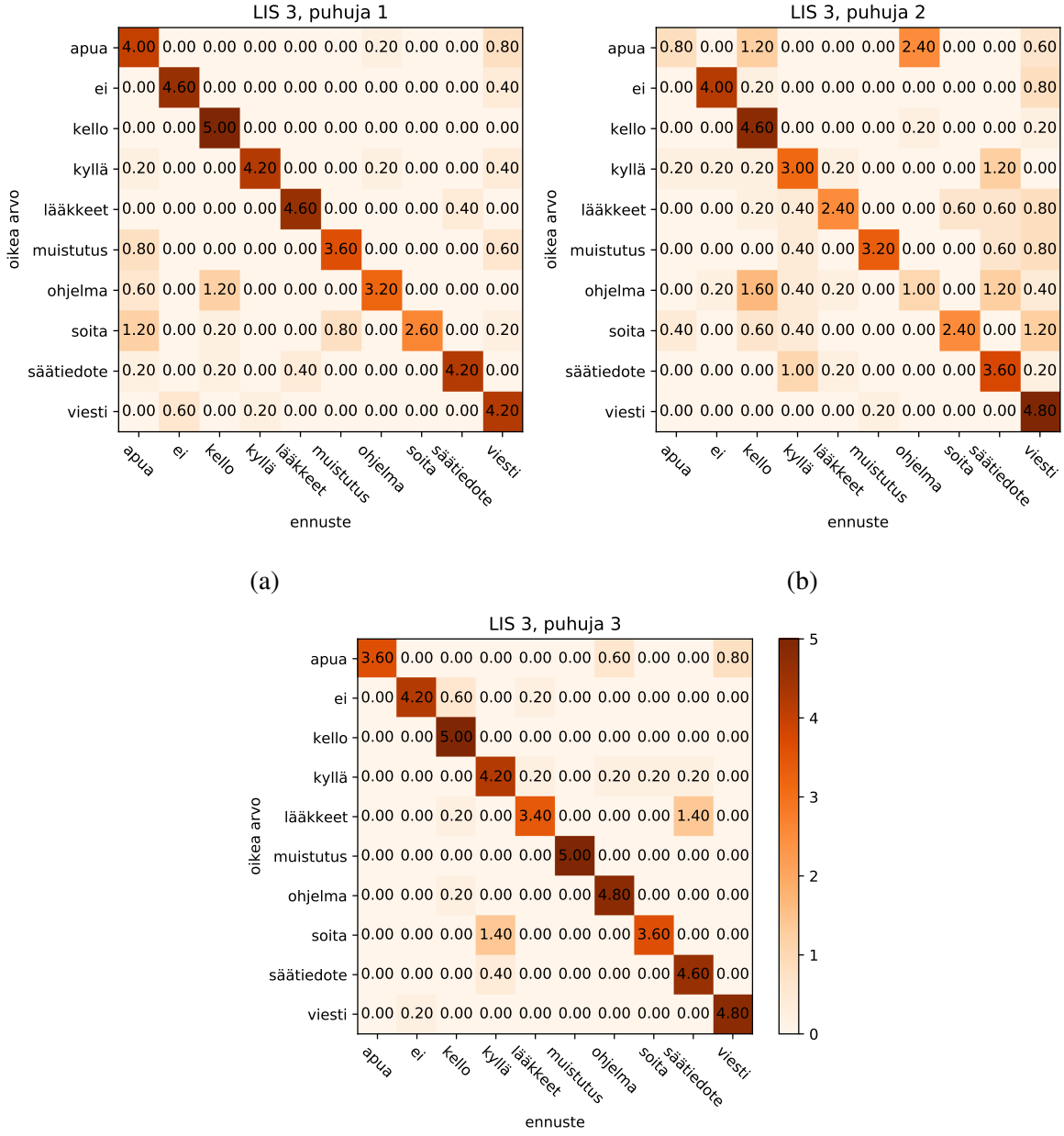
Taulukko 4: LIS-palikoiden ja LIS-ytimien LC sekä konvoluutioiden ulostulokanavien määrän N_F vaikutus LIS-Netin tunnistustarkkuuteen

