

3D konvoluutioneuroverkkojen soveltamisesta piikiekkoaineiston analyysissä

Otto Pajunen

Perustieteiden korkeakoulu

Kandidaatintyö
Espoo 27.1.2025

Vastuuopettaja

Prof. Pauliina Ilmonen

Työn ohjaaja

DI Mikko Vaulanen



Copyright © 2025 Otto Pajunen

The document can be stored and made available to the public on the open internet pages of Aalto University.
All other rights are reserved.

Tekijä Otto Pajunen**Työn nimi** 3D konvoluutioneuroverkkojen soveltamisesta piikiekkoaineiston analyysissä**Koulutusohjelma** Teknistieteellinen kandidaattiohjelma**Pääaine** Matematiikka ja systeemitietee**Pääaineen koodi** SCI3029**Vastuuopettaja** Prof. Pauliina Ilmonen**Työn ohjaaja** DI Mikko Vaulanen**Päivämäärä** 27.1.2025**Sivumäärä** 36+1**Kieli** Suomi**Tiivistelmä**

MEMS-piikiekkojen manuaalinen laadunvalvonta on työlästä ja altista ihmisen tekemille virheille. Lähes jokaisessa teollisessa prosessissa valmistetussa piikiekossa havaitaan ainakin muutama neliömicrometriin kokoinen vika. Piikiekossa on yli 3300 anturia, jotka sahataan erilleen vasta valmistuksen jälkeen, joten satunnaiset anturikohtaiset viat eivät vielä tarkoita, että koko komponentti olisi viallinen.

Tämän kandidaatintutkielman tavoitteena on luoda Murata Electronics Oy:n toimeksiannosta pohjaa ratkaisulle, jossa piikiekot esitetään kaksi- tai kolmiulotteisina vektoreina niiden pääpiirteet säilyttäen. Piikiekkoaineiston ulottuvuuden alentaminen tehdään pääkomponenttianalyysillä (PCA) sekä 3D konvoluutioneuroverkosta (CNN) eristetyllä enkooderilla, ja saatuja tuloksia vertaillaan. Alempiulotteinen esitys mahdollistaa piikiekkojen vertailun tulkitsemalla kuvaa, jossa samankaltaiset piikiekot ryhmittyvät. Tutkielmana halutaan myös selvittää, kuinka teollista aineistoa tulee esikäsitellä parhaan tuloksen saamiseksi. Aineiston esitys alemmassa ulottuvuudessa pyritään klusterointimaan todennäköisesti, eli ryhmittelemään ohjaamattomasti piikiekkojen kategorioihin.

Kun laadunvalvonta saa tuekseen piikiekon pääpiirteitä kuvaavan vektorin, piikiekko voidaan arvioida tukeutuen menneisyydessä valmistettujen piikiekkojen vektoritietokantaan. Tarkka viantunnistus johtaa nopeampiin jatkotoimenpiteisiin, jotka tukevat koko tuotantolinjaa.

Tulosten perusteella molemmat mallit ulottuvuuden alentamiseen hukkasivat paljon tietoa. Pääkomponenttianalyysin lineaarinen piirteiden muunnos todettiin epätarkaksi ja metodi hylättiin. Enkooderi löysi aineistosta epälineaarisia riippuvuuksia ja onnistui erottelemaan joitakin piikiekkojen valmistusvaiheen yleisvikoja. Enkooderin todettiin olevan herkkä, sillä sen tekemässä muunnoksessa oli selkeitä eroja konvoluutioneuroverkon koulutuskertojen välillä. Stabiilimman mallin saavuttamiseksi tarvitaan merkittävästi enemmän dataa CNN:n koulutusvaiheeseen. Klusterointi oli liian yleistävä, mutta sen tulos on hyödyllinen todellisten ryhmittymien manuaalisessa etsinnässä.

Avainsanat Kategorisointi, klusterointi, neuroverkko, piikiekko, pääkomponenttianalyysi, vian tunnistus



Author Otto Pajunen

Title Application of 3D Convolutional Neural Networks in the Analysis of Wafer Data

Degree programme Bachelor's Programme in Science and Technology

Major Mathematics and Systems Sciences

Code of major SCI3029

Teacher in charge Prof. Pauliina Ilmonen

Advisor DI Mikko Vaulanen

Date 27.1.2025

Number of pages 36+1

Language Finnish

Abstract

Manual quality control of MEMS wafers is labor-intensive and prone to human error. In almost every industrially manufactured wafer, at least a few square-micrometer-sized defects are observed. A wafer contains over 3300 sensors, which are only separated after production, meaning that random sensor-specific defects do not necessarily render the entire component faulty.

The aim of this bachelor's thesis is to lay the groundwork for a solution assigned by Murata Electronics, where wafers are represented as two- or three-dimensional vectors while preserving their key features. Dimension reduction of the wafer data is done using principal component analysis (PCA), and an encoder extracted from a 3D convolutional neural network (CNN). The results obtained are compared. The lower-dimensional representation enables the comparison of wafers by interpreting a visualization where similar wafers are grouped together. The thesis also seeks to determine how the industrial wafer data should be preprocessed to achieve the best results. One goal is also to cluster the data in the lower-dimensional representation accurately, that is grouping the wafers into categories in an unsupervised manner.

When quality control is supported by a vector representing the key features of a wafer, the wafer can be categorized by comparing it to a vector database of previously manufactured wafers. Accurate defect detection enables faster follow-up actions, which benefit the entire production line.

The results showed that both models for dimension reduction lost a significant amount of information. The linear feature transformation of principal component analysis was found to be inaccurate, leading to the method being discarded. The encoder identified nonlinear dependencies in the data and succeeded in distinguishing some common manufacturing defects in the wafers. However, the encoder was found to be sensitive, as the transformations it produced showed noticeable variations between different training iterations of the convolutional neural network. Achieving a more stable model will require significantly more data for CNN training. The clustering was found to be overly generalizing, but its output is helpful in the manual identification of actual groupings.

Keywords Categorization, clustering, fault detection, neural network, principal component analysis, wafer

Sisällys

Tiivistelmä	3
Tiivistelmä (englanniksi)	4
Sisällys	5
1 Johdanto	6
2 Aineisto	7
2.1 Teollinen data	7
2.2 Muunnos bittikartoiksi	8
3 Ulottuvuuden alentaminen	11
3.1 Piirteiden valinta ja muunto	11
3.2 Datan skaalaus	13
3.3 Pääkomponenttianalyysi	14
3.4 Konvoluutioneuroverkko	17
4 Klusterointi	24
5 Tulokset	27
5.1 Lopullinen malli ulottuvuuden alentamiseen	27
5.2 Enkooderin herkkyys	29
5.3 Enkooderin validointi	30
5.4 Klusterointi	31
6 Yhteenvetö	32
6.1 Yhteenvetö	32
6.2 Rajoitteet ja tulevaisuuden tutkimus	32

1 Johdanto

Teollisessa prosessissa valmistetun MEMS-piikiekon laatua arvioidaan manuaalisesti 28 kuvan perusteella, jolloin yhdellä piikiekolla on yli 120 000 niitä karakterisoivaa piirrettä. Manuaalinen laadunvalvonta on työlästä ja altista ihmisen tekemille virheille. Tässä opinnäytetyössä luotiin Murata Electronics Oy:n toimeksiannosta pohjaa ratkaisulle, jossa laadunvalvonnan suorittajalla on tukenaan piikiekon pääpiirteitä kuvaava vektori. Vektoriesityksen avulla valmistettua piikiekkoja voitaisiin arvioida tukeutuen massiviiseen menneisyydessä valmistettujen piikiekkojen vektoritietokantaan. Tarkka viantunnistus johtaa nopeampiin jatkotoimenpiteisiin, jotka tukevat koko tuotantolinjaa. Mikäli piikiekkojen vektoriesitys olisi todella tarkka komponenttien kategorisoinnissa, laadunvalvonta voitaisiin automatisoida. Tämä parantaisi tuotannon tarkkuutta ja laskisi kustannuksia.

Opinnäytetyön pääkysymys on, pystytäänkö kahden valitun koneoppimismenetelmän eli 1) Pääkomponenttianalyysin ja 2) 3D-konvoluutioneuroverkon avulla muuntamaan piikiekkoaineisto kaksi- tai kolmiulotteisiksi vektoreiksi niin, että alemmassa ulottuvuudessa niiden pääkategoriat erottuisivat todennäköisesti ryhminä kuvaa tulkitessa. Muita aineiston muuntomenetelmiä ei käytetä. Dimension alentaminen tehdään maksimissaan kolmeen ulottuvuuteen, vaikka pidempi vektoriesitys voisi karakterisoida aineistoa tarkemmin. Tällä valinnalla muunnettua aineisto on mahdollista havainnollistaa euklidisessa koordinaatistossa kuvana. Osaongelmina ovat 1) miten aineistoa tulisi esikäsitellä parhaiden tulosten saavuttamiseksi ja 2) mikäli aineiston esitys alemmassa ulottuvuudessa on hyväksyttävä, voidaanko aineisto klusteroida eli ryhmitellä ohjaamattomalla koneoppimismenetelmällä todennäköisintä kategoriaihin.

MEMS-piikiekot (Micro Electro Mechanical System) ovat Muratan valmistamia puolijohdekomponentteja. Piikiekolla olevat anturit sahataan yksittäisiksi, jonka jälkeen niitä käytetään muun muuassa autojen ajonvakautusjärjestelmissä. Yhdellä piikiekolla on lukuisia antureita, jotka kuvataan sahauksen jälkeen piikiekkohtaisesti automaattisella kuvantamislaitteella. Kuvantamislaitteen tuottama data on se aineisto, jolla opinnäytetyö tehdään. Kuvantamislaitte tunnistaa automaattisesti antureiden pinnalta 14 eri esikategorisoitua vikaluuokkaa. Tietyllä tavalla vioittuneet anturit muodostavat piikiekon vikakategoriat, joiden toivotaan hahmottuvan muunnetussa aineistossa. [Vaulanen](#)

Osiossa [2](#) esitellään teollinen aineisto ja sen muunto bittikartoiksi, eli kuviksi. Osiossa [3](#) esitellään tapoja suorittaa ulottuvuuden alentaminen, kerrotaan aineiston skaalaamisesta menetelmille, sekä esitellään pääkomponenttianalyysi ja konvoluutioneuroverkko. Osio [4](#) käsittelee klusterointia, eli muunnetun aineiston ohjaamatonta ryhmittelyä. Opinnäytetyön tulokset esitellään osiossa [5](#), jossa myös vastataan johdannossa esitettyihin tutkimuskysymyksiin ja arvioidaan tulosten herkyyttä ja luotettavuutta. Lopuksi osiossa [6](#) kerrotaan tiiviisti, päästiinkö tavoitteisiin ja ohjataan tulevaisuuden tutkimuksen suuntaa antamalla ideoita kattavamman analyysin suorittamiseen. Lisäksi esitetään, miten laadunvalvonnan automatisointi voitaisiin toteuttaa tulevaisuudessa, pohditaan työtä rajoittaneita tekijöitä, sekä ehdotetaan lisäselvitystä vaativia rakenne- ja parametrimuutoksia käytetylle menetelmille.

2 Aineisto

2.1 Teollinen data

Työssä käytettävä data on Murata Technologies Oy:n työntekijän ja samalla tämän kandidaatintyon ohjaajan DI Mikko Vaulasen kokoama. Data koostuu 3949:stä unikista Muratan valmistamasta piikiekosta, missä jokainen kiekko muodostaa yksittäisen datapisteen. Piikiekkojen tunnistenumerot (ID:t) ovat anonymisoitu ja dataan merkittyjä kiekkojen valmistuseriä ei huomioida. Yksittäisellä piikiekolla on 3300 anturia, ja antureiden ajatellaan olevan jokaisella piikiekolla samoissa euklidisen avaruuden kaksiulotteisen tason pisteissä. Piikiekon keskikohta on tason origossa. Antureilta on mitattu vikoja automaattisella kuvantamislaitteella (WAVI, Wafer automatic visual inspection). Laite pystyy havaitsemaan 13 erilaista esikategorisoitua vikaa, eli defektityyppiä, sekä yhden kategorian, jolla esitetään ominaisuutta 'ei vikaa'. Joitakin vikakategorioita havaitaan piikiekolla useissa kohdissa, toisia ilmaantuu harvemmin, jos ollenkaan.

Havaintoja antureiden defekteistä on ilmaistu datassa kahdella eri tavalla. Defektityyppien ilmaisutavat selviävät taulukosta 1. Tapa 1 kuvailee havaitun defektityyppin kappalemäärää kyseisellä anturilla. Tämä havainto on merkity dataan luonnollisena lukuna. Anturin mittaan defektityyppin lukumäärä on useimmissa tapauksissa 0 tai 1. Tapa 2 kuvailee, mikä on tietyn defektityyppin suuruus kyseisellä anturilla. Tämä on ilmaistu defektityyppin yhteenlaskettuna pinta-alana anturin mittausalueella. Pinta-alaa kuvataan datassa jatkuvana lukuna. Pinta-alan yksikköä ei ole kerrottu. Mikäli anturin mittausalueella ei ole havaittu kyseistä vikatyyppiä, pinta-alan arvoksi määrityy 0. Yksittäisten anturien havaitsemien defektien pinta-alat ovat tyypillisesti keskenään samaa suuruuluokkaa. Pinta-alaltaan suurimmat defektit ovat noin 300-kertaisia tyypilliseen defektiin verrattuna. Defektien pinta-alojen suuruudet ovat samaa luokkaa eri vikakategorioiden välillä. Tavassa 2 'ei vikaa'-kategoria on turha, sillä viattomalle alueelle ei voi laskea vian pinta-alaa. Vikatyyppien tarkempia tietoihin ei syvennytä, sillä ne ovat yrityksen ominaisuutta. [Vaulanen](#)

Taulukko 1: Anturin havaitsemien defektien ilmaisutavat datassa eri kategorioille.

Ilmaisutapa	Vika	Ei vikaa
Kappalemäärä	Luonnollinen luku \mathbb{N}	Binäär $\{0, 1\}$
Pinta-ala	Jatkuva luku $[0, \infty)$	Aina 0

Datan komponentteja voidaan ilmaista merkinnällä x_{i,j,k_a} , jossa $i \in \{1, \dots, I\}$ viittaa datapisteeseen, $j \in \{1, \dots, J\}$ anturin koordinaatteihin tasossa ja $k_a \in \{1, \dots, K\}$ anturin kategorian. Muuttujan k_a alaindeksi $a \in \{1, 2\}$ määrittää, mitä kategorian k ilmaisutapaa tutkitaan. Anturit sekä niiden kategoriat muodostavat datapisteiden piirteet. Jokaista piikiekkoa karakterisoi $m = J \times K_1 \times K_2 \sim 226000$ piirrettä. Koneoppimisen sovelluksessa datapisteiden mahdollisten piirrevektorien joukkoon viitataan nimellä piirreavaruuus $\chi \in \mathbb{R}^m$ ([Jung, 2022](#), s. 23).

