Tampereen yliopisto

Jesse Räihä

TUTKASEURANTA KONEOPPIMISMENETELMIN

Diplomityö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastajat: Ari Visa
Tapio Elomaa
Tammikuu 2024

TIIVISTELMÄ

Jesse Räihä: Tutkaseuranta koneoppimismenetelmin
Diplomityö
Tampereen yliopisto
Sähkötekniikan diplomi-insinöörin tutkinto-ohjelma
Tammikuu 2024

Tässä työssä tutkittiin koneoppimismenetelmien ja näistä tarkemmin neuroverkkojen soveltuvuutta tutkaseurannassa. Tavoitteena oli toteuttaa kokonainen koneoppimiseen perustuva tutkaseuranta-algoritmi yhden kohteen seurantaan. Algoritmin kouluttamiseen ja testaamiseen käytettiin valmista tutkasimulaattoria. Lisäksi tavoitteena oli tutkia kyseisille menetelmille soveltuvia tutkahavaintojen assosiointialgoritmeja. Työ tehtiin Insta Advance Oy:lle ja siinä käytettiin yrityksen toteuttamaa tutkasimulaatioympäristöä.

Työn alussa tutkittiin tutkien sekä perinteisten seuranta-algoritmien toimintaa ja toimintaperiaatteita. Lisäksi tutkittiin neuroverkkojen teoriaa ja toimintaperiaatteita. Näiden ja aikaisempien neuroverkkopohjaista monen kohteen seurantaa koskevien tutkimuksien perusteella lähdettiin muotoilemaan ratkaisua työssä käytettävää ympäristöä varten. Työssä kehitettiin menetelmät erilaisten simuloitavien reittien luomiseen. Valittiin reittien kanssa käytettävä koordinaatisto, reittien ja havaintojen keskittämiseen ja skaalaukseen käytettävät menettelyt sekä harjoitusdatasetin tasapainotukseen käytettävä menettely. Tämän jälkeen luotiin neuroverkkopohjainen seuranta-algoritmi, jonka rakenne sai vaikutteita perinteisistä seuranta-algoritmeista sekä aikaisemmissa neuroverkkopohjaista seurantaa tutkivissa julkaisuissa käytetyistä menettelyistä. Algoritmia varten toteutettiin kolme erilaista havaintojen assosiointimenetelmää. Näitä olivat vakioikkunainen assosiointi ennustusta käyttäen, muuttuvaikkunainen assosiointi ja autoenkoodaajaan perustuva assosiointi. Seuranta-algoritmin suorituskykyä testattiin ja arvioitiin näillä kaikilla.

Työn lopputuloksena oli, että yksinkertainen vakioikkunainen ennustusta käyttävä assosiointi oli suorituskyvyn ja menetelmän vähäisten muutettavien parametrien vuoksi testatuista assosiointimenetelmistä paras käytetyssä ympäristössä ja käytetyllä testausdatalla. Tuloksista havaittiin myös se, että toteutettu seuranta-algoritmi toimi kokonaisuutena kohtalaisesti työssä käytetyssä ympäristössä ja käytetyillä yksinkertaisilla reiteillä. Toteutetun seuranta-algoritmin neuroverkkomallien erillisissä evaluoinneissa havaittiin merkkejä siitä, ettei niiden suorituskyky ollut toivotulla tasolla. Algoritmia ei myöskään käytännön syistä pystytty vertaamaan perinteisempiin menetelmiin, jonka takia sen suorituskyvystä verrattuna niihin ei ole tietoa. Käytetyt menettelyt vaativat täten jatkotutkimusta ennen kuin niiden soveltuvuudesta tehtävään voidaan tehdä kunnollisia johtopäätöksiä.

Avainsanat: tutka, tutkaseuranta, koneoppiminen, neuroverkot

ABSTRACT

Jesse Räihä: Radar tracking using machine learning methods
Master's Thesis
Tampere University
Master's degree Programme in Electrical Engineering
January 2024

This thesis focused on studying machine learning methods, specifically neural networks, for radar tracking. The goal was to develop a complete radar tracking algorithm for single target tracking scenarios which utilizes machine learning methods. The method was developed and tested using an existing radar simulator. Additionally, the aim was to explore suitable plot to track association algorithms for the tracking algorithm. The thesis was conducted for Insta Advance Oy, utilizing their ready-made radar simulator.

At the beginning of the thesis, the theory and working principles of radars and traditional tracking algorithms were examined. Furthermore, the theory and principles of neural networks were explored. Drawing from these and prior studies on neural network-based multi-object tracking, a solution was formulated tailored to the radar simulator environment used in this work. Methods were developed for creating various simulated routes. Suitable coordinate system was chosen. Procedures for centering and scaling routes and observations, and techniques for balancing the training data set were developed. Subsequently, a neural network-based tracking algorithm was designed, influenced by traditional tracking algorithms and methodologies used in previous publications on neural network-based tracking. Three different radar plot to track association methods were implemented for the tracking algorithm: fixed-window nearest neighbor association using prediction, variable-window nearest neighbor association, and autoencoder based association. The performance of the tracking algorithm was tested using these methods, and their performance was evaluated based on the tests.

The outcome indicated that the simple fixed-window association with prediction method outperformed the other tested association methods in terms of performance and the limited adjustable parameters of the method in the utilized environment and with the testing data. The results also revealed that the implemented tracking algorithm functioned moderately well in the employed environment and with the simple routes used. However, evaluations of the neural network models within the implemented tracking algorithm indicated signs that their performance did not meet the desired level. Additionally, due to practical constraints, a direct comparison with more traditional methods was not feasible, thus lacking information about its performance compared to those methods. Hence, the methodologies used require further investigation before drawing conclusive insights regarding their suitability for the intended purpose.

Keywords: radar, radar tracking, machine learning, neural networks

ALKUSANAT

Yliopisto-opinnot ovat antaneet minulle paljon. Päällimmäisenä ja tärkeimpänä ne ovat opettaneet oppimisesta sekä antaneet työkaluja ymmärtää ja tehdä tutkimusta. Ne ovat myös antaneet mahdollisuuden oppia laaja-alaisesti monista kiinnostavista asioista, jotka ovat lopulta auttaneet löytämään omat mielenkiinnonkohteet. En edelleenkään pysty vastaamaan kaikkiin kysymyksiin, joita uteliaisuuteni tuottaa, mutta ainakin nyt tiedän miten vastauksia kannattaisi lähteä etsimään. Tämä työ on ollut pitkä ja hyvin opettavainen ja sen valmistuminen merkitsee myös sähkötekniikan opintojeni päätöstä.

Työ suoritettiin Insta Advance Oy:lle vuoden 2023 aikana. Haluan kiittää kaikkia Instalta, jotka mahdollistivat työn suorittamisen. Erityiskiitokset Instan ohjaajilleni Marko Latvalalle ja Petri Korpisaarelle neuvonnasta, ohjauksesta sekä tuesta työhön liittyen.

Yliopiston puolelta haluan kiittää Ari Visaa sekä Tapio Elomaata työn tarkistamisesta. Lisäksi kiitos Arille tuesta ja ohjauksesta työn suorittamisen aikana.

Tämän lisäksi haluan kiittää opiskelukavereitani tuesta, vertaistuesta ja kannustuksesta niin työn kuin opiskelujen aikana sekä mieleenpainuneista opiskeluvuosista. Jaetut oppimisen ja oivaltamisen tunteet ovat olleet korvaamattomia. Haluan myös kiittää perhettäni, ystäviäni ja tyttöystävääni, jotka jaksoivat kannustaa ja tukea koko työn ajan.

Tampereella, 22. tammikuuta 2024

Jesse Räihä

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

EKF	Laajennettu Kalman-suodin (engl. Extended Kalman filter)
GRU	engl. Gated Recurrent Unit
LSTM	engl. Long Short-Term Memory
PRF	Pulssin toistotaajuus (engl. Pulse Repatation Frequency)
PRI	Pulssin toistoaika (engl. Pulse Repetation Interval)
RCS	Tutkapoikkipinta-ala (engl. Radar Cross Section)
ReLU	engl. Rectified Linear Unit
RNN	Takaisinkytketty neuroverkko (engl. Recurrent Neural Network)
UKF	engl. Unscented Kalman Filter

1. JOHDANTO

Digitalisaatio ja tekninen kehittyminen on johtanut siihen, että 2020-luvun maailmassa dataa tuotetaan ja kerätään enemmän kuin koskaan ennen. Monia aikaisemmin pelkästään ihmisen tehtävissä olevia tehtäviä on näin ollen pystytty automatisoimaan ja siirtämään koneiden ja algoritmien vastuulle. Monet näistä käytettävistä algoritmeista ovat edelleen ihmisten ohjelmoimia sekä suunnittelemia ja näin ollen noudattavat tarkasti niille annettuja käskyjä. Koneoppimisen ja niihin liittyvien syvien neuroverkkojen kehittyminen on kuitenkin johtanut tilanteeseen, joissa osia perinteisistä algoritmeista on korvattu näillä datasta oppivilla metodeilla. Kyseisten metodien hyötynä on se, että ne pystyvät oppimaan monimutkaisia riippuvuussuhteita suoraan datasta. Näiden riippuvuussuhteiden mallintaminen perinteisin menetelmin olisi hyvin haastavaa. Viime aikoina syvät neuroverkot ovatkin suoriutuneet monista tehtävistä, kuten kuvan ja äänen tunnistuksesta, paremmin kuin perinteisemmät menetelmät. Lisäksi esimerkiksi generatiiviset mallit ovat kehittyneet viimevuosina paljon ja ovat saaneet hyvin laajaa näkyvyyttä myös valtamediassa.

Tutkaseuranta on monimutkainen ongelma, joka koostuu monista eri vaiheista. Perinteisten algoritmien ytimessä on kuitenkin yleisesti Kalman-suodin tai joku sen kehittyneimmistä versioista. Se suorittaa seurantojen tilaestimoinnin Bayesilaista päättelyä hyödyntäen. Kyseiset menetelmät tarvitsevat toimiakseen kuitenkin monia ennalta määritettyjä parametreja. Nämä parametrit liittyvät käytettävien mittauslaitteiden mittausvirheisiin sekä seurattavien kohteiden oletettuihin dynaamisiin malleihin. Sopivien parametrien valinta vaatii näin ollen sovellusaluekohteen ja käytettävien laitteiden tuntemusta, eikä se tällöinkään ole triviaali tehtävä, vaan voi vaatia monia Monte Carlo -simulaatioita. Koska koneoppimismallit pystyvät oppimaan systeemiin liittyvät tärkeimmät riippuvuudet datan avulla, niiden avulla toteutettu tutkaseuranta-järjestelmä olisi helpompi ottaa käyttöön erilaisilla tutkilla. Monimutkaisesti liikkuvat ja nopeasti liikehtivät kohteet ovat perinteisille seuranta-algoritmeille haastavia, sillä ne voivat sisältää vain rajallisen määrän erilaisten liikemallien estimointiin käytettäviä estimaattoreita. Koneoppimismalli voisi oppia dataan sisällytetyt liikemallit ja näin ollen estimoida näitä kaikkia. Tässä työssä tutkitaankin koneoppimisen hyödyntämistä tutkaseurannassa ja toteutetaan neuroverkkoihin perustuva tutkaseuranta-algoritmi. Toteutusta varten tarvittava data generoidaan valmista tutkasimulaatiota hyödyntäen.

Käydään seuraavaksi läpi työn rakenne. Luvussa 2 esitellään tutkan toiminnan perusperi-

aatteet sekä niihin liittyvät yhtälöt. Lisäksi esitellään perinteisiin tutkaseuranta-algoritmeihin liittyvät perusperiaatteet sekä niihin liittyvät rajoitteet. Luvussa 3 esitellään neuroverkkojen toimintaan liittyvää teoriaa yksinkertaisten eteenpäin kytkettyjen verkkojen ja takaisinkytkettyjen verkkojen osalta. Lisäksi käydään läpi niiden kouluttamiseen liittyvä teoria. Tämän jälkeen luvussa 4 käydään läpi työssä toteutetun ja testatun koneoppimiseen perustuvan seuranta-algoritmin rakenne, kouluttamiseen liittyvän datan generointi, mallien koulutus sekä testattavien assosiointimenetelmien toiminta. Luvussa 5 käydään läpi toteutetun seuranta-algoritmin tulokset erilaisilla assosiointimenetelmillä sekä tarkastellaan näiden merkittävyyttä. Kyseinen luku sisältää myös pohdintaa työssä käytettätyistä menettelyistä, niiden rajoitteista sekä mahdollisia kehitysehdotuksia ja mielenkiintoisia jatkotutkimusaiheita.

2. TUTKAT JA TUTKASEURANTA

2.1 Tutkan yleinen toimintaperiaate

Tutka (engl. Radar) on pohjimmiltaan radiolähettimen ja -vastaanottimen yhdistelmä. Sen englanninkielinen nimi tulee sanoista radio detection and ranging, joka kertoo paljon sen alkuperäisestä käyttötarkoituksesta. Tutkien alkuperäinen tehtävä oli siis havaita kohteita ja määrittää niiden etäisyys radiosignaaleja käyttämällä. Nykyäänkin tutkien toiminta perustuu näihin hyvin yksinkertaisiin periaatteisiin, mutta niiden toimintaa on kehitetty vastaamaan uusia käyttökohteita.

Yksinkertaisimmillaan tutkan voidaan ajatella koostuvan antennista, lähettimestä, vastaanottimesta ja signaalinkäsittelyyn liittyvästä laitteistosta. Lähettimen tehtävänä on luoda tutkassa käytettävä signaalimuoto ja vahvistaa sitä siten, että se voidaan siirtää antennille. Antenni siirtää lähettimeltä vastaanottamansa signaalin radiosignaalin etenemistielle, joka on tutkan tapauksessa yleensä ilmakehä. Antennin lähettämä radiosignaali, joka on pohjimmiltaan elektromagneettinen aalto, etenee ilmakehässä, kunnes se törmää kohteeseen, joka heijastaa elektromagneettisia aaltoja. Tällöin signaali heijastuu kohteen pinnanmuotojen mukaan eri suuntiin ja osa signaalista saattaa palata takaisin tutkalle. Tutkan antenni vastaanottaa tämän palaavan kaiun ja toimittaa sen vastaanottimelle, joka vahvistaa signaalia ja siirtää sen korkeilta radiotaajuuksilta takaisin alemmille taajuuksille, jotta se voidaan prosessoida signaalinkäsittelyyn liittyvässä laitteistossa. Vastaanotetusta signaalista selvitetään erilaisia metodeja käyttämällä, että tuliko se kohteesta ja pitäisikö siitä siten tehdä havainto, vai oliko se vain ei haluttua häiriötä.[24, s.3-5]

Oikeiden havaintojen tekeminen on monimutkainen tehtävä, koska vastaanotetut signaalit ovat voimakkuudeltaan todella heikkoja ja sekoittuvat siten helposti häiriöön tai mahdolliseen häirintään. Esimerkiksi siviili-ilmailuun käytettävän ilmavalvontatutkan lähetetyn ja vastaanotetun signaalin ero voi tyypillisillä parametreillä olla noin 190 dB [23]. Mahdollisia häiriölähteitä ovat esimerkiksi laitteiston kohina, ulkoinen kohina, muista kuin kohteista syntyvät kaiut, muista lähteistä olevat sähkömagneettiset aallot, sähkömagneettinen häiriö sekä tahallinen häirintä [24]. Nämä kaikki vaikeuttavat havaintojen tekemistä ja aiheuttavat sen, että tutkan tekemissä havainnoissa on epävarmuutta. Tutka saattaa siis raportoida havaintoja, jotka eivät ole lähtöisin kohteista tai saattaa jättää raportoimatta kohteista tulleita havaintoja, mikäli ne ovat liian heikkoja ja peittyivät häiriön alle. Lisäksi

mittauksissa on laitteiston epätarkkuudesta aiheutuvaa mittausvirhettä.

Tutkalla on mahdollista siis mitata kohteen etäisyys, sen säteisnopeus, avaruuskulma, sekä koko ja muoto. Näistä ja tutkan tarkasta sijainnista on koordinaattimuutoksien avulla mahdollista saada tietoon kohteen globaali sijainti. Etäisyysmittaus perustuu lähetetyn ja vastaanotetun signaalin aikaeroon ja sen tarkkuuteen vaikuttaa esimerkiksi tutkasignaalin kaistanleveys. Yhtälöt ja tarkemmat määrittelyt esitetään kappaleessa 2.3. Säteisnopeus on mahdollista määrittää, joko Doppler-ilmiön avulla tai laskemalla se kahden peräkkäisen havainnon etäisyys- ja kulmamuutoksista. Tutkassa käytettävän antennin keilan tulee olla suhteellisen kapea, jotta kohteen kulma tutkan suhteen on tarkasti määriteltävissä. Tämä vaatii, että tutka tekee ainakin neljä mittausta kohteen ympäriltä, jotta kohteen atsimuutti- ja korkeuskulma on määritettävissä. Kohteen reunojen skannaus mahdollistaa näin ollen myös sen koon ja muodon määrittelemisen, tutkan kulma- ja etäisyysmittauksien tarkkuuksien rajoissa.

2.2 Tutkien luokittelu

Tutkat voidaan jaotella niiden ominaisuuksien perusteella eri ryhmiin. Tämä jaottelu voidaan tehdä esimerkiksi käytettävien antennien lukumäärän tai antennikonfiguraation perusteella tai niiden käyttämien aaltomuotojen perusteella. Aaltomuodon perusteella on olemassa kaksi eri pääluokkaa: jatkuvan aallon tutkat (engl. Continuous waveform, CW) ja pulssitutkat (engl. Pulsed waveform). Käytettävien antennikonfiguraatioiden suhteen on kaksi yleisesti käytössä olevaa luokkaa: yhden aseman ja kahden aseman tutkat. Kuitenkaan useamman kuin kahden aseman käytölle ei ole teoreettisia rajoitteita ja kahden aseman tekniikat ovat yleistettävissä useammillekin tutka-asemille. Erillaisilla tutkatyypeillä on omat etunsa ja sovelluskohteensa. Seuraavaksi käydään läpi hieman niiden ominaisuuksista, toimintatavoista ja hyödyistä.

Yhden antennin tutkissa sekä lähetin että vastaanotin on kytkettynä yhteen tutka-antenniin. Tällaisissa tutkissa lähettäminen ja vastaanottaminen on yleisesti aikajakoista. Kahden antennin tutkissa sekä lähettimellä että vastaanottimella on omat antennit, jolloin niitä voidaan käyttää myös samaan aikaan. Kahden antennin käyttäminen ei kuitenkaan automaattisesti tee järjestelmästä kahden aseman tutkaa. Tämä johtuu siitä, että lähekkäin olevien antennien kulmat ja etäisyydet suhteessa havaittaviin kohteisiin on liian samankaltaiset, tuottaakseen kahden aseman tutkan etuja.[24, s.18-20]

Lähettimen lähettämän signaalin teho voi suurimmillaan olla jopa megawatteja, kun taas tutkalle saapuvat kaiut voivat olla pienimmillään jopa vain nanowatteja. Kahden aseman tutkan tarkoituksena on erottaa lähetin ja vastaanotin siten, ettei lähettimen lähettämä vahva elektromagneettinen signaali häiritse vastaanottimen toimintaa tai pahimmassa tapauksessa riko sitä.[24, s.18-19] Kahden aseman käyttäminen mahdollistaa myös monia muita etuja. Koska vastaanotin ja lähetin voivat olla hyvin erillään toisistaan, voidaan to-

teuttaa esimerkiksi pienempiä ja halvempia laitteita, jotka sisältävät vain vastaanottimen [24, s.18-20]. Koska vastaanottimen ei tällöin tarvitse lähettää itse signaalia, sitä ei voida havaita ja paikantaa tutkimalla alueella lähetettäviä signaaleja. Tutkahäivettä hyödyntävät laitteet ovat suunniteltu heijastamaan sähkömagneettisia aaltoja mahdollisimman huonosti lähettimen suuntaan. Tämä tarkoittaa, että niiden tutkapoikkipinta-ala (engl. Radar cross section, RCS) on pieni. Tutkapoikkipinta-alaa voidaan pienentää käyttämällä sähkömagneettisia aaltoja absorboivia materiaaleja sekä muotoilemalla laite siten, että signaali ei heijastu suoraan takaisin vaan siroaa moniin eri suuntiin. Koska kahden aseman tutkassa vastaanotin voi sijaita kohteen näkökulmasta eri avaruuskulmassa kuin lähetin, se voi vastaanottaa nämä siroavat heijasteet.[24, s.18-20]

Yhden aseman tutkat ovat kuitenkin yleisempiä, koska niiden rakenne ja toteuttaminen on yksinkertaisempaa. Niissä lähettimen ja vastaanottimen erottaminen tehdään yleisesti käytettävän signaalimuodon mukaan kiertoelimiä tai kytkimiä käyttämällä. [24, s.18-20]

Jatkuvan aallon tutkissa lähetin lähettää ja vastaanotin vastaanottaa signaalia jatkuvasti. Lähettimen ja vastaanottimen erottaminen on näin ollen haastavaa ja se tehdään joko käyttämällä kahden aseman konfiguraatiota tai kytkemällä antenni kiertoelimeen. Kiertoelin mahdollistaa antennille menevien lähetettävien signaalien ja antennilta tulevien vastaanotettavien signaalien erottelun. Jotta jatkuvan aallon signaaleista voidaan määrittää vastaanotetun kaiun etäisyys, tulee siihen pystyä sisällyttämään jotenkin lähetysaika. Tämä tehdään esimerkiksi käyttämällä taajuusmodulointia. Kuitenkin lähettimen ja vastaanottimen erottelu ei jatkuvan aallon tutkissa ole täydellistä ja näin ollen lähetettävä signaali aiheuttaa häiriötä vastaanottimessa, jolloin käytettävät tehot ovat yleisesti pieniä ja havainnointialueet pieniä. Tästä syystä yleisimmät jatkuvan aallon tutkien sovelluskohteet ovat nopeuden mittauksessa ja korkeuden havainnoinnissa. [24, s.20]

Pulssitutkissa lähetys ja vastaanotto tapahtuu eri aikoihin. Tutka lähettää lyhyen pulssin, joka on pituudeltaan yleensä joitakin mikrosekunteja. Tämän ajanhetken vastaanotinlaitteisto on irrotettu antennista kytkimen avulla, joka suojaa sitä vaurioitumiselta. Tämän jälkeen lähetin irrotetaan antennista ja vastaanotin aloittaa kaikujen kuuntelemisen, joka yleensä kestää muutamista mikrosekunneista joihinkin millisekunteihin. Tällaista yhtä lähetysvastaanottosykliä kutsutaan pulssin toistoajaksi PRI (engl. Pulse repetition interval). Pulssin toistoajasta voidaan laskea pulssin toistotaajuus PRF (engl. Pulse repetation frequency) $PRF = \frac{1}{PRI}$. Tässä työssä tullaan keskittymään ilmavalvontatutkiin, jotka ovat yleisesti pulssitutkia. Luvussa 2.3 käydäänkin seuraavaksi läpi pulssitutkiin liittyvät yhtälöt.

2.3 Tutkayhtälöt

Tutkan lähettämät elektromagneettiset aallot kulkevat valonnopeudella, jolloin kohteen etäisyys on mahdollista laskea yksinkertaisesti lähetyn ja vastaanotetun signaalin aikae-

rosta seuraavaa yhtälöä käyttämällä,

$$R = \frac{c\Delta t}{2} \tag{2.1}$$

jossa Δt kuvaa lähetetyn ja vastaanotetun signaalin aikaeroa [24, s.27]. Pulssitutkan tapauksessa maksimietäisyys, josta tutka voi havaita kohteen yksiselitteisesti on

$$R_{umax} = \frac{c \cdot PRI}{2} = \frac{c}{2 \cdot PRF}. \tag{2.2}$$

Tämä johtuu siitä, että jos lähetetyllä signaalilla kestää pitempään kuin pulssintoistoajan verran palata vastaanottimelle, ei voida olla enää varmoja johtuuko kaiku jo seuraavasta lähetetystä pulssista.[24, s.22]

Liikkuvasta kohteesta kimpoavan signaalin taajuus muuttuu Doppler-ilmiön mukaisesti. Näin ollen kohteen säteisnopeus tutkan suhteen on selvitettävissä, kun vertaillaan lähetetyn ja vastaanotetun signaalin taajuuksia. Doppler-taajuus voidaan laskea seuraavan yhtälön mukaisesti,

$$f_d \approx \frac{2v}{c} f \approx \frac{2v}{\lambda}, \tag{2.3}$$

jossa f on lähetetyn signaalin taajuus ja λ sen aallonpituus. Kohteen ja tutkan välinen säteisnopeus on v. Vastaanotettujen kaikujen Doppler-nopeutta voidaan käyttää tutkissa parantamaan signaali-välke-suhdetta [24, s.625]. Tämä on mahdollista, sillä tutkavälke johtuu yleisesti maastosta tai muista paikallaan olevista kohteista, jolloin ne voidaan suodattaa pois käyttämällä liikkuvan kohteen indikaattoria (engl. Moving target indicator MTI). Kyseinen menetelmä kuitenkin olettaa, että tutkalla halutaan havaita vain liikkuvia kohteita, jolloin sen käyttämisen mahdollisuus riippuu täysin sovelluskohteesta.

Jotta kohteen nopeus voidaan havaita pulssitutkalla, on sen keilan osoitettava samaa kohdetta kohdin usean pulssin ajan. Tutka vastaanottaa kyseisestä suunnasta M pulssia, jota jokaista kohden se ottaa N näytettä. Tutka lähettää pulsseja sen pulssin toistoajan PRI välein ja sen pulssin toistotaajuus on PRF . Näin ollen tutkan keila lähettää ja vastaanottaa signaaleja samasta suunnasta $M \cdot PRI$ ajan, jota kutsutaan koherentiksi prosessointiajaksi. Jos tutkan käyttämä pulssi on suorakulmapulssi, jonka pituus on τ tällöin pulssin kaistanleveys on noin $B \approx 1/\tau$. Näytteistystaajuudeksi riittää $f_s \geq B$, jossa B on signaalin kaistanleveys [24, s.625-628]. Nyquistin teoreema on yleisesti muotoiltu siten, että signaalin näytteistystaajuuden tulisi olla $f_s \geq 2B$. Tässä tulee kuitenkin huomioida se, että tällä tarkoitetaan yleisesti kantataajuudella olevan signaalin kaistanleveyttä, joka huomioi näin ollen vain positiivisen taajuusalueen. Koska tutkasignaalit ovat taajuusmoduloituja, tässä työssä kaistanleveyden oletetaan, jos ei toisin mainita, tarkoittavan signaalin molemminpuolista kaistanleveyttä.

Näytteistä voidaan muodostaa NxM matriisi. Kyseisessä matriisissa rivit kuvaavat samal-

ta etäisyydeltä olevia näytteitä eri pulsseista. Kun jokaiselle riville lasketaan erikseen Fourier-muunnos, saadaan lopputuloksena matriisi, joka kuvaa eri etäisyyksiltä vastaanotettujen signaalien Doppler-taajuuksia . Koska suurin mahdollinen yksiselitteisesti mitattava Doppler-taajuus on $f_{dua} = PRF$, jakautuvat Doppler-taajuudet välille $-PRF/2-+PRF/2$. Yleisesti suurin osa välkkeestä syntyy paikallaan olevista kohteista, joten suodattamalla nollataajuuden ympäristö saadaan merkittävä osa välkkeestä poistettua .[24, s.625-628]

Tutkan maksimikantamaan liittyvä yhtälö voidaan intuitiivisesti ymmärtää aloittamalla siitä, kuinka isotrooppisen säteilijän teho jakautuu pallopinnalle

$$P_d = \frac{P_t}{4\pi R^2 L_t}, \tag{2.4}$$

jossa P_t on lähettimen teho ja L_t on lähettimeen liittyvä siirtohäviö [14, s.93-97]. Pulssitutkan tapauksessa, kun tiedetään tutkan huippulähetysteho P_I sekä yksittäisen lähetyksen pituus τ, voidaan tutkan keskimääräinen lähetysteho laskea seuraavasti

$$P_{avg} = P_I \cdot \tau \cdot PRF, \tag{2.5}$$

jolloin yhtälössä 2.4 olevan lähetystehon P_t voidaan ajatella olevan pulssitutkan tapauksessa $P_t = P_{avg}$. Tutkissa käytetään suuntaavia antenneja, jotka tuottavat kapean korkean vahvistuksen keilan. Antennin keilaan liittyvät ominaisuudet kuvataan yleisesti 3 dB:n kulmien avulla, jotka kertovat atsimuutti ja korkeuskulmat, joilla antennin vahvistus on 3 dB pienempi kuin sen maksimivahvistus. Antennivahvistus kertoo siis pääkeilan vahvistuksen isotrooppisen säteilijän suhteen ja sitä merkitään kirjaimella G. Koska antennivahvistus on vain suhdeluku, voidaan tutkan pääkeilan suuntaan kohdistama teho laskea kertomalla yhtälöä 2.4 antennivahvistuksella G,

$$P_d = \frac{P_t G}{4\pi R^2 L_t} \tag{2.6}$$

Elektromagneettisen aallon törmätessä kohteeseen osa energiasta absorboituu rakenteisiin ja loput siroaa moniin eri suuntiin. Vain pieni osa saapuneesta energiasta kimpoaa takaisin tutkan suuntaan. Tälle kohteen muodoista, materiaaleista, koosta ja orientaatiosta riippuvalle ominaisuudelle on määritelty termi tutkapoikkipinta-ala. Tutkapoikkipinta-ala kertoo siis kohteelta tutkan suuntaan kimpoavan ja kohteelle tutkalta tulleen tehon suhteen. Sitä merkitään merkillä σ ja se voidaan laskea siis seuraavalla yhtälöllä,

$$\sigma = \frac{P_r}{P_d} \tag{2.7}$$

jossa P_r on kohteelta tutkan suuntaan kimpoava teho ja P_d kohteeseen osunut teho. [14, s.93-97] Elektromagneettinen aalto vaimenee kulkiessaan ilmakehässä ilmakehän

vaimennuskertoimen mukaisesti. Koska kaikki vaimennukset vaikuttavat tulokseen vain kertoimena, voidaan niitä merkitä tästä lähtien yhtenä kertoimena $L = L_t L_{atm} L_r$, jossa L_{atm} on ilmakehän vaimennus molempiin suuntiin ja L_r on vastaanottimeen liittyvä siirtohäviö. Antennin sieppauspinta A_e, kuvaa kuinka hyvin antenni vastaanottaa tietyn taajuista signaalia. Sen arvo voidaan laskea seuraavaa yhtälöä käyttämällä

$$A_e = \frac{G\lambda^2}{4\pi}. \tag{2.8}$$

Näin ollen tutkan vastaanottama teho voidaan kirjoittaa seuraavassa muodossa

$$P_r = \frac{P_t G\sigma}{4\pi R^2} \frac{G\lambda^2}{4\pi} \frac{1}{L} = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R^4 L}. \tag{2.9}$$

Tutkan vastaanottamaan signaaliin summautuu monista eri lähteistä peräisin olevaa häiriötä, jota voidaan kuvata valkoisena kohinana, joka on rajoittunut tutkan käyttämälle taajuusalueelle. Vastaanottimelle saapuvan kohinan määrä on tällöin

$$N_i = kT_s B \tag{2.10}$$

jossa K on Boltzmannin vakio, T_s on järjestelmän kohinalämpötila ja B tutkan kaistanleveys. Tutkan vastaanottimen signaali-kohinasuhteeksi saadaan tällöin

$$SNR = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R^4 k T_s B L}. \tag{2.11}$$

Kyseistä yhtälöä 2.11 kutsutaan yleisesti tutkayhtälöksi ja se kuvaa tietyillä suorituskykyparametreillä olevan tutkan herkkyyttä eri etäisyyksillä oleville tai eri tutkapoikkipinta-alan omaaville kohteille.[14, s.93-97]. Tutkayhtälöstä voidaan myös helposti laskea tutkan maksimikantama seuraavaa yhtälöä hyödyntämällä,

$$R_{max} = \sqrt[4]{\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 k T_s B L} \frac{1}{SNR_{min}}} \tag{2.12}$$

mikäli tiedetään pienin havaintoon riittävä teho SNR_{min} [14, s.93-97].

