Pinola Juho

Neuroverkkotyökalun rakentaminen valitulla kirjastolla

Insinööri (AMK)
Tieto- ja viestintätekniikka
Syksy 2022

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Pinola Juho

Työn nimi: Neuroverkkotyökalun rakentaminen valitulla kirjastolla

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: neuroverkot, Python, Scikit-learn, TkInter

Keinotekoiset neuroverkot ovat nykyään hyvin yleisesti käytetty koneoppimismenetelmä. Neuroverkkoja käytetään laajasti erilaisten ongelmien ratkaisemiseen. Tälle on perustana, että neuroverkkojen potentiaaliset käyttötarkoitukset ovat moninkertaistuneet vuonna 2012 ja sen jälkeen neuroverkkotutkimuksessa saavutettujen edistysaskeleiden vaikutuksesta. Tämäkin opinnäytetyö rakentuu tuon kehityksen varaan. Kuten millä tahansa muullakin asialla, myös neuroverkoilla on hyvät ja huonot puolensa. Tämä pitää paikkansa niin etiikan kuin käytännön ohjelmoinninkin kannalta. Tämän opinnäytetyön tarkoituksena oli selvittää, kuinka hyvin tietyntyyppinen neuroverkko soveltui annetun tutkimusongelman ratkaisemiseen.

Tässä työssä käytiin läpi aluksi kattavasti neuroverkkojen teoriaa, jota tarvittiin työn myöhemmissä vaiheissa. Tämän jälkeen toteutettiin Python-ohjelmointikieltä käyttäen kokeellinen tutkimus, jossa aluksi tarkasteltiin kolmea erilaista neuroverkkokirjastoa (Pytorch, Scikit-learn ja Tensorflow) ja valittiin niistä tämän työn tarkoitusta varten paras (Scikit-learn) työn loppuosan toteuttamista varten.

Työn pääosan muodosti raporttimainen kuvaus työhön soveltuvan neuroverkon toteuttamisesta ja soveltamisesta valitulla neuroverkkokirjastolla Python-ohjelmointikieltä käyttäen. Työssä käytetty neuroverkon opettamisessa ja testaamisessa käytetty datasetti esiteltiin omassa luvussaan. Työn toteuttamisen aikana luotiin myös graafinen käyttöliittymä (TkInter-kirjastoa käyttäen), jonka avulla neuroverkko-ohjelman loppukäyttäjä voi säätää neuroverkon suorituskykyyn vaikuttavia parametreja. Opinnäytetyön lopputulos oli Windows 10 - tai Windows 11 -ympäristössä toimiva tietokoneohjelma, jota pystyy käyttämään sellainenkin henkilö, jolle ohjelmointi ja neuroverkot eivät ole ennestään erityisen tuttuja.

Neuroverkon suorituskykyä havainnollistettiin tässä työssä käyttäen sekä toimeksiantajalta saatua neuroverkon toiminnan tavoitetasoa että visualisointeja, jotka edelleen paransivat neuroverkon suorituskyvyn havainnollistumista annetulla datasetillä. Työn loppuosassa käytiin läpi mahdollisia keinoja työssä toteutetun neuroverkon toiminnan optimoimiseen edelleen. Lopussa oli yhteenveto työssä saavutetuista asioista sekä tekijän oma pohdinta.

KAMK • University of Applied Sciences

Abstract

Author: Pinola Juho

Title of the Publication: Building of Neural Network Tool with Chosen Library

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: neural networks, Python, Scikit-learn, TkInter

Artificial neural networks are a very commonly used machine learning technique nowadays. Neural networks are widely used to solve different kinds of problems. This is since potential use cases of neural networks have increased tremendously because of remarkable progress made in neural network research in the year 2012 and after it. Also, this bachelor's thesis is based on that progress. Like any other thing, also neural networks have good and bad aspects. This is true concerning both ethics and practical programming. The aim of this thesis is to resolve how well a certain type of neural network can be applied to solve a given research problem.

In this thesis, neural networks are first discussed extensively on theoretical level. This approach is needed later in this thesis. After that, experimental research is conducted using Python programming language. In this research, three different neural network libraries (Pytorch, Scikit-learn and Tensorflow) are first evaluated. In the end, one of those libraries (Scikit-learn) is chosen to be used in the rest of this thesis.

The main part of this thesis is composed of a report-like depiction about the implementation and application of a neural network suitable for the purpose of this work with the chosen neural network library using the Python programming language. The dataset used in this thesis to train and test the neural network is represented in its own chapter. Also, during the making of this thesis a graphical user interface (using TkInter library) is implemented. Through it, the end user of the neural network program can change parameters connected to the performance of the neural network. The result of this thesis is a computer program that can be run on Windows 10 or Windows 11 operating system and that can be used by a person who is not already very familiar with programming and neural networks.

The performance of the neural network is clarified in this thesis using both the desired level of neural network performance given by the contractor and visualizations that further enhance the understanding of the performance of the neural network with the given dataset. In the last chapters of this thesis, some ways to further optimize the performance of the neural network are discussed. In the very last chapter, there are a summary of things achieved in this thesis and the author's own reflection.

Alkusanat

Valitsin opinnäytetyölleni tämän aiheen pääasiassa siksi, että aihetta ehdotettiin Prometec Tools Oy:n puolesta. Oli sekä minun että Prometecin kannalta hyödyllistä, että sain jatkaa hyvin sujuneen opintoihin kuuluvan harjoittelun jälkeen samassa yrityksessä opinnäytetyön parissa. Osasyynä aiheen valintaan oli myös se, että AMK-opintojeni aikana olen tutustunut monenlaisiin neuroverkkoihin ja neuroverkkokirjastoihin. Tämän huomioon ottaen oli loogista valita opinnäytetyölleni juuri neuroverkkoihin liittyvä aihe. Kolmas syy aiheen valintaan oli oma kiinnostukseni ohjelmointia, neuroverkkoja ja graafisten käyttöliittymien luomista kohtaan. Nuo kaikki kolme ovat asioita, joiden parissa haluan työskennellä myös sitten, kun olen saanut opintoni Kajaanin ammattikorkeakoulussa päätökseen.

Haluan kiittää tämän työn tekemisessä Prometecin taholta minua työn sisällön suhteen neuvonutta ohjaajaani Olli Kakkoa. Lisäksi haluan kiittää Kajaanin ammattikorkeakoulun taholta opinnäytetyötäni ohjannutta opettajaa Pekka Huttusta.

Sisällys

Termit ja lyhenteet

Aktivaatiofunktio	Matemaattinen funktio, joka määrää jokaisesta neuroverkon neuronista tulevan ulostulon suuruuden.
Alisovittuminen	Tilanne, jossa neuroverkon suorituskyky sekä opetus- että validointijoukossa jää heikoksi.
Binaarivektori	Vektori, jossa yksi arvoista on arvossa 1 ja kaikki muut ovat arvossa 0.
Epookki tai iteraatio	Neuroverkkojen tapauksessa yksittäinen opetuskierros, jonka aikana koko sisään tuleva datajoukko syötetään kertaalleen neuroverkon läpi.
Häviöfunktio	Matemaattinen funktio, jonka avulla määritetään neuroverkon ennusteissa jokaisella epookilla esiintyvä virhe oikeisiin tuloksiin verrattuna.
Neuroverkkokirjasto	Ohjelmistokirjasto, joka mahdollistaa toimivien neuroverkkojen toteuttamisen ilman, että käyttäjän täytyy itse lähteä toteuttamaan kaikkea itse.
Opetusjoukko	Datajoukko, jota käytetään neuroverkon opettamiseen. Opetusjoukko sisältää sekä varsinaisen neuroverkolle syötettävän datan että target-vektorin.
Optimointialgoritmi	Algoritmi, jonka avulla neuroverkon ennusteissa esiintyvä virhe pyritään minimoimaan jokaisella epookilla.
Sisään tuleva data	Matriisi, joka syötetään neuroverkolle sitä opetettaessa tai sen toimintaa validoitaessa.
Target-vektori	Vektori, joka sisältää oikeat tulokset, joita neuroverkon tulisi oppia ennustamaan.

Validointijoukko	Datajoukko, jota käytetään neuroverkon toiminnan testaamiseen opettamisen jälkeen. Validointijoukko sisältää eri dataa kuin opetusjoukko.
Ylisovittuminen	Tilanne, jossa neuroverkko suoriutuu hyvin opetusjoukon datalla, mutta huonosti ennen näkemättömällä datalla (esimerkiksi validointijoukon datalla).

1 Johdanto

Tämän opinnäytetyön toimeksiantaja on Prometec Tools Oy, joka on suomalainen vuonna 2012 toimintansa aloittanut kone- ja prosessisuunnittelun alalla toimiva yritys [1]. Prometec Tools Oy:n kotipaikka on Kajaani, ja työntekijöitä sillä on ollut joulukuussa 2021 päättyneellä tilikaudella 12 [2]. Prometecin toimisto sijaitsee Kajaanissa Renforsin rannan yritysalueella osoitteessa Sokajärventie 9 [3].

Kuten tämän opinnäytetyön otsikko vihjaa, tavoitteena on neuroverkkotyökalun rakentaminen valittua neuroverkkokirjastoa käyttäen. Toteutettavan neuroverkkotyökalun toiminnallisuuden vaatimusmäärittely on Prometec Tools Oy:n toimittama. Vähimmäisvaatimus Prometecin puolesta tälle työlle on luoda maallikon käytettävissä oleva Windows 10 - tai Windows 11 -ympäristössä toimiva tietokoneohjelma, joka käyttää valitun tyyppistä neuroverkkoa ennusteiden tekemiseen. Kajaanin ammattikorkeakoulun puolesta tavoite on tuottaa selkeä ja tieto- ja viestintätekniikan ammattilaisille ymmärrettävä dokumentti, jossa on käsitelty riittävällä tarkkuudella neuroverkkojen teoriaa ja toteutettu käytännöllinen tutkimus valitusta aiheesta.

Työn toteutuksen aikana valitaan kolmesta vaihtoehdosta ensin paras neuroverkkokirjasto ja sitten tuon kirjaston avulla toteutetaan toimiva neuroverkko. Neuroverkolle on Prometecin puolesta asetettu tietty tavoitetarkkuus, jonka saavuttaminen tuottaa huomattavasti lisäarvoa toimeksiantajalle. Tuon tavoitetarkkuuden saavuttamiseen käytetään yleisesti neuroverkkojen kanssa käytettyjä optimointimenetelmiä. Työn lopussa tehtävät johtopäätökset riippuvat siitä, saavutettiinko tuo tavoitetarkkuus.

Työn tutkimusongelmana on selvittää, onko mahdollista tekijän käytettävissä olevilla taidoilla rakentaa neuroverkko, joka saavuttaa edellä mainitun tavoitetarkkuuden. Opinnäytetyötä voisi edelleen laajentaa esimerkiksi siten, että otettaisiin käyttöön jokin muu kuin tässä työssä valittu neuroverkkokirjasto ja tutkittaisiin samoja menetelmiä käyttäen, paraneeko neuroverkon suorituskyky.

Ongelmaa, jota tässä työssä käsitellään, olisi voitu periaatteessa lähestyä myös käyttäen muita koneoppimismenetelmiä kuin neuroverkkoja. Käytettäväksi vaihtoehdoksi valikoituivat kuitenkin

neuroverkot niiden hyvän virheensietokyvyn ja monipuolisten optimointimahdollisuuksien vaikutuksesta. Tieto siitä, että neuroverkkojen sisäistä toimintaa ei tunneta täysin tarkasti vielä nykyään, ei tässä tapauksessa haittaa.

Toimeksiantajan puolesta tässä työssä toteutettavalle neuroverkolle on määritelty toiminnan tavoitetasoksi, että 95 % kaikista neuroverkon tekemistä ennusteista menee +/- 3 %-yksikön vaihteluvälin sisään silloin, kun asiaa tarkastellaan x-akselia pitkin histogrammin luokkien lukumäärän keskikohdasta molempiin suuntiin lähtien. Tämä tarkoittaa, että vaihteluvälin sisään kelpaavat vain ne arvot, joiden etäisyys keskikohdasta on alle tai tasan 3 %-yksikköä. Näin ollen neuroverkon suorituskyky on sitä parempi, mitä suurempi prosentuaalinen osuus kaikista ennusteista menee tuon vaihteluvälin sisään. Tässä työssä toteutettu ohjelma esittää lopuksi graafisessa käyttöliittymässä myös niiden arvojen osuuden, jotka ovat 5 ja 7 %-yksikön päässä keskikohdasta. Nämä kaksi ylimääräistä luokkaa ovat tarpeellisia siksi, että niiden avulla saadaan selville neuroverkon tarkkuus myös halutun tavoitetarkkuuden ympäristössä.

Työn edetessä edellä mainitun tavoitetason saavuttaminen osoittautui varsin vaativaksi toimenpiteeksi, jonka saavuttaminen vaatii monialaista neuroverkon optimointia. Tämän työn lopussa selostetaan vaihtoehtoja tuon optimoinnin toteuttamiseksi. Varsinaista tutkimusta optimoinnista ei kuitenkaan tässä työssä esitetä, koska toimeksiantajan mukaan luvussa 6 toteutettu selostus ohjelman käyttöliittymästä, käyttöliittymän toiminnasta ja tulosten tulkintaohjeista on oleellisempaa jatkokäytön kannalta.

2 Neuroverkkojen teoreettinen perusta

” We cannot solve our problems with the same thinking we used when we created them.”
Albert Einstein

2.1 Keinotekoisen neuroverkon määritelmä

Lyhyesti sanottuna keinotekoinen neuroverkko on ihmisen luoma tietoverkko, joka pyrkii jäljittelemään ihmisen tai eläimen aivojen näköaivokuoren rakennetta ja toimintaa. Aivan kuten biologinen hermoverkko ihmisen tai eläimen aivojen näköaivokuoressa, myös keinotekoinen neuroverkko koostuu tuoja- ja viejähaarakkeista sekä tietoa käsittelevistä yksiköistä. [4.]

Tuoja- ja viejähaarakkeet pelkästään välittävät tietoa neuronien (tietoa käsittelevät yksiköt) välillä. Kuten myöhemmin tämän työn teoriakappaleessa kuvataan, yksittäisen neuronin tehtävä tiedonkäsittelyn kannalta on periaatteessa varsin yksinkertainen myös keinotekoisissa neuroverkoissa. Jos neuroneita on riittävästi, ne kuitenkin pystyvät muodostamaan yhdessä suuriakin määriä tietoa käsittelevän verkoston. Sekä biologisten että keinotekoisten neuroverkkojen tapauksessa voidaankin sanoa, että kokonaisuus on enemmän kuin osiensa summa.

2.2 Historiaa

Keinotekoisten neuroverkkojen historia ulottuu yllättävän kauas, aina 1940-luvulle asti [5]. Tämän jälkeen keinotekoisten neuroverkkojen kehittäminen on edennyt välillä nopeammin ja välillä hitaammin aina vuoteen 2012 asti, jolloin alkoi keinotekoisten neuroverkkojen nopea nousukausi alalla saavutettujen suurten edistysaskeleiden vaikutuksesta [6]. Poikkeavaa tässä nousukaudessa verrattuna sitä edeltäneisiin on, että sille ei näy loppua ja uusien keinotekoisiin neuroverkkoihin liittyvien innovaatioiden määrä jatkaa kasvuaan eksponentiaalisesti [6].

Aiemmassa kappaleessa kuvattu vuoden 2012 edistysaskel johtuu sekä laskentatehon kasvamisesta että teoreettisesta edistyksestä keinotekoisten neuroverkkojen alalla. Vuoden 2012 jälkeen

keinotekoisista neuroverkoista on tullut maailmanlaajuisesti suosittu tutkimusaihe, jolla on myös monia merkittäviä käytännön sovelluksia. Keinotekoisten neuroverkkojen käyttöönottoa helpottavat nykyään huomattavasti monet saatavilla olevat valmiit ohjelmakirjastot, joita voidaan käyttää ilmaiseksi monien erilaisten ohjelmointikielten kanssa. Tämä tekee mahdolliseksi neuroverkkojen käytännön soveltamisen sellaisillekin tahoille, jotka eivät ole matemaattisten taitojensa puolesta sillä tasolla, että voisivat luoda täysin itsenäisesti uusia neuroverkkotyyppejä.