Dataassa anturit on järjestetty niiden sijainnin mukaan piikiekon pinnalla kaksilottiseen koordinaatistoon. Anturien sijainteja piikiekolla kuvataan pisteinä, mutta todellisuudessa ne ovat muutaman neliömillimetrin kokoisia. Dataan pohjalta pystymme siis paikantamaan jokaisen vikaluojan määrään sekä sijainnin suurella tarkkuudella piikiekon pinnan eri kohdissa.

Datapisteitä ei ole nimetty, eli piikiekkoja ei olla luokiteltu ammattilaisen toimesta ehjiin, viallisiin tai muuten massasta poikkeaviin. Ilman tietämystä piikiekoista ja sen antureiden vikakategorioista ei pystytä käytettävissä olevan teollisen dataan pohjalta kertomaan piikiekkojen eli datapisteiden yleispiirteistä. Dataan nimeämättömyys ohjaa työn suuntaa. Nimeämättömillä datapisteillä pystytään tutkimaan kuinka samankaltaisia tai erilaisia piikiekot ovat toisiinsa nähdyn ja minkälaiset datapisteiden piirteet vaikuttavat eniten siihen, kuinka datapisteet ryhmittyvät. Tämän pohjalta pystytään arvioimaan ryhmittyvien datapisteiden nimiä, toisin sanoen etsimään vikaryhmiä.

2.2 Muunnos bittikartoiksi

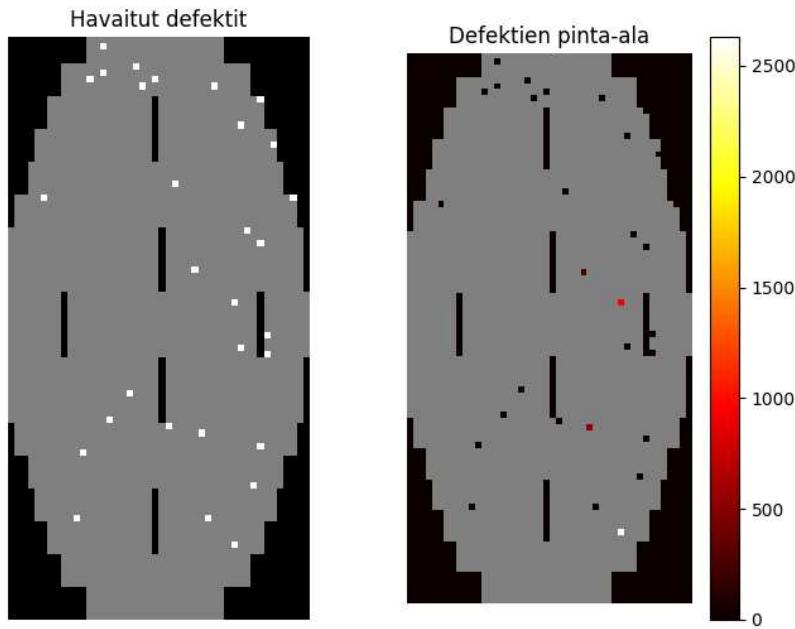
Ennen analysointia, aineisto esikäsitetään muuntamalla datapisteet bittikartoiksi. Muunnettua aineistoa voidaan myöhemmin käyttää koneoppimismallien soveltamiseen. Aineiston esikäsittelyyn tarkoitettun Python-implementaation on kehitänyt [Vaulanen](#). Esittämällä alkuperäinen data bittikarttoina jokaista datapistettä kuvataan sen kaikkien kategoroiden molemmilla ilmaisutavoilla. Kuvaus kategorioilla toteutuu jo teollisessa dataassa, mutta dataa muokataan niin, että se voidaan esittää eri tavalla.

Datapisteitä kuvaavat kategoriat muunnetaan bittikartoiksi eli kuviksi. Jokaista datapisteen kategoriaa ja sen esitystapaa k_a varten luodaan siis oma kuva, toisin sanoen kanava. Täten yksittäistä datapistettä karakterisoiti $K \cdot a = 28$ kanavaa.

Ennen implementaatioon perehtymistä esitellään esimerkki bittikartoista. Kuvassa 1 nähdään bittikartoaksi muunnettua dataa erään piikiekon vikaluoosta 1. Kuvista selviää piikiekon ellipsiä muistuttava muoto. Anturit eivät todellisuudessa ole neliön muotoisia, mutta kuitenkin suorakulmaisia. Bittikartassa jokainen anturi on pikseli, eli neliön muotoinen, joten piikiekon todellinen pyöreä muoto näyttää bittikartoissa soikealta. Vasemmalla on kategorian 'vikaluokka 1' ensimmäinen kanava. Siinä valkoiset pistet esittävät havaittujen defektien sijainteja piikiekolla. Kuvassa pystysuorat mustat kaistaleet ovat alueita piikiekolla, joissa ei ole antureita. Nämä alueet ovat samat jokaisella kiekolla. Oikealla nähdään toinen piikiekon vikaluoikkaa 1 kuvaava kanava. Tutkimalla kuvaa voidaan todeta antureilta mitattujen defektien pinta-alojen olevan pääosin pieniä, mutta muutama alueeltaan suurempi defekti on havaittavissa.

Implementaation ensimmäisessä vaiheessa kaikki data luetaan datakehikoksi käyttäen Pandas-pakettia. Tämän jälkeen alustetaan ja täytetään 4. asteen tensori, käyttäen Numpy-pakettia. Matematiikassa tensori on termi, joka yleistää skalaarin, vektorin, matriisiin ja korkeampien asteiden lineaarikuvaukset. Neljänneksen asteen tensorilla tarkoitetaan multidimensionaalista taulukkoa, jossa sen jokainen alkio määrityy moni-indeksin perusteella ([Bocci ja Chiantini, 2019](#), s. 83–84).

Eräään piikiekon antureiden havainnot, vikaluokka 1



Kuva 1: Satunnaisesti valitun piikiekon vikaluokkaa 1 kuvaavat kanavat.

Tässä työssä datakehikkoa kuvataan tensorilla $T(i, k, x, y)$. Siinä i viittaa datapisteeseen, k anturin kanavaan ja x sekä y datapisteiden eli piikiekkojen pinnalla olevien antureiden koordinaatteihin. Tensori T tuottaa vektorin, kun tensorin kaikki paitsi yksi indeksi on kiinnitetty. Kun kaksi indeksiä on kiinnitetty ja kaksi ovat vapaita, tensori T tuottaa matriisin. Tietyillä indeksivalinnoilla saadaan tensori T tuottamaan matriiseja, jotka voidaan esittää Matplotlib-paketin avulla bittikarttoina: kiinnittämällä i_a ja k_b tensori $T(i_a, k_b, x, y) \in \mathbb{R}^{X \times Y}$ tuottaa datapisteen i_a kanavan k_b piirteiden arvoja kuvaavan matriisin

$$T(i_a, k_b, x, y) = \begin{bmatrix} T(i_a, k_b, 1, 1) & T(i_a, k_b, 2, 1) & \dots & T(i_a, k_b, x, 1) \\ T(i_a, k_b, 1, 2) & T(i_a, k_b, 2, 2) & \dots & T(i_a, k_b, x, 2) \\ \vdots & \vdots & \ddots & \vdots \\ T(i_a, k_b, 1, y) & T(i_a, k_b, 2, y) & \dots & T(i_a, k_b, x, y) \end{bmatrix}.$$

Datapisteet, eli piikiekot muistuttavat muodoltaan ellipsiä, eli niillä ei kuitenkaan ole jokaisessa rivissä x määräää antureita. Tämän takia tensoria alustettaessa kaikkien alkioiden arvoksi asetetaan -1. Tätä arvoa ei löydy datasta. Täytääessä 4. asteen tensoria datalla vain ne koordinaattialkiot, joissa on antureita saavat päivitettyt arvot. Bittikartassa tulee siis olemaan mukana myös alkioita, eli pikseleitä, jotka eivät kuulu piikiekkoon. Nämä pikselit ovat piikiekkoa ympäröivää aluetta, josta emme ole kiinnostuneita. Kyseiset pikselit ovat jokaisella datapisteellä samoissa alkioissa. Koska alkioiden arvoiksi oli alustettu -1, täytääessä piirteitä koordinaattien ja kategorioiden perusteella bittikartalle ne koordinaatit, missä piikiekkö ei ole tai sillä

ei ole antureita erottuvat selvästi. Piikiekkojen sisällä on myös muutamia kaistaleita, joissa ei ole antureita. Myös näiden kaistaleiden alkioiden arvoiksi jää -1, sillä niiden paikalle ei ole täytettävää anturidataa.

Implementaation lopputuloksena on Numpy-array eli 4. asteen tensori, jonka koko on (3948, 109, 74, 28). Tämä tensori on muunnetun datan piirreavaruus. Numpyarrayn koko on ilmoitettu Pythonin indeksoinilla, joka alkaa nollasta kun matematiisesti esitetyn tensorin indeksointi alkaa yleensä yhdestä.

3 Ulottuvuuden alentaminen

3.1 Piirteiden valinta ja muunto

Ulottuvuuden eli dimension alentamisella pyritään muuntamaan korkeaulotteinen data matalampaan ulottuvuuteen. Piikiekodatan käsittely ja vertailu helpottuu, jos kiekkojen pääpiirteet saadaan kätevästi tallennettua matalammassa ulottuvuudessa. Dimension alentamisessa lasketaan piirrevektoreiden ulottuvuutta valinta- ja muuntomenetelmillä. Piirrevektori on yleisesti käytetty tapa esittää datapiste numeerisessa muodossa koneoppimis- ja analyysitehtävissä. Annettuna datapiste i_a , voimme esittää sen kaikki piirteet piirteina \mathbf{x}_{i_a} yhtälön

$$\begin{aligned}\mathbf{x}_{i_a} &= (T(i_a, k_1, x_1, y_1), T(i_a, k_1, x_1, y_2), \dots, T(i_a, k_K, x_X, y_Y)) \\ &= (x_{i_a,1}, \dots, x_{i_a,K \cdot X \cdot Y}) \in \mathbb{R}^{K \cdot X \cdot Y}\end{aligned}$$

muodossa. Tässä K, X, Y ovat tensorin T ulottuvuuksien maksimi-indeksit. Datan ulottuvuudella tarkoitetaan datapisteen piirrevektorin ulottuvuutta. Datan ulottuvuus on sama datajoukon kaikille datapisteille. Ulottuvuuden alentamisella tarkoittamme metodeita, jotka muuntavat datapistettä kuvaavan piirrevektorin muotoon

$$\mathbf{x}'_{i_a} = (x'_{i_a,1}, \dots, x'_{i_a,n}) \in \mathbb{R}^n$$

niin, että $n < K \cdot X \cdot Y$.

Mikäli datan korkea ulottuvuus aiheuttaa ongelmia, olisi hyödyllistä yrittää alentaa sitä datan tärkeää informaatio säilyttäen. Silloin ulottuvuuden alentamista käytetään esikäsittelytyökaluna ennen data-analyysimallien soveltamista dataan alemmassa ulottuvuudessa [Verleysen ja François \(2005\)](#). Kuten tässä työssä, data-analyysimallit ovat usein koneoppimismalleja. Yleisesti koneoppimismenetelmän kouluttamiseen käytettävien datapisteiden määrä tulisi olla huomattavasti suurempi kuin piirteiden määrä datapistettä kohden [Jung \(2022\)](#), tosin tämä tavoite jää usein toteutumatta datan puutteen takia.

Datan analysoinnissa ja käsittelyssä dimensioiden alentamisesta on monia hyötyjä. Redundanttien piirteiden datasta pudottamisen myötä vaadittava laskentateho analyssissä laskee ja analysointitehtävien algoritmit nopeutuvat [Ladla ja Deepa \(2011\)](#). Datan ymmärrettävyys ja analyysin tarkkuus paranee, kun dimension alentamisen jälkeen aineistoa käsittellään laadukkaammalla datalla eli selkeämmin ja tarkemmin kuvaavilla piirteillä. Ulottuvuuden alentaminen jaetaan yleisellä tasolla kahteen eri menetelmään: piirteiden valintaan ja piirteiden muuntamiseen. Piirteiden muuntaoa kutsutaan usein nimenomaan ulottuvuuksien alentamiseksi [S. Khalid ja Nasreen \(2014\)](#), mutta tässä työssä käytetään termiä piirteiden muunto. Parhaan ulottuvuuden alentamismetodin löytäminen on todella vaikeaa. Samalle aineistolle voidaan haluta käyttää täysin erilaisia ulottuvuuden alentamismetodeita, jos aineistolle myöhemmin tehtävän analyysin tavoitteet muuttuvat edes vähän.

Ensimmäisenä esiteltävä metodi on piirteiden valinta, toisin sanoen piirteiden suodattaminen. Se on pohjimmiltaan heuristinen hakuprosessi, jossa arvioidaan erilaisten piirreosajoukkojen hyväyyttä [Liu ja Yu \(2005\)](#). Siinä valitaan jollakin metodilla

osajoukko kaikista piirteistä ja hylätään loput. Piirteiden valinnan etu on, että yksittäisen piirteen tietoa ei hukku. Ongelmaksi voi muodostua, että valittaessa vain pieni osajoukko kaikista piirteistä alkuperäisen joukon ollessa suuri ja monipuolinen, yleistä tietoa datapisteistä voi hukkua. Piirteiden suodatuukseen on lukuisia metodeita. Suodatusmenetelmät valitsevat piirteitä niille ominaisen suorituskykymittarin perusteella, riippumatta käytettävästä koneoppimismenetelmästä. Vasta suodatusmenetelmän luokittelua parhaan piirteiden osajoukon koneoppimismenetelmät käyttäävät niitä. Suodatusmenetelmät voivat asettaa yksittäiset piirteet järjestykseen tai arvioida kokonaisia piirrejoukkoja. Suodatusmenetelmien suorituskykymittarit voidaan karkeasti luokitella informaatio-, etäisyys-, johdonmukaisuus-, samankaltaisuus- ja tilastollisiin mittareihin. [Jović et al. \(2015\)](#) Halutessaan lukija voi perehdyä suodustekniikoihin ja niiden suorituskykymittareihin edellämainitussa julkaisussa, jossa käsitellään tarkemmin niiden soveltuvuutta erilaisiin koneoppimismenetelmiin.

Seuraavaksi esitellään dimension alentaminen piirteitä muuntamalla. Sen tavoitteena on manipuloida dataa niin, että vain analyysille relevantti tieto jää piirteisiin. Toteutus tehdään matemaattisilla malleilla, jotka voivat olla lineaarisia, sekä epälineaarisia ([Laine, 2003](#), s. 42). Yksinkertainen esimerkki piirteiden muunnosta on summata piirrevektorissa jokaisen kahden vierekkäisen indeksin arvot yhteen. Esimerkki on lineaarinen muunto. Useimmissa sovelluksissa tämä ei ole hyvä tapa tuoda tärkeää tietoa piirteistä esiin. Esimerkistä voidaan havaita, että muunnetut piirteet eivät usein kerro mitään alkuperäisestä datasta.