2.4 Kohteen seuranta

Seurannalla tarkoitetaan samalta tosimaailman kohteelta saapuvien havaintojen yhdistämistä yhtenäiseksi kokonaisuudeksi, jonka tarkoituksena on kuvata sen liikettä eri ajanhetkillä. Tutkan tarkoituksena on tehdä havaintoja mahdollisesti hyvinkin etäällä olevista elektromagneettisia aaltoja heijastavista kohteista. Näin ollen tutkan vastaanottamat signaalit ovat yleisesti heikkoja, jonka takia myös kohina saattaa aiheuttaa ylimääräisiä havaintoja. Lisäksi kohteesta palaavaan kaikuun on summautunut kohinaa, joka aiheuttaa

epätarkkuutta mittauksissa. On myös hyvin mahdollista, että kohde on hetkellisesti tutkalta näkymättömissä esimerkiksi vuorien tai muun maastonmuodon takia, eikä siitä tällöin saada havaintoja. Kohteen sijainnista on kuitenkin tärkeää olla tarkka arvio jokaisella ajanhetkellä, koska sillä on suuri merkitys muun muuassa lentoturvallisuudelle. Seurannan on siis tarkoitus tuottaa tarkempaa ja helpommin ymmärrettävää informaatiota kohteen liikkeistä. Lisäksi sen on tarkoitus myös eritellä eri kohteista saatavat havainnot erillisiksi seurannoiksi.

Seurantajärjestelmät voidaan yleisesti jakaa kahteen tyyppiin, suljettuihin ja avoimiin seurantajärjestelmiin. Suljettuja seurantajärjestelmiä käytetään lähinnä seurantatutkissa tai monitoimitutkien seurantaominaisuudessa. Tämä johtuu siitä, että niissä seurantajärjestelmän sisäisiä parametrejä käytetään säätämään tutkan muita parametrejä. Tämä tarkoittaa esimerkiksi sitä, että antennia käännetään havainnon jälkeen kohti sen seuraavaa oletettua suuntaa. Avoimia seurantajärjestelmiä käytetään ”trak-while-scan”-tekniikkaa hyödyntävissä tutkissa. Tällöin tutka skannaa tiettyä aluetta jollain nopeudella ja muodostaa alueella olevista kohteista havaintoja. Havainnot syötetään avoimelle seurantajärjestelmälle, joka luo ja ylläpitää havainnoista syntyviä seurantoja.[4, s.1-3]. Tässä työssä tullaan käsittelemään seurantoja ainoastaan tutkalta saatavien havaintojen kautta, ilman mahdollisuutta vaikuttaa sen toimintaan. Tästä syystä tulemme tarkastelemaan ainoastaan avoimen seurantajärjestelmän toimintaa tarkemmin. Mikäli ei toisin mainita, kaikki työssä olevat viittaukset seurantajärjestelmään viittaa juuri tällaiseen avoimeen seurantajärjestelmään.

Seurannan tekeminen koostuu monesta eri askeleesta. Kun tutkalta saadaan uusi havainto täytyy ensin päättää, kuuluuko havainto johonkin jo olemassa olevaan seurantaan, onko se uusi seurattava kohde vai häiriötä. Tässä vaiheessa siis tehdään havaintojen assosiointi erillisille seurannoille ja sitä kutsutaankin havaintojen assosiointivaiheeksi. Samalla tulee huolehtia jo olemassa olevista seurannoista, onko niihin tullut päivityksiä lähiaikoina, vai onko seurattava kohde kadonnut tutkan havaintoalueelta ja tulisiko se näin ollen poistaa seurattavien kohteiden listalta. Tätä kutsutaan seurantojen hallinnoiniksi. Havainnot, jotka liittyvät jo olemassa oleviin seurantoihin, tulee myös liittää niihin, jotta arvio kohteen tilasta päivittyy havainnosta saatavalla informaatiolla. Tätä vaihetta kutsutaan seurannan tilaestimaatin päivittämiseksi tai suodattamiseksi. Nämä kaikki kohdat ovat oleellisia käytännön seurantajärjestelmän luonnissa, ja niistä tullaan kertomaan lisää seuraavissa kappaleissa.

2.4.1 Seurantasuotimet

Käydään ensin läpi seurantaan liittyvä kohteen tilan estimointi. Koska havaintojen assosiointi käydään läpi myöhemmässä luvussa, voidaan nyt olettaa, että kaikki havainnot saapuvat yhdestä tutkittavasta kohteesta ja tällöin seurannassa on todellisuudessa kyse

vain kohteen tilan estimoinnista. Seurantasuotimien tarkoituksena on vähentää havaintojen häiriöisyydestä aiheutuvaa virhettä sen tilaestimaatista. Näin ollen on luonnollista puhua havaintojen suodattamisesta, sillä kyseessä on pohjimmiltaan samasta ilmiöstä kuin yksiulotteisen signaalin tai aikasarjan suodattamisessakin. Seurantasuotimissa estimoitavat tilat sekä havainnot ovat kuitenkin eri avaruuksista, joka erottaa ne tavallisista signaalisuotimista. Tämä tarkoittaa, että havainnot eivät suoraan sisällä kaikkia estimoitavia parametrejä, vaan niiden välillä on tehtävä muunnoksia.

Seurantaongelmaa on hyvin luonnollista lähestyä Baysilaisen päättelyn avulla, jonka takia tässä työssä esitellään perinteisistä ratkaisuista vain siihen perustuvia menetelmiä. Bayesialaisen päättelyn perusperiaatteet seurannan tapauksessa on asettaa seurattavan kohteen tilalle priori-todennäköisyysjakauma ja kuvata kuinka kohteen tila voi muuttua ajan myötä. Mittauksista saatava informaatio muutetaan kohteen tila-avaruudessa olevaksi havainnon uskottavuudeksi (engl. Likelyhood). Tämän jälkeen kohteen tilaan liittyvä priori-jakauma ja mittauksesta saatava havainnon uskottavuus yhdistetään Bayesin teoreeman avulla kohteen tilan posteriotodennäköisyydeksi, josta tilaestimaatti ja sen epävarmuudet ovat laskettavissa.[23, s.2] Luvussa esiteltävä Kalman-suodin onkin optimaalisen Bayesialaisen-suotimen approksimaatio tilanteissa, jossa tarkasteltava systeemi on lineaarinen ja siihen liittyvät virheet ovat normaalijakautuneita sekä toisistaan riippumattomia. Tarkoituksena on siis estimoida tilastollisia menetelmiä hyödyntäen kohteen tilasta sellaisia parametrejä, joita ei voida suoraan havainnoida. Seuraavaksi käydään läpi, kuinka tällaista systeemiä kuvataan yhtälömuodossa, sekä sen estimointiin liittyvät Bayesilaisen menetelmän yhtälöt.

Todellisuudessa kohteiden toiminta täyttää stokastisen epälineaarisen systeemin piirteet, joten sille voidaan muodostaa stokastinen tilayhtälö. Myös mittausprosessi täyttää nämä piirteet, joten sitä voidaan kuvata stokastisena mittausprosessina. Tilayhtälö voidaan kirjoittaa siis geneerisessä muodossaan seuraavasti

$$x_n = f_n(x_{n-1}, w_{n-1}). \tag{2.13}$$

Stokastinen mittausyhtälö voidaan kirjoittaa geneerisessä muodossaan seuraavasti

$$z_n = h_n(x_n, v_n). \tag{2.14}$$

Kyseiset yhtälöt evaluoituvat ajanhetkelle t_n. x_n kuvaa kohteen tilaa, joka ei yleisesti ole havainnoitavissa ja z_n kuvaa havaintovektoria. w_n on dynaaminen häiriövektori, jota kutsutaan myös prosessihäiriöksi ja v_n on havaintoon liittyvä mittausvirhevektori. Sekä f_n että h_n ovat deterministisiä funktioita, jotka kuvaavat kohteen edellisen tilan vaikutusta nykyiseen tilaan sekä kohteen tilan vaikutusta saatavaan havaintovektoriin. Estimoidessa kohteen tilaa Bayesialaista päättelyä käyttämällä tarkoituksena on siis laskea posteriotodennäköisyys $p(x_n \mid z_{1:n})$, jossa $z_{1:n} \triangleq \begin{bmatrix} z_1 & z_2 & z_3 & \dots & z_n \end{bmatrix}$.[6, s.22-23]

Tutkitaan seuraavaksi kohteen mallintamista diskreettinä lineaarisena dynaamisena systeeminä. Kohteen tilaa voidaan merkitä vektorilla $x(k)$, jossa k on kyseinen ajanhetki. Vektori $x(k)$ sisältää n_x elementtiä, jotka kuvaavat kohteen tilaa. [1] Kyseinen vektori voisi siis sisältää esimerkiksi kohteen koordinaatit, nopeuden ja kulmannopeuden. Lisäksi tarvitaan tiedossa oleva ohjausvektori $u(k)$, joka sisältää n_u elementtiä, jotka kuvaavat kohteen tilaan vaikuttavia ohjauskomentoja. Yhtälöön tarvitaan myös normaalijakautuneesta kohinasta koostuva n_x elementtiä sisältävä vektori, joka mallintaa arvaamattomia häiriöitä, jotka vaikuttavat kohteen tilaan. Näin ollen kohteen tila ajanhetkellä $k+1$, saadaan seuraavalla yhtälöllä,

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k). \tag{2.15}$$

$F(k)$ on $n_x n_x$ matriisi, joka kuvaa kohteen dynaamista mallia ja näin ollen kuinka kohteen edellinen tila vaikuttaa seuraavaan tilaan.[1, s.200] Tämä tapahtuu esimerkiksi kohteen nopeuden kautta. Matriisi $F(k)$ voi muuttua ajan edetessä, mutta seurantasuotimissa sitä pidetään yleisesti vakiona ja merkitään pelkästään kirjaimella F. $G(k)$ on $n_x n_u$ matriisi, joka kuvaa kuinka kohteen ohjauskomennot vaikuttavat sen tilaan. Tutkaseurannan tapauksessa ohjauskomennot eivät kuitenkaan ole tiedossa, jolloin matriisi $G(k)$ ja vektori $u(k)$ voidaan jättää merkitsemättä ja näin ollen sisällyttää sen vaikutus satunnaisvektoriin. Näin ollen kohteen tilalle käytetään seuraavaa yhtälöä

$$x(k+1) = Fx(k) + v(k). \tag{2.16}$$

Yhtälöön liittyvää satunnaisvektoria $v(k)$ kutsutaan myös prosessihäiriöksi ja sen kovarianssiksi saadaan matriisi $Q(k) = \mathbb{E}[v(k)v(k)']$ Kohteesta saatavaa havaintoa merkitään vektorilla $z(k)$, käyttämällä kohteen tilaa $x(k)$ ja lisäämällä siihen normaalijakautunutta kohinaa mittausvirheeksi saadaan sille seuraava yhtälö,

$$z(k) = H(k)x(k) + w(k). \tag{2.17}$$

$H(k)$ on $n_z n_x$ matriisi, joka muuttaa kohteen tilan havainnoksi.[1, s.200] Yleisesti tutkaseurannan tapauksessa tämäkin oletetaan ajasta riippumattomaksi, jolloin sitä merkitään kirjaimella H. Lineaarisessa tapauksessa kyseinen mittausmatriisi voi esimerkiksi poistaa havainnosta arvoja, joita kohteen tila sisältää. Tämä johtuu siitä tosiseikasta, että etäällä oleva mittausväline ei yleisesti pysty mittaamaan kaikkia kohteeseen liittyviä parametreja. Tutkat mittaavat kohteista yleensä etäisyyttä, suuntaa ja säteisnopeutta. Tutkalta saatavat havainnot täytyy siis esiprosessoida, mikäli halutaan säilyttää mittausmatriisin lineaarisuus. Mittausvirhevektori $w(k)$ sisältää n_z elementtiä, jotka ovat lähtöisin normaalijakautunutta kohinaa sisältävästä jakaumasta. Mittausvirhevektorille voidaan laskea kovarianssi $R(k) = \mathbb{E}[w(k)w(k)']$. Mikäli matriisit F,H,Q ja R ovat tiedossa kaikille ajanhetkille ja täyttävät aikaisemmin esitetyt vaatimukset, käytettävä alkuarvo täyttää edellä mainitut

ominaisuudet, sekä kaikki satunnaisvektorit ovat toisistaan riippumattomia on kyseessä Gauss-Markov prosessi. Näin ollen ehdollinen tilaestimaatti voidaan laskea

$$\hat{x}(j \mid k) = \mathbb{E}[x(j) \mid Z^k]. \tag{2.18}$$

Kyseinen estimaatti tuottaa j:n ja k:n arvoista riippuen, joko estimaatin kohteen tilasta, suodatetun kohteen tilan tai ennusteen tulevasta kohteen tilasta.[1, s.201] Merkinnällä Z^k tarkoitetaan kaikkia estimointiin liittyviä mittauksia ajanhetkeen k saakka $Z^k = z(i), i < k$ Tilaestimaatille voidaan laskea myös kovarianssi käyttämällä seuraavaa yhtälöä

$$P(j \mid k) = \mathbb{E}[[x(j) - \hat{x}(j \mid k)][x(j) - \hat{x}(j \mid k)]' \mid Z^k] \tag{2.19}$$

Kalman-suodin on rekursiivinen tilaestimointialgoritmi. Tilanteessa, jossa kaikki aikaisemmin esitetyt vaatimukset systeemin lineaarisuudesta sekä häiriöiden normaalijakautuneisuudesta toteutuu, se on optimaalinen pienimmän neliösumman estimaattori. Mikäli satunnaismuuttujat eivät ole todellisuudessa normaalijakautuneita, on Kalman-suodin silti paras lineaarinen estimaattori.[1, s.207] Algoritmi koostuu kahdesta osasta: tilan ennustamisesta sekä tilan päivityksestä. Estimointia varten tarvitaan myös alkuarvot tilaestimaatille $\hat{x}(0|0)$ ja sen kovarianssille $P(0|0)$. Aloitetaan tilan ennustukseen ja suotimen sisäisten parametrien laskemisella. Ensimmäiseksi lasketaan tilalle ennuste ajanhetkelle k käyttämällä seuraavaa yhtälöä

$$\hat{x}(k|k-1) = F(k)\hat{x}(k-1|k-1) \tag{2.20}$$

Tämän jälkeen ennustetaan estimaatin kovarianssi

$$P(k|k-1) = F(k)P(k-1|k-1)F(k)' + Q(k-1) \tag{2.21}$$

Uuden havainnon tuottama informaatio huomioidaan laskemalla sen ja ennustetun tilan välinen erotus seuraavasti

$$\hat{y} = z(k) - H(k)\hat{x}(k|k-1). \tag{2.22}$$

Erotuksen kovarianssi saadaan seuraavasta yhtälöstä

$$S(k) = H(k)P(k|k-1)H(k)' + R(k) \tag{2.23}$$

Suotimen vahvistus $K(k)$, jota kutsutaan myös Kalmanin vahvistukseksi (engl. Kalman gain) lasketaan seuraavasti

$$K(k) = P(k|k-1)H(k)'S(k)^{-1} \tag{2.24}$$

Tämän jälkeen suoritetaan tilaestimaatin sekä siihen liittyvän kovarianssimatriisin päivitys. Tilaestimaatin päivitys tapahtuu seuraavaa yhtälöä hyödyntäen

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)\hat{y}(k) \tag{2.25}$$

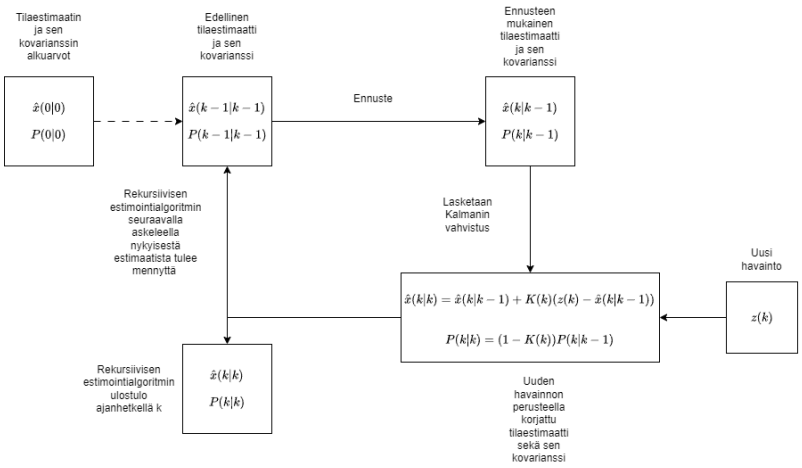
Viimeisenä päivitetään tilaestimaatin kovarianssi

$$P(k|k) = (I - K(k)H(k))P(k|k-1) \tag{2.26}$$

Yhtälöt 2.20,2.21,2.22,2.23,2.24,2.25,2.26 muodostavat rekursiivisen Kalman-suotimen yhden prosessointikierroksen.[1, s.208] Suotimen vahvistukseen liittyvästä yhtälöstä 2.24 voidaan tehdä seuraavat havainnot. Se on suoraan verrannollinen tilaestimaatin ennusteen epävarmuuteen sen kovarianssin 2.21 kautta. Lisäksi se on kääntäen verrannollinen mittauksen ja ennustetun tilan erotuksen kovarianssille 2.23. [1, s.207] Näin ollen sen voidaan ajatella kuvaavan tilaennusteen estimointivirheen ja mittausvirheen välistä suhdetta. Tällöin suotimen vahvistus on suuri, jos tilaestimaatin ennuste on epätarkka suhteessa mittauksiin ja pieni mikäli tilaestimaatin ennuste on tarkka suhteessa mittauksiin. Tämä suhde voidaan havaita myös yhtälöstä 2.25.

Normaalijakautuneiden prosessihäiriöiden ja mittausvirheiden oletukset ovat monissa käytännön seurantasovelluksissa riittäviä, varsinkin tilanteissa, jossa algoritmin tulee olla laskennallisesti tehokas. Kuitenkaan lineaaristen tilamuutoksien ja mittausyhtälöiden käyttäminen ei tuota epälineaarisissa tapauksissa tarkkoja tuloksia. Tämän takia Kalman suotimesta onkin luotu versioita, joiden tarkoituksena on toimia myös epälineaarisissa tapauksissa.[6, s.31] Tunnettuja esimerkkejä epälineaarisuuden huomioivista Kalman-suotimen varianteista ovat Extended Kalman filter (EKF) ja Unscented Kalman filter (UKF), jotka toimivat linearisoimalla tilaestimointiongelman. Perusmuotoinen Kalman-suodin on tarkoitettu lineaarisesti muuttuvien tilojen estimointiin, käyttämällä lineaarista tilanmuutosmatriisia F. Tilanmuutosmatriiseja on kehitetty monia, mutta yksinkertaisimpia niistä ovat esimerkiksi vakionopeuden- ja vakiokiihtyvyyden-tilamatriisit [24, s.730]. Mallin valinta riippuu kohteen maksimikiihtyvyydestä, mittauksien tarkkuudesta sekä mittaustaajuudesta. Parhaan mallin valintaan ei ole kuitenkaan olemassa algoritmeja, minkä takia mallin valintaan tarvitaan Monte Carlo -simulaatioita sekä tietoa seurattavien kohteiden ominaisuuksista. Suotimiin lisätyn prosessihäiriön tarkoituksena on huolehtia mahdollisista kohteen suorittamista liikkeistä, joita valittu liikemalli ei huomioi.[24, s.730]

Kaikki seurattavat kohteet eivät kuitenkaan noudata yksittäistä liikemallia, vaan voivat liikehtiä ja vaihtaa liikemalleja yllättävästikin. Yksinkertainen Kalman-suodin, joka sisältää vain yksittäisen liikemallin ei pysty mukautumaan näihin muutoksiin. Tämän ongelman ratkaisemiseen on kehitetty monia eri menetelmiä, joista yksi on monen mallin estimaattorit. Näissä oletetaan, että kohteen dynaamiset muutokset ovat aina estimoitavissa käyttäen rajattua määrää ennalta valittuja malleja. Kohteen liikehtiessä sen dynaaminen malli



Kuva 2.1. *Yksinkertaistettu kuvaus Kalman-suotimen toiminnasta*

siis hyppelehtii näiden mallien mukaisesti. Mikäli kohteen erilaiset liikemallit pystytään arvioimaan ennalta ja esittämään käyttämällä pientä määrää erillisiä malleja, monen mallin estimaattorit toimivat hyvin verrattuna muihin liikehtivän kohteen tilaestimointialgoritmeihin verrattuna. Kyseisten algoritmien heikkouksia ovat kuitenkin niiden ennalta määritetty rakenne ja, se ettei kohteiden seurantaa tarvittavat erilaiset dynaamiset mallit todellisuudessa ole yleensä tiedossa. Lisäksi useiden mallien käyttäminen kasvattaa algoritmin laskennallista kuormaa.[6, s.62-63] Näin ollen näissäkin sopivien tilamuutosmatriisien ja prosessihäiriöiden valinta perustuu Monte Carlo -simulaatioihin sekä sovellusaluekohtaiseen ennakkotietoon.

Tässä luvussa esitetyn rekursiivisen estimoinnin periaatteet on kuvattu yksinkertaistetusti kuvassa 2.1. Sen voidaan huomata sisältävän kaksi pääkohtaa: tilan ennustamisen sekä kyseisen ennusteen korjaus uutta havaintoa hyödyntäen. Luvussa 4.3 esitellään neuroverkkoihin perustuva ratkaisu, joka mukailee näitä periaatteita. Ratkaisu sisältää kaksi neuroverkkoa. Ensimmäistä kutsutaan ennustusmalliksi, jonka tarkoitus on suorittaa kuvassa 2.1 esitetty tilan ennustaminen. Toinen näistä on suodatusmalli, joka päivittää seurannan tilaa saapuvaa havaintoa hyödyntäen. Ennustusmallin tuottamaa ennustetta päivitetään kuvan 2.1 periaatteita vastaavalla tavalla, kun molemmat neuroverkot eivät ole liian lyhyen seurannan takia käytössä. Muulloin suodatusmalli päivittää tilaa itsenäisesti, pelkästään seurantaan liittyviä havaintoja hyödyntäen.

2.4.2 Havaintojen assosiointi

Tutkat tekevät havaintoja mittaamalla vastaanotettujen kaikujen vahvuuksia ja raportoimalla niistä, jotka ylittävät tietyt raja-arvot. Havaintojen tekeminen mittauksista ja siihen liittyvät tekniikat sekä algoritmit ovat monimutkaisia, eivätkä tämän työn kannalta oleellisia, sillä työssä keskitytään seurantojen tekemiseen tutkan tuottamien havaintojen avulla. Näin ollen havaintojen tekemiseen liittyvä teoria ohitetaan tässä työssä. Kuitenkin on hyvä

tietää, että tutkalta saatavat havainnot eivät ole täydellisiä ja sisältävät näin ollen mittausvirheitä. Tutkien tuottamien havaintojen todellisesta lähteestä ei myöskään ole varmuutta ja ne sisältävät harhahavaintoja (engl. False alarm), jotka voivat johtua häiriöstä, häirinnästä tai esimerkiksi tutkavälkkeestä [24, s.760-761] Havaintojen assosiointi sisältää yleisesti kaksi vaihetta. Havaintojen hyväksynnän ja ikkunoinnin sekä seurantojen päivittämisen näillä hyväksytyillä havainnoilla.[24, s.761] Työssä keskitytään yhden kohteen ja tutkan tapaukseen, jonka takia usean kohteen seurantaan liittyviä assosiointi-algoritmeja ei esitellä.

Ensimmäinen askel havaintojen hyväksynnässä ja ikkunoinnissa on yleisesti karkea ikkunointi, joka suoritetaan käyttämällä neliön muotoista ikkunaa seurantojen ympärillä. Karkean ikkunoinnin läpäisseet mittaukset otetaan mukaan tarkempaan ellipsoidilliseen ikkunointiin. Ellipsoidillinen ikkunointi suoritetaan laskemalla Mahalanobiksen etäisyys seuraavaa yhtälöä käyttämällä

$$d^i(k) = (\hat{y}^i(k))'(S^i(k))^{-1}\hat{y}^i(k), \tag{2.27}$$

jossa $\hat{y}$ on havainnon i ja seurantaan liittyvän ennustetun tilan välinen erotus, joka voidaan laskea käyttämällä yhtälöä 2.22. Yhtälössä esiintyvä toinen termi $S^i(k)$ on vastaava kovarianssi, joka lasketaan käyttämällä yhtälöä 2.23. Laskettaessa mittauksen i ja valitun seurannan välille sen yhtälö on

$$S^i(k) = H(k)P(k|k-1)H(k)' + R^i(k) \tag{2.28}$$

Seurantaan kuuluvien mittauksien Mahalanobiksen etäisyydet ovat khiin neliö-jakautuneita, jossa vapausasteiden määrä on n_z. Muuttuja n_z kuvaa havainnon sisältämien arvojen lukumäärää. Tämän jälkeen valitaan todennäköisyys P_G, jolla seurantaan liittyvien havaintojen tulisi läpäistä ellipsoidillinen ikkunointi. Arvo P_G voi olla esimerkiksi 0.995. Tällöin todennäköisyydellä $1 - P_G$, seurantaan liittyvä havainto ei läpäise ikkunointia. Tämän jälkeen lasketaan khiin neliö-jakauman häntätodennäköisyyttä $1 - P_G$ vapausasteella n_z vastaava arvo ja käytetään sitä ikkunoinnin kynnysarvona d_{th}.[24, s.763-764] Näin ollen kynnysarvo d_{th} vapausasteella 1 ja todennäköisyydellä $P_G = 0.955$ on 7,88. Kaikki havainnot, joiden $d^i(k) < d_{th}$ läpäisevät ikkunoinnin ja otetaan huomioon assosioitaessa seuraavaa havaintoa seurantaan [24, s.764].

Lähimmän naapurin assosiointi on yksinkertainen menetelmä, joka assosioi havainnon todennäköisimpään seurantaan. Lähimmän naapurin menetelmää käyttävän seurantasuotimen tilaestimaatin kovarianssi on kuitenkin optimistinen, sillä se ei sisällytä mahdollisista vääristä assosioinneista aiheutuvaa virhettä tilaestimaattiin. Lähimmän naapurin assosiointi toimii päivittämällä seurantaa tilastollisesti lähimmällä ikkunoinnin läpäisseellä havainnolla $\mathrm{argmin}_{1 \leq i \leq m_k} d_k^i$, jossa m_k kuvaa ikkunoinnin läpäisseitä havaintoja. Tämän jälkeen seurannan päivitys suoritetaan käyttämällä tätä yksittäistä havaintoa. Käytettä-

västä tila-avaruudesta sekä seurantasuotimesta riippuen lähimmällä naapurilla voidaan tarkoittaa myös euklidisesti lähintä havaintoa. Tämä aspekti on oleellinen lähinnä niiden seurantasuotimien osalta, jotka eivät paljasta niiden tilastollisia ominaisuuksia ulospäin. Neuroverkkoihin perustuvat ratkaisut ovat yleisesti tällaisia.

Todennäköisyyspohjainen data assosiaatio (engl.Probalistic data assosiation, PDAF) käyttää kaikkia ikkunoinnin läpäisseitä havaintoja seurannan päivittämiseen. Sen tilaestimaatin kovarianssi on näin ollen hyvin vakaa ja se sisällyttää myös havaintojen assosiointiin liittyvän epävarmuuden estimaattinsa. Tämä tarkoittaa siis, että sen luomat seurannat eivät ole yksittäisestä kohteesta syntyviä todellisia seurantoja vain enemmänkin hypoteettisia seurantoja. Niissä seurannan lähialueen kaikki havainnot huomioidaan tilapäivityksessä painotettuna todennäköisyydellä millä ne ovat alkujaan seurattavasta kohteesta.[24] Kyseisellä tavalla toimittaessa on intuitiivisesti todennäköisempää, että kohteesta todella lähtöisin oleva havainto huomioidaan seurannan päivityksessä kuin lähimmän naapurin assosiaatiossa. Näin ollen yksittäinen harhahavainto, joka noudattaa seurattavan kohteen oletettua tilamuutosta tarkemmin kuin kohteesta saatava havainto, ei pääse suistamaan seurantaa täysin pois radaltaan.

3. KONEOPPIMINEN

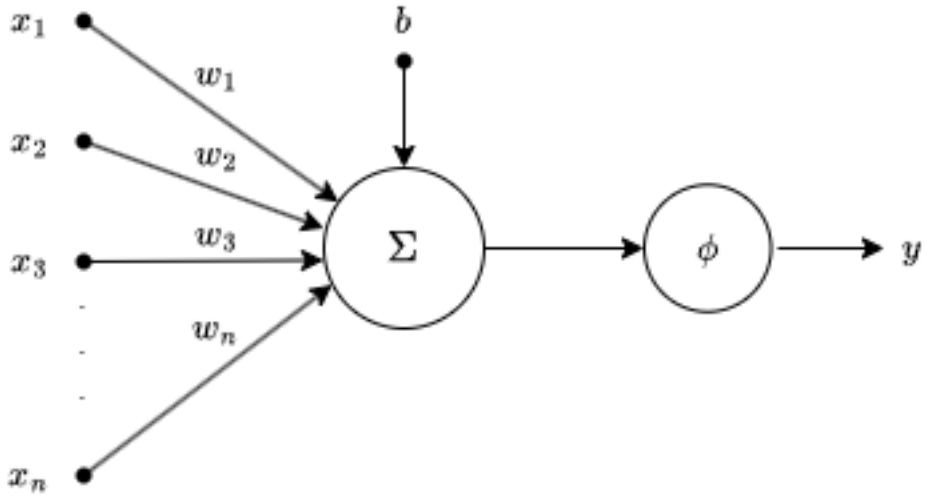
Tekoäly ja koneoppiminen ovat kehittyneet viimeisien vuosien aikana huomattavasti. Vaikka kyseisiä termejä käytetäänkin arkikielessä välillä samaa tarkoittavina, ne ovat kuitenkin erillisiä konsepteja eikä niitä tulisi käyttää synonyymeinä.

