LAB University of Applied Sciences

Tekoälypohjainen videodatan indeksointi

LAB-ammattikorkeakoulu
Insinööri (YAMK), IoT:stä tekoälyyn
2022
Joel Pesu

Tiivistelmä

Tekijä	Julkaisun laji	Valmistumisaika
Pesu, Joel	Opinnäytetyö, YAMK	Syksy 2022
	Sivumäärä	
	77	

Työn nimi

Tekoälypohjainen videodatan indeksointi

Tutkinto ja koulutusala

Insinööri (ylempi AMK), IoT:stä tekoälyyn

Tiivistelmä

Videodatan määrän jatkuvasti kasvaessa on tullut haastavaksi löytää etsimänsä nopeasti. Käyttöjärjestelmien hakumenetelmät eivät mahdollista videosisällön perusteella tiedostojen etsimistä. Hakemisen helpottamiseksi videosisältöä tulee voida indeksoida sisällön mukaan. Tekoälyn hyödyntäminen on mahdollistanut kuvadatan tulkitsemisen ihmissilmän tarkkuudella ja siksi tässä opinnäytetyössä tarkoituksena oli löytää soveltuvin neuroverkkoarkkitehtuuri videosisällön indeksointiin.

Opinnäytetyössä käytettiin dokumenttianalyysia tutkittaessa olemassa olevia vaihtoehtoja neuroverkkoarkkitehtuurin löytämiseksi. Konstruktiivisella tutkimuksella opinnäytetyössä iteroitiin eri neuroverkkoarkkitehtuureja ja empiirisesti testattiin niiden toimivuutta käytännössä. Kokeiltuja arkkitehtuureja olivat: konvoluutioneuroverkko ResNet50, siamilainen konvoluutioneuroverkko ja Autoencoder. Opinnäytetyössä pyrittiin ratkaisemaan neuroverkkomallin sovittaminen vähäisellä datamäärällä. Soveltuvimmaksi arkkitehtuuriksi valittiin siamilainen konvoluutioneuroverkko Precision-lukemalla 78,2 % ja siitä innovoitiin ja kehitettiin prototyyppipalvelu. Prototyyppipalvelun toimivuutta testattiin käytännössä rajatulla materiaalilla. Prototyyppipalvelun kelvolliset tulokset kannustivat projektin jatkokehitykseen. Lopuksi opinnäytetyössä esitettiin mahdollisia jatkokehitystoimenpiteitä palvelun parantamiseksi.

Asiasanat

Tekoäly, koneoppiminen, konvoluutioneuroverkko, indeksointi

Abstract

Author	Type of Publication	Published
Pesu, Joel	Master's Thesis	Fall 2022
	Number of Pages	
	77	

Title of Publication

Neural network-based video indexing service

Degree and field of study

Master of Engineering, From IoT to AI

Abstract

With the amount of video data constantly increasing, it has become challenging to find what one is looking for. Search methods of the operating systems do not enable searching for files based on video content. To make such searching possible, video content should be indexed by content. The use of artificial intelligence has made it possible to interpret image data with the accuracy of the human eye. To leverage recent developments in artificial intelligence, the purpose of this thesis was to find the most suitable neural network architecture for video content indexing.

Document analysis was used in the thesis to investigate existing solutions and research for finding a neural network architecture to fit the needs of indexing. Using constructive research different neural network architectures were empirically tested in practice and the most suitable neural network architecture was chosen. Empirically tested architectures were convolutional neural network ResNet50, Siamese neural network and an autoencoder. The aim of the thesis was to find a neural network architecture capable of fitting with a small amount of data. Siamese neural network was chosen as the most suitable architecture with a Precision value of 78,2 %. A prototype service was implemented around the chosen Siamese network and its functionality was tested in practice with limited material. Adequate results of the prototype encouraged for further development of the prototype. Finally, possible further development measures to improve the service were identified.

Keywords

Artificial Intelligence, machine learning, convolutional neural networks, indexing

Sisällys

Liitteet

1 Johdanto

Teknologian kehittyessä kuvadataa syntyy koko ajan enemmän. Puhelimien muistit kasvavat ja palveluntarjoajat tuovat markkinoille lisää pilvitallennustilaa. Kuvadatan tulkitseminen neuroverkoilla on kehittynyt viime vuosikymmenen aikana merkittävästi. Kuvadataa voidaan kategorisoida, kategorioita lokalisoida ja seurata sekä generoida täysin uutta kuvadataa.

Neuroverkkojen ansiosta kuvadatan analysointi on kehittynyt merkittävästi. Nykyisin analysointi on niin hyvää ja tarkkaa, että neuroverkkoihin perustuvaa kuva-analyysia käytetään miltei kaikkialla: ihmisten kasvojen tunnistuksessa, terveydenhoidon magneettikuvien analysoinnissa, autonomisesti kulkevissa ajoneuvoissa ja jopa erilaisten turvallisuusuhkien tunnistamisessa (Palash & Young-Gab 2022, 2).

Avoimeen lähdekoodiin perustuvat neuroverkkojen kehitysympäristöt kuten Googlen TensorFlow, ovat mahdollistaneet laajan neuroverkkojen käytön. (TensorFlow 2022a.) Niitä käyttävät niin valtiot, yksityiset yritykset, kuin opiskelijatkin. Näytönohjainten laskutehon räjähdysmäinen kehitys ja neuroverkkojen matriisilaskentaan tarkoitettujen matrsiisilaskentaydinten (TPU) markkinoille tulo ovat helpottaneet neuroverkkojen käyttöönottoa.

Kuvadatan tulkitseminen ohjelmallisesti on kehittyvä ala, jossa yhä parempaan tarkkuuteen, ja nopeampaan laskutoimitukseen kykeneviä algoritmeja kehitetään vuosi vuodelta lisää (Kang ym. 2020). Näin ollen kuvadataan liittyvän analyysiin ammattilaisilta vaaditaan uusien lähestymistapojen opiskelua ja niihin tutustumista jatkuvasti.

1.1 Rajaus

Tässä opinnäytetyössä tutkitaan eri neuroverkkoarkkitehtuureja ja niiden soveltuvuutta videopankin analyysiin. Tässä yhteydessä videopankilla tarkoitetaan verkkolevyiltä tai muista lähteistä löytyviä suuria videomääriä. Aluksi opinnäytetyössä käydään läpi kuvadatan kehitystä ja tutkitaan tarkemmin muutamaa mielenkiintoista vaihtoehtoa ongelman ratkaisemiseksi. Seuraavaksi tehdään sarja kokeiluja mahdollisesti soveltuvilla neuroverkkoarkkitehtuureilla ja lopuksi paras arkkitehtuuri valitaan. Soveltuvimmasta arkkitehtuurista sovitetaan malli ja tuotetaan niin sanottu Proof of Concept, eli konseptipalvelu PoC. Palvelun tarkoitus on helpottaa videoiden etsimistä sisällön perusteella. Lopuksi konseptipalvelu toteutetaan hyödyntäen soveltuvinta mallia ja palvelun toimintaa kokeillaan käytännössä tekemällä sarja testejä valitun datan kanssa.

Opinnäytetyössä ei yritetä kehittää uutta arkkitehtuuria, parantaa tutkittujen arkkitehtuurien tarkkuutta tai nopeuttaa lasku- tai sovitusaikaa. Tässä työssä ei myöskään oteta kantaa mallien sovitusaikaan tai mallin sovitukseen käytetystä laitteistosta. Lisäksi opinnäytetyön

neuroverkkomallien sovittaminen ja validointi tapahtuu rajatulla datasetillä, sillä monien eri datasettien tutkiminen ja validointi tekisi tutkimuksesta liian laajan.

1.2 Tavoitteet ja tutkimuskysymykset

Indeksoinnista käyttöjärjestelmissä ja hakukoneissa on tullut itsestäänselvyys. Tekstipohjaista informaatiota on helppo hakea esimerkiksi Googlen kautta, tai omalta lähiverkon verkkolevyltä hakusanan mukaan. Kuva- ja videomateriaalin määrän kasvaessa ongelmaksi on tallennustilan sijaan muodostunut datan jälleen hakemiseen liittyvät ongelmat. Kuva- tai videomateriaalin tallentamisen yhteydessä kuvamateriaalia ei useimmiten kategorisoida. Tämän vuoksi videomateriaalin hakeminen sen sisältävän informaation kautta voi olla haastavaa. Suuret teknologiayhtiöt ovat alkaneet kehittää ongelmaan ratkaisuja, mutta ainoastaan niiden alustoja hyödyntäen. Esimerkiksi Facebook voi automaattisesti tunnistaa kuvista henkilöitä ja Google Photos kykenee kategorisoimaan otettuja kuvia erilaisiin luokkiin, kuten asiakirjoihin, ihmisiin, eri vuodenaikoihin ja niin edelleen.

Opinnäytetyön tavoitteena on tuottaa edellä mainitun tapainen palvelu luokittelemaan verkkolevyn videomateriaalia sen hakemisen helpottamiseksi hyödyntäen mahdollisimman vähäistä esimerkkidataa haluttujen hakutulosten saavuttamiseksi. Kuvadatan luokitteluun on monia eri lähestymistapoja, joilla jokaisella vahvuuksia ja heikkouksia, joten tutkimuksen tavoitteena on löytää palveluun sopiva arkkitehtuuri, sovittaa malli ja validoida sen toimivuus PoC-palvelussa.

Opinnäytetyössä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

1. Mitä neuroverkkoarkkitehtuureja on olemassa kuvadatan luokitteluun?
2. Mitkä ovat valittujen neuroverkkoarkkitehtuurien vahvat ja heikot ominaisuudet?
3. Mikä valituista neuroverkkoarkkitehtuureista soveltuu parhaiten ratkaisemaan verkkolevylle tallennetun videomateriaalin luokittelun ja miten se toimii käytännössä?

Opinnäytetyö on konstruktiivinen tutkimus, jossa tutkimusmenetelmänä käytetään dokumenttianalyysia. Konstruktiivisella tutkimuksella perehdytään syvällisesti aiheeseen ja innovoidaan ongelmaan ratkaisu. (Lukka 2001.) Asiaan perehdyttäessä apuna käytetään dokumenttianalyysia, mikä tarkoittaa lähdemateriaalin etsimistä ja sen ymmärtämistä (Anttila 1998).

Lähteinä hyödynnetään laajasti artikkeleita, alan ammattilaisten erikoisalansa konferenseissa ja julkaisuissa esittelemiä tutkimuksia (ns. research papers), asiantuntijoiden blogikirjoituksia ja luentoja.

2 Pikselidata neuroverkoissa

2.1 Konvoluutioneuroverkko

Konvoluutioneuroverkot, eli CNN:t (Convolutional Neural Network) ovat neuroverkkojen alakategoria, ja niitä käytetään muun muassa kuvan, videon, äänen ja aikasarjadatan analysoinnissa. CNN:t ovat erinomaisia analysoimaan dataa, jossa spatiaalinen tai kronologinen järjestys on tärkeää. CNN:istä on tullut yleisimmin käytetty malli konenäössä. Kuvadatan tulkintaan CNN:t ovat erityisen sopivia, sillä vierekkäiset pikselit muodostavat havaittavia kokonaisuuksia. (Wikipedia 2022a.)

2.2 CNN kerrokset

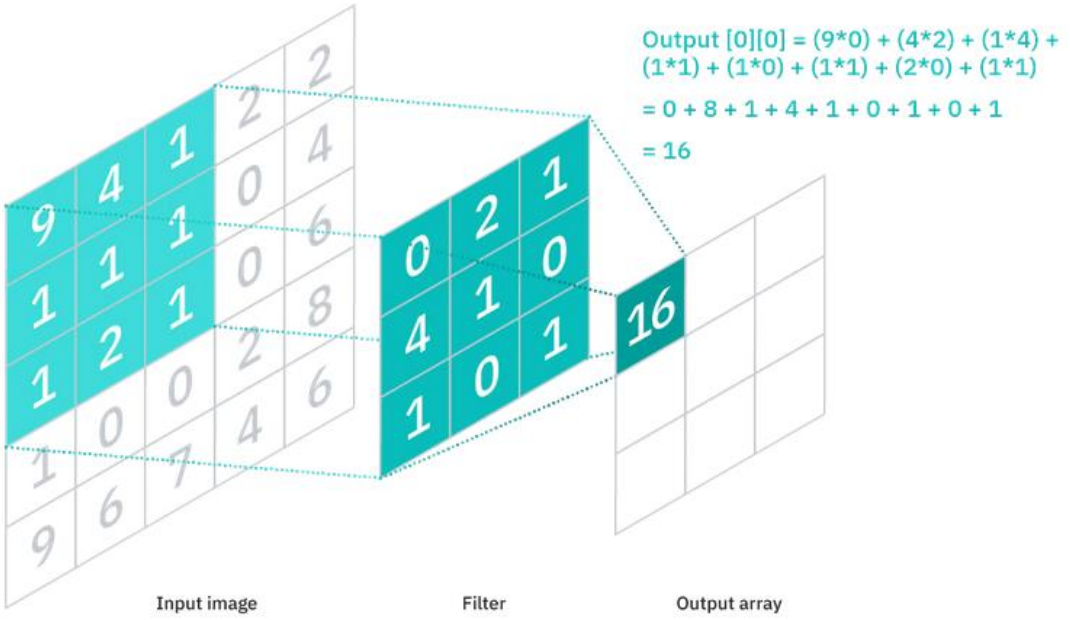
Keskeisiä kerroksia CNN:ssä ovat: Convolutional Layer, eli konvoluutiokerros, Pooling Layer ja Fully-connected Layer. Kyseisiä kerroksia hyödynnetään usein eri CNN arkkitehtuurissa, tai niiden johdannaisissa.

2.2.1 Konvoluutiokerros

Konvoluutiokerros on CNN:n tärkein osa. Konvoluutiokerros koostuu niin sanotuista Filtereistä, joita liikuttelemalla kartoitetaan syötteestä löytyvät samankaltaisuudet Filtereiden kanssa. Filterin koko on yksi konvoluutiokerroksen hyperparametreistä. Samankaltaisuus syötteen kanssa lasketaan pistetulolla. Kun Filteriä liikutetaan koko syötteen läpi, niin Filterin mukainen piirre voi esiintyä missä tahansa kohdassa syötettä. Tämä on yksi konvoluutiokerroksen tärkeimmistä piirteistä, joka tekee siitä erinomaisen työkalun kuvadatan analysoinnissa. Muita hyperparametrejä konvoluutiokerroksessa on Stride, joka määrittää pikselimäärän, jonka Filter liikkuu jokaisen pistetulolaskutoimituksen välissä, Padding, jonka avulla syötteen ympärille voidaan lisätä tilaa, jotta Filter mahtuu jokaiseen kohtaan syötettä sekä Filtereiden lukumäärä. (IBM 2020.)

Konvoluutiokerroksen tuloste on niin sanottu Feature Map. Nimensä mukaisesti Feature Map on matriisi, joka sisältää informaation siitä, kuinka samanlaisia piirteitä syötteestä löytyy Filtereiden kanssa. Tulostettavien Feature Mapien määrä määräytyy konvoluutiokerroksen Filtereiden lukumäärän mukaan (IBM 2020). CNN:n sovitusvaiheessa konvoluutiokerroksen optimoitavat parametrit ovat Filterit.

Peräkkäin kytketyt konvoluutiokerrokset mahdollistavat monimutkaisten piirteiden erottelun ja tämän vuoksi konvoluutioneuroverkkoja kutsutaan usein Feature Extractoreiksi (Rosebrock 2019).

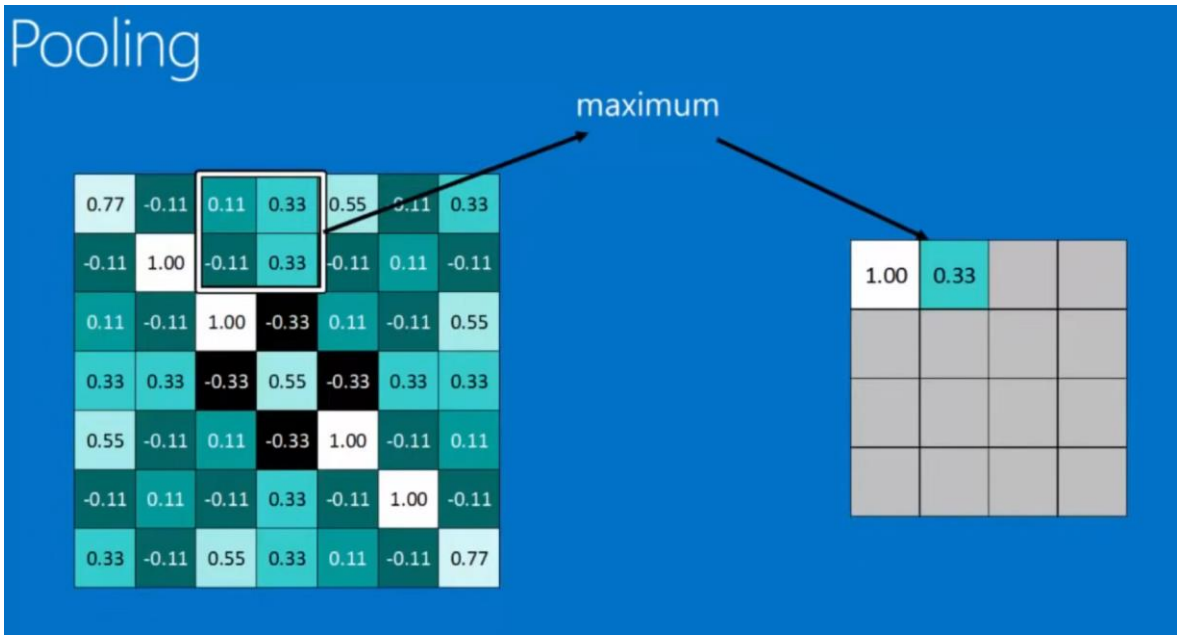


Kuva 1. Konvoluutiokerros 3x3 filtterillä (IBM 2020)

2.2.2 Pooling Layer

Pooling Layer on kerros, jossa konvoluutiokerroksen tapaan liikutetaan Filteriä hyperparametri Striden mukaisesti. Pooling Layerissä Filterin maksimi tai keskiarvo kirjataan tulosteeseen. Jos Pooling Layerissä kirjataan maksimiarvo, kutsutaan sitä Max Poolingiksi, ja jos kyse on keskiarvosta, kutsutaan sitä Average Poolingiksi (IBM 2020). Pooling Layerit ilmenevät useimmiten konvoluutiokerrosten jälkeen, eli konvoluutiokerroksen tuloste syötetään Pooling Layerille.

Pooling Layeriä kutsutaan myös Down Samplingiksi, sillä Pooling Layerin tuloste on tyypillisesti dimensioltaan syötettä pienempi (IBM 2020). Pooling Layer auttaa myös mallia generalisoimaan muotoja lokaatiosta riippumatta, sillä syötteestä kartoitettu piirteiden ja Filtereiden välinen samankaltaisuuden informaatio siirtyy maksimi- tai keskiarvon mukana eteenpäin.



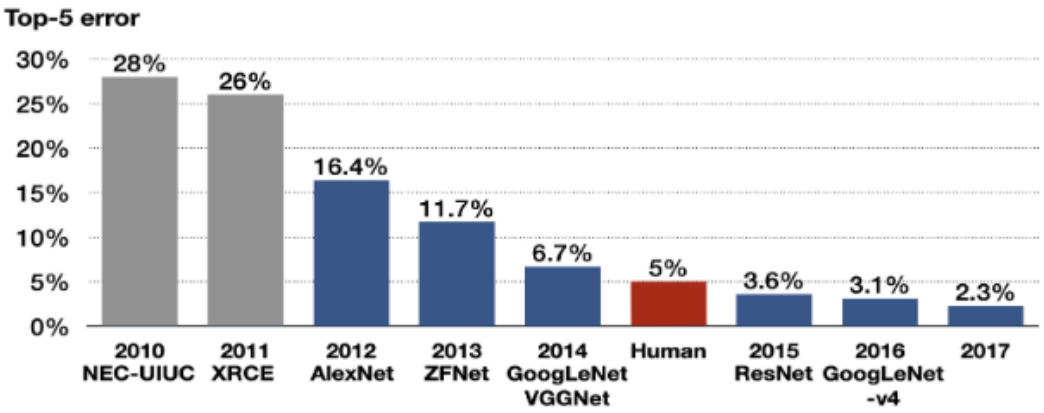
Kuva 2. Max Pooling -taso 2x2 filtterin koolla (Rohrer 2016)

2.2.3 Fully-Connected Layer

Fully-Connected Layerissä konvoluutiokerroksen ja Pooling Layerin tuottamista piirteistä voidaan toteuttaa kategorisointi (IBM 2020). Tässä yhteydessä kategorisoinnilla tarkoitetaan luokan määrittämistä kerrokseen syötetylle datalle.

2.3 CNN-Malleja

CNN:ien kehitys on ollut varsin nopeaa vuoden 2012 jälkeen, jolloin AlexNet voitti ImageNet Large Scale Visual Recognition Challenge (ILSVRC) -kilpailun (Stanford Vision Lab 2012).

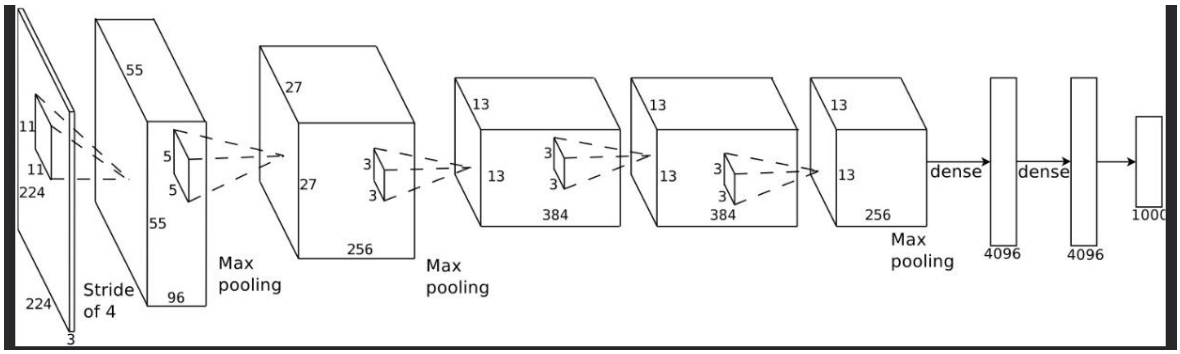


Kuvio 1. CNN-mallien tarkkuuden kehittyminen (Kang ym. 2020)

ILSVRC on vuosittainen konenäkökilpailu, jossa on yleensä muutamia eri kategorioita, kuten kuvien luokittelua ja lokalisointia. Kilpailu on yksi alan arvostetuimmista ja sen järjestävät professorit muun muassa Yhdysvaltojen Stanfordin ja Princetonin yliopistoista. Kilpailua sponsoroivat suuret teknologiayhtiöt, esimerkiksi Google ja Nvidia. (ImageNet 2021.)

2.3.1 AlexNet

AlexNet oli ensimmäinen CNN:iin perustuva neuroverkko, joka voitti ILSVRC-kilpailun. AlexNet voitti edellisen vuoden parhaan mallin yli kymmenen prosenttia pienemmällä virheellä. (Kang ym. 2020, 5.) AlexNetin arkkitehtuuri on nykypäivän malleihin verrattuna melko yksinkertainen; se sisältää vain viisi konvoluutiokerrosta, kolme Max Pooling - ja kaksi Fully Connected -kerrosta. (Krizhevsky ym. 2012.)



Kuvio 2. AlexNetin arkkitehtuuri (Krizhevsky ym. 2012)

2.3.2 VGGNet

VGGNet (Visual Geometry Group) voitti ILSVRC:n lokalisaatiokilpailukategorian ja sijoittui toiseksi klassifiointikategoriassa vuonna 2014 GoogLeNetin jälkeen (Das 2017). Vaikka VGGNet sijoittuikin vasta toiseksi klassifioinnissa, voidaan todeta sen vaikutuksen olleen konenäön kehityksen kannalta suurempi kuin GoogLeNetin. Google Scholarin mukaan VGGNetin tutkimusta on siteerattu yli 81 000 kertaa ja GoogleNetin hieman yli 40 000 kertaa. VGGNet tutki hyvin pienten konvoluutiokerrosten Filterien kokoa, kuten 3 kertaa 3 pikseliä, ja samalla syvensi CNN:iä merkittävästi. Klassifioinnissa parhaiten menestynyt, 19 kerrosta syvä VGG19, saavutti kahdeksan prosentin Top-5 virheen. VGG19 sisältää 19 minimoitavaa kerrosta, joista 16 on konvoluutiokerroksia ja 3 viimeistä on Fully Connected -kerroksia (Simonyan & Zisserman 2015, 2-3).

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

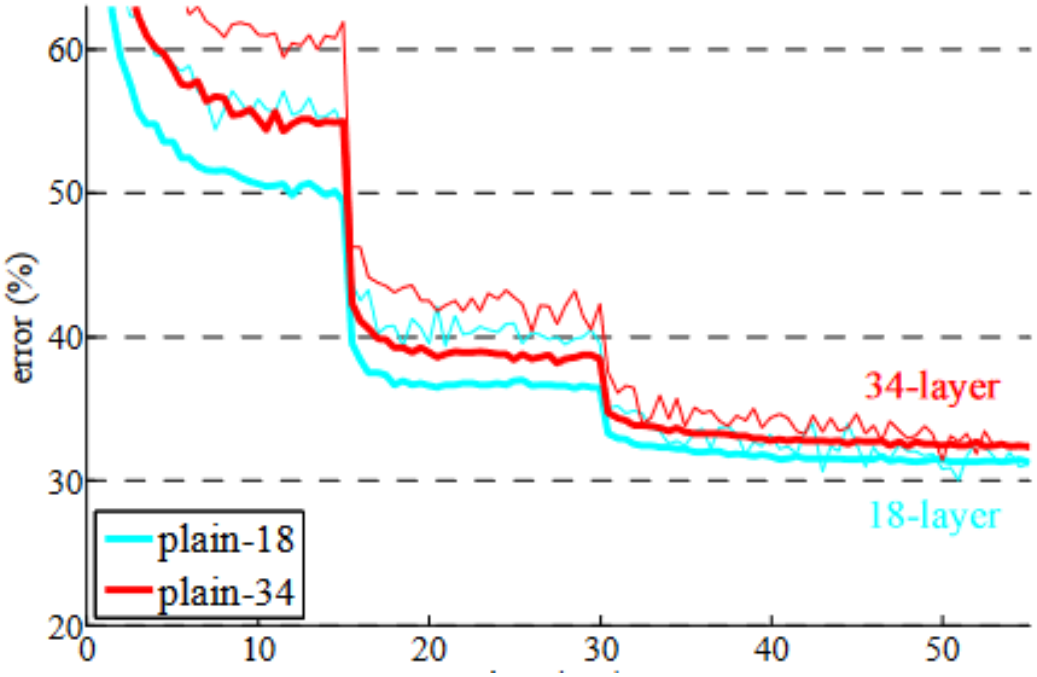
Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Taulukko 1. Eri VGG-arkkitehtuureja (Simonyan & Zisserman 2015)

2.3.3 ResNet

Vuoden 2015 ILSVRC:n voittaja oli neuroverkkoarkkitehtuuri nimeltä ResNet, joka voitti sekä lokalisaatio-, että klassifiontikategoriat. Se ratkoi myös monia ongelmia liittyen todella monikerroksisiin CNN:iin, kuten parametrien määrän kasvamiseen jokaisen lisätyn kerroksen jälkeen. Lisäksi se auttoi ylisovittamisen estämisessä. ResNetin tutkimuksessa oletetaan, että syvemmän neuroverkkoarkkitehtuurin pitäisi kyetä ainakin yhtä pieneen virheeseen kuin matalamman arkkitehtuuri, sillä ylimääräisten kerrosten pitäisi pystyä kopioimaan tarvittava informaatio syvemmille kerroksille. Tutkimuksessa kuitenkin havaittiin, että lisätyt kerrokset heikentävät useimmiten tarkkuutta. (He ym. 2015, 1-5.)



