Tampereen yliopisto

Iiro Sundberg

RADIOAMATÖÖRILÄHETYKSIEN PUHEENTUNNISTUS

Informaatioteknologian ja viestinnän tiedekunta
Pro gradu -tutkielma
Huhtikuu 2023

TIIVISTELMÄ

Iiro Sundberg: Radioamatöörilähetyksien puheentunnistus
Pro gradu -tutkielma
Tampereen yliopisto
Laskennallisesti suurien tietoaineistojen analysoinnin tutkinto-ohjelma
Huhtikuu 2023

Viimeaikaiset edistysaskeleet tekoälyn käytössä (erityisesti syvät neuroverkot) ovat tehneet mahdolliseksi ns. päästä-päähän-puheentunnistusjärjestelmien suunnittelun ja rakentamisen. Aiemmat menetelmät perustuivat tutkijoiden tunnistamiin puheen toistuviin ominaisuuksiin kuten äänteisiin. Uudet menetelmät käyttävät tyypillisesti kielimallia korjaamaan tunnistettuja lauseita ja nämä on koulutettu häiriöttömällä ja selvillä äänillä, jotka on tallennettu hyvälaatuisella mikrofonilla.

Radioamatöörien puhelähetykset ovat kuitenkin usein kohinaisempia kuin esimerkiksi matkapuhelimen välityksellä käydyt (tallennetut) keskustelut, ja puhe saattaa häipyä radiotiellä olevien ilmastollisten ilmiöiden vuoksi hetkittäin lähes tai kokonaan kuulumattomiin. Lisäksi keskustelijoita voi olla samaan aikaan useita (yhtäaikaisia lähetyksiä) ja radiovastaanottimesta kuulua useita ääniä yhtä aikaa. Tässä opinnäytetyössä tutkitaan nykyaikaisten puheentunnistusmenetelmien toimintaa radioamatöörien puhelähetysten litteroinnissa tekstiksi. Työ aloitettiin tutustumalla kirjallisuuteen ja erilaisiin puheentunnistusmenetelmiin sekä etsittiin Internetistä litteroituja puhemateriaaleja ja esikoulutettuja neuroverkkoja. Tällä hetkellä on vain muutamia avoimesti käytettävissä olevia koulutettuja suomen kieltä osaavia neuroverkkoja. Muille kielille näitä kuitenkin löytyy ja näiden uudelleen kouluttamista suomenkielisellä materiaalilla tutkittiin.

Koulutusmateriaaliksi kerättiin 2021–22 aikana Internetiin liitetyllä ohjelmistoradiolla noin 100 tunnin aineisto. Tästä aineistosta litteroitiin käsin noin tunnin mittainen opetus- ja kahden tunnin todennusaineisto, jolla vertailtiin eri tunnistimien suorituskykyä. Lopputuloksena Microsoft Office 365 -puheentunnistuksella saavutettiin 64,17 %, Googlen SpeechRecognition-Python-kirjastolla 78,30 %, ja itsekoulutetuilla QuartzNet ja Conformer-CTC Large -puheentunnistusmalleilla 99,99 % ja 100,14 % sanavirhesuhteet. Työn lopputuloksena voidaan päätellä, että suurten materiaalien litteroimiseksi tehokkaasti tarvitaan suuri joukko henkilöitä ja että pieni koulutusmateriaalimäärä johtaa helposti ylisovittumiseen. Tämä puolestaan haittaa ennen kuulemattoman puheen tunnistusta, koska se ei vastaa opetusaineistoa.

Avainsanat: Puheentunnistus, radioamatööri, QuartzNet, Conformer

ABSTRACT

Iiro Sundberg: Amateur radio transmissions analyzed with automatic speech recognition
Pro gradu -thesis
University of Tampere
Computational Big Data Analytics
April 2023

Recent advances in artificial intelligence (especially deep neural networks) have made it possible for designing and building automatic speech recognition systems which can be trained end to end with transcribed speech data. In comparison with earlier methods which utilized the repeatable characteristics of the speech - for example - phonemes, these new methods use a language model to correct the results of the recognized sentences. Typically, these systems are trained with non-disturbed and clear voices recorded with a good quality microphone.

However, the speech transmissions by the amateur radio enthusiasts are typically noisier than, e.g., speech transmitted by the mobile phones. In addition, the speech may temporarily fade away completely due to atmospheric phenomena. Also, several transmissions can exist simultaneously and thus several different voices heard in chorus with a radio receiver. This thesis concentrates on using the modern automatic speech recognition methods for transcribing the noisy amateur radio speech to text. The thesis work was started by studying literature and different speech recognition methods and searching the web for transcribed speech sets and pre-trained neural networks. Currently, only a few openly available trained neural networks exist for Finnish language. However, there are several networks available for other languages. These are studied as a starting point for transfer learning, i.e., retrain the neural network with Finnish language material.

To collect the radio amateur speech for training of the neural networks over 100 hour of transmissions were recorded during the 2021-22 with a websoftwareradio and an hour of these was transcribed by hand for the training material of the neural networks and two hours was transcribed for validation material which was used to compare different speech recognition systems. The results of the comparison were that Microsoft speech recognition engine could achieve word error rate of 64,17 %, Google speech recognition engine 78,30 %, and own implementations with QuartzNet and Conformer-CTC Large achieved 99,99 % and 100,14 %. It can be concluded that transcribing such a large material requires several persons to be efficient and that the small amount of teaching data results easily to over-fitting which in turn leads to poor generalization (high word error rate for unheard speech).

Key words: Automatic Speech Recognition, Amateur radio, QuartzNet, Conformer

Sisällysluettelo

Esipuhe

Tämän opinnäytetyön aikana tutustuin uusiin aloihin: laskennallisesti suurten tietomassojen käsittelyyn, neuroverkkoihin, puheenmuodostukseen, ja puheentunnistukseen em. neuroverkoilla. Koin oppineeni paljon suurten tietomassojen käsittelyyn liittyvistä haasteista ja erityisesti puheentunnistuksesta. Työn tekemisen aikana ala eteni vauhdikkaasti ja kaksi vuotta kului opiskellessa ennen kuin aloin ymmärtämään riittävästi puheentunnistuksesta pystyäkseni seuraamaan viimeisimpiä artikkeleita.

Työn idea lähti halusta luoda radioamatöörilähetysten puheentunnistusjärjestelmä suomalaisten radioamatöörien kiinnostavien keskusteluiden seuraamiseen jälkikäteen. Näiden keskusteluiden erityisenä vaikeutena on vastaanotetun lähetyksen kohinaisuus sekä radioamatöörien käyttämä erikoissanasto.

Ajankäytöllisesti työ oli haastava kokonaisuus, koska jouduin opiskelemaan puheen muodostuksen teorian ja neuroverkot. Osoittautui, että useat tutkimukset käsittelevät tunnettujen aineistojen käyttöä uuden algoritmin suorituskyvyn arviointiin. Lopulta käytännön puheentunnistustulos ratkaisee ja mielestäni monissa tutkimuksissa käytetty koulutusaineisto on verrattain pieni ongelman haasteellisuuteen nähden.

Kiitokset ohjaajalleni professori Tuomas Virtaselle työtä eteenpäin vieneistä kommenteista ja professori Jaakko Peltoselle etenkin teknisen osan ohjaamisesta seminaaritapaamisissa sekä Mika Anttilalle Narvi-klusterin käyttöavustas. Lisäksi jään kiitollisuuden velkaan perheelleni ja aivan erityisesti vaimolleni Iinalle tämän kirjan valmistumisen odottamisesta ja tuesta. Ilman sitä tämä ei olisi valmistunut. Useiden poikani päiväunien ajan, useat illat töiden jälkeen ja automatkat menivät tätä työtä valmistellessa, artikkeleita lukiessa, radioamatöörilähetyksiä litteroidessa sekä laskentapalvelinta ohjatessa.

73:t de Iiro, OH2GXM

1 Johdanto

Ihmiset ovat kuvitelleet jo antiikin mytologiassa koneen (pronssinen Talos-robotti), joka ymmärtäisi ihmisen puheen ja osaisi toimia sen merkityksen perusteella [1]. Kesti kuitenkin 1980-luvulle asti ennen kuin tietokoneiden laskentakyky ja puheentunnistusalgoritmit (erityisesti Markovin piilomalli) kehittyivät tasolle, jolla sanojen tunnistaminen puheesta olisi mahdollista [2].

Ensimmäinen onnistunut karkea puheentunnistusjärjestelmä, Bellin laboratorioiden täysin analoginen Audrey vuodelta 1952 kykeni tunnistamaan yhden puhujan puheesta numeroita foneemien perusteella, ks. lähteet [3] ja [4]. Myöhemmin 1970-luvulla puheentunnistusjärjestelmät (Hearsay-I ja Dragon) perustuivat erilaisten äänteiden tunnistamiseen ja tilastolliseen mallinnukseen [5]. Näiden järjestelmien pohjalta kehitettiin DARPA:n tutkimusohjelmassa HARPY, joka kykeni tunnistamaan kokonaisia lauseita ja 1011 sanaa [5]. Nämäkin järjestelmät kuitenkin toimivat parhaiten tietyn tyyppisille puhujille, äänille ja sanoille.

Perinteiset puheentunnistusmenetelmät ovat perustuneet kieli-, äänne- ja akustisen mallien käyttämiseen. Ensin kielimallin perusteella on muodostettu todennäköisimmät sanajärjestykset (esimerkiksi käyttäen n-grammeja, ks. luku 5.1), sitten äännemallista haettiin sanojen ääntämys ja lopuksi akustinen malli muodosti äänteistä ääniaaltomuodot (esimerkiksi Gaussian Mixture Model, GMM, ks. yksityiskohdat esimerkiksi [6]). Tämän jälkeen maksimoitiin mallin antaman arvauksen ja puhetallenteen todennäköisyys $P(\text{audio}|\text{transkriptio}) \cdot P(\text{transkriptio})$ ja etsittiin sellaiset sanat, jolla todennäköisyys maksimoitui (argumentin arvot, joilla saatiin suurin arvo) [7].

Kun tietokoneiden laskentatehot alkoivat kasvaa merkittävästi 1990-luvulta lähtien (ns. Mooren lain [8] mukaisesti eli samalle pinta-alalle mahtuvien prosessorien transistorien määrän kaksinkertaistuessa noin kahden vuoden välein) ja markkinoille tuli suorituskykyisiä rinnakkaiseen laskentaan kykeneviä laskentakortteja (ns. Graphical Processing Unit, GPU) 2000-luvun alkupuolella, ks. lähde [9], niin myös sellaisten neuroverkkojen koulutus mahdollistui suurilla puhemäärillä, joilla kyetään kuvaamaan monimutkaisia funktioita (esimerkiksi useiden eri puhujien puhesignaalit) ”riittävän” tarkasti [10].

Viime vuosien neuroverkkojen kehitys on mahdollistanut kokonaisten sanojen ja lauseiden opettamisen tietokoneelle eli ns. päästä-päähän-menetelmän (eng. end-to-end) hyödyntämisen. Nämä puheentunnistusalgoritmit ovat laskennallisesti monimutkaisia, koska niillä kyetään mallintamaan useiden erilaisten (miehet, naiset, lapset, jne.) puhujien ääniaaltomuotoja. Niiden suorituskyky on rajoittunutta ja sitä voidaan parantaa tallentamalla hyvälaatuista puhetta sekä litteroimalla se opetusaineistoksi neuroverkolle. Tämä vie hyvin paljon aikaa litteroinnin suorittavalta henkilöltä.

Teksti on usein nopeampi lukea ja ymmärtää kuin katsella kokonainen video tai kuunnella useampi puhe. Suomenkielistä puhetta ymmärtävä puheentunnistus on tarpeellista mm. lääkärien potilaskertomuksien litteroinnissa. Lisäksi kuuloltaan vammautuneiden tai heikentyneet ihmiset hyötyvät esimerkiksi keskusteluohjelmien tekstityksestä.

Puheella on myös mahdollista ohjata esimerkiksi kotiautomaatiota, jossa näppäimistön sijoittelu on rajattu esimerkiksi tiettyyn huoneeseen tai älypuhelinta, jonka näppäimistö ei voi olla fyysisesti kovin suuri. Ensimmäinen keinotekoinen henkilökohtainen assistentti Siri kehitettiin vuonna 2007 ja Apple julkaisi sen puhelimilleen 2011 [11]. Vastaavasti kotiautomaatiota (esimerkiksi valoja ja musiikkisoitinta) ohjaava Amazon Alexa julkaistiin marraskuun 2014 alussa [12].

Kirjoittajan kiinnostus radioamatöörien keskusteluiden litterointiin tekstiksi toimi tämän pro gradu -tutkielman motiivina. Tutkielman tutkimuskysymykseksi muodostui millä puheentunnistusalgoritmilla tai -menetelmällä tämä olisi mahdollista toteuttaa opetusaineiston rajallisuuden vuoksi ja voisiko yhdistämällä useampia menetelmiä tai painottamalla niiden tuloksia saavuttaa parempaa suorituskykyä. Lisäksi selvitettiin puheentunnistusmenetelmien koulutusvaatimuksia eli vaaditaanko grafiikkakiihdytinkortteja vai riittävätkö suorituskyvyltään vähäisemmät laitteet.

Tutkimuksessa kerättiin noin 100 tunnin tallenteet suomenkielistä radioamatööripuhetta. Tästä aineistosta ehdittiin litteroida tekstiksi opetusaineistoksi noin tunnin verran eri puhujien puheenvuoroja ja todennusaineistoksi 121 minuuttia. Tämä on vähän verrattuna kehittyneimpiin puheentunnistusjärjestelmiin, joiden opetusaineistona saattaa olla satoja tuhansia tai miljoonia tunteja puhetta eri puhujilta [13].

Opinnäytetyössä tutkitaan konvoluutionaalisen neuroverkon (esimerkiksi äänisignaalin spektrogrammi-kuvaajaan perustuva) [14] ja konnektionistisen temporaalisen luokittelija-algoritmin (eng. Connectionist Temporal Classification) [15] yhdistämistä puheentunnistusalgoritmissa sekä ns. Transformer-mallin [16] soveltuvuutta radioamatööripuheen muuntamiseen tekstiksi. Yhteistä näille menetelmille on se, että ne käsittelevät peräkkäisiä tapahtumia, jotka voi olla riippuvia toisistaan, esimerkiksi tietyt merkit tai sanat seuraavat toisiaan.

Tutkielmassa perehdytään ensin puheen syntymiseen ja luodaan lyhyt katsaus puhesignaalin ominaisuuksiin sekä puheentunnistuksen teoriaan luvussa 2. Tämän jälkeen esitellään yleisesti koneoppimista ja erityisesti neuroverkkojen toimintaa matalalla tasolla luvussa 3. Nämä neuroverkkoteoriat ja -menetelmät esitellään lyhyesti, jotta niihin perustuvat puheentunnistusneuroverkkorakenteet voisi ymmärtää. Työssä käytettyjä erilaisia moderneja päästä-päähän-puheentunnistusneuroverkkorakenteita avataan luvussa 4. Osa puheentunnistusjärjestelmistä hyödyntää tunnistuksessa kielimalleja täydentämään sanojen tunnistusta. Näiden toimintaa selostetaan luvussa 5 vain lyhyesti, sillä työssä ei tutkita näiden toimintaa.

Testijärjestely soveltuvien radioamatööripuhelähetysten tallentamiseksi, litteroimiseksi ja työssä käytettyjen neuroverkkojen kouluttamiseksi tehdyt asetukset on kuvattu luvussa 6. Neuroverkkojen oppimistulokset käyttäen eduskunnan puheaineistoa ja tallennettuja radioamatööripuhelähetteitä on esitelty luvussa 7. Tutkimuksen yhteenveto on luvussa 8 ja lähteinä käytetty viitemateriaali listattu luvussa 9.

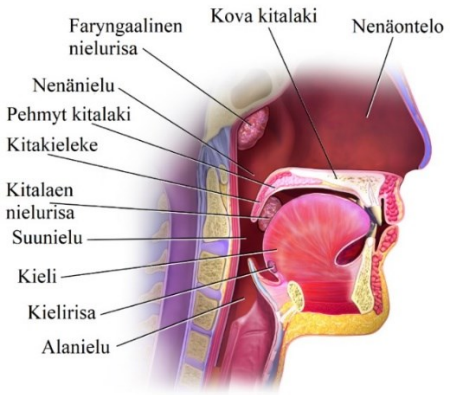
2 Puheentunnistuksen teoriaa

Tässä luvussa käsitellään lyhyesti mikrofonilla tallennetun puhesignaalin ominaisuuksia myöhemmin esiteltävän puheentunnistuksessa käytettävän koneoppimisen ymmärtämiseksi. Puheenmuodostus on laaja tutkimusalue ja ensimmäiset kehitetyt puheentunnistusmallit perustuivatkin juuri puheen syntymisen ymmärtämiseen sekä äänen muodostuksessa käytettävien elinten mallinnukseen, esimerkiksi miten ääntöväylän pituus vaikuttaa äänenkorkeuteen. Atte Virtanen on käsitellyt diplomityössään [17] äänen muodostusta sekä puhujalle ominaisen perustaajuuden tunnistusta neuroverkoilla tallennetusta puhelinpuheesta. Aalto-yliopiston verkkomateriaali [18] tarjoaa hyvät perusteet puhesignaalin muodostuksen ymmärtämiselle ja esittelee aiemmin puheentunnistukseen kehitetyt menetelmät (ennen laajamittaista neuroverkkojen hyödyntämistä). Puheentutkimuksen suomenkielisenä sanastona on käytetty Helsingin yliopiston Fonetiikan perussanastoa [19].

Tässä luvussa esitellään kirjoittajan tallentamin esimerkein puhesignaalin ominaisuuksia ja luodaan pohja myöhemmin työssä esiteltävien neuroverkkototeutuksien yksityiskohtien ymmärtämiselle. Lisäksi sanojen tunnistukseen liittyviä haasteita ja sanojen tunnistuksen onnistumisen mittarit esitellään.

2.1 Puhesignaalin ominaisuudet

Puhesignaali muodostuu, kun keuhkoista tulevaa ilmavirtaa muokataan (moduloidaan) värisyttämällä äänihuulia eli aiheuttamalla paineen vaihtelua. Tästä syystä puhesignaali on jaksollinen ja siinä on erotettavissa useita taajuuskomponentteja. Puhesignaalin perustaajuus muodostuu äänihuulten pituussuuntaisesta jännityksestä, paine-erosta äänihuulten ylä- ja alapuolella sekä äänihuulten massasta. Ihminen pystyy muuttamaan äänenkorkeutta painetta ja äänihuulten jännitystä säätelemällä, mutta ei äänihuulten fyysistä kokoa. Näin ollen puhujan (äänihuulten) fyysinen koko korreloi myös äänen korkeuden kanssa eli miesten ääni on tyypillisesti matalampi kuin naisten ja lasten. [18]

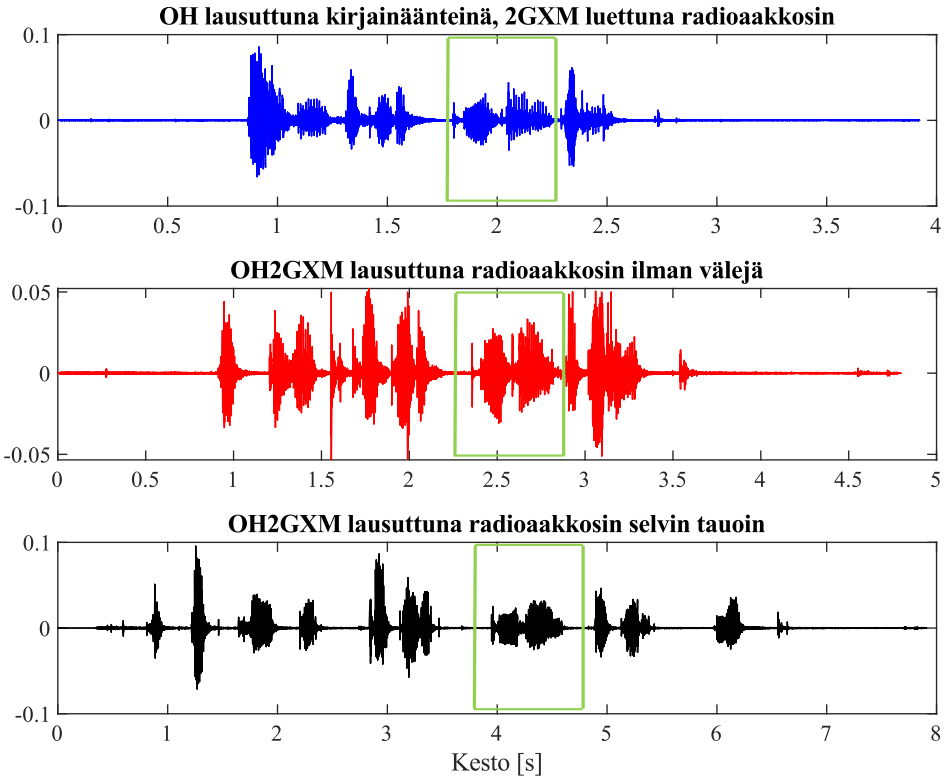


Kuva 1 Ihmisen puhe-elimet (nielurisat ja kurkku) lähteen [20] mukaan (CC-lisenssi) suomeksi nimettynä lähteen [19] mukaan.

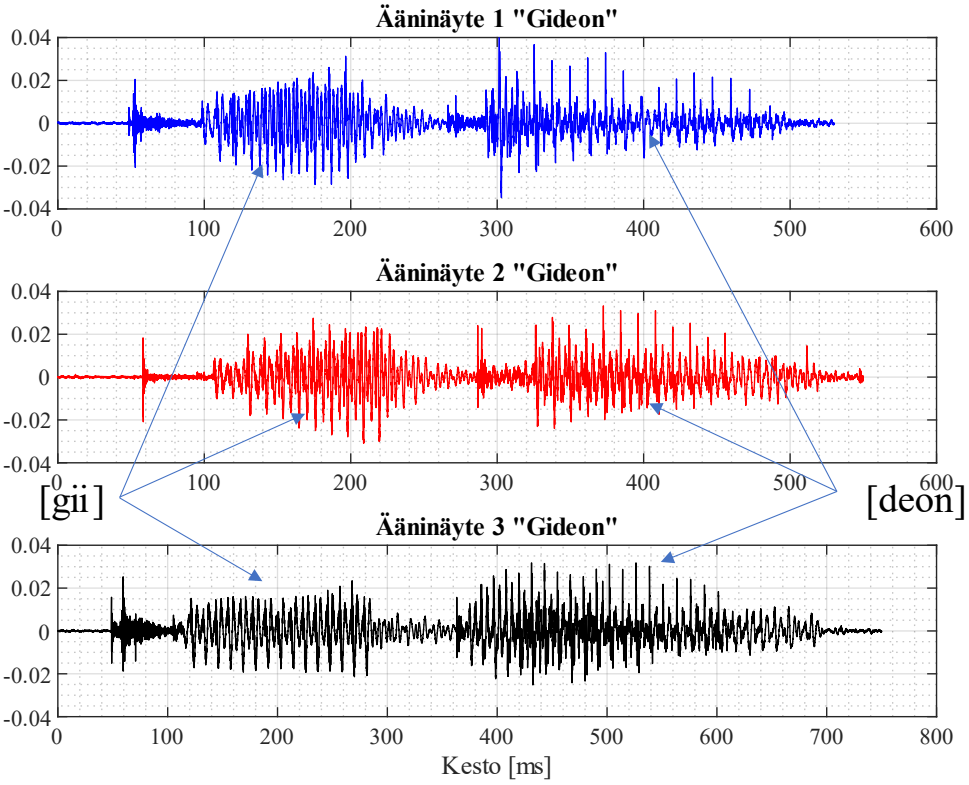
Vokaaliäänteet muodostuvat ääntöväylään (kurkunpää, nielun ja nenän ontelot, ks. kuva 1) aiheutetuista resonansseista, kun taas konsonanttiäänteet vaativat ääntöväylän osittaisen tai kokonaisen estymisen [18]. Tässä työssä ei tarkastella laajemmin puheen mallinnukseen kehitettyjä malleja, vaan painopiste on opetella suoraan tunnistamaan neuroverkolla sanoja.

2.2 Mikrofonilla tallennettu puhesignaali

Puhesignaali muodostuu paineen vaihteluista ja se voidaan tallentaa mikrofonilla, joka muuttaa nämä ilmavirran paineenvaihtelut sähköiseksi signaaliksi. Kuvassa 2 on esitetty kannettavan tietokoneen mikrofonilla tallennetut radioamatöörikutsun OH2GXM-puhesignaalit luettuna (ensin sininen käyrä: ”oo-hoo-kaksi-gideon-äksä-matti, sitten punainen käyrä: ”otto-heikki-kaksi-gideon-äksä-matti” ja lopuksi musta käyrä: ”otto heikki kaksi gideon äksä matti”) ajan suhteen. Voidaan havaita, että vaikka signaalit on puhuttu eri nopeudella, niin niiden muodoissa on yhtäläisyyksiä. Erityisesti loppuosan sanojen aaltomuodot ovat perusmuodoltaan samankaltaisia, vrt. vihreällä suorakaiteella reunustettu Gideon-sana, jonka mikrofonin tallentama aaltomuoto on suurennettuna kuvassa 3.



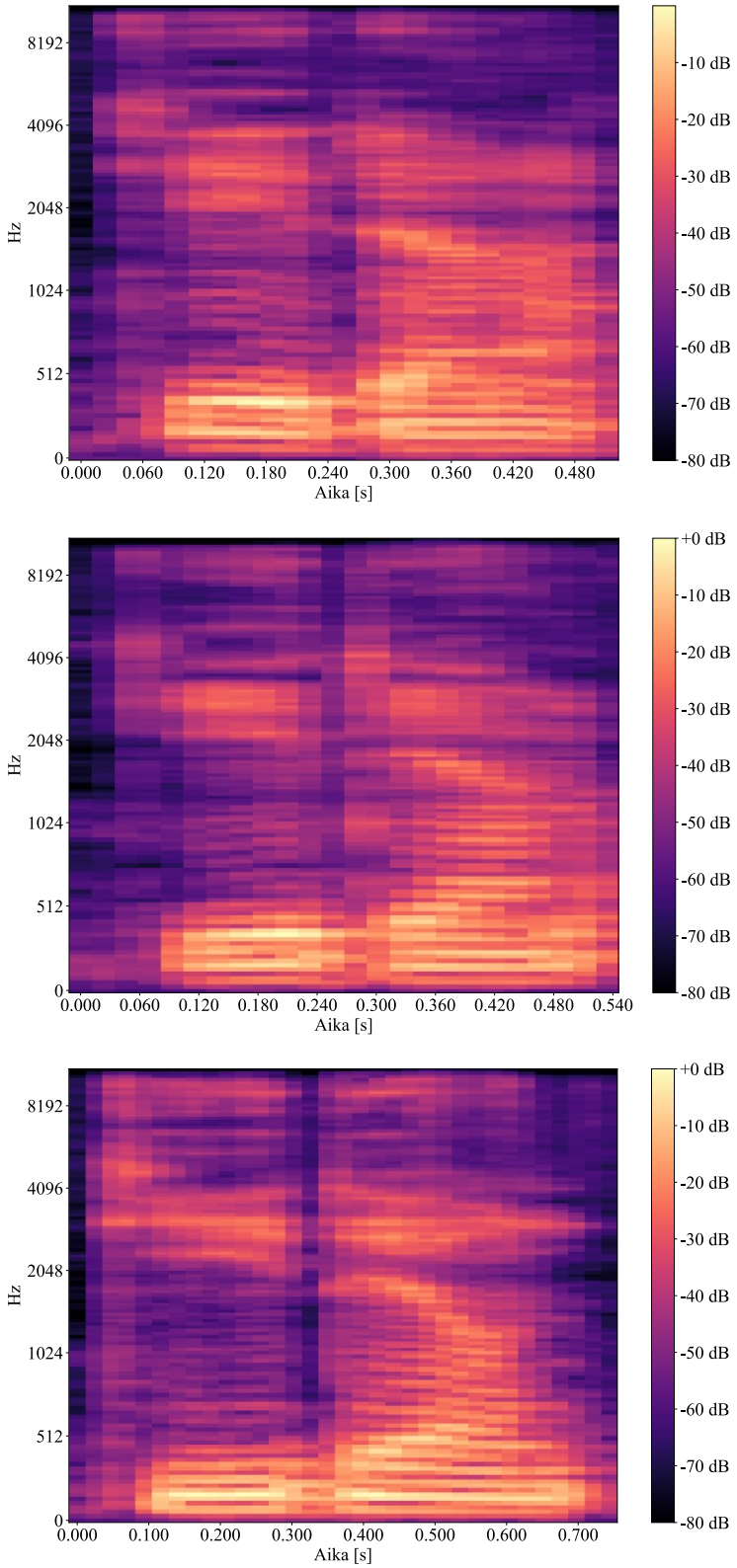
Kuva 2 Saman puhujan lausuma radioamatööriasematunnus OH2GXM kolmella poikkeavalla tavalla (”oohoo-kaksi-gideon-äksä-matti”, ”ottoheikkikaksigideonäksämatti”, ja ”otto heikki kaksi gideon äksä matti”). Tallenteiden loppu on identtinen, mutta sanan ”Gideon” -aaltomuodot (ympäröity vihreillä suorakulmioilla) ajan suhteen poikkeavat toisistaan hieman etenkin amplitudiltaan.



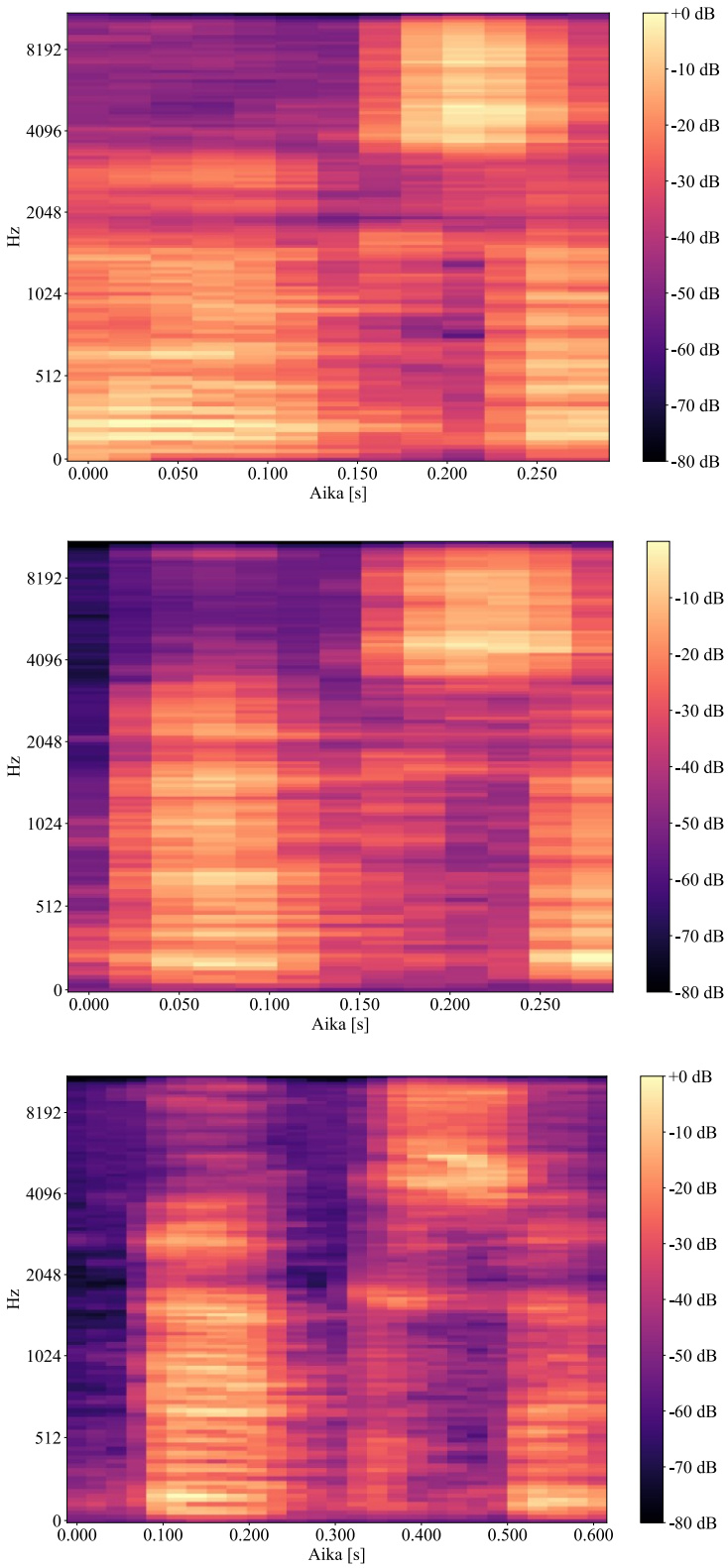
Kuva 3 Sanan ”Gideon” jänniteaaltomuodot ajan suhteen (sekunteja). Aaltomuodot muistuttavat toisiaan, mutta niissä on myös eroavaisuuksia.

Sanat voidaan tunnistaa neuroverkolla näiden jänniteaaltomuotojen perusteella (jännitevektori ajan suhteen), mutta valokuvien kuvankäsittelyyn kehitetyt konvolutiiviset neuroverkot toimivat matriisimuotoisen datan (kuvan pikseleiden arvo) kanssa. Näin ollen alun perin puhesignaalin taajuussisällön tarkasteluun kehitetyt spektrogrammiesitykset (aika vs. eri taajuuksilla oleva energia) voidaan syöttää neuroverkolle äänteiden ja niistä sanojen tunnistamiseksi. Esimerkiksi edellä olevan Gideon-sanan energia muodostaa tunnistettavan muodon, ks. Mel-spektrogrammiesitys kuvassa 4. Vertailun vuoksi sanan ”äksä” Mel-spektrogrammiesityksessä energiaa on korkeilla taajuuksilla aina 20 kHz asti, ks. kuva 5.

