Koneoppiminen pään ja kaulan alueen syövän tunnistamisessa

Henri Hellström

Pro gradu -tutkielma
Syyskuu 2021

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

HELLSTRÖM, HENRI: Koneoppiminen pään ja kaulan alueen syövän tunnistamisessa
Pro gradu -tutkielma, 47 s.
Sovellettu matematiikka
Syyskuu 2021

Tämän tutkielman päätavoitteena on kehittää koneoppimista hyödyntävä menetelmä pään ja kaulan alueen syövän tunnistamiseen kaksiulotteisista PET-MRI-kuvista. PET-MRI-kuvat ovat tärkeässä roolissa syövän diagnosonoissa sekä hoidon suunnittelussa, mutta syöpäkudoksen leimaaminen kuviin on aikaa vievä ja työläs prosessi. Tutkielmassa kehitettävä koneoppimissysteemi ennustaa tietyllä luottamustasolla, sisältääkö yksittäinen PET-MRI-leike syöpäkudosta vai ei. Asiantuntija voi sitten käyttää ennustetta leimausprosessin aikana suuntaa antavana arviona syöpäkudoksen esiintymistodennäköisyydestä tietyssä leikkeessä.

Tutkielmassa käytettävät menetelmät lukeutuvat konvoluutioneuroverkoiksi. Tutkimusongelmaa lähestytään kouluttamalla kaksi mallia, joista toinen on valmiiksi koulutettu ResNet50-malli ja toinen perinteinen konvoluutioverkko, jonka parametrit optimoidaan alusta alkaen PET-MRI-aineistolla. Mallien rakentaminen, optimointi ja suorituksen evaluointi toteutetaan käyttämällä Python-ohjelmointikieltä sekä syviin neuroverkkoihin erikoistunutta Tensorflow-kirjastoa.

Perinteinen konvoluutioverkko yltää kohtalaisen hyviin tuloksiin, mutta esikoulutetulla mallilla ei löydetä datasta mielenkiintoista signaalia. Vaikka malleista parempi luokittelee sille syötetyn kuvaparin kohtuullisen usein oikein, ei se vielä kykene täysin autonomiseen toimintaan. Tutkielman malli on sovelluskohteeseen nähden vielä varsin yksinkertainen ja jatkotutkimuskohteita on useita.

Asiasanat: koneoppiminen, PET, MRI, konvoluutioneuroverkko, ResNet50, luokittelu, Python, Tensorflow.

Sisällys

1 Johdanto

Syöpä on nimitys joukolle sairauksia, jossa solujen jakautuminen on epänormaalia ja invasiivista. Pään ja kaulan alueen syöpiä puolestaan karakterisoi niiden sijainti ihmiselimistössä. Yleisin pään ja kaulan alueen syövistä on levyepiteelikarsinooma. [25] Vaikeasta sijainnistaan johtuen pään ja kaulan syöpien diagnosoinnissa joudutaan usein turvautumaan lääketieteelliseen kuvantamiseen, erityisesti PET- ja MRI-kuvantamiseen.

PET-MRI-kuvat jälkikäsitellään leimaamalla ne pikselit, jotka sisältävät syöpäkudosta. Kuvat ovat usein kolmiulotteisia ja sisältävät siis monta poikkileukkausta potilaan kuvatusta elimistön osasta. Tämän vuoksi leimaaminen on usein hyvin työläs ja aikaavievä prosessi, joka vähentää asiantuntijoiden aikaa hoitaa muita työtehtäviä. Tämä motivoi uuden ratkaisun löytämistä, missä ihmisen sijaan leimaamisen hoitaisikin älykäs tietokoneohjelma. Tämän tutkielman tavoitteena onkin saattaa hanke aluilleen ja rakentaa koneoppimismalli, joka estimoi todennäköisyyden sille, että yksittäinen PET-MRI-leike sisältää syöpäkudosta.

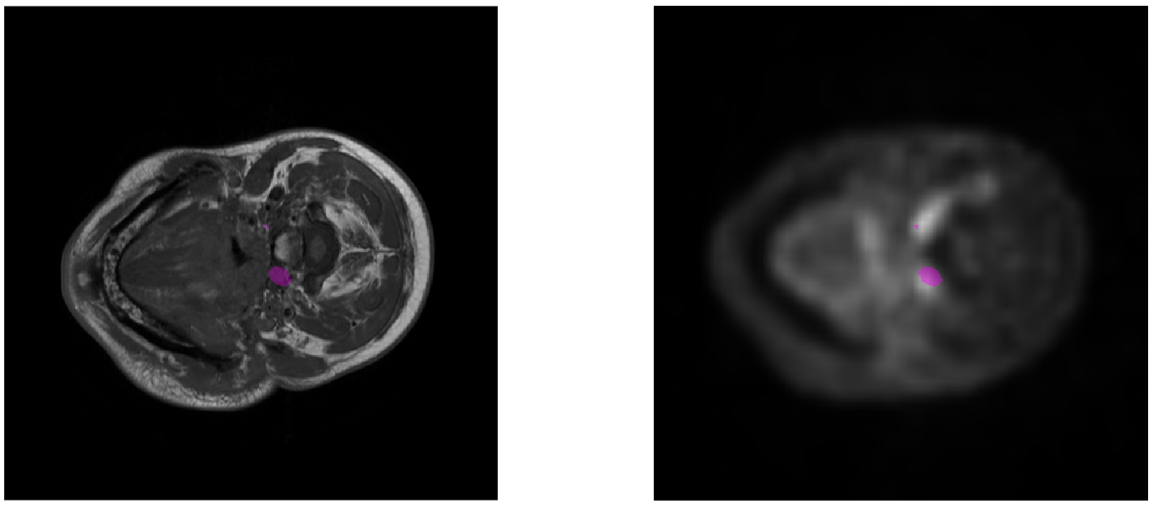
Tutkielmassa käytetyt koneoppimismallit lukeutuvat konvoluutioneuroverkoiksi, jotka ovat hyvin suosittuja kuvantunnistuksen saralla. Tutkielmassa verrataan, miten tutkielman aineistosta rippumattomalla valtavalla datalla valmiiksi koulutetun mallin tulokset eroavat puhtaasti tutkielman aineistolla koulutetun mallin tuloksista. PET-MRI-aineisto on varsin pieni konvoluutioverkon tarpeisiin nähden, minkä vuoksi olisi otollista, että esikoulutettu malli suoriutuisi tehtävästä hyvin.

2 Aineisto

Aineistona on 63 potilaan kolmiulotteisia PET-MRI-kuvia, joista 18 on positiivisia ja 45 negatiivisia. Positiivisilla potilailla on todettu löytyvän levyeptieelikarsinoomaa ja PET-MRI-kuvat on näissä tapauksissa kuvattu 3 kuukauden mittaisen sädehoitorutiinin jälkeen. Positiivisten potilaiden osalta MRI- ja PET-kuvien lisäksi jokaiselle kuvaparille on käytössä binäärimaski, joka on dimensioiltaan identtinen vastaavien kuvien kanssa. Binäärimaskit ovat asiantuntijan koostamia ja esittävät mahdollisen syöpäkudoksen sijaintia poikkileukkauksessa. PET- ja MRI- kuvat sekä binäärimaskit ovat *NIfTI*-formaattia.

Aineiston kuvien poikkileikkaukset sisältävät 512×512 pikseliä ja yhden värikanavan. Poikkileikkausten lukumäärä vaihtelee potilaittain, mutta keskimäärin poikkileikkauksia on noin 38. Positiivisten potilaiden kuvat sisältävät keskimäärin 10 leikettä, joissa esintyy binäärimaskissa leimattua syöpäkudosta. Tutkielman neuroverkkomalli ottaa syötteenään PET- ja MRI-kuvat pareittain. Binäärimaskeja ei

käytetä koulutusvaiheessa, vaan tässä vaiheessa leimattujen pikseleiden avulla niistä vain määritetään, sisältääkö yksittäinen leike syöpäkudosta vai ei.



Kuva 1: Vasemmalla eräs aineiston MRI-leikkeistä, jossa syöpäkudos on merkitty violetilla. Oikealla vastaava PET-leike.

2.1 MRI- kuvantamisen periaatteet

Mageneettikuvaus (Magnetic Resonance Imaging, MRI) on keino kuvantaa elimistön anatomisia rakenteita korkealla resoluutiolla [22]. Magneettikuvauksen mahdollistava fysikaalinen ilmiö liittyy vetyatomiytimen protonin pyörimisliikkeeseen. Tätä lähes kaikille alkeishiukkasille ominaista liikettä kutsutaan *spiniksi*. Pyörimisliikkeen vuoksi positiivisesti varatulla protonilla on ympärillään magneettikenttä ja siten ulkoisilla magneettikentillä on vaikutus protonin orientaatioon. Protonin magneettikenttä on puolestaan havaittavissa signaalina sopivan detektorin avulla. [21]

Käytännössä MRI-kuvantamisessa generoidaan tietyllä taajuudella sähkömagneettisia pulsseja, jotka muuttavat elimistön vetyatomien orientaatiota havaittavalla tavalla. Nimensä mukaan MRI-kuvantamisessa hyödynnetään *resonanssia*, sillä protonit virittäytyvät vain tietyn suuruisella taajuudella. Tätä pulssia ei kohdisteta koko ruumiseen kerralla, vaan MRI-kuvantamisessa tuloksena on joukko poikkileikkauksia elimistön rakenteista. Poikkileikkausten muodostuminen on mahdollista varioimalla tätä taajuutta halutun ruumiinosan yli. Näin magneettikenttä vaikuttaa ainoastaan niihin protoneihin, jotka ovat tietyn taajuuden piirissä samalla kun

muun elimistön toiminta pysyy muuttumattomana. Näiden muutosten spatiaalisten sijaintien havaitseminen mahdollistaa MRI-kuvan konstruoinnin, mutta juuri signaalin lähteen sijainnin tunnistaminen lienee MRI-kuvantamisen suurin haaste. [21]

MRI-kuvantamisen heikkoutena voidaan kenties pitää signaalin herkkyyttä kohinalle. Useat tekijät voivat aiheuttaa merkittävää kohinaa mittausdataan. Esimerkiksi magneettipulssin generoivien käämien lämpeneminen, potilaan ruumiin liike ja hengityksestä aiheutuva liike voivat aiheuttaa kuvan laatua merkittävästi huonontavaa kohinaa. Magneettikuvauksessa on kuitenkin usean parametrin avulla mahdollista kontrolloida mittauksen herkkyyttä. Esimerkiksi poikkileukkauksen leveyden, kuvan pikseleiden lukumäärän sekä magneettikentän voimakkuuden avulla on mahdollista hienosäätää mittauksen tarkkuutta. [21]

MRI-kuvantamisella on useita etuja verrattuna muihin kuvantamismenetelmiin, mikä selittäneekin sen suuren suosion useiden eri ruumiinosien kuvantamisessa. Eräs keskeinen syy, miksi magneettikuvantamisella anatomiset rakenteet saadaan hyvin näkyviin kuten kuvasta 1 nähdään, on luiden näkymättömyys MRI-kuvassa. Tällöin kaikkia elimistön osia on mahdollista tarkastella luukudoksen häiritsemättä. Usein lääketieteellinen kuvantaminen pitää sisällään jonkinlaista altistumista ionisoivalle säteilylle, kuten esimerkiksi röntgenkuvantamisessa. Magneettikuvauksessä tätä riskiä ei kuitenkaan ole. Tosin potilaille, joilla on implantteja tai muuta epäorgaanista (magnetisoituvaa) ainesta elimistössä, magneettikuvaus saattaa sisältää enemmän riskejä. MRI-kuvantamista voidaan yhdistää muihin kuvantamismenetelmiin, kuten tämän tutkielman aineiston valossa PET-kuvantamiseen. Tällä on varsinkin syövän tunnistuksessa se etu, että MRI mahdollistaa potilaan anatomiassa poikkeavien rakenteiden tutkimisen, kun taas PET havainnolistaa elimistössä tapahtuvia metabolisia poikkeavuuksia. [27]

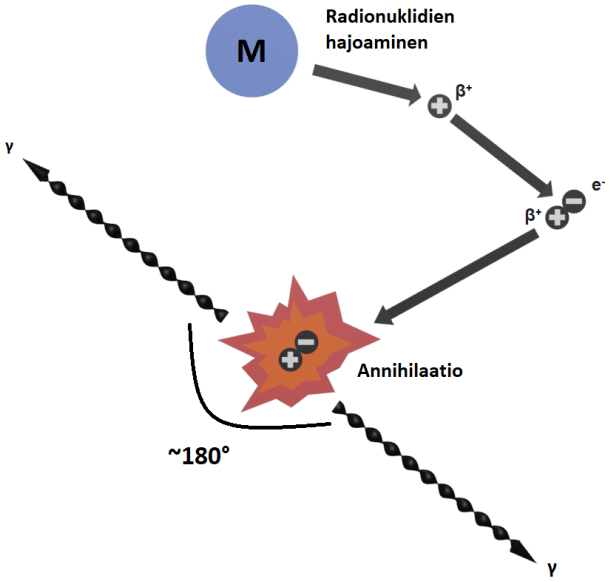
2.2 PET- kuvantamaisen periaatteet

Positroniemissiotomografia (PET) on varsin suosittu menetelmä monien eri syöpätyyppien tunnistamiseksi. PET- kuvantaminen perustuu pahalaatuisen kasvaimen solujen poikkeavaan metaboliaan, mikä mahdollistaa elimen epänormaalin toiminnan havaitsemisen jo ennen minkäänlaisia morfologisia muutoksia. Tämä ominaisuus on myös etu jo mahdollisesti operoitujen potilaiden tutkimisessa, sillä kirurgisten toimenpiteiden aiheuttamat muutokset eivät häiritse kuvantamista. [20]

Atomitasolla PET- kuvantaminen perustuu radiofarmateuttisten merkkiaineiden aiheuttamaan fotonien emissioon. Radionuklideilla varustettu merkkiaine injektoidaan potilaaseen, jolloin se kulkeutuu verenkierron mukana kohde-elimiin. Kasvaimessa merkkiaineen, kuten fluorodeoksiglukoosin (FDG), radionuklidit emittoivat

positroneja eli hiukkasia, jotka ovat massaltaan yhtä suuria kuin elektronit, mutta positiivisesti varautuneita [20]. Positronit puolestaan annihiloituvat elektronien kanssa vapauttaen kaksi vastakkaisiin suuntiin emittoivaa gammakvanttia (fotonia) kuten kuvasta 2 näkyy. PET- laitteen detektori lopulta havaitsee nämä hiukkaset. [30]

Annihilaatiossa emittoutujen fotonien emissiosuunnan vastakkaisuutta käyte-



Kuva 2: Merkkiaineen (M) radionuklidien hajoaminen ja positronin sekä elektronin annihilaatio.

tään tehokkaasti hyväksi PET- tutkimuksissa. Rengasmaiset fotodetektorit rekisteröivät vain sellaiset emittoidut fotonit, jotka osuvat lähes samanaikaisesti kahteen vastakkaiseen detektoriin. Tämän avulla PET- mittauksissa on mahdollista pienentää taustasäteilyn vaikutusta sekä kohinaa ja korostaa mielenkiintoisen signaalin intensiteettiä. [30]

PET- kuvat eivät kuitenkaan sellaisenaan mahdollista anatomisten rakenteiden tunnistamista, vaan ne yksinomaan korostavat metabolisia eroavaisuuksia. Tämän vuoksi PET- kuvantamista sekä tietokonetomografiaa tai MRI- kuvantamista hyödynnetäänkin usein rinnakkain. MRI kuitenkin usein mahdollistaa anatomisten rakenteiden tunnistamisen jopa tarkemmin kuin tietokonetomografian avulla. [20]

2.3 Tavoitteet

Tutkielman tavoitteena on rakentaa konvoluutioneuroverkko, joka kykenee mahdollisimman hyvin tunnistaman, sisältääkö mallille syötetty kuva syöpäkudosta vai ei.

Tämän lisäksi tutkitaan, kuinka merkittävästi konvoluutioneuroverkoille ominaista *siirtovaikutusta* (transfer learning) voidaan hyödyntää mallin kouluttamisessa. Esikoulutetuilla malleilla on se etu, että ne on valmiiksi koulutettu valtavalla määrällä dataa, jonka jälkeen ne voidaan hienosäätää muihin tarpeisiin pienemmälläkin aineistolla. Tällä tavoin vältytään kouluttamasta täysin uutta neuroverkkoa, mikä on usein laskennallisesti tietokoneelle hyvin raskas prosessi. Lisäksi valmiiksi koulutetut mallit ovat yksinkertaisia käyttää, sillä niiden arrkitehtuuri ja parametrien arvot ovat kiinnitettyjä.

Mallin ei kuitenkaan ole tarkoitus toimia täysin autonomisesti. Sen on tarkoitus vielä tässä vaiheessa vain avustaa asiantuntijaa leimausprosessissa. Esimerkiksi, mikäli malli ennustaa tietyllä luottamustasolla, että leike ei sisällä syöpäkudosta, voi asiantuntija kiinnittää leikkeeseen vähemmän huomiota. Näin voidaan käyttää enemmän aikaa muiden leikkeiden yksityiskohtaisempaan tutkimiseen.

3 Koneoppiminen

Koneoppimisessa on kyse runsaasti saatavilla olevan datan hyödyntämisestä arvokkaan tiedon jalostamiseksi. Nykyään dataa on useimmiten yltäkylläisesti saatavilla, mutta sellaisenaan siitä ei ole mitään ilmeistä hyötyä. Koneoppimisen viitekehyksessä tiedon jalostaminen tapahtuu käyttämällä algoritmeja, jotka nähtyään luokiteltua tai luokittelematonta harjoitusdataa kykenevät havaitsemaan datasta yleisiä trendejä tai kuvioita. Kouluttamisen jälkeen kiinnostuksen kohteena on usein mallin soveltaminen uuteen dataan, jota malli ei ole vielä koulutusvaiheessa nähnyt. [11]

Koneoppiminen voidaan karkeasti jakaa ohjattuun oppimiseen sekä epäohjattuun oppimiseen. Ohjatussa oppimisessa valmiiksi luokiteltua dataa käytetään koneoppimisalgoritmin koulutuksessa, minkä jälkeen mallin avulla voidaan ennustaa luokitteluongelmissa uuden datapisteen luokka tai regressio-ongelmissa arvo [18]. Epäohjattua oppimista sovelletaan puolestaan ekspolaritiivisessa data-analyysissa tai piirteiden valinnassa [3]. Erona ohjattuun oppimiseen on, että tässä tapauksessa data ei ole etukäteen ryhmiteltyä tai luokiteltua. Tässä tutkielmassa käytetään neuroverkkomallia, joka lukeutuu ohjatuksi koneoppimismalliksi.

3.1 Neuroverkko

Ihmisbiologiasta inspiraationsa saaneet neuroverkot hyödyntävät useita itsenäisiä yksiköitä monimutkaisen funktion approksimoimiseksi [3]. Yksinkertaisen painotetun summan avulla syötetty data virtaa neuroverkossa eteenpäin mahdollisesti usean

kerroksen läpi lopulta saavuttaen viimeisen kerroksen, jolloin neuroverkko on jalostanut tarkoituksenmukaisella tavalla datasta informaatiota.