Kuvio 3. 18 ja 34:n kerroksen CNN virheprosentti ImageNet-dataset:iin sovitettuina (He ym. 2015, 5)

ResNet toi ongelman ratkaisemiseksi neuroverkkoihin täysin uudenlaisen lähestymistavan: residuaalisen lohkon. Residuaalisessa lohkossa lohkon syöte lisätään lohkon tulosteeseen (Ng 2017a). Lisäyksen ansiosta lohko estää syvyyden aiheuttamaa ylisovitusta mahdollistamalla informaation siirtymisen syötteestä suoraan tulosteeseen.

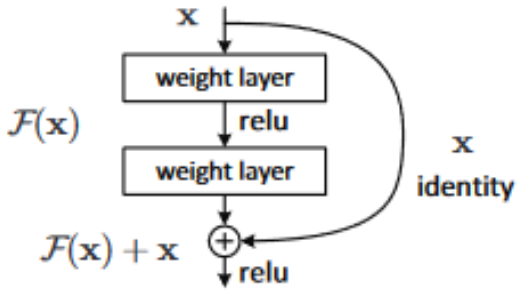
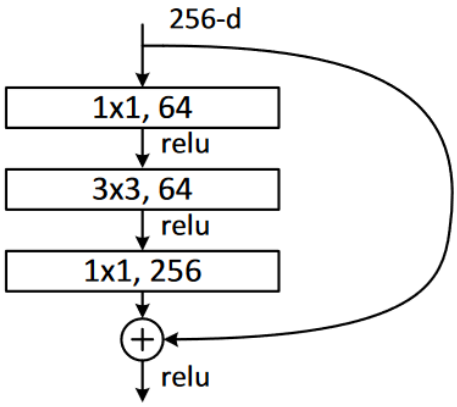


Figure 2. Residual learning: a building block.

Kuvio 4. Residuaalinen lohko (He ym. 2015, 2)

Residuaalisen lohkon lisäksi ResNetin tutkimuksessa esiteltiin myös Bottleneck-lohko, jonka tehtävä on mahdollistaa kahden syvyysdimensioltaan eroavan kerroksen summaaminen ja korkeamman syvyysdimension syötteen projisointi pienempään dimensioon sovitettavien parametrien määrän minimoimiseksi. Bottleneck-lohkossa syöte ajetaan 1x1 konvoluutiokerroksen läpi, laskemalla syötteestä N-määrä Filtereitä. Täten Bottleneck-lohkoa hyödyntäen kaksi eri syvyysdimension piirrematriisia voidaan summata lohkon viimeisessä kerroksessa. Bottleneck-lohkoa käytetään tutkimuksessa myös pienentämään syötteen dimensioita hetkellisesti lohkon sisällä, jotta laskennallisesti suurten 3x3 konvoluutiokerrosten parametrimäärät voidaan pitää hallittuina. (He ym. 2015, 6).



Kuvio 5. Bottleneck-lohko (He ym. 2015, 6)

ResNet saavutti vuoden 2015 ILSVRC -kilpailussa 3,6 prosentin top-5 virheen klassifiointi-kategoriassa, mikä oli merkittävä parannus aiempaan 6,7 prosenttiin, johon VGGNet pystyi edellisenä vuonna. Tämä oli merkittävää, koska ResNet-50 -malli sisälsi vain noin 25,6 miljoonaa parametriä ja oli 107 kerrosta syvä, kun taas VGG19 oli 19 kerrosta syvä ja sisälsi noin 143,7 miljoonaa parametriä (Keras 2022).

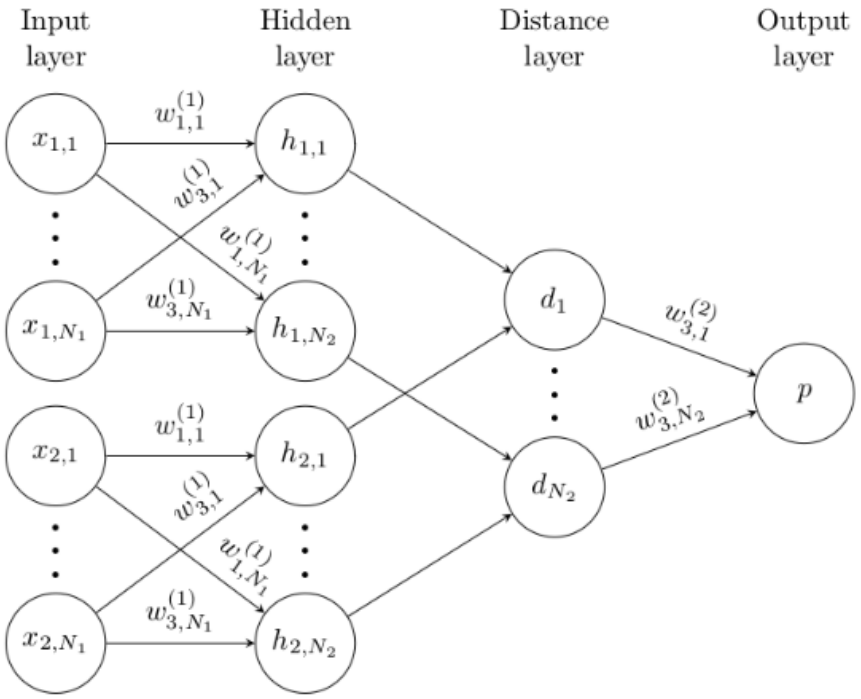
ResNetin tutkimuksessa kokeiltiin myös erittäin syvää arkkitehtuuria 1 202:lla kerroksella. Arkkitehtuurissa ei havaittu ongelmia sovituksessa, mutta testissä todettiin, että 1 202:n kerroksen arkkitehtuurin tarkkuus oli huonompi kuin 110-kerroksisen arkkitehtuurin. (He ym. 2015, 8).

ResNetin tutkimuksessa kehitetty residuaalinen lohko on paljon käytetty myös monissa muissa arkkitehtuureissa. Tunnettuja arkkitehtuureita, joissa esiintyy residuaalisia lohkoja ovat muun muassa: YOLO, SENet ja MobileNet (Redmon & Farhadi 2018, 2; Hu ym. 2019, 4; Sandler ym. 2019, 5).

3 Siamilainen neuroverkko

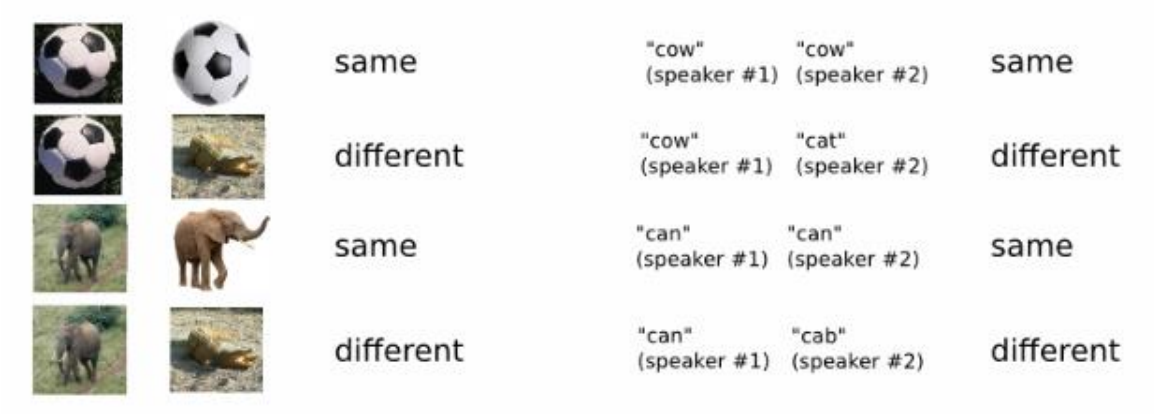
3.1 Similariteetti

Yleisesti neuroverkoissa hyödynnetään yhtä syötettä ja mallia, jonka tarkoitus on tehdä jokin ennuste syötteestä, kuten luokitella kuva. Siamilaisessa neuroverkossa sen sijaan tulkitaan kahden tai useamman syötteen samankaltaisuutta. Yhden mallin ennusteen sijaan voidaan hyödyntää kahta tai useampaa. Mallit ovat identtisiä keskenään ja niillä on samat parametrit (Koch ym. 2015, 4). Molempien mallien viimeinen kerros yhdistetään etäisyyskerroksessa, jonka pohjalta tehdään yksi johtopäätös: kuinka samanlaisia syötteet ovat.



Kuvio 6. Siamilainen neuroverkko (Koch ym. 2015, 4)

Opetusvaiheessa malleihin syötetään positiivisia ja negatiivisia pareja. Positiiviset parit ovat datapareja, jotka kuuluvat samaan kategoriaan ja joiden piirteiden etäisyys etäisyyskerroksessa pyritään minimoimaan. Negatiiviset parit sen sijaan kuuluvat eri kategorioihin ja näiden piirteiden välisen etäisyys pyritään maksimoimaan (Chopra ym. 2005, 3).



Kuva 3. Esimerkki siamilaisen mallin sovitusdatasta (Koch ym. 2015)

Parin ensimmäinen datapiste syötetään yhteen sisääntuloista ja toinen toiseen, jonka jälkeen mallien tulosteista lasketaan etäisyys. Etäisyys voidaan laskea esimerkiksi L1-etäisyysmetriikkaa hyödyntämällä (Koch ym. 2015, 3-4). Tappiofunktiona siamilaisessa neuroverkossa voidaan käyttää esimerkiksi Binary Cross-entropy -lossia, jota käytetään yleisesti binäärisissä luokitteluongelmissa, joissa toivottu tulos on joko 0 tai 1 (TensorFlow 2022b).

Siamilainen neuroverkko oppii tunnistamaan eroavaisuuksia syötteiden välillä ja täten mallia voidaan käyttää tunnistamaan sisältävätkö syötteet samoja piirteitä. Käytännössä samanlaisuuden tunnistamisen herkkyyttä voidaan hienosäätää marginaalilla, jonka alle neuroverkon similariteettiarvon pitää sijoittua, jotta syötteet tulkitaan samoiksi.

3.2 N-shot learning

Siamilaisia neuroverkkoja käytetään esimerkiksi tilanteissa, joissa sovitusdata on vähäistä, eli niin kutsutuissa N-Shot learning -tilanteissa. N-Shot learningin etuliite N kuvaa opetusdatan määrää. Esimerkiksi: Few-Shot Learning tarkoittaa, että opetusdataa löytyy vähän, One-Shot Learning puolestaan tarkoittaa, että opetusdataa löytyy vain yksi kappale. On myös olemassa Zero-Shot Learning, missä lopullisen käyttötarkoituksen dataa ei ole olemassa ollenkaan. (Lyashenko 2022.)

3.3 Anchor, Support Set ja K-Way N-Shot

Similariteetin tulkitsemiseksi siamilaisessa neuroverkossa tarvitaan niin sanottu Support Set. Support Set on joukko esimerkkejä, joita syötetään siamilaisen mallin toiseen syötepäähän. Toiseen syötepäähän syötetään niin sanottu Anchor, eli se data, jonka similariteetti Support Setin kanssa halutaan laskea. K-Way N-Shot kuvaa Support Setin kokoa. K kuvaa Support Setin kategorioiden lukumäärää, eli esimerkiksi 10-Way tarkoittaa, että Support

Setissä on 10 eri vertailukategoriaa. (Shusen 2020a.) N-Shot taas tarkoittaa kuinka monta esimerkkiä jokaisesta luokasta Support Setissä on. Täten esimerkiksi 10-Way 5-Shot tarkoittaa Support Settiä, jossa on kymmenen eri luokkaa joista jokaisessa on viisi vertailuesimerkkiä. (Shusen 2020a.)

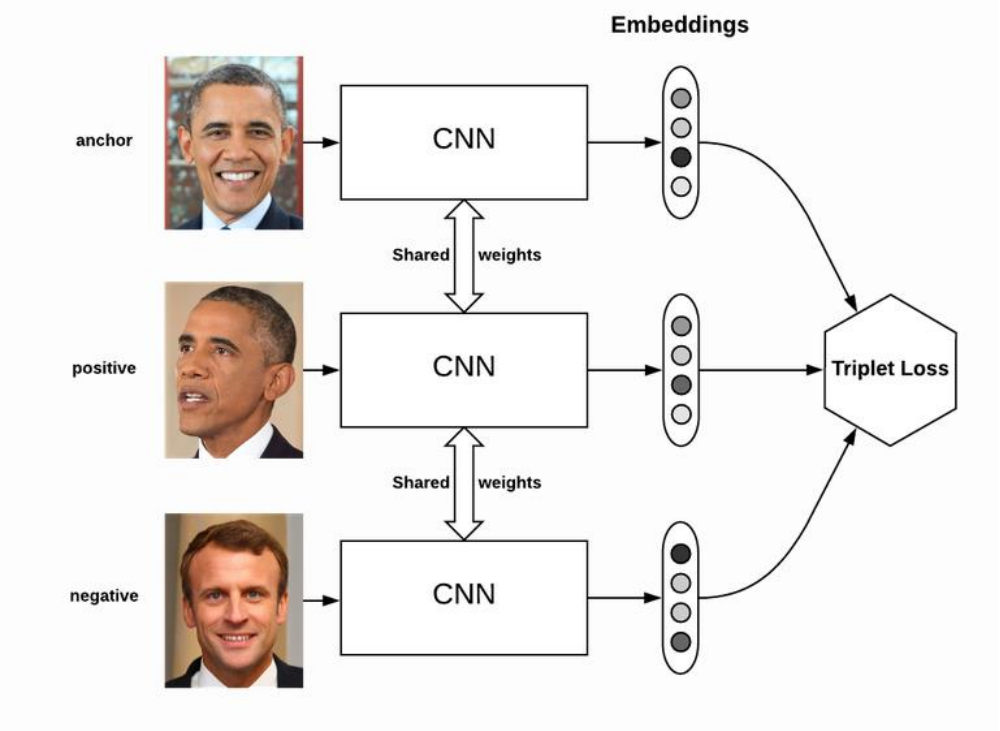
Kuva 4. 3-Way 2-Shot Support Set (Shusen 2020b)

3.4 Triplet Loss

Siamilaisissa neuroverkoissa piirrevektorit lasketaan kahdelle tai useammalle syötteelle, joiden välinen etäisyys lasketaan etäisyysfunktiolla. Kyseinen etäisyys pyritään minimoimaan, minkä seurauksena eri luokkiin kuuluvat piirrevektorit voivat päätyä hyvinkin lähelle toisiaan piirrevektorien koordinaatistossa. Tämä voi vaikeuttaa similariteettilaskentaa ja aiheuttaa vääriä tuloksia.

FaceNetin tutkimuksessa esitetyn Triplet Lossin on tarkoitus parantaa siamilaisen neuroverkon tarkkuutta minimoimalla samaan luokkaan kuuluvien piirrevektorien etäisyyttä ja maksimoimalla eri luokkiin kuuluvia (Schroff ym. 2015, 3).

Triplet Lossia varten siamilaiseen neuroverkkoon tarvitaan myös kolmas syötepää. Kolmanteen päähän syötetään esimerkkejä negatiivisista luokista. Tätä kolmen datapisteen syötettä kutsutaan tripletiksi.



Kuvio 7. Esimerkki kolmipäisestä siamilaisesta neuroverkosta (Moindrot 2018)

FaceNetin tutkimuksessa esitetty Triplet Loss on kaavassa 1. Tappiofunktion ensimmäinen termi $f(x_i^a) - f(x_i^p)$ pyrkii minimoimaan Anchorien ja positiivisten parien etäisyyden, ja toinen termi $-f(x_i^a) - f(x_i^n)$ pyrkii maksimoimaan Anchorien ja negatiivisten parien etäisyyden. Marginaali α on asetettu etäisyys eli kuinka kaukana toisistaan positiiviset ja negatiiviset dataparit pyritään pitämään (Schroff ym. 2015, 3).

$$\sum_i^N \left[\left\| f(x_i^a) - f\left(x_i^p\right) \right\|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + \alpha \right] \quad (1)$$

3.4.1 Triplettien kategorointi

Satunnaisesti valitut tripletit voivat sisältää useita triplettejä, joissa Anchorin ja positiivisen parin piirrevektorit ovat jo lähellä toisiaan ja Anchorin ja negatiivisen pari piirrevektorit kaukana toisistaan. Nämä tripletit tuottavat vain pienen virheen ja hidastavat mallin sovitusta.

Tämän estämiseksi FaceNetin tutkimuksessa esitetään kolme kategoriaa tripleteille: Easy, Hard ja Semi-Hard. (Schroff ym. 2015, 3.)

Easy, eli helpot tripletit, sisältävät kaikki tripletit, joissa positiivinen pari on lähellä Anchoria ja negatiivinen pari kaukana Anchorista, eli kyseisten triplettien erottelu on helppoa. Hard, eli vaikeat, tripletit viittaavat tripletteihin, joissa Anchorin ja positiivisen parin piirteet ovat otannassa kauimpana toisistaan, sekä Anchorin ja negatiivisten parien piirteet mahdollisimman lähellä toisiaan. Semi-hard, eli vaikeahkot tripletit, viittaavat tripletteihin, joissa Anchor ja positiivinen pari ovat lähempänä toisiaan kuin Anchor ja negatiivinen pari, mutta negatiivinen pari on α:n, eli marginaalin, päässä Anchorista. (Schroff ym. 2015, 3-4.)

Tutkimuksesta ilmenee, että kasvojen luokittelussa Semi-Hard tripletit toimivat mallin sovituksessa parhaiten (Schroff ym. 2015, 4).

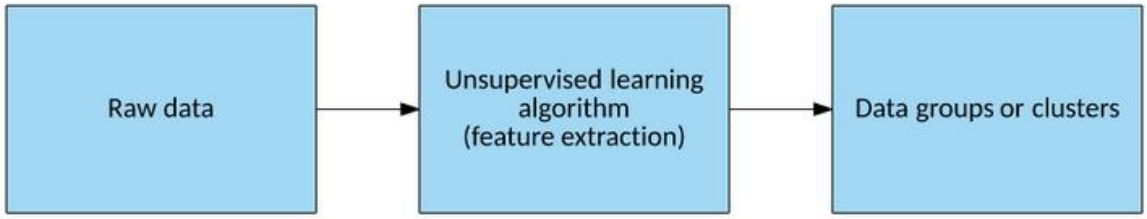
3.4.2 Triplettien valinta

FaceNetin tutkimuksessa esitetään kaksi tapaa valita Semi-Hard ja Hard triplettejä: Offline ja Online. Offline-valinta merkitsee, että mallin viimeisimpiä parametrejä hyödyntäen valitaan sovitusdatasta uudet Hard tai Semi-Hard tripletit jokaisen n-sovitusvaiheen jälkeen. Online-valinta merkitsee, että sovitukseen käytettävät tripletit lasketaan jokaisesta Batchista eli erästä. (Schroff ym. 2015, 3.)

4 Automaattinen luokittelu

4.1 Ohjaamaton oppiminen ja luokittelu

Ohjaamattomalla oppimisella tarkoitetaan koneoppimismenetelmää, jossa mallin sovituksessa käytetään merkitsemätöntä dataa (engl. unlabeled data). Neuroverkkoja voidaan sovittaa erottelemaan syötteestä piirrevektoreita näyttämättä oikeaa kategoriaa mallille sovitusvaiheessa. Hyvin eritellyt piirteet voivat olla erittäin hyödyllisiä esimerkiksi poikkeamien tunnistamisessa ja hyvin eroteltavia piirteitä voidaan myös luokitella. (IBM 2017.)



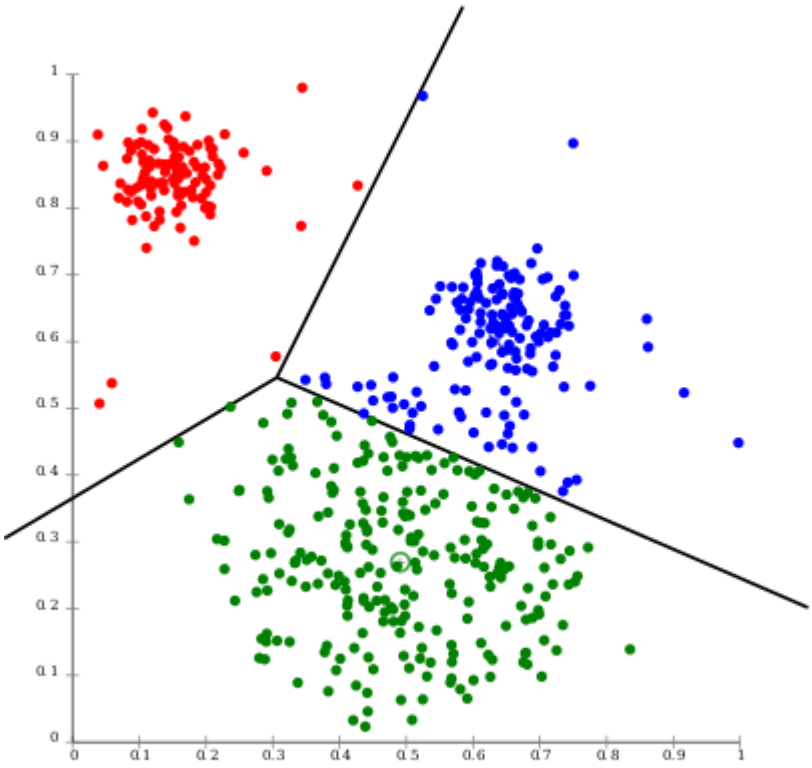
Kuva 5. Datan klusterointi ohjaamattomalla oppimisella (IBM 2017)

4.2 K-Means

K-Means on luokittelualgoritmi, jolla voidaan klusteroida havaintoja-arvoja hyödyntämällä niiden etäisyyksiä toisistaan (Piech 2017).

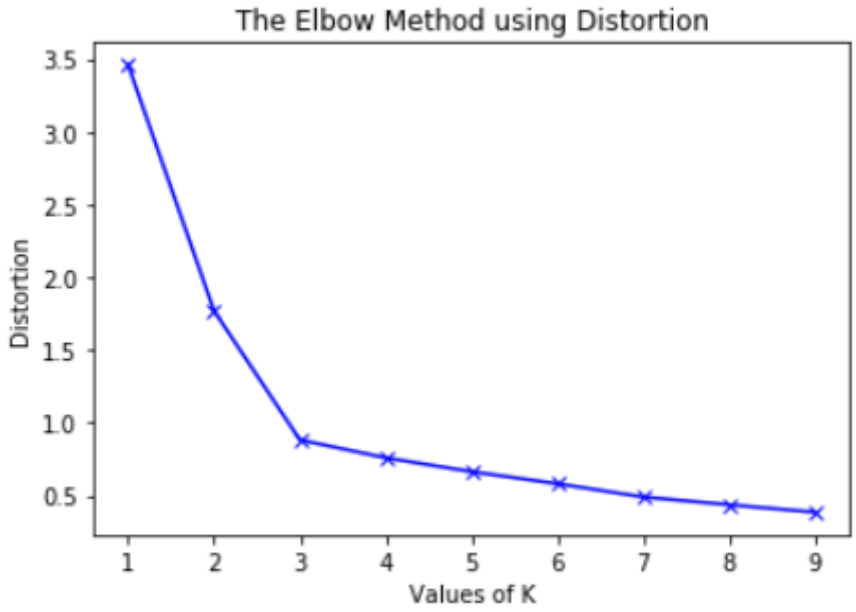
K-Means algoritmi toimii seuraavasti:

1. Asetetaan haluttujen luokkien lukumäärä K.
2. Asetetaan luokkien sentroidit satunnaisesti.
3. Jokaiselle havainto-arvolle lasketaan lähin sentroidi, ja piirre lisätään sen luokkaan.
4. Jokaiselle luokalle lasketaan uusi sentroidi, joka on kaikkien luokkaan kuuluvien havaintoarvojen keskipiste.
5. Kohtaa 3 ja 4 toistetaan, kunnes muutoksia ei enää tapahdu. (Lavrenko 2014.)



Kuvio 8. Dataa klusteroituna K-Means algoritmillä (Chandha ym. 2018)

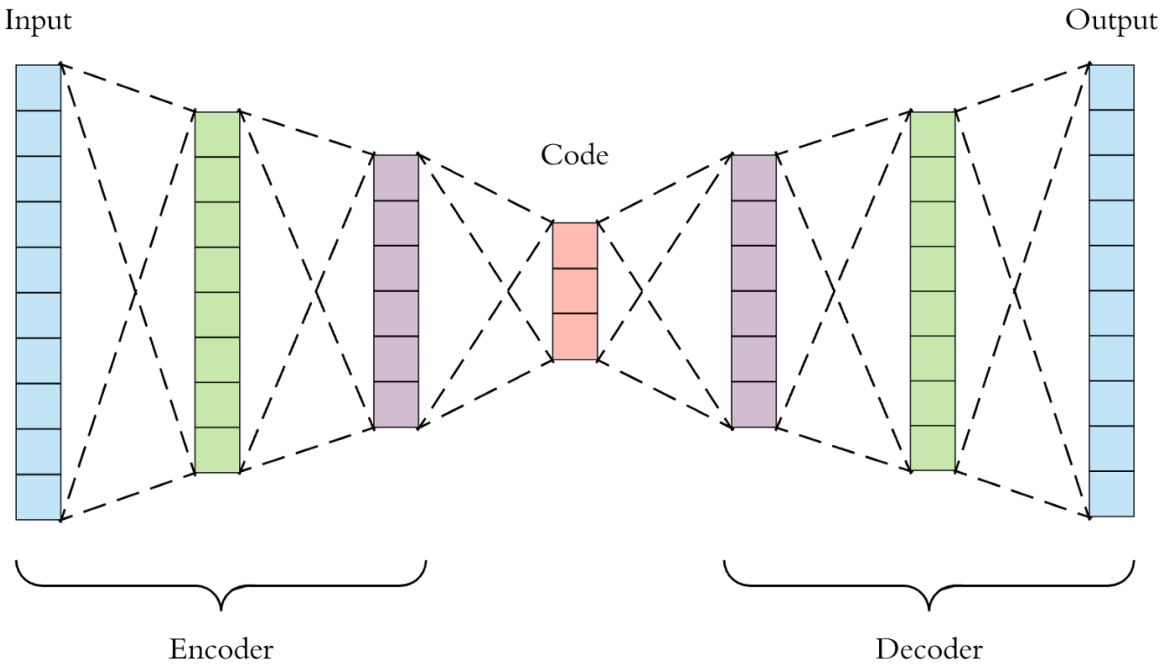
K-Means luokitteluun voi vaikuttaa lähes ainoastaan säätämällä klustereiden lukumäärää eli K:ta. Täten on tärkeää löytää optimaalinen K-arvo. Yksi tyypillinen lähestymistapa on käyttää niin sanottua Elbow-metodia. Elbow-metodissa K-Means luokittelualgoritmi sovitetaan dataan monella eri K-arvolla ja sovitetusta mallista lasketaan mallin Inertia ja Distortion -metriikat. Inertia tarkoittaa piirrevektorien ja niille lähimpien sentroidien välisten pituuksien neliöiden summaa. Distortion taas kaikkien sentroidien välisten pituuksien neliöiden keskiarvoa. Testatut K-arvot ja niiden tuottamat Inertia ja Distortion -metriikat piirretään graafiin, jossa X-akselille piirretään K-arvo ja Y-akselille Inertia ja Distortion. Se kohta, jossa Inertian ja Distortion lasku hidastuu merkittävästi on etsitty Elbow-piste, josta kohdan K-arvo otetaan käyttöön. (Gupta 2022.)