Kuvien tunnistamiseen alun perin kehitetty konvolutiivinen neuroverkko voidaan opettaa tunnistamaan spektrogrammiesityksien perusteella eri sanat (ja äänteet). Tämä vaatii laajan opetusaineiston paitsi puhuttuja eri sanoja, niin myös eri puhujien puhumina. Valtakielille, kuten englanti, on olemassa vapaasti saatavilla olevia tietokantoja (vrt. Mozilla Common Voice [21]), mutta pienempien kielialueiden kielten tallennettua puhuttua kieltä on vaikeammin saatavilla. Esimerkiksi Yleisradio on kampanjoinut puhutun suomen kielen keräämiseksi puheentunnistusmallien kehittämisen mahdollistamiseksi tai helpottamiseksi [22].



Kuva 4 Mel-spektrogrammikuvaajat ääninäytteiden ”Gideon”-sanasta. Pääosa energiasta on keskittynyt matalille, alle 1 kHz taajuuksille ja kuvaajissa on yhdenmukaisuutta keskenään.



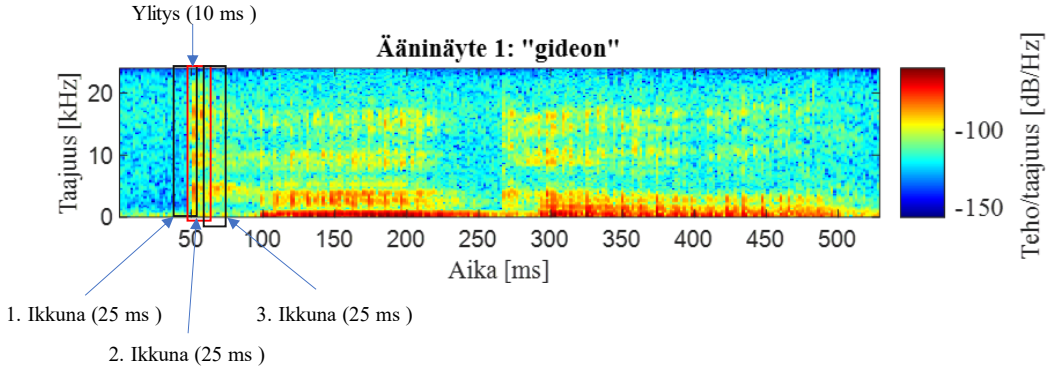
Kuva 5 Mel-spektrogrammikuvaajat ääninäytteiden ”äksä”-sanasta. Sanojen kesto vaihtelee (ylimmässä kuvaajassa lyhin), mutta etenkin s-kirjaimen energiaa keskittyy korkeammille taajuuksille (> 10 kHz) aiempaan ”Gideon”-sanaan (< 10 kHz) verrattuna.

2.3 Sanojen tunnistaminen

Puhe on ajallisesti jatkuvaa signaalia, jossa äänteet, joista sanat muodostuvat seuraavat toisiaan. Eri puhujilla ja eri murteissa eri äänteet voivat kestää eri mittaisia aikoja. Tästä johtuen on vaikeaa laatia tarkkaa yleistä algoritmia, joka osaisi muuntaa jokaisen puhujan puheen ilman koulutusmateriaalia ko. puhujalta. Puheentunnistaminen on tietokoneohjelmalle haastava ongelma. Puhe on paineenvaihtelua, joka on muunnettava tietokoneen ymmärtämään muotoon eli digitaaliseksi ennen kuin sitä voidaan analysoida. Puheentunnistusjärjestelmien voidaan ajatella koostuvan koodaimesta, jolla puheesta poimitaan tunnistettavat akustiset piirteet ja koodin purkajasta, jolla sitten näistä piirteistä tunnistetaan sanat [23].

2.3.1 Puheen näytteistys ja signaalinkäsittely

Puheentunnistusmenetelmät käsittelevät mikrofonilla digitoitua äänisignaalia paloissa eli 25–40 ms mittaiseen kehykseen (tai ikkunaan) otetaan kerrallaan esimerkiksi 10 ms puhesignaalia käsiteltäväksi. Näitä kehyksiä syntyy edellä esiteltyjen 600 ms pituisista tallenteista yhteensä noin 50 kpl (riippuen kehyksen pituudesta ja ylimenevästä osuudesta), ks. kolme ensimmäistä kehystä piirrettynä kuvassa 6.



Kuva 6 Ensimmäisen ääninäytteen kolme ensimmäistä kehystä (keskimmäinen punainen suorakaide, reunoilla mustat) havainnollistettuna sekä kehysten päällekkäinen ylitys.

Kehykseen osuville näytepisteille tehdään Fourier-muunnos, jolla saadaan ratkaistua signaalin muodostavat siniaallot (ja niiden taajuudet eli spektri). Koska Fourier-muunnos lasketaan rajallisesta määrästä näytepisteitä, niin spektriin syntyy näytteistyksestä johtuvia harhasignaaleja (tai energiaa esiintyy harhataajuuksilla verrattuna todelliseen signaalin taajuuteen), mikäli jokaista näytepistettä käytetään sellaisenaan painottamattomana, niin ikkunointia kutsutaan suorakaideikkunaksi (eng. rectangle window). Näitä harhasignaaleja voidaan vähentää pienentämällä kehyksen alussa ja lopussa olevien näytepisteiden vaikutusta ns. ikkunoinnilla (esim. kirjallisuudessa käytettyjä ikkunointifunktioita ovat mm. Hamming ja Hann) [24]. Erityyppisillä ikkunointifunktioilla voidaan optimoida esimerkiksi signaalikohinasuhdetta tai signaalien erottelua taajuustasossa.

2.3.2 Mel-kepstrikertoimet

Edellä kuvatut raakaspektriesitykset poikkeavat jokaiselle puhujalle (eri äänenkorkeus ja muut ominaisuudet esimerkiksi harmonisten taajuuskomponenttien määrä). Puhesignaalista voidaan löytää puhujariippumattomia ominaisuuksia, esimerkiksi ratkaisemalla ns. Mel-kepstrikertoimet (eng. Mel-frequency Cepstral coefficient), jotka poistavat harmoniset taajuudet spektristä eli puhujan äänenkorkeuden ja jäljelle jää esimerkiksi vokaaleille ominaiset tunnistettavat taajuuskomponentit.

Laskemalla edellä kuvatuista raakaspektriesityksistä logaritmi voidaan löytää toistuvia ns. harmonisia taajuuksia (log-spektri korostaa näitä) [18], koska ihmiset aistivat äänenvoimakkuuden ns. epälineaarisella mitta-asteikolla eli esimerkiksi äänenvoimakkuuden kaksinkertaistuessa havaitaan ero (toisin kuin äänien lineaarisissa voimakkuuseroissa). Useissa puheentunnistusmenetelmissä tehdään tämän jälkeen diskreetti Fourier-muunnos tai diskreetti kosinimuunnos log-spektrille ja saadaan ns. kepstri (eng. cepstrum). Tämän jälkeen saadut taajuudet muunnetaan Mel-asteikolle valittujen taajuustasossa kolmiomuotoisten suodattimien avulla seuraavalla kaavalla (1) lähteen [24] mukaan (alkuperäinen lähde [25] ja muokattu skaala esitetty lähteessä [26]):

$$\mathrm{mel}(f) = 1127 \ln\left(1 + \frac{700}{f}\right), \qquad (1)$$

jossa f on taajuus Hertzeinä ja ln() luonnollinen logaritmi.

Näin saadaan Mel-painotettu spektri, jossa äänen muodostavien harmonisten taajuuskomponenttien hienorakenne on suodattunut pois ja jäljelle on jäänyt perustaajuiset taajuuskomponentit eli ääntä kuvaava spektrogrammikuvaaja on yksinkertaistunut tai ns. yleistynyt kuvaamaan eri puhujien samankaltaisia äänteitä [18]. Näitä kertoimia käytetään neuroverkkojen piirrevektoreina, joiden avulla verkot koulutetaan tunnistamaan äänteet ja sanat.

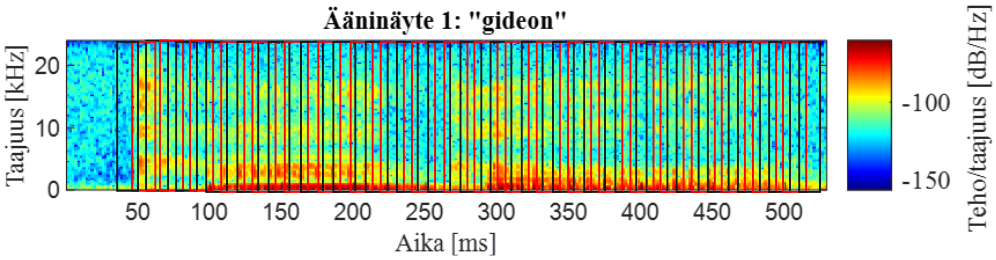
2.3.3 Merkkien tunnistus ja sanojen muodostus

Kun puhesignaali on digitoitu ja sen Mel-kertoimet on ratkaistu jokaiselle kehykselle, niin voidaan ratkaista mitä merkkiä (tai kirjainta) tämä kehys vastaa (tai sitä edustavat Mel-kertoimet vastaavat). Haasteena on etenkin se, että kehyksiä on huomattavasti enemmän kuin lopullisia kirjaimia puhutussa sanassa. Luvun 2.3.1 esimerkissä (ks. kuva 6) kehyksiä oli yhteensä noin 50 kappaletta ja niistä pääteltäviä kirjaimia yhteensä kuusi kappaletta (”gideon”). Tätä vaihetta voidaan pitää koodaimena (eng. encoder) eli akustisena mallina.

Kun puhesignaalin kehyksistä lasketut piirrevektorit syötetään esimerkiksi takaisinkytketylle neuroverkolle (eng. Recurrent Neural Networks, RNN, ks. yksityiskohdat luvussa 3.4), se laskee eri merkkien todennäköisyydet jokaiselle kehykselle. Tämän jälkeen konnektionistista aikajaottelu -algoritmia (eng. Connectionist Temporal Classification,

ks. yksityiskohdat luvussa 4.1) voidaan käyttää tunnistamaan sopivat merkit, merkkien määrä ja tauot sekä mahdollisesti tunnistamattomat merkit [15].

Näin kyetään yhdistämään tunnistetut peräkkäiset kirjaimet yhdeksi ja toisaalta erottamaan, mikäli onkin peräkkäin kaksi samaa kirjainta, joita ei tule yhdistää, ks. havainnollistava esimerkki kuvassa 7. Tämän jälkeen lasketaan todennäköisyydet kaikille mahdollisille yhdistelmille, jotka em. merkkijonoista voisi muodostua. Tätä vaihetta puolestaan voidaan pitää koodin purkamisena (eng. decoder).

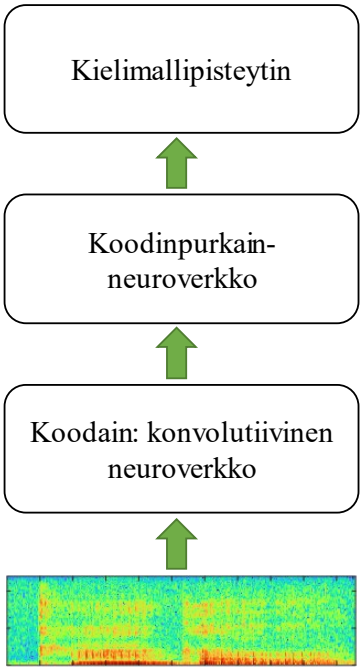


1: є g g g є i i i i i i i i i i i i є є є d d d є e e e e e e e e є є o o o є n n є

2: g i d e o n

3: g i d e o n

Kuva 7 Puhesignaalin jakaminen kehyksiin ja kehysten äänteen tunnistaminen (1) sekä tunnistettujen merkkien yhdistäminen (2) ja lopullinen tunnistettu sana (3). Huom. esimerkki on luotu havainnollistamiseen eikä vastaa todellista algoritmin tulosta.



Kuva 8 Useiden kirjoitushetkellä käytössä olevien puheentunnistusalgoritmien ylätason rakenne koostuu Mel-kepstrikertoimien avulla tapahtuvasta merkkien/äänteiden tunnistuksesta (opittu äännemalli) ja näiden perusteella koodinpurkaimessa muodostetuista sanoista (opittu kielimalli). Lisäksi voidaan käyttää kielimallipisteytintä korjaamaan tunnistettuja sanoja kielimallin mukaiseksi osaksi lausetta.

Nykyiset puheentunnistusmenetelmät oppivat tunnistamaan (koodaamaan) äänteet konvolutiivisten neuroverkkojen avulla Mel-skeptrikertoimia esittävistä kuvaajista. Näiden kertoimien avulla koodinpurkainneuroverkko laskee todennäköisyydet eri merkeille. Lopuksi merkeistä muodostetaan sanat ja tässä voidaan käyttää kielimallipisteytintä painottamaan todennäköisimpiä sanoja ja lauseita, ks. ylätason rakenne kuvassa 8.

Puheentunnistusmenetelmiä, jotka oppivat opetusaineistosta sekä akustisen mallin puhesignaaleista, että kielimallin litteroinneista kutsutaan päästä-päähän-menetelmiksi. Näiden menetelmien käyttäminen edellyttää laajaa opetusaineistoa, jossa on useiden eri puhujien puhumana sama sana (ja suomen kielessä myös eri taivutusmuotoineen) tai mallin käyttökelpoisuus rajoittuu pääasiassa opetusaineiston puhujien puheentunnistukseen.

2.4 Puheentunnistuksen onnistumisen tunnusluvut

Puheentunnistuksessa syntyy virheitä johtuen puhujista, koulutusaineiston ominaisuuksista ja sanastosta sekä tunnistusalgoritmien ominaisuuksista. Virheen suuruutta voidaan arvioida tunnettuun litteroituun aineistoon merkki- ja sanavirhesuhteilla (eng. Character/Word Error Rate). Ensimmäinen virhesuhde kertoo kuinka monta merkkiä vastaa litterointia ja jälkimmäinen, kuinka monta muutosta (lisäystä, poistoa, muunnosta) tarvittaisiin, jotta koneen tekemä ja käsin litteroitu aineisto vastaisivat toisiaan.

Virhe voi johtua siitä, että jokin sana on jäänyt pois (poisto) puheentunnistusalgoritmin tekstistä tai sitten algoritmi on saattanut lisätä (lisäys) jonkin sanan, toisinaan algoritmi päätyy muuttamaan sanan taivutusta tai muotoa (muunnos). Näiden avulla voidaan laskea sanavirhesuhde:

$$\text{Sanavirhesuhde} = \frac{\text{poistot} + \text{lisäykset} + \text{muunnokset}}{\text{kaikki virheettömän vertailulitteroinnin sanat}} \qquad (2)$$

Koska kaavan osoittajassa on myös algoritmin mahdollisesti tekemät lisäykset virhesuhde voi olla suurempi kuin 100 %.

Esimerkiksi Googlen puheentunnistusalgoritmia käyttävä SpeechRecognition-python-algoritmi tunnisti Yleisradion FM-radiolähetyksen keskusteluohjelman 60 lausutusta sanasta 47 kappaletta, joista 39 tunnistui oikein (ks. taulukko 1). Sanavirhesuhteeksi tuli

$$\text{Sanavirhesuhde} = \frac{7+3+8}{60} \cdot 100\ \% = 30{,}0\ \%$$

Taulukko 1 Esimerkki YLE:n Puhe-FM-radiokanavan (kohinattoman) keskusteluohjelman automaattisesta puheentunnistustulokset Pythonille kehitetyn SpeechRecognition-kirjaston [27] avulla.

Litteroitu teksti	**Tunnistettu teksti**	**Virheet**
rahaa ja voi päättää siitä minkälainen auto tai asunto on ja miten se	kaahaaja voi täyttää sitä Minkälainen auto tai asuntoon	**Poistot**: 0 kpl **Lisäykset**: 0 kpl **Muunnokset**: 6 kpl rahaa ja → kaahaaja päättää → täyttää siitä → sitä asunto on → asuntoon
lämmitetään et se mihin mihin nuorten ehkä kannattaa erityisesti pistää aikaa on tän ruokavallankumouksen edistäminen	lämmitetään että meen tänne kannattaa erityisesti pistää aikaa on vallankumouksen eristäminen	**Poistot**: 3 kpl se, mihin, mihin, nuorten, ehkä, tän, ruoka- **Lisäykset**: 2 kpl meen, tänne **Muunnokset**: 1 kpl edistäminen → eristäminen
koska se on kuitenkin kaikkein tärkein yksittäinen juttu, jos me halutaan maapallo pelastaa	koska se on kuitenkin kaikkein tärkein yksittäinen juttu, jos ne auttaa	**Poistot**: 3 kpl halutaan, maapallo, pelastaa **Lisäykset**: 1 kpl auttaa **Muunnokset**: 1 kpl me → ne
Kiitoksia Risto Isomäki ja Perttu Pölönen mielenkiintoisesta keskustelusta	Kiitoksia Risto Isomäki ja Perttu Pölönen mielenkiintoisesta keskustelusta	**Poistot**: - **Lisäykset**: - **Muunnokset**: -
ja kiitos kuuntelijoille seurasta ja ensi tiistaina taas uudet aiheet	Kiitos kuuntelijoille seurasta ja ensi tiistaina taas uudet aiheet	**Poistot**: 1 kpl ja **Lisäykset**: - **Muunnokset**: -
Moi	Moi	**Poistot**: - **Lisäykset**: - **Muunnokset**: -
YHTEENSÄ: 60 sanaa	47 tunnistettua sanaa	18 sanaa

Kahden eri valmistajan puheentunnistuskirjaston sanavirhesuhteen vertailua vaikeuttaa käytetyt opetusaineistot ja niiden käsin tehdyt litteroinnit eli onko käytetty ns. virallista kirja- vai epävirallista puhekieltä. Lisäksi puheentunnistimen valmistaja on voinut estää joidenkin sanojen litteroinnin kuten esimerkiksi kirosanat.

Pääosa tässä työssä käytettävästä litteroitavasta radioamatööripuheaineistosta on luonnollisesti puhekielistä, koska harva puhuja puhuu kirjakielisesti. Lisäksi lukusanat kirjoitettiin kirjaimin arabialaisten numeroiden sijaan.

3 Neuroverkkokäsitteitä

Tietokoneella luodut neuroverkot (eng. artificial neural networks) syntyivät aivojen mikrorakenteen innoittamina, mutta tarkalleen ottaen ne eivät täysin vastaa monimutkaista biologista esikuvaansa. Tietokoneneuroverkot koostuvat useista neuroneista (syvät neuroverkot useista kerroksista neuroneita) ja ne pystyvät kuvaamaan mielivaltaisen funktion vasteen, kunhan funktion toimintaa kuvaava opetusjoukko on riittävän kattava.

Monimutkaisista funktioista on usein vaikea saada riittävää näytejoukkoa, jolla neuroverkon painokertoimet säädettäisiin kuvaamaan funktiota tarkasti ja siksi neuroverkkokin erehtyy eikä saavuta täydellistä tarkkuutta – harvoin edes ihmisen suorituskykyä. Toisaalta mikäli opetusaineisto on liian suppea neuroverkko saattaa säätyä kuvaamaan sitä täydellisesti mukaan lukien pienet yksityiskohdat ja kohina eli ns. ylisovittua (eng. overfitting) ja tällöin uusi, opetusaineistosta poikkeava näyte tunnistetaan väärin, koska uusi näyte ei vastaa yksityiskohdiltaan täysin mitään opetusaineiston näytteistä (eli verkko ei kykene yleistämään opetusaineistoa riittävästi).

Jatkossa työssä esitellyt neuroverkot voidaan kouluttaa kahdella eri tavalla (ks. luku 3.1), jotka määrittävät verkkojen käyttötavan. Ensimmäisenä esitellään luvussa 3.1.1 ohjatun opetuksen menetelmä, jota hyödynnetään tässä työssä ja tämän jälkeen lyhyesti ohjaamattoman opetuksen menetelmä luvussa 3.1.2 sekä vähäisen käytettävissä olevan koulutusmateriaalin haasteen ratkaisemiksi kehitetty siirto-oppiminen luvussa 3.1.3. Tämän jälkeen luvuissa 3.2 ja 3.3 käsitellään perseptroniverkkojen perusteoria ja neuroverkkojen toimintaan liittyvät haasteet ja ratkaisuehdotuksia niihin. Peräkkäisien tapahtumien kuvaamiseen soveltuvan takaisinkytketyn neuroverkon toimintaa käsitellään luvussa 3.4 kahden yleisesti käytetyn mallin perusteella. Luvussa 3.5 on konvolutiivisen neuroverkon toiminnan esittely puheentunnistusneuroverkkojen toiminnan ymmärtämiseksi, sekä käydään läpi kirjoitushetkellä kehittyneitä puheentunnistukseen käytettyjä neuroverkkorakenteiden toteutusta PyTorch Lightning -kirjastolla luvussa 3.6. Lopuksi luvussa 3.7 esitellään lyhyesti, miten yhdistämällä useampi samaan tehtävään koulutettu neuroverkko voitaisiin mahdollisesti parantaa tunnistusta.

3.1 Koneoppimismenetelmän opetus

3.1.1 Ohjatun opetuksen menetelmä

Ohjatun opetuksen menetelmät perustuvat tunnettuun opetusaineistoon, jossa on jollakin tavalla määritetty neuroverkolle (tai yleisesti koneoppimisen menetelmälle) mikä on odotettu päätelmä tietyillä syötteillä. Hyödyntäen tätä tietoa opetusvaiheessa lasketaan erotus oikean päätelmän ja neuroverkon antaman päätelmän välillä eli virhe ja korjataan

neuroverkon kertoimia antamaan odotettu päätelmä (ns. gradientti kohti tappiofunktion minimiä, eng. loss function).

Hyvälaatuisen opetusmateriaalin laadinta on työlästä ja vaatii ihmisen määrittämään ns. perustotuuden eri syötteille. Esimerkiksi Amazon tarjoaa tätä Mechanical Turk -nimisenä palveluna [28]. Toisaalta ihmiset tekevät virheitä ja toisaalta eri ihmisillä saattaa olla eri tulkinta samasta aineistosta, esimerkiksi mitä jossakin valokuvassa on kuvattuna. Jopa opetusaineiston tulkitsijoiden maailmankatsomus saattaa siirtyä neuroverkon tekemiin päätelmiin. Ongelmaa on käsitelty NeurIPS-konferenssin datakeskeisessä tekoälytyöpajassa [29] sekä Aalto Yliopiston tutkimusryhmä on julkaissut aiheesta artikkeleita [30].

Ohjatun opetuksen menetelmät pystyvät tekemään päätelmiä uudesta aineistosta opetusaineistossa esiintyvien tapausten mukaisesti eli esimerkiksi tunnistamaan ja nimeämään eri näytteet eri ryhmiin. Mikäli opetusaineisto on riittävän monipuolinen, niin on todennäköistä, että algoritmit luokittelevat myös opetusaineiston ulkopuoliset tapaukset opetusaineistoa vastaavasti. Tämä ei kuitenkaan ole välttämättä haluttu luokittelu. Poikkeavat havainnot voidaan tunnistaa ja luokitella omaksi opetusaineistosta poikkeavien ryhmään.

3.1.2 Ohjaamattoman opetuksen menetelmä

Ohjaamattoman opetuksen menetelmä voi perustua esimerkiksi samankaltaisten tapausten järjestämiseen ryhmiksi (ns. klusterit) aineistosta tapausten ominaisuuksien perusteella. Näissä menetelmissä ei tiedetä kuuluvatko tapaukset samaan ryhmään vaan koneoppimisen menetelmä tunnistaa samankaltaisten syötteiden perusteella toisistaan vähiten poikkeavat tapaukset. Tämä luonnollisesti johtaa epävarmuuteen lopullisissa päätelmissä, mutta voi auttaa analysoimaan tuntematonta aineistoa ja löytämään yhtäläisyyksiä eri tapausten välillä. Tätä menetelmää ei käytetä tässä työssä.

3.1.3 Siirto-oppiminen

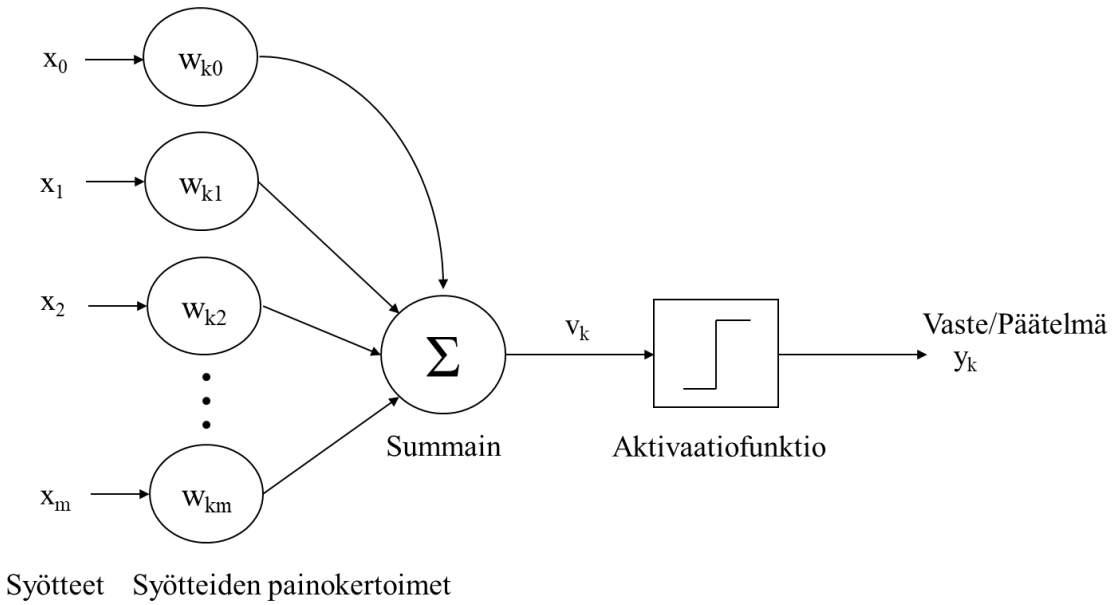
Kun neuroverkko on koulutettu jollakin opetusaineistolla, sen alimmat kerrokset osaavat tunnistaa osia, joista aineistossa esiintyvien rakenteet koostuvat, esimerkiksi kasvontunnistuksessa erilaisia silmiä, neniä, kaaria. Tällainen neuroverkko voidaan siirto-opettaa (eng. transfer learning) tunnistamaan toisenlaisesta aineistosta vastaavia rakenteita. Tällöin tarvittavan opetusmateriaalin määrä voi pienentyä merkittävästi, kun esimerkiksi englantia ymmärtämään opetettu puheentunnistusneuroverkko voidaan siirto-opettaa tunnistamaan opetusaineistosta poikkeavaa kieltä.

Tällöin voidaan saavuttaa muutamia prosentteja pienempi sanavirhesuhde verrattuna mallin kouluttamiseen alusta, esimerkiksi lähteessä [31] Huang tutkimusryhmineen on tutkinut englantia puhuvan verkon opettamista tunnistamaan englannin kielen eri murteiden, saksan, espanjan ja venäjän puhetta. Neuroverkko saattaa ylisovittua määrältään vähäiseen opetusaineistoon ja tällöin määrältään suuremmalla aineistolla opettaminen on

hyödyllistä paremman yleistämistoiminnallisuuden saavuttamiseksi ja lopuksi verkon hienosäätäminen pienemmällä opetusmateriaalilla.

3.2 Eteenpäin syöttävä yksikerroksinen perseptroniverkko

Ensimmäiset askeleet neuroverkkojen osalta pyrkivät mallintamaan matemaattisesti aivojen neuronien toimintaa, ks. McCulloch ja Pitts [32] ja myöhemmin Rosenblatt mallinsi silmän retinan valoherkän perseptronin, johon tulee useita syötteitä eteenpäin syöttävän yksikerroksisen perseptonin avulla (eng. single layer perceptron) [33]. Mallissa painotetaan eri syötteitä (x_1, x_2, … x_m) kertoimilla (w_{k1}, w_{k2}, … w_{km}) ja nämä signaalit summataan. Summa syötetään epälineaariselle aktivaatiofunktiolle, joka muodostaa perseptronin vasteen (y_k). Tämä vastaa kuvassa 9 esitettyä rakennetta.



Kuva 9 Yksikerroksinen perseptroniverkko jäljennetty lähteen [34] mukaisesti.

Alkuperäinen Rosenblattin perseptroni käytti aktivaatiofunktiona Heavisiden askelfunktiota. Aktivaatiofunktioita käsitellään tarkemmin luvussa 3.3.3. Tässä työssä ei käsitellä tarkemmin yksikerroksisia perseptroniverkkoja.

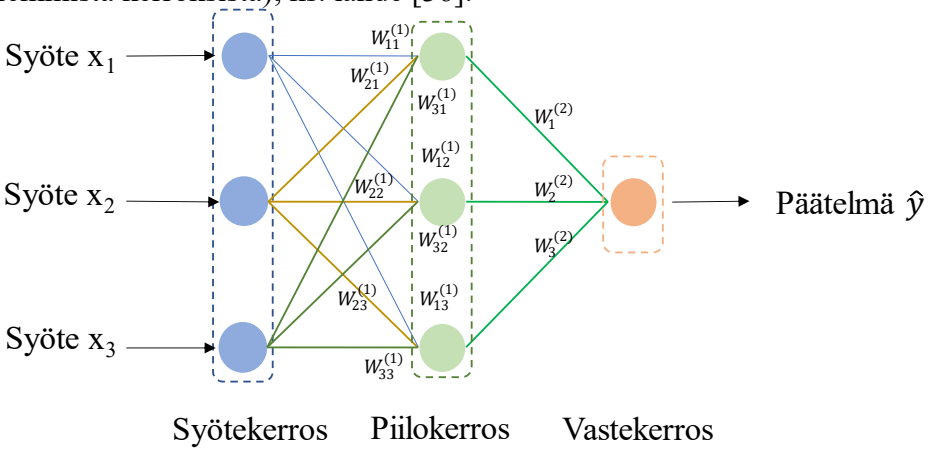
3.3 Eteenpäin kytketty monikerrosperseptroniverkko

Eteenpäin kytketty monikerrosverkko (eng. Multilayer Perceptron) koostuu syötekerroksesta, piilokerroksesta ja vastekerroksesta. Monikerrosverkkoja, joissa on useita piilokerroksia, kutsutaan syväoppiviksi neuroverkoiksi (eng. deep learning network). Piilokerroksia tarvitaan kuvaamaan epälineaarisia päätösrajoja (eng. decision boundary). Piilokerros nimitys tulee siitä, ettei sen syötettä ja vastetta voida määrittää opetusaineiston perusteella, ks. lähde [35].

Neuroverkon piilokerroksia tulee olla riittävästi, jotta neuroverkko kykenee kuvaamaan monimutkaisen funktion. Jos piilokerroksia on liian monta, niin neuroverkko ylisovittuu opetusaineistoon, eikä kykene yleistämään päätelmiä uuteen aineistoon. Neuroverkon piilokerrosten ja perseptronien lukumäärään on olemassa ns. nyrkkisääntöjä (kuten esimerkiksi lähteen [36] mukaisesti piilokerroksen perseptronien lukumäärän tulisi olla syötteiden ja vasteiden lukumäärän välissä, lukumäärä tulisi olla 2/3 syötteiden lukumäärästä + vasteiden lukumäärä ja perseptroneja ei tulisi olla enempää kuin kaksi kertaa syötteiden määrä), mutta lopulta kyse on yhä osittain kokeilemisesta – mikä määrä perseptroneja ja piilokerroksia toimii parhaiten.

Piilokerrosten perseptronit kuvaavat lineaarisen funktion, joten kahdessa ulottuvuudessa piilokerrosten perseptronit kuvaavat suoria ja näiden yhdistäviä pisteitä. Näin ollen hahmottelemalla karkeasti datan perusteella tarvittavan päätösrajan eri pisteiden välille voidaan arvioida piilokerroksissa olevien neutronien määrä tarvittavien eri suuntaisien viivojen perusteella ja piilokerrosten määrän niitä yhdistävien pisteiden perusteella, ks. lähde [37].

Mikäli neuroverkossa ei ole yhtään piilokerrosta, se pystyy tekemään päätelmiä vain lineaarisesti eroteltavissa olevista funktioista (näyte on jommallakummalla puolella viivaa ja saa luokittelun tai päätelmän sen perusteella). Yksi piilokerros mahdollistaa sellaisten funktioiden esittämisen, jotka kuvaavat äärellisen avaruuden toiseen äärelliseen avaruuteen. Kahdella piilokerroksella voidaan kuvata mielivaltainen päätösraja ja mikäli piilokerroksia on useampia (kuin kaksi) ne voivat oppia yksittäisiä ominaisuuksia datasta (tai aiemmista kerroksista), ks. lähde [36].



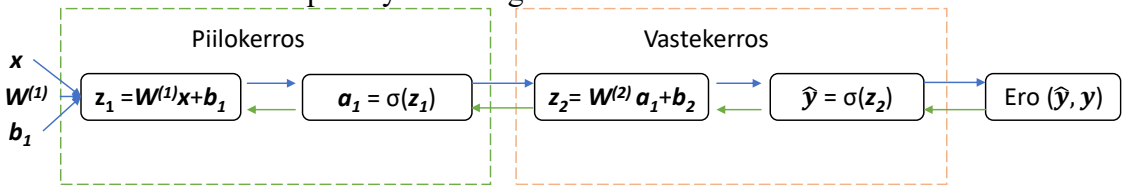
Kuva 10 Moninkertaisen eteenpäin kytketyn perseptronirakenteen kulkukaavio, jossa on yksi piilotettu kerros ja kerroinmatriisin painokerroinalkiot $W_{(mn)}^{(k)}$ merkitty (k on kerros, m on lähtöneuroni, n on tuloneuroni).

Esimerkiksi päätösrajan, joka koostuu seitsemästä viivasta, jotka yhdistyvät yhdessä pisteessä voisi esittää kuvassa 10 esitetyllä neuroverkolla. Koska syötteitä on kolme, niin neuroverkko voisi kuvata kolmessa ulottuvuudessa olevan päätöspinnan (eng. decision

boundary). Tämän havainnollistaminen visuaalisesti on mahdollista, mutta syötteiden määrän kasvaessa neuroverkkojen päätöspintojen ymmärtäminen muuttuu haastavaksi ja käytännössä mahdottomaksi ihmiselle.