Piirteitä muuntamalla pyritään esittämään data sellaisessa muodossa, jossa data-analyysimallit erottavat dataa karakterisoivat ominaisuudet paremmin. Kun data-analyysimallit oppivat kuvaamaan datan piirteitä tarkemmin, tutkittavan tehtävän näkyvyyttä eli dataan tulkinnan kyky paranee. Piirteiden muunnossa piirrevektorin kokoa voidaan laskea hukkaamatta paljon tietoa alkuperäisestä piirreavaruudesta. Valinta muuntamiseen käytettyjen metodien välillä riippuu datan tyypistä ja sovelluksen kohteesta [Liu ja Yu \(2005\)](#).

Käytettäessä lineaarista piirteiden muuntometodia, olisi syytä tutkia sen sopivuutta datan käsittelyyn jopa silloin kun datan luokat ovat lineaarisesti eroteltavissa, sillä [Martinez ja Zhu \(2005\)](#) mukaan tulokset voivat silti olla virheellisiä. Muuntomenetelmän sopivuutta voidaan siis arvioida suoraan dataa katsomalla, mutta mikäli mahdollista, muuntojen hyvyyttä kuvaavia suorituskykymittareita olisi hyödyllistä tutkia metoden toimivuuden varmistamiseksi. Lineaaristen piirteiden muuntometodien käytettävyyden arvointiin voi perehdyä lukemalla lisää edellä mainitusta lähteestä.

Lineaarinen piirteiden muunto on käsitteliesti yksinkertaista, ja sitä on käytetty monilla sovellusalueilla. Sillä on kuitenkin rajoituksensa dataan, joka ei ole lineaarisesti eroteltavissa, sillä epälineaarisia riippuvuuksia on vaikea kuvata lineaarisella muunnoksella [Park et al. \(2004\)](#). Kun tärkeää tietoa datasta hukkuu lineaarisella menetelmällä, ongelman ratkaisemiseksi voidaan sen sijaan käyttää epälineaarisia piirteiden muuntamismenetelmiä. Tämä ei kuitenkaan takaa parempia tuloksia, ja niitä onkin syytä vertailla.

Tässä työssä tapahtuu juuri kuvailtu tilanne. Datassa tiedetään olevan epälineaarisia riippuvuuksia [Vaulanen](#), joten tulemme vertailemaan lineaarista ja epälineaarista

metodia piirteiden muuntoon sekä niiden vaikutusta muunnon jälkeisen analyysin hyvyyteen. Tässä työssä käytettäväksi lineaariseksi menetelmäksi valikoitui pääkomponenttianalyysi ja epälineaariseksi konvoluutioneuroverkosta eristetty enkooderi. Menetelmät valittiin Muratan toiveiden perusteella, ja niihin perehdystään osioissa 3.3 ja 3.4. Menetelmien vaikutusta analyysin tuloksiin tarkastellaan osiossa 5, jossa valitaan lopullinen käytettävä menetelmä.

Joissakin sovelluksissa halutaan käyttää piirteiden valintaa, sekä muuntoa pääallekkäin ulottuvuuden alentamiseen. Koska muunnetut piirteet eivät välittämättä kuvaa alkuperäistä dataa mitenkään, tulisi ensin valita dataa parhaiten karakterisoivat ja analyysin kannalta hyödyllisimmät piirteet, jonka jälkeen muuntaa niitä. Piirteiden muunnon jälkeen voidaan kuitenkin valita käytettäväksi vielä pienempi piirteiden osajuokko, jos jotkin muunnetut piirteet todetaan hyödyttömiksi tai haitallisiksi analyysin kannalta. Kyseisen toimintatavan voi huomata olevan käytössä tämän työn pääkomponenttianalyysin implementaatiossa, johon syvennytään osiossa 3.3.

3.2 Datan skaalaus

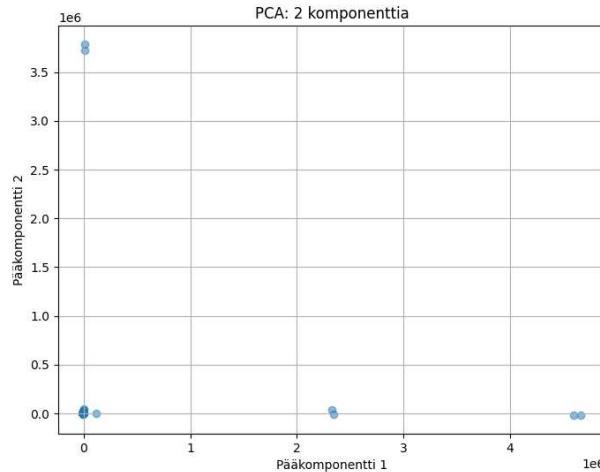
Datan skaalaus on esikäsittelytekniikka, jonka tarkoituksena on mukauttaa datan arvojen vaihteluväli tai jakauma niin, että ne ovat vertailukelpoisia tai parantavat myöhemmin sovellettavan algoritmin suorituskykyä. Skaalaus varmistaa, että kaikki datan piirteet vaikuttavat analyysiin yhtä paljon, eikä mikään piirre dominoi suurempien arvojensa tai eri yksiköidensä vuoksi. Yleisesti datan skaalaaminen ennen piirteiden muuntoa on välittämätöntä, jos datan piirteet ovat eri mittakaavoissa, sillä datan skaalaamattomuus johtaa usein väärin tuloksiin. [Ahsan et al. \(2021\)](#) Kuitenkin väärän skaalausmenetelmän valinta voi olla enemmän haitallinen kuin hyödyllinen analyysille.

Useimmat koneoppimisen algoritmit ja piirteiden muuntomenetelmät tekevät samat operaatiot jokaiselle piirteelle, eli ne olettavat, että kaikki data on samassa mittakaavassa. Esimerkiksi tässä työssä dataa kuvataan kahdella eri yksiköllä: defektien määrällä ja niiden pinta-alaa kuvaavilla piirteillä. Ilman skaalausta pinta-alaa kuvaavat piirteet määräväät muunnetun datan projektiion, koska niiden arvojen vaihteluvälit ovat merkittävästi suurempia kuin defektien määrää kuvaavien piirteiden vaihteluvälit. Piirteiden muuntomenetelmät eivät anna painoarvoa defektien määrää kuvaaville piirteille, koska ne ovat arvoltaan pienempiä kuin pinta-alaa kuvaavat piirteet.

Tässä työssä halutaan käyttää niin sanottua standardointia: datapisteistä vähennetään datan keskiarvo, jonka jälkeen erotus jaetaan datan keskihajonnalla. [Pedregosa et al. \(2011\)](#) Kyseisen skaalaimen käyttö on toivottavaa, koska pääkomponenttianalyysi perustuu kovarianssimatriisin käyttöön, joka on herkkä piirteiden mittakaavoille. Skaalaimen tekemä datan keskiarvon nollaaminen on myös hyödyllistä useimille datan muunnon menetelmille, koska se varmistaa muunnetun datan keskittymisen origoon.

Kuvasta 2 näkyy, kuinka massiivisia eroja datapisteille muodostuu muuntamalla piirteitä pääkomponenttianalyysillä, kun datasta on vähennetty sen keskiarvo, mutta piirteitä ei olla skaalattu samaan kokoluokkaan. Melkein kaikki datapisteet keskittyvät

origoon, ja vain muutamat datapisteet erottuvat, jääden todella kauas muista. Origon lähettyvyydessä on yli 3900 datapistettä, vaikka sitä on mahdotonta havaita kuvasta ilman tarkennusta. Pääkomponenttianalyysistä kerrotaan seuraavassa luvussa.



Kuva 2: Skaalaamatonta dataa voi olla vaikea tulkita: skaalaamaton data esitetynä sen PCA:n kahdella komponentilla.

3.3 Pääkomponenttianalyysi

Pääkomponenttianalyysi (PCA) on laajasti käytetty monimuuttujainen tilastollinen menetelmä. Se on ulottuvuuden alentamisen tekniikka, joka suoritetaan lineaarisesti piirrevektoreita muuntamalla. PCA:ssa alkuperäiset piirrevektorit muunnetaan pääkomponenttien suuntaan. Tämä tehdään datapisteiden ja pääkomponenttien pistetulolla, jolloin jokainen datapiste kuvataan uudessa koordinaatistossa, pääkomponenttiavaruudessa. Tässä avaruudessa datapisteiden väliset samankaltaisuudet ilmenevät etäisyksinä, eli projektio pääkomponenttien suuntaan tiivistää datapisteiden informaation ja mahdollistaa niiden esittämisen alemmassa dimensiossa. [Abdi ja Williams \(2010\)](#) Datapisteet voidaan esittää esimerkiksi kahden pääkomponentin muodostamalla kartalla, joka havainnollistaa niiden ryhmittelyä ja eroja.

Datapisteen i_a piirrevektorin \mathbf{x}_{i_a} projektiokoordinaattia $PC_{i_a,j}$ pääkomponentin v_j suuntaan voidaan kuvata yhtälöllä

$$PC_{i_a,j} = v_{j,1}\mathbf{x}_{i_a,1} + v_{j,2}\mathbf{x}_{i_a,2} + \dots + v_{j,K \cdot X \cdot Y}\mathbf{x}_{i_a,K \cdot X \cdot Y} \in \mathbb{R}$$

sillä ehdolla, että yhtälö

$$\|v_j\| = \sqrt{v_{j,1}^2 + v_{j,2}^2 + \dots + v_{j,K \cdot X \cdot Y}^2} = 1$$

toteutuu, eli että vektori v_i on yksikkövektori.

Pääkomponentit ovat vektoreita, jotka maksimoivat alkuperäisen datan varianssin suunnassa, johon ne osoittavat. Ensimmäinen pääkomponentti osoittaa siihen

suuntaan, jossa datan varianssi on suurin. Jokainen seuraava pääkomponentti maksimoidaan jäljelle jääväästä varianssista, mutta ne ovat ortogonaalisia aiempiin pääkomponentteihin. [Holland \(2008\)](#) Esimerkiksi käyttämällä kahta pääkomponenttia, voidaan alkuperäinen data kuvata 2-ulotteiseen avaruuteen niin, että datan varianssista säilyy mahdollisimman paljon. Toisaalta PCA on lineaarinen menetelmä, eli se olettaa, että datan tärkeimmät piirteet ovat suoraan havaittavissa lineaarisista suhteista. PCA ei siis tunnistaa monimutkaisempia epälineaarisia rakenteita datassa.

Tavallista PCA:ta käytettäessä datakehikon, eli tensorin T dimensioita on muokkattava sopivaksi pääkomponenteille. Tensorin ulottuvuuksia yhdistetään niin, että $T \in \mathbb{R}^{I \times K \times X \times Y} \rightarrow Z \in \mathbb{R}^{I \times P}$, missä $P = K \cdot X \cdot Y$. Käytännössä piirteet dimensioissa K, X, Y listataan peräkkäin uuteen yksittäiseen ulottuvuuteen P , jolloin Z muodostaa datamatriisin, joka sisältää riveinään datan piirrevektorit \mathbf{x}_i . Piirrevektorit voidaan projektoida pääkomponenttien avulla.

Matemaattisesti pääkomponenttien etsintä perustuu positiivisesti semidefinitisten matriisien ominaisarvohajotelmaan [Abdi ja Williams \(2010\)](#). Olkoon $C = \text{Cov}(Z) \in \mathbb{R}^{P \times P}$ datamatriisin Z otoskovarianssimatriisi. Oletetaan, että Z on skaalattu siten, että sen keskiarvovektori

$$\bar{\mathbf{x}} = \frac{1}{I} \sum_{i=1}^I \mathbf{x}_i^T = 0.$$

Koska C on positiivisesti semidefinitiitti, sillä on ominaisarvoesitys

$$C = GDG^\top.$$

Tässä $G \in \mathbb{R}^{P \times P}$ on ortonormaali ja sen pystyvektorit ovat C :n ominaisvektorit ja diagonaalimatriisin $D \in \mathbb{R}^{P \times P}$ diagonaalialkiot ovat niitä vastaavat ominaisarvor siten, että $d_1 \geq d_2 \geq \dots \geq d_P$. Aineiston pääkomponenttimuunnos saadaan asettamalla

$$U = ZG \in \mathbb{R}^{I \times P}.$$

Nyt

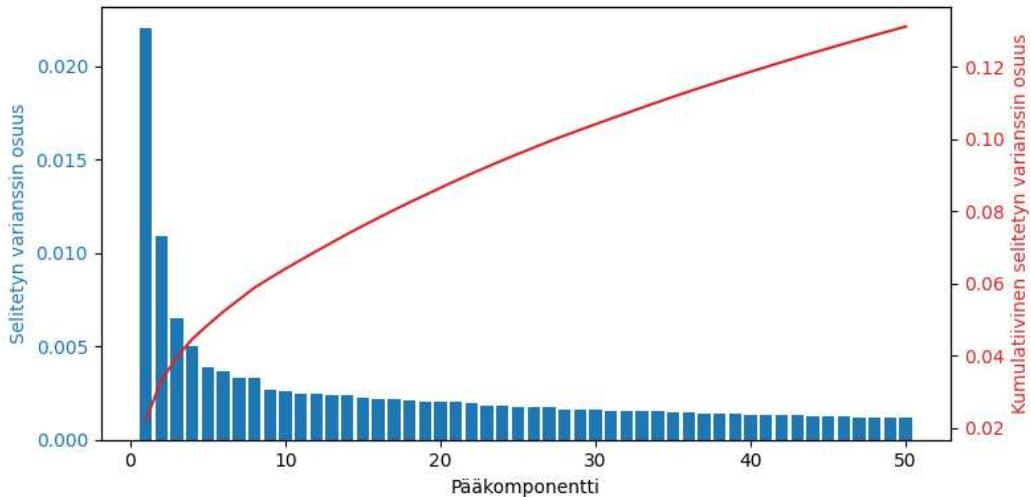
$$\text{Cov}(U) = \text{Cov}(ZG) = G^\top \text{Cov}(Z)G = G^\top GDG^\top G = D.$$

Näin ollen muunnetun aineiston kovarianssimatriisi $\text{Cov}(U)$ on diagonaalimatriisi D , jonka diagonaaliset alkiot ovat suuruusjärjestyksessä. Lisäksi uudet muuttujat ovat ortogonaalisia, toisin sanoen korreloimattomia. Nyt dimensiota voidaan pienentää siten, että käytetään vain muutamaa ensimmäistä pääkomponenttia, jolloin uusi datamatriisi koostuu matriisin U ensimmäisistä sarakkeista.