Tekoälyllä tarkoitetaan koneen tai muun ihmisestä poikkeavan toimijan kykyä ilmentää tietoisuutta, oppia ja tehdä päätöksiä. Tällaista tietoisuutta ilmentävää tekoälyä voidaan kutsua supertekoälyksi [18]. Toimijan tietoisuutta on kuitenkin haastava tutkia sekä määrittää, eikä tällaisia supertekoälyn ilmentymiä ole vielä kehitetty. Muita tekoälyn luokkia ovat heikko ja vahva tekoäly. Vahva tekoäly pystyy suoriutumaan monista eri tehtävistä, oppimaan niistä sekä toimimaan itsenäisesti. Myöskään vahvaa tekoälyä ilmentäviä toimioita ei ole vielä kehitetty. Heikolla tekoälyllä tarkoitetaan hyvin rajattuun tehtävään suuniteltuja algoritmeja, jotka ihminen on asettanut suorittamaan kyseistä tehtävää. Ne eivät siis pysty toimimaan itsenäisesti, vaan vaativat ihmisen tekemää ohjausta. [18]. Tässä työssä keskitytään koneoppimiseen, joka on heikon tekoälyn alaluokka. Koneen tai algoritmin voidaan määritellä oppivan, mikäli se pystyy keräämään kokemusta suorittamistaan tehtävistä ja pystyy suoriutumaan tulevaisuudessa vastaavanlaisista tehtävistä paremmin tämän avulla [7]. Algoritmien tapauksessa tämä tapahtuukin keräämällä kokemusta käyttötarkoitukseen kerätyn datan ja valitun virhefunktion avulla. Oppimista seurataan tarkkailemalla virhefunktion kehitystä.

Monet todellisen maailman ongelmat ovat hyvin haastavia ratkaista käyttäen perinteisiä algoritmipohjaisia menetelmiä. Esimerkiksi ihmiselle hyvin intuitiivisen kuvantunnistamisongelman muotoilu käyttäen perinteistä algoritmillista menettelyä on todella haastavaa.[13] Koneoppimismenetelmät tarjoavat korkeamman abstraktiotason, jossa oppimisalgoritmit oppivat tehtävän kannalta tärkeimmät päättelyketjut suoraan datasta. Seuraavaksi esitellään yhden yleisimmin käytössä olevan ja paljon huomiota saaneen koneoppimismenetelmän eli neuroverkkojen toimintaperiaatteet.

3.1 Neuroni

Neuroverkot ovat biologista informaation prosessoinnista inspiraation saaneita oppimisalgoritmeja. Ne koostuvat prosessointi solmuista eli neuroneista ja niiden välisistä kytkennöistä. Neuronit perustuvat McCullohin ja Walter Pittsin vuonna 1943 tekemässä ”A



Kuva 3.1. *Yksinkertaisen neuronin rakenne.*

logical calculus of the ideas immanent in nervous activity"-julkaisussa [20] esitettyyn neuronin laskennalliseen malliin. Kyseinen McCulloh-Pitts-neuroni vastaanottaa siihen kytketyiltä muilta neuroneilta tulevat aktivaatiot ja muodostaa niistä ulostulon kytkentöihin liittyvien painokertoimien ja oman aktivointifunktionsa avulla. Neuronin sisääntuloilla eli synapseilla on omat painokertoimet, joita merkitään merkillä w. Lisäksi neuronille asetetaan lineaarinen tai epälineaarinen aktivointifunktio, jonka avulla sisääntuloista muodostetaan neuronin aktivaatio.[9, s.5-6] Neuronin aktivaatio eli ulostulo voidaan laskea siis seuraavaa yhtälöä käyttäen

$$y = \phi(\sum_{j=1}^{n}[w_j x_j] + b). \tag{3.1}$$

Yhtälö kuvaa neuronin ulostuloa, kun siihen on liitetty n sisääntuloa. Näitä sisääntulo arvoja on merkitty vektorilla x. Jokaiselle sisääntulolle löytyy vastaava painokerroin vektorista w, joka kuvaa kyseisen sisääntulon vaikutuksen merkitsevyyttä neuronin ulostuloon. Lisäksi yhtälössä on bias-termi b, joka on kyseiselle neuronille vakio. Käytännössä neuroni siis summaa painotettuja sisääntuloja lisäten lopuksi neuronikohtaisen vakion. Tämän jälkeen tuloksesta lasketaan ulostuloarvo aktivaatiofunktion avulla. Kuvassa 3.1 on esitetty yhtälöä 3.1 vastaavan neuronin rakenne.

Yksinkertaisin aktivaatiofunktio on lineaarinen kuvaus, jolloin neuronin ulostulo on summayhtälön mukainen. Kuitenkin yleisesti käytetyt aktivaatiofunktiot ovat luonteeltaan epälineaarisia. Tarpeeksi suuren epälineaarisia aktivaatioita käyttävän neuroverkon on todettu pystyvän approksimoimaan mielivaltaisia funktioita ja niitä voidaan näin ollen pitää universaaleina aproksimaattoreina [16]. Neuroverkkoja hyödynnetään monilla tieteenaloilla analyyttisena työkaluna. Niitä hyödynnetään esimerkiksi signaalinkäsittelyssä, regressioanalyysissa, ennustuksessa, hahmontunnistuksessa, järjestelmän karakterisoinnissa, klusteroinnissa ja luokittelussa. Nämä kaikki pohjautuvat niiden kykyyn approksimoida

erilaisia funktioita. Lisäksi jotain neuroverkkotyyppejä kuten Hopfieldin-verkkoa voidaan hyödyntää kombinatorisessa optimoinnissa [9, s.11].

3.2 Aktivointifunktiot

Neuroverkkojen kyky kuvata monimutkaisia epälineaarisia funktioita johtaa juurensa niissä käytettäviin epälineaarisiin aktivointifunktioihin. Mikäli tarkastellaan luvussa 3.1 esiteltyä neuronin yhtälöä, voidaan huomata, että mikäli käytettävä aktivointifunktio on lineaarinen kuvaus, neuronin ulostulo pystyy kuvaamaan vain ensimmäisen asteen polynomeja. Tällöin syväkin neuroverkko yksinkertaistuu vain lineaariseksi regressiomalliksi, jolla ei ole kykyä approksimoida korkeamman asteen funktioita. [26]

Epälineaariset aktivointifunktiot tuottavat yksittäisen neuronin kohdalla epälineaarisen kuvauksen sisään- ja ulostulojen välille. Kun tällaisista epälineaarisia kuvauksia sisältävistä neuroneista koostetaan neuroverkko, se pystyy kuvaamaan monimutkaisia datasta löytyviä epälineaarisia yhteyksiä. Neuroverkkojen kouluttamiseen liittyvän vastavirta-algoritmin käyttäminen kuitenkin vaatii, että käytettävien aktivointifunktioiden tulee olla derivoituvia [26].

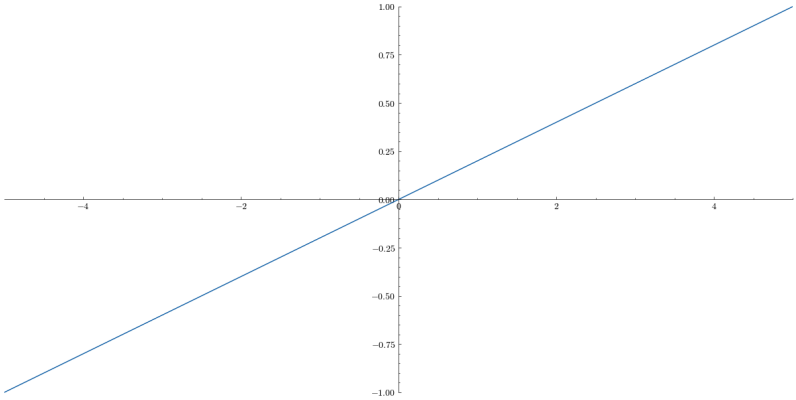
Aktivointifunktiot ovat skalaarimuunnoksia. Ne jakautuvat kahteen eri luokkaan: muunnoksiin, joissa ulostulojen arvot eivät ole rajattu millekään tietylle arvovälille ja muunnoksiin, joissa ulostulo on rajattu tiettyyn arvoväliin. Epälineaarisia rajaamattomalla ulostuloväliillä olevia aktiovointifunktioita ovat esimerkiksi ReLU (engl. Rectified linear unit) ja sen variantit Leaky ReLu ja Parametrized ReLU. Rajatulla arvovälillä toimivia epälineaarisia aktivointifunktioita ovat esimerkiksi Sigmoid, Tanh ja Binary Step Function.

Käydään seuraavaksi tarkemmin läpi työssä hyödynnetyt aktivointifunktiot. Näitä ovat lineaarinen aktivointifunktio sekä Sigmoid- ja Tanh-aktivointifunktio.

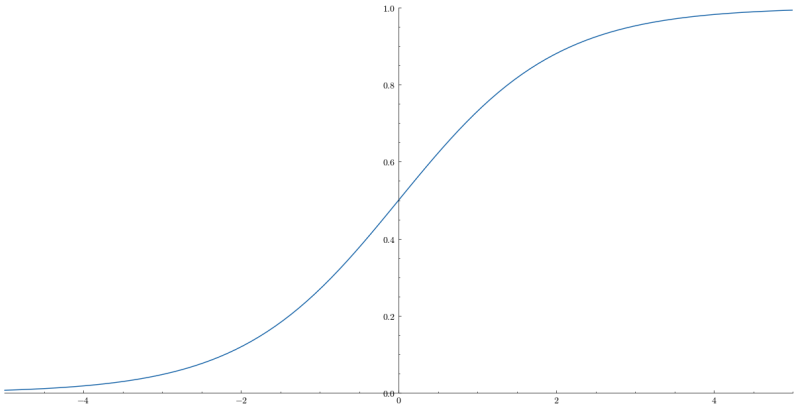
Lineaarinen aktivointifunktio toimii seuraavan yhtälön mukaisesti $f(x) = ax$, jossa a on käyttäjän valitsema vakio. Tästä voidaan huomata, ettei funktion ulostulojen arvot ole rajoittunut millekään välille, vaan ne riippuvat pelkästään funktion sisääntulosta ja valitusta kertoimesta a. Koska kyseessä on yksinkertainen lineaarinen funktio, sen derivaatta on sisääntulosta riippumatta vakio. Vakioderivaatan vuoksi kyseisen aktivaatiofunktion käyttämisestä neuroverkon piilotetuilla kerroksilla ei ole juurikaan hyötyä virhefunktion optimointia ajatellen [26]. Kuvassa 3.2 on esitetty yksinkertaisen lineaarisen aktivointifunktion $f(x) = 0.2x$ ulostulot arvovälillä [-5,5].

Sigmoid-aktivaatiofunktio on yleisesti käytetty ei lineaarinen aktivaatiofunktio, jonka ulostulot on rajoittunut välille $(0, 1)$. Muodoltaan funktio on kuvan 3.3 mukainen S-kurvi. Sen yhtälö voidaan kirjoittaa muodossa

$$sig(t) = \frac{1}{1 + e^{-}t}, \tag{3.2}$$



Kuva 3.2. *Kuvassa yksinkertainen lineaarinen aktivointifunktio.*



Kuva 3.3. *Kuvassa sigmoid-aktivointifunktio*

jossa t kuvaa funktion sisääntuloa. Sigmoid-funktio on derivoituva ja sen derivaatta eri pisteissä voidaan laskea seuraavaa yhtälöä käyttämällä

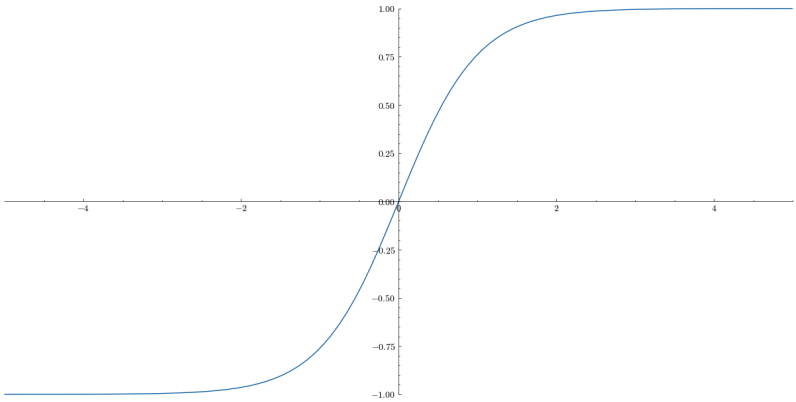
$$\frac{\mathrm{d}sig(t)}{\mathrm{d}t} = 1 - sig(t) \tag{3.3}$$

Sigmoid-funktio ei ole kuitenkaan nollan suhteen symmetrien, jonka takia kaikki ulostulot ovat positiivisia, mikä ei sovi kaikkiin sovelluskohteisiin. Tanh-aktivointifunktio on muodoltaan vastaavanlainen kuin edellä esitelty Sigmoid-aktivointifunktio. Sen yhtälö voidaankin kirjoittaa muodossa

$$f(t) = 2sig(2t) - 1, \tag{3.4}$$

josta voidaan nähdä että se on vain skaalattu ja siirretty versio Sigmoid-aktivointifunktiosta. Tanh-aktivointifunktio on kuitenkin nollan suhteen symmetrinen ja näin ollen sen ulostulo voi saada sekä positiivisia että negatiivisia arvoja. Kyseisen ominaisuuden vuoksi sitä käytetään yleisesti Sigmoid-aktivointifunktion sijaan.[26]

Oikean aktivointifunktion valitseminen haluttuun tehtävään ei kuitenkaan ole yksinkertainen tehtävä ja sen vaikutus neuroverkon lopulliseen suorituskykyyn on merkittävä. Tästä syystä suunniteltaessa neuroverkkoihin perustuvaa ratkaisua, on syytä tehdä tarkkaa



Kuva 3.4. *Kuvassa Tanh-aktivointifunktio*

tutkimusta ja testausta erilaisilla aktivaatiofunktioilla. Aktivaatiofunktion valinta on täysin aplikaatiokohtaista, eikä sen valintaan ole saatavilla ohjesääntöjä. Kuitenkin jokaisella aktivaatifunktiolla on tiettyjä heikkouksia ja vahvuuksia, jotka on hyvä huomioida tutkimusta tehtäessä. ReLU-aktivointifunktio on yleisimmin käytössä oleva ja se yleensä suoriutuu muita aktivointiofunktioita paremmin. Se ei kuitenkaan yleisesti sovi käytettäväksi neuroverkon ulostulokerroksessa, sillä se ei ole sileä funktio ja voi näin ollen tuottaa epävakaita ennusteita. [26] Tanh- ja Sigmoid-aktivaatiofunktiot voivat johtaa gradientin katoamiseen (engl. Vanishing gradient problem). Häviävät gradientit voivat näin ollen pysäyttää koko neuroverkon oppimisen [2]. ReLu:n käyttäminen voi johtaa myös niin sanottuihin kuolleisiin neuroneihin, jossa neuronin ulostulo on nolla sisääntulosta riippumatta. Koska funktion derivaatta on tällä alueella myös nolla, neuroni voi jäädä lopullisesti tähän tilaan ja olla neuroverkon toiminnan kannalta hyödytön.

3.3 Neuroverkkojen kouluttaminen

Neuroverkkojen kouluttamisen tarkoituksena on löytää sille sellaiset painot W, joilla se pystyy kuvaamaan harjoitusdatassa olevat sisääntulot x_p toivottuihin ulostuloihin y_p. Samalla opitun hypertason tulisi myös pystyä kuvaamaan datan lähteenä olevaa prosessia myös niissä pisteissä, joista harjoitusdataa ei ole saatavilla. Tällä tarkoitetaan neuroverkon generalisoitumista, joka tarkoittaa neuroverkon kykyä kuvata harjoitusdatan lähteenä olevan prosessin tilastollisia ominaisuuksia. Neuroverkon kouluttamisen voidaan ajatella tarkoittavan korkeauloitteisen tason sovittamista harjoitusdataan. Tällöin generalisoitumisella tarkoitetaan sitä kuinka tämän tason interpolointi sekä ekstrapolointi vastaa samasta prosessista syntyvien muiden näytteiden ominaisuuksia. [9, s. 15–20]

Neuroverkkojen kouluttamiseen on olemassa monia erilaisia strategioita riippuen harjoitusdatan rakenteesta. Näitä ovat esimerkiksi ohjattu oppiminen (engl. Supervised learning), ohjaamaton oppiminen (engl. Unsupervised learning) ja vahvistusoppiminen (engl. Reinforcement learning). Vahvistusoppiminen on kuitenkin ohjatun oppimisen erikoista-

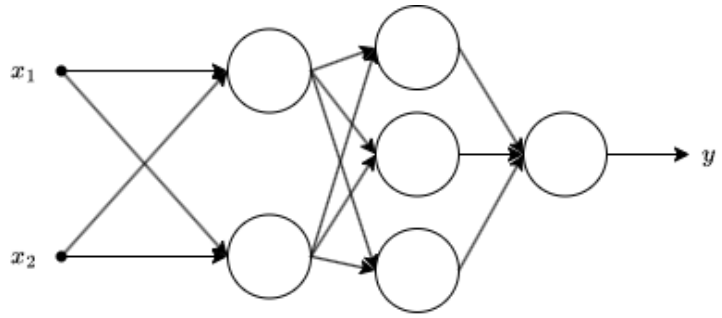
paus, joten sitä ei käsitellä tarkemmin. Matemaattisesti, ohjatun ja ohjaamattoman oppimisen voidaan ajatella eroavan siinä, että ohjatussa oppimisessa neuroverkon on tarkoitus oppia tiheysfunktio $p(y|x)$ joka kuvaa tiedettyjen ulostulojen ja niitä vastaavien sisääntulojen ehdollista todennäköisyyttä, kun taas ohjaamattomassa oppimisessa neuroverkon on tarkoitus oppia tiheysfunktio $p(x)$, joka kuvaa sisääntulojen tilastollisia ominaisuuksia. Ohjattua oppimista käytetään yleisimmin kategorisoinnissa, approksimoinnissa sekä optimoinnissa. Ohjaamatonta oppimista taas käytetään lähinnä datan klusteroinnissa ja piirteiden erottelussa. [9, s.15]. Ohjatun oppimisen toimintaperiaatteita kuvataan tarkemmin luvussa 3.6

3.4 Neuroverkot

Neuroverkot ovat yleisesti useista laskentayksiköistä eli neuroneista koostuvia kokonaisuuksia. Niiden rakenne on usein kerrosmainen, jossa jokainen kerros sisältää useita neuroneita. Jokainen neuroverkko sisältää sisääntulokerroksen ja ulostulokerroksen. Näiden välissä on sovelluskohteesta riippuva määrä piilokerroksia, jotka prosessoivat sisääntulokerrokselta saadun datan [28]. Sisääntulokerroksen tarkoituksena on vastaanottaa neuroverkolle annettava data ja välittää se piilokerroksille prosessoitavaksi. Ulostulokerros vastaanottaa piilokerroksien prosessoiman datan ja muuntaa sen neuroverkon ulostuloksi. Piilokerrokset voivat sisältää kerroksia, joissa neuroneita on huomattavasti enemmän kuin sisääntulokerroksen piirteitä, joka mahdollistaa korkeamman asteen approksimaation [9]. Kuitenkin on hyvä huomata, ettei piilokerroksilla olevien neuronien määrä riipu sisääntulokerroksen piirteiden lukumäärästä, vaan todellisen approksimoitavan funktion muodosta [9]. Eri kerroksilla olevat neuronit ovat kytkeytyneet toisiinsa neuroverkon tyypin mukaisesti. Esimerkkejä erilaisista syvistä neuroverkkoarkkitehtuureista ovat eteenpäin kytketty verkko (engl. Feedforward network), takaisinkytketty neuroverkko RNN (engl.Recurrent neural network), konvoluutioneuroverkkoverkko CNN (engl. Convolutional neural network) ja autoenkoodaajaneuroverkko (engl. Autoencoder network)

Eteenpäin kytketyt neuroverkot ovat arkkitehtuuriltaan yksinkertaisimpia. Nimensä mukaisesti niissä neuronien väliset yhteydet ovat aina edelliseltä kerrokselta seuraavalle. Yksinkertaisimmillaan täysin kytketyssä eteenpäin kytketyssä neuroverkossa kaikki edellisen kerroksen neuronit ovat liittyneet kaikkiin seuraavan kerroksen neuroneihin. Kuvassa 3.5 on esitetty tällaisen yksinkertaisen eteenpäin kytketyn neuroverkon rakenne. Neuroneita on kuvattu valkoisilla ympyröillä ja niiden välisiä yhteyksiä nuolilla. Painokertoimet sekä yksittäisten neuronien ulostulot on jätetty yksinkertaisuuden vuoksi pois kuvasta. Kuvasta voidaan havaita verkon kerrosmainen rakenne. Se sisältää yhteensä kolme kerrosta. Ensimmäinen kerros on sisääntulokerros ja viimeinen ulostuloskerros. Näin ollen kyseinen verkko sisältää vain yhden piilokerroksen.

Koska neuroverkkoja kouluttaessa ja niiden ulostuloja laskettaessa on yleistä, että tä-



Kuva 3.5. *Yksinkertaisen kolmikerroksisen eteenpäin kytketyn neuroverkon rakenne.*

mä tehdään useille sisääntulo-ulostulo pareille, käytetään eri näytteiden erottelemiseksi alaindeksia p. Käydään seuraavaksi läpi tällaisen eteenpäin kytketyn neuroverkon rakenne yhtälömuodossa. Verkon ulostuloa merkitään vektorilla y_p. Yläindekseillä kuvataan kerroksen järjestysnumeroa. Kerroksen m ulostuloa merkitään vektorilla $o^{(m)}$. Sisääntulovektoria näytteellä p kuvataan merkillä x_p, mutta se voidaan kirjoittaa aikaisempaa notaatiota käyttämällä myös $o_p^{(1)}$. Verkon painovektoreita merkitään matriisilla W ja näin ollen esimerkiksi kerroksien m ja $m-1$ välinen painovektori on W^{m-1}. Kerroksen m bias vektoria merkitään merkillä $\theta^{(m)}$ ja sillä käytettyä aktivointifunktiota merkillä $\phi^{(m)}$. Aktivointifunktiota käytetään aina elementtikohtaisesti ja tässä notaation yksinkertaistamiseksi on oletettu, että samalla kerroksella jokaisen neuronin ulostulolle sovelletaan samaa aktivaatiofunktiota. Neuroverkon kerroksella m olevien neuronien lukumäärää merkitään tässä työssä merkillä J_m. Verkon sisältämien kerroksien lukumäärää merkitään tässsä työssä merkillä M. Koska kyseessä on täysin kytketty neuroverkko, voidaan kerroksen m neuroneille kirjoittaa ulostulot seuraavasti:

$$net_p^{(m)} = [W^{(m-1)}]^T o_p^{(m-1)} + \Theta^{(m)} \tag{3.5}$$

$$o_p^{(m)} = \phi^{(m)}(net_p^{(m)}) \tag{3.6}$$

Yleisesti kerroksilla on useampia neuroneita, jonka takia kaikki laskut ovat vektorimuotoisia.

3.5 Takaisinkytketyt neuroverkot

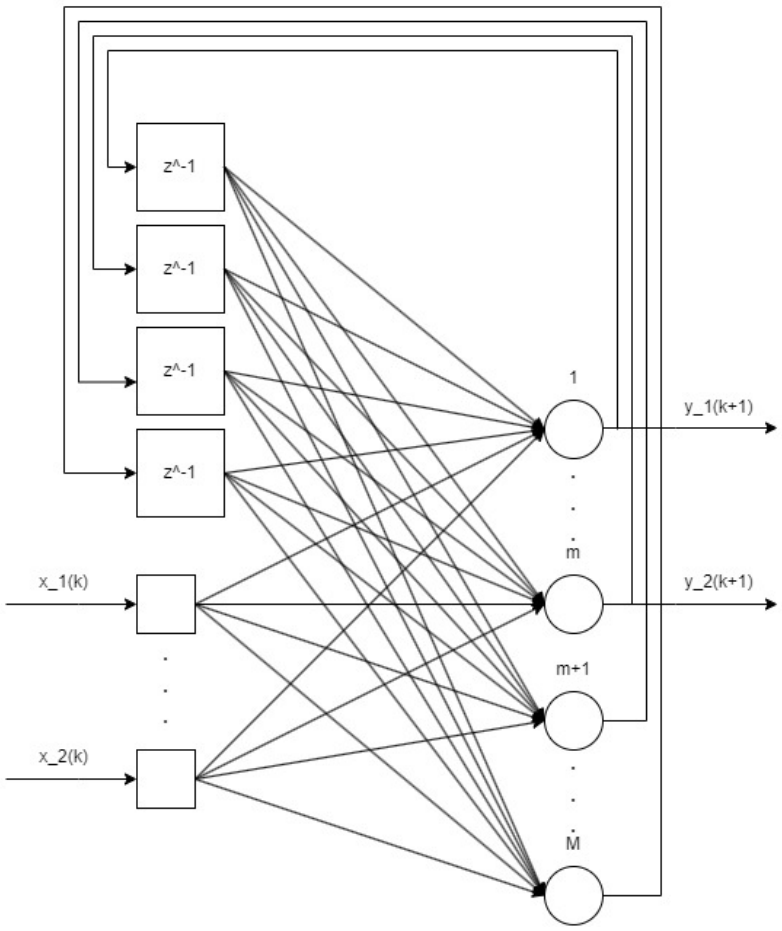
Takaisinkytketyt neuroverkot ovat aikasarjojen ja aikariippuvaisten dynaamisten systeemien prosessointiin sopivia syvän oppimisen menetelmiä. Eteenpäin kytketyistä neuroverkoista eroten, ne pystyvät rakenteensa ansiosta toimimaan muuttuvan pituisten aikasarjojen kanssa. Tällaisia signaaleja tai aikasarjoja, joissa näytteiden aikarelaatioilla on analysoinnin kannalta merkitystä, on olemassa monia. Esimerkkinä tällaisista sovelluskohteista voidaan ottaa hahmontunnistus EEG-signaalista, jossa näytteiden aikadimensiolla on merkitystä lopputuloksen kannalta.[12, luku 11]. Dynaamisia prosesseja ja niistä

syntyviä signaaleja tutkittaessa käytettävän neuroverkon tulisi myös olla rakenteeltaan dynaaminen. Näin ollen neuroverkon täytyisi pystyä ylläpitämään mallinnettavan systeemin tilaa eri ajanhetkillä ja täten sisältää muistia.[21, s.32] Erilaiset muistityypit voidaan jakaa lyhyeen ja pitkään muistiin. Lyhyellä muistilla tarkoitetaan systeemin hetkellisestä tilasta säilytettävää oleellista informaatiota. Pitkällä muistilla tarkoitetaan taas pitkään tai lopullisesti ylläpidettävää informaatiota systeemin perustavanlaatuisesta dynamiikasta. Muisti voidaan sisällyttää neuroverkkoihin monilla eri tavoilla. Yksi esimerkki on aikasarjan sisällyttäminen suoraan neuroverkon sisääntuloon. [21, s.32] Yksinkertaisten eteenpäinkytkettyjen neuroverkkojen tapauksessa tämä kuitenkin tarkoittaa, että käytettävän aikaikkunan pituus täytyy määrittää etukäteen, eikä ne täten pysty toimimaan muuttuvan pituisten aikasarjojen kanssa. Toinen mahdollinen tapa sisällyttää systeemin dynamiikkaa ja näin ollen muistia neuroverkkoon on käyttämällä takaisinkytkentään neuronien välillä [21, s.32].

Korkeimmalla tasolla takaisinkytketyt neuroverkot voidaan jakaa globaalisti takaisinkytkettyihin ja lokaalisti takaisinkytkeittyihin verkkoihin. Globaalisti takaisinkytketyt verkot voivat sisältää takaisinkytkentöjä kaikkien neuroniparien välillä. Yksi esimerkki tällaisesta globaalista takaisinkytketystä verkosta on täysin takaisinkytketty neuroverkko, jota kutsutaan myös reaaliaikaiseksi takaisinkytketyksi neuroverkoksi (engl. Real Time Recurrent Network). Sen rakenne on esitetty kuvassa 3.6. Se sisältää kokonaisuudessaan M neuronia, joista indeksit $1 - m$ toimivat verkon ulostuloina ja loput ovat verkon sisäisiä piiloneuroneita. Jokaisen neuronin välillä on takaisinkytkentä viive-elementtien z^{-1} kautta, jotka kuvaavat systeemin tilaa. Kyseinen verkkomalli ei sisällä kerrosrakennetta eikä näin ollen sisällä mitään eteenpäinkytketyn verkon piirteitä. Kyseinen rakenne mahdollistaa monimutkaisten dynaamisten riippuvuuksien kuvaamisen, mutta sen rakenteen suuri kompleksisuus $O(n^2)$, heikko stabiilisuus ja hidas konvergoituminen tekevät siitä yleisesti soveltumattoman käytännön ongelmien ratkaisemiseen.[21, s.32]

Lokaalisti takaisinkytketyt neuroverkot taas sisältävät dynaamisia sisäisiä takaisinkytkentöjä, kuitenkin niin, että neuronien väliset yhteydet ovat aina verkossa eteenpäin. Näissä piilokerroksilla olevat neuronit kuvaavat kuvattavan systeemin tilaa ja niiden arvot riippuvat myös systeemin aikaisemmista tiloista. Kyseisissä verkoissa käytetään aikadimensiossa identtisiä prosessointiyksiköitä, jonka takia niiden kouluttamiseen voidaan soveltaa perinteistä vastavirta-algoritmia sekä gradienttimenetelmiä. [12, luku 11] Käytännössä tämä tarkoittaa, että jokaisella ajanhetkellä neuroverkko prosessoi sisääntulon samaa funktiota käyttämällä huomioiden aikaisemmilta ajanhetkiltä kerrytetyt systeemin tilaa kuvaavat arvot. Lokaalisti takaisinkytketyistä neuroverkoista on olemassa monia arkkitehtuuriltaan eroavia variantteja. Käydään seuraavaksi läpi niistä kolme. Ensimmäisenä vanilla RNN, jota tullaan jatkossa kutsumaan yksinkertaisesti termillä RNN. Toisena RNN:stä kehitetty uudempi variantti GRU (engl. Gated Recurrent Unit) ja viimeisimpänä LSTM (engl. Long-short-term-memory) variantti.