3.1.1 Perseptroni

Perseptronit ovat neuroverkkomalleista yksinkertaisimpia. Perseptronissa jokaista syötettä $x_j \in \mathbb{R}$, $j = 1, ..., d$, missä d on syötteen dimensio ja $\mathbb{R} = (-\infty, \infty)$ on reaalilukujen joukko, vastaa yksittäinen *neuroni*, jolla on jokin *synaptinen painokerroin* $w_j \in \mathbb{R}$. Lisäksi perseptroniin liitetään yksi ylimääräinen yksikkö, joka saa aina arvon 1 ja jonka synaptinen paino on w_0. Yksinkertaisimmassa tapauksessa perseptroni laskee syötteen sekä ylimääräisen vakion avulla tulosteen y arvon painotettuna summana:

$$y = \sum_{j=1}^{d} w_j x_j + w_0. \tag{1}$$

Merkitsemällä $\boldsymbol{w} = [w_0, ...w_d]^T$ ja $\boldsymbol{x} = [1, x_1, ..., x_d]^T$ perseptronin laskutoimitus voidaan esittää vektorimuododssa

$$y = \boldsymbol{w}^T \boldsymbol{x}. \tag{2}$$

Yhden perseptronin mallit kykenevät ratkaisemaan joitakin kahden luokan luokittelu- tai regressio-ongelmia. Mikäli luokkia on useampia, rinnakkain olevien perseptronien lukumäärän on vastattava luokkien lukumäärää. Laskutoimitukset ovat oleellisesti samanlaisia kuin yhden perseptronin tapauksessa, mutta nyt painotettu keskiarvo lasketaan jokaiselle perseptronille erikseen ja tulosteita on yhtä monta kuin perseptroneja. Tätä rinnakkaisuutta on havainnollistettu kuvassa 3.

Olkoon luokkien määrä K. Jokaiselle $i = 1, ..., K$ lasketaan

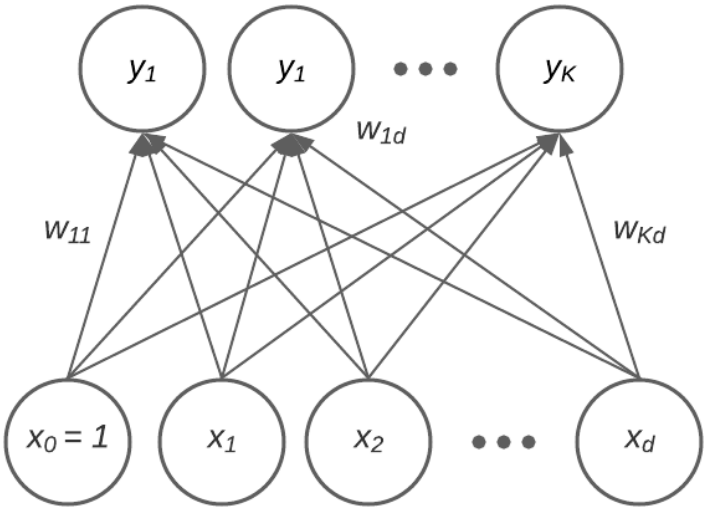
$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_0 = \boldsymbol{w}_i^T \boldsymbol{x}, \tag{3}$$

missä $\boldsymbol{w}_i = [w_{i0}, ..., w_{id}]^T$. Tämä voidaan esittää matriisimuodossa

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}, \tag{4}$$

missä $\boldsymbol{W}$ on $K \times (d+1)$- matriisi, jonka riveinä ovat kunkin perseptronin painokertoimista muodostetut vektorit. [3]

Perseptronimallin tekemät laskutoimitukset ovat oleellisesti lineaarisia. Usein on kuitenkin hyödyllistä lisätä perseptronimalliin epälineaarisuutta syöttämällä mallin laskema painotettu keskiarvo $\boldsymbol{w_i}^T \boldsymbol{x}$ jollekin *aktivaatiofunktiolle*. Ne ovat derivoituvia



Kuva 3: Graafinen esitys rinnakkaisten perseptronien toiminnasta.

funktioita ja useimmiten määriteltyjä kaikilla reaaliluvuilla, mutta niiden arvojoukot vaihtelevat tarpeen mukaan. Esimerkiksi kahden luokan luokitteluongelmissa on usein mielenkiintoista tutkia, millä todennäköisyydellä annettu syöte kuuluu jompaankumpaan luokkaan. Näissä tapauksissa tulostekerroksessa käytetään yleisesti funktiota $sigmoid : \mathbb{R} \to [0, 1]$, jolle

$$sigmoid(\boldsymbol{w}_i^T \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}_i^T \boldsymbol{x}}}, \tag{5}$$

jonka saamat arvot voidaan tulkita todennäköisyyksinä. [3] Jos luokkien lukumäärä K on suurempi kuin kaksi, voidaan vastaavanlaiset todennäköisyydet generoida käyttämällä tulostetasolla funktiota $softmax : \mathbb{R} \to [0, 1]$, missä

$$softmax(\boldsymbol{w}_i^T \boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^T \boldsymbol{x}}}{\sum\limits_{k \in K} e^{\boldsymbol{w}_i^T \boldsymbol{x}}} \tag{6}$$

kaikille tulostetason yksiköille $k = 1, ..., K$.

Yhden perseptronin malli voi luokitella täydellisesti ainoastaan lineaarisesti separoituvia pistejoukkoja. Toisin sanoen jos $X = \{x_1, ..., x_n\}$ ja $Y = \{y_1, ..., y_m\}$ on oltava

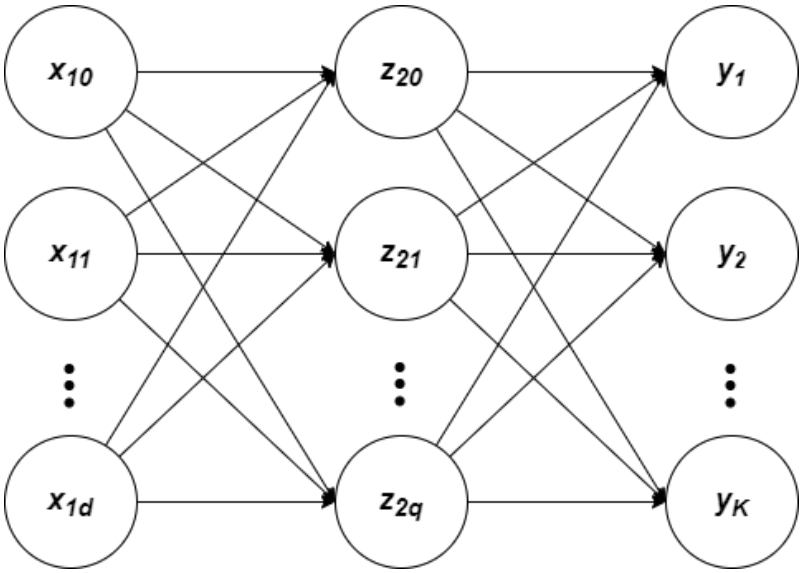
$$\text{conv}(X) \cap \text{conv}(Y) = \emptyset, \quad x_i \in \mathbb{R}^d \ \ \forall i = 1, ..., n \text{ ja } y_j \in \mathbb{R}^d \ \ \forall j = 1, ..., m, \tag{7}$$

eli pistejoukkojen konveksien peitteiden leikkauksen on oltava tyhjä. Todellisuudessa

tämä ehto harvoin toteutuu, jolloin yhden perseptronin malli ei ole kompleksisuudeltaan riittävä mallintamaan tilannetta. Monimutkaisempia funktioita voidaankin approksimoida lisäämällä yhden kerroksen perseptronimalliin *piilotettuja kerroksia*. Tällöin piilotetut kerrokset saavat syötteeksi edellisen kerroksen aktivoidut tulosteet. Useamman kerroksen neuroverkoissa sekä konvoluutioverkoissa piilotetuilla tasoilla käytetään usein aktivaatiofunktiona $ReLU$- funktiota (Rectified Linear Unit), missä

$$ReLU(x) = \max\{0, x\}. \tag{8}$$

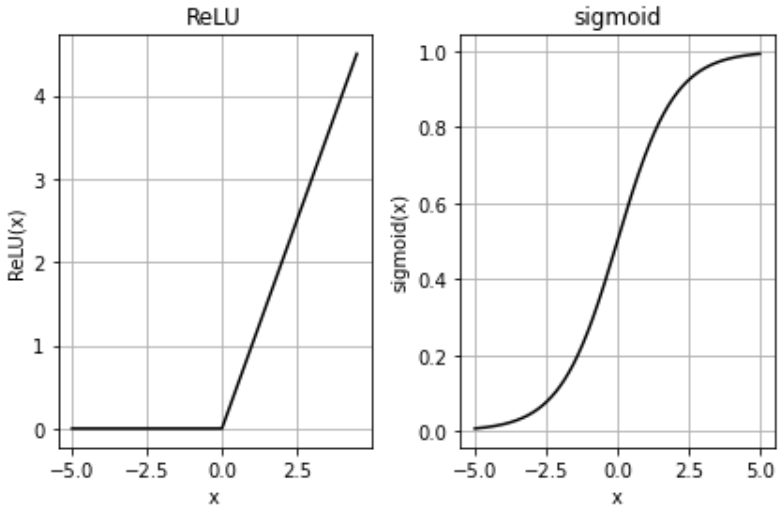
Kuvasta 5 nähdään, että funktio ei ole selvästikään derivoituva pisteessä $x = 0$, mutta usein sopimuksenvaraisesti asetetaan $ReLU'(0) = 0$. Käytännössä usean kerroksen perseptroniverkko rakentuu monesta kerroksesta rinnaikkaisia perseptroneja ja jokainen kerros sisältää vakioyksikön sekä vaihtelevan määrän neuroneita. [6]



Kuva 4: Graafinen esitys perseptroniverkosta, joka sisältää yhden piilotetun kerroksen $\{z_{20}, z_{21}, ..., z_{2q}\}$.

Esimerkki 1. Tarkastellaan kuvan 4 neuroverkoa ja oletetaan, että kaikki kerroksen k yksiköt ovat yhteydessä kaikkiin kerroksen $k + 1$ yksiköihin. Olkoon kerroksella 1 aktivaatiofunktio $f_1 : \mathbb{R} \to \mathbb{R}$ ja ulostulokerroksella aktivaatiofunktio $f_o : \mathbb{R} \to \mathbb{R}$. Olkoon $\boldsymbol{W}$ $(q+1) \times (d+1)$- matriisi, jonka riveinä ovat kerroksen 1 painokertoimista muodostetut vektorit ja $\boldsymbol{V}$ $K \times (q + 1)$-matriisi, jonka riveinä ovat kerroksen 2 painokertoimista muodostetut vektorit. Nyt kerroksella 1 kaikille $i = 0, ..., q$ saadaan

$$z_i = f_1(\sum_{j=0}^{d+1} w_{ij} x_j), \tag{9}$$



Kuva 5: Aktivaatiofunktioiden kuvaajia.

missä w_{ij} on matriisin $\boldsymbol{W}$ alkio. Vastaavasti ulostulokerroksella eli kerroksella 2 saadaan kaikille $k = 1, ..., K$

$$y_k = f_o(\sum_{j=0}^{q+1} v_{ij} z_j), \tag{10}$$

missä v_{ij} on matriisin $\boldsymbol{V}$ alkio. Matriisimuodossa koko neuroverkon tekemä laskutoimitus voidaan vielä esittää seuraavasti:

$$\boldsymbol{y} = f_o(\boldsymbol{V} \underbrace{f_1(\boldsymbol{W}\boldsymbol{x})}_{\text{z}}), \tag{11}$$

missä funktioita f_1 ja f_o sovelletaan matriisituloista syntyviin vektoreihin alkoittain.

3.1.2 Monikerroksisen perseptroniverkon kouluttaminen

Oppiminen tapahtuu neuroverkkojen kohdalla synaptisten painokertoimien välityksellä. Aluksi painokertoimet alustetaan joillakin satunnaisilla pienillä luvuilla, jonka jälkeen neuroverkolle syötetään dataa, joka puolestaan virtaa kerrokselta kerrokselle. Ohjatun oppimisen tapauksessa neuroverkko määrää viimeisellä kerroksella datapisteille leiman, jota vertaamalla kyseisen pisteen todelliseen leimaan voidaan hienosäätää neuroverkon painokertoimia, jotta verkon ennustamat leimat olisivat mahdollisimman lähellä oikeita leimoja. Data ikään kuin virtaa neuroverkossa väärään suuntaan ulostulokerrokselta syötekerrokselle, mistä syvien neuroverkkojen koulutuksessa käytetty algoritmi *vastavirta* (backpropagation) onkin saanut nimensä. [6]

Vastavirta-algoritmi mahdollistaa jonkin differentioituvan tappiofunktion gradientin laskemisen rekursiivisesti ulostulokerrokselta aina syötekerrokselle asti. Usein kuitenkin monikerroksisten neuroverkkomallien kompleksisuus vaikeuttaa painokerrointen optimoimista muuttamalla tappiofunktion epäkonveksiksi. [6] Tämän vuoksi optimaalisia painokertoimia ei ole mahdollista määrittää analyyttisesti, vaan on turvauduttava iteratiivisiin optimointimenetelmiin. Luonnollinen vaihtoehto (mahdollisimman) optimaalisten painokertoimien löytämiseksi on siis soveltaa *nopeimman laskeutumisen menetelmää* (Gradient descent). [26]

Algoritmi 1. ***Nopeimman laskeutumisen menetelmä (NLM)***
Olkoon $\boldsymbol{x_0} \in \mathbb{R}^n$ *aloituspiste,* $\epsilon > 0$, $f : \mathbb{R}^n \to \mathbb{R}$ *differentioituva funktio ja* $k = 0$. *Neuroverkkojen tapauksessa* x_0 *vastaa alustettuja painokertoimia. Jos* $||\nabla f(\boldsymbol{x_k})|| < \epsilon$, *algoritmi päättyy. Muutoin määrätään suunta*

$$\boldsymbol{d_k} = -\frac{\nabla f(\boldsymbol{x_k})}{||\nabla f(\boldsymbol{x_k})||}, \tag{12}$$

siirrytään pisteeseen $\boldsymbol{x_{k+1}} = \boldsymbol{x_k} + \eta \boldsymbol{d_k}$, *missä askelpituus* $\eta > 0$ *ja asetetaan* $k = k + 1$. *Iteraatioita toistetaan kunnes gradientin normi on riittävän pieni, jolloin algoritmi on löytänyt lokaalin tai parhaassa tapauksessa globaalin minimin.*

Lause 1. *Olkoon* $\boldsymbol{x_0} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ *differentioituva pisteessä* $\boldsymbol{x}$ *ja oletetaan, että* $||f(\boldsymbol{x})|| \neq 0$. *Tällöin pisteessä* $\boldsymbol{x}$ *suunta, jossa funktion* f *arvo pienenee nopeiten on*

$$\boldsymbol{d} = -\frac{\nabla f(\boldsymbol{x})}{||\nabla f(\boldsymbol{x})||}. \tag{13}$$

Todistus. Koska f on differentioituva pisteessä $\boldsymbol{x}$, on sillä olemassa suuntaderivaatta suuntaan $\boldsymbol{d} \in \mathbb{R}^n$. On siis löydettävä $\boldsymbol{d}$, joka ratkaisee optimointitehtävän $\min_{||\boldsymbol{d}||=1} f'(\boldsymbol{x}; \boldsymbol{d})$. Koska f on differentioituva,

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \nabla f(\boldsymbol{x})^T \boldsymbol{d}. \tag{14}$$

Koska $||\boldsymbol{d}|| = 1$, niin Cauchy-Schwartzin lauseen avulla saadaan

$$-||\nabla f(\boldsymbol{x})|| = -||\nabla f(\boldsymbol{x})|| \cdot ||\boldsymbol{d}|| \leq \nabla f(\boldsymbol{x})^T \boldsymbol{d}. \tag{15}$$

Siis suuntaderivaattaa rajoittaa alapuolelta negatiivinen gradientin normi $-||\nabla f(\boldsymbol{x})||$. Koska $\nabla f(\boldsymbol{x})^T \nabla f(\boldsymbol{x}) = ||\nabla f(\boldsymbol{x})||^2$, niin sijoittamalla $\boldsymbol{d} = -\frac{\nabla f(\boldsymbol{x})}{||\nabla f(\boldsymbol{x})||}$ epäyhtälöön

nähdään, että yhtäsuuruus pätee:

$$\nabla f(\boldsymbol{x})^T \boldsymbol{d} = -\nabla f(\boldsymbol{x})^T \cdot \frac{\nabla f(\boldsymbol{x})}{||\nabla f(\boldsymbol{x})||} = -||\nabla f(\boldsymbol{x})||. \tag{16}$$

Siis negatiivisen gradientin suunta minimoi suuntaderivaatan $f'(\boldsymbol{x}; \boldsymbol{d})$. □

Vastavirta-algoritmin tavoitteena on rekursiivisesti hyödyntää ulostulotasolla määritetyn virheen vaikutusta koko neuroverkon painokertoimien päivittämiseen ja lopulta näin minimoida kaikista datapisteistä summattu virhe. Keräämällä kaikki neuroverkon parametrit vektoriin $\boldsymbol{\theta}$, tämä päivitys voidaan esittää yleisellä tasolla NLM-menetelmän avulla yhtälöllä

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \frac{\partial E(X; \boldsymbol{\theta}^t)}{\partial \boldsymbol{\theta}}, \tag{17}$$

missä X kuvaa neuroverkolle syötettyä aineistoa, $E : \mathbb{R} \to \mathbb{R}$ virhefunktiota ja t iteraatiokierroksen numeroa. Ohjattujen neuroverkkojen kouluttamisessa varsin suosittu tappiofunktio on keskineliövirhe

$$E^n = \frac{1}{2}(\hat{y_n} - y_n)^2, \tag{18}$$

missä y_n on pisteen $\boldsymbol{x}^n$ todellinen leima ja $\hat{y_n}$ on pisteen $\boldsymbol{x}^n$ ennustettu leima. [23] Kun summataan jokaisesta koulutusdatan pisteestä aiheutunut virhe yhteen, saadaan vastavirta-algoritmin tavoitefunktio

$$E(X; \boldsymbol{\theta}) = \frac{1}{2N} \sum_{n=1}^{N} (\hat{y_n} - y_n)^2. \tag{19}$$

Virheen gradientin määrittäminen tuollaisenaan ei ole käytännöllistä, vaan se on jaettava osiin ketjusäännön avulla. [15]

Lause 2. *(Ketjusääntö) Olkoon funktio $f : A \to B$ derivoituva pisteessä $x \in A$ ja funktio $g : B \to \mathbb{R}$ derivoituva pisteessä $f(x) \in B$ ja lisäksi $A, B \subset \mathbb{R}$. Nyt yhdistetty fuktio $h(x) = g(f(x))$ on differentioituva pisteessä x ja sille pätee*

$$\frac{\partial h}{\partial x} = \frac{\partial g}{\partial f} \frac{\partial f}{\partial x}. \tag{20}$$