Tässä työssä PCA:n toteutus tehdään käyttäen Pythonin scikit-learn-kirjastoa. Scikit:in PCA-työkalulla datamatriisin keskiarvo nollataan automaattisesti, mutta piirteitä ei skaalata samaan mittakaavaan [Pedregosa et al. \(2011\)](#). Tämän takia ennen PCA:n soveltamista dataa skaalataan samaan mittakaavaan scikit-learn-kirjaston työkalulla StandardScaler.

Pylväskuviosta 3 selviää ensimmäisen 50 pääkomponentin selittämä varianssi tämän työn datasta. Pylväskuvioita tutkimalla voidaan etsiä sopivaa pääkomponenttien osajoukkoa, jolla data muunnetaan. Pääkomponenttien kumulatiivisen selitetyn

varianssin osuutta kuvaavaa käyrää katsomalla voi huomata, ettei PCA onnistu alentamaan ulottuvuutta varianssia tehokkaasti säilyttäen. Ensimmäiset pääkomponentit ovat kuitenkin selvästi muita parempia varianssin selittämiseen.



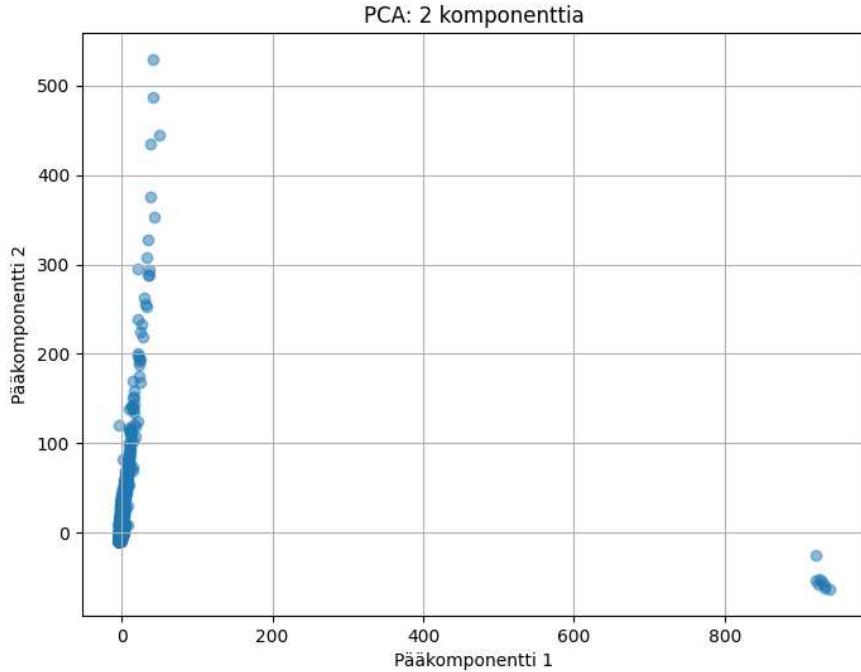
Kuva 3: Yksittäisten pääkomponenttien selittämä varianssi datasta

Tässä työssä käytettävien pääkomponenttien määrää ei päättää selitetyn varianssin perusteella. Tavoitteena on tehdä kuvaan klusteranalyysi, joten aluksi käyttöön valikoitui kaksi pääkomponenttia. Tällä valinnalla datan varianssista pääkomponenttimuunnoksessa säilyy noin 3,5%. Tämä on todella heikko arvo, sillä Holland (2008) mainitsee kriteereissään, että voisi olla hyödyllistä käyttää ainakin niin monta pääkomponenttia, että varianssista selittyisi ainakin 90%. Matalan selitetyn varianssin perusteella on todennäköistä, että jotkin vialliset piikiekot ovat ryhmittyneet lähelle ehjiä, koska PCA ei havaitse niiden eroja pienellä pääkomponenttimääräällä.

Dataalle yritettiin tehdä piirteiden valintaa ennen PCA:n soveltamista tulosten parantamiseksi. Kun PCA:ta sovellettiin datapisteille, joissa piirteinä olivat vain defektien kappaalemääräät tai defektien pinta-alat, selitetty varianssi ensimmäisillä pääkomponenteilla oli vielä pienempää. Tämän perusteella piirteiden valintaa ei tehdä lopullisessa PCA-implementaatiossa.

Kuvassa 4 on standardisoivasti skaalattu data projisoituna kahdella pääkomponentilla. Kuva tutkimalla voidaan huomata, että pääkomponentti 1 selittää eniten varianssista siksi, että muutama oletettavasti viallinen piikiekkö on piirteiltään niin eroavaista muusta datasta. Jos kuvan oikeassa alalaadissa makaavat vialliset piikiekot poistettaisiin datasta, ja PCA tehtäisiin uudestaan, projektio pääkomponentti 1 suuntaan olisi aivan erilainen, ja datapisteiden välillä olisi selkeästi enemmän vaihtelua myös Pääkomponentti 1-akselin suuntaan.

Kuvassa 5 on havainnollitettu dataan projektio kolmelle pääkomponentille. Pääkomponenttien 2 ja 3 huomataan vaikuttavan paljon suurempaan osaan datapisteistä. Pääkomponentti 1 suunnassa isoarvoiset datapisteet vaikuttavat olevan harvinaisen vikatyypin komponenetteja, joiden arvot muiden pääkomponenttien suuntaan ovat



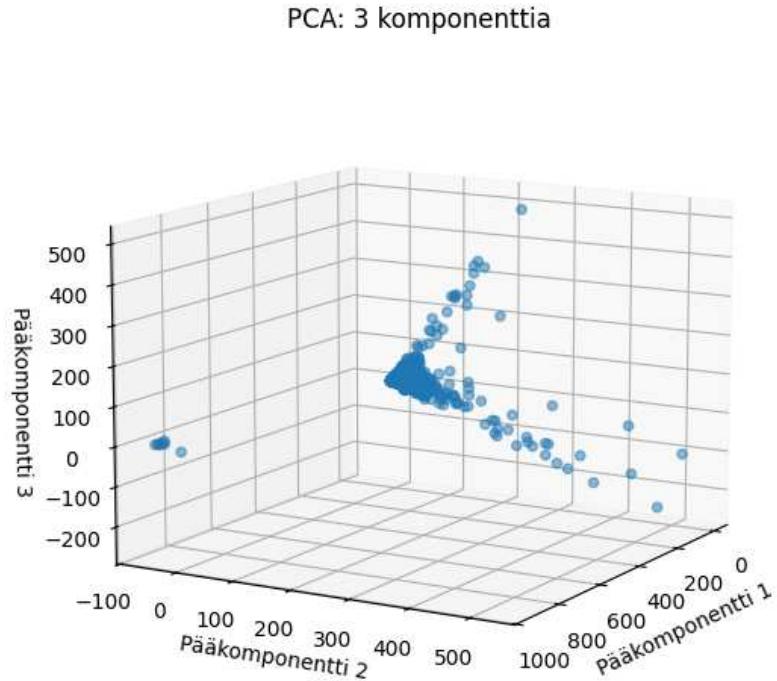
Kuva 4: Skaalattu data esitettyynä sen PCA:n kahdella komponentilla.

kuitenkin lähellä 0, eli eivät poikkea massasta. Jos valittaisiin käytettäväksi pääkomponentit 2 ja 3, nämä selkeät vierashavainnot jäisivät huomaamatta. Toisaalta katsoessa vain pääkomponentteja 1 ja 2, kolmannen pääkomponentin suuntaan on myös merkittäviä eroja datapisteissä, jotka jävät kokonaan huomaamatta. Syvempi analyysi havaintojen rakenteista, PCA:n hyvyydestä ja vikatyypien etsinnästä esitetään osiossa [5](#).

3.4 Konvoluutioneuroverkko

Deterministiset koneoppimismenetelmät, kuten PCA, toimivat tietyllä, etukäteen määritetyllä menetelmällä. Tällaisten menetelmien käyttö epälineaaristen järjestelmien datan käsittelyssä tuottaa usein vääräitä tuloksia, koska ne eivät pysty mukautumaan datan erityispiirteiden mukaan. Jotta dataa voitaisiin käsittää piirteiden monimutkaiset yhteydet säilyttää, olisi hyödyllistä käyttää syväoppimisen menetelmiä. Syväoppiminen on koneoppimismenetelmien alaluokka, jolla tarkoitetaan keinotekoisia neuroverkkoja, jotka mallintavat ja oppivat monimutkaisia rakenteita ja kuvioita datasta. Toisin kuin perinteiset koneoppimismenetelmät, neuroverkot pystyvät automaattisesti oppimaan piirteitä datan pohjalta, ohjatusti tai ohjaamattaa, kun sille annetaan selvät kriteerit, joihin pyrkiä. Ohjatussa oppimisessä neuroverkko koulutetaan luokitettulla datalla, ohjaamattomissa malleissa dataa ei ole luokiteltu. [Shrestha ja Mahmood \(2019\)](#)

Sama kirjoittaja toteaa myös, että konvoluutioneuroverkko (CNN) on syväop-



Kuva 5: Skaalattu data esitettyynä sen PCA:n kolmella komponentilla.

pimisen malli, joka on suunniteltu käsitlemään rakenteellista dataa, kuten kuvia, videoita, sekä muita monimutkaisia aineistoja. CNN:t ovat erityisen tehokkaita tehtävissä, jotka liittyvät piirteiden tunnistamiseen ja kuvioiden havaitsemiseen, näistä esimerkkeinä kasvojentunnistus, houkuttelevan sisällön suosittelun käyttäjälle ja kuvien luokittelu. Neuroverkkojen rakenteet ovat hyvin yksilöllisiä riippuen siitä, mitä ongelmaa ratkaistaan ja minkä tyypistä data on.

Tässä työssä CNN oli toteutettu ohjaamattomasti. Työssä käytetyn tyylistä neuroverkkoa kutsutaan autoenkooderiksi, joka on kuvien uudelleenrakentamiseen pohjautuva metodi. CNN:n tavoitteena on alentaa datakehikon dimensiota ja sen jälkeen uudelleenrakentaa alkuperäinen data mahdollisimman tarkasti alennetussa dimensiossa olevan datan pohjalta. Onnistuakseen CNN pyrkii opettelemaan sopivat parametrit muunnosta varten minimoimalla keskineliövirhettä alkuperäisen ja uudelleenrakennetun datan välillä. Kun sopivat parametrit on opittu, CNN:stä erotetaan dimensiota alentava algoritmi, eli enkooderi. Koulutettu enkooderi on nyt työkalu, joka on suunniteltu kyseisen tyypisen datan dimension alentamiseen. Sen etu on kyky löytää myös datan epälineaariset yhteydet, joita perinteiset menetelmät eivät usein pysty havaitsemaan. [Tschanne et al. \(2018\)](#)

Kokonaisesta CNN-algoritmista on myös hyötyä. Kouluttamalla CNN vain ehjillä

piikiekoilla ja sen jälkeen käyttämällä koko CNN:n rakennetta pystytään arvioimaan, mitkä datapisteistä ovat viallisia tutkimalla keskineliövirheiden suuruuksia uudelleenrakennettaessa dataa. Metodi perustuu siihen, että tällä CNN:n koulutustavalla datapiste, jonka piirteissä ei ole juurikaan poikkeamia on tarkemmin uudelleenrakennettavissa kuin viallinen, suuria poikkeamia piirteissään sisältävä datapiste. [Principi et al. \(2019\)](#)

Neuroverkkojen yhteydessä konvoluutiolla tarkoitetaan operaatiota, jossa liikkuva numeromaski kulkee datakehikon yli, ja tekee maskin kokoisille datakehikon osille operaatiot yksi kerrallaan. Datakehikon osille tehdään alkiokohtainen tulo ja sen jälkeen summaus. Kuvassa 6 on esitetty yksinkertainen esimerkki konvoluutiosta yksikanavaiselle kaksiulotteiselle kuvalle. Alkiokohtaisen tulon tuottaman matriisin alkioiden summa on konvoluution tuottaman matriisin alkio. Pehmusteilla tarkoiteitaan ylimääräisiä nolla-alkioita, jotka reunustavat kuvan alkioita. Yhden kerroksen paksuiset pehmusteet varmistavat tässä tilanteessa sen, että matriisi säilyttää alkuperäisen muotonsa konvoluutiossa. Pehmustekerroksen paksuutta ja niiden alkioiden arvoja voidaan muuttaa tarpeen mukaan. Askel puolestaan määritää sen, kuinka suuria askelia otetaan pistetulojen välissä. Yhden askel tarkoittaa, että kaikille kuvan alkioille tehdään konvoluutio, kahden askeleella joka toiselle, ja niin edelleen. Jos konvoluution lopputuloksen dimensioiden koon haluttaisiin olevan puolet alkuperäisestä, askel nostettaisiin kahteen. CNN:n opetusvaiheessa se etsii optimaalisia arvoja maskin alkioille, eli painoille tappiofunktion minimoimiseksi. Yleisesti maskilla tehtyjen operaatioiden jälkeen CNN summaa vakion koko ulostulokanavaan. Vakiot ovat siis kanavakohtaisia säätöparametreja. [Alzubaidi et al. \(2021\)](#)

Pehmustettu yksikanavainen "kuva"								
0	0	0	0	0	0	0	0	0
0	5	0	8	7	8	1	0	
0	1	9	5	0	7	7	0	
0	6	0	2	4	6	6	0	
0	9	7	6	6	8	4	0	
0	8	3	8	5	1	3	0	
0	7	2	7	0	1	0	0	
0	0	0	0	0	0	0	0	
$1 \times 1 \times (6+2) \times (6+2)$								

Lopputulos								
-19	22	-20	-12	-17	11			
16	-30	-1	23	-7	-14			
-14	24	7	-2	1	-7			
-15	-10	-1	-1	-15	1			
-13	13	-11	-5	13	-7			
-18	9	-18	13	-3	4			
$1 \times 1 \times 6 \times 6$								

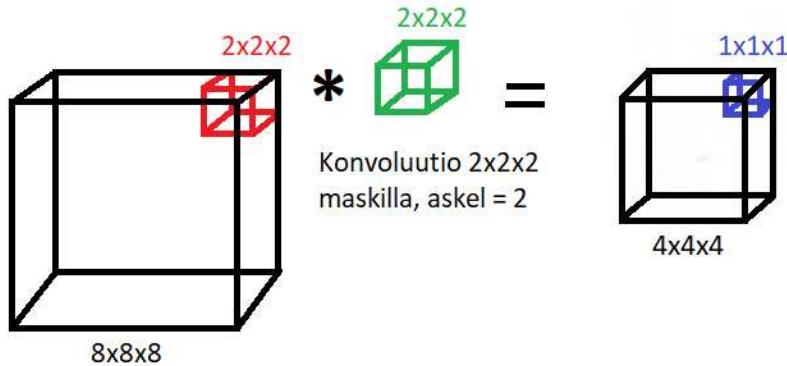
Pistetulo								
9	5	0	x	0	1	0	=	0 5 0
0	2	4		1	-4	1		0 -8 4
7	6	6		0	1	0		0 6 0
Elementtien summa on 7								

Kuva 6: Konvoluutio yksikanavaiselle kaksiulotteiselle kuvalle. Kuvan luomisessa on käytetty [Godoy \(2021\)](#) luomaa pohjaa.