3.3.1 Eteenpäin syöttävä algoritmi

Eteenpäin syöttävällä algoritmilla lasketaan verkon tuottama vaste $\hat{y}$ (päätelmä) syötteille x_1, x_2, … x_n. Koulutuksen alussa verkon painokertoimet alustetaan satunnaisesti. Tämän etuna, että painokertoimet todennäköisimmin muuttuvat oikeaan (eli tunnistustehtävän suorittamista tukevaan) suuntaan taaksepäin syöttävää algoritmia käytettäessä. Mikäli painokertoimet olisi alustettu esimerkiksi nolliksi, niin kestäisi useamman askeleen verran ennen kuin taaksepäin syöttävä algoritmi saisi ne muokattua oikeiksi.



Kuva 11 Neuroverkon vasteen laskemiseksi tarvittavat vaiheet. Ensin piilokerroksessa kerrotaan syötevektorin $\boldsymbol{x}$ alkioita kerroinmatriisin $\boldsymbol{W^{(1)}}$ kertoimilla ja nämä tulot summataan yhteen sekä lisätään bias-termi $\boldsymbol{b_n}$, sitten vastekerros laskee piilokerroksen vasteista päätelmän $\hat{\boldsymbol{y}}$. Tämän jälkeen lasketaan ero tunnettuun opetusaineiston vasteeseen y ja lasketaan taaksepäin syöttävällä algoritmilla uudet painokertoimet $\boldsymbol{W}$-matriiseille.

Kuvan 10 monikertainen eteenpäin kytketty neuroverkko voidaan mallintaa kulkukaaviolla, jossa eri kerroksissa tehtävät matemaattiset operaatiot kuvataan, ks. kuvan 11 siniset nuolet. Piilokerroksessa summataan syötevektorin $\boldsymbol{x}$ ja kerroinmatriisin $\boldsymbol{W^{(1)}}$ alkioiden tulot ja lisätään tämä bias-termivektorin $\boldsymbol{b_1}$ kanssa (lineaarinen operaatio):

$$\boldsymbol{z_1} = \boldsymbol{W^{(1)}x} + \boldsymbol{b_1}, \tag{3}$$

Näille lasketaan logistisen aktivaatiofunktion (myös käytetty nimitystä sigmoid-) arvo seuraavasti:

$$\mathbf{a_1} = \sigma(\mathbf{z_1}). \tag{4}$$

Tämän jälkeen vastekerroksessa summataan nämä vektorin $\mathbf{a_1}$-alkioiden arvot kerrottuna kerroinmatriisin $\boldsymbol{W^{(2)}}$arvoilla ja summattuna bias-termivektorin $\boldsymbol{b_2}$ kanssa:

$$\boldsymbol{z_2} = \boldsymbol{W^{(2)}}\mathbf{a_1} + \boldsymbol{b_2} \tag{5}$$

ja lopulta lasketaan tälle yhdelle skalaariarvolle logistisen aktivaatiofunktion arvo (päätelmä $\hat{\boldsymbol{y}}$):

$$\hat{\mathbf{y}} = \sigma(\mathbf{z_2}). \tag{6}$$

Kun verrataan tämän lasketun $\hat{\boldsymbol{y}}$-vektorin arvoja opetusaineiston tunnettuun vastevektoriin $\boldsymbol{y}$, saadaan tappiofunktion arvo (eng. cost function/loss function). Tappiofunktiona voidaan käyttää esimerkiksi virheen neliösummaa:

$$\boldsymbol{\mathcal{L}} = \sum_{i=1}^{n}(\mathbf{y} - \hat{\mathbf{y}})^2. \tag{7}$$

Esimerkki eteenpäin syöttävän algoritmin hyödyntämisestä neuroverkon vasteen laskemiseksi kuvan 11 mukaiselle kulkukaaviolle matemaattisella ohjelmistolla (Matlab, versio R2021b [38]) on esitetty liitteessä 1. Eteenpäin syöttävää algoritmia käytetään, kun neuroverkko on opetettu (kertoimet säädetty opetusaineiston mukaiseksi) tekemään päätelmä (ts. luokittelu) uusista näytteistä.

3.3.2 Taaksepäin syöttävä algoritmi

Taaksepäin syöttävällä algoritmilla (lähteessä [39] käytetty myös nimitystä vastavirta-algoritmi) lasketaan uudet painokertoimet kerroinmatriiseille $\boldsymbol{W^{(n)}}$ neuroverkolle ja ohjataan vastetta (eli opetetaan neuroverkkoa) kohti opetusaineiston tunnettuja päätelmiä. [40] Taaksepäin syöttävä algoritmi mahdollistaa monimutkaisten neuroverkkojen perseptronien painokertoimien säätämisen ilman työlästä eri arvojen kokeilemista tunnetun lopputuloksen saavuttamiseksi (mikä olisi mahdollista yhden perseptronin tai yksinkertaisen neuroverkon tapauksessa) toisin sanoen minimoidaan derivoituva tappiofunktio gradienttimenetelmällä.

Käytännössä tämä tapahtuu laskemalla osittaisderivaatat jokaisen muuttujan suhteen (ks. kulkukaavio kuvassa 11), esimerkiksi ensimmäisen piilokerroksen $\boldsymbol{W^{(1)}}$-muuttujan suhteen voidaan ratkaista seuraavasti:

$$\frac{\mathrm{d}\boldsymbol{\mathcal{L}}}{\mathrm{d}\boldsymbol{W_{11}^{(1)}}} = \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z_2}} \cdot \frac{\partial \mathbf{z_2}}{\partial \boldsymbol{a_1}} \cdot \frac{\partial \mathbf{a_1}}{\partial \mathbf{z_1}} \cdot \frac{\partial \mathbf{z_2}}{\partial \boldsymbol{W^{(1)}}}. \tag{8}$$

Ensin lasketaan tappiofunktion (7) osittaisderivaatta $\hat{\boldsymbol{y}}$ suhteen:

$$\frac{\partial \boldsymbol{\mathcal{L}}}{\partial \hat{\boldsymbol{y}}} = 2(\hat{\boldsymbol{y}} - \boldsymbol{y}) \equiv (\hat{\boldsymbol{y}} - \boldsymbol{y}), \tag{9}$$

sitten välituloksen $\hat{\boldsymbol{y}}$ (6) muuttujan $\boldsymbol{z}$ suhteen

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_2}} = \hat{\boldsymbol{y}}(1 - \hat{\boldsymbol{y}}), \tag{10}$$

missä on hyödynnetty logistisen funktion derivaattaa
$\frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{x})} - \frac{1}{(1+e^{x})^2} = f(x)[1 - f(x)]$.
Seuraavaksi lasketaan vastekerroksen tulosmuuttujan osittaisderivaatta $\boldsymbol{a_1}$ suhteen ks. yhtälö (5):

$$\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{a_1}} = \boldsymbol{W^{(2)}} \tag{11}$$

Nyt voidaan laskea seuraavan logistisen funktion (4) derivaatta kuten edellä:

$$\frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{z_1}} = \boldsymbol{a_1}(1 - \boldsymbol{a_1}). \tag{12}$$

Ja lopulta voidaan ratkaista painokertoimen $\boldsymbol{W^{(1)}}$ ja bias-termien $\boldsymbol{b_1^{(1)}}$ osittaisderivaatat osittaisderivoimalla yhtälöä (3):

$$\frac{\partial \boldsymbol{z_1}}{\partial \boldsymbol{W^{(1)}}} = \boldsymbol{x} \tag{13}$$

$$\frac{\partial \boldsymbol{z_1}}{\partial \boldsymbol{b_1^{(1)}}} = 1 \tag{14}$$

Nyt voidaan laskea eri kertoimien vaikutus tappiofunktioon. Painokertoimen $\boldsymbol{W_1^{(2)}}$ päivittämiseksi pitää laskea osittaisderivaatat:

$$\frac{\mathrm{d}\boldsymbol{\mathcal{L}}}{\mathrm{d}\boldsymbol{W_1^{(2)}}} = \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z_2}} \cdot \frac{\partial \mathbf{z_2}}{\partial \boldsymbol{W_1^{(2)}}} = (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot \hat{\boldsymbol{y}}(1 - \hat{\boldsymbol{y}}) \cdot \boldsymbol{a_1} \tag{15}$$

ja bias-termin $\boldsymbol{b_1^{(2)}}$:

$$\frac{\mathrm{d}\boldsymbol{\mathcal{L}}}{\mathrm{d}\boldsymbol{b_1^{(2)}}} = \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z_2}} \cdot \frac{\partial \mathbf{z_2}}{\partial \boldsymbol{b_1^{(2)}}} = (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot \hat{\boldsymbol{y}}(1 - \hat{\boldsymbol{y}}) \cdot 1 \tag{16}$$

Vastaavasti lasketaan ensimmäisen piilokerroksen osittaisderivaatat $\boldsymbol{W^{(1)}}$:lle

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{\mathcal{L}}}{\mathrm{d}\boldsymbol{W^{(1)}}} &= \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z_2}} \cdot \frac{\partial \mathbf{z_2}}{\partial \boldsymbol{a_1}} \cdot \frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{z_1}} \cdot \frac{\partial \boldsymbol{z_1}}{\partial \boldsymbol{W^{(1)}}} \\ &= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot \hat{\boldsymbol{y}}(1 - \hat{\boldsymbol{y}}) \cdot \boldsymbol{W^{(2)}} \cdot \boldsymbol{a_1}(1 - \boldsymbol{a_1}) \cdot \boldsymbol{x} \end{aligned} \tag{17}$$

ja bias-termille $\boldsymbol{b_1^{(1)}}$:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{\mathcal{L}}}{\mathrm{d}\boldsymbol{b_1^{(1)}}} &= \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z_2}} \cdot \frac{\partial \mathbf{z_2}}{\partial \mathbf{a_1}} \cdot \frac{\partial \boldsymbol{a_1}}{\partial \boldsymbol{z_1}} \cdot \frac{\partial \boldsymbol{z_1}}{\partial \boldsymbol{b_1^{(1)}}} \\ &= (\hat{\boldsymbol{y}} - \boldsymbol{y}) \cdot \hat{\boldsymbol{y}}(1 - \hat{\boldsymbol{y}}) \cdot \boldsymbol{W^{(2)}} \cdot \boldsymbol{a_1}(1 - \boldsymbol{a_1}) \cdot 1 \end{aligned} \tag{18}$$

Epokin päätteeksi päivitetään painokertoimet ja bias-termit:

$$\boldsymbol{W^{(2)}} = \boldsymbol{W^{(2)}} - \alpha_{\boldsymbol{W^{(2)}}} \cdot \frac{\mathrm{d}\boldsymbol{L}}{\mathrm{d}\boldsymbol{W^{(2)}}} \tag{19}$$

$$\boldsymbol{b_1^{(2)}} = \boldsymbol{b_1^{(2)}} - \alpha_{b_1^{(2)}} \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}\boldsymbol{b_1^{(2)}}} \tag{20}$$

$$\boldsymbol{W^{(1)}} = \boldsymbol{W^{(1)}} - \alpha_{\boldsymbol{W^{(1)}}} \cdot \frac{\mathrm{d}\boldsymbol{L}}{\mathrm{d}\boldsymbol{W^{(1)}}} \tag{21}$$

$$\boldsymbol{b_1^{(1)}} = \boldsymbol{b_1^{(1)}} - \alpha_{b_1^{(1)}} \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}\boldsymbol{b_1^{(1)}}}, \tag{22}$$

jossa $\alpha_{W_1^{(2)}}$, $\alpha_{b_1^{(2)}}$, $\alpha_{W_1^{(1)}}$ ja $\alpha_{b_1^{(1)}}$ ovat eri termien oppimisnopeudet. Nämä voivat olla (keskenään) samoja tai sitten mukautua eri tavoin opetusaineiston perusteella, ks. tarkemmat yksityiskohdat luvussa 3.3.7.

Epokin laskemisen jälkeen jatketaan laskemalla eteenpäin syöttävällä algoritmilla näillä päivitetyillä painokertoimilla seuraava virhe sekä päivitetään painokertoimia, kunnes koko opetusaineisto on käyty läpi. Esimerkki taaksepäin syöttävän algoritmin hyödyntämisestä neuroverkon vasteen laskemiseksi kuvan 11 mukaiselle kulkukaaviolle matemaattisella ohjelmistolla (Matlab, versio R2021b [38]) on esitetty liitteessä 1.

3.3.3 Aktivaatiofunktio

Aktivaatiofunktio määrittelee sen, miten syötteiden painotettu summa muunnetaan perseptronin vasteeksi tai päätelmäksi. Mikäli koulutuksessa käytetään takaisinpäin syöttävää algoritmia, aktivaatiofunktion on lisäksi oltava derivoituva. Monimutkaisia funktioita voidaan approksimoida useamman kerroksen eteenpäin syöttävillä neuroverkoilla, jotka käyttävät epälineaarista aktivaatiofunktiota [41]. Seuraavaksi esitellään neljä epälineaarista aktivaatiofunktiota: logistinen, hyberbolinen, tasasuunnattu lineaariyksikkö- ja normalisoitu eksponenttifunktio (eng. soft max -funktio).

Ensimmäisissä neuroverkoissa käytettiin **logistista aktivaatiofunktiota** [35]. Se saa arvoja [0…1] välillä. Logistinen aktivaatiofunktio (yleisessä muodossa) voidaan määritellä seuraavasti:

$$f(x) = \frac{M}{1 + e^{-k(x-x_0)}} \tag{23}$$

jossa M on käyrän suurin arvo, k on logistinen kasvunopeus tai käyrän kaltevuus x_0 on arvo, jolla sigmoidin keskipiste saavutetaan, ks. kuva 12, jossa on käytetty arvoja $M = 1$, $k = 1$, $x_0 = 0$. Mikäli x on erittäin suuri positiivinen luku (tai lähestyy positiivista ääretöntä), funktion arvo on hyvin lähellä yhtä (lähestyy yhtä). Vastaavasti x:n ollessa suuri negatiivinen luku funktion arvo on lähes nolla (sitä kuitenkaan saavuttamatta, lähestyy äärettömyydessä). Tästä ominaisuudesta seuraa, että neuroverkkoa alustettaessa on hyödyllistä pitää aktivaatiofunktiolle syötettävät luvut mahdollisimman pieninä tai opetuksesta voi tulla hidasta, koska suurilla luvuilla logistisen aktivaatiofunktion vaste ei juuri muutu (joitain desimaaleja).

Hyberbolisen (tangentti-) funktion arvojoukko on [-1…1] ja kuten logistinen aktivaatiofunktio, niin myös hyberbolinen funktion arvo lähestyy x:n arvojen kasvaessa yhtä,

mutta arvojen pienetessä funktion arvo lähestyy -1. Hyberbolisen funktion määrittely on seuraava:

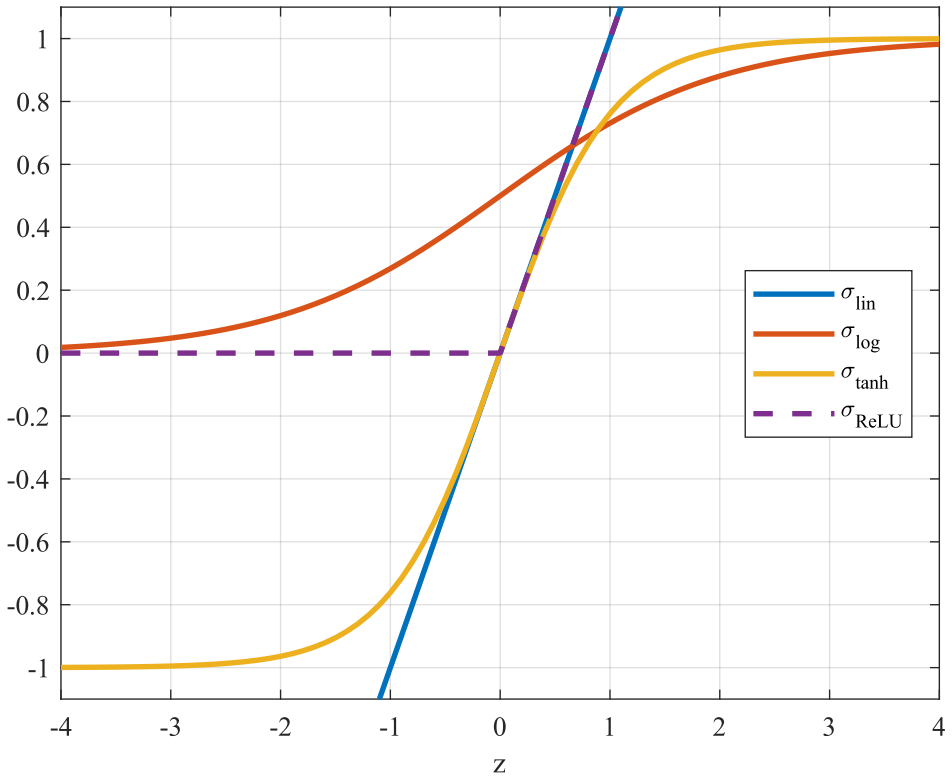
$$f(x) = \tanh(\mathrm{x}) = \frac{\sinh(x)}{\cosh(\mathrm{x})} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{24}$$

missä sinh(x) on hyberbolinen sini- ja cos(x) vastaavasti kosinifunktio ja e^x on eksponenttifunktio.

Tasasuunnattu lineaariyksikköaktivaatiofunktio (eng. Rectified Linear Unit, ReLU) on paloittain lineaarinen funktio, joka antaa syötteen arvon, mikäli se on positiivinen ja muuten nollan, ks. määrittely alla ja kuvaaja kuvassa 12:

$$f(x) = \begin{cases} \mathrm{x} & \text{jos x} > 0, \\ 0 & \text{muutoin} \end{cases} \tag{25}$$

Tasasuunnatun lineaariyksikköaktivaatiofunktion etuna on, että sitä käyttävät neuroverkot ovat helpompia kouluttaa kuin edellä esiteltyjä logistista ja hyperbolista funktiota käyttävät neuroverkot.

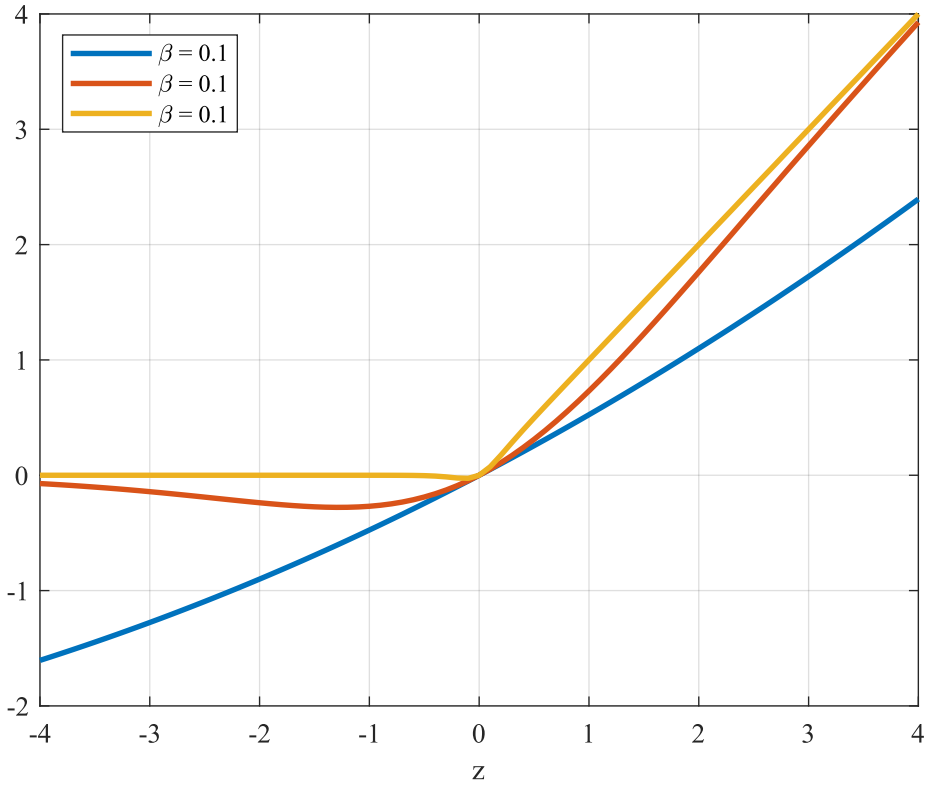


Kuva 12 Lineaarisen, logistisen, hyberbolisen tangentti- ja tasasuunnattu lineaariyksikkö- (Rectified Linear Unit) aktivaatiofunktioiden käyrät.

Tasasuunnatun lineaariyksikköfunktion ongelma on $x = 0$ syntyvä epäjatkuvuuskohta. Tätä voidaan kiertää käyttämällä ns. swish-funktiota [42]:

$$swish(x) = x \cdot \text{sigmoid}(\beta x) = \frac{x}{1 + e^{-\beta x}}, \tag{26}$$

jossa β-parametri vaikuttaa funktion vasteeseen. Kun $\beta = 1$, niin funktio vastaa logistista aktivaatiofunktiota ja kun $\beta \to \infty$, niin funktio lähestyy tasasuunnattua yksikköaktivaatiofunktiota, ks. funktion vasteet eri β arvoilla kuvassa 13.



Kuva 13 Swish-funktion vaste eri β-parametrin arvoilla.

Normalisoidulla eksponenttifunktiolla (eng. softmax) voidaan ratkaista neuroverkon ulostulojen todennäköisyydet sen sijaan, että ulostulos olisi esimerkiksi kaksijakoinen (0 tai 1) [43]. Tämän funktion voi ajatella olevan logistisen funktion yleistys useampaan ulottuvuuteen. Normalisoitu yksikköeksponenttifunktio σ määritellään seuraavasti:

$$\sigma(\mathbf{z})_i = \frac{e^{\mathbf{z}_i}}{\sum_{j=1}^{K} e^{\mathbf{z}_j}}, \tag{27}$$

kun i = 1, …, K ja $\boldsymbol{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$.

Normalisoitua eksponenttifunktiota voidaan käyttää diskreetin muuttujan n eri kategorian tai luokan todennäköisyyksien esittämiseen. Esimerkiksi neuroverkko, jonka tulee

tunnistaa käsin kirjoitetusta tekstistä otetusta valokuvasta numerot voi esittää todennäköisyyden eri numeroille, ks. yksityiskohtainen tunnistusesimerkki lähteessä [44] .

3.3.4 Katoavien/räjähtävien gradienttien ongelma

Neuroverkoissa korjataan taaksepäin syöttävällä algoritmilla painokertoimia hyödyntäen eteenpäin syöttävällä algoritmilla laskettua virhettä kertoimen arvon ja opetusaineistosta lasketun arvon välillä, ks. luku 3.3.2. Katoavien/räjähtävien gradienttien (eng. vanishing/exploding gradient problem) ongelmalla tarkoitetaan tilannetta, jossa useamman kerroksen läpi kuljettuaan virheen suuruus joko pienenee niin pieneksi, ettei siitä voida käytännössä päätellä mihin suuntaan painokerrointa tulisi muuttaa tai niin suureksi ettei välttämättä tietokoneen lukualue riitä esittämään sitä. Näin ollen neuroverkon toiminta tässä perseptronissa muuttuu hyödyttömäksi oppimisen (eli kertoimien muuttamisen) kannalta – joko se kestää liian kauan tai virhettä ei kyetä laskemaan.

3.3.5 Eräoppiminen ja kerrosnormalisointi

Neuroverkkojen koulutuksessa yhtä eteenpäin ja yhtä takaisinpäin syöttävän algoritmin laskemista kaikille näytteille kutsutaan epookiksi (eng. epoch). Neuroverkon kertoimia voidaan päivittää laskemalla erikseen jokaisen näytteen tuottaman tappiofunktion gradientin suuntavektorit yhteen yhtäaikaisesti, ns. **täyden eräoppimisen -menetelmällä** (tai deterministisen gradientin -menetelmällä) (eng. full batch-learning/deterministic gradient method) [45]. Tämän menetelmän etuna on, että tappiofunktion gradientin suuntavektorin todellinen suunta näytteiden perusteella saadaan ratkaistua. Haittana on, että koko aineisto on kyettävä lataamaan tietokoneen muistiin yhtä aikaa käsiteltäväksi (käytännössä suurten matriisien operaatioina). Tämän vuoksi tämä menetelmä soveltuu pienille aineistomäärille [46].

Minieräoppimisessa (eng. mini-batch learning) valitaan kokeilemalla parhaan virheen suppenemisen (gradientti lähestyy tappiofunktion minimiä nopeasti) tai pienimmän virheen tuottava määrä näytteitä, joista laskettujen vektorien perusteella päivitetään kertoimet. Tämä menetelmä vaatii jonkin verran kokeilemista, millä näytemäärällä saavutetaan paras tulos [46]. Suurista näytejoukoista laskettu gradientin suuntavektori johtaa yleisemmin tappiofunktion lokaaliin minimiin, joka on jyrkkäreunainen ja siten menetelmältä jää löytämättä [47].

Mikäli päivitys tehdään jokaisen näytteen tuottaman vasteen jälkeen, tätä kutsutaan **jatkuvaksi eräoppimiseksi tai stokastiseksi** (eng. online batch learning/stochastic). Tämä menetelmä on satunnaisin, koska yksittäiset näytteet saattavat muuttaa gradientin suuntavektoria eri suuntiin. Jatkuvan eräoppimisen menetelmä soveltuu tuotantokäytössä olevien neuroverkkojen säätämiseen [45] [46].

Kerrosnormalisointia (eng. layer normalization) käytetään muuntajamalleissa (ks. luku 4.2). Tämä menetelmä on esitelty lähteessä [48] ja menetelmän etuna on keskiarvon

ja varianssin laskeminen koko kerrokselle yksittäisen neuronin sijaan, jolloin minieräoppimisen erän koko ei vaikuta tulokseen.

3.3.6 Neuronisammutus

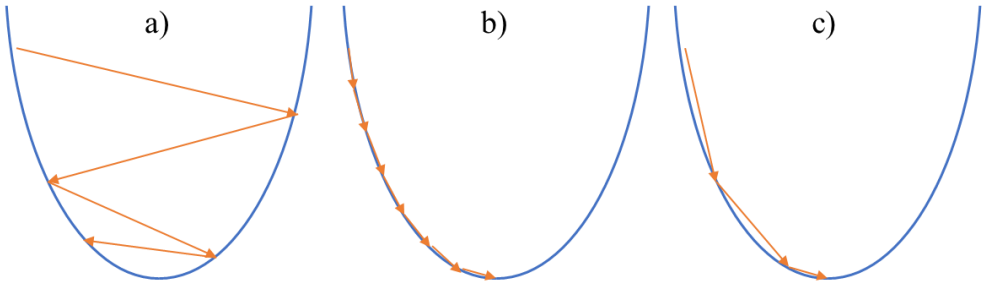
Laajojen ja syvien neuroverkkojen ylisovittumisongelmaa voidaan yrittää ratkaista eräoppimisen lisäksi poistamalla (tai sammuttamalla) satunnaisesti eri neuroneita verkosta epookin ajaksi eli laskemalla eteenpäin ja taaksepäin syöttävillä algoritmeilla vasteet sekä muuttamalla kertoimet vastaavasti (eng. dropout). Seuraavalla epookilla palautetaan edellisellä sammutetut neuronit ja sammutetaan satunnaisesti seuraavat neuronit ja lasketaan neuroverkon vasteet [49]. Menetelmä on Googlen patentoima [50].

Neuronisammutusmenetelmä yksinkertaistaa monimutkaista neuroverkkoa hieman rakenteeltaan poikkeaviksi verkoiksi ja lopputuloksen voidaan ajatella olevan keskiarvo näiden neuroverkkojen vasteista. Lisäksi neuroneista tulee itsenäisempiä, koska viereiset neuronit eivät välttämättä ole mukana verkossa. Tämä lisää myös neuroverkon toimintavarmuutta, koska voidaan ajatella useamman reitin verkossa johtavan tai useamman neuronin vaikuttavan haluttuun lopputulokseen myönteisesti. [39]

3.3.7 Mukautuva oppimisnopeusparametri

Neuroverkkojen taaksepäin syöttävä algoritmi muuttaa gradienttimenetelmässä kertoimia kohti funktion minimiä (minimoi tappiofunktiota). Tätä muutosnopeutta on mahdollista säätää oppimisnopeusparametrin (eng. learning rate) avulla. Liian suuri oppimisnopeus, ks. kuva 14 a) voi estää suppenemisen minimiin tai algoritmi saattaa ohittaa minimin ja jäädä värähtelemään sen ympärille. Liian pieni oppimisnopeus puolestaan hidastaa minimin löytymistä ks. kuva 14 b).

Gradientin lähestyessä funktion minimiarvoa on hyödyllistä hidastaa oppimisnopeutta ks. c)-kuvaaja kuvassa 14, jotta gradientti ei jäisi heilumaan pienimmän arvon ympärille kuten a)-kuvaajassa. Kuvaajat ovat yksinkertaistuksia kahdessa ulottuvuudessa, käytännössä menetelmät optimoivat gradientin useamman ulottuvuuden suhteen.



Kuva 14 Vasemmalla a) havainnollistus liian suuren oppimisnopeuden vaikutuksesta minimin löytämiseen, keskellä b) liian pieni oppimisnopeus hidastaa minimin löytymistä ja oikealla c) oppimisnopeuden pienentyminen lähestyttäessä funktion pienintä arvoa.

Oppimisnopeutta voidaan säätää esimerkiksi eksponentiaalisen pienentymisen avulla, mutta tällöin joudutaan sitomaan tämä johonkin kokeellisesti määritettyyn

epookkien määrään. Ensimmäisiä oppimisnopeutta mukauttavia algoritmeja oli vuonna 2011 julkaistu **AdaGrad**-algoritmi (eng. adaptive gradient algorithm) [51], joka laskee jokaiselle painokertoimelle ja bias-termille oman oppimisnopeuden perustuen niiden tiheyteen eli harvat parametrit, joiden arvo on lähellä nollaa saavat suuremman oppimisnopeuden kuin ne, joiden arvo poikkeaa nollasta.

RMSProp-algoritmi (eng. Root Mean Square Propagation) [52] laskee keskiarvoa viimeisimpien gradienttien suuruudesta ko. painokertoimelle (tai parametrille) ja jakaa oppimisnopeuden näillä. Näin ollen sekin toimii jokaiselle yksittäiselle parametrille.

Adam-algoritmi (eng. Adaptive Moment Estimation) [53] on kehittynyt versio RMSProp-algoritmista, sillä se käyttää parametrien gradienttien ensimmäisten, että toisten momenttien keskiarvoja oppimisnopeuden säätämiseen.

NVIDIA:n tutkijoiden julkaiseman **Novograd**-algoritmin [54] etu verrattuna Adam-algoritmiin on (heidän mukaansa) pienempi muistin käyttö sekä numeerisesti vakaampi toiminta. Novograd-algoritmi perustuu stokastisen gradientin pienenemiseen ja toiset momentit lasketaan kerroskohtaisesti toisin kuin Adam-algoritmissa, jossa ne lasketaan painokerroinkohtaisesti. Novograd-algoritmia käytetään tässä työssä QuartzNet-puheentunnistusneuroverkon (ks. luku 4.1.3) oppimisnopeuden säätämiseen.

3.3.8 Yksi-pois-ristiinvalidointimenetelmä

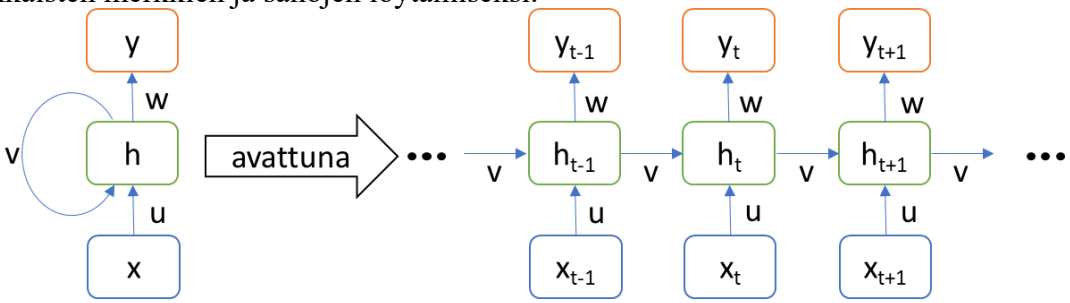
Kappaleessa 3.1.1 esitelty ohjattu koneoppiminen perustuu tuntemattoman funktion vasteen määrittämiseen tunnettujen esimerkkien avulla. Koulutetun neuroverkon vastetta halutaan todentaa testaamalla sitä esimerkeillä tai näytteillä, joita ei ole käytetty opetuksessa. Tähän on kehitetty erilaisia menetelmiä kuten esimerkiksi opetusaineiston jakaminen osiin (esimerkiksi opetusaineisto, testiaineisto ja todennusaineisto). Jakosuhteet hieman vaihtelevat eri koneoppimismenetelmien välillä, mutta tyypillisesti käytetään suurta osuutta (esim. 70 %) opetusaineistona ja pienempiä osuuksia (10–15 %) testi- ja todennusaineistolle.

Koska suurten datamäärien kyseessä ollessa satunnaisesti valittu testi- tai todennusaineisto (aineistosta pääteltävä jakauma) saattaa poiketa merkittävästi verrattuna opetusaineistosta opeteltuun jakaumaan, niin neuroverkko voidaan opettaa useamman kerran käyttämällä eri näytteitä ns. yksi-pois-ristiinvalidointimenetelmällä (eng. leave-one-out cross validation). Tällöin neuroverkon opetus tehdään käyttämällä muita näytteitä, mutta testaus vain yhdellä näytteellä ja tämä toistetaan jokaiselle näytteelle. Tätä menettelyä voidaan soveltaa siten, että jaetaan aineisto esimerkiksi kymmeneen osaan ja käytetään opetukseen kulloinkin yhdeksää muuta osaa ja testaukseen jäljelle jäävää yhtä osaa.