Todistus. Todistetaan ensin, että funktiolle $g : B \to \mathbb{R}$, joka on derivoituva pisteessä $x_0 \in B$, pätee *differentiaalihajotelma*

$$g(x_0 + h) = g(x_0) + g'(x_0)h + hu(h). \tag{21}$$

Yhtälössä (21) $h \in (r_1, r_2)$, missä $r_1, r_2 > 0$ ja $(x_0 - r_1, x_0 + r2) \subset B$. Lisäksi asetetaan funktiolle $u : (r_1, r_2) \to \mathbb{R}$, että $u(0) = 0$ ja $u(h) = \frac{g(x_0+h)-g(x_0)}{h} - g'(x_0)$. Nyt yhtälölle (21) saadaan

$$\begin{aligned} g(x_0 + h) &= g(x_0 + h) - g(x_0) + g(x_0) + g'(x_0)h - g'(x_0)h \\ &= g(x_0) + g'(x_0)h + g(x_0 + h) - g(x_0) - g'(x_0)h \\ &= g(x_0) + g'(x_0)h + \left(\frac{g(x_0 + h) - g(x_0)}{h} - g'(x_0)\right)h \\ &= g(x_0) + g'(x_0)h + hu(h). \end{aligned} \tag{22}$$

Todistetaan seuraavaksi lauseessa 2 esitetty tulos. Olkoon $x_0 \in A$. Funktion g derivoituvuuden ja yhtälön (21) avulla saadaan, että pisteessä $f(x_0)$ pätee

$$g(f(x)) = g(f(x_0)) + g'(f(x_0))(f(x) - f(x_0)) + (f(x) - f(x_0))u((f(x) - f(x_0)). \tag{23}$$

Nyt funktion h erotusosamäärä saadaan muotoon

$$\begin{aligned} &\frac{h(x) - h(x_0)}{x - x_0} = \frac{g(f(x)) - g(f(x_0))}{x - x_0} \\ &= \frac{g(f(x_0)) + g'(f(x_0))(f(x) - f(x_0)) + (f(x) - f(x_0))u((f(x) - f(x_0)) - g(f(x_0))}{x - x_0} \\ &= g'(f(x_0))\frac{f(x) - f(x_0)}{x - x_0} + u(f(x) - f(x_0))\frac{f(x) - f(x_0)}{x - x_0}. \end{aligned}$$

Koska f on derivoituva pisteessä x_0 ja $\lim_{h\to 0} u(h) = 0$, niin

$$\begin{aligned} h'(x_0) &= \lim_{x\to x_0} \frac{h(x) - h(x_0)}{x - x_0} \\ &= \lim_{x\to x_0} g'(f(x_0)) \cdot \lim_{x\to x_0} \frac{f(x) - f(x_0)}{x - x_0} + \lim_{x\to x_0} u(f(x) - f(x_0)) \cdot \lim_{x\to x_0} \frac{f(x) - f(x_0)}{x - x_0} \\ &= g'(f(x_0)) \cdot f'(x_0) + 0 \cdot f'(x_0) = g'(f(x_0)) \cdot f'(x_0). \end{aligned}$$

□

Tämän luvun lausekkeissa käytetään seuraavia merkintöjä:

w_{ij}^k	:	solmujen i ja j välinen paino kerroksella k
a_i^k	:	solmun i syötteiden ja vastaavien painojen tulo kerroksella k
o_i^k	:	solmun i aktivoitu tulos kerroksella k
n_k	:	solmujen lukumäärä kerroksella k
f	:	piilotettujen kerrosten aktivaatiofunktio
f_o	:	ulostulotason aktivaatiofunktio.

Lisäksi joka kerroksen harhayksikkö sisällytetään syötevektoriin ja tätä vastaavien painokertoimien symbolit ovat muotoa w_{0j}^k.

Huomataan, että tappiofunktio voidaan esittää yhdistettynä funktiona

$$E(x; w_{ij}^k) = \frac{1}{2N} \sum_{i=1}^{N} (f(a_i^k) - y_i)^2 = (E \circ a_i^k)(x; w_{ij}^k), \tag{24}$$

jolloin ketjusäännön avulla saadaan

$$\frac{\partial E(x; w_{ij}^k)}{\partial w_{ij}^k} = \frac{\partial E(x; w_{ij}^k)}{\partial a_i^k} \frac{\partial a_i^k}{\partial w_{ij}^k} = \delta_j^k \frac{\partial}{\partial w_{ij}^k} \sum_{i=0}^{n_{k-1}} w_{ij}^k o_i^{k-1} = \delta_j^k o_i^{k-1}, \tag{25}$$

jossa δ_j^k on kerroksen k solmun j *lokaali gradientti*. Tämän johdattelun mukaan on siis aluksi määritettävä δ_j^m, missä m viittaa neuroverkon ulostulokerrokseen. Nyt tappiofunktion määrittelyn avulla saadaan

$$\delta_j^m = \frac{\partial E(x; w_{ij}^m)}{\partial a_j^m} = \frac{\partial}{\partial a_j^m} \frac{1}{2} (f_o(a_j^m) - y_j)^2 = f_o'(a_j^m)(\hat{y}_j - y_j). \tag{26}$$

Sijoittamalla nyt yhtälön (26) tulos yhtälöön (25) saadaan

$$\frac{\partial E(x; w_{ij}^m)}{\partial w_{ij}^m} = f_o'(a_j^m)(\hat{y}_j - y_j) o_j^{m-1}. \tag{27}$$

Piilotetuilla kerroksilla lokaalien gradienttien laskeminen hyödyntää jo laskettuja gradientteja. Kerroksen $k \in 1, ..., m-1$ solmut ovat täysin yhdistettyjä seuraavan tason solmuihin. Tämä on otettava huomioon lokaalien gradienttien laskemisessa summaamalla yli kerroksen $k+1$ lokaalien gradienttien. [17] Tämän tiedon sekä ketjusäännön avulla saadaan

$$\frac{\partial E^n}{\partial a_j^k} = \sum_{i=1}^{n_{k+1}} \frac{\partial E^n}{\partial a_i^{k+1}} \frac{\partial a_i^{k+1}}{\partial a_j^k}. \tag{28}$$

Koska algoritmi aloitettiin viimeiseltä kerrokselta, kerroksella k lokaali gradientti δ_j^{k+1} on tunnettu. Sijoittamalla tämä ja $a_j^{k+1} = \sum_{t=0}^{n_k} w_{ij}^{k+1} f(a_j^k)$ yhtälöön (28) saadaan

$$\frac{\partial E^n}{\partial a_j^k} = \sum_{i=1}^{n_{k+1}} \delta_i^{k+1} \frac{\partial a_i^{k+1}}{\partial a_j^k} = \frac{\partial}{\partial a_j^k} \sum_{t=0}^{n_k} \delta_i^{k+1} w_{ij}^{k+1} f(a_i^k) = \sum_{t=0}^{n_k} \delta_i^{k+1} w_{ij}^{k+1} f'(a_i^k), \tag{29}$$

missä f' on kerroksen k aktivaatiofunktion derivaatta. Huomataan, että piilotettujen kerrosten lokaalit gradientit voidaan määrittää hyödyntämällä jo laskettuja

gradientteja. Lopuksi painokertoimet päivitetään yhtälön 17 mukaan jollakin $\eta > 0$, jonka valitsemiseen on useita eri tapoja. Usein painojen optimoinnissa käytetään myös klassisen NLM-mentelmän sijaan jotain sen muunnosta, kuten sen stokastista varianttia (*Stochastic Gradient Descent*).

Usein neuroverkkojen optimoinnissa NLM-algoritmin mukainen kohdefunktio ei ole konveksi, jolloin uniikkiia globaalia minimiä on huomattavasti vaikeampi löytää kuin konveksissa tapauksessa. NLM-algoritmi löytää lähestulkoon aina vain lokaalin minimin, jos kyse on epäkonveksista funktiosta. Stokastisessa NLM-menetelmässä hyödynnetään nimensä mukaan sattumaa algoritmin tehostamiseksi sekä mahdollisesti paremman minimin löytämieksi. Algoritmin ensimmäinen vaihe on tismalleen sama kuin perinteisessäkin NLM-algortimissa; painokertoimet alustetaan jollain arvoilla $\boldsymbol{\theta}^0$. Alustuksen jälkeen iteraatioilla $t = 1, 2, ...T$ valitaan satunnaisesti mielenkiinnon kohteena olevasta aineistosta ilman palautusta piste $\boldsymbol{x}^t$, minkä jälkeen painokertoimet $\boldsymbol{\theta}^{t+1}$ lasketaan päivityskaavan (17) mukaan. Algoritmin siis generoi approksimaatioita tutkittavan funktion todellisesta gradientista sattumanvaraisissa aineiston pisteissä. Tällainen menettely on huomattavasti paljon tehokkaampi kuin todellisen gradientin arvon laskeminen koko aineistosta jokaisella iteraatiokierroksella. [32]

Koulutusprosessin selkäranka on esitetyn vastavirta-algoritmin mukainen, mutta käytännössä on päätettävä, kuinka usein painokertoimia päivitetään ja kuinka kauan virhettä kumuloidaan. Koulutuksessa iteroidaan saatavilla olevan aineiston yli useita kertoja ja yhtä koko aineiston läpikäyntiä kutsutaan *epookiksi* (epoch). *Batch-oppimisessa* neuroverkolle syötetään koko aineisto kumuloiden samalla virhettä. Kun koko erä on virrannut verkon läpi, aloitetaan vastavirtaprosessi, jossa painokertoimet päivitetään. *Mini-batch-oppimisessa* neuroverkolle kerralla syötettävä aineiston määrä on pienempi. Aineisto jaetaan satunnaisesti etukäteen spesifioituun määrään joukkoja, jotka syötetään yksi kerraallaan neuroverkolle. Virhe kumuloidaan yksittäisen joukon yli ja parametrit päivitetään joka erän syötön jälkeen. *Online-oppiminen* on mini-batch-oppimisen erikoistapaus, jossa joukkojen lukumäärä on aineiston objektien lukumäärä. Neuroverkolle siis syötetään aineisto yksi piste kerrallaan, jonka jälkeen virheen perusteella parametrit päivitetään. [7]

Mini-batch-oppiminen lienee suosituin tapa kouluttaa neuroverkko, mutta eri menettelyillä on omat etunsa sekä haittansa. Batch-oppiminen on yksinkertaista implementoida, mutta vaatii koko aineiston tallentamisen titeokoneen muistiin. Usein syväoppivat neuroverkot vaativat valtavan määrän koulutusdataa, joten batch-learning ei ole aina mahdollinen vaihtoehto. Mini-batch-oppimisen etu on puolestaan huomat-

tavasti nopeammat koulutusajat kuin batch-oppimisessa. Toisaalta aineiston jakaminen alijoukkoihin satunnaisesti saattaa aiheuttaa painokerrointen oskillointia päivitysten yhteydessä. Online-oppiminen ei puolestaan ole laajalti käytössä menetelmän merkittävän satunnaisuuden vuoksi. Se on kuitenkin hyödyllinen sovelluksissa, jossa koulutus tehdään reaaliajassa ja koko aineisto ei ole alusta alkaen saatavilla. [7]

3.2 Konvoluutioneuroverkot

Tavallisessa neuroverkossa syötteet ovat vektorimuodossa. Näin ollen tämän tutkielman aineisto tulisi muuntaa vektoreiksi ennen neuroverkkoon syöttämistä. Tällöin ei kuitenkaan ole mahdollista hyödyntää pikseleiden keskinäisten sijaintien suomaa informaatiota hyväksi. Siksi kuvien analysoimisessa onkin suosittua käyttää konvoluutioneuroverkkoja, jotka matriisikonvoluution avulla mahdollistavat aineiston syöttämisen neuroverkolle matriiseina. [14]

3.2.1 Matriisikonvoluutio

Neuroverkkoa kutsutaan konvoluutioneuroverkoksi, mikäli se hyödyntää perineisen matriisitulon sijaan *matriisikonvoluutiota* vähintään yhdessä sen kerroksista. [6] Perinteisesti konvoluutio määritellään kahden funktion f ja g välisenä operaationa seuraavasti:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt. \tag{30}$$

Matriisikonvoluutio on puolestaan kahden matriisin välinen operaatio, jossa konvoluutio suoritetaan mielenkiinnon kohteena olevan matriisin ja *kernelin* välillä. Kerneli on toinen matriisi, jonka avulla konvoluutioneuroverkot voivat poimia kohdematriisista tiettyjä piirteitä ja yksityiskohtia. Kun näitä piirteitä louhitaan tarpeeksi, saadaan korkean abstaktiotason esitys konvoluutioverkolle syötetystä kuvasta, jonka perusteella lopullinen sovelluskohteesta riippuva operaatio tehdään syötteelle. [6], [4]

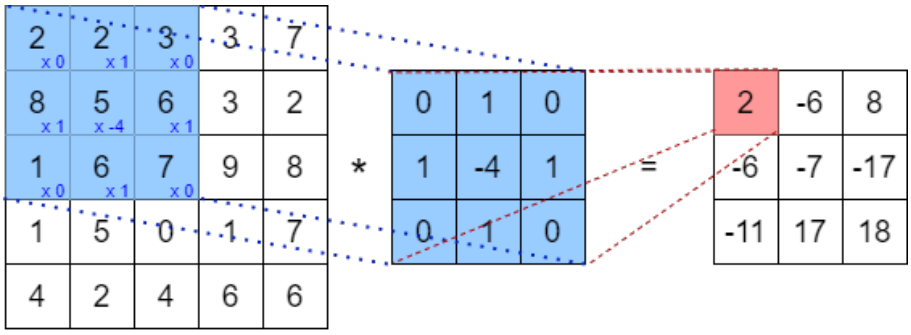
Määritelmä 1. Olkoon $X \in \mathbb{R}_{N\times M}$ ja $Y \in \mathbb{R}_{P\times Q}$ matriiseja. Lisäksi oletetaan, että $P < N$ ja $Q < M$. Matriisien X ja Y välinen konvoluutio on

$$(X * Y)(i,j) = \sum_{p=1}^{P}\sum_{q=1}^{Q} X(i+p-1, j+q-1)Y(p,q), \tag{31}$$

missä $1 \leq i \leq P$ ja $1 \leq j \leq Q$.

Yllä määriteltyä konvoluutiomatriisia kutsutaan oikeaoppisesti *ristikorrelaatioksi*, mutta konvoluutioverkkojen yhteydessä se kuitenkin tunnetaan yleensä konvo-

luutiona, sillä se on hyvin samankaltainen operaatio oikean matriisikonvoluution kanssa. [6] Käytännössä matriisikonvoluutiossa kernelimatriisi toimii ikään kuin "ikkunana", joka liukuu kohdematriisin yli kooten informaatiota tulosmatriisiin. Tätä vertauskuvaa havainnollistetaan kuvassa 6. Määritelmästä huomataan, että konvoluutiomatriisin dimensiot muuttuvat alkuperäisen matriisin dimensioista; konvoluutiomatriisi on $\mathbb{R}_{N-P+1 \times M-Q+1}$-matriisi.



Kuva 6: Visualisaatio konvoluution etenemisestä.

Esimerkki 2. Olkoon

$$X = \begin{pmatrix} 2 & 2 & 3 & 3 & 7 \\ 8 & 5 & 6 & 3 & 2 \\ 1 & 6 & 7 & 9 & 8 \\ 1 & 5 & 0 & 1 & 7 \\ 4 & 2 & 4 & 6 & 6 \end{pmatrix} \text{ ja } K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Niiden väliselle konvoluutiomatriisille saadaan

$$\begin{aligned} (X * K)(1,1) =& 2 \cdot 0 + 1 \cdot 2 + 0 \cdot 3 + 1 \cdot 8 - 4 \cdot 5 + 1 \cdot 6 + 0 \cdot 1 + 1 \cdot 6 + 0 \cdot 7 \\ =& 2 \\ (X * K)(1,2) =& 2 \cdot 0 + 1 \cdot 3 + 0 \cdot 3 + 1 \cdot 5 - 4 \cdot 6 + 1 \cdot 3 + 0 \cdot 6 + 1 \cdot 7 + 0 \cdot 9 \\ =& -6 \\ (X * K)(1,3) =& 0 \cdot 3 + 1 \cdot 3 + 0 \cdot 7 + 1 \cdot 6 - 4 \cdot 3 + 1 \cdot 2 + 0 \cdot 7 + 1 \cdot 9 + 0 \cdot 8 \\ =& 8. \end{aligned}$$

Samoin jatkamalla konvoluutiomatriisiksi saadaan lopulta 3×3 matriisi:

$$X * K = \begin{pmatrix} 2 & -6 & 8 \\ -6 & -7 & -17 \\ -11 & 17 & 18 \end{pmatrix}.$$

Muuttamalla kernel-matriisien elementtejä voidaan konvoluution kohteena olevista kuvista korostaa eri asioita. Tämän ominaisuuden avulla konvoluutioverkot kykenevät eristämään syötteistä piirteitä, joita voidaan edelleen käyttää muun muassa luokittelussa. Esimerkiksi *Sobel-filtterit* ovat varsin suosittuja tiettyyn suuntiin osoittavien reunojen korostamisesssa. [2] Horisontaalisia reunoja tunnistava Sobel-kerneli on muotoa

$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

ja vertikaalisten puolestaan muotoa

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}.$$

Kuvasta 7 nähdään selkeästi, kuinka konvoluutio vertikaalisen sobel-filtterin kanssa korostaa kuvan pystysuoria linjoja ja vastaava ilmiö voidaan havaita myös vaakasuorassa tapauksessa. Kuvia voidaan terävöittää käyttämällä esimerkiksi kerneliä

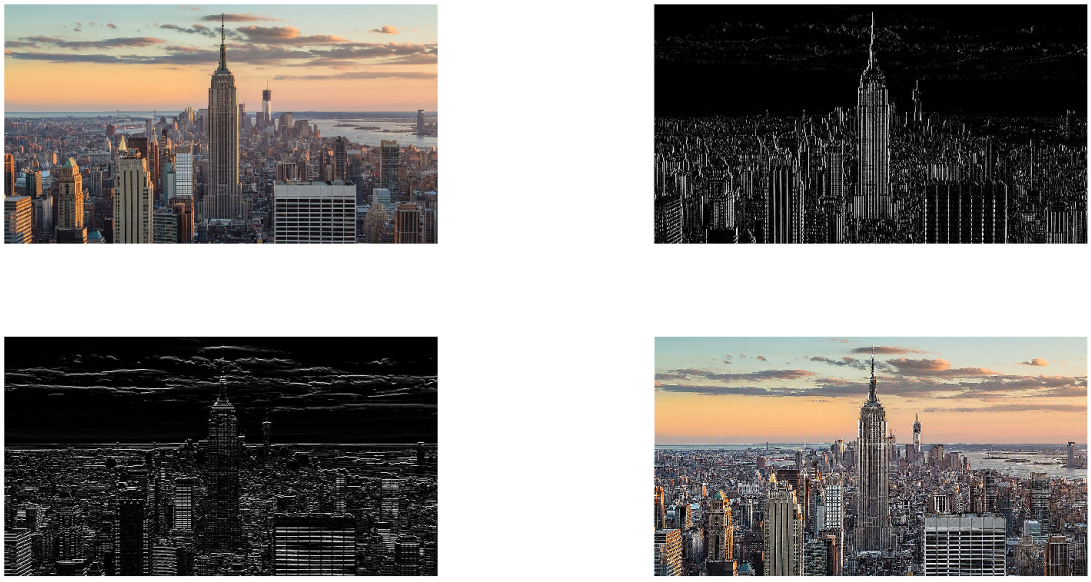
$$S = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

tai sumentaa kernelillä

$$B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

3.2.2 Pehmustaminen ja stride

Konvoluutiomatriisin dimensiot ovat alkuperäistä matriisia pienemmät, joten selvästi jotain informaatiota menetetään konvoluutiossa. Tämä johtuu siitä, että muokkaamattomassa kuvassa konvoluutiota ei pystytä hyödyntämään kuvan reunoilla tehokkaasti. Lisäksi syvissä konvoluutioverkoissa on useita konvoluutiokerroksia, jotka omalta osaltaan pienentävät syötteen kokoa jatkuvasti. Pahimmassa tapauksessa syöte kutistuu olemattoman pieneksi, jolloin oppiminen selvästikin hankaloituu. Näiden ongelmien välttämiseksi syötteet on usein hyödyllistä *pehmustaa* (padding). Syötteet kehystetään esimerkiksi sopivalla määrällä nollia, jotta konvoluutiota pystytään hyödyntämään myös kuvan reunoilla tehokkaasti ja syötteen liiallinen kutis-



Kuva 7: Konvoluution vaikutus kuvaan eri kerneleillä. Kuvat vasemmalta oikealle ja ylhäältä alas: alkuperäinen, vertikaalinen Sobel-kerneli, horisontaalinen Sobel-kerneli ja terävöittävä kerneli (alkuperäinen kuva: Wikimedia Commons [29]).

tuminen eliminoituisi. [8]

Esimerkki 3. Olkoon $I \in \mathbb{R}_{m\times n}$ matriisi ja $F \in \mathbb{R}_{f\times f}$ konvoluutiossa käytetty kerneli. Olkoon $p \in \mathbb{N}_0$ matriisin I pehmustamiseen käytettyjen rivien määrä. Olettaen, että pehmustaminen suoritetaan symmetrisesti, pehmustetun matriisin I dimensiot ovat nyt muotoa $m + 2p$ ja $n + 2p$. Tällöin konvoluutiomatriisin dimensiot ovat muotoa $m + 2p - f + 1$ ja $n + 2p - f + 1$.