Malli	**Koulutus-näytteet**	**Koulutusaika (m)**	**Testiaika (s)**	**Puhujan 1 testitarkkuus (%)**	**Puhujan 2 testitarkkuus (%)**	**Puhujan 3 testitarkkuus (%)**	**Keskiarvo testitarkkuus (%)**
MBox 1	5	0,81	1,07	74,4 (5,95)	54,4 (4,53)	82,8 (2,33)	70,53 (4,0)
MBox 1	10	0,89	1,06	78,8 (4,32)	53,2 (5,46)	95,6 (1,72)	75,87 (5,16)
MBox 1	15	0,93	1,05	85,6 (2,48)	62,8 (4,27)	**96,0 (0,63)**	81,47 (4,01)
MBox 1	20	1,08	1,06	80,8 (3,38)	70,8 (3,98)	92,4 (1,72)	81,33 (2,91)
MBox 2	5	0,91	1,15	75,2 (3,98)	39,6 (7,76)	72,4 (14,82)	62,4 (6,84)
MBox 2	10	1,03	1,14	82,0 (3,16)	70,0 (1,67)	93,2 (2,24)	81,73 (2,85)
MBox 2	15	1,14	1,16	89,6 (1,17)	72,0 (2,28)	95,2 (1,36)	**85,6 (2,79)**
MBox 2	20	1,29	1,15	73,6 (3,97)	70,8 (2,58)	90,0 (2,0)	78,13 (2,76)
MBox 3	5	0,91	1,14	77,6 (2,23)	50,4 (9,7)	83,6 (2,64)	70,53 (5,0)
MBox 3	10	0,99	1,13	79,2 (2,42)	60,8 (2,73)	94,8 (1,02)	78,27 (3,89)
MBox 3	15	1,09	1,13	84,0 (2,83)	68,8 (5,08)	94,8 (2,33)	82,53 (3,44)
MBox 3	20	1,29	1,13	72,8 (1,36)	**75,6 (4,17)**	88,4 (3,37)	78,93 (2,49)
LIS 1	5	3,46	3,22	10,0 (0,0)	10,0 (0,0)	10,0 (0,0)	10,0 (0,0)
LIS 1	10	7,09	3,21	35,6 (6,94)	13,2 (2,06)	44,0 (2,19)	30,93 (4,19)
LIS 1	15	10,38	3,19	75,2 (2,06)	56,0 (3,63)	80,0 (4,05)	70,4 (3,3)
LIS 1	20	12,17	3,2	80,4 (1,94)	53,6 (3,66)	76,0 (4,69)	70,0 (3,68)
LIS 2	5	4,14	3,58	10,0 (0,0)	10,0 (0,0)	10,0 (0,0)	10,0 (0,0)
LIS 2	10	8,5	3,58	56,0 (4,86)	10,0 (0,0)	57,2 (2,24)	41,07 (6,1)
LIS 2	15	12,87	3,55	82,0 (3,74)	51,6 (3,37)	85,6 (2,64)	73,07 (4,44)
LIS 2	20	15,04	3,54	**90,4 (1,83)**	50,4 (10,72)	79,6 (1,17)	73,47 (5,64)
LIS 3	5	4,8	3,38	10,0 (0,0)	10,0 (0,0)	10,0 (0,0)	10,0 (0,0)
LIS 3	10	10,15	3,36	36,8 (3,83)	10,0 (0,0)	59,2 (6,18)	35,33 (5,82)
LIS 3	15	15,66	3,42	80,4 (2,56)	59,6 (4,62)	86,4 (1,6)	75,47 (3,51)
LIS 3	20	18,24	3,36	82,4 (2,04)	48,0 (6,99)	78,0 (4,77)	69,47 (4,89)