3.4 Takaisinkytketty neuroverkko

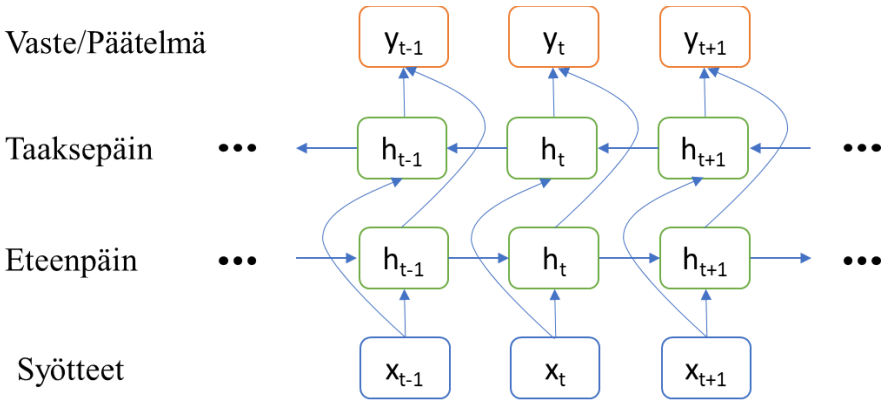
Takaisinkytkeytyvät neuroverkot (engl. Recurrent Neural Networks) säilyttävät osan tilatiedoistaan eri tasoilla. Tämä on hyödyllinen ominaisuus, kun verkolla kuvataan

ilmiöitä, joilla on peräkkäinen rakenne, sekvenssi, esimerkiksi sanat lauseessa (eli jono sanoja) tai sanojen äänteet, jotka ovat riippuvaisia toisistaan. Yksisuuntaiset takaisinkytkeytyvät neuroverkot ovat hyödyllisiä aikasarjojen kuvaamisesta, mutta puheentunnistuksessa kaksisuuntaisuudesta (eng. bidirectional recurrent neural network [55]) on etua peräkkäisten merkkien ja sanojen löytämiseksi.



Kuva 15 Yksisuuntaisen takaisinkytketyn neuroverkon rakenne kompressoituna (vasemmalla) ja avattuna (oikealla). Syötteet ovat *x*, *h* kuvaa piilotettua kerrosta, *y* vastekerrosta ja *u* on syöte painotettuna.

Yksisuuntainen takaisinkytketty neuroverkko muodostuu syötteestä ***x***, piilotetusta kerroksesta ***h***, vastekerroksesta ***y*** ja ***u*** on syötteen painotukset, ks. rakenne avattuna kuvassa 15. Puheentunnistuksessa on etua, mikäli takaisinkytketty neuroverkko kykenee käymään sekvenssin myös taaksepäin ja tämän kerroksen tulokset huomioidaan vasteessa tai päätelmässä, ks. kaksisuuntaisen takaisinkytketyn neuroverkon ylätason arkkitehtuuri kuvassa 16.

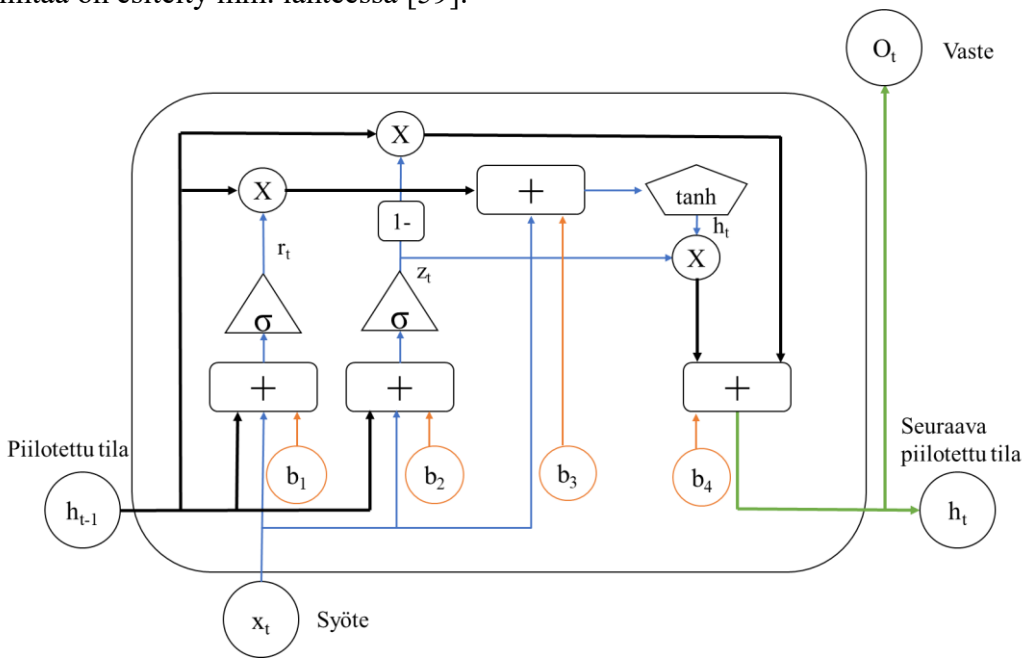


Kuva 16 Kaksisuuntaisen takaisinkytketyn neuroverkon rakenne.

Siinä missä monikerrosperseptroniverkolla kyetään approksimoimaan funktioita, kun niiden sisääntuloja vastaavat ulostulosarvot tunnetaan, niin takaisinkytketyillä neuroverkoilla kyetään löytämään jokaisen peräkkäisen tapahtuman vaikutus ulostuloon [56]. Tämä mahdollistaa edeltävän ja seuraavan sanan tai äänteen hyödyntämisen tunnistuksessa [57].

3.4.1 Portitettu toistuva yksikkö

Portitettu toistuva yksikkö (eng. Gated Recurrent Unit, GRU) koostuu erityyppisitä porteista, joilla verkko painottaa edellisen tilan ja uuden syötteen x_t merkitysportilla r_t (eng. relevance gate) sekä säätelee seuraavan tilan päivitystä päivitysporteilla z_t ja h_t (eng. update gate) [58]. Portitetun toistuvan yksikön rakenne on esitetty kuvassa 17. Yksikön toimintaa on esitelty mm. lähteessä [59].



Kuva 17 Portitetun toistuvan yksikön rakenne, jossa + kuvaa summainta, X matriisien alkiokohtaista tuloa (myös Hadamard- tai Schur-tulo), σ logistista funktiota ja tanh hyberbolista tangenttifunktiota. Eri porteilla säädetään kuinka paljon edellisestä tilasta ja uudesta datasta säilytetään tietoa seuraavaan tilaan.

Portitetun toistuvan yksikön toimintaa kuvaavat seuraavat yhtälöt: nollausportti $\boldsymbol{r_t}$ (eng. reset tai relevance gate)

$$\boldsymbol{r_t} = \sigma(\boldsymbol{W_r x_t} + \boldsymbol{U_r h_{t-1}} + \boldsymbol{b_1}), \tag{28}$$

jossa $\sigma(\,)$ on logistinen funktio (ks. luku 3.3.3), $\boldsymbol{W_r}$ on kerroinmatriisi, jolla kerrotaan sisääntulovektoria $\boldsymbol{x_t}$, $\boldsymbol{U_r}$ on kerroinmatriisi, jolla kerrotaan edellisen askeleen piilotetun tilan $\boldsymbol{h_{t-1}}$-vektoria, ja $\boldsymbol{b_1}$ on bias-termi.

Päivitysportti z_t (eng. update gate)

$$\boldsymbol{z_t} = \sigma(\boldsymbol{W_z x_t} + \boldsymbol{U_z h_{t-1}} + \boldsymbol{b_2}), \tag{29}$$

jossa $\boldsymbol{W_z}$ on sisääntulovektorin $\boldsymbol{x_t}$ kertoimet, $\boldsymbol{U_z}$ on edellisen askeleen piilotetun tilan $\boldsymbol{h_{t-1}}$ kertoimet, ja $\boldsymbol{b_2}$ on bias-termi.

Ehdokastilan aktivaatio $\hat{h}_t$ (eng. candidate activation)

$$\hat{h}_t = \tanh\left(\boldsymbol{W_h x_t} + \boldsymbol{U_h}(r_t \odot \boldsymbol{h_{t-1}}) + \boldsymbol{b_h}\right), \tag{30}$$

jossa tanh() on hyberbolinen tangenttifunktio (ks. luku 3.3.3), ⊙ kuvaa matriisien/vektorien alkiokohtaista tuloa (myös Hadamard- tai Schur-tulo), $\boldsymbol{W_h}$ on kerroinmatriisi, jolla kerrotaan sisääntulovektoria $\boldsymbol{x_t}$, $\boldsymbol{U_h}$ on kerroinmatriisi, jolla kerrotaan nollausportin ja edellisen askeleen piilotetun tilan $\boldsymbol{h_{t-1}}$-vektoria, ja $\boldsymbol{b_h}$ on bias-termi.

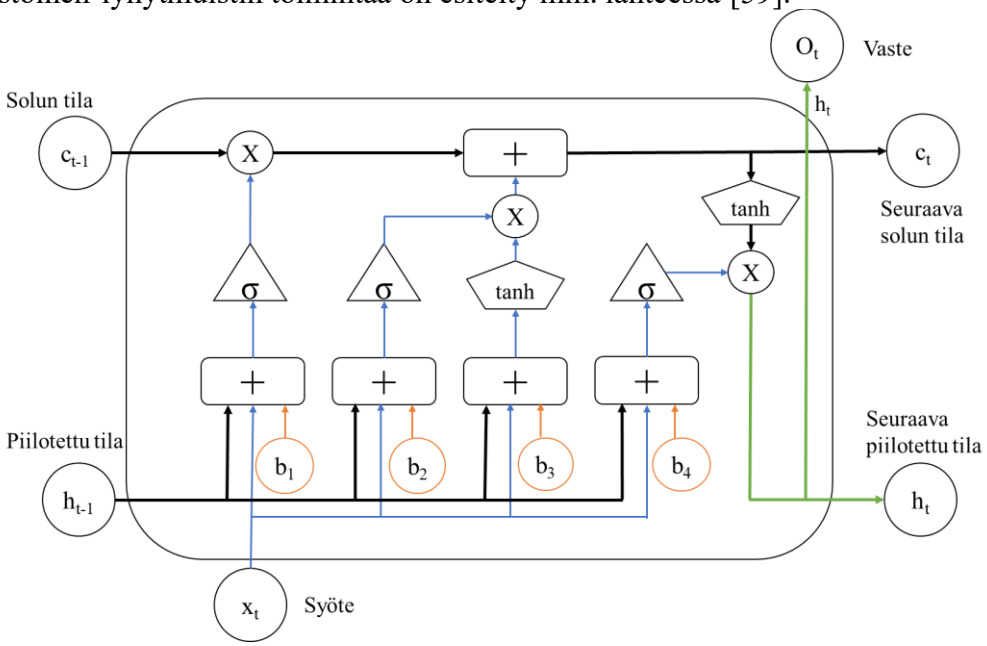
sekä vaste $\boldsymbol{h_t}$ (eng. output vector)

$$\boldsymbol{h_t} = \boldsymbol{z_t} \odot \hat{\boldsymbol{h}}_t + (1 - \boldsymbol{z_t}) \odot \boldsymbol{h_{t-1}}, \tag{31}$$

jossa ⊙ kuvaa matriisien/vektorien alkiokohtaista tuloa, $\boldsymbol{z_t}$ on päivitysportin vastevektori, $\hat{\boldsymbol{h}}_t$ on ehdokastilan aktivaatio ja $\boldsymbol{h_{t-1}}$ edellisen askeleen piilotetun tilan vektori.

3.4.1 Pitkäkestoinen-lyhytkestomuisti-rakenne

Takaisinkytkeytyvä neuroverkko voidaan toteuttaa käyttämällä pitkäkestoinen-lyhytkestomuisti-rakennetta (engl. Long-Short Term Memory) pohjana, ks. yksityiskohdat esimerkiksi lähteistä [60] ja [61]. Pitkäkestoinen-lyhytkestomuisti-rakenteessa on yksi portti enemmän kuin edellä esitellyssä portitetussa toistuvassa yksikössä, vrt. kuvat 17 ja 18. Pitkäkestoinen-lyhytmuistin toimintaa on esitelty mm. lähteessä [59].



Kuva 18 Pitkäkestoinen-lyhytmuisti-rakenne, jossa + kuvaa summainta, X matriisien alkiokohtaista tuloa (myös Hadamard- tai Schur-tulo), σ logistista funktiota ja tanh hyberbolista tangenttifunktiota. Eri porteilla säädetään kuinka paljon edellisestä tilasta ja uudesta datasta säilytetään informaatiota seuraavaan tilaan.

Pitkäkestoinen-lyhytmuisti-rakenteen toimintaa kuvaavat seuraavat yhtälöt: unohda-portti (eng. forget gate)

$$\boldsymbol{f_t} = \sigma\left(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b_1}\right), \tag{32}$$

jossa $\sigma(\,)$ on logistinen funktio (ks. luku 3.3.3), $\boldsymbol{W_f}$ on kerroinmatriisi, jolla kerrotaan sisääntulovektoria $\boldsymbol{x_t}$, $\boldsymbol{U_f}$ on kerroinmatriisi, jolla kerrotaan edellisen askeleen piilotetun tilan $\boldsymbol{h_{t-1}}$ vektoria ja $\boldsymbol{b_1}$ on bias-termi.

Päivitysportin (eng. update gate) yhtälöt:

$$i_t = \sigma(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_2}), \tag{33}$$

jossa $\sigma(\,)$ on logistinen funktio (ks. luku 3.3.3), $\boldsymbol{W_i}$ on kerroinmatriisi, jolla kerrotaan sisääntulovektoria $\boldsymbol{x_t}$, $\boldsymbol{U_i}$ on kerroinmatriisi, jolla kerrotaan edellisen askeleen piilotetun tilan $\boldsymbol{h_{t-1}}$-vektoria, ja $\boldsymbol{b_2}$ on bias-termi.

Vasteensäätelyportin (eng. output gate) laskentakaava:

$$o_t = \sigma(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{b_4}) \tag{34}$$

jossa $\sigma(\,)$ on logistinen funktio (ks. luku 3.3.3), $\boldsymbol{W_o}$ on kerroinmatriisi, jolla kerrotaan sisääntulovektoria $\boldsymbol{x_t}$, $\boldsymbol{U_o}$ on kerroinmatriisi, jolla kerrotaan edellisen askeleen piilotetun tilan $\boldsymbol{h_{t-1}}$-vektoria, ja $\boldsymbol{b_4}$ on bias-termi.

Ehdokastilan aktivaatio $\hat{c}_t$ (eng. candidate activation)

$$\hat{c}_t = \tanh\,(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t-1}} + \boldsymbol{b_3}), \tag{35}$$

jossa tanh on hyberbolinen tangenttifunktio (ks. luku 3.3.3), $\boldsymbol{W_c}$ on kerroinmatriisi, jolla kerrotaan sisääntulovektoria $\boldsymbol{x_t}$, $\boldsymbol{U_c}$ on kerroinmatriisi, jolla kerrotaan nollausportin ja edellisen askeleen piilotetun tilan $\boldsymbol{h_{t-1}}$-vektoria, ja $\boldsymbol{b_3}$ on bias-termi.

Vastevektori muodostetaan seuraavasti:

$$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \hat{\boldsymbol{c}}_{\boldsymbol{t}}, \tag{36}$$

jossa $\boldsymbol{f_t}$ on unohda-portin vastevektori, $\boldsymbol{c_{t-1}}$ on edellisen solun tilan vektori, $\boldsymbol{i_t}$ on päivitysportin vastevektori ja $\hat{\boldsymbol{c}}_{\boldsymbol{t}}$ ehdokastilan aktivaatiovektori ja $\odot$ kuvaa matriisien/vektorien alkiokohtaista tuloa.

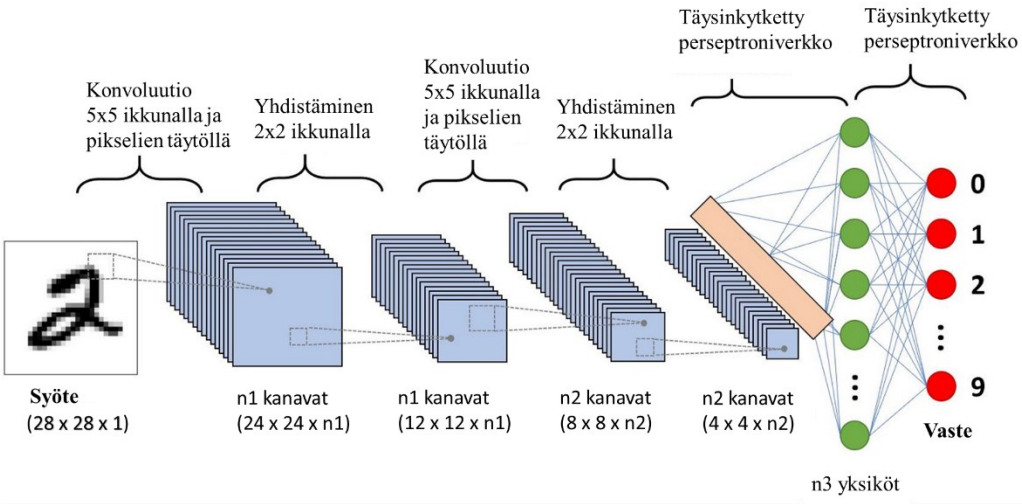
Seuraavan solun piilotettu vektori voidaan laskea

$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t}), \tag{37}$$

jossa $\boldsymbol{o_t}$ on vasteensäätelyportin vektori, $\boldsymbol{c_t}$ on vastevektori ja funktiot kuten edellä.

3.5 Konvoluutionaalinen neuroverkko

Konvoluutionaaliset neuroverkot (Convolutional neural networks) on kehitetty digitaalisten valokuvien käsittelyn ja tunnistuksen tarpeisiin, ja ne ovat hyödyllisiä esimerkiksi datan pakkaamisessa, sillä ne kutistavat datan alkiot pienemmiksi kokonaisuuksiksi, ks. lähteet [62] [63]. Tämä on hyödyllistä, koska tunnistukseen käytetyn täysin kytketyn monikerrospersetronineuroverkon perseptronien väliset kytkennät kasvattavat hyvin nopeasti laskenta-ajan epäkäytännöllisen pitkäksi (ns. ulottuvuuksien kirous, eng. curse of dimensionality).

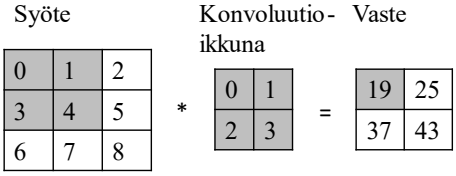


Kuva 19 Konvoluutionaalinen neuroverkko käsin kirjoitettujen numeroiden tunnistamiseen digitaalisista valokuvista. Alussa useita pikseleitä (28x28x1) sisältävä kuva pienenee täysinkytketylle perseptronineuroverkolle käyttökelpoisempaan kokoon (4x4xn2), mutta tunnistuksessa käytettävissä olevien kanavien määrä kasvaa. Muokattu alkuperäisestä lähteen [64] kuvasta (joka on julkaistu CC-BY-SA-lisenssillä).

Pikselien määrän pienentämiseksi on kehitetty erilaisia menetelmiä, kuvassa 19 on esitetty yksinkertainen konvolutiivinen neuroverkko, jolla voidaan tunnistaa 28x28x1-kokoisista digitaalisista valokuvista käsin kirjoitettuja numeroita. Ensin 28x28x1-kokoiseen kuvaan kohdistetaan 5x5 konvoluutioikkuna, sitten 2x2-yhdistysikkuna, sitten 5x5-konvoluutioikkuna ja ennen syöttämistä täysinkytketylle perseptronineuroverkolle kohdistetaan vielä yksi 2x2-yhdistysikkunaoperaatio, jolloin kuvan koko on enää 4x4, mutta erilaisia kanavia on syntynyt useita. Näistä täysinkytketty verkko tunnistaa kuvassa olevan numeron (tai sen todennäköisyyden).

3.5.1 Pikselien konvoluutio

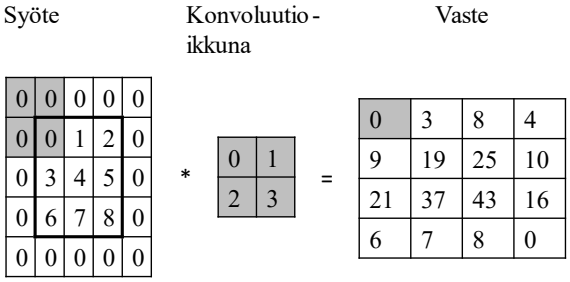
Pikselien konvoluutiossa lasketaan ristikorrelaatio konvoluutioikkunan (eng. convolution window tai kernel) kanssa, kuvan 20 esimerkin mukaisesti ensin kerrotaan elementtikohtaisesti 0x0+1x1+3x2+4x3 = 19. Tämä pienentää alkuperäisen kuvan kokoa pikselimäärältään.



Kuva 20 Konvoluutiokerroksen operaatioiden tuloksena syntyvä uusi matriisi.

3.5.2 Pikselien täyttö

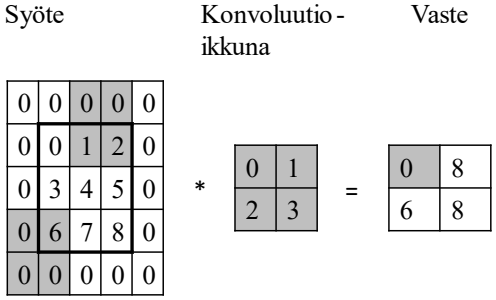
Täytöllä (eng. padding) poistetaan edellä mainittu kuvan reunojen pienemisestä aiheutuva ongelma lisäämällä kuvan reunoille pikseleitä, ks. operaatiot ja syntyvä matriisi kuvassa 21. Käytännössä lisätään nolla-arvoiset alkiot, jotka eivät lisää laskennallista taakkaa.



Kuva 21 Täyttämällä alkuperäisen kuvan ympärille pikseleitä ennen konvoluutiota voidaan säilyttää alkuperäisen kuvan koko.

3.5.3 Pikselien askellus

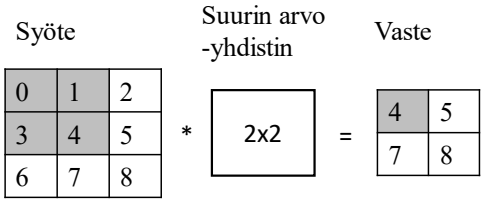
Askelluksella (eng. stride) siirretään konvoluutioikkunaa n_v pikselin verran vaakasuunnassa ja n_p verran pystysuunnassa. Tämä pienentää kuvan tarkkuutta (resoluutiota), koska lopputulos lasketaan vähemmästä määrästä pikseleitä ja siten lopputuloksessakin on vähemmän pikseleitä kuin alkuperäisessä digitaalisessa kuvassa.



Kuva 22 Askeltamalla konvoluutioikkunaa vaakasuunnassa kaksi pikseliä ja pystysuunnassa kolme pikseliä alkuperäinen 5x5 kokoinen kuva pienenee 2x2 kokoiseksi.

3.5.4 Pikselien yhdistys

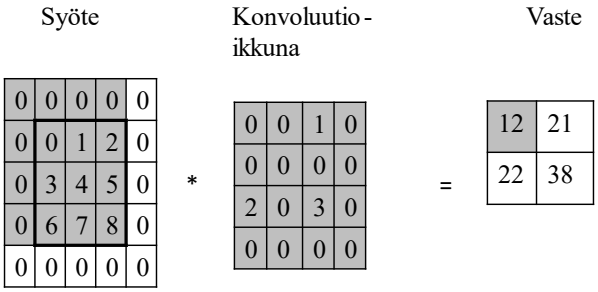
Yhdistämisellä (eng. pooling) valitaan pikseli-ikkunan (eng. pooling window, myös käytetty englanninkielistä nimitystä kernel eli ydin) arvoista suurin (eng. max-pooling) tai lasketaan kaikista pikseleistä keskiarvo. Näin alkuperäinen kuva käytännössä pienenee kooltaan kuten pikselien konvoluutiossa – esimerkiksi 2x2 kokoinen ikkuna pienentää alkuperäistä kuvaa neljänneksellä. Toisin kuin konvoluutiossa, yhdistämisessä ei painoteta pikselien arvoja mitenkään.



Kuva 23 Yhdistämisellä voidaan valita pikseli-ikkunan arvoista suurin (tai laskea niistä keskiarvo). Käyttämällä 2x2 suurin arvo -yhdistintä alkuperäisen kuvan koko pienenee.

3.5.5 Konvoluutioikkunan laajennus

Konvoluutioikkuna voidaan myös laajentaa (eng. dilation), ks. yksityiskohdat esimerkiksi lähteessä [65]. Laajennus kattaa suuremman osan alkuperäisestä kuvasta ilman edellä esitettyä pikselien yhdistämistä. Kuvassa 20 esitetty konvoluutio vastaa laajennusta $l = 1$ ja alla kuvassa 24 on esitetty konvoluutiovaste konvoluutioikkunan laajennuksella 2.



Kuva 24 Konvoluutioikkunan laajennus $l = 2$.

Konvoluutioikkunan laajennus yksinkertaistaa laskentaa, jos pikselien yhdistäminen voidaan jättää tekemättä ja suurempi ikkuna vähentää konvoluutioiden määrää.

3.5.6 Syvyyssuuntainen konvoluutio

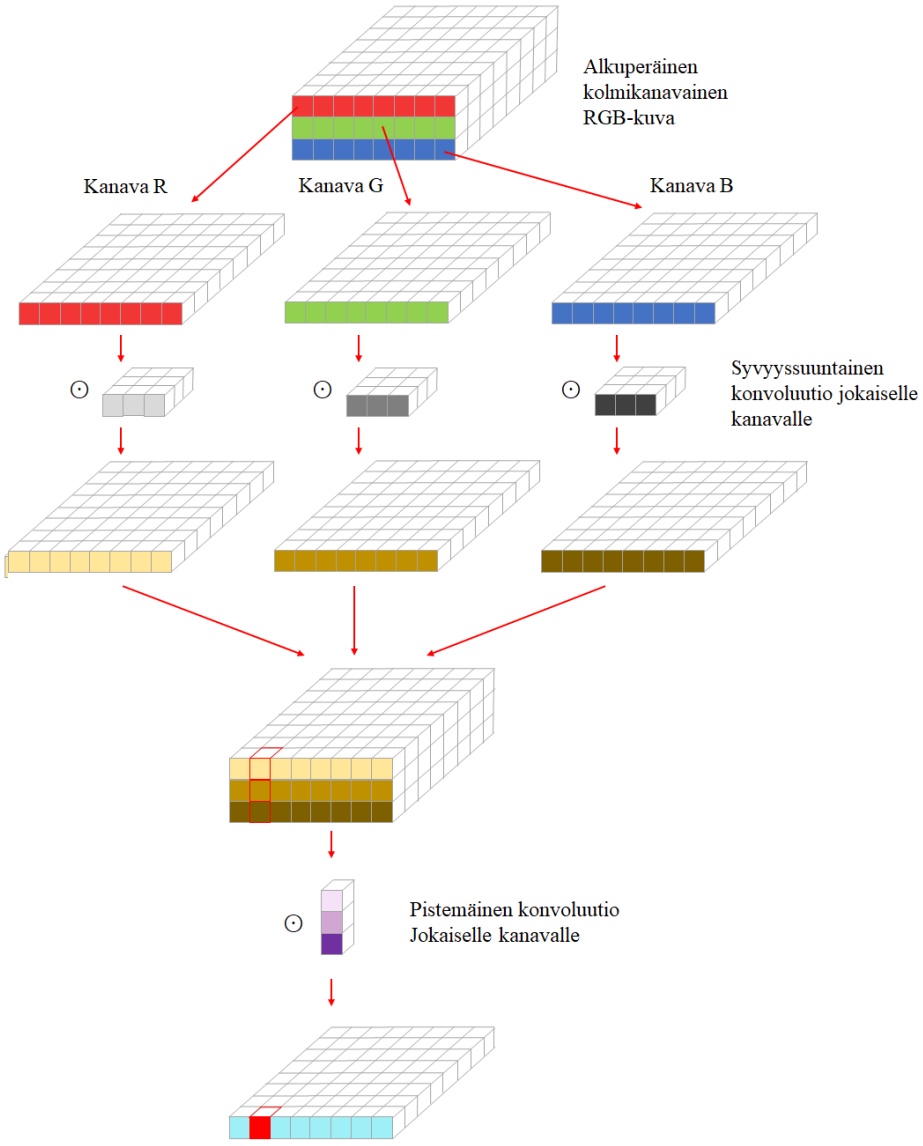
Syvyyssuuntaisessa konvoluutiossa (eng. depthwise convolution) tehdään avaruudellinen (eli kuvan vaaka ja pystysuunta) konvoluutio erikseen jokaiselle kanavalle (jotka kuvaavat syvyyttä). Esimerkiksi digitaalisissa valokuvissa käytetään yleensä kolmea kanavaa esittämään pikselin punaisen, vihreän ja sinisen komponentteja (ns. RGB-värikartta).

3.5.7 Pistemäinen konvoluutio

Pistemäisessä konvoluutiossa (eng. pointwise convolution) tehdään 1x1 konvoluutio jokaiselle pikselille, mutta tämän konvoluutioikkunan syvyys vastaa kanavien määrää ja esimerkiksi edellä olevan RGB-kuvan tapauksessa olisi kolme. Näin ollen lopputuloksena syntyy projektio uuteen kanava-avaruuteen.

3.5.8 Syvyyssuunnassa eroteltavissa oleva konvoluutio

Syvyyssuunnassa eroteltavissa oleva konvoluutio (eng. depthwise separable convolution) yhdistää edellä esitellyt syvyyssuuntaisen ja pistemäisen konvoluution, ks. yksityiskohdat kuvassa 25. Lopputuloksena syntyy huomattavasti vähemmän kertolaskuja kuin mikäli tehtäisiin sama määrä tavallisia konvoluutioita alkuperäiselle kuvalle, ks. MobileNet [66]-lähteessä. Tämä nopeuttaa laskentaa.



Kuva 25 Syvyyssuunnassa eroteltavissa olevan konvoluution havainnollistus kolmikanavaisen digitaalisen RGB-kuvan pikselien avulla.

3.5.9 Puristusheräte-lohko

Puristusheräte-lohkossa (eng. squeeze-and-excitation block) konvoluutiolohkojen kanaviin lisätään parametrejä, jolloin neuroverkko voi sopeuttaa jokaisen piirrekartan (eng. feature map) painotusta [67]. Tavallinen konvolutiivinen lohko painottaa kaikkia piirrekarttoja samalla tavalla [68]. Tämän menetelmän soveltamisella saavutettiin vuonna 2017 noin 25 prosentin parannus ImageNet-luokittelukilpailussa [69] verrattuna edelliseen vuoteen.

Menetelmässä kuvataan tai ”puristetaan” ensin kanavan piirrekartat ensin yhdeksi niitä kuvaavaksi luvuksi keskiarvoisella yhdistämisellä (ks. luku 3.5.4). Tämän jälkeen käyttämällä perseptronineuroverkkoa tasasuunnatuilla lineaariyksikköaktivaatiofunktiolla saadaan tarvittava epälineaarisuus ja logistisella aktivaatiofunktiolla saadaan portitus jokaiselle kanavalle (vrt. luvussa 3.4.1 esiteltyjen portitetun yksikön ja luvussa 3.4.1 pitkäkestoinen-lyhytkestomuistirakenteen portit). Lopulta jokaista piirrekarttaa painotetaan edellä lasketulla kertoimella tai herätteellä.

3.6 Neuroverkkojen toteutus PyTorch Lighting -moduulina

PyTorch Lightning -neuroverkkokehityskirjasto [70] tarjoaa valmiita toteutuksia konvoluutioille, aktivaatiofunktioille ja muille neuroverkkomalleille. Esimerkiksi kuvan 10 kaksiasteinen neuroverkko määriteltäisiin PyTorch Lighning -toteutuksena seuraavasti:

```
nn.Sequential(nn.Linear(1,3),
              nn.ReLU(),
              nn.Linear(3,3),
              nn.ReLU(),
              nn.Linear(3,1),
              nn.Sigmoid()
             ),
```

jossa eteenpäin kytkevät lineaariset perseptronikerrokset on määritelty Linear(*x*,*y*)-funktiolla. Edellä *x* kertoo neuronille tulevien syötteiden lukumäärän ja *y* siitä lähtevien vasteiden. Nämä menevät epälineaariselle aktivaatiofunktiolle - tässä tapauksessa ensimmäisistä kerroksista tasasuunnatulle yksikköaktivaatiofunktiolle, ReLU ja lopuksi logistiselle aktivaatiofunktiolle eli viimeisenä olevalle Sigmoid()-funktiolle ylläolevassa määrittelyssä.

Vastaavasti voidaan esittää käsin kirjoitettujen numeroin tunnistamiseksi kehitetyn konvolutiivisen neuroverkon määrittely (ks. kuva 19) käyttäen valmiita funktioita:

```
nn.Sequential( nn.Conv2d(1,32,kernel_size=5,stride=1,padding=”valid”),
               nn.MaxPool2D(2,stride=2)
               nn.Conv2d(12, 64, kernel_size=5, stride=1)
               nn.MaxPool2D(2,stride=2)
               nn.Flatten()
               nn.Linear(1024,10)
               nn.Dropout(0.1)
               nn.Softmax(dim=1)),
```

jossa on konvoluutiot kuvalle pikselien lisäyksellä ja askelluksella, maksimiarvon yhdistämisellä, tasoitus, lineaarinen neuroverkkokerros neuronisammutuksella. Jotta verkko tuottaa todennäköisyydet eri numeroille tarvitaan lisäksi normalisoitu eksponenttifunktio. Lähteessä [71] on esitetty tarkemmin tarvittava koodi.

3.7 Nippumenetelmät

Koneoppimismenetelmien luokittelutuloksia voidaan yhdistää ns. nippuoppimiseksi (eng. ensemble learning) ja mikäli menetelmien tulosten välillä on paljon hajontaa (poikkeavia havaintoja tai tunnistuksia), ne voivat tuottaa yhdistettynä (esimerkiksi valitaan yleisin menetelmien tuottamista päätelmistä tai painotetaan eri menetelmien päätelmiä esimerkiksi niiden ilmoittaman tuloksen varmuuden perusteella) paremman lopputuloksen kuin yksinään. Useamman tunnistusmenetelmän yhtäaikainen käyttö ei välttämättä ole kovin tehokasta.