Pehmustamista on useaa eri tyyppiä, mutta kenties intuitiivisin niistä on *saman pehmustamisen* idea (same padding). Siinä pehmustuksen määrä lasketaan siten, että konvoluutiomatriisin dimensiot ovat samat kuin sen alkuperäisen vastineen. [8] Siis jos alkuperäisen matriiisin dimensiot ovat m ja n ja konvoluutiossa käytettävän kernelin dimensiot ovat f_x ja f_y, niin konvoluutiomatriisin dimensiot ovat $m - f_x + 1$ ja $n - f_y + 1$. Tästä voidaan helposti päätellä tarvittavien pehmustusten määrät p_x ja p_y:

$$m = m + 2p_x - f_x + 1 \iff p_x = \frac{f_x - 1}{2}$$
$$m = m + 2p_y - f_y + 1 \iff p_y = \frac{f_y - 1}{2}.$$

Yleensä kernelit ovat neliömatriiseja, joiden dimensiot ovat parittomia, jolloin ei ole

pelkoa siitä, että pehmustavien rivien tai sarakkeiden lukumäärä ei olisi kokonaisluku.

Toinen konvoluutioverkoille ominainen konsepti on *stride*. Stridella tarkoitetaan konvoluutiossa kernelin vertikaalisen ja horisontaalisen liikkeen määrää konvoluution kohteena olevan matriisin yli. Oletusarvoisesti stride on konvoluutiossa symmetrinen sekä leveys- että korkeussuuntaan ja sen arvo on 1, jolloin kyseessä on tavallinen, jo aikaisemmin esitelty, konvoluutio. Joskus on kuitenkin hyödyllistä pienentää suurta syötematriisia kompaktimman esityksen aikaansaamiseksi, jolloin stridea voidaan kasvattaa arvoon s. Tällöin aina liikkuessaan kerneli siirtyy s pikseliä leveys- tai korkeussuuntaan. [8]

Esimerkki 4. Olkoon $I \in \mathbb{R}_{m \times n}$ matriisi ja $F \in \mathbb{R}_{f \times f}$ konvoluutiossa käytetty kerneli. Olkoon $p \in \mathbb{N}_0$ matriisin I pehmustamiseen käytettyjen rivien määrä ja $s \in \mathbb{N}_1$ symmetrinen stride. Oletetaan vielä, että vain ne yksittäiset konvoluutiot suoritetaan, jotka mahtuvat kokonaisuudessaan pehmustettuun matriisiin ja että pehmustus on symmetrinen. Koska käytössä on stride s, niin jokaisella konvoluutiomatriisin rivillä on ohitettu $s - 1$ konvoluutiota. Nyt konvoluutiomatriisin dimensioiksi saadaan $\frac{m+2p-f}{s} + 1$ ja $\frac{n+2p-f}{s} + 1$. Koska yksittäisistä konvoluutioista lasketaan vain ne, jotka mahtuvat kokonaisuudessaan syötematriisiin, voidaan vielä taata, että dimensiot ovat kokonaislukuja ottammalla lattiafunktiot dimensioiden murtolukuosuuksista, jolloin saadaan korkeudeksi $\lfloor \frac{m+2p-f}{s} \rfloor + 1$ ja leveydeksi $\lfloor \frac{n+2p-f}{s} \rfloor + 1$.

3.2.3 Konvoluutioverkon komponentit

Konvoluution lisäksi konvoluutioverkot sisältävät useita muita syötteelle suoritettavia operaatioita. Konvoluutio on yleensä vasta ensimmäinen osa verkon arkkitehtuuria ja usein syötteelle tehdäänkin monta konvoluutiota eri kerneleillä. Toisaalta perinteisempien koneoppimisongelmien kuten luokittelun kannalta konvoluutioverkkoon usein sisällytetään täysin yhdistettyjä kerroksia kuten monikerroksisissa perseptronimalleissa. [5]

Konvoluutiokerroksissa syötteelle suoritetaan konvoluutioita ottaen huomioon mahdolliset pehmustukset sekä stride. Usein kuvina syötetyt aineistot eivät kuitenkaan ole tavallisia kaksiulotteisia matriiseja, vaan aineistona voi olla värikuvia, jolloin kvantifioituina kuvat ovat useampiulotteisia matriiseja. Konvoluutioverkkojen yhteydessä niitä kutsutaan usein *tensoreiksi*. Esimerkiksi useimmat värikuvat ovat RGB-formaattia, jolloin punaiselle, vihreälle ja siniselle on jokaiselle oma *kanavansa* (color channel). Tällöin konvoluutiossa kernelien dimensioiden on vastattava värikanavien lukumäärää eli esimerkiksi RGB-kuvien kohdalla kerneleiden on oltava dimensiota $p \times q \times 3$. Sitten konvoluutio voidaan suorittaa normaalisti jokaiselle

värikanavalle erikseen, jonka jälkeen kolmen konvoluutiomatriisin vastaavat alkiot summataan keskenään yhteen ja lopputuloksena on kaksiulotteinen matriisi. Usein yhdessä konvoluutiokerroksessa suoritetaan kuitenkin monta konvoluutiota useiden eri kernelien kanssa. Jos siis kerneleiden lukumäärä on n_f, niin kaikkien konvoluutioiden jälkeen kerroksen tuloste on tensori, joka sisältää n_f kanavaa. [5]

Konvoluutiota seuraava toimenpide muistuttaa perseptronin toimintaa. Konvoluution tuloksena syntyneeseen tensoriin summataan alkioittain harhoja, jotka ovat konvoluutioverkon opittavia parametreja. Lopuksi tensoriin sovelletaan vielä jotain epälineaarista aktivaatiofunktiota alkioittain, jonka jälkeen tuloste syötetään verkon seuraavalle kerrokselle. Aktivaatiofunktioista varsinkin *ReLU*-funktio on hyvin laajalti käytössä konvoluutiokerroksissa. Harhojen lisäämisellä ja aktivaatiofunktioilla on konvoluutioverkoissa käytännössä sama rooli kuin perspetroniverkoissakin: epälineaarisuuden lisääminen verkon suorittamiin lineaarisiin operaatioihin. Kokonaisuudessaan konvoluutiota ja epälineaarista aktivaatiota kutsutaan yhdessa *detektorivaiheeksi* (detector stage). [5],[6]

Tyypillisessä konvoluutioverkossa edellä mainittuja vaiheita seuraa usein *pooling*-kerros. Pooling-kerroksessa jokin funktio ikään kuin kokoaa informaatiota tietystä syötteen kohdasta tiiviimpään muotoon esimerkiksi jonkin tilastollisen tunnusluvun muodossa. Pooling-operaatiota voidaan havainnollistaa samanlaisella kaaviokuvalla kuin konvoluutiotakin, mikä nähdään kuvasta 8. Poolingin ansiosta syötteestä tulee invariantti pienille siinä tapahtuville muutoksille. Koska pooling-kerrosten tulosteet ovat syötteen yleisempiä representaatioita, eivät pienet erot syötteessä vaikuta konvoluutioverkon tekemään lopputulokseen. Tämä on usein hyödyllistä tapauksissa, jossa ollaan kiinnostuneita jonkin tietyn ominaisuuden prevalenssista syötteessä eikä niinkään jonkin entiteetin tarkasta sijainnista. Tämän tutkielman aineiston valossa pooling-kerrokset olisivat hyödyllisiä konvoluutioverkossa, jonka tavoitteena olisi tunnistaa, sisältääkö syötetty kuva syöpäkudosta vai ei. [6]

Eri tarpeisiin on olemassa useita erityyppisiä pooling- funktioita. Myös pooling-operaatio suoritetaan konvoluution tavoin syötteen sekä dimensioiltaan spesifioiduin "ikkunan"välillä. Esimerkiksi eräs laajalti käytetty pooling- operaatio on *average pooling*- kerros. Siinä pooling- funktiona käytetään otoskeskiarvoa, joka lasketaan lokaalisti syötteestä suorakulmion muotoiselta alueelta. Toinen hyvin suosittu pooling- kerros on *max pooling*. Siinä tutkitaan jälleen syötteen lokaaleja suorakulmioita, joista valitaan pikselin arvo, joka on suurin. Myös pooling- kerroksissa voidaan käyttää konvoluutiokerroksista tuttuja pehmustuksia sekä stridea. Usein pooling- operaatioon liitetäänkin yhdestä poikkeava stride komptaktimman esitysmuodon aikaansaamiseksi. Toisin kuin konvoluutiossa, kolmiulotteisten tensoreiden

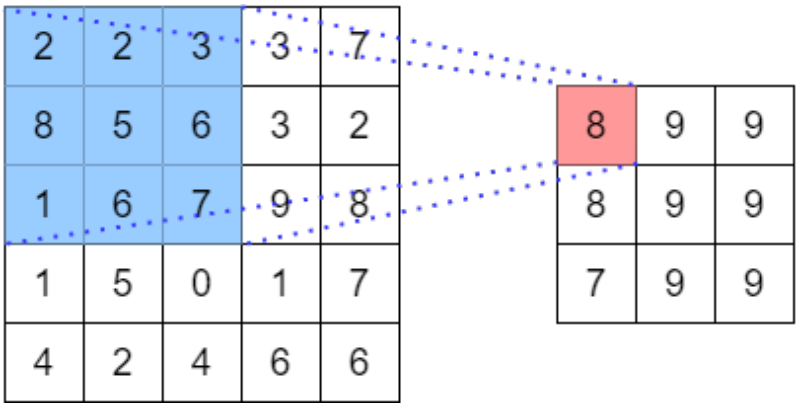
kohdalla pooling- operaatio suoritetaan jokaiselle kanavalle erikseen, jolloin syötteen kanavien lukumäärä vastaa pooling operaation- tuloksen kanavien lukumäärää. [6], [12]

Esimerkki 5. Tutkitaan erään konvoluutioverkon average pooling- kerroksen suorittamaa operaatiota syötematriisille X. Olkoon $X \in \mathbb{R}_{M\times N}$ ja P ja Q pooling-operaation dimensiot. Nyt pooling- operaatio voidaan esittää muodossa

$$P(i,j) = \frac{1}{pq}\sum_{p=1}^{P}\sum_{q=1}^{Q} X(i+p-1, j+q-1), \tag{32}$$

missä $P(i,j)$ on pooling- operaation tuloksena syntyvän matriisin P rivin i ja sarakkeen j alkio. Max pooling voidaan vastaavasti esittää muodossa

$$P(i,j) = \max\{X(i,j), X(i+1,j), ..., X(i+p,j), X(i,j+1), ..., X(i+p,j+q)\}. \tag{33}$$



Kuva 8: Visualisaatio max pooling- kerroksen operaatiosta.

Konvoluutiokerrokset ja pooling- kerrokset esiintyvät usein konvoluutioverkossa pareittain ja tätä konvoluutio-pooling- sekvenssiä yleensä toistetaan useamman kerran sovelluskohteesta riippuen. Neuroverkon loppuosa voi kuitenkin sovelluskohteesta riippuen vaihdella. Perinteisemmissä luokitteluongelmissa useita konvoluutio- ja pooling- kerroksia seuraa vaihteleva määrä täysin yhdistitettyjä perseptronikerroksia, joiden tarkoitus on konvoluutioverkon oppimien piirteiden perusteella luokitella syötteet. Ero klassiseen perspetroniverkkoon on kuitenkin se, että konvoluutioverkossa piirteet generoituvat automaattisesti neuroverkon toimesta. Perseptroniverkolle on syötettävä piirrevektori, joka on usein ihmisen konstruoima. Tämä on tosielämän sovelluksissa kallis proseduuri, minkä vuoksi konvoluutioverkot lienevät varsin suosittuja kuvantunniksen saralla. Konvoluutioverkoissa ihmisen tehtävänä on ainoastaan kiinnittää arkkitehtuuri. [5]

3.2.4 Konvoluutioverkon kouluttaminen

Kuten perinteisenkin neuroverkon kouluttamisessa, konvoluutioverkon koulutus koostuu kahdesta vaiheesta: syötteen kulkemisesta syötetasolta tulostetasolle ja virheen virtaaminen verkossa takaisinpäin. Konvoluutioverkkojen kohdalla syötteelle suoritetaan konvoluutio, jonka jälkeen konvoluution tulokseen sovelletaan vielä alkioittaain aktivaatiofunktiota, mitä seuraa vielä mahdollinen pooling-kerros. Verkon loppuosa voi sisältää sovelluksesta riippuen perseptronikerroksia, joissa syöte virtaa täysin samalla tavalla eteenpäin kuin perinteisissä neuroverkoissakin.

Vastavirta-osuus on konvoluutioverkoilla hieman poikkeava. Aiemmin esitettyä tavallista vastavirta-menettelyä voidaan soveltaa verkon MLP-osuuteen. Pooling-kerroksissa syötteeseen sovelletaan jotain kokoavaa funktiota, joten näissä kerroksissa ei ole koulutettavia parametreja. Konvoluutiokerroksessa on otettava huomioon, että syötteelle suoritettava operaatio on konvoluutio (ristikorrelaatio) eikä tavallinen matriisitulo. Täten lokaalien gradienttien, painokerrointen sekä harhojen päivityksen yhteydessä on myös käytettävä ristikorrelaatio-operaattoria. Käytännössä konvoluutioverkkojen koulutuksessa vastavirta-algoritmin yhteydessä käytetään yleisimmin optimointialgoritmina stokastista nopeimman laskeutumisen menetelmää, jossa stokastisuus aiheutuu satunnaistamalla mini-batch joukkojen generointi epookkien välillä. [13]

3.2.5 Konvoluutioverkon hyperparametrit

Konvoluutioverkoissa on monta tekijää, jotka on päätettävä etukäteen ennen mallin kouluttamista. Näitä tekijöitä kutsutaan mallin *hyperparametreiksi* ja ne määräävät käytännössä mallin arkkitehtuurin. Arkkitehtuurin selkärangan muodostavat konvoluutiokerrosten sekä mahdollisten piilotettujen kerrosten lukumäärät.

Konvoluutioissa käytettävien kernelien dimensiot ovat keskeisessä roolissa syötteen dimensioiden kontrolloimisessa. Usein kernelit ovat varsin pieniä neliömatriiseja (tai tensoreita), joiden dimensiot ovat parittomia. Tällöin konvoluutiomatriisien dimensiot pysyvät kokonaislukuina. Toinen kerneleihin liittyvä hyperparametri on niiden lukumäärä eri kerroksilla. Usein syötteeseen sovelletaan monia kerneleitä yhdellä konvoluutiokerroksella usean piirteen eristämiseksi samanaikaisesti.

Pooling-kerrokset eivät sisällä opittavia parametreja, mutta niiden suhteen on kuitenkin tehtävä useampi päätös ennen kouluttamista. Kuten konvoluutiokerroksissa, myös pooling-kerrosten kernelien dimensiot on määritettävä. Dimensioiden lisäksi on päätettävä, mitä pooling-menetelmää käytetään. Näiden lisäksi sekä konvoluutioettä pooling-kerroksissa on päätettävä mahdollisista pehmustuksista sekä stridesta.

Myös konvoluutiokerrosten sekä piilotettujen kerrosten aktivaatiofunktiot las-

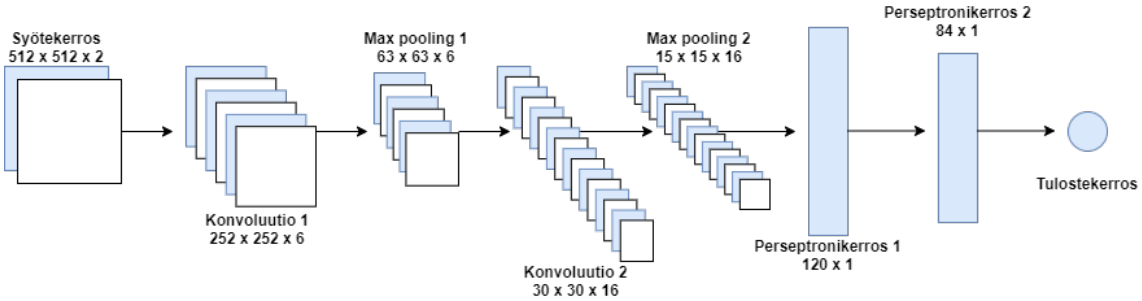
ketaan hyperparametreiksi. Usein koko verkossa käytettään viimeistä kerrosta lukuunottamatta samaa aktivaatiofunktiota. [5] *ReLU*-funktiolla on erityisen suosittu rooli konvoluutioverkoissa, mutta muitakin vaihtoehtoja on olemassa. Viimeisen kerroksen aktivaatio on sovelluskohteen mukainen eli esimerkiksi binäärisessä luokitteluongelmassa *sigmoid*-funktio tai regressiosssa identiteettifunktio.

Useaa konvoluutiokerrosta seuraa tämän tutkielman tapauksessa joukko täysin yhdistettyjä perseptronikerroksia. Näiden suhteen on jokaiselle kerrokselle päätettävä yksiköiden lukumäärä. Viimeisen kerroksen yksiköiden lukumäärä vaihtelee jälleen sovelluskohteesta riippuen, mutta binääriluokitteluongelmassa viimeisellä kerroksella riittää vain yksi *sigmoid*-funktiolla varustettu tulosteyksikkö.

