Katsaus transformer-arkkitehtuureihin ja niiden hyödyntämiseen konenäkötehtävissä

Turun yliopisto
Tietotekniikan laitos
Pro gradu -tutkielma
Tietojenkäsittelytiede
Marraskuu 2024
Juuso Haapanen

TURUN YLIOPISTO
Tietotekniikan laitos

Juuso Haapanen: Katsaus transformer-arkkitehtuureihin ja niiden hyödyntämiseen konenäkötehtävissä

Pro gradu -tutkielma, 57 s.
Tietojenkäsittelytiede
Marraskuu 2024

Konvoluutioneuroverkkoihin perustuvat kuvantunnistusalgoritmit ovat pitkään hallinneet konenäön menetelmiä. Viime vuosina alun perin luonnollisen kielen käsittelyä varten kehitetyt transformer-arkkitehtuurit ovat kuitenkin saavuttaneet merkittävää suosiota myös konenäön tehtävissä, kuten kuvien luokittelussa, hahmontunnistuksessa ja semanttisessa segmentoinnissa. Tässä työssä tarkastellaan transformer-arkkitehtuureja, erityisesti konenäön tarpeisiin kehitettyjä transformer-malleja, sekä analysoidaan niiden menetelmällisiä ja suorituskyvyn eroja.
Transformer-arkkitehtuureita vertaillaan työssä konvoluutioneuroverkkoihin sekä kirjallisuudessa raportoitujen tuloksien perusteella sekä kokeellisin menetelmin.
Kokeelliset tulokset osoittavat, että transformer-pohjaiset mallit ovat kilpailukykyisiä verrattuna konvoluutioneuroverkkoihin. Tässä työssä hienosäädetty ViT-malli saavutti 91,2% ja Swin Transformer 85,78% tarkkuuden, kun taas EfficientNet-malli, joka edustaa konvoluutioneuroverkkoja, saavutti 82,01% tarkkuuden kuvien luokittelutehtävässä. Semanttisen segmentoinnin osalta tuloksia arvioitiin kirjallisuudessa raportoitujen arvojen perusteella, ja niiden pohjalta voidaan todeta transformer-mallien olevan hyvin vertailukelpoisia suhteessa konvoluutioneuroverkkoihin.
Tulosten perusteella transformer-arkkitehtuureja voidaan pitää konenäön ongelmien ratkaisemisessa tarkkuuden suhteen kilpailukykyisinä konvoluutioneuroverkkoihin verrattuna. Lisäksi transformer-arkkitehtuurit osoittautuvat monikäyttöisemmäksi, mahdollistaen erilaisten konenäkötehtävien ratkaisemisen yhdenmukaisella lähestymistavalla.

Asiasanat: transformer, kuvamuunnin, konenäkö, kuvien luokittelu, semanttinen segmentointi

UNIVERSITY OF TURKU
Department of Computing

JUUSO HAAPANEN: Katsaus transformer-arkkitehtuureihin ja niiden hyödyntämiseen konenäkötehtävissä

Master of Science Thesis, 57 p.
Computer Science
November 2024

Convolutional networks have dominated machine vision methods, but transformer architectures developed initially for natural language processing have also gained popularity in machine vision tasks such as image classification, pattern recognition, and semantic segmentation. In this study, we will learn about transformer architectures and models developed especially for machine vision and their methodical and performance differences.

The study compares transformer architectures to convolutional network-based architectures, comparing the results presented in the literature with experimental work. Based on this study, models based on transformer architecture are comparable to convolutional neural networks. In image classification, the fine-tuned ViT model achieved 91.2% accuracy, Swin Transformer 85.78% accuracy, while the EfficientNet model based on convolutional networks achieved 82.01% accuracy. The semantic segmentation results were evaluated based on the values presented in the literature, and it appeared that the transformer models were very comparable to the best models based on convolutional networks.

According to this study, transformer architectures are comparable in accuracy to architectures based on convolutional networks in machine vision problems. Transformer architectures are more general-purpose than convolutional networks and can solve several machine vision tasks.

Keywords: transformers, computer vision, image classification, semantic segmentation

Sisällys

Kuvat

Taulukot

Symbolit

σ_i, ϕ Neuroverkon aktivointifunktio

$\mathcal{D}$ Koneoppimisongelman koulutukseen käytettävä datajoukko

E Koneoppimisalgoritmin virhefunktio

H Hypoteesijoukko

$\mathbf{b}$ Harha

$\mathbb{1}$ Indikaattorifunktio

$\mathcal{L}$ Hävikkifunktio

MLP Monikerroshavaitsin, (*engl. Multi Layer Perceptron*)

η Mallin opetusaste

s Neuroverkon kerroksen syöte

T Koneoppimisalgoritmin tehtävä

$\mathbf{W}$ Mallin painomatriisi

$\mathbf{X}$ Mallin syötematriisi

1 Johdanto

1.1 Motivaatio

Konenäkö ja kuvien tunnistaminen tietokoneen avulla on erittäin pitkään tutkittu ongelma. Modernit koneoppimisen työkalut ovat mahdollistaneet konenäköön perustuvien ratkaisuiden tuomisen lähes kaikkien ulottuville. Käyttäessämme mobiililaitteiden kameroita kamera tunnistaa kuvasta reaaliajassa ihmiset, ja puhelimen kuvakirjasto pystyy luokittelemaan kuvat niissä esiintyvien ihmisten sekä heidän toimintansa perusteella oikeisiin albumeihin.

Kuvien tunnistaminen syvien neuroverkkojen avulla on koneoppimistehtävä, jossa parhaiten suorituvat menetelmät perustuvat tällä hetkellä konvoluutioneuroverkkoihin. Konvoluutioneuroverkkojen idean pohjalta on rakennettu kehittyneitä neuroverkkoarkkitehtuureita, joita voidaan käyttää konenäkötehtävien suorittamiseen joko sellaisenaan tai pohjana erikoistuneemmille arkkitehtuureille.

Transformer-arkkitehtuurit ovat alunperin luonnollisen kielen käsittelyyn suunniteltuja neuroverkkoja, mutta viimeaikainen tutkimus on osoittanut että niitä voidaan hyödyntää myös konenäön tehtäviin. Transformer-arkkitehtuurit ovat tällä hetkelle erittäin kiinnostavia siksi, että monet tekoälyn sovellukset kuten GPT-mallit ovat syntyneet transformer-arkkitehtuureiden pohjalta sekä siksi, että perinteisesti erilaisiin tehtäviin kehitetyt mallit ovat perustuneet erilaisiin arkkitehtuureihin: konenäön ongelmien kuten kuvantunnistuksen tehtäviin on käytetty konvoluutio-

neuroverkkoja, kun taas tekstin käsittelemiseen on käytetty esimerkiksi LSTM tai RNN -arkkitehtuureita, mutta transformereihin perustuvia malleja voidaan hyvin samankaltaisilla, tässä työssä myöhemmin esitettävillä, ideoilla soveltaa sekä käännöstehtäviin että kuvien semanttiseen segmentointiin.

1.1.1 Tutkimuskysymykset

Tässä pro gradu-työssä perehdytään konvoluutioneuroverkkojen ja transformaatioarkkitehtuureiden välisiin eroihin ja verrataan niiden kyvykkyyksiä toisiinsa, erityisesti semanttisessa segmentoinnissa. Semanttinen segmentointi on vaativa konenäön tehtävä, jossa tarkoitetuksena on luokitella kuvan elementit pikseleittäin oikeaan luokkaan. Tämän työn tutkimuskysymyksiä ovat

- TK1: Miten transformer-arkkitehtuuri eroaa konvoluutioneuroverkon arkkitehtuurista?
- TK2: Mikä on transformer-arkkitehtuurin pohjalta tehdyn neuroverkon suorituskyky verrattuna *state-of-art* konvoluutioverkkoon kuvien luokittelussa ja semanttisessa segmentoinnissa?

1.2 Tutkimusmetodologia

Konvoluutioneuroverkkojen ja transformer-neuroverkkojen vertailu on pitkälti empiiristä tutkimusta sekä teoreettista tarkastelua. Empiirinen tukimusmenetelmä perustuu tutkimuskohteen havainnointiin ja mittaamiseen, joka tässä työssä tarkoittaa tutkittavien algoritmien hyvyyden arviointia tietyn aineiston ja ennustamistehtävän ratkaisussa. Tässä työssä verkkoja testataan empiirisesti vertaamalla mallien kyvykkyyttä kuvien luokittelulla sekä arvioimalla teoreettisesti eri menetelmien eroja kirjallisuuteen pohjaten.

1.3 Työn rakenne

Luvussa 2 perehdytään koneoppimisen perusterminologiaan sekä siihen miten koneoppimismalleja sekä niiden hyvyyttä voidaan analysoida.

Luvussa 3 esitellään syväoppimisen sekä konvoluutioverkkojen perusteet sellaisessa laajuudessa kuin niiden ymmärtäminen on tarpeen transformer-arkkitehtuurien ymmärtämisen kannalta.

Luvussa 4 käsitellään transformer-arkkitehtuurin perusteita sekä kuvamuuntimia. Tässä luvussa perehdytään perusmuuntimiin sekä niiden ominaisuuksiin ja luokitteluihin.

Luvussa 5 esitellään kuvien luokitteluun kehitettyjä transformer-arkkitehtuureita ja luvussa 6 esitetään kuvien semanttiseen segmentointiin liittyviä transformer-arkkitehtuureita.

Luvussa 7 esitellään ja vertaillaan täysin konvoluutionaalisiin verkkoihin ja transformereihin liittyviä kuvien luokittelumalleja.

2 Koneoppiminen yleisesti

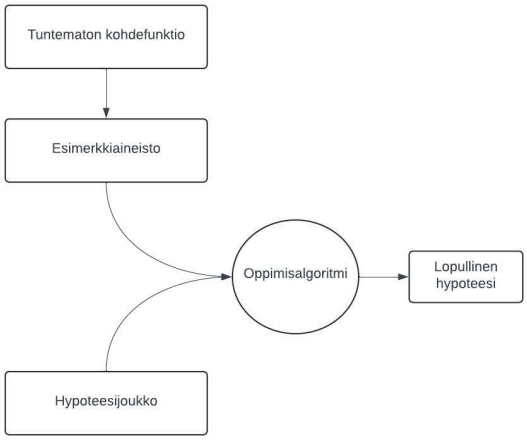
Tässä luvussa perehdytään koneoppimisen perusteisiin, kuten erilaisiin koneoppimisongelmien tyyppeihin sekä siihen miten koneoppimismenetelmiä sekä niiden hyvyyttä voidaan arvioida. Lopuksi perehdytään neuroverkkoihin ja syväoppimiseen.

2.1 Johdanto koneoppimiseen

Koneoppimisella tarkoitetaan sitä, että tietokoneohjelma pystyy ratkaisemaan jonkin tehtävän T, käyttäen saatavilla olevaa dataa $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i)$ niin, että virhefunktio E saadaan minimoitua ja löydetään jokin tuntematon kohdefunktio $f : \mathcal{X} \to \mathcal{Y}$, jossa $\mathcal{X}$ on joukko kaikkia mahdollisia syötteitä ja $\mathcal{Y}$ joukko kaikkia mahdollisia arvoja löydetylle funktiolle. Tyypillisesti joukkoa $\mathcal{D}$ kutsutaan datajoukoksi. Koneoppimistehtävän tarkoituksena on löytää jokin sellainen algoritmi $\mathcal{A}$, jolla löydetään funktio g joka approksimoi funktiota f jostakin hypoteesijoukoksi kutsutusta joukosta $\mathcal{H}$ [1], [2].

Mitchell [2] määritelmään koneoppimisesta kuuluu virhefunktio E, jonka avulla arvioidaan miten hyvin koneoppimisalgoritmi on ratkaissut tehtävän T. Virhefunktion E tarkoituksena on arvioida miten hyvin koneoppimisalgoritmin löytämä funktio g voidaan yleistää sellaisella datajoukolle, jota ei ole vielä aiemmin havaittu.

Marslandin [3] mukaan koneoppimisen soveltaminen koostuu muutamasta vaiheesta. Ensimmäisenä tarvitaan jokin aineisto (eli joukko $\mathcal{D}$), jonka perusteella pyritään löytämään vastauksia haluttuun kysymykseen. Data voi olla esimerkiksi mit-



Kuva 2.1: Koneoppimisen kuvaus [1] mukaan: Tuntematon kohdefunktio $f : \mathcal{X} \to \mathcal{Y}$ pyritään löytämään oppimisalgoritmi $\mathcal{A}$ hypoteeseista $\mathcal{H}$, siten että $g \approx f$

tausaineistoa, kuvia, ääntä tai tekstiä. Ratkaistava ongelma määrittää minkälaista dataa on käytettävä. Toinen vaihe koneoppimisen soveltamisessa on piirteiden valinta, joka tarkoittaa sitä, että pyritään määrittelemään mitkä datan piirteistä ovat relevantteja halutun ongelman ratkaisemiseen. Joissakin aineistoissa muuttujia voi olla paljon, jolloin kaikkien käyttäminen mallissa voi vaikeuttaa mallin käyttöä tai aiheuttaa jopa kustannuksia tai kärsitymistä, jos kyse on esimerkiksi ihmisen terveyteen liittyvästä datasta jonka kerääminen vaatii näytteiden ottamista ihmisestä. Algoritmin valinta perustuu ratkaistavaan ongelmaan ja siihen mitä siitä tiedetään. Usein voi olla tarjolla useita vaihtoehtoisia algoritmeja, joista on löydettävä sopivin annettuun ongelmaan tai löydettävä olemassa olevaan malliin sopivimmat parametrit. Sopivan tai sopivien menetelmien löydyttyä malli on koulutettava. Koneoppimismallit koulutetaan niin, että data jaetaan oppimisaineistoon sekä testiaineistoon - sekä joskus myös validointiaineistoon. Oppimisaineiston avulla koulutetaan malli ja testiaineistolla testataan miten hyvin koulutettu malli yleistyy uuteen, ennen näkemättömään dataan. Menetelmien arviointia käsitellään tarkemmin kappaleessa 2.3.

2.2 Koneoppimismenetelmien tyypit

Ohjattu oppiminen (*engl. supervised learning*) tarkoittaa koneoppimismenetelmiä, joissa datassa $\mathcal{D}$ on mukana ennaltamäärättyjä luokkia tai arvoja, joiden perusteella algoritmi $\mathcal{A}$ pyrkii löytämään halutun funktion $g \approx f$, jonka avulla saadaan minimoitua hävikkifunktiota $\mathcal{L}$. Ohjatun oppimisen voi jakaa karkeasti kahteen alikategoriaan, jotka ovat regressio-ongelmat sekä luokitteluongelmat. Regressio-ongelmissa hävikkifunktion $\mathcal{L}$ arvon toivotaan olevan sellainen, joka minimoi todellisen arvon sekä ennustetun arvon erotusta esimerkiksi pienimmän neliösumman menetelmällä $\mathcal{L}(f, (x, y)) = (f(x) - y)^2$. Luokittelussa pyritään löytämään sellaiset luokittelut arvoille, joilla hävikki $\mathcal{L}(f, (x, y)) = \mathbb{1}_{(-\infty,0)}(yf(x))$ on indikaattorifunktio sille onko mallin tuottama arvo oikein vai väärin [4].

Sähköpostiohjelmien roskapostisuodatin on klassinen esimerkki ohjatusta oppimisesta, jos saapuneita sähköposteja ajatellaan aineistona yhdessä roskapostiluokittelun kanssa. Tällöin ohjattua oppimista käyttäen voidaan pyrkiä määrittämään onko uusi, ennen näkemätön sähköposti roskapostia vaiko ei. Kuvantunnistuksessa ohjattua oppimista esiintyy esimerkiksi kuvien luokittelussa: voi olla tarpeen luokitella onko jokin kuvassa esiintyvä eläin kissa vai koira tai näkeekö auton kamera edessään ihmisen vai liikennemerkin. Tässä työssä käsiteltävässä semanttisessa segmentoinnissa luokittelutehtävä voidaan tulkita jokaiselle kuvan pikselille erikseen siten, että kuuluuko se haluttuun luokkaan. Numeerisen aineiston perusteella ohjatun oppimisen tehtäviä ovat kaikki regressio-ongelmat, joiden perusteella pyritään oppimaan opetusaineiston perusteella miten jokin arvo muuttuu tai kehittyy uusissa tilanteissa. Eräs esimerkki tällaisesta tehtävästä on malli, jonka avulla it-yhtiö pyrki ennustamaan (tai selittämään) asuntojen hintoja eri puolella Suomea [5] demokratisten muuttujien perusteella tai huoneiden lukumäärän perusteella [6]. Muita esimerkkejä ennustemalleista on esimerkiksi asiakkaan elinkaaren arvon ennustaminen, sääennustukset tai vaikkapa sen ennustaminen kuinka monta pyöräilijää

pyöräilee yliopiston ohi päivässä. Generatiivisen tekoälyn eli tekoälyn, joka tuottaa tekstiä, kuvia tai ääntä, suorittamia tehtäviä voidaan tulkita myös ennustemalleina koska niissä malleille on koulutettu suuri määrä aineistoa jonka perustella pyritään luomaan uutta tekstiä [7].