2.3 Eettiset näkökohdat

Keinotekoinen neuroverkko voi käsitellä periaatteessa mitä tahansa tietokoneen luettavissa olevaa dataa (tekstiä, kuvaa, ääntä, videota). Tämä tekee keinotekoisista neuroverkoista moneen pystyviä sillä oletuksella, että opetusdataa on saatavilla tarpeeksi. Juuri tämä neuroverkkojen potentiaalisen käyttämisen monialaisuus tekee niistä hyviä tutkimuskohteita niin luonnontieteen (varsinkin matematiikan ja tietojenkäsittelytieteen) kuin etiikankin kannalta. Eettiset näkökohdat ovat tulleet erityisen näkyviksi jo 2010-luvun viimeisinä vuosina, mutta varsinkin 2020-luvulle tultaessa. Keinotekoisten neuroverkkojen avulla voidaan saada aikaan paljon hyvää, mutta varsinkin suurvaltojen (joilla on käytettävissä valtavasti resursseja esim. neuroverkkojen opettamiseen) käsissä niillä voidaan saada aikaan myös harmia. Tällainen kehitys avaa uusia näkökulmia etiikkaan ja filosofiaan.

Oman lisänsä keinotekoisten neuroverkkojen tutkimukseen antaa sekin, että kaikki nykyään suositut ja helppokäyttöiset suurten datamäärien käsittelyyn soveltuvat neuroverkkokirjastot ovat alun perin Yhdysvalloissa päämajaansa pitävien suuryritysten (Google ja Meta) kehittämiä [7][8]. Itse keinotekoisten neuroverkkojen taustalla olevan matematiikan ja sen sovellusten kannalta tämä ei vielä ole suuri ongelma, mutta esim. esiopetettujen neuroverkkojen tapauksessa eettisiä kysymyksiä voi herätä. Nämä ongelmat liittyvät nykymaailmassa usein siihen, että esim. kuvia käsittelevän verkon esiopetuksessa käytetyssä datassa on ollut pääasiassa valkoihoisia ihmisiä, varsinkin miehiä. Tällaisen esiopetetun neuroverkon käyttäminen monista erilaisista etnisistä taustoista tulevien ihmisten kuvien luokitteluun johtaa usein epätarkkoihin tuloksiin. Edellä mainitulla

tavalla opetetuista neuroverkoista on hankala päästä eroon laajemmalla tasolla, koska vain harvalla keinotekoisista neuroverkoista kiinnostuneella taholla on käytettävissä niin paljon resursseja (dataa ja laskenta-aikaa), että he voisivat tehdä esiopettamisen itse uudelleen.

Datatieteessä yleinen sanonta on, että kilpailun voittaa se, jolla on käytettävissä eniten dataa eikä se, jolla on paras algoritmi. Tämä on keinotekoisten neuroverkkojen kannalta sikäli huolestuttavaa, että monissa totalitaristisissa maissa kerätään systemaattisesti dataa kansalaisista. Esimerkkinä keinotekoisten neuroverkkojen eettisestä näkökulmasta kyseenalaisesta käytöstä voidaan mainita kiinalaisten tutkijoiden kansainvälisessä tiedelehdessä vuonna 2018 julkaisema tutkimus, jossa oli kehitetty keinotekoinen neuroverkko uiguurien tunnistamiseen valvontakameroiden kuvista [9, s. 178].

Keinotekoisilla neuroverkoilla on kuitenkin myös monia eettisesti oikeita käyttötarkoituksia. Näitä ovat esimerkiksi niiden käyttäminen lääketieteessä sairauksien tunnistamiseen kuvista sekä tähtitieteessä tunnistamaan uusia avaruuden kohteita aiemman datan perusteella [10][11]. Monesti eettisesti oikeisiin käyttötarkoituksiin kuuluvat myös useat chattibotit, suosittelujärjestelmät sekä esimerkiksi Googlen ilmainen kielenkäännösohjelma [10].

On vaikeaa sanoa, onko keinotekoisista neuroverkoista etiikan kannalta enemmän hyötyä vai haittaa. Uudet tieteelliset innovaatiot, ilmastonmuutoksen eteneminen tai vaikka (paikallinen tai globaali) energiapula voivat saada aikaan aivan uusia käyttötarkoituksia keinotekoisille neuroverkoille. Todellisen suunnan voi näyttää vain tulevaisuus.

2.4 Keinotekoisten neuroverkkojen rakenne ja toiminta

Keinotekoinen neuroverkko on yksinkertaisimmillaankin monesta eri osasta koostuva kokonaisuus. Ensinnäkin tarvitaan datan käsittelyn tyypin määräävä tietoverkon rakenne (neuroverkon tyyppi) sekä algoritmit, jotka käsittelevät dataa halutulla tavalla ennen ulostuloa. Oleellisia keinotekoisen neuroverkon suorituskyvyn kannalta ovat myös erilaiset parametrit, joita algoritmeille voidaan antaa.

Keinotekoisissa neuroverkoissa kaksi oleellista, niiden olemuksen ja käyttötarkoituksen määräävää tekijää ovat neuroverkon tyyppi ja sen käyttämä aktivaatiofunktio. Kuten arvata saattaa sekä

keinotekoisten neuroverkkojen tyyppejä että aktivaatiofunktioita on olemassa useita erilaisia. Nykyisen neuroverkkojen nousukauden vaikutuksesta molempia tullaan todennäköisesti keksimään lisää seuraavien vuosien aikana.

Keinotekoisissa neuroverkoissa käytetyistä algoritmeista oleellisia ovat optimointialgoritmi sekä neuroverkon suorituskyvyn muuttumista epookkien määrän funktiona mittaava häviöfunktio. Näille molemmille on olemassa monia eri vaihtoehtoja, ja myös nämä algoritmit ovat aktiivisen tutkimuksen ja kehittämisen kohteena nykyään.

Seuraavaksi esitellään ensin yleisimmät ja nykyään merkittävimmät neuroverkkojen tyypit, kerrotaan aktivaatiofunktioista, määritellään optimointialgoritmi ja häviöfunktio sekä kerrotaan yleisimmistä vaihtoehdoista noille kahdelle. Lopuksi kerrotaan neuroverkkojen hyvistä ja huonoista puolista sekä neuroverkkojen toteuttamisesta ja optimoimisesta käytännön tasolla.

2.4.1 Neuroverkkojen tyypit

Keinotekoisten neuroverkkojen tyypeistä vanhin (ensimmäisenä keksitty) tyyppi on perseptroni (engl. perceptron), joka koostuu sisään- ja ulostulokerroksesta sekä yhdestä neuronista ja jossa data virtaa vain eteenpäin. Aktivaatiofunktiona toimi tällöin porrasfunktio, jonka arvo on joko 0 tai 1. Tällaisenaan perseptroni pystyy käsittelemään ongelmia, joissa data pitää luokitella kahteen eri luokkaan. Myöhemmin perseptronia on laajennettu monikerroksiseksi ja käyttämään muitakin aktivaatiofunktioita kuin porrasfunktiota. Tällöin se pystyy käsittelemään tehokkaasti myös ongelmia, joissa data pitää saada luokiteltua useampaan kuin kahteen eri luokkaan. [12.]

Konvoluutioneuroverkko (engl. convolutional neural network) käsittää tavallisen neuroverkon rakenteen lisäksi konvoluutiokerroksen, jonka avulla voidaan määrittää havaitun piirteen sijainti ja vahvuus. Konvoluutio toimii siten, että jokainen lähtödatan alkio syötetään tietyn dataa analysoivan algoritmin läpi. Tämä algoritmi määrittää painoarvon jokaiselle analysoidulle yksikölle. Mitä korkeampi painoarvo on, sitä enemmän tietty piirre vaikuttaa neuroverkon ulostuloon. Edellä mainittu on tarpeellista silloin, kun neuroverkon käsittelemästä datasta pitää korostaa joitakin asioita ja jättää toiset vähemmälle huomiolle. [13.]

Toistuva neuroverkko (engl. recurrent neural network) on tavallaan omaa toimintaansa korjaamaan pyrkivä neuroverkko. Käytännössä oman toiminnan korjaaminen tapahtuu siten, että data liikkuu neuroverkon sisällä edestakaisin: ulostulo syötetään tarvittaessa takaisin neuroverkolle, jolloin aluksi väärin mennyt ennuste pystytään korjaamaan ennen lopullista ulostuloa. Tämän tyyppiset neuroverkot ovat hyödyllisiä luonnollisen kielen käsittelyä vaativissa tehtävissä. [14.]

Modulaarinen neuroverkko (engl. modular neural network) koostuu kahdesta tai useammasta toisistaan riippumatta toimivasta neuroverkosta. Jos käsiteltävä ongelma on tyypiltään sellainen, että se voidaan jakaa osiin, modulaarisista neuroverkoista on hyötyä, sillä rinnakkain tapahtuva datan käsittely nopeuttaa laskentaa kokonaisuutena. [12.]

2.4.2 Aktivaatiofunktiot

Aktivaatiofunktio on matemaattinen funktio, jonka läpi jokaisen neuroverkon neuronin läpi mennyt data lopuksi ajetaan. Se määrittää vaihteluvälin niille arvoille, jotka tulevat ulos neuroverkon eri kerroksista ja lopuksi koko neuroverkosta. Vaihteluvälin määrittäminen helpottaa tulosten tulkintaa ja jatkokäyttöä. Aktivaatiofunktion ulostulon matemaattinen tarkastelu määrittää, onko neuroverkon tuottama ulostulo tyypiltään vain erillisiä arvoja sisältävä (diskreetti) vai jatkuva. Yleisimpiä aktivaatiofunktioita nykyään ovat sigmoidi/logistinen funktio, tanh eli hyperbolinen tangentti, ReLU (Rectified Linear Unit) ja Softmax [15]. Uusin ja aloittelevan neuroverkkojen toteuttajan tarpeita ajatellen käyttökelpoisin näistä on ReLU [15]. Seuraavaksi jokainen edellä luetelluista aktivaatiofunktioista esitellään pääpiirteittäin. Tämän alaluvun seuraavissa tekstikappaleissa aiheen käsittelyyn on käytetty lähdettä [15].

Sigmoidifunktio on graafisessa muodossa S-kirjaimen muotoisen käyrän muodostava funktio. Sen ulostulo rajautuu aina välille]0,1[, joka tekee siitä käytännöllisen varsinkin binaarisiin luokitteluongelmiin, joissa ulostulona on vain kaksi erillistä luokkaa. Tämä johtuu siitä, että väli]0,1[on helppo jakaa keskikohdan suhteen kahteen yhtä suureen osaan.

Hyperbolinen tangenttifunktio on pitkälti samantapainen kuin sigmoidifunktio, mutta siitä ulostuleva arvojoukko on välillä]-1,1[. Tämän vaikutuksesta ulos tulevien arvojen keskiarvo on hyvin lähellä nollaa, mikä helpottaa oppimista seuraavan kerroksen osalta.

ReLU on laskennallisesti huomattavasti kevyempi kuin kaksi edellä mainittua aktivaatiofunktiota, koska se sisältää määritelmänsä perusteella yksinkertaisempaa matematiikkaa: se palauttaa suoraan sille syötetyn arvon, jos tuo arvo on positiivinen, muussa tapauksessa funktio palauttaa nollan. Laskennallinen keveys näkyy siinä, että ReLU:a käyttävä neuroverkko oppii nopeammin kuin sigmoidi- tai hyperbolista tangenttifunktiota käyttävä neuroverkko.

Softmax on myös tietyntyyppinen sigmoidifunktio, joka on käyttökelpoinen erityisesti ongelmissa, joissa data pitää saada jaettua moneen eri luokkaan. Se rajaa ensin jokaisen luokan ulostulon välille]0,1[ja jakaa äskeisen tuloksen vielä ulostulojen summalla. Laskennallisesti vaativien laskutoimitusten takia Softmax on tässä työssä esitellyistä aktivaatiofunktioista hitain oppimaan.

2.4.3 Optimointialgoritmit

Optimointialgoritmeja (engl. optimization algorithm, optimizer, solver) käytetään määrittämään pienin mahdollinen virhe neuroverkon ulostulolle. Käytännössä tämä tapahtuu optimoimalla häviöfunktio jokaisella epookilla mahdollisimman lähelle globaalia minimiä muuttamalla sopivassa vaiheessa painokertoimia ja oppimisnopeutta. Aihetta käsitellään tässä alaluvussa käyttäen lähdettä [16].

Gradient descent on yksinkertaisin, mutta myös eniten käytetty, optimointialgoritmi. Se käyttää optimointiin vain häviöfunktion ensimmäisen kertaluvun derivaattaa. Tätä tietoa käyttämällä se laskee, millä tavalla painokertoimia pitää muuttaa optimoidun tuloksen saavuttamiseksi. Painokertoimia päivitetään vain silloin, kun gradientti on ensin laskettu koko datasetille. Yksinkertaisuutensa vuoksi gradient descent on nopea optimointialgoritmi, mutta riskinä on jäädä jumiin paikalliseen minimiin ja datasetin ollessa suuri suppenemiseen voi mennä todella pitkä aika.

Stokastinen gradient descent (engl. stochastic gradient descent) on tavallisen gradient descentin paranneltu muunnelma. Suurin ero näiden kahden välillä on, että stokastinen gradient descent päivittää painokertoimia häviön jokaiselle näytteelle laskemisen jälkeen. Tästä seuraa, että stokastinen gradient descent suppenee nopeammin eikä jää läheskään niin helposti jumiin paikalli-

seen minimiin. Se myös kuluttaa vähemmän muistia, koska gradienttia ei tarvitse laskea koko datasetille kerralla. Huonoja puolia tässä optimointialgoritmissa ovat suuri varianssi painokertoimissa sekä se, että globaalin minimin löytyminen ei ole täysin varmaa.

Kolmas gradient descent -pohjainen optimointialgoritmi on mini-batch gradient descent, jossa on pyritty yhdistämään sekä tavallisen gradient descentin että sen stokastisen muunnelman hyvät puolet. Se jakaa datasetin yhteen tai useampaan osajoukkoon (engl. batch) ja päivittää painokertoimet jokaiselle osajoukolle erikseen. Tästä seuraa, että painokertoimissa esiintyy vähemmän varianssia kuin stokastisen gradient descentin tapauksessa ja muistinkäyttö on hillitympää kuin tavallisessa gradient descentissä. Tässäkin optimointialgoritmissa haasteeksi jää edelleen varsinkin se, että optimaalisen oppimisnopeuden löytäminen voi olla vaikeaa.

Momentum-algoritmi vähentää stokastisen gradient descentin korkeaa varianssia ja myös pehmentää suppenemista siten, että algoritmi ”ottaa vauhtia” kohti minimiä ja vähentää samalla aaltoilua ei-toivottuun suuntaan. Tämä optimointialgoritmi suppenee nopeammin kuin tavallinen gradient descent. Huono puoli on, että liikemäärää (engl. momentum) varten tarvitaan uusi hyperparametri, joka pitää valita manuaalisesti.

Nesterovin kiihdytetty gradientti (engl. Nesterov Accelerated Gradient, NAG) pyrkii ennustamaan sitä suuntaa, johon gradientti lähtee tulevaisuudessa aiempien tulosten perusteella. Se laskee häviön käyttämällä arvioitua (tulevaa) gradientin arvoa. Näin ollen algoritmi ei ohita paikallista minimiä ja hidastaa minimin lähestyessä. Tässä on kuitenkin edelleen ongelmana, että liikemäärää kuvaava hyperparametri pitää valita manuaalisesti.