[Alzubaidi et al. \(2021\)](#) toteaa myös, että koneoppimisalgoritmin tehokkuus on yleisesti riippuvainen datan esitystavasta. Jotta työn konvoluutioneuroverkko onnistuisi kuvaamaan piikiekkojen välistä suhteita oikein, työssä käytettyä tensoria T muokataan konvoluutioneuroverkolle 5. asteen tensorikksi. Tensorin T muuttujan

k sisältämät vikatyypit ja niiden esitystavat jaetaan kahteen ulottuvuuteen, jotta konvoluutioneuroverkko pystyy käsittelemään vikojen määriä ja pinta-aloja samanaikaisesti: $T(i, k, x, y) \rightarrow T'(i, k, z, x, y)$, missä z kuvailee defektityyppien ilmaisutapaa. Muuttuja k on nyt vikatyyppi, eli kanava. Muuttuja z on vikatyypin esitystapa, joka on datapiesteiden kanavien syvyys. Viisiulotteisen tensorin T' avulla pystytään käyttämään Pythonin PyTorch-kirjaston Conv3d-operaatioita, jotka ovat kolmiulotteisia konvoluutioita [Paszke et al. \(2019\)](#).

Kuvassa 7 on havainnollistettu konvoluutio kolmiulotteiselle yksikanavaiselle datakuutiolle. Pehmusteita ei ole piirretty kuvaan. Konvoluution askel on 2, joten lopputulokseksi on puolet pienempi datakuutio. Jos datakuutiolla on useampi kanava, kuten tässä työssä, maski on 4-ulotteinen. Tämä johtuu siitä, että maskilla on jokaiselle kanavalle niitä vastaavat alkiot, eli painot. Uniikit painot eri kanaville auttavat säilyttämään kanaville ominaisia piirteitä. Monikanavaisessa konvoluutiossa maski tekee pistetulot kaikille kanaville samanaikaisesti. Kun kanavien määärää halutaan muuttaa, käytetään haluttavien kanavien määärää maskeja. Jokainen maski suorittaa samat laskuoperaatiot datakehikon kanssa omilla painoillaan, jolloin maskit oppivat säilyttämään erilaisia piirteitä.



Kuva 7: Kolmiulotteinen konvoluutio yksikanavaiselle datakuutiolle.

Konvoluutioneuroverkko koostuu useista kerroksista, joissa seuraava kerros opii edeltävän piirteistä. Tässä työssä enkooderin kerrokset sisältävät kolmiulotteisia konvoluutioita, epälineaarisia aktivointifunkcioita ja lineaarisia muunnoksia. Epälineaarinen aktivointifunktio ReLU: $f(x) = \max(0, x)$ on jokaisen muun operaation välissä. ReLU on nopeasuoritteen ja hyvin yksinkertainen. Inspiraatio sen käyttöön kumpuaa [Kunang et al. \(2018\)](#):n sillä saaduista erinomaisista tuloksista autoenkoodilla, joka pyrkii havaitsemaan järjestelmiin tunkeutumisia. ReLU on se osa CNN:ää, joka toteuttaa epälinearisoinnin ja mahdollistaa monimutkaisten rakenneiden oppimisen. ReLU on kaikista käytetyin funktio konvoluutioneuroverkoissa yleisesti [Alzubaidi et al. \(2021\)](#). Kerroksissa kanavien määärää nostetaan samalla kun datan muita dimensioita lasketaan. Enkooderin viimeisessä kerroksessa kanavat ja data alennetaan lineaarisesti kolmeen ulottuvuuteen. Dekooderin kerrokset ovat enkooderin kerrokset käänteisenä, joissa konvoluutio on korvattu konvoluution transpoosilla.

Kerros	Kanavat sisään → ulos	Dimensio ulos (z, x, y)	Maski, Askel
Enkooderi			
Konv1	$14 \rightarrow 32$	$2 \times 109 \times 74$	$3 \times 3 \times 3, 1$
Konv2	$32 \rightarrow 64$	$2 \times 55 \times 37$	$2 \times 3 \times 3, 2$
Norm1	$64 \rightarrow 64$	$2 \times 55 \times 37$	-
Konv3	$64 \rightarrow 128$	$2 \times 55 \times 37$	$3 \times 3 \times 3, 1$
Konv4	$128 \rightarrow 256$	$1 \times 28 \times 19$	$2 \times 3 \times 3, 2$
Norm2	$256 \rightarrow 256$	$1 \times 28 \times 19$	-
Lin1	$256 \rightarrow 1$	$1 \times 256 \times 1$	-
Lin2	$1 \rightarrow 1$	$1 \times 3 \times 1$	-
Dekooderi			
LinK1	$1 \rightarrow 1$	$1 \times 256 \times 1$	-
LinK2	$1 \rightarrow 256$	$1 \times 136192 \times 1$	-
KonvT1	$256 \rightarrow 128$	$2 \times 55 \times 37$	$2 \times 3 \times 3, 2$
KonvT2	$128 \rightarrow 64$	$2 \times 55 \times 37$	$3 \times 3 \times 3, 1$
KonvT3	$64 \rightarrow 32$	$2 \times 109 \times 73$	$2 \times 3 \times 3, 2$
KonvT4	$32 \rightarrow 14$	$2 \times 109 \times 74$	$3 \times 3 \times 3, 1$

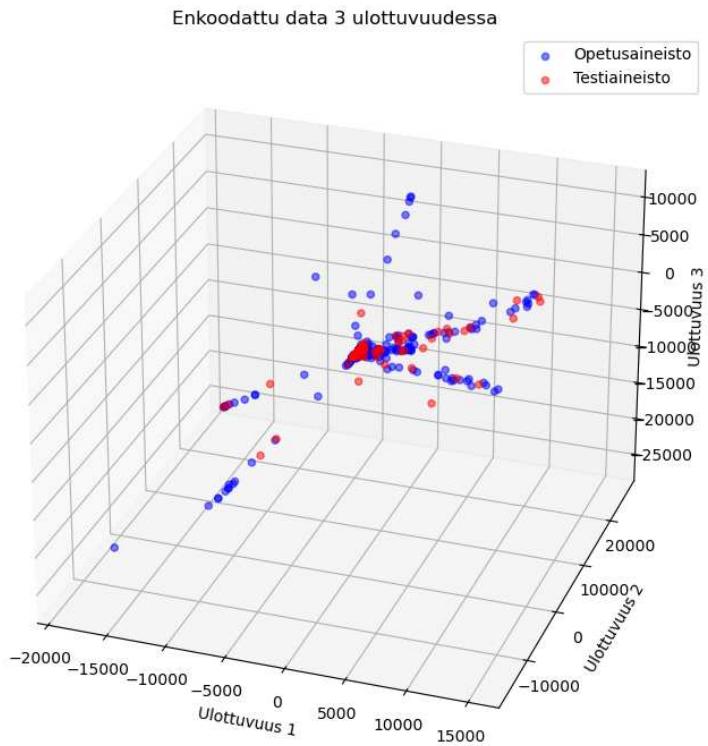
Taulukko 2: Työn konvoluutioneuroverkon rakenne. Kaikkien paitsi Konv- ja Normkerrosten välissä käytettiin ReLU:a.

Taulukossa 2 on esitelty tässä työssä toteutetun konvoluutioneuroverkon kerrokset. Kanavien määrään kasvatetaan samalla kun muut dimensiot pienenevät. Alennetun dimension kooksi valikoitui kolme. Tällä valinnalla vertailu PCA:han on kätevämpi ja datan oletetaan myös ryhmittivän selkeämmin. CNN-mallin maskikoko- ja rakennevalinnat perustuvat [Ruan et al. \(2023\)](#) artikkeliin, jossa on tutkittu kaksilottaisen CNN:n hyvyyttä laakerien vika-analysissä riippuen parametrivalinnoista. Konv-kerrokset ovat kolmiulotteisia konvoluutioita. Norm-kerrokset ovat operaatioita, joissa datan keskiarvo siirretään nollaan ja varianssi yhteen. Norm kerrokset otettiin käyttöön mallin ylisovittamisen vähentämiseksi. Ennen Norm-kerrosten käytöönottoa mallin testiaineiston uudelleenrakennusvirhe oli 79.2% korkeampi kuin opetusaineistolla. Käytöönnoton jälkeen vastaava arvo oli 55.7%, eli parantamisen varaa löytyy edelleen. Norm-kerroksia ei tarvita dekooderin puolella, koska dekooderi pystyy omilla painoillaan rakentamaan normalisoidunkin datan uudestaan. Näiden kerrosten käyttö ja sijoittelu pohjautuu [Garbin et al. \(2020\)](#) artikkeliin, jossa empiirisesti tutkitaan ylisovittamisen vähentämisen mahdollisuksia pudotus- ja normalisointiteknikoilla. Kaikkien paitsi Konv- ja Norm-kerrosten välissä käytettiin epälineaarista aktivointifunktio ReLU:a. Vasta Konv4-keroksessa laskettiin datakehikon syvyyttä z , joka kuvaa vikojen ilmaisutapoja. Tällä valinnalla yritetään opettaa konvoluutioneuroverkko huomioimaan molemmat vikojen ilmaisutavat ja löytämään niiden välisiä yhteyksiä. Muuttamalla askelta, pehmustetta ja maskin kokoa pystytettiin säätelymään ulostulodimensioita. Konv4-keroksessa pehmusteet otettiin pois käytöstä syvyysakselilla, muissa se oli 1. Kyseisessä keroksessa $2 \times 3 \times 3$ maskin käyttö kahden askelkoulla takaa syvyyden putoamisen yhteen. Kaikissa muissa Konv- ja KonvT-kerroksissa, paitsi tietysti KonvT4-keroksessa, pehmustus oli jokaisella ulottuvuudella aina 1. KonvT-kerrokset ovat kolmiulotteisen konvoluution transpooseja, jotka toteuttavat käänteiset muunnokset neuroverkossa. KonvT-kerrokset yrittävät uudelleenrakentaa dataa mahdollisimman tarkasti alennetusta dimensiosta. Lin-kerrokset ovat lineaarisia muunnoksia, ja LinK-kerrokset ovat niiden käänteismuunnokset. Myös lineaarimuunnoksilla on omat painot ja vakiot. Konv4- ja Lin1-kerrosten välissä datan dimensioit litistettiin yhteen. Dekooderissa tehtiin vastaavasti dimensioita korottava käänteismuunnos. Kun enkooderille syötetään datapiste, se palauttaa kolmen pituisen vektorin, joka voidaan havainnollistaa kolmessa ulottuvuudessa. Enkooderin tuottaman vektorin pituutta on helppo muuttaa. Dataa voidaan tutkia kuvantamalla se myös 1 tai 2 ulottuvuudessa vaihtamalla Lin2-keroksen ulostulodimensiota. Vaihtoehtoisesti aineisto voidaan esittää pidemmilläkin vektoreilla, mutta silloin niiden visualisointi koordinaatistossa ei ole mahdollista.

Konvoluutioneuroverkkoa yritettiin luoda myös yhdistäjä-kerroksia (pooling) käytäen. Yhdistäjä-kerrokset ovat dimension alentamiseen käytettäviä kerroksia, missä valitaan erilaisilla algoritmeilla osia dataasta tai yhdistellään niitä. Näiden kerrosten käyttö oli laskennallisesti raskasta. Enkooderin puolella kahdella yhdistäjäkerroksella ja dekooderin puolella kahdella CNN:n kouluttamisaika kertaantui noin viidellä ja samalla uudelleenrakennusvirhe kasvoi. Työssä yritettiin käyttää keskiarvoyhdistämistä (average pooling) ja maksimiyhdistämistä (max pooling) huonolla menestyksellä. Yhdistämismenetelmiin voi perehtyä lisää lukemalla [Gholamalinezhad ja Khosravi \(2020\)](#) artikkelin.

Aineistoa yritettiin skaalata standardoivasti ennen konvoluutioneuroverkon käytöä, mutta se johti tulokseen, joka muistuttaa hyvin paljon kuvaa 4, joka saatiin kahden pääkomponentin muunnuksella. Tämän takia lopullinen CNN-malli käyttää skaalaamatonta dataa. Konvoluutioneuroverkon koulutus tehtiin satunnaisesti sekoitetulla datan osajoukolla. Koulutukseen käytettävän datan osuuus oli 80%, loput datasta säädettiin CNN:n testivaiheeseen. Koulutusvaiheessa CNN:lle syötetään dataa pienissä osissa, tässä työssä kymmenen datapisteen minierissä. Kun CNN on prosessoinut minierän, se päivittää maskien painojen ja vakioiden arvot käyttäen Adam-optimointialgoritmia (Adaptive moment estimation), joka tekee optimoinnin gradienttilaskualgoritmilla minimoidakseen keskineliövirheen alkuperäisen ja prosessoidun minierän välillä. Empiiriset tulokset näyttävät, että Adam toimii paremmin kuin muut stokastiset optimointimetodit. [Diederik ja Kingma \(2015\)](#) Halutessaan lukija voi perehtyä Adamiin ja konvoluutioneuroverkkojen gradienttilaskuoptimointiin syvemmin lukemalla [Dogo et al. \(2018\)](#) mainion artikkelin aiheesta. Koko testidata käytettiin läpi yhteensä 50 kertaa ja datapisteet sekoitettiin kierrosten välissä. Oppimisaste, joka asetettiin arvoon 0.00005, määrittää kuinka suuria päivityksiä optimointialgoritmi voi tehdä painoihin yhdellä muutoksella. Liian suuri oppimisaste johtaa epävakaaseen oppimiseen, koska optimointialgoritmi päivittää painoja liikaa, jolloin se ei onnistu laskemaan keskineliövirheen arvoa prosessin edetessä. Tämä ongelma kohdattiin myös tässä työssä. Oppimisaste oli aluksi asetettu arvoon 0.001, jolloin huomattiin keskineliövirheen kasvavan ja pienenevän jo ensimmäisten 10 kierroksen välillä. Laskemalla oppimisastetta päästiin yli 5 kertaa pienempään koulutusdatan keskineliövirheeseen. Toteutetussa CNN:ssä on hienosäädön varaa, mutta täydellisytyen tähdättäessä jäädään helposti jumiin neuroverkon rakentamiseen.