4 Päästä-päähän-puheentunnistusmenetelmät

Sanojen tunnistaminen koneellisesti on monimutkainen ongelma johtuen äänteiden ja sanojen eri pituudesta eri puhujilla ja murteissa, kuten luvussa 2.3 taustoitettiin. Puheentunnistusmenetelmiä, jotka pyrkivät oppimaan opetusaineistostaan sekä äänimallin, että kielelliset ominaisuudet kutsutaan päästä-päähän-menetelmiksi. Tässä työssä perehdytään ns. konnektionistiseen aikajaottelu -algoritmiin perustuviin neuroverkkorakenteisiin sekä ns. muuntaja-arkkitehtuuriin (eng. transformer) perustuviin malleihin. Lisäksi esitellään konvolutiiviset verkot ja muuntaja-arkkitehtuurin yhdistävä Conformer-malli.

Taulukko 2 Eräiden päästä-päähän puheentunnistusmenetelmien parametrit ja saavutetut sanavirhesuhteet erillisen kielimallin kanssa ja ilman sitä lähteen [72] mukaan. Parametrimäärän kasvaessa sanavirhesuhde pienenee ko. testijoukossa.

Menetelmä	**Parametrien määrä (miljoonaa)**	**Sanavirhesuhde % ilman kielimallia (testclean-näytejoukko)**	**Sanavirhesuhde % kielimallin avustuksella (testclean-näytejoukko)**
QuartzNet (CTC) [73]	19	3,90	2,69
Transformer [74]	270	2,89	2,33
Transformer Transducer [75]	139	2,4	2,0
ContextNet(L) (CNN-RNN-transducer) [76]	112,7	2,1	1,9
Conformer (S) [72]	10,3	2,7	2,1
Conformer (M) [72]	30,7	2,3	2,0
Conformer (L) [72]	118,8	2,1	1,9
Conformer XXL puoli-itseoppiva [77]	1000	1,5	1,4

Konnektionistinen aikajaottelu -algoritmi tekee oletuksen ehdollisesta riippumattomuudesta algoritmin tunnistamien sanojen osalta tietyllä äänisignaalilla. Tästä seuraa, ettei konnektionistinen aikajaottelu -algoritmi opi kielimallia yhtä hyvin kuin mallit, jotka ovat ehdollisesti riippuvia [23]. Erillinen kielimalli saattaa parantaa tunnistustulosta ks. englanninkieliselle LibriSpeech-aineistolle saavutetut sanavirhesuhteet taulukossa 2 ja lähde [78].

4.1 Konnektionistinen aikajaottelu -algoritmi

Kuten aiemmin luvussa 2.3.3 kuvattiin, niin puheentunnistuksen ongelmana on yhdistää useat samaa äännettä kuvaavat kehykset mahdollisesti vain yhdeksi ainoaksi kirjaimeksi. Tämän ongelman ratkaisemiseksi kehitettiin konnektionistinen aikajaottelu -

algoritmi (eng. Connectionist Temporal Classification) [15], joka laskee todennäköisyydet konvolutiivisen neuroverkon Mel-skeptrikertoimista tunnistamille äänteille. Ensin koodain määrittää ns. väli- tai piiloesityksen *H*, (eng. hidden representation) ja sen jälkeen ns. koodin purkain, (eng. decoder) erottelee (ideaalitapauksessa) kirjaimet toisistaan ja yhdistää samat äänteet yhdeksi kirjaimeksi tai merkiksi väliesityksen *H* perusteella. Tunnistusta voi joissain tapauksissa parantaa käyttämällä kielimallia täydentämään merkkejä tai sanoja, ks. kielimallien esittelyt luvussa 5.

	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
a	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
b	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
c	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
d	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
e	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
f	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
g	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
h	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
i	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,970	0,000	0,850	0,000
j	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
k	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
l	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
m	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
n	0,000	0,000	0,000	0,000	0,000	0,000	0,990	0,000	0,000	0,000	0,000	0,000
o	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
p	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
q	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
r	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
s	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
t	0,000	0,990	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
u	0,000	0,000	0,000	0,910	0,930	0,000	0,000	0,000	0,000	0,000	0,000	0,000
v	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
w	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
x	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
y	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
z	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
ä	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
ö	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
å	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
””	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
-	0,960	0,000	0,980	0,000	0,000	0,970	0,000	0,980	0,000	0,760	0,000	0,980

Kuva 26 Konnektionistisen aikajaottelu -algoritmin esimerkkitunnistus ”-t-uu-n-i-i-” maksimihakumenetelmän mukaisesti.

Esimerkiksi kuvassa 26 on laskettu 11 aika-askeleen todennäköisyydet eri merkeille. Mikäli otetaan yksinkertaisesti suurimmat todennäköisyydet, saadaan seuraava merkkijono: ”-t-uu-n-i-i-” ja kun yhdistetään samat merkit (joiden välissä ei ole ”-”-välimerk-

kiä), saadaan ”-t-u-n-i-i-”. Tämän jälkeen kielimallista voitaisiin tarkistaa, onko ”tunii” tunnettu sana vai kenties esimerkiksi ”tuni”. Huomaa, että mikäli merkistöön otettaisiin lisäksi isot kirjaimet ja välimerkit, niin tunnistettavien merkkien määrä kasvaisi varsin suureksi.

Jotta edellä esitettyyn päättely saavutetaan, niin algoritmi pyrkii maksimoimaan opetusaineistosta laskemalla ehdollisen todennäköisyyden $P(\boldsymbol{Y}|\boldsymbol{X})$, jossa $\boldsymbol{X} = [x_1, x_2, \dots, x_T]$ esittää mallin opetuksessa käytettävää ääniaineistoa ja $\boldsymbol{Y} = [y_1, y_2, \dots, y_U]$ esimerkiksi käsin litteroimalla laadittua transkriptiota. Esimerkiksi lähteessä [78] on esitetty tarkemmat yksityiskohdat takaisinkytketyillä neuroverkoilla toteutetusta konnektionistisesta aikajaottelualgoritmista, mm. dynaamisen algoritmin käyttämisestä todennäköisimmän merkkijonon löytämiseen. Yhdistämällä edellä esitelty konvolutiivinen neuroverkkomalli (ks. luku 3.5) konnektionistisen aikajaottelu -algoritmin kanssa voidaan rakentaa monimutkaisia puheentunnistusneuroverkkoja kuten NVIDIA:n kehittämät Jasper ja Quartznet. Näitä esitellään seuraavaksi lyhyesti seuraavissa luvuissa.

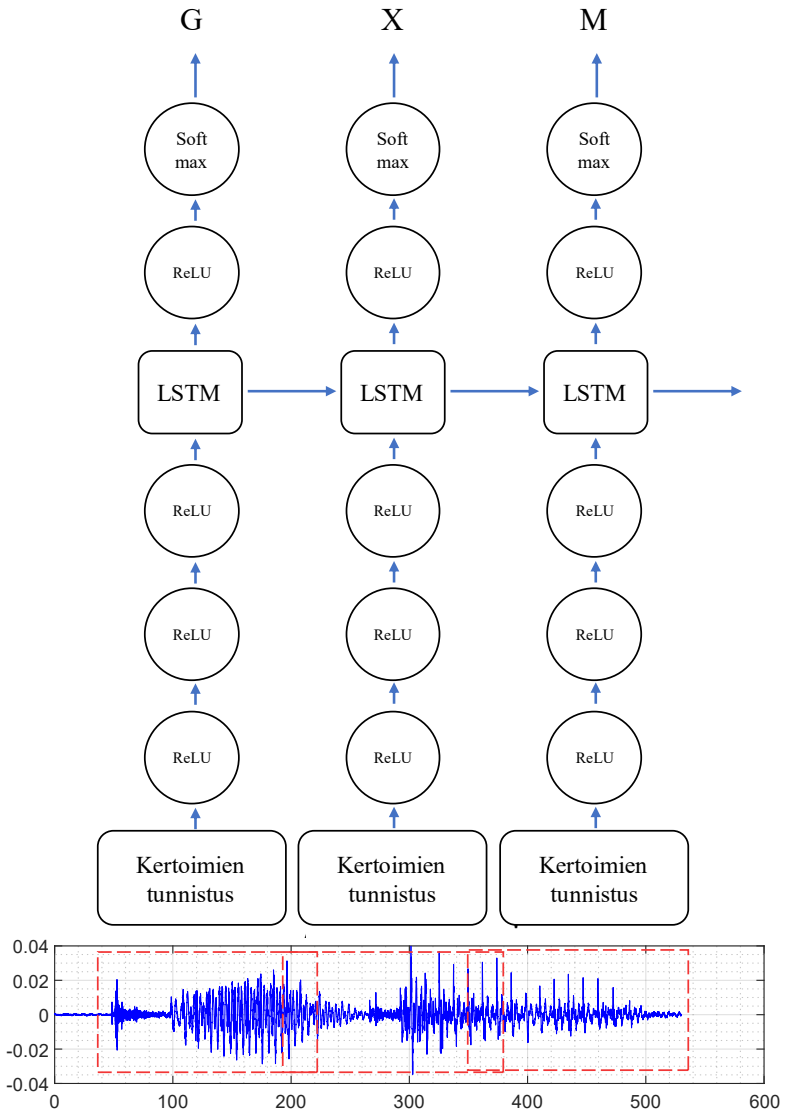
4.1.1 Mozilla DeepSpeech-puheentunnistusalgoritmi

Mozilla DeepSpeech -puheentunnistusalgoritmi [79] perustuu pitkäkestoinen-lyhytkestomuisti-rakenteilla (ks. luku 3.4.1) toteutetun takaisinkytketyn neuroverkon (ks. 3.4) ja konnektionistisen aikajaottelutappiofunktion käyttöön, ks. kuva 27. Mallin kehittivät ja julkaisivat ensimmäisenä Baidun tutkijat lähteessä [80].

Mallissa on viisi piilotettua kerrosta. Ensimmäiset kolme kerrosta kohdistavat äänteitä kuvaavasta spektristä laskettuihin Mel-kepstrikertoimiin tasasuunnatun lineaariyksikköaktivaatiofunktion. Tämän jälkeen mallissa on neljännessä kerroksessa toteutettu takaisinkytketty neuroverkko pitkäkestoinen-lyhytkestomuistirakenteilla, jotka huomioivat äännejonon todennäköisyyden. Viimeinen piilotettu kerros kohdistaa taas tasasuunnatun lineaariyksikköaktivaatiofunktion ja lopuksi verkko laskee eri merkkien todennäköisyyden normalisoidulla eksponenttifunktiolla. Verkon tunnistamaan merkkiin sovelletaan tappiofunktiona edellä esiteltyä konnektionistista aikajaottelua virheen laskemiseksi.

Tällaisen takaisinkytketyn neuroverkon koulutettavissa olevien parametrien määrä on kymmeniä miljoonia. Esimerkiksi Mozillan opettamassa englannin kielen puheentunnistusmallissa on 47 224 861 koulutettavissa olevaa eri parametria (versiossa 0.9.0) [81].

DeepSpeech-neuroverkko on toteutettu Googlen neuroverkkojen koodaamiseen kehittämän TensorFlow-kirjaston [82] avulla. Neuroverkosta on tarjolla raskaampi, erityisesti grafiikkakiihdytinkorteilla laskettavaksi tarkoitettu malli sekä kevyempi vähemmän suorituskykyisille laitteille (kuten matkapuhelimet) tarkoitettu malli [83].

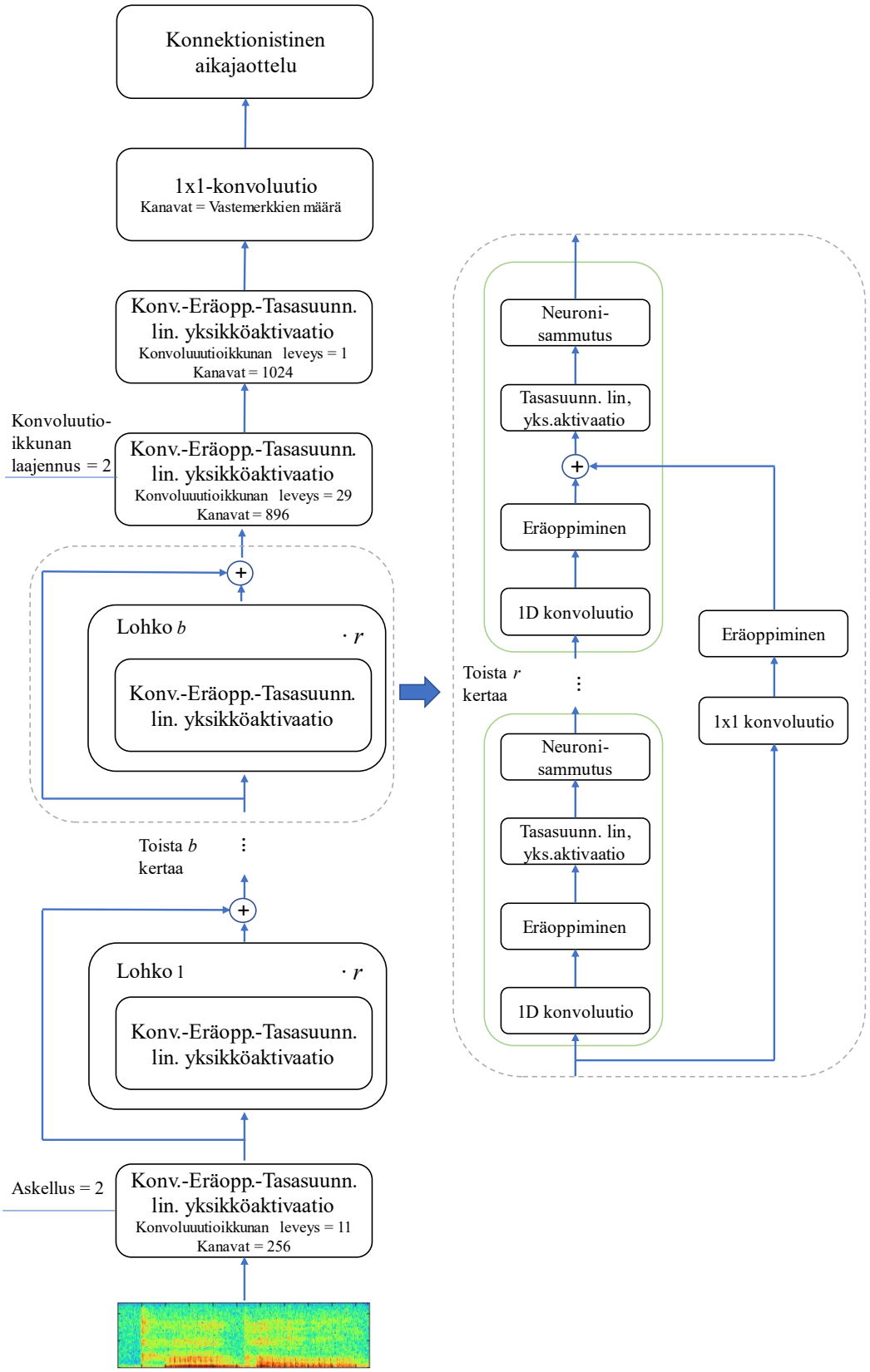


Kuva 27 Mozillan DeepSpeech-neuroverkon arkkitehtuuri (ääniaaltomuoto ja ikkunointi ovat havainnollistus eivätkä vastaa todellista tunnistinta).

4.1.2 Jasper-puheentunnistusneuroverkko

Jasper *b*×*r* (eng. Just Another Speech Recognizer, ks. lähde [84]) on NVIDIA:n tutkijoiden kehittämä syviin aikaviivästettyihin neuroverkkoihin (eng. time delay neural network, TDNN) perustuva puheentunnistusneuroverkkomalli, joka koostuu *b* kappaleesta lohkoja, joiden sisällä toistetaan konvoluutiota alilohkoissa *r*-kertaa muodostaen syvän neuroverkon, ks. lohkokaavio kuvassa 28. Esimerkiksi 10 lohkon (54 konvolutiivista kerrosta) Jasper-malli sisältää 333 miljoonaa parametria, joilla voidaan kuvata äänteitä ja sanoja.

Jasper käyttää 20 ms näytteistysikkunaa eli kehystä (eng. frame) ja peräkkäisten ikkunoiden ylityksenä 10 ms (ks. luku 2.3.1). Verkon ulostulo on todennäköisyysjakauma eri merkeille näytteistysikkunassa. Verkon virhe lasketaan ja verkko koulutetaan konnektionistisella aikajaottelutappiofunktiolla.



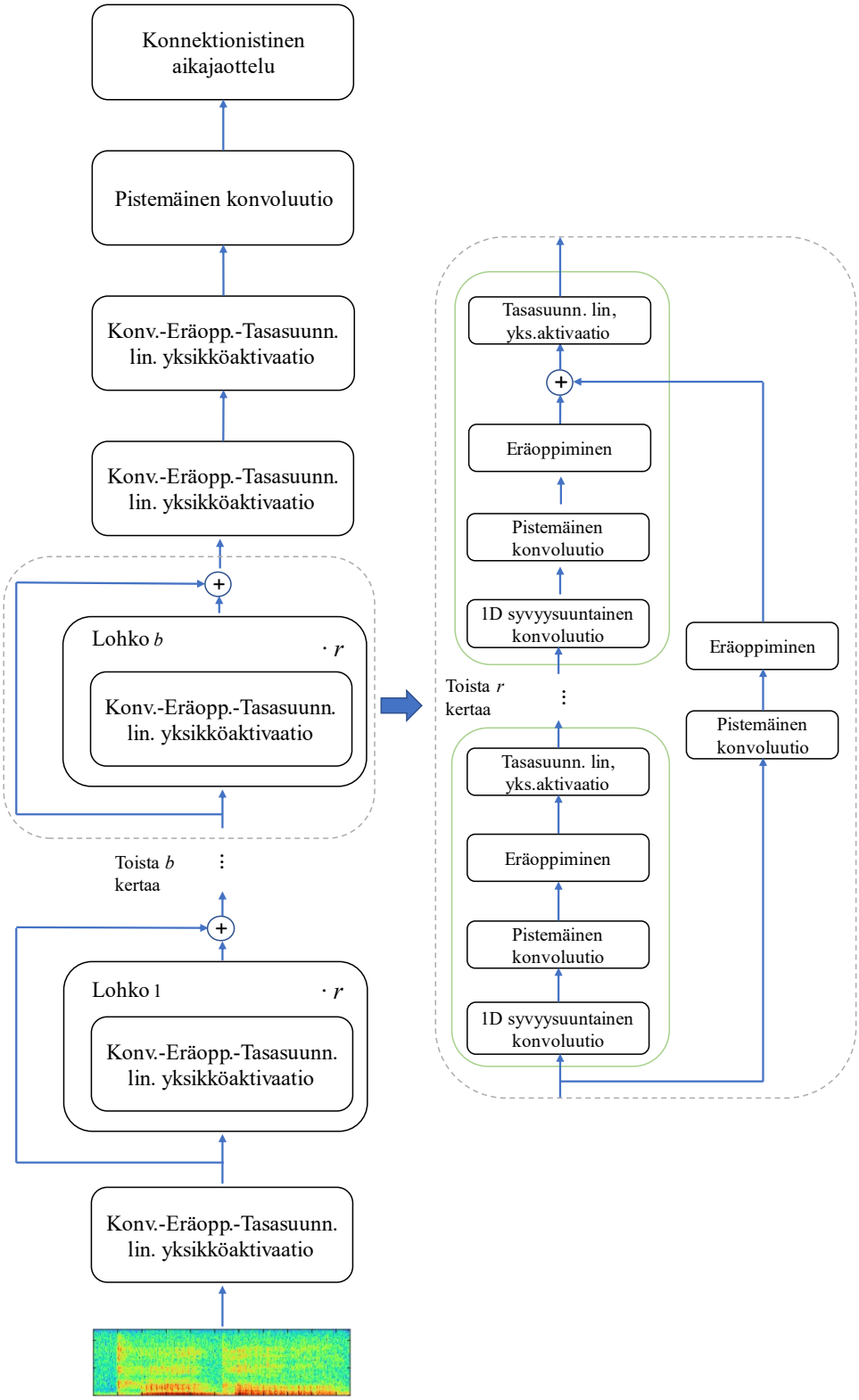
Kuva 28 Jasper-puheentunnistusverkon rakenne, käsitteet suomennettu alkuperäisen lähteen [84] lohkokaaviosta.

4.1.3 QuartzNet-puheentunnistusverkko

Kuten edellä esitellyt verkot, niin myös QuartzNet perustuu konnektionistisen aikajaottelualgoritmin käyttöön kielimallina, ks. rakenne kuvassa 29. Siinä on käytettävissä 18,9 miljoonaa parametria kuvaamaan eri äänteitä ja sanoja. Erotuksena Jasper-verkkoon QuartzNet käyttää aikakanavissa eroteltavia konvoluutioita (ks. luku 3.5.8), mikä pienentää opetettavissa olevien kertoimien määrää, mutta säilyttää tarkkuuden [85]. Jos mallin määrittää käyttämään puristus-herätettä lohkoissa, niin sitten se vastaa Citrinet-mallia [86]. Citrinet-malli käyttää lisäksi alisanakoodausta.

QuartzNet-mallin käyttämisessä on kuitenkin huomioitava, että mallin kehittäneen yrityksen ilmoituksen se kykenee oppimaan pienten (alle 10 000 tuntia) ääniaineistojen materiaalin riittävällä tarkkuudella, ks. Somshubra Majumdarin kommentti NVIDIA:n keskustelualueella [87]. Lisäksi suositus on käyttää suurempien oppimisaineistojen kanssa suurempia malleja.

Jasper- ja QuartzNet-verkot on luotu hyödyntäen PyTorch Lightning neuroverkko-ohjelmointikirjastoa. Molemmat verkot ovat myös ns. avoimen lähdekoodin tuotteita eli ne ovat vapaasti käyttäjien tutustuttavissa ja muokattavissa.

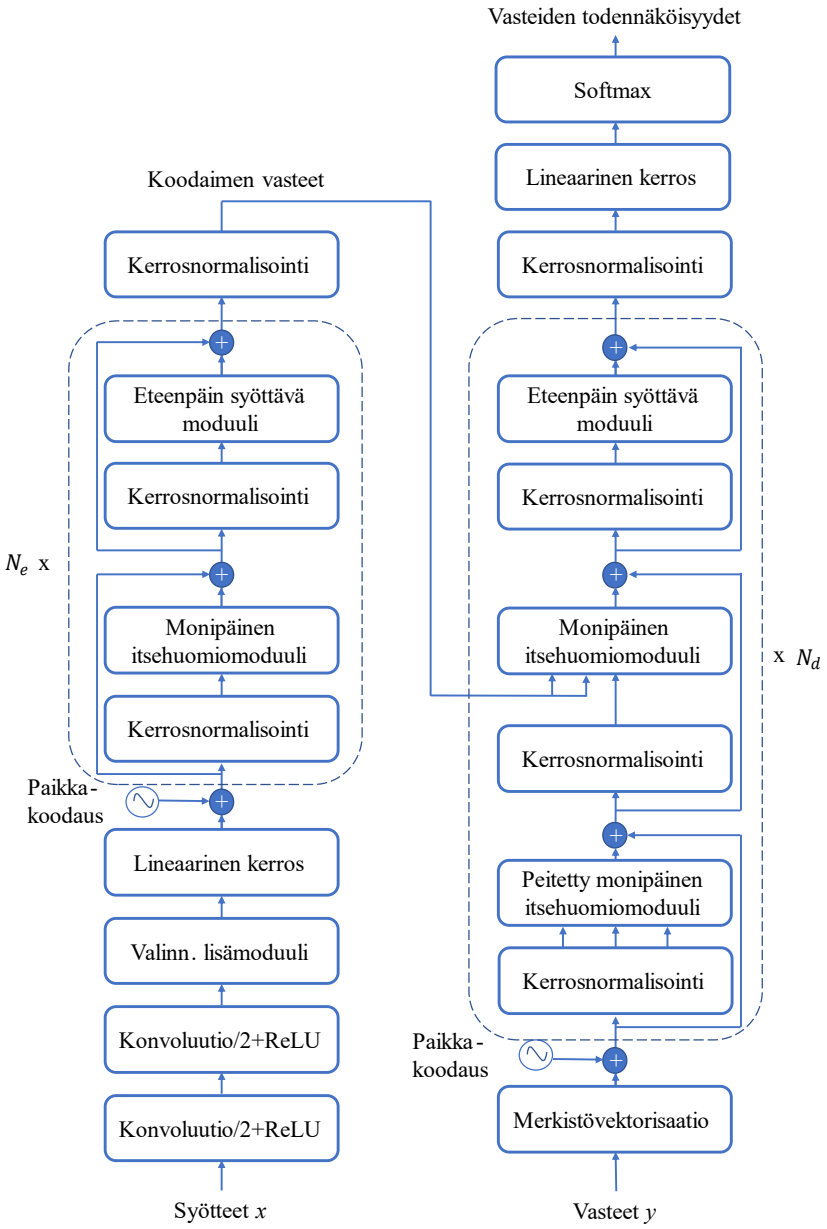


Kuva 29 Konvolutiivisen QuartzNet-puheentunnistusverkon rakenne, käsitteet suomennettu alkuperäisen lähteen [73] lohkokaaviosta.

4.2 Muuntaja-arkkitehtuuri

Kirjoitushetkellä yksi kehittyneimmistä neuroverkkoarkkitehtuurimalleista perustuu Googlen tutkijoiden vuonna 2017 (ks. lähde [16]) esittelemään ns. muuntajarakenteeseen

(eng. transformer) ks. kuva 30. Tämä arkkitehtuuri kehitettiin ratkaisemaan takaisinkytkettyjen neuroverkkojen ongelmia, mm. laskennan sarjamuotoisuuden aiheuttama hitaus. Rakenteessa on koodaimia (eng. encoder) ja koodinpurkajia (eng. decoder). Lisäksi mallissa on sisähuomio, joka huomioi syötteiden todennäköisyyden laajemmassa asiayhteydessä esimerkiksi tietyt sanat lauseessa kuuluvat yhteen.



Kuva 30 Puheentunnistuksessa käytetty muuntajarakenne lähteen [88] mukaan (suomennos kirjoittajan). Mallissa Mel-kertoimet syötetään ensin konvolutiiviselle verkolle ja sitten koodaimista ja koodinpurkajista, joissa hyödynnetään sisähuomiota koostuvalle muuntajalle.

Aiemmin esiteltyjen neuroverkkojen laskentahaasteena on ollut mm. muistin riittäminen ja esimerkiksi takaisinkytketyillä neuroverkoilla laskennan peräkkäisyys (ks. yksityiskohdat luvussa 3.4), mutta muuntajien etuna on niiden laskennan rinnakkaistuminen (etenkin niiden koodaimissa). Muuntaja-mallien parametrien määrä voi olla miljardeja,

vertaa esimerkiksi luonnollisen kielen käsittelyyn (eng. natural language processing) kehitetty Megatron-Turing NLG 530B-malli, jossa on 530 miljardia parametria [89].

4.2.1 Koodain

Puheentunnistusmuuntajan koodaimessa (eng. encoder) muodostetaan konvolutiiviselta verkolta saatavista Mel-kertoimia kuvaavista vektoreista sisähuomiolla painotettu matriisiesitys. Koodaimen ulostulo lasketaan eteenpäin syöttävällä neuroverkolla. Lisäksi koodaimessa käytetään kerrosnormalisointia (eng. layer normalizing, ks. yksityiskohdat luvussa 3.3.5) ja residuaalihaaraa.

4.2.2 Syötteiden vektorisaatio ja paikkakoodaus

Puheentunnistusta ei tehdä suoraan mikrofonista saatavalle aikajatkuvalle signaalille vaan siitä lasketaan Mel-kertoimet (ks. luku 2.3.3). Näistä muodostetaan jokaiselle äänteelle (merkille) ja sanalle kuvaavat vektorit ns. vektorisaatiossa (eng. embedding). Termi juontaa juurensa luonnollisen kielen käsittelyyn, jossa eri sanojen samankaltaisuutta arvioitiin muodostamalla sanoja kuvaavat vektorit (eng. word embedding) ja laskemalla niiden etäisyydet esimerkiksi euklidista etäisyyttä (eng. Euclidean distance). Myös muuntaja-arkkitehtuurissa muodostetaan vastesanoja kuvaavat vektorit, joiden Mel-kerroinesitykset malli oppii tunnistamaan. Vektorin pituus määräytyy opetusaineiston pisimmän lauseen mukaan.

Koneoppimismenetelmät hyötyvät sanojen järjestyksen tai keskinäisen etäisyyden tuntemisesta lauseessa ja tätä varten on kehitetty paikkakoodaus (eng. positional encoding). Eräs tapa aikaansaada paikkakoodaus on painottaa edellä saatua vektoria esimerkiksi siniaallosta lasketuilla kertoimilla.

4.2.3 Sisähuomio

Sisähuomio (eng. self-attention) vertaa jokaista syötevektorin $\boldsymbol{x} \in \mathbb{R}^{m \times d_v}$ alkiota itsensä kanssa vastevektorin $\boldsymbol{y} \in \mathbb{R}^{m \times d_v}$ saamiseksi. Tämä tapahtuu kertomalla syötevektoria kolmella kerroinmatriisilla, kyselymatriisi $\boldsymbol{Q}$ (eng. query) saadaan $\boldsymbol{W^q x}$-tulona ja vastaavasti avainmatriisi $\boldsymbol{K}$ (eng. key) $\boldsymbol{W^k x}$-tulona ja arvomatriisi $\boldsymbol{V}$ (eng. value) $\boldsymbol{W^v x}$-tulona. Huomioitavaa on, että kyselymatriisin $\boldsymbol{Q}$ dimensio on $m \times d_k$ ja avainmatriisin dimensio on $n \times d_k$ ja arvovektorien $n \times d_v$. Tällöin seuraavan laskentakaavan sisähuomion dimensioksi tulee $m \times d_v$:

$$\boldsymbol{y} = softmax(\frac{\boldsymbol{Q K^T}}{\sqrt{d_k}})\boldsymbol{V} \tag{38}$$

Tällöin edellä esitetty kaava tuottaa ns. skaalatun pistetulosisähuomion [16]. Normalisoitu eksponenttifunktio (eng. softmax) laskee skaalauksen, jolla arvovektoria kerrotaan eli kysely- ja avainmatriisien alkioiden samankaltaisuus (tulo) määrää paljonko arvomatriisin alkion arvosta otetaan huomioon. Suurilla kysely- ja avainvektorien dimensioilla

d_k pistetulon arvo kasvaa suureksi ja normalisoidun eksponenttifunktion arvo häviää, ks. katoavien gradienttien ongelma luvussa 3.3.4. Tämän vuoksi kysely- ja avainmatriisien tuloa skaalataan $1/\sqrt{d_k}$. Havainnollinen esitys sisähuomion laskemisesta on esitetty lähteessä [90].

Monipäisessä (tai monikerroksisessa) sisähuomiossa käytetään ensimmäisen koodaimen ulostuloa (jossa on mukana edellä mainittu paikkakoodaus) muiden päiden lähtötietona ilman paikkakoodausta. Alkuperäisen artikkelin [16] kirjoittajat käyttivät kahdeksanpäistä sisähuomiota ja lopuksi näiden päiden tulosmatriisit yhdistetään yhdeksi suureksi matriisiksi, jota kerrotaan kerroinmatriisilla. Tätä tulosta sitten käytetään eteenpäin syöttävän neuroverkon syötteenä.

4.2.4 Koodinpurkaja

Koodinpurkajat (eng, decoder) vastaavat rakenteeltaan koodaimia, ks. kuva 31, mutta lauseiden loput on peitetty sisähuomion laskennassa. Näin koodinpurkajat joutuvat tekemään päätelmät pelkästään näkemistään äänteistä tai sanoista.

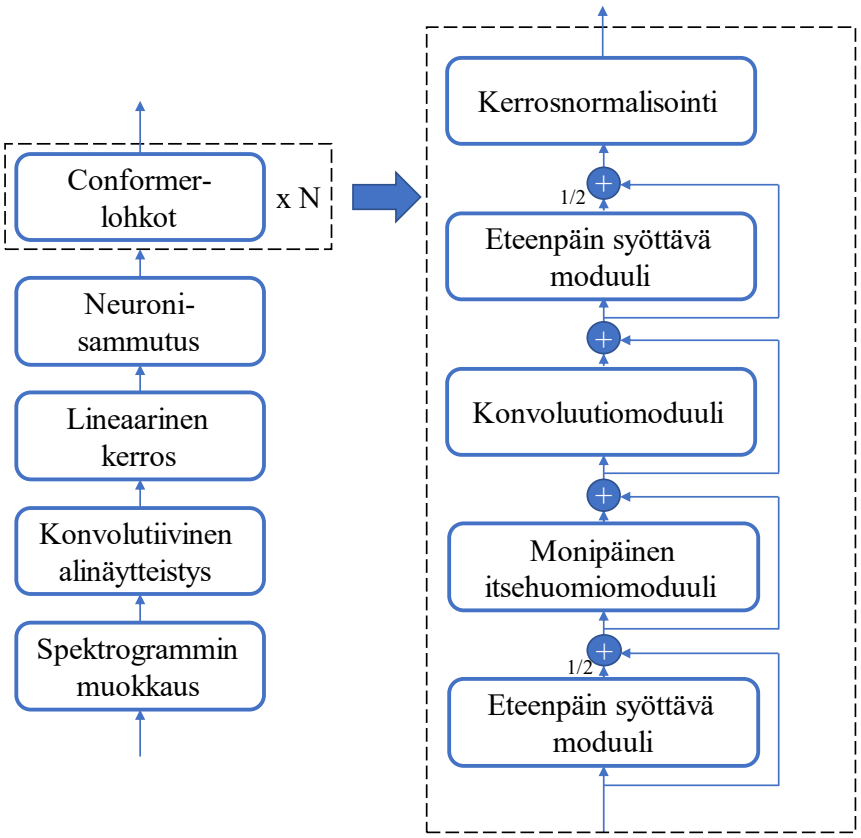
4.2.5 Lineaarinen kerros ja normalisoitu eksponenttifunktio

Lineaarinen neuroverkkokerros (eng. linear) projisoi koodinpurkajalta saadun vektoriesityksen koko opetusaineiston sanaston kattavaksi isoksi vektoriksi, jossa jokainen sana saa pisteytyksen. Normalisoitu eksponenttifunktio (eng. softmax) puolestaan muuntaa nämä pisteytykset todennäköisyyksiksi ja korkeimman todennäköisyyden saanut sana valitaan tunnistustulokseksi.