Adagradissa oppimisnopeutta muutetaan jokaiselle painokertoimelle erikseen jokaisella mahdollisella ajanhetkellä. Algoritmi tekee isoja muutoksia harvinaisiin parametreihin ja pieniä muutoksia usein esiintyviin parametreihin. Tässä on etuna varsinkin, että oppimisnopeutta ei tarvitse valita manuaalisesti. Huonoja puolia ovat algoritmin laskennallinen raskaus toisen kertaluvun derivaattojen käsittelyn takia sekä se, että oppimisnopeus pienenee jatkuvasti aiheuttaen hitautta oppimiseen.

AdaDelta pyrkii korjaamaan edellisessä algoritmissa esiintyvän oppimisnopeuden pienenemisongelman rajoittamalla aiempien tarkasteltavien gradienttien määrän tiettyyn vakiomäärään sen si-

jaan, että käytettäisiin kaikkia aiemmin laskettuja gradientteja. Algoritmi käyttää eksponentiaalisesti liikkuvaa keskiarvoa kaikkien gradienttien summan sijasta. Tässäkin algoritmissa on huonona puolena oppimisnopeuden manuaalisen asettamisen vaatimus sekä laskennallinen raskaus.

Adam (Adaptive Moment Estimation) perustuu ensimmäisen ja toisen kertaluvun liikemääriin. Se on tässä työssä esitellyistä optimointialgoritmeista ”älykkäin”, sillä se osaa vähentää suppenemisnopeutta aina, kun mahdollinen minimi lähestyy. Näin vältetään minimin ohittaminen ja samalla kauempana minimeistä tapahtuva hidastelu. Näitten seikkojen takia algoritmi suppenee nopeasti, ei pienennä oppimisnopeutta ajan kuluessa ja korjaa korkean varianssin ongelman. Oikeastaan ainoa suurempi haitta on laskennallinen vaativuus. Adamia pidetäänkin usein parhaana ja monesti käyttäjän kannalta helpoimpana optimointialgoritmina.

2.4.4 Häviöfunktiot

Häviöfunktio (engl. loss function, error function) määrittää neuroverkon ulostulon ja oikean arvon välisen eroavaisuuden (häviön tai virheen). Sama asia voidaan esittää myös niin, että häviöfunktion avulla voidaan löytää erilaisia mittaustapoja käyttäen parhaiten annettuun arvojoukkoon sopiva regressiosuora. Häviöfunktiot voidaan jakaa jatkuvaa ulostuloa vaativissa regressio-ongelmissa käytettäviin sekä diskreettiä ulostuloa vaativissa luokitteluongelmissa käytettäviin. Seuraavaksi esitellään ensin regressio-ongelmissa yleisesti käytettävät kolme häviöfunktiota ominaisuuksineen ja sen jälkeen vielä kolme luokitteluongelmissa käytettävää häviöfunktiota. Aihetta käsitellään tässä alaluvussa lähteen [17] avulla, ellei muuta ole erikseen mainittu.

Keskimääräinen absoluuttinen virhe (engl. Mean Absolute Error, MAE) toimii siten, että se laskee oikean arvon ja neuroverkon ulostulon välisten erotusten itseisarvojen summan ja jakaa lopuksi tuloksen näytteiden määrällä. Neliöllinen keskivirhe (engl. Mean Squared Error, MSE) on pitkälti samankaltainen kuin MAE, mutta erona on, että MSE neliöi yksittäiset virheet sen sijaan, että se laskisi noiden virheiden itseisarvon. Kolmas regressio-ongelmissa käytettävä häviöfunktio on neliöllisen keskivirheen neliöjuuri (engl. Root Mean Squared Error, RMSE), joka määritellään MSE:n neliöjuurena.

MSE:tä käytetään varsin usein sekä neuroverkkojen kanssa regressio-ongelmissa että myös yleisemmin insinööritieteissä virhettä mitattaessa. Haluttaessa käsitellä kaikkia virhetermejä samanarvoisina on MAE hyvä valinta virhefunktioksi, kun taas haluttaessa antaa enemmän painoarvoa suurille virheille ovat MSE ja RMSE parempia vaihtoehtoja. MAE:n ja MSE:n/RMSE:n erona on myös, että ensin mainittu ei ole herkkä outlier-arvoille, kun taas kaksi jälkimmäistä ovat. Monesti herkkyys outlier-arvoille ei ole varsinkaan neuroverkkojen kanssa suuri ongelma, koska nuo arvot voidaan suodattaa datasta pois ennen datan syöttämistä neuroverkolle. Kaikkien tässä ja edellisessä tekstikappaleessa mainittujen häviöfunktioiden arvojoukko on $[0, \infty[$. [18.]

Binaarista ristientropiaa (engl. binary cross-entropy) käytetään häviön laskemisessa sellaisissa luokitteluongelmissa, joissa data pitää saada jaettua kahteen eri luokkaan. Kategorinen ristientropia (engl. categorical cross-entropy) suoriutuu myös tapauksista, joissa luokkia on enemmän kuin kaksi. Siinä sisään tulevat arvot ovat one-hot-muodossa, joka tarkoittaa, että jokaisella käsiteltävällä rivillä yksi arvoista on 1 ja kaikki muut ovat arvossa 0. Harva kategorinen ristientropia (engl. sparse categorical cross-entropy) eroaa tavallisesta kategorisesta ristientropiasta siten, että siinä käsiteltävät rivit eivät ole one-hot-muodossa, vaan jokaista käsiteltävää riviä vastaa vain yksi lukuarvo. Harva kategorinen ristientropia on käytännöllinen silloin, kun luokkia on hyvin paljon.

2.4.5 Vahvuudet ja heikkoudet

Keinotekoisia neuroverkkoja kannattaa käyttää mieluummin kuin muita koneoppimisalgoritmeja varsinkin silloin, kun virheensietokyvyn täytyy olla hyvä (kun täytyy neuroverkon opettamisen jälkeen käyttää sisään tulevaa dataa, jossa on puuttuvia arvoja). Keinotekoisten neuroverkkojen etuja ovat myös, että ne voivat oppia monimutkaisia ja ei-lineaarisia riippuvuussuhteita sekä se, että ne osaavat yleistää oppimiaan asioita (voivat ennustaa ennen näkemättömiä riippuvuussuhteita aiemmin oppimiensa riippuvuussuhteiden perusteella). Joissakin yhteyksissä (esim. hyvin suurten datasettien tapauksessa) neuroverkkojen kyky hajautettuun tiedonkäsittelyyn on myös vahvuus. [19.]

Keinotekoisten neuroverkkojen suosio nykyään perustuu pitkälti saatavilla olevan datan suureen määrään, saatavilla olevan laskentatehon kaksinkertaistumiseen vuosittain, algoritmien nopeaan

kehittymiseen sekä keinotekoisten neuroverkkojen saamaan usein varsin positiiviseen julkisuuteen [20]. Nämä kaikki ovat asioita, jotka ennestään helpottavat ja nopeuttavat keinotekoisten neuroverkkojen käyttöä ja joista hyötyvät sekä teoreettinen tutkimus että käytännön sovellukset. Juuri tämä monialaisuus tekee keinotekoisista neuroverkoista varteenotettavan vaihtoehdon nykyään moneen tiedonkäsittelyn ongelmaan. Kaikkia mahdollisia ongelmia niiden avulla ei kuitenkaan ole järkevää tai edes mahdollista ratkaista.

Keinotekoisten neuroverkkojen suurin yksittäinen huono puoli on, että on vaikea saada selville tarkat syyt sille, miksi neuroverkko on päätynyt juuri tietynlaiseen lopputulokseen. Tämä on ongelma varsinkin silloin, kun neuroverkko ei tuota haluttua lopputulosta. Tämän vuoksi keinotekoiset neuroverkot sopivat huonosti sellaisten ongelmien ratkaisemiseen, joissa pitää tietää tarkasti koko lopputulokseen johtanut päättelyketju. Toinen huono puoli on keinotekoisten neuroverkkojen vaatima pitkä kehitysaika. Tämä tarkoittaa, että vaikka käytettäisiinkin jotakin valmista neuroverkkokirjastoa, kirjaston käyttöönottoon kuluu usein huomattavan pitkä aika verrattuna muiden koneoppimismenetelmien käyttöönottoon kuluvaan aikaan. Huonoiksi puoliksi voidaan lukea myös keinotekoisten neuroverkkojen vaatima suuri datamäärä verrattuna muihin koneoppimistekniikoihin sekä edellä mainitusta seuraava laskennallinen raskaus. [20.]

2.4.6 Opettaminen ja validointi

Joissakin neuroverkkokirjastoissa (esim. Tensorflow ja Scikit-learn) on mahdollista käyttää valmista metodia neuroverkon opettamiseen [21][22]. Useimmissa tavalliselle käyttäjälle vastaan tulevissa tilanteissa tämä on paras vaihtoehto yksinkertaisuutensa takia, mutta on myös tilanteita, joissa opettaminen täytyy tehdä manuaalisesti. Ilmeisin näistä tilanteista on, kun valmista opetusmetodia ei ole saatavissa. Tällainen tilanne on suosituimmista neuroverkkokirjastoista Pytorchin tapauksessa. Toinen tilanne, jossa neuroverkon opettaminen pitää tehdä manuaalisesti on, kun halutaan hallita opetusprosessia tarkemmin kuin valmiit metodit mahdollistavat. Kolmas manuaalista opettamista vaativa tilanne on, kun neuroverkon laskenta pitää saada hajautettua esimerkiksi useammalle kuin yhdelle GPU:lle.

Neuroverkon opettaminen tarkoittaa prosessia, jossa dataa syötetään neuroverkolle sisään (input-data) ja verrataan toistuvasti mallidataan (target-data) niin kauan, kunnes halutut tavoitteet

joko opetusaskelten (epookkien) määrän tai oppimistavoitteiden suhteen on täytetty. Opettamisen jälkeen neuroverkkoa voidaan käyttää ennusteiden tekemiseen sellaisella datalla, jota neuroverkolle ei ole ennen syötetty.

Neuroverkon toiminnan taso (se, kuinka hyvin neuroverkko on oppinut) testataan validoinnin avulla. Validoinnissa neuroverkolle syötetään uutta, ennen sille syöttämätöntä dataa ja tutkitaan, kuinka hyvin neuroverkon tekemät ennusteet vastaavat validointijoukon mallidataa. Käytännössä usein neuroverkon suorituskyky on validointijoukossa hieman heikompi kuin opetusjoukossa, koska välttämättä kaikkia validointijoukossa olevia näytteitä ei ole ollut mukana opetusjoukossa. Tämä saa aikaan epävarmuutta neuroverkon toiminnassa.

2.4.7 Suorituskyvyn arviointi

Käytännössä aina, kun neuroverkkoa opetetaan, on tarpeellista saada selville, kuinka hyvin neuroverkko oppii opetusaskelten määrän kasvaessa. Oppimista mitataan aiemmin esiteltyjen virhefunktioiden avulla. Niiden avulla saadaan selville, kuinka hyvä on neuroverkon yleistävyys (kyky toimia ennen näkemättömän datan kanssa).

Yleisin neuroverkon suorituskyvyssä esiintyvä ongelma on ylisovittuminen (engl. overfitting). Siinä neuroverkko suoriutuu hyvin opetusjoukon datalla, mutta huonosti ennen näkemättömällä datalla validointijoukossa tai silloin, kun neuroverkkoa käytetään myöhemmin esimerkiksi yksittäisten ennusteiden tekemiseen. Ylisovittuminen johtuu siitä, että neuroverkkoa on opetettu liian pitkään. [23.]

Toinen ongelma, joka voi periaatteessa esiintyä liian vähän opetetun neuroverkon tapauksessa, on alisovittuminen (engl. underfitting). Siinä neuroverkon suorituskyky sekä opetus- että validointijoukossa jää kehnoksi. Periaatteessa tämä ongelma on helppo kiertää opetusaskelten määrää lisäämällä, mutta käytännössä tämä voi olla ylimääräinen haaste esimerkiksi silloin, kun opetusdataa on hyvin paljon ja opetukseen käytettävät resurssit ovat rajalliset. [23.]

Aika yleisesti ajatellaan, että neuroverkon suorituskykyä voidaan parantaa suoraan lisäämällä opetusdatan määrää. Pitkälti tämä pitääkin paikkansa, mutta myös datan laatu vaikuttaa opetus-

tuloksiin. Esimerkiksi jos opetusjoukon datassa on vain vähän toisistaan poikkeavia arvoja, neuroverkko ei opi yleistämään sellaisessa sisääntulojoukossa, jossa on ennen näkemättömiä arvoja. Toisaalta jos opetusjoukon arvoissa on liikaa toisistaan poikkeavia arvoja, lopputulos on usein huono, koska neuroverkko oppii ennustamaan vain suuren joukon yksittäisiä, ennalta määrättyjä arvoja.

2.4.8 Suorituskyvyn optimointi

Koska keinotekoiset neuroverkot ovat nykyään niin suosittuja, myös niiden toiminnan optimointi on paljon tutkittu aihe. Monimutkaisuutta tähän aiheeseen luo, että kaikki keinot eivät sovi kaikkiin tilanteisiin. Tämän takia neuroverkon toteuttajan pitää tietää ainakin jossain määrin, minkälaista optimointia voidaan käyttää missäkin tilanteessa. Ilmeisimmät (ja usein myös helpoimmat) keinot parantaa neuroverkon suorituskykyä ovat piilokerrosten määrän kasvattaminen, neuronien lukumäärän kasvattaminen piilokerroksissa sekä epookkien määrän lisääminen opetusvaiheessa. Edellä mainitut ovat myös tehokkaimpia keinoja parantaa neuroverkon suorituskykyä.

Seuraavaksi esitellään joukko muita keinoja, joilla neuroverkon suorituskykyä voidaan edelleen optimoida edellä mainittujen keinojen lisäksi ylisovittumisen ennaltaehkäisemisen avulla. Seuraavat tekstikappaleet pohjautuvat lähteeseen [23].

Datan täydentämisessä (engl. data augmentation) datasettiä kasvatetaan keinotekoisesti luomalla uusia näytteitä jo olemassa olevien pohjalta. Kuvamuotoisen datan tapauksessa tämä voi tarkoittaa esimerkiksi kuvien rajaamista, kääntämistä tai zoomaamista. Tämä käytäntö on tehokkuutensa vaikutuksesta nykyään varsin yleinen kuvia käsittelevien neuroverkkojen kanssa. Etuna on datasetin koon kasvamisen lisäksi, että täydennetyllä datalla opetettu neuroverkko oppii tunnistamaan myös kuvia, jotka ovat esimerkiksi epätarkkoja, niissä esiintyy kohinaa tai tunnistettava kohde ei näy kokonaan.

Joukkonormalisointi (engl. batch normalization) tarkoittaa kaikkien neuroverkolle syötettävien arvojen skaalaamista -1:n ja 1:n välille. Tämän menetelmän hyötyjä ovat muun muassa parantunut gradienttien virtaus ja korkeammat oppimisnopeudet.

Dropout-menetelmässä jätetään satunnaisesti pois arvojen aktivaatiofunktion läpi ajaminen osasta jokaisen neuroverkon kerroksen neuroneita ennen arvojen syöttämistä seuraavalle kerrokselle. Ylisovittumisen todennäköisyyden pienentämisen lisäksi tämä menetelmä auttaa mallia yleistämään paremmin ja nopeammin.

Painokertoimien pienentäminen (engl. weight decay) ottaa käyttöön valitun kokoiset painokertoimet, jotka pienenevät edettäessä kohti ulostulokerrosta. Koska neuroverkko koostuu matemaattisesti jokaisen neuronin painokertoimesta (engl. weight) ja vakiotermistä (engl. bias), näitä kahta muuttamalla voidaan vaikuttaa myös koko neuroverkon ulostuloon. Painokertoimien pienentämisen tapauksessa pienennetään asteittain vain painokertoimia.