Kun verkon arkkitehtuuri on kiinnitetty, on vielä suunniteltava koulutusprosessin kulku tarkkaan. Tämä sisältää päätöksen siitä, käytetäänkö koulutuksessa batch-, mini-batch- vai online-oppimista. Lisäksi on päätettävä vastavirta-algoritmin yhteydessä käytettävä optimointialgoritmi sekä määritttävä siihen liittyvien hyperparametrien arvot. Esimerkiksi stokastisen NLM-menetelmän päivityskaava on yhtälön (17) mukaan muotoa

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \frac{\partial E(X; \boldsymbol{\theta}^t)}{\partial \boldsymbol{\theta}}. \tag{34}$$

Askelpituuden η valitsemiseen on olemassa useita menetelmiä, ja useimmiten oppimisalgoritmin yhteydessä askelpituutta päivitetään interaktiivisesti, jotta vältytään *katoavan gradientin ongelmalta* (vanishing gradients). Lisäksi NLM-menetelmä vaatii, että opittavat parametrit on alustettu joillain alkuarvoilla. Yksinkertaisimillaan ne voidaan alustaa joillain pienillä satunnaisilla luvuilla, mutta alkuarvojen valitsemiseen on myös olemassa hienostuneimpia proseduureja.



Kuva 9: Graafinen esitys LeNet-arkkitehtuurista modifioituna tutkielman aineiston dimensioille. Kuvassa tekstit kertovat syötteen dimension konvoluutioverkon eri kerroksilla.

Esimerkki 6. Tutkitaan kuvan 9 LeNet-arkkitehtuurin opittavien parametrien lukumäärää, kun syötteenä on tutkielman aineiston $512 \times 512 \times 2$ tensoreita. Oletetaan, että pehmustusta ei käytetä. Otetaan taulukossa 1 käyttöön seuraavat merkinnät:

Kerroksen nimi	Hyperparametrit	Tulos	Parametrien lukumäärä
Syötekerros		$512 \times 512 \times 2$	0
Konvoluutio 1	$s = 2$ $d_k = 9 \times 9 \times 2$ $n_f = 6$	$252 \times 252 \times 6$	$9{\times}9 \times 2 \times 6 + 6 = 978$
Max pooling 1	$s = 4$ $d_k = 4 \times 4$ $n_f = 6$	$63 \times 63 \times 6$	0
Konvoluutio 2	$s = 2$ $d_k = 5 \times 5 \times 6$ $n_f = 16$	$30 \times 30 \times 16$	$5 \times 5 \times 6 \times 16 + 16 = 2416$
Max pooling 2	$s = 2$ $d_k = 2 \times 2$ $n_f = 16$	$15 \times 15 \times 16$	0
Perseptronikerros 1	Neuroneita = 120	120×1	$15 \times 15 \times 16 \times 120 + 120$ $= 432120$
Perseptronikerros 2	Neuroneita = 84	84×1	$120 \times 84 + 84 = 10164$
Tulostekerros	$n_p = 1$	1×1	$84 \times 1 + 1 = 85$

Taulukko 1: Kuvan 9 vastaavan konvoluutioverkon parametrien lukumäärät.

s : stride
d_k : kernelin dimensio
n_k : kerneleiden lukumäärä.

Yhteenlaskut taulukossa 1 parametrien lukumäärissä seuraavat harhojen summaamisista konvoluution sekä perseptronikerroksissa sisätulon tuloksiin. Yhteensä mallissa on siis 445763 optimoitavaa parametria.

3.2.6 Konvoluutioverkkojen edut

Konvoluutioverkoilla on useita ominaisuuksia, jotka tekevät niistä otollisempia vaihtoehtoja kuvien analysoiintiin perinteisten neuroverkkojen sijaan. Tärkeimpänä voidaan pitää konvoluutioverkkojen kykyä käsitellä minkä kokoisia objekteja tahansa, kunhan niiden dimensiot aineiston sisällä ovat koherentteja. Tällöin säilytetään pikseleiden arvon lisäksi spatiaalinen informaatio, joka riippuu niiden fyysisistä sijainneista syötteessä.

Perinteisessä monikerroksisessa perseptroniverkossa jokaisella tasolla jokainen yksikkö linkittyy jokaiseen seuraavan tason yksikköön. Tämä aiheuttaa usein syvissä perseptroniverkoissa optimoitavien parametrien lukumäärän kombinatorisen räjähtämisen, joka puolestaan heikentää verkon suorituskykyä. Konvoluutioverkot tarjoavat vastauksen tähän ongelmaan, sillä useimmiten niissä kerrosten väliset lin-

kit ovat harvoja (*sparse connectivity*). Käytännössä tämä on mahdollista toteutttaa valitsemalla konvoluution kernelin dimensiot pienemmiksi kuin syötteen dimensiot. Tällöin konvoluution vältyksellä neuroverkko eristää tuhansien pikseleiden syötteistä abstrakteja piirteitä samalla pienentäen syötteen kokoa. Tämä vähentää parametrien lukumäärää huomattavasti, mikä puolestaan tekee mallista yksinkertaisemman ja tilastollisesti tehokkaamman. Käytännön toteuksen kannalta tällä on monia hyötyjä. Esimerkiksi mallin säilöminen vie vähemmän muistia ja parametrien optimointi vaatii pienemmän määrän laskutoimituksia. [6]

Perseptroniverkoissa jokaisella yksiköllä on yksi painokerroin, jota käytetään verkossa kerran. Sen sijaan konvoluutioverkoissa parametrit ovat kerroksittain yhteiskäytössä, sillä jokaisessa suoritetussa konvoluutiossa kernelin painokertoimia sovelletaan kaikkiin syötteen pikseleihin (parameter sharing). Tämä pienentää entisestään konvoluutioverkon optimoimiseen tarvittavaa muistia sekä parantaa optimointialgoritmin tehokkuutta ja parametrien laatua. [6]

3.2.7 Siirtovaikutus konvoluutioneuroverkoissa

Syväoppivien neuroverkkojen heikkous lienee se, että ne vaativat valtavan määrän dataa löytääkseen mahdollisimman optimaaliset parametrien arvot. Kuten esimerkissä 6 huomattiin, jo varsin matalassa konvoluutioverkossa on satoja tuhansia optimoitavia parametreja. Usein käytännössä tarpeeksi suuren aineiston kerääminen on kuitenkin liian kallista, jolloin aineistoa ei ole neuroverkon kouluttamiseen tarpeeksi. Tämän ongelman ratkaisemiseksi on useita vaihtoehtoja kuten esimerkiksi datan kasvattaminen keinotekoisesti augmentoimalla saatavilla olevaa aineistoa. Toinen vaihtoehto on hyödyntää syväoppivien neuroverkkojen *siirtovaikutusta*, kuten tässä tutkielmassa tehdään. [5]

Siirtovaikutus mahdollistaa syväoppivien neuroverkkojen käytön myös pienempien aineistojen ohella. Ajatus siirtovaikutuksesta perustuu siihen, että neuroverkko eristää koulutusvaiheessa datasta yleishyödyllisiä piirteitä, joita voi mahdollisesti uudelleenkäyttää eri kontekstissa. Käytännössä tämä tarkoittaa sitä, että neuroverkko koulutetaan aluksi jollain valmiiksi annotoidulla aineistolla, joka on kooltaan tarpeeksi suuri. Tällaisia aineistoja on esimekriksi *ImageNetin ILSVRC-aineisto* (*ImageNet Large-Scale Visual Recognition Challenge*), joka sisältää yli miljoona annotoitua kuvaa, jotka on jaettu tuhanteen eri luokkaan. [19] Sitten mallin parametrit koulutetaan alkuperäiseen tarpeeseen saatavilla olevalla pienemmällä aineistolla. [5]

Kun valittu konvoluutioverkkomalli on koulutettu suurella aineistolla, neuroverkon viimeiseen perseptronikerrokseen tehdään muutoksia, jotta se vastaisi alkuperäistä sovelluskohdetta. Esimerkiksi ImageNetin ILSVRC-aineistolla koulutettaes-

sa konvoluutioverkon viimeisellä kerroksella on oltava 1000 yksikköä, yksi jokaiselle luokalle, $softmax$-funktiolla varustettuina. Kuitenkin tässä tutkielmassa käsiteltävä luokitteluongelma on binäärinen, mikä vaatii vain yhden $sigmoid$-funktiolla varustetun yksikön tulostekerroksella. Sopivien muutosten jälkeen esikoulutettu neuroverkko voidaan kouluttaa uudelleen pienemmällä aineistolla. Kaikkia kerroksia ei tarvitse uudelleenkouluttaa. Esimerkiksi joissain tapauksissa saattaa olla mielekästä uudelleenkouluttaa ainoastaan konvoluutioverkon perseptroniosuus ja käyttää konvoluutiokerroksia piirteiden eristämiseen, jolloin neuroverkko on käytännössä perseptroniverkko, joka eristää automaattisesti piirteet (niitä ei siis tarvitse konstruoida käsin). [5]

4 Mallin valinta, koulutus ja evaluointi

Tutkielman empiirisessä osassa tutkitaan, kuinka hyvin konvoluutioneuroverkot suoriutuvat PET-MRI-kuvien luokittelusta. Lisäksi tutkitaan, soveltuvatko valmiiksi koulutetut mallit tämänkaltaisen ongelman ratkaisemiseen. Tutkielmassa koulutetaan kaksi mallia ja analysoidaan niiden kouluttamista sekä suoritusta testiaineiston luokittelussa.

4.1 Aineiston esikäsittely

MRI- ja PET- kuvat eivät käytännössä sellaisinaan sovellu käytettäväksi koneoppimisessa, mutta raakadata voidaan normalisoida konvoluutioneuroverkoille käyttökelpoisempaan muotoon. Ongelmat raakadatassa aiheutuvat pikseliarvojen suurista vaihteluista. Esimerkiksi globaali maksimi kaikkien saatavilla olevien PET- kuvien pikseleiden yli on arvoltaan noin 24000 yksikköä, kun taas pienin mahdollinen PET-aktiivisuustaso saa arvon nolla. Magneettikuvissa vastaavat arvot ovat 2800 yksikköä ja nolla.

Suurien pikseliarvojen välttämiseksi raakadata normalisoidaan globaalisti välille $[0, 255]$ lauseen 3 avulla, jotta kuvat voisi mieltää perinteisten RGB-kuvien kaltaisiksi. Jotta PET- aktiivisuuden suoma informaatio saadaan pysymään ennallaan, normalisoidaan PET- ja MRI- kuvat toisistaan erillään. Lopuksi kaksiulotteisten kuvien vastinpareista muodostetaan kolmiulotteisia tensoreita, jotka syötetään neuroverkkomalleille. Siirtovaikutusta tutkittaessa käytetään mallia, jossa syötteen on sisällettävä kolme värikanavaa. Tämän vuoksi esikoulutettua mallia käytettäessä tensoreihin konkatenoidaan kolmannessa ulottuvuudessa vielä yksi kerros, joka sisältää pelkästään nollia. Näin syöte saadaan yhteensopivaksi neuroverkkomallille vääristämättä aineistoa lainkaan.

Lause 3. *Olkoon X kaksiulotteisten $n \times m$-matriisien joukko ja olkoon kaikkien matriisien alkiot ei-negatiivisia kokonaislukuja. Oletetaan lisäksi, että kaikkien matriisien kaikki alkiot eivät ole yhtä suuria. Olkoon $U = \{1, ..., n\}$ ja $V = \{1, ..., m\}$. Matriisit voidaan normalisoida globaalisti välille $[0, 1]$ alkioittain kaavalla*

$$x_{ij}^{norm} = \frac{x_{ij} - \min\limits_{x \in X} \left\{ \min\limits_{u \in U, v \in V} x_{uv} \right\}}{\max\limits_{x \in X} \left\{ \max\limits_{u \in U, v \in V} x_{uv} \right\} - \min\limits_{x \in X} \left\{ \min\limits_{u \in U, v \in V} x_{uv} \right\}}, \tag{35}$$

missä $i \in U$ ja $j \in V$.

Todistus. Olkoon x eräs joukon X matriisin alkio. Merkitään $a = \min\limits_{x \in X} \left\{ \min\limits_{u \in U, v \in V} x_{uv} \right\}$ ja $b = \max\limits_{x \in X} \left\{ \max\limits_{u \in U, v \in V} x_{uv} \right\}$. Lukujen a ja b määritelmistä seuraa, että

$$a \leq x \leq b,$$

ja vähentämällä puolittain luvulla a saadaan edelleen

$$0 \leq x - a \leq b - a.$$

Jakamalla puolittain luvulla $b - a > 0$ saadaan lopuksi

$$0 \leq \frac{x - a}{b - a} \leq 1,$$

mikä todistaa lauseen tuloksen. □

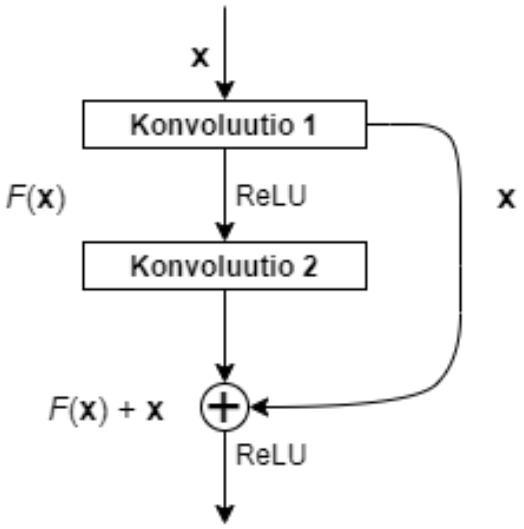
4.2 Mallien arkkitehtuurit

Konvoluutioverkkojen siirtovaikutuksen tutkimiseksi MRI-PET- aineistolle koulutetaan kaksi mallia. Yksi malleista rakennetaan arkkitehtuuritasolta lähtien, kun taas osassa kokeista käytetään valmiiksi koulutettua mallia, jonka arkkitehtuuri on viimeistä kerrosta lukuunottamatta kiinnitetty. Esikoulutettuja malleja on useaa eri tyyppistä ja jokaisella mallilla on vahvuutensa ja heikkoutensa. Tässä työssä valmiiksi malliksi valikoitui kuitenkin *ResNet50*-neuroverkko, sillä RGB-kuvilla koulutettua mallia on aikaisemmin onnistuneesti käytetty lääketieteellisten (mustavalkoisten) kuvien luokittelussa. [1] ResNet50-mallilla on se etu, että normaalista poiketen malliin voi sisällyttää suuren määrän kerroksia kuitenkin säilyttäen koulutusprosessin tehokkaana. Kustomoiduksi malliksi räätälöitiin puolestaan varsin yksinkertainen perinteinen konvoluutioneuroverkko.

4.2.1 ResNet50

Syvillä neuroverkoilla tarkoitetaan malleja, jotka sisältävät selkeästi enemmän kerroksia kuin perinteisemmät neuroverkkomallit. Syvillä neuroverkkomalleilla on onnistuttu peittoamaan perinteiset kuvantunnistumallit esimerkiksi konenäkömaailman suurimassa kilpailussa ILSVRC:ssä. [28] Syvien konvoluutioverkkojen koulutus on kuitenkin laskennallisesti hyvin raskasta, joten tämän ongelman ratkaisemiseksi on kehitetty *residuaalimalleja*, joista kuuluisimmat lienevät *ResNet*-mallit. [16]

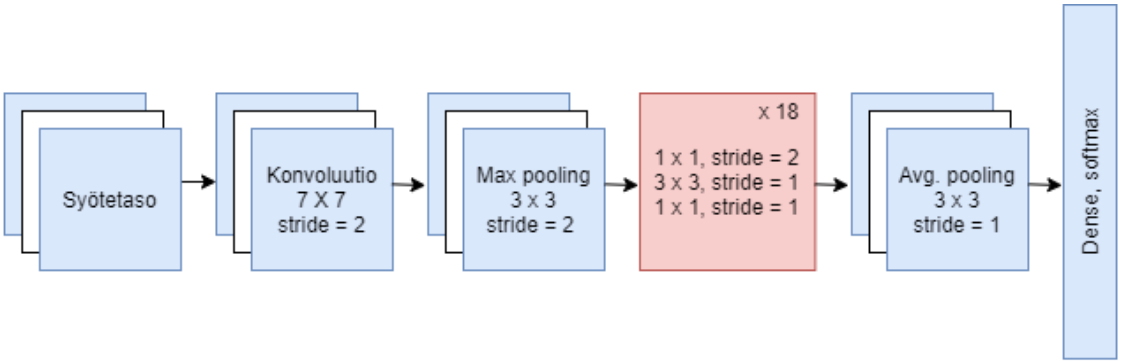
Koneoppimisessa yleisesti tavoitteena on oppia jokin syötteen kuvaus $\mathcal{H}(x)$. Residuaalioppimisessa kuvauksen $\mathcal{H}(x)$ sijaan approksimoidaan residuaalikuvausta $\mathcal{F}(x) = \mathcal{H}(x) - x$, jonka jälkeen alkuperäinen kuvaus saadaan residuaalikuvaksen avulla ilmaistuna $\mathcal{H}(x) = \mathcal{F}(x) + x$. Teoriassa kummankin funktion avulla on mahdollista mallintaa sovelluksen kohteena olevaa ilmiötä, mutta kuvauksen oppimisen vaivattomuus saattaa vaihdella tapauksittain. Tehoton koulutusprosessi luonnollisesti heikentää mallin suorituskykyä. [16]



Kuva 10: Graafinen esitys ResNet-mallien residuaalioppimista implementoivista komponenteista, missä $\boldsymbol{x}$ edustaa syötettä ja $\mathcal{F}(\boldsymbol{x})$ opittavaa residuaalikuvausta.

Käytännössä residuaalioppiminen implementoidaan rakentamalla neuroverkko kuvan 10 komponenteista. Ideana on rakentaa malli residuaalikuvauksen oppimiselle. Residuaalikuvauksesta päästään takaisin mielenkiinnon kohteena olevaan kuvaukseen summaamalla syöte alkioittain residuaalikuvauksen kanssa. Tämä menettely ei lisää mallin opittavien parametrien lukumäärää lainkaan, kunhan syötteen ja residuaalikuvauksen summa on määritelty. ResNet50-arkkitehtuurissa residuaalikomponentti sisältääkin kahden konvoluutiokerroksen sijaan kolme kerrosta, jolloin

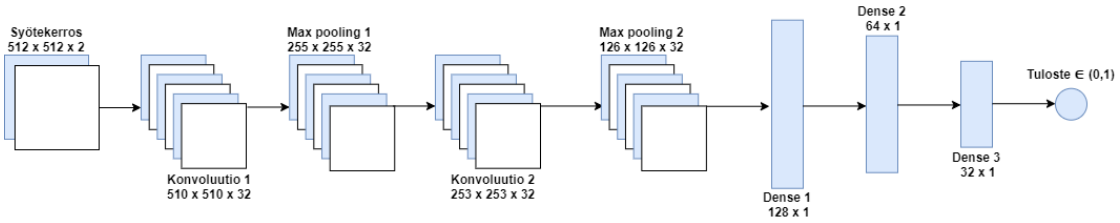
kolmen kerroksen joukkoa kutsutaan *pullonkaulauksi*. Neuroverkon arkkitehtuuri esitellään tarkemmin kuvassa 11. [16]



Kuva 11: Yksinkertaistettu kaaviokuva alkuperäisestä ResNet50-mallista. Punainen laatikko esittää kolme konvoluutiota sisältävää residuaalikomponenttia ja kerneleiden dimensiot on lueteltuin laatikossa. Komponentteja on mallissa yhteensä 18 peräkkäin, jolloin mallissa on syöte- ja tulostekerroksia lukuunottamatta $2+3\times18 = 50$ kerrosta. [16]

4.2.2 Perinteinen konvoluutioneuroverkko

Alusta alkaen koulutettu malli on arkkitehtuuriltaan perinteinen konvoluutioverkko, mikä nähdään kuvasta 12. Verkon alussa konvoluutiokerrokset sekä pooling-kerrokset eristävät esikäsitellystä datasta piirteitä, joita täysin yhdistetty perseptroniverkko ottaa syötteeksi. Viimeisellä kerroksella sigmoid-funktion avulla malli laskee syötteenä annetulle tensorille regressionomaisesti reaalilukuarvon, jonka perusteella binääriluokittelu suoritetaan. Konvoluutiokernelit ovat dimensioiltaan 3×3 ja pooling-kerroksissa puolestaan 2×2. Konvoluutiokerroksissa käytetään vertikaalisen ja horisontaalisen striden arvoa yksi, kun taas pooling-kerroksissa vastaava arvo on kaksi. Aktivaatiofunktioina käytetään pääsääntöisesti ReLU-funktiota paitsi viimeisellä kerroksella käytetään binääriluokittelulle tyypillistä sigmoid-funktiota.