Kuvio 9. Eri K-arvojen Distorion-metriikka viivadiagrammissa (Gupta 2022)

4.3 Autoencoder

Autoencoderit ovat neuroverkkojen alakategoria, joka koostuu kahdesta pääasiallisesta puolesta: Encoderista ja Decoderista. Encoder on se Autoencoderin puoli, jonka tarkoitus on koodata syötteen informaatio. Tyypillisesti Encoderin tuottama koodaus on dimensioiltaan huomattavasti alkuperäistä syötettä pienempi (Zhu & Wang 2019, 3-4). Decoderin tehtävä taas puolestaan on yrittää tuottaa koodauksesta alkuperäinen syöte.



Kuvio 10. Autoencoder (Dertat 2017)

Autoencoderit ovat erinomaisia erittelemään syötteestä sen olennaisia piirteitä. Täten Autoencodereita on käytetty paljon esimerkiksi kuvankäsittelyssä, kuten mustavalkokuvien värittämisessä, kohinanpoistossa ja pakkauksessa (Deshpande ym. 2017, 2; Monn 2017; Alexandre ym. 2019). Lisäksi Autoencodereita käytetään paljon poikkeavaisuuksien tunnistamisessa ja uuden datan generoinnissa (Zhu & Wang 2019, 15; Alfeo ym. 2020).

Autoencoder-malli sovitetaan syöttämällä mallin Encoderille dataa, joka on useimmiten jokin augmentaatio, tai korruptoitu versio syötteestä. Esimerkiksi kuva, jossa on kohinaa, tai mustavalkoinen versio kuvasta. Encoder koodaa syötteen, mikä puolestaan syötetään Decoderille. Decoder yrittää rekonstruoida koodattua dataa haluttuun muotoon, kuten värittää syötettä tai poistaa kohinaa syötteestä. (Monn 2017.)

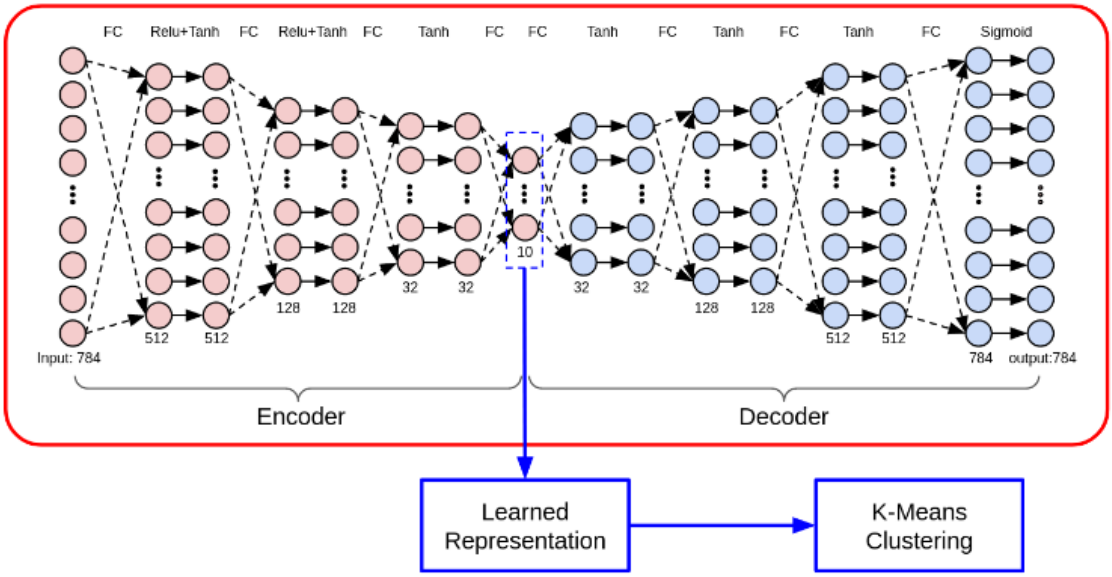
4.3.1 Reconstruction Loss

Decoderin tulosteen ja toivotun datan välistä virhettä kutsutaan Reconstruction Lossiksi. Keskineliövirhe on yleisesti käytetty virhefunktio rekonstruktiovirheen laskemisessa.

4.3.2 Automaattinen luokittelu Autoencoderilla

Autoencoder-malli voidaan sovittaa koodaamaan syötetty data ja tuottamaan koodatusta datasta syötettä vastaava data, eli kopioimaan. Operaatio ei vaadi merkittyä dataa, joten toimenpide voidaan suorittaa täysin ohjaamattomalla oppimisella. Jos Decoder kykenee

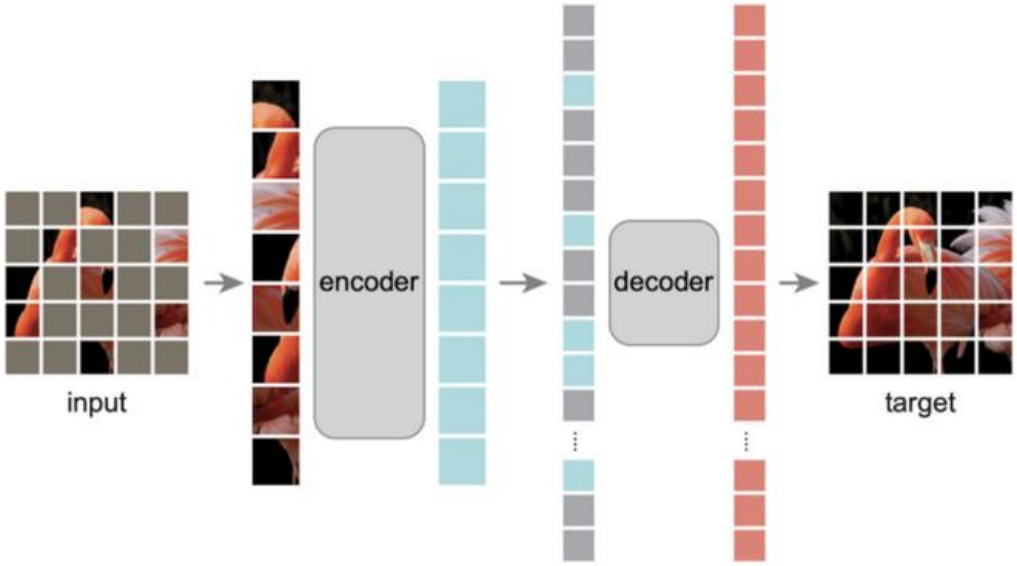
tuottamaan syötettä vastaavan datan Encoderilla koodatusta datasta, voidaan sitä teoriassa klusteroida hyödyntämällä erinäisiä klusterointialgoritmeja. (Lu & Li 2021, 2-4.)



Kuvio 11. K-Means klusterointi Encoderin tuottamista piirrevektoreista (Lu & Li 2021, 3)

4.3.3 Self-Supervised Learning

Autoencodereilla on taipumus löytää hyvin alhaisen tason piirteitä, kuten värejä, tekstuureja tai kontrasteja. Tämän estämiseksi on tehty paljon tutkimustyötä ja kehitetty erilaisia metodeja. Yksi metodeista on niin sanottu Self-Supervised Learning, jossa syötettä augmentoidaan jollain tapaa ja täten Autoencoderia yritetään estää tarttumasta piirteisiin, jotka eivät välttämättä ole olennaisia. Syötteen yleisiä augmentointitapoja Self-Supervised Learningissa ovat muun muassa kuvan pyörittäminen, peilaus ja rajaus. (Gansbeke ym. 2020, 2.)

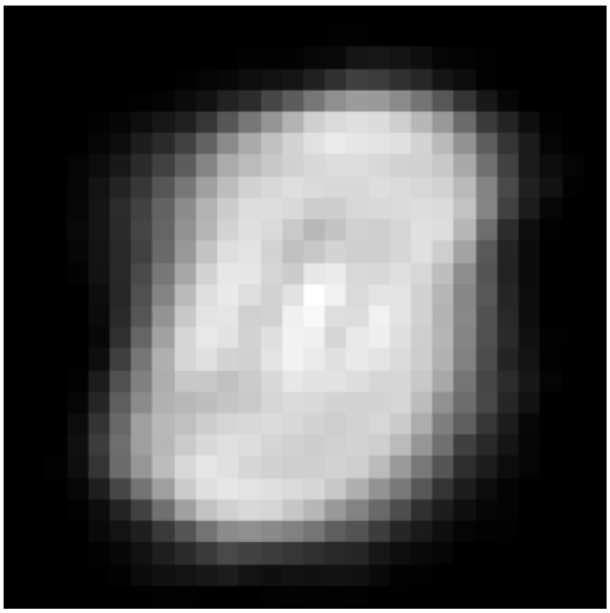


Kuvio 12. Esimerkki Self-Supervised Learningista ja syötteen augmentoinnista (Sick 2021)

Yksi menetelmistä, joilla epäoleellisiin piirteisiin takertumista voidaan estää on DAC (Deep Autoencoder-based Clustering). DAC lisää Autoencoderin sovitusvaiheeseen painoarvo-kartan, jonka avulla syötteen eri elementeille voidaan asettaa painoarvo mallin sovituksen Reconstruction Lossia laskiessa. Autoencodereille tyypillisen keskineliövirheen lisäksi DAC:ssa keskineliövirhe kerrotaan painoarvokartalla ɯ kaavan 2 mukaisesti. (Lu & Li 2021, 3-4.)

$$L_{cmse} = \frac{\sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2}{n} \qquad (2)$$

Tämä ehkäisee mallia kiinnittämästä huomiota esimerkiksi kuvassa sijaitsevien objektien ympäröiviin pikseleihin (Lu & Li 2021, 5).



Kuva 6. MNIST datasetistä laskettu painoarvokartta (Lu & Li 2021, 5)

SCAN (Semantic Clustering by Adopting Nearest neighbours) -metodissa mallin sovittamiseen lisättiin Self-Labeling -vaihe, jossa klusterointialgoritmin klustereita käytettiin vastemerkintänä sovittamaan uusi neuroverkkomalli. Toiminnon ideana on minimoida samaan klusteriin kuuluvien piirrevektorien etäisyys piirteiden koordinaatistossa. (Gansbeke ym. 2021, 6-7.)

5 Transfer learning ja Continual learning

5.1 Transfer Learning

Transfer Learning on koneoppimisen ongelmakenttä, jossa yhteen käyttötarkoitukseen sovitettua mallia sovitetaan uudelleen toiseen käyttötarkoitukseen. Esimerkiksi kissojen tunnistukseen sovitettua CNN:ää voidaan sovittaa uudelleen tunnistamaan koiria koiriin liittyvällä kuvadatasetillä.

Neuroverkoissa Transfer Learning tapahtuu käytännössä siten, että otetaan aikaisemmin sovitettu neuroverkkomalli, jonka viimeinen kerros, jossa luokittelu tapahtuu, sovitetaan uudelleen uuteen dataan. Täten mallin jo oppima funktionaalisuus voidaan yrittää hyödyntää uutta käyttötarkoitusta varten. Riippuen uuden datan määrästä mallin muitakin kerroksia voi yrittää sovittaa uudelleen. Jos uusi käyttötarkoitus poikkeaa vanhasta paljon, voidaan mallin arkkitehtuuriakin muuttaa, kuten lisätä tai poistaa kerroksia. Transfer Learning on suosittu tapa lähestyä ongelmaa, jos lopullista tehtävää varten dataa on vain vähän, mutta muuta samankaltaista tehtävää varten dataa on paljon. (Ng 2017b.)

5.1.1 Pre-Training ja Fine-Tuning

Pre-Training on Transfer Learningiin liittyvä termi, joka tarkoittaa mallin alustamista sovittamalla malli dataan, joka ei vielä ole lopullinen. Fine-Tuning on puolestaan Pre-Trainingin jälkeinen vaihe, jossa malli sovitetaan suorittamaan toista tehtävää uudella datalla. Pre-Trainingin tarkoitus on löytää mallille hyvät lähtöparametrit lopullista käyttötarkoitusta, eli Fine-Tuningia varten. (Ng 2017b.)

Kyseinen lähestymistapa on yleinen, kun Fine-Tuning osuuteen liittyvää dataa on vähän, mutta sopivan Pre-Training tehtävän dataa on paljon. Esimerkiksi CNN-neuroverkkomalli voidaan sovittaa Pre-Training vaiheessa kategorioimaan erilaisia objekteja kuvista, hyödyntäen suuria avoimen lähdekoodin datasettejä. Malli voidaan sitten jatkosovittaa Fine-Tuning vaiheessa merkittävästi pienempään datamäärään hyödyntäen Pre-Training vaiheen optimoituja parametrejä.

5.2 Continual Learning

Continual Learning on Transfer Learningin alakategoria, jossa mallia sovitetaan uuteen dataan säilyttäen mallin vanha toiminnallisuus, johon lisätään uutta toiminnallisuutta. Esimerkiksi klassifiointineuroverkon täysin uudelleensovittamisen sijaan pyritään sovittamaan malli, joka kykenee kategorioimaan vanhojen luokkien lisäksi myös uusia luokkia. (Papers With Code.)

5.2.1 Catastrophic Forgetting

Catastrophic Forgetting on Continual Learningin olennainen ongelma, jossa mallia sovitettaessa uuteen toiminnallisuuteen, malli ei enää kykene suorittamaan aiempaa toiminnallisuutta (Gido & Andreas 2019, 1). Ongelmaa voidaan yrittää ehkäistä esimerkiksi yhdistämällä sovitusdataa vanhasta ja uudesta datasetistä, ja jäädyttämällä valmiiksi sovitettuja parametrejä (Li & Hoiem 2017, 4).

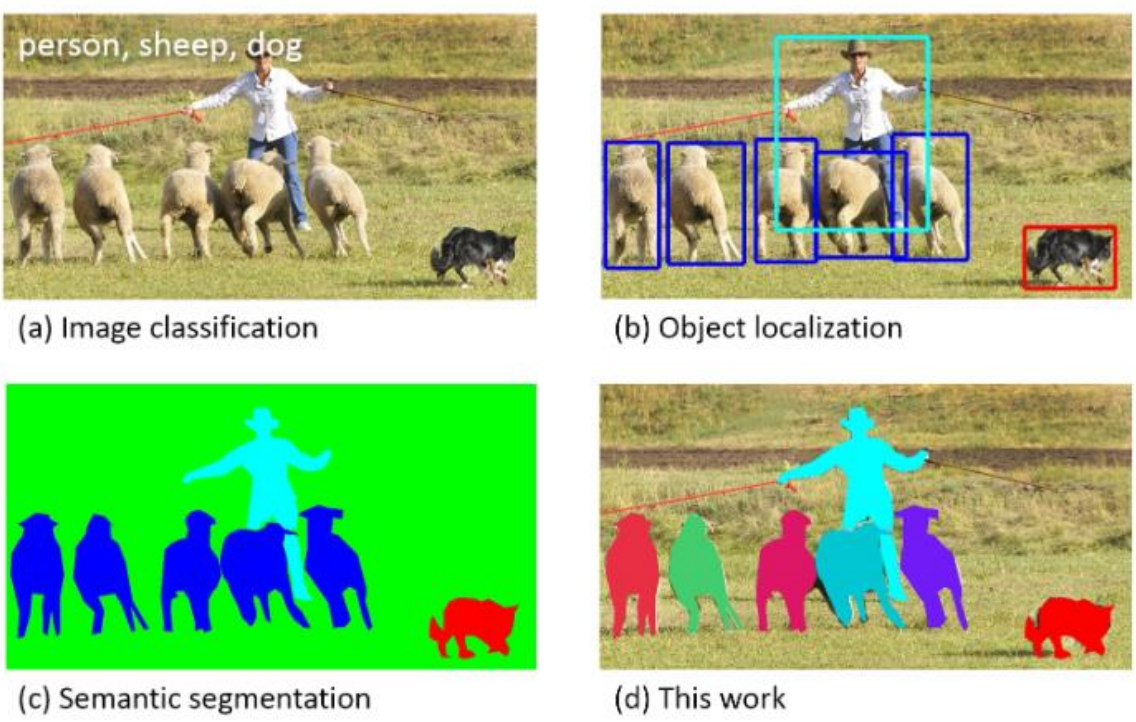
6 Sovitus- ja validointidata

6.1 COCO 2017

Tässä opinnäytetyössä eri neuroverkkomallien sovittamiseen käytettiin COCO 2017 kuvadatasettiä. COCO on Microsoftin, Googlen ja Facebookin sponsoroima ja ylläpitämä kuvadatasetti (Common Objects in Context). Nimi COCO on akronyymi sanoista Common Objects In Context. Numero 2017 merkitsee vuosilukua, johon asti kuvadataa on kerätty ja annotoitu. Annotointi tarkoittaa datan oikeiden vasteiden, kuten kuvasta löytyvien luokkien merkitsemistä. Kuvadatan keräys aloitettiin vuonna 2014 ja sitä hyödyntäen on järjestetty vuosittain kilpailuja esimerkiksi kuvantunnistuksessa, lokalisoinnissa ja segmentoinnissa.

COCO:n kuvadata on vuodesta riippumatta pitkälti samaa, mutta annotaatioita on vuosien varrella lisätty kuviin paljon. COCO 2017 kuvadatapankki sisältää 123 287 kuvaa ja 886 284 annotaatiota. COCO:n itse keräämän kuvadatasetin lisäksi siihen on liitetty PASCAL VOC kuvadatasetti (Lin ym. 2015, 2).

Vuoden 2017 COCO datasetti koostui kolmesta eri annotaatiosta jokaista kuvaa kohden: Detection, Keypoints ja Stuff. Vuoden 2018 datasetti on sama kuin vuoden 2017, mutta siihen on lisätty Panoptic-annottaatio. Detection-annotaatio sisältää jokaista kuvaa kohden kaikkien kuvasta löytyvien luokkien sijainnin ja rajauksen sekä segmentoinnin, eli tunnistettavan luokan ääriviivat pistejonona. Keypoints taas sisältää annotaation kuvissa esiintyvien ihmisten asennosta; jalkojen, käsien ja pään sijainnin. Stuff-annotaatio on segmentaatiomerkintä vaikeasti rajattaville kohteille kuten vedelle, taivaalle ja ruohikolle. Panoptic on puolestaan Stuff ja Detectionin segmentointien yhdistelmä. (Lin ym. 2015, 3.)

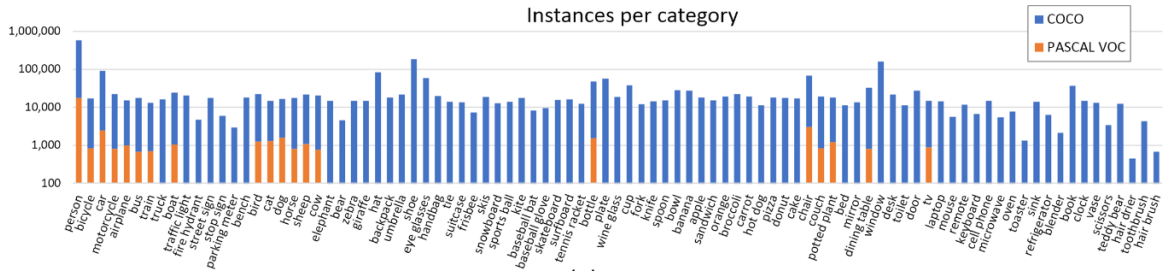


Kuva 7. COCO:n annotaatioita (Lin ym. 2015, 1)

6.2 Datan esikäsittely

Opinnäytetyössä hyödynnetään COCO 2017 kuvadatapankkia ja COCO 2017 tarjoamaa Detection-annotaatiota, joista hyödynnetään kuvista löytyviä luokkia. Datasetistä löytyy 91 kategoriaa COCO:n tutkimuksen mukaan (Lin ym. 2015, 4). Kuvista erotellaan eri kategoriat omiksi kuvikseen niiden rajauksen mukaan. Tämän seurauksena sovituskuvadataa saadaan 860 001 kappaletta ja validointidataa 36 781 kappaletta.

COCO 2017 annotaatioissa on suurta luokkaepätasapainoa. Osasta kategorioita on löydettävissä paljon annotaatioita, osasta hyvin vähän ja joistain ei lainkaan. Luokkaepätasapaino on myös esitetty COCO:n tutkimuksessa kuviossa 13.

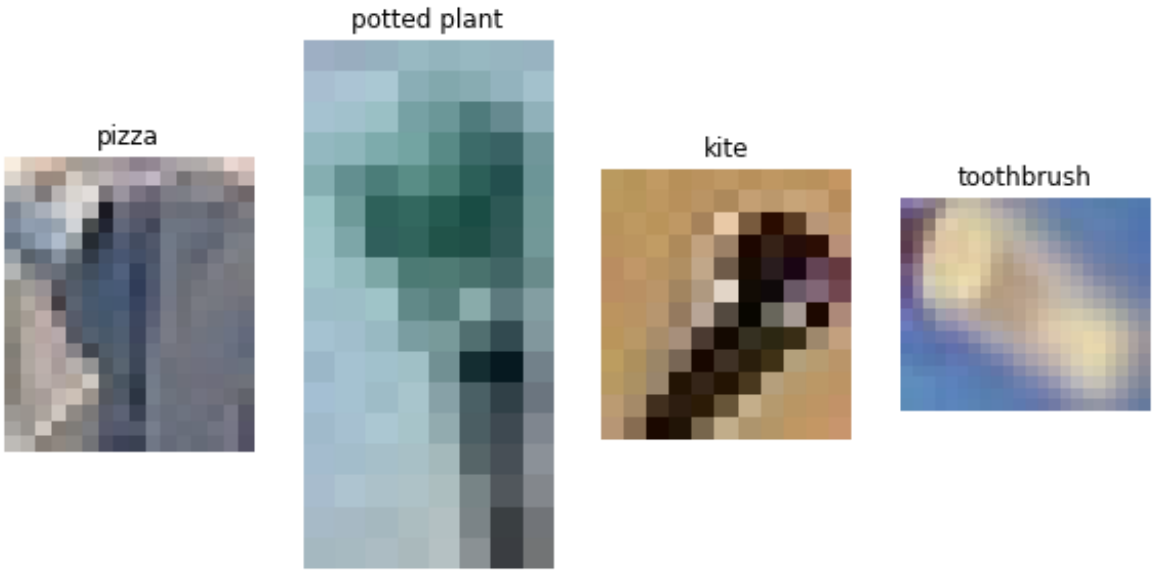


Kuvio 13. Annotaatioiden määrä kategorioittain COCO-datasetissä (Lin ym. 2015, 7)

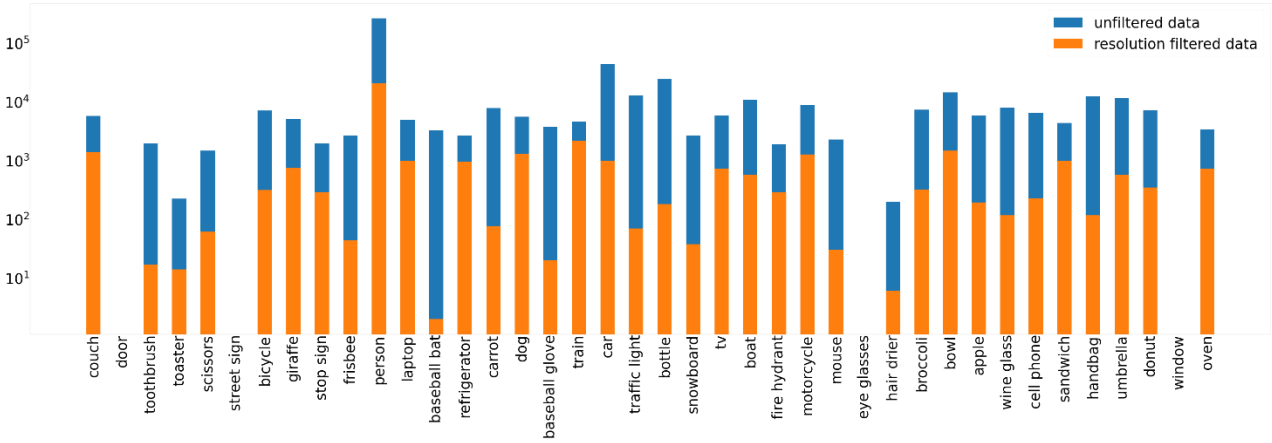
Syy miksi osasta kategorioista ei löydy yhtään annotaatiota on luultavasti se, että niistä ei löydy Detection-annotaatioita. COCO:n tutkimuksessa mainitaan myös, että vuoden 2014 julkaisu on rajoitettu vain 80:een kategoriaan (Lin ym. 2015, 7). Tyhjiä kategorioita ei käsitellä opinnäytetyössä.

Kategoriaepätasapaino korjataan sovitettavasta mallista riippuen Oversamplingia hyödyntämällä. Oversampling on datatieteessä yleisesti käytetty metodi, jossa vähemmän edustetun luokan datapisteitä monistetaan kokonaisdatan tasapainottamiseksi.

COCO 2017 annotaatioissa on myös muutamia esimerkkejä merkittävästi toivottua pienemmästä pikselimäärästä. Osassa annotaatioita tunnistettava objekti kattaa hyvin pienen osan kuvaa ja täten rajattu alue on epäselvä. Tämän vuoksi sovitusdatasta filtteröidään myös annotaatiot, joiden rajausalue on alle 14 400 neliöpikseliä, mikä vastaa minimissään noin 120x120 pikselin kokoista aluetta.



Kuva 8. Esimerkkejä alhaisen resoluution kuvista



Kuvio 14. Otanta alkuperäisestä- ja resoluutiolla filtteröidystä datasta

Lisäksi osa COCO 2017:n rajauksista poikkeaa 1:1 kuvasuhteesta. Siksi kuville joiden rajaukset poikkeavat paljon toivotusta, tehdään replikointioperaatio, jossa reunassa olevaa viimeistä pikseliä kopioidaan täyttämään kuvan puuttuva alue. Tämä auttaa sovitusdataa pysymään oikeassa kuvasuhteessa ja tunnistettavia kategorioita pitämään muotonsa ilman uudelleenskaalauksen aiheuttamaa vääristymää.

Kuva 9. Esimerkkejä replikoiduista kuvista

6.3 Sovitus- ja validointidatan jako

Filtteröinnin jälkeen käyttöön jäi 80 kategoriaa, joissa on 160 493 kappaletta sovitusdataa ja 6 845 kappaletta validointidataa. Koska validointidataa jäi alle toivotun 10 %:n ja validointidatasetistä löytyi vain 76 kategoriaa, osa sovitusdatasta siirrettiin validointidataan. Lopullinen Pre-Training sovitusdata koostui 151 333:stä kuvasta ja 16 005:stä validointikuvasta.

Jatkosovitusta, eli Fine-Tuningia varten COCO:n tarjoaman datan lisäksi otettiin käsittelyyn kuusi täysin uutta kategoriaa, joista etsittiin manuaalisesti 10 sovituskuvaa. Jatkosovitukseen valitut luokat olivat: näytönohjain, prosessori, keskusmuistikampa, F1 kilpa-auto, 3D-

tulostin ja robotti-imuri. Lopullinen työssä käytetty kategoriataulukko sisälsi täten 86 kategoriaa (Liite 1).