Aikainen lopettaminen (engl. early stopping) tarkoittaa neuroverkon opettamisen lopettamista heti, kun häviöfunktion arvo validointijoukossa alkaa kasvaa. Edellä mainittu on selvä merkki ylisovittumisesta, jonka esiintyessä neuroverkon suorituskyky alkaa laskea. Vaikka onkin yksinkertainen parannus, aikainen lopettaminen auttaa paljon suurten datasettien tapauksessa.

L1/L2-regularisaatio ottaa käyttöön neuroverkon sisäisen sakkojärjestelmän pitääkseen neuroverkolle syötettävät sarakkeet hallinnassa. Tämä menetelmä on erityisen käyttökelpoinen suurten mallien, joissa on paljon sisään tulevia sarakkeita, opettamisessa. L1/L2-regularisaatio on suosittu tutkimusaihe nykyään erityyppisille neuroverkoille erikseen sovellettuna.

3 Tekniikat ja menetelmät

3.1 Käytetyt ohjelmistot ja suoritusympäristö

Tässä opinnäytetyössä käytetään kaiken ohjelmointiin liittyvän toteuttamisessa Python-ohjelmointikieltä, joka on korkean tason oliopohjainen tulkattava ohjelmointikieli. Suoritusympäristönä on koko työn ajan Windows 11 Home, mutta seuraavissa kappaleissa kuvatuilla ohjeilla tässä työssä käytetyn kanssa identtinen ohjelmistoympäristö on mahdollista ottaa käyttöön myös Windows 10:ssä. Lisäksi ei ole vaikutusta sillä, onko kyseessä Home- vai esimerkiksi Professional-editio Windowsista.

Python-tulkki voidaan ladata Python Software Foundationin nettisivuilta [24]. Lataamisen jälkeen tulkki täytyy asentaa. Asentaminen on varsin helppo toimenpide tavallisellekin käyttäjälle, joten tässä työssä ei mennä tämän syvemmälle siihen. Nykyään myös Pythonin käyttämä Pip-pakettienhallintaohjelma tulee automaattisesti mukana virallisilta Pythonin nettisivuilta ladatussa asennuspaketissa. Pip mahdollistaa tätä kirjoitettaessa (marraskuu 2022) yli 400 000 lisäpaketin asentamisen Python-tulkkiin [25]. Pip:n käyttäminen on yksinkertaista, seuraavaksi käydään läpi pakettien asentaminen sen avulla Windows 10 - tai Windows 11 -ympäristössä.

Ensimmäisessä vaiheessa avataan Windowsin komentokehote (engl. command prompt). Koska nykyään asennusohjelma osaa asettaa Python-tulkin suoritettavan tiedoston sisältävän kansion suoraan Path-ympäristömuuttujaan, Pip:tä käytettäessä ei tarvitse huolehtia siitä, missä kansiossa Windowsin komentokehote on avattu. Näin ollen paketin asentamiseksi Pip:llä riittää kirjoittaa komentokehotteeseen *pip install paketin_nimi,* jossa *paketin_nimi* korvataan asennettavan paketin oikealla nimellä. Sitten painetaan Enter, ja Pip etsii automaattisesti uusimman ja asennettuna olevan Python-tulkin kanssa yhteensopivan version valitusta paketista. Pip osaa myös asentaa automaattisesti kaikki ne paketit, jotka vaaditaan valitun paketin asentamiseen (riippuvuudet). Jos pakettia ei löydy tai sitä ei voitu asentaa, Pip tulostaa komentokehotteeseen virheilmoituksen.

Kaikki tämän työn mahdollistavat paketit voidaan asentaa suoraan Pip:n avulla. Näitä paketteja ovat ainakin Scikit-learn, Numpy, Pandas ja Pandastable. Jos ohjelmaa komentokehotteesta käynnistettäessä paketteja puuttuu, tulee näkyviin virheilmoitus, jossa näkyy puuttuvien pakettien nimet.

Tämän työn tapauksessa Pythonilla toteutetun ohjelman ajaminen tehdään suoraan komentokehotteesta, eikä käytössä ole yksinkertaisuuden vuoksi esimerkiksi graafista JupyterLab-ympäristöä. Pythonilla toteutetun ohjelman ajaminen komentokehotteesta tapahtuu kirjoittamalla siihen *python ohjelman_nimi.py* ja painamalla Enter. *Ohjelman_nimi.py* täytyy käyttäjän korvata suoritettavan (tulkattavan) tiedoston nimellä. Päätös olla käyttämättä esim. JupyterLabia johtuu myös siitä, että tämän työn tuloksena olevaa ohjelmaa tullaan sen lopullisessa käyttöympäristössä ajamaan exe-tiedostoksi (suoritettava ohjelma) käännettynä. Tämä mahdollistaa sen, että exe-tiedoston kuvaketta klikkaamalla aukeaa suoraan ohjelman graafinen käyttöliittymä.

3.2 GPU-tuki vs. ohjelman ajaminen pelkällä CPU:lla

Alusta lähtien tavoitteena oli Prometecin vaatimuksesta toteuttaa ohjelma, joka toimii Windows 10 - tai Windows 11 -ympäristössä. Näissä käyttöjärjestelmissä olisi myös ollut mahdollista ottaa käyttöön GPU-laitteistokiihdytys neuroverkon opettamisessa. GPU-laitteistokiihdytyksen mahdollistavia lisäkirjastoja ei kuitenkaan Prometecin pyynnöstä tarvinnut asentaa ohjelman suoritusympäristön pitämiseksi mahdollisimman yksinkertaisena. Tämän vuoksi kaikki neuroverkkoihin liittyvä laskenta on tehty CPU:lla, mikä tarkoittaa, että suoritusnopeus on parhaassakin tapauksessa ollut huomattavasti hitaampi kuin jos GPU-laitteistokiihdytys olisi ollut käytettävissä.

Vaikka koodin ajaminen pelkällä CPU:lla hidastaa neuroverkon opettamista GPU:lla ajamiseen verrattuna, asia ei kuitenkaan vaikuta opetuksen laatuun. Toisin sanoen tulokset lopussa ovat samat molemmissa tapauksissa.

3.3 Vaihtoehdot neuroverkkokirjastoksi

Alussa tässä opinnäytetyössä käytettäväksi neuroverkkokirjastoksi oli ehdolla kolme eri vaihtoehtoa: Pytorch, Scikit-learn ja Tensorflow. Näistä Tensorflow on pääasiassa Googlen kehittämä ja Pytorch omaa nimeään kantavan Linux Foundationin alaisuudessa toimivan säätiön (aiemmin Pytorchia kehittivät aluksi Facebook ja sitten Meta AI) [7][8].

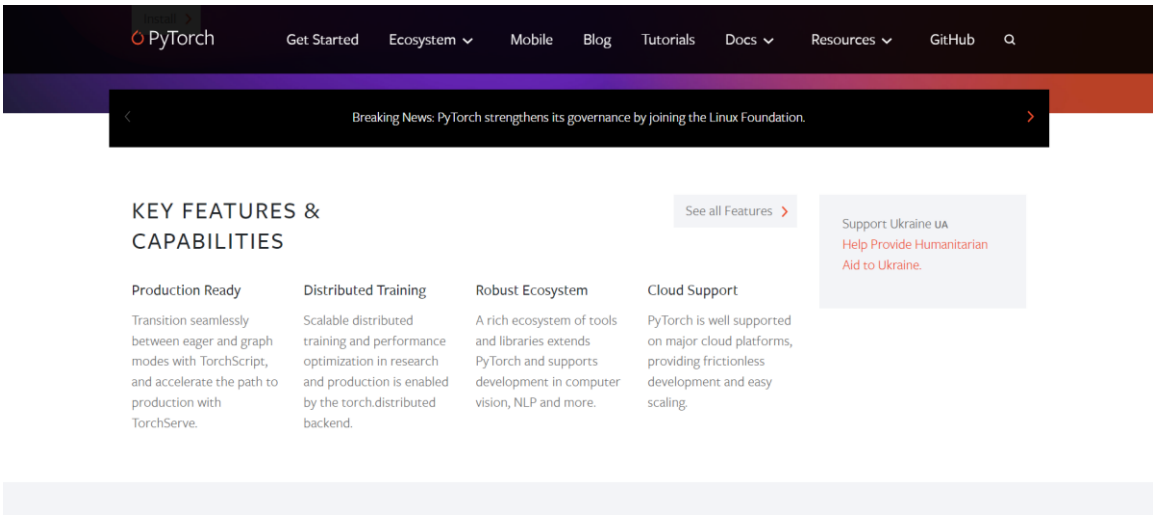
Scikit-learn on alun perin luotu vuonna 2007 Google Summer of Code -projektina. Tämän jälkeen projekti on laajentunut ja on tällä hetkellä kansainvälisen ohjelmistokehittäjien yhteisön kehittämä ja ylläpitämä. [26].

Aluksi kaikilla näillä kolmella neuroverkkokirjastolla luotiin esimerkkiohjelma, joka luki sisään jokaisen ohjelman kohdalla saman datasetin CSV-formaatissa olevasta tiedostosta. Sitten tuo datasetti ajettiin neuroverkon läpi ja tulokset visualisoitiin. Seuraavaksi esitellään kaikkien noiden kolmen neuroverkkokirjaston avulla saavutetut tulokset.

Jokaisen neuroverkkokirjaston tulosten käsittelyn yhteydessä on myös vertailun vuoksi kuvakaappaus kyseisen kirjaston nettisivuston etusivusta. Kuvassa 1 on kuvakaappaus Pytorchin etusivusta, kuvassa 2 kuvakaappaus Scikit-learnin etusivusta sekä kuvassa 3 kuvakaappaus Tensorflown etusivusta. Näiden lisäksi kuvassa 4 esitellään kuvakaappauksen muodossa aloittelijoille suunnattua esimerkkikoodia Tensorflown nettisivuilta.

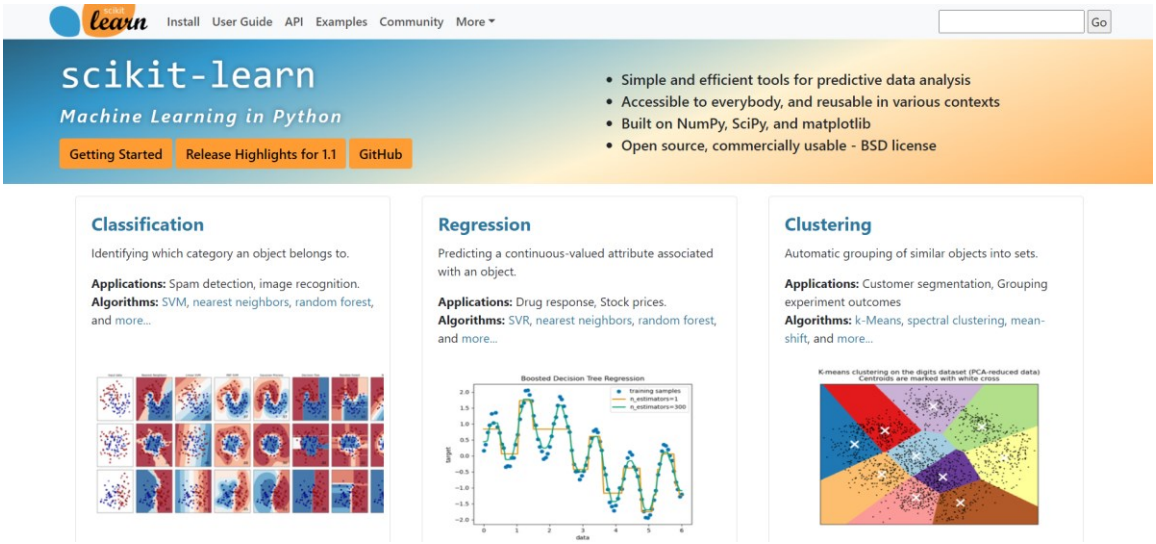
Pytorchin tapauksessa ei saatu aikaan muiden kirjastojen kanssa yhteensopivia tuloksia. Tähän oli kaksi pääasiallista syytä. Ensinnäkin Pytorchin tapauksessa neuroverkon opettamiseen käytetty silmukka piti toteuttaa kokonaan itse. Tässä ilmeni alun jälkeen huomattavia vaikeuksia varsinkin sen suhteen, että opetus- ja validointijoukossa mitatut häviöt olisi saatu vertailukelpoisiksi muiden testattujen neuroverkkokirjastojen kanssa. Toiseksi koska tässä opinnäytetyössä toteutetun tutkimuksen tapauksessa neuroverkon ulostulon pitää olla tyypiltään jatkuva (ei-diskreetti) ja koska koulun puolesta annettu esimerkkikoodi olettaa ulostulon olevan diskreetti, olisi tämän opinnäytetyön tekijälle jäänyt liikaa työtä, jos jatkuvan ulostulon kanssa mitatut häviöt opetus- ja validointijoukossa olisi pitänyt saada näkymään oikein. Näin ollen Pytorchiin perehtyminen jätettiin kesken ja päätettiin keskittyä kahteen muuhun tutkittavana olevaan neuroverkkokirjastoon.

Edellä mainitut seikat huomioon ottaen tekijä huomauttaa Pytorchin olevan ainakin nykyisessä muodossaan liian vaikea käytettäväksi suoraan tämäntyyppisissä ja -laajuisissa projekteissa.



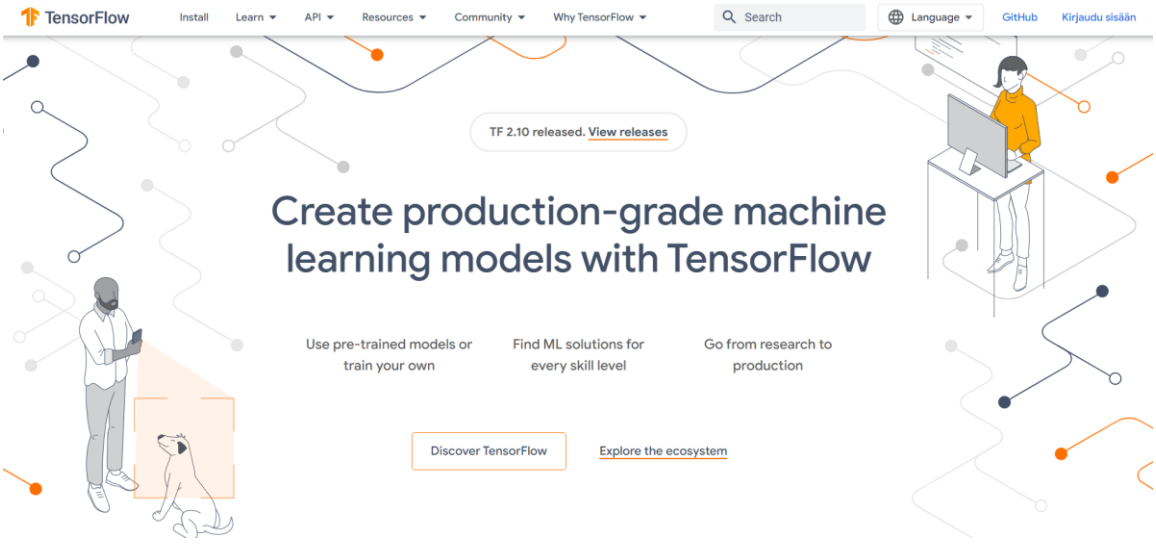
Kuva 1. Kuvakaappaus Pytorchin nettisivuston etusivusta [27], jossa esitellään kyseisen neuroverkkokirjaston tärkeimpiä ominaisuuksia lyhyessä muodossa.

Scikit-learn oli puolestaan varsin näppärä ottaa käyttöön. Tämä johtui pitkälti siitä, että Scikit-learnissä oli valmiiksi metodi jatkuvan ulostulon omaavan neuroverkon opettamiseen. Myös häviö opetusjoukossa oli helppo ottaa talteen jokaisen epookin tapauksessa ja lopuksi visualisoida.