Kuva 12: Graafinen esitys koulutetusta mallista sekä syötteen dimensioista neuroverkon eri osissa. Kuvassa 'Dense'- kerrokset viittavat täysin yhdistettyihin perseptronikerroksiin.

4.3 Kouluttaminen ja validointi

4.3.1 Datan jakaminen

Koulutusprosessi aloitetaan määrittämällä koulutus-, validointi- sekä testiaineistot, jotka ovat varsinaisen aineiston osajoukkoja. Yksinkertaisimillaan aineiston jakamisen koulutus- ja testijoukkoihin voidaan suorittaa satunnaisotannalla ja sisällyttämällä tietty osuus aineistosta testijoukkoon. Tässä sovelluskohteessa on kuitenkin otettava huomioon, että mikäli tensoreita, jotka ovat peräisin samalta potilaalta, päätyy sekä testi- että koulutusjoukkoon, voi mallin evaluoinnin yhteydessä määritetyt metriikat olla hyvinkin harhaisia. Tätä ilmiötä kutsutaan yleisesti koneoppimisen viitekehyksessä *datan vuotamiseksi* (*data leakage*). [10]

Datan vuotaminen on tässä tutkielmassa vältetty satunnaistamalla koulutus-, validointi- ja testijoukkojen valinta yksittäisten potilaiden suhteen. Jaossa positiivisilta potilailta käytettiin kaikki syöpää sisältävät leikkeet ja negatiivisilta potilailta valittiin leikkeitä satunnaisesti siten, että joka potilaalta käytettiin neljä leikeittä. Näin varmistettiin, että koulutus-, validointi- ja testiaineistoihin saatiin likimain saman verran positiivisia sekä negatiivisia esimerkkejä. Lisäksi, koska vain yhdellä positiivisista potilaista löytyi syöpäkudosta nenänielun alueelta, negatiivisten leikkeiden valinnassa ei otettu huomioon 20% ylimpiä pään ja nenänielun yläpuolen alueella sijaitsevia leikkeitä. Koulutusaineistoon valittiin satunnaisesti 8 positiivista ja 29 negatiivista potilasta. Vastaavasti validointi- ja testijoukkoihin valittiin kumpaankin 4 positiivisen ja 8 negatiivisen potilaan leikkeistä muodostetut tensorit.

	Positiiviset	Negatiiviset	Yhteensä
Koulutus	86	116	202
Validointi	43	32	75
Testi	49	32	81
Koko aineisto	178	180	358
Potilaiden lukumäärät	18	45	63

Taulukko 2: Esimerkkien lukumäärät koulutus-, validointi- ja testiaineistoissa sekä koko aineistossa yhteensä.

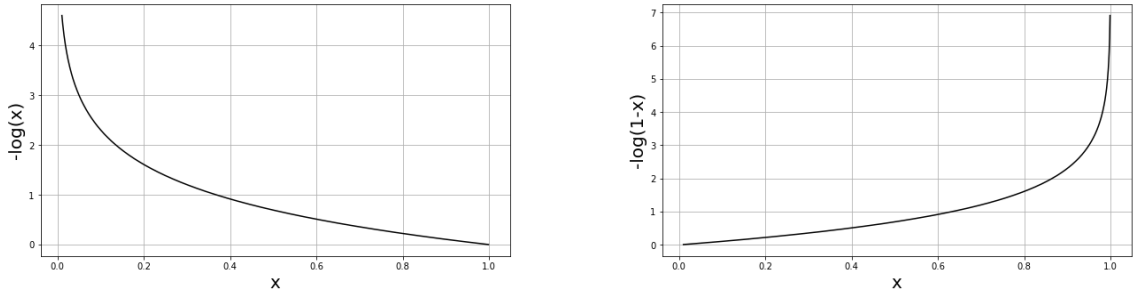
4.3.2 Tappiofunktio

Koska sovelluskohteena oleva luokitteluongelma on luonteeltaan binäärinen, ei aiemmin esiteltyä keskineliövirhettä ole mielekästä käyttää tappiofunktiona. Sen sijaan tässä tutkielmassa koulutusvaiheen tappiofunktioksi on valittu *binäärinen ristientropia* (*binary cross-entropy*), joka on tyypillinen tappiofunktio binääriluokittelussa.

Määritelmä 2. Olkoon $\boldsymbol{y}$, $y_i \in \{0,1\}$ ja $\hat{\boldsymbol{y}}$, $\hat{y}_i \in (0,1)\ \forall i \in \{1,...,N\}$. Vektoreiden $\boldsymbol{y}$ ja $\hat{\boldsymbol{y}}$ välinen binäärinen ristientropia saadaan kaavasta

$$\mathcal{H}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \tag{36}$$

Binäärisen ristientropian ideana on mitata jakaumien $\boldsymbol{y}$ ja $\hat{\boldsymbol{y}}$ similaarisuutta. Täten käyttämällä sitä tappopfunktiona konvoluutioverkon kouluttamisessa tavoitteena on minimoida oikeiden leimojen ja ennusteiden välistä eroavaisuutta. Tulkinnallisesti ristientropia on intuitiivinen; kun malli tunnistaa esimerkin oikein positiiviseksi, malli tuottaa kyseiselle esimerkille regressioarvon, joka on lähellä luku yksi. Tällöin kaavasta 36 sekä kuvasta 13 nähdään, että summan toinen termi saa arvon nolla ja ensimmäinen termi puolestaan itseisarvoltaan pienen arvon. Mikäli malli tunnistaisi väärin, että esimerkki on negatiivinen todellisen leiman ollessa positiiviinen, summan ensimmäisen termin logaritmi saa itseisarvoltaan suuren arvion, sillä similaarisuus todellisen leiman ja ennustetun arvon välillä pienenee. [24]



Kuva 13: Binäärisen ristientropian komponenttien kuvaajat välillä $x \in (0,1)$.

4.3.3 Optimointi

Keskeisenä optimointialgoritmina kummankin mallin koulutuksessa käytetään stokastista nopeimman laskeutumisen menetelmää (*stochastic gradient descent, SGD*) ja minimoitavana kohdefunktiona on edellä esitelty ristientropia. Kustomoidun mallin kohdalla mallia koulutetaan 10 epookin verran vakioaskelpituudella $\eta = 0{,}0001$. ResNet50-mallin koulutuksessa on otettava huomioon, että mallin parametrit ovat valmiiksi kiinnitettyjä ja jotta siirtovaikutuksen hyödyntämisen mahdollisuutta päästään tutkimaan, on koulutusrutiinia hieman muokattava. ResNet50-malli on alunperin koulutettu käyttämällä aineistolle, joka sisältää RGB-kuvia 10 eri luokasta. Jotta mallia pystyttiin hyödyntämään binäärisessä luokittelussa, on viimeisen kerroksen 10 softmax-yksikköä korvattava yhdellä sigmoid-yksiköllä. Tämän vuoksi viimeisen kerroksen parametrit on koulutettava uudelleen, mutta loput mallin parametreista

pidetään ennallaan. Viimeisen kerroksen kertoimet määritetään niin ikään käyttämällä SGD:tä 10 epookin ajan askelpituudella $\eta = 0{,}0001$.

ResNet50-malli on optimoitu suoriutumaan parhaalla mahdollisella tavalla *CIFAR-10*-aineiston luokittelusta. [16] Tämän vuoksi on syytä vielä viimeisen kerroksen parametrien koulutksen jälkeen hienosäätää koko mallin parametrit tämän tutkielman PET-MRI-aineiston luokitteluun sopivaksi. Tämä toteutetaan käytännössä jatkamalla mallin koulutusta vielä viiden epookin verran entistä pienemmällä askelpituudella $\eta = 1 \cdot 10^{-}5$. Tarkoitus ei kuitenkaan ole uudelleenkouluttaa koko mallia, vaan muokata marginaalisesti sen parametreja sopivimmiksi.

4.3.4 Metriikat

Kenties intuitiivisin luokittelevan mallin hyvyyden mittari on (binäärinen) tarkkuus (*Binary Accuracy*) eli oikein menneiden ennustusten osuus kaikista tarkasteltavista esimerkeistä.

Määritelmä 3. Olkoon $\boldsymbol{y}$, $y_i \in \{0, 1\}$ ja $\hat{\boldsymbol{y}}$, $\hat{y}_i \in \{0, 1\}$ $\forall i \in \{1, ..., N\}$. Todellisten leimojen ja ennustusten välinen tarkkuus on

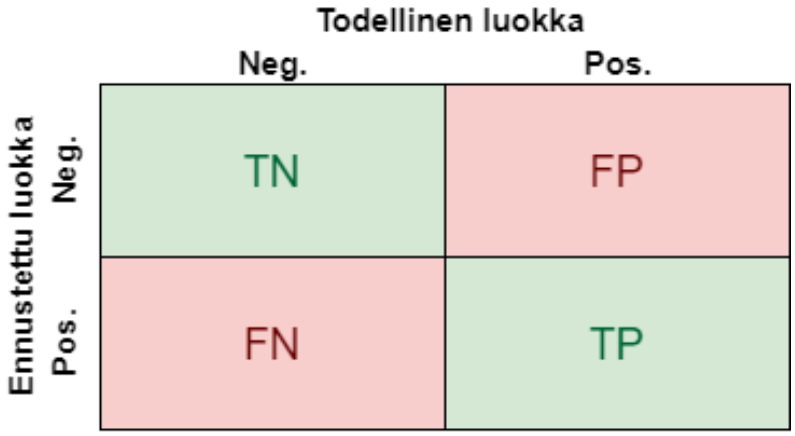
$$ACC = \frac{1}{N} \sum_{i=1}^{N} I(y_i = \hat{y}_i), \tag{37}$$

missä

$$I(x = y) = \begin{cases} 1, & \text{kun } x = y \\ 0, & \text{muutoin.} \end{cases}$$

Binääriluokittelussa koneoppimismallit usein generoivat syötteen perusteella jonkun jatkuvan reaaliluvun, josta pitäisi vielä päätellä syötteen leima (positiivinen tai negatiivinen). Tätä varten on päätettävä jokin sopiva rajapyykki, jonka avulla leimat määritetään. Toisin sanoen, jos x on valittu raja-arvo ja $\hat{x}$ syötteelle ennustettu reaaliluku, saa kyseinen esimerkki leiman 1, jos $\hat{x} \geq x$ ja leiman 0 muutoin. Eräs binääriluokittelumalleille tyypillinen metriikka on *AUC-indeksi* (*Area Under receiver operator characteristic Curve*), jonka avulla voidaan tutkia mallin ennustevoimaa valitsematta binääriluokittelun rajapyykkiä eksplisiittisesti.

Binääriluokittelussa voidaan tehdä kahdentyyppisiä virheitä: *tosi positiivinen* esimerkki (*True Positive, TP*) voidaan luokitella negatiiviseksi, jolloin kyseessä on *väärä negatiivinen* (*False Negative, FN*), ja toisaalta *tosi negatiivinen* (*True Negative, TN*) esimerkki voidaan luokitella positiiviseksi, jolloin tapausta kutsutaan *vääräksi positiiviseksi* (*False Positive, FP*). Usein mallin evaluoinnin yhteydessä ennustukset esitetään *sekaannusmatriisin* (*confusion matrix*) muodossa. [24]



Kuva 14: Kaaviokuva sekaannusmatriisista. Kuvassa TN, FP, FN ja TP merkitsevät vastaavien käsitteiden esiintymisten lukumääriä luokitellussa aineistossa.

Sekaannusmatriisista voidaan määrittää useampi tunnusluku, jotka ovat ilmaisuvoimaisempia kuin pelkkä tarkkuus. *Sensitiivisyys* (*Sensitivity, True Positive Rate, TPR*) kertoo niiden ennustusten osuuden, jotka on luokiteltu oikein positiiviksi niistä esimerkeistä, joiden tiedetään olevan positiivisia. Sensitiivisyys voidaan määrittää kaavalla

$$TPR = \frac{TP}{N_+} = \frac{TP}{TP + FN}, \tag{38}$$

missä N_+ on posiitivisten esimerkkien lukumäärä. *Spesifisyys* (*Specificity, True Negative Rate, TNR*) puolestaan kertoo oikein ennustettujen negatiivisten esimerkkien osuuden niistä esimerkeistä, joiden tiedetään olevan negatiivisia. Se voidaan niin ikään määrittää kaavalla

$$TNR = \frac{TN}{N_-} = \frac{TN}{TN + FP}, \tag{39}$$

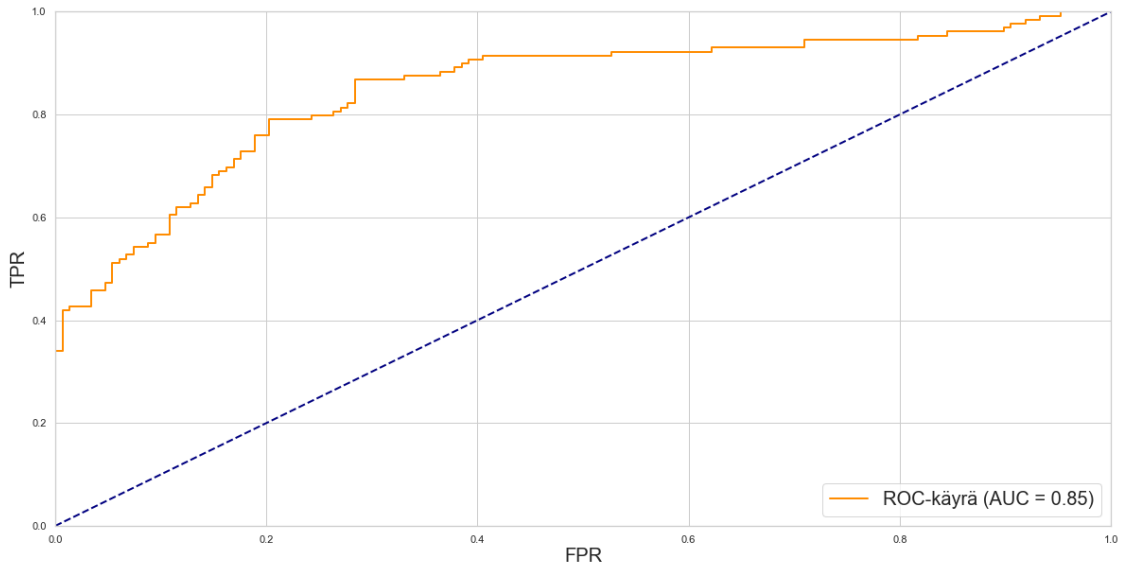
missä N_- on negatiivisten esimerkkien lukumäärä. AUC-indeksin määritykselle oleellinen suure on vielä lisäksi *FPR* (*Fall-out, False Positive Rate*), joka saadaan myös sekaannusmatriisista kaavalla

$$FPR = \frac{FP}{N_-} = \frac{FP}{TN + FP} = 1 - TNR. \tag{40}$$

Kuten kuvasta 14 nähdään, kun sekaannusmatriisi on määritetty, voidaan kummankin mainitun tunnusluvun arvo laskea suoraan matriisia käyttämällä.

Edellä esitellyt tunnusluvut liittyvät kaikki binääriluokitteluun, jonka vuoksi niiden määrittämistä varten on ensin valittava raja-arvo τ, jonka perusteella luokittelu

tehdään. Kun FPR ja TPR määritetään usealla $\tau \in (0, 1)$ ja asetetaan ne koordinaatistoon, saadaan *ROC-käyrä* (*Receiver Operating Characteristic*), joka saa arvoja joukossa $[0, 1] \times [0, 1]$. AUC-indeksi voidaan nimensä mukaisesti määrittää tämän käyrän ja FPR-akselin rajaaman alueen pinta-alana.



Kuva 15: Koulutetun konvoluutioverkkomallin tuottama ROC-käyrä koulutusaineistolla. Sininen katkoviiva esittää täysin satunnaisesti ennustavan mallin ROC-käyrää.

Eräänlaisena vertailupisteenä voidaan pitää täysin satunnaisesti ennustavan mallin ROC-käyrää. Tällöin malli tuottaa vääriä positiivisa yhtä paljon kuin se tuottaa oikeita positiivisa, jolloin sensitiivisyyden ja FPR:n suhde on yhtä suuri kaikilla rajaarvoilla $\tau \in (0, 1)$. Tällöin ROC-käyrästä saadaan kuvan 15 katkoviivan mukainen suora, jonka AUC-indeksi on luonnollisesti 0.5. Täten systemaattisesti oikein ennustavan mallin ROC-käyrän tulisi olla huomattavasti tämän rajapyykin yläpuolella. Tämän vuoksi malli, joka erottelee positiiviset ja negatiiviset esimerkit toisistaan, saa AUC-arvon 1 ja puolestaan malli, jolla ei esiinny minkäänlaista ennustevoimaa, saa AUC-arvoksi aiemmin mainitun 0.5. [9]

AUC on useimmissa tapauksissa parempi binäärisen luokittelijan hyvyyden mittari kuin pelkkä binäärinen tarkkuus ja yleensä tutkittaessa mallien suorituskykyä hyödynnetäänkin monia metriikoita. Koneoppimisen määränpäänä voidaan pitää jonkun tietyn ilmiön tunnistamista saatavilla olevasta datasta. Usein lääketieteellisissä tutkimuksissa mielenkiinnon kohteena oleva ilmiö voi olla jokin harvinainen sairaus, jonka prevalenssi potilaissa voi olla hyvin pieni. Tässä tutkielmassa on tavallaan samanlainanen tilanne, sillä syöpäkudosta sisältävien PET-MRI-leikkeiden

osuus kaikista leikkeistä on varsin pieni. Tällöin malli, joka leimaisi kaikki esimerkit negatiivisiksi riippumatta datasta, saavuttaisi varsin korkeita tarkkuusestimaatteja, sillä suurin osa esimerkeistä on negatiivisia. Tällöin tarkkuus on harhaanjohtava estimaatti, jonka rinnalla tulisi käyttää robustimpeja metriikoita, kuten AUC:ta.