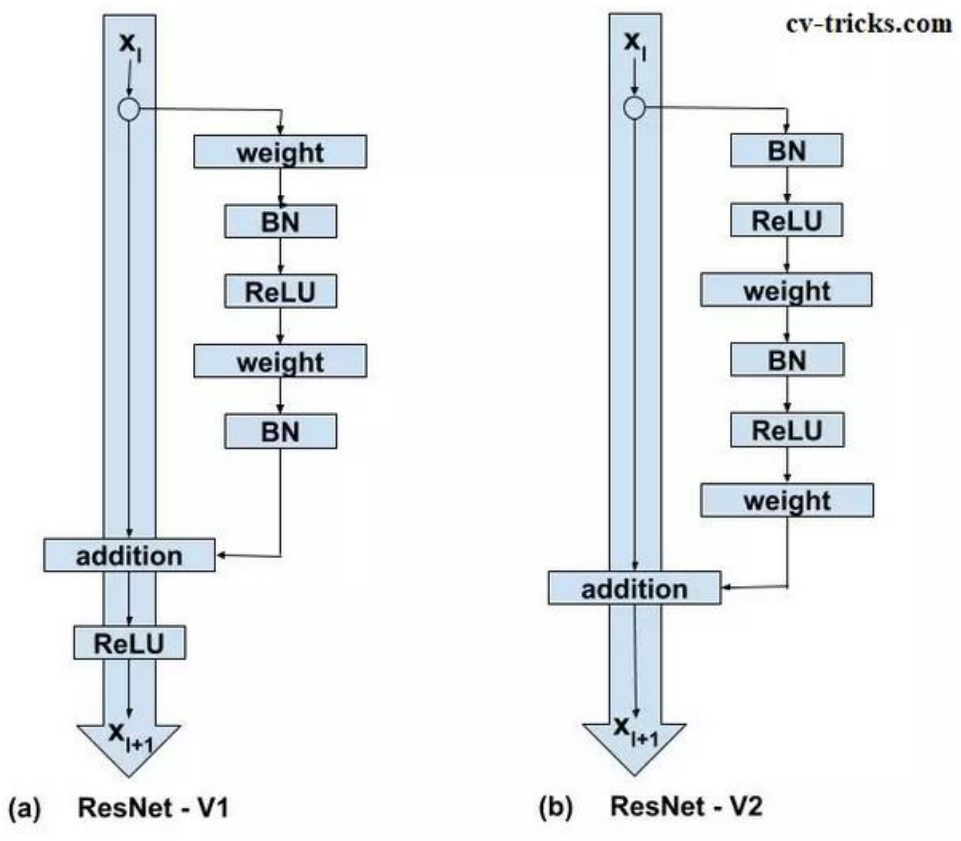
7 Eri arkkitehtuurien testaus

7.1 Testauksen määrittely

Opinnäytetyössä kokeiltiin erilaisia neuroverkkomalleja, joilla pystyy saavuttamaan hyväksyttävän tarkkuuden videoiden sisällön merkitsemisestä. Hyväksyttävä tarkkuus tässä tapauksessa on subjektiivista. Videoiden indeksoinnissa tärkeää on maksimoida oikeat positiiviset tulokset ja minimoida väärät positiiviset tulokset. Se, että osa kartoitettavista luokista jää havaitsematta on vähemmän tärkeää kuin väärien hakutulosten löytyminen. Sovitusaikana käytettiin kuvien augmentointia, eli kuvien sattumanvaraista rotatointia, värimuutoksia ja kirkkauden säätämistä, mikäli sovitettavan mallin nähtiin hyötyvän siitä.

7.2 ResNet50V2

Opinnäytetyön perustana käytettiin Keraksen ResNet50V2 CNN-mallin implementaatiota. ResNet50V2 on päivitetty versio aiemmin mainitusta ResNet50:stä. Batch Normalization ja Rectified Linear Unit (ReLU) -aktivointifunktion paikka residuaalisissa lohkoissa eroaa ResNet50V2:ssa alkuperäisestä. ResNet50:ssä koko residuaalisen lohkon viimeinen aktivointifunktio ajettiin syötteen ja konvoluutiokerrosten elementtikohtaisen addition jälkeen. ResNet50V2:ssa sen sijaan aktivointifunktio ajetaan vain haarautuvien konvoluutiokerrosten jälkeen. (Sachan 2017.) Muutos on esitetty kuviossa 15.

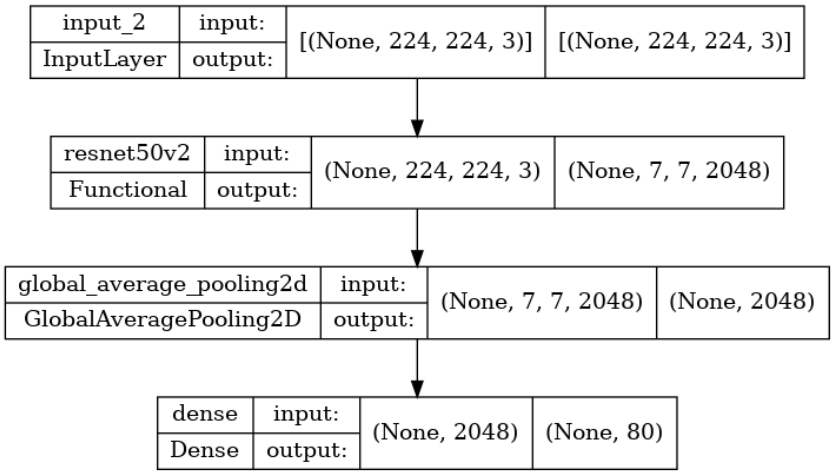


Kuvio 15. ResNet50 ja ResNet50V2 eroavaisuus (Sachan 2017)

ResNet valittiin tutkimuksen perustaksi sen hyvän tarkkuuden vuoksi. Malli on myös kevyt sisältäen vain noin 25,6 miljoonaa parametriä (Keras). Tämän vuoksi kyseistä mallia voitiin hyödyntää myös muissa tutkittavissa malleissa piirteiden erottelussa.

7.2.1 Mallin arkkitehtuuri

Mallin arkkitehtuuri ladattiin Keraksen syväoppimisrajapinnasta. Mallissa hyödynnettiin myös Keraksen tarjoamia ImageNet-datasettiin sovitettuja parametrejä nopeuttamaan sovitusaikaa. Mallista poistettiin sen kategorisointipää, joka ImageNetin dataan sovitettuna sisälsi 1 000 kategoriaa. Tilalle lisättiin kategorisointipää 80 kategorialle.



Kuvio 16. Sovitettava ResNet-malli 80 kategorian kategorisointipäällä

7.2.2 Mallin sovittaminen

Mallin kaikki parametrit sovitettiin uudelleen. Sovituksen virhefunktiona käytettiin Categorical Crossentropyä, joka on yleisesti käytetty virhefunktio malleille, joiden tarkoituksena on kategorisoida dataa (TensorFlow 2022c).

Sovitus- ja validointiaikana käytettiin 32-kokoisia Batchejä. Optimoijana käytettiin Adam optimointialgoritimiä ja sen oppimisnopeudeksi (Learning Rate) asetettiin 0.001. Mallia sovitettiin 10 Epochia, joista jokaisessa käytettiin 262 144 sattumanvaraisesti valittua ja augmentoitua kuvaa. Kuvat valittiin niin, että jokaisen haettavan kuvan kohdalla valittiin sattumanvarainen kategoria ja sattumanvaraisen kategorian kuvakansiosta sattumanvarainen kuva. Näin mitätöitiin COCO 2017 datasetin kategorioiden välinen epätasapaino. Validoinnissa käytettiin aina kaikki 16 005 validointikuvaa ilman augmentointia.

Sovitusvaiheessa seurattiin mallin validointidatan virhe-, Accuracy- ja Precision -metriikoita, sekä oikeiden positiivisten ja väärien positiivisten määriä. Kaikkien 10 Epochin jälkeen paras malli tallennettiin kyseisten metriikoiden perusteella.

Loss, eli virhe, on mallin tulostaman kategorioinnin todennäköisyysjakauman ja oikean todennäköisyysjakauman välinen etäisyys.

Accuracy, eli virheettömyys, puolestaan lasketaan kaavan 3 mukaan (Google 2022a).

$$Accuracy = \frac{Oikeat\ positiiviset + Oikeat\ negatiiviset}{Oikeat\ positiiviset + Oikeat\ negatiiviset + Väärät\ positiiviset + Väärät\ negatiiviset} \quad (3)$$

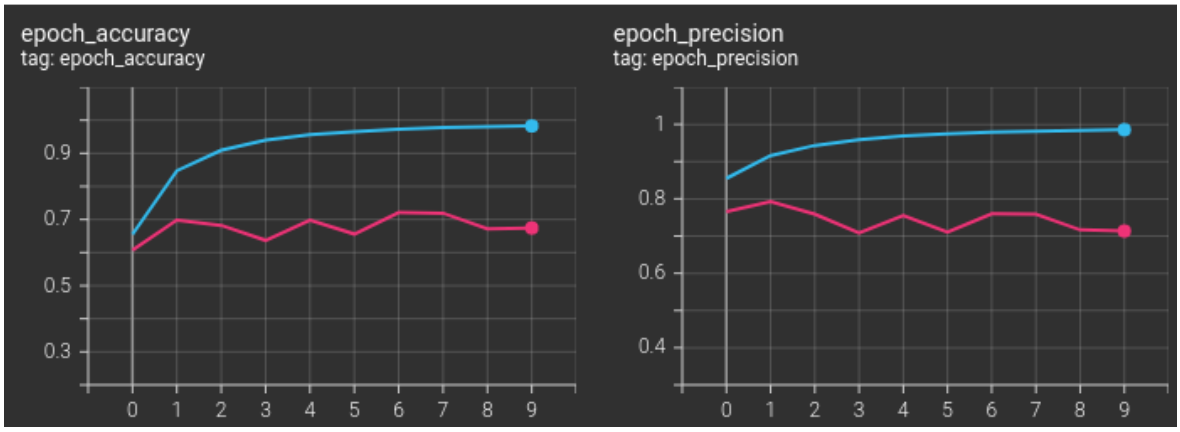
Accuracy ilmaisee sen, kuinka hyvin malli suoriutuu kaikista luokista (Gad 2020), mutta Accuracy-metriikka voi nousta nopeasti, jos kategorioita on paljon, sillä oikeiden negatiivisten summa kasvaa kategorioiden määrän kasvaessa (Google 2022a).

Täten on hyvä myös seurata Precision-metriikkaa, eli positiivisen testituloksen ennustearvoa, joka lasketaan kaavan 4 mukaan (Google 2022b).

$$Precision = \frac{Oikeat\ positiiviset}{Oikeat\ positiiviset + Väärät\ positiiviset} \quad (4)$$

Precision on tutkimuksen käyttötarkoituksen kannalta tärkeä metriikka, sillä se kertoo mallin kategorioinnin relevanssista (Northeastern University 2015). Precision puolestaan ei kärsi kategorioiden määrän kasvaessa, kuten Accuracy, sillä se ottaa huomioon vain oikeat positiiviset ja oikeat negatiiviset vastaukset. Precision-metriikka vaikutti yhteensopivimmalta, sillä väärät positiiviset luokittelut merkitsisivät hakukoneen toiminnassa sitä, että väärät luokat päätyisivät hakutuloksiin. Hakukoneen kannalta relevanssi on tärkeämpää kuin oikeiden hakutuloksien poisjääminen.

Ensimmäisen sovituksen parhaaksi malliksi osoittautui Epoch 7, jossa mallin Accuray validointidatassa oli 72,08 %, Precision 75,97 % ja Loss 1,4172. Vaikka Epochilla 2 Precision-metriikka oli hiukan parempi: 79,25 %, valittiin Epoch 7 parhaaksi malliksi, koska sillä saatiin oikeita positiivisia vastauksia 11 284 kappaletta ja vääriä 3 569. Epochilla 2:lla oikeita positiivisia saatiin 10 280 ja vääriä 2 691. Accuracystä laskettiin virhesuhde vähentämällä 100 %:sta Accuracyn lukema. Täten virhesuhteeksi saatiin 27,92 %. Virhesuhde on hieman huonompi kuin ResNetin tutkimuksessa esitetty 22,85 % ImageNet-datasetissä (He ym. 2015).



Kuvio 17. Ensimmäisen sovitusvaiheen tulokset. Sinisellä sovitusdatan ja punaisella validointidatan metriikat.

Validoinnissa parhaiten toiminutta mallia testattiin vielä ajamalla validointidata mallin läpi ja kirjaamalla oikeat positiiviset-, sekä väärät positiiviset -vastaukset valitsemalla mallin viimeisestä luokittelukerroksesta suurimman arvon tulostanut indeksi ja vertaamalla sitä oikean kategorian indeksiin. Toimenpiteen suoritettua saatiin laskettua Confusion Matrix ja testimetriikat. Mallia testatessa kirjattiin 11 373 oikeaa positiivista luokittelua ja 4 632 väärää positiivista luokittelua. Täten testin Precision-arvoksi saatiin 71,06 %. Confusion Matrixista voidaan todeta, miten malli kykenee tulkitsemaan eri kategorioita. Confusion Matrixissa Y-akselilla esitetään kuvan oikea kategoria ja X-akselilla mallin luokittelema kategoria. Täydellisessä luokittelussa Confusion Matrixiin piirtyisi selkeä diagonaalinen viiva vasemmalta ylhäältä oikealla alas.

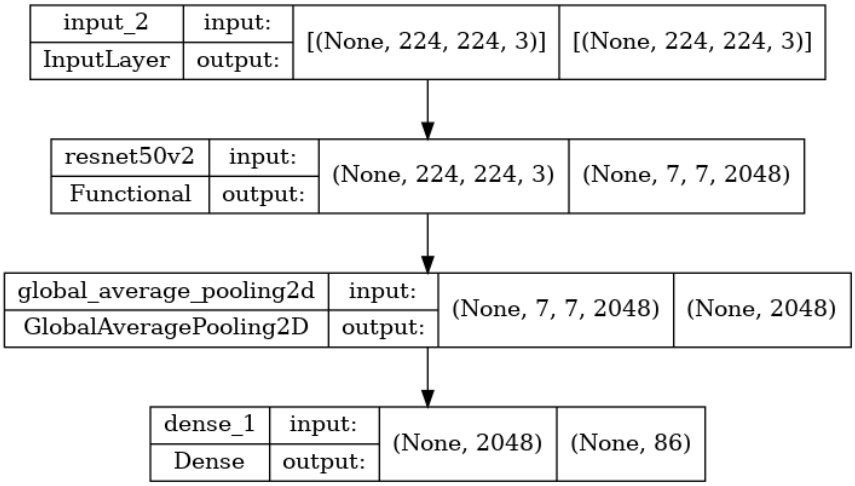
Ensimmäisen sovitusvaiheen Confusion Matrixista (Liite 2) nähdään, että malli toimii luokittelijana keskinkertaisesti. Ihmiskategoriassa näkyy vahva horisontaalinen viiva, sillä ihmiskategorian kuvia on merkittävästi enemmän kuin muiden kategorioiden kuvia. Matriisista huomataan myös, että hiustenkuivaajan kohdalla näkyy vertikaalinen tyhjä viiva. Tämä tarkoittaa, että malli ei luokitellut yhtään validointikuvaa kyseiseen kategoriaan. Mallilla on myös vaikeuksia luokitella pesäpallomailoja. Tämä voidaan havaita Confusion Matrixin pesäpallomailakategorian X ja Y-akseleiden leikkauksen tyhjästä kohdasta.

Voidaan myös todeta, että malli kategorisoi merkittävän paljon ihmisiä repuiksi ja lumilaudoiksi. Esimerkiksi lumilaudoissa tämä ilmiö mahdollisesti selittyy tarkastelemalla sovitus- ja validointidataa kyseisessä kategoriassa. Tarkastelusta ilmeni, että monissa lumilauta-kategorian kuvissa esiintyy myös osa ihmistä.

Kuva 10. Esimerkkejä lumilauta-kategorian kuvista, joissa esiintyy myös ihminen.

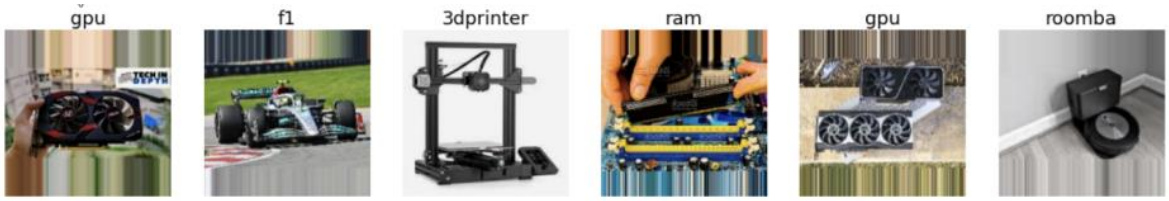
7.2.3 Uusien kategorioiden lisäys malliin

Tämän jälkeen arkkitehtuuriin tehtiin kuuden uuden kategorian lisäys Fine-Tuningia varten. Edellä tehdyn sovitusvaiheen parhaan mallin parametrit ladattiin, mutta viimeinen aktivointikerros korvattiin uudella, joka sisälsi 80 ulostulon sijaan 86 ulostuloa.



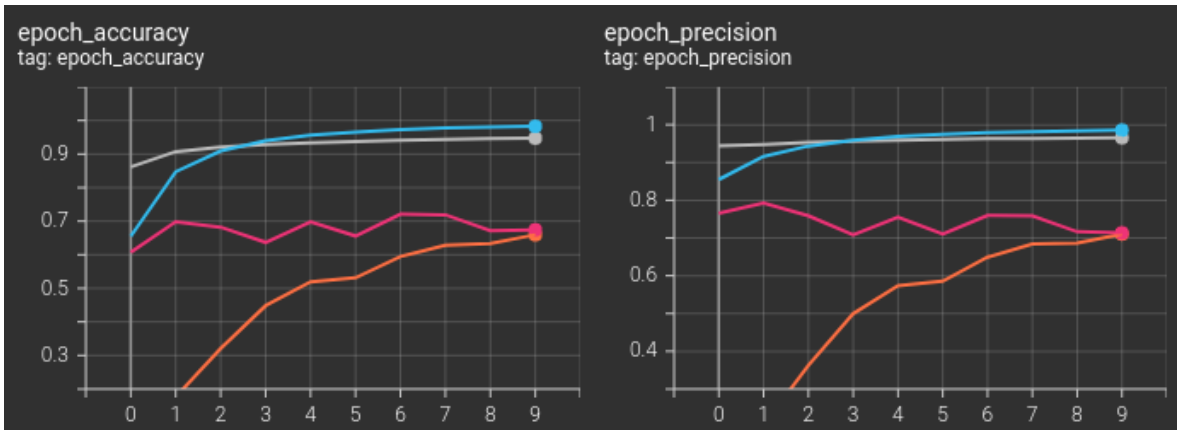
Kuvio 18. ResNet-malli 86 kategorian kategoriointipäällä

Fine-Tuning -vaiheessa malli jatkosovitettiin uuden ja vanhan sovitusdatan yhdistelmään. Kaikki muut parametrit, paitsi viimeinen kategoriointikerros, jäädytettiin vanhan kategorisointifunktionaalisuuden ylläpitämiseksi. Uudelleensovitusvaiheen datan vähäisen määrän vuoksi uusi data jaettiin 80/20 % -jaolla, joka tarkoitti, että kahdeksan kuvaa kymmenestä päätyi sovitusdataksi ja kaksi validointidataksi.



Kuva 11. Näyte uusien luokkien sovitusdatasta

Fine-Tuning vaihetta ajettiin 10 Epochia hyödyntäen samaa Adam-optimointialgoritmiä. Fine-Tuning vaiheessa havaittiin, että malli sovittui dataan paljon Pre-Training vaihetta hitaammin. Myös metriikat pysyivät paljon stabiilimpina Pre-Training -vaiheeseen verrattuna. Ilmiö selittyi todennäköisesti sillä, että sovitettavia parametrejä oli kerrosten jäädyttämisen vuoksi paljon vähemmän ja täten muutokset näin ollen paljon pienempiä. Parhaaksi malliksi valittiin viimeinen Epoch 10. Kyseisellä Epochilla mallin Accuracy oli 65,95 %, Loss 1,59 ja Precision 71,03 %, mitkä olivat hieman huonompia Pre-Training vaiheeseen verrattuna. Oikeita positiivisia vastauksia oli 10 116 kappaletta ja vääriä positiivisia 4 125.



Kuvio 19. Fine-Tuning -vaiheen metriikat. Harmaalla sovitusdatan ja oranssilla validointidatan metriikat

Tämän jälkeen mallista laskettiin uudet testimetriikat. Testauksessa oikeita positiivisia luokitteluja kirjattiin 10 513 kappaletta ja vääriä positiivisia 5 504. Täten Fine-Tuning osion Precisioniksi saatiin 65,64 %. Testauksesta laskettiin myös uusi Confusion Matrix (Liite 3), josta voitiin selvittää kuinka hyvin malli kykeni kategorisoimaan uusia luokkia. Kyseisestä Confusion Matrixista ilmeni, että malli pystyi luokittelemaan melko hyvin viisi kuudesta lisätystä

luokasta. GPU:n kohdalla mallilla oli vaikeuksia. Confusion Matrixista ilmeni myös, että Fine-Tuning sovitus oli myös vaikuttanut muiden kategorioiden luokittelukykyyn. Confusion Matrixista todettiin, että esimerkiksi pesäpallohanskojen tunnistamistamiskyky oli heikentynyt Pre-Training vaiheeseen verrattuna.

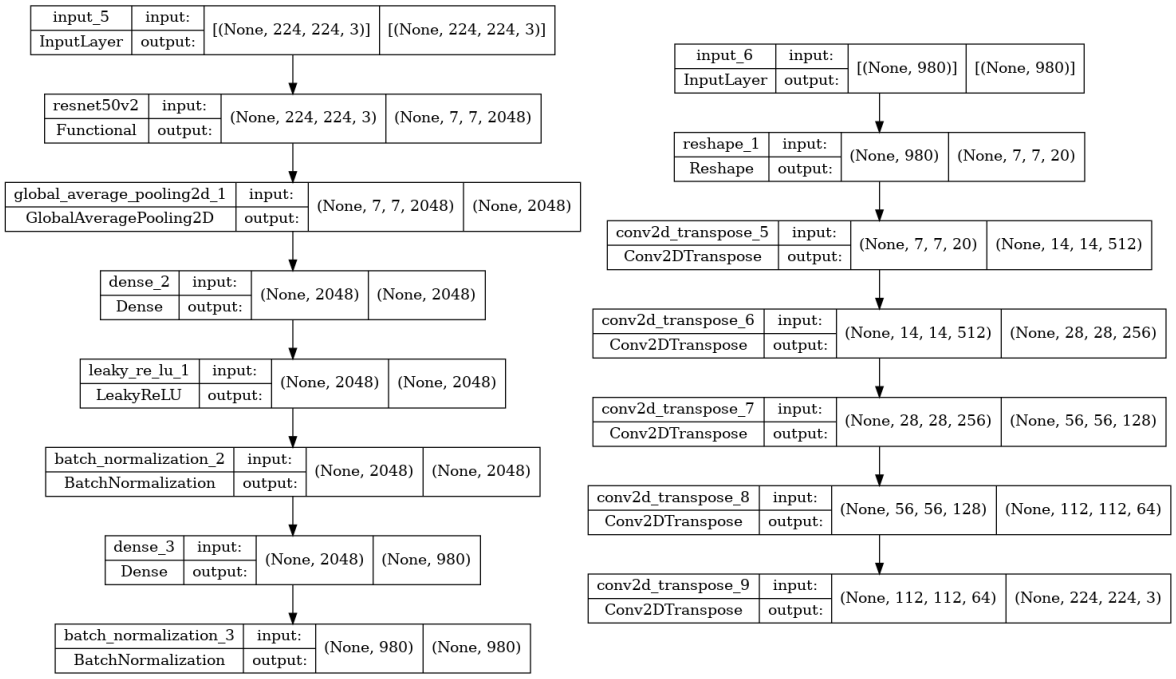
Kokeilusta voidaan päätellä, että tyypillisen CNN:n jatkosovittaminen luokittelemaan uusia kategorioita on mahdollista, mutta vaikeaa. On tärkeää pystyä tulkitsemaan, kuinka paljon erilaiset Fine-Tuning sovitusvaiheet vaikuttavat ja etsiä menetelmiä, joilla Catastrophic Forgetting voidaan ehkäistä.

7.3 Autoencoder merkitsemättömän datan luokittelu

Autoencoderit tarjoavat teoriassa ainutlaatuisen mahdollisuuden tehdä luokittelua hyödyntämällä merkitsemätöntä dataa. Merkitsemätön data tarkoittaa dataa, jolle ei ole asetettu oikeaa vastearvoa. Esimerkiksi kuvadatassa kuvan sisältämää tai sisältämiä luokkia.

7.3.1 Mallin arkkitehtuuri

Testausta varten kehitettiin Autoencoder-malli, jonka Encoder-puoleksi asetettiin aiemmin opinnäytetyössä käytetty ResNet50V2 hyödyntäen Keraksen tarjoamia ImageNet-datasettiin sovitettuja parametrejä. Encoder-puolen viimeiseksi kerrokseksi asetettiin 980:n yksikön Dense-kerros piirrevektoreita varten. Decoder-puoleksi asetettiin Reshape-kerros, joka muuttaa koodatun piirrevektorin kaksiulotteiseksi. Tämän jälkeen asetettiin viisi Conv2DTranspose-kerrosta, joiden tehtävä oli tuottaa piirrevektorin informaatiosta alkuperäinen kuva. Viidessä ensimmäisessä Conv2DTranspose-kerroksessa käytettiin ReLU-aktivoinitfunktiota ja viimeisessä Sigmoidia, jonka avulla Decoderin tuloste saatiin pidettyä 0:n ja 1:n välillä.

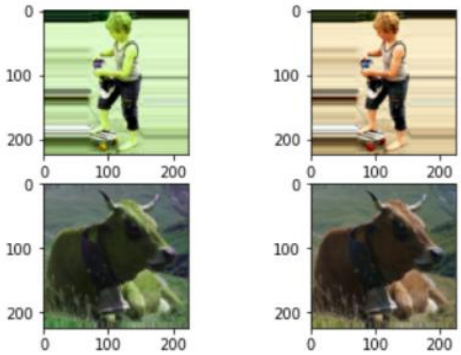


Kuvio 20. Sovitetun Autoencoder-mallin arkkitehtuuri; vasemmalla Encoder ja oikealla Decoder

Koko Autoencoderin sovittamisen jälkeen käytettiin vain Encoder-puolta. Encoderin tulostamista piirrevektoreista sovitettiin K-Means klusterointimalli.

7.3.2 Datan käsittely mallin sovittamista varten

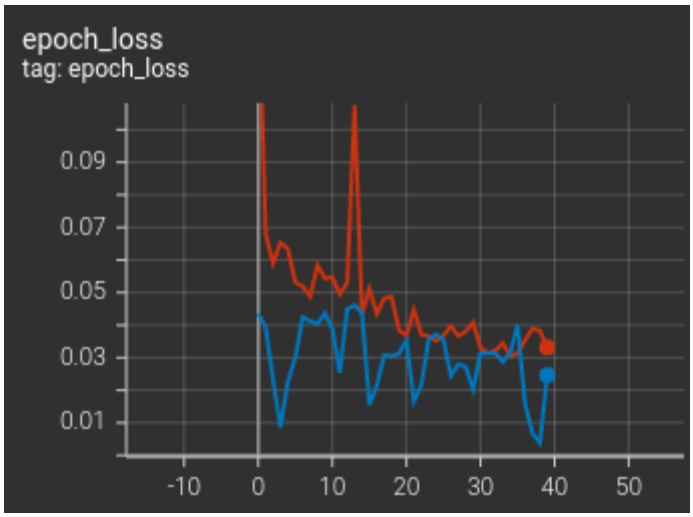
Mallin sovitus tapahtui Self-Supervised menetelmällä, joten sovitusdatan ei tarvinnut olla vastemerkittyä. Sen sijaan syöte ja vasteparina toimi sama kuva, mutta syötettä augmentoitiin, jotta voitiin estää Encoderia takertumasta alhaisen tason piirteisiin kuten väriin. Augmentointina käytettiin sattumanvaraista värimaailman ja kirkkauden säätöä.