Kuvassa 8 on havainnollistettu konvoluutioneuroverkosta eristetyn enkooderin muuntamat datapisteet kolmiulotteisessa avaruudessa. Datapisteiden muodostamat 'viivat' voivat esittää piikiekkojen yleisimpiä vikoja. Datapiste kaukana muiden pisteen ryppäästä on tällöin enemmän viallinen kuin lähempänä oleva. Kuvan perusteella aineisto ei ole niin suuri, että analyysi voitaisiin tehdä vain testiaineistolle. Datan puutteen takia enkooderin tulosten analysoinnissa tullaan käyttämään opetus-, sekä testiaineistoa.



Kuva 8: Työn konvoluutioneuroverkosta eristetyn enkooderin muuntamat datapisteet kolmiulotteisessa avaruudessa.

4 Klusterointi

Klusteroinnin päätavoitteena on auttaa ymmärtämään ja ryhmittelemään dataa löytämällä siitä samankaltaisuksia ja piileviä rakenteita, jotka voivat tukea päätöksentekoa ja analyysiä. Se on koneoppimisen ja datan analysoinnin tekniikka, jossa data jaetaan ryhmiin (klustereihin) siten, että saman klusterin jäsenet ovat keskenään samanlaisia, mutta eri klustereihin kuuluvat jäsenet eroavat toisistaan. Useimmat klusterointimenetelmät perustuvat pisteiden välisten etäisyysien eroihin, eli klusterointi on ohjaamaton koneoppimismenetelmä. Klusterointialgoritmi ryhmittelee datapisteet siten, että saman klusterin pisteet ovat mahdollisimman lähellä toisiaan ja eri klustereiden pisteet mahdollisimman kaukana toisistaan. [Fung \(2001\)](#) Jotta klusterointimenetelmän tuloksia voitaisiin analysoida visualoimalla, sillä käsiteltävän datan ollisi oltava maksimissaan kolmiulotteista.

Tässä työssä käytettäväksi menetelmäksi valikoitui k:n-keskiarvon-klusterointi. Kyseinen menetelmä on hyvä vaihtoehto, kun arvio haluttavien klusterien määrästä on jo tiedossa, sillä k:n-keskiarvon-menetelmässä niiden lukumäärä määritetään ennen menetelmän sovellusta. Kuvan 8 perusteella enkooderilla muunnettulla datalla tulisi olla ainakin 5 klusteria. Kuvassa näkyvät lineaariset viivat ovat selvästi omia ryhmittymiä ja datan suurin ryhmittymä on myös selkeästi muista poikkeava. PCA:n tuottamat datapisteet kolmessa ulottuvuudessa, jotka näkyvät kuvassa 5 halutaan jakaa vähintään 4 klusteriin: selvästi poikkeavat havainnot, suurin ryhmittymä ja

dimensioiden 1 sekä 3 suunnassa suuriarvoiset datapisteet. Menetelmän valintaan vaikutti myös se, että dimension alentamisen seurauksena datapisteiden erot selittyyvät niiden euklidisilla etäisyyskäytävillä toisistaan, johon myös k:n-keskiarvon-menetelmä perustuu. K:n-keskiarvon-klusteroinnin heikkous on siinä, että se ryhmittää jokaisen datapisteen johonkin klusteriin. [Sinaga ja Yang \(2020\)](#) Tämä ei ole toivottua, sillä kuvaan 8 perusteella enkooderilla dimensiota alennetussa datassa on selvästi yksittäisiä, kaikista muista datapisteistä merkittävästi poikkeavia havaintoja, tarkoittaen etteivät kyseiset havainnot välittämättä kuulu mihinkään ryhmään. Tämän työn kannalta klusterointialgoritmi toimii optimaalisesti silloin, kun se pystyy alennetussa dimensiossa ryhmittelemään piikiekkojen todelliset vikaluokat ohjaamattomasti. Toistaalta klusteroinnin onnistuminen riippuu myös dimension alennuksen hyvyydestä, eikä klustereiden oikeellisuutta voida varmistaa ilman luokkia, mutta tulosta voidaan silti analysoida.

K:n-keskiarvon-klusterointi perustuu yhtälöön

$$J = \sum_{n=1}^k \sum_{x \in C_n} \|x - \mu_n\|^2$$

minimointiin. Funktio J on klusterien sisäinen hajonta, C_n on joukko datapisteitä, jotka kuuluvat klusteriin n , μ_n on klusterin n keskipiste ja $\|x - \mu_n\|^2$ on euklidinen etäisyys datapisteen x ja klusterikeskipisteen μ_n välillä. Funktion J minimin etsintä aloitetaan valitsemalla klusterikeskipisteet μ_n satunnaisesti. Satunnaisesti valitut klusterien aloituskeskipisteet saattavat vaikuttaa klusteroinnin lopputulokseen. Datapisteet x_i liitetään siihen klusteriin

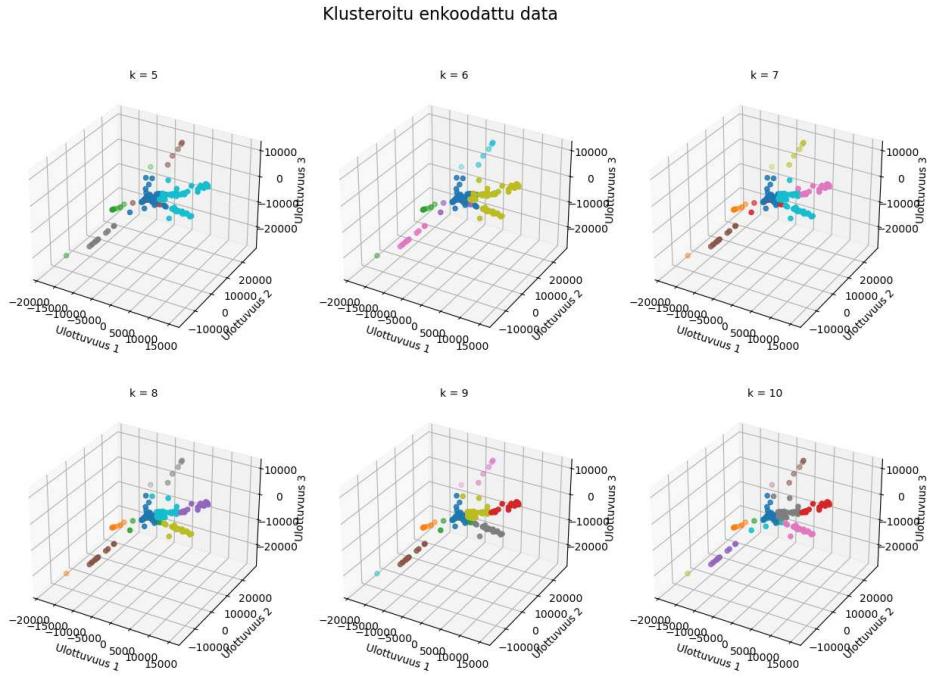
$$C_n = \{x_i : \|x_i - \mu_n\|^2 \leq \|x_i - \mu_\ell\|^2, \forall \ell \in \{1, \dots, k\}\} \quad (1)$$

jonka keskipistettä μ_n ne ovat lähimpänä. Tämän jälkeen jokaisen klusterin keskipiste

$$\mu_n = \frac{1}{|C_n|} \sum_{x \in C_n} x \quad (2)$$

päivitetään laskemalla sen datapisteiden keskiarvo. Muuttuja $|C_n|$ on klusterin C_n pisteiden lukumäärä. Jotta klusterointimenetelmä löytäisi funktion J minimin, vaiheita 1 ja 2 toistetaan, kunnes keskipisteet eivät enää muudu. [Fung \(2001\)](#) K:n-keskiarvon-klusterointi toteutettiin Pythonin scikit-learn paketilla [Pedregosa et al. \(2011\)](#).

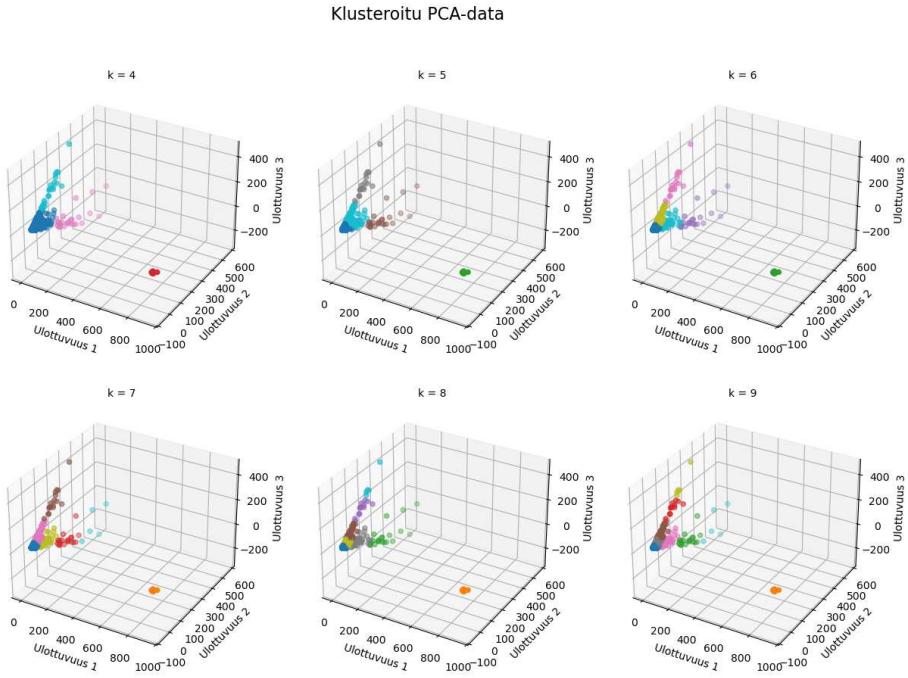
Kuvassa 9 on havainnollistettu enkooderin läpi syötetty piikiekkoista k:n-keskiarvon-menetelmällä klusteroituna. Arvolla $k = 5, 6, \dots, 10$ tarkoitetaan sitä, kuinka moneen klusteriin data on ryhmitetty. K:n-keskiarvon-klusterointi onnistuu monella eri k-avulla löytämään selkeitä pääryhmiä. Valinnoilla $k = 5, 6, 7$ data jaetaan suuriin ryhmiin. Tutkimalla kuvaan voidaan huomata, että esimerkiksi valinnalla $k = 7$ vaaleansinisellä värijätty klusteri voisi sisältää kahta eri vikaryhmää. Valinnoilla $k = 8, 9, 10$ data alkaa jakautumaan pienempiin ryhmiin, ja valinnoilla $k = 9, 10$ luodaan kokonaan oma klusteri yksittäiselle poikkeavalle havainnolle, joka näkyy kuvien vasemmassa alareunassa. Pelkästään klustereita tutkimalla ei pystytä tarkemmin



Kuva 9: K:n-keskiarvon-klusterointu enkoodattu data havainnollistettuna monella eri klusterimäärällä. Muuttuja k viittaa klusterien määärään.

määrittämään sopivaa k -arvoa, vaan olisi syytä vertailla pisteiden ryhmittymiä ja prosessoimatonta dataa.

Kuvassa 10 on havainnollistettu PCA:lla käsitelty piikiekodatata k:n-keskiarvon-menetelmällä klusteroituna. PCA:lla dimensiota alennetuilla datapisteillä on vähemmän muutoksia kaikkien kolmen akselin suuntaan, joten klusteroinnin tulos on helpompi hahmottaa. Kaikki $k = 4, 5, \dots, 9$ valinnat näyttävät onnistuvan jakamaan datan pääpiirteitäin, suuremmilla k -arvoilla tarkemmin. Erityisen kiinnostavaa on, kuinka enkooderin ja PCA:n klusterit eroavat toisistaan esimerkiksi arvoilla $k = 5, 7, 9$. Sisältävätkö klusterit samat datapisteet vai ovatko dimensionalennustekniikat tuottaneet täysin erilaisia tuloksia? Analyysi on toteutettu seuraavassa osiossa.



Kuva 10: K:n-keskiarvon-klusterointi PCA:lla käsitelty data havainnollistettuna monella eri klusterimäärällä. Muuttuja k viittaa klusterien määrään.

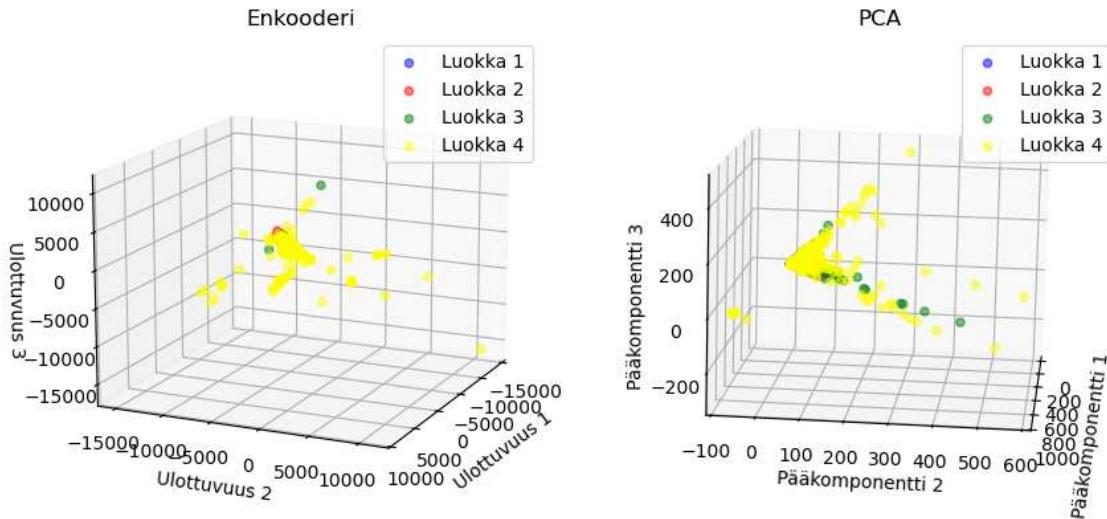
5 Tulokset

5.1 Lopullinen malli ulottuvuuden alentamiseen

Työn tärkein tavoite oli selvittää, pystytääkö PCA:n tai CNN:n avulla esittämään piikiekkoaineisto alemmassa ulottuvuudessa niin, että aineisto säilyttää pääpiirteensä ja olisi tällöin kätevästi vertailtavissa. Kuvassa 11 on havainnollistettu PCA:lla ja CNN:llä käsitelty luokiteltu aineisto. Luokittelut ovat tehty ainoastaan defektien summien pohjalta. Defektien pinta-alojen summat ovat luotettavampi mittari kuin määrät, koska aineistossa on paljon marginaalisen pieniä defektejä. Datapisteiden defektien pinta-alojen summat on ensin summattu datapistekohtaisesti yhteen, minkä jälkeen ne on luokiteltu neljään luokkaan tasaisin prosenttipistein (kvartiilein) käyttäen summien jakaumaa. Luokkien summarajat ovat 5051, 12628, 19647, 36525, 25523119. Ensimmäinen arvo on luokan 1 alaraja ja niin edelleen. Tämän jälkeen visualisoitiin enkooderilla ja PCA:lla muunnettua aineistoa näitä luokkia käyttäen. Kuvassa 11 luokat 1 ja 2 ovat niin tiukasti ryhmittyneitä, ettei niitä voi edes hahmottaa luokan 4 seasta. Tässä kuvassa PCA:n ja enkooderin tulokset näyttävät molemmat onnistuneilta. Tarkentamalla datapisteiden suurimpaan ryhmittymään erot mallien välillä hahmottuvat selkeästi, katso kuva 12.