Kuva 2. Kuvakaappaus Scikit-learnin nettisivuston etusivusta [28], jossa on selvästi käytetty enemmän värejä kuin Pytorchin ja Tensorflown etusivujen tapauksessa. Myös neuroverkkokirjaston rakentamiseen käytetyt alemman tason kirjastot sekä lisenssi ovat näkyvissä heti etusivulla.

Tensorflow oli myös yllättävän helppo ottaa käyttöön. Siinäkin oli valmis metodi tässä työssä tarvittavan neuroverkon opettamiseen. Tuosta metodista oli myös mahdollista saada suoraan ulos häviöt opetus- ja validointijoukossa jokaisella epookilla visualisoimista varten.



Kuva 3. Kuvakaappaus Tensorflown nettisivuston etusivusta [29], joka on perusajatukseltaan varsin samanlainen Pytorchin etusivun kanssa.

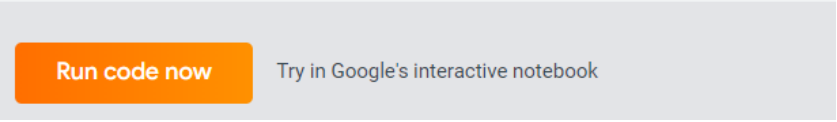
```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```



Kuva 4. Aloittelijoille tarkoitettu koodiesimerkki Tensorflown nettisivuilta [30] kuvakaappauksen muodossa. Koodiesimerkin tarkoituksena on havainnollistaa, kuinka helppoa Tensorflow on ottaa käyttöön ja oppia tekemään sen avulla omia, toimivia neuroverkkoja. Oltaessa Tensorflown nettisivuilla painamalla oranssia nappia koodin voi ajaa Google Colab -ympäristössä (linkki ei toimi suoraan tästä opinnäytetyöstä).

3.4 Sopivan neuroverkkokirjaston valinta

Kun kahta esitestaamisen jälkeen jäljelle jäänyttä neuroverkkokirjastoa, Scikit-learniä ja Tensorflowta, verrattiin toisiinsa, havaittiin Scikit-learn loppujen lopuksi paremmaksi vaihtoehdoksi tämän työn tarkoitukseen. Tälle on perusteena pääasiassa, että koska datasetti sisältää kuitenkin alle 10 000 riviä dataa, ei ole tarpeen käyttää suurista dataseteistä hyvin suoriutuvaa mutta muuten laitteistoresurssien kannalta raskasta Tensorflowta.

Tensorflown hyväksi puoleksi mainitaan, että tämän työn tekijä on tutustunut siihen Kajaanin ammattikorkeakoulun kurssien ja projektiopintojen aikana. Tästä huolimatta tekijä kokee, että Scikit-learnin neuroverkkoa opettava metodi oli todella helppo ottaa käyttöön. Näin ollen Scikit-learn on myös vaivattomampi ainakin käyttöönoton ja peruskäytön kannalta kuin Tensorflow, jonka oppiminen vaati kuitenkin kurssien käymistä ja aiheeseen tutustumista projektiopintojen aikana.

Kun työn toteuttamisessa käytettäväksi neuroverkkokirjastoksi oli valittu Scikit-learn, lopetettiin esimerkkiohjelman kehittäminen kahdella muulla neuroverkkokirjastolla ja keskityttiin lopun aikaa pelkästään Scikit-learnin hyödyntämiseen tutkimusongelman ratkaisemisessa. Näin toimittiin työn laajuuden pitämiseksi hallinnassa.

3.5 Neuroverkkokirjastojen lisensoinnista

Toimeksiantajan pyynnöstä myös tässä työssä tutkittavien neuroverkkokirjastojen käyttämät lisenssit täytyy kartoittaa ja esittää selkeässä muodossa. Tämä johtuu siitä, että lisenssi saattaa rajoittaa ohjelmiston (tässä tapauksessa neuroverkkokirjaston) käyttöä yrityskäytössä. Seuraavaksi esitellään jokaisen Luvussa 3.3 esitellyn neuroverkkokirjaston lisenssiehdot.

Pytorch käyttää 3-osaista BSD-lisenssiä. Sen mukaan Pytorchia käyttävien, lähdekoodina levitettävien ohjelmistojen täytyy kuljettaa mukanaan alkuperäistä Pytorchin lisenssitiedostoa, joka sisältää tiedot vuosista, joina Pytorchia on kehitetty sekä kunakin aikana kehitystä tehneestä tahosta. Edellisessä lauseessa mainitut ehdot pätevät myös silloin, kun Pytorchia käyttävää ohjelmistoa levitetään binaarimuodossa. Lisäksi lisenssi kieltää Pytorchin avulla tehtyjen tuotteiden (ohjelmistot) mainostamisen käyttäen tekijänoikeuden omistajan tai kehittäjien nimeä/nimiä ilman kirjallista lupaa. [31.]

Tensorflown käyttö perustuu Apache-lisenssin versioon 2.0. Apache 2.0 -lisenssi on muuten samanlainen kuin 3-osainen BSD-lisenssi, mutta siitä puuttuu viimeinen, ohjelmistojen mainostamisen tekijänoikeuden omistajan tai kehittäjien nimeä/nimiä käyttäen kieltävä osio. [32.]

Scikit-learn käyttää Pytorchin tavoin 3-osaista BSD-lisenssiä [33]. Ainoastaan tällä on tämän työn kannalta enemmän merkitystä, koska vain Scikit-learnillä tehty neuroverkko-ohjelmisto jää toimeksiantajan käyttöön tämän opinnäytetyön valmistumisen jälkeen.

4 Tässä työssä käytetty neuroverkko ja sen parametrit

Sen jälkeen, kun työssä valittiin käytettäväksi neuroverkkokirjastona pelkästään Scikit-learn, oli myös selvää, minkä tyyppistä neuroverkkoa tullaan käyttämään. Tämä johtuu siitä, että Scikit-learnissä ei pysty itse vaikuttamaan käytettävän neuroverkon tyyppiin, vaan pelkästään neuronien määrään jokaista piilokerrosta kohti, oppimisnopeuteen, epookkien määrään sekä muihin suoraan neuroverkon tyypistä riippumattomiin parametreihin. Näin ollen tässä työssä oli käytössä tavallinen yhden piilokerroksen sisältävä perseptroni, kuten seuraavissa kappaleissa tarkemmin selitetään.

Työssä käytetään Scikit-learnin MLPRegressor-metodia, joka tuottaa tässä työssä vaaditun jatkuvan ulostulon [34]. Neuroverkon opettamisen jälkeen sen avulla tehdään ennuste valmista predict-metodia käyttäen. Ennusteen tekemisessä käytetään testaus-/validointijoukon dataa. MLPRegressor-metodilla on monia parametreja, joista vain osa on tämän työn kannalta oleellisia. Ne parametrit, joita ei tarvitse tässä työssä säätää, on säilytetty oletusarvossa. Huomionarvoista on, että virhefunktiota ei ole ollenkaan mahdollista valita MLPRegressorin parametreistä, vaan virhefunktio on vakiona neliöllinen keskivirhe (MSE). Tässä työssä graafisen käyttöliittymän kautta säädettävissä olevat parametrit ja niiden perustiedot on esitetty Taulukossa 1. Taulukon jälkeen tulevissa kappaleissa esitellään jokainen noista parametreista tarkemmin. Lueteltuina ovat myös ne arvot, joihin kunkin parametrin voi MLPRegressorin tapauksessa asettaa.

Parametrin nimi	**Mahdolliset arvot**	**Oletusarvo**	**Mahdolliset optimointialgoritmit (Kaikki = lbfgs, stokastinen gradient descent, adam)**
Neuronien määrä piilokerrosta kohti	Tuple-tietotyyppi, pituus 0 – (kerrosten lukumäärä - 2)	(100,)	Kaikki

Aktivaatiofunktio	Identiteettifunktio, logistinen funktio, tanh, ReLU	ReLU	Kaikki
Optimointialgoritmi	Lbfgs, stokastinen gradient descent, adam	Adam	-
Oppimisnopeuden tyyppi	Constant, invsca-ling, adaptive	Constant	Stokastinen gradient descent
Oletusoppimisnopeus	Kaikki positiiviset reaaliluvut	0.001	Stokastinen gradient descent, adam
Epookkien tai gra-dienttiaskelten määrä	Kaikki positiiviset kokonaisluvut	200	Kaikki

Taulukko 1. Säädettävän parametrin nimi, sen saamat mahdolliset arvot, Scikit-learnin sille määräämä oletusarvo sekä ne optimointialgoritmit, joiden kanssa kyseisen parametrin käyttäminen on mahdollista.

Ensimmäinen säädettävä parametri on neuronien määrä piilokerrosta kohti. Se voi MLPRegressorin määritelmän mukaan saada tuple-tietotyypin arvoja pituudelta 0 – (kerrosten lukumäärä - 2). Esimerkiksi tuplella (10, 5, 6) MLPRegressorin luoma neuroverkko sisältää 3 piilokerrosta, joista ensimmäisessä on 10, toisessa 5 ja kolmannessa 6 neuronia. Nollan asettaminen arvoksi ei ole suositeltavaa, koska silloin neuroverkko sisältää vain sisään- ja ulostulokerroksen eikä käsittele dataa millään tavalla. Käytännössä voidaan sanoa, että mitä suurempia tämän parametrin arvot jokaista piilokerrosta kohti ovat, sitä laskennallisesti vaativampi neuroverkko on opettamisen suhteen ja sitä tarkempia tulokset todennäköisesti ovat. Oletusarvo tälle parametrille on (100,). Myöhemmin tässä työssä neuroverkkoa testattaessa tämä parametri on säilytetty oletusarvossa, jolloin käytössä on ollut vain yksi piilokerros, jossa on 100 neuronia. Myöhemmin tässä työssä esiteltävän graafisen käyttöliittymän avulla on mahdollista luoda vain sellainen neuroverkko,

jossa on yksi piilokerros. Tällöin käytännössä tässä työssä toteutetussa ohjelmassa ainoa parametri, jota on mahdollista säätää piilokerrosten suhteen, on neuronien määrä yhdessä piilokerroksessa.

Toinen säädettävä parametri on aktivaatiofunktio. Tälle vaihtoehtoja ovat identiteettifunktio (funktio, joka palauttaa saman arvon kuin sille syötetään), logistinen funktio, tanh ja ReLU. Oletuksena on yleisimmissä tapauksissa hyvin toimiva vaihtoehto eli ReLU.

Sitten tulee optimointialgoritmi. Vaihtoehtoja tälle ovat lbfgs (kvasinewtonilaisten menetelmien joukkoon kuuluva optimointialgoritmi), stokastinen gradient descent ja adam. Scikit-learnin dokumentaatiossa MLPRegressor-metodille kerrotaan, että vähintään tuhansien näytteiden dataseteillä adam on paras optimointialgoritmi, mutta tuota pienemmillä dataseteillä lbfgs voi supeta nopeammin ja toimia tehokkaammin. Oletusarvo optimointialgoritmiksi on adam.

Oppimisnopeuden tyyppi määrittää sen, muutetaanko oppimisnopeutta opetuksen aikana ja jos muutetaan, millä tavalla. Tässä tapauksessa oppimisnopeutta voidaan ylipäänsä muuttaa vain silloin, kun optimointialgoritmina on stokastinen gradient descent (sgd). Oletusarvo on constant, joka tarkoittaa, että seuraavassa kappaleessa määritelty parametri määrää oppimisnopeuden vakioksi koko opetuksen ajaksi. Muita vaihtoehtoja ovat invscaling (oppimisnopeuden asteittainen pienentäminen käyttäen käänteistä skaalauseksponenttia) sekä adaptive (Oppimisnopeus pidetään vakiona siihen asti, kunnes opetushäviö lopettaa pienenemisen tiettyjen ehtojen mukaan. Tuon tapahtuessa oppimisnopeus jaetaan luvulla 5).

Oletusoppimisnopeus (learning_rate_init) määrää aluksi käytettävän oppimisnopeuden. Tuo oppimisnopeus myös pysyy vakiona, jos siihen ei tehdä muutoksia opettamisen aikana edellisessä kappaleessa kuvatuilla tavoilla. Tätä parametria käytetään vain, jos optimointialgoritmi on sgd tai adam ja sen oletusarvo on 0.001.

Epookkien (iteraatioiden) lukumäärä määrää stokastisten optimointialgoritmien (sgd ja adam) tapauksessa, kuinka monta kertaa kutakin näytettä käytetään opettamiseen. Lbfgs-optimointialgoritmin tapauksessa tämä luku määrää vastaavasti gradienttiaskelten määrän. Neuroverkon opettamista jatketaan niin kauan, kunnes tämä luku saavutetaan tai kunnes optimointialgoritmi suppenee. Oletusarvo on 200.