Kuvassa 3.7 on esitetty RNN verkon yksinkertaistettu rakenne sekä oikealla kuvattu kuin-



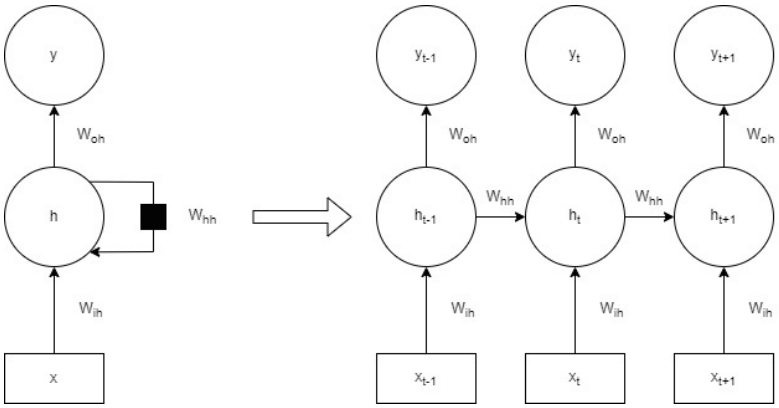
Kuva 3.6. *Täysin takaisinkytketty neuroverkko*

ka piilokerroksien tilat siirtyvät ajanhetkestä toiseen. RNN verkon piilokerroksen h ulostulo sekä verkon ulostulo y voidaan laskea seuraavia yhtälöitä käyttämällä

$$h_t = \phi(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \tag{3.7}$$

$$y_t = W_{ho}h_t + b_o \tag{3.8}$$

Näissä alaindeksit t kuvaavat ajanhetkeä aikasarjoissa ja ovat täten kokonaisluvut väliltä $1 - T$, kun T on aikasarjan pituus. Vektori x_t kuvaa verkon sisääntuloa ajanhetkellä t. Vektori h_t kuvaa RNN-neuroverkon piilotettua tilaa ajanhetkellä t. Painomatriisi W sisältää erilliset painot ulkoisen sisääntulon ja edellisen ajanhetken piilotetun tilan huomiointiin sekä ulostulon laskemiseen. Merkillä ϕ tarkoitetaan piilotetulla kerroksella käytettäviä aktivointifunktioita. [28, s.27] Yhtälöistä voidaan nähdä, että painomatriisin indeksillä ih huomioidaan ulkoinen vaikute eli sen hetkinen sisääntulo. Indeksillä hh olevilla painoilla huomioidaan edellisen ajanhetken piilotettu tila. Yksikön ulostulo lasketaan hyödyntämällä painomatriisin indeksillä ho löytyviä painoja. Termit b tarkoittavat verkon bias-termejä ja myös ne ovat eri piilotetun tilan ja ulostulon laskentaan. Yksinkertaisuu-



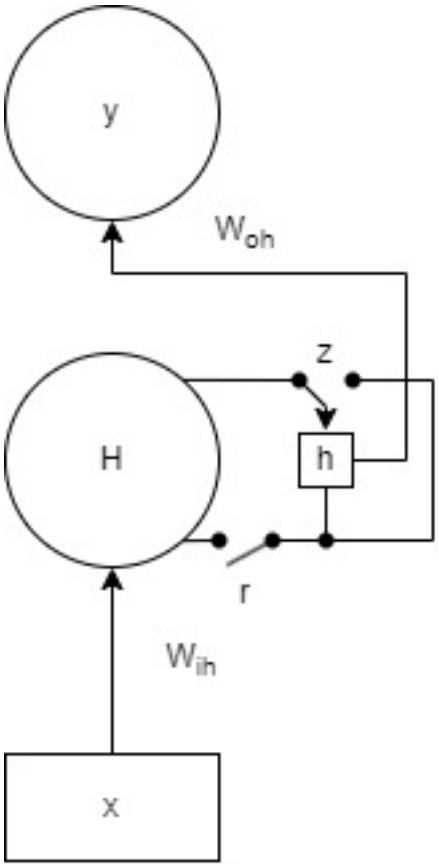
Kuva 3.7. *RNN verkko sekä sen aika-avaruudessa levitetty versio*

den vuoksi verkon ulostuloon ei ole lisätty aktivointifunktiota. RNN-neuroverkko voidaan kouluttaa vastavirta-algoritmia hyödyntäen, kun se suoritetaan ajassa levitetylle verkkorakenteelle (engl. Backpropagation through time, BBTT).[28, s.27]. Yksinkertainen RNN-rakenne kärsii kuitenkin katoavista gradienteista, jonka takia sen kouluttaminen pitkillä aikasarjoilla on haastavaa[28, s.28] [3]

Pitkät aikasarjat ja näin ollen gradienttien vaikutuksien ulottaminen useiden aikakerroksien läpi voi aiheuttaa lähellä nollaa olevia osittaisderivaattoja, joka aiheuttaa katoavan gradientin ongelman. [12, luku 11][28, s.28]. GRU- ja LSTM-verkot pyrkivät vähentämään tämän ongelman vaikutuksia lisäämällä prosessointiyksiköihin portteja, jotka vaikuttavat kuinka verkot hyödyntävät niiden historiallisia tiloja [28]. Kyseiset yksiköt sisältävät prosessointiyksiköiden sekä porttien ansiosta suoria yhteyksiä historiallisiin tiloihin, jotka mahdollistavat hitaasti muuttuvien riippuvuuksien löytämisen. Koska verkot oppivat portteihin liittyvät painot suoraan datasta, ne pystyvät löytämään ne datariippuvaiset piirteet, joita tilasta on lopputuloksen kannalta tarpeellista pitää muistissa.[12]

Käydään ensimmäisenä läpi näistä rakenteeltaan yksinkertaisempi eli GRU. Sen rakennetta on kuvattu kuvassa 3.8. Kyseinen verkkorakenne esiteltiin konekääntämiseen liittyvässä julkaisussa vuonna 2014 [8]. Kyseinen rakenne sisältää yksinkertaiseen RNN rakenteen lisäksi nollausportin r (engl. Reset port) ja päivitysportin z (engl. Update port). Nollausportin tarkoituksena on määrittää, kuinka paljon verkon piilotettua tilaa käytetään uutta tilaa laskettaessa. Päivitysportin tarkoituksena taas on päättää, kuinka paljon aikaisemmasta tilasta käytetään uutta tilaa laskettaessa. Näin ollen nollausportti mahdollistaa tilan pitämisen kompaktina poistamalla tilasta pidemmällä aikavälillä hyödyttömiä piirteitä kun taas päivitysportti mahdollistaa pitkän aikavälin muistin.[8] Kuvassa 3.8 olevasta esityksestä huolimatta, kyseessä ei ole binääriset portit. Kyseiset portit toimivat sigmoid-aktivaatiofunktiota käyttämällä ja sallivat näin arvot välillä 0–1. GRU-neuroverkon porttien toimintaa voidaan kuvata yhtälömuodossa seuraavasti

$$r_t = \sigma(W_r x_t + U_r h_{(t-1)}) \tag{3.9}$$



Kuva 3.8. *Gru verkon rakenne*

$$z_t = \sigma(W_z x_t + U_z h_{(t-1)}) \tag{3.10}$$

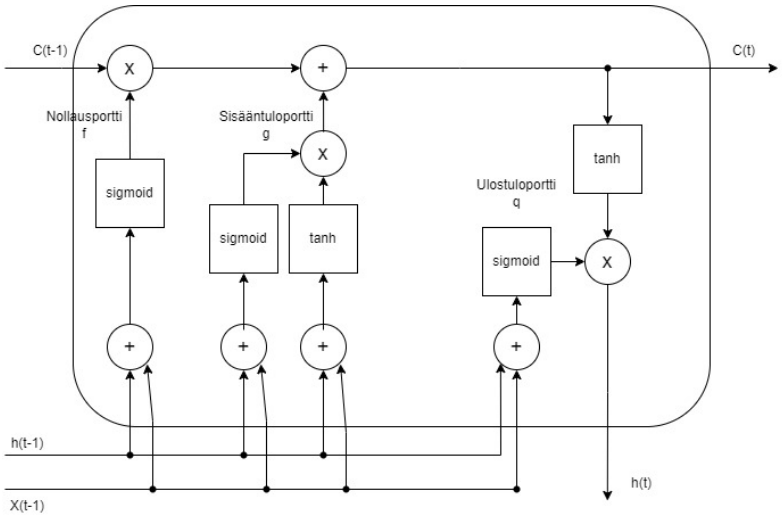
Tilakandidaatti H ja tila h lasketaan seuraavasti

$$H = \phi(W_i h x_t + U_h(r_t \odot h_{(t-1)}) \tag{3.11}$$

$$h_t = (1 - z_t) \odot h_{(t-1)} + z_t \odot H_t \tag{3.12}$$

Alaindeksit, jotka sisältävät merkin t kuvaavat aikariippuvaisia muuttujia. Näissä x_t kuvaa verkon sisääntuloa, σ kuvaa sigmoid-aktivaatiofunktiota, ϕ tanh-aktivaatiofunktiota, $h_{(t-1)}$ verkon tilaa edellisenä ajanhetkenä. Matriisit W ja U kuvaavat neuroverkon oppimia painovektoreita. Merkillä $\odot$ kuvataan matriisien elementtikohtaista tuloa eli Hadamardin tuloa. [8] Vaikkakaan bias-termejä ei ole tässä implisiittisesti mainittu, voidaan ne sisällyttää painomatriiseihin helposti muokkaamalla sisääntulovektoreita sopivalla tavalla. Verkon ulostulo ajanhetkellä t voitaisiin siis laskea vastaavasti kuin RNN verkon tapauksessa käyttämällä yhtälöä 3.8.

LSTM-verkon rakenne esiteltiin vuonna 1997 ”Long short-term memory”-julkaisussa [15]. Sen rakenne sisältää kolme porttia, jonka takia sen rakenteen esittäminen GRU-verkkoa



Kuva 3.9. *LSTM-prosessointiyksikkö*

vastaavalla tavalla on haastavaa. Tästä syystä kuvassa on 3.9 on esitetty vain yksittäinen LSTM-yksikkö. LSTM-verkko kuitenkin rakentuu vastaavasti kuin normaali RNN-verkko, jossa verkon piilotetut tilat, on korvattu LSTM-prosesointiyksiköillä. Vastaavasti kuin GRU-verkossa, LSTM-prosessointiyksikön nollausportin tehtävänä on poistaa tarpeetonta informaatiota verkon tilasta. Sisääntuloportti kontrolloi, mitä verkon sisäisen tilan arvoja päivitetään ja sen vieressä oleva tanh-aktivaation sisältämä haara luo vektorin, joilla näitä arvoja päivitettään. Ulostuloportti suodattaa oppimiensa painojen perusteella yksikön sisäisestä tilasta tarpeelliset arvot yksikön ulostuloon.[28, s.28-29] Koska LSTM- ja GRU-prosessointiyksiköiden toiminta on hyvin samankaltaista, ohitetaan LSTM-yksikön yhtälöiden esittäminen tässä työssä.

3.6 Ohjattu oppiminen

Ohjatussa oppimisessa neuroverkon harjoitusdatasetti sisältää alkuarvot x_p sekä niitä vastaavat tosiarvot y_p. Esimerkiksi kuvien kategorisointiin tarkoitetun neuroverkon tapauksessa tämä tarkoittaisi, että x_p voisi vastata kuvaa ja y_p siinä esiintyvää kategoriaa. Tämän lisäksi täytyy määrittää käytettävä virhefunktio, joka voi datan tyypin mukaan olla esimerkiksi keskineliövirhe MSE (engl. Mean-squared error). Se voidaan kirjoittaa yhtälömuodossa seuraavasti

$$E = \frac{1}{N} \sum_{p=1}^{N} \|y_p - \hat{y}_p\|^2. \tag{3.13}$$

Tässä N kuvaa harjoitusdatassa olevien näytteiden määrää, y_p kuvaa yksittäistä harjoitusnäytteeseen liittyvää tosiarvoa ja $\hat{y_p}$ vastaa neuroverkon ulostuloa vastaavalle harjoitusnäytteelle.[9]. Neuroverkon kouluttaminen on iteratiivinen prosessi, jossa virhefunktion arvoa yritetään minimoida. Tämä tehdään näyttämällä harjoitusdata kokonaisuudes-

saan neuroverkolle useita kertoja. Yhtä tällaista optimointikierrosta kutsutaan kierrokseksi (engl. Epoch). Jokaisen kierroksen jälkeen lasketaan keskimääräinen virhe käyttämällä valittua virhefunktiota, jonka jälkeen painoja muokataan hieman gradientin vastaiseen suuntaan. Tämä iteratiivinen virheen laskeminen sekä painojen muokkaaminen johtaa neuroverkon oppimiseen.

3.7 Vastavirta-algoritmi

Vastavirta-algoritmi on neuroverkkojen optimointiin käytetty algoritmi, joka hakee virhefunktion globaalia minimiä käyttäen virhefunktion jyrkintä laskusuuntaa [25][9]. Koska vastavirta-algoritmi käyttää virhefunktion gradientteja eri neuroneiden suhteen optimoinnissaan, tulee jokaisen verkossa olevan neuronin aktivaatiofunktion olla jatkuva ja differentioituva [9, s.85-88]. Verkon virhefunktio voidaan määrittää käyttämällä verkon kouluttamiseen käytettyjä sisääntulo ja toivottuja ulostulo pareja $x_p, y_p \in S$. Tässä S kuvaa optimointiin käytettävää osaa harjoitusdatasta, joka voi käytännössä olla satunnaisesti valittu alijoukko koko harjoitusdatasta tai koko harjoitusdatasetti. Yksinkertaisuudessaan vastavirta algoritmissa siis lasketaan neuroverkon ennuste, jonka virhettä tiedettyyn tosiulostuloon verrataan, jonka jälkeen painoja muokataan siinä suhteessa, missä ne vaikuttavat syntyvään virheeseen. Kokonaiskeskineliövirhe osajoukolle S voidaan laskea seuraavasti

$$E = \frac{1}{2}\sum_{p \in S} E_p = \frac{1}{2N}\sum ||\hat{y}_p - y_p||^2, \tag{3.14}$$

jossa N kuvaa käytettävän osajoukossa olevien näytteiden määrää. Näin ollen yksittäisen opetusnäytteen virhe E_p voidaan laskea seuraavasti

$$E_p = \frac{1}{2}||\hat{y}_p - y_p||^2 = \frac{1}{2}e_p^T e_p, \tag{3.15}$$

$$e_p = \hat{y}_p - y_p \tag{3.16}$$

Kun kaikilla neuroverkon kerroksilla olevat painot $W^{(m-1)}$, jossa $m = 2, ..., M$ yhdistetään yhdeksi matriiksiksi saadaan neuroverkon parametrimatriisi $\boldsymbol{W}$. Kun virhefunktio osittaisderivoidaan tämän neuroverkon parametrimatriisin $\boldsymbol{W}$ suhteen saadaan suunta, johon neuroverkon parametrejä tulee muuttaa, jotta virhefunktion arvo pienenee kyseisen sijainnin suhteen nopeinten. Koska käytettävät optimointialgoritmit ovat iteratiivisia, käytetään myös askelkokoa μ, joka varmistaa, että jokaisella iteroinnin kierroksella siirrytään pieni askel kohti virhefunktion minimiä.[9, s.85-87] Yhtälömuodossa tämä voidaan kirjoittaa seuraavasti

$$\Delta_p \boldsymbol{W} = -\mu \frac{\partial E_p}{\partial \boldsymbol{W}} \tag{3.17}$$

Tarvittavat osittaisderivaatat ratkaistaan käyttämällä ketjusääntöä. Neuroverkon parametrimatriisista $\boldsymbol{W}$, on haettavissa kerroksen m neuronin i ja seuraavan kerroksen neuronin j välinen painokerroin $w^m_{(i,j)}$. Yksittäisestä harjoitusdataparista syntyvän virheen gradientti, joka on siis Jacobin matriisi, voidaan kirjoittaa muodossa $\frac{\mathrm{d}E_p}{\mathrm{d}\boldsymbol{W}}$. Seuraavissa yhtälöissä käytetttävä merkintä $net^{(m)}_{p,v}$ kuvaa neuroverkon m:nen kerroksen neuronin v ulostuloa näytteellä p. Merkinnällä $\Theta^{(m)}_v$ kuvataan kerroksen m neuronin v bias-termiä. Kerros m sisältää J_m neuronia. Ketjusääntöä käyttämällä, yksittäisen painon derivaatta voidaan laskea seuraavasti

$$\frac{\partial E_p}{\partial w^{(m)}_{(u,v)}} = \frac{\partial E_p}{\partial net^{(m+1)}_{(p,v)}} \frac{\partial net^{(m+1)}_{(p,v)}}{\partial w^m_{(u,v)}} \tag{3.18}$$

Kun yhtälön 3.18 toisessa osassa oleva verkon ulostulo kerroksella $m+1$, korvataan sitä kuvaavalla yhtälöllä 3.5 voidaan osittaisderivaatta kirjoittaa seuraavassa muodossa

$$\frac{\partial net^{m+1}_{(p,v)}}{\partial w^{(m)}_{(u,v)}} = \frac{\partial}{\partial w^{(m)}_{(u,v)}} \sum_{\omega=1}^{J_m} (w^{(m)}_{(\omega,v)} o^{(m)}_{(p,\omega)}) + \Theta^{(m+1)}_v = o^{(m)}_{(p,u)}. \tag{3.19}$$

Yhtälön 3.18 ensimmäinen osa voidaan myös ratkaista käyttämällä ketjusääntöä sekä aikaisemmin määriteltyä yhtälöä 3.6. Sen yhtälö muotoutuu seuraavanlaiseksi

$$\frac{\partial E_p}{\partial net^{(m+1)}_{(p,v)}} = \frac{\partial E_p}{\partial o^{(m+1)}_{(p,v)}} \frac{\partial o^{(m+1)}_{(p,v)}}{\partial net^{(m+1)}_{(p,v)}} = \frac{\partial E_p}{\partial o^{(m+1)}_{(p,v)}} \dot{\phi}^{(m+1)}(net^{(m+1)}_{p,v}). \tag{3.20}$$

Kyseinen yhtälössä 3.20 ratkaistu gradientti riippuu kerroksen ulostulosta, jonka takia piilokerroksille $m = 1, ..., M-2$ ja ulostulokerrokselle tarvitaan erilliset määrittelyt. Ulostulokerroksella oleville neuroneille kyseinen gradientti on yksinkertaisesti neuronin tuottama virhe eli se voidaan merkitä seuraavasti

$$\frac{\partial E_p}{\partial o^{(m+1)}_{p,v}} = e_{p,v}, m = M-1. \tag{3.21}$$

Muilla kerroksilla yhtälö on muotoa

$$\frac{\partial E_p}{\partial o^{m+1}_{(p,v)}} = \sum_{\omega=1}^{J_{m+2}} \frac{\partial E_p}{\partial net^{(m+2)}_{(p,\omega)}} w^{(m+1)}_{(v,\omega)}, m = 1, ..., M-2. \tag{3.22}$$

Delta-funktioksi määritellään

$$\delta^{(m)}_{(p,v)} = -\frac{\partial E_p}{\partial net^{(m)}_{(p,v)}}, m = 2, ..., M. \tag{3.23}$$

Kun tämän jälkeen yhtälöiden 3.21 ja 3.22 yhtälöt sijoitetaan yhtälöön 3.20 voidaan deltafunktioiden yhtälöt ulostulokerrokselle sekä piilokerrokselle kirjoittaa seuraavissa muo-

doissa

$$\delta_{(p,v)}^{(M)} = -e_{(p,v)}\dot{\phi}_v^{(M)}(net_{(p,v)}^{(M)}), m = M-1 \tag{3.24}$$

$$\delta_{(p,v)}^{(m+1)} = \dot{\phi}_v^{m+1}(net_{(p,v)}^{(m+1)}) \sum_{\omega=1}^{J_{m+2}} \delta_{(p,\omega)}^{(m+2)} w_{(v,\omega)}^{(m+1)}, m = 1..., M-2. \tag{3.25}$$

Yhtälöiden 3.24 ja 3.25 avulla delta-funktioiden arvot on määritettävissä jokaiselle verkon kerrokselle rekursiivisesti. Näin ollen verkon painomatriisin $\boldsymbol{W}$ arvoja voidaan päivittää yksi kerrallaan käyttämällä seuraavaa yksinkertaista yhtälöä

$$\frac{\partial E_p}{\partial w_{(u,v)}^{(m)}} = -\delta_{(p,v)}^{(m+1)} o_{(p,u)}^{(m)}. \tag{3.26}$$

Neuronien sisältämät biastermit voidaan päivittää samalla kerralla sisällyttämällä ne verkon parametrimatriisiin sekä kerroksien ulostulovektoreihin asianmukaisesti. Ne on kuitenkin mahdollista optimoida myös erikseen vaihtamalla painokertoimista koostuva matriisi $\boldsymbol{W}$ biastermeistä koostuvaan matriisiin $\boldsymbol{\Theta}$. [9, s.85-87]

3.8 Gradienttimenetelmät

Neuroverkkojen oppiminen tapahtuu gradienttimenetelmän avulla. Gradienttimenetelmä on iteratiivinen prosessi, jossa virhefunktion arvoa minimoidaan askeleittain. Tarkoituksena on löytää virhefunktion globaali minimi. Tämä ei kuitenkaan aina ole mahdollista ja virhefunktion muodosta sekä käytetystä optimointimenetelmästä riippuen algoritmi voi konvergoitua lokaaliin minimiin. Menetelmässä lasketaan virhefunktion gradientti neuroverkon sisäisten parametrien eli painojen w ja bias termien b suhteen. Merkitään näitä parametrejä seuraavaksi merkillä Θ. Yksinkertaisimmassa erägradienttimenetelmässä (engl. Batch gradient descent) lasketaan keskimääräinen gradientti koko harjoitusdatasetille, jonka jälkeen parametrien arvoa päivitetään gradientin vastaiseen suuntaan. Tämä voidaan kirjoittaa yhtälömuodossa

$$\Theta = \Theta - \eta \nabla \Theta E(\Theta). \tag{3.27}$$

Oppimisnopeus η on parametri, joka vaikuttaa algoritmin konvergoitumisnopeuteen sekä ylipäätään konvergoitumiseen. Oikean oppimisnopeuden valinta on haastava tehtävä, sillä liian pienillä arvoilla algoritmi konvergoituu hitaasti ja liian suurilla arvoilla virhefunktio voi heittelehtiä ja aiheuttaa virhefunktion hajaantumisen.[25]

Stokastisessa gradienttimenetelmässä SGD (engl. Stochastic Gradient Descent) neuroverkon painoja sekä bias termejä päivitetään jokaisen harjoitusdatasta valitun esimerkin

jälkeen. Tämä voidaan kirjoittaa yhtälömuodossa

$$\Theta = \Theta - \eta \nabla \Theta E(\Theta, x^{(i)}, y^{(i)}). \tag{3.28}$$

Tämä nopeuttaa algoritmia tapauksissa, joissa harjoitusdatasetti sisältää useita samankaltaisia näytteitä, mutta kasvattaa myös virhefunktion heilahtelua, sillä painoja päivitetään tiheämmin. Virhefunktion suurempi heilahtelu ja varianssi mahdollistavat myös joissain tapauksessa paremman lokaalin minimin löytämisen. Tämä perustuu mahdollisuuteen poistua tasomaisen virhefunktion mahdollisesti sisältämistä uomista. Stokastinen gradienttimenetelmä ei takaa konvergoitumista globaaliin tai lokaaliin virhefunktion minimiin, mutta käytännössä hitaalla oppiminopeuden pienentämisellä on havaittu päästävän vastaaviin tuloksiin kuin erägradienttimenetelmässä. [25]

Nykyään yleisimmin käytössä olevat gradienttimenetelmät perustuvat pienempiä eriä käyttävään ”mini-batch gradient descent”-algoritmiin, joka on yhdistelmä aikaisemmin esitellyistä erägradienttimenetelmästä sekä stokastisesta gradienttimenetelmästä. Siinä päivitysten tekemiseen käytetään useammasta kuin yhdestä harjoitusnäytteestä laskettua gradienttia ja kuitenkin pienempää osajoukkoa kuin erägradienttimenetelmästä. Tämän takia virhefunktion varianssi on pienempi kuin stokastisessa menetelmässä ja konvergoituminen on nopeampaa kuin erägradienttimenetelmässä. Se voidaan kirjoittaa yhtälömuodossa seuraavasti

$$\Theta = \Theta - \eta \nabla \Theta E(\Theta, x^{(i:i+n)}, y^{(i:i+n)}), \tag{3.29}$$

jossa n kuvaa käytettävän osajoukon kokoa. Yleisesti käytettävät osajoukkojen koot ovat välillä 50–256. [25]

Näissä kaikissa perinteisissä gradienttimenetelmissä on kuitenkin ongelmansa, joiden takia niistä on kehitetty erilaisia optimoituja versioita. Näitä ongelmia on esimerkiksi oppimisnopeuden valinta, jolla on suuria vaikutuksia algoritmin nopeuteen sekä ylipäätään konvergoitumiseen. Tätä varten on kehitetty menetelmiä, jotka muuttavat oppimisnopeutta ennalta määritetyn taulukon mukaan optimointiaskeleiden välissä. Tätä kutsutaan oppimisnopeuden jaksotukseksi (engl. Learning rate scheduling), mutta sen ongelmana on, ettei oppimisnopeus mukaudu datasetin perusteella. Lisäksi virhefunktiot voivat olla muodoltaan hyvin monimutkaisia, sisältää monia lokaaleja minimejä sekä useita satulapisteitä, joka vaikeuttaa globaalin minimin löytämistä.[25] Yksi näistä optimoiduista gradienttimenetelmistä on ADAM-optimisaattori (engl. Adaptive Moment Estimation), jota käsitellään luvussa 3.9.

Momenttitermin lisääminen gradienttimenetelmään tiedetään nopeuttavan sen konvergoitumista. Momenttitermin lisääminen stokastisen gradienttimenetelmän tapauksessa voidaan tehdä lisäämällä yhtälöön 3.28 momenttia kuvaava termi, jolloin sen painojen päivi-

tys saa seuraavan muodon

$$\Theta = \Theta_{t-1} - \eta \nabla \Theta E(\Theta, x^{(i)}, y^{(i)}) + p \nabla \Theta_{t-1}, \tag{3.30}$$

jossa p kuvaa momenttikerrointa. Momenttitermin tuomia hyötyjä voidaan ajatella esimerkiksi tilanteessa, jossa tasomainen virhefunktio sisältää kapean mutta pitkän laakson. Tällöin vakio-oppimiskertoimella oleva menetelmä, ilman momenttia saattaa heilahdella laakson reunoilla ilman että se koskaan konvergoituu laakson pohjalle. Tällaisessa tilanteessa momentin käyttö keskiarvottaa laakson reunojen välillä olevan oskiloinnin ja vähentää laakson läpi kulkemiseen kuluvien askelien määrää summaamalla peräkkäisiä samansuuntaisia gradienttikomponentteja [22].

3.9 ADAM

Adam on syväoppimisessa yleisesti käytetty optimointialgoritmi, jonka tarkoituksena on suorittaa adaptiivista oppimisnopeuden muutosta datan ominaispiirteiden perusteella sekä yhdistää gradientteihin momenttia kuvaava komponentin, joka mahdollistaa paremman optimointituloksen kompleksisen ja satulapisteitä sisältävien virhefunktioiden kohdalla. Sen toimintaa voidaan kuvata yhtälömuodossa seuraavasti. Se toimii laskemalla eksponentiaalista pienenemistä noudattavat harhattomat liukuvat keskiarvot gradienteille. Nämä kuvaavat gradienttien odotusarvoa ja varianssia. Virhefunktion gradientille askeleella t käytetään tässä merkkiä g_t ja se voidaan laskea seuraavasti

$$g_t = \nabla_\Theta E_t(\Theta_{t-1}) \tag{3.31}$$

Odotusarvoa vastaava liukuva keskiarvo voidaan kirjoittaa yhtälömuodossa seuraavasti

$$\hat{m_t} = \frac{\beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t}{(1 - \beta_1^t)} \tag{3.32}$$

Varianssia vastaava liukuva keskiarvo lasketaan seuraavasti

$$\hat{v_t} = \frac{\beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2}{(1 - \beta_2^t)} \tag{3.33}$$

Näistä voidaan laskea päivitetyt painot

$$\Theta_t = \Theta_{t-1} - \alpha \cdot \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon} \tag{3.34}$$

Algoritmi tarvitsee toimiakseen hyperparametrit β_1, β_2, ϵ ja α. Näistä α kuvaa algoritmin maksimiaskelkokoa, joka skaalataan aikaisempien gradienttien mukaan, jotta saavutetaan dataan perustuva adaptiivinen askelkoko. Hyperparametrit β_1 ja $\beta_2 \in [0, 1)$, ovat liukuvien keskiarvojen eksponentiaaliset hajoamiskertoimet. Lisäksi algoritmiin kuuluu hy-

perparametri ϵ, jonka tarkoituksena on estää mahdollinen nollalla jakaminen ja sen arvoksi on algoritmin julkaisussa asetettu arvo $\epsilon = 10^{-8}$. Koska algoritmi on rekursiivinen, vektoreille $\hat{m}_t$ ja $\hat{v}_t$ asettaa arvot ajanhetkelle $t = 0$. Julkaisussa molemmille vektoreille on valittu alkuarvoiksi nollavektorit ja niiden aiheuttamat harhat estimaatteihin on poistettu jakamalla harhaiset estimaatit termeillä $(1 - \beta_1^t)$ ja $(1 - \beta_2^t)$.[17]

Adam on pienellä muistintarpeella oleva laskennallisesti tehokas stokastinen optimointialgoritmi. Se sisältää vain muutaman säädettävän hyperparametrin, jotka tyypillisesti vaativat vain pieniä muutoksia. Näiden lisäksi sen on havaittu toimivan empiirisissä tutkimuksissa hyvin verrattuna muihin vastaavanlaisiin algoritmeihin verrattuna monenlaisella datalla.[17] Nämä ominaisuudet tekevät siitä helppokäyttöisen, tehokkaan ja varsin robustin optimointialgoritmin, jota on hyödynnetty paljon osana syväoppimisongelmien ratkaisuja.

4. SEURANTA-ALGORITMIN TOTEUTUS

4.1 Yleistä

Yleisesti käytössä olevat seuranta-algoritmit käyttävät Bayesialaista estimointia, jossa kohteen todellista tilaa estimoidaan virhettä sisältävien havaintojen perusteella. Algoritmit koostuvat yleisesti kahdesta osasta: kohteen sijainnin ennustus käyttämällä edellistä kohteen tilaa koskevaa estimaattia ja sen epävarmuutta, ja päivityksestä, jossa lasketaan uusi kohteen tilaestimaatti ennusteen ja sen epävarmuuden sekä kohteesta saadun havainnon avulla. Koneoppimista käytettäessä seuranta on mahdollista toteuttaa seuraamatta tarkasti Bayesialaisen estimoinnin periaatteita. Tällaisia lähestymistapoja on esitelty esimerkiksi ”Online multi-target intelligent tracking using a deep long-short term memory network”-julkaisuissa [29] ja ”Maneuvering Target Tracking with Recurrent Neural Networks for Radar Application”-julkaisussa [11], joita tullaan hyödyntämään tässä työssä. Lisäksi lentokoneen seuraavan sijainnin ennustamista, jonka voidaan ajatella olevan merkittävä yhden kohteen seurantaa, on tutkittu LSTM-neuroverkkoja hyödyntäen esimerkiksi ”LSTM-based Flight Trajectory Prediction”-julkaisussa [27]. Kaikki tässä työssä käytettävät koordinaatit ovat ECEF-muotoisia, ellei toisin mainita. Tämä valinta tehtiin koska järjestelmän alkuperäisenä tarkoituksena oli toimia myös monen tutkan tapauksessa. Yhteisen koordinaatiston käyttö prosessoitaessa monen tutkan havaintoja keskitetysti on ainoa mielekäs tapa suorittaa seurantaa. Tietysti mikä tahansa muukin tutkien välinen yhteinen koordinaatisto olisi mahdollinen, mutta maakeskeinen ECEF-koordinaatisto on luonnollinen ja yksinkertainen valinta.