Ohjaamaton oppiminen (*engl. unsupervised learning*) tarkoittaa sitä, että aineistosta $\mathcal{D}$ pyritään oppimaan ilman annettuja luokitteluja. Tyypillinen esimerkki ohjaamattomasta oppimisesta on klusterointi, jossa pyritään löytämään jokin luokka annetuille datapisteille. Esimerkiksi yritykset voivat pyrkiä luokittelemaan asiakkaitaan erilaisiin segmentteihin käyttäen klusterointimenetelmiä sekä asiakkaista tiedossa olevia demografisia tai käyttäytymiseen liittyvää dataa. Ohjaamaton oppiminen soveltuu hyvin myös poikkeamien tunnistamiseen datasta. Klassinen esimerkki on luottokortin käyttö, josta luottokorttiyhtiö kerää paljon dataa liittyen käytettäviin summiin, ostopaikkoihin sekä maihin. Jos luottokorttiyhtiö huomaa yllättäviä poikkeamia käyttäytymisessä, se voi sulkea luottokortin tai informoida käyttäjää poikkeavista tapahtumista. Poikkeamien havainnointia voidaan hyödyntää myös osana laajempaa koneoppimisratkaisua siihen, että tarkistetaan onko havaintoaineistossa poikkeavia havaintoja [6].

Monissa tapauksissa datan merkitseminen (*engl. labeling*) voi olla työlästä ja siten myös kallista, jolloin voi olla tilanne että on merkitsemätöntä ja merkittyä dataa jota koneoppimisratkaisun pitäisi tukea. Puoliohjattu oppiminen (*engl.semi-supervised learning*) voi auttaa tällaisen datan kanssa yhdistäen sekä ohjaamattomia sekä ohjattuja algoritmeja ongelmien ratkaisussa [6]. Esimerkkinä tästä Géron [6] mainitsee kuvapalvelut, joissa osasta kuvia tiedetään kuka kuvassa esiintyy mutta on myös kuvia joita ei ole merkitty.

2.3 Koneoppimismenetelmien arviointi

Koneoppimisen määritelmän [2] mukaan koneoppimisessa pyritään löytämään ongelmalle sopivat parametrit mimimoimalla virhefunktiota E. Mallin arvioinnin tarkoituksena on löytää sellainen funktio, joka virhefunktiota E minimoiden löytää sellaisen funktion g joka yleistyy mahdollisimman hyvin sellaiselle datalle jota malli ei ole vielä nähnyt. Jotta koneoppimismallin hyvyyttä voidaan arvioida niin tarvitaan sellainen menetelmä jossa malli voidaan kouluttaa tietyllä opetusaineistolla ja verrata saatuja tuloksia tunnettuun aineistoon, jota ei ole käytetty mallia kouluttaessa. Yksinkertaisin menetelmä tähän on jakaa data $\mathcal{D}$ sekä opetusjoukkoon $\mathcal{D}_{opetus} \in \mathcal{D}$ että testijoukkoon $\mathcal{D}_{testi} \in \mathcal{D}$, siten että $\mathcal{D}_{opetus} \cap \mathcal{D}_{testi} = \emptyset$ ja $\mathcal{D}_{opetus} \cup \mathcal{D}_{testi} = \mathcal{D}$.

Opetusaineiston tulokset validoidaan testijoukkoa vasten siten, että testijoukossa olevia tunnettuja arvoja verrataan mallin tuottamiin tuloksiin käyttäen haluttuja metriikoita[6]. Käytännössä mallin validointi perustuu siihen, että opetusdatan ulkopuolelle on jätetty testi- tai validointijoukko havaintoja käytössä olevasta aineistosta ja tähän aineistoon tehtyjä ennustuksia verrataan aineistossa oleviin todellisiin havaintoihin.

Opetus- ja testiaineistojen käyttöä hienostuneempia menetelmiä ovat erilaiset ristiinvalidoinnin menetelmät, jotka perustuvat tilastollisiin uudelleenotantamenetelmiin eli siihen että opetusaineistosta otetaan tietyt havainnot mallin koulutusta varten ja mallia testataan jäljellejääneillä havainnoilla. Tällaisia menetelmiä on esimerkiksi *K-Fold Cross-Validation* sekä *LOOCV* eli *Leave-One-Out-CrossValidation*.

K-Fold-menetelmässä datajoukko $\mathcal{D}$ jaetaan k eri joukkoon

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3 \cdots \cup \mathcal{D}_k \tag{2.1}$$

, joista mallin opetus tapahtuu vuoron perään jokaiselle joukolle siten, että joukko $\mathcal{D}_i$, jolle $i \in 1 \ldots k$ toimii testiaineistona eli kyseistä dataa pidetään otannan

ulkopuolisena datana. $LOOCV$-menetelmän avulla mallin opetusdata on $k-1$ datapistettä ja yksi datapiste on jätetty testiaineistoksi. $LOOCV$-menetelmä vastaa *K-Fold*-menetelmää kun $K = N$ eli aineistossa olevien havaintojen määrä. [8], [9]

Koneoppimismallin hyvyyden arviointi perustuu siihen ongelmaan, jota mallilla ollaan ratkaisemassa. Luokitteluongelman ratkaisevan algoritmin hyvyys voidaan mitataan mallin tarkkuudella (*engl. accuracy*), jossa arvioidaan miten hyvin malli on onnistunut luokittelemaan havainnon oikeaan kategoriaan. Regressio-ongelmissa voidaa mitata mikä on oikean ja ennustetun arvon välinen etäisyys.

3 Syväoppiminen

Tässä kappaleessa perehdytään syviin neuroverkkoihin sekä niihin liittyvään peruskäsitteistöön, joka on oleellista konvoluutioneuroverkkojen sekä transformer-arkkitehtuureiden ymmärtämisessä. Syväoppimisen käsittely aloitetaan siitä mitä erityisesti syvät neuroverkot ovat, jonka jälkeen tutustutaan yksinkertaiseen monikerroksiseen havaitsijaan, jolla voidaan luokitella tai ennustaa haluttuja asioita. Tämän jälkeen käsitellään neuroverkkojen tärkeitä ominaisuuksia, joita ovat erilaiset neuronien aktivointifunktiot sekä hyperparametrit. Tämän jälkeen käsitellään neuroverkkojen toiminnan arviointiin kehitettyjä menetelmiä. Lopuksi käsitellään siirto-oppimista sekä erityisesti konvoluutioverkkoja, joita voidaan hyödyntää piirteiden erotteluun esimerkiksi kuvista.

3.1 Syväoppiminen

Neuroverkko on suunnattu asyklinen graafi, jonka solmut suorittavat laskutoimituksia. Syväoppiminen tarkoittaa funktioiden approksimointia neuroverkkojen avulla perustuen johonkin datajoukkoon $\mathcal{D} = \{x_i\}$. Roberts [10] esittää yksinkertaisena ideana funktion $f(x)$ approksimointia, joka ottaa syötteenä kuvan ja palauttaa 1 jos kuvassa x_i on koira tai 0, jos kuvassa ei ole koiraa. Tavoitteena on löytää joukko mallin parametreja θ_μ, joukolle funktioita $\{f(x;\theta)\}$, joiden parametreja θ_μ muuttamalla voidaan löytää approksimaatio $f(x;\theta^*) \approx f(x)$. Prosessia, jossa mallin parametreja θ_μ muokataan kutsutaan oppimisalgoritmiksi.

Yksinkertaisin yksikkö syvässä neuroverkossa on neuroni, joka voidaan esittää Robertsin [10] mukaan matemaattisesti kahtena funktiona. Ensimmäinen on esiaktivointifunktio $z_i(s)$, jossa b_i on kyseisen kerroksen harha, $\mathbf{W}$ on painotusmatriisi ja s on syöte tai signaali kyseiselle kerrokselle.

$$z_i(s) = b_i + \sum_{j=1}^{n_{in}} \mathbf{W_{ij}} s_j, i = 1 \dots n_{out} \tag{3.1}$$

ja toinen on funktio on kyseisen neuronin aktivointifunktio kerrokselle i

$$\sigma_i \equiv \sigma(z_i) \tag{3.2}$$

.

Géron [6] esittää saman asian matriisimuodossa syötematriisille $\mathbf{X}$, painolle $\mathbf{W}$ sekä harhalle $\mathbf{b}$, siten että ϕ on aktivointifunktio.

$$h_{\mathbf{W},\mathbf{b}}(\mathbf{X}) = \phi(\mathbf{XW} + \mathbf{b}). \tag{3.3}$$

Aktivointifunktion symboli näyttää vaihtelevan kirjallisuudessa riippuen siitä kuka siitä kirjoittaa. Robertsin [10] ja Géronin [6] lisäksi Bishop [11] käyttää neuroverkkojen piilokerroksen aktivointifunktiosta merkintää h, mutta merkintää ϕ kun kyse on kantafunktion (*basis function*) aktivoinnista. Tässä työssä, selkeyden vuoksi käytetään aktivointifunktiosta merkintää σ.

Näiden komponenttien pohjalta voidaan rakentaa monimutkaisempia verkkoja, joista yksinkertaisin on monikerroksinen havaitsin (*engl. Multilayer Perception*, MLP). Monikerroksinen havaitsin on niin kutsuttu täysin yhdistetty verkko (*engl. fully connected network*) eli jokainen neuroni on linkitetty toiseen [6], [10]. Kuvassa 3.1 on esitetty yksinkertainen monikerroksinen havaitsija. Ensimmäinen kerros, jolle $l = 1$, on kerros joka saa syötteenä neuroverkkoon syötettävän datan. Kerros $l = 2$

on niin kutsuttu piilokerros ja viimeisenä on yksi ulostulokerros.

Roberts [10] esittää monikerroksisen havainnoijan arvojen päivittämisen seuraavalla tavalla rekursiivisena kaavoina

$$z_i^{(1)}(x_\alpha) \equiv b_i^{(1)} + \sum_{j=1}^{n_0} \mathbf{W_{ij}^1} x_{j;\alpha} \tag{3.4}$$

$$z_i^{(l+1)}(x_\alpha) \equiv b_i^{(l+1)} + \sum_{j=1}^{n_l} \mathbf{W}_{ij}^{l+1} \sigma(z_j^l(x_\alpha)), \tag{3.5}$$

joissa kaava 3.4 kuvaa ensimmäistä kerrosta, ja kaava 3.5 kuvaa myöhempiä kerroksia. Jokainen neuroni laskee seuraavan kerroksen painojen W^{l+1} ja oman kerroksen aktivointifunktion tulon. Yksinkertaisin tapa edetä verkossa on se, että arvo $\sum_{j=1}^{n_0} \mathbf{W_{ij}^1} x_{j;\alpha}$ on suurempi kuin $-b_i$, jolloin piilokerroksen neuroni aktivoituu. Syötevektori ja ensimmäisen kerroksen painot yhdessä aktivointifunktion kanssa päättävät laukeaako piilokerrokset lainkaan ja syvempien kerroksien painot yhdessä edellisten kerroksien ulostulojen kanssa päättävät laukeaako kerroksen neuronit vaiko eivät [3], [10].

Aktivointifunktion valinta perustuu siihen minkälaista jakaumaa ollaan mallintamassa ja mitä ongelmaa ratkaisemassa [11]. Ilman aktivointifunktiota neuroverkko olisi vain lineaarinen regressiomalli $\mathbf{y} = \mathbf{WX} + b$, jolla ei pystyisi mallintamaan monimutkaisia datan välisiä suhteita [3], [6], [10], [12]–[14]. Neuroverkkoja rakennettaessa aktivointifunktion voi valita melko vapaasti, mutta se on oltava derivoituva sekä huomioitava verkon rakenne mahdollisimman hyvin eli neuronien sekä kerroksien määrä. Verkon eri kerroksilla voi olla eri aktivointifunktiot.

Yksinkertaisin aktivointifunktio voi olla identiteettifunktio, mutta sen käyttö ei eroaisi siitä kuin jos mallintaisi verkkoa lineaarisena funktiona, jolloin valinta kohdistuu derivoituviin funktioihin, jotka saturoituvat eli ovat vakioita määrittelyvälin kummassakin päässä ja vaihduttava saturaatioarvojen välillä nopeasti, jotta aktivointi voi tapahtua. Joidenkin aktivointifunktioiden, erityisesti logistisen funktion

tanh kanssa käy niin, että funktion arvo lähestyy nollaa jos painot ovat negatiivisia [3], [11].

Viimeisen kerroksen eli ulostulokerroksen aktivointifunktion valinnassa vaikuttaa se mitä ongelmaa ollaan ratkaisemassa. Binäärisessä luokittelussa eli luokittelussa, jossa mahdollisia arvoja on kaksi, aktivointifunktio on usein logistinen sigmoidifunktio eli

$$\sigma(a) = \frac{1}{1 + exp(-a)}. \tag{3.6}$$

Jos tavoitteena on tehdä luokittelua useampaan luokkaan, tyypillinen verkon ulostulokerroksen aktivointifunktio on ns. Softmax-funktio vektorille $\mathbf{z}$

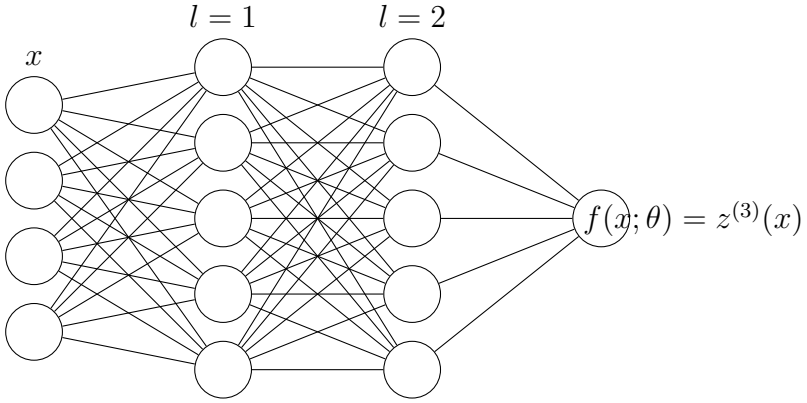
$$\sigma(\mathbf{z}) = \frac{e^z}{\sum_{j=1}^{K} e^z}. \tag{3.7}$$

Taulukossa 3.1 on esitetty tyypillisimpiä aktivointifunktioita lähteen [6] mukaan.

Funktio	**Kuvaus**	**Kaava**
Sigmoid	Sigmoid-funktio palauttaa arvon väliltä 0 ja 1 ja soveltuu hyvin malleihin, joissa lasketaan todennäköisyyksiä.	$\sigma(x) = \frac{1}{1+e^{-x}}$
ReLU	ReLU on lineaarinen funktio positiiviselle syötteelle ja nolla negatiivisille	$f(x) = \max(0, x)$
Tanh	Tanh palauttaa arvon -1 ja 1 välillä, samaan tapaan kuin sigmoid. Tanh-arvot ovat normalisoituja nollan lähellä, mikä tekee siitä nopeamman tietyntyyppisille kerroksille	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Softmax	Softmaxia käytetään tyypillisesti tehtävissä, joissa on tarkoituksena luokitella moneen luokkaan esimerkiksi kuvia.	$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$
Leaky ReLU	ReLU-funktion variantti, jossa sallii pienen vakion ϵ gradienttiin $x < 0$.	$f(x) = \max(\epsilon x, x)$

Taulukko 3.1: Tyypillisiä aktivointifunktioita

Kuva 3.1: Yksinkertainen monikerroksinen havaitsija



3.2 Neuroverkkojen koulutus ja optimointi

Neuroverkko koulutetaan ja koulutus vaatii tiettyjen parametrien määrittämistä. Neuroverkon arkkitehtuuri valitaan ratkaistavan ongelman perusteella ja erilaisia tehtäviä varten on erilaisia lähestymistapoja. Kappaleessa 3.6 esitetään konvoluutioneuroverkot, jotka ovat kuvantunnistuksessa tyypillisiä arkkitehtuureita, mutta yleisesti neuroverkon rakentaminen aloitetaan siitä että suunnitellaan verkon syvyys eli kuinka monta piilokerrosta on, kuinka leveitä kerrokset ovat, mikä on hävikkifunktio (*eng. loss function*) ja minkälaiset painot eri kerroksille asetetaan aluksi [13].

Princen [14] mukaan hävikkifunktio $L(x)$ on funktio, joka kuvaa mallin ennustaman arvon $f(\mathbf{x};\theta)$ sekä todellisen arvon y_i välistä eroa. Matemaattisesti, täsmällisemmin määriteltynä, hävikkifunktio on generoidun todennäköisyysmallin suurimman uskottavuuden estimaatti. Suurimman uskottavuuden estimaattia laskettaessa pyritään löytämään jokin todennäköisyysfunktio $P(y_i|x_i)$ ennustetulle arvolle y_i syötteellä x_i tietyin parametrein θ, jotka minimoivat hävikkiä eli

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}[\prod P(y_i|f(x_i;\theta)]. \tag{3.8}$$

Hävikkifunktiota käytetään osana virheen laskentaa neuroverkossa. Goodfellown [12]

mukaan se voidaan esittää yksinkertaistetusti esitettynä hävikkifunktion odotusarvona. Virhefunktion $E(\mathbf{W})$ avulla voidaan arvioida kuinka suuri virhe oikean arvon sekä mallin ennustaman virheen välillä on. Käytännössä suurimman uskottavuuden estimaatti lasketaan logaritmisena, koska liukulukuaritmetiikan vuoksi tulo voi hyvin pienillä arvoilla nolla. Logaritmin laskusäännöillä tämä voidaan esittää kuitenkin summana.