5 Datasetti ja sen esikäsittely

Työssä käytettävä data on tallennettuna CSV-muotoiseen tiedostoon, josta se luetaan jokaisella ohjelman suorituskerralla ja esikäsitellään ennen neuroverkolle syöttämistä. Datasetin lukemisessa CSV-tiedostosta ja datasetin esikäsittelyssä käytetään Pandas- ja NumPy-kirjastoja [35][36]. Datasetin pituus ennen esikäsittelyä on noin 7 900 riviä. Datasetin pituus esikäsittelyn jälkeen riippuu valituista sarakkeista, koska eri sarakkeissa on eri määrä poistamista edellyttäviä rivejä. Tämän työn tekemisen alkuvaiheessa valittujen sarakkeiden tapauksessa esikäsittelyn jälkeen datasetin pituus lyheni noin 7 800 riviin. Yleisesti voidaan sanoa, että tässä työssä käytössä olevan datasetin tapauksessa datasetti lyhenee noin 0–500 riviä esikäsittelyn jälkeen valittujen sarakkeiden mukaan.

Datasetin sisältö on luokiteltu salassa pidettäväksi, joten sitä ei voida käydä tarkemmin läpi tässä työssä. Se voidaan kuitenkin mainita, että jokainen datasetin rivi sisältää aikaleiman ja joukon muita nimeltä mainitsemattomia sarakkeita. Datasetissä on yhteensä 22 saraketta, joista periaatteessa mitkä tahansa 20 voidaan valita opetusdatan featureiksi ja loput kaksi target-vektorin pohjaksi. Suurimmat datasettiä koskevat epävarmuudet liittyvät seuraavassa kappaleessa esiteltyyn esikäsittelyn ensimmäiseen vaiheeseen. Tämä tarkoittaa, että jos jossakin sarakkeessa on paljon poistettavaksi määritettyjä arvoja, datasetin pituus (rivien määrä) voi lyhentyä esikäsittelyn jälkeen niin paljon, että se haittaa neuroverkon suorituskykyä. Toinen datasettiä koskeva epävarmuustekijä ovat hyvin lähellä nollaa olevat arvot, jotka ovat potentiaalisesti mittausvirheestä aiheutuvia. Koko tämän työn tekemisen aikana käytetty datasetti on samasta paikasta ja se on mitattu samoilla mittalaitteilla. Tulokset ovat yleistettävissä melko hyvin muilta osin, paitsi vuodenajan mukaan. Tämän takia päivämäärästä erotetaan aina kuukausi erilleen, kuten seuraavissa kappaleissa on kuvattu.

Datasetin esikäsittelyn ensimmäinen vaihe sisältää tässä tapauksessa niiden rivien poistamisen, joissa esiintyy 0- tai None-arvoja tai joissa jokin lukuarvo on suurempi kuin 100 000. Esikäsittelyn toisessa vaiheessa valitaan alkuperäisestä datasetistä käyttäjän graafisen käyttöliittymän kautta valitsemat sarakkeet neuroverkon opetusmatriisiin ja luodaan target-vektori, joka sisältää ne arvot, joita neuroverkon tulisi oppia ennustamaan.

Esikäsittelyn kolmannessa vaiheessa aikaleimasta (jos se on valittuna käytettävien sarakkeiden joukkoon) erotetaan kuukausi ja luodaan uusi vektori, joka sisältää tiedon kuukaudesta (vektorin rivien arvo on aina välillä [1,12]). Tämä uusi vektori liitetään osaksi opetusmatriisia ja alkuperäinen aikaleimasarake poistetaan. Tämä vaihe tehdään riippumatta siitä, muutetaanko tässä vaiheessa luotu vektori myöhemmin binaarivektoriksi.

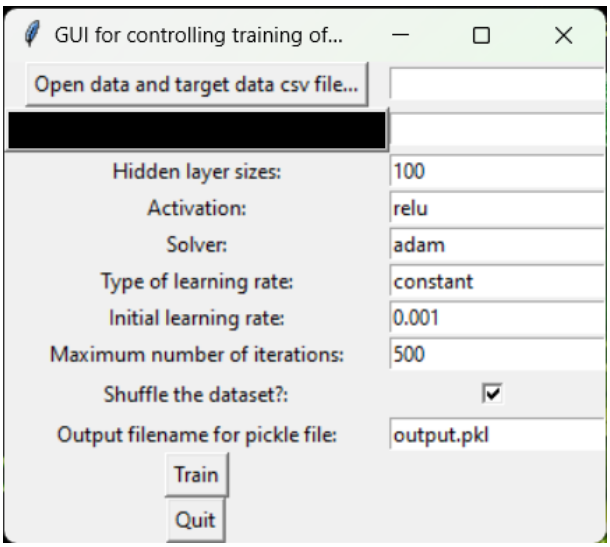
Esikäsittelyn neljäs ja viimeinen vaihe käsittää käyttäjän määrittämien opetusmatriisin sarakkeiden muuttamisen binaarivektoreiksi (vektori, jossa yksi arvo on arvossa 1 ja kaikki muut arvot arvossa 0) rivi kerrallaan. Aluksi tämän työn tekemisen aikana yksittäisen binaarivektorin alkioiden määrä pohjautui käsiteltävässä sarakkeessa olevien toisistaan eroavien arvojen lukumäärään. Kuitenkin myöhemmin toimeksiantajan pyynnöstä toiminnallisuutta muutettiin siten, että binaarivektorin alkioiden määrän määrittää sisään tulevan sarakkeen suurimman yksittäisen arvon lukuarvo. Muunnoksen jälkeen binaarivektoreista koostuvat matriisit lisätään opetusmatriisiin ja lopuksi alkuperäiset sarakkeet, joiden perusteella binaarivektorien matriisit luotiin, poistetaan. Edellä mainittujen toimien jälkeen datasetin leveys (yhdellä rivillä olevien sarakkeiden määrä eli neuroverkolle datana syötettävien parametrien määrä) riippuu siitä, kuinka monta saraketta on päätetty muuttaa binaarivektoriksi sekä siitä, kuinka suuri on ollut suurin yksittäinen lukuarvo jokaisessa käsiteltävässä sarakkeessa.

Datasetti jaetaan ennen sen syöttämistä neuroverkolle opetus- ja validointijoukkoihin. Tämä tehdään Scikit-learnin train_test_split-metodin avulla käyttäen automaattisesti määrättyä jakosuhdetta, jonka mukaan ¾ datasta menee opetusjoukkoon ja loput testaus-/validointijoukkoon [37]. Lisäparametreina ovat satunnaislukugeneraattorille asetettu siemenarvo 1 sekä oletuksena valinta siitä, että datasetti sekoitetaan ennen sen jakamista opetus- ja validointijoukkoihin. Valintaa datasetin sekoittamisesta tässä vaiheessa voi käyttäjä muuttaa myöhemmin esiteltävän graafisen käyttöliittymän kautta.

6 Graafinen käyttöliittymä ja tuloksen tallentaminen Pickle-tiedostoon

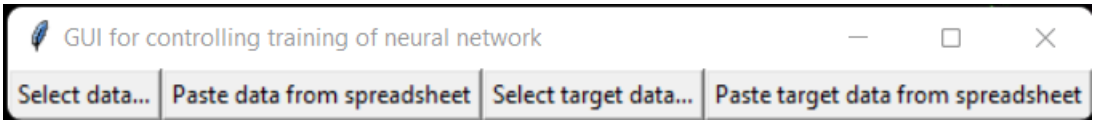
6.1 Graafisen käyttöliittymän toiminnallisuudet

Opinnäytetyön tekijälle annettiin tehtäväksi tehdä myös yksinkertainen graafinen käyttöliittymä (Graphical User Interface, GUI), jonka kautta pystytään muuttamaan monia erilaisia neuroverkon opettamiseen vaikuttavia parametreja ennen kuin neuroverkon opettaminen aloitetaan. Graafinen käyttöliittymä on toteutettu Pythonille portatulla TkInter-kirjastolla [38]. Jokaisella graafisen käyttöliittymän parametrilla on oletusarvo, jota käytetään, jos käyttäjä ei tee parametriin muutoksia. Graafisessa käyttöliittymässä on myös nappi, jota painamalla neuroverkon opettamisen voi käynnistää. Neuroverkon opettamisen lopetusnappia ei ole, koska graafinen käyttöliittymä menee ”jumiin” opetuksen ajaksi. Tämä johtuu siitä, että TkInter ei täysin tue ohjelman jakamista useampaan kuin yhteen säikeeseen. Koska neuroverkon opettaminen kaappaa käyttöönsä ohjelman käytettävissä olevan yhden säikeen, ei jää resursseja tarkistaa graafiseen käyttöliittymään kohdistuvia tapahtumia. Kun neuroverkon opettaminen on valmistunut ja kaikki ohjelman avaamat grafiikkaikkunat on suljettu, graafinen käyttöliittymä toimii taas normaalisti. Koko ohjelman lopettamista varten graafisessa käyttöliittymässä on myös erillinen nappi. Graafinen käyttöliittymä on englanninkielinen. Kuvassa 5 on graafisen käyttöliittymän pääikkunan alkutila.



Kuva 5. Graafisen käyttöliittymän pääikkuna, josta salaisiksi luokitellut tiedot on peitetty.

Graafisen käyttöliittymän ylin painike mahdollistaa ohjelman sisään tulevana datana käyttämän CSV-tiedoston valitsemisen. Nappia painamalla avautuu ensin tiedostonvalintaikkuna, josta käyttäjä voi valita avattavan tiedoston. Kun tiedosto on valittu ja käyttäjä painaa tiedostonavaamisnappia, avautuu erillinen uusi datanvalintaikkuna, jonka nappeja painamalla käyttäjä voi valita ne sarakkeet, joita käytetään neuroverkon sisään tulevana datana ja neuroverkon target-vektorin pohjana. Tämä ikkuna on esitetty Kuvassa 6.



Kuva 6. Datanvalintaikkuna silloin, kun dataa ei ole vielä valittuna.

Neuroverkolle syötettävä data valitaan Select data… -napista avautuvasta taulukkolaskentaikkunasta, jossa on avattuna edellä mainitun tiedostonvalintaikkunan avulla valittu CSV-tiedosto. Tuosta ikkunasta käyttäjä voi valita yhden tai useamman sarakkeen klikkaamalla sarakkeen otsikkoa (useampaa saraketta valittaessa pitää painaa Ctrl- tai Shift-näppäin pohjaan ennen otsikon klikkaamista). Valitut sarakkeet saa kopioitua leikepöydälle painamalla jonkin datariveistä kohdalla hiiren oikeaa nappia ja valitsemalla avautuvasta valikosta Copy. Kun data on leikepöydällä, se voidaan tuoda ohjelmaan painamalla yksi askel ennen taulukkolaskentaikkunaa avautuneesta ikkunasta (datanvalintaikkuna) Paste data from spreadsheet -nappia. Tämän jälkeen datanvalintaikkunan käyttöliittymä päivittyy: näkyviin ilmestyvät valittujen sarakkeiden otsikot ja jokaisen niistä kohdalle valintaruutu, joka määrittää, muutetaanko kyseinen sarake binaarivektoriksi ennen neuroverkolle syöttämistä.

Target-vektorin data valitaan samaan tapaan kuin neuroverkolle syötettävä data, mutta tässä tapauksessa painetaan Select target data… -painiketta. Tällöin avautuu samanlainen taulukkolaskentaikkuna kuin aiemmassakin tapauksessa. Tuosta ikkunasta voidaan valita kaksi saraketta, joiden perusteella muodostetaan neuroverkon target-vektori. Tämän ohjelman tapauksessa vain tietyt kaksi saraketta ovat sopivia jatkokäyttöä ajatellen (varsinainen target-vektori muodostetaan vähentämällä noiden sarakkeiden arvot toisistaan). Ohjelma esittää virheilmoituksen, jos valitut sarakkeet eivät ole juuri oikeat. Kun sopivat sarakkeet on valittu, ne pitää kopioida leikepöydälle edellisessä kappaleessa kuvatulla tavalla. Sarakkeet saa liitettyä ohjelman käyttöön painamalla Paste target data from spreadsheet -painiketta. Tämän jälkeen valittujen sarakkeiden nimet

ilmestyvät näkyviin datanvalintaikkunan alalaitaan. Kuvassa 7 on datanvalintaikkuna silloin, kun valittuna on neljä saraketta neuroverkon opettamiseen sekä kaksi sopivaa saraketta target-vektorin muodostamista varten.

Kuva 7. Datanvalintaikkuna silloin, kun kaikki neuroverkon opettamisen kannalta välttämätön data on valittu.

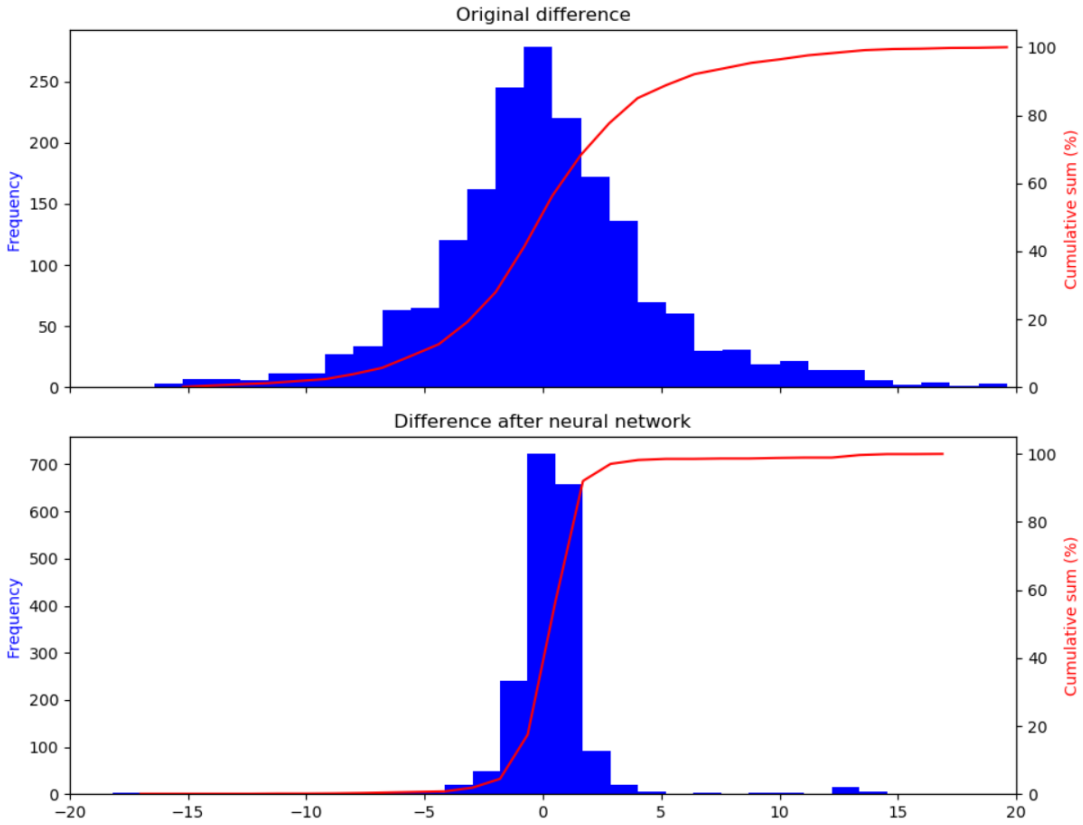
Kahdessa edellisessä kappaleessa mainittujen toimenpiteiden jälkeen ohjelma on valmis opettamaan neuroverkkoa. Ennen opettamisen aloittamista kannattaa vielä varmistaa, että kaikki ensimmäisenä ohjelman käynnistämisen jälkeen avautuneessa ikkunassa olevat parametrit ovat halutuissa arvoissa.

6.2 Tulokset ja niiden tulkinta