Kohteen seurantaa voidaan ajatella myös vain kohteesta saapuvien havaintojen suodattamisena siten, että mittalaitteistosta ja ympäristöstä peräisin olevan häiriön vaikutus saadaan poistettua havainnosta. Tämä tarkottaisi, että suodatetut havainnot kuvaisivat kohteen tilaa täydellisesti mittaushetkinä. Kyseisen suotimen tulisi siis pystyä approksimoimaan todennäköisintä kohteen liikemallia peräkkäisten havaintojen perusteella. Tämä lähestymistapa vaatii kuitenkin sen, että kohteeseen liittyvät havainnot tulee pystyä assosioimaan oikealle kohteelle häiriöhavaintojen (engl. False plot) seasta. Tähän tarvitaan erillinen ennustusalgoritmi, jonka tarkoituksena on arvioida todennäköisintä kohteen seuraavaa tilaa aikaisempien kohteesta tulleiden suodatettujen havaintojen perusteella. Mikäli tutkan mittauksiin aiheuttama häiriö olisi puhtaasti normaalijakautunutta ja koh-

teen tilanmuutokseen liittyvät epävarmuudet olisivat myös normaalijakautuneita, Kalmansuodin olisi todistetusti optimaalinen tähän häiriön poistoon. Todellisuudessa tutkahavainnon sisältämä häiriö on kuitenkin tutkakohtaista ja sen jakauman muoto poikkeaa normaalijakaumasta.

Seuraavaksi esitellään työssä toteutetun seuranta-algoritmin rakenne ja luomiseen vaaditut vaiheet. Ensimmäisenä käydään läpi käytettävä simulaatiomalli ja koulutus- sekä evaluointimateriaalinen luominen sitä hyödyntäen. Kyseistä simulaatiomallia ei luotu tätä työtä tehtäessä vaan se oli valmis kokonaisuus, joka saatiin käyttöön työn suorittamista varten. Sen toimintaa kuvataan luvussa 4.2.

Koulutus- ja evaluointimateriaalin luomiseeen ja käsittelyyn liittyvät vaiheet esitellään luvuissa 4.2.1, 4.2.2, 4.2.3 ja 4.2.4. Tämän jälkeen luvussa 4.3 luodaan neuroverkkopohjainen seurantasuodin, joka koostuu ennustusmallista sekä suodatusmallista. Luvussa 4.4 esitellään työssä seurantojen hallinnointiin liittyvä logiikka. Viimeisenä seuranta-algoritmin osana esitellään työssä käytettävät havaintojen assosiointimenetelmät luvussa 4.5. Näistä esitellään vakioikkunainen assosiointi 4.5.1, vakiosäteinen assosiointi ennustusta käyttämällä 4.5.2, muuttuvaikkunainen assosiointi 4.5.3 sekä autoenkooderiin perustuva assosiointi 4.5.4. Tämän jälkeen luvussa 4.6 kuvataan kokonaisen seuranta-algoritmin toimintaa havainnon saapumisesta uuden seurannan luomiseen tai vanhan seurannan päivittämiseen.

Luvussa 5 vertaillaan käytettyjen assosiointimenetelmien tuottamia tuloksia seuranta-algoritmissa. Vakiosäteistä assosisiointia ei oteta mukaan näihin vertailuihin, sillä se lisättiin työhön vain huolehtimaan seurantojen alussa tapahtuvista assosioinneista. Tämä johtuu siitä, että työssä luotu seurantasuodin sekä muut esitellyt assosiointimetodit eivät pysty aloittamaan seurantaa, vaan vaativat toimiakseen seurantoja joihin on liitetty jo useampi havainto. Luvussa myös pohditaan tuloksien merkitystä, seuranta-algoritmiin liittyviä rajoitteita sekä mahdollisia jatkotutkimusaiheita.

4.2 Lentodatan generointi

Koneoppimista käyttävien mallien koulutukseen tarvitaan aina sovellusalueeseen liittyvää dataa. Koska tässä työssä on tarkoitus rakentaa malli, joka pystyy seuraamaan ilmakohteita tutkahavaintojen avulla, tarvitaan siis tutkahavaintoja ja kohteiden todellisia sijainteja kyseisillä hetkillä. Tässä työssä rajoitutaan kuitenkin tarkastelemaan vain kiinteäsiipisten ilmakohteiden toimintaa eli esimerkiksi helikoptereita ja droneja ei niiden hyvin erillaisten lento-ominaisuuksien vuoksi huomioida. Koska todellisten tutkahavaintojen ja koneiden oikeiden lokaatioiden saaminen tutkimuskäyttöön on hyvin haastavaa tietoturva- ja turvallisuussyistä, päädyimme tässä työssä käyttämään valmista simulaatiomallia datan generointiin. Kyseistä simulaatiomallia ei siis kehitetty tätä tutkimusta varten, vaan se saatiin tutkimuksen aiheen antaneelta yritykseltä käyttöön työn suorittamista varten.

Työn kannalta käytettävä simulaatiomalli sisältää kaksi oleellista toiminnallisuutta. Se pystyy simuloimaan kohtuullisella tarkkuudella erillaisten lentokoneiden liikehdintää monissa erillaisissa skenaarioissa ja simuloimaan erilaisten tutkien toimintaa ja niiden saamia havaintoja näissä tilanteissa. Simulaattorille syötettävät skenaariot voivat sisältää useita eri reiteillä lentäviä lentokoneita ja useita eri lokaatioissa olevia tutkia. Yksinkertaisuuden vuoksi työssä kuitenkin päädyttiin käyttämään vain skenaarioita, jotka sisälsivät yhden lentävän kohteen ja yhden tutkan.

Käydään seuraavaksi läpi tarkemmin tutkasimulaation toimintaperiaatteet. Simulaatiossa käytettäville lentokoneille on mahdollista asettaa RCS eli tutkapoikkipinta-alatiedot. Näin ollen tutkan vastaanottamien heijastusten voimakkuus riippuu lentokoneen asennosta, sijainnista tutkan suhteen ja käytettävästä taajuudesta. Simulaatiomalli tuottaa havaintoja sellaisista kohteista, joista saapuvan signaalin signaali-kohinasuhde on tarpeeksi suuri. Signaali-kohinasuhde, jolla tutka luo havainnon, riippuu tutkalle asetetusta havainnon luomistodennäköisyydestä (eng. Probability of detection) sekä asetetusta väärien havaintojen esiintymistiheydestä (engl. False alarm rate). Tutkille on myös asetettu virherajat korkeuskulma, atsimuuttikulman ja etäisyyden mittauksessa, joiden sisältä luotuun havaintoon lisätään normaalijakautunutta kohinaa. Näiden huomiointi aiheuttaa sen, että riippuen koneen asennosta ja etäisyydestä, se voi hävitä tutkan näkymättömiin pitkäksikin aikaa. Tämä aiheuttaa suuria ongelmia seurannoissa, sillä mikäli kohteesta ei saavu havaintoja useaan mittauskierrokseen, se yleisesti poistetaan seurattavien kohteiden listalta. Tässä työssä tähän ongelmaan ei kuitenkaan pureuduta tarkemmin, vaan tällaiset perinteisillekkin tutkaseuranta-algoritmeille vaikeat tilanteet sivuutetaan testauksessa.

Simulaatioon kuuluvan lentosimulaation avulla lentokoneille voidaan siis luoda erillaisia reittejä koordinaattien ja näihin liitettävien tavoitenopeuksien avulla. Tämän jälkeen malli toteuttaa lentoradan, jos se on kyseiseen lentokoneeseen liitettyjen lennettävyysominaisuuksien rajoissa mahdollista. Mikäli lentorata ei ole sellaisenaan kyseisellä lentokoneella lennettävissä, malli toteuttaa lentoradan mahdollisimman lähelle tavoitetta, mutta rikkomatta lentokoneen lentomallia. Näin ollen simulaatiomallille voidaan syöttää satunnaisesti myös sellaisia reittejä, joita simulointiin valittu lentokone ei todellisuudessa pysty lentämään aiheuttamatta lentodataan sellaisia lentoratoja, jotka eivät todellisuudessa ole mahdollisia. Kyseinen ominaisuus on työn kannalta hyödyllinen, sillä koneoppimismallin harjoitusdatan tulisi optimitilanteessa sisältää kaikki erilaiset mahdolliset lentoradat.

Kaikkien mahdollisten lentoratojen sisällyttäminen harjoitusdataan ei kuitenkaan käytännössä ole mahdollista, joten lentoratoja päädyttiin luomaan satunnaisuutta hyödyntäen sellainen määrä, jolla lentoratojen simulointiin kuluva aika pysyisi työn aikataulun kannalta järkevänä. Lopullisessa työssä käytettävien lentoratojen simulointi kesti noin 48 tuntia. Lentoradat päädyttiin luomaan kahta erilaista strategiaa käyttämällä. Molemmat strategiat perustuvat siihen, että kaikki reitin sisältämät pisteet ovat sellaisen suorakulmaisen särmiön sisällä, jonka korkeus on tutkan maksimi korkeuskantama R_{aMax} ja pohjan reu-

nojen pituus voidaan laskea seuraavasti $\frac{2}{\sqrt{2}}R_{max}$, jossa R_{max} on käytettävän tutkan maksimikantama. Tämän jälkeen lentoradat keskitetään siten, että suorakulmaisen särmiön pohjan keskipiste on tutkan ECEF-koordinaatissa. Kyseinen valinta aiheuttaa sen, että simuloitavat lentoradat eivät ole jakautuneet koko tutkan kantama-alueelle tasaisesti, vaan tutkan kantama-alueella on joitain alueita joita harjoitusdata ei kata. Tämä valinta luonnollisesti huonontaa järjestelmän suorituskykyä kyseisillä alueilla. Tämä ongelma olisi korjattavissa kohtuullisen yksinkertaisesti parantamalla lentoratojen generointiin käytettävää algoritmiä, mutta se sivuutettiin tässä työssä, koska se ei ole työn ydinasian kannalta oleellista. On myös oleellista huomioida, että seuranta-algoritmin koneoppimismallit koulutetaan lyhyillä reitin paloilla, sillä pitkällä aikavälillä olevien riippuvuuksien opettaminen takaisinkytketyille neuroverkoille on haastavaa. Koneoppimismallit siis tarkastelevat ajallisesti vain hyvin lyhyitä riippuvuuksia lentokoneen liikehdinnässä. Tämä tarkoittaa, että generoitavien lentoratojen tulee sisältää hyvin nopeita käännöksiä ja liikkeitä, jotta mallit voivat oppia nämä riippuvuudet.

4.2.1 Lentoreittien luominen

Ensimmäisen strategian tarkoituksena on muodostaa reitti satunnaisten reittipisteiden välillä käyttämällä constant jerk-mallia. "Jerk" eli nykäisy on kiihtyvyyden derivaatta ja täten kyseinen malli luo reitin siten että pisteiden välillä kohteen kiihtyvyys on vakio. Jotta kyseistä mallia voidaan käyttää, tulee jokaiselle satunnaisesti valitulle pisteelle määrittää tavoitenopeus, jota malli käyttää reittivälin aloitusnopeutena. Kyseinen nopeus määritellään tässä työssä yksinkertaisesti siten, että suunta on kohti seuraavaa satunnaisesti valittua pistettä kohti ja vauhti valitaan satunnaisesti koneelle sallitulta väliltä. Jotta reittien pituudet pysyvät järkevinä käytetään jokaisen reitin luomiseen vain muutamia satunnaisesti valittuja pisteitä. Tämän jälkeen valittujen pisteiden väliltä lasketaan lisää reittipisteitä käyttämällä stonesoup-kirjaston constant jerk-mallia [19]. Näin ollen saadaan reittejä, joissa satunnaisesti valittujen pisteiden välillä kohde liikkuu vakiokiihtyvyydellä.

Toinen strategia on luoda reitit käyttämällä tietynlaista random-walk-tyyppistä ratkaisua. Lentoradan alkupiste valitaan ensimmäisen strategian tapaan tutkan kantama-alueelta, jonka jälkeen valitaan satunnainen kääntymisnopeus ja kääntymisaika tietyltä etukäteen valitulta arvoväliltä. Myös lentonopeus ja lentokorkeus valitaan satunnaisesti ennalta määritellyltä väliltä ja näitä pidetään vakiona koko reitin ajan. Yksinkertaisuuden vuoksi kaikki simuloitavat käännökset tapahtuivat vakiokorkeudella. Reittipisteet lasketaan käyttämällä käännösnopeutta, käännösaikaa ja lentonopeutta. Jokaisen reittipisteen jälkeen arvotaan uusi käännösnopeus ja käännösaika. Reitille arvotaan alussa myös satunnainen kokonaispituus väliltä 300–500 km. Reitin luominen päättyy mikäli se menee tutkan kantama-alueen ulkpuolelle tai reitin pituus ylittää valitun kokonaispituuden. Erillaisille lentokoneille on mahdollista valita erillaiset arvovälit niiden lento-ominaisuuksien perusteella. Tämä

on tarpeellista, koska erilaisilla lentokoneilla kuten matkustajalentokoneilla, pienlentokoneilla ja hävittäjillä on hyvin erillaiset lento-ominaisuudet ja seuranta-algoritmin tulisi kuitenkin pystyä seuraamaan kaiken tyyppisiä lentokoneita. Yksinkertaisuuden vuoksi kaikki luotavat reitit tullaan kuitenkin simuloimaan vain yhtä lentokonetyyppiä hyödyntäen, jotta tarvittavan datan määrä pysyy maltillisena.

Työssä käytettävä simulaatiomalli ei mahdollista satunnaisten reittien luomista automaattisesti, vaan se tarvitsee tavoitereitin simulointia varten. Tässä luvussa esiteltyjä strategioita hyödynnetään luomaan näitä tavoitereittejä simulaation käytettäväksi.

4.2.2 Tutkahavainnon muuttaminen ECEF-koordinaatistoon

Tutkasimulaation tuottama data on Asterix-standardin mukaista. Havainnot vastaanotetaan Asterix CAT048-sanomalla ja tutkainformaatio Asterix CAT034-sanomalla, joka vastaa tutkan pohjoissanomaa. Työtä varten nämä sanomat kirjoitetaan tiedostoon, jonka jälkeen niistä erotellaan työn kannalta oleellinen informaatio. Havaintoja sisältävistä sanomista työhön tarvitaan havainnon aika TOD (engl. Time Of Day), joka muutetaan Unix-aikaleimaksi, jotta myös vuorokauden vaihtuminen seurannan aikana on mahdollista. Lisäksi tarvitaan tutkan tunniste ja havainnon sijainti. Havainnon sijainti ilmoitetaan viistoetäisyyden, pohjoiskulman ja korkeuden avulla, jotta havainnon sijainti ECEF-koordinaatistossa on laskettavissa tarvitaan myös havainneen tutkan sijainti. Tutkien sijainnit saadaan tutkien ilmoittamista pohjoissanomista.

Havainnon ECEF-koordinaatin laskenta viistoetäisyyden ψ, pohjoiskulman Θ, havaitun korkeuden h ja havaitsijan sijainnin perusteella suoritetaan geometrisesti. Laskentaan tarvittavat yhtälöt on esitetty dokumentissa [5]. Selvyyden vuoksi seuraavaksi kuitenkin esitetään tarvittavat yhtälöt. Pohjoissanomasta saapuvat tutkan koordinaatit on ilmoitettu maantieteellisinä koordinaatteina. Sen latituudi koordinaattia merkitään merkillä Lat_R, longituudi koordinaattia merkillä Lon_R ja korkeutta maanpinnasta merkillä h_R. Maan säde tutkan latituudi koordinaatissa voidaan laskea seuraavaa yhtälöä käyttämällä

$$R = \frac{a(1-e^2)}{\sqrt{(1-e^2\sin^2(Lat_R))^3}}, \tag{4.1}$$

jossa a vastaa maan ellipsoidin puoliakselin pituutta ja e vastaa ellipsoidin eksentrisyyttä. Tästä saadaan laskettua R_r, joka vastaa maan sädettä tutkan kohdalla. Mikäli oletetaan, että maan säde on havainnon kohdalla sama kuin tutkan kohdalla, voidaan havainnon korkeuskulma Ψ laskea seuraavasta yhtälöstä

$$\sin\Psi = \frac{2R_r(h-h_r)+h^2-h_r^2-\psi^2}{2\psi(R_r+h_r)}. \tag{4.2}$$

Koska tutka voi havaita kohteita monen sadan kilometrinkin päästä, tällöin maan säde on

todellisuudessa eri tutkan ja havaitun kohteen kohdalla, joilloin laskennasta aiheutuu systemaattista virhettä [5]. Tätä virhettä ei kuitenkaan tässä työssä eksplitiittisesti huomioida, vaan oletetaan koneoppimismallin oppivan kyseisen virhelähteen koulutusmateriaalista. Näitä arvoja käyttämällä havainnon koordinaatit voidaan muuttaa tutkakeskeisestä pallokoordinaatiosta tutkakeskeiseen karteesiseen koordinaatistoon seuraavilla yhtälöillä

$$\begin{aligned} x &= \psi \cos(\Psi) \sin(\theta) \\ y &= \psi \cos(\Psi) \cos(\theta) \\ z &= \psi \sin(\Psi) \end{aligned} \tag{4.3}$$

Tutkakeskeisen karteesisen koordinaatiston z-akseli on maanpintaan suhteen normaali, y-akseli on pohjoiseen osoittava tangentti ja x-akseli on näiden suhteen normaali siten, että syntyy kohtisuorassa oleva oikeakätinen koordinaatisto. Muunnos maakeskeiseen koordinaatistoon tapahtuu siis laskemalla tutkan sijainti maakeskeisessä koordinaatistossa ja lisäämällä siihen tutkakeskeisessä koordinaatistossa lasketut havainnon koordinaatit siten, että akselit on käännetty vastaamaan maakeskeisen koordinaatiston akseleita. Tätä varten ensin täytyy laskea tutkan sijainti maakeskeisessä koordinaatistossa T_r seuraavalla yhtälöllä

$$\begin{aligned} T_x &= (\mu_1 + h_r) \cos(Lat_r) \cos(Lon_r) \\ T_y &= (\mu_1 + h_r) \cos(Lat_r) \sin(Lon_r) \\ T_z &= (\mu_1(1 - e^2) + h_r) \sin(Lat_r). \end{aligned} \tag{4.4}$$

Näitä käyttämällä koostetaan $T_r = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$, jossa $\mu_1 = \frac{a}{\sqrt{1-e^2 \sin^2(Lat_r)}}$ Näin ollen havainnon koordinaatit muutetaan maakeskeiseen ECEF-koordinaatistoon seuraavalla yhtälöllä

$$\begin{bmatrix} X_{Ecef} \\ Y_{Ecef} \\ X_{Ecef} \end{bmatrix} = S_1^T \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T \tag{4.5}$$

, jossa

$$S_1 = \begin{bmatrix} -\sin(Lon_r) & \cos(Lon_r) & 0 \\ -\sin(Lat_r)\cos(Lon_r) & -\sin(Lat_r)\sin(Lon_r) & \cos(Lat_r) \\ \cos(Lat_r)\cos(Lon_r) & \cos(Lat_r)\sin(Lon_r) & \sin(Lat_r) \end{bmatrix} \tag{4.6}$$

4.2.3 Koulutusmateriaalin luominen

Kaikki työhön generoitu data on luotu siten, että simuloidaan ainoastaan yhden tutkan ja yhden kohteen tapauksia. Simulaattori tuottaa lennetystä reitistä kolme tiedostoa. Ensimmäinen tiedosto sisältää kaikkien tutkalta saapuneiden havaintojen Asterix CAT048-sanomaan kuuluvat tiedot. Lisäksi tiedostoon on lisätty jokaiselle havainnolle tieto siitä, onko se peräisin oikeasta kohteesta. Toinen sisältää tutkan kaikkien Asterix CAT034-sanomien tiedot. Koska kaikki työssä käytettävät tutkat ovat mittausten ajan paikallaan, työtä varten tarvitaan oikeastaan ainoastaan ensimmäisen sanoman tiedot. Kolmas tiedosto sisältää kohteen oikeat paikkatiedot tutkan mittaushetkillä. Havainnon oikeallisuudesta kertovaa informaatiota käytetään ainoastaan mallien harjoitusdatan generointiin. On kuitenkin hyvä huomata, että tämän informaation saaminen on mahdollista saada ainoastaan simulaatiomalleista, eikä myöhemmissä kappaleissa esiteltäviä suodatus- tai autoenkooderimallia näin ollen pystytä kouluttamaan todellisilta tutkilta saaduilla havainnoilla. Tämä johtuu siitä, ettei tutkalta saatuja havaintoja voida koskaan varmuudella yhdistää tiettyyn kohteeseen. Myöhemmässä luvussa esitellään myös ennustusverkko, jonka kouluttamiseen tarvitaan ainoastaan kohteen todellisia sijainteja. Näin ollen se ainakin teoreettisesti voidaan kouluttaa käyttämällä lentokoneen GPS-järjestelmästä saatua dataa. Todellisuudessa tämäkin tietty sisältää häiriötä ja epävarmuutta jotka voivat vaikuttaa ennustusmallin kouluttamiseen.

Koneoppimismallien kouluttamista varten luodaan harjoitus- ja testidatasetti käyttämällä 4.2.1 esiteltyjä strategioita. Molemmilla strategioilla luodaan 150 reittiä, jolloin harjoitusdata koostuu yhteensä 300 reitistä. Tämän jälkeen nämä reittiskenaariot ajetaan simulaatiomallilla, jolloin saadaan yhteensä 300 edellisessä kappaaleessa esitettyä kolmen tiedoston rykelmää. Työssä sovellettavien koneoppimismallien tavoitteena on oppia lentokoneiden liikehdinnästä sekä tutkan aiheuttaman häiriön muodosta. Näin ollen malleja on mielekästä opettaa vain oikeista kohteista saapuvilla havainnoilla. Tästä syystä harjoitusdatasettiin liitettävät tiedostot esikäsitellään siten, että havainnoista säilytetään vain oikeilta kohteilta saapuneet havainnot. Tämän jälkeen reitit katkotaan lyhyempiin reitinpalasiin, siten että peräkkäisten havaintojen aikaero on maksimissaan 18 sekuntia. Tämä valinta perustuu siihen että työssä käytettävän simulaation tutkan pyörimisnopeudeksi on valittu 10 RPM ja seurantojen hallinta on toteutettu siten, että vahvistetun seurannan seuranta lopetetaan mikäli siihen ei ole liitetty havaintoa noin kolmeen tutkan pyörähdykseen. Tällä varmistetaan, ettei harjoitus- tai testidataan synny sellaisia aikasarjoja, jotka rikkoisivat luvussa 4.2.4 esitettyjä skaalauksessa käytettäviä sääntöjä. Näistä lyhyemmistä reitinpaloista erotetaan satunnaisesti 5%, joita tullaan käyttämään luotujen mallien erilliseen evaluointiin.

Pitkät lentoreitit sisältävät kuitenkin kokonaisuutena huomattavasti enemmän suoraan lentämistä kuin käännöksiä. Tämä havaittiin erilaisia koulutettuja malleja evaluoitaessa,

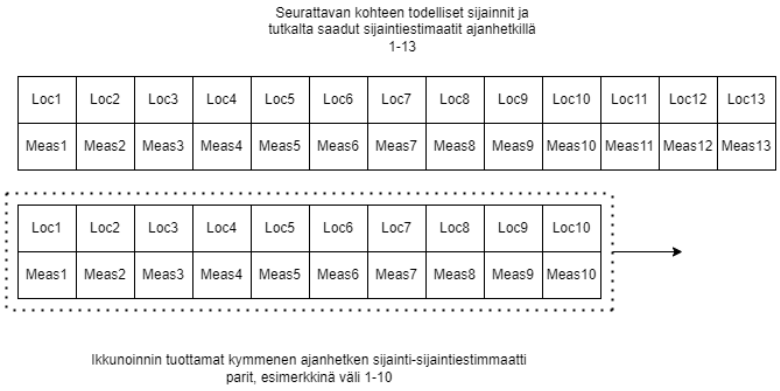
jonka takia datasetti päädyttiin tasapainottamaan ennen koulutusta, siten että lopullinen harjoitusdata sisältää saman määrän lähes suoraan lentäviä reitinpaloja sekä käännöksiä sisältäviä reitinpaloja. Jokaiselle havaintosarjalle lasketaan reitin lineaarisuutta kuvaava arvo käyttämällä kyseisiin havaintoihin liittyviä todellisia reittipisteitä. Laskennassa tulee käyttää havaintoihin liittyviä todellisia reittipisteitä, sillä muuten arvon merkitys muuttuisi havaintoihin liittyneen satunnaisen kohinan takia. Todellisia reittipisteitä sisältäville sarjoille lineaarisuutta kuvaava arvo voidaan laskea suoraan. Reitin lineaarisuutta kuvaavaksi arvoksi valikoitui yksinkertainen reitin kokonaispituus kaikkien reittipisteiden kautta suhteessa reitin alku- ja loppupisteiden väliseen etäisyyteen. Tämä voidaan kirjoittaa yhtälömuodossa

$$L_{linearity} = \frac{\|c_k - c_0\|}{\sum_{i=1}^{k} \|c_i - c_{i-1}\|}, \tag{4.7}$$

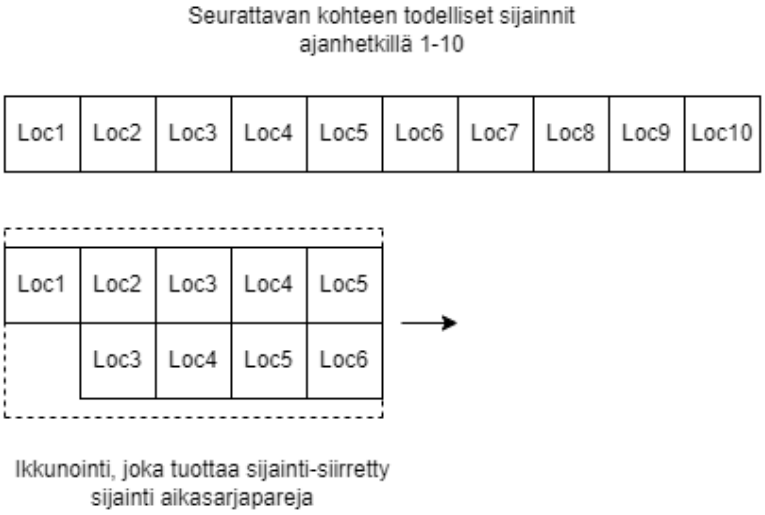
jossa vektori $c_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}$ sisältää todellisen reittipisteen sijainnin ECEF-koordinaatistossa. Kaikki kyseisen lineaarisuutta kuvaavan arvon $L_{linearity}$ ovat luonteensa vuoksi arvovälillä [0,1]. Lähellä 0 olevat arvot tarkoittavat hyvin mutkittelevia reittejä ja lähellä 1 olevat arvot lähes lineaarisia reittejä. Kaikille suodatusverkon koulutukseen ja ennustusverkon koulutukseen tarkoitetulle materiaaleille lasketaan lineaarisuuskertoimet erikseen. Tämän jälkeen arvoväli jaetaan tulosten perusteella 100 tasaväliseen väliin ja lasketaan esiintymistaajuudet. Kaikkien materiaalien tapauksessa suurin osa arvoista on kolmessa viimeisessä arvovälissä, jonka takia lasketaan kuinka monen sarjan lineaarisuus on ensimmäisessä 97 arvovälissä ja viimeisistä kolmesta arvovälistä valitaan mukaan satunnaisesti vain yhtä monta sarjaa.

Esikäsittelyn jälkeen jokainen havaintosarjan palanen ajetaan 10 havainnon pituisen liukuvan ikkunoinnin läpi, jolloin havaintosarjoista saadaan useita suodatusmallille sopivia osittain päällekäisiä sarjoja. Sama prosessi suoritetaan myös kohteen oikeaa liikettä kuvaaville tiedostoille, jolloin saadaan 10 pituisia kohteen oikeita lokaatioita kuvaavia sarjoja. Havaintosarjat ja kohteen lokaatiota kuvaavat sarjat yhdenmukaistetaan ajan perusteella, jolloin saadaan havaintosarja-sijaintisarja-pareja, jotka voidaaan syöttää suodatusmalliin. Tästä syntyy 83309 havaintosarja-sijaintisarja-paria. Kun näistä sarjoista poistetaan lineaarisia tapauksia, jotta datasetistä saadaan tasapainoisempi, jäljelle jää 39682 sarjaa.

Ennustumallin koulutusta varten tarvitaan kohteen oikeita sijainteja sisältäviä aikasarjoja, sekä niihin liittyviä yhdellä aikavälillä eteenpäin siirrettyjä sijainteja. Nämä saadaan aikaan ajamalla jokainen esikäsittelystä saatu kohteen todellisia sijainteja kuvaava sarja viiden pituisen ikkunoinnin läpi, siten että samalla luodaan ennustesarjat, jotka ovat yhden aika-askeleen edellä. Kyseisissä aikasarjapareissa siirretyt aikasarjat ovat pituudeltaan yhden lyhyempiä kuin alkuperäiset aikasarjat, sillä yhden ajanhetken perusteella ennustaminen ei ole millään tavoin kiinnostavaa. Ennustusverkon koulutuksessa käytettävät sisäänmenovektorit ovat siis pituudeltaan 5 ajanhetkeä ja ulostulovektorit 4 ajanhetkeä. Tästä syntyy yhteensä 93675 aikasarja-paria. Kun näistä poistetaan ylimääräiset li-



Kuva 4.1. *Esimerkki havaintosarja-sijaintisarja-parejen luomisesta*



Kuva 4.2. *Havainnekuva harjoitusdatan tuottamisesta ennustusmallille*

neaariset tapaukset, jotta datasetistä saadaan tasapainoisempi, jäljelle jää 50730 sarjaa. Evaluointiin varatuille reitinpaloille suoritetaan muuten samat esiprosessointiin kuuluvat vaiheet kuin muillekkin reitinpaloille, mutta niistä saatavia datasettejä ei linearisoida eikä skaalata. Näitä käytetään ennustus- ja suodatusmallien erilliseen evaluointiin, jossa virheet lasketaan normaalissa skaalassa. Tätä evaluointiin käytettävää datasettiä ei linearisoida, jotta saadaan selville mallien keskimääräiset virheet, siten että ne ovat painottuneet oikein simulaatioista syntyneiden erillaisten lentoliikkeiden esiintyvyyden perusteella.

Autoenkoodajan koulutukseen tarvittavat havaintosarjat saadaan lähes samalla tavalla kuin suodatusmallillekkin. Ainoa erotus on se, että autoenkoodaajan tapauksessa käytetään liukuvan 10 näytteen ikkunan sijaan vain liukuvaa 5 näytteen ikkunaa. Näin ollen syntyy 96751 sarjaa, joista liiallisten lineaaristen tapauksien poiston jälkeen jää jäljelle 12464 sarjaa, joilla autoenkoodaaja koulutetaan.

Autoenkoodaajaan perustuvan assosiaattorin kouluttamiseen tarvitaan myös toinen harjoitusdatasetti. Tätä varten luodaan luodaan ja simuloidaan 20 reittiä kummallakin luvus-

sa 4.2 esitetyllä menetelmällä. Näistä luotavaa datasettiä tullaan käyttämään oikean rajaarvon löytämiseen, joten sen tulee sisältää myös sellaisia havaintosarjoja, jossa kaikki havainnot eivät ole peräisin kohteesta. Kuitenkin yksinkertaisuuden vuoksi luodaan liukuvan 5 näytteen ikkunalla sellaisia havaintosarjoja, joissa 4 ensimmäistä näytettä on kohteesta ja viidenneksi näytteeksi valitaan kaikki sellaiset havainnot, jotka ovat syntyneet alle 8 sekuntia neljännen näytteen jälkeen ja ovat alle 20 kilometrin päästä siitä. Näin ollen yhdestä tällaisesta ikkunoinnista voi syntyä monta harjoitusnäytettä, joihin kaikkiin liitetään lisäksi tieto ovatko kaikki sarjan havainnot kohteesta syntyneitä vai ei. Näin ollen syntyy harjoitusdatasetti, joka sisältää 5 havainnon sarjoja, jotka on leimattu sen mukaan, onko sarjan 5. havainto oikea kohteesta syntynyt havainto vai harhahavainto.