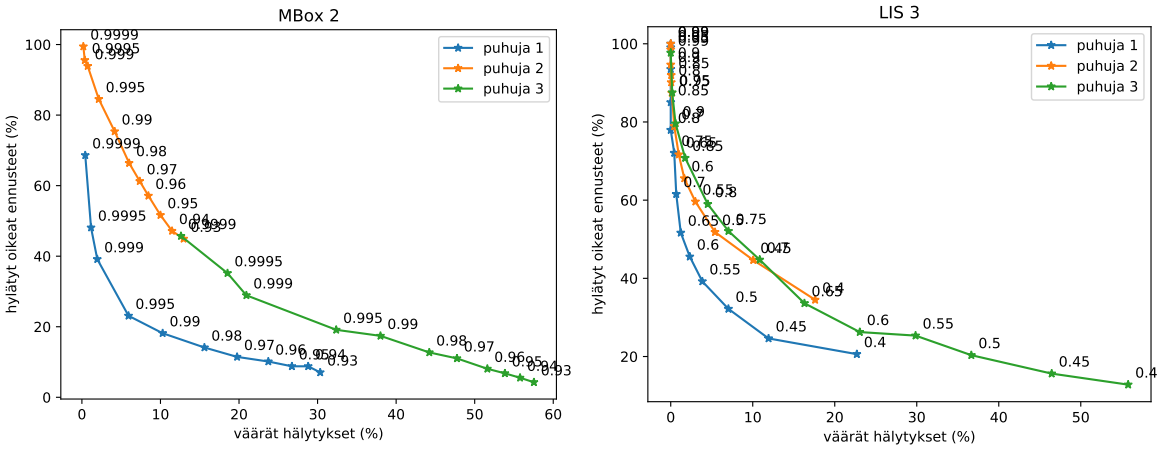
Taulukko 5: MatchboxNet- ja LIS-Net-mallien testitulokset eri puhujien ja koulutusnäytemäärien osalta. Tulokset ovat viiden eri koulutuksen pohjalta laskettuja keskiarvoja. Testitarkkuuksille on ilmoitettu keskivirheet suluissa.



Kuvio 9: Sekaannusmatriisit MBox 2 -mallin puhujakohtaisista ennusteista 15 komentokohtaisen näytteen tapauksessa viiden koulutuksen keskiarvoina. Testinäytteitä oli 5 kutakin komentoa kohden.



Kuvio 10: Sekaannusmatriisit LIS 3 -mallin puhujakohtaisista ennusteista 15 komentokohtaisen näytteen tapauksessa viiden koulutuksen keskiarvoina. Testinäytteitä oli 5 kutakin komentoa kohden.



Kuvio 11: Väärien hälytysten osuus kaikista ei komennoiksi luokitelluista äänitteistä ja hylättyjen oikeiden ennusteiden osuus kaikista oikeista ennusteista eri raja-arvoilla viiden eri koulutuskerran keskiarvoina MBox 2 ja LIS 3 -mallien osalta. Arvot kuvaajien pisteiden vieressä kertovat käytetyn raja-arvon.

6 Johtopäätökset

Tässä tutkimuksessa vertailtiin MLP:n, MatchboxNetin ja LIS-Netin eri konfiguraatioita puhujariippuvaisen puhekomentojentunnistuksen tapauksessa. Koulutus- ja testidatana käytettiin kolmelta eri henkilöltä kerättyjä ääninäytteitä. Kaikki muodostetut konfiguraatiot koulutettiin aluksi yhden puhujan datalla, minkä jälkeen ensimmäisten testien perusteella lupaavimmat mallit koulutettiin kaikkien puhujien datoilla.