Jotta neuroverkko voi oppia, on virheen $E(\mathbf{W})$ oltava mahdollisimman pieni, eli on pystyttävä minimoimaan tai löytämään neuroverkon virhefunktiolle mahdollisimman pieni arvo. Neuroverkon virhefunktiot voivat olla hyvin monimutkaisia, koska verkon rakenne on harvoin yksinkertainen jolloin yhtälölle $\nabla E(\mathbf{W}) = 0$ ei löydy helposti analyyttista ratkaisua [11]. Ratkaisu sopivan virheen arviointiin on niin kutsuttu gradienttimenetelmä (*engl.gradient descent*), jonka avulla voidaan löytää optimaaliset painomatriisille $\mathbf{W}$ arvot iteratiivisesti. Tällöin voidaan pienesti lisätä arvoja tähän ja löytää optimaaliset painot iteroimalla $\mathbf{W}^{(\tau)} = \mathbf{W}^{(\tau-1)} + \Delta\mathbf{W}^{(\tau-1)}$, jossa τ merkitsee kuinka monennesta iteraatiosta on kysymys. Painojen päivitysarvo $\Delta\mathbf{W}$ riippuu siitä, millaisella algoritmilla painoja halutaan päivittää [11].

Stokastisessa gradienttimenetelmässä (*engl. Stochastic Gradient Descent*) painojen päivitys tapahtuu iteroimalla läpi datan n pistettä algoritmia 1 käyttäen. Algoritmin tärkeä osa on kaava 3.9, jossa määritetään miten algoritmi päivittää painovektorin arvoja kaavalla:

$$\mathbf{W} = \mathbf{W} - \eta\nabla E_n(\mathbf{W}), \tag{3.9}$$

missä η tarkoittaa mallin opetusastetta, $\mathbf{W}$ on mallin painomatriisi ja $E_n(\mathbf{W})$ on virhefunktio aineiston n:lle pisteelle.

Ohjelmalistaus 1 Stokastinen gradienttimenetelmä

Syöte: Opetusjoukko indeksoituna $n \in \{1 \dots N\}$, Virhefunktio $E[\mathbf{w}]$, Opetusaste η, alustava painovektori $\mathbf{w}$

Tulos: Lopullinen painovektori

1: $n \leftarrow 1$
2: **repeat**
3: $\quad \mathbf{W} \leftarrow \mathbf{W} - \eta \nabla E_n(\mathbf{W})$
4: $\quad n \leftarrow n + 1(modN)$
5: **until** konvergoituu
6: **return W**

Algoritmissa 1 esiintyy parametri η, joka neuroverkon opetusaste (*learning rate*, jonka avulla kontrolloidaan miten nopeasti neuroverkon kerroksien painot optimoidaan. Jos oppimisaste on liian pieni, optimointialgoritmi joutuu ottamaan paljon askelia löytääkseen optimaaliset verkon painot, mutta vastaavasti jos oppimisaste on liian iso, optimointialgoritmi saattaa hypätä joidenkin arvojen yli ja ei pysty löytämään hyvää ratkaisua painoille [6].

Konvergoituminen (*engl. convergence*) tarkoittaa gradienttimenetelmissä sitä, että algoritmi löytää mahdollisen optimaalisen minimiarvon. Se, millainen virhefunktio on käytössä määrittää sitä, miten hyvin algoritmi konvergoituu [6], [11].

Syvien neuroverkkojen kouluttamisessa voidaan käyttää gradienttimenetelmän sijaan kehittyneempiä optimointimenetelmiä, kuten minierägradienttimenetelmää, *AdaGrad*-, *RMSProp*- tai *Adam*-menetelmää. Minierämenetelmässä gradientti lasketaan pienelle erälle erikseen. AdaGrad-menetelmän ajatuksena on pienetää oppimiastetatta opetuksen edetessä käyttäen jokaisen parametrin derivaatan neliösummia. Menetelmän ongelmana on kuitenkin se, että arvot voivat muuttua opetuksen edetessä hyvin pieneksi aiheuttaen opetuksen hidastumista. *RMSProp*-menetelmä

(esitelty esim [6]) pyrkii korjaamaan *AdaGrad*-menetelmän [15] ongelmia sillä että gradienttien neliösummat korvataan eksponentiaalisesti painotetulla keskiarvolla. Näiden menetelmien yhdistelmä on Adam [16], joka on on stokastinen gradienteihin perustuva optimointimenetelmä, jolla on hyvin pieni muistivaatimus. Menetelmässä jokaiselle parametrille lasketaan omat oppimisasteet siten että huomioidaan sekä eksponentiaalisesti painotetut keskiarvot gradienteille sekä gradienttien neliöille.

3.3 Vastavirta-algoritmi

Vastavirta-algoritmi (*engl. backpropagation*) on automaattiseen derivointiin perustuva menetelmä, jonka avulla voidaan löytää gradientit monimutkaisista funktioista, kuten esimerkiksi neuroverkoista ja käyttää näitä myöhemmin optimointialgoritmin, kuten aiemmin esitetyn stokastisen gradienttimenetelmän avulla.

Vastavirta-algoritmissa neuroverkkoa ajatellaan joukkona yhdistettyjä funktioita, jotka välittävät viestejä eteenpäin kerrokselta toiselle, eli matemaattisesti ilmaistuna kerroksen $\mathbf{z}_{l-1}$ tulos toimii syötteenä kerrokselle $\mathbf{z_l} \triangleq f_l(\mathbf{z}_{l-1}; \theta_l)$. Verkon tulos, eli hävikkifunktio $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ voidaan ajatella verkon kerroksena $L + 1$. Näiden kerroksien perusteella voidaan laskea koko verkolle funktio yhdistämällä rekursiivisesti kerroksia toisiinsa ja tätä kutsutaan päättelyksi neuroverkossa, koska sen tehtävänä on päätellä ulostulokerroksen jakauma annetuilla parameterillä. Kouluttaessa neuroverkkoa jokainen kerros saa seuraavalta kerrokselta takaisin päin viestinä seuraavan kerroksen gradientin ja tuottaa kaksi takaisinpäin menevää viestiä ulostulona: gradientin edelliseltä kerrokselta sekä kyseisen kerroksen parametrien gradientin. Näihin voidaan soveltaa derivaatan tulosääntöä, jolloin saadaan koko verkolle laskettua gradientti hävikkifunktion suhteen. [17], [18]

3.4 Siirto-oppiminen ja mallien hienosäätö

Siirto-oppimisella tarkoitetaan sitä, että yhden ongelman ratkaisuun koulutettua neuroverkkoa voidaan hyödyntää jonkin toisen ongelman ratkaisussa käyttämällä valmiita verkon painoja siten, että käytetään laajemmalla aineistolla koulutetun mallin luokittelukerros uudelleenkoulutetaan tehtävään paremmin sopivalla, pienemmällä aineistolla jos aineistossa on samankaltaisuuksia. Esimerkkinä voidaan ajatella vaikkapa mallia, joka on koulutettu tunnistamaan laajalti erilaisia eläinlajeja, ja jonka pohjalta myöhemmin halutaan kouluttaa malli tunnistamaan jotain tiettyjä eläimiä kuten vaikkapa lintuja.

Siirto-oppimisen toiminta perustuu siihen, että jossakin laajemassa luokittelutehtävässä voi olla käytössä sellaista dataa joka sisältää yhteisiä piirteitä myös pienemmälle luokittelulle ja verkosta voidaan kouluttaa vain yksittäisiä tai muutamia kerroksia.

Siirto-oppimisen hyötynä on se, että mallista voidaan kouluttaa vain pieni osa ja tällöin saadaan malli nopeammin käyttöön. Käytännön sovelluksissa on syytä huomioida myös se, että monien mallien kouluttaminen voi kestää hyvinkin pitkään eikä kaikilla ole saatavilla dataa tai resursseja kouluttaa isoja malleja.

Käytännöllisissä sovelluksissa, esimerkiksi Keras-kirjastoa käyttäessä,
siirto-oppiminen tapahtuu niin että käytetään olemassa olevaa, koulutettua mallia jonka kerrokset määritellään niin että niiden painot eivät päivity opetuksessa ja malliin lisätään uusia kerroksia.

Mallin hienosäätö tarkoittaa sitä, että kun malli on saatu koulutettua jollakin aineistolla, eli se on konvergoitunut, malli koulutetaan uudestaan erittäin pienellä oppimisasteella.

3.5 Tietämyksen siirtäminen

Tietämyksen siirtäminen (*knowledge distillation*) tai opettaja-oppilas-arkkitehtuuri, on koneoppimismenetelmä, jossa pienempi koneoppimismalli oppii jostain suuremmasta neuroverkosta. Pienempää neuroverkkoa kutsutaan oppilaaksi ja suurempaa opettajaksi. Hinton et al. [19] vertaavat neuroverkkojen kehittämistä ja koulutusta hyönteisiin, joilla on kaksi muotoa: toukka ja aikuinen. Toukan muoto on optimoitu keräämään mahdollisimman paljon energiaa ja ravinteita ympäristöstä, kun taas aikuisen hyönteisen rakenne mahdollistaa liikkumisen sekä lisääntymisen. Hinton vertaa hyönteisiä neuroverkkoihin siten, että erilaisia käyttötarpeita varten on erilaisia verkkoja: kouluttaminen edellyttää laajempia, isoja verkkoja kun taas operationaalisessa käytössä tarvitaan laskennallisesti tehokkaampia ratkaisuja.

Tietämyksen siirtämisen menetelmät voidaan jakaa karkeasti kolmeen kategoriaan: vastaperustaiseen, ominaisuusperustaiseen ja suhdeperustaiseen [20]. Tässä rajataan perehtyminen vasteperusteiseen tietämykseen ja sen siirtämiseen, koska se on myöhemmin käsiteltävien transformer-arkkitehtuureiden kannalta keskeisempi menetelmä. Vasteperustanen tietämys (*engl. Response-Based Knowledge*) tarkoittaa sellaista tietämystä, joka on opettajaverkon viimeisessä kerroksessa ja tavoitteena on matkia opettajaverkon luokittelua. Oppilasverkon koulutuksessa on käytössä kaksi tavoitefunktiota, joita optimoidaan. Ensimmäinen tavoite on optimoida ristientropia hävikkifunktiosta oppilasverkon ennustusten ja opettajamallin tarjoamien "pehmeiden"ennustusten välillä. Pehmeillä ennnusksilla tarkoitetaan ennustuksia, joita on muokattu $softmax$-funktion lämpötilaparametrilla T, jotta se sisältäisi enemmän informaatiota. Toinen tavoite on minimoida ristientropiaa hävikkifunktiosta oppilaan ennustusten ja oikeiden, todellisten arvojen välillä. Tässä on tavoitteena, että oppilasmalli oppii tekemään ennustuksia myös itse eikä vain täysin matki opettajamallin käytöstä. Nämä kaksi tavoitetta yhdistetään usein yhdeksi hävikkifunktioksi,

joka voi olla kahden hävikkifunktion ristientropia siten, että

$$L_{\text{ResD}}(p(z_t, T), p(z_s, T)) = \mathcal{L}_R(p(z_t, T), p(z_s, T)), \tag{3.10}$$

jossa z_t on opettajaverkon ja z_s oppilasverkon logit-arvot, T lämpötila ja L_{ResD} ja $\mathcal{L}_R$ hävikkifunktioita.

3.6 Konvoluutioarkkitehtuurin perusteet

Konvoluutioneuroverkko pyrkii ottamaan huomioon kuvien ominaisuudet niin, että kuvia ei käsitellä yksittäisten pikseleiden avulla vaan huomio keskittyy kuvissa isompiin alueisiin kuten ihmisen näköaistissa eli aivokuoren visuaalista informaatiota käsittelevässä osassa. Konvoluutioneuroverkot perustuvat kahden funktion väliseen integraalimuunnokseen $(f * g)(\mathbf{x}) = \int_{\mathbb{R}^D} f(\mathbf{u})g(\mathbf{z} - \mathbf{u})d\mathbf{u}$ funktioille $f, g : \mathbb{R}^D \to \mathbb{R}$ [17].

Konvoluutioneuroverkoissa konvoluutio-operaatio tehdään yksiulotteisen datan sijaan kaksiulotteisille matriiseille, jolloin konvoluutio esitetään kuvalle I ja filtterille K yhtälönä

$$C(i, j) = \sum_l \sum_m I(j + l, k + m)K(l, m), \tag{3.11}$$

jossa $\mathbf{K}$ on suodatin tai ydin (*engl. kernel*). Tämä voidaan esittää myös matriisimuodossa $\mathbf{C} = \mathbf{I} * \mathbf{K}$ Konvoluution avulla voidaan tunnistaa kuvasta ominaisuuksia ajatellen operaatiota mallin sovituksena (*engl. template matching*) ja lopputulos on ominaisuuskartta. Konvoluutio pienentää alkuperäistä kuvaa uuteen kokoon riippuen konvoluutiokernelin koosta. [11], [17]. Kaksiulotteinen konvoluutio soveltuu harmaasävyisille tai mustavalkokuville. Käytännössä kuvat voivat kuitenkin olla värikuvia, mikä tarkoittaa sitä että kuvassa on useampia kerroksia. Kuva, jossa on useampia päällekkäisiä kerroksia voidaan ajatella matemaattisesti moniulotteisena

matriisina eli tensorina. Konvoluutio voidaan kohdistaa myös tensoriin, jolloin saadaan joka kerroksesta omat ominaisuuskartat.

3.7 Konvoluutioneuroverkon kerrokset

Konovoluutioneuroverkkojen kerrokset eroavat täysin yhdistetyistä verkoista siten, että kerroksien neuronit eivät suoraan liity yksittäisiin syötteen pikseleihin vaan kappaleessa 3.6 esitetyllä tavalla pyritään ymmärtämään kuvien piirteitä.

Pooling eli yhdistämiskerroksen tarkoituksena on pienentää kuvan kokoa peräkkäisissä konvoluutiokerroksissa, jotta neuroverkon laskennallinen vaatimus ja mallin parametrien määrä laskee. Yhdistämiskerroksessa kuvan kokoa pienennetään niin, että kuvasta tietystä alueesta kuvaa otetaan jokin arvo joka edustaa aluetta seuraavassa kerroksessa. Tyypillisesti tähän voidaan käyttää maksimiarvoa.

Padding eli täyttökerroksien tarkoitus on huolehtia konvoluutioneuroverkoissa siitä, että ominaisuudet pysyvät konvoluutioneuroverkoissa käyttökelpoisina. Ilman täyttöoperaatiota kuvien koko pienenisi mentäessä kerroksia etenepäin, jolloin ne muuttuisivat lopulta käyttökelvottomaksi. Toinen ongelma voi tulla vastaan kuvan reunapikseleiden kanssa, joita ei otettaisi yhtä paljoa huomioon jos ei tehtäisi täyttöä.

Suodatin-kerroksilla (*engl. filter*) kontrolloidaan konvoluutiokerroksen ulostulon kokoa ja määritellä siihen täyttö tai yhdistäminen, tarpeen mukaan.[6]

3.8 Neuroverkon sääntely

Neuroverkkojen sekä muiden koneoppimismallien sääntelyn tarkoituksena on varmistaa se, että mallit yleistyvät hyvin uuteen, ennen näkemättömään dataan. Koneoppimisen *No Free Lunch Theorem* [12], [21] sanoo, että kaikki koneoppimismallit ovat keskimäärin yhtä hyviä tai huonoja kun tarkastalleen kaikkia mahdollisia aineisto-

ja. Tämä tarkoittaa sitä, että neuroverkkojen kehittäessä on pyrittävä löytämään annettuun tehtävään parhaiten soveltuva malli. Induktiivinen harha (*engl. inductive bias*) tarkoittaa sitä, että mallia valitessa otetaan huomioon joitakin oletuksia dataan liittyen. Vaikka koneoppiminen tapahtuu pitkälti opetusdatan perusteella, mallien parametrit vaikuttavat myös vahvasti siihen miten hyvin ne soveltuvat oppimiseen [11].

Soveltuvan mallin löytäminen edellyttää malliin liittyvän virheen tai hävikin minimointia, mutta voi myös edellyttää myös sopivien mallin parametrien etsimistä ja säätämistä. Oikeaa mallia etsittäessä voidaan huomata, että jokin malli yli- tai alisovittuun annettuun aineistoon ja ei sen vuoksi yleisty hyvin uuteen dataan. Mallin parantamiseksi tarvitaan erilaisia sääntelytekniikoita.

Yksi tapa säädellä mallia on rajoittaa mallin kapasiteettia asettamalla tavoitefunktiolle jonkinlainen rangaistusfunktio $\Omega(\theta)$ niin, että $\mathbf{J}(\theta) = \mathbf{J}(\theta) + \mathbf{\Omega}(\theta)$.

Rangaistusfunktioon perustuvia sääntelytekniikoita on L^1 ja L^2-normeihin perustuvat sääntelyt. L^2-normiin perustuvaa sääntelyä kutsutaan myös painojen pudotukseksi (*engl. weight decay*).