Kuva 12. Esimerkki syöte- ja vasteparista

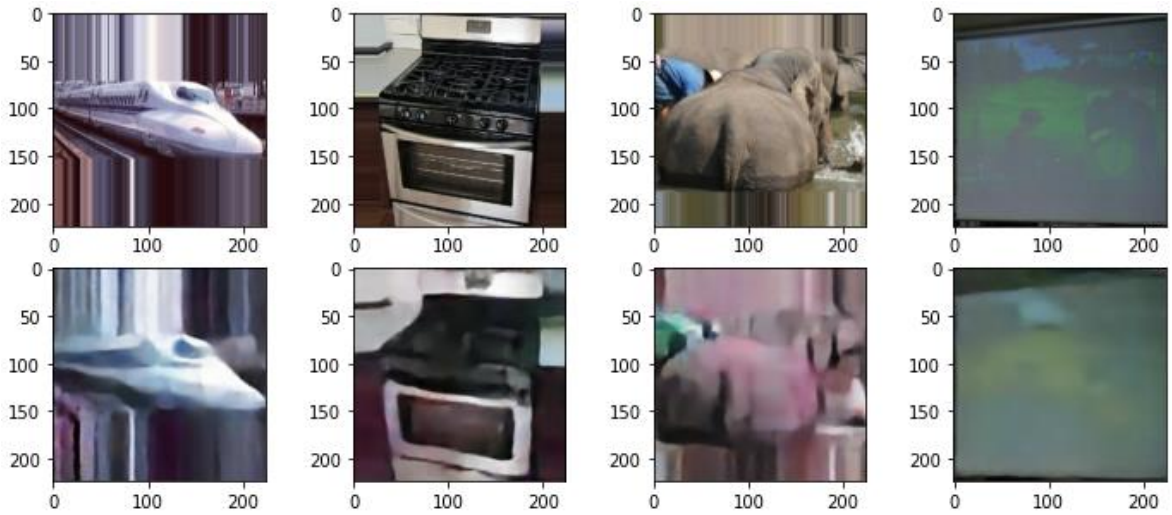
7.3.3 Mallin sovittaminen

Mallin rekonstruktiovirheeksi asetettiin keskineliövirhe, optimointialgoritmiksi Adam ja sen oppimisnopeudeksi 0.002. Koska Autoencoderin sovitusvaiheessa tarkoitus oli tuottaa vain mahdollisimman tarkka kopio vastekuvasta, sovitettaessa seurattiin tarkasti virhettä. Keskineliövirhe Decoderin tuottaman kuvan ja alkuperäisen kuvan välillä kertoi kuinka tarkan kopion Decoder pystyi tuottamaan. Mallia sovitettiin 40 Epochia, joista jokaisessa käytettiin 100:aa kappaletta 16:ta kuvaparin Batchia.



Kuvio 21. Sovitusvaiheen metriikat; sinisellä sovitusdatan ja punaisella validointidatan metriikat

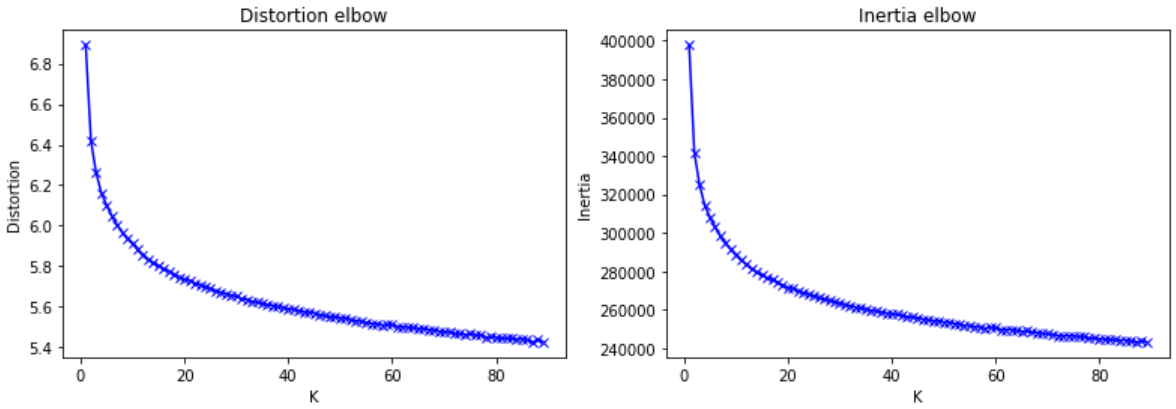
Parhaaksi malliksi valittiin Epoch 35, jolloin validointidatan keskineliövirhe oli 0,3132. Autoencoder-mallin toimintaa tarkasteltiin ajamalla muutama satunnaisesti valittu validointikuva koko Autoencoder-mallin läpi. Mallin tulosteena saatiin kuvia ja näiden kuvien samanlaisuutta verrattiin itse syötteeseen silmämääräisesti. Kuvassa 13 nähdään ylärivillä syötekuvia ja alarivillä mallin rekonstruktoidut kuvat. Rekonstruktioista voidaan huomata, että malli kykenee hyvin koodaamaan syötteen 980:n yksikön piirrevektoriksi ja rekonstruktoimaan tästä piirrevektorista alkuperäisen kuvan.



Kuva 13. Kuvia Autoencoder-mallin rekonstruktioista

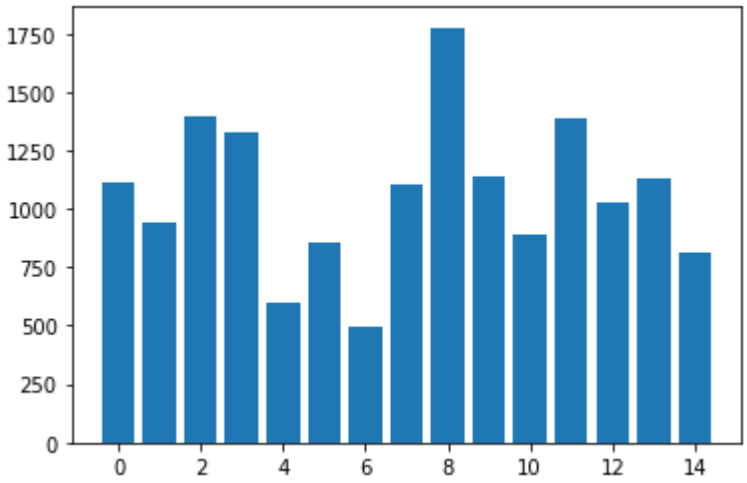
Tämän jälkeen sovitusdatasta valittiin 80 000 kuvaa, ne ajettiin Encoderin läpi ja piirrevektorit kirjattiin. Optimaalinen K-Means klusterimäärä piirrevektoreille laskettiin aiemmin esitetyllä Elbow-menetelmällä. K-Means malleja sovitettiin piirrevektoreihin K:n arvoilla 1-90. Malleista kirjattiin Inertia- ja Distortion -metriikat, joiden avulla optimaalinen K:n arvo valittiin.

Distortion ja Inertian graafeista kuviossa 22 voidaan havaita, että Elbow-piste esiintyy K:n arvossa noin 15. K:n arvo 15 merkitsee datan luokittelua vain 15:een luokkaan, mikä on huomattavasti vähemmän datassa esiintyvään 80 kategoriaan verrattuna. Lopullinen K-Means -malli sovitettiin piirrevektoreihin hyödyntämällä edellämainittua K:n arvoa 15.



Kuvio 22. Distortion- ja Inertia -metriikat

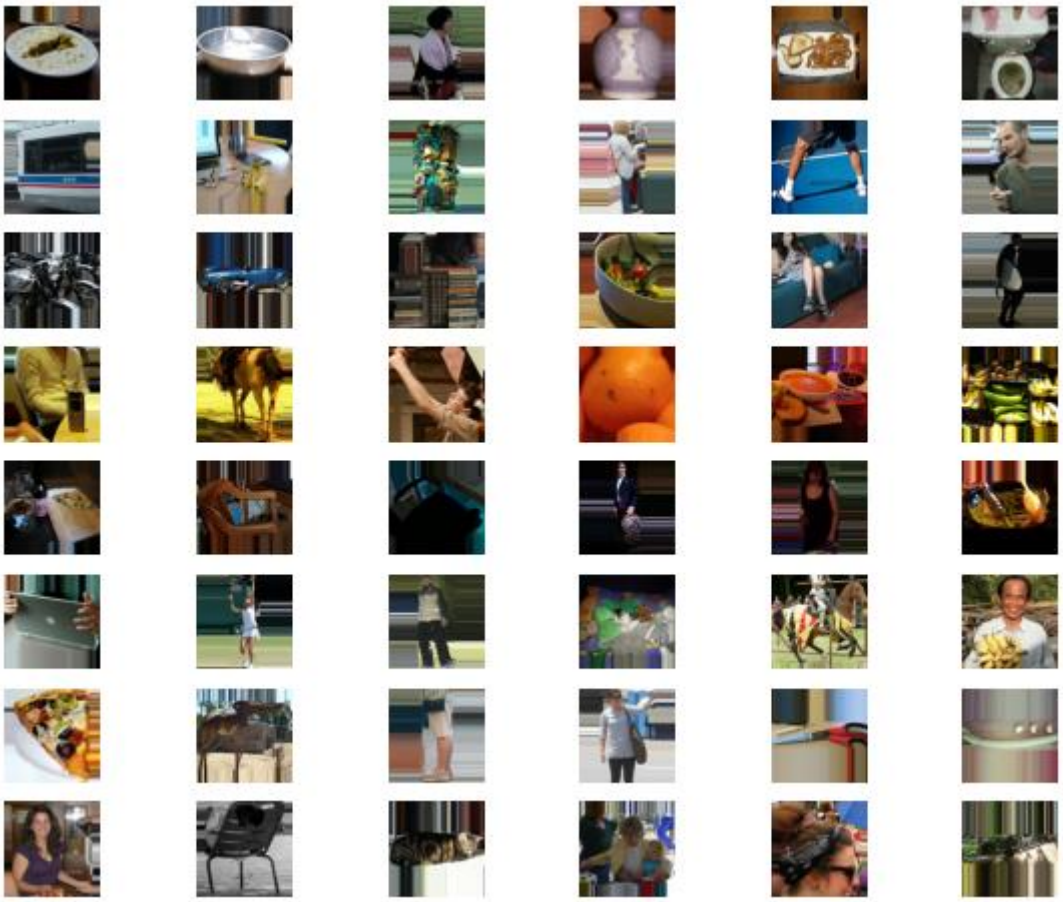
Tämän jälkeen kaikki 16 005 validointikuvaa ajettiin Encoderin läpi. Koodatut piirrevektorit syötettiin K-Means -mallille, jonka tulosteena saatiin piirrevektorin klusterin indeksi. Klusterin indeksi oli mallin tuottama kategoria kuvalle. Klusterijakaumassa kuviossa 23, voitiin huomata suurta hajontaa. Ilmiö oli odotettavissa, sillä validointidata oli myös vahvasti epätasapainoista.



Kuvio 23. Validointidatan jakauma K-Means klustereihin

Klustereiden kuvia tarkastelemalla huomattiin, että klusterointi ei ollut kovinkaan järkevää. Klusteroinnista otettiin näyte, joka esitetään kuvassa 14. Kun näytettä tarkasteltiin, huomattiin suuri määrä kellertäviä kuvia klusterilla 4 neljännellä rivillä ja pimeitä kuvia klusterissa 5

viidennellä. Tästä voitiin päätellä, että malli oli takertunut vahvasti alhaisen tason piirteisiin. Ilmiö oli odotettavissa, ja sen estämiseksi on olemassa useita lähestymistapoja; esimerkiksi SCAN-malli, eli Semantic Clustering by Adopting Nearest Neighbours. Suurimpana ongelmana automaattisessa klusteroinnissa on kuitenkin klustereiden arvaamattomuus. On vaikea ennalta määrittää minkälaisista piirteistä klusterit tulevat muodostumaan, ja mitkä piirteet päätyvät mihinkin klusteriin. Tämän takia automaattisen klusterointimallin jatkosovitus ja jo sovitettujen klustereiden ylläpito voi olla erittäin haastavaa. Tästä syystä päätettiin kokeilla toista lähestymistapaa valitsemalla uusi arkkitehtuuri.



Kuva 14. Näyte kahdeksan ensimmäisen klusterin kuvista

7.4 Siamese CNN

Siamilaisen mallin olisi tullut teoriassa sopia tutkittavaan käyttökohteeseen erinomaisesti. Jos similariteettilaskenta toimisi halutulla tavalla, pitäisi mallin luokittelemiseen pystyä lisäämään kategorioita vain lisäämällä esimerkkikuvia Support Settiin.

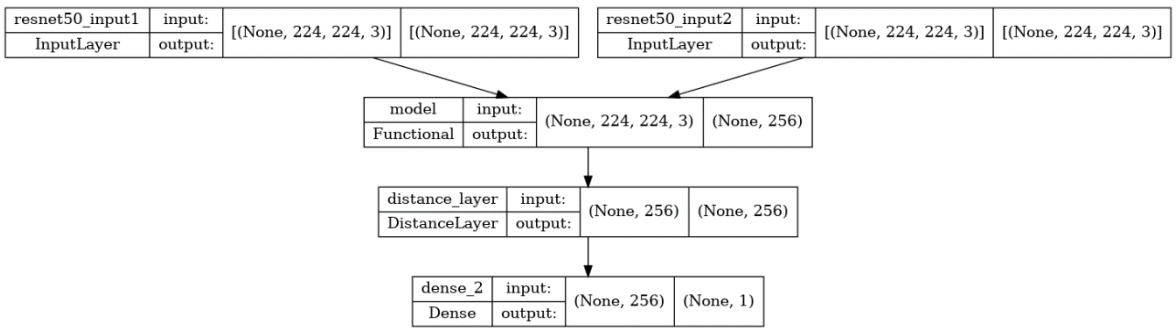
7.4.1 Mallin arkkitehtuuri

Kokeilua varten kehitettiin siamilainen neuroverkko, johon asetettiin kaksi syötepäätä. Toiseen syötepäähän syötettiin Support Setin Anchor-kuvat ja toiseen luokiteltava kuva. Piirteiden erottelua varten syötepäihin ladattiin Keraksen ResNet50V2 ImageNetiin sovitetuilla parametreillä. Loppuun lisättiin 2 048:n, 1 024:n ja 256:n yksikön Dense-kerrokset. Viimeisen 256 yksikön Dense-kerros sisälti syötteen piirrevektorin.

Piirrevektoreista laskettiin absoluuttinen etäisyys jokaisen piirteen väliltä kaavan 5 mukaan.

$$d(a,b) = |a_i - b_i| \quad (5)$$

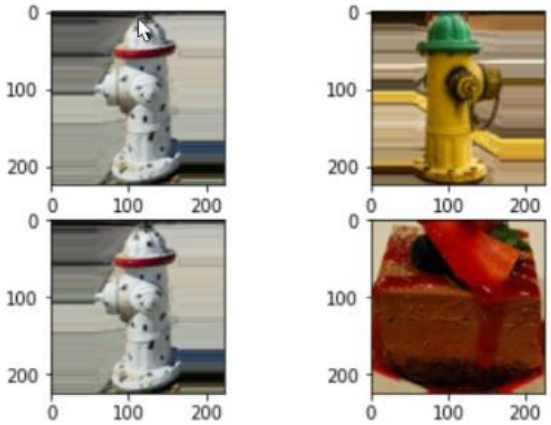
Kaavan 5 *a* on ensimmäisen syötteen piirrevektori ja *b* toisen syötteen piirrevektori. Piirteiden välisen etäisyysvektorin jälkeen lisättiin yhden yksikön aktivointikerros Sigmoid-aktivointifunktiolla ilmaisemaan kuvien samankaltaisuutta.



Kuvio 24. Sovitettavan siamilaisen CNN:n arkkitehtuuri

7.4.2 Datan käsittely mallin sovittamista varten

Sovitusdatana käytettiin samaa dataa kuin aiemminkin, mutta datasetin lataamista piti muuttaa uudelle tarkoitukselle soveltuvaksi. Sovituskuvista tehtiin pareja valitsemalla satunnainen kuva Anchoriksi ja toinen satunainnen kuva samasta kategoriasta positiiviseksi pariksi. Positiivisen parin vasteeksi asetettiin 1. Lisäksi Anchorille valittiin satunnaisesti toinen eri kategoriaan kuuluva kuva negatiiviseksi pariksi ja tämän vasteeksi arvo 0. Näin positiivisten ja negatiivisten sovituskuvien paritus saatiin toteutettua 50/50 % -jakaumalla positiivisten ja negatiivisten parien suhteen. Kyseistä mallia sovitettaessa datalle ei tehty augmentointia. Sovitukseen kuvapareja generoitiin 262 144 kappaletta.



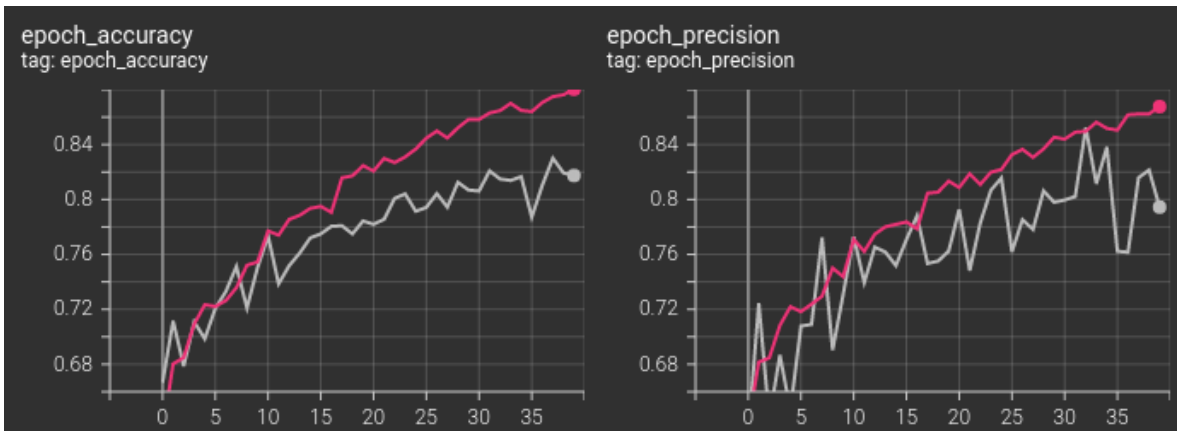
Kuva 15. Generoituja kuvapareja

7.4.3 Mallin sovittaminen

Optimointialgoritmiksi valittiin Adam ja sen oppimisnopeudeksi asetettiin 0.001. Virhefunktiona käytettiin Keraksen Binary Cross Entropy:ä.

Seuratut metriikat olivat samat kuin ResNet50V2:n sovitusvaiheessa. Mallin haluttiin maksimoivan oikeiden positiivisten vastauksien ja minimoivan väärien positiivisten vastauksien määrän. Sovitusvaiheessa seuratut metriikat eivät kuitenkaan ole täysin verrattavissa ResNetin metriikoihin, sillä tarkkuus käytännössä riippuu käytettävistä kuvapareista.

Mallia sovitettiin 40 Epochia, joista jokaisessa käytettiin 8 000:ta satunnaisesti ladattua kuvaparia. Kuvaparit jaettiin 16:n kokoisiin Batcheihin. Validoinnissa käytettiin 32 010 kuvaparia valitsemalla jokaiselle validointidatan kuvalle sattumanvarainen positiviinen ja negatiivinen pari. Lisäksi jokaisen Epochin jälkeen validointidatan piirrevektorit tallennettiin, jotta eroteltuja piirrevektoreita voitiin visualisoida.



Kuvio 25. Sovitusvaiheen metriikat; punaisella sovitusdatan ja valkoisella validointidatan metriikat

Parhaaksi malliksi valittiin Epoch 37, jossa Precision oli 81,58 %, Accuracy 83,01 % ja Loss 0,376. Oikeita positiivisia vastauksia oli 13 648 kappaletta ja virheellisiä positiivisia 3 081.

Sovitusvaiheessa esitetyt metriikat ovat suuntaa-antavia, sillä jokaista validaatiokuvaa kohden valittiin ainoastaan yksi satunnainen positiivinen ja negatiivinen pari. Lopullista testausta varten sovitusdatasta valittiin 16 kuvaa per kategoria eli 80-Way 16-Shot Support Set. Support Setin ja kaikkien 16 005:n validointikuvan piirrevektorit tallennettiin.

Testausta varten Support Setin piirrevektoreista laskettiin keskiarvo merkitsemään kategorian keskipistettä piirrevektorien koordinaatistossa. Tämän jälkeen jokaisen validointikuvan piirrevektorin ja Support Setin piirrevektorien keskipisteiden etäisyys laskettiin kosinietäisyysfunktiolla. Kosinietäisyyden kaava esitetään kaavassa 6 (Scipy 2022).

$$1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2} \quad (6)$$

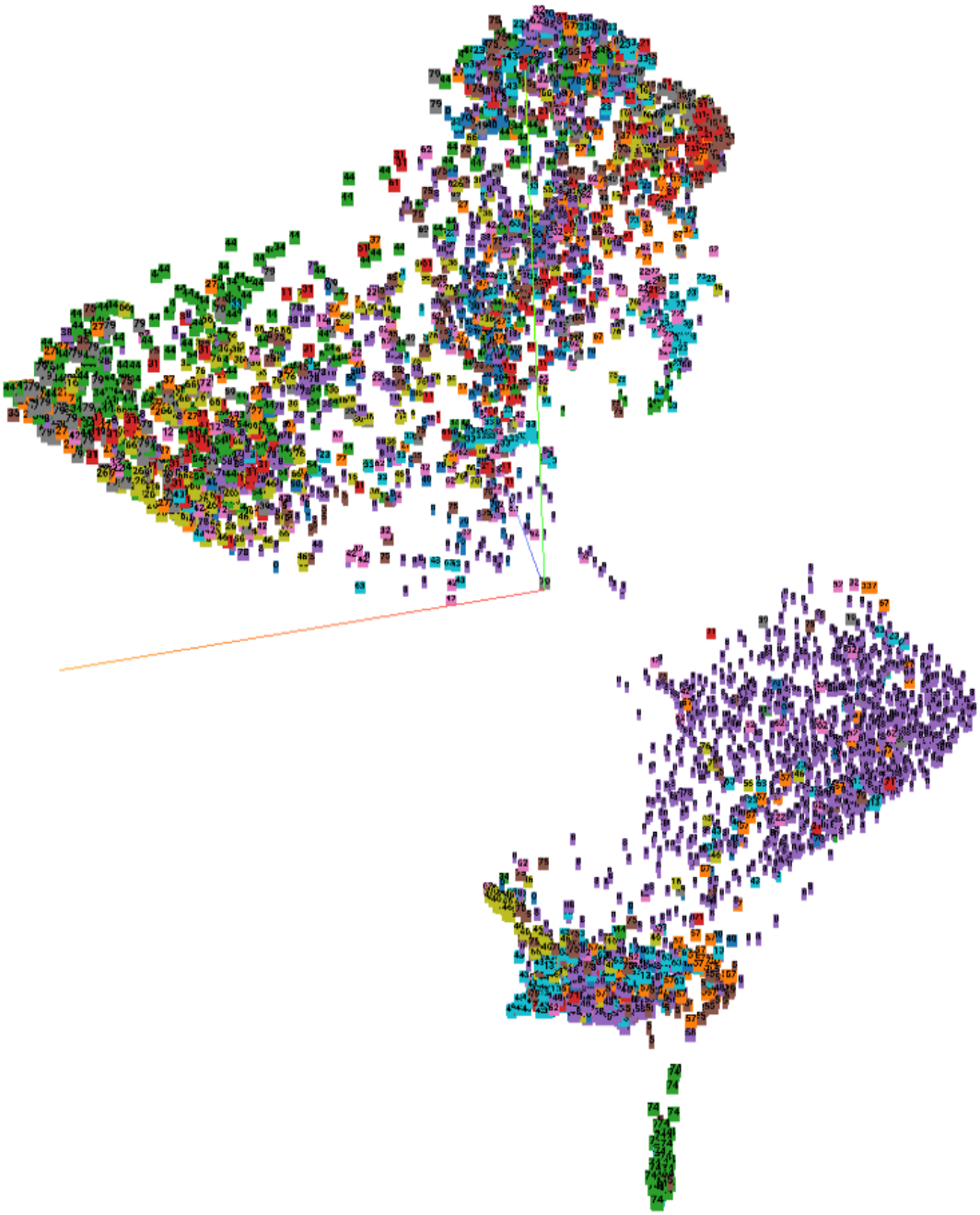
Kosinietäisyysfunktion $u \cdot v$ tarkoittaa piirrevektorien välistä pistetuloa. Kosinietäisyyksistä pienin eli läheisin arvo kirjoitettiin muistiin merkitsemään validointikuvalle parhaiten sopivaa kategoriaa. Tämän minimietäisyyden pääteltiin olevan mallin kategorioinnin tuloste.

Tulosteista ja oikeista vasteista laskettiin Confusion Matrix (Liite 4), josta voitiin analysoida mallin toimintaa. Tulosteista ja oikeista vasteista kirjattiin oikeat ja väärät positiiviset vastaukset sekä Precision-metriikka.

Malli pystyi kategorioimaan oikein 7 302 ja väärin 8 703 validointikuvaa. Täten mallin Precision oli vain 45,62 %. Confusion Matrixista huomattiin, että se sisälsi paljon enemmän kohinaa optimaalisen diagonaalisen viivan ulkopuolella verrattuna aiemmin tutkimuksessa sovitettuun ResNet50V2-malliin. Confusion Matrixin mukaan mallilla oli vaikeuksia etenkin hammasharjojen, pesäpallomailojen, lumilautojen ja hiustenkuivaajien kategorioinnissa.

Mallin huonoa luokittelua voitiin tarkastella sovitusvaiheessa tallennetuilla validointidatan piirrevektoreilla. Ensimmäisen ja 37:n Epochin piirrevektorit ladattiin Tensorflown sivuston selainpohjaiseen projektorityökaluun. Työkalun UMAP-ulottuvuuksien vähennysalgoritmillä pystyttiin projisoimaan kategorioiden klustereiden kehitys kolmiulotteisessa tilassa.