Tutkimuksen tutkimuskysymyksillä haettiin eroja vertailtujen mallien tunnistustarkkuudessa, tunnistusnopeudessa, laskennallisessa vaativuudessa, muistinkäytössä ja koulutusdatan käytössä. Tunnistustarkkuus mitattiin oikein tunnistettujen testinäytteiden osuutena, tunnistusnopeus mitattiin testierän tunnistukseen kuluvana aikana ja laskennallinen vaativuus mitattiin mallin suoritukseen tarvittavien liukulukulaskuoperaatioiden määränä. Muistinkäyttö yhdistettiin mallien parametrimääriin. Koulutusdatavaatimuksia arvioitiin kouluttamalla mallit eri komentokohtaisilla äänitemäärillä ja tarkastelemalla niillä saavutettuja tunnistustarkkuuksia. Tunnistustarkkuuksissa suurempi oli parempi ja muiden metriikoiden mittaustavoissa pienempi oli parempi.

Osa MatchboxNet-malleista saavutti yli 90 % tunnistustarkkuuden puhujan 3 tapauksessa, mitä voidaan pitää käyttökelpoisena toimintavarmuutena. Muiden puhujien kohdalla tunnistustarkkuudet jäivät kuitenkin alle 90 %. Erityisesti puhuja 2 erottuu joukosta noin 70 % tai alle tunnistustarkkuuksineen. Erot puhujien välillä voivat johtua äänitystyylien vaihtelusta. Henkilöitä oli ohjeistettu lausumaan komentoja monin eri tavoin, mutta kukin päätti itse missä määrin vaihteli puhetyyliään. Jotkut ovat voineet äänittää enemmän monotonisia toistoja toisten puhuessa monipuolisesti eri äänenkorkeuksilla ja nopeuksilla. Käytetyt äänityslaitteet saattavat selittää osan havaitusta vaihtelusta. Huonompia mikrofoneja käytettäessä signaaliin jää enemmän kohinaa, jolloin komentojen yksityiskohdat saattavat peittyä.

Apua-komento tunnistui huonosti useiden eri malli- ja puhujayhdistelmien kohdalla, mutta muuten mikään komento ei ollut selkeästi ongelmallinen kaikkien puhujien kohdalla. Eri komennot pärjäsivät eri tavoin puhujasta riippuen, mikä voi olla seurausta avoimesta ohjeistuksesta, jolloin puhujat ovat saattaneet panostaa joidenkin komentojen monipuoliseen

lausumiseen enemmän kuin toisten, mikä on johtanut niiden pienempään tunnistustarkkuuteen. Ei-komento tunnistui hyvin kaikkien puhujien kohdalla, mikä voi selittyä sillä, että se on selkeästi lyhyin komento, jolloin se erottuu helposti muista. Asiaa auttaa, että muut komennot eivät sisällä ei-sanan kirjaimia vastaavassa järjestyksessä.

Koulutetut puheentunnistusmallit luokittelevat minkä tahansa äänitteen joksikin komennoksi, sillä niille ei muuta opetettu. Jotta malleja voitaisiin käyttää käytännössä, väärät hälytykset pyrittiin estämään hyväksymällä vain sellaiset ennusteet, joiden softmax-aktivoidun ulostulon suurin arvo ylittää tietyn raja-arvon. Väärien hälytysten minimointi on välttämätöntä jatkuvasti esimerkiksi kotioloissa kuuntelevan puheentunnistusjärjestelmän tapauksessa. Muuten satunnainen ihmisten välinen keskustelu, television äänet, radio tai muut vastaavat voisivat aktivoida komentoja, vaikka toivottua avainsanaa ei sanottaisi sellaisenaan. Testien perusteella väärien hälytysten laskeminen muutaman prosentin tasolle tarkoittaisi, että mallit hylkäisivät puhujasta riippuen lähes tai yli 50 % kaikista komennoista, jotka muuten olisivat tunnistuneet oikein. Puhujien välisten erojen takia ei voida valita yleistä raja-arvoa, vaan sopiva raja tulisi hakea yksilöllisesti kullekin puhujalle.