4 Transformer-arkkitehtuurit

Tässä kappaleessa käsitellään transformer-arkkitehtuureita ja niiden käyttöä erityisesti kuvantunnistustehtävissä. Aluksi käsitellään mitä transformerit tarkoittavat, jonka jälkeen tarkastellaan tranformer-arkkitehtuureiden taksonomiaa sekä kuvamuuntimia.

4.1 Perusteet

Transformereita voidaan pitää yhtenä tärkeimmistä koneoppimisen kehitysaskeleista [11]. Transformer-arkkitehtuurin on kehittänyt Googlen tutkijat vuonna 2017, jolloin he julkaisivat artikkelin *Attention is all you need* [22]. Transformer-arkkitehtuurien konsepti perustuu huomion eli englanniksi *attention* käsitteeseen ja funktioon, joka yhdistää kyselyn ja arvo-avainparit haluttuun ulostuloon, siten että funktion arvo on arvojen painotettu summa ja jokainen paino on laskettu syötteenä annettujen avaimien mukaisesti [22].

Transformer-arkkitehtuuria hyödyntävä neuroverkko laskee eri painot eri verkon syötteille niin, että saadaan muodostettua voimakas induktiivinen harha.

Transformer-arkkitehtuurien nimitys *transformer* (*suom. muunnin*) tulee siitä, että arkkitehtuurissa muunnetaan syötevektorit tietystä esitystavasta uuteen esitystapaan siten että se soveltuu paremmin ongelman ratkaisuun.

Yleinen approksimaatiolause (*engl. Universal Approximation Theorem*) osoittaa, että yksikerroksinen rajoittamattoman monen parametrin neuroverkko voi mallintaa

mielivaltaista jatkuvaa funktiota millä tarkkuudella hyvänsä [11], [23]. Yun et al. [24] ovat osoittaneet, että transformer-arkkitehtuuri on universaali approximaatio jatkuville ja permutaatioekvivalenteille (*engl. Permutation Equivariant*) sekvenssistä-sekvenssiin-funktoille, ja sijaintikoodauksien myötä myös jatkuville sekvenssi -sekvenssi-funktioille.

Alkuperäinen sovellusala transformer-arkkitehtuureille on ollut luonnollisen kielen käsittely ja siihen liittyvä koneoppiminen, mutta arkkitehtuuria voi soveltaa myös kuviin sekä vaikkapa proteiinirakenteisiin [11], aikasarjoihin [25] tai puheen tunnistamiseen [26]. Kuvamuuntimia käsitellään kappaleessa 4.4.

Transformer-arkkitehtuurissa syötteenä on joukko $\{x_n\}$ D-ulotteisia vektoreita (missä $n = 1 \dots N$), jossa yhteen vektoriin viitataan *tokenina* joka voi olla sana, kuvan osa tai jokin muu syötedataan liittyvä elementti. Transformer eli muunnin muuttaa syötteenä saadut vektorit $\mathbf{x_m}$ uuteen muotoon $\mathbf{y_n}$ esittäen ne lineaarikombinaationa

$$\mathbf{y_n} = \sum_{m=1}^{N} a_{mn}\mathbf{x_m} \tag{4.1}$$

siten, että parametreille $a_{mn} \geq 0$, joita kutsutaan huomiopainoksi (*attention weight*) pätee ehto $\sum_{n=1}^{N} a_{mn} = 1$. Huomiopainon arvo riippuu siitä kuinka tärkeä tietty token on, joten jos ei ole tärkeä niin arvo on lähellä nollaa. Jos parametrin arvo $a_{mn} = 0$, niin se tarkoittaa sitä että parametrille ei tehdä minkäänlaista muunnosta. Esitetty kaava 4.1 voidaan esittää matriisimuodossa

$$Y = XX^T. \tag{4.2}$$

Huomiopainojen määrittely perustuu siihen, miten lähellä kaksi vektoria $\mathbf{x_m}$ ja $\mathbf{x_n}$ ovat toisiaan normalisoituna *softmax*-funktiolla.

$$a_{mn} = \frac{x_n^T x_m}{\sum_{m=1}^{N} exp(x_m^T x_n)} \tag{4.3}$$

Softmax-funktion tarkoitus transformer-arkkitehtuurissa ei ole laskea todennäköisyyksiä vaan ainoastaan normalisoida huomiopainot sopivaan muotoon. Koska painojen lineaarikombinaatio kaavassa 4.1 voidaan esittää matriisimuodossa kuten kaavassa 4.2 esitetään, niin kaavalle 4.3 saadaan esitysmuoto

$$\mathbf{Y} = Softmax[\mathbf{X}\mathbf{X}^T]X, \tag{4.4}$$

jossa $Softmax[\mathbf{L}]$ normalisoi syöteenä saadun matriisin siten, että jokaisesta matriisin $\mathbf{L}$ elementistä eksponentin ja normalisoi jokaisen rivin itsenäisesti siten että summaksi muodostuu yksi.

Kaavan 4.4 avulla transfromer-arkkitehtuurin verkko ei kuitenkaan voi vielä oppia mitään, koska se ei sisällä mitään opittavia parametreja. Kaavaan voidaan lisätä kokoa $D \times D$ -kokoinen matriisi $\mathbf{U}$, joka käyttäytyy kuin kerros tavallisessa neuroverkossa. Tällöin muunnokselle saadaan muoto

$$\mathbf{Y} = Softmax[\mathbf{X}\mathbf{U}\mathbf{U}^\mathbf{T}\mathbf{X}^\mathbf{T}]\mathbf{X}\mathbf{U}, \tag{4.5}$$

jonka ongelmana on kuitenkin matriisin $\mathbf{X}\mathbf{U}\mathbf{U}^\mathbf{T}\mathbf{X}^\mathbf{T}$ symmetrisyys, kun huomiomekanismissa olisi syytä huomioida asymmetrisyys, jotta malli voi oppia joustavammin. Tämän vuoksi määritellään erikseen kysely (query), avain (key) ja arvo (value) matriisit, joilla on omat itsenäiset lineaariset transformaatiot

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathbf{q})} \tag{4.6}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(\mathbf{k})} \tag{4.7}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(\mathbf{v})}, \tag{4.8}$$

jossa matriisit $\mathbf{W}^{(\mathbf{q})}$, $\mathbf{W}^{(\mathbf{k})}$ ja $\mathbf{W}^{(\mathbf{v})}$ ovat painoja, joiden arvot opitaan arkkitehtuu-

rin myöhemmässä vaiheessa. Matriisien koot on valittava siten, että niiden välillä voidaan laskea pistetulot. Näiden matriisien avulla yhtälö 4.4 voidaan esittää muodossa

$$\mathbf{Y} = Sotfmax[\mathbf{Q}\mathbf{K}^T]V. \tag{4.9}$$

Visuaalisesti matriisin $\mathbf{Q}\mathbf{K}^T$ laskenta tapahtuu kuvan 4.2 mukaisesti. Näiden pohjalta voidaan muodostaa elementit, joiden avulla voidaan muodostaan transformer-arkkitehtuurin pohjalta tehtäviä neuroverkkoja [11], [22]. Näitä käsitellään seuraavassa kappaleessa.

4.2 Transformer-kerrokset

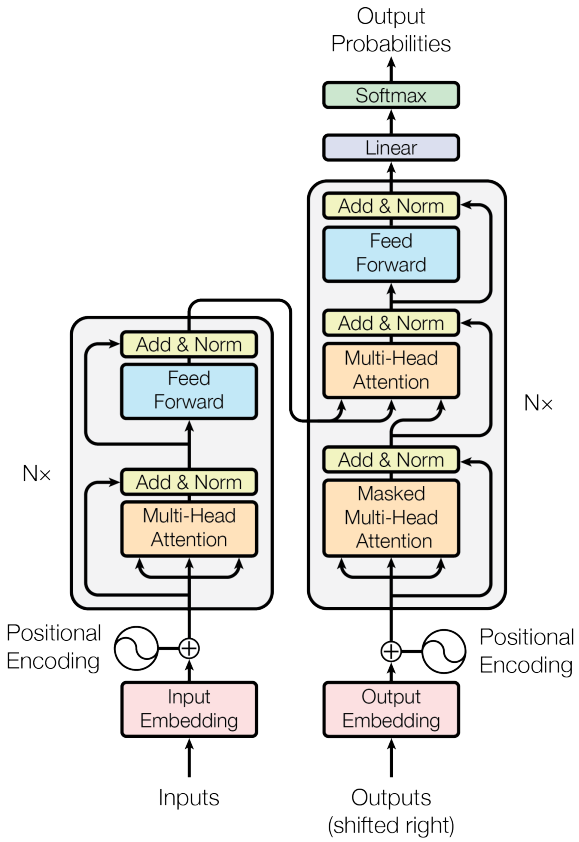
Edellisessä kappaleessa esitettiin transformer-arkkitehtuurin peruselementti eli huomiomekanismi, jonka pohjalta voidaan muodostaa kaksi erilaista transformer-arkkitehtuurissa tarvittavaa rakennetta, skaalattu itsehuomio (*engl. scaled self-attention*) ja monipäinen huomio (*engl. multi-head attention*).

Skaalattu itsehuomio muodostuu kaavojen 4.6, 4.7 ja 4.8 pohjalta siten, että

$$\mathbf{Y} = Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \equiv Softmax[\frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{D_k}}]\mathbf{V}, \tag{4.10}$$

joka tarkoittaa käytännössä sitä että mekanismi laskee ensin matriiseille $\mathbf{Q}$ sekä $\mathbf{Q}^T$ matriisitulon, joka skaalataan arvolla $\sqrt{D_k}$ ja tämä kerrotaan matriisilla $\mathbf{V}$. Tätä kutsutaan myös skaalatuksi pistetulo-itsehuomioksi (*engl. scaled dot-product self attention*) ja se on esitetty algoritmissa 2 [11], jossa riveillä 1-3 on kaavojen 4.6 - 4.8 kaavat, ja lopullinen palautusarvo algoritmista on 4.9.

Käytännössä yksittäinen itsehuomiokerros itsessään ei riitä arkkitehtuurin pohjaksi, koska se saattaisi aiheuttaa sen että eri piirteet olisi laskettu vain keskiarvoilla. Tämän ongelman korjaamiseksi käytetään monipäistä huomiota, joka koos-



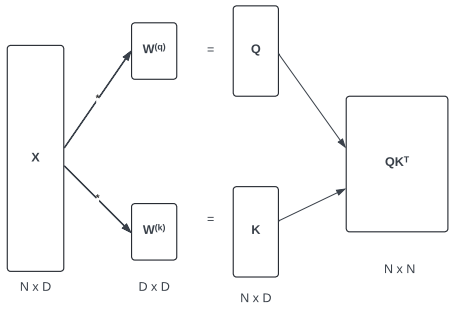
Kuva 4.1: Transformer-arkkitehtuurin rakenne. Kuvan lähde: [22]

tuu useammasta itsenäisesti oppivasta itsehuomiosta, jotka yhdistetään. Monipäinen huomio esitetään siis muodossa

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_n)\mathbf{W}^O, \tag{4.11}$$

missä $head_i = Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$, jossa $\mathbf{Q}, \mathbf{K}$ ja $\mathbf{V}$ ovat kaavoista 4.6 - 4.8 saadut arvot ja matriiseille pätee $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ ja $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ Lopuksi saadut projektiot vielä skaalataan uudestaan käyttäen skaalattua itsehuomiota, siten että sille tehdään vielä lineaarinen transformaatio käyttäen painomatriisia $\mathbf{W}^O$, jolloin saadaan lopullinen muoto monipäiselle huomiolle.

Huomiomekanismia käytettäessä on mahdollista, että kyselymatriisi $\mathbf{Q}$ sekä avain- ja arvomatriisit $\mathbf{K}$ ja $\mathbf{V}$ tulevat eri syötteistä. Käännöksissä tämä voi tarkoittaa esi-



Kuva 4.2: Huomion laskeminen syötematriisista $\mathbf{X}$

merkiksi sitä, että $\mathbf{Q}$ on peräisin lähdekielisestä datasta sekä $\mathbf{K}$ ja $\mathbf{V}$ kohdekielisestä [27] tai niin, että data on multimodaalista eli syötteenä saadaan kuvaa ja tavoitteena tuottaa tekstiä [28]. Tätä kutsutaan termillä ristihuomio (*engl. cross-attention*).

Kuvassa 4.1 esitetään transformer-arkkitehtuurin rakenne alkuperäisen [22] artikkelin mukaan. Arkkitehtuuri on dekooderi-enkooderi-arkkitehtuuri, jossa enkooderi koostuu kuudesta kerroksesta, joissa kussakin kerroksessa on kaksi alikerrosta joista ensimmäinen monipäinen huomiokerros ja toinen on sijaintiriippuvainen (*engl. position-wise*) täysin myötäkytketty verkko, siten että jokaisen alikerroksen yhteydessä on myös jäännösyhteys, jota seuraa normalisointikerros siten että

$$\mathbf{Z} = LayerNorm[\mathbf{Y}(\mathbf{X}) + \mathbf{X}] \tag{4.12}$$

$$\tilde{\mathbf{X}} = LayerNorm[MLP(\mathbf{Z}) + Z], \tag{4.13}$$

jossa MLP on monikerroshavaitsin. Dekooderi-kerros vastaa muuten enkooderia, mutta siinä on lisänä vielä kolmas alikerros, jonka syötteenä on enkooderikerroksen ulostulo jolle tehdään monipäähuomion laskeminen. Tämä monipäinen huomiokerros on muokattu peitoksilla (*mask*) niin, että sijainnit eivät muutu, jolloin voidaan varmistua että ennusteet riippuvat ainoastaan tunnetuista sijainneista [22].

Koska transformer-arkkitehtuurissa ei mallinneta syötteiden toistuvuussuhteita, syötevektorin sijainnit on koodattava jotenkin malliin mukaan. Tämän vuoksi mal-

lissa on mukana sekä enkooderissa että dekooderissa sijainnin enkoodaus (kuvassa 4.1 *Positional encoding*) joka huolehtii mallissa siitä, että malli huomioi syötteenä saatavien tokenien järjestyksen, siten että sijainnit koodataan mukaan malliin mukaan jaksollisilla siniaalloilla, suhteessa sijaintiin *pos* sekä kulloiseenkin dimensioon i, siten että

$$PE_{(pos,2i)} = sin(pos/10000^{(2i/d_{\text{model}})})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{(2i/d_{\text{model}})}),$$

jossa d_{model} on upotusvektorin (*engl. embeddings*) ulottuvuus.

Ohjelmalistaus 2 Skaalattu itsehuomio[11]

Syöte: Joukko tokeneita $\mathbf{X} \in \mathbb{R}^{N \times D}$,Painomatriisit $\{\mathbf{W}^{(q)}, \mathbf{W}^{(k)}\} \in \mathbb{R}^{d \times d_k}$ ja $\mathbf{W}^{v} \in \mathbb{R}^{D \times D_v}$

Tulos: Attention($\mathbf{Q}$, $\mathbf{K}$,$\mathbf{V}$)

1: $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$

2: $\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$

3: $\mathbf{K} = \mathbf{X}\mathbf{W}^{(v)}$

4: **return** Attention($\mathbf{Q}$, $\mathbf{K}$,$\mathbf{V}$) $= Softmax[\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}}]\mathbf{V}$

Ohjelmalistaus 3 Monipäinen itsehuomio[11]

Syöte: Joukko tokeneita $\mathbf{X} \in \mathbb{R}^{N \times D}$,Kyselyiden painomatriisit $\{\mathbf{W}_1^{(q)} \dots \mathbf{W}_h^{(q)}\}$, Avainten painomatriisit $\{\mathbf{W}_1^{(k)} \dots \mathbf{W}_H^{(k)}\}$, Arvojen painomatriisit $\{\mathbf{W}_1^{(v)} \dots \mathbf{W}_H^{(h)}\}$, Ulostulon painomatriisit $\{\mathbf{W}_1^{(o)} \dots \mathbf{W}_H^{(o)}\}$

Tulos: $\mathbf{Y} \in \mathbb{R}^{N \times D}$

1: **for** $h = 1 \dots H$ **do**
2: $\quad \mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(q)}$
3: $\quad \mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(k)}$
4: $\quad \mathbf{V}_h = \mathbf{X}\mathbf{W}_h^{(v)}$
5: $\quad \mathbf{H}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \text{Softmax}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^T}{\sqrt{D_k}}\right)\mathbf{V}_h$
6: **end for**
7: $\mathbf{H} = Concat(\mathbf{H}_1, \dots, \mathbf{H}_N)$
8: **return** $\mathbf{Y}(\mathbf{X}) = \mathbf{H}\mathbf{W}^{(o)}$

4.3 Transformereiden taksonomia

Edellisessä osioissa käsitelty transformerin perusrakennetta ja kerrokset ovat tärkeitä rakennuspaloja myös muille transformer-malleille. Lin et al. [29] esittää transformereille taksonomian, jonka avulla niitä voidaan jaotella. Moduulitasolla tarkoitetaan sellaisia variantteja, joissa yhden moduulin sisällä on tavallisten huomiomekanismien ja täysin kytkettyjen kerroksien sijaan jotain muuta tai sijaintikoodaus toteutetaan eri tavalla. Arkkitehtuuritason muutokset tarkoittavat sitä, että yhden transformerverkon lohkoja käsitellään eri tavalla kuin tavallisessa transformerissa esimerkiksi niin, että lohkojen välillä on paluuyhteyksiä tai kevyempiä rakenteita. Esikoulutettujen mallien kategoria sisältää esikoulutetut kooderit ja dekooderit kuten erilaiset GPT- ja BERT-mallit [30]. Transformerin sovellusalan perusteella jaetut mallit liittyvät joko tekstiin, kuviin tai vaikkapa ääneen ja niiden käsittelyyn.