4.4 Optimaalisen raja-arvon valinta

Voisi tuntua intuitiiviselta luokitella esimerkki positiiviseksi, mikäli binääriluokittelulle tarkoitetun konvoluutioverkon viimeisen kerroksen sigmoid-yksikkö palauttaisi arvon, joka on suurempi kuin 0.5. Sigmoid-regression tulosteethan voidaan tulkita todennäköisyyksinä, ja tämän kaltaisessa tilanteessahan olisi todennäköisempää, että esimerkki on positiivinen kuin että se olisi negatiivinen. Todellisuudessa koneoppimisongelmat harvoin ovat näin yksinkertaisia, ja erilaistan päätösten kustannukset ovat usein hyvinkin erilaisia. Esimerkiksi lääketieteessä väärä positiivinen ennuste saataa aiheuttaa tarpeettomia tutkimuksia ja lääkityksiä, mistä voi puolestaan seurata suuriakin turhia kustannuksia ja ajankäytön hukkaa. Väärä negatiivinen ennuste voi puolestaan aiheuttaa tarpeellisen hoidon saamattomuutta, mikä voi pahimmissa tapauksissa johtaa potilaan kuolemaan. Joissain tapauksissa kustannusten arviointi voi olla helppoa, mutta usein näin ei kuitenkaan ole. Esimerkiksi potilaan kuolemalle tuskin voidaan määrittää mitään tarkkaa rahallista arvoa. Tämän vuoksi useat konioppimisen luokitteluongelmat ovat todella epäsymmetrisiä kustannusten suhteen, ja luokittelun raja-arvon määritys ei ole aina yksinkertaista.

Eräs objektiivinen tapa määrittää luokittelun raja-arvo on käyttää *Youdenin J-statistiikkaa*. [31]

Määritelmä 4. Olkoon TP, FP, TN ja FN vastaavien suureiden lukumäärät luokitellussa aineistossa, missä luokittelun raja-arvo on τ. Youdenin J-statistiikka saadaan kaavalla

$$J(\tau) = \frac{TP \cdot TN - FN \cdot FP}{(TP + FN)(FP + TN)}. \tag{41}$$

Käytännössä raja-arvon τ määritys Youdenin J-statistiikan avulla muistuttaa kovasti ROC-käyrän määritystä: J-statistiikka lasketaan useilla $\tau \in (0, 1)$ ja raja-arvoksi valitaan τ, joka maksimoi Youdenin J-statistiikan. On helppo nähdä, että ROC-

käyrän avulla on tosiaan mahdollista määrittää Youdenin J:

$$\begin{aligned} J(\tau) &= \frac{TP \cdot TN - FN \cdot FP}{(TP + FN)(FP + TN)} \\ &= \frac{TP \cdot TN + TP \cdot FP - TP \cdot FP - FN \cdot FP}{(TP + FN)(FP + TN)} \\ &= \frac{TP \cdot (TN + FP)}{(TP + FN)(FP + TN)} - \frac{FP \cdot (TP + FN)}{(TP + FN)(FP + TN)} \\ &= \frac{TP}{TP + FN} - \frac{FP}{TN + FP} = TPR - FPR. \end{aligned} \tag{42}$$

Siis Youdenin J-statistiikan arvo jollain τ saadaan suoraan ROC-käyrästä akselien arvojen erotuksena. Edelleen optimaalisen luokittelun tuottava raja-arvo τ saadaan kaavasta

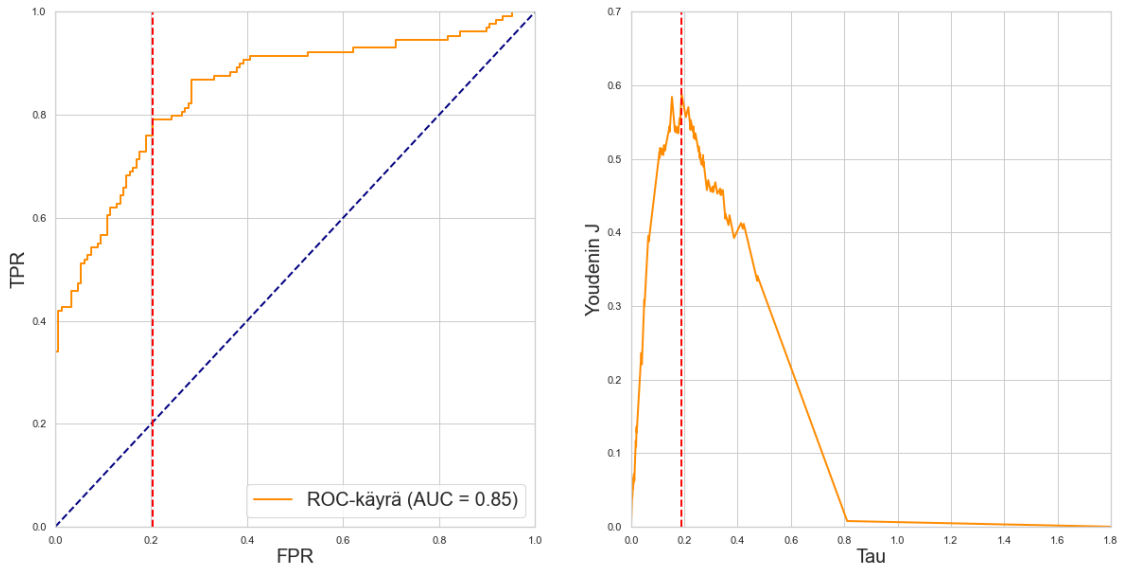
$$\hat{\tau} = \arg\max_{\tau \in (0,1)} J(\tau). \tag{43}$$

Youdenin J-statistiikkalle on olemassa myös graafinen tulkinta ROC-käyrän avulla. J-statistiikka ilmaisee ROC-käyrän etäisyyden satunnaisesti ennustavan mallin ROC-käyrästä (eli funktiosta $f(x) = x$). Koska pyrkimyksenä on luoda malli, jonka AUC-indeksi olisi mahdollisimman lähellä lukua 1, on intuitiivista valita luokittelun raja-arvoksi sellainen τ, jolla ROC-käyrä on mahdollisimman kaukana satunnaisesti ennustavan mallin ROC-käyrästä. Tämä huomataan myös kuvasta 16, missä punaisella katkoviivalla merkitty optimaalinen rajapyykki asettuu Youdenin J-statistiikkaa taun funktiona kuvaavaan käyrän maksimipisteeseen.

4.5 Tulokset

4.5.1 ResNet50-malli

Kuvasta 17 nähdään tuloksia ResNet50-mallin koulutuksesta. Näyttäisi siltä, että malli ei ole löytänyt mitään merkittävää PET-MRI-kuvista, sillä validointi-AUC pysyy lähes vakiona koko koulutusprosessin ajan. AUC-indeksi on pienempi kuin rajapyykkiarvo 0.5, mikä tarkoittaa, että malli käytännössä ennustaa huonommin kuin satunnaisesti esimerkkejä leimaava malli ennustaisi. Binääriluokittelussa voidaan kuitenkin tarvittaessa kääntää leimat vastakkaisiksi, jolloin AUC-indeksiksi saadaan $1 - AUC$, mikä on vastaavasti suurempi kuin 0.5. Tässä tapauksessa tuo muunnos ei kuitenkaan tuo lisäarvoa malliin, sillä alkuperäisetkin AUC-arvot ovat niin lähellä arvoa 0.5. Validoinnissa binääritarkkuus vaihtelee hieman enemmän, mutta pysyy kuitenkin noin 40 prosentin tasolla. Ristientropia puolestaan laskee loivasti, mutta lasku on suuruusluokaltaan varsin pieni ja näyttäisi tasaantuneen 15 epookin kohdalla. Kokonaisuudessaan näyttäisi siltä, että malli ei ole löytänyt mi-



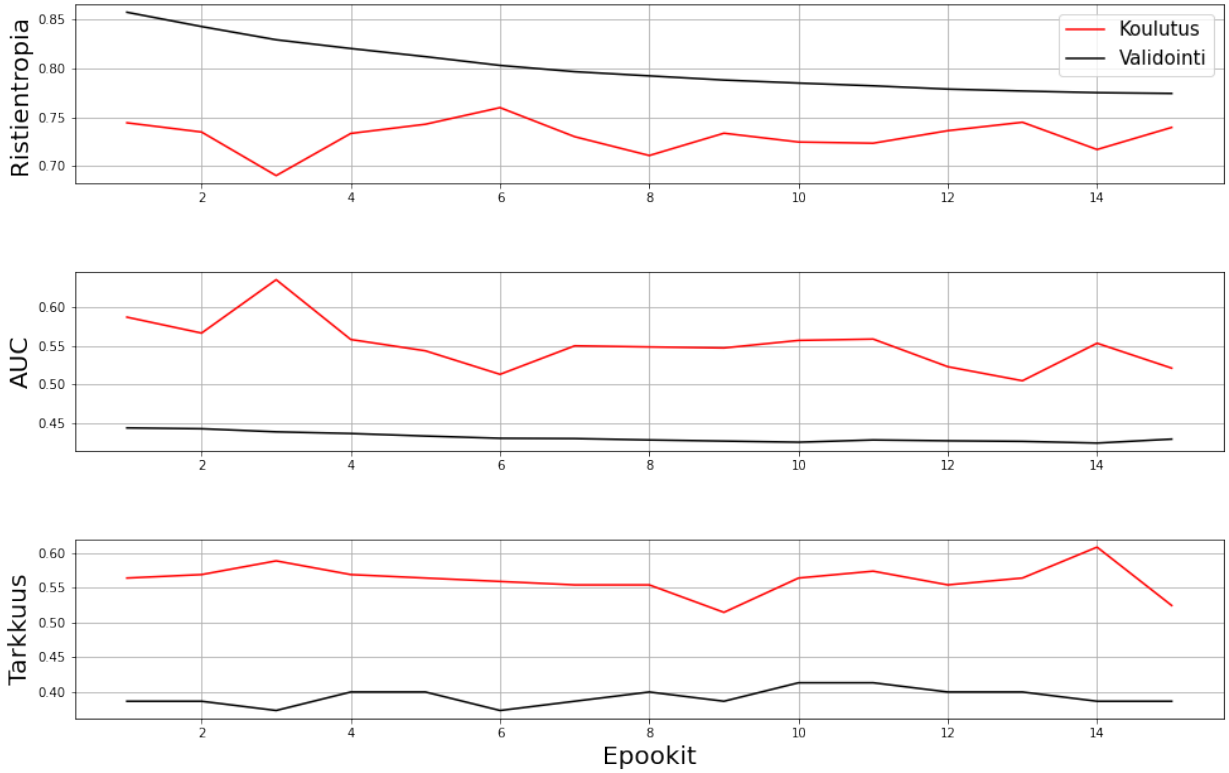
Kuva 16: Vasemmalla kuvan 15 ROC-käyrä, johon on lisätty punaisella katkoviivalla piste, jossa Youdenin J-statistiikka saa suurimman arvonsa. Oikealla näkyy puolestaan Youdenin J raja-arvon τ funktiona ja punaisen katkoviivan kohdalta löytyy optimaalinen luokittelun raja-arvo.

tään kiinnostavaa signaalia datasta.

Taulukkoon 3 on koottu tunnuslukuja mallin suoriutumisesta testijoukon kuvilla. Malli on luokitellut testikuvista noin 35 prosenttia oikein, mikä on varsin alhainen lukema. AUC-indeksin arvo tukee väitettä siitä, että malli ei ole oppinut mitään merkittävää. Alhainen sensitiivisyys puolestaan viestii, että mallilla on vaikeuksia tunnistaa juuri mielenkiinnon kohteena olevan syöpäkudoksen läsnäolo leikkeissä. Toisaalta myös spesifisyys on varsin alhainen, mikä kertoo, että malli ei tunnista negatiivisiakaan esimerkkejä tehokkaasti.

Tunnusluku	Arvo
Tarkkuus	0.35
AUC	0.43
Sensitiivisyys	0.33
Spesifisyys	0.38
τ	0.37

Taulukko 3: ResNet50-mallin suorituksen evaluointi testijoukon kuvien leimaamisessa.

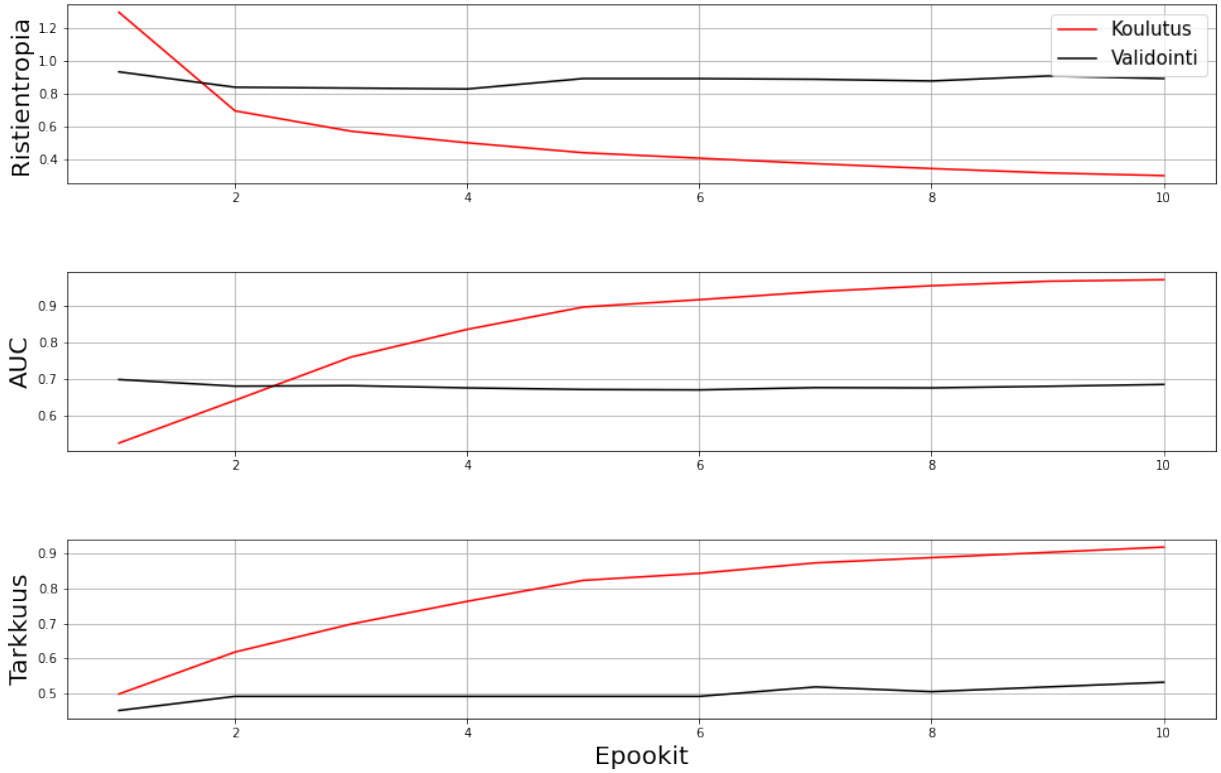


Kuva 17: Tunnuslukuja ResNet50-mallin koulutusprosessista. Ylimpänä ristientropia, keskellä AUC ja alimpana binääritarkkuus.

4.5.2 Perinteinen konvoluutioverkko

Kuvasta 18 huomataan, että validointi-AUC on huomattavasti korkeampi kuin ResNet50-mallin tapauksessa. Tarkkuus ja AUC koulutusdatalla kasvavat aidosti epookkien myötä ja näyttäisivät saturoituvan juuri 10 epookin kohdalla, jolloin koulutuskin päättyy. Ristientropia vastaavasti laskee.

Taulukkoon 4 on jälleen koottu tunnuslukuja konvoluutioverkon suoriutumisesta testijoukolla. Huomataan, että myös testijoukolla alusta asti koulutetun mallin AUC on huomattavasti parempi kuin ResNet-mallilla. AUC on toisaalta myös merkityksellisesti suurempi kuin 0.5, mikä indikoi, että malli on löytänyt signaalin datasta. Spesifisyys on erityisen korkea, mikä kertoo, että malli tunnistaa hyvin sellaiset esimerkit, joissa ei esiinny syöpäkudosta. Toisaalta myös sensitiivisyys on kohtuullisella tasolla. Tämä ilmiö voidaan havaita vielä kuvan 19 sekaannusmatriisista, josta nähdään, että vääriä positiivisia on tosiaan huomattavasti vähemmän kuin vääriä negatiivisia. Mallilla on siis taipumus ennustaa esimerkille negatiivinen leima useammin,



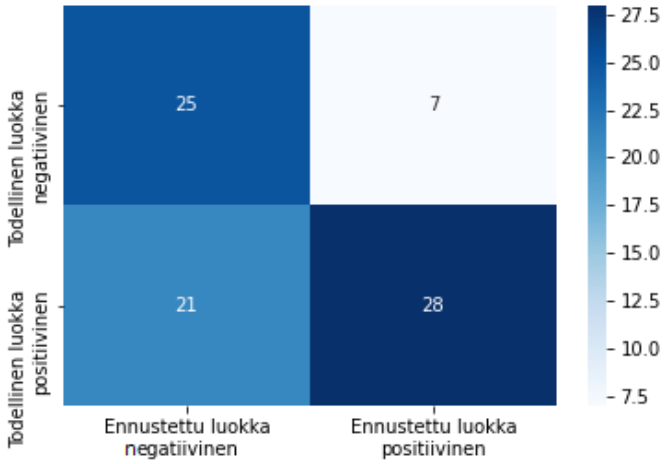
Kuva 18: Tunnuslukuja konvoluutioverkon koulutusprosessista. Ylimpänä ristientropia, keskellä AUC ja alimpana binääritarkkuus.

vaikkakin se myös tunnistaa syöpäkudoksen leikkeessä kohtalaisen hyvin.

Tunnusluku	Arvo
Tarkkuus	0.65
AUC	0.72
Sensitiivisyys	0.57
Spesifisyys	0.78
τ	0.26

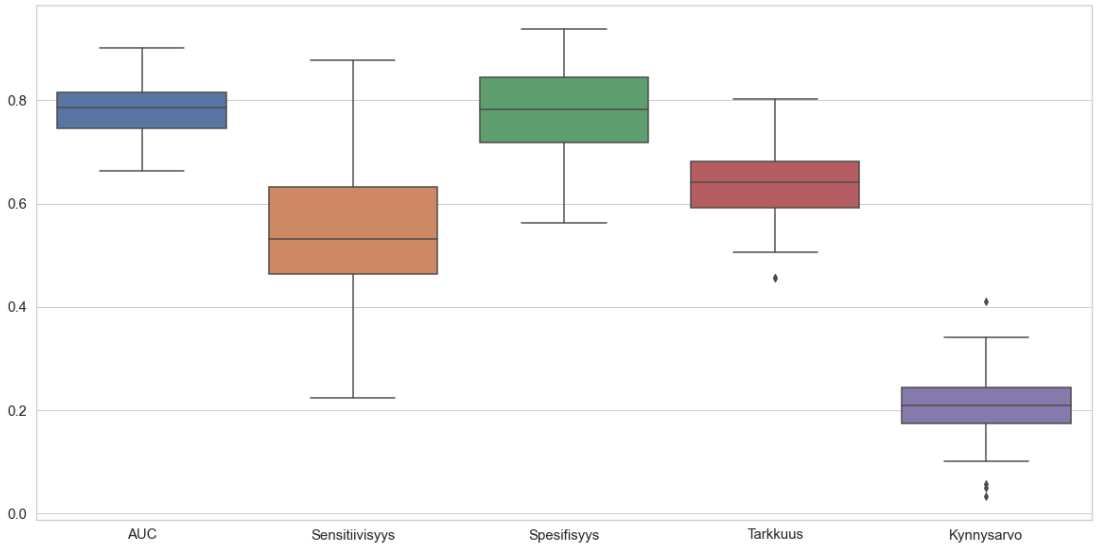
Taulukko 4: Perinteisen konvoluutioneuroverkkomallin suorituksen evaluointi testijoukon kuvien leimaamisessa.