Vertailluista malleista MLP suoriutui heikoiten tunnistustarkkuuksien osalta, joten se hylättiin jo ensimmäisten testien perusteella. Tuloksesta tekee yllättävän sen poikkeavuus Lopezin ym. (2015) sekä Medhin ja Taulukdarin (2015) MLP:llä saavuttamista hyvistä tunnistustarkkuuksista. Kyseisten tutkimusten data ei ole vapaasti saatavilla, joten syitä erolle ei voitu tutkia tarkemmin. MLP ei ollut suuren parametrimääränsäkään takia houkutteleva vaihtoehto varsinkin, kun MatchboxNet osoitti hyvän tunnistustarkkuuden olevan saavutettavissa parametrimäärällä, joka on alle kymmenesosa MLP:n lukemasta. LIS-Netin parhaiten tunnistukseen soveltuneet rakenteet olivat parametrimääriltään vielä MLP:täkin suurempia, mutta jäivät silti tunnistustarkkuudessa MatchboxNetin taakse. Suuri parametrimäärä mahdollisesti edesauttoi ylioppimista, kun koulutusdataa oli vain vähän. MatchboxNet saavutti keskimäärin parhaat tunnistustarkkuudet. Se sisälsi vähiten parametreja, joten sen muistinkulutus on vähäisintä, ja se vaati LIS-Nettiä vähemmän liukulukulaskuoperaatiota, joten se on laskennallisesti vähemmän vaativa kuin LIS-Net. Lisäksi sen koulutus- ja tunnistusajat olivat LIS-Nettiä lyhyemmät. Sekä LIS-Net- että MatchboxNet-mallit saavuttivat keskimäärin parhaat tulokset käytettäessä 15 koulutusnäytettä jokaista komentoa kohden, mutta LIS-Netin tulok-

set olivat huonommat pienemmillä koulutusnäytemäärillä, joten MatchboxNettien voidaan katsoa vaativan vähemmän koulutusdataa.

Kaiken kaikkiaan MatchboxNet -mallit vastasivat parhaiten esitettyihin kriteereihin, mutta nekään eivät saavuttaneet tuloksia, joiden perusteella ne soveltuisivat hyvin jatkuvaan komentojen kuunteluun. Merkittävimmät ongelmakohdat olivat tunnistustarkkuuksien suuri vaihtelu puhujien välillä ja väärien hälytysten välttäminen heikentämättä merkittävästi komentojen tunnistuskykyä. Puhujien välistä vaihtelua voitaisiin vähentää ohjeistamalla puhujat muuntelemaan puhetyyliään mahdollisimman vähän äänitteitä tehdessä. Tällöin tietyllä tapaa lausutut komennot voisivat tunnistua varmemmin, mutta kääntöpuolena on, että hiemankin eri tavalla sanotut komennot eivät välttämättä tunnistu, mikä voi tuottaa ongelmia esimerkiksi puhuttaessa etäällä mikrofonista tai käheän kurkun kanssa. Mikrofonin tallennustarkkuuden vaikutus tunnistustarkkuuteen tulee myös selvittää. Väärien hälytysten välttämiseksi koulutukseen voitaisiin lisätä uusi luokka kaikkia komentoihin kuulumattomia ääniä varten. Kyseinen ratkaisu vaatii paljon dataa, sillä komentoihin kuulumattomia ääniä on paljon enemmän kuin komentoja. Komentoihin kuulumattomien äänien koulutusdataa voi siis olla moninkertainen määrä muihin luokkiin verrattuna, jolloin koulutusta tulee tasapainottaa, jottei malli opi tunnistamaan vain komentoihin kuulumattomia äänitteitä. Mahdollisia lähestymistapoja ovat muun muassa satunnaisten näytteiden monistaminen pienemmissä luokissa, niin että kaikista luokista tulee saman kokoisia, tai kouluttaessa äänitteiden kustannusten painottaminen sillä perusteella paljonko näytteitä kunkin äänitteen edustamaan luokkaan kuuluu (Cui ym. 2019, 1-2). Vähemmän edustetuille luokille annetaan suurempi paino esimerkiksi kertomalla kunkin äänitteen kustannus äänitteen luokan koon käänteisluvulla.