Tässä työssä olemme kiinnostuneita tavallisten transformer-mallien lisäksi vain kuvamuuntimista sekä erityisesti semanttiseen segmentointiin kehitetyistä malleista, joita käsitellään myöhemmin.

4.4 Kuvamuuntimet

Alkuperäinen [22] transformer-arkkitehtuuri on kehitetty luonnollisen kielen käsittelyä varten, mutta vastaavan idean pohjalta on rakennettu myös malleja kuvien käsittelyyn. Tässä yhteydessä käsitellään yksinkertainen VISION TRANSFORMER ViT-malli [31], sekä kehittyneempiä malleja kuten DeIT [32], BEiT [33] sekä SWIN TRANSFORMERS [34].

Yksinkertainen, alkuperäinen kuvamuunnin perustuu ideaan, jossa kaksiulotteinen kokoa $\mathbf{x}_p \in \mathbb{R}^{H\times W\times C}$ oleva kuva muutetaan yksiulotteisiksi litistetyiksi 2d-vektoreiksi $\mathbf{x}_p \in \mathbb{R}^{N\times(P^2\cdot C)}$ siten, että (H, W) on alkuperäisen kuvan resoluutio, C on kanavien määrä ja $N = HW/P^2$ on transformer-malliin menevän syötteen merkityksellinen koko, mutta malli itsessään käyttää joka kerroksessa latenttia vektoria, jonka pituus on D joten syöte on kuvattava tämän pituisena.

Kuvamuuntimen mallissa käytetään samankaltaista ideaa kuin luonnollisen kielen käsittelyyn kehitetyssä BERT-mallissa [30], [31], jossa käytetään erityistä `[CLS]`-tokenia merkitsemään malliin tulevat syötteet. Mallissa tämä esitetään matemaattisesti muodossa $\mathbf{z}_0^0 = \mathbf{x}_{\text{class}}$, joka on mallin viimeisellä kerroksella. Kuvien kaksiuloitteisesta (tai kolmiulotteisesta, jos värikanavat otetaan huomioon) luonteesta huolimatta sijainnin koodaus ViT-kuvamuuntimessa tapahtuu yksiulotteisella sijantikoodauksella, koska kaksiulotteinen ei tuo malliin lisää hyötyjä.

Mallin enkoodauskerros sisältää [22] mukaisesti vaihdellen monipäisen itsehuomion (MSA, *engl. Multihead Self-Attention)* sekä MLP-osion. Ennen jokaista kerrosta on kerroksen normalisointi (*eng. layer normalization*) ja joka kerroksen jälkeen jäännösyhteys. Kerroksen normalisointi tarkoittaa sitä, että syötteet normalisoidaan

kerroksen aktivointifunktion suorittamisen jälkeen, ennen kuin tulos annetaan syötteenä seuraavalle kerrokselle [35].

Matemaattisesti kuvamuunninneuroverkko voidaan esittää neljällä kerroksella siten, että

$$\mathbf{x}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \ldots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{\text{pos}} \tag{4.14}$$

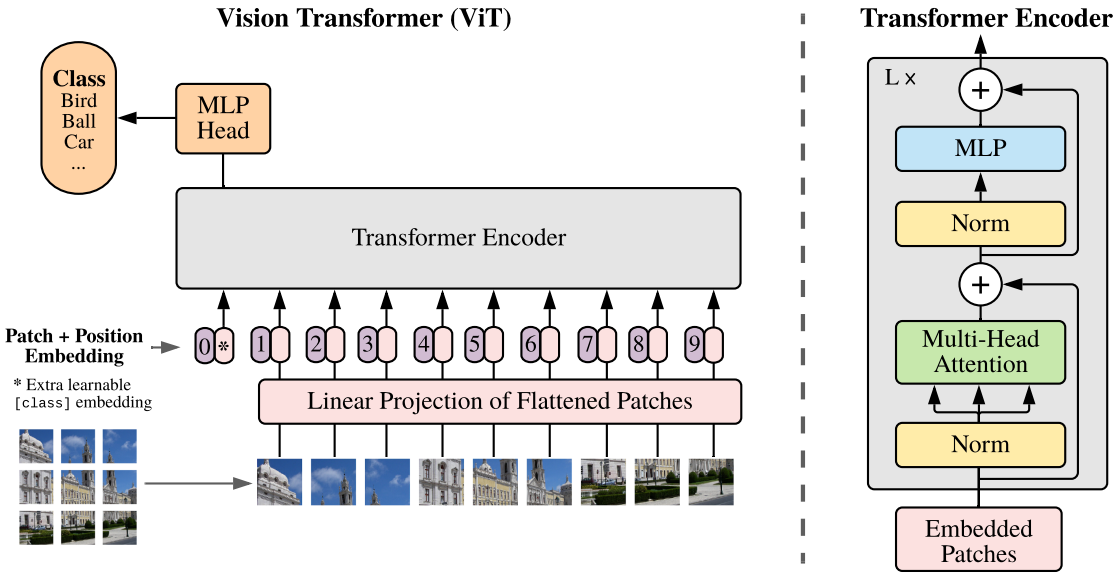
$$\mathbf{z'}_l = MSA(LN(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \tag{4.15}$$

$$\mathbf{z}_l = MLP(LN(\mathbf{z'}_l)) + \mathbf{z'}_l \tag{4.16}$$

$$\mathbf{y} = LN(\mathbf{z}_\mathbf{L}^\mathbf{0}), \tag{4.17}$$

missä $\mathbf{E} \in \mathbb{R}^{P^2 \cdot C) \times D}$, $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$ ja $l = 1 \ldots L$. ViT-mallin rakenne on visualisoitu kuvassa 4.3.

Dosovitskiyn et al. [31] esittämä alkuperäinen kuvamuunnin ei toimi hyvin, jos sen kouluttaa riittämättömällä määrällä dataa. Touvron et al. [32] ovat luoneet alkuperäisen kuvamuuntiminen pohjalta DEiT-mallin joka perustuu siihen, että mallin opettaminen perustuu tietämyksen siirtoon (*engl. knowledge distillation*) siten, että mallissa on `CLS`-tokenin lisäksi erityinen siirtotoken (*engl. distillation token*), jonka tarkoituksena on mahdollistaa se, että verkko oppii opettajaverkon ulostulokerrokselta kuitenkin täydentäen verkon luokkavektoria. DEiT-mallissa isehuomiokerroksen päällä on eteenpäinkytketty verkko.



Kuva 4.3: ViT-mallin rakenne. Kuvan lähde: [31]

5 Transformerit kuvien luokittelussa ja tunnistamisessa

Luvussa 4.4 esiteltiin kuvamuuntimet eli transformer-arkkitehtuurit, jotka on kehitetty kuvadatan käsittelyyn. Tässä luvussa perehdytään kuvien luokitteluun sekä tunnistamiseen sekä käydään läpi transformer-arkkitehtuureiden erityispiirteitä näiden ongelmien ratkaisemiseen.

5.1 Kuvien luokittelu ja kohteen tunnistus

Kuvien luokittelulla (*eng. Image classification)* tarkoitetaan ongelmaa, jossa joukko kuvia luokitellaan kategorioihin sen perusteella mitä kuvat esittävät. Johdettuna luvun 2 koneoppimisen määritelmästä tämä tarkoittaa sitä, että on olemassa joukko $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i)$ kuvia $\mathbf{x}_i$ sekä niihin liittyvä luokka $y_i \in \mathcal{C}$, jossa on $\mathcal{C}$ on joukko mahdollisia kategorioita kuvajoukossa. Kuvan $\mathbf{x_i}$ luokittelu on siis tehtävä T, jossa estimoidaan funktiota $f : \mathbb{R}^{H \times W \times K} \rightarrow \{0, 1\}^C$.

Joissain tapauksisssa yksittäisen luokan sijaan on tarpeen tunnistaa kuvista niissä esiintyviä kohteita kuten vaikkapa autoja, kasvoja tai eläimiä. Tätä kutsutaan kohteiden tunnistamiseksi (*eng. object detection*). Kohteiden tunnistamisen koneop-

pimistehtävä on oppia funktio

$$f_\theta : \mathbb{R}^{H\times W\times K} \rightarrow \underbrace{[0,1]^{A\times A}}_{p_{ij}\in[0,1]} \times \underbrace{\{1,\ldots,C\}^{A\times A}}_{y_i ij\in\{1,\ldots,C\}} \times \underbrace{(\mathbb{R}^4)^{A\times A}}_{\delta_{ij}\in\mathbb{R}^4}, \tag{5.1}$$

jossa A on ankkurialueiden lukumäärä, C mahdollisten luokkien lukumäärä. Ankkurilaatikolla tarkoitetaan aluetta, jolla jokin kohde voi kuvassa olla eli ne ovat ehdotuksia sille missä tietty kohde voisi sijaita. Jokaisella kandidaattilaatikolle lasketaan todennäköisyys $p_{ij} \in [0,1]$, kategoria $y_{ij} \in \{1,\ldots,C\}$ sekä kaksi kaksiulotteista leikkausvektoria $\delta_{ij} \in \mathbb{R}^4$, jotka asetetaan kohteen keskiosan päälle siten että saadaan kohteen ylävasen ja alaoikea kulma. [17]

Konvoluutioneuroverkkoihin perustuva luokittelu ja tunnistus perustuu hyvin syviin neuroverkkoihin, jotka on koottu luvussa 3.6 esitetyistä konvoluutioneuroverkon blokeista. Tunnettuja malleja on esimerkiksi VGG [36] , GoogLeNet [37] sekä AlexNet [38] [6], [17].

5.2 Transformer-pohjainen kuvien luokittelu ja kohteen tunnistus

Osiossa 4.4 esiteltiin transformer-arkkitehtuuriin perustuvien kuvamuuntimien perusidea ViT eli Vision transformer-mallin pohjalta sekä käsiteltiin haasteita sekä miten kuvamuuntimia saadaan paremmiksi esimerkiksi tietämyksen siirron avulla, kuten DeIT-mallissa. Sekä ViT ja DeIT on erityisesti kuvien luokitteluun soveltuvia malleja. Tässä osiossa käsittelen näiden lisäksi monimutkaisempia malleja, jotka soveltuvat sekä kuvien luokitteluun että kohteen tunnistamiseen kuvasta, koska monet mallit soveltuvat molempiin tehtäviin. Vaativampi, semanttisen segmentoinnin tehtävä esitellään luvussa 7. Taulukossa 5.1 esitetään tässä työssä käsittellyt kuvamuuntimet sekä tehtävät, joihin alkuperäisistä artikkeleista löytyy maininta. Näistä

malleista Swin Transformer on monipuolisin ja pystyy sekä luokitteluun, tunnistamiseen että semanttiseen segmentointiin. Tähän työhön valittiin eri arkkitehtuureita tekemällä haku Scopus-tietokantaan *"image classification"AND transformers*.

BEiT [33] eli *Bidirectional Encoder representation from Image Transformers* on luonnollisen kielen käsittelyyn luodun BERT-mallin idean pohjalta kehitetty malli, joka perustuu peitettyjen kuvien mallintamiseen (*engl. Masked Image Modeling*), jossa kuvat jaetaan kahteen erilaiseen syötteeseen mallille: sekä peitteitä, että tavallisia syötteitä siten, että C-kanavainen ja (H, W)-kokoinen kuva $x \in \mathbb{R}^{H \times W \times C}$ jaetaan $N = HW/P^2$ osaan, siten että yksittäinen osa $\mathbf{x}^p \in \mathbb{R}^{N \times (P^2 C)}$, jossa P on yksittäisen erän resoluutio. Kuvaosiot $\{\mathbf{x}^p\}_{i=1}^N$ litistetään vektoreiksi joille tehdään lineaarinen projektio. BEiT-malli soveltuu sekä kuvien luokitteluun että semanttiseen segmentointiin.

Swin Transformers eli *Shifted Window Transformers* on kuvantunnistustehtäviä varten kehitetty transformer-arkkitehtuuri, jonka tavoitteena on toimia tehokkaana runkona erilaisille tehtäville. Mallissa kuvat jaetaan epälimittäisiin palasiin siten, että tavoitteena on hiearkinen esitystapa kuvan ominaisuuksille jossa kuvien koko pienenee mitä syvemmälle verkossa edetään, samaan tapaan kuin esimerkiksi konvoluutioverkoissa VGG tai ResNet. Mallin päärakennuspalanen on niin kutsuttu *Swin Transformer Block*, joka perustuu tavalliseen monipäiseen huomioon (esitetty kaavassa 4.11 ja algoritmissa 3), jota seuraa MLP-kerros sekä kerroksen normalisointi (LN). Itsehuomio lasketaan limittäisille lohkoille, siten että normaalinen huomiomekanismin neliöllisen laskennallisen vaatimuksen sijaan saadaan laskennallisesti tehokkaampi tapa laskea. Ikkunan siirtojen vuoksi osa yhteyksistä kuvien välillä voi hävitä, jolloin mallissa hyödynnetään yhteyksiä eri kerroksien välillä. Swin Transformers mallia voidaan käyttää niin kuvantunnistamisessa, luokittelussa kuin semanttisessa segmentoinnissakin [34].

Taulukkossa 5.2 on esitetty tässä työssä kuvailtujen mallien raportoidut tark-

	Tehtävä			**Tutkimuksen tiedot**	
Malli	**Luokittelu**	**Tunnistus**	**Segmentointi**	**Vuosi**	**Viittaukset**
ViT	X			2021	7883
DeIT	X			2021	3295
BeIT	X		X	2022	432
Swin	X	X	X	2021	12617

Taulukko 5.1: Transformer-mallien tehtävät sekä julkaisuvuodet ja viittaukset (viitattu 1.11.2024)

Malli	**Tarkkuus**
ViT-H/14	88,55%
DeIT-B 384	85,20%
BEiT-L	88,60%
Swin-L	87,30%
SwinV2-G	90%
EfficientNet	88,4%

Taulukko 5.2: Transformer-luokittimien sekä EfficientNet-mallin luokittelutarkkuudet

kuudet ImageNet aineiston perusteella sekä NoisyStudent eli erään EfficientNet-mallin variaation tarkkuus. Monissa papereissa (esim. [31]) mallin tarkkuutta on verrattu EfficientNet-mallin NoisyStudent-versioon, joten se valittiin mukaan tähän vertailuun.

6 Transformerit semanttisessa segmentoinnissa

6.1 Semanttinen segmentointi

Kuvien segmentoinnilla tarkoitetaan kuvien jakamista erilaisiin osiin esimerkiksi kuvan osien, kuten reunojen tai eri värien perusteella. Semanttisella segmentoinnilla tarkoitetaan tunnistustehtävää, jossa kuvassa esiintyvät objektit pyritään samankaltaisesti tunnistamaan että rajaamaan. Koneoppimistehtävänä tämä tarkoittaa sitä, että kuvan jokainen pikseli pyritään luokittelemaan johonkin luokkaan. Semanttisella segmentoinnilla on hyvin monenlaisia sovellusalueita. Lääketieteessä semanttista segmentointia voidaan käyttää vaikkapa patologisten kuvien analysointiin, joihin esimerkiksi U-Net-malli [39] on alunperin kehitetty. Semanttista segmentointia voi hyödyntää solukuvien lisäksi myös tietokonetomografialla, röntgenillä tai ultraäänellä saatujen kuvien tunnistamiseen jolloin menetelmää voi käyttää vaikkapa erilaisten kasvaimien tunnistamiseen liittyvät ongelmat. Esimerkiksi myöhemmin esiteltävä U-Net-arkkitehtuuri on alunperin kehitetty biolääketieteellisten kuvien analysointiin. Toinen sovelluskohde kuvien segmentoinnille on kaukokartoitus, jossa aineistona käytetään esimerkiksi satelliitilla otettuja kuvia. Tämän sovelluskohteita voi olla esimerkiksi luontoon liittyvien muutoksien tarkastelu tai kuvissa näkyvien objektien luokittelu. [13], [40], [41]. Kuvassa 6.1 esitetään esimerkki SegFormer-mallin

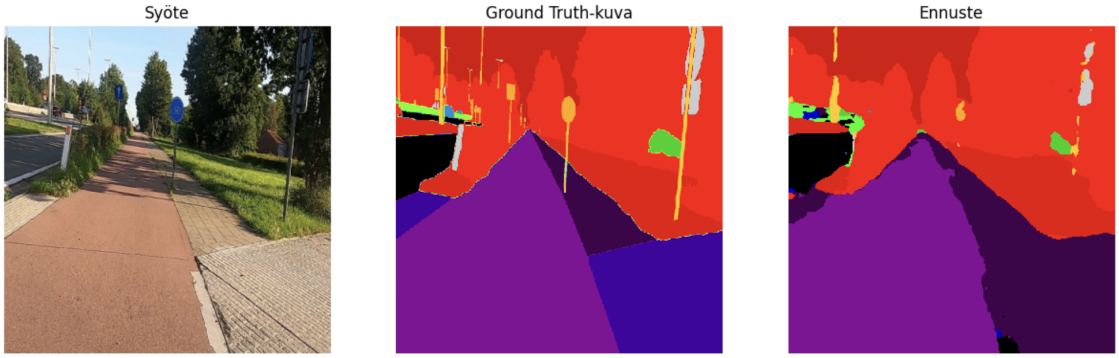
saamasta syötteestä, oikeasta segmentoinnista sekä mallin tekemästä ennusteesta.