Kuvassa 12 nähdään molempien metodien tuottamat ydinryppää ja PCA:n heikkous paljastuu. PCA:n kuvan akseleiden arvoista huomataan, että tutkimme

Luokittelu defektien pinta-alan suuruuden mukaan

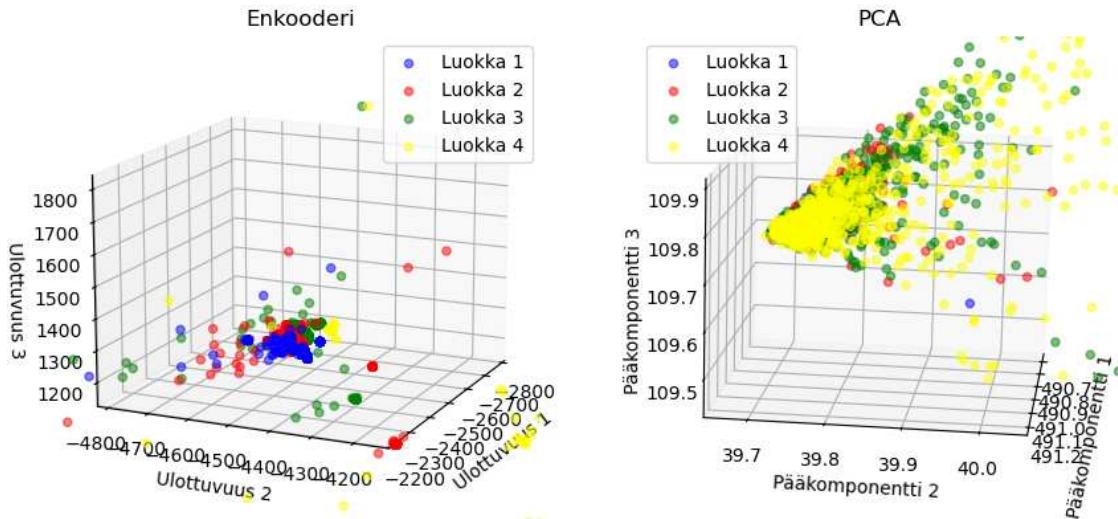


Kuva 11: PCA:n ja enkooderin tulokset luokiteltuna kvartaalien piikiekkojen defektien kokonaispinta-alan mukaan. Luokka 1 sisältää vähiten defektipinta-alaa, 4 eniten. Luokat 1 ja 2 ovat niin tiiviisti ryhmittyneet, etteivät ne näy luokan 4 alta. Kuvassa 12 ollaan tarkennettu tämän kuvan ydinryhmittymiin.

datapisteitä nyt todella pienellä alueella, mutta silti luokan 4 ja 1 datapisteet ovat täysin sekoittuneet. PCA ei erota datapisteitä, jotka ovat täysin ehjiä, ja niitä joiden defektien summien pinta-alan summa on yli 36525. PCA:n tulosta voidaan puolustaa sillä, että sen dataa on skaalattu ja enkooderi ei ole, mutta toisaalta PCA ei tuottanut järkeviä tuloksia edes skaalaamattomalla datalla, kuten kuvasta 2 näkyy. PCA käsittelee piirteitä yhdessä ulottuvuudessa, minkä takia se ei ota huomioon kanavia, tai defektien ilmaisutapoja. Neuroverkon kolmiulotteinen monikanavainen rakenne oli selkeästi oikea piikiekkonaineiston dimension alentamiseen. PCA:n lineaarisuus on myös suuri ongelma piikiekkodatan käsittelyssä. Oletetaan, että edes 50 pääkomponentin projektiolla PCA ei pysty säilyttämään datapisteiden pääpiirteitä muunnoksessa yhtä hyvin kuin enkooderi kolmessä ulottuvuudessa. Tämän pohjalta CNN:stä eristetty enkooderi taulukon 2 rakenteella valittiin työn lopulliseksi metodikksi dimension alentamiseen.

Kuvan 12 perusteella enkooderin tulokset ovat lupaavat. Se tuo todella tiivistyneen ryppään melkein kaikille datapisteille luokassa 1, joiden voidaan olettaa olevan täysin tai melkein kokonaan ehjiä. Myös se, että kaikki luokan datapisteet eivät ole samassa ryypäässä on lupaava tulos. Ideaalin enkooderin pitäisi ottaa huomioon myös kanava- ja anturikohtaiset defektit, eikä vain pelkkää defektien pinta-alojen yhteenlaskettua summaa. Otetaan esimerkiksi datapiste luokassa 1, jonka defektien pinta-alojen kokonaissumma kanavien välillä on pieni, mutta yksittäisen anturin defektiarvo on silti suuri. Tällöin datapisteen ei kuuluviskaan ryhmittyä muiden, täysin ehjien luokan 1 pisteiden joukkoon.

Luokittelu defektien pinta-alan suuruuden mukaan

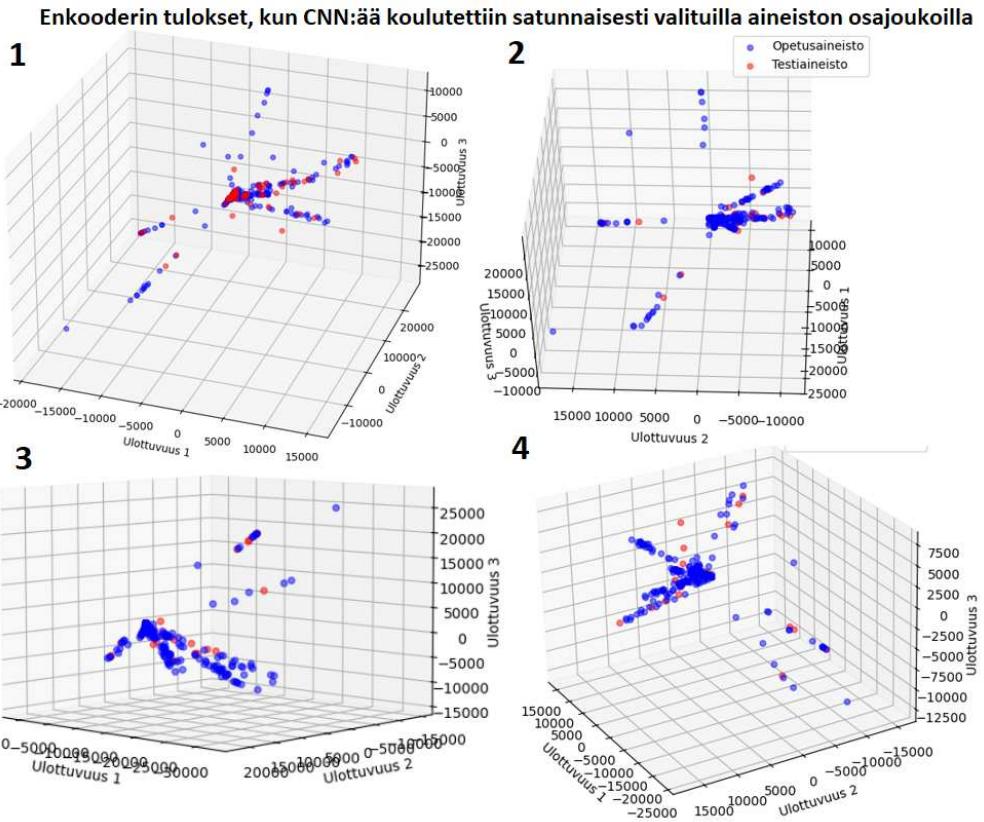


Kuva 12: Tarkennettu kuva PCA:n ja enkooderin tuloksista luokiteltuna kvartaaleihin piikiekkojen defektien kokonaispinta-alan mukaan. Kuvassa näkyy menetelmien luomat pääryhmätyt, missä oletettavasti ehjät piikiekot sijaitsevat. PCA ryhmittelee useita luokan 4 datapisteitä luokan 1 sekaan.

5.2 Enkooderin herkkyyys

Konvoluutioneuroverkkoa koulutettaessa opetusaineistoa sekoitetaan. Jokainen koulutuksen minierä on sekoitettu, joten neuroverkon tapa alentaa dimensiota vaihtelee koulutusten välillä, vaikka opetusaineisto pysyisi samana. Konvoluutioneuroverkon todettiin olevan melko stabiili käytettäessä samaa opetusaineistoa koulutusten välillä, jonka suuruus oli 80% kaikesta aineistosta. Tällöin keskineliövirhe pysyi koulutusten välillä samassa kokoluokassa ja datapisteiden ryhmittymät olivat samanlaiset, vaikka akselit joiden mukaan data ryhmittyi muuttuivat.

Kun CNN:n opetusaineistoa vaihdettiin satunnaisesti koulutusten välillä, enkooderin keskineliövirhe opetusaineistolle vaihteli jopa 30%. Kuvassa 13 näkyy 4 eri koulutuskertaa, missä koulutus 1 on alkuperäinen, osiossa 3.4 esitelty tulos. Kuvien tarkastelu eri näkökulmista oli vältämätöntä, koska datapisteet ryhmittyivät koulutusten välillä akselien suhteeseen eri tavalla. Kaikilla koulutuskerroilla enkooderi onnistuu ryhmittelemään dataa samankaltaisiin melko lineaarisii viivoihin ja pääryhmään. Tulosten poikkeavuus viittaa kuitenkin siihen, ettei aineisto ollut tarpeeksi suuri stabiilin tuloksen saavuttamiseksi. Toisaalta opetusaineiston kokoa voitaisiin kasvattaa testiaineistoa pienentämällä, mutta pienellä testiaineistolla CNN:n validointi olisi epäluotettavampaa.



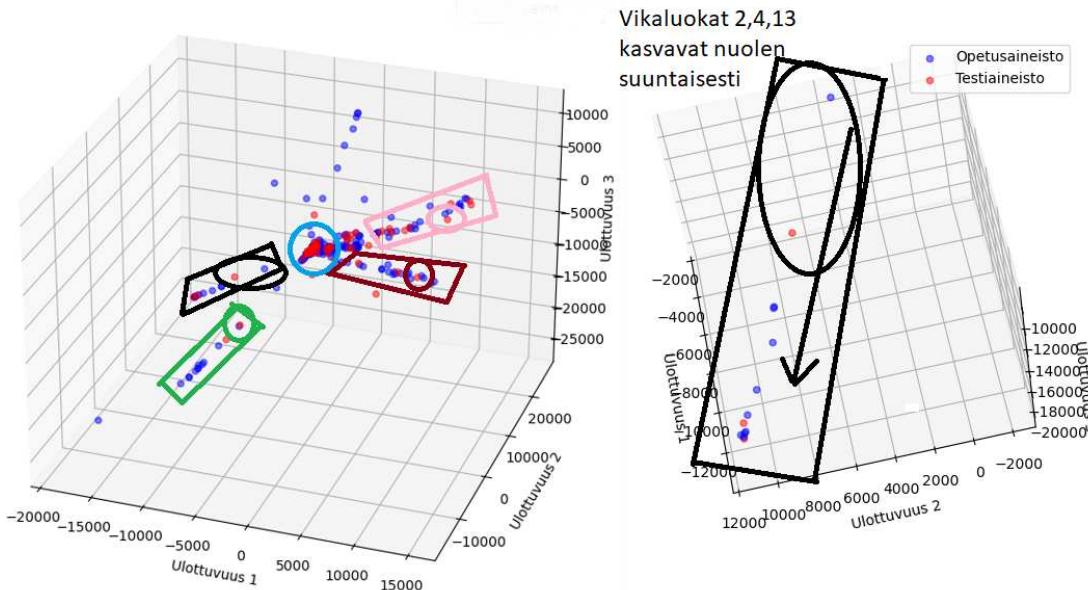
Kuva 13: Enkooderin tuottama dimension alentaminen, kun opetusaineistoa vaihdettiin satunnaisesti koulutusten välillä.

5.3 Enkooderin validointi

Enkooderin hyvyyttä tarkasteltiin valitsemalla muunnetun aineiston datapisteitä, joita sitten vertailtiin niiden alkuperäisessä dimensiossa. Käytetty enkooderi oli alkuperäinen, osion 3.4 tulos. Kuvassa 14 näkyvät datapisteet, joita tutkittiin. Ympyröiden sisällä olevien opetus- ja testidatapisteiden piireteisiin perehdyttiin. Sinisen ympyrän sisällä olevien datapisteiden todettiin olevan pääosin ehjiä, sillä niillä ei ole ollenkaan, tai on vain yksittäisiä merkittäviä defektejä. Tällä kertaa vertailtiin datapisteiden kanavakohtaisia defektien pinta-alasummia. Vihreässä laatikossa olevien datapisteiden antureissa oli erityisesti defektejä vikaluoissa 5 ja 8, mustassa laatikossa vastaavasti vikaluoikkia 2, 4 ja merkittävästi luokkaa 13. Pinkissä ja punaisessa laatikossa olevat datapisteet sisälsivät molemmat merkittävästi vikaluoikkaa 1 ja melko paljon luokkaa 4. Vikaluokkien bittikarttoja tutkimalla huomattiin, että läheisimpien pisteen defektit olivat samankaltaisia. Defektien kokonaispinta-ala kanavakohtaisesti kasvaa, kun tutkitaan datapisteitä, jotka ovat kauempana ehjistä piikiekoista, eli pääryhmästä. Kuvassa 14 on havainnollistettu mustan laatikon datapisteiden käyttäytymistä. Suurempi matka pääryhmästä tarkoittaa suurempia arvoja vikaluoille 2, 4 ja 13. Yleisesti kaukana pääryhmästä olevissa datapisteissä oli merkittäviä määriä montaa eri vikaluoikkaa.