4.3 Conformer-rakenne

Konvolutiivisiin verkkoihin (eng. convolutive neural networks) ja muuntajaan (eng. transformer) perustuva Conformer-rakenne, ks. lohkokaavio kuvassa 31, pyrkii yhdistämään molempien mallien etuja. Lähteen [72] kirjoittajat esittävät konvolutiivisten verkkojen kykenevän lyhyen aikavälin piirteiden – käytännössä erilaisten kuvioiden tunnistamiseen ja muuntaja puolestaan pitkien aikariippuvuuksien havaitsemiseen eli esimerkiksi lauseen alun ja lopun sanojen.

Conformer-rakentesta on olemassa kolmea eri kokoista neuroverkkoa (pieni, keskikokoinen ja suuri), ks. taulukko 2. Näiden koulutettavissa olevien parametrien lukumäärä kasvaa noin kymmenestä miljoonasta aina 118 miljoonaan.



Kuva 31 Conformer-mallin lohkokaavio lähteen [72] mukaan (suomennos kirjoittajan).

Conformer-lohkojen eteenpäin syöttävissä moduuleissa on käytetty swish-aktivaatiofunktiota aiemmissa malleissa käytetyn tasasuunnatun lineaariyksikköfunktion sijaan. Lähteen [72] kirjoittajat ovat tutkineet monipäisen itsehuomiomoduulin ja konvoluutiomoduulin keskinäistä sijoitusta Conformer-lohkossa ja päätyivät sijoittamaan konvoluutiomoduulin itsehuomiomoduulin jälkeen – joskin artikkelissa esitetyt sanavirhesuhteen erot ovat pieniä (n. 0,1 % LibriSpeech-aineistossa.

4.4 Puheentunnistusmenetelmien vaatimat laskentaresurssit ja energiankulutus

Monet suuret neuroverkkomallit vaativat suuren määrän muistia, jotta laskentamatriisit mahtuvat muistiin ja niillä voidaan laskea. Transformer-mallit toimivat rinnakkaisesti, joten muistivaatimus on maltillisempi ja käytännössä laskentakortin rinnakkaiset ytimet vaikuttavat enemmän.

Laskennan nopeuttamiseksi on alettu käyttämään pienintä numeerista tarkkuutta, joka antaa ”riittävän” tarkkoja tuloksia. Esimerkiksi NVIDIA:n A100-laskentakortti on suunniteltu käyttämään 16-bittisiä liukulukuja ja uusin H100-kortti käyttää valintansa mukaan 32-, 16- tai 8-bittisiä liukulukuja [91].

NVIDIA-yrityksen valmistaman V100-laskentakortin sähkönkulutus on noin 300 wattia (riippuen mm. kuormituksesta). Tyypillisessä laskentapalvelimessa (ks. esimerkiksi lähde [92]) on esimerkiksi kahdeksan V100-laskentakorttia ja näiden yhteenlaskettu

kulutus on tällöin 2,4 kW eli tunnissa nämä kuluttavat 2,4 kWh. Mikäli neuroverkkolaskenta kestää esimerkiksi kolme vuorokautta, niin nämä kortit kuluttavat yhteensä noin 172,8 kWh sähköenergiaa. Tämä muuttuu pääosin lämmöksi, mikä täytyy jäähdyttää palvelimesta ja konesalista pois. Koska jäähdytykseen kuluu energiaa (puhaltimia pyörittävissä moottoreissa, jäähdytysaineen kierrätykseen), niin kulutettu kokonaissähköenergia on suurempi kuin pelkkien laskentakorttien kuluttama. Vertailun vuoksi tuolla energiamäärällä voisi lämmittää lähes kymmenen kertaa kuuden kilowatin kiuasta kolmen tunnin ajan (tai lähes 29 tuntia yhtäjaksoisesti).

5 Luonnollisen kielen käsittelyn kielimallit

Luonnollisen kielen käsittelyn (eng. natural language processing) tavoitteena on ennustaa suureen opetusaineistoon perustuen lauseen sanojen todennäköisyydet eli kun lauseessa on tietyt sanat, niin mikä on seuraava tai suomen kielen ollessa kyseessä mikä on todennäköisin taivutusmuoto huomioiden muiden lauseen sanojen taivutusmuodot. Seuraavissa luvuissa esitellään lyhyesti puheentunnistustuloksen parantamisessa usein käytettyjä kielimalleja.

5.1 Maksimihakuun perustuva malli

Maksimihaku (eng. max decoding tai greedy decoding) on yksinkertaisin malleista. Se perustuu todennäköisimmän sanan valitsemiseen riippumatta lauseyhteydestä. Haasteena on kuitenkin, että valitsemalla todennäköisin sana voi käydä niin, että valittua sanajärjestystä ei löydy lainkaan opetusaineistosta. Tällöin tuotettu lause voi olla täysin käsittämätön.

5.2 N-grammeihin perustuvat kielimallit

Eräs yksinkertaisista kielimalleista perustuu opetusaineistosta laskettujen peräkkäisten sanojen todennäköisyyksien laskemiseen, ns. N-grammeihin. Bigrammiksi kutsutaan opetusaineistosta laskettujen kahden toisiaan seuraavan sanan todennäköisyyttä, trigrammiksi kolmen toisiaan seuraavien sanojen todennäköisyyttä ja niin edelleen.

Ideaalitapauksessa opetusaineisto sisältäisi kaikki kuviteltavissa olevat sanajärjestykset ja näin ollen myös lauseet. Tällöin olisi mahdollista laskea todennäköisyys uusille ennalta tuntemattomille lauseille. Pitkien N-grammien ongelmana on kuitenkin, että niitä esiintyy opetusaineistossa harvoin eikä näin ollen ennustus onnistuisi (kuin ylisovittumalla ko. lauseeseen).

5.3 Sädehaku-malli

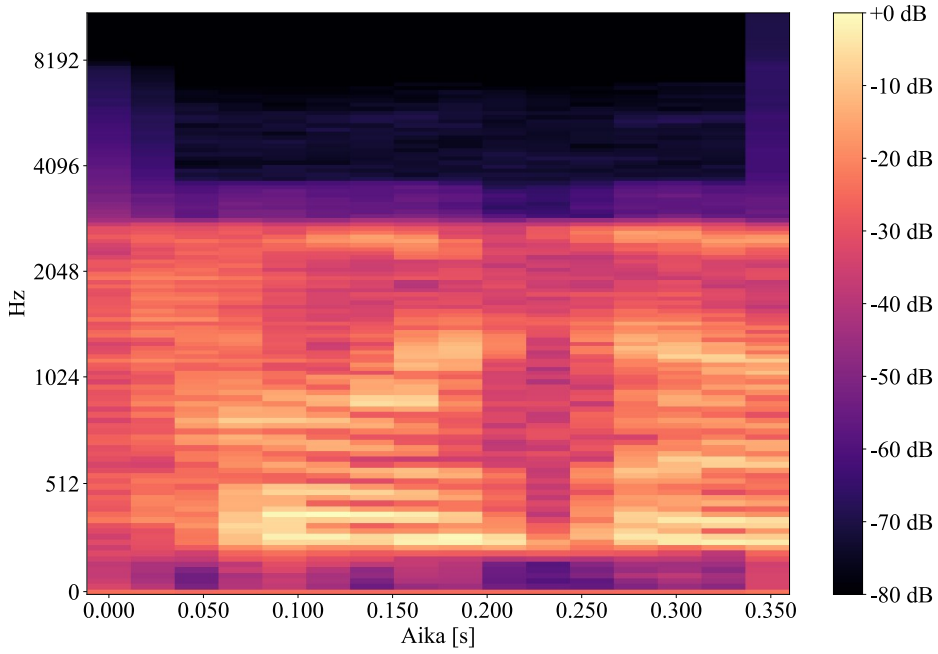
Edellä esitelty maksimihaku valitsi todennäköisimmän sanan lauseeseen. Sädehaku (eng. beam search) tarkastelee koko tiedossa olevaa sanajonoa ja valitsee sen perusteella korkeimman todennäköisyyden tuottavan yhdistelmän seuraavalle sanalle. Säteenleveysparametrin (eng. beam width) avulla voidaan määrittää, kuinka monta sanakandidaattia huomioidaan. Sädehaku voidaan toteuttaa perustuen pitkäkestoinen-lyhytmuistirakenteilla toteutettuihin takaisinkytkettyihin neuroverkkoihin tai muuntajarakenteisiin.

6 Testijärjestely

Tätä työtä varten tallennettiin noin 100 tuntia radioamatöörien puhelähetyksiä, tarkemmat yksityiskohdat luvussa 6.1. Tästä määrästä ehdittiin litteroida käsin noin 121 minuuttia vertailuaineistoksi ja noin tunnin verran opetusaineistoksi, ks. luku 6.2. Luvussa 6.3 esitellään puheentunnistusneuroverkkojen kouluttamista Eduskunnassa vuosina 2008–2020 tallennettujen ja litteroitujen kansanedustajien puheenvuorojen (noin 3300 tunnin verran) avulla.

6.1 Puheentallennus neuroverkolle

Radioamatööripuhelähetteiden vaatima taajuuskaista on pyritty pitämään kapeana, jotta useat radioamatöörit voisivat pitää yhteyksiä yhtäaikaisesti radioamatöörien käyttöön varatuilla taajuuskaistoilla (ks. Liikenne- ja viestintäviraston ylläpitämä taajuusjakotaulukko [93]). Tässä työssä on käytetty materiaalina radioamatöörien 3,6 MHz:n ympäristössä (useimmat lähetteet tallennettiin 3,699 MHz taajuudella) lähettämiä sivunauhamoduloituja (eng. single sideband) puhelähetyksiä, joiden kaistanleveys on rajoittunut noin 3,4 kHz:in, kuten Mel-spektrogrammiesityksestä kuvasta 32 havaitaan (teho on painottunut alle 3 kHz:n taajuuksille, punaisin alue).

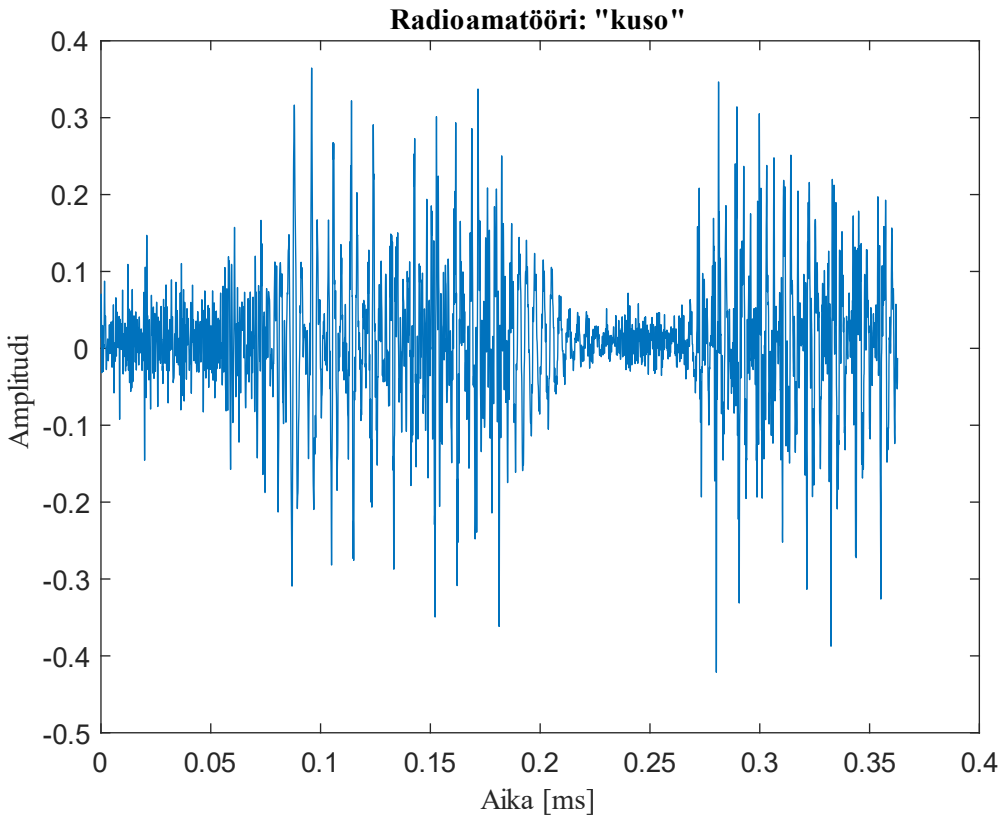


Kuva 32 Tallenteen 40100 ”kuso”-sanan spektrogrammikuvaajassa voidaan havaita radioamatöörilähetyksen puhesignaalin 3 kHz korkeammilla taajuuksilla olevan energian suodattuminen.

Edellä esitetty tekninen kaistarajoitus vähentää puheen ymmärrettävyyttä, koska puhesignaalin kaistanleveys on suurempi. Tämä on havaittavissa, kun vertaa edellä kuvan 32 Mel-spektrogrammikuvaajan tehon jakautumista yli 3 kHz:n taajuuksilla

esimerkiksi mikrofonilla tallennettujen puhesignaalien kuvien 4 ja 5 Mel-spektrogrammiesitysten tehon jakautumiseen.

Radioamatööripuheen aaltomuodossa voidaan myös havaita hieman kohinaa pienillä amplitudeilla etenkin tallennuksen alussa (n. 0…0,05 ms), ks. kuva 33. Tätä voi verrata luvussa 2.2 esitettyihin kuviin mikrofonilla tallennetuista signaaleista. Niissä signaalin amplitudi on suurempi, joten signaali-kohina-suhde on myös suurempi.

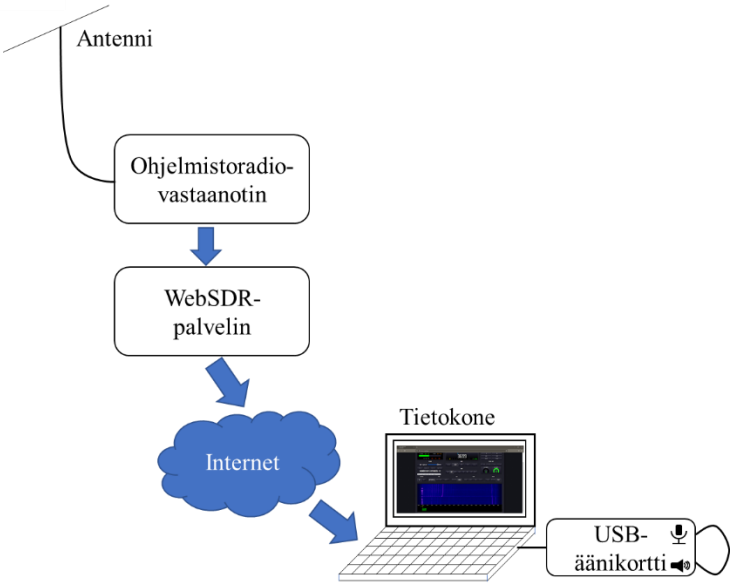


Kuva 33 Tallenteen 40100 ”kuso”-sanan aaltomuodon esitys aika-alueessa.

Radioamatöörien käyttämällä 3,5 MHz:n taajuusalueella radiosignaali heijastuu auringon säteilyn ionisoimasta ilmakehän molekyylikerroksesta (ns. D-kerros 50–90 km korkeudessa) päivällä, mutta yöllä D-kerros katoaa ja radioaallot pääsevät etenemään korkeammalle ennen heijastumistaan F-kerroksesta (160-400 km korkeudessa). Näin ollen 3.5 MHz:n taajuusalue soveltuu päivällä kotimaan radioamatööriyhteyksien pitämiseen ja yöllä kansainvälisien yhteyksien pitämiseen [94]. Tämä tarkoitti tutkimuksen kannalta sitä, että suomalaisten radioamatöörien suomeksi pitämiä yhteyksiä kyettiin tallentamaan kello 6–22 välisenä aikana ilman häiritseviä lähetyksiä, esimerkiksi lähellä Suomen rajaa sijaitsevilta venäläisiltä asemilta.

Radioamatööripuhelähetteiden tallennukset tehtiin tallentamalla Internetiin liitetyn radioamatööriaseman vastaanottamaa lähetystä tietokoneella, johon oli kytketty USB-äänikortti [95], jonka ulostulo oli liitetty sisääntuloon (eli kaiutin- ja mikrofoniliittimien välille kytketty johto 3,5 mm kaapelilla). Tämä tehtiin siksi, ettei radioamatööriaseman

vastaanotinpalvelun jakamiseen kehitetty selaimella ohjattava WebSDR-ohjelmisto [96] (Internet-pohjainen ohjelmistoradio, eng. Software Defined Radio) tarjonnut suoratoiston tallennukseen rajapintaa. Lisäksi ohjelmassa oli mahdollista käyttää kohinasalpaa, jonka tarkoitus on estää vastaanotinta toistamasta pelkkää kohinaa ja avautua ainoastaan lähetyksen aikana. Tämä oli käytössä ensimmäisissä tallennuksissa, mutta poistettiin käytöstä jälkimmäisissä, jotta voidaan tutkia erittäin kohinaisen puheen tunnistamista.



Kuva 34 Puhelähetysten tallentamiseen käytetty mittauskytkentä.

Äänikortille voidaan määrittää sen digitointiin käyttämä bittien määrä ja näytteenottonopeus (tai -taajuus). Näihin tallennuksiin määritettiin 16 bittiä ja näytteenottotaajuudeksi 16 kHz, jotka ovat käytössä esimerkiksi LibriSpeech-näytekannassa (korpuksessa) [97]. Tämä mahdollistaa sellaisten neuroverkkojen käyttämisen siirto-oppimiseen, jotka on koulutettu hyödyntäen näitä edellä mainituilla ominaisuuksilla tallennettuja ja opetukseen käytettyjä aineistoja. Tallennukset tehtiin liitteessä 2 esitetyllä Python-ohjelmakoodilla.

Äänikorttiin perustuvan mittausjärjestelyn olennaiset ongelmat liittyvät suurempaan näytteistysnopeuteen kuin mitä varsinainen puhelähete on sekä sopivan äänenvoimakkuustason valitsemiseen. Koska äänikortin mikrofoni- ja kaiutinlinjat on kytketty yhteen, niin äänikortin lähtöamplitudi täytyy säätää sopivaksi tarkkailemalla äänitallenteen aaltomuotoa, jotta se ei leikkaudu.

Taulukko 3 Työssä käytettyjen tallenteiden luontipäivämäärät ja numerointi sekä huomiot.

Päivämäärä	Tallenteen juokseva numerointi	Kommentit
27.6.2021	0…469	Kohinasalpa päällä, minuutin tallenteet
24.6.2022	40000…40290	Kohinasalpa pois päältä, kahden eri maantieteellisissä sijainneissa olleiden asemien yhtäaikaiset tallenteet, 30 sekunnin tallenteet

6.2 Puheen litterointi neuroverkolle

Tallennettujen radioamatööripuhelähetysten koneellista automatisoitua litterointia kokeiltiin ensin Pythonille kehitetyllä SpeechRecognition-kirjastolla [27] käyttäen alkuun Googlen kehittämää Speech Recognition -tunnistinta. Jotta kaikki noin 10 000 tallennusminuuttia saatiin litteroitua koneellisesti järkevässä ajassa, niin useampia prosesseja käynnistettiin eri terminaali-ikkunoihin. Tämä tehtiin, jotta saataisiin käsitys kaupallisten toimijoiden puheentunnistimien suorituskyvystä ja mahdollisista kehityskohteista. Lisäksi työssä myöhemmin kehitettävän puheentunnistusverkon suorituskykyä voitaisiin verrata näihin.

Seuraavaksi tehtiin käsin litterointi kuuntelemalla ja kirjoittamalla tekstiksi. Yhden minuutin litterointiin meni aikaa noin 10–15 minuuttia riippuen puhujien määrästä ja puheenvuorojen kestosta (osassa kohinasalpa oli leikannut vasta-aseman vastauksen pois ja osassa oli koko minuutin ajan puhetta).

Tätä tulosta verrattiin edellä tehtyyn automaattiseen litterointiin Googlen palvelulla ja todettiin, että laatu oli niin heikko (ks. luku 7.1), ettei automaattista tunnistusta voitu käyttää apuna. Tätä voisi tutkia lisää eli olisiko mahdollista kehittää sellainen tunnistin, joka parantaisi tulosta koko ajan, kun käyttäjä litteroisi uutta aineistoa. Muuten puheen muuttaminen tekstiksi muodostaa työläimmän vaiheen prosessissa eikä käyttäjämäärältään pienten kielten puheentunnistusta ole mahdollista kehittää ilman korkeita kustannuksia (vrt. aiemmin mainittu Amazon Mechanical Turk -palvelu [28]).

Lopuksi rakennettiin koulutusaineisto tallentaen Audacity-ohjelmalla eri puhujien puheenvuorot omiin tiedostoihinsa ja kirjoittamalla litteroinnit. Näitä käytettiin

6.3 Neuroverkkojen koulutus

Tässä työssä käytetään NVIDIA:n englanninkielisten aineistojen pohjalta kouluttamia QuartzNet- ja Conformer-CTC-neuroverkkoja. Ne pyritään siirtokouluttamaan kansanedustajien vuosina 2008–2020 Eduskunnassa esittämien puheenvuorojen avulla.

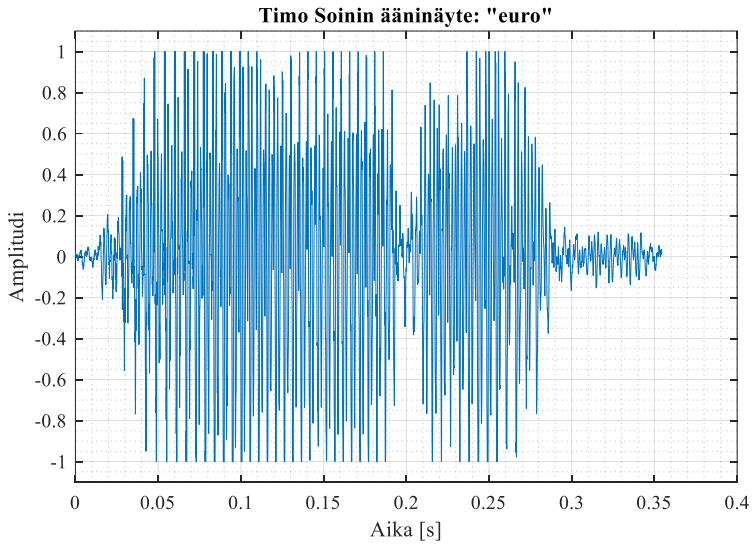
6.3.1 Käytetty Eduskunnan koulutusaineisto

Eduskunnan litteroitu puheaineisto vuosilta 2008–2020 (n. 3130 tuntia, 19 356 831 erillistä sanaa, 1 422 318 erillistä litteroitua äänitiedostoa, noin 311 Gt pakattuna

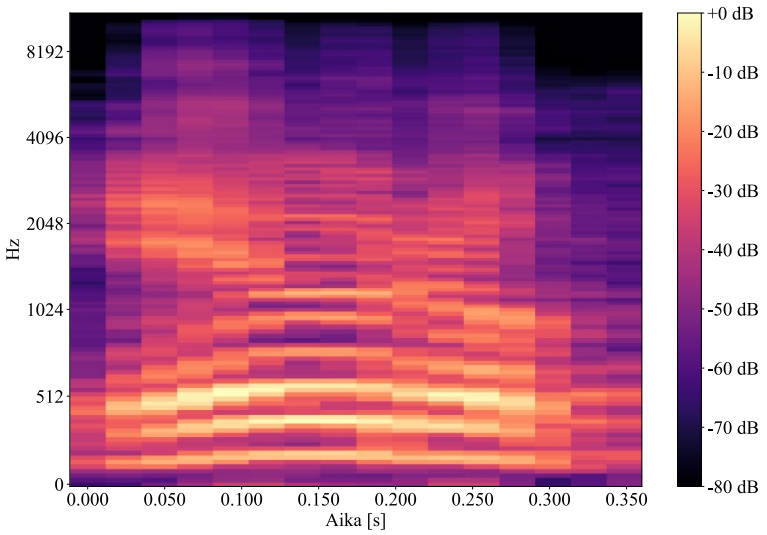
kahdeksaan pakettiin, ks. lähteet [98] ja [99]) jaettiin opetusaineistoon ja testiaineistoon. Tätä varten luettiin ensin jokaisen äänitiedoston polku Python-skriptillä tekstitiedostoon ja jaoteltiin yksittäisen kansanedustajan äänitiedostoista ensimmäiset 20 prosenttia testitiedostoiksi ja jälkimmäiset 80 prosenttia opetustiedostoiksi. Tämän jaon tarkoitus on varmistaa, että jokaisen puhujan ainutlaatuisia äänitiedostoja käytetään sekä opetuksessa, että mallin toiminnan todennuksessa. Täysin satunnainen otanta saattaisi tuottaa eroja opetus- ja todennusaineistojen välille ja huonontaa puheentunnistusneuroverkon tunnistustulosta.

Esimerkki eduskunnan puheaineiston ääninäytteistä on Timo Soinin lausuma ”euro”. Nämä tallenteet ovat vähäkohinaisempia verrattuna radioamatööriläheteyksiin, ks. aaltomuodon esitys aikatasossa kuvassa 36 (ylempi kuvaaja). Näiden näytteiden signaali-kohina-suhde on parempi kuin aiemmin radioamatööripuheen vastaavissa tallennuksista esitetyissä kuvaajissa. Lisäksi tehoa on havaittavissa 3…10 kHz taajuuksilla, ks. spektrogrammikuvaaja kuvassa 36 (alempi kuvaaja).

NVIDIA:n puheentunnistusmallit mahdollistavat omien ääniaineistojen käytön puheentunnistuksen opettamiseen. Tätä varten on kuitenkin luotava ns. manifestitiedosto, joka sisältää kuvauksen jokaisen opetukseen käytetyn äänitiedoston parametreista tai liitännäistiedosta (eng. metadata). Tämä tiedosto on JSON-tyyppinen ja sen rivien tiedostomuoto on `{"audio_filepath": "/path/to/audio.wav", "text": "the transcription of the utterance", "duration": 23.147}`, missä on ensin tarkka tiedoston sijainti hakemistorakenteessa, sitten puhe litteroituna tekstiksi ja lopuksi äänitiedoston pituus. Narvi-laskentapalvelimelta varattiin prosessori laatimaan manifestitiedostot ja testimanifestin laatiminen vei aikaa 44 minuuttia ja opetusmanifestin 115 minuuttia.



Kuva 35 Timo Soinin lausuman sanan ”euro” aikatason esitys. Huomaa aaltomuodon amplitudin leikkaantuminen -1…+1 välille.



Kuva 36 Timo Soinin lausuman sanan ”euro” Mel-spektrogrammikuvaaja.

Eduskunnan puheiden manifestitiedostoista tuli suuria, yli 200 Mt. Tämän vuoksi niiden avaaminen tekstieditorissa oli hidasta tai mahdotonta. Mikäli tiedostosta löytyi virhe, niin tätä varten koodattiin Python-skripti, joka haki kyseisen rivin tiedostossa esimerkiksi rivillä olevien merkkien hakemiseen tehtyä ”Regular Expression” -kirjastoa hyväksikäyttäen tai mikäli rivinumero oli tiedossa niin sen avulla.

6.3.2 Litteroitu radioamatöörien puheaineisto

Radioamatöörien puheaineistosta litteroitiin 450 puheenvuoroa Audacity-äänikäsittelysovellusta, yhteensä noin 49 minuuttia. Tämä jaettiin opetus- ja testiaineistoksi kuten edellä suhteessa 80/20 % eli 40 minuuttia opetusaineistoksi ja kahdeksan minuuttia testiaineistoksi. Jos yksittäiseltä puhujalta ei ollut riittävästi puheenvuoroja, niin puheita käytettiin vain koulutukseen.

6.3.3 QuartzNet-puheentunnistusneuroverkkomallin asetukset

Tässä työssä käytettiin QuartzNet 15x5 -puheentunnistusneuroverkkomallin asetuksina NVIDIA:n oletusasetuksia (ks. [100]). Kuitenkin opetusnopeusparametria pienennettiin siirto-oppimisen vuoksi, koska painokertoimia ei haluttu muuttaa merkittävästi, sillä uskottiin, että verkko oli oppinut jo englannin äänteet.

QuartzNet-mallin erän kokona (eng. batch size) käytettiin 32, ks. yleiset asetukset taulukossa 4. NVIDIA suosittelee käyttämään mahdollisimman suurta lukua, jotta grafiikkakiihdytinkortin muistin käyttö maksimoituu, ks. NVIDIA:n keskustelualue [101]. QuartzNet-malli osaa poistaa äänitiedoston alussa ja lopussa olevat hiljaiset hetket. Tällä asetuksella voidaan kohdistaa oppiminen äänitiedoston kiinnostavaan osaan. Mallissa on asetettu leikkuri leikkaamaan yli 16,7 sekunnin mittaiset äänitiedostot pois opetusmateriaalista (NVIDIA:n NeMo-keskustelualueella suositellaan maksimissaan 20 sekunnin opetusäänitiedostojen käyttämistä, ks. lähde [102]). Lisäksi mallissa voidaan käyttää

opetusaineiston sekoitusta (eng. shuffle). Tästä on kuitenkin haittaa mallin suorituskyvyn todentamisessa ja siksi se ei ole käytössä validoinnissa. Tässä työssä ei muutettu näitä asetuksia tai tutkittu näiden vaikutusta puheentunnistustulokseen.

Taulukko 4 QuartzNet 15x5 -mallin yleiset asetukset

Parametri	**Asetus**
Erän koko	32
Hiljaisten hetkien poisto	Käytössä
Pisin äänitiedoston pituus	16,7 sekuntia
Opetusaineiston sekoitus	Käytössä opetuksessa, ei validoinnissa

QuartzNet-puheentunnistusneuroverkkomallissa on esikäsittelijä, jonka asetuksia voi muuttaa. NVIDIA on määritellyt näytteistysikkunan (josta lasketaan tunnistettavan äänen spektri, ks. luku 10) pituudeksi 20 ms ja ikkunan askellukseksi 10 ms, ks. esikäsittelijän asetukset koottuna taulukossa 5. Ikkunointifunktiona käytetään Hann-ikkunaa ja piirrevektori lasketaan 64 Mel-suodattimen perusteella, nopean Fourier-muunnoksen (eng. Fast Fourier Transform, FFT) ikkunan pituus on 512 taajuusresoluutioalkiota (eng. frequency bin). Lisäksi malli lisää valkoista kohinaa näytespektreihin (eng. dither). Tämän tarkoitus on vähentää (satunnaistaa) näytteistyksen kvantisoinnista aiheutuvaa vääristymää. Tässä työssä ei muutettu näitä asetuksia tai tutkittu näiden vaikutusta puheentunnistustulokseen.

Taulukko 5 QuartzNet 15x5 -mallin esikäsittelijän asetukset.

Parametri	**Asetus**
Ikkunan pituus	0,02 sekuntia
Ikkunan askellus	0,01 sekuntia
Ikkunafunktio	Hann
Piirrevektori	64 MEL
FFT-ikkunan pituus N	512
Kehysten halkaisu	1
Valkoisen kohinan lisäys	0,00001
STFT	Ei käytössä

Yleisesti neuroverkkojen opetusaineistoja kannattaa muuttaa satunnaisesti ja siten ehkäistä mahdollista mallin ylioppimista. Tällä pyritään samaan lopputulokseen kuin luvussa 3.3.6 esitellyllä neuronisammutuksella. QuartzNet-puheentunnistusneuroverkkomallin spektrogrammeja on mahdollista muokata opetusta varten (esimerkiksi peittämällä osan spektrogrammia) joko taajuuden tai ajan suhteen (eng. spectrogram augmentation), ks. asetukset taulukossa 6. Tämä menetelmä on esitelty lähteessä [103].

Taulukko 6 QuartzNet 15x5 -mallin spektrogrammin muokkauksen asetukset.

Parametri	**Asetus**
Suorakaideleikkaus taajuuden suhteen	50
Suorakaideleikkausten määrä	5
Suorakaideleikkaus ajan suhteen	120

QuartzNet-puheentunnistusneuroverkkomallin koodain käyttää edellä mainittuja 64 Mel-piirrettä ja aktivaatiofunktionaan tasasuunnattua lineaariyksikköfunktiota. Lisäksi käytössä on konvoluutioiden peittäminen (ks. lähde [104]) eli koodainneuroverkko ei näe tulevia spektrogrammin pikseleitä eikä voi siten käyttää niitä päättelyssään.

Työssä käytetyn QuartzNet-mallin toistuvien lohkojen kanavien määrät, pikseli-ikkunan koot (eng. kernel), lomituksen (eng. stride) ja konvoluutioikkunan laajennuksen (eng. dilation) asetukset on esitetty taulukoissa 7-12. Lisäksi on mahdollista säätää neuronisammutuksen määrää, ottaa residuaalihaara (ks. lohkojen ohitushaarat kuvassa 29), QuartzNet-lohkoissa on käytössä konvoluution eroteltavuus erotuksena Jasper-verkkoon (ks. luku 3.5.8) ja puristusheräte (ks. luku 3.5.9) käyttöön (asetettu laskemaan kaikkien aika-askeleiden yli eli kontekstin koko on -1).

Taulukko 7 QuartzNet 15x5 -mallin lohkon 1 asetukset.

Parametri	**Asetus**
Kanavien määrä	128
Toisto	1 krt
Pikseli-ikkunan koko	11
Lomitus	1
Konvoluutioikkunan laajennus	1
Neuronisammutus	0.0
Residuaalihaara	Käytössä
Eroteltavissa olevuus	Käytössä
Puristus-heräte	Käytössä
Puristus-heräte kontekstin koko	-1

Taulukko 8 QuartzNet 15x5 -mallin lohkon 2 asetukset.

Parametri	**Asetus**
Kanavien määrä	256
Toisto	1 krt
Pikseli-ikkunan koko	13
Residuaalihaara	Käytössä

Taulukko 9 QuartzNet 15x5 -mallin lohkon 3 asetukset.