Ensimmäiset algoritmit segmentointiin perustuivat Bayesialaiseen analyysiin sekä siihen, että kuvat analysoitiin eri värien näkökulmasta ja pyrittiin löytämään rajat. Konvoluutioneuroverkkojen myötä semanttinen segmentointi voidaan toteuttaa yhden neuroverkon avulla [13]. Transformer-arkkitehtuurin yleistymisen myötä myös siihen perustuvia verkkoja on kehitetty ja sovellettu.

Semanttisen segmentoinnin algoritmeja voidaan arvioida erilaisilla metriikoilla. Nopeus millisekunteina kertoo kuinka nopeasti malli pystyy tekemään päättelyn ja se raportoidaan tyypillisesti millisekunteina. Tarkkuus mitataan keskimääräisellä IoU-arvolla (*Intersection over Union*), joka voidaan määritellä kaavalla $MIoU = \frac{1}{k}\sum_{i=0}^{k}\frac{P_{ij}}{\sum_{j=0}^{k}P_{ii}+\sum_{j=0}^{k}P_{ji}-P_{ii}}$, jossa k on luokkien lukumäärä, P_{ij} on kategorian i pikselit luokiteltuna kategoriaan j, P_{ii} oon oikeat positiiviset ja P_{ij} ja P_{ji} ovat väärät positiiviset ja negatiiviset luokittelut. Tarkkuus voidaan laskea myös pikselitasolla, keskiarvona tai painottaen frekvenssejä [42]. Semanttisessa segmentoinnin malleissa voidaan käyttää hävikkifunktiona myös pikseleittäin laskettu ristientropiaa $CE_{loss}(p,q) = -\sum_{i=0}^{n} p_i log(q_i)$, jossa p_i on oikea pikselin luokka ja q_i mallista tullut ennuste. Tätä voidaan myös muokata painottamalla entropiaa painolla w_i. Biolääketieteellisessä kuvantamisessa käytetään niin kutsuttua *dice loss*-menetelmää, joka esitetään matemaattisesti muodossa

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}, \tag{6.1}$$

jossa lasketaan yhteen N ennustettua vokselia (pikselin vastike kolmiulotteisissa kuvissa) siten, että $p \in P$ on ennustettu arvo sja $g_i \in G$ on totuus (*engl. ground truth*) [43]. Lääketieteellisissä sovelluksissa käytetään pikselien sijaan vokseleita, koska monet sovellusalueet kuvaavat kudoksia kolmiulotteisesti.



Kuva 6.1: Esimerkki Segformer-mallin syötteestä ja tuloksesta

6.2 Konvoluutioverkkoihin perustuva semanttinen segmentointi

Yksinkertaisin tapa lähestyä semanttista segmentointia konvoluutioneuroverkkojen avulla olisi tehdä verkko, joka ottaa syötteenä haluttuun pikseliin keskitetty neliö kuvasta ja luokitella se oikeaan luokkaan. Tämänkaltaisen verkon käyttäminen vaatisi erittäin paljon laskentatehoa eikä olisi tarpeeksi tehokasta. Konvoluutioneuroverkoissa kuvan ominaisuuskarttojen koko pienenee verkon kerroksissa edetessä, jolloin verkon parametrien määrä pysyy tarkoituksenmukaisena ja kuvista saadaan tunnistettua isompia piirteitä [11].

Täysin konvolutionaaliset verkot (*engl. fully convolutional network, FCN*) ovat neuroverkkoja, joissa konvoluutioneuroverkoista poiketen puuttuu viimeisen täysin yhdistetty kerros ja se on korvattu konvoluutiokerroksella. Yleiset syvät neuroverkot laskevat yleisen, epälineaarisen funktion mutta verkot joissa on pelkästään konvoluutiokerroksia muodosta epälineaarisen filtterin jota kutsutaan yleisemmin täysin konvoluutionaaliseksi verkoksi [42].

Artikkelissa [42] esitetty FCN-tekniikkaan perustuva segmentointiarkkitehtuuri jota testattiin runkoinaan (*engl. backbone*) VGG-16, GoogLeNet ja AlexNet-verkkoja. Runkoverkkovaihtoehdot ovat hyviksi havaittuja konvoluutioverkkoja, joi-

den viimeinen kerros poistetaan ja muutetaan täysin yhdistetystä kerroksesta takaisin konvoluutiokerrokseksi. Lisäksi malliin lisättin 21-ulotteinen, 1×1 -kokoinen konvoluutio joka vastaa käytettävän PASCAL-datajoukon luokkien määrää, jonka jälkeen lisättiin dekonvoluutiokerroksia jotka kasvattavat (*up-sampling*) kuvaa vastaa pikselien määrää ulostulokerroksella. Artikkelin pohjalta parhaat tulokset saavutettiin VGG-rungolla.

Toinen yleisesti käytetty neuroverkkoarkkitehtuuri semanttiseen segmentointiin on U-Net, jossa syötedatan ulottuvuutta (H, W) pienennetään ja ja kasvatetaan symmetrisesti, siten että jokainen alentamiskerros (*down-sampling*) on liitetty vastaavaan kasvattamiskerrokseen [11].

Kolmas semanttiseen segmentointiin käytetty menetelmä on laajentava konvoluutio (*dilated convolution*), jossa konvoluutio kasvattaa vastaanottavaa kenttää (*receptive field*) pitäen kuitenkin parametrien määrän samana [44].

6.3 Transformer-arkkitehtuurin perustuva semanttinen segmentointi

Kappaleessa 4 esitettiin yleisesti transformer-arkkitehtuuri sekä muutamia tyypillisimpiä arkkitehtuureita konenäkötehtäviin. Pelkät transformer-mallit eivät sellaisenaan sovellu segmentointitehtäviin, koska niiden dekooderikomponentti ei sovellu siihen [45]. Semanttiseen segmentointiin on kehitetty transformer-pohjaisia arkkitehtuureita, jotka tehtävään soveltuvat. Työhön valitut semanttisen segmentoinnin muunntimet valittiin mukaan lähteestä [41], johon oli koottu erilaisia malleja sekä raportoitu ominaisuuksia. Valittuja malleja vertasin Scopus-tietokantaan tehtyyn hakuun *"semantic segmentation"AND transformers*, joka rajattiin koskemaan vain englanniksi kirjoitetttuja artikkeleita, jotka on julkaistu vuoden 2017 jälkeen. Tavoitteena oli valita mukaan sellaiset mallit, joista löytyi IoU-arvot yleisimmistä data-

	Datajoukko				
Malli	**Ade20k**	**Cityscapes**	**Pascal VOC**	**Viittauksia**	**Vuosi**
Swin	X	-	-	12617	2021
SETR	X	X	-	2135	2021
Segmenter	X	X	X	987	2021
SegFormer	X	X	X	2641	2021

Taulukko 6.1: Valitut segmentointimuuntimet ja aineistot, joilla mallit on artikkeleissa testattu (viitattu 1.11.2024)

joukoista sekä viittauksia. Taulukossa 6.1 on esitetty mallit viittausmäärineen sekä datajoukot, joilla niitä on testattu.

Tähän työhön olen valinnut esiteltäväksi kolme semanttista segmentointia varten suunniteltua kuvamuunninta, jotka ovat SETR eli *Segmentation Transformer*, SEGMENTER sekä SEGFORMER. SWIN TRANSFORMERS esiteltiin jo kuvamuuntimien yhteydessä. On olemassa myös monimutkaisempia ratkaisuja, mutta ne jätetään tässä yhteydessä huomioimatta.

SEGMENTATION TRANSFORMER eli SETR-malli on enkooderi-dekooderipohjainen transformer-malli kuvien segmentointiin, joka ottaa syötteen kuvan $x \in \mathbb{R}^{H \times W \times 3}$, joka alennetaan ominaisuuskartaksi jolle $x_f \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times 3}$. Tämän perusteella saadaan ominaisuussyöte Z pituudella $L = \frac{H}{16} \times \frac{W}{16} = \frac{HW}{256}$, jolloin transformerin ulostulo voidaan muuntaa samankokoiseksi. Jotta saadaan aikaiseksi $\frac{HW}{256}$ kokoinen syöte, kuva pitää jakaa $\frac{H}{16} \times \frac{W}{16}$-kokoisiksi palasiksi paloiksi p. Jokaiselle palaselle p tehdään lineaarinen kuvause $f \to e \in \mathbb{R}^C$ kuvaus, jolle enkoodataan palasen spatiaalinen informaatio käyttäen upotusvektoria p_i, jolloin lopullisen syötteen muoto on $E = e_1 + p_1, \ldots, e_L + p_L$. Mallin toiminta perustuu luvussa 4 esitettyjen muuntimien perustoimintaan. Jokainen syöte E annetaan enkooderille, joka pyrkii oppimaan kuvan piirteitä siten, että jokaisella transformer-kerroksella on oma havaitseva kenttä. Enkooderikerros koostuu L_e-kerroksesta MSA- ja MLP-kerroksia, joista jokaiselle kerrokselle annetaan syötteenä kolmikko $(query, key, value)$ siten, että $query = Z^{l-1}\mathbf{W}_q$, $key = Z^{l-1}\mathbf{W}_k$ ja $value = Z^{l-1}\mathbf{W}_v$, jossa $\mathbf{W}_q/\mathbf{W}_k/\mathbf{W}_v$ ovat

opittavia parametreja. SETR-muuntimen itsehuomio ja monipäinen huomio vastaavat normaalien muuntimien kerroksia, mutta ne ovat kerrottu parametrilla Z^{l-1}. SETR-mallista on olemssa kolme eri varianttia: *naive upsampling (naive)*, *progressive upsampling (pup)* sekä *multi-level feature aggregation (MLA)*. NAIVE-variantin dekooderi kuvaa ominaisuudet Z^L luokkien lukumäärää vastaavaan ulottuvuuteen käyttäen yksinkertaista kaksikerroksista verkkoa, jossa 1×1-konvoluutio sekä normalisointi RELU-aktivoinnilla sekä toinen 1×1-konvoluutio, jonka jälkeen syöte kasvatetaan alkuperäisen kuvan kokoon ja luokittelu tehdään käyttäen pikselikohtaista ristientropiaa. *Progressive upsampling*-variantissa vaihdellaan konvoluutio-operaatiota sekä kasvattamista siten, että kasvattaminen rajoitetaan kuitenkin vain kaksinkertaistamiseeksi. Tämän johdosta tarvitaan neljä operataatiota koosta $Z^l = \frac{H}{16} \times \frac{W}{16}$.

Kolmas variantti on nimeltään *Multi-level feature Aggregation* eli MLA, jossa syötteenä saadaan ominaisuuksien esitykset $Z^m m \in \{\frac{L_e}{M}, 2\frac{L_e}{M}, \ldots, M\frac{L_e}{M}\}$ M:stä eri tasaisesti jakautuneesta kerroksesta niin, että saadaan eri ryhmää jotka ovat erikoistuneet eri kerroksiin. Joka kerroksen ominaisuudet Z^l muutetaan koosta $\frac{HW}{256} \times C$ kokoon $\frac{H}{16} \times \frac{W}{16} \times C$. Tämän jälkeen kolmikerroksista verkkoa (1×1, 3×3 ja 3×3) sovelletaan ensimmäiseen ja kolmanteen kerrokseen ja resoluutio kasvatetaan nelinkertaiseksi kolmannen kerroksen jälkeen. Eri kerroksien välisiä suhteita parannetaan elementtitason yhdistämisellä ensimämisen kerroksen jälkeen, jonka jälkeen kerrokseen sovelletaan 3×3-konvoluutiota ja lopulta kaikki kerrokset yhdistetään kanavittain ja kasvatetaan nelinkertaiseksi jotta saadaan täysi resoluutio. [46].

SEGMENTER-mallissa [47] kuva $x \in \mathbb{R}^{H\times W\times C}$, jossa $H \times W$ on kuvan koko ja C kanavien määrä, jaetaan syötteeksi $\mathbf{x} = [x_1, \ldots, x_n] \in \mathbb{R}^{N\times P^2\times C}$, jossa P on osan koko samaan tapaan kuin alkuperäisissä kuvamuuntimissa, DEiT tai BEiT -malleissa. Syöte litistetään yksiulotteiseksi vektoriksi ja sille tehdään lineaarikuvaus sekä sijainnin enkoodaus, joka lisätään syötteeseen alkioksi $\mathbf{x}_0$. Transformerin enkooderikerros koostuu L-kerroksesta, joiden pohjalta tehdään kontekstuaalinen enkoodaus

$\mathbf{z}_L \in \mathbb{R}^{N \times D}$. Jokainen kerros sisältää monipäisen itsehuomion (MSA) jota seuraa MLP-kerros normalisoituna sekä jäännösyhteys, siten että

$$\mathbf{a}_{i-1} = MSA(LN(\mathbf{z}_{i-1})) + \mathbf{z}_{i-1} \tag{6.2}$$

$$\mathbf{z}_i = MLP(LN(\mathbf{a}_{i-1})) + \mathbf{a}_{i-1}. \tag{6.3}$$

Mallissa MSA lasketaan 4 esitetyllä monipäisen itsehuomion kaavalla ilman poikkeuksia. Mallin dekooderi muuttaa sekvenssin koodattuja arvoja $\mathbf{z_L} \in \mathbb{R}^{N \times D}$ segmentointikartaksi $s \in \mathbb{R}^{H \times W \times K}$, missä K on luokkien lukumäärä. Dekooderi oppii kuvaamaan enkooderilta tulevat enkoodaukset patchi-tason luokkien pisteiksi, jotka kasvatetaan bilineaarisella interpolaatiolla pikselitason arvoiksi.

SEGFORMER [48] on semanttista segmentointia varten kehitetty malli,, joka yhdistää ominaisuuksia sekä transformer-malleista että MLP-malleista. Mallin enkooderi perustuu transformereihin ja sen tehtävänä on generoida kuvasta sekä hienotasoiset matalan resoluution, että karkeat suuren resoluution ominaisuudet. Mallin dekooderi on MLP-malli, joka yhdistää nämä ominaisuudet lopulliseksi segmentointimaskiksi. Toisin kuin alkuperäisessä [31] ViT-mallissa, Segformerissa käytetään 4×4 kokoisia palasia mallin syötteenä, koska ne soveltuvat paremmin tiheän ennustamisen tehtävään jollainen semanttinen segmentointi on. Mallin enkooderi saa syötteenä $H \times W \times 3$-kokoisia kuvia, jotka lohkotaan $\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C_i$-kokoisiksi hierarkisiksi ominaisuuskartoiksi, joissa $i \in \{1, 2, 3, 4\}$ ja $C_{i+1} > C_i$. Mallissa hyödynnetään tavallista itsehuomiokerrosta, mutta sitä tehostetaan jotta se olisi laskennallisesti nopeampi siten, että syötteenä annettavan sekvenssin kokoa muutetaan. Viimeinen komponentti enkooderissa on Mix-FFN-kerros, joka saa syötteenä ominaisuudet $\mathbf{x}_{in}$ itsehuomiolta ja määritellään $\mathbf{x}_{out} = MLP(GELU(Conv_{3\times3}(MLP(\mathbf{x}_{in})))) + \mathbf{x}_{in}$. Mallin kehittäjien mukaan 3×3-konvoluution avulla voidaan ylläpitää sijaintitietoa mallissa. Mallin dekooderi on kevyt MLP-verkko, joka määritellään nelikerroksisene-

likerroksisena siten että ensimmäisen kerroksen tehtävänä yhtenäistää ominaisuuksien F_i ulottuvuudet, toinen kerros kasvattaa ja yhdistää ominaisuudet, kolmas kerros sulattaa (*engl. fuse*) yhdistetyt ominaisuudet ja viimeinen kerros luo lopullisen segmentointimaskin. Matemaattisesti tämä esitetään niin, että

$$\hat{F}_i = Linear(C_i, C)(F), \forall i \tag{6.4}$$

$$\hat{F}_i = Upsample(\frac{W}{4} \times \frac{W}{4})(\hat{F}_i), \forall i \tag{6.5}$$

$$F = Linear(4C, C)(Concat(\hat{F}_i), \forall i \tag{6.6}$$

$$M = Linear(C, N_{cls})(F), \tag{6.7}$$

jossa $Linear(C_i, C)$ tarkoittaa neuroverkon lineaarista kerrosta ja $Upsample$ viittaa kasvattamiskerrokseen, jolla mallissa kulkevaa dataa kasvatetaan systemaattisesti vastaamaan halutun kuvan tarkkuutta kuten vastaavissa konvoluutioverkkopohjaisissa malleissa [11]. Esiteltyjen mallien perusteella voidaan huomata, että monet transformer-pohjaiset semanttisen segmentoinnin mallit ovat perustoiminnaltaan hyvin samankaltaisia, johtuen siitä että jokainen perustuu lähtökohtaisesti samaan huomiomekanismiin. Erot malleissa sekä niiden toteutuksissa liittyvät syötteisiin sekä siihen malleissa hyödynnetään opittavia parametreja tai muita neuroverkkokomponentteja kuten MLP:ta tai kerrosnormeja (LN). Lisäksi syötteenä käytettävien kuvien lohkomisessa on jonkin verran eroja mallien välillä.