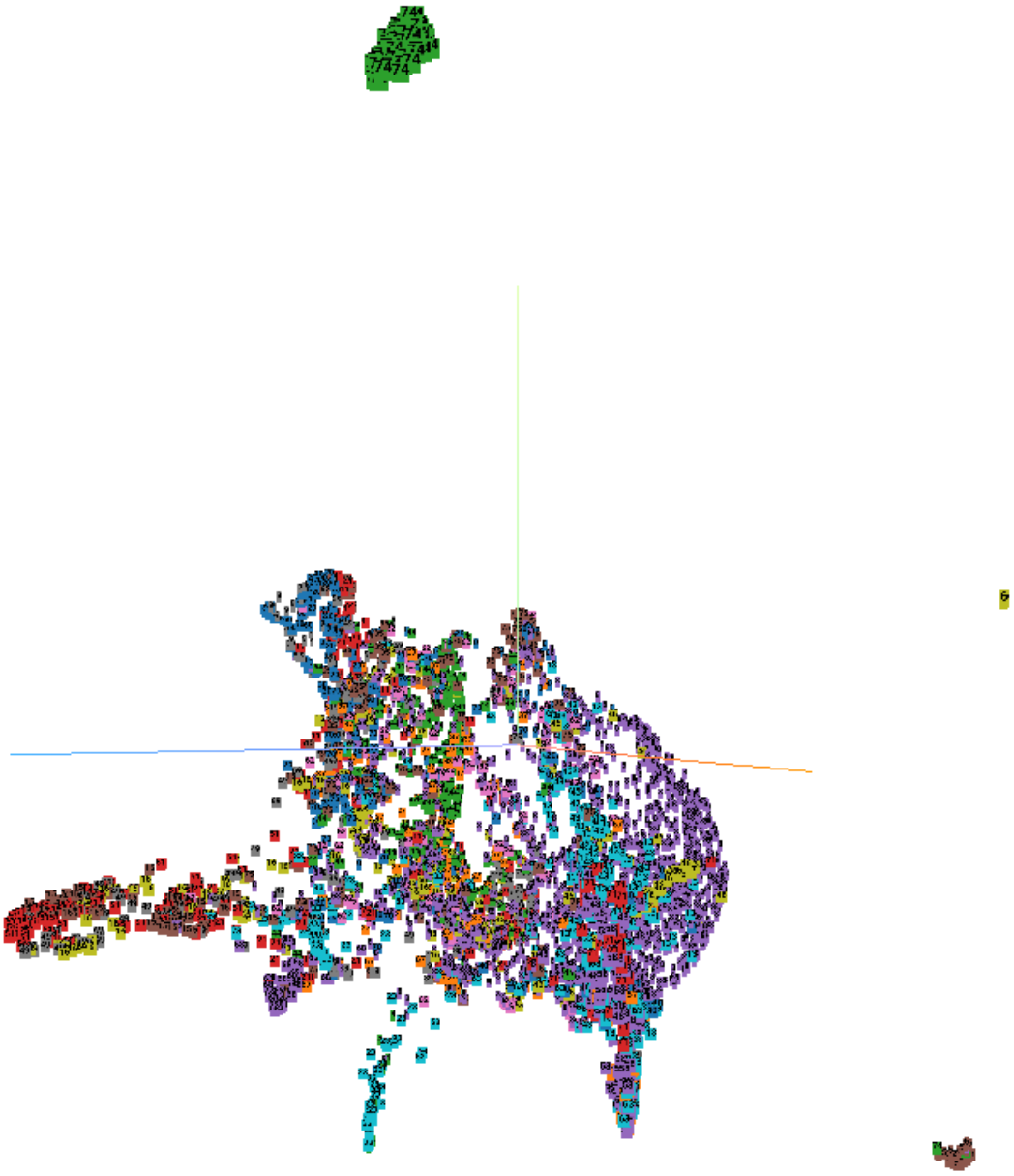
Ensimmäisen Epochin UMAP-projisiosta kuviossa 26 saatiin vertailukelpoinen pohja viimeisen Epochin projisiolle. Kuvasta kävi ilmi, että malli oli jo yhden Epochin jälkeen onnistunut erottelemaan luokan 74, eli seepran, piirteet omaksi klusterikseen kuvan alareunassa.



Kuvio 26. Ensimmäisen Epochin jälkeen projisoidut validointidatan piirrevektorit

Epochin 37 kohdalla osa luokista, esimerkiksi stop-merkki ja kirahvi, olivat hyvin eroteltuina muista. Muutkin kategoriat piirtyvät suhteellisen hyviin kasoihin, mutta kasojen väliset etäisyydet olivat pieniä, mikä oli todennäköinen syy mallin huonoon kategoriointikykyyn. Hajonta on nähtävissä kuviossa 27. Kategoriointikykyä voisi mahdollisesti parantaa optimoimalla hyperparametrejä, kuten piirrevektorien ulottuvuuden kasvattamisella suuremmaksi kuin 256, tai etsimällä parempia vertailukuvia Support Setin kuvadataksi. Kasojen pienten

etäisyyksien vuoksi mallin todettiin olevan käyttötarkoitukseen sopimaton ja uutta arkkitehtuuria kokeiltiin Triplet Loss -virhefunktiota hyödyntämällä.



Kuvio 27. Epochin 37 jälkeen projisoidut validointidatan piirrevektorit

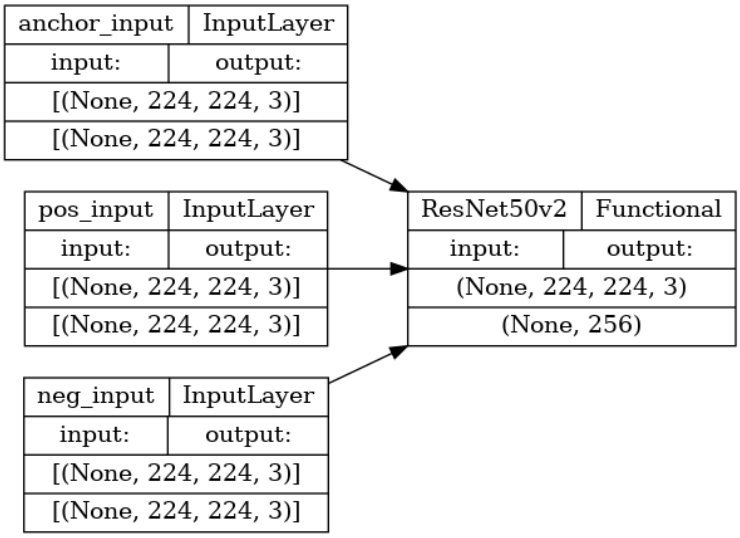
7.5 Siamese CNN Triplet Loss

Aiemmin opinnäytetyössä kokeiltu siamilainen CNN kahdella syötteellä tuotti toivottua heikompia tuloksia. Kokeilussa todettiin, että eri kategorioiden piirrevektorit päätyivät

piirrevektorien koordinaatistossa hyvin lähelle toisiaan. Tästä syystä seuraavaksi opinnäytetyössä kokeiltiin siamilaista CNN:ää Triplet Loss virhefunktiolla.

7.5.1 Mallin arkkitehtuuri

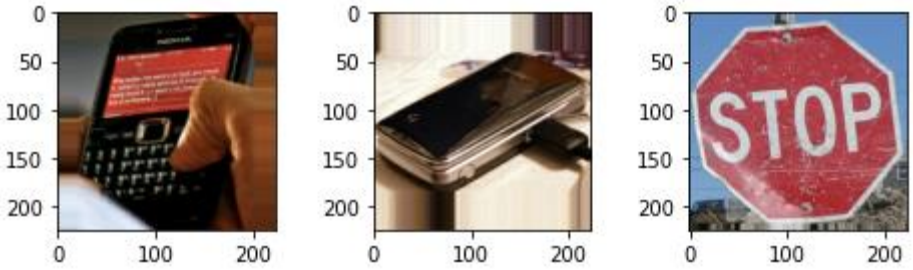
Kokeilua varten kehitettiin neuroverkkomalli, jossa piirteiden erottelua varten ladattiin aiemminkin opinnäytetyössä käytetty Keraksen ResNet50V2 ImageNetiin sovitetuilla parametreillä. ResNet mallin perään lisättiin 2 048:n, 1 024:n ja 256:n yksikön Dense-kerrokset. Viimeksi mainittua 256:n yksikön Dense-kerrosta käytettiin piirrevektoreina, kuten opinnäytetyössä aiemmin toteutetussa siamilaisessa CNN:ssä. Sovitusvaihetta varten lisättiin kolme syötepäätä: Anchor, Positive ja Negative. Kaikki syötepäät liitettiin ResNet50V2:een ja täten kaikista kolmesta syötepäästä saatiin tulostettua 256:n yksikön piirrevektorit.



Kuvio 28. Sovitettu siamilainen CNN:n kolmella syötepäällä

7.5.2 Datan käsittely mallia varten

Mallia varten sovitusdatasta tehtiin kolmen kuvan triplettejä. Triplettien valinnassa käytettiin vastaavaa metodiikkaa kuin aiemmin esitetyssä kahden syötteen siamilaisessa CNN:ssä. Jokaisen tripletin kohdalla valittiin satunnainen Anchor-kategoria ja kategorian kuvakansiosta ladattiin satunnainen kuva. Positiivinen pari ladattiin valitsemalla toinen satunnainen kuva samasta kategoriasta ja negatiivinen kuva ladattiin valitsemalla satunnainen toinen kategoria, josta valittiin satunnainen kuva. Sovitusdatan augmentoinnin ei nähty olevan tarpeellista.



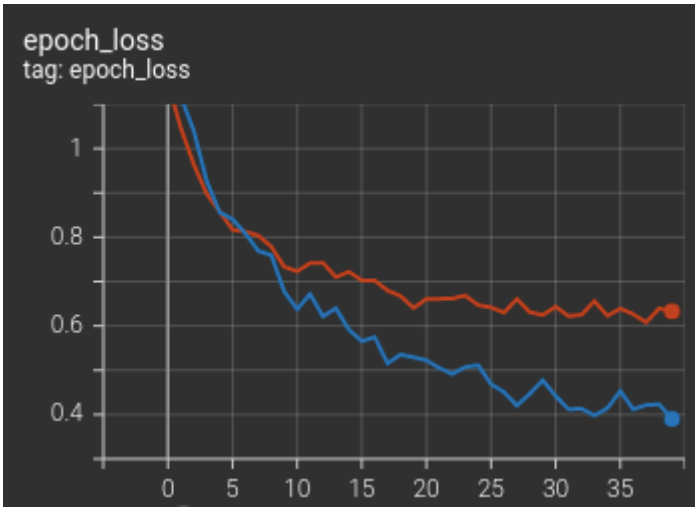
Kuva 16. Esimerkki sovituksessa käytetystä tripletistä

Sovitusdataa generoitiin 262 144 triplettiä ja validointia varten 4 096 triplettiä validointidatasta.

7.5.3 Mallin sovittaminen

Malli sovitettiin sovitusdataan käyttäen TensorFlow Addons -kirjaston TripletHardLoss-virhefunktiota. TripletHardLoss-virhefunktio toimii opinnäytetyössä aiemmin esitetyllä Online-metodilla valitsemalla jokaisesta Batchista vaikeimman positiivisen ja negatiivisen parin (TensorFlow 2022d). TripletHardLossin marginaaliksi asetettiin arvo 1. Sovituksen optimointialgoritmiksi valittiin Adam ja sen oppimisnopeudeksi asetettiin 0.00005.

Sovitusvaiheessa seurattiin ainoastaan Lossia, mikä TripletHardLossin kohdalla merkitsi kaavan 1 mukaista laskutoimitusta, jossa $f(x_i^a)$, $f(x_i^p)$ ja $f(x_i^n)$ valittiin tappiofunktion toimesta täyttämään vaikean tripletin vaatimukset. Mallia sovitettiin 40 Epochia, joista jokaisessa käytettiin 6 400 triplettiä 32:n kokoisissa Batcheissä. Jokaisen Epochin jälkeen kaikkien 16 005:n validointikuvan piirrevektorit tallennettiin myöhempää klusteroinnin tarkastelua varten. Pienimmän virheen saavuttanut malli valittiin jatkoon.

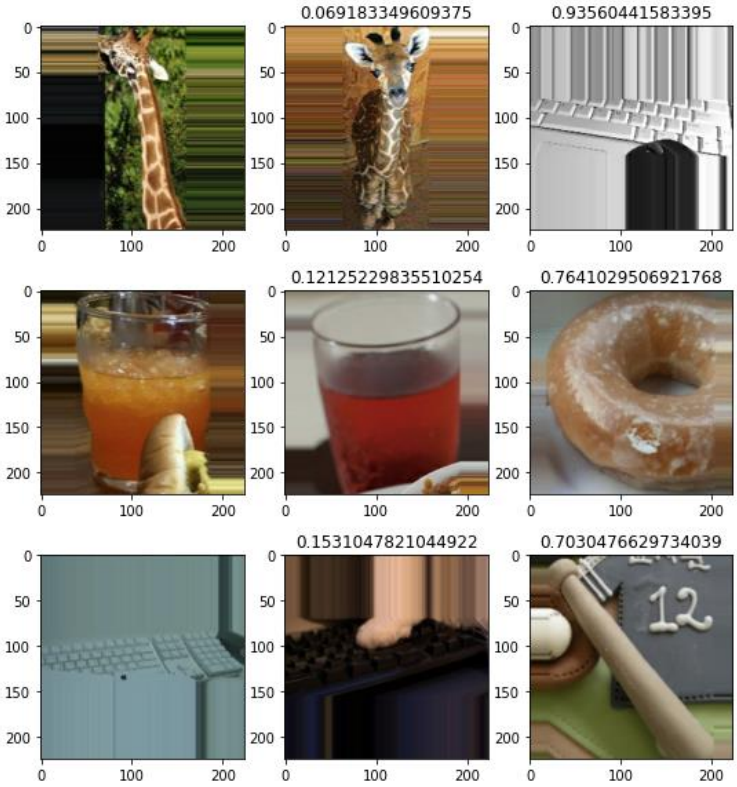


Kuvio 29. Sinisellä sovitus- ja punaisella validointidatan Pre-Training virhemetriikat

Pienin validointivirhe saavutettiin Epochissa 37 arvolla 0,6078. Sovituksen jälkeen sovitusdatasta valittiin sattumanvarainen 80-Way 16-Shot Support Set vertailukelpoista testausta varten. Support Setin kuvat ajettiin sovitetun mallin läpi ja piirrevektoreiden keskiarvot otettiin talteen. Tämän jälkeen validointidatan kaikki 16 005 kuvaa ajettiin sovitetun mallin läpi ja näiden piirrevektorien etäisyys Support Setin keskiarvoihin laskettiin samalla kosini-etäisyysfunktiolla kuin opinnäytetyössä aiemmin esitetyssä siamilaisessa CNN:ssä.

Support Setin piirrevektorien keskiarvojen ja syötteen piirrevektorien minimi kosinietäisyyden katsottiin olevan mallin antama kategoria syötteelle. Täten validointidatasta laskettiin Confusion Matrix, oikeat positiiviset- ja väärät positiiviset- luokittelut ja Precision -metriikka. Oikeita positiivisia vastauksia malli luokitteli 12 437 kappaletta ja vääriä positiivisia 3568 kappaletta. Täten mallin Precision oli 77,71 %.

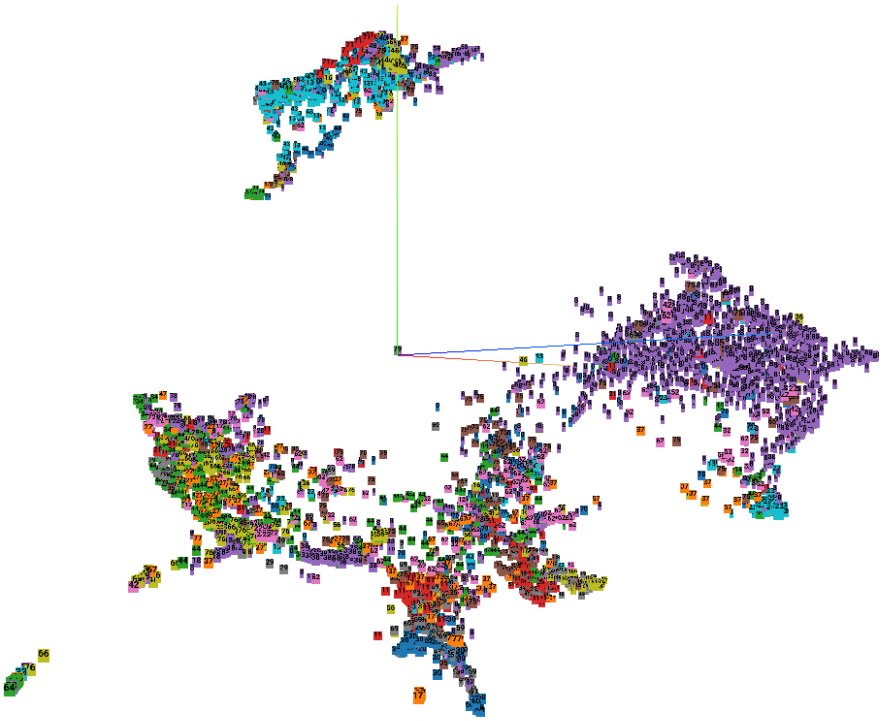
Confusion Matrixista huomattiin, että malli piirtää suhteellisen vahvan diagonaalisen viivan, mikä tarkoittaa, että mallin luokittelu toimi hyvin. Ongelmat ovat pitkälti samoja kuin pelkän ResNetin tapauksessa, eli pesäpallomailat ja hiustenkuivaajat tuottivat ongelmia.



Kuva 17. Esimerkkejä mallin tuottamasta similariteettilaskennasta

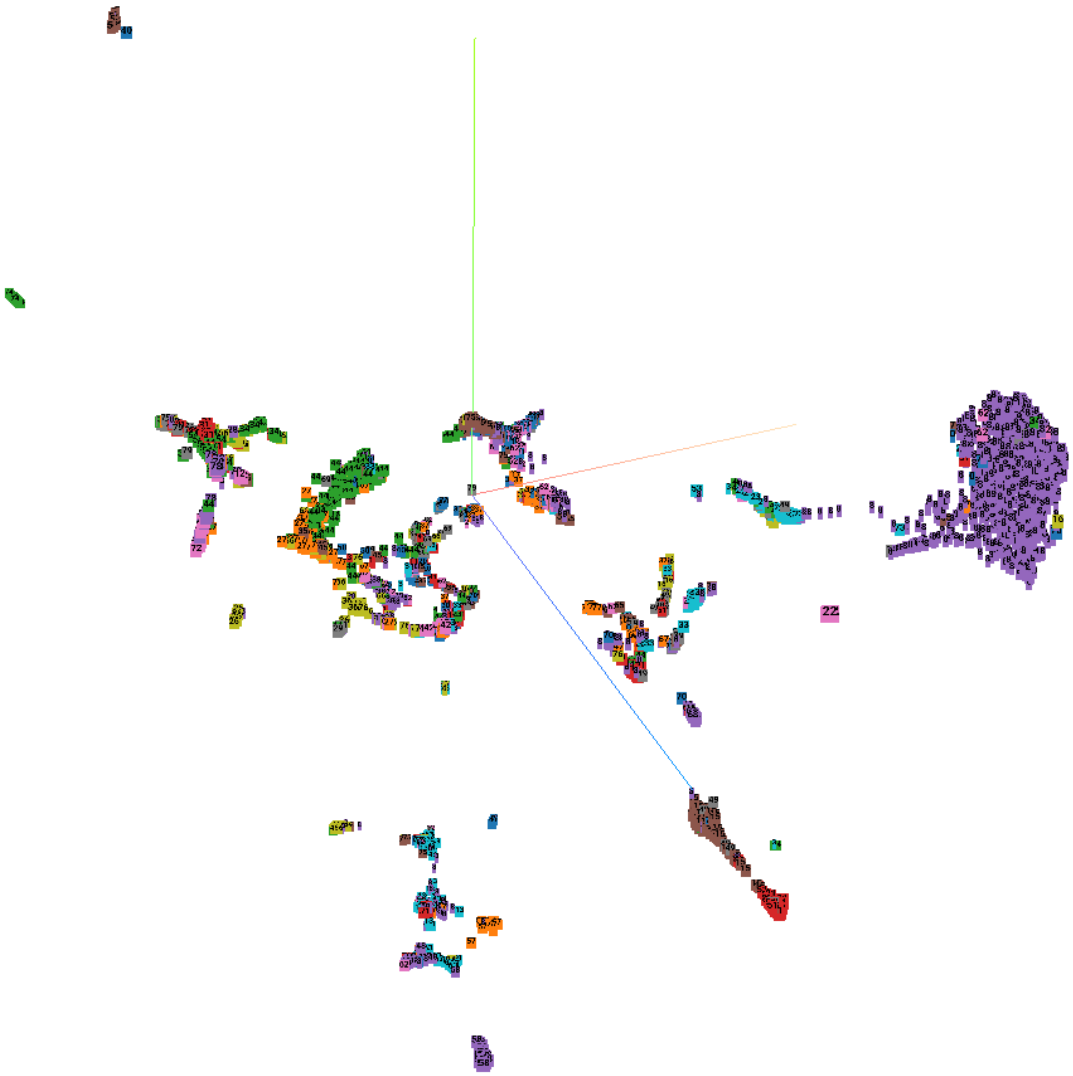
Sovitusvaiheessa tallennetuista piirrevektoreista pystyttiin tulkitsemaan, kuinka hyvin malli kykeni erottelemaan kategorioita toisistaan hyödyntäen opinnäytetyössä aiemmin käytettyä TensorFlown selainpohjaista UMAP projisointialgoritmiä.

Kuviossa 30 esitetyssä ensimmäisen Epochin jälkeisestä piirrevektoreiden projisiosta voitiin havaita kategorioiden erkaantumista toisistaan.



Kuvio 30. Ensimmäisen Epochin jälkeiset validointidatan piirrevektorit projisoituina

Parhaiten sovitetulla mallilla Epoch 37:llä huomataan kuviosta 31, että kategoriat ovat hyvin erottuneet toisistaan.

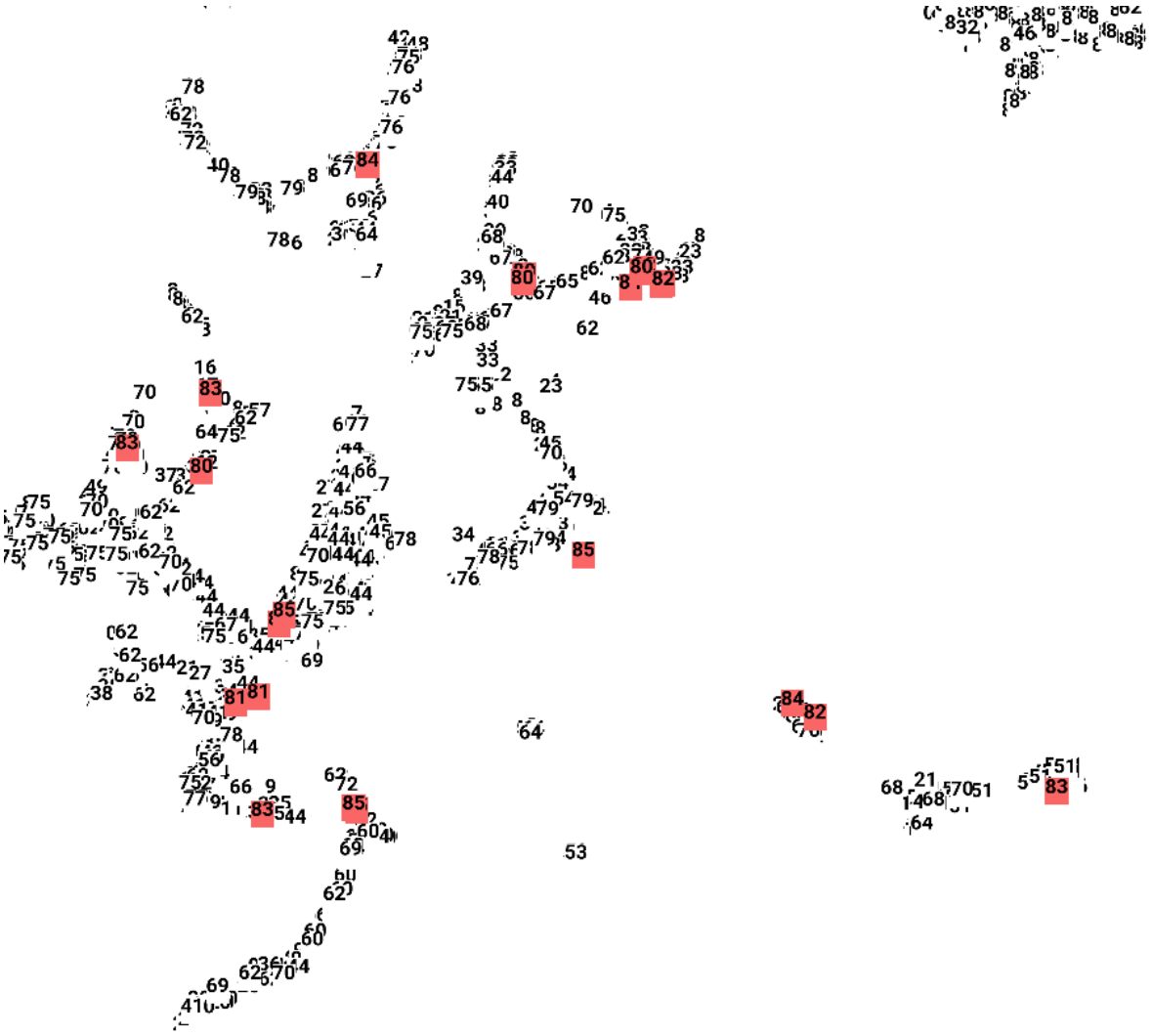


Kuvio 31. Epoch 37:n jälkeen projisoidut validointidatan piirrevektorit

7.5.4 Uusien kategorioiden lisäys

Seuraavaksi mallin kategoriointikykyä testattiin lisäämällä validointidataan opinnäytetyössä esitetyt kuusi uutta kategoriaa. Ensin mallia testattiin ilman Fine-Tuning sovitusvaihetta, koska kyseinen arkkitehtuuri mahdollisti tämän. Uusien kategorioiden Anchor-kuvat lisättiin Support Setin sovitusdatasta. Testaus tehtiin uudelleen lisäämällä validointidataan uudet kuusi kategoriaa. Kun uudet kategoriat oli lisätty eikä Fine-Tuning -sovitusta tehty, oikeita positiivisia vastauksia saatiin 12 377 kappaletta, vääriä positiivisia 3 640 ja näin ollen Precision-metriikaksi saatiin 77,27 %. Malli luokitteli validointidataa jonkin verran edellistä vaihetta huonommin, mutta Confusion Matrixista (Liite 6) havaittiin, että malli pystyi tunnistamaan oikein osan lisätyistä kuudesta kategoriasta oikein ilman Fine-Tuning sovitusta uusien

kategorioiden datalla. Projisoimalla uusien kategorioiden piirrevektorit huomataan kuviosta 32, että piirteet päätyivät osittain kauas toisistaan.