4.2.4 Koulutusmateriaalin ja vastaanotettujen havaintojen skaalaus

Neuroverkkojen tiedetään suoriutuvan paremmin tilanteista, jossa lähtödata on skaalattu siten, että kaikki sisääntulopiirteet on skaalattu samalle arvovälille. Tästä syystä sekä koulutusmateriaali että seuranta-algoritmille syötettävä sisääntulo skaalataan ennen seurannan tekemistä. Aikasarjat skaalataan siten että ensimmäisen havainnon piirteet vähennetään kaikista kyseisen sarjan piirrevektoreista. Näin ollen sarjojen sijainnit on keskitetty origon ympärille ja sarja siirretty alkamaan ajanhetkestä 0. Kaikki työssä käytettävät lentoreitit on simuloitu ECEF-koordinaatiston positiivisessa kahdeksannessa, jonka takia keskitys origon ympärille toimii edellä kuvatusti. Yleisesti ensimmäisen koordinaatin merkit tulisi kuitenkin huomioida keskitystä tehtäessä.

Koska havaintosarjat ja kohteen liikettä kuvaavat aikasarjat sisältävät eri määrän piirteitä, niille on määritelty erilliset skaalausvektorit. Havaintosarjojen skaalaukseen käytettävä vektori riippuu sarjaan kuuluvien havaintojen lukumäärästä N_u, joka on tässä työssä käytettävän neuroverkon rakenteen vuoksi määritelty vakioksi $N_u = 10$. Havaintosarjojen skaalausvektori on siis muotoa

$$Scale_{measurments} = \begin{bmatrix} \Delta T_{max} N_u & X_{maxDiff} N_u & Y_{maxDiff} N_u & Z_{maxDiff} N_u & R_{max} \end{bmatrix}$$

Koska kohteen sijaintiin liittyviin piirteisiin on sisällytetty myös aikaväli seuraavaan sijaintiin ΔT, täytyy myös se huomioida skaalausvektorissa. Kohteen suodatettuun sijaintiin liittyvä aikasarja skaalataan siis vektorilla, joka on muotoa

$$Scale_{state} = \begin{bmatrix} \Delta T_{max} N_p & \Delta T_{max} & X_{maxDiff} N_p & Y_{maxDiff} N_p & Z_{maxDiff} N_p & R_{max} \end{bmatrix}$$

Joissa $\Delta T_{max} = 18$ s ,$X_{maxDiff} = Y_{maxDiff} = Y_{maxDiff} = 400$ m ja $R_{max} = 400$ km. Kohteen suodatettuun sijaintiin liittyvä aikasarjan skaalausvektori $Scale_{state}$ riippuu aikasarjan pituudesta N_p, joka vastaa aikasarjan pituutta. Kohteen suodatukseen käytettävän aikasarjan pituudeksi on työssä määritelty $N_p = 5$. Skaalauksissa käytetyt arvot on valittu

siten, että ne mahdollistaisivat skaalauksen kaikissa työssä käytettävissä skenaarioissa. Tämä tarkoittaa, sitä että lentokoneiden oletetaan lentävän maksimissaan 400 m/s nopeudella tiettyyn suuntaan. Geometriasta johtuen suuremmatkin nopeudet ovat mahdollisia, mikäli ne eivät ole täysin akseleiden suuntaisia. Simulaatiossa käytettävän tutkan maksimikantamaksi asetettiin 400 km, jota merkittiin skaalausvektoreissa merkillä R_{max}.

4.3 RNN-pohjainen seuranta-algoritmi

RNN-neuroverkkojen kehittyneempiä versioita eli esimerkiksi LSTM-malleja on käytetty aikaisemmissa tutkimuksissa onnistuneesti aikasarjaennustuksessa, ja ne ovat näissä tuottaneet hyviä tuloksia. Tämä perustuu LSTM-verkon rakenteesta johtuvaan kykyyn löytää ja muistaa aikasarjan näytteistä tärkeimmät piirteet ja unohtaa sellaiset piirteet, jotka eivät ole tulevan näytteen kannalta oleellisia. LSTM-mallin esityskykyä on mahdollista kasvattaa käyttämällä useita LSTM-kerroksia päällekkäin, kuten muissakin syvissä neuroverkoissa. Tällöin muodostuu syvä LSTM-verkko, joka pystyy mallintamaan monimutkaisempia piirreyhdistelmiä.

Yleisesti tutkaseurantaa koskevissa tutkimuksissa kohteista saapuvien havaintojen oletetaan saapuvan prosessoitavaksi vakio-aikavälillä, jolloin Many-to-one LSTM verkon voidaan ajatella ennustavan kohteen tilaa seuraavaan sallittuun ajanhetkeen. Tässä työssä käytettävän seuranta-algoritmin on kuitenkin tarkoitus pystyä seuraamaan kohteita myös tapauksissa, jossa havainnot saapuvat muuttuvalla aikavälillä. Tämä on käytännön toteutuksissa lähes pakollista, sillä esimerkiksi nopeasti liikkuvasta kohteesta saapuvat peräkkäiset havainnot eivät saavu tasaisella välillä, koska tutkan pyörimisnopeus on yleisesti vakio. Mikäli käytössä olisi useampia tutkia, näidenkin havainnot saapuisivat muuttuvalla aikavälillä, sillä eri tutkat havaitsevat kohteet eri aikoihin.

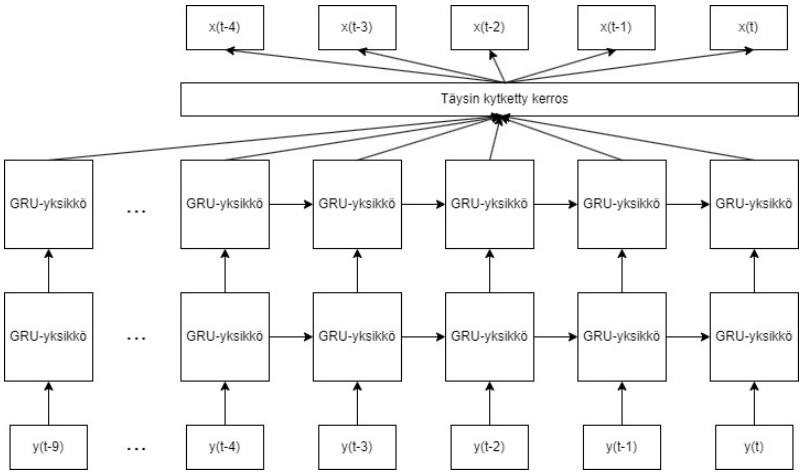
Työssä käytetty GRU-pohjainen seuranta-algoritmi perustuu artikkelissa [29] esiteltyyn ratkaisuun. Kyseisessä artikkelissa esitelty toteutus sopii vain tasaisella aikavälillä saapuville havainnoille, jonka takia tässä työssä havaintojen tilavektoreihin lisätään aikaan liittyviä parametrejä. Näitä käyttämällä malleille yritetään opettaa erilaisia aikariippuvuuksia, jotka mahdollistavat muuttuvan aikavälin seurannan. Lisäksi siinä käytetyt LSTM-verkot korvattiin tässä työssä rakenteeltaan yksinkertaisimmilla GRU-verkoilla, sillä niiden välillä ei havaittu malleja koulutettaessa merkittäviä eroja.

Algoritmi pohjautuu kahteen erilliseen syvään GRU-malliin. Ennustusmalli koostuu kahdesta GRU-kerroksesta ja se vastaanottaa kohteen 5 viimeisintä tilaa, joiden avulla se ennustaa kohteen sijainnin seuraavana ajanhetkenä. GRU kerrosten päällä on yksi täysin kytketty lineaarinen kerros, jonka tarkoituksena on muuttaa GRU-mallin ulostulosta saapuvat arvot väliltä -1 – 1 vapaalle arvovälille. Kohteen tila sisältää ajanhetken T, ajan seuraavaan tiedettyyn kohteen tilaan ΔT, kohteen koordinaatit X, Y, Z ECEF-koordinaatistossa sekä kohteen etäisyyden tutkasta jota merkitään merkillä R. Näistä

koostuu kohteen tilavektori $\begin{bmatrix} T & \Delta T & x & y & z & R \end{bmatrix}$. Aika seuraavaan tiedettyyn kohteen tilaan ΔT mahdollistaa kohteen tilan ennustamisen 0–18 sekuntia tulevaisuuteen. Kyseistä ominaisuutta tarvitaan, sillä käytettävä seurantamalli perustuu siihen, että kohteista saatavat havainnot pystytään assosioimaan oikeille kohteille ja näistä koostuvista havaintosarjoista poistetaan häiriön vaikutus suodatusverkon avulla. Mikäli tutka onnistuu tekemään havainnon kohteesta jokaisella pyörähdyksellään, kohteesta saatavat havainnot saapuvat aina 0-6 s sisään edellisestä havainnosta. Todellisuudessa on kuitenkin myös tilanteita, joissa tutka ei pysty tuottamaan kohteesta havaintoa jokaisella pyörähdyksellään, jolloin kohteesta saapuvien peräkkäisten havaintojen aikaero on yli tutkan pyörähdysajan. Ennustusmallille annettava harjoitusdata kuitenkin sisältää näitäkin tilanteita ja näin ollen sen tulisi onnistua ennustamaan kohteen sijaintia myös näissä tilanteissa.

Suodatusmalli sisältää myös 2 kerroksisen syvän GRU-verkon, jonka päällä on yksi täysin kytketty lineaarinen kerros, kuten ennustumallissakin. Sen sisääntulo sisältää kymmenen viimeisintä kohteeseen assosioitua havaintoa ja sen ulostulo sisältää viisi viimeisintä havaintoa suodatettuna. Tutkalta saapuvat havainnot ovat vektoreita, jotka sisältävät ajanhetken jolloin havainto on tehty T, havainnon koordinaatit ECEF-koordinaatistossa X, Y,Z ja havainnon etäisyyden tutkasta R. Kyseinen vektori on siis muotoa $\begin{bmatrix} T & x & y & z & R \end{bmatrix}$. Kohteen tilan päivittämiseen käytetään kuitenkin vain viimeistä suodatettua havaintoa, joka vastaa kohteen viimeisintä tilaa.

Molemmat mallit kuitenkin tarvitsevat seurannan historiaan liittyviä tietoja toimiakseen. Ennustusmalli tarvitsee viisi edellistä kohteen suodatettua tilaa ja suodatusmalli 10 viimeisintä kohteeseen liittyvää havaintoa. Näin ollen tarvitaan erillinen tapa suorittaa seurantaa tilanteissa, joissa nämä ehdot eivät täyty. Näissä tilanteissa työssä päädyttiin hyödyntämään artikkelissa [29] esiteltyjä menettelyjä. Tilannetta, jossa seurantaan liitettyjen havaintojen määrä oli välillä $2 - N_p$ artikkelissa ennustusmallin sisääntulo täydennettiin niin, että ensimmäistä arvoa kopioitiin sarjan alkuun tarvittava lukumäärä. [29]. Kuitenkin tilanteessa, jossa seurantaan on vasta alkanut ja näin ollen sisältää vain yhden havainnon, ei ennustamista voida tehdä. Tällöin havainon assosiointi suoritetaan käyttämällä yksinkertaista vakiosäteistä assosiointi-ikkunaa suoraan havainnoille. Suodatusmalli tarvitsee toimiakseen N_u havaintoa. Tilanteissa joissa seurantaan on liitetty alle N_u kappaletta havaintoja päädyttiin käyttämään menettelyä, jossa lasketaan seurannan ennustetun tilan ja uuden havainnon sijaintien sekä tutkasta olevan etäisyyden keskiarvo ja käytetään tätä seurannan uutena tilana. Kyseinen menettely vastaa artikkelissa [29] käytettyä menettelyä sillä erotuksella, että siihen on lisätty työssä käytettävän tilavektori sisältää myös kohteen etäisyyden tutkasta.



Kuva 4.3. *Suodatusmallin yksinkertaistettu rakenne*

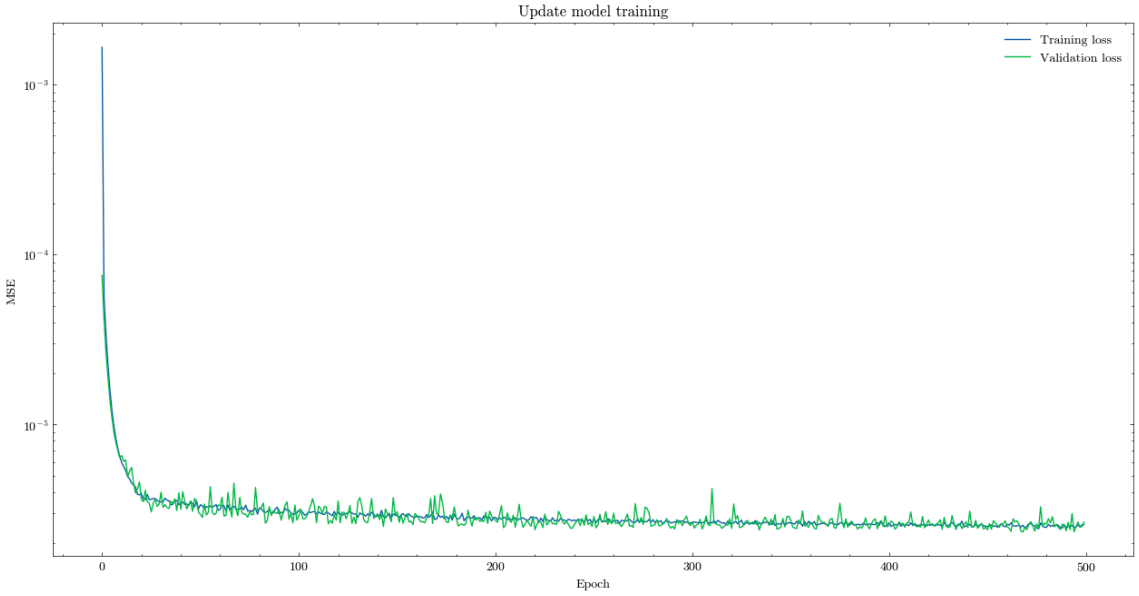
4.3.1 Suodatusmallin rakenne ja koulutus

Suodatusmalli koostuu siis kahdesta päällekkäisestä GRU-kerroksesta. Ensimmäisellä kerroksella on 8 neuronia ja toisella kerroksella 16 neuronia. GRU-kerroksissa aktivaatiofunktiona käytettiin Tanh-funktiota ja verkon ajallisesti peräkkäin olevien komponenttien välillä Sigmoid-funktiota. Nämä ovat myös toteutuksessa käytetyn Keras-kirjaston vakioarvot kyseiselle verkkorakenteelle. GRU-kerrosten päällä on yksi täysin kytketty kerros, jossa on 25 neuronia. Nämä arvot kuvaavat viittä viimeisintä verkkoon syötettyä havaintoa suodatettuna, kun saatu vektori muotoillaan 5x5 matriisiksi. Verkon koulutuksessa käytettävä virhefunktio on keskineliövirhe MSE, joka soveltuu hyvin regressio-ongelmiin. Optimointi suoritetaan käyttämällä Adam-optimisaattoria.

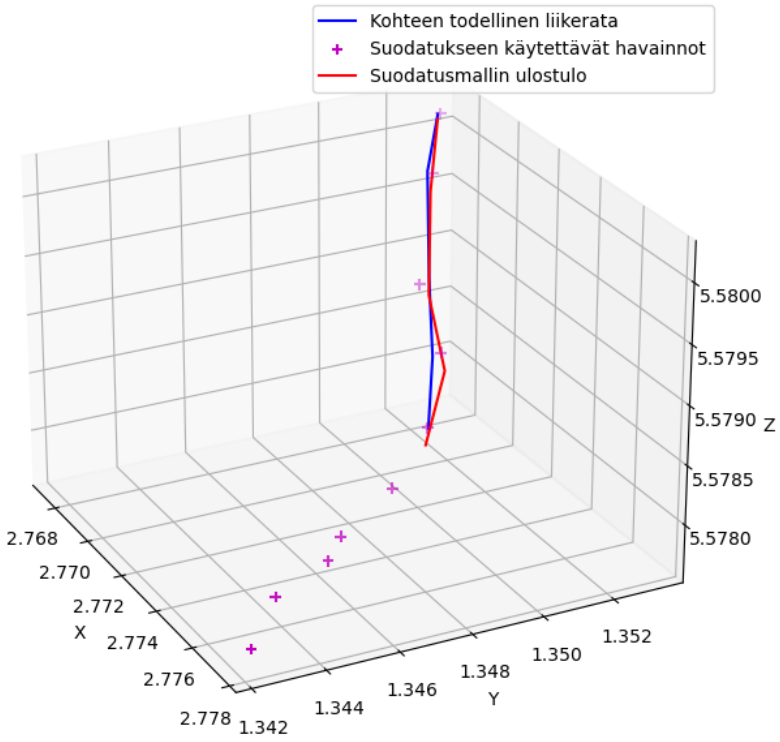
Harjoitusdata näytetään verkolle koulutuksen aikana yhteensä 500 kertaa ja optimoinnin nopeuttamiseksi käytetään 128 näytteen jaottelua. Tämän jälkeen suodatusmalli evaluoitiin käyttämällä harjoitusdatasta aikaisemmin eroteltua 5% osaa, jota malli ei koulutuksen, eikä sen aikaisen evaluoinnin aikana nähnyt. Arvot skaalattiin käyttämällä 4.2.4 kappaleessa esiteltyä skaalausta. Kun jokaiselle ajanhetkelle laskettiin kuinka paljon tutkalta saatu mittaustulos eroaa kohteen todellisesta sijainnista, ja nämä keskiarvotettiin saatiin keskimääräiseksi mittausvirheeksi 218,8 metriä. Suodatuskerrokselta saatava ulostulo erosi kohteen todellisesta sijainnista keskimäärin 225,1 metriä. Suodatusverkolta saatava erillinen etäisyysestimaatti erosi todellisesta etäisyydestä keskimäärin 107,2 metriä.

4.3.2 Ennustusmallin rakenne ja koulutus

Myös ennustusmalli koostuu kahdesta GRU-kerroksesta ja yhdestä täysin kytketetystä kerroksesta. Verkon sisääntulo on viiden aikasemman ajanhetken tilat ja ulostulona neljä kohteen tilaa. Näistä viimeinen on verkon ennuste tulevalle ajanhetkelle, jota käy-

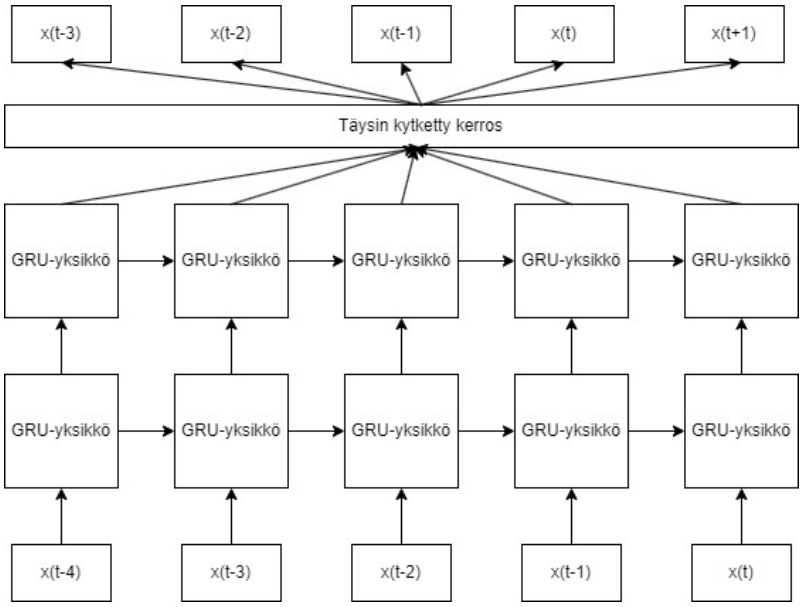


Kuva 4.4. *Suodatusmallin virheen kehittyminen mallia koulutettaessa*

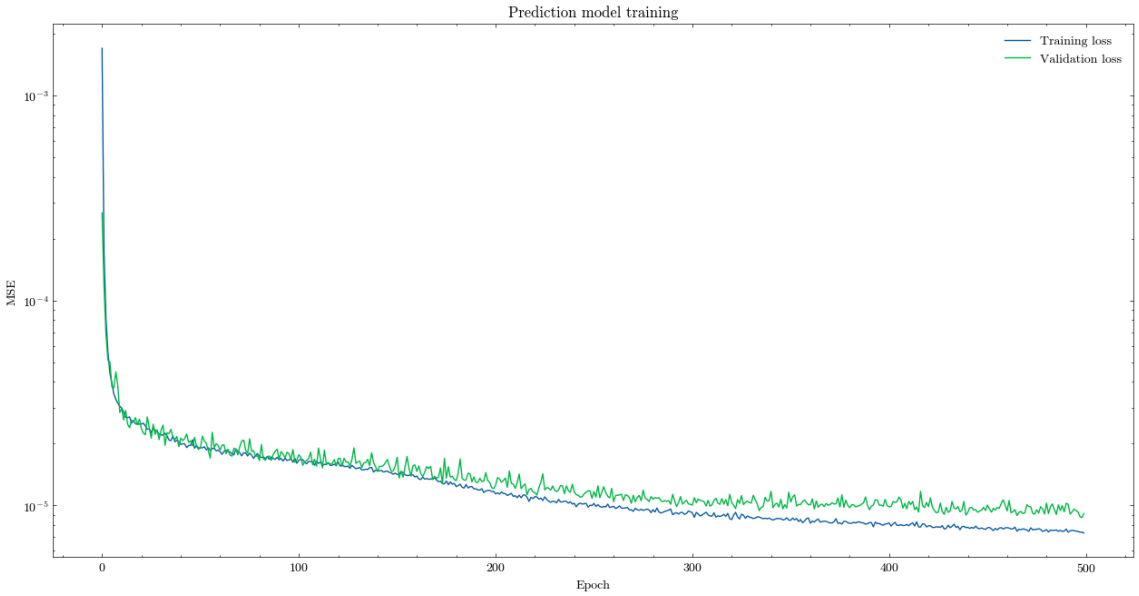


Kuva 4.5. *Esimerkkikuva suodatusmallin evaluoinnista*

tetään kohteen tilan ennusteena. Kahdella ensimmäisellä GRU-kerroksella on 32 neuronia kullakin. Täysin kytketty kerros sisältää viisi neuronia, jotka muodostavat ennusteen toisen GRU-kerroksen ulostulojen perusteella. Kuvassa 4.6 on esitetty mallin yksinkertaistettu rakenne. Viimeisellä kerroksella aktivointifunktiona käytetään lineaarista aktivointia. Virhefunktiona käytetään keskineliövirhettä MSE ja verkon optimointiin ADAM-optimisaattoria.

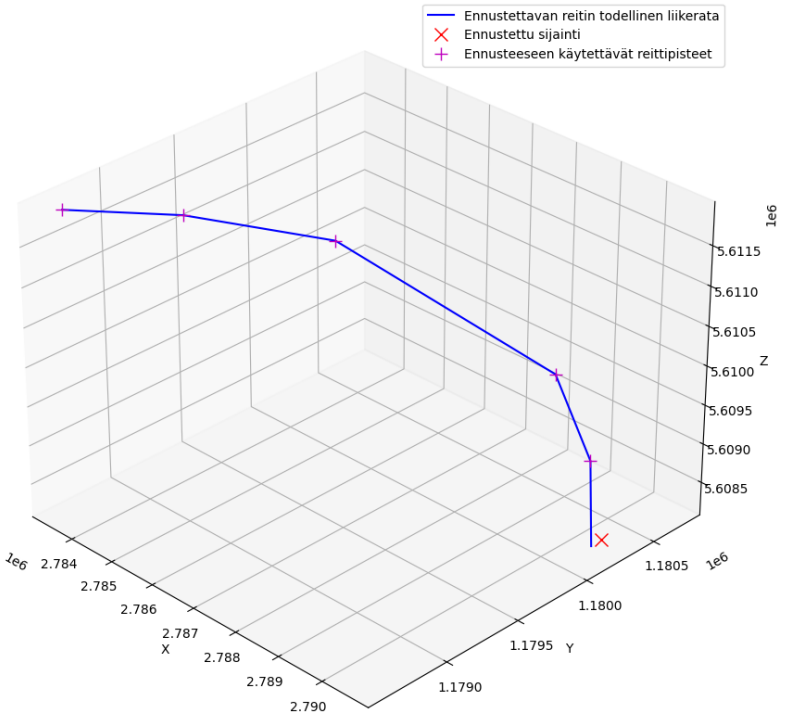


Kuva 4.6. *Ennustusmallin yksinkertaistettu rakenne*



Kuva 4.7. *Ennustusmallin virheen kehittyminen koulutettaessa*

Harjoitusdata ajetaan verkon läpi koulutuksessa yhteensä 500 kertaa pienemmissä 128 näytteen seteissä, joka nopeuttaa verkon kouluttamista. Tämän jälkeen ennustusmalli evaluoitiin käyttämällä erillistä pienempää evaluointisettiä, joka sisälsi harjoitusdatasta eroteltua 5% pienemmistä reitinpaloista. Nämä olivat siis sellaisia reitinpaloja, joita malli ei ollut ennen nähnyt. Kun reitinpaloille laskettiin ennusteet käyttämällä yksinkertaista lineaarista approksimaatiota niiden keskivirheeksi saatiin 478.8 metriä. Kun samat reitinpalat skaalattiin ja syötettiin ennustusmalliin, jonka jälkeen ulostulo skaalattiin takaisin alkuperäiseen mittakaavaan, saatiin ennusteiden keskivirheeksi 182 metriä. Erillisen etäisyysapproksimaation virhe neuroverkolta oli 317.1 metriä. Näissä laskennoissa käytettiin ainoastaan verkon viimeistä ulostuloa, joka kuvaa ennustetta seuraavalle ajanhetkelle.

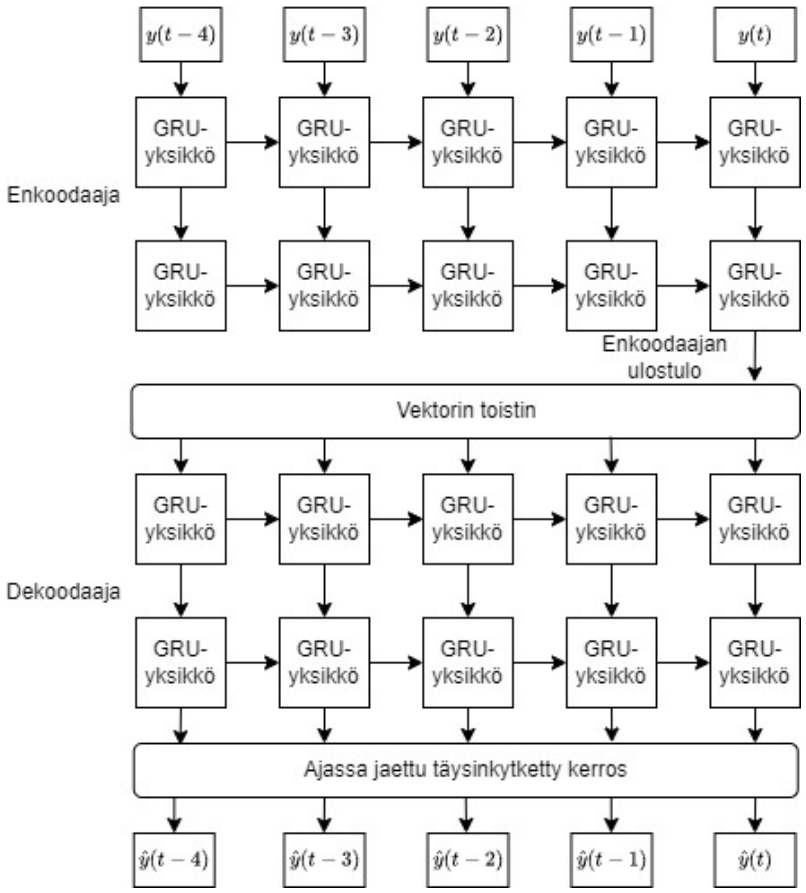


Kuva 4.8. *Esimerkkikuva ennustusmallin evaluoinnista*

4.3.3 Autoenkoodajamallin rakenne ja kouluttaminen

Yleisesti autoenkoodaaja-tyyppisten neuroverkkojen voidaan ajatella etsivän sille annetusta koulutusdatasta mahdollisimman tehokas ja kuitenkin tarpeeksi kuvaava koodaus. Käytännössä se tekee kontekstisidonnaista häviöllistä pakkausta, jossa se hyödyntää koulutusdatan tyypille luonnollisia riippuvaisuuksia ja piirteitä. Käytännössä tämä tapahtuu luomalla enkoodaaja- ja dekoodaajamallit, jotka yhdistetään yhtenäiseksi autoenkoodaajaksi. Yleisesti neuroverkkoja koulutettaessa koulutusdata koostuu näytepareista (X,Y), jossa neuroverkolle annettava lähtöarvo X on eri kuin tavoitearvo Y. Autoenkooderimallit kuitenkin koulutetaan näytepareilla (X,X), jossa lähtöarvo on sama kuin tavoitearvo. Enkooodaajan on siis tarkoitus löytää painokertoimet, joiden avulla se voi tuottaa tietyn tyyppisestä datasta pienemmän piirrevektorin, joka sisältää kyseiselle datatyypille oleellisimmat piirteet. Dekoodaaja taas etsii sellaiset painokertoimet, jotka mahdollistavat palauttamaan pienemmästä piirrevektorista virhefunktion kannalta lähinnä alkuperäistä arvoa olevan vektorin.