Neuroverkon opettamisen jälkeen ohjelma esittää saavutetut tulokset graafisessa muodossa käyttäen Matplotlib-kirjaston [39] erillisiä grafiikkaikkunoita. Ensimmäisessä näistä ikkunoista esitetään histogrammien muodossa validointijoukon target-vektorin arvojen jakauma sekä neuroverkon validointijoukossa saavuttamien ennusteiden jakauma. Samassa grafiikkaikkunassa esitetään myös arvojen prosentuaalinen kertymä erillisen käyrän avulla. Toisessa ikkunassa esitetään neuroverkon opetushäviö epookkien määrän funktiona. Seuraavaksi molemmat grafiikkaikkunat esitellään kuvakaappausten muodossa kahdella eri epookkien määrällä ja tuloksille esitetään myös tulkintaohjeet. Tämän jälkeen esitellään vielä graafisen käyttöliittymän pääikkunan päivitetty lopputila, jossa näkyy saavutettujen tulosten tarkkuus asetettuun tavoitetasoon nähden.

Kuvassa 8 esitetään datapisteiden jakauma validointijoukossa ilman neuroverkkoa ja neuroverkon jälkeen histogrammien sekä prosentuaalisten kertymien muodossa silloin, kun epookkien

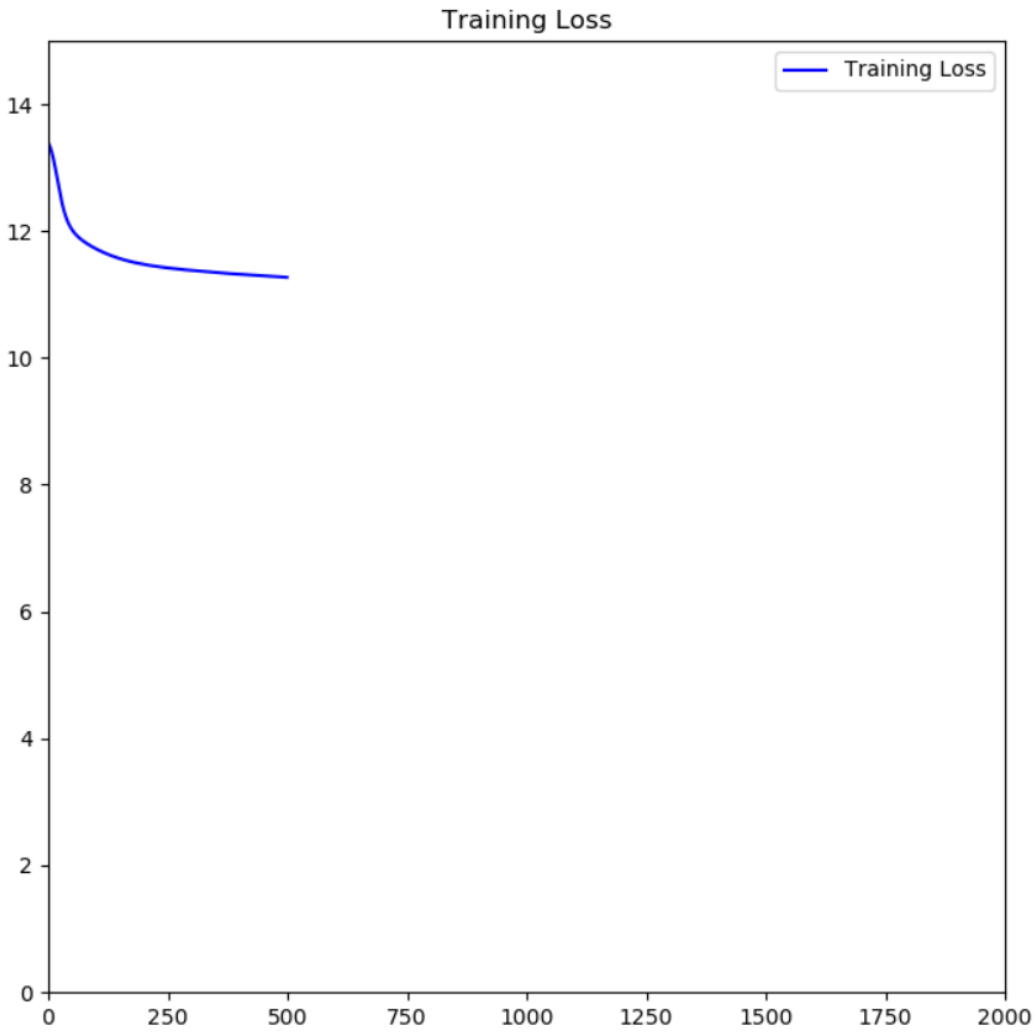
määrä on ollut 500. Ylemmässä kuvaajassa näkyy datapisteiden jakauma validointijoukossa (validointijoukon target-vektori) ilman, että dataa on ajettu lainkaan neuroverkon läpi. Alemmassa kuvaajassa näkyy neuroverkon tekemien ennusteiden jakauma validointijoukon datan pohjalta. Neuroverkko tekee ennusteensa validointijoukossa sen perusteella, mitä se on oppinut opetusjoukon datan pohjalta. Kuvaajia vertaamalla huomataan, että neuroverkon tekemät ennusteet ovat keskittyneet huomattavan paljon lähemmäs x-akselin nollakohtaa kuin alkuperäiset validointijoukon target-vektorin arvot. Johdannossa esitellyn neuroverkon toiminnan tavoitetason huomioon ottaen edellä mainittu ero kuvaajissa on toivottava.



Kuva 8. Validointijoukon target-vektorin arvojen ja neuroverkon ulostulona tuottamien arvojen jakaumien vertailu 500 epookilla.

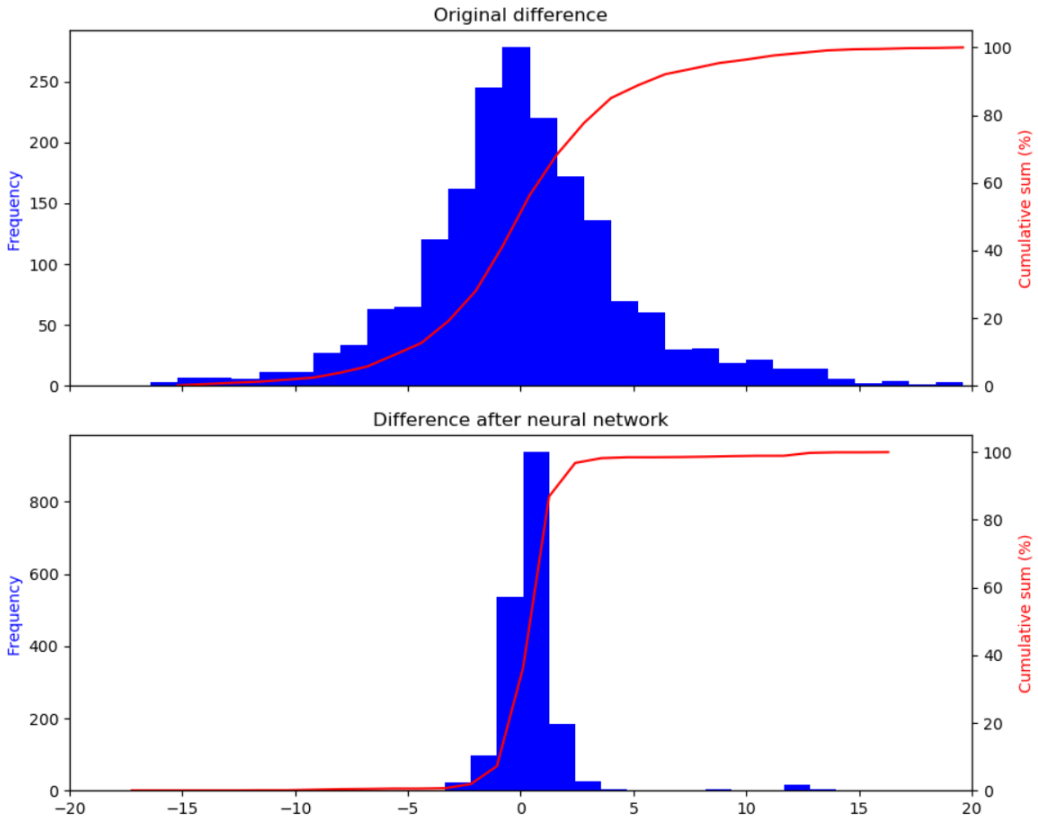
Kuvassa 9 on esitettynä neuroverkon opetushäviön kehittyminen epookkien määrän funktiona, kun epookkien maksimimäärä on ollut 500 ja häviöfunktiona on käytetty neliöllistä keskivirhettä (MSE). X-akselin pituus on 2 000, jotta kuvaaja olisi suoraan vertailukelpoinen 2 000 epookilla

tehdyn vastaavan kuvaajan kanssa. Kuvaajasta huomataan, että opetushäviö jatkaa pienenemistä edelleen silloin, kun neuroverkon opettaminen loppuu (epookkien maksimimäärä on saavutettu). Tämän tuloksen saamiseksi neuroverkolle syötettävät sarakkeet piti valita tarkoituksellisen tarkasti.



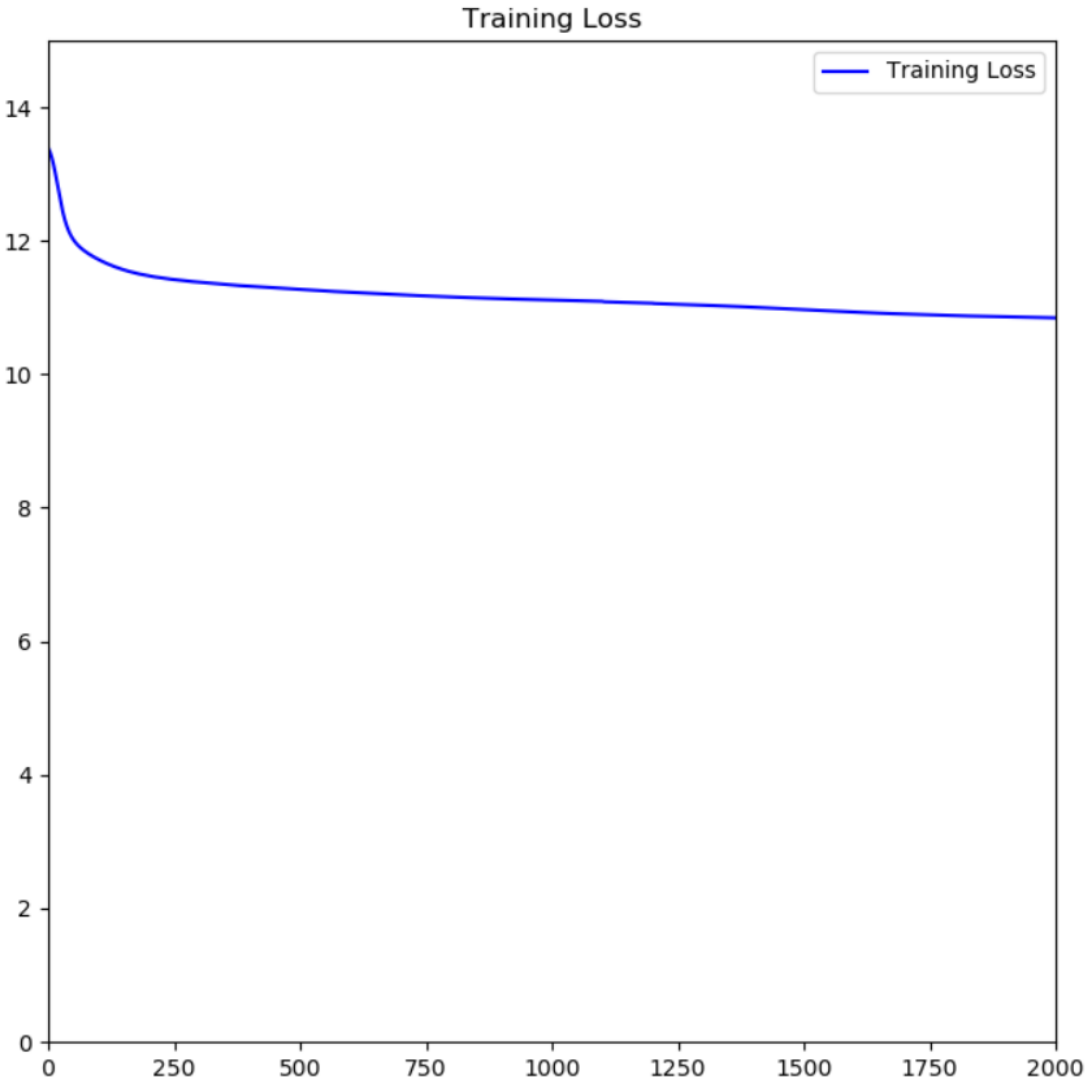
Kuva 9. Neuroverkon opetushäviön kehittyminen epookkien määrän funktiona, kun epookkien määrä on ollut 500.

Kuvassa 10 on datapisteiden jakauma validointijoukossa ilman neuroverkkoa ja neuroverkon jälkeen histogrammien sekä prosentuaalisten kertymien muodossa silloin, kun epookkien määrä on ollut 2 000 ja käytettävät sarakkeet ovat olleet samat kuin edellisissä kuvaajissa. Kuvaajista nähdään, että nyt neuroverkon tuottaman jakauman huippu on hieman kapeampi kuin äsken.



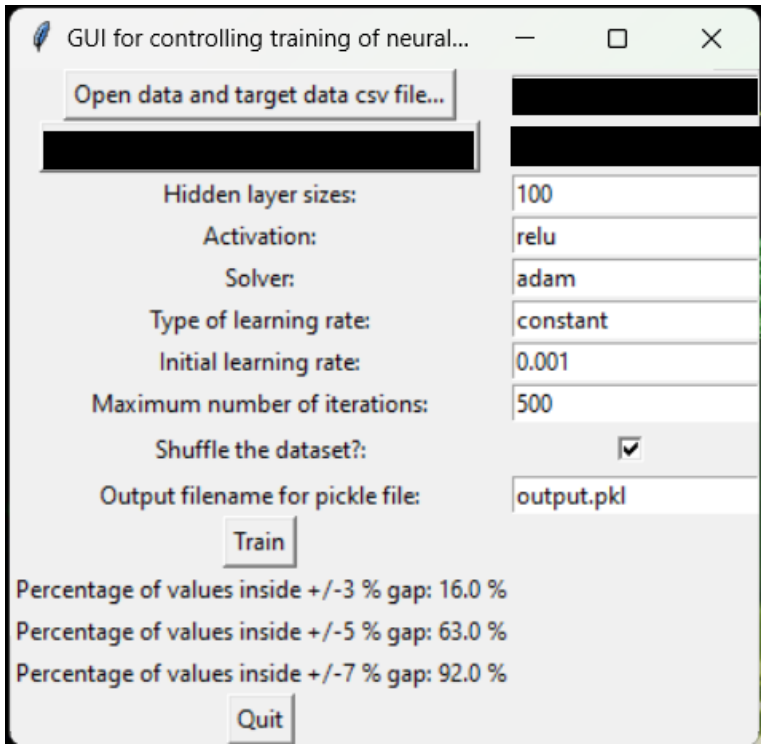
Kuva 10. Validointijoukon target-vektorin arvojen ja neuroverkon ulostulona tuottamien arvojen jakaumien vertailu 2 000 epookilla.

Kuvassa 11 kuvataan neuroverkon opetushäviön kehittyminen epookkien määrän funktiona, kun epookkien maksimimäärä on ollut 2 000 ja käytettävät sarakkeet ovat olleet samat kuin edellä. Häviöfunktiona on käytetty tässäkin neliöllistä keskivirhettä (MSE). Kuvaajasta huomataan, että opetushäviö ei aivan ehdi saavuttaa vakioarvoa ennen epookkien määrän loppumista. Tämä tarkoittaa, että myöskään optimointialgoritmi ei ehdi supeta.



Kuva 11. Neuroverkon opetushäviön kehittyminen epookkien määrän funktiona, kun epookkien määrä on ollut 2 000.

Kuvassa 12 on graafisen käyttöliittymän pääikkunan lopputila molempien grafiikkaikkunoiden sulkemisen jälkeen. Erona saman ikkunan aiempaan tilaan on, että nyt alalaidassa Train- ja Quit-nappien välissä näkyy neuroverkon validointijoukossa tuottamien ennusteiden tarkkuus asetettuun tavoitetasoon nähden. Tarkempi määrittely asetetulle tavoitetasolle esiteltiin Johdannossa.



Kuva 12. Graafisen käyttöliittymän pääikkunan lopputila, josta salaisiksi luokitellut tiedot on peitetty.

6.3 Neuroverkon tallentaminen Pickle-tiedostoon ja jatkokäyttö

Neuroverkon opettamisen, validoinnin ja tulosten visualisoimisen jälkeen opetettu neuroverkko tallennetaan toimeksiantajan pyynnöstä vielä binaarimuotoiseen Pickle-tiedostoon neuroverkon jatkokäyttöä varten. Tämän toiminnon toteuttamiseen käytetään Pythonin kanssa toimivaa valmista Pickle-kirjastoa [40].

Pickle-tiedostosta valmiiksi opetettu neuroverkko voidaan myöhemmin lukea ja ottaa käyttöön toisessa Python-skriptissä. Näin toimittaessa säästyy aikaa ja vaivaa, koska neuroverkkoa ei tarvitse opettaa uudelleen aina, kun sitä käytetään ennusteiden tekemiseen uudella datalla.

Tämän työn tapauksessa neuroverkon jatkokäyttö tarkoittaa sen ottamista käyttöön erillisessä Linux-ympäristössä toimivassa sovelluksessa, joka on toteutettu Pythonilla. Tuo sovellus toimii siten, että se lukee levyltä käyttäjän toisen sovelluksen kautta palvelimelle lataaman valmiiksi

opetetun neuroverkon Pickle-muodossa ja käyttää tuota neuroverkkoa ennusteen tekemiseen tietokannasta luettua dataa käyttäen. Tämän jälkeen sovellus tallentaa neuroverkon tekemät ennusteet takaisin tietokantaan sopivaan tauluun.

Huolimatta siitä, että Pickle-kirjasto ei ole turvallinen käyttää sellaisen datan kanssa, jota ei etukäteen tunneta, tässä tapauksessa edellä mainittu ei kuitenkaan ole ongelma. Tämä johtuu siitä, että nyt tiedetään, että purettavana datana on aina ennestään tunnetulla datalla opetettu neuroverkko. Koska tämän työn tapauksessa neuroverkko pitää uudelleenopettaa vain harvoin, saavutetaan Pickle-tiedostoon tallennettua valmiiksi opetettua neuroverkkoa erillisessä sovelluksessa useita kertoja uudelleen käyttämällä huomattavaa etua.

7 Neuroverkon toiminnan optimoiminen

7.1 Lähtökohdat

Tämän työn neuroverkon tapauksessa sisään tuleva data määrittää oleellisella tavalla sen, kuinka hyviä tuloksia neuroverkko pystyy tuottamaan. Tämä johtuu siitä, että jos datasetistä valitaan vääränlaisia sarakkeita (sellaisia, joissa lähes kaikki arvot ovat erilaisia), neuroverkon tulokset jäävät pakostakin vaatimattomiksi. Jos taas sarakkeet valitaan oikein, neuroverkon tuottamat ennusteet ovat siinä määrin käyttökelpoisia, että niille on jatkokäyttöä. Näin ollen on pohjimmiltaan tämän työn aikana luodun ohjelman käyttäjän vastuulla, mitä sarakkeita neuroverkolle syötettäväksi valitaan. Lisärajoitteena ohjelmalle syötettäville sarakkeille on, että ne saavat sisältää otsikkoa lukuun ottamatta vain numeromuotoista dataa. Tämä johtuu siitä, että tekstimuotoisen datan muuttamista numeromuotoiseksi ei ole toteutettu ohjelmallisella tasolla.

Kokeillessaan neuroverkkoa satunnaisilla datasetin sarakkeilla tämän työn tekijä havaitsi, että sarakkeiden tultua valituiksi epäoptimaalisesti opetushäviö (häviöfunktion arvo) pienenee hyvin hitaasti saaden aikaan epätehokasta oppimista. Tämän tapahtuessa huolimatta epookkien määrän asettamisesta korkeaksi (jopa arvoon 5 000 asti) oppiminen ei ollut kovin tehokasta ja lisäksi saatiin varoitus Scikit-learniltä siitä, että optimointialgoritmi ei ole tuonkaan epookkien määrän loppuun kuluessa supennut.

7.2 Vaihtoehtoja optimointikeinoiksi

Epookkien määränä tässä työssä toteutetun ohjelman käyttöliittymässä määritetty oletusarvo 500 on riittävä testikäyttöä ajatellen, jos data on huolella valittu edellä kuvatulla tavalla. Tuon epookkien määrän aikana optimointialgoritmi ei usein ehdi supeta, mutta neuroverkon tulokset ovat jo kuitenkin melko tarkkoja. Varsinaisessa tuotantokäytössä 500 on epookkien määränä kuitenkin liian vähän, koska opetushäviön kuvaajan perusteella neuroverkko alioppii annetulla datasetillä tuolla epookkien määrällä. Alioppiminen on helppo kiertää kasvattamalla epookkien mää-

rää varsinkin tässä työssä käytetyn varsin pienen datasetin tapauksessa. Kuten Luvussa 6.2 esitetyn 2 000 epookilla tehdyn testiajon perusteella havaitaan opetushäviön kuvaajasta, tuollakaan epookkien määrällä ei saavuteta tilannetta, jossa suorituskyky olisi toimeksiantajan määrittelemän optimaalisen tason rajojen sisällä. Juuri tämän takia on syytä mainita muitakin optimointikeinoja.

Epookkien määrän suhteen jatkokehityskohteena voisi periaatteessa olla aikaisen lopettamisen toteuttaminen. Tämä tarkoittaisi sitä, että neuroverkon opettaminen lopetetaan heti, kun virhefunktion arvo ei enää pienene. Näin olisi vältettävissä mahdollinen ylisovittuminen, joka saa aikaan epätarkkoja tuloksia ennen näkemättömällä datalla. Ylisovittumisen estämistä ei kuitenkaan tämän työn puitteissa lähdetty toteuttamaan, koska se olisi laajentanut työn aihepiiriä liikaa. On myös huomionarvoista, että tämän optimointikeinon tapauksessa epookkien määrää pitäisi nostaa edelleen suuremmaksi kuin tässä työssä käytetty suurin arvo 2 000 on ollut silloin, kun datasetti on niinkin pieni kuin se tässä työssä on ollut.

Yksi tapa parantaa neuroverkon suorituskykyä on myös neuronien lukumäärän kasvattaminen piilokerroksessa. Vaikka tämä tekeekin neuroverkosta laskennallisesti raskaamman, asia ei ole tässä työssä ongelma, koska neuroverkolle syötettävää dataa on kuitenkin sen verran vähän. Tämänkin parametrin valinta on käytännössä käyttäjän vastuulla.

Optimointikeinona voidaan tässä tapauksessa pitää sitäkin, että pyritään valitsemaan graafisen käyttöliittymän taulukkolaskentaikkunan avulla mahdollisimman monta oikeanlaista saraketta neuroverkolle syötettäväksi. Jo pelkästään tällä toimella on huomattavan paljon vaikutusta neuroverkon tulosten tarkkuuteen.

Oppimisnopeuden säätäminen on periaatteessa paljon lupaava optimointikeino. Käytännössä kuitenkin sen arvon muuttaminen johtaa helposti työlääseen vertailuun eri oppimisnopeuksien välillä.