Kuvio 32. Kuuden uuden kategorian piirrevektorit projisoituna ilman Fine-Tuning sovitusvaihetta

7.5.5 Fine-Tuning

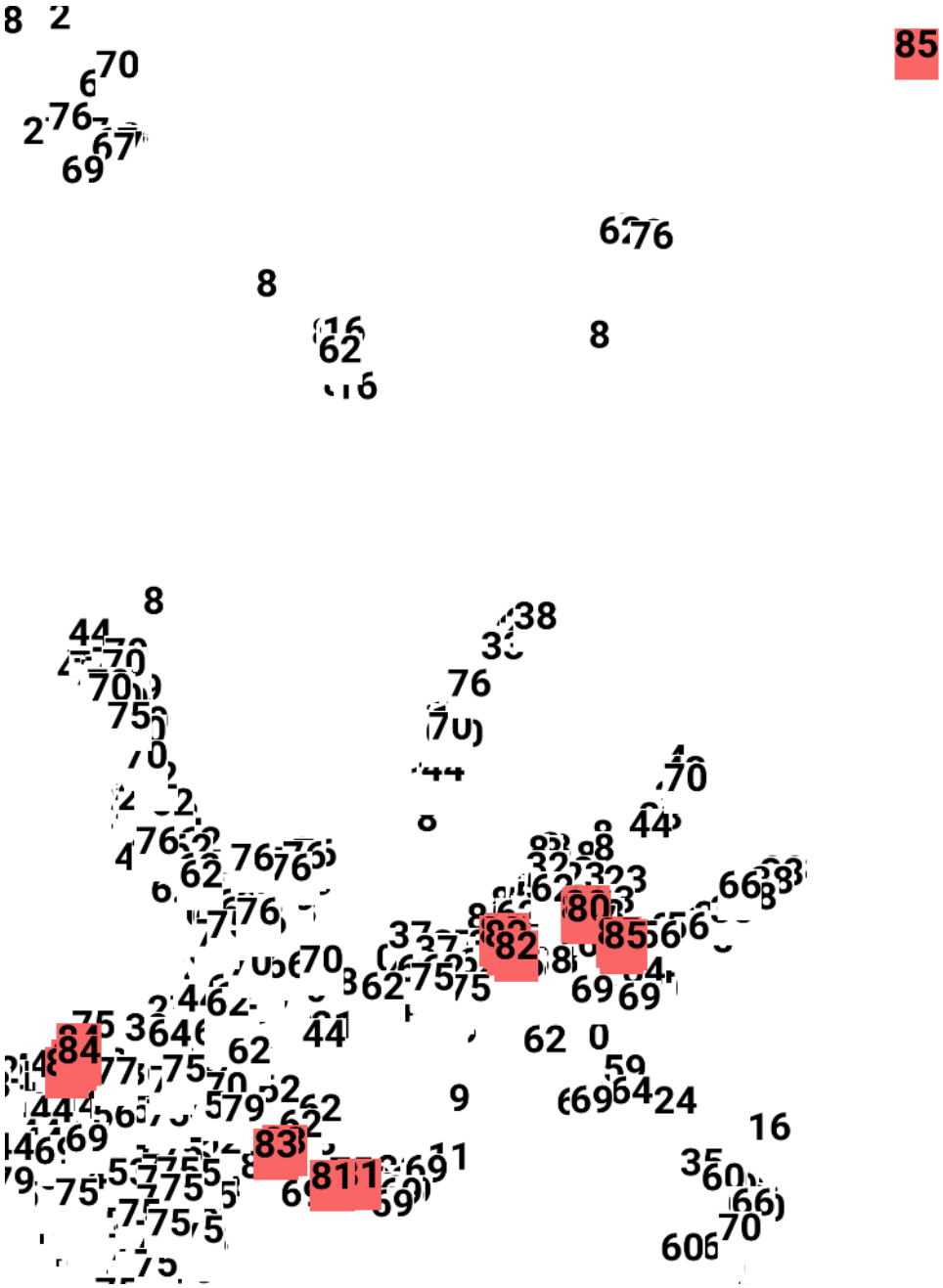
Uusien kategorioiden luokittelun parantamista varten mallia jatkosovitettiin lisäämällä sovitusdataan kuuden uuden kategorian sovituskuvat eli kahdeksan kuvaa per kategoria. Mallin konvoluutiokerrokset jäädytettiin ja sovitettavaksi jäi vain viimeiset kolme Dense-kerrosta. Mallia sovitettiin samalla virhefunktiolla, optimointialgoritmillä ja oppimisnopeudella, 10 Epochin ajan.

Kuvio 33. Fine-Tuningin virhemetriikat, jossa sovitusvirhe harmaalla ja validointivirhe oranssilla

Fine-Tuning sovitusvaiheen pienin validointivirhe 0,5418 mitattiin Epochilla 8. Fine-Tuning sovitusvaiheen jälkeen Support Setin piirrevektorien keskipisteet laskettiin uudelleen ja validointidatasta laskettiin uusi Confusion Matrix, oikeat positiiviset luokittelut, väärät positiiviset luokittelut ja Precision-metriikka samoin kuin ensimmäisessä sovitusvaiheessa. Oikeita positiivisia luokitteluja kirjattiin 12 525 kappaletta, vääriä positiivisia 3 492 ja näin ollen Precisioniksi saatiin 78,20 %.

Precision oli hieman parempi edelliseen sovitusvaiheeseen verrattuna. Confusion Matrixista todettiin (Liite 7), että matriisin oikealla alhaalla näkyvien kuuden uuden kategorian diagonaalinen viiva oli vahvistunut. Tämä tarkoitti, että malli kykeni erottelemaan kyseisiä luokkia paremmin kuin ilman Fine-Tuning sovitusta.

Fine-Tuningin jälkeen uusien kategorioiden piirrevektorit projisoitiin uudelleen. Projisiosta kuviossa 34 nähtiin, että uudet kategoriat olivat Fine-Tuning sovitusvaiheen jälkeen lähempänä toisiaan, mahdollistaen paremman luokittelun. Poikkeuksena oli yksi kategorian 85, eli robotti-imurin, piirrevektori, joka päätyi melko kauas muiden robotti-imurien piirrevektoreista.



Kuvio 34. Kuuden uuden kategorian piirrevektorit projisoituna Fine-Tuningin jälkeen

8 Videohakupalvelun prototyyppi

8.1 Prototyypin määrittely

Opinnäytetyön lopputuloksena oli tarkoitus tuottaa prototyyppitason indeksointipalvelu videotiedostoille. Palvelu tuli pystyä yhdistämään verkkolevyyn tai kansioon helposti, ja palvelun oli tarkoitus indeksoida videoita automaattisesti hyödyntämällä parhaiten soveltuvaa empiirisessä osiossa testattua neuroverkkomallia. Palveluun tuli kehittää rajapinta, jonka avulla videoita voitaisiin hakea tutkimuksessa käytettyjen kategorioiden avulla. Prototyypin ei tarvinnut lokalisoida löytyneitä luokkia videoista.

8.2 Valittu neuroverkkomalli

Parhaiten soveltuvaksi neuroverkkomalliksi valittiin viimeisenä tutkimuksessa empiirisesti testattu Triplet Loss -tappiofunktiolla sovitettu siamilainen CNN. Malli valittiin parhaiten soveltuvaksi sen saavuttaman Precision-metriikan takia. Malli myös antoi paljon säätövaraa ilman että mallia joutui uudelleensovittamaan. Se mahdollisti esimerkiksi Support Setin uudelleenmäärittelyn ja hyväksyttävän kategorioinnin marginaalin muuttamisen ilman uudelleensovittamista.

8.3 Arkkitehtuuri

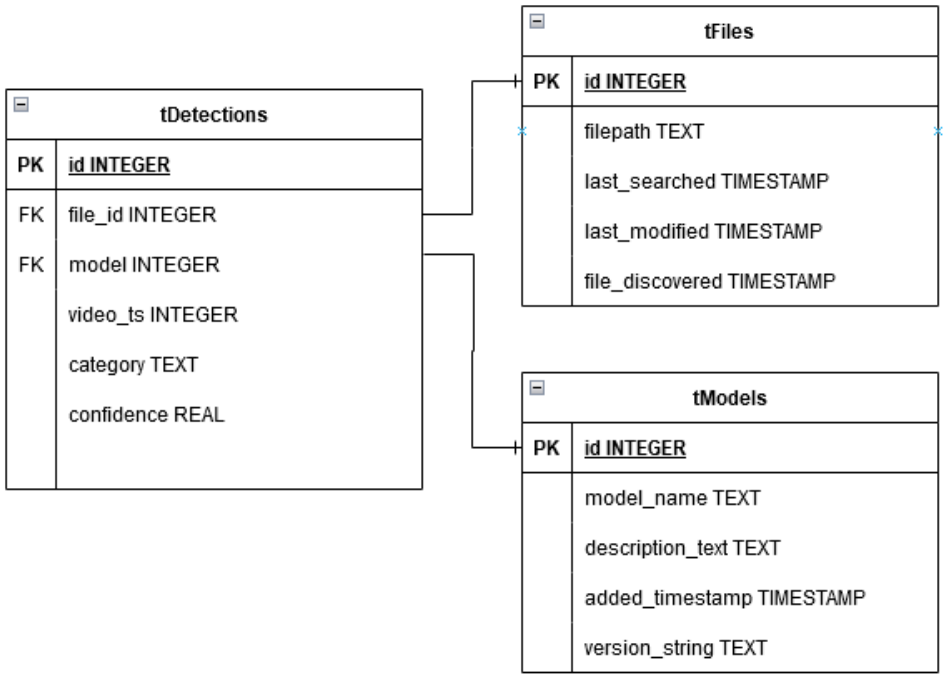
Palvelu kehitettiin kokonaan Docker-virtualisointikontteihin perustuen. Dockerin käyttöjärjestelmätason virtualisointi valittiin helpon liikuteltavuuden takia. Tämän vuoksi palvelu voitiin asentaa sitä hyödyntävälle palvelimelle vaivattomasti. Asennukseen vaadittiin ainoastaan Docker-virtualisointiympäristö ja palvelun Docker-imaget. Palvelu koostui kolmesta Docker imagesta ja niiden tarjoamista alapalveluista: tietokanta-, kartoitus- ja merkintäpalvelusta.

8.4 Tietokantapalvelu

Palvelun toimintaa ja hakutuloksia varten tarvittiin tietokanta. Tietokannaksi valittiin SQLite sen helppokäyttöisyyden vuoksi. SQLiten koko tietokanta on yhdessä tiedostossa, jota oli helppo siirtää ja kopioida palvelua testattaessa ja kehitettäessä. SQLite tukee melkein kaikkia SQL-92 standardin toimintoja, joiden nähtiin olevan tarpeellisia hakutuloksia etsittäessä (Wikipedia 2022b).

Tietokantaan lisättiin kolme taulukkoa: tDetections, tFiles ja tModels. tFiles-taulukkoon kirjattiin kaikki löytyneet tiedostot ja niiden tiedostopolut. Taulukossa pidettiin myös kirjaa tiedoston muokkaamisen ja viimeisen indeksoinnin ajankohdista. tModels-

tietokantataulukkoon kirjattiin käytetyn mallin tiedot, kuten nimi, versio ja kuvaus. tModels-taulukko lisättiin projektin mahdollisen jatkokehitysten vuoksi, esimerkiksi uusien mallien lisäystä varten. tDetections-taulukkoon kirjattiin kaikki yksittäiset mallin tuottamat havainnot. Taulukko koostui kahdesta viiteavaimesta. Yhdellä viiteavaimella viitattiin tiedostoon tFiles-taulukossa, josta havainto oli löydetty, ja toisella viiteavaimella viitattiin malliin, jolla havainto oli tehty. Taulukossa määriteltiin myös ajankohta millisekunteina, kategoria tekstinä ja todennäköisyysarvo reaalilukuna.



Kuvio 35. Tietokantakuvaus

8.5 Kartoituspalvelu

Indeksoitavien tiedostojen etsimistä varten kehitettiin kartoituspalvelu. Palvelun tehtävä oli hakea kaikki videotiedostot määritetystä tiedostopolusta. Uusista, muuttuneista ja poistetuista videotiedostoista lähetettiin tietokantapalvelun rajapintaan viesti kirjanpitoa varten.

8.6 Merkitsemispalvelu

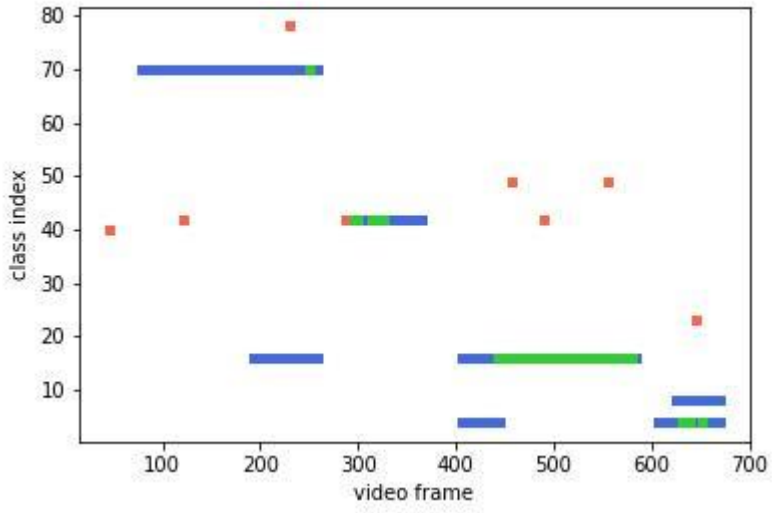
Uusien ja muokattujen tiedostojen merkitsemistä varten kehitettiin erillinen merkitsemispalvelu. Merkitsemispalvelu haki yksi kerrallaan tietokantapalvelusta videotiedoston, jonka last_searched -kentän arvo oli pienempi kuin last_modified, tai file_discovered. Tällaisen

tiedoston löydyttyä merkitsemispalvelu kopio tiedoston tietokannan antamasta polusta merkitsemispalvelun omaan tiedostojärjestelmään. Kopioinnin jälkeen tiedosto avattiin ja video eriteltiin kuviin OpenCV-konenäkökirjastoa hyödyntäen. Jokainen videon kuva skaalattiin uudelleen siten, että suurempi akseli oli maksimissaan 672 pikseliä. Toista akselia uudelleenskaalattiin niin, että kuvasuhde pysyi alkuperäisenä. Uudelleenskaalattu kuva jaettiin pysty- ja vaaka-akseliltaan 224:n pikselin kokoisiin ikkunoihin siten, että ikkunat peittivät toisiaan 50 %. Edellä mainittu operaatio suoritettiin siksi, että mallia ei ole opinnäytetyössä kehitetty lokalisoimaan objekteja kokonaisesta kuvasta. Tällainen toiminnallisuus olisi vaatinut suuria muutoksia neuroverkkomallin arkkitehtuuriin, sovitusdataan sekä perehtymistä aiheeseen Object Detection, jossa neuroverkon viimeiseen kerrokseen lisätään löydetyn kategorian vastaavat pikselikoordinaatit (Brownlee 2021). Kaikki ikkunat ajettiin neuroverkkomallin läpi ja niistä löytyneet kategoriat lähetettiin tietokantapalvelun rajapintaan.

8.7 Prototyypin testaus

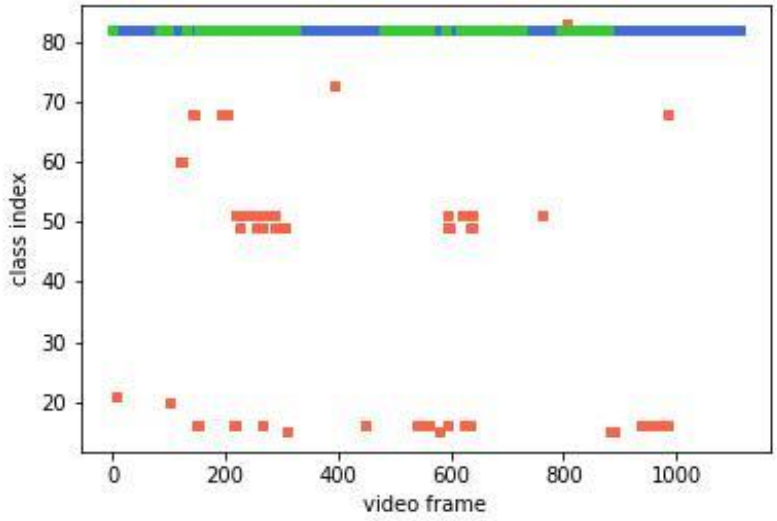
Prototyyppipalvelua kokeiltiin kolmella videolla. Videoista kirjattiin manuaalisesti neuroverkkomallin sovituksessa käytetyt kategoriat. Prototyypin annettiin indeksoida videot, minkä jälkeen mallin havaitsemia kategorioita verrattiin manuaalisesti kirjattuihin vasteisiin. Prototyyppipalvelun havaitsemat ja manuaalisesti merkatut kategoriat piirrettiin aikajanoille videota kohden antamaan paremman tilannekuvan palvelun toiminnasta.

Ensimmäisessä videossa kuvattiin katua, jossa oli penkkejä, ruukutettuja kasveja, autoja, ihmisiä ja polkupyörä. Malli havaitsi neljä viidestä videolla löytyvästä vasteesta. Ainoastaan ihmistä malli ei havainnut. Videolla esiintyneen penkin malli havaitsi vain yhdestä kuvasta noin 250:n kuvan joukosta. Vääriä kategorioita malli havaitsi viisi kappaletta. Näihin kuuluivat: lintu, banaani, moottoripyörä ja rekka. Vääriä kategorioita havaittiin kuitenkin vain yhden kuvan ajan, ja osa väärin tulkituista kategorioista oli vasteiden kanssa hyvin samanlaisia; esimerkiksi moottoripyörä ja polkupyörä.



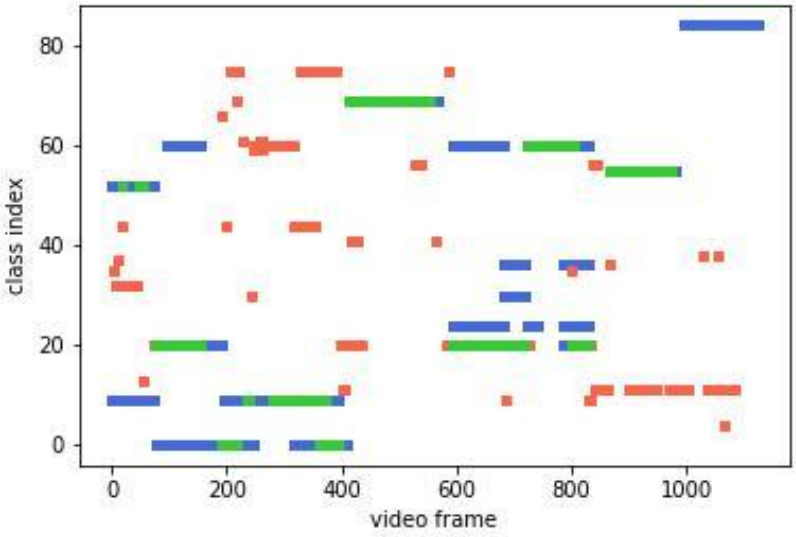
Kuvio 36. Ensimmäisen videon aikajana

Toisella videoista oli osa formulakilpailua, jolloin jokaisessa kuvassa oli havaittavissa vain formuloita. Malli havaitsi formuloita videosta noin puolet ajasta, mutta myös melko paljon vääriä kategorioita. Kategorioihin kuului selittämättömiä vääriä positiivisia havaintoja, kuten näppäimistö ja lentokone. Selvästi väärät kategorioinnit olivat kuitenkin harvinaisia. Enemmän vääriä positiivisia havaintoja löytyi kategorioista: auto, linja-auto ja rekka, jotka ovat keskenään samankaltaisia.



Kuvio 37. Toisen videon aikajana

Viimeisessä videossa oli asunto, jossa oli paljon mallin sovituksessa käytettyjä kategorioita. Malli havaitsi monia kategorioita hyvin; esimerkiksi TV:n, kannettavan tietokoneen, näppäimistön, sohvan, kirjan ja repun. Malli ei kuitenkaan onnistunut havaitsemaan kahvikuppia tai videolla paljon esitettyä 3D-tulostina. 3D-tulostimen tunnistus oli sovitettu mallin Fine-Tuning vaiheessa hyödyntäen ainoastaan kahdeksaa kuvaa, joten malli ei välttämättä ollut onnistunut generalisoimaan 3D-tulostimen kategorisointiin vaadittavia muotoja.



Kuvio 38. Kolmannen videon aikajana

9 Päätelmät

Videoiden indeksointipalvelun kehittäminen on haasteellinen tehtävä, varsinkin jos tarkoituksena on pystyä lisäämään etsittäviä kategorioita pienellä datamäärällä. Merkittävän datamäärän keräämisen esteenä ovat yleensä inhimilliset tekijät, kuten suoritettava manuaalisen työn määrä esimerkkikuvien etsimisessä.

Tässä opinnäytetyössä tutkittiin kolmea eri neuroverkkoarkkitehtuuria: ResNet50 CNN:ää, ohjaamatonta oppimista Autoencoderilla ja siamilaista CNN:ää. ResNet50:n todettiin sovittuvan hyvin suurella datamäärällä, mutta ongelmaksi muodostui uusien kategorioiden lisäys pienellä datamäärällä. Ohjaamaton oppiminen Autoencoderilla ja klusterointi K-Meansilla tuotti käyttökelvottomia klustereita. Siamilaisen CNN:n todettiin soveltuvan käyttötarkoitukseen parhaiten, sillä se pystyi kategorisoimaan uusia kategorioita ilman jatkosovitusta, vaikkakin heikosti. Hienosäätämällä Support Settiä sen toimivuutta voisi parantaa. Edellä mainituista syistä siamilainen CNN valittiin prototyyppipalvelun neuroverkkoarkkitehtuuriksi.

Opinnäytetyössä huomattiin, että lisättäessä paljon uusia kategorioita ja jatkosovitettaessa mallia, on tärkeää pystyä analysoimaan Catastrophic Forgetting -ilmiön vaikutusta mallin toimintaan. On myös löydettävä toimenpiteet sen ehkäisemiseksi. Haasteena on, että jatkosovitettaessa vanhaa dataa joudutaan monissa ehkäisemisen menetelmissä kierrättämään. Tällöin sovitusdatan määrä ja sovitukseen käytettävä aika kasvavat merkittävästi. Tässä opinnäytetyössä malleja testattiin pienellä kategoriamäärällä, joka pitkäaikaisessa käytössä voi kasvaa jopa tuhansiin. Tutkimuksessa sovitettu siamilainen CNN ja sen ympärille kehitetty PoC-palvelu kuitenkin antoivat varmuutta siitä, että ongelma on ratkaistavissa. Oli rohkaisevaa, että siamilainen CNN pystyi rajoitetusti kategorisoimaan uusiin luokkiin kuuluvaa dataa ilman jatkosovitusta.

Kuvadatan kategorisointiin ja lokalisointiin on kehitetty monia muitakin neuroverkkoarkkitehtuureja, joita tässä opinnäytetyössä ei testattu. Esimerkiksi Meta AI:n kehittämä DINO-malli on mielenkiintoinen kokeellinen lähestymistapa kuvadatan kategoriointiin, jolla voidaan hyödyntää täysin vastemerkitsemätöntä dataa (Meta AI 2021). Eri malleja testaamalla opinnäytetyössä kuvattua palvelua voisi jatkokehittää ja sen tunnistuskykyä tarkentaa. Tutkimuksessa käsiteltyjä kategorioita voisi myös yhdistää suurempiin yläkategorioihin, joiden avulla hakutuloksia voisi laajentaa, esimerkiksi yhdistämällä kulkuneuvot omaksi yläkategoriakseen. Lisäksi palveluun voisi lisätä ominaisuuden, jolla käyttäjä pystyisi suodattamaan hakutuloksia todennäköisyysarvon mukaan. Tämän avulla vääriä positiivisia hakutuloksia voitaisiin mahdollisesti vähentää.

Tässä opinnäytetyössä datan käsittely, mallien sovittaminen ja PoC-palvelun koeajo tehtiin yhdellä pöytätietokoneella. Laajemmassa käytössä kaikki toiminnallisuus olisi syytä toteuttaa pilvipalveluissa. Se kuitenkin vaatisi pääomaa, suunnittelua, henkilöresursseja ja työtunteja. Tästä syystä olisi myös hyvä tutkia valmiita samankaltaisia palveluja, kuten Microsoft Azure Video Indexeriä. Azure Video Indexer näyttäisi kykenevän indeksoimaan videoita video- ja audiosisällön mukaan. (Microsoft).

Lähteet

Alexandre, D., Chang, C., Peng, W. & Hang, H. 2019. An Autoencoder-based Learned Image Compressor. Viitattu 15.7.2022. Saatavissa https://arxiv.org/pdf/1902.07385.pdf

Alfeo, A., Manco, G., Ritacco, E. & Giovanni, M. 2020. Using an autoencoder in the design of an anomaly detector for smart manufacturing Viitattu 15.7.2022. Saatavissa https://www.researchgate.net/publication/342310989_Using_an_autoencoder_in_the_design_of_an_anomaly_detector_for_smart_manufacturing

Anttila, P. 1998. Tutkimisen taito ja tiedon hankinta. Viitattu 8.9.2022. Saatavissa https://metodix.fi/2014/05/17/anttila-pirkko-tutkimisen-taito-ja-tiedon-hankinta/#9.2.4%20Dokumenttianalyysi

Brownlee, J. 2021. A Gentle Introduction to Object Recognition With Deep Learning. Viitattu 5.8.2022. Saatavissa https://machinelearningmastery.com/object-recognition-with-deep-learning/

Chandha, G., Das, D. & Karnin, Z. 2018. K-means clustering with Amazon SageMaker. Viitattu 14.7.2022. Saatavissa https://aws.amazon.com/blogs/machine-learning/k-means-clustering-with-amazon-sagemaker/

Chopra, S., Hadsell, R. & LeCun, Y. 2005. Learning a Similarity Metric Discriminatively, with Application to Face Verification. Viitattu 6.7.2022. Saatavissa http://yann.lecun.com/exdb/publis/pdf/chopra-05.pdf

Common Objects in Context. Viitattu 20.7.2022. Saatavissa https://cocodataset.org

Das, S. 2017. CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more. Viitattu 2.8.2022. Saatavissa https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Dertat, A. 2017. Applied Deep Learning – Part 3: Autoencoders. Viitattu 15.7.2022. Saatavissa https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Deshpande, A., Lu, J., Yeh, M., Chong, M. & Forsyth, D. 2017. Learning Diverse Image Colorization. Viitattu 15.7.2022. Saatavissa https://arxiv.org/pdf/1612.01958.pdf

Gad, A. 2020. Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall. Viitattu 21.7.2022. Saatavissa https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/

Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M. & Van Gool, L. 2020. Learning To Classify Images Without Labels. Viitattu 16.7.2022. Saatavissa https://arxiv.org/pdf/2005.12320v1.pdf

Gido, M. & Andreas, S. 2019. Three scenarios for continual learning. Viitattu 18.7.2022. Saatavissa https://arxiv.org/pdf/1904.07734.pdf

Gondara, L. 2016. Medical image denoising using convolutional denoising autoencoders. Viitattu Viitattu 15.7.2022. Saatavissa https://arxiv.org/pdf/1608.04667.pdf

Google. 2022a. Classification: Accuracy. Viitattu 9.9.2022. Saatavissa https://developers.google.com/machine-learning/crash-course/classification/accuracy

Google. 2022b. Classification: Precision and Recall. Viitattu 9.9.2022. Saatavissa https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

Gupta, A. 2022. Elbow Method for optimal value of K in KMeans. Viitattu 14.7.2022. Saatavissa https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

He, K., Zhang X., Ren, S. & Sun J. 2015. Deep Residual Learning for Image Recognition. Viitattu 2.7.2022. Saatavissa https://arxiv.org/pdf/1512.03385.pdf

Hu, J., Shen, Li., Albanie, S., Sun G. & Wu E. 2019. Squeeze-and-Excitation Networks. Viitattu 2.7.2022. Saatavissa https://arxiv.org/pdf/1709.01507.pdf

IBM. 2020. Convolutional Neural Networks. Viitattu 28.6.2022. Saatavissa https://www.ibm.com/cloud/learn/convolutional-neural-networks

IBM. 2017. Unsupervised learning for data classification. Viitattu 14.7.2022. Saatavissa https://developer.ibm.com/articles/cc-unsupervised-learning-data-classification/

ImageNet. 2021. About ImageNet. Viitattu 31.6.2022. Saatavissa https://image-net.org/about.php

Kang, D., Park J. & Duong P. 2020. Application of Deep Learning in Dentistry and Implantology. Viitattu 31.6.2022. Saatavissa https://www.researchgate.net/profile/Dae-YoungKang/publication/346091812_Application_of_Deep_Learning_in_Dentistry_and_Implantology/links/5ffedb0492851c13fe0a39cc/Application-of-Deep-Learning-in-Dentistry-and-Implantology.pdf