Parametri	**Asetus**
Kanavien määrä	256
Toisto	1 krt
Pikseli-ikkunan koko	15
Residuaalihaara	Käytössä

Taulukko 10 QuartzNet 15x5 -mallin lohkon 4 asetukset.

Parametri	**Asetus**
Kanavien määrä	256
Toisto	1 krt
Pikseli-ikkunan koko	17
Residuaalihaara	Käytössä

Taulukko 11 QuartzNet 15x5 -mallin lohkon 5 asetukset.

Parametri	**Asetus**
Kanavien määrä	256
Toisto	1 krt
Pikseli-ikkunan koko	19
Residuaalihaara	Käytössä

Taulukko 12 QuartzNet 15x5 -mallin lohkon 6 asetukset.

Parametri	**Asetus**
Kanavien määrä	256
Toisto	1 krt
Pikseli-ikkunan koko	21
Residuaalihaara	Ei käytössä

QuartzNet-mallin viimeisenä lohkona on koodinpurkainlohko (eng. decoder), ks. kuva 29. Siinä on 1024 kanavaa (eng. filters) ja sen pikseli-ikkuna on yhden pikselin kokoinen. Siinä ei ole residuaalihaaraa. QuartzNet-mallin oppimisnopeusoptimointiin käytettiin NovoGrad-algoritmia, ks. luku 3.3.7. Sen parametrit on esitetty taulukossa 13.

Taulukko 13 QuartzNet 15x5 -mallin oppimisnopeusoptimoijan asetukset.

Parametri	**Asetus**
Oppimisnopeus	0,001
Beta-parametrit	0,8 ja 0,5
Painokertoimien pieneneminen	0,001
Oppimisnopeuden vuorottaja	CosineAnnealing
Tarkkailtava arvo (PyTorch Lightning)	Todennustappio
Oppimisnopeuden pienennys muuttumattomassa tilanteessa (eng. plateau)	Ei käytössä
Lämmittelyaskeleet	Null
Lämmittelysuhde	Null

6.3.4 QuartzNet-puheentunnistusneuroverkon koulutus

Yliopiston Narvi-laskentapalvelimelta varattiin laskentaresursseja SLURM-nimisellä taakanjako-ohjelmalla [105] Pythonilla toteutetuille laskentaskripteilleni. Laskentaa varten pyydettiin käyttöoikeus grafiikkakiihdytinkorttiryhmään. Tämän jälkeen käytettävissäni oli esimerkiksi Tesla V100 -laskentakortteja 16 kappaletta maksimissaan seitsemän vuorokauden ajan (kerrallaan yhdelle työlle). Valitettavasti näin monen kortin saaminen käyttöön vaatisi, että kortit olisivat yhtä aikaa vapaana, joten yhden laskentakortin saa käyttöönsä lähes välittömästi ja useamman saaminen käyttöön voi olla käytännössä mahdotonta, koska muilla käyttäjillä on ne varattuna.

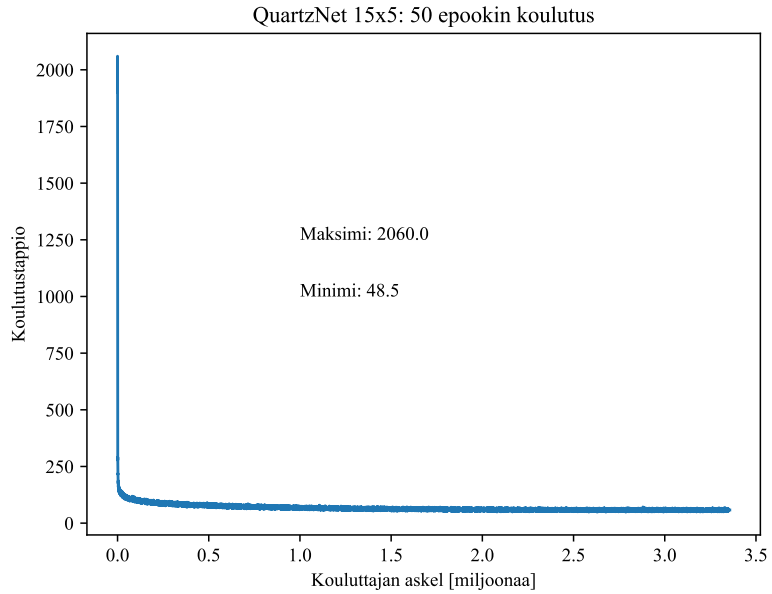
Koulutus aloitettiin NVIDIA:n kehittämästä QuartzNet-neuroverkosta hyödyntäen alun perin englanninkielisellä puheella (LibriSpeech äänikirjatietokanta, noin 1000 tuntia puhetta [97]) koulutettua puheentunnistusneuroverkkoa eli tehtiin kertoimien hienosäätö (siirto-oppiminen, eng. transfer learning, ks. luku 3.1.3) eduskunnan litteroidulla puheaineistolla. Opetusta varten lisättiin käytettyyn merkistöön ”ä”, ”ö” ja ”å” merkit sekä poistettiin heittomerkki. Tämä aiheutti käytännössä koodinpurkaimen uudelleen koulutustarpeen.

Taulukko 14 QuartzNet-neuroverkon koulutettavissa olevat parametrit.

Moduuli	**Nimi**	**Parametrit**
Esikäsittelijä	AudioToMelSpectrogramPreprocessor	0
Koodain	ConvASREncoder	1 200 000
Koodinpurkaja	ConvASRDecoder	31 800
Tappio	CTCLoss	0
Spektrogrammin muuntelu	SpectrogramAugmentation	0
Sanavirhesuhde	WER	0

NeMo-kirjaston QuartzNet-koulutin (eng. trainer) ilmoittaa skriptin suorituksen alussa, että mallissa on noin 1,2 miljoonaa koulutettavissa olevaa parametria, ks. taulukko 14. Tämä poikkeaa aiemmin luvussa 4.1.3 esitetystä 18,9 miljoonasta luultavasti siksi, että koulutus aloitettiin esikoulutetulla verkolla.

Opetusta aloitettaessa osoittautui, että manifestitiedostot käyttivät unicode-merkistöä, joten ne piti ajaa uudelleen käyttäen UTF-8-merkistöä. Eduskunnan puheaineistosta käytettiin opetukseen 1477317 äänitiedostoa puhetta eli yhteensä 2864,73 tuntia. Äänitiedostojen pituuden ylittymisen vuoksi 51823 tiedostoa (321,99 tuntia) suodatettiin pois opetusaineistosta. Testiaineisto sisälsi 382623 tiedostoa eli yhteensä 786,95 tuntia. Aineiston yhden epookin laskemiseen meni yhdellä grafiikkakiihdytinkortilta (eng. Graphical Processing Unit) noin viisi tuntia. Jostain syystä kolmannen epookin laskeminen kaatui yhdellä grafiikkakiihdytinkortilla.



Kuva 37 QuartzNet neuroverkon koulutustappion pieneneminen suppeni arvoon 48,5 kouluttimen (eng. trainer) askeleiden ylittäessä noin 1 500 000 askelta.

Kahdella grafiikkakiihdytinkortilla viiden epookin laskeminen onnistui yhdeksässä tunnissa ja 15 minuutissa ja 50 epookin laskentaan kului aikaa 75 tuntia 53 minuuttia (QuartzNet-artikkelin kirjoittajat olivat kouluttaneet 400 epookkia kahdeksalla V100-graafiikkakiihdytinkortilla viiden päivän ajan, ks. [73]). Koulutustappio suppeni arvoon 48,5 ja se ei enää merkittävästi pienentynyt noin 1,5 miljoonan askeleen jälkeen, ks. kuva 37. Mielestäni tästä voidaan päätellä, että neuroverkon koulutus saavutti lakipisteensä eikä koulutusta jatkamalla olisi kyetty parantamaan suorituskykyä. NVIDIA:n keskustelupalstan vastauksesta (ks. S. Majumdarin kommentti [87]) ilmeni, että CTC:hen perustuvien puheentunnistusmallien koulutustappio jää johonkin nollasta poikkeavaan arvoon, koska malli käyttää edellä mainittua spektrogrammin osien peittämistä (eng. SpecAugment [103]), mikä estää ylioppimista.

Yritin käyttää kahdeksaa grafiikkakiihdytinkorttia laskentaan, mutta tämä kaatui minulle tuntemattomasta syystä – mahdollisesti eri laskentakortit alustuivat (käynnistyivät) väärässä järjestyksessä laskennan jakamisen kannalta. Yliopiston laskentapalvelimella on neljä grafiikkakiihdytinkorttia samassa emolevyssä (solmussa) ja lisäksi eri solmujen välillä on erityyppisiä laskentakortteja, joten on mahdollista, että eri solmuissa olevat kortit alustuivat eri aikaisesti tai kortit olivat erityyppisiä.

6.3.1 Conformer-CTC-puheentunnistusneuroverkkomallin asetukset

Conformer-mallina (ks. luku 4.3) käytettiin NVIDIA:n NeMo-keskustelutekoälykirjaston (eng. Neural Module) Conformer-CTC-puheentunnistusmallia. Sen asetustiedostoon [106] tehtiin samat merkistömuutokset kuin QuartzNet-mallille edellä eli lisättiin ”ä”, ”ö” ja ”å” sekä poistettiin heittomerkki. Mallissa on asetettu leikkuri leikkaamaan yli 16,7 sekunnin mittaiset äänitiedostot pois opetusmateriaalista. Lisäksi mallissa voidaan käyttää opetusaineiston sekoitusta (eng. shuffle). Tästä on kuitenkin haittaa mallin suorituskyvyn todentamisessa ja siksi se ei ole käytössä validoinnissa. Tässä työssä ei muutettu näitä asetuksia tai tutkittu näiden vaikutusta puheentunnistustulokseen.

Taulukko 15 Conformer-CTC-mallin yleiset asetukset

Parametri	**Asetus**
Erän koko	16
Hiljaisten hetkien poisto	Ei käytössä
Pisin äänitiedoston pituus	16,7 sekuntia
Lyhin äänitiedoston pituus	0,1 sekuntia
Opetusaineiston sekoitus	Käytössä opetuksessa, ei validoinnissa
Opetusaineiston sekoituksen koko *n*	2048

Taulukko 16 Conformer-CTC-mallin esikäsittelijän asetukset.

Parametri	**Asetus**
Ikkunan pituus	0,025 sekuntia
Ikkunan askellus	0,01 sekuntia
Ikkunafunktio	Hann
Piirrevektori	80
FFT-ikkunan pituus *N*	512
Kehysten halkaisu	1
Valkoisen kohinan lisäys	0,00001
STFT	Ei käytössä

Taulukko 17 Conformer-CTC-mallin spektrogrammin muokkauksen asetukset.

Parametri	**Asetus**
Taajuuspeitteiden lukumäärä	2
Aikapeitteiden lukumäärä	10
Taajuuspeitteen leveys	27
Aikapeitteiden leveys	0.05

Conformer-CTC-mallin koodaimessa on sisääntulossa syötteinä merkistö ja siinä on 16 Conformer-lohkoa ja mallin piilotetun ulostulon koko on 256. Alinäytteistyksenä pienentämään tarvittavaa laskentaa käytettiin askellusta (eng. striding), ks. lähde [107] kertoimella 4. Eteenpäinkytkevän moduulin laajenemiskerroin oli 4 ja monipäisen sisähuomiomoduulin päitä oli kahdeksan. Konvoluutiomoduulin ikkunan koko oli 31 ja konvoluutionormalisointi tehtiin eräajotyyppisesti. Regularisaationa useimmissa Conformer-lohkoissa käytettiin neuronisammutusta arvolla 0,1 kuten myös ennen koodainta. Paikkakoodauksissa käytettiin arvoa 0,0 ja monipäisessä sisähuomiomoduulissa myös arvoa 0,1. Conformer-CTC-mallin oppimisnopeusoptimointiin käytettiin painokertoimien pienemisen suhteen optimoitua AdamW-algoritmia, ks. luku 3.3.7 ja lähde [108]. Sen parametrit on esitetty taulukossa 13.

Taulukko 18 Conformer-CTC-mallin AdamW-oppimisnopeusoptimoijan asetukset.

Parametri	**Asetus**
Oppimisnopeus	2,0
Beta-parametrit	0,9 ja 0,98
Painokertoimien pieneneminen	0,001
Oppimisnopeuden vuorottaja	NoamAnnealing
Lämmittelyaskeleet	10 000
Lämmittelysuhde	Null
Pienin oppimisnopeus	1e-6

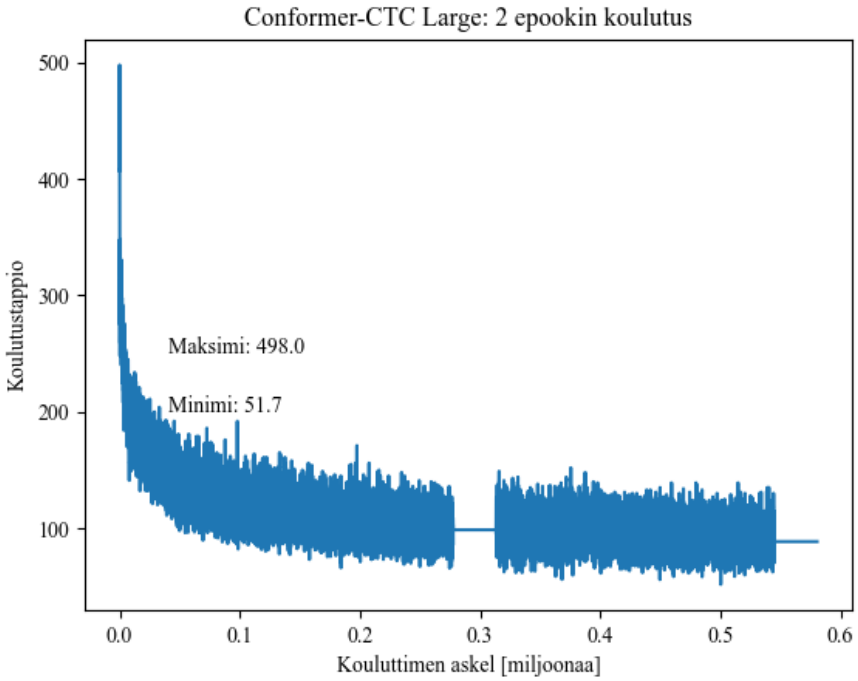
Conformer-CTC-kirjoitusmerkkipuheentunnistusneuroverkkomallin koulutuksen saamiseksi käyntiin jouduttiin tekemään jonkin verran enemmän töitä – osittain siksi, ettei tästä mallista ollut esimerkkitoteutusta tarjolla NVIDIA:n kotisivuilla. Näiden mallien asetusten määrittely on hieman työlästä, sillä esikoulutettu verkko sisältää alkuasetukset ja näitä tulee sitten muokata joko asetustiedostossa tai komennettaessa Python-skriptiä komentoriviparametrina.

NeMo-kirjaston Conformer-CTC-koulutin ilmoittaa skriptin suorituksen alussa, että mallissa on noin 27,3 miljoonaa koulutettavissa olevaa parametria, ks. Taulukko 14. Tämä poikkeaa aiemmin luvussa 4.1.3 esitetystä 118 miljoonasta luultavasti siksi, että koulutus aloitettiin esikoulutetulla verkolla.

Taulukko 19 Conformer-CTC-neuroverkon koulutettavissa olevat parametrit.

Moduuli	**Nimi**	**Parametrit**
Esikäsittelijä	AudioToMelSpectrogramPreprocessor	0
Koodain	ConvASREncoder	27 300 000
Koodinpurkaja	ConvASRDecoder	8 000
Tappio	CTCLoss	0
Spektrogrammin muuntelu	SpectrogramAugmentation	0
Sanavirhesuhde	WER	0

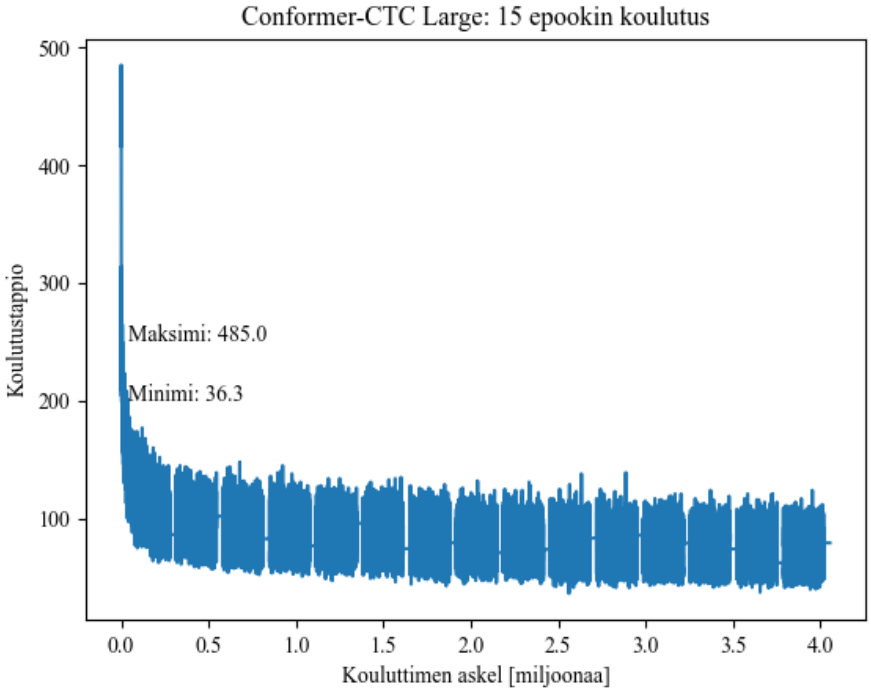
Lopulta sain koulutuksen käyntiin samalla edellä mainitulla eduskunnan puheaineistolla. Aluksi kokeiltiin koulutusta viidellä epookilla ja kahdella grafiikkakiihdytinkortilla. Jostain syystä koulutus kuitenkin kaatui ensimmäisen epookin laskennan loppupuolella ilmeisesti Slurm-skriptin keskeyttämänä (mahdollisesti muistin loppumiseen). Tähän mennessä oli kulunut aikaa seitsemän tuntia ja 12 minuuttia.



Kuva 38 Conformer-CTC Large -mallin koulutustappion suppeneminen kahden epookin aikana Eduskunnan puheaineiston koulutuksessa.

Conformer-CTC Large -malli suppeni kahden epookin aikana kohtuullisesti eikä koulutustappio enää juurikaan pienentynyt epookin loppupuolella, ks. kuva 38. Jatkoin kokeiluja ja sain ajettua 15 epookkia sekä Eduskunnan puhemateriaalia opetusaineistona käyttäen että litteroimaani radioamatööriaineistoa lisäksi käyttäen. Aikaa em. koulutukseen kului yhteensä viisi vuorokautta ja 14 tuntia. Jos simulointiaikaa olisi ollut enemmän, niin olisi voitu ajaa pidempiä (ehkä satojen epookkien) testejä. Näin ollen voidaan päätellä, että monimutkaisten neuroverkkojen koulutuksessa käytännöllisessä ajassa

tarvitaan useita grafiikkakiihdytinkortteja laskennan nopeuttamiseksi. Mahdollisesti johtuen spektrogrammin satunnaispeitosta (em. SpecAugment lähteessä [103]) koulutustappio jäi värähtelemään noin 36…150 välille, ks. käyrä kuvassa 39.



Kuva 39 Conformer-CTC Large -mallin koulutustappion suppeneminen 15 epookin aikana Eduskunnan ja litteroidun radioamatööriaineiston koulutuksessa.

Verrattuna aiempaan QuartzNet-malliin ja sen suppenemiseen (ks. kuva 37) Conformer-mallin suppeneminen tapahtui vähemmillä kouluttimen askeleilla. Näin ollen transformer-mallin koulutus on nopeampaa kuin konvolutiivisiin ja lineaarisiin neuroverkkoihin perustuvan QuartzNet-mallin.

7 Kokeelliset tulokset

7.1 Litterointitulokset Googlen ja Microsoftin palveluissa

Googlen SpeechRecognition -kirjastoa käyttänyt puheentunnistin saavutti ensimmäisten tallennettujen 121 minuutin ajalta 9875 sanan tunnistuksessa sanavirhesuhteen 78,30 % (2214 sanaa oikein, 1881 korvattua, 5780 poistettua/tunnistamatta jäänyttä ja 71 lisättyä sanaa) ja Microsoft Word Onlinen tarjoama litterointi hieman pienemmän 64,17 % (4465 sanaa oikein, 4502 korvattua, 908 tunnistamatta jäänyttä ja 927 lisättyä). Näissä tallenteissa (0…120) kohinasalpa oli käytössä, joten osa kohinaan häipyneistä sanoista ei ollut käytettävissä. Microsoftin palvelu osasi lisäksi tunnistaa puhujien vaihtumisen.

7.2 Litterointitulokset käyttäen siirtokoulutettua neuroverkkoa

7.2.1 QuartzNet-litterointitulokset eduskunnan aineistolla

Quartznet-puheentunnistusneuroverkko siirtokoulutettiin eduskunnan litteroidulla puheaineistolla. Ensin kokeiltiin miten englanniksi koulutettu neuroverkko muuttaa Aila Paloniemen puheenvuoron: *”venäjä on osallistunut korkean tason valtuuskunnalla yleiskokouksen kokouksiin”*. Verkon tuottama teksti on *”veniya on osalis to note cork an tasonvaltus onala hulescocoaks and cocok si”*. Tästä on tunnistettavissa oikeita merkkejä ja äänteitä, mutta ääkköset puuttuvat, koska ne eivät ole englannin merkistössä.

QuartzNet-neuroverkkoa siirtokoulutetttiin kahden epookin verran ja tunnistustulos paranee *”pen äja on osalistunut korkenta sen valtuskunnalla yleskokouksen kokouksii”*, mutta sanat ovat vaillinaisia. Koulutusta jatkettiin ja 50 epookin jälkeen tunnistustulos on *”venä ja on osallistunut korkeantasoon valtuuskunnalla yleiskokouksen kokouksiin”*. Tämä on jo varsin lähellä ihmisen litteroimaa tekstiä ja neuroverkko saattaa olla ylisovittunut.

Eduskunnan puheaineisto sisältää kehittäjäaineiston (eng. developer test, devtest), jossa on äänitiedostoja, joita ei ole käytetty neuroverkon opetuksessa. Näistä valittiin kaksi lyhyttä tallennetta ja saatiin seuraavat tekstit: kansanedustaja Mikko Kärnä: *”kannatan täydestä sydämestäni tätä lakialoitetta”* tunnistettiin *”kannatan täydista sydämästänikatä vakialoitetta”*. Englannin kielellä koulutettu verkko *”gone up on douest te tuta mir tenitata a yell”* tuotti ääntämyksellisesti lähellä olevan tekstin, mutta koska englannin kieltä äännetään eri tavalla kuin kirjoitetaan, niin teksti ei vastaa suomenkielistä. Kansanedustaja Veera Ruoho: *”joo kiitos”* ja QuartzNetin vastine ”jokietosseuaavaksi”. Tämä on lyhyt tallenne ja silti siitä tuotettu teksti on melko pitkä.

QuartzNet-neuroverkko saavutti edellä mainitussa eduskunnan puheaineiston suorituskyvyn tarkistamiseen tarkoitetussa ns. todennusaineistossa sanavirhesuhteen 50,90 % (18158 oikein litteroitua sanaa, 13042 korvattua, 3695 poistettua/tunnistamatta jäänyttä

ja 1025 lisättyä sanaa. Tämä on heikko tulos kohinattomalle aineistolle ottaen huomioon, että todennukseen käytettiin opetusaineiston kanssa samojen puhujien verkolle tuntemattomia puhenäytteitä. Merkkivirhesuhde on kuitenkin parempi 10,93 % (oikeita merkkejä 275552 kpl, korvattuja 8573, poistettuja/puuttuvia 17027 ja lisättyjä 7303 kpl). Tästä voitaneen päätellä, että suomen kielen sijapäätteet ja toistuvat vokaalit/konsonantit tuottavat haasteita konnektionistiselle aikajaottelu -algoritmille. Toimintaa voitaisiin kuitenkin parantaa käyttämällä erillistä kielimallia korjaamaan virheitä sanojen tunnettujen muotojen ja lauseyhteyden perusteella.

7.2.2 QuartzNet-litterointitulokset radioamatööripuheaineistolla

Seuraavaksi kokeiltiin seuraavansisältöisen radioamatöörilähetyksen litterointia: *”No se olis mahdollisesti vaikuttanut tuolla tohon noin tohon noin 20 bandiin että se ei ollut oikein se sitten aikaisemmin se meni niinku koko 20 oli niinku riittävästi vireessä mutta nyt sitten Ei mennytkään enää elikkä se meni sinne 200 tai 14200 ja siitä ylöspäin oli niinku vireessä mutta että sitten sieltä tuli nyt niinku käytännössä kaikki kvarkit tuli mukaan että se on vähän tämmönen hmm emmä ole sitä lähtenyt säätämään että kuinka pitkä sen langan pitäs olla koska se oli vähän sattumanvarainen lanka löytyi jonka mä siihen laitoin. Joojoo. Tämäpä roogeri. Mites 21 megaa? Se on muistaakseni koko alue ihan ok. Joo joo ilman tätä tätä ilman tätä kelaa se ei kyllä menis koska tuota se menee siihen parittomaan niin jos mä oikein tuossa eiku vai tuleeko siitä kolme lambdaa lambdaa? Vai laskenko”*.

Neuroverkon tuottama teksti on kaukana tavoitellusta: *”ätaa litneltiteutttlaataan taholla etaollkaden kykäielä tytre olooikeetiteitätellitemälei koko koka kytypiealieli odariittäväpyvid täopa nytyttälei mennytkäeläömenitiletatotläneltootkatotn työtäino lililunviräetittedä tolenyuikumää tänä jokaki vakituli okaanettäeovähä tämmäleem tälää nytää tämälet kulta pikkatollaa pitä olla koka oliähätämäne attoma varlänalakyomaväntilaejaelä vareomeki meterväkkovynäkatomota pitlekkkoko alueeikotäota ilpanpataakatakelpotavakeaa v iköllä varihkottatopaampäläkiepariittovaanitäa jaka oikea tota e ko patolekatetä kalmella nkolmella on ptäovain laken”*.

Edellinen tallenne ei ollut kohinaisimpia ja lopuksi kokeiltiin vielä QuartzNet-neuroverkon suoritusta niin kohinaisessa näytteessä, että kirjoittajallakin oli haasteita löytää sanoja kohinan joukosta. Tallenteen Rec_20210627_17_.wav: *”ukkonen aika menee jo desibelin alle välillä OH7UK”* neuroverkon vastine on lyhyt: *”tue u”*. Vertailun vuoksi Microsoft Office 365 -litterointipalvelu löysi seuraavat sanat: *”Ukkonen aika menee. Välillä on 7 3.”* Tämä osoittaa, että Microsoft on kouluttanut verkkoaan myös kohinaisilla äänitallenteilla.

QuartzNet-neuroverkko saavutti radioamatööripuheaineiston tunnistamisessa sanavirhesuhteen 99,75 % (28 oikein litteroitua sanaa, 1514 korvattua, 8518 poistettua/tunnistamatta jäänyttä ja kolme lisättyä sanaa). Merkkivirhesuhde oli hieman pienempi

85,75 %, oikein 10638 merkkiä, korvatut 10610 kpl, poistetut 49358 kpl ja lisätyt 577 kpl. Neuroverkkoa jatkokoulutettiin vajaan tunnin mittaisella litteroidulla radioamatööripuheaineistolla. Tämä ei kuitenkaan parantanut tilannetta vaan WER oli 99,99 % ja CER 99,41 %.

QuartzNet-neuroverkon heikkoon suoritukseen radioamatöörilähetteiden kanssa on luultavasti useita syitä, joista suurin lienee radioamatöörien nopeampi puhekielinen puhetapa verrattuna kansanedustajien hyvin artikuloituihin puheenvuoroihin, ja radioamatöörien sanasto poikkeaa merkittävästi kansanedustajien puheenvuoroissaan käyttämistä sanoista. Lisäksi radioamatöörien käyttämien lähetteiden sisältämä kohina on suurempaa suhteessa signaaliin verrattuna mikrofonilla tallennettuihin puheisiin. Saattaa olla, ettei malli kykene löytämään riittävää yleistystä aineistosta rajallisten koulutettavissa olevien parametrien vuoksi.

7.2.3 Conformer-CTC-litterointitulokset eduskunnan aineistolla

Conformer-CTC-mallista oli tarjolla ns. kirjoitusmerkki- (eng. character encoding) ja alisanakoodainmallit (eng. subword encoding). Ensin kokeiltiin kirjoitusmerkkipohjaista koodainta lisäämällä sen merkistöön ”ä”, ”ö” ja ”å” kuten edellä QuartzNet-verkon kanssa. Tämän johtaa käytännössä koodinpurkajan uudelleen kouluttamiseen. Koulutus ei kuitenkaan onnistunut vaan päättyi virheilmoituksiin.

Tämän jälkeen opiskelin lisää alisanakoodauksesta ja neuroverkkomallin määrittelyssä on tarjolla esikäsittelyyn ns. Google Sentencepiece- (joko tavuparikoodausta (eng. byte-pair-encoding, BPE) [109] tai unigram-mallia [110]) tai HuggingFace WordPiece -saneistimet (eng. tokenizer) sanojen pilkkomiseksi tunnistusta helpottaviin osiin. Näiden molempien käyttö eduskunnan aineistolle kuitenkin kaatui mahdollisesti puuttuviin käyttöoikeuksiin yliopiston laskentapalvelimella, kun skripti ei saanut tehtyä uusia hakemistoja ja lopulta muodostetut tiedostot eivät sisältäneet saneistettuja sanoja eikä siten ollut perusteita aloittaa koulutustakaan.

Palasin kirjoitusmerkkikouluttimen käyttöön, mutta vaikka sainkin opetus- ja testiaineistot ladattua, niin koodin suoritus kaatui asetustiedostossa olleeseen pre-encoder-asetukseen. Tämän asetuksen poistaminen asetustiedostosta ja asetustiedostopolkujen saaminen toimiviksi saattoi mallin toimintakelpoiseksi. Conformer-CTC Large -mallin ensimmäisen epookin koulutuksen aikana tulostui aluksi seuraava tunnistustulos:

```
öäyiröroxrmxärxkörururutäääqr öhgråuräräianröåråfrävqzärqrqvmäi-
luarwårvätröärvmzburukqriuöäwdxiyorg
```

ja sen tavoiteteksti oli:

```
huippuyksikköjen menestyminen ei ole tällaisten leikkausten
```

Kun koulutusta oli jatkettu epookin loppuun, niin tunnistustulos oli parantunut huomattavasti:

```
pankkivaltuuston näkemys on että suomen pankin vakavaraisuus oli
kertomus vuoden lopulaan riittävä kattamaan pankin tehtävien hoi-
tamisesta
```

ja tämän tavoiteteksti on lähes identtinen:

```
pankkivaltuuston näkemys on että suomen pankin vakavaraisuus oli
kertomusvuoden lopulla riittävä kattamaan pankin tehtävien hoita-
misesta johtuvat riskit
```

Näin ollen Conformer-CTC Large -mallin koulutus tuottaa nopeammin hyvän puheentunnistuskyvyn verrattuna QuartzNet-malliin. Conformer-CTC Large -malli saavutti 15 epookin koulutuksen jälkeen myös paremman sanavirhesuhteen 34,35 % (oikeat sanat 24035, vaihdot 9064, poistot 1796 lisäykset 1128 kpl) ja merkkivirhesuhteen 9,41 % (oikeat merkit 284027, vaihdot 5593, poistot 11532, lisäykset 11215) eduskunnan todennusaineistossa kuin QuartzNet-malli.

7.2.4 Conformer-CTC-litterointitulokset radioamatööripuheaineistolla

Myös Conformer-CTC-mallia käytettiin litteroimaan radioamatööripuheaineisto sekä koulutettuna pelkästään eduskunnan puheaineistolla, että koulutettuna molemmilla puheaineistoilla (eduskunnan ja radioamatööripuhekoulutusmateriaali). Nyt sanavirhesuhde ja merkkivirhesuhteet heikkenivät ollen 99,96 % (oikeat sanat 117 kappaletta, vaihdot 3394, poistot 6549 ja lisäykset 113 kpl) ja 83,67 % (oikeat merkit 13001 kpl, vaihdot 11747, poistot 45858 ja lisäykset 1462 kpl). Radioamatöörimateriaalin käyttäminen koulutuksessa eduskunnan puheaineiston lisäksi eri parantanut suorituskykyä vaan sanavirhesuhde kasvoi hieman ollen nyt 100,14 % (oikeat sanat 190, vaihdot 4557, poistot 5313 ja lisätyt 204 kpl). Merkkivirhesuhde kuitenkin hieman parantui ollen 78,01 % (oikeat merkit 18498 kappaletta, vaihdot 16684, poistot 35424, lisäykset 2975 kpl).

7.2.5 Tiivistelmä työssä saavutetuista tunnistustuloksista

Microsoftin puheentunnistin saavutti pienimmän sanavirhesuhteen 64,17 % kohinaiselle radioamatööripuheelle ja myös pienimmän merkkivirhesuhteen 34,03 %. Työssä koulutetuilla neuroverkoilla saavutettiin noin 100 % sanavirhesuhde ja monimutkaisemmalla mallilla parhaimmillaan 78,01 % merkkivirhesuhde, ks. yksityiskohdat taulukossa 20. Koska työssä koulutetut puheentunnistusmallit saavuttivat paremmat virhesuhteet koulutusaineistoa paremmin vastaavassa todennusaineistossa, on luultavaa, että suuremmalla litteroidulla radioamatööripuhekoulutusmateriaalilla olisi saavutettu matalammat virhesuhteet.

Taulukko 20 Eri puhetunnistimilla radioamatööripuhemateriaalille saavutetut sana- ja merkkivirhesuhteet.

Tunnistin	**WER**	**CER**
Google SpeechRecognition	78,30 %	78,49 %
Microsoft Office 365	64,17 %	34,03 %
QuartzNet 5x15	99,99 %	99,41 %
Conformer-CTC Large	100,14 %	78,01 %.