Optimointialgoritmin vaihtaminen on myös periaatteessa mahdollista. Koska kuitenkin adam on käytännössä käytettävissä olevista optimointialgoritmeista tehokkain annetulla datasetillä, ei vaihtamisesta olisi juurikaan hyötyä.

8 Tulokset

8.1 Yhteenveto opinnäytetyössä saavutetuista asioista

Tähän opinnäytetyöhön liittyvä ohjelmistokehitys on siinä määrin täyttänyt sille asetetut tavoitteet, että tuloksena on saatu toteutettua ohjelma, joka tekee niitä asioita, joita se on suunniteltu tekemään. Kirjallisena raporttina tuon ohjelman toteuttamisesta toimii tämä opinnäytetyödokumentti. Sekä ohjelman sisäinen toiminta (datan esikäsittely ja neuroverkko) että sen käyttäjälle näkyvä graafinen käyttöliittymä täyttävät toimeksiantajan niille asettamat kriteerit.

Katsottaessa tavoitteiden toteutumista toimeksiantajan neuroverkolle asettaman tavoitetason näkökulmasta, tavoitetta kokonaisuudessaan ei saatu täytettyä. Tähän oli syynä rajallinen aika, mutta myös se, että asetettu tavoite tiedettiin alun perinkin varsin haastavaksi toteuttaa käytännössä.

Taulukosta 2 huomataan, että vaikka +/- 7 %-yksikön sisään menee jo 92 % neuroverkon ennustamista arvoista, niin +/- 3 %-yksikön sisään menee kuitenkin vain 16 % noista arvoista. Tämä tarkoittaa, että kapeimmalla määritetyllä välillä arvoja on huomattavasti vähemmän kuin tavoitetason mukaan pitäisi olla. Tämä ei tämän opinnäytetyön tekijän puolesta haittaa, koska hänen tehtävänsä oli vain toteuttaa toimiva neuroverkko eikä niinkään optimoida sitä tuottamaan parhaan mahdollisen tuloksen. Neuroverkon toiminnan tarkempi optimoiminen jää toimeksiantajan tehtäväksi.

Arvoja +/- 3 %-yksikön sisällä	**Arvoja +/- 5 %-yksikön sisällä**	**Arvoja +/- 7 %-yksikön sisällä**
16,0 %	63,0 %	92,0 %

Taulukko 2. Parhaat tämän opinnäytetyön tekemisen aikana saavutetut neuroverkon tulokset validointijoukossa taulukkomuodossa esitettynä. Nämä tulokset ovat samat, jotka ovat näkyvillä Kuvan 12 graafisessa käyttöliittymässä.

Mahdollisia jatkokehityskohteita tekijän tässä opinnäytetyössä toteuttamalle ohjelmalle voisivat olla esimerkiksi, että ohjelmaa kokeiltaisiin ajaa useammalla kuin yhdellä datasetillä tai että kokeiltaisiin Scikit-learnin lisäksi jotakin toista neuroverkkokirjastoa. Jälkimmäiseen jatkokehityskohteeseen liittyen voisi olla myös mielenkiintoista ajaa käytettävän neuroverkon kanssa sen sisäistä toimintaa analysoivaa ohjelmaa, jos vain käytetty neuroverkkokirjasto tämän mahdollistaa.

8.2 Tekijän omia ajatuksia opinnäytetyön tekemisestä ja oppimisesta

Kaiken kaikkiaan opinnäytetyön tekeminen on ollut antoisa prosessi, jonka aikana tekijä on oppinut paljon uusia asioita niin ohjelmoinnin maailmasta kuin yrityksessä työskentelystäkin. Kaikkein mielenkiintoisinta on tekijän mielestä ollut perehtyä lisää Python-ohjelmointikielen maailmaan. Tieto- ja viestintätekniikan suhteen olisi ollut mielenkiintoista perehtyä opinnäytetyöprosessin aikana enemmän myös muihin käyttöjärjestelmiin kuin Windowsiin.

Ajateltaessa sitä, kuinka hyvin opinnäytetyön aihe vastasi opinnoissa käytyjä asioita, voidaan mainita, että vastaavuus on ollut hyvä. Tämä johtuu siitä, että opintojen pääpaino on ollut koneoppimismenetelmissä (varsinkin neuroverkoissa), joten tehdessään tätä neuroverkkoihin oleellisella tavalla liittyvää opinnäytetyötä tekijä on saanut käyttää opinnoissaan oppimiaan asioita hyödyksi.

Seikka, jota tekijä olisi ehkä toivonut olevan enemmän mukana opinnäytetyön aihepiirissä, ovat relaatiotietokannat, joita on kuitenkin käyty opintojenkin aikana läpi. Ottaen huomioon tekijän oman kiinnostuksen luonnontieteitä kohtaan myös matematiikan ja fysiikan rooli opinnäytetyön aihepiirissä olisi voinut olla suurempikin.

Lähteet

1 Prometecin kotisivut: Prometecin tarina [Internet]. [Viitattu 8.9.2022]. Saatavilla: https://prometec.fi/tarina/

2 Finder.fi: Prometec Tools Oy [Internet]. [Viitattu 8.9.2022]. Saatavilla: https://www.finder.fi/Suunnittelutoimisto/Prometec+Tools+Oy/Kajaani/yhteystiedot/3092973

3 Prometecin kotisivut: Yhteystiedot [Internet]. [Viitattu 8.9.2022]. Saatavilla: https://prometec.fi/yhteystiedot/

4 Neuroverkot: Neuroverkkojen periaatteet. Helsingin yliopisto, Elements of AI -kurssin materiaali. [Internet]. [Viitattu 9.9.2022]. Saatavilla: https://course.elementsofai.com/fi/5/1

5 A Concise History of Neural Networks. Towards Data Science, julkaistu 14.8.2016. [Internet]. [Viitattu 9.9.2022]. Saatavilla: https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec

6 2012: A Breakthrough Year for Deep Learning. Medium, julkaistu 17.7.2019. [Internet]. [Viitattu 9.9.2022]. Saatavilla: https://medium.com/neuralmagic/2012-a-breakthrough-year-for-deep-learning-2a31a6796e73

7 PyTorch strengthens its governance by joining the Linux Foundation. Pytorch, julkaistu 12.9.2022. [Internet]. [Viitattu 15.9.2022]. Saatavilla: https://pytorch.org/blog/PyTorchfoundation/

8 Google Just Open Sourced TensorFlow, Its Artificial Intelligence Engine. Wired, julkaistu 9.11.2015. [Internet]. [Viitattu 15.9.2022]. Saatavilla: https://www.wired.com/2015/11/google-open-sources-its-artificial-intelligence-engine/

9 Ursa ry. Tiedonjano. Helsinki: Ursa ry; 2021.

10 Artificial Neural Networks and its Applications. Geeksforgeeks, viimeksi päivitetty 31.8.2021. [Internet]. [Viitattu 30.9.2022]. Saatavilla: https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

11 New Deep Learning Method Adds 301 Planets to Kepler's Total Count. Nasa Jet Propulsion Laboratory, julkaistu 22.11.2021. [Internet]. [Viitattu 30.9.2022]. Saatavilla: https://www.jpl.nasa.gov/news/new-deep-learning-method-adds-301-planets-to-keplers-total-count

12 Types of Neural Networks. Educba. [Internet]. [Viitattu 13.9.2022]. Saatavilla: https://www.educba.com/types-of-neural-networks/

13 A Comprehensive Guide to Convolutional Neural Networks. V7 Labs, julkaistu 3.10.2022. [Internet]. [Viitattu 8.11.2022]. Saatavilla: https://www.v7labs.com/blog/convolutional-neural-networks-guide

14 Classification of Neural Network. Educba. [Internet]. [Viitattu 8.11.2022]. Saatavilla: https://www.educba.com/classification-of-neural-network/

15 Activation functions in Neural Networks. Geeksforgeeks, viimeksi päivitetty 22.8.2022. [Internet]. [Viitattu 13.9.2022]. Saatavilla: https://www.geeksforgeeks.org/activation-functions-neural-networks/

16 Various Optimization Algorithms For Training Neural Network. Towards Data Science, julkaistu 13.1.2019. [Internet]. [Viitattu 14.9.2022]. Saatavilla: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

17 Neural Network Basics: Loss and Cost Functions. Medium, julkaistu 6.11.2021. [Internet]. [Viitattu 15.9.2022]. Saatavilla: https://medium.com/artificialis/neural-network-basics-loss-and-cost-functions-9d089e9de5f8

18 Mean Squared Error – Explained | What is Mean Square Error? Great Learning, julkaistu 8.8.2020. [Internet]. [Viitattu 7.10.2022]. Saatavilla: https://www.mygreatlearning.com/blog/mean-square-error-explained/

19 What is Neural Networks? Educba. [Internet]. [Viitattu 27.9.2022]. Saatavilla: https://www.educba.com/what-is-neural-networks/

20 4 Reasons Why Deep Learning and Neural Networks Aren't Always the Right Choice. Builtin, julkaistu 24.7.2019. [Internet]. [Viitattu 27.9.2022]. Saatavilla: https://builtin.com/data-science/disadvantages-neural-networks

21 Tensorflow: tf.keras.Model.fit. [Internet]. [Viitattu 30.9.2022]. Saatavilla: https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit

22 Scikit-learn: MLPRegressor.fit. [Internet]. [Viitattu 30.9.2022]. Saatavilla: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor.fit

23 Overfitting Neural Network. Educba. [Internet]. [Viitattu 27.9.2022]. Saatavilla: https://www.educba.com/overfitting-neural-network/

24 Python: Downloads. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://www.python.org/downloads/

25 PyPI: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://pypi.org/

26 Scikit-learn: About us. [Internet]. [Viitattu 16.9.2022]. Saatavilla: https://scikit-learn.org/stable/about.html

27 Pytorch: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://pytorch.org/

28 Scikit-learn: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://scikit-learn.org/stable/index.html

29 Tensorflow: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://www.tensorflow.org/

30 Tensorflow: Overview. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://www.tensorflow.org/overview

31 Pytorch: LICENSE. GitHub, viimeksi päivitetty 4.3.2022. [Internet]. [Viitattu 11.10.2022]. Saatavilla: https://github.com/pytorch/pytorch/blob/master/LICENSE

32 Tensorflow: LICENSE. GitHub, viimeksi päivitetty 29.11.2021. [Internet]. [Viitattu 11.10.2022]. Saatavilla: https://github.com/tensorflow/tensorflow/blob/master/LICENSE

33 Scikit-learn: COPYING. GitHub, viimeksi päivitetty 12.9.2022. [Internet]. [Viitattu 11.10.2022]. Saatavilla: https://github.com/scikit-learn/scikit-learn/blob/main/COPYING

34 Scikit-learn: MLPRegressor. [Internet]. [Viitattu 30.9.2022]. Saatavilla: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

35 Pandas: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://pandas.pydata.org/

36 NumPy: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://numpy.org/

37 Scikit-learn: train_test_split. [Internet]. [Viitattu 7.10.2022]. Saatavilla: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

38 TkInter: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://docs.python.org/3/library/tkinter.html

39 Matplotlib: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://matplotlib.org/

40 Pickle: Etusivu. [Internet]. [Viitattu 18.11.2022]. Saatavilla: https://docs.python.org/3/library/pickle.html