Mallin koulutukseen käytetty data on 4.2.3 luvussa esitettyjä 5 havainnon pituisia skaalattuja havaintosarjoja. Näin ollen verkolle annettava yksi näyte on 5x5 matriisi. Työssä käytetään GRU-pohjaista autoenkooderimallia, joka vastaanottaa näitä viiden mittaisia havaintosarjoja. Enkoodaaja koostuu kahdesta päällekkäisestä kerroksesta, jossa ensimmäisellä on 12 neuronia ja toisella 10 neuronia. Enkoodaja antaa ulostulona toisen kerroksen viimeisen GRU-solun ulostulon eli vektorin joka sisältää 10 elementtiä. Tämä on



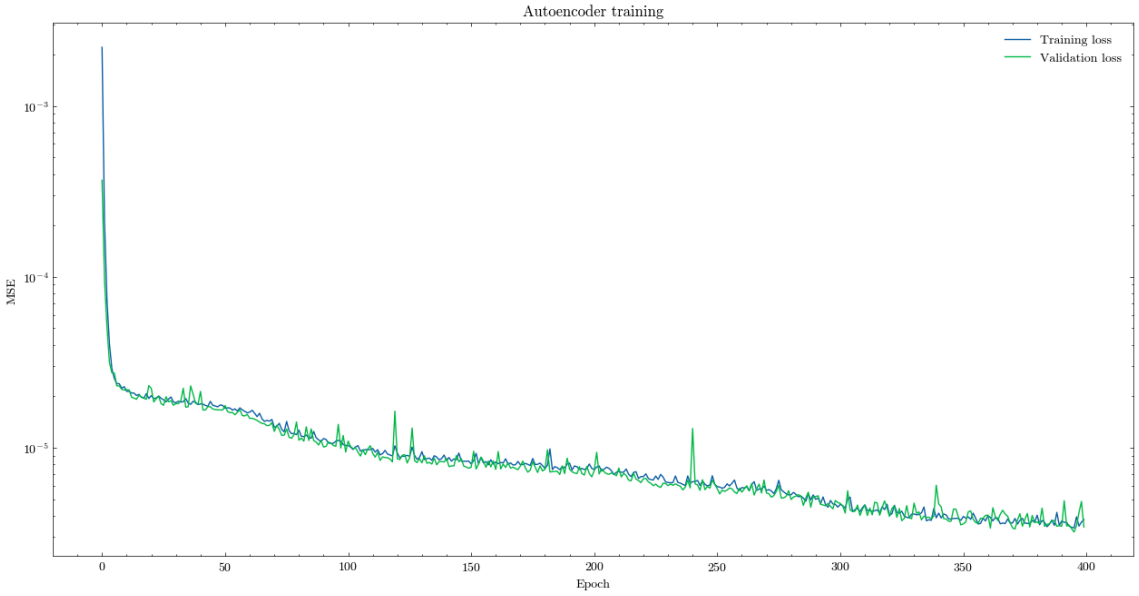
Kuva 4.9. *Autoenkoodajamallin yksinkertaistettu rakenne*

autoenkooderissa käytettävä piirrevektori, jonka koko on siis 2.5 kertaa pienempi kuin sisääntulovektorin. Myös dekoodaaja koostuu kahdesta GRU-kerroksesta, jossa ensimmäisellä on 12 neuronia ja toisella 10 neuronia. Koska myös dekoodaaja on takaisinkytketty neuroverkko, tulee enkoodaajalta saatu piirrevektori kopioida jokaiseen GRU-soluun. Koska dekoodaajan toiselta kerrokselta saatava ulostulo on muotoa 5x10, täytyy jokaisen takaisinkytketyn GRU-solun perään lisätä 5 neuronin täysin kytketty kerros lineaarisilla aktivointifunktioilla, jotta saadaan sisäänmenoa vastaava 5x5 ulostulo.

Kuvassa 4.9 on kyseisen mallin yksinkertaistettu rakenne. Malli siis vastaanottaa havaintosarjan y ja antaa ulostulona enkoodauksen ja dekoodauksen johdosta aproksimaation $\hat{y}$. Verkon kouluttamiseen käytetään muiden työssä esiteltyjen verkkojen tapaan keskineliövirhettä ja Adam-optimisaattoria. Harjoitusdata myös näytetään verkolle vastaavasti 400 kertaa käyttämällä 64 näytteen jaottelua.

4.3.4 Neuroverkkojen rakenteen optimointi

Neuroverkkojen tyypiksi valikoitui takaisinkytkeyyu neuroverkko, sillä työssä haluttiin tutkia kuinka kirjallisuudesta löytyvien tutkaseurantaan käytettävät neuroverkkoarkkitehtuurit toimisivat tutkasimulaatioympäristöön luodulla tutkaseuranta-algoritmilla. Mallien kou-



Kuva 4.10. *Autoenkoodaajamallin virheen kehittyminen koulutettaessa*

lutusvaiheessa testattiin myös eteenpäin kytkettyjä neuroverkkoja, mutta niiltä saadut evaluointitulokset olivat huonompia kuin valikoituneelta mallilta saadut, jonka takia niiden tutkimisesta luovuttiin alkumetreillä ja päädyttiin keskittymään pelkästään takaisinkytkettyihin neuroverkkoihin.

Käytettävien aikasarjojen pituudet pohjautuivat artikkelissa [29] esiteltyihin arvoihin. Niitä kuitenkin muokattiin työhön sopivimmiksi artikkelissa käytetyn ja tässä työssä käytettävän tutkasimulaation näytteistystaajuuksien eron takia. Aikasarjojen todellista ajallista kestoa ei voida myöskään liikaa kasvattaa, koska sillä on suora vaikutus neuroverkkopohjaisen seurannan käynnistymisnopeuteen. Tämä johtuu siitä että mallit tarvitsevat tietyn määrän seurantaan liitettyjä havaintoja tai seurannan tiloja ennen kuin ne toimivat.

Malleissa käytettäviin aktivointifunktioihin päädyttiin myös testaamalla erillaisia vaihtoehtoja ja seuraamalla niiden vaikutusta evaluointituloksiin. Viimeisen täysinkytketyn kerroksen aktivointifunktiolla oli suurimmat vaikutukset, sillä minkä tahansa muun kuin lineaarisen aktivoinnin käyttö aiheutti ongelmia toteutetuissa malleissa. Tämä johtuu todennäköisesti siitä, että muut testatut aktivointifunktiot rajoittavat ulostulonsa tietylle arvovälille ja tällöin myös koko verkon ulostulo on rajoittunut tälle arvovälille. Työssä käytettävän ennustusmallin kohdalla tämä tarkoittaisi sitä, että verkolta saatavat ennusteet olisivat rajautuneet skaalausvektorin määrittelemälle arvovälille, joka ei ennusteen kannalta ole toivottavaa.

Käytettävien neuronien sekä piilokerroksien määrää vaihdeltiin etsittäessä soveltuvinta mallia. Suuremmilla ja pienemmillä neuronimäärillä mallien evaluoinnista saatavat tulokset, joko huononivat tai pysyivät samoina, jolloin päädyttiin valitsemaan nämä työssä käytettävät neuronien lukumäärät siten, että ne antoivat parhaan evaluointituloksen pienim-

mällä mahdollisella neuroneiden lukumäärällä. Kyseinen valinta tehtiin sillä suuremman verkon käyttäminen aiheuttaa todennäköisemmin ylisovittamista.

4.4 Seurantojen hallinnointi

Seurantojen hallinnointi tehdään aikaperusteisesti käyttämällä erilaisia seurannan tasoja. Tasoja ovat seurannan alku, jota kuvataan painolla 0,4, seurantakandidaatti, jota kuvataan painolla 0,8 ja vahvistettu seuranta, jota kuvataan painolla 1,0. Seurannan painoa kasvatetaan aina 0,2 yksikköä mikäli siihen assosioituu havainto. Mikäli seurantaan ei ole assosioitunut havaintoa eikä sen painoa ole päivitetty viimeisimmän 12 sekunnin aikana sen painoa pienennetään 0,2 yksikköä. Mikäli seurannan painoa pienennetään tilanteessa jossa se on 0,8 yksikköä, se poistetaan kokonaan seurattavien kohteiden listalta.

Tämän tyyppinen hallinnointi mahdollistaa sen, ettei yksittäinen oikeasta kohteesta puuttuva havainto poista seurantaa. Samalla se pitää myös huolen siitä, että häiriöhavainnoista syntyvät seurannan alut poistetaan nopeasti seurattavien kohteiden listalta. Tämä on oleellista, koska jokainen tutkalta tuleva havainto, joka ei assosioidu mihinkään olemassa olevaan seurantaan, aiheuttaa uuden seurannan alun syntymisen, mikä aiheuttaa lisää laskentakuormaa seuranta-algoritmille.

4.5 Assosiaatiomenetelmät

Työssä aikaisemmin esitellyn neuroverkkoratkaisun suodatusmallin on tarkoitus suodattaa häiriöitä kohteelta saapuvista havainnoista ja ennustusmallin ennustaa niiden avulla kohteen todennäköisintä sijaintia vapaavalintaisella ajanhetkellä. Ennustumallia käytetään siis approksimoimaan kohteen sijaintia jollain ajanhetkellä, jotta alueelta saapunut havainto pystytään assosioimaan oikealle seurannalle. Tämän lisäksi sitä hyödynnetään kohteen tilan päivityksessä, sellaisissa seurannan tilanteissa, joissa seurantaan ei ole liitetty tarpeeksi havaintoja suodatusmallin käyttämiseksi.

Tutkaseurannan suorituskyky on hyvin riippuvainen näistä oikeista assosioinneista, joten työssä päädyttiin etsimään parasta assosiointitekniikkaa kyseiselle neuroverkkoihin perustuvalle rakenteelle. Kalman-suotimiin perustuvissa seuranta-algoritmeissa havainnon assosiointiin on yksinkertaisimmillaan käytetty ennusteen epävarmuuden avulla muodostettua ellipsoidin muotoista assosiointi-ikkunaa. Koska neuroverkot eivät luonnostaan tuota ennusteellensa epävarmuuslukuja, tätä lähestymistapaa ei voida kuitenkaan suoraan soveltaa tässä työssä. Monen kohteen seurantaan liittyvässä kirjallisuudessa havaintojen assosiointi suoritetaan yleisesti kaikille samaan aikaan saapuville havainnoille samanaikaisesti, valitsemalla parhaat assosiaatiot käyttämällä unkarilaista algoritmia. Tässä työssä assosiointi kuitenkin suoritetaan jokaiselle havainnolle erikseen niiden saavuttua, koska simulaattori tuottaa havaintoja epätasaisin aikavälein.

Toimiakseen odotetulla tavalla suodatusverkko tarvitsee 10 kohteeseen assosioitua havaintoa ja kohteen liikettä ennustava ennustusmalli tarvitsee 5 kohteeseen assosioitua suodatettua havaintoa. Kuitenkin seuranta-algoritmin seurannan aloitusta on mahdollista nopeuttaa täydentämällä ennustuverkon sisääntuloa kopioimalla ensimmäistä kohteen tilaa niin monta kertaa, että sisääntulovektorissa on yhteensä 5 tilaa. Yksittäisen tilan perusteella ennustaminen ei kuitenkaan ole mielekästä, jonka takia näissä tilanteissa päädyttiin käyttämään luvussa 4.5.1 esiteltyä vakiosäteistä assosiointi-ikkunaa. Tilanteessa, jossa seurantaan on assosioitunut alle 10 havaintoa, kohteen tilaa päivitetään ennusteen ja assosioidun havainnon keskiarvolla.

Kaikissa assosioinneissa käytetään havaintojen esisuodatusta, jossa kaikista sellaisista saapuvista havainnoista joiden etäisyys sen hetkisten seurantojen viimeisimpään tilaan on yli 20 km luodaan uusi seuranta. Tällä vähennetään algoritmin laskennallista kuormaa, sillä näin kaukaiset havainnot eivät voi mittausvirheiden rajoissa kuulua millekkään seuranalle.

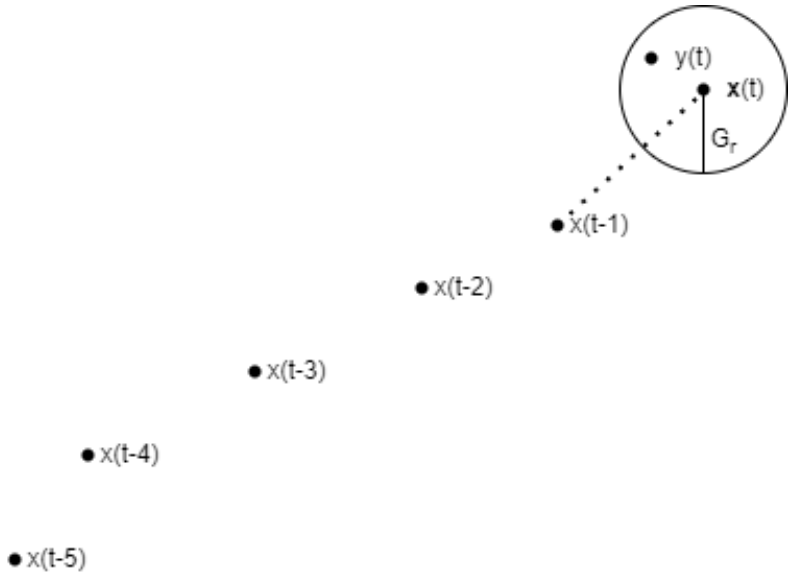
4.5.1 Vakiosäteinen assosiointi-ikkuna

Yksinkertaisimmillaan havaintojen assosiointi on mahdollista suorittaa käyttämällä vakiosäteistä assosiointi-ikkunaa. Tätä metodia päädyttiin käyttämään myös kaikissa seuraavissa assosiointimenetelmissä tilanteissa, jossa seurantaan liitettyjen havaintojen ja näin ollen päivitettyjen tilojen lukumäärä ei ole assositaatiomenetelmälle riittävä. Yksikertaisuudessaan vakiosäteistä asssosiointi-ikkunaa käyttävä algoritmi assosioi havainnon seurantaan, mikäli havainnon ja seurantaan liitetyn viimeisimmän havainnon välinen euklidinen etäisyys on alle ennalta määritellyn säteen. Mikäli useampi seuranta täyttää tämän ehdon, assosioidaan havainto siihen seurantaan johon etäisyys on lyhin. Mikäli mikään seuranta ei täytä tätä ehtoa luodaan uusi seuranta.

Menetelmän yksinkertaisuus kuitenkin aiheuttaa tiettyjä rajoitteita, jotka huonontavat sen suorituskykyä. Koska kyseinen assosiointimenetelmä ei mitenkään huomioi kohteen liikettä eikä sitä, kuinka kauan aikaa on kulunut seurantaan viimeiseksi liitetystä havainnosta, käytettävän assosiointi-ikkunan tulee olla varsin suuri, jotta nopeasti liikkuvasta kohteesta saapuvat havainnot voivat assosioitua kohteelle. Suuri assosiointi-ikkuna taas kasvattaa todennäköisyyttä, jolla myös muut kuin kohteesta lähtöisin olevat havainnot assosioituvat kyseiselle kohteelle.

4.5.2 Vakiosäteinen assosisointi ennustusta käyttämällä

Kun seurantaan on liitetty alle kaksi havaintoa, käytetään assosiointiin luvussa 4.5.1 esiteltyä metodia. Tämän jälkeen metodi käyttää viimeisimpiä seurantaan liittyviä päivitettyjä tiloja ja vastaanotetun havainnon aikaleimaa ennustamaan kohteen sijaintia kyseisellä



Kuva 4.11. *Kaksiulotteinen havainnekuva vakiosäteisen ennustusta hyödyntävän assosioinnin toiminnasta.*

ajanhetkellä. Koska ennustusmalli tarvitsee seurannan viisi viimeisintä tilaa toimiakseen, kopioidaan ensimmäistä tilaa tarvittaessa jotta tämä ehto täyttyy. Assosiointi suoritetaan ennustetun sijainnin ympäriltä käyttämällä vakiosäteistä assosiointi-ikkunaa. Mikäli havainto osuu useamman seurannan assosiointi-ikkunaan, valitaan se seuranta, jossa ennustetun sijainnin ja havainnon sijainnin välinen etäisyys on pienin. Mikäli havainto ei osu minkään seurannan assosiointi-ikkunaan, aloitetaan uusi seuranta.

Kyseinen assosioitimenetelmä huomioi kohteen liikkeen ennustamalla sen paikkaa havainnon saapumishetkelle. Tämän tulisi parantaa assosiointitarkkuutta verrattuna vakiosäteiseen assosiointiin mahdollistamalla pienemmän assosiointi-ikkunan käytön. Ikkunan koko pysyy kuitenkin edelleen vakiona riippumatta kohteen liikemallin epävarmuudesta, jonka voisi ainakin teoreettisesti ajatella huonontavan metodin assosiointikykyä.

Kyseisen metodin toimintaa on havainnollistettu kuvassa 4.11. Siinä tilat $x(t-5)...x(t-1)$ kuvaavat seurannan aikaisempia tiloja. Havainto $y(t)$ kuvaa ajanhetkellä t saapunutta havaintoa. Näin ollen seurannan aikaisempien tilojen avulla lasketaan tilaennuste $\boldsymbol{x}(t)$. Merkillä G_r kuvataan käytetyn vakioikkunan kokoa. Kuvan 4.11 tilanteessa havainto $y(t)$ voi assosioitua kuvatulle seurannalle, koska sen etäisyys ennusteesta on pienempi kuin G_r.

4.5.3 Muuttuvaikkunainen assosiointi ennustusta käyttämällä

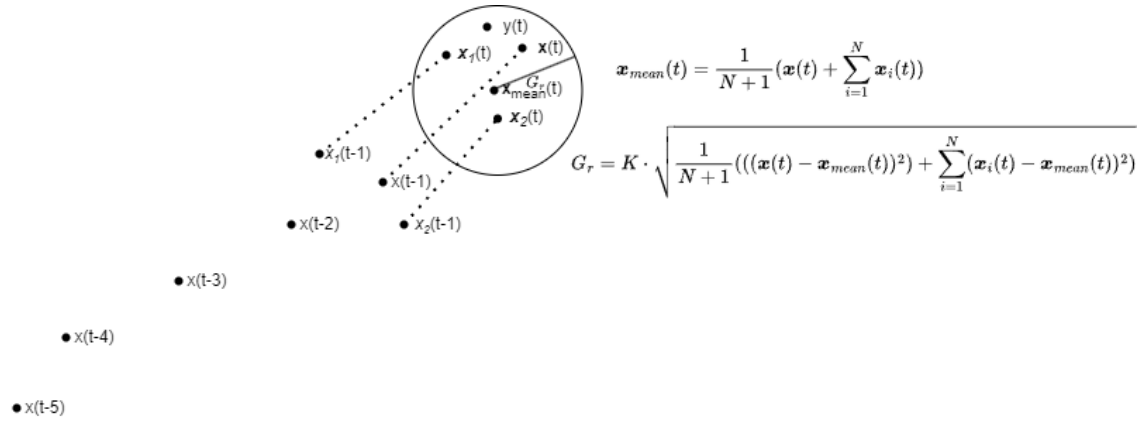
Assosiointi-ikkunan koon muuttaminen ennusteen epävarmuuden mukaan on yleisesti käytetty assosiointimetodi Kalman-suotimia käyttävien tutkaseuranta-algoritmien ohessa. Koska neuroverkkoihin perustuva ennustemalli ei suoraan kuitenkaan muodosta ennus-

teellensa epävarmuutta, täytyy tämä toiminnallisuus lisätä siihen käyttämällä muita metodeita. Yksi lähestymistapa kyseiseen ongelmaan on hyödyntää UKF-suotimessa käytettyä tapaa luoda satunnaisesti näytteistettyjä pisteitä suodatetun seurannan viimeisen pisteen ympärille ja tehdä ennuste näiden avulla.

Kun seurantaan on liitetty alle kaksi havaintoa käytetään luvussa 4.5.1 esiteltyä metodia. Mikäli seuranta sisältää alle viisi päivitettyä tilaa käytetään vastaavaa kopiointimenettelyä kuin luvussa 4.5.2. Menetelmä toimii käyttämällä seurannan viimeisimpiä päivitettyjä tiloja, ja näytteistämällä 100 pistettä viimeisimmän tilan ympäriltä. Näytteistys suoritetaan vain tilan koordinaattipisteille ja muut tilaan liittyvät arvot kopioidaan seurannan viimeisimmästä tilasta. Nämä pisteet näytteistettiin säteeltään 500 m kokoisesta pallosta viimeisen pisteen ympäriltä, jossa pisteet oli normaalijakautuneita. Nämä kaikki 100 tilavektoreita sisältävää sarjaa ajetaan ennustusverkon läpi, jolloin saadaan ennusteita sisältävä pistepilvi. Tämän jälkeen saadusta pistepilvestä laskettiin keskiarvo sekä keskihajonta. Keskiarvoa käytettiin assosiointi-ikkunan keskipisteenä ja keskihajontaa vakiolla skaalattuna ikkunan säteenä. Tämä ratkaisu kuitenkin vaatii, että sopiva pistepilven säde sekä näytteiden määrä tulee optimoida ennalta valittua dataa käyttäen, eikä näin ole käytännön kannalta yksinkertaisin ratkaisu. Lisäksi myös pistepilven jakauman valinta vaikuttaa lopputuloksiin eikä parhaan jakauman löytäminen hyvin epälineaarisen neuroverkon käyttöön ole yksinkertainen tehtävä.

Hajonnan tarkoituksena oli kuvata ennusteen epävarmuutta ennustustilanteessa. Kyseisen arvon ei ollut tarkoituksena siis suoraan toimia assosiointi-ikkunan säteenä, eikä se tehtyjen testien perusteella sellaisena suoraan toiminutkaan. Huomioitavaa on myös se, että ennustusmallin tuottama hajonta oli pienempää tilanteissa, jossa siihen syötettävä tilavektorisarja sisälsi kopioituja arvoja. Tämän takia skaalauskerrointa muutettiin siten että sen arvo oli $K = 20 - M_s$ kun $M_s > 0$ ja $K = 10$ kun $M_s = 0$. Arvo M_s kuvaa kuinka monta suodatettua havaintoa puuttuu ennustusverkon sisääntulosta. Nämä arvot valittiin etsimällä iteratiivisesti arvoja, jolla verkko tekee eniten oikeita assosiaatioita ja vähiten vääriä assosiaatioita. Tämä testaus ei kuitenkaan käynyt kaikkia mahdollisia arvoja läpi, vaan sopivat arvot etsittiin empiirisesti kokeilemalla eri arvoja pientä erillistä datasettiä hyödyntäen.

Kuvassa 4.12 on havainnollistettu metodin toimintaa yksinkertaisessa kaksiulotteisessa tilanteessa. Seurannan aikaisempia tiloja on kuvattu merkein $x(t-5)...x(t-1)$. Viimeisimmän tilan ympäriltä satunnaisesti näytteistettyjä tiloja on esitetty yksinkertaisuuden vuoksi vain kaksi kappaletta ja niitä on merkitty merkein $x_1(t-1)$ ja $x_2(t-1)$. Näille kaikille lasketaan ennustusmallia hyödyntäen ennusteet $\boldsymbol{x}(t)$, $\boldsymbol{x}_1(t)$ ja $\boldsymbol{x}_2(t)$. Ennusteista lasketaan keskiarvo, jota käytetään assosiointi-ikkunan keskipisteenä. Skaalauskertoimella K kerrottua keskijakaumaa käytetään assosiointi-ikkunan säteenä. Kuvan 4.12 tapauksessa havainto $y(t)$ on assosiointi-ikkunan sisäpuolella, jonka takia havainto voidaan assosioida kyseiseen seurantaan.



Kuva 4.12. *Kaksiulotteinen havainnekuva muuttuvaikkunaisen assosioinnin toiminnasta*

Neuroverkoilta on mahdollista saada epävarmuutta kuvaavia lukuja myös näytteistämällä sen ulostuloa käyttämällä ennustuksen aikaista dropouttia. Tämä perustuu ”Dropout as Bayessian approximation: Representing model uncertainity in Deep Learning”-julkaisussa esitettyyn teoriaan [10]. Käytännössä tämä vaatii kuitenkin sen, että ennustusmalliin tulee lisätä dropout-kerros. Kyseistä mallia voitaisiin sitten käyttää suorittamalla useita ennusteita samalla sisääntulovektorilla, siten että dropout toiminnallisuus pidetään päällä myös ennustuvaiheessa. Työssä käytettävällä Keras ja Tensorflow kirjastolla toteutusta on kuitenkin vaikea tehdä nopeaksi, sillä vaikkakin kaikki ennustukset ovat toisistaan riippumattomia ja näin ollen voitaisiin tehdä samanaikaisesti, ei tämä käytännössä ole helposti toteutettavissa. Näistä syistä kyseistä menettelyä ei työssä tutkittu tarkemmin, sillä muutenkin pitkät evaluointiajat olisivat kasvaneet työn aikataulun kannalta liian pitkiksi.

4.5.4 Autoenkoodaajaan perustuva assosiointi

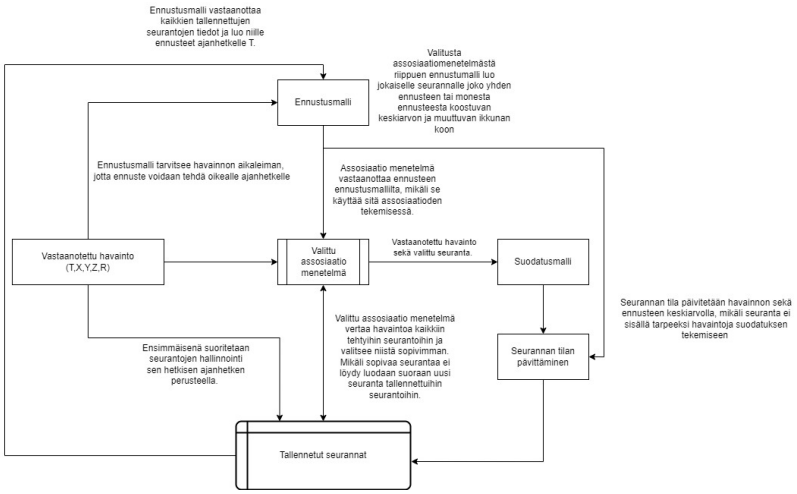
Viimeinen työssä kokeiltu assosiointimenetelmä on täysin koneoppimiseen perustuva. Siinä käytetään kappaleessa 4.3.3 esiteltyä autoenkoodaajamallia. Koska se on opetettu toisintamaan kohteista saatavia havaintosarjoja, voidaan sitä käyttää poikkeavuuksien havainnointiin. Näin ollen autoenkoodajamalli voidaan kouluttaa kokonaan kohteilta saapuvilla havaintosarjoilla ilman negatiivisia koulutusesimerkkejä.

Autoenkoodajamalliin perustuva assosiaattori toimii siis vastaanottamalla 5 havainnon pituisen havaintosarjan ja tekemällä sen jälkeen luokittelun sen perusteella kuinka todennäköinen kyseisen tyyppinen havaintosarja on verrattuna harjoitusdataan. Jotta autoenkoodaajamalli toimii assosiaattorina täytyy sen virheelle eli sisääntulevan havaintosarjan ja ulostulon väliselle erotukselle etsiä iteroimalla sopiva raja-arvo. Tämä tehdään käyttämällä kappaleessa 4.2.3 esitettyä autoenkooderi-mallille tarkoitettua toista harjoitusdatasettiä. Kyseistä dataa hyödyntäen mallille valittiin raja-arvo, jolla jokainen oikea havaintosarja luokittui oikein. Kyseinen menentelmä tarvitsee siis toimiakseen viisi seurantaan liitettyä havaintoa, jonka takia lyhyemmillä seurannoilla käytetään yksinkertaista luvussa

4.5.1 esiteltyä vakiosäteistä assosiointia.

4.6 Seuranta-algoritmin toiminta

Käydään seuraavaksi läpi seuranta-algoritmin toiminta. Seuranta-algoritmi toimii käyttämällä ATERIX-standardin mukaisia CAT034- ja CAT048-sanomia. Käyttämällä näitä sanomia ja luvussa 4.2.2 esiteltyjä yhtälöitä, voidaan tutkalta saatavien havaintojen sijaintitiedot muuttaa ECEF-koordinaatistoon. Koska sanomat sisältävät aikaleiman joka on suhteessa viimeisimpään keskiyöhön, tulee ne samalla muuttaa Unix-aikaleimoiksi, jotta seuranta-algoritmi toimii myös vuorokauden vaihtuessa. Havaintoon liittyvä sanoma sisältää paljon informaatiota, mutta työssä käytettävä algoritmi on suuniteltu toimimaan käyttämään ainoastaan havainnon ajanhetkeä, sijaintia ja etäisyyttä tutkasta. Vastaanotetuista havainnoista välitetään vain nämä tiedot seuranta-algoritmille.



Kuva 4.13. *Seuranta-algoritmin lohkokaaviokuva*

Saapuvat havainnot prosessoidaan seuranta-algorimissa yksi kerrallaan. Ensimmäisenä uuden havainnon saapuessa suoritetaan seurantojen hallinnointi. Tässä sen hetkisten olemassa olevien seurantojen painoja päivitetään ja vanhentuneita seurantoja poistetaan luvun 4.4 mukaisesti. Tämän jälkeen saapuvan havainnon yhteensopivuutta sen hetkisiin seurantoihin tutkitaan yrittämällä assosioida se niihin kaikkiin ja tämän jälkeen valitsemalla se assosiointi, jossa assosiointivirhe on pienin. Mikäli kaikki assosiointivirheet ovat suurempia kuin algoritmille asetettu assosiointiin käytettävä raja-arvo, havainnosta luodaan uusi seuranta. Muuttuvaikkunaisen assosioinnin tapauksessa, jokaiselle seurannalle on oma erillinen raja-arvo joka määrittää voidaanko havainto assosioida kyseiseen seurantaan. Se lasketaan luvussa 4.5.3 esitellyin menetelmin. Autoenkooderin perustuvan assosioinnin tapauksessa, jokaisen seurannan viisi viimeistä havaintoa syötetään autoenkoodaajamalliin ja verrataan ulostulon ja sisääntulon välisen erotuksen summaa kyseiselle metodille asetettuun raja-arvoon.

Mikäli havainto assosioituu olemassa olevaan seurantaan, sen tila päivitetään käyttämällä luvussa esiteltyä suodatusmallia. Mallille syötetään 10 seurantaan viimeiseksi liitettyä havaintoa ja seurannan päivitetyksi tilaksi asetetaan mallin antama viimeinen suodatettu tila. Mikäli seurantaan on liitetty alle 10 havaintoa, sen tilaa päivitetään saadun ennusteen ja havainnon keskiarvolla. Tämän jälkeen päivitettyyn seurantaan liitettyä painoa kasvatetaan luvussa 4.4 esitellyn säännön mukaisesti 0,2 yksikköä ja se sijoitetaan takaisin tallennettuihin seurantoihin.

5. TULOKSET

Tässä luvussa käsitellään työssä esitellyn kokonaisen seuranta-algoritmin tuloksia erillaisia assosiointimenetelmiä käyttämällä. Seurantojen laadun kuvaaminen käyttämällä yksittäistä tunnuslukua on haastavaa, jonka takia lisäksi esitellään käytettävät tunnusluvut. Jotta seuranta-algoritmin toiminnasta käytettävässä ympäristöstä saadaan tuloksia, täytyy se testata kokonaisuutena. Erillisten osien vaikutusten tarkkaa arviointia tuloksiin on kuitenkin tästä syystä vaikea arvioida, jonka takia työssä tarkastellaan ainoastaan käytettävien assosiointi menetelmien mahdollisia vaikutuksia. Luvun lopussa pohditaan tutkimuksen tuloksia, niiden merkittävyyttä ja käytetyn menetelmän haasteita.