Mallien yleistyvyyttä voidaan yrittää parantaa data-augmentoinnin avulla (Majumdar ja Ginsburg 2020, 2). Esimerkiksi SpecAugment-augmentoinnilla mallin kykyä tunnistaa komennot pienistä aikamuutoksista, taajuuspuutteista ja komennon osien puuttumisesta huolimatta pyritään edistämään tekemällä koulutusdataan pieniä vääristymiä ajan suhteen sekä peittämällä satunnaisia taajuuksia ja aikajaksoja (Park ym. 2019, 1-2). Piirteenirrotusmetodillakin on vaikutus mallin suorituskykyyn, joten vertailu eri metodien ja parametrien suhteen, mitä ei tässä tutkimuksessa tehty eri yhdistelmien koulutukseen vaadittavan ajan takia, on mahdollinen kehityskohde. Tutkittavaksi voidaan ottaa muitakin neuroverkkorakenteita, kuin tässä

tutkimuksessa vertaillut, kuten Wein ym. 2020 tutkima neuroverkko, jossa on käytetty piirteenirrotuksessakin erilaista käsittelyä kuin tässä tutkimuksessa. Pienen datamäärän takia oppimisen siirrostakin voisi olla hyötyä. Esimerkiksi MatchboxNet voitaisiin kouluttaa aluksi Google Speech Commands -data-aineistoa käyttäen, jonka jälkeen MatchboxNetin palikat ja niitä edeltävät kerrokset voitaisiin jäädyttää, mikä mahdollistaisi viimeisten kerrosten uudelleenkoulutuksen kohdealueen datalla. Mikäli puhekomennontunnistusmalli saadaan käytännölliselle tasolle, tarvitaan hoivarobottiin myös helppo tapa luoda henkilökohtaiset äänitteet, joilla malli koulutetaan kullekin henkilölle sopivaksi. Robotissa voisi olla esimerkiksi graafinen käyttöliittymä, joka ohjaa komentojen valinnassa ja näytteiden äänityksessä sekä automatisoi mallin koulutuksen ja käyttöönoton.

Lähteet

Anh, N., Y. Hu, Q. He, T. Linh, H. Dung ja C. Guang. 2021. “LIS-Net: An end-to-end light interior search network for speech command recognition”. *Computer Speech & Language* 65:101131. https://doi.org/https://doi.org/10.1016/j.csl.2020.101131.

Anusuya, M. A., ja S. K. Katti. 2010. *Speech Recognition by Machine, A Review.* arXiv: 1001.2267 [cs.CL].

Bäckström, T., O. Räsänen, A. Zewoudie ja P. Zarazagaa. 2019. *Introduction to Speech Processing.* Viitattu 24. helmikuuta 2021. https://wiki.aalto.fi/display/ITSP/Introduction+to+Speech+Processing.

Chai, S., Z. Yang, C. Lv ja W. Zhang. 2019. “An End-to-End Model Based on TDNN-BiGRU for Keyword Spotting”. Teoksessa *2019 International Conference on Asian Language Processing (IALP),* 402–406. https://doi.org/10.1109/IALP48816.2019.9037714.

Chen, X., S. Yin, D. Song, P. Ouyang, L. Liu ja S. Wei. 2019. “Small-Footprint Keyword Spotting with Graph Convolutional Network”. Teoksessa *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU),* 539–546. https://doi.org/10.1109/ASRU46091.2019.9004005.

Cui, Y., M. Jia, T. Lin, Y. Song ja S. Belongie. 2019. *Class-Balanced Loss Based on Effective Number of Samples.* arXiv: 1901.05555 [cs.CV].

Fontaine, V., ja H. Bourlard. 1997. “Speaker-dependent speech recognition based on phone-like units models-application to voice dialling”. Teoksessa *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing,* 2:1527–1530. https://doi.org/10.1109/ICASSP.1997.596241.