6.4 Menetelmien erojen tarkastelu

Edellisessä osiossa esiteltiin kaksi täysin konvoluutionaalisiin verkkoihin ja neljä transformer-arkkitehtuuriin perustuvaa semanttisen segmentoinnin algoritmia. Semanttisen segmentoinnin tehtävässä algoritmeja vertaillaan tyypillisesti mIoU-arvolla käyttäen tunnettuja datajoukkoja PASCAL VOC, Ade20k ja Cityscapes. PASCAL

VOC on objektien luokitteluun tehty datajoukko, joka on kehitetty kuvantunnistusalgoritmien paremmuuden selvittämiseen niin luokittelussa, tunnistamisessa kuin segmentoinnissakin kuvantunnistushaasteessa [49]. Ade20k-datajoukko on kehitetty erityisesti kuvien semanttista ymmärtämistä varten ja sisältää kuvista luokittelut siitä, mihin luokkaan kuvan elementit kuuluvat [50]. Cityscapes-datajoukko on erityisesti katukuviin keskittyvä joukko, jossa annotoidut kuvat ovat on otettu eri kaupungeista Euroopassa [51]. Taulukossa 6.2 on esitetty tässä vertailussa käytettyjen aineistojen koot lähteen [50] mukaan.

Datajoukko	Kuvia	Luokkia
PASCAL VOC	10103	540
Ade20k	22210	2639
Cityscapes	25000	30

Taulukko 6.2: Segmentointidatajoukkojen koot

Taulukossa 6.3 kuvataan tässä työssä esiteltyjen mallien tuloksia näillä datajoukoilla. Näiden tuloksien perusteella transformer-pohjaiset mallit pärjäävät tasaväkisesti FCN-mallien kanssa.

Mallien julkaisuissa esitetään usein ablaatiotutkimus (*engl. ablation study*), joka tarkoittaa sitä että mallista poistetaan tai muutetaan ominaisuuksia systemaattisesti ja tarkastellaan mallia näiden muutoksien jälkeen. Semanttisen segmentoinnin menetelmissä voidaan vaihdella esimerkiksi runkoa tai mallin parametrien määrää.

Algoritmisessa kompleksisuudessa transformer-arkkitehtuuriin perustuvat mallit eroavat jonkin verran konvoluutioneuroverkoista. Transformer-kerroksen kompleksisuus on $\mathcal{O}(n^2 \cdot d)$ ja konvoluutiokerroksen $\mathcal{O}(k \cdot n \cdot d^2)$, jossa n on syötteen pituus, d kerroksen ulostulon ulottuvuus ja k konvoluutioverkon ydinfunktion koko [22]. Transformer-kerroksen kompleksisuus on neliöllinen suhteessa syötevektorin kokoon ja lineaarinen ulostulokerroksen kokoon nähden, kun taas konvoluutiokerros on lineaarinen konvoluutiokerroksen ydinfunktion sekä syötteen koon suhteen, mutta neliöllinen ulostulon suhteen.

	mIoU			
	ADE20K	Cityscapes	Pascal VOC	Lähde
FCN-VGG16	29.39	-	56	[42]
FCN-AlexNet	-	-	39.8	[42]
FCN-GoogLeNet	-	-	42.5	[42]
SETR	50.28	82.15	55.83	[46]
Segmenter	52.63	-	59	[47]
Segformer	51.8	84	-	[48]
Swin-L	53.5	-	-	[34]

Taulukko 6.3: Työssä esiteltyjen mallien Intersection over Union arvot

7 Mallien empiirinen testaus

Edellä esiteltiin kirjallisuuden pohjalta kuvamuuntimien käyttöä sekä kuvien luokittelussa että semanttisessa segmentoinnissa, sekä käytiin läpi esitettyjä tuloksia. Tässä luvussa esitetään käytännön kokeen tulokset siitä, miten hyvin mallit toimivat hienosäädetyllä esikoulutetulla verkolla.

7.1 Valitut mallit ja aineisto

Tässä tutkielmassa testataan sekä transformer- että konvoluutioverkkoihin perustuvia malleja luokitteluongelmalla käyttäen tunnettua datajoukkoa sekä esikoulutettuja malleja.

Valitut mallit olivat ViT, Swin Transformer sekä EfficientNet B0, jotka ovat kirjallisuudessa esitellyiltä tuloksiltaan hyviä. Vertailuun valittu ongelma oli kuvien luokittelu eri luokkiin. Kuvien luokittelu valittiin vertailun tehtäväksi siksi, että se on konenäön tehtävistä yksinkertaisin. Aineistona testissä käytetään Cifar100-aineistoa [52]. Cifar-100 ainestossa kuvat on luokiteltu sataan erilaiseen luokkaan.

7.2 Mallien koulutus

Mallien koulutus tehtiin hyödyntämällä kappalessa 3.4 esitettyä hienosäätöä kuvan 7.1 prosessilla. Aluksi otetaan käyttöön esikoulutettu malli, jossa on valmiiksi lasket-

tuna painot tietylle aineistolle, tässä tapauksessa ImageNet-aineistolle. Muut kuin mallin päällikerros jäähdytetään (*engl. freeze*) siten, että sen painot eivät voi muuttua koulutuksen aikana vaan ainoastaan päällimmäisen kerroksen painot lasketaan uudestaan. Näin saadaan laskettua ennustetukset. Päällikerroksen kouluttamisen jälkeen koko malli voidaan kouluttaa uudelleen, mutta tässä työssä tämä jätettiin tekemättä. Valmiina malleina käytettiin kirjastoista saatavia valmiita malleja.

7.2.1 Testien tekninen toteutus

Mallien toteutus toteutettiin kahdella erillisellä toteutuksella, joista ensimmäinen keskittyi transformer-mallien käyttöön ja toinen konvoluutioneuroverkkojen käyttöön. Datan käsittely toteutettiin luokalla, joka haki halutun aineiston Huggingface-palvelun datajoukoista ja muutti sen testissä vaadittuun muotoon, riippuen mallista ja sen vaatimasta muodosta riippuen. Datan käsittelyyn tarkoitettu luokka mahdollisti myös muunnokset kuville, joita tarvittiin konvoluutioverkon toteutuksessa.

Sekä ViT että Swin Transformer-mallien hienosäätö toteutettiin Huggingface Transformers-kirjaston [53] avulla, mutta mallit olisivat olleet saatavilla esikoulutetuilla painoilla myös esimerkiksi torchvision- tai Keras CV-kirjastoista. Huggingface Transformers-kirjasto tarjoaa kuvien esiprosessointiin valmiit tiedot, jolloin mallin hienosäädössä voidaan käyttää samoja parametrejä kuin alkuperäisen mallin kouluttamisessa. Jokainen kuva muutettiin kokoon $(224, 224)$ ja normalisoitiin sekä kuvankäsittelyluokan avulla kuviin sovellettiin samoja muunnoksia kuin alkuperäisessä esikoulutetussa mallissa. Näitä on esimerkiksi normalisoitiin liittyvät parametrit arvoineen.

EfficientNet B0-mallin hienosäätö tehtiin hyödyntämällä torchvision-kirjastosta löytyvää esikoulutettua mallia ja parametrit valittiin samoin kuin Swin Transformer-mallin kohdalla. EffientNet-mallin pohjalta olevan mallin luokitin korvattiin uudella neuroverkkokerroksella, joka koulutettiin uudestaan.

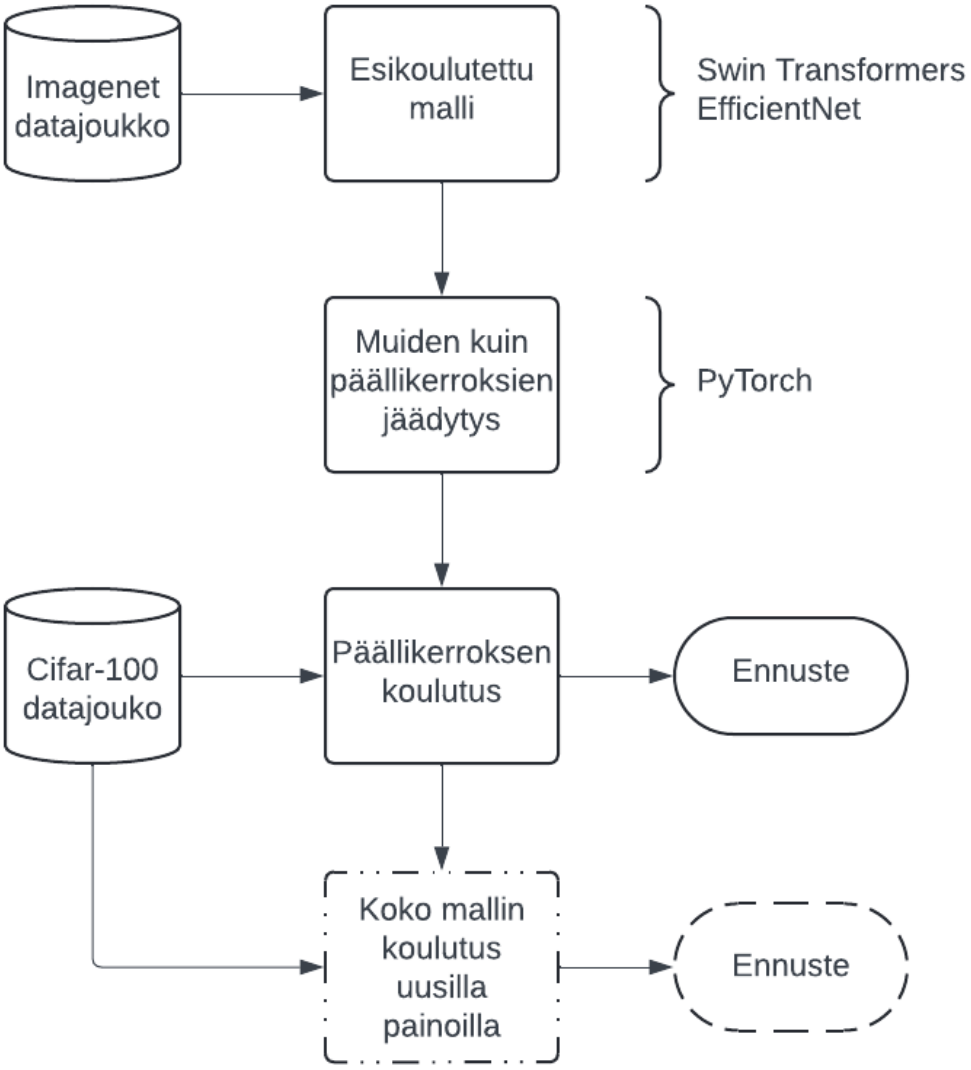
	Malli		
Parametri	ViT	Swin Transformer	EfficientNet B+
Opetusaste	$3 \times 10^{-}5$		
Epokkien lkm	5		
Eräkoko	16		
Kuvan koko	$(224, 224)$		
Tarkkuus	91,2%	85,78%	82,01%

Taulukko 7.1: Työssä tarkastellut mallit parametreineen

Normalisoinnin lisäksi malleihin ei tehty mitään datan muutoksia. Malleissa käytetyt parametrit on esitetty taulukossa 7.1. Mallien parametrit valittiin eri mallien välillä yhtälaisiksi, jotta tulokset olisivat mahdollisimman vertailukelpoisia.

7.3 Tulokset

Mallien hienosäädön perusteella, ilman kattavaa hyperparametrien säätämistä voidaan todeta että sekä ViT että Swin Transformer-pohjaisen mallin tarkkuus on parempi kuin konvoluutioverkkoihin perustuvan mallin. Kirjallisuudessa esitettyjen tuloksien, jotka on esitetty taulukossa 5.2, perusteella EfficientNet-mallin pitäisi päihittää Swin Transformer, koska käytössä oli mallin ensimmäinen versio. Tässä kokeellisessa osuudessa ei kuitenkaan ole käytetty samoja parametreja kuin kirjallisuudessa, joten eroja teoreettisiin tuloksiin voi esiintyä. Yleisesti ottaen sekä kokeellisten että kirjallisuudessa esiintyvien tuloksien perusteella voi arvioida, että sekä transformer- että konvoluutioneuroverkot pärjäävät hyvin luokittelutehtävässä. Molempien mallien koulutuksissa merkille pantavaa oli mallin kouluttamisen kesto, joka oli yli 20 minuuttia kaikilla malleilla.



Kuva 7.1: Hienosäätöprosessin kuvaus

8 Yhteenveto

Tässä työssä perehdyttiin transformer-arkkitehtuureihin konenäön eri tehtävissä, sekä vertailtiin transformer-malleja konvoluutioneuroverkkoihin kirjallisuudessa esitettyjen tuloksien että kokeellisin keinoin. Konvoluutioneuroverkkoihin verrattuna transformer-mallit ovat monikäyttöisempiä, mahdollistaen erilaisten konenäköongelmien ratkaisemisen samalla mallilla kun taas konvoluutioneuroverkoissa käytetään erikoistuneempia malleja. Mallien tarkkuudessa transformer-mallit pärjäävät hyvin konvoluutioneuroverkoille sekä kuvien luokittelussa että semanttisessa segmentoinnissa.

8.1 Tulokset tutkimuskysymyksien valossa

Ensimmäinen tutkimuskysymys oli selvittää miten transformer-arkkitehtuuri eroaa konvoluutioverkkojen arkkitehtuurista (TK1). Konvoluutioverkkojen periaatteena on hyödyntää kuvista löytyviä ominaisuuksia, jotka perustuvat kuvien vierekkäisiin pikseleihin sekä niiden välisiin suhteisiin eli kuvien spatiaalisiin ominaisuuksiin.

Transformer-arkkitehtuureissa, joiden tausta on luonnollisen kielen käsittelyssä, mallien syötteenä on vektori johon kuva on jaettu halutun säännön mukaan: esimerkiksi esitellyssä ViT-mallissa kuva jaettiin 16×16-kokoisiin lohkoihin ja Swin Transformer-mallissa kuvia käsiteltiin hierarkisesti siten, että kuvien koko pienenee kun mallin kerroksissa edetään. Konvoluutioverkossa haluttuun lopputulokseen päästään kerroksissa tapahtuvien operaatioiden kautta, joissa esimerkiksi syötteen

eli kuvan kokoa pienennetään tai täytetään. Lopulta kerroksittain lasketaan kyseisen kerroksen aktivointifunktio, jonka tuloksen pohjalta seuraava kerros laskee omat laskutoimituksensa. Transformer-arkkitehtuurissa mallin oppiminen perustuu huomiomekanismiin, joka perustuu yksinkertaisiin matriisilaskutoimituksiin eli matriisituloihin. Tässä työssä esiteltiin huomiomekanismeista sekä skaalattu itsehuomio että monipäinen itsehuomio. Molemmissa huomiomekanismeissa malli saa syötteenä matriisin $\mathbf{X}$, jolle pyritään löytämään kolme painomatriisia $\mathbf{W}^{(K)}$, $\mathbf{W}^{(V)}$ sekä $\mathbf{W}^{(Q)}$ joiden arvot malli oppii. Näiden matriisien skaalatun matriisitulon *Softmax*-funktio on skaalattu itsehuomio, ja näiden usealle arvolle laskettujen arvojen yhdistelmä muodostaa monipäisen itsehuomion. Transformer-mallit perustuvat näiden yhdistelmille. Näiden lisäksi transformer-malleissa on oleellisena osana myös kerrosnormi sekä jäännösyhteys normalisointikerroksen jälkeen (kaavat 4.12 sekä 4.13). Näiden merkitys on tasapainottaa verkkoa ja vähentää opetusaikaa. Myös konvoluutioneuroverkoissa käytetään jäännösyhteyksiä kerroksien välillä, esimerkiksi ResNet-mallissa [54].

Tässä työssä keskityttiin konenäön sovellutuksiin, erityisesti luokitteluun kuin semanttiseen segmentointiin, mutta näiden lisäksi transformer-arkkitehtuuriin perustuvia malleja voidaan hyödyntää myös muunlaisiin ongelmiin.

Toinen tutkimuskysymys oli selvittää miten transformer-arkkitehtuurin pohjalta tehdyn neuroverkon suorituskyky eroaa *state-of-art* konvoluutioverkkoon kuvien luokittelussa (TK2). Työssä esitettyjen tuloksien perusteella transformer-arkkitehtuuriin perustuvat mallit toimivat melko hyvin jo hyvin pienellä hienosäädöllä: ViT-mallin tarkkuus oli $91,2\%$, ja Swin Transformer-mallin tarkkuus oli $85,78\%$. Kirjallisuuden perusteella EfficientNet-mallin tarkkuus on Swin Transformer-mallin suuruusluokkaa.

Semanttisen segmentoinnin tehtävässä transformer-arkkitehtuuripohjaiset mallit pärjäävät hyvin verrattuna konvoluutioverkkopohjaisiin malleihin. ADE20K-aineis-

tolla tulokset olivat huomattavasti parempia kuin konvoluutioverkkoihin perustuvalla mallilla sekä Pascal VOC-aineistolla ero oli pieni.