5.1 Seuranta-algoritmin evaluointi ja tulokset

Kokonaisen seuranta-algoritmin testaus suoritettiin simuloimalla 30 Constant-Jerk-tekniikalla luotua skenaariota ja 30 satunnaisia käännöksiä luovalla tekniikalla luotua skenaariota. Näin ollen testaukseen käytettävät simuloidut reitit sisältävät vastaavanlaisia liikkeitä kuin mallien koulutukseen käytetty datasetti. Käytetttävä testidata on myös simuloitu käyttämällä samaa tutkakonfiguraatiota kuin harjoitusdatan generoinnissa, joten sen tuottamat virheet ovat vastaavanlaisia. Kaikki skeraariot sisälsivät siis vain yhden tutkan ja yhden kohteen. Näistä saatavat tiedostot, jotka vastasivat Asterix CAT048- ja Asterix CAT034-sanomista saatavia tietoja syötettiin seuranta-algoritmin evaluointiin tarkoitettuun skriptiin. Koska työssä ei haluttu tutkia skenaarioita, joissa kohteesta tulevat tutkahavainnot saapuvat todella harvoin eli yli 18 sekunnin päästä toisistaan, nämä poistettiin prosessointivaiheessa, jotteivat ne vaikuta saataviin lopputuloksiin. Evaluointi suoritettiin käyttämällä luvussa 4 esiteltyä seuranta-algoritmia. Evaluoinnin tarkoituksena oli tarkastella seuranta-algoritmin suorituskykyä kappaleessa 4.5 esiteltyjen erillaisten assosiointimenetelmien kanssa.

Mittareina käytettiin seurannan ja kohteen todellisen sijainnin välistä absoluuttista keskivirhettä MAE, niillä ajanhetkillä, jolloin seuranta-algoritmi oli luonut vahvistetun seurannan. Lisäksi mitattiin, kuinka monta harhahavaintoa (eng. False plot) vahvistettuihin seurantoihin yhteensä assosioitui. Tätä merkitään työssä lyhenteellä NOFPA (engl. Number of false plots assosiated). Vahvistettuihin seurantoihin liitettyjen havaintojen kokonaismäärää merkitään lyhteellä NOPA (engl.Number of plots assosiated). Tämän lisäksi mi-

tattiin, kuinka monta vahvistettua seurantaa reittiä kohden keskimäärin syntyi. Tätä merkitään lyhenteellä NOMT (engl. Number of mature tracks). Lisäksi mitattiin vahvistettujen seurantojen pituudet käyttäen pituuden mittarina seurantaan liitettyjen havaintojen lukumäärää. Tästä laskettiin vahvistettujen seurantojen pituuksien keskiarvo reittiä kohden. Tämä arvo laskettiin jokaiselle käytetylle reitille erikseen ja näistä laskettiin keskiarvo, joka kuvaa keskimääräistä vahvistetun seurannan pituutta. Keskimääräistä vahvistetun seurannan pituutta merkitään tässä työssä lyhenteellä MMTL (engl. Mean mature track length).

Algoritmin samanaikaisesti ylläpitämien vahvistettujen seurantojen lukumäärää ei rajoitettu evaluointia varten. Kuitenkin kaikilla käytetyillä assosiointimenetelmillä seuranta-algoritmi ylläpiti evaluoinnin aikana vain yhtä samanaikaista vahvistettua seurantaa. Tämä oli toivottu tulos sillä, jokainen käytetty skenaario sisälsi vain yhden kohteen. Näin ollen seuranta-algoritmi ei millään käytetyistä assosiointimenetelmistä tuottanut sellaisia vahvistettuja seurantoja, jotka olisivat sisältäneet vain häiriöstä aiheutuneita havaintoja. Tämä tarkoittaa myös sitä, että vahvistettujen seurantojen keskimääräinen lukumäärä sekä vahvistettujen seurantojen keskimääräinen pituus yhdessä kuvaavat seuranta-algoritmin kykyä jatkuvaan seurantaan ilman seurantojen katkeilua.

Seuranta-algorimista testattiin lopulta seuraavat eri versiot: vakioikkunainen assosiointi ennustusta käyttäen (merkitään lyhenteellä VIA), muuttuvaikkunainen assosiointi ennustusta käyttäen (MIA) sekä autoenkooderiin perustuva assosiointi (AE). Kaikki assosiaatiomenetelmät toimivat käyttämällä luvussa 4.5.1 esiteltyä vakioikkunaista assosiointia ilman ennustetta, ennen kuin seurantaan on liitetty ennustusverkon tarvitsemat 2 havaintoa tai autoenkoodaajamallin kohdalla 5 näytettä. Tästä syystä myös muuttuvaikkunaiselle sekä autoenkoodaajaan perustuvalle assosiaattorille on määritetty seurannan alussa käytettävä vakioikkunan koko. Vakioikkunaisen ja muuttuvaikkunaisen assosioinnin kohdalla testattiin ikkunan koot 3000 m, 4000 m, 5000 m ja 10000 m. Autoenkooderiin perustuvalla assosioinnilla testattiin ainoastaan 5000 m ikkunan koko, sillä sen voitiin havaita suoriutuvan huomattavasti huonommin kuin muut menetelmät.

Taulukossa 5.2 on esitetty seuranta-algoritmin tulokset edellä mainituille assosiointimenetelmille käyttäen erilaisia assosiointi-ikkunan kokoja. Taulukossa menetelmän perään suluissa merkitty numero kertoo käytetyn assosiointi-ikkunan koon. Vakioikkunaisen ennustusta käyttävän menetelmän kohdalla tämä kuvaa sekä seurannan alussa käytettävän vakioikkunaisen assosioinnin käyttämän että seurannan käynnistyttyä käytetyn ikkunan koon. Muilla menetelmillä tämä kuvaa vain seurannan alussa käytettävän vakio-ikkunan kokoa.

Testidatassa oli yhteensä 24873 kohteesta saapunutta havaintoa, joten taulukosta 5.1 voidaan huomata, ettei mikään assosiointimenetelmä onnistunut seuraamaan kaikkia reittejä kokonaan, vaan reittien varrelle on sattunut pisteitä, joissa seuranta on katkennut. Har-

Assosiaatiomenetelmä	MAE	NOFPA	NOPA	NOMT	MMTL
VIA (3000m)	615.5	2	22469	7.25	51.7
VIA (4000m)	635.2	5	22720	6.83	55.4
VIA (5000m)	642.6	9	22781	6.68	56.8
VIA (10 000m)	685.7	53	22861	6.68	55.9
MI (3000m)	597.8	22	22354	6.25	59.7
MI (4000m)	624.7	23	22678	6.48	58.4
MI (5000m)	656.2	25	22805	6.48	58.7
MI (10 000m)	697.4	26	22856	6.56	58.0
AE (5000m)	810.0	448	22721	10.68	36.1

Taulukko 5.1. *Seuranta-algoritmin tulokset eri assosiaatiomenetelmillä. Seurannan keskimääräinen etäisyysvirhe (MAE), vahvistettuihin seurantoihin assosioitujen harhahavaintojen kokonaislukumäärä (NOFPA), vahvistettuihin seurantoihin assosioitujen havaintojen kokonaislukumäärä (NOPA), keskimääräinen vahvistettujen seurantojen lukumäärä reittiä kohden (NOMT), keskimääräinen vahvistetun seurannan pituus havaintoina (MMTL). Todellisesta kohteesta saapuneiden havaintojen kokonaislukumäärä oli 24873.*

hahavaintoja testidatassa oli yhteensä 242419. Tuloksista voidaan huomata, että seurantojen ja todellisten reittien välinen etäisyysvirhe MAE kasvaa ikkunan koon kasvaessa. Samalla seurantoihin liitettyjen harhahavaintojen lukumäärä NOFPA kasvaa, joka selittää tätä, sillä harhahavaintojen liittäminen seurantaan siirtää sitä pois oikealta reitiltä ja vaikeuttaa seuraavan oikean assosiaation tekemistä.

Kuitenkin myös se, että mahdollisimman suuri osa todellisista havainnoista assosioituu vahvistetuille seurannoille on seuranta-algoritmin toimivuuden kannalta hyvin oleellista. Tuloksista voidaan havaita, että vakioikkunaisen ja muuttuvaikkunaisen assosiointimenetelmän erot evaluointiin käytetyllä datalla on mittakaavat huomioon ottaen varsin pieniä. Kuitenkin voidaan huomata että vakioikkunainen ennustusta käyttävä assosiointimenetelmä assosioi vähiten harhahavaintoja seurantaan tapauksissa, joissa todellisia havaintoja assosioidaan lähes vastaava määrä kuin muissa menetelmissä. Tämä voidaan havaita tarkastelemalla esimerkiksi rivien VIA(4000m) ja MI(4000m) eroja. Tuloksista voidaan myös huomata, että yli 5000m vakioikkunan käyttö aiheuttaa tulosten huomattavaa huonontumista. Tuloksista voidaan myös havaita, että muuttuvaikkunaisen assosioinnin NOFPA kasvaa suhteessa hitaammin ikkunan kokoa muutettaessa kuin vakioikkunaisen assosioinnin tapauksessa. Tämä kuitenkin selittyy täysin sillä, että siinä ikkunan koko määrittyy täysin dataperustaisesti seurannan alun jälkeen.

Muuttuvaikkunaisen assosiointimetodin tuottamat seurannat ovat keskimääräisesti hieman pidempiä kuin vakioikkunaisen menetelmän tuottamat sekä niitä on yksittäistä reittiä kohden keskimääräisesti hieman vähemmän. Nämä pienet erot selittynevät kuitenkin siitä, että kyseisiin seurantoihin on assosioitunut enemmän harhahavaintoja ja näin ollen ne

ovat pituudeltaan hieman pidempiä.

Koska tuloksissa havaittavat erot ovat pieniä ja valittujen mittareiden painoarvoa menetelmien vertailussa on haastavaa tehdä, tarkastellaan vielä assosiaatiomenetelmän suorituskyvyn kannalta oleellisempia mittareita. Näitä ovat: vahvistettuihin seurantoihin assosioituneiden todellisten havaintojen suhde kohteesta saapuneiden havaintojen lukumäärään (merkitään lyhenteellä ROCA, engl. Ratio of correct assosiations) sekä vahvistettuihin seurantoihin assosioitujen harhahavaintojen lukumäärä suhteessa kaikkiin tehtyihin assosiointeihin (merkitään lyhenteellä ROFA, engl. Ratio of false assosiations).

Assosiaatio metodi	ROCA	ROFA
VI(3000m)	$0,903$	$8,0 \cdot 10^{-5}$
VI(4000m)	$0,913$	$2,2 \cdot 10^{-4}$
VI(5000m)	$0,915$	$4,0 \cdot 10^{-4}$
VI (10 000m)	$0,919$	$2,3 \cdot 10^{-3}$
MI (3000m)	$0,898$	$9,8 \cdot 10^{-4}$
MI (4000m)	$0,912$	$1,0 \cdot 10^{-3}$
MI (5000m)	$0,916$	$1,1 \cdot 10^{-3}$
MI (10 000m)	$0,919$	$1,1 \cdot 10^{-3}$
AE (5000m)	$0,913$	$1,9 \cdot 10^{-2}$

Taulukko 5.2. *Tutkittujen assosiointimenetelmien suorituskyky käytetyllä testidatalla.*

Taulukon 5.2 tuloksista voidaan havaita, että vakioikkunainen assosiointi 4000 m tai 5000 m ikkunalla tuottaa parhaimmat tulokset sillä vastaavien ROCA-lukujen saaminen muilla metodeilla aiheuttaa suuruusluokkia suuremmat ROFA-arvot. Taulukosta 5.2 voidaan myös huomata, että lähes kaikki menetelmät assosioivat yli 90% kohteista saapuneista havainnoista vahvistetuille seurannoille. Tämä osoittaa, että toteutettu seuranta-algoritmi ylläpiti valtaosan ajasta vahvistettua seurantaa testidatassa olleille kohteille ja näin ollen toimi käytetyllä testidatalla.

5.2 Pohdintaa

Luvussa 5.1 havaitut erot assosiointimenetelmien välillä ovat pieniä ja niiden todellinen merkitys käytännön sovelluksessa on todennäköisesti lähes merkityksetön. Vakiosäteinen ennustusta käyttävä assosiointi on kuitenkin menetelmistä yksinkertaisin ja sen käyttäminen vaatii vain yhden parametrin valinnan toimiakseen. Muuttuvaikkunaisen sekä autoenkoodaajaan perustuvan assosioinnin tapauksessa parametrejä on useampia, joka tekee niiden optimoinnista huomattavasti haastavampaa. Yksi mahdollinen syy niillä saaduille vakioikkunaista assosiaatiota huonommille tuloksille onkin se, että valitut parametrit saattoivat olla rajatun optimoinnin takia epäedulliset.

Luvussa 4.6 esitellyn suodatusmallin heikko keskimääräinen suorituskyky mallia erikseen evaluoitaessa aiheuttanee ongelmia kokonaiselle seuranta-algoritmille. Kuitenkin luvussa 4.3.2 koulutetun ennustusmallin suorituskyky oli huomattavasti lineaarista ekstrapolointia parempi käytetyllä datalla. Työssä kuvattu seuranta-algoritmi toimi työssä käytetyllä simulaatiomallilla sekä luoduilla erilaisilla reiteillä ja tuotti todellisista kohteista vahvistettuja seurantoja. Simuloidulta tutkalta saapuneet harhahavainnot eivät myöskään aiheuttaneet ylimääräisiä vahvistettuja seurantoja, joten seuranta-algoritmi tuotti tässä hyvin yksinkertaisessa tilanteessa toivottuja lopputuloksia.

Käytettyä menetelmää ei kuitenkaan aikarajoitteitten sekä objektiivisen vertailun haastavuuden takia vertailtu perinteisiin seuranta-algoritmeihin, joten sen absoluuttisesta suorituskyvystä ei voida vetää johtopäätöksiä. Tämä johtuu siitä, ettei kokonaisia valmiita perinteisiin menetelmiin perustuvia seuranta-algoritmeja ole julkisesti saatavilla. Tällaisen rakentaminen ja optimointi työssä käytettävälle datalle oli työn aikataulun sekä työn laajuuden kannalta mahdotonta. Näin ollen aihe sekä käytetyt menettelyt vaatisivat huomattavasti jatkotutkimusta, jotta sen todellisesta suorituskyvystä sekä soveltuvuudesta käytännön sovelluskohteisiin voitaisiin tehdä tarkkoja johtopäätöksiä.

Koneoppimiseen perustuvan seuranta-algoritmin toteuttaminen on käytännössä haastavaa ja siihen liittyy monia mahdollisia ongelmia. Mallien kouluttamiseen tarvittavan todellisen datan hankinta on monista syistä vaikeaa, jonka takia simulointimalleihin turvautuminen on käytännössä ainoa mahdollinen ratkaisu. Ohjattua oppimista hyödyntävät menetelmät tarvitsevat opetusaineistokseen näytteitä, jotka sisältävät lähtöarvon ja toivotun ulostulon. Todellisia tutkia käyttämällä tällaisten opetusnäytteiden kerääminen on kuitenkin havainnointiin liittyvien epävarmuuksien takia mahdotonta. Näin ollen näytteet täytyy luokitella käyttämällä heuristiikkaa, joka voi vääristää opetusmateriaalia. Työssä käytetty suodatusmalli koulutettiin kohteesta lähtöisin olevilla havaintosarjoilla sekä sarjoilla, jotka kuvasivat kohteen tilaa näillä ajanhetkillä. Tällainen havaintojen ja tilojen välinen täydellinen assosiaatio on kuitenkin mahdollista vain simulaatioita hyödyntäen. Ennustusmalli koulutettiin käyttämällä kohteen todellisia tiloja. Tällaisia tarkkoja tiloja ei myöskään käytännössä ole mahdollista saada kuin simulaatioita käyttämällä. Lentokoneiden GPS-paikantimet kuitenkin pystyvät tuottamaan varsin tarkkoja estimaatteja niiden tiloista, joita voitaisiin mahdollisesti hyödyntää vastaavien mallien kouluttamiseen.

Toinen ongelma on erilaisten liikemallien sisällyttäminen opetusdataan. Työssä käytettiin erilaisia strategioita, joiden tarkoituksena oli tuottaa satunnainen jakauma erilaisia mahdollisia liikkeitä. Käytetyt menetelmät olivat kuitenkin yksinkertaisia ja ne todennäköisesti tuottivat vain pienen osan käytetyn koneen mahdollisista liikkeistä. Näin ollen seuranta-algoritmi ei toimisi esimerkiksi erilaisilla lentokoneilla ja niiden liikkeillä, joita opetusdata ei sisältänyt. Riittävän laajan ja kuvaavan sovelluskohteeseen liittyvän datasetin kerääminen on siis yksi koneoppimismallien kouluttamiseen liittyvistä merkittävistä ongelmista.

Neuroverkkoratkaisuja käytettäessä on myös hyvä huomioida se, että niihin liittyy paljon parametrejä, joiden valintaan ei ole vakiintuneita menetelmiä. Sopivan arkkitehtuurin, piirrevektorin, käytettävien neuronien lukumäärän ja koulutukseen käytettävien hyperparametrien valinta on haastava ja aikaa vievä iteratiivinen prosessi. Tämän vuoksi työssä käytettiin kirjallisuudesta löytynyttä arkkitehtuuria, jota muokattiin vastaamaan työssä käytettävän erilaisen simulaatiomallin haasteisiin. Työssä käytettyihin piirrevektoreihin päädyttiin lyhyiden testien ja päättelyn avulla. Muuttuva-aikavälisiä sarjoja varten piirrevektoreihin lisättiin aikamuuttujat. Kuitenkin esimerkiksi kaksisuuntaiset takaisinkytketyt neuroverkot (engl. Bidirectional recurrent neural network) tai differentiaaliyhtälöihin perustuvat neuroverkot (engl. Neural ordinary differential equations) olisivat olleet varteenotettavia vaihtoehtoja mallien arkkitehtuureiksi.

Koneoppimiseen ja neuroverkkoihin perustuva tutkaseuranta on varsin vähän tutkittu aihealue. Videoon liittyviä neuroverkkopohjaisia monen kohteen seuranta-algoritmeja on kuitenkin tutkittu huomattavasti enemmän ja niissä käytettyjen metodien voitaisiin yhtäläisyyksien vuoksi ajatella generalisoituvan myös tutkaseurantaan. Kuitenkin varsinkin ilmavalvontatutkiin liittyvän datan vaikea saatavuus ja näin ollen yleisesti hyväksyttyjen datasettien puuttuminen vaikeuttaa aiheen tutkimista. Tämän vuoksi tällaisen datasetin kerääminen tai laajan ja tutkitun datasetin simulointi ja julkaisu voisivat olla kiinnostavia jatkotutkimusaiheita.

6. YHTEENVETO

Työssä tutkittiin koneoppimismenetelmien soveltuvuutta yhden tutkan ja yhden kohteen tapauksessa. Tavoitteena oli rakentaa tutkasimulaatioympäristössä toimiva koneoppimiseen perustuva seuranta-algoritmi nämä rajoitteet huomioon ottaen. Tämän kokonaistavoitteen saavuttamiseksi työssä tutkittiin aikaisempia tutkaseurannassa käytettyjä algoritmeja ja menettelyjä sekä koneoppimisen teoriaa keskittyen neuroverkkoihin. Tämän jälkeen tuotettiin lentoreittejä sekä tutkahavaintoja sisältävä datasetti, jota käytettiin toteutettavan seuranta-algoritmin kouluttamiseen sekä evaluointiin. Tämän jälkeen toteutettiin kokonainen yhden tutkan ja yhden kohteen seurantaan tarkoitettu algoritmi. Algoritmia varten toteutettiin useita tutkahavaintojen assosiointimenetelmiä. Näitä olivat vakioikkunainen ennustusta käyttävä assosiointi, muuttuvaikkunainen assosiointi sekä autoenkoodaajaan perustuva assosiointi. Näiden suorituskykyä testattiin ja vertailtiin osana kokonaista seuranta-algoritmia.

Työn alussa tutkittiin tutkien toimintaperiaatteita sekä perinteisten tutkaseuranta-algoritmien toimintaa sekä niiden rajoitteita. Tämän jälkeen tutkittiin koneoppimismenetelmien ja niistä tarkemmin neuroverkkojen toimintaperiaatteita. Näistä havaittiin, että takaisinkytketyillä neuroverkoilla on monia ominaisuuksia, jotka tekisivät siitä seuranta-algoritmiin soveltuvan tekniikan. Perinteisiä tutkaseuranta-algoritmeja tutkittaessa havaittiin myös, että yhden kohteen seurantatapauksissa yleisesti käytettävät havaintojen assosiointimetodit perustuvat seurantoihin liittyviin epävarmuuksiin, joiden saaminen neuroverkkopohjaiselta ratkaisulta ei ole yksiselitteistä. Nämä tutkimukset antoivat hyvän pohjan seuranta-algoritmin toteuttamiseen sekä eri ongelmista, joita sen toteuttamisessa tulisi huomioida. Näitä oppeja ja huomioita hyödynnettiin seuraavassa luvussa, jossa toteutettiin koneoppimiseen pohjautuva seuranta-algoritmi.

Koneoppimiseen perustuvan tutkaseuranta-algoritmin toteuttamisessa oli monia eri askelia. Ensimmäisenä toteutettiin algoritmit, joilla tuotettiin simulaatiomallilla simuloitavat lentoreitit. Tämän jälkeen valittiin reittipisteiden kanssa käytettävä koordinaatisto. Koska työn alkuperäisenä tarkoituksena oli soveltaa menetelmää myös monen tutkan tapaukseen, päädyttiin käyttämään globaalisti yhtenevää ECEF-koordinaatistoa. Neuroverkkopohjaisten ratkaisujen tiedetään toimivan parhaiten, kun käytettävät arvot on skaalattu pienelle arvovälille nollan lähettyville. Tämän vuoksi kehitettiin metodi, jolla reittipistevektoreista tai havaintovektoreista koostuvat aikasarjat keskitettiin origon ympärille ja niiden

arvot skaalattiin välille [-1,1] . Perinteisissä seuranta-algoritmeissa tilaestimointiin käytettävä Kalman-suodin päädyttiin korvaamaan kahdella neuroverkolla. Näihin neuroverkkoihin päädyttiin tutkimalla aikaisempia neuroverkkopohjaisia monen kohteen seuranta-algoritmeja koskevia julkaisuja. Lisäksi työssä käytettävä seurantojen hallinnointi pohjautui näissä julkaisuissa käytettyihin ratkaisuihin. Käytettävän tutkasimulaation rakenteesta tiedettiin, että se voi tuottaa epätasaisilla aikaväleillä syntyviä havaintoja, jonka takia piirrevektoreihin lisättiin erillinen aikamuuttuja. Suunnitteluvaiheessa neuroverkkojen ennusteiden havaittiin painottuvan lineaarisiin tapauksiin. Tästä syystä harjoitusdatasetin tasapainotukseen kehitettiin metodi, jolla sen sisältämien erilaisten reittityyppien välisiä suhdemääriä korjattiin. Seuraavaksi luvussa keskityttiin seuranta-algoritmiin liittyvään havaintojen assosiointiin. Koska tutkimuksissa ei löydytty yhden kohteen seurantatapaukseen liittyviä neuroverkkoratkaisulle sopivia havaintojen assosiointimenetelmiä, päädyttiin näitä kehittämään muutamia sekä testaamaan niiden suorituskykyä. Koska neuroverkkopohjaisilla algoritmeilla on tietyt käynnistymisajat, eli ne tarvitsevat tietyn määrän havaintoja toimiakseen, päädyttiin seurantojen aluissa käyttämään yksinkertaista havaintoihin perustuvaa lähimmän naapurin menettelyä. Työtä varten kehitettyjä havaintojen assosiointimenetelmiä olivat: vakioikkunainen assosiointi ennustusta käyttäen, muuttuvaikkunainen assosiointi ennustetta käyttäen sekä autoenkoodaajaan perustuva assosiointi.

Työn viimeisessä luvussa testattiin työssä kehitetyn kokonaisen seuranta-algoritmin suorituskykyä näillä erilaisilla assosiointimenetelmillä. Tätä varten reittien generointiin käytettävillä menetelmillä tuotettiin yhteensä 60 kokonaista lentoreittiä, jotka simuloitiin käytettävällä tutkasimulaatiolla. Eri assosiaatiomenetelmien tuottamia keskietäisyysvirheitä, assosioitujen harhahavaintojen lukumäärää, vahvistettujen seurantojen lukumäärää yksittäistä reittiä kohden sekä vahvistettujen seurantojen keskipituutta vertailtiin toisiinsa. Tuloksista havaittiin, että autoenkoodaajaan perustuva assosiointi toimi menetelmistä huonointen ja vakioikkunaisen ennustusta käyttävän sekä muuttuvaikkunaisen assosioinnin väliset erot olivat hyvin pieniä. Tuloksista voitiin kuitenkin havaita, että menetelmät assosioivat suurimman osan kohteelta saapuneista havainnoista vahvistettuun seurantaan ja näin ollen seuranta-algoritmin voidaan todeta toimivan tässä yksinkertaisessa tilanteessa. Vakioikkunaisen ennustusta käyttävän ja muuttuvaikkunaisen assosioinnin väliset erot evaluointituloksissa olivat kuitenkin pieniä, jonka takia rakenteeltaan ja toiminnaltaan yksinkertaisempaa vakioikkunaista ennustusta käyttävää assosiointia pidettiin parhaana ratkaisuna.

Viimeiseksi pohdittiin vielä työn tulosten merkitystä ja niihin liittyviä rajoitteita sekä käytiin läpi mahdollisia kiinnostavia jatkotutkimusaiheita. Tuloksista havaittiin, että kehitetty algoritmi toimi työssä rajatussa yksinkertaisessa tilanteessa. Algoritmin suorituskykyä ei kuitenkaan pystytty käytännön syistä vertaamaan perinteisiin seuranta-algoritmeihin, jonka takia sen soveltuvuudesta todelliseen seurantaongelmaan ei voida tehdä johtopäätöksiä. Tutkimusta aiheesta voisi tulevaisuudessa jatkaa vertailemalla toteutetun algoritmin suo-

rituskykyä perinteisiin seuranta-algoritmeihin. Lisäksi tutkimusta voitaisiin jatkaa vähentämällä työssä asetettuja rajoitteita kohteiden liikemalleille, samanaikaisten seurantojen lukumäärälle sekä käytettävien tutkien lukumäärälle.

LÄHTEET

[1] Yaakov Bar-Shalom, X.-Rong Li ja Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. eng. New York: Wiley, 2004. ISBN: 047141655X.

[2] Sunitha Basodi et al. ”Gradient amplification: An efficient way to train deep neural networks”. *Big Data Mining and Analytics* 3.3 (2020), s. 196–207. DOI: 10.26599/BDMA.2020.9020004.

[3] Y. Bengio, P. Simard ja P. Frasconi. ”Learning long-term dependencies with gradient descent is difficult”. *IEEE Transactions on Neural Networks* 5.2 (1994), s. 157–166. DOI: 10.1109/72.279181.

[4] Mervin C. Budge. *Basic radar tracking*. eng. Artech House radar series. Boston: Artech House, 2018. ISBN: 1-63081-336-2.

[5] *CAT017 - EUROCONTROL Specification for Surveillance Data Exchange Part 5 Category 017 (Appendix A)*. Kesäkuu 2005. URL: https://www.eurocontrol.int/publication/cat017-eurocontrol-specification-surveillance-data-exchange-part-5-category-017-0 (viitattu 21. 12. 2023).

[6] Sudha Challa ja Sudha Challa. *Fundamentals of object tracking*. eng. Cambridge: Cambridge University Press, 2011. ISBN: 1-280-88667-6.

[7] Subramanian Chandramouli, Saikat Dutt ja Amit Das. *Machine Learning*. eng. 1st edition. Pearson Education India, 2018. ISBN: 93-89588-13-8.

[8] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[9] K.L. Du ja M.N.S. Swamy. *Neural Networks and Statistical Learning*. SpringerLink : Bücher. Springer London, 2013. ISBN: 9781447155713. URL: https://books.google.fi/books?id=wzK8BAAAQBAJ.

[10] Yarin Gal ja Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2016. arXiv: 1506.02142 [stat.ML].

[11] Chang Gao et al. ”Maneuvering Target Tracking with Recurrent Neural Networks for Radar Application”. Teoksessa: *2018 International Conference on Radar (RADAR)*. 2018, s. 1–5. DOI: 10.1109/RADAR.2018.8557284.

[12] Arash Gharehbaghi. *Deep learning in time series analysis*. eng. Boca Raton, FL: CRC Press, 2023. ISBN: 0-429-32125-2.

[13] Marco Gori. *Machine learning*. eng. Morgan Kaufmann, 2017. ISBN: 0-08-100670-5.

[14] *Handbook of radar signal analysis*. eng. First edition. Advances in applied mathematics. Boca Raton: C&H \CRC Press, 2021. ISBN: 9781351665247.

[15] Sepp Hochreiter ja Jürgen Schmidhuber. ”Long short-term memory”. *Neural computation* 9.8 (1997), s. 1735–1780.

[16] Kurt Hornik, Maxwell Stinchcombe ja Halbert White. ”Multilayer feedforward networks are universal approximators”. *Neural networks* 2.5 (1989), s. 359–366.

[17] Diederik P. Kingma ja Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[18] Jukka Kolari ja Aleksi Kallio. *Tekoäly 123 : matkaopas tulevaisuuteen*. fin. Jyväskylä: Docendo, 2023. ISBN: 978-952-382-375-4.

[19] David Last et al. ”Stone Soup: announcement of beta release of an open-source framework for tracking and state estimation”. Teoksessa: *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVIII*. Toim. Ivan Kadar, Erik P. Blasch ja Lynne L. Grewe. Vol. 11018. International Society for Optics ja Photonics. SPIE, 2019, s. 1101807. DOI: 10.1117/12.2518514. URL: https://doi.org/10.1117/12.2518514.

[20] Warren S McCulloch ja Walter Pitts. ”A logical calculus of the ideas immanent in nervous activity”. *The bulletin of mathematical biophysics* 5 (1943), s. 115–133.

[21] Krzysztof Patan. *Artificial neural networks for the modelling and fault diagnosis of technical processes*. eng. 1st ed. 2008. Lecture notes in control and information sciences ; 377. Berlin, Germany: Springer, 2008. ISBN: 3-540-79872-2.

[22] Ning Qian. ”On the momentum term in gradient descent learning algorithms”. *Neural Networks* 12.1 (1999), s. 145–151. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00116-6. URL: https://www.sciencedirect.com/science/article/pii/S0893608098001166.

[23] *Radar handbook*. eng. 3rd ed. Maidenhead: McGraw-Hill Professional, 2008. ISBN: 0-07-162113-X.

[24] M.A. Richards et al. *Principles of Modern Radar: Basic Principles, Volume 1*. Electromagnetics and Radar. Institution of Engineering ja Technology, 2010. ISBN: 9781891121524. URL: https://books.google.fi/books?id=nD7tGAAACAAJ.

[25] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[26] Siddharth Sharma, Simone Sharma ja Anidhya Athaiya. ”ACTIVATION FUNCTIONS IN NEURAL NETWORKS”. *International Journal of Engineering Applied Sciences and Technology* 04 (toukokuu 2020), s. 310–316. DOI: 10.33564/IJEAST.2020.v04i12.054.

[27] Zhiyuan Shi et al. ”LSTM-based Flight Trajectory Prediction”. Teoksessa: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, s. 1–8. DOI: 10.1109/IJCNN.2018.8489734.

[28] Amit Kumar Tyagi ja Ajith Abraham. *Recurrent Neural Networks : concepts and applications*. eng. First edition. Boca Raton: CRC Press, 2023. ISBN: 9781032310565.

[29] Yongquan ZHANG et al. ”Online multi-target intelligent tracking using a deep long-short term memory network”. *Chinese Journal of Aeronautics* (2023). ISSN: 1000-9361. DOI: https://doi.org/10.1016/j.cja.2023.02.006. URL: https://www.sciencedirect.com/science/article/pii/S1000936123000274.