Kun tarkasteltiin pelkkiä kanavakohtaisia defektien pinta-alasummia, pinkkei-

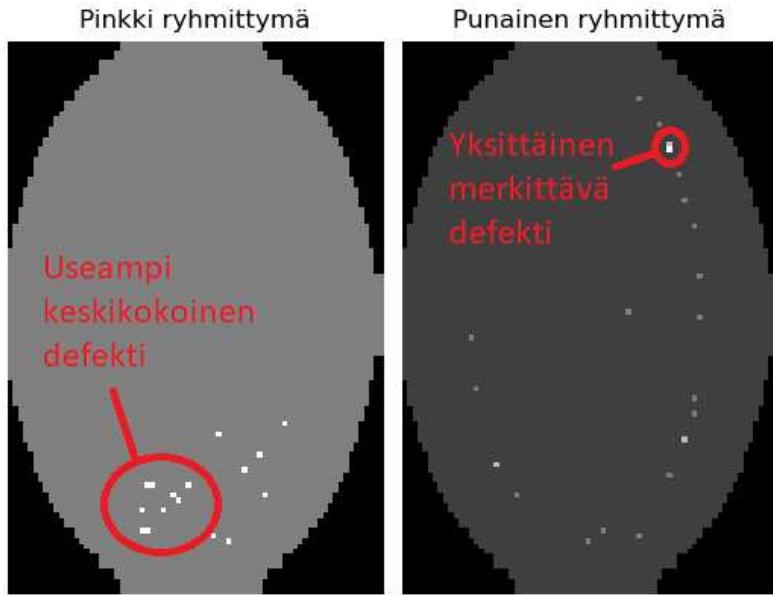
hin ja punaisiin lukeutuvat datapisteet vaikuttivat samanlaisilta. Tutkimalla näiden datapisteiden bittikarttoja erot selvenivät. Kuvassa 15 on havainnollistettu näihin väriryhmiin sisältyvien piiekkejä vikaluokan 1 bittikarttoja. Kuvassa oikealla olevan, punaisiin kuuluvilla datapisteillä oli vain yksi merkittävä vikaluokan 1 defekti, kun pinkillä oli useampia keskikokoisia. Tämän perusteella enkooderi onnistuu erottamaan joitakin piiekkejä valmistusvaiheen yleisviikoja. Jotta voitaisiin todeta, että enkooderi löytää kaikki erilaiset vikaluokat ja erottelee ne oikein, analysointi täytyisi suorittaa laajemmin.



Kuva 14: Enkooderin hyvyyttä tutkittiin jakamalla datapisteitä väreillä merkittyihin ryhmiin. Ryhmien datapisteitä vertailtiin keskenään ja muiden ryhmien kanssa.

5.4 Klusterointi

Edellisessä osiossa enkooderia validoidessa tutkittiin samalla epäsuorasti klusterien hyvyyttä, koska osa enkooderilla muunnetusta dataasta ryhmiteltiin samanlaisiin klustereihin kuin mitä k:n-keskiarvon-menetelmä tuotti. Edellisen osion tulosten peerusteella osiossa 4 enkooderille tehty klusterointi kuvassa 9 valinnalla $k = 8, 9, 10$ tuottaa yleistäviä, pääosin oikean kaltaisia perusryhmiä. Kuitenkin klusterointia voidaan pitää liian yleistävänä, sillä jopa valinnalla $k = 10$ voidaan valita mistä tahansa klusterista kaksi eri ääripäiden piiekkejä ja todeta, että ne ovat pääpiirteiltään erilaiset. Suhtellisen lyhyt matka enkooderin dimensioissa voi tarkoittaa jo uuden vian esiintymistä, sekä vikatyypikombinaatioita on niin paljon, ettei klusterointi näillä parametrivalinnoilla erottele niitä kaikkia. Täten k:n-keskiarvon-klusteroinnin tulosta ei hyväksytä, mutta valinnalla $k = 10$ sen käyttö ryhmien pohjarakenteena katsotaan hyödylliseksi, kun halutaan luoda tarkempia klustereita manuaalisesti.



Kuva 15: Kuvan 14 pinkin ja punaisen ympyrän sisällä olevien piikiekkojen vikaluuokkaa 1 esittävät bittikartat.

6 Yhteenvetö

6.1 Yhteenvetö

Työssä toteutettu enkooderi pystyi muuntamaan piikiekkoaineiston kolmiulotteiseksi vektoreiksi niin, että alemmassa ulottuvuudessa ainakin osa pääkategorioista erottui todennäköisintä ryhminä kuvaan tulkitessa. Työn päättävöitytseenen päästiin, mutta paljon kattavampi konvoluutioneuroverkon analysointi olisi hyödyllistä sen mahdollisten puutteiden löytämiseksi. Analyysin tekijän tulisi tietää piikiekkojen yleisvieroista, jotta mallin todellista vikojen erottelukykyä voitaisiin kritisoida.

6.2 Rajoitteet ja tulevaisuuden tutkimus

Osiossa 5.2 konvoluutioneuroverkolla näytettiin, että enkooderin tulos on aina erilainen nykyisellä aineistolla. Kuvaan 13 katsomalla voi kuitenkin huomata, että tulos on juuri sen verran stabiili, että aineisto muodostaa selviä, samankaltaisia ryhmiä jokaisella opetuskerralla. Kun opetusaineiston koko laskettiin 50%:iin koko datasta, tulokset olivat jo todella huonot, sillä silloin enkooderi tuotti vain yhden vikaluuokan esittävän 'viivan' ja ryhmitti loput datapisteet suureen pallon muotoa muistuttavaan pääryhmään. Tämän perusteella todettiin, että noin 4000 piikiekon aineisto, josta 80% käytetään CNN:n kouluttamiseen, on kokoluokaltaan pienin datajoukko, jolla saadaan tavoitteet tyydyttävä tulos.

Työn enkooderin suurin rajoite on opetusaineiston koko ja sen tulisi olla merkitsevästi suurempi. Lisäksi koulutusvaiheen oppimisastetta tulisi laskea esimerkiksi arvoon (0.00001 tai 0.000005) ja opetusaineiston läpikäyntikierroksia nostaa (100-200 kierrosta). Tämän työn enkooderin koulutus kesti noin 10 minuuttia, kun käytet-

tiin Nvidian GTX2080-näytönohjainta koodin ajamiseen. Kyseinen näytönohjain on laskuteholtaan kuluttajamarkkinoiden parhaita, ja tyypillisen toimistopöytäkoneen prosessorilla koulutus olisi kestänyt arviolta yli 4 tuntia. Suuremmilla opetusaineistoilla kannattaa siis varautua hyvin pitkiin, jopa yli päivän pituisiin opetusprosesseihin tarkan tuloksen saavuttamiseksi.

Kattavaan enkooderin tuloksen analysointiin suositellaan menetelmäksi laajaa manuaalista pisteen tarkastelua ja vertailua, kuten kappaleessa 5.3 tehtii pintapuolisesti. Olisi hyödyllistä aloittaa arvioimalla k:n-keskiarvon-klusteroinnin 10 klusterin tulosta, josta ryhmiä voidaan tarkentaa määrittämällä niiden rajat kolmiulotteisessa euklidisessa avaruudessa. Manuaaliseesti tarkennettuja datapisteryhmiä, sekä k:n-keskiarvon-klusteroinnin tuloksia voitaisiin analysoida myös klusterikohtaisella lineaarisella regressiolla, sillä kuvassa 9 on havaittavissa selkeitä klusterikohtaisia lineaarisia riippuvuuksia. Klusterikohtainen lineaarinen regressio kertoo enemmän koko aineiston tuotantoprosessista kuin yksittäisistä piikiekoista. Sovittamalla lineaarisia regressiomalleja klusterikohtaisesti, voitaisiin ennustaa vikatyyppien suuruuksia luottamusväleillä, sekä yrittää selittää vikojen esiintymiseen vaikuttavia tekijöitä Bagirov et al. (2017). Toisaalta vikojen esiintymisen syitä voi olla käytännöllisempää etsiä suoraan niistä tuotantolaitteista, jotka ovat olleet mukana viallisten komponenttien valmistuksessa. Lisäksi aineiston luokittelun kuvien perusteella vikaryhmiin ja sitten vertailu alemmassa ulottuvuudessa kertoisi suoraan, mitä vikoja enkooderi ei vältämättä erota ja missä se onnistuu hyvin. Jos saatu tulos on onnistunut, uudet määritetyt ryhmät kolmessa ulottuvuudessa määräätään luokiksi. Sitten enkooderia uudelle piikiekolle sovellettaessa saadaan suoraan sen pääkategoria.

MEMS-piikiekkojen manuaalinen laadunvalvonta on työlästä ja altista ihmisen tekemille virheille. Lähes jokaisessa teollisessa prosessissa valmistetussa piikiekossa havaitaan ainakin muutama neliomikrometrikkoinen vika. Piikiekossa on yli 3300 anturia, jotka sahataan erilleen vasta valmistuksen jälkeen, joten satunnaiset anturikohtaiset viat eivät vielä tarkoita, että koko komponentti olisi viallinen. Tämän kandidaatintutkielman tavoitteena on luoda Murata Electronics Oy:n toimeksiannosta pohjaa ratkaisulle, jossa piikiekot esitetään kakso- tai kolmiulot- teisina vektoreina niiden pääpiirteet säilyttäen. Piikiekkoaineiston ulottuvuuden alentaminen tehdään lineaarisella mallilla eli pääkomponenttianalyysillä (PCA), se- kä epälineaarisen 3D konvoluutioneuroverkosta (CNN) eristetyllä enkooderilla, ja saatuja tuloksia vertaillaan. Alempiulotteinen esitys mahdolistaa piikiekkojen vertailun tulkitsemalla kuvaa, jossa samankaltaiset piikiekot ryhmittyvät. Tutkielmassa halutaan myös selvittää, kuinka teollista aineistoa tulee esikäsitellä parhaan tuloksen saamiseksi. Aineiston esitys alemmassa ulottuvuudessa pyritään klusterointimaan todennäköisesti, eli ryhmittelemään ohjaamattomasti piikiekkojen kategorioihin. Kun laadunvalvonta saa tuekseen piikiekon pääpiirteitä kuvaavan vektorin, piikitko voidaan arvioida tukeutuen menneisyydessä valmistettujen piikiekkojen vektoritietokantaan. Tarkka viantunnistus johtaa nopeampiin jatkotoimenpiteisiin, jotka tukevat koko tuotantolinjaa. Tulosten perusteella molemmat mallit ulottuvuuden alentamiseen hukkasivat paljon tietoa. Pääkomponenttianalyysin lineaarinen piirteiden muunnos todettiin epätarkaksi ja metodi hylätätiin. Enkooderi löysi aineistosta epälineaarisia riippuvuuksia ja onnistui erottelemaan joitakin piikiekkojen valmistusvaiheen yleisvikoja.

Enkooderin todettiin olevan herkkä, sillä sen tekemässä muunnoksessa oli selkeitä eroja konvoluutioneuroverkon koulutuskertojen välillä. Stabiilimman mallin saavuttamiseksi tarvitaan merkittävästi enemmän dataa CNN:n koulutusvaiheeseen. Klusterointi oli liian yleistävä, mutta sen tulos on hyödyllinen todellisten ryhmittymien manuaalisessa etsinnässät.

Viitteet

- H. Abdi ja L. Williams. Principal component analysis. *WIREs Computational Statistics: Volume 2, Issue 4*, 2(4):433–459, 2010.
- M. Ahsan, M. Mahmud, P. Saha, K. Gupta, ja Z. Siddique. Effect of data scaling methods on machine learning algorithms and model performance. *Technologies*, 9(52), 2021.
- L. Alzubaidi, J. Zhang, A. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santa-maria, M. Fadhel, M. Al-Amidie, ja L. Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.
- A. Bagirov, A. Mahmood, ja A. Barto. Prediction of monthly rainfall in Victoria, Australia: Clusterwise linear regression approach. *Atmospheric Research*, 2017.
- C. Bocci ja L. Chiantini. *An Introduction to Algebraic Statistics with Tensors*. Springer, Università di Siena, Italy, 2019.
- P. Diederik ja J. Kingma. Adam: A method for stochastic optimization. *Arxiv*, 1412(6980), 2015.
- E. Dogo, O. Afolabi, N. Nwulu, B. Twala, ja C. Aigbavboa. A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*, pages 92–99, 2018.
- G. Fung. A comprehensive overview of basic clustering algorithms. *Citeseer*, 2001.
- C. Garbin, X. Zhu, ja O. Marques. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia tools and applications*, 79(19):12777–12815, 2020.
- H. Gholamalinezhad ja H. Khosravi. Pooling methods in deep neural networks, a review. *arXiv*, 2009(07485), 2020.
- D. Godoy. DL-visuals, figures and diagrams of the most popular deep learning architectures and layers, <https://github.com/dvgodoy/dl-visuals>, 2021.
- S. Holland. Principal component analysis (PCA). 2008.

- A. Jović, K. Brkić, ja N. Bogunović. A review of feature selection methods with applications. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205, 2015.
- A. Jung. *Machine Learning, The Basics*. Springer, Singapore, 2022.
- Y. Kunang, S. Nurmaini, D. Stiawan, ja A. Zarkasi. Automatic features extraction using autoencoder in intrusion detection system. *International Conference on Electrical Engineering and Computer Science (ICECOS)*, pages 219–224, 2018.
- L Ladla ja T. Deepa. Feature selection methods and algorithms. *International Journal on Computer Science and Engineering (IJCSE)*, 3(5):1787–1797, 2011.
- S. Laine. *Using visualization, variable selection and feature extraction to learn from industrial data*. Väitöskirja, Helsinki University of Technology, Espoo, 2003.
- H. Liu ja L. Yu. Toward integrating feature selection algorithms for classification and clustering. *Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005.
- A. Martinez ja M. Zhu. Where are linear feature extraction methods applicable? *Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1934–1944, 2005.
- C. Park, H. Park, ja P. Pardalos. A survey of feature selection and feature extraction techniques in machine learning. *Fourth IEEE International Conference on Data Mining (ICDM'04), Brighton, UK*, pages 495–498, 2004.
- A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, and S. Chilamkurthy. A. Tejani, B. Steiner, L. Fang, J. Bai, ja S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8024–8035, 2019.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, ja E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- E. Principi, D. Rossetti, S. Squartini, ja F. Piazza. Unsupervised electric motor fault detection by using deep autoencoders. *EEE/CAA Journal of Automatica Sinica*, 6 (2):441–451, 2019.
- D. Ruan, J. Wang, J. Yan, ja C. Gühmann. CNN parameter design based on fault signal analysis and its application in bearing fault diagnosis. *Advanced Engineering Informatics*, 55, 2023.

- T. Khalil S. Khalid ja S. Nasreen. A survey of feature selection and feature extraction techniques in machine learning. *Science and Information Conference, London, UK*, pages 372–378, 2014.
- A. Shrestha ja A. Mahmood. Review of deep learning algorithms and architectures. *IEEE access*, 7:53040–53065, 2019.
- K. Sinaga ja M. Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8: 80716–80727, 2020.
- M. Tschannen, O. Bachem, ja M. Lucic. Recent advances in autoencoder-based representation learning. *ArXiv*, abs/1812.05069, 2018.
- M. Vaulanen. Specialist, Quality Control. Murata Technologies Oy. Myllykivenkuja 6, 01620 Vantaa. Haastattelu 17.9.2024.
- M. Verleysen ja D. François. The curse of dimensionality in data mining and time series prediction. *in: Cabestany, J., Prieto, A., Sandoval, F. (eds)*, pages 758–770, 2005.