8.2 Transformer-mallien haasteet ja rajoitukset

Tämän työn tuloksien perusteella transformer-mallit pärjäävät hyvin konvoluutioneuroverkoille, etenkin kun malleja kouluttaa hienosäätämällä valmiita, isompia malleja. Transformer-mallien kouluttaminen vaatii kuitenkin laskentatehoa, koska huomiomekanismin laskeminen on neliöllinen operaatio, ja syötteen koko voi osoittautua pullonkaulaksi [55]. Jos valmista mallia ei ole hienosäädettäväksi, mallin koulutus omalla aineistolla vaatii erittäin paljon aineistoa sekä myös laskentatehoa. Etenkin gradun tai vastaavan työn resursseilla tällaiseen ei välttämättä ole varaa. Laskentatehon vaatimukset saattavat rajata transformer-mallien käyttöä sellaisiin ympäristöihin, joissa tehoa on saatavilla ja jättää ulkopuolelle vähävirtaiset ympäristöt.

Toinen rajoite transformer-malleissa konenäön tehtävissä on se, että ne eivät huomioi kuvissa olevia spatiaalisia ominaisuuksia toisin kuin konvoluutioneuroverkot. Tässä työssä esitetyissä malleissa, kuten Swin Transformer sekä Segmenter pyrkivät ratkaisemaan kuvien spatiaalisiin piirteisiin liittyviä ongelmia mallien kerroksien hierarkioiden sekä kytkentöjen avulla.

Kolmas haaste transformer-malleissa on myös se, että valmiita malleja etenkin konenäön tehtäviin on tarjolla melko vähän verrattuna konvoluutioneuroverkkoihin perustuviin malleihin. Tämä vaikeuttaa esimerkiksi ensimmäisen haasteen ratkaisemista. Kokeellisessa osiossa tehdyt testit tehtiin Huggingface Transformers-kirjastosta [53] löytyneillä malleilla. Vaikka Huggingface Transformers-kirjastosta löytyy paljon esikoulutettuja malleja, niiden luotettavuus voi olla kyseenalaista, koska kaikkia hienosäädettyjä malleja ei ole vertaisarvioitu tai tekijöiden osaamista ei ole voitu varmistaa.

8.3 Mahdolliset jatkotutkimukset

Tässä työssä perehdyttiin transformer-arkkitehtuureihin erityisesti konenäön tehtävissä: luokittelussa sekä segmentoinnissa. Työssä käytettiin vertailuun hyvin yleisluontoista aineistoa, Cifar-100-aineistoa, jossa on erittäin paljon luokkia kuville. Käytännön sovelluksissa luokkien määrä voi olla paljon pienempi tai dataa voi olla vähemmän. Vaikka Cifar-100 voi olla hyvä mallien ensimmäiseen kokeiluun siksi, että tuloksia on helppo verrata muihin samalla aineistolla tehtyyn tutkimukseen, niin tulokset voivat riippua myös varsinaisesta sovelluskohteesta johon etsitään mallia.

Tämän työn pohjalta transformereiden ja konvoluutioneuroverkkojen eroja voisi tarkastella tarkemmin erilaisilla hyperparametrien optimoinnin menetelmillä sekä erilaisilla aineistoilla, sekä vertailla menetelmien eroja erilaisiin sovelluskohteisiin. Lääketieteellisten kuvien osalta on tehty jo tutkimusta siitä, voisiko transformermallit korvata konvoluutioneuroverkot näissä tehtävissä [56]. Tässä tutkimuksessa käytettiin hyvin yleisiä lääketieteellisiä aineistoja erilaisilla mallien alkuarvoilla, jolloin vain satunnaisesti alustettu transformer-malli oli huonompi kuin konvoluutioverkko eli transformerit voisivat hyvin korvata konvoluutioverkot näissä tehtävissä. Lääketieteen sovellusalueella voisi kuitenkin olla vielä hyvin paljon tutkittavaa mallien eroissa, koska datajoukot eivät välttämättä ole merkittävän suuria [57] ja konenäköalgoritmien tulokset voivat vaikuttaa merkittävästi siihen miten hyviä diagnooseja potilaat saavat.

Toinen kiinnostava sovellusala transformer-mallien ja konvoluutioneuroverkkojen vertailuun on itseohjautuvat ajoneuvot tai kuljettajaa avustavat järjestelmät ajoneuvoissa. Transformer-mallien käytössä itseohjautuvissa autoissa avoimina kysymyksinä on mallien toteutus, laskennallinen vaativuus sekä laitteistokiihdytys sekä mallien tulkinta ja selitettävyys [58].

Kolmas lääketieteellistä ja ajoneuvoja yhdistävä tekijä on mallien selitettävyys, koska mallien vaikutukset ihmisille voivat olla erittäin kohtalokkaita vaikkapa on-

nettomuuksien tai väärän diagnoosin aiheuttaman hoitovirheen tai inhimillisen kärsimyksen vuoksi. ViT-mallin artikkelissa [31] esitettiin miten mallien huomiomekanismeja voi arvioida ja visualisoida. Koneoppimismalleille on kehitetty selitettävyysmenetelmiä (esim. [59]), koska mallit voivat olla herkkiä väärille tuloksille tai tuottaa ennakoimattomia tuloksia [60]. Vaikka transformer-mallien selitettävyyttä on tutkittu [61]), mallien ymmärtämisen kannalta voisi olla hyvä myös vertailla eri tyyppisiä malleja mallien tuottamien tuloksien oikeellisuuden sekä selitettävyyden näkökulmasta.

Transformer-arkkitehtuureissa käytetyn huomiomekanismin pohjalta on kehitetty uusia arkkitehtuureita, kuten multimodaalinen Perceiver-arkkitehtuuri [28], joka pyrkii ratkaisemaan huomiomekanismin neliölliseen laskentavaatimukseen liittyviä haasteita sekä Segment Anything 2 [62], jossa hyödynnetään huomiomekanismeja siihen että voidaan luoda malli jolla voi segmentoida mitä tahansa kuva- tai videoaineistoa. Näiden mallien sekä tehokkaampien huomiomekanismien tutkiminen saattaisi lisätä ymmärrystä siitä mihin transformer-mallit ovat kehittymässä ja miten neliölliseen laskentavaatimukseen liittyviä rajoituksia voisi ratkaista.

8.4 Johtopäätökset

Tämän työn perusteella transformer-arkkitehtuureihin perustuvat mallit soveltuvat hyvin monenlaisiin konenäön tehtäviin, ja ne kykenevät suoriutumaan niistä hyvällä tarkkuudella. Verrattuna konvoluutioverkkoihin perustuviin malleihin, osa transformer-arkkitehtuureihin perustuvista malleista ovat erittäin monikäyttöisiä ja mahdollistavat monien erilaisten konenäön tehtävien suorittamisen. Transformer-mallien huono puoli on kuitenkin laskennallinen vaativuus, koska huomiomekanismin laskeminen perustuu matriisituloon, jonka laskennallinen vaativuus on neliöllinen ajan sekä tilan suhteen.

Käytännön sovelluksissa transformer-mallit edellyttävät kuitenkin valmiita mal-

leja, koska niiden kouluttaminen on laskennallisesti hidasta. Valmiiden mallien hienosäätö, siirto-oppiminen sekä tietämyksen siirto kuitenkin mahdollistavat mallien hyödyntämisen myös ilman konesalitason laskentakapasiteettiä.

Lähdeluettelo

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail ja H.-T. Lin, *Learning From Data*. AMLBook, 2012. url: https://amlbook.com.

[2] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[3] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman ja Hall/CRC, 2011.

[4] J. Berner, P. Grohs, G. Kutyniok ja P. Petersen, "The Modern Mathematics of Deep Learning", teoksessa *Mathematical Aspects of Deep Learning*. Cambridge University Press, joulukuu 2022, s. 1–111, ISBN: 9781316516782. DOI: 10.1017/9781009025096.002. url: http://dx.doi.org/10.1017/9781009025096.002.

[5] Reaktor, "Kannattaako kauppa", Kannattaako kauppa -verkkopalveu. url: https://reaktor.github.io/Neliohinnat/202205-comp-blog.html.

[6] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. "O'Reilly Media, Inc.", 2022.

[7] D. Foster, *Generative deep learning*. "O'Reilly Media, Inc.", 2022.

[8] T. Hastie, R. Tibshirani, J. H. Friedman ja J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[9] M. Mohri, A. Rostamizadeh ja A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[10] D. A. Roberts, S. Yaida ja B. Hanin, *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge University Press, toukokuu 2022, ISBN: 9781316519332. DOI: 10.1017/9781009023405. url: http://dx.doi.org/10.1017/9781009023405.

[11] C. M. Bishop ja H. Bishop, *Deep Learning: Foundations and Concepts*. Springer, 2024.

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

[13] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[14] S. J. Prince, *Understanding Deep Learning*. MIT Press, 2023. url: http://udlbook.com.

[15] J. Duchi, E. Hazan ja Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.", *Journal of machine learning research*, vol. 12, nro 7, 2011.

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

[17] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. url: probml.ai.

[18] K. P. Murphy, *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023. url: http://probml.github.io/book2.

[19] G. Hinton, O. Vinyals ja J. Dean, *Distilling the Knowledge in a Neural Network*, 2015. arXiv: 1503.02531 [stat.ML].

[20] J. Gou, B. Yu, S. J. Maybank ja D. Tao, "Knowledge Distillation: A Survey", *International Journal of Computer Vision*, vol. 129, nro 6, s. 1789–1819, maaliskuu 2021, ISSN: 1573-1405. DOI: 10.1007/s11263-021-01453-z. url: http://dx.doi.org/10.1007/s11263-021-01453-z.

[21] D. H. Wolpert ja W. G. Macready, "No free lunch theorems for optimization", *IEEE transactions on evolutionary computation*, vol. 1, nro 1, s. 67–82, 1997.

[22] A. Vaswani, N. Shazeer, N. Parmar et al., *Attention Is All You Need*, 2023. arXiv: 1706.03762 [cs.CL].

[23] K. Hornik, "Approximation capabilities of multilayer feedforward networks", *Neural networks*, vol. 4, nro 2, s. 251–257, 1991.

[24] C. Yun, S. Bhojanapalli, A. S. Rawat, S. J. Reddi ja S. Kumar, *Are Transformers universal approximators of sequence-to-sequence functions?*, 2020. arXiv: 1912.10077 [cs.LG].

[25] S. Ahmed, I. E. Nielsen, A. Tripathi, S. Siddiqui, R. P. Ramachandran ja G. Rasool, "Transformers in time-series analysis: A tutorial", *Circuits, Systems, and Signal Processing*, vol. 42, nro 12, s. 7433–7466, 2023.

[26] S. Kim, A. Gholami, A. Shaw et al., "Squeezeformer: An efficient transformer for automatic speech recognition", *Advances in Neural Information Processing Systems*, vol. 35, s. 9361–9373, 2022.

[27] C. Raffel, N. Shazeer, A. Roberts et al., *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*, 2023. arXiv: 1910.10683 [cs.LG]. url: https://arxiv.org/abs/1910.10683.

[28] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals ja J. Carreira, *Perceiver: General Perception with Iterative Attention*, 2021. arXiv: 2103.03206 [cs.CV]. url: https://arxiv.org/abs/2103.03206.

[29] T. Lin, Y. Wang, X. Liu ja X. Qiu, *A Survey of Transformers*, 2021. arXiv: 2106.04554 [cs.LG].

[30] J. Devlin, M.-W. Chang, K. Lee ja K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov et al., *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, 2021. arXiv: 2010.11929 [cs.CV].

[32] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles ja H. Jégou, *Training data-efficient image transformers & distillation through attention*, 2021. arXiv: 2012.12877 [cs.CV].

[33] H. Bao, L. Dong, S. Piao ja F. Wei, *BEiT: BERT Pre-Training of Image Transformers*, 2022. arXiv: 2106.08254 [cs.CV].

[34] Z. Liu, Y. Lin, Y. Cao et al., *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*, 2021. arXiv: 2103.14030 [cs.CV].

[35] J. L. Ba, J. R. Kiros ja G. E. Hinton, *Layer Normalization*, 2016. arXiv: 1607.06450 [stat.ML].

[36] K. Simonyan ja A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[37] C. Szegedy, W. Liu, Y. Jia et al., *Going Deeper with Convolutions*, 2014. arXiv: 1409.4842 [cs.CV]. url: https://arxiv.org/abs/1409.4842.

[38] A. Krizhevsky, I. Sutskever ja G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, vol. 25, 2012.

[39] O. Ronneberger, P. Fischer ja T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015. arXiv: 1505.04597 [cs.CV].

[40] G. Csurka, R. Volpi ja B. Chidlovskii, *Semantic Image Segmentation: Two Decades of Research*, 2023. arXiv: 2302.06378 [cs.CV].

[41] H. Thisanke, C. Deshan, K. Chamith, S. Seneviratne, R. Vidanaarachchi ja D. Herath, *Semantic Segmentation using Vision Transformers: A survey*, 2023. arXiv: 2305.03273 [cs.CV].

[42] J. Long, E. Shelhamer ja T. Darrell, *Fully Convolutional Networks for Semantic Segmentation*, 2015. arXiv: 1411.4038 [cs.CV].

[43] F. Milletari, N. Navab ja S.-A. Ahmadi, *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*, 2016. arXiv: 1606.04797 [cs.CV]. url: https://arxiv.org/abs/1606.04797.

[44] P. Wang, P. Chen, Y. Yuan et al., *Understanding Convolution for Semantic Segmentation*, 2018. arXiv: 1702.08502 [cs.CV].

[45] Z. Xu, W. Zhang, T. Zhang, Z. Yang ja J. Li, "Efficient transformer for remote sensing image segmentation", *Remote Sensing*, vol. 13, nro 18, s. 3585, 2021.

[46] S. Zheng, J. Lu, H. Zhao et al., *Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers*, 2021. arXiv: 2012.15840 [cs.CV].

[47] R. Strudel, R. Garcia, I. Laptev ja C. Schmid, "Segmenter: Transformer for semantic segmentation", teoksessa *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, s. 7262–7272.

[48] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez ja P. Luo, *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers*, 2021. arXiv: 2105.15203 [cs.CV].

[49] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn ja A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective", *International Journal of Computer Vision*, vol. 111, nro 1, s. 98–136, tammikuu 2015.

[50] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso ja A. Torralba, "Scene parsing through ade20k dataset", teoksessa *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, s. 633–641.

[51] M. Cordts, M. Omran, S. Ramos et al., *The Cityscapes Dataset for Semantic Urban Scene Understanding*, 2016. arXiv: 1604.01685 [id='cs.CV' full$_name$ =' $ComputerVisionandPatternRecognition'is_active = True alt_name = None in_archive =' cs'is_general = False description =' Coversimageprocessing, computervision, patternrec$

[52] A. Krizhevsky ja G. Hinton, "Learning multiple layers of features from tiny images", University of Toronto, Toronto, Ontario, Technical Report UTML TR 2009-003, 2009. url: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[53] T. Wolf, L. Debut, V. Sanh et al., *HuggingFace's Transformers: State-of-the-art Natural Language Processing*, 2020. arXiv: 1910.03771 [cs.CL]. url: https://arxiv.org/abs/1910.03771.

[54] K. He, X. Zhang, S. Ren ja J. Sun, *Deep Residual Learning for Image Recognition*, 2015. arXiv: 1512.03385 [cs.CV]. url: https://arxiv.org/abs/1512.03385.

[55] S. Wang, B. Z. Li, M. Khabsa, H. Fang ja H. Ma, *Linformer: Self-Attention with Linear Complexity*, 2020. arXiv: 2006.04768 [cs.LG]. url: https://arxiv.org/abs/2006.04768.

[56] C. Matsoukas, J. F. Haslum, M. Söderberg ja K. Smith, *Is it Time to Replace CNNs with Transformers for Medical Images?*, 2021. arXiv: 2108.09038 [cs.CV]. url: https://arxiv.org/abs/2108.09038.

[57] I. M. Perez, "Data analysis with limited data availability: prostate cancer prediction and characterization as a case study", Doctoral Dissertation, University of Turku, toukokuu 2024, ISBN: 978-951-29-9646-9. url: https://www.utupub.fi/handle/10024/176943.

[58] Q.-V. Lai-Dang, *A Survey of Vision Transformers in Autonomous Driving: Current Trends and Future Directions*, 2024. arXiv: 2403.07542 [cs.CV]. url: https://arxiv.org/abs/2403.07542.

[59] M. Munn ja D. Pitman, *Explainable AI for practitioners*. "O'Reilly Media, Inc.", 2022.

[60] A. Ghorbani, A. Abid ja J. Zou, *Interpretation of Neural Networks is Fragile*, 2018. arXiv: 1710.10547 [stat.ML]. url: https://arxiv.org/abs/1710.10547.

[61] R. Kashefi, L. Barekatain, M. Sabokrou ja F. Aghaeipoor, *Explainability of Vision Transformers: A Comprehensive Review and New Perspectives*, 2023. arXiv: 2311.06786 [cs.CV]. url: https://arxiv.org/abs/2311.06786.

[62] N. Ravi, V. Gabeur, Y.-T. Hu et al., *SAM 2: Segment Anything in Images and Videos*, 2024. arXiv: 2408.00714 [cs.CV]. url: https://arxiv.org/abs/2408.00714.