Koska sekä sanavirhe- että merkkivirhesuhteet jäivät verrattain korkeaksi omilla koulutusmateriaaleilla, voidaan todeta, ettei kohinaiselle radioamatööripuheelle löydy suoraan täydellisesti toimivaa puheentunnistinta. Lisäksi eri tunnistimien tuottamat merkit vaihtelevat paljon ja osalla menetelmistä ovat satunnaisia, ettei todennäköisesti saavuteta merkittävää etua yhdistämällä tuloksia nippumenetelmällä (ks. luku 3.7).

8 Yhteenveto

Opinnäytetyössä yritettiin löytää päästä-päähän-neuroverkkoalgoritmeja kohinaisen radioamatööripuhelähetysten muuntamiseksi automaattisesti tekstiksi. Työn alussa luotiin katsaus puheentunnistuksen perusteisiin sekä puheentunnistuksessa käytettyihin neuroverkkomalleihin ja niiden taustatekniikoihin. Viimeisimmät puheentunnistuksessa käytetyt päästä-päähän-menetelmät sisältävät paljon viimeaikaisissa tutkimuksissa havaittuja suorituskykyä parantavia keksintöjä. Tästä syystä työn teoriaosuuden laatimiseen kului huomattavan paljon aikaa.

Suomalaisten radioamatööripuhelähetysten tallentaminen tapahtui pääosin kesällä päivän aikana, jolloin lähetyksiä oli paljon. Tallennuksia syntyi paljon ja suhteellisen nopeasti, mutta osassa oli ongelmia tallennusvaiheessa kohinasalvan ja erillisen äänikortin äänitasojen kanssa. Lisäksi äänikortin näytteistyskaistanleveys (16 kHz) oli tarpeettoman suuri verrattuna lähetyksen kaistanleveyteen (3 kHz) ja se saattoi lisätä tallennuksiin 3–8 kHz taajuuksille ylimääräisiä signaaleja.

Työmäärä radioamatööripuhelähetysten litteroimiseksi osoittautui erittäin suureksi – yhden minuutin litteroimiseen saattoi kulua 10–15 minuuttia ja koulutusaineiston puhujien puheenvuorojen erottamisessa erillisiin tiedostoihin 15–30 minuuttia. Tätä aineistoa käytettiin kuitenkin sellaisenaan Googlen ja Microsoftin automaattisten puheesta tekstiksi palveluiden arvioimiseen. Niillä saavutettiin kohtalainen suorituskyky (sanavirhesuhteet 78,3 % ja 64,17 %) kohinaiselle radioamatööripuhelähetteelle. Microsoftin tekstitys osasi tunnistaa ja erotella myös eri puhujat tallenteista.

Työssä koulutettiin NVIDIA:n kehittämä QuartzNet 15x5 konvolutiivinen neuroverkko kansanedustajien eduskunnassa vuosina 2016–2020 pitämien tekstitettyjen puheenvuorojen avulla. Tämä neuroverkko saavutti aineiston todennusaineistossa sanavirhesuhteen 50,90 %, mutta radioamatööripuheaineistossa sana- ja merkkivirhesuhteet olivat korkeita (99,99 % ja 99,41 %), vaikka verkkoa hienosäädettiin radioamatööripuheaineistolla. Vastaavasti tunnistusta kokeiltiin kehittyneemmällä (ja monimutkaisemmalla) Conformer-CTC Large -mallilla ja tällä saavutettiin eduskunnan todennusaineistossa 34,35 % sanavirhesuhteen, mutta radioamatööripuheaineistossa sanavirhesuhde heikkeni ollen 100,14 % ja merkkivirhesuhde oli 78,01 %. Koska suuremmalle aineistolle Conformer-malli tuotti varsin hyvän sanavirhesuhteen, on mahdollista, että tuottamalla enemmän koulutusmateriaalia voitaisiin saavuttaa parempi sanavirhesuhde myös kohinaisemmalle radioamatööripuheelle.

Koska sanavirhesuhteet olivat hyvin korkeat, niin vastauksena tutkimuskysymyksiin voidaan todeta, että menetelmien tuottamien tulosten yhdistämisellä, ns. nippumenetelmä, ei voida saavuttaa suurta etua rajallisen opetusaineiston kanssa. Lisäksi havaittiin, että puheentunnistus vaatii etenkin neuroverkkomallien koulutukseen käytännöllisessä

ajassa käytännössä useita grafiikkakiihdytinkortteja. Koulutettua mallia voidaan kuitenkin ajaa resursseiltaan rajoittuneemmassa laitteessa (esim. älypuhelin).

Yleisenä havaintona todettiin, että automaattisen puheentunnistuksen tarkkuutta voitaisiin parantaa käyttämällä monipuolisempaa ja suurempaa aineistoa. Työssä tallennettujen radioamatöörien puheaineistossa esiintyi vain miehiä, joten naisten äänien tunnistamiseksi kohinaa voisi lisätä esimerkiksi eduskunnan puheaineiston tallenteisiin jälkikäsittelyllä ja näin auttaa verkkoa tunnistamaan myös harvoin aineistossa esiintyviä näytteitä. Tätä ei kuitenkaan ehditty tutkia tässä työssä.

Olisi myös hyödyllistä, jos käsin litterointia tekevän henkilön apuna olisi sovellus, joka tunnistaisi eri puhujat ja oppisi muuntamaan puheen tekstiksi litteroitujen näytteiden määrän kasvaessa yhä paremmin. Lopulta sovellus osaisi ehdottaa lähes täydellistä tekstitystä ja henkilön tehtäväksi jäisi vain hienosäätää kohdalleen. Tällaisen sovelluksen luominen esimerkiksi hyödyntäen Googlen tai Microsoftin palvelujen rajapintoja voisi olla mahdollista.

Työ oli opettavainen ja tutkittavaa varmasti riittää kohinaisen puhesignaalin litteroimiseksi tekstiksi. Haasteena suuresta tietoaineistosta tehtäviin malleihin on kuitenkin paitsi opetusmateriaalin tuottaminen, niin myös oikeasti vaikuttavien parannusten todentaminen, koska suurten neuroverkkomallien koulutukseen kuluu usein paljon aikaa.

9 Viiteluettelo

[1] A. Shashkevich, ”Stanford researcher examines earliest concepts of artificial intelligence, robots in ancient myths,” Stanford News, 28. helmikuuta 2019. [Online]. Available: https://news.stanford.edu/2019/02/28/ancient-myths-reveal-early-fantasies-artificial-life/. [Haettu 16. toukokuuta 2022].

[2] C. Kikel, ”A Brief History of Voice Recognition Technology,” The Total Voice, 14. huhtikuuta 2022. [Online]. Available: https://www.totalvoicetech.com/a-brief-history-of-voice-recognition-technology/. [Haettu 16. toukokuuta 2022].

[3] K. Moskvitch, ”The machines that learned to listen,” BBC, 15. helmikuuta 2017. [Online]. Available: https://www.bbc.com/future/article/20170214-the-machines-that-learned-to-listen. [Haettu 16. toukokuuta 2022].

[4] R. Pieraccini, Kirjoittaja, *From Audrey to Siri.* [Performance]. International Computer Science Institute, 2012.

[5] B. T. Lowerry, The HARPY Speech Recognition System - Thesis Abstract, Pittsburgh, Pennsylvania: Carnegie-Mellon University, 1976.

[6] D. Yu ja L. Deng, ”Gaussian Mixture Models,” tekijä: *Automatic Speech Recognition - A Deep Learning Approach*, Lontoo, Springer Verlag, 2015, pp. 13-21.

[7] NVIDIA Corporation, ”Automatic Speech Recognition (ASR),” 20. toukokuuta 2022. [Online]. Available: https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/asr/intro.html. [Haettu 22. toukokuuta 2022].

[8] G. E. Moore, ” Cramming more components onto integrated circuits,” *Electronics Magazine,* osa/vuosik. 38, nro 8, 1965.

[9] S. Puri, ”Training convolutional neural networks on graphics processing units”. Yhdysvallat Patentti US7747070B2, 31. elokuuta 2005.

[10] D. Yu ja D. Li, Automatic Speech Recognition, London: Springer, 2015.

[11] SRI International, ”Siri,” 2022. [Online]. Available: https://www.sri.com/hoi/siri/. [Haettu 23. toukokuuta 2022].

[12] L. Lorenzetti, ”Forget Siri, Amazon now brings you Alexa,” Fortune, 6. marraskuuta 2014. [Online]. Available: https://fortune.com/2014/11/06/forget-siri-amazon-now-brings-you-alexa/. [Haettu 23. toukokuuta 2022].

[13] A. Ng, ”Foreword,” *The Batch - Essential news for deep learners,* 28. huhtikuuta 2021.

[14] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano ja K. Lang, ”Phoneme recognition using time-delay neural networks,” *IEEE Transaction on Acoustics, Speech, and Signal Processing,* osa/vuosik. 37, nro 3, pp. 328-339, 1989.

[15] A. Graves, S. Fernández, F. Gomez ja J. Schmidhuber, ”Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks,” tekijä: *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, 2006.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser ja I. Polosukhin, ”Attention is all you need,” tekijä: *Neural Information Processing System*, Long Beach, CA, USA, 2017.

[17] A. Virtanen, Robust f0 Estimation of Telephony Speech Using Artificial Neural Networks, Tampere: Tampere University Faculty of Information Technology and Communication Sciences, 2020.

[18] T. Bäckström, O. Räsänen, A. Zewoudie, P. P. Zarazaga ja L. Koivusalo, ”Introduction to Speech Processing,” Aalto University, [Online]. Available: https://wiki.aalto.fi/display/ITSP/Speech+production+and+acoustic+properties. [Haettu 16. tammikuuta 2022].

[19] A. Iivonen, M. Horppila, M. Heikkonen ja O. Rissanen, ”Fonetiikan perussanasto,” Helsingin yliopisto Fonetiikan laitos, 2000. [Online]. Available: http://hdl.handle.net/10224/3513. [Haettu 16. tammikuuta 2022].

[20] Blausen.com staff, ”Medical gallery of Blausen Medical 2014,” 2014. [Online]. Available: https://en.wikiversity.org/wiki/WikiJournal_of_Medicine/ Medical_gallery_of_Blausen_Medical_2014. [Haettu 16. tammikuuta 2022].

[21] Mozilla Corporation, ”Mozilla Common Voice,” Mozilla Corporation, [Online]. Available: https://commonvoice.mozilla.org/fi. [Haettu 3. helmikuuta 2022].

[22] Yleisradio, ”YLE - Lahjoita puhetta,” Yleisradio, [Online]. Available: https://yle.fi/aihe/lahjoita-puhetta. [Haettu 3. helmikuuta 2022].

[23] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, D. Seetapun, A. Sriram ja Z. Zhu, ”Exploring neural transducers for end-to-end speech recognition,” tekijä: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Okinawa, Japan, 2017.

[24] D. Jurafsky ja J. H. Martin, Speech and Language Processing, 2019.

[25] S. S. Stevens, J. Volkmann ja E. B. Newman, ”A Scale for the Measurement of the Psychological Magnitude Pitch,” *The Journal of the Acoustical Society of America,* osa/vuosik. 8, pp. 185-190, 1937.

[26] S. S. Stevens ja J. Volkmann, ”The Relation of Pitch to Frequency: A Revised Scale,” *The American Journal of Psychology,* osa/vuosik. 53, nro 3, pp. 329-353, 1940.

[27] A. Zhang, ”SpeechRecognition 3.8.1,” Python Package Index (PyPI), 5. joulukuuta 2017. [Online]. Available: https://pypi.org/project/SpeechRecognition/. [Haettu 21. kesäkuuta 2022].

[28] Amazon, ”Amazon Mechanical Turk,” [Online]. Available: https://www.mturk.com/. [Haettu 5. tammikuuta 2022].

[29] ”NeurIPS Data-Centric AI Workshop,” [Online]. Available: https://datacentricai.org/. [Haettu 5. tammikuuta 2022].

[30] L. Meronen, M. Trapp ja A. Solin, ”Periodic Activation Functions Induce Stationarity,” tekijä: *Advances in Neural Information Processing Systems 34*, 2021.

[31] J. Huang, O. Kuchaiev, P. O'Neill, V. Lavrukhin, J. Li, A. Flores, G. Kucsko ja B. Ginsburg, ”Cross-Language Transfer Learning and Domain Adaptation for End-to-End Automatic Speech Recognition,” tekijä: *IEEE International Conference on Multimedia and Expo (ICME)*, Shenzhen, 2021.

[32] W. S. McCulloch ja W. Pitts, ”A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics,* pp. 115-133, Joulukuu 1943.

[33] F. Rosenblatt, ”The perceptron - a perceiving and recognizing automaton,” Cornell Aeronautical Laboratory, 1957.

[34] S. Haykin, Neural Networks: a comprehensive foundation, Upper Saddle River, New Jersey: Prentice-Hall, 1999.

[35] I. Goodfellow, Y. Bengio ja A. Courville, Deep Learning, Cambridge, Massachusetts: The MIT Press, 2016.

[36] J. Heaton, ”The Number of Hidden Layers,” Heaton Research, 1. kesäkuuta 2017. [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html. [Haettu 26. tammikuuta 2022].

[37] A. Gad, ”Beginners Ask “How Many Hidden Layers/Neurons to Use in Artificial Neural Networks?”,” Towards Data Science, 27. kesäkuuta 2018. [Online]. Available: https://towardsdatascience.com/beginners-ask-how-many-hidden-layers-neurons-to-use-in-artificial-neural-networks-51466afa0d3e. [Haettu 26. tammikuuta 2022].

[38] *Mathworks MATLAB.*

[39] H. Tuominen, ”Johdatus tekoälyn taustalla olevaan matematiikkaan,” Jyväskylän yliopisto, 2017. [Online]. Available: https://tim.jyu.fi/view/143092#WQNTKt6Q8IMk. [Haettu 2. kesäkuuta 2022].

[40] ”Learning Internal Representations by Error Propagation,” tekijä: *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, MIT Press, 1986, p. 318–362.

[41] K. Hornik, M. Stinchcombe ja H. White, ”Multilayer Feedforward Networks are Universal Approximators,” *Neural Networks,* osa/vuosik. Vol 2, pp. 359-366, 1989.

[42] P. Ramachandran, B. Zoph ja Q. V. Le, ”Searching for Activation Functions,” 16. lokakuuta 2017. [Online]. Available: https://arxiv.org/abs/1710.05941v2. [Haettu 14. elokuuta 2022].

[43] J. Bridle, ”Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition,” *Neurocomputing: Algorithms, Architectures and Applications,* osa/vuosik. 68, pp. 227-236, 1990.

[44] A. Singh, ”Digit Recognition using MNIST dataset using Softmax and ReLu Activation function,” Medium Inc, 20. elokuuta 2020. [Online]. Available: https://medium.com/da-tum/digit-recognition-using-mnist-dataset-using-softmax-and-relu-activation-function-7aacd595dc53. [Haettu 3. kesäkuuta 2022].

[45] I. Goodfellow, Y. Bengio ja A. Courville, ”Optimization for Training Deep Learning Models,” tekijä: *Deep Learning*, Cambridge, Massachusetts, MIT Press, 2016, pp. 271-325.

[46] A. Bilogur, ”Full batch, mini-batch, and online learning,” Kaggle, 29. heinäkuuta 2018. [Online]. Available: https://www.kaggle.com/code/residentmario/full-batch-mini-batch-and-online-learning/notebook. [Haettu 9. toukokuuta 2022].

[47] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy ja P. T. P. Tang, ”On large-batch training for deep learning: Generalization gap and sharp minima,” tekijä: *5th International Conference on Learning Representations*, Toulon, Ranska, 2017.

[48] J. L. Ba, J. R. Kiros ja G. E. Hinton, ”Layer Normalization,” tekijä: *NIPS 2016 Deep Learning Symposium*, 2016.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever ja R. Salakhutdinov, ”Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research,* osa/vuosik. 15, pp. 1929-1958, 2014.

[50] G. E. Hinton, A. Krizhevsky, I. Sutskever ja N. Srivastva, ”System and method for addressing overfitting in a neural network”. Yhdysvallat Patentti US2014180986A1, 30. elokuuta 2013.

[51] J. Duchi, E. Hazan ja Y. Singer, ”Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research,* osa/vuosik. 12, pp. 2121-2159, 2011.

[52] G. Hinton, ”Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude,” Coursera, 2012.

[53] D. P. Kingma ja J. L. Ba, ”Adam: A Method for Stochastic Optimization,” tekijä: *3rd International Conference for Learning Representations*, San Diago, 2015.

[54] B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, H. Nguyen, Y. Zhang ja J. M. Cohen, ”Training Deep Networks with Stochastic Gradient Normalized by Layerwise Adaptive Second Moments,” 27. toukokuuta 2019. [Online]. Available: https://arxiv.org/abs/1905.11286. [Haettu 14. elokuuta 2022].

[55] M. Schuster ja K. K. Paliwal, ”Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing,* osa/vuosik. 45, nro 11, p. 2673–2681, 1997.

[56] B. Hammer, ”On the approximation capability of recurrent neural networks,” *Neurocomputing,* osa/vuosik. 31, nro 1-4, pp. 107-123, 2000.

[57] A. Graves, ”3.2.4 Bidirectional Networks,” tekijä: *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer Berlin, Heidelberg, 2012, pp. 21-22.

[58] K. Cho, B. von Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk ja Y. Bengio, ”Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,” tekijä: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, 2014.

[59] A. Zhang, Z. C. Lipton, M. Li ja A. J. Smola, ”10.1. Gated Recurrent Units (GRU),” 21. kesäkuuta 2021. [Online]. Available: https://d2l.ai/chapter_recurrent-modern/gru.html. [Haettu 13. elokuuta 2022].

[60] S. Hochreiter ja J. Schmidhuber, ”Long Short Term Memory,” Technishe Universität München Institut für Informatik, München, 1995.

[61] S. Hocheiter ja J. Schmidhüber, ”Long Short-Term Memory,” *Neural Computation,* osa/vuosik. 9, nro 8, pp. 1735-1780, 1997.

[62] P. Simard, D. Steinkraus ja J. Platt, ”Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis,” tekijä: *Proceedings of the Seventh International Conference on Document Analysis and Recogniti*, Edinburgh, UK, 2003.

[63] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Y. Simard ja V. Vapnik, ”Comparison of

Machine Learning Algorithms for Handwritten Digit Recognition," tekijä: *International Conference on Artificial Neural Networks*, 1995.

[64] Meta AI Research, "Papers with Code," [Online]. Available: https://paperswithcode.com/methods/category/convolutional-neural-networks. [Haettu 25. heinäkuuta 2022].

[65] F. Yu ja V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," tekijä: *International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[66] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto ja H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 17. huhtikuuta 2017. [Online]. Available: https://arxiv.org/abs/1704.04861. [Haettu 11. elokuuta 2022].

[67] J. Hu, L. Shen, S. Albanie, G. Sun ja E. Wu, "Squeeze-and-Excitation Networks," tekijä: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA , 2019.

[68] P.-L. Pröve, "Squeeze-and-Excitation Networks," 17. lokakuuta 2017. [Online]. Available: https://towardsdatascience.com/squeeze-and-excitation-networks-9ef5e71eacd7. [Haettu 5. marraskuuta 2022].

[69] "ImageNet Large Scale Visual Recognition Challenge (ILSVRC)," [Online]. Available: https://image-net.org/challenges/LSVRC/. [Haettu 5. marraskuuta 2022].

[70] "PyTorch Lightning The Ultimate Pytorch Research Framework," [Online]. Available: https://www.pytorchlightning.ai/. [Haettu 13. marraskuuta 2022].

[71] P. team, "Introduction to Pytorch Lightning," 15. elokuuta 2022. [Online]. Available: https://pytorch-lightning.readthedocs.io/en/stable/notebooks/lightning_examples/mnist-hello-world.html. [Haettu 14. marraskuuta 2022].

[72] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Y. W. Zhengdong Zhang ja R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," tekijä: *Interspeech*, Shanghai, 2020.

[73] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li ja Y. Zhang, "Quartznet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions," tekijä: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020.

[74] G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko, E. Grave, V. Pratap, A. Sriram, V. Liptchinsky ja R. Collobert, "End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures," 19. marraskuuta 2019.

[Online]. Available: https://arxiv.org/abs/1911.08460. [Haettu 28. elokuuta 2022].

[75] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo ja S. Kumar, ”Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss,” tekijä: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Espanja, 2020.

[76] W. Han, Z. Zhang, Y. Zhang, J. Yu, C.-C. Chiu, J. Qin, A. Gulati, R. Pang ja Y. Wu, ”ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context,” tekijä: *Interspeech*, Shanghai, 2020.

[77] Y. Zhang, J. Qin, D. S. Park, W. Han, C.-C. Chiu, R. Pang, Q. V. Le ja Y. Wu, ”Pushing the Limits of Semi-Supervised Learning for Automatic Speech Recognition,” 20. heinäkuuta 2022. [Online]. Available: https://arxiv.org/abs/2010.10504. [Haettu 2. syyskuuta 2022].

[78] A. Hannun, ”Sequence Modeling With CTC,” 27. marraskuuta 2017. [Online]. Available: https://distill.pub/2017/ctc/. [Haettu 10. elokuuta 2022].

[79] ”Project DeepSpeech,” Mozilla Corporation, [Online]. Available: https://github.com/mozilla/DeepSpeech. [Haettu 16. marraskuuta 2022].

[80] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates ja A. Y. Ng, ”Deep Speech: Scaling up end-to-end speech recognition,” 19. joulukuuta 2014. [Online]. Available: https://arxiv.org/abs/1412.5567. [Haettu 16. marraskuuta 2022].

[81] R. Morais, ”How many parameters in DeepSpeech 0.8.2 and 0.9.0 models?,” Mozilla, 2. marraskuuta 2020. [Online]. Available: https://discourse.mozilla.org/t/how-many-parameters-in-deepspeech-0-8-2-and-0-9-0-models/69997. [Haettu 11. elokuuta 2022].

[82] G. B. Team, ”TensorFlow,” 9. marraskuuta 2015. [Online]. Available: https://www.tensorflow.org/. [Haettu 16. marraskuuta 2022].

[83] Mozilla Corporation, ” Welcome to DeepSpeech’s documentation!,” [Online]. Available: https://deepspeech.readthedocs.io/en/r0.9/. [Haettu 16. marraskuuta 2022].

[84] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguen ja R. T. Gadde, ”Jasper: An End-to-End Convolutional Neural Acoustic Model,” tekijä: *Interspeech*, Graz, Austria, 2019.

[85] O. Kuchaiev, ”ASR with NeMo tutorial,” NVIDIA, 6. elokuuta 2020. [Online]. Available: https://github.com/NVIDIA/NeMo/blob/main/tutorials/asr/ASR_with_NeMo.ipynb. [Haettu 2. elokuuta 2022].

[86] S. Majumdar, J. Balam, O. Hrinchuk, V. Lavrukhin, V. Noroozi ja B. Ginsburg, ”Citrinet: Closing the Gap between Non-Autoregressive and Autoregressive End-to-End Models for Automatic Speech Recognition,” 5. huhtikuuta 2021. [Online]. Available: https://arxiv.org/abs/2104.01721. [Haettu 18. marraskuuta 2022].

[87] S. Majumdar, ”[Question] ASR finetuning is in progress, but the train loss is no longer decreasing. #4492,” NVIDIA, 3. heinäkuuta 2022. [Online]. Available: https://github.com/NVIDIA/NeMo/discussions/4492#discussioncomment-3071529. [Haettu 12. marraskuuta 2022].

[88] L. Dong, S. Xu ja B. Xu, ”Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition,” tekijä: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018.

[89] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song ja M. Shoey, ”Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model,” 28. tammikuuta 2022. [Online]. Available: https://arxiv.org/abs/2201.11990. [Haettu 20. elokuuta 2022].

[90] J. Alammar, ”The Illustrated Transformer,” 27. kesäkuuta 2018. [Online]. Available: http://jalammar.github.io/illustrated-transformer/. [Haettu 6. marraskuuta 2022].

[91] S. K. Moore, ”Nvidia’s Next GPU Shows That Transformers Are Transforming AI,” IEEE Spectrum, 8. huhtikuuta 2022. [Online]. Available: https://spectrum.ieee.org/nvidias-next-gpu-shows-that-transformers-are-transforming-ai. [Haettu 9. elokuuta 2022].

[92] AIME GmbH, ”AIME A4000 - Multi GPU HPC Rack Server,” 2020. [Online]. Available: https://www.aime.info/shop/product/aime-a4000/?pid=A4000-4A100-Y7232-256R-1M-4U-4U. [Haettu 22. elokuuta 2022].

[93] TRAFICOM - Liikenne- ja viestintäministeriö, ”RADIOTAAJUUSMÄÄRÄYS 4 AC/2021M,” 15. joulukuuta 2021. [Online]. Available: https://www.finlex.fi/fi/viranomaiset/normi/480001/47642. [Haettu 11. kesäkuuta 2022].

[94] H. E. Heinonen, ”Luku 7. Radioaaltojen eteneminen,” tekijä: *Tiimissä hamssiksi Radioamatööritekniikan perusteita*, Helsinki, Suomen Radioamatööriliitto ry, 1997, p. 164.

[95] DealExtreme, ”7.1 External USB Sound Card USB to Jack 3.5mm Headphone Audio Adapter Micphone Sound Card,” [Online]. Available: https://www.dx.com/p/71-external-usb-sound-card-usb-to-jack-35mm-

headphone-audio-adapter-micphone-sound-card-for-mac-win-compter-android-linux-2738469.html#.Yrdf5OxBxPY. [Haettu 25. kesäkuuta 2022].

[96] P.-T. d. Boer, ”A WebSDR is a Software-Defined Radio receiver connected to the internet,” [Online]. Available: http://www.websdr.org/. [Haettu 11. kesäkuuta 2022].

[97] V. Panayotov, G. Chen, D. Povey ja S. Khudanpur, ”Librispeech: An ASR corpus based on public domain audio books,” tekijä: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, QLD, Australia, 2015.

[98] A. Mansikkaniemi, P. Smit ja M. Kurimo, ”Automatic Construction of the Finnish Parliament Speech Corpus,” tekijä: *Interspeech 2017*, Tukholma, Ruotsi, 2017.

[99] A. Virkkunen, A. Rouhe, N. Phan ja M. Kurimo, ”Finnish Parliament ASR corpus - Analysis, benchmarks and statistics,” 27. maaliskuuta 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10579-023-09650-7. [Haettu 23. huhtikuuta 2023].

[100] NVIDIA Corporation, ”QuartzNet15x5-asetukset,” [Online]. Available: https://raw.githubusercontent.com/NVIDIA/NeMo/main/examples/asr/conf/config.yaml. [Haettu 18. marraskuuta 2022].

[101] NVIDIA, ”Batching,” 10. lokakuuta 2022. [Online]. Available: https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/nlp/nemo_megatron/batching.html. [Haettu 6. marraskuuta 2022].

[102] S. Majumdar, ”num_samples is zero, but 1172 files were filtered #3483,” NVIDIA, 21. tammikuuta 2021. [Online]. Available: https://github.com/NVIDIA/NeMo/issues/3483#issuecomment-1018890632. [Haettu 12. marraskuuta 2022].

[103] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Z. E. D. Cubuk and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Interspeech*, Graz, Austria, 2019.

[104] A. van den Oord, N. Kalchbrenner ja K. Kavukcuoglu, ”Pixel Recurrent Neural Networks,” tekijä: *Proceedings of the 33 rd International Conference on Machine Learning*, New York, NY, USA, 2016.

[105] ”SLURM workload manager software,” SchedMD®, [Online]. Available: https://slurm.schedmd.com/documentation.html. [Haettu 6. marraskuuta 2022].

[106] NVIDIA , [Online]. Available: https://github.com/NVIDIA/NeMo/blob/main/examples/asr/conf/conformer/conformer_ctc_char.yaml. [Haettu 18. marraskuuta 2022].

[107] M. Burchi ja V. Vielzeuf, ”Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition,” tekijä: *IEEE Automatic Speech Recognition and Understanding Workshop*, Cartagena, Colombia, 2021.

[108] I. Loshchilov ja F. Hutter, ”Decoupled Weight Decay Regularization,” tekijä: *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[109] R. Sennrich, B. Haddow ja A. Birch, ”Neural Machine Translation of Rare Words with Subword Units,” tekijä: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berliini, Saksa, 2016.

[110] T. Kudo, "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates," in *The 56th Annual Meeting of the Association for Computational Linguistics* , Melbourne, Australia, 2018.

Liite 1: Matlab-koodiesimerkki yksinkertaisen neuroverkon laskennasta

```
%% Skripti neuroverkon laskutoimituksien havainnollistamiseksi
% Iiro Sundberg
% 16.1.2022
clear

%% Alustetaan neuroverkko ja koulutetaan se syötteillä
% Syötteet 3 kpl (sarakkeet), 4 kpl opetusaineistoa (rivit)
x = [0 0 0;
     1 0 0;
     0 1 0;
     1 1 0]';

% Odotettu tulosvektori (vaste tai päätelmät)
y = [1;
     0;
     0;
     1];

% Piilokerroksen painokertoimet
W1 = [0.1 0.2 0.9
    0.2 0.4 0.9;
    0.9 0.5 0.7];

b1 = [0;
      0;
      0];
% Syötekerroksen painokertoimet
W2 = [0.6;
      0.1;
      0.2];

b2 = [0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lasketaan piilokerroksen vaste
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z1 = W1*x + b1;
a1 = sig(z1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lasketaan vastekerroksen vaste
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z2 = W2'*a1+b2;
y_pred = sig(z2)';
disp('Verkon ennustamat arvot eri syötteille:');
for i = 1:length(y_pred)
    disp("y_pred"+num2str(i)+": "+num2str(y_pred(i)))
end

%% Virhefunktio
% Eteenpäin syöttävän algoritmin suorituksen jälkeen ratkaistaan virhefunktio
% eli paljonko saatu tulos poikkeaa tunnetusta odotetusta arvosta.
ero = y-y_pred;
virhe = sum(ero.^2);
```

```
%% Taaksepäin syöttävä algoritmi
% Tämän jälkeen lasketaan taaksepäin syöttävällä algoritmilla uudet
painokertoimet
lr = 1; % oppimisnopeuden skaalauskerroin, learning rate

dL_per_dy_pred = (y_pred-y); % [4x1]
dy_pred_per_dz2 = y_pred.*(1-y_pred); % [4x1]
dz2_per_dW2 = y_pred; % [4x1]
dz2_per_b2 = 1; % [1x1]

% Osittaisderivaatat W2 ja b2 suhteen
dL_per_dW2 = dL_per_dy_pred.*dy_pred_per_dz2.*dz2_per_dW2; %
[4x1][4x1][4x1]
dL_per_b2 = dL_per_dy_pred.*dy_pred_per_dz2.*dz2_per_b2; %
[4x1][4x1][4x1]

% Päivitetään W2- ja b2-arvot:
W2 = W2-lr*sum(dL_per_dW2);
b2 = b2-lr*sum(dL_per_b2);

% Osittaisderivaatat W1 ja b1 suhteen
dz2_per_da1 = a1.*(1-a1);
da1_per_z1 = W2;
dz1_per_W1 = x;
dz1_per_b1 = 1;

dL_per_dW1 =
dL_per_dy_pred.*dy_pred_per_dz2.*dz2_per_da1'.*da1_per_z1'.*dz1_per_W1
';
dL_per_db1 =
dL_per_dy_pred.*dy_pred_per_dz2.*dz2_per_da1'.*da1_per_z1'.*dz1_per_b1
;
% Päivitetään W1 ja b1-arvot:

W1 = W1-lr*sum(dL_per_dW1);
b2 = b2-lr*sum(dL_per_db1);


% Apufunktio
function u = sig(z)
   u = 1./(1+exp(-z));
end
```

Liite 2: Python-koodi tallenteiden tekemiseen USB-äänikortilta

```
# Iiro Sundberg 2021
import sounddevice as sd
from scipy.io.wavfile import write

fs = 16000
recSecs = 60

i = 20021 # määritetään juokseva numerointi tallenteelle
while 1:
    rec = sd.rec(int(recSecs*fs), samplerate = fs, channels = 1,
dtype='int16')
    sd.wait()
    fileName = 'Rec_'+str(i)+'.wav'
    write(fileName,fs,rec)
    i += 1
    print('Tallennettiin '+fileName)
```

Liite 3: Python-koodi tallenteiden litteroimiseksi Google Speech Recognition -kirjastolla

```
# Iiro Sundberg 2021
import numpy as np
import glob

import speech_recognition as sr
from os import path

def logger(s): # luodaan teksti litterointitiedostoon
    # {"audio_filepath": "/path/to/audio.wav", "text": "puheen litterointi", "duration": 60.000}
    f = open('audioComp.json', 'a+', encoding='utf-8')
    json_str = "\"{audio_filepath\":" +" \""+ fileName +"\""+ ", \"text\":" +"\""+ s +"\""+ ", " + "\"duration:\" " + str(60) + "}\n"
    f.write(json_str)
    f.write("\x0A")
    f.close()

r = sr.Recognizer()

# Rec_20021.wav

for x in range(20021,20037): # määritetään halutut tallenteet
  fileName = "Rec_"+str(x)+".wav"
  AUDIO_FILE = path.join(path.dirname(path.realpath(__file__)), fileName)
  with sr.AudioFile(AUDIO_FILE) as source: # luetaan 0...30 sekuntia
    audio1 = r.record(source, offset = 0, duration = 30)
  with sr.AudioFile(AUDIO_FILE) as source: # luetaan 30...60 sekuntia
    audio2 = r.record(source, offset = 30, duration = 30)

# tunnistetaan ensimmäisten 30 sekunnin puhe Google Speech Recognition avulla
  try:
    result1 = r.recognize_google(audio1, language = "fi-FI")
  except sr.UnknownValueError:
    result1 = "WHITE NOISE in first 30s?" # jos puhetta ei löytynyt
  except sr.RequestError as e:
    print("Google Speech Recognition -palvelu ei palauttanut tuloksia; {0}".format(e))

 # tunnistetaan jälkimmäisten 30 sekunnin puhe Google Speech Recognition avulla
  try:
    result2 = r.recognize_google(audio2, language = "fi-FI")
  except sr.UnknownValueError:
    result2 = "WHITE NOISE in latter 30s?"
  except sr.RequestError as e:
    print("Google Speech Recognition -palvelu ei palauttanut tuloksia; {0}".format(e))

  logger(result1 + " " + result2)
  print(fileName + " litteroitu.")
```