Keras. Keras Applications. Viitattu 2.7.2022. Saatavissa https://keras.io/api/applications/

Krizhevsky, A., Sutskever, I. & Hinton, G. 2012. ImageNet Classification with Deep Convolutional Neural Networks. Viitattu 1.7.2022. Saatavissa https://www.image-net.org/static_files/files/supervision.pdf

Koch, G., Zemel, R. & Salakhutdinov, R. 2015. Siamese Neural Networks for One-shot Image Recognition. Viitattu 6.7.2022. Saatavissa https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

Lavrenko, V. 2014. K-means clustering: how it works. Viitattu 14.7.2022. Saatavissa https://www.youtube.com/watch?v=_aWzGGNrcic

Li, Z. & Hoiem, D. 2017. Learning without Forgetting. Viitattu 18.7.2022. Saatavissa https://arxiv.org/pdf/1606.09282.pdf

Lin, T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. & Dollár, P. 2015. Microsoft COCO: Common Objects in Context. Viitattu 20.7.2022. Saatavissa https://arxiv.org/pdf/1405.0312.pdf

Lu, S. & Li, R. 2021. DAC: Deep Autoencoder-based Clustering, a General Deep Learning Framework of Representation Learning. Viitattu 16.7.2022. Saatavissa https://arxiv.org/pdf/2102.07472.pdf

Lukka, K. 2001. Konstruktiivinen tutkimusote. Viitattu 8.9.2022. Saatavissa https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/

Lyashenko, V. 2022. Understanding Few-Shot Learning in Computer Vision: What You Need to Know. Viitattu 7.7.2022. Saatavissa https://neptune.ai/blog/understanding-few-shot-learning-in-computer-vision

Meta AI. 2021. Advancing the state of the art in computer vision with self-supervised Transformers and 10x more efficient training. Viitattu 6.8.2022. Saatavissa https://ai.facebook.com/blog/dino-paws-computer-vision-with-self-supervised-transformers-and-10x-more-efficient-training/

Microsoft. Unlock video insights. Viitattu 6.8.2022. Saatavissa https://vi.microsoft.com/en-us

Moindrot, O. 2018. Triplet Loss and Online Triplet Mining in TensorFlow. Viitattu 11.7.2022. Saatavissa https://omoindrot.github.io/triplet-loss

Monn, D. 2017. Denoising Autoencoders explained. Viitattu 15.7.2022. Saatavissa https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2

Ng, A. 2017a. C4W2L04 Why ResNets Work. Viitattu 2.7.2022. Saatavissa https://www.youtube.com/watch?v=RYth6EbBUqM

Ng, A. 2017b. Transfer Learning (C3W2L07). Viitattu 18.7.2022. Saatavissa https://www.youtube.com/watch?v=yofjFQddwHE

Northeastern University. 2015. Relevance, Precision, and Recall. Viitattu 9.9.2022. Saatavissa https://course.ccs.neu.edu/cs6200sp15/slides/m06.s02%20-%20relevance,%20precision,%20and%20recall.pdf

Palash, Y. & Young-Gab, K. 2022. Real-Time Abnormal Object Detection for Video Surveillance in Smart Cities. Viitattu 26.6.2022. Saatavissa https://mdpi-res.com/d_attachment/sensors/sensors-22-03862/article_deploy/sensors-22-03862-v2.pdf

Papers With Code. Continual Learning. Viitattu 18.7.2022. Saatavissa https://paperswithcode.com/task/continual-learning

Piech, C. 2013. K Means. Viitattu 14.7.2022. Saatavissa https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Redmon, J. & Farhadi, A. 2018. YOLOv3: An Incremental Improvement. Viitattu 2.7.2022. Saatavissa https://arxiv.org/pdf/1804.02767.pdf

Rohrer, R. 2016. How Convolutional Neural Networks Work. Viitattu 28.6.2022. Saatavissa https://www.youtube.com/watch?v=FmpDIaiMIeA

Rosebrock, A. 2019. Keras: Feature extraction on large datasets with Deep Learning. Viitattu 3.7.2022. Saatavissa https://pyimagesearch.com/2019/05/27/keras-feature-extraction-on-large-datasets-with-deep-learning/

Sachan, K. 2017. Detailed Guide to Understand and Implement ResNets. Viitattu 21.7.2022. Saatavissa https://cv-tricks.com/keras/understand-implement-resnets/

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L. 2019. MobileNetV2: Inverted Residuals and Linear Bottlenecks. Viitattu 3.7.2022. Saatavissa https://arxiv.org/pdf/1801.04381.pdf

Schroff, F., Kalenichenko, D. & Philbin, J. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. Viitattu 10.7.2022. Saatavissa https://arxiv.org/pdf/1503.03832.pdf

Scipy. 2022. Scipy.spatial.distance.cosine. Viitattu 25.7.2022. Saatavissa https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cosine.html

Shusen, W. 2020a. Few-Shot Learning: Basic Concepts. Viitattu 9.7.2022. Saatavissa https://www.youtube.com/watch?v=hE7eGew4eeg

Shusen, W. 2020b. Few-Shot Learning: Pretraining + Fine-tuning. Viitattu 9.7.2022. Saatavissa https://www.youtube.com/watch?v=U6uFOIURcD0

Sick, L. 2021. Paper explained: Masked Autoencoders Are Scalable Vision Learners. Viitattu 16.7.2022. Saatavissa https://towardsdatascience.com/paper-explained-masked-autoencoders-are-scalable-vision-learners-9dea5c5c91f0

Simonyan, K. & Zisserman A. 2015. Very Deep Convolutional Networks For Large-Scale Image Recognition. Viitattu 1.7.2022. Saatavissa https://arxiv.org/pdf/1409.1556.pdf

Stanford Vision Lab. 2012. Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). Viitattu 31.6.2022. Saatavissa https://image-net.org/challenges/LSVRC/2012/results.html

TensorFlow. 2022a. Why TensorFlow?. Viitattu 8.9.2022. Saatavissa https://www.tensorflow.org/about

TensorFlow. 2022b. tf.keras.losses.BinaryCrossentropy. Viitattu 7.7.2022. Saatavissa https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy

TensorFlow. 2022c. tf.keras.losses.CategoricalCrossentropy. Viitattu 21.7.2022. Saatavissa https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

TensorFlow. 2022d. tfa.losses.TripletHardLoss. Viitattu 1.8.2022. Saatavissa https://www.tensorflow.org/addons/api_docs/python/tfa/losses/TripletHardLoss

Wikipedia. 2022a. Convolutional Neural Network. Viitattu 28.6.2022. Saatavissa https://en.wikipedia.org/wiki/Convolutional_neural_network

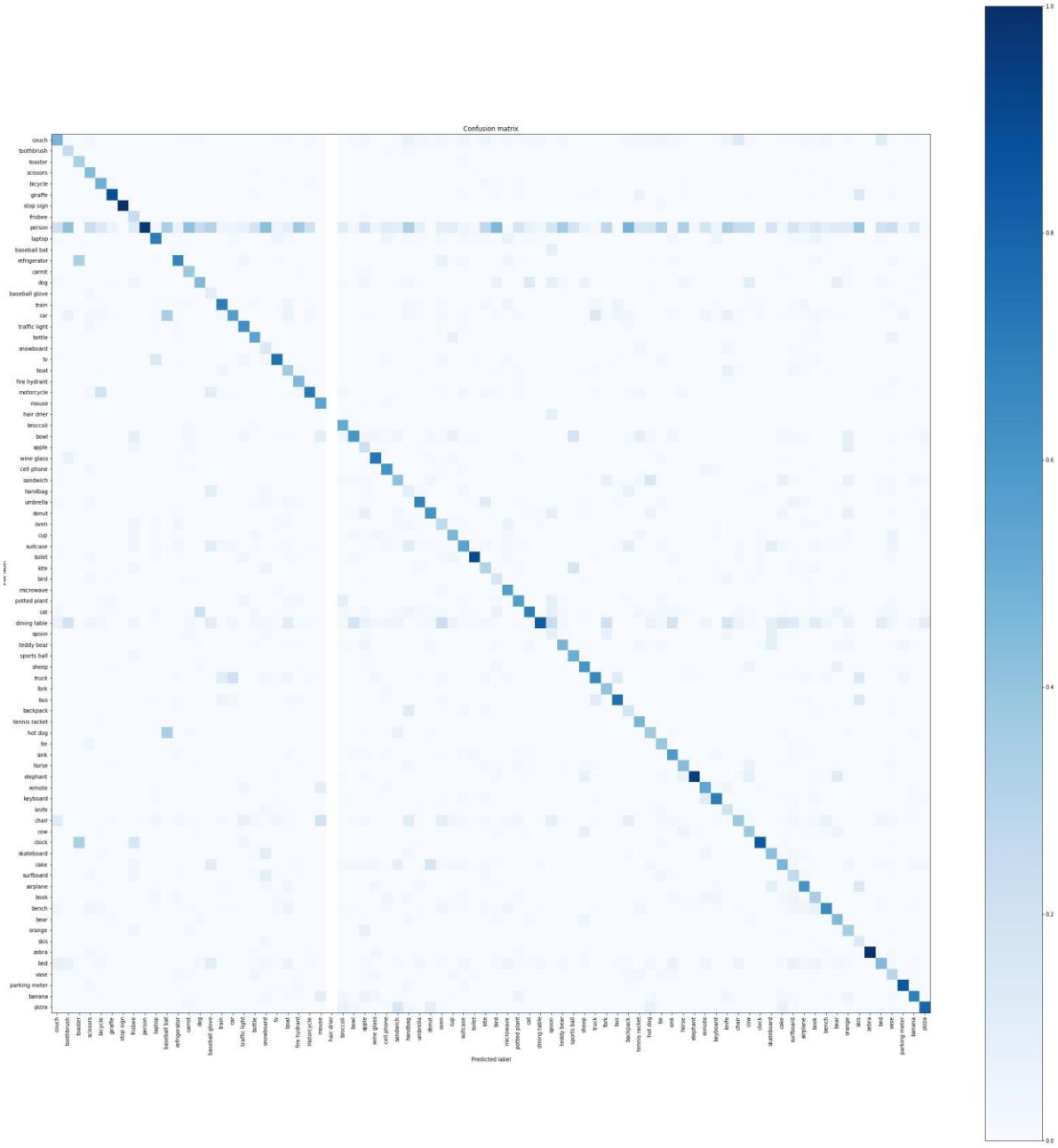
Wikipedia. 2022b. SQLite. Viitattu 5.8.2022. Saatavissa https://en.wikipedia.org/wiki/SQLite

Zhu, Q. & Zhengyong, W. 2019. An Image Clustering Auto-Encoder Based on Predefined Evenly-Distributed Class Centroids and MMD Distance. Viitattu 15.7.2022. Saatavissa https://arxiv.org/pdf/1906.03905.pdf

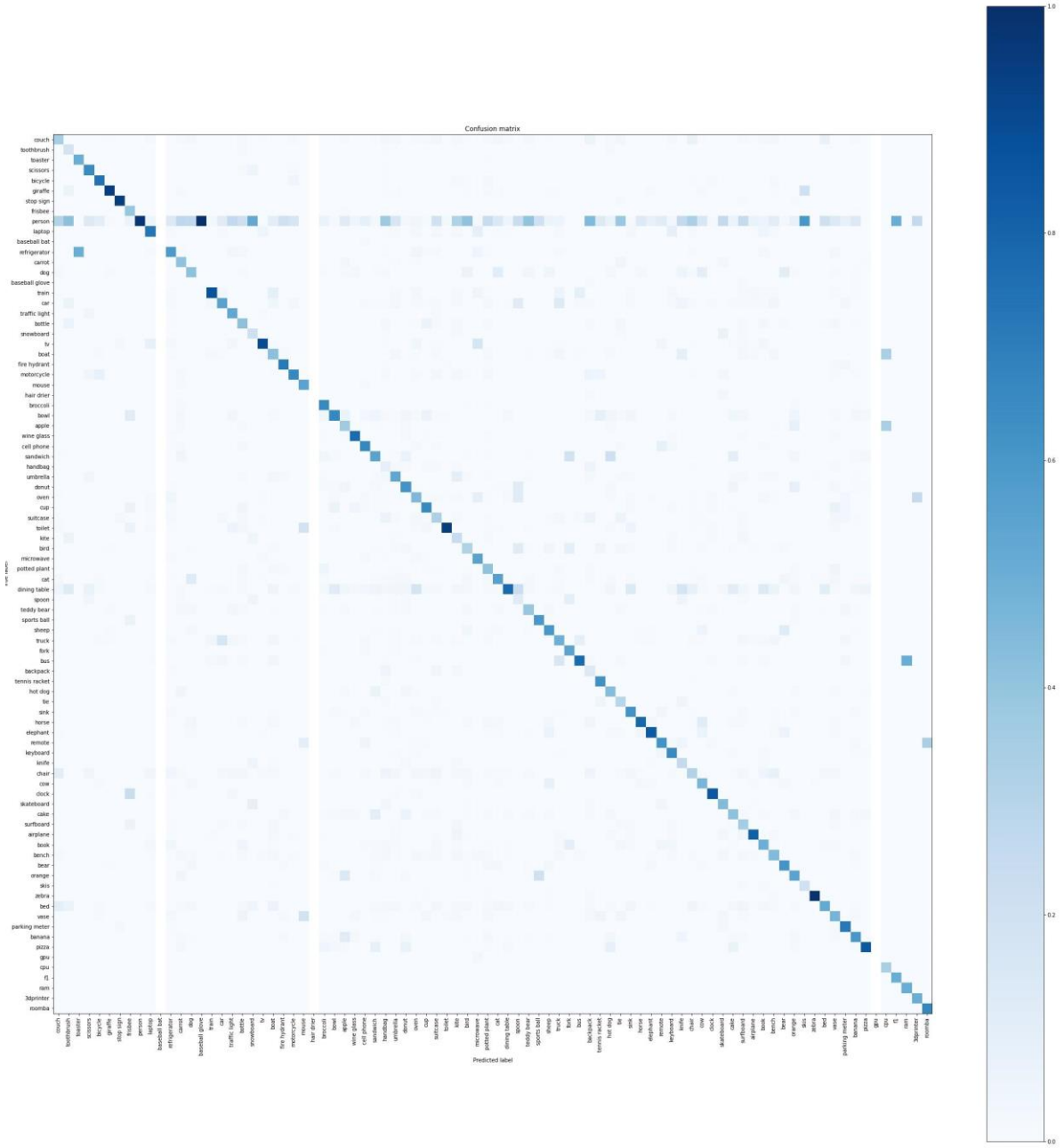
Liite 1. Kategoriataulukko

0	couch	43	cat
1	toothbrush	44	dining table
2	toaster	45	spoon
3	scissors	46	teddy bear
4	bicycle	47	sports ball
5	giraffe	48	sheep
6	stop sign	49	truck
7	frisbee	50	fork
8	person	51	bus
9	laptop	52	backpack
10	baseball bat	53	tennis racket
11	refrigerator	54	hot dog
12	carrot	55	tie
13	dog	56	sink
14	baseball glove	57	horse
15	train	58	elephant
16	car	59	remote
17	traffic light	60	keyboard
18	bottle	61	knife
19	snowboard	62	chair
20	tv	63	cow
21	boat	64	clock
22	fire hydrant	65	skateboard
23	motorcycle	66	cake
24	mouse	67	surfboard
25	hair drier	68	airplane
26	broccoli	69	book
27	bowl	70	bench
28	apple	71	bear
29	wine glass	72	orange
30	cell phone	73	skis
31	sandwich	74	zebra
32	handbag	75	bed
33	umbrella	76	vase
34	donut	77	parking meter
35	oven	78	banana
36	cup	79	pizza
37	suitcase	80	gpu
38	toilet	81	cpu
39	kite	82	f1
40	bird	83	ram
41	microwave	84	3dprinter
42	potted plant	85	roomba

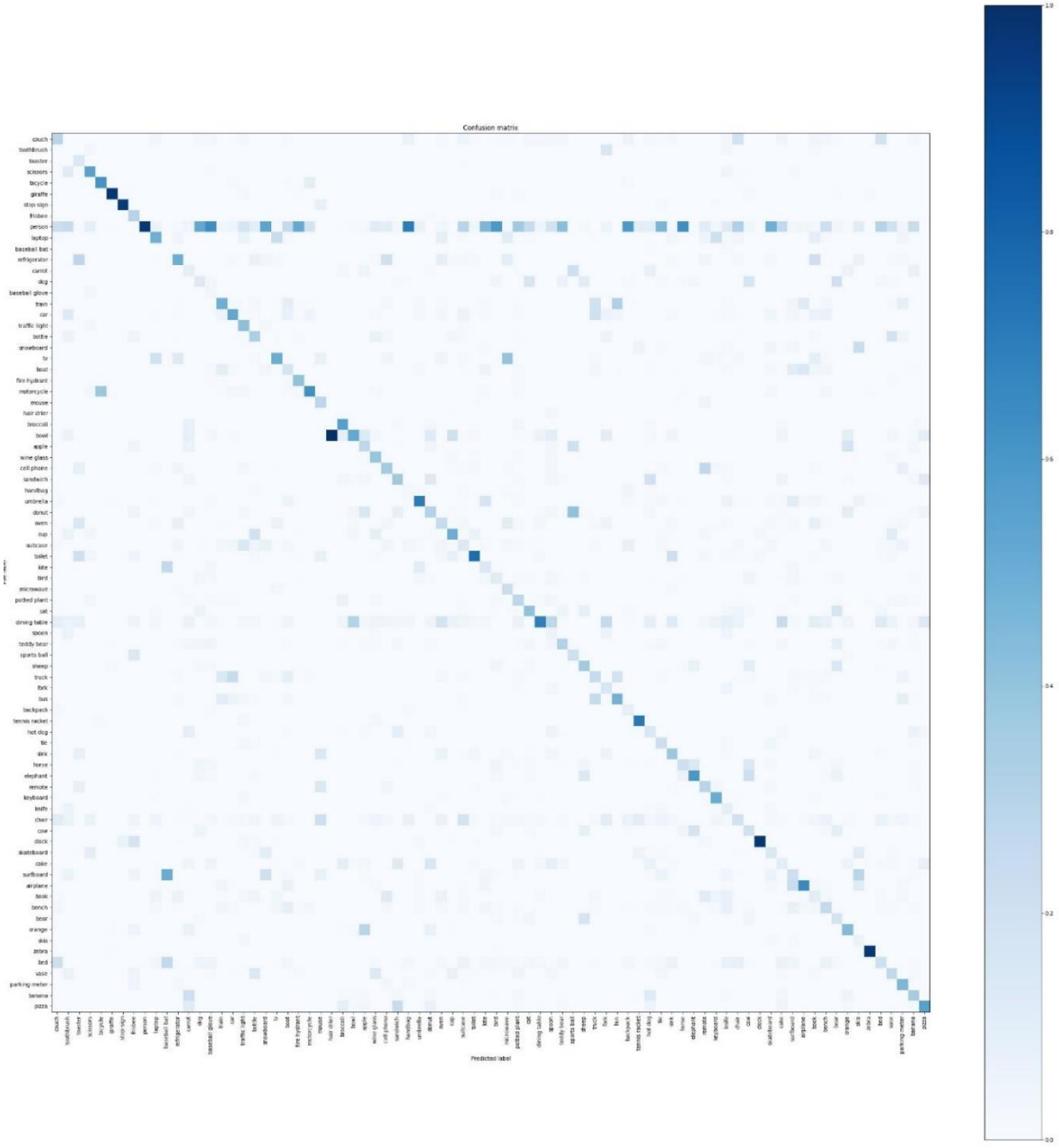
Liite 2. ResNet50V2 Pre-Training Confusion Matrix



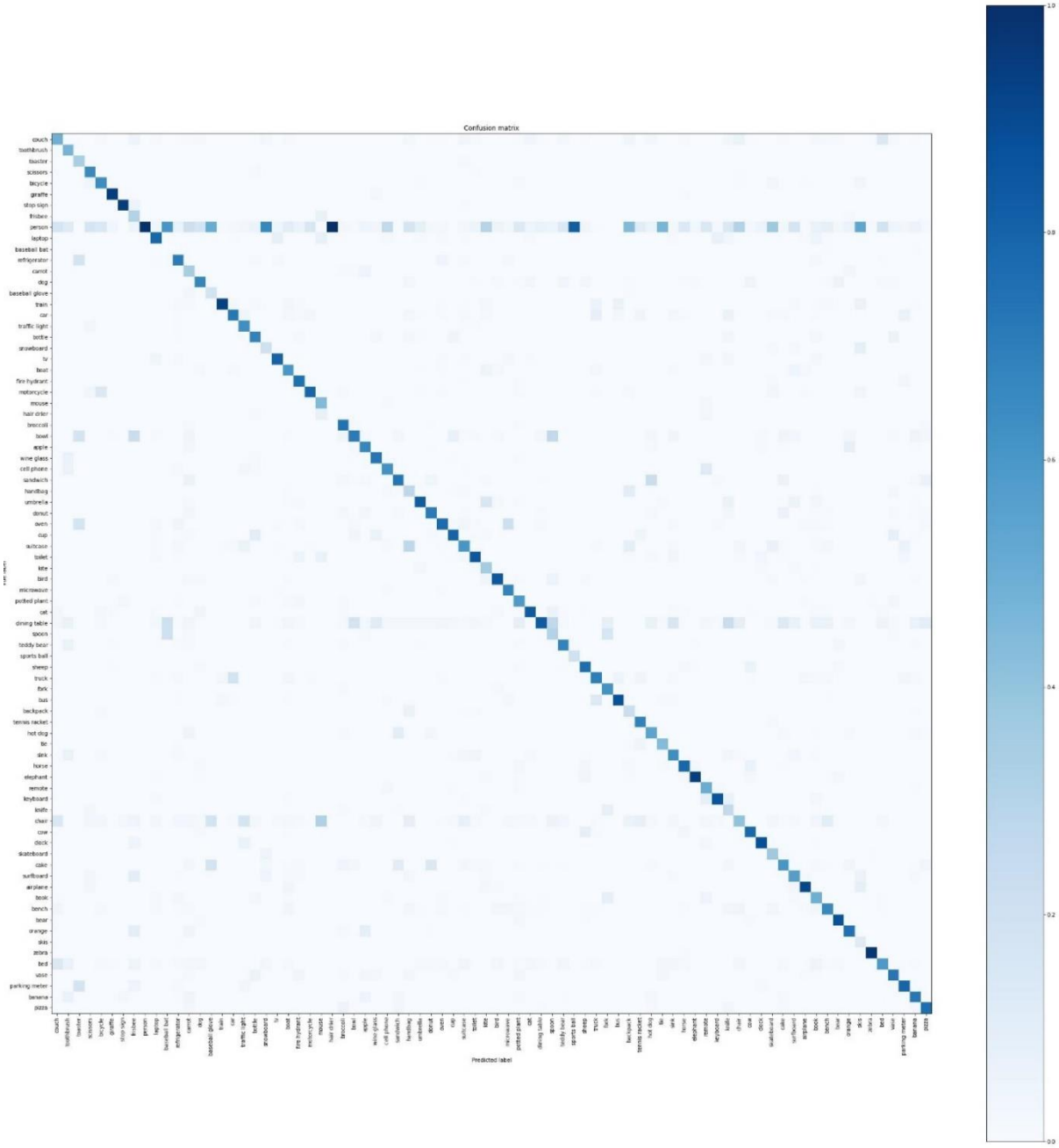
Liite 3. ResNet50V2 Fine-Tuning Confusion Matrix



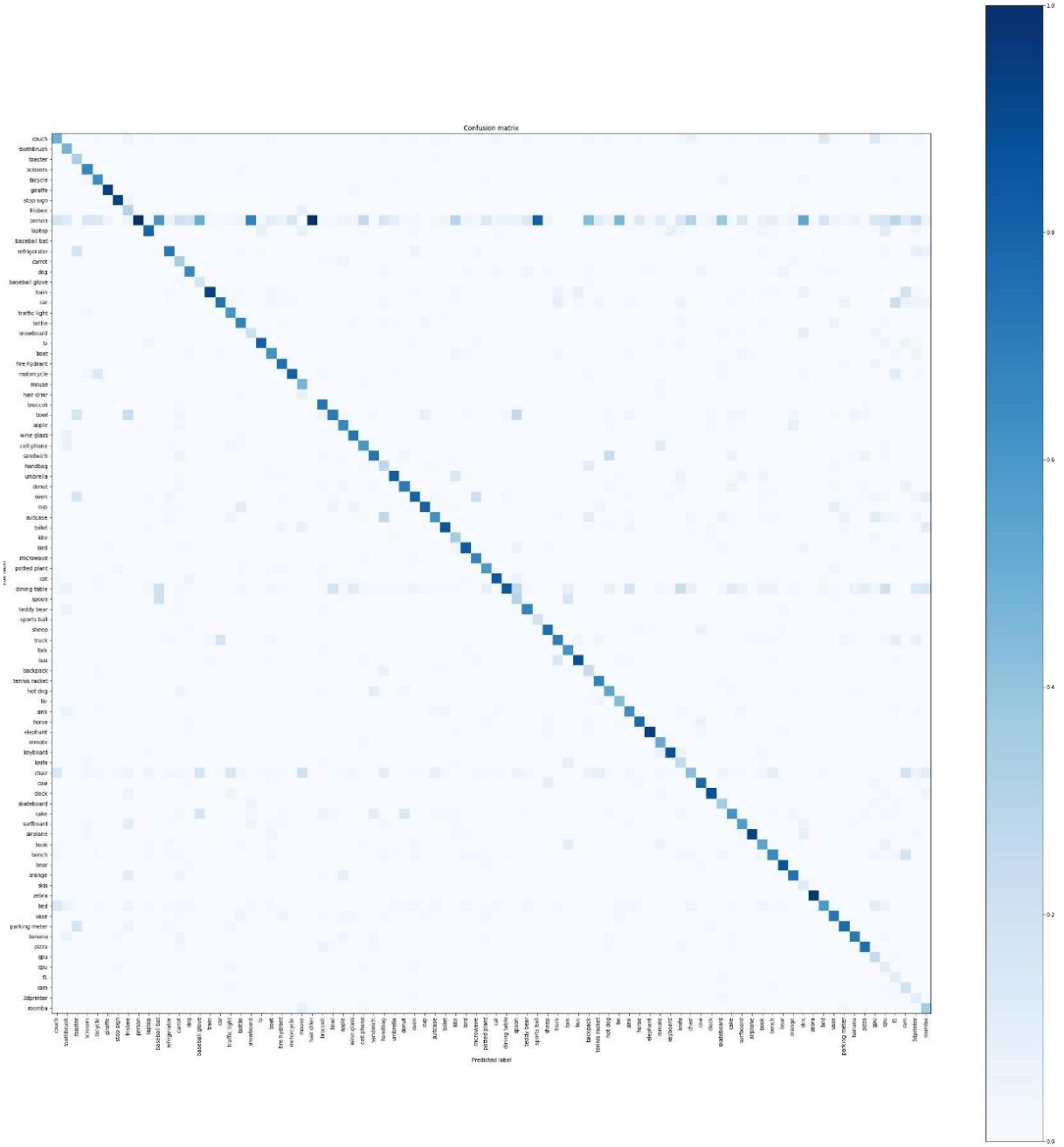
Liite 4. Siamilainen CNN Confusion Matrix



Liite 5. Siamese CNN Triplet Loss Pre-Training Confusion Matrix



Liite 6. Siamese CNN Triplet Loss Pre-Training Confusion Matrix kuudella lisätyllä kategorialla



Liite 7. Siamese CNN Triplet Loss Fine-Tuning Confusion Matrix