Ginsburg, B., P. Castonguay, O. Hrinchuk, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, H. Nguyen, Y. Zhang ja J. M. Cohen. 2020. *Stochastic Gradient Methods with Layer-wise Adaptive Moments for Training of Deep Networks.* arXiv: 1905.11286 [cs.LG].

Glorot, X., A. Bordes ja Y. Bengio. 2011. “Deep Sparse Rectifier Neural Networks”. Teoksessa *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics,* toimittanut Geoffrey Gordon, David Dunson ja Miroslav Dudík, 15:315–323. Proceedings of Machine Learning Research. JMLR Workshop / Conference Proceedings. http://proceedings.mlr.press/v15/glorot11a.html.

Goodfellow, I., Y. Bengio ja A. Courville. 2016. *Deep Learning.* Http://www.deeplearningbook.org. MIT Press.

Gu, Y., Z. Du, H. Zhang ja X. Zhang. 2020. “An Efficient Joint Training Framework for Robust Small-Footprint Keyword Spotting”. Teoksessa *Neural Information Processing,* toimittanut Haiqin Yang, Kitsuchart Pasupa, Andrew Chi-Sing Leung, James T. Kwok, Jonathan H. Chan ja Irwin King, 12–23. Springer International Publishing. https://doi.org/10.1007/978-3-030-63830-6_2.

He, K., X. Zhang, S. Ren ja J. Sun. 2015. *Deep Residual Learning for Image Recognition.* arXiv: 1512.03385 [cs.CV].

Huang, X., A. Acero, H. Hon ja R. Reddy. 2001. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Prentice Hall PTR. https://dl.acm.org/doi/10.5555/560905.

Ioffe, S., ja C. Szegedy. 2015. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* arXiv: 1502.03167 [cs.LG].

Jurafsky, D., ja J. H. Martin. 2020. *Speech and Language Processing.* Viitattu 19. helmikuuta 2021. https://web.stanford.edu/~jurafsky/slp3/.

Karunanayake, Y., U. Thayasivam ja S. Ranathunga. 2019. “Transfer Learning Based Free-Form Speech Command Classification for Low-Resource Languages”. Teoksessa *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop,* 288–294. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-2040.

Kingma, D. P., ja J. Ba. 2017. *Adam: A Method for Stochastic Optimization.* arXiv: 1412.6980 [cs.LG].

Lin, M., Q. Chen ja S. Yan. 2014. *Network In Network.* arXiv: 1312.4400 [cs.NE].

Lopez-Meyer, P., H. Cordourier-Maruri, A. Quinto-Martinez ja O. Tickoo. 2015. “Analyzing Artificial Neural Networks and Dynamic Time Warping for spoken keyword recognition under transient noise conditions”. Teoksessa *2015 9th International Conference on Sensing Technology (ICST),* 274–277. https://doi.org/10.1109/ICSensT.2015.7438406.

Lu, Y., W. Shan ja J. Xu. 2019. “A Depthwise Separable Convolution Neural Network for Small-footprint Keyword Spotting Using Approximate MAC Unit and Streaming Convolution Reuse”. Teoksessa *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS),* 309–312. https://doi.org/10.1109/APCCAS47518.2019.8953096.

Lyons, J. 2013. “Mel Frequency Cepstral Coefficient (MFCC) tutorial”. Viitattu 1. maaliskuuta 2021. http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

Majumdar, S., ja B. Ginsburg. 2020. *MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition.* arXiv: 2004.08531 [eess.AS].

McMahan, B., ja D. Rao. 2017. *Listening to the World Improves Speech Command Recognition.* arXiv: 1710.08377 [cs.SD].

Medhi, B., ja P. H. Talukdar. 2015. “Isolated Assamese Speech Recognition using Artificial Neural Network”. Teoksessa *2015 International Symposium on Advanced Computing and Communication (ISACC),* 141–148. https://doi.org/10.1109/ISACC.2015.7377331.

Mittermaier, S., L. Kürzinger, B. Waschneck ja G. Rigoll. 2020. “Small-Footprint Keyword Spotting on Raw Audio Data with Sinc-Convolutions”. Teoksessa *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 7454–7458. https://doi.org/10.1109/ICASSP40776.2020.9053395.

Muda, L., M. Begam ja I. Elamvazuthi. 2010. *Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques.* arXiv: 1003.4083 [cs.MM].

Myers, C., L. Rabiner ja A. Rosenberg. 1980. “Performance tradeoffs in dynamic time warping algorithms for isolated word recognition”. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28 (6): 623–635. https://doi.org/10.1109/TASSP.1980.1163491.

Nassif, Ali Bou, Ismail Shahin, Imtinan Attili, Mohammad Azzeh ja Khaled Shaalan. 2019. “Speech Recognition Using Deep Neural Networks: A Systematic Review”. *IEEE Access* 7:19143–19165. https://doi.org/10.1109/ACCESS.2019.2896880.

Nielsen, M. A. 2015. *Neural Networks and Deep Learning.* Determination Press. Viitattu 18. helmikuuta 2021. http://neuralnetworksanddeeplearning.com/.

Park, D. S., W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk ja Q. V. Le. 2019. “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition”. *Interspeech 2019,* https://doi.org/10.21437/interspeech.2019-2680.

Pascanu, R., T. Mikolov ja Y. Bengio. 2013. *On the difficulty of training Recurrent Neural Networks.* arXiv: 1211.5063 [cs.LG].

Pervaiz, A., F. Hussain, H. Israr, M. A. Tahir, F. R. Raja, N. K. Baloch, F. Ishmanov ja Y. B. Zikria. 2020. “Incorporating Noise Robustness in Speech Command Recognition by Noise Augmentation of Training Data”. *Sensors* 20 (8). https://doi.org/10.3390/s20082326.

Rao, K. Sreenivasa, ja Manjunath K.E. 2017. *Speech Recognition Using Articulatory and Excitation Source Features.* Springer International Publishing. https://doi.org/10.1007/978-3-319-49220-9.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever ja R. Salakhutdinov. 2014. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. *Journal of Machine Learning Research* 15 (56): 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Sørensen, P. M., B. Epp ja T. May. 2020. “A depthwise separable convolutional neural network for keyword spotting on an embedded system”. *EURASIP Journal on Audio, Speech, and Music Processing* 2020 (1): 10. https://doi.org/10.1186/s13636-020-00176-2.

Tokusumi, T. 2020. *keras-flops.* Viitattu 19. toukokuuta 2021. https://github.com/tokusumi/keras-flops/tree/0.1.2.

Warden, P. 2018. *Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition.* arXiv: 1804.03209 [cs.CL].

Wei, Y., Z. Gong, S. Yang, K. Ye ja Y. Wen. 2020. “A New Lightweight CRNN Model for Keyword Spotting with Edge Computing Devices”. Teoksessa *Machine Learning for Cyber Security,* toimittanut Xiaofeng Chen, Hongyang Yan, Qiben Yan ja Xiangliang Zhang, 195–205. Springer International Publishing. https://doi.org/10.1007/978-3-030-62223-7_17.

Yang, S., Z. Gong, K. Ye, Y. Wei, Z. Huang ja Z. Huang. 2020. “EdgeRNN: A Compact Speech Recognition Network With Spatio-Temporal Features for Edge Computing”. *IEEE Access* 8:81468–81478. https://doi.org/10.1109/ACCESS.2020.2990974.

Yu, F., ja V. Koltun. 2016. *Multi-Scale Context Aggregation by Dilated Convolutions.* arXiv: 1511.07122 [cs.CV].

Zhao, Y., ja L. Zhu. 2017. “Speaker-Dependent Isolated-Word Speech Recognition System Based on Vector Quantization”. Teoksessa *2017 International Conference on Computer Network, Electronic and Automation (ICCNEA),* 133–137. https://doi.org/10.1109/ICCNEA.2017.103.