Koska näyttää siltä, että perinteinen konvoluutioverkko toimii tässä tapauksessa paremmin, tutkitaan vielä lisää sen toimintaa tilastollisemmasta näkökulmasta. Vaikka koulutus-, validointi-, ja testiaineisto pysyvät vakioina koulutuksesta toiseen, pitää koulutuksessa käytössä oleva optimointimenetelmä silti sisällään satunnaisuutta. Tämän vuoksi toistetaan aiemmin esitelty koulutusprosessi identtisenä 100 ker-



Kuva 19: Sekaannusmatriisi testijoukon ennustuksista.

taa, jotta voidaan tutkia, millaisia tuloksia malli tuottaa keskimäärin.



Kuva 20: Laatikko-janakuviot testiaineistolla määritetyistä tunnusluvuista sadalta erilliseltä koulutusajolta.

Kuvan 20 laatikko-janakuvioista huomataan, että AUC-indeksi saa varsin pienellä hajonnalla suuria arvoja mediaanin ollessa likimain 0.8. Aiemmin esitelty malli ei siis vaikuttanut olevan vain sattumaa, vaan tämä neuroverkkomalli näyttäisi systemaattisesti onnistuvan löytämään datasta mielenkiintoisen signaalin. Sensitiivi-

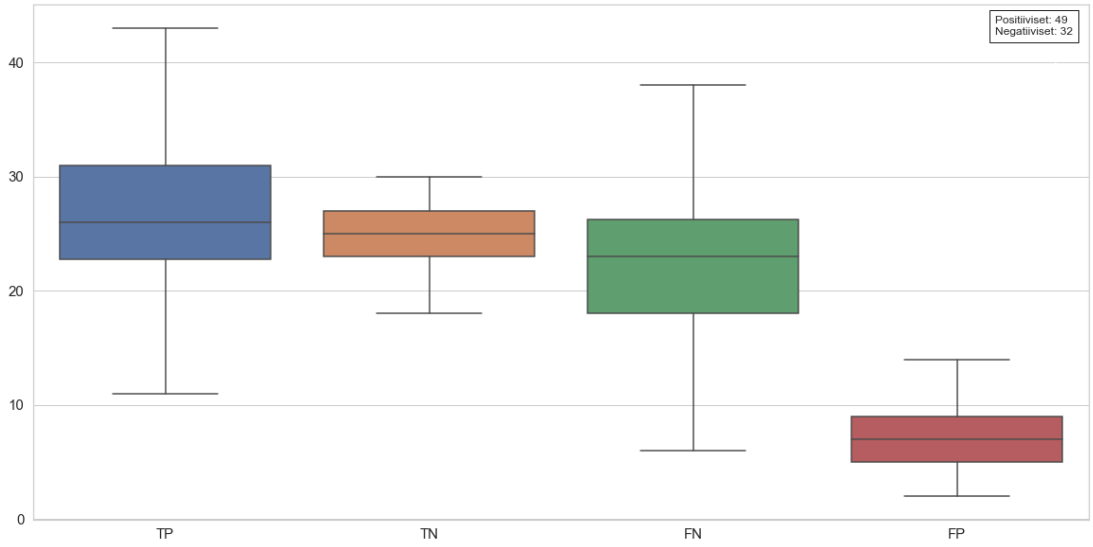
syydessä huomataan suurta vaihtelua, kun taas spesifisyys saa paljon kapeammalta väliltä arvoja. Tämä selittynee edelleen sillä, että malli ennustaa negatiivisia leimoja herkemmin. Luokittelun kynnysarvo ei sinänsä mittaa malliin hyvyyttä, mutta on syytä panna merkille, että kynnysarvot ovat jakautuneet varsin tiiviisti arvon 0.2 ympärille. Tämä tarkoittaa, että malli leimaa esimerkin positiiviseksi varsin pienellä kynnyksellä. Kuvan 21 laatikko-janakuvioista nähdään vielä, että väärien positiivisten lukumäärät ovat tosiaan olleet matalia, mikä selittää korkean spesifisyyden. Korkeampi väärien negatiivisten lukumäärä puolestaan selittää matalamman sensitiivisyyden.

Eri koulutusiteraatioiden malleissa on siis selvästi jonkin verran eroja, mutta suorituskyvyn mittarina AUC pysyy kohtuullisen hyvin samalla tasolla. Näin ollen se malli, joka vastaa AUC-indeksien mediaania edustaa hyvin keskimääräistä mallia, jonka tässä tutkielmassa käytetty menetelmä tuottaa.

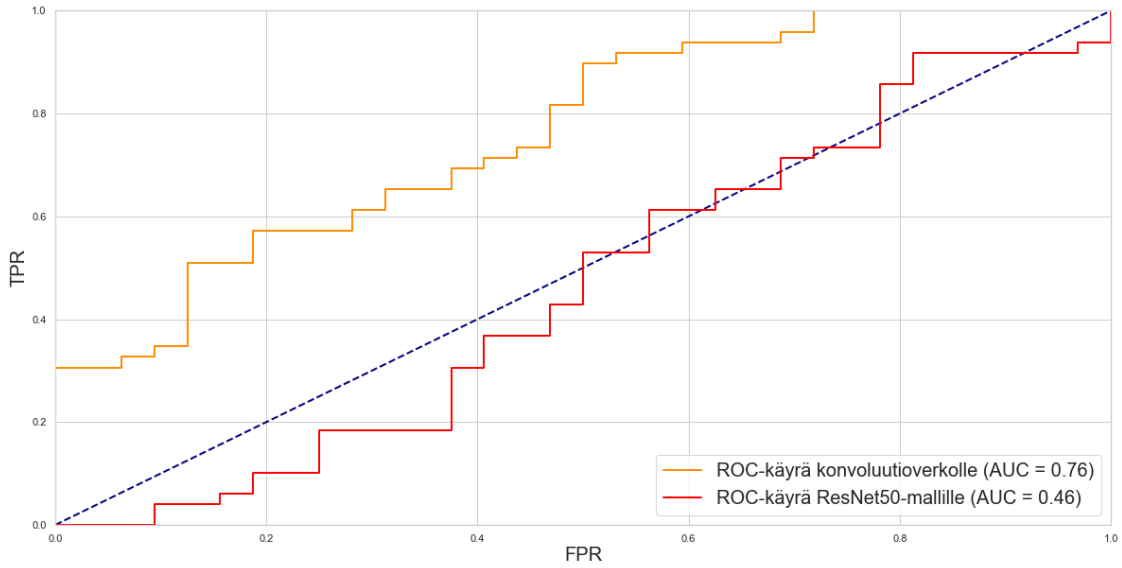
Mediaanimallin ROC-käyrä on odotetusti selvästi vertailuarvomallin sekä ResNet50-mallin ROC-käyrien yläpuolella kuten kuvasta 22 nähdään. Taulukossa 5 on listattuna metriikoita mediaanimallin suoriutumisesta testijoukon sekä vertailuksi myös koulutusjoukon kuvilla. Suorituskykymittarit ovat koulutusjoukolla luonnollisesti parempia, sillä malli on optimoitu käyttämällä nimenomaan koulutusjoukon leikkeitä. AUC-indeksi putoaa testijoukon kohdalla arvosta 0.85 arvoon 0.76, mikä vaikuttaa järkevältä. Selkeästi malli on testijoukostakin löytänyt samankaltaisen signaalin, minkä se on löytänyt jo koulutusvaiheessa. Spesifisyys tippuu testijoukkoon siirryttäessä vain vähän, mikä viittaa siihen, että malli tunnistaa negatiiviset esimerkit varsin hyvin, vaikkei se olisikaan nähnyt kyseistä esimerkkiä aikaisemmin koulutuksen tai validoinnin yhteydessä. Sensitiivisyys puolestaan tippuu, mikä näkyy myös selkeästi kuvassa 21 väärien negatiivisten suurempana määränä verrattuna väärien positiivisten määrään.

Tehtyjen testien ja analyysien perusteella vaikuttaisi siis siltä, että valmiiksi koulutetusta ResNet50-mallista ei ole apua PET-MRI-leikkeiden luokittelussa syöpäkudosta sisältäviksi tai ei. Tämä saattaa johtua esimerkiksi siitä, että ResNet-malleja ei ole alunperin kehitetty lääketieteellisten aineistojen analysoimiseen, vaan motivaatio ResNet-mallien kehityksen takana on syvempien neuroverkkojen koulutuksen tehostaminen. [16] Lisäksi Pythonin Tensorflow-kirjaston tarjoamat valmiiksi koulutetut ResNet-mallit on koulutettu tuhansilla kaksiulotteisilla RGB-kuvilla, kun taas PET-MRI-fuusiot sisältävät vain kaksi "värikanavaa". Puolestaan alusta asti koulutettu yksinkertainen konvoluutioverkko näyttäisi suoriutuvan tehtävästä kohtalaisen hyvällä menestyksellä. Tämä selittynee sillä, että neuroverkon parametrit räätälöidään siten, että malli suoriutuu hyvin nimenomaan tässä tutkielmassa käytettävän

aineiston kaltaisien PET-MRI-kuvien luokittelusta.



Kuva 21: Laatikko-janakuviot ennustusten ja ennustevirheiden lukumääristä testiaineistolla.



Kuva 22: Mediaanimallin ROC-käyrä testiaineistolla. Sinisellä katkoviivalla merkitty satunnaisia ennustuksia tuottavan mallin ROC-käyrä. Vertailuksi kuvaan on merkitty myös ResNet50-mallin ROC-käyrä testiaineistolla.

Aineisto	Tarkkuus	AUC	Sensitiivisyys	Spesifisyys
Testi	0.64	0.76	0.57	0.75
Koulutus	0.79	0.85	0.79	0.80

Taulukko 5: Mediaanimallin suorituksen evaluointi testijoukon ja koulutusjoukon kuvien leimaamisessa.

5 Jatkotutkimus

PET-MRI-leikkeiden luokittelu on vain yksi sovelluskohde, jossa konvoluutioneuroverkkoja voidaan käyttää. Seuraava askel olisi luonnollisesti siirtyä yksittäiden leikkeiden luokittelusta kuvien segmentoimiseen, jossa tarkoituksena on generoida aineiston avulla uudelle leikeparille binäärimaski, joka vastaa mahdollisimman hyvin asiantuntijan koostamaa maskia kyseisestä leikkeestä. PET-MRI-kuvat ovat usein, kuten tämänkin tutkielman aineiston kohdalla, kolmiulotteisia, joten lopullinen tavoite olisi luoda malli, joka pystyisi ottamaan syötteenä leimaamattoman kolmiulotteisen PET-MRI-pinon ja luomaan tälle binäärimaskin. Tätä varten on tutkittava, miten vierekkäisissä leikkeissä esiintyvät syöpäkudokset korreloivat keskenään. On paljon todennäköisempää, että leikkeessä, jonka ylä- ja alapuolelta löydetään syöpää, löytyyy myös syöpäkudosta kuin että sitä löytyisi sellaisestä leikkeestä, jonka kummaltakaan puolelta ei löydy syöpää.

Luokittelun saralla mallia voisi kehittää vielä useammalla tavalla. Youdenin J-statistiikan avulla määritetty raja-arvo on objektiivinen, mutta ei ota kantaa väärien luokitteluiden epäsymmetrisiin kustannuksiin. On vaikea arvioida tarkkaa kustannusta väärälle ennustukselle, mutta kustannusarvioiden lisääminen raja-arvon valintaan varmasti vaikuttaisi lopputulokseen.

Tässä tutkielmassa koulutettu malli suoriutuu sille tarkoitetusta tehtävästä varsin hyvin, mutta ennen kuin sitä voisi varsinaisesti käyttää tuotannossa, tulisi sen tarkkuutta ja erityisesti matalaa sensitiivisyyttä parantaa. Yksi lähestymistapa tämän saavuttamiselle on kenties kokeilla vaihtoehtoisia arkkitehtuureja. Lisäksi saatavilla oleva aineisto on tässä tutkielmassa varsin pieni. Kasvattamalla aineiston kokoa saadaan yhtäältä enemmän koulutusdataa, jolloin mallille voidaan syöttää suurempi kirjo eri tautitapauksia. Toisaalta näin saadaan kasvatettua myös testiaineiston kokoa, jolloin mallin suorituksen evaluointi on tehokkaampaa ja luotettavampaa.

Tässä tutkielmassa esiteltävän luokittelevan koneoppimismallin kehittämisen tavoitteena on loppujen lopuksi käyttää sitä uusien potilaiden PET-MRI-leikkeiden analysoimisen helpottamiseen. Mallin ei ole tässä vaiheessa vielä tarkoitus toimia autonomisesti, vaan sitä voisi soveltaa uuden potilaan kohdalla binäärimaskien koos-

tamisen apuna. Uuden potilaan PET-MRI-kuvat syötetään mallille, jolloin se generoisi alustavat ennustukset siitä, sisältävätkö yksittäiset leikkeet syöpäkudosta vai ei. Jos malli on tietyllä luottamustasolla esimerkiksi ennustanut, että yksittäinen leike sisältää syöpäkudosta, voisi binäärimaskia koostava asiantuntija kiinnittää erityistä huomioita tutkiessaan leikettä. Puolestaan negatiivisen leiman saaneiden leikkeiden tutkimiseen voisi käyttää vähemmän aikaa, jolloin leimaaminen olisin kenties nopeampaa. Tuotantoonvientiä ajatellen olisi kuitenkin päätettävä, miten mahdollinen uusi materiaali otettaisiin huomioon mallissa. Yksi vaihtoehto olisi normalisoida uusi pino leikkeitä sisällyttäen ne alkuperäiseen aineistoon. Näin lauseen 3 avulla uudet leikkeet tulee normalisoitua samalla tavalla kuin miten aineisto on skaalattu koulutus- ja testausvaiheissa. Mallia voisi myös kouluttaa uudelleen sitä mukaa, kun aineistoa saadaan lisää. Uusien potilaiden PET-MRI-leikkeistä kerätty informaatio ikään kuin sulautettaisiin malliin ja näin mallin suorituskykyä voitaisiin marginaalisesti parantaa ajan myötä.

Kirjallisuutta

[1] S. Abu Bakr, S. Shadman, K. R. M Mohammad, Y. Nowrin, A. Anas, c. Madiha, T. K. Ihtyaz: *Detection of COVID-19 Disease from Chest X-Ray Images: A Deep Transfer Learning Framework*. https://www.medrxiv.org/, luettu 15.08.2021.

[2] A. Akasapu, V. Sailaja, G. Prasad: *Implementation of Sobel filter using CUDA*. https://iopscience.iop.org/article/10.1088/1757-899X/1045/1/012016/pdf, luettu 04.04.2021.

[3] E. Alpaydin: *Introduction to Machine Learning*. MIT Press, 2014.

[4] R. Bhatia: *Convolutions*. https://link-springer-com.ezproxy.utu.fi/article/10.1007/s12045-017-0525-7, luettu 03.04.2021.

[5] S. Bhattacharyya, E. Hassanien, V. Snasel, B. K. Tripathy: *Deep Learning: Research and Applications*. De Gruyter, 2020.

[6] Y. Bengio, A. Courville, I. Goodfellow: *Deepl Learning*. MIT Press, 2016.

[7] A. Bilogur: *Full batch, mini-batch, and online learning*. https://www.kaggle.com/residentmario/full-batch-mini-batch-and-online-learning, luettu 10.04.2021.

[8] J. Brownlee: *A Gentle Introduction to Padding and Stride for Convolutional Neural Networks*. https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/, luettu 05.04.2021.

[9] J. Candlish, Z. Hoo, D. Teare: *What is an ROC-curve*. BMJ Publishing Group, 2017.

[10] *Data Leakage*. https://www.kaggle.com/alexisbcook/data-leakage, luettu 16.06.2021.

[11] M. P. Deisenroth, A. A. Faisal, C. S. Ong: *Mathematics for Machine Learning*. Cambridge University Press, 2019.

[12] Dive Into Deep Learning: *Pooling*. http://d2l.ai/chapter_convolutional-neural-networks/pooling.html, luettu 06.04.2021.

[13] W. Zhao, H. Fu, W. Luk, T. Yu, S. Wang, B. Feng, Y. Ma, G. Yang: *F-CNN: An FPGA-based Framework for Training Convolutional Neural Networks*.

https://spiral.imperial.ac.uk/bitstream/10044/1/50990/2/asap16wz.pdf, luettu 10.04.2021.

[14] H. Greenspan, D. Shen, K. S. Zhou: *Deep Learning for Medical Image Analysis*. Academic Press, 2017.

[15] P. Harjulehto, R. Klén, M. Koskenoja: *Analyysiä reaaliluvuilla*. Unigrafia, 2014.

[16] K. He, X. Zhang, S. Ren, J. Sun: *Deep Residual Learning for Image Recognition*. https://arxiv.org/pdf/1512.03385.pdf, luettu 08.06.2021.

[17] T. Hill: *Gradient Descent and Backpropagation*. https://towardsdatascience.com/part-2-gradient-descent-and-backpropagation-bf90932c066a, luettu 21.03.2021.

[18] IBM Cloud Education: *Supervised Learning*. https://www.ibm.com/cloud/learn/supervised-learning, luettu 25.12.2020.

[19] *ImageNet*. http://www.image-net.org/download, luettu 11.04.2021.

[20] V. Kapoor, B. M. McCook, F. S. Torok: *An Introduction to PET-CT Imaging*. https://doi.org/10.1148/rg.242025724, luettu 01.01.2021, kello 20.33.

[21] V. Köchli, B. Marinek, D. Weishaupt: *How does MRI work? : an introduction to the physics and function of magnetic resonance imaging*. Springer, 2006.

[22] H. LeVine: *Medical Imaging*. Greenwood, 2010.

[23] S. Marsland: *Machine Learning: An Algorithmic Perspective*. CRC Press, 2015.

[24] K. Murphy: *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[25] National Cancer Institute: *Head and Neck Cancers*, https://www.cancer.gov/types/head-and-neck/head-neck-fact-sheetwhat-are-cancers-of-the-head-and-neck, luettu 11.04.2021.

[26] M. M. Mäkelä: *Optimointialgoritmit*. Turun yliopisto, matematiikan ja tilastotieteen laitos, 2020.

[27] W. Oldendorf, W. Jr. Oldendorf: *Basics of Magnetic Resonance Imaging*. Martinus Nijhoff Publishing, 1988.

[28] K. Simonyan, A. Zisserman: *Very Deep Convolutional Networks for Large-scale Image Recognition*. https://arxiv.org/pdf/1409.1556.pdf, luettu 08.06.2021.

[29] S. Valadi: *New York City*. https://commons.wikimedia.org/wiki/File:EmpireStateNewYokCity.jpg, luettu 04.04.2021.

[30] A. van Waarde: *Introduction to PET: Description of Basics and Principles*. https://www.researchgate.net/publication/300137514_Introduction_on_PET_Description_of_Basics_and_Principles, luettu 02.01.2021

[31] W.J. Youden: *Index for Rating Diagnostic Tests*. Wiley Subscription Services, 1950.

[32] T. Zhang: *Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms*. Watson Research